A flexible SNARK via the monomial basis

Steve Thakur

Panther Protocol

Abstract

We describe a pairing-based SNARK with a universal updateable CRS that can be instantiated with any pairing-friendly curve endowed with a sufficiently large prime scalar field. We use the monomial basis, thus sidestepping the need for large smooth order subgroups in the scalar field. In particular, the scheme can be instantiated with outer curves to widely used curves such as Ed25519, secp256k1, BN254 and BLS12-381. This allows us to largely circumvent the overhead of non-native field arithmetic for succinct proofs of valid signatures in Ed25519 and secp256k1 and one layer recursion with BN254 or BLS12-381.

The proof size is constant (10 $\mathbb{G}_1$, 20 $\mathbb{F}_p$), as is the verification time, which is dominated by a single pairing check (i.e. two pairings). The Prover time is dominated by the 10 multi-scalar multiplications in $\mathbb{G}_1$ - with a combined MSM length of $22 \cdot |\text{Circuit}|$ - and, to a lesser extent, the computation of a single sum of polynomial products over the scalar field via multimodular FFTs [1].

The scheme supports succinct lookup arguments for subsets as well as subsequences. Our construction relies on homomorphic table commitments, which makes them amenable to vector lookups. The Prover algorithm runs in runtime $O(M \cdot \log(M))$, where $M = \max\{|\text{Circuit}|, |\text{Table}|\}$.

Furthermore, the scheme supports custom gates, albeit at the cost of a larger proof size. As an application of the techniques in this paper, we describe a protocol that supports multiple *univariate* custom gates $\mathcal{G}_i$ of high degree that are sparsely distributed in the sense that

$$\sum_i \deg(\mathcal{G}_i) \cdot \#(\mathcal{G}_i \text{ gates}) \;=\; O(|\text{Circuit}|).$$

This comes at the cost of three additional $\mathbb{G}_1$ elements and does not blow up the proof generation time, i.e. it does not entail MSMs or FFTs of length larger than the circuit size.

At the moment, Panther Protocol's Rust implementation with a 570-bit pairing-friendly outer curve to Ed25519 has a (not yet optimized) Prover time of 32 seconds for a million gate circuit on a 64 vCPU AWS machine.

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 compatibilty 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 pairing-friendly curve. 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 ([Gro16], [GWC19], [CHMMVW20] etc.) 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.

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

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

The proof size is 10 $\mathbb{G}_1$ elements and 20 $\mathbb{F}_p$ elements in the version we present here. We have tried to keep the number of $\mathbb{G}_1$ elements to a bare minimum since for our primary use cases, they are over twice as large as the $\mathbb{F}_p$ elements and determine the number of MSMs, which is the most expensive part of the proof generation.

We note that this version is optimized for the proof size and the verification time rather than the Prover time. It is straightforward to reduce the combined MSM length (and consequently, the Prover time) by allowing for more $\mathbb{G}_1$ elements in the proof.

3 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 such as 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 a single pairing check, i.e. two pairings. As usual, the Miller loops in these two pairings can be parallelized and the two final exponentiations can be batched.

4 Universal updateable trusted setup and a CRS size linear in the circuit size

We use the 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.

5 Support for lookup arguments for subsets and subsequences

The scheme supports lookup arguments for subsets as well as subsequences. Our construction relies on homomorphic table commitments, which makes them amenable to vector lookups. The Prover algorithm runs in runtime $O(M \cdot \log(M))$, where $M := \max\{|\text{Circuit}|, |\text{Table}|\}$.

We do not yet have a protocol in the monomial basis that enables a Prover time independent of the table size after preprocessing, as is the case with [EFG22].

6 As few polynomial multiplications as possible and preferably in the same step so that they can be parallelized.

This scheme uses a monomial basis rather than a Lagrange basis. Consequently, the only juncture where the scheme needs FFTs is a sum of six polynomial products that occurs in Step 7 of the SNARK (in the version we present in this draft). The products can be parallelized and there are further savings if the Prover uses a single inverse FFT for the sum rather than using an inverse FFT per product.

Unlike the uses cases targeted by the impressive body of work in recent years ([CBBZ22] etc) that avoids FFTs altogether, we found that for our use cases and circuit sizes, the polynomial products were not the primary bottleneck. The MSMs continue to be the biggest contributors to the Prover time, even in the cases where the scalar fields have low 2-adicity and the polynomial products require the multimodular FFT or the Schönhage-Strassen algorithm. This discrepancy between the total MSM runtime and the polynomial product runtime is even more pronounced (as one would expect) when the base field is twice the size of the scalar field, as is the case for Cocks-Pinch or Brezing-Weng outer curves.

To that end, we have included some benchmarks for polynomial products in a few relevant FFT-*unfriendly* fields using the NTL library. These include the base fields of the widely used curves Ed25519, secp256k1, BN254 and BLS12-381, all of which allow for the construction of pairing-friendly outer curves of embedding degree 6 (12 in the case of Ed25519).

7 Support for custom gates

The scheme 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.

The scheme also supports multiple *univariate* custom gates of high degree that are sparsely distributed, in the sense that

$$\sum_i \deg(\mathcal{G}_i) \cdot \#(\mathcal{G}_i \text{ gates}) \;=\; O(\text{Circuit Size}).$$

This comes at the cost of 3 extra $\mathbb{G}_1$ elements in the proof and does not blow up the proof generation time because the MSMs and the FFTs are of lengths $O\big(\max(|\text{Circuit}|,\ \deg(\mathcal{G}_i)\big)$ with this approach. We note, however, that this approach does not seem to be easily adaptable to high degree gates other than those defined by univariate polynomials.

This protocol hinges on the Hadamard product protocol and an elementary lemma we use throughout the paper (lemma 2.1), namely the fact that a randomized sum of rational functions being a polynomial implies that all of the rational functions are polynomials with overwhelming probability. The Verifier does not need to store the potentially large univariate polynomials that define these custom gates. Instead, he stores the KZG10 commitments to these polynomials.

8 Constant-sized storage for the Verifier

Despite the CRS being of length linear in the circuit size, the Verifier stores a constant number of $\mathbf{G}_1$ and $\mathbf{G}_2$ points. He also stores the KZG10 commitments to a few polynomials pre-processed during the circuit generation.

9 A wider class of amicable curve pairs for pairing-based recursion

Arbitrarily deep recursion with pairings requires an *amicable* pair E_1, E_2 over prime fields $\mathbb{F}_{p_1}$, $\mathbb{F}_{p_2}$ such that $p_1 = |E_2(\mathbb{F}_{p_2})|$ and $p_2 = |E_1(\mathbb{F}_{p_1})|$. At present - as far as we know - the pairing based recursive schemes require both $p_1 - 1$ and $p_2 - 1$ to have large smooth divisors. Since our scheme works with scalar fields without large smooth order subgroups, our approach allows for a wider class of amicable curve pairs, albeit at the cost of more expensive polynomial products.

10 Immunity to Cheon-type attacks ([Che10])

Cheon's attack against common reference strings brings the security of the scheme down to $O(\sqrt{\frac{p-1}{d}} + \sqrt{d})$, where d is the largest divisor of $p-1$ such that the element $\mathbf{g}_1^{\mathbf{s}^d}$ is part of the CRS. Pairing-based SNARKs including [Gro16], [GWC19]. [CHMMVW20] require a 2-power larger than the circuit size to divide $p-1$ and hence, are affected by this attack to some extent. If the scalar field $\mathbb{F}_p$ is such that $p-1$ has no large smooth divisors - as could be the case with our scheme - Cheon's attack (and others similar to it) do not appear to affect the security level. That said, this was one of our lower priorities.

Implementation: Panther Protocol has implemented a bare-bones version of the scheme - i.e. without higher arity, custom gates or lookup arguments - in Rust. The implementation uses a

576-bit pairing friendly outer curve to Ed25519, constructed using the Cocks-Pinch algorithm. The curve has j-invariant 0 and embedding degree 12 with respect to the prime $2^{255} - 19$. At the moment, a circuit size of one million with this curve has a (not yet optimized) Prover time of 32 seconds on a 64 vCPU AWS machine.

1.1 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 KZG10 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

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

1.2 Notations and terminology

As usual, $\mathbb{F}_p$ denotes the finite field with p elements for a prime power p and $\overline{\mathbb{F}}_p$ denotes its algebraic closure. $\mathbb{F}_p^*$ denotes the cyclic multiplicative group of the non-zero elements of $\mathbb{F}_p$. $\mathbb{F}_p[X]$ denotes the ring of univariate polynomials over $\mathbb{F}_p$, which is a principal ideal domain. $\mathbb{F}_p(X)$ denotes the field $\text{Frac}(\mathbb{F}_p[X])$, the fraction field of $\mathbb{F}_p[X]$. 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 $f(X)$. Throughout this paper, p will be a large odd prime.

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.

By the *circuit size*, we mean the total number of addition and multiplication gates in a fan-in two circuit.

We denote by $\boldsymbol{\lambda} \in \mathbb{Z}^+$ a security parameter. We denote by $\texttt{negl}(\boldsymbol{\lambda})$ an unspecified function that is *negligible* in $\boldsymbol{\lambda}$ (namely, a function that vanishes faster than the inverse of any polynomial in $\boldsymbol{\lambda}$). When a function can be expressed in the form $1 - \texttt{negl}(\boldsymbol{\lambda})$, we say that it is *overwhelming* in $\boldsymbol{\lambda}$. By PPT, we refer to a probabilistic polynomial time algorithm.

Definition 1.1. An argument system is *complete* if an honest Prover can efficiently output an accepting transcript with probability 1.

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 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 $\{\mathbf{g}^{\mathbf{s}^i} : 1 \leq i \leq n\}$ 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*

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

A special case of the KEA assumption is that given the elements $\{g^{\mathbf{s}^i} : 0 \leq i \leq n\}$, if a PPT algorithm $\mathcal{A}_1$ is able to output a triplet $(c_1, c_2, f(X)) \in \mathbb{G} \times \mathbb{G} \times \mathbb{F}_p[X]$ with $\deg(f(X) \geq 1$ such that $c_2 = c_1^{f(\mathbf{s})}$, then there is a PPT algorithm $\mathcal{A}_2$ that with overwhelming probability, outputs a polynomial $e(X)$ such that

$$c_1 = g_1^{e(\mathbf{s})} \ , \ c_2 = g_1^{e(\mathbf{s}) \cdot f(\mathbf{s})}.$$

1.4 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 $\mathcal{A}_{\mathtt{PPT}}$ in a CRS-based protocol, we mean a PPT algorithm which satisfies the following:

Whenever $\mathcal{A}_{\mathtt{PPT}}$ 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

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

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.

1.5 Commitments to index sets

For an index set $I \subseteq [0,\ \text{length}(\texttt{CRS})]$, 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 *characteristic polynomial* of $\mathcal{I}$. Thus, the commitment is given by

$$\texttt{Com}(\mathcal{I}) \ := \ \texttt{Com}(\chi_{\mathcal{I}}(X)) \ = \ \mathbf{g}_1^{\chi_{\mathcal{I}}(\mathbf{s})} \ = \ \mathbf{g}_1^{\sum_{i \in \mathcal{I}} \mathbf{s}^i}.$$

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 $I \subseteq [0,\ n]$.

1.6 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

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

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

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

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

For a fixed integer N 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

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

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.

Showing that the high degree part $f_{\Pi,\gamma,+}(X)$ of $f_{\Pi,\gamma}(X)$ is divisible by X^{N+1} requires a simple divisibility check. And since X^{N+1} is a monomial, the division requires a left shift by $N+1$ positions rather than any expensive polynomial division. Lastly, the Prover show that the polynomial

$$f_{\Pi,\gamma}(X) \ - \ f_{\Pi,\gamma,+}(X) \ - \ f_{1,2}(\gamma) \cdot X^N$$

is of degree $\leq \ N$, where $f_{1,2}(X)$ is the committed polynomial that is claimed to be the Hadamard product $f_1 \odot f_2(X)$.

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 $\deg(f) \leq n$.

We note that Hadamard products are batchable in the sense that to prove the equations

$$f_{j,1} \odot f_{j,2}(X) \ = \ f_{j,1,2}(X) \ \text{ for } \ j = 1, \cdots, k \ ,$$

for committed polynomials $f_{j,1},\ f_{j,2}(X),\ f_{j,1,2}(X)$, it suffices to show that for randomly generated challenges γ, λ, the sum

$$f_\lambda(X) \ := \ \sum_{j=1}^{k} \lambda^{j-1} \cdot f_{j,1}(\gamma \cdot X) \cdot X^N \cdot f_{j,2}(X^{-1})$$

has coefficient $\sum\limits_{j=1}^{k} \lambda^{j-1} \cdot f_{j,1,2}(\gamma)$ at the position X^N. This boils down to expressing the difference

$$f_\lambda(X) \ - \ \big[\sum_{j=1}^{k} \lambda^{j-1} \cdot f_{j,1,2}(\gamma)\big] \cdot X^N$$

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}.

Since this scheme uses the monomial basis, the Prover needs to show that certain triples of committed polynomials satisfy the Hadamard product equation. In particular, for a fan-in two arithmetic circuit, if polynomials $L(X)$, $R(X)$, $O(X)$ represent the left, right and output wire polynomials respectively and $\mathcal{I}_A$, $\mathcal{I}_M$ are the index sets corresponding to the addition and multiplication gates, the Prover needs to show that

$$[L(X) + R(X) - O(X)] \odot \chi_{\mathcal{I}_A}(X) = 0 \ , \ \ [L(X) \odot R(X) - O(X)] \odot \chi_{\mathcal{I}_M}(X) = 0.$$

1.7 Degree upper bounds

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 KZG10 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 2.1). Thus, a Prover can demonstrate all of these degree upper bounds by *verifiably* sending the KZG10 commitment to $f_\lambda(X)$.

1.8 The Permutation argument

The scheme hinges on a permutation argument akin to and influenced by PLONK's ([GWC19]). The Prover succinctly shows that certain wire values are equal by showing that the committed wire polynomial is stable under the action of a prescribed permutation, a commitment to which is stored by the Verifier. As one would expect, we need to commit to permutations in a

different manner compared to the schemes that use the Lagrange basis. 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 $P_{\mathtt{id},N}(X) := \sum_{i=0}^{N-1} k \cdot X^k$.

For polynomials $f(X),\ \widetilde{f}(X) \in \mathbb{F}_p[X]$, we say $\boldsymbol{\sigma}(f) = \widetilde{f}$ if the coefficients of $f(X),\ \widetilde{f}(X)$ satisfy the equations:

$$\mathtt{Coef}(\widetilde{f}\ ,\ j) \ =\ \mathtt{Coef}(f\ ,\ \boldsymbol{\sigma}(j)) \quad \forall\ j \in [0,\ N-1].$$

In response to two randomly and uniformly generated challenges $\delta_1,\ \delta_2$, the Prover shows that he knows a polynomial $F_{\delta_1,\delta_2}(X)$ of degree $\leq n-1$ and with cyclic right shift

$$F^{\mathtt{RShift}}_{\delta_1,\delta_2}(X) \ :=\ X \cdot F_{\delta_1,\delta_2}(X) \ (\mathrm{mod}\ X^N - 1)$$

such that:

$$\mathtt{Coef}(F^{\mathtt{RShift}}_{\delta_1,\delta_2}\ ,\ 0) \ =\ 1 \ =\ \mathtt{Coef}(F_{\delta_1,\delta_2}\ ,\ N-1)$$

and the following equation of Hadamard products holds:

$$[f(X)+\delta_1 \cdot \sum_{j=0}^{N-1} j \cdot X^j + \delta_2 \cdot \sum_{j=0}^{N-1} X^j] \odot F_{\delta_1,\delta_2}(X) \ =\ [\widetilde{f}(X)+\delta_1 \cdot \sum_{j=0}^{N-1} \boldsymbol{\sigma}(j) \cdot X^j + \delta_2 \cdot \sum_{j=0}^{n-1} X^j] \odot F^{\mathtt{RShift}}_{\delta_1,\delta_2}(X).$$

The equation implies that for every $i \in [0,\ N-1]$,

$$\mathtt{Coef}(F_{\delta_1,\delta_2}\ ,\ i) \ =\ \prod_{j=0}^{i} \frac{\mathtt{Coef}(f\ ,\ j) + \delta_1 \cdot j + \delta_2}{\mathtt{Coef}(\widetilde{f}\ ,\ j) + \delta_1 \cdot \boldsymbol{\sigma}(j) + \delta_2}.$$

In particular,

$$\prod_{j=0}^{N-1} \mathtt{Coef}(f\ ,\ j) + \delta_1 \cdot j + \delta_2 \ \ =\ \ \prod_{j=0}^{N-1} \mathtt{Coef}(\widetilde{f}\ ,\ j) + \delta_1 \cdot \boldsymbol{\sigma}(j) + \delta_2.$$

Since the challenge δ_2 was randomly and uniformly generated, the Schwartz-Zippel lemma implies that with overwhelming probability, the *multisets* $\{\mathtt{Coef}(f\ ,\ j)+\delta_1 \cdot j\}_j$, $\{\mathtt{Coef}(\widetilde{f}\ ,\ j)+\delta_1 \cdot \boldsymbol{\sigma}(j)\}_j$ coincide. Furthermore, since the challenge δ_1 was also randomly and uniformly generated, any equation

$$\mathtt{Coef}(f\ ,\ j) + \delta_1 \cdot j \ =\ \mathtt{Coef}(\widetilde{f}\ ,\ i) + \delta_1 \cdot \boldsymbol{\sigma}(i) \ \ ,\quad i,j \in [0,\ N-1]$$

implies that with overwhelming probability,

$$j = \boldsymbol{\sigma}(i) \ \ ,\ \ \mathtt{Coef}(\widetilde{f}\ ,\ i) = \mathtt{Coef}(f\ ,\ \boldsymbol{\sigma}(i)).$$

Thus, with overwhelming probability, $\boldsymbol{\sigma}(f) = \widetilde{f}$.

1.9 Structure of the paper

In section 2, we discuss a few preliminary lemmas that will be used in the subsequent sections. We also discuss a protocol that allows us to have no more than two $\mathbb{G}_1$ elements for the multiple polynomial commitment openings in the Snark. We also describe the protocol for batched divisibility which allows us to keep the $\mathbb{G}_1$ elements in the Snark proof to a bare minimum.

In section 3, we describe the Snark in the batched form. This version has 10 $\mathbb{G}_1$ elements and 20 $\mathbb{F}_p$ elements. The combined length of the 10 MSMs is $22 \cdot |\text{Circuit}|$. We note that it is possible to reduce the total MSM length at the cost of a few more $\mathbb{G}_1$ elements in the proof. This reduces the proof generation time but results in a larger proof size.

In section 4, we describe the various subprotocols that underpin the Snark In particular, we describe the protocols for the Hadamard product and the permutation argument in the monomial basis. Unlike in the case of the Snark, we have made no particular effort to make these modular subprotocols efficient, beyond ensuring that the proof sizes and verification times are constant and the proof generation times quasi-linear. The purpose of this section is merely to help explain the ideas used in the Snark.

In section 5, we describe the lookup protocol. It allows the Prover to show that the coefficients of a committed polynomial lie in a committed table, a commitment to which is stored by the Verifier. The scheme supports succinct arguments for subsets as well as subsequences.

In section 6, we describe the protocol for univariate custom gates of high degree. The protocol allows for multiple *univariate* custom gates $\mathcal{G}_i$ of high degree that are sparsely distributed, in the sense that

$$\sum_i \deg(\mathcal{G}_i) \cdot \#(\mathcal{G}_i \text{ gates}) \ = \ O(\text{Circuit Size}).$$

This comes at the cost of 3 extra $\mathbb{G}_1$ elements in the proof and does not blow up the proof generation time. The protocol hinges on the Hadamard product subprotocol and an elementary lemma we use throughout the paper, namely the fact that a randomized sum of rational functions being a polynomial implies that all of the rational functions are polynomials.

In the Appendix, we provide sketches of the security proofs for the Snark and for the subprotocols.

2 Preliminary lemmas

We will need the following lemma repeatedly.

Lemma 2.1. *For rational functions $h_i(X) \in \mathbb{F}_p(X) := \text{Frac}(\mathbb{F}_p[X])$, 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 $\text{val}_{q(X)}(h_j(X)) \leq -1$, 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. □

Lemma 2.2. *Let $f_1(X), f_2(X) \in \mathbb{F}_p[X]$. Suppose there exists a polynomial $F(X)$ of degree $\leq N-1$ with cyclic right shift*

$$F^{\texttt{RShift}}(X) \ := \ X \cdot F(X) \ (\text{mod } X^N - 1)$$

such that:

- $\texttt{Coef}(F^{\texttt{RShift}} \ , \ 0) \ = \ 1 \ = \ \texttt{Coef}(F \ , \ N-1)$

- *For a randomly generated $\gamma \in \mathbb{F}_p$, the polynomials $f_1(\gamma \cdot X) \cdot X^N \cdot F(X^{-1})$ and $f_2(\gamma \cdot X) \cdot X^N \cdot F^{\texttt{RShift}}(X^{-1})$ have the same coefficient at the position X^N.*

Then with overwhelming probability, the polynomials $f_1(X) \ (\text{mod } X^N)$, $f_2(X) \ (\text{mod } X^N)$ have the same product over the multisets of coefficients.

Proof. Replacing $f_1(X)$, $f_2(X)$ by the residues modulo X^N is necessary, we may assume without loss of generality that they are of degree $\leq N-1$.

Let $f_1 \odot F(X)$, $f_2 \odot F^{\texttt{RShift}}(X)$ denote the Hadamard products. The coefficients of

$$f_1(\gamma \cdot X) \cdot X^N \cdot F(X^{-1}) \quad , \quad f_2(\gamma \cdot X) \cdot X^N \cdot F^{\texttt{RShift}}(X^{-1})$$

at the position X^N are given by $f_1 \odot F(\gamma)$, $f_2 \odot F^{\texttt{RShift}}(\gamma)$ respectively. Thus,

$$f_1 \odot F(\gamma) \ = \ f_2 \odot F^{\texttt{RShift}}(\gamma)$$

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

$$f_1 \odot F(X) \ = \ f_2 \odot F^{\texttt{RShift}}(X).$$

Now,

$$\texttt{Coef}(F \ , \ N-1) = 1 \ \ , \ \ F^{\texttt{RShift}}(X) = X \cdot F(X) - (X^N - 1) = 1 + \big[\sum_{i=0}^{N-2} \texttt{Coef}(F \ , \ i) \cdot X^{i+1} \big]$$

Induction on i implies that with overwhelming probability,

$$\texttt{Coef}(F(X) \ , \ i) \ = \ \prod_{j=0}^{i} \frac{\texttt{Coef}(f_2 \ , \ j)}{\texttt{Coef}(f_1 \ , \ j)} \quad \forall \, i \ \in \ [0, \ N-1]$$

and in particular,

$$\prod_{j=0}^{N-1} \frac{\texttt{Coef}(f_2 \ , \ j)}{\texttt{Coef}(f_1 \ , \ j)} = 1,$$

which completes the proof. □

2.1 Batched proof of divisibility

The Snark requires 7 polynomial commitment openings at 7 different points. Naïvely, this would mean 7 extra $\mathbb{G}_1$ elements, which would increase the proof size as well as the number of MSMs in $\mathbb{G}_1$, thus increasing the Prover time. Instead, we describe a protocol that requires the Prover to send 2 $\mathbb{G}_1$ elements and 7 $\mathbb{F}_p$ elements. We note that in the primary uses cases we deal with, the base field is more than twice the size of the scalar field, which makes the $\mathbb{G}_1$ elements substantially larger than the $\mathbb{F}_p$ elements. So this protocol improves the proof size as well as the Prover time.

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 Snark proof size low and to keep the number of MSMs to a bare minimum.

The protocol hinges on the simple observation (lemma 2.1) that for a set of rational functions in $\mathbb{F}_p(X) := \text{Frac}(\mathbb{F}_p[X])$, if a randomized sum of these rational functions is a polynomial, then with overwhelming probability, all of the rational functions are polynomials.

We need the polynomials $e_i(X)$ to be sparse so that the Verifier can evaluate them at a challenge. The Snark (our primary use case for this protocol) has 7 such pairs $f_i(X)$, $e_i(X)$ and all of the $e_i(X)$ are linear polynomials.

Protocol 2.3. *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 $[\mathbf{g}_1,\ \mathbf{g}_1^{\mathbf{s}},\ \cdots,\ \mathbf{g}_1^{\mathbf{s}^M}]\ ,\ [\mathbf{g}_2,\ \mathbf{g}_2^{\mathbf{s}}]$
Inputs: Elements $\mathbf{a}_i\ \in\ \mathbb{G}_1$; sparse public polynomials $e_i(X) \in \mathbb{F}_p$; $i = 1, \cdots, k$
Claim: The Prover knows polynomials $f_i(X)$ such that

$$\mathbf{a}_i = \mathbf{g}_1^{f_i(\mathbf{s})}\ \ ,\ \ f_i(X) \equiv 0 \ (\text{mod}\ e_i(X)).$$

1. The hashing algorithm $\texttt{Hash}_{\text{FS}}$ generates a challenge $\widetilde{\lambda}$.

2. The Prover $\mathcal{P}$ computes the polynomial

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

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

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

3. The hashing algorithm $\texttt{Hash}_{\text{FS}}$ generates a challenge $\widetilde{\alpha}$.

4. $\mathcal{P}$ sends the $\mathbb{F}_p$-elements $\beta_i := f_i(\widetilde{\alpha})\quad (i = 1, \cdots, k)$.

5. The hashing algorithm $\texttt{Hash}_{\text{FS}}$ generates a challenge $\widetilde{\lambda}_1$.

6. $\mathcal{P}$ computes

$$q(X)\ :=\ (X - \widetilde{\alpha})^{-1} \cdot \Big[[f_{\widetilde{\lambda}}(X) - f_{\widetilde{\lambda}}(\widetilde{\alpha})]\ +\ \sum_{i=1}^{k} \widetilde{\lambda}_1^{i} \cdot [f_i(X) - \beta_i] \Big]$$

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

$$\widetilde{\beta}\ :=\ \big[\sum_{j=1}^{k} \widetilde{\lambda}^{j-1} \cdot \beta_j \cdot e_j(\widetilde{\alpha})^{-1} \big]\ +\ \sum_{i=1}^{k} \widetilde{\lambda}_1^{i} \cdot \beta_i$$

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

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

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

3 The Snark in the batched form

We now describe the scheme in its basic form. i.e. without higher arity, custom gates or lookups. The proof in this version consists of 10 $\mathbb{G}_1$ elements and 20 $\mathbb{F}_p$ elements. The verification is constant and is dominated by a single pairing check. The Prover time is dominated by the 10 MSMs in $\mathbb{G}_1$ with a combined length of $22 \cdot |\text{Circuit}|$.

The protocol largely boils down to the Hadamard product protocol and the permutation argument. We use the protocol for batched divisibility to reduce the number of $\mathbb{G}_1$ elements arising from the multiple polynomial openings. This results in a smaller proof size and fewer MSMs than would be the case otherwise.

We note that rather than sending a commitment to the permutation polynomial $F(X)$ defined in step 3 below, it seemed more convenient for the Prover to send a commitment to the reverse of the polynomial

$$F^{\texttt{RShift}}(X) := X \cdot F(X) \pmod{X^N - 1},$$

i.e. the reverse of the cyclic right shift of $F(X)$ instead.

Protocol 3.1. The 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 n, $N = 3n$; The CRS $[\mathbf{g}_1,\ \mathbf{g}_1^{\mathbf{s}},\ \cdots,\ \mathbf{g}_1^{\mathbf{s}^N}]$, $[\mathbf{g}_1,\ \mathbf{g}_2^{\mathbf{s}}]$.
- Index sets $\mathcal{I}_{\mathcal{A}}$, $\mathcal{I}_{\mathcal{M}}$, $\mathcal{I}_{\texttt{Pub}}$ corresponding to the addition, multiplication gates and public inputs respectively.
- Characteristic polynomials $\chi_{\mathcal{I}_{\mathcal{A}}}(X)$, $\chi_{\mathcal{I}_{\mathcal{M}}}(X)$, $\chi_{\mathcal{I}_{\texttt{Pub}}}(X)$ and the reverses

$$\widehat{\chi}_{\mathcal{I}_{\mathcal{A}}}(X) := X^N \cdot \chi_{\mathcal{I}_{\mathcal{A}}}(X^{-1}) \ ,\ \widehat{\chi}_{\mathcal{I}_{\mathcal{M}}}(X) := X^N \cdot \chi_{\mathcal{I}_{\mathcal{M}}}(X^{-1}) \ ,\ \widehat{\chi}_{\mathcal{I}_{\texttt{Pub}}}(X) := X^N \cdot \chi_{\mathcal{I}_{\texttt{Pub}}}(X^{-1}).$$

- The polynomials

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

for a permutation $\boldsymbol{\sigma}$ of $[0,\ N-1]$.

Verifier preprocessed inputs:

$$\mathbf{g}_1,\ \mathbf{g}_1^{\mathbf{s}},\ \mathbf{g}_1^{\mathbf{s}^N} \in \mathbb{G}_1, \quad \mathbf{g}_2,\ \mathbf{g}_2^{\mathbf{s}} \in \mathbb{G}_2.$$

$$\widehat{\mathbf{C}}_{\mathcal{A}} := \mathbf{g}_1^{\widehat{\chi}_{\mathcal{I}_{\mathcal{A}}}(\mathbf{s})} \ , \ \widehat{\mathbf{C}}_{\mathcal{M}} := \mathbf{g}_1^{\widehat{\chi}_{\mathcal{I}_{\mathcal{M}}}(\mathbf{s})} \ , \ \widehat{\mathbf{C}}_{\mathtt{Pub}} := \mathbf{g}_1^{\widehat{\chi}_{\mathcal{I}_{\mathtt{Pub}}}(\mathbf{s})} \ , \ \mathbf{A}_{\mathtt{Pub}} := \mathbf{g}_1^{P_{\mathtt{pub}}(\mathbf{s})}$$

$$\mathbf{C}_{\boldsymbol{\sigma}} := \mathbf{g}_1^{P_{\boldsymbol{\sigma}}(\mathbf{s})} \ , \ \mathbf{C}_{\mathrm{id}} := \mathbf{g}_1^{P_{\mathrm{id}}(\mathbf{s})} \ , \ \mathbf{C}_1 := \mathbf{g}_1^{\sum_{i-0}^{N-1}\mathbf{s}^i}$$

Claim: The Prover knows polynomials $L(X),\ R(X),\ O(X)$ of degree $\leq n-1$ such that:

- $[L(X) \odot R(X) - O(X)] \ \odot \ \chi_{\mathcal{I}_{\mathcal{M}}}(X) \ = \ 0$
- $[L(X) + R(X) - O(X)] \ \odot \ \chi_{\mathcal{I}_{\mathcal{A}}}(X) \ = \ 0$
- The concatenated polynomial $\mathtt{Wires}(X) := L(X) + X^n \cdot R(X) + X^{2n} \cdot O(X)$ is stable under the action of the permutation $\boldsymbol{\sigma}$, i.e.

$$\mathtt{Coef}(\mathtt{Wires}\ ,\ \boldsymbol{\sigma}(i)) = \mathtt{Coef}(\mathtt{Wires}\ ,\ i) \ \ \forall\, i \in [0,\ N-1].$$

- $\mathtt{Wires}(X) \ \odot \ \chi_{\mathcal{I}_{\mathtt{Pub}}}(X) \ = \ P_{\mathtt{pub}}(X)$, or equivalently, $[\mathtt{Wires}(X) - P_{\mathtt{pub}}(X)] \odot \chi_{\mathcal{I}_{\mathtt{pub}}}(X) = 0$.

The proof generation:

Committing to the wire polynomials

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

$$\mathbf{A}_L \ := \ \mathbf{g}_1^{L(\mathbf{s})} \ , \ \mathbf{A}_R \ := \ \mathbf{g}_1^{R(\mathbf{s})} \ , \ \mathbf{A}_O \ := \ \mathbf{g}_1^{O(\mathbf{s})} \ , \ \mathbf{A}_{L,R} \ := \ \mathbf{g}_1^{L \odot R(\mathbf{s})}.$$

Committing to the reverse of the cyclic right shift of the permutation polynomial

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

3 $\mathcal{P}$ computes the polynomial

$$F(X) \ := \ \sum_{i=0}^{N-1} \prod_{j=0}^{i} \frac{\mathtt{Coef}(\mathtt{Wires}(X)\ ,\ j) + \delta_1 \cdot \boldsymbol{\sigma}(j) + \delta_2}{\mathtt{Coef}(\mathtt{Wires}(X)\ ,\ j) + \delta_1 \cdot j + \delta_2} \cdot X^i$$

and the right shift

$$F^{\mathtt{RShift}}(X) \ := \ X \cdot F(X) \ (\mathrm{mod}\ X^N - 1).$$

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

$$\mathbf{A}_{\widehat{F},1} := \mathbf{g}_1^{\widehat{F}^{\mathtt{RShift}}(\mathbf{s})},$$

where

$$\widehat{F}(X) := X^N \cdot F(X^{-1}) \ \ , \ \ \widehat{F}^{\mathtt{RShift}}(X) := X^N \cdot F^{\mathtt{RShift}}(X^{-1})$$

The twist

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

5 $\mathcal{P}$ sends the evaluation $\gamma_{L,R} \ := \ L \odot R(\gamma) \ \in \ \mathbb{F}_p$.

Aggregation of the multiple twisted products (for the Hadamard products)

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

7 $\mathcal{P}$ computes the polynomial $\widetilde{f}_{\gamma,\lambda}(X) :=$

$$L(\gamma \cdot X) \cdot \widehat{R}(X) \ + \ \lambda \cdot [L(\gamma \cdot X) + R(\gamma \cdot X) - O(\gamma \cdot X)] \cdot \widehat{\chi}_{\mathcal{I}_{\mathcal{A}}}(X)$$

$$+ \lambda^2 \cdot [L \odot R(\gamma \cdot X) - O(\gamma \cdot X)] \cdot \widehat{\chi}_{\mathcal{I}_{\mathcal{M}}}(X)$$

$$+ \lambda^3 \cdot \Big[\mathtt{Wires}(\gamma \cdot X) + \delta_1 \cdot P_{\mathtt{id}}(\gamma \cdot X) + \delta_2 \cdot \sum_{i=0}^{N-1} (\gamma \cdot X)^i\Big] \cdot \widehat{F}(X)$$

$$- \lambda^3 \cdot \Big[\mathtt{Wires}(\gamma \cdot X) + \delta_1 \cdot P_{\boldsymbol{\sigma}}(\gamma \cdot X) + \delta_2 \cdot \sum_{i=0}^{N-1} (\gamma \cdot X)^i\Big] \cdot \widehat{F}^{\mathtt{RShift}}(X)$$

$$+\lambda^4 \cdot \big[\mathtt{Wires}(\gamma \cdot X) - P_{\mathtt{pub}}(\gamma \cdot X)\big] \cdot \widehat{\chi}_{\mathcal{I}_{\mathtt{pub}}}(X) \ .$$

The low-degree part

8 $\mathcal{P}$ computes the the residue $\widetilde{f}_{\gamma,\lambda,-}(X) \ := \ \widetilde{f}_{\gamma,\lambda}(X) \ (\mathrm{mod}\ X^N)$ and sends the $\mathbb{G}_1$-element

$$\widetilde{\mathbf{A}}_{\gamma,\lambda,-} \ := \ \mathbf{g}_1^{\widetilde{f}_{\gamma,\lambda,-}(\mathbf{s})}.$$

The high-degree part

9 $\mathcal{P}$ computes the polynomial

$$\widetilde{f}_{\gamma,\lambda,+}(X) \ := \ \sum_{i=N+1}^{2N} \mathtt{Coef}(\widetilde{f}_{\gamma,\lambda} \ , \ i) \cdot X^{i-N-1}$$

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

Degree upper bounds

10 The hashing algorithm $\mathtt{Hash}_{\mathrm{FS}}$ generates a challenge $\widehat{\lambda}$.

11 $\mathcal{P}$ computes $\widehat{f}_{\widehat{\lambda}}(X) \ := \ X^{n-1} \cdot L(X^{-1}) \ + \ \widehat{\lambda} \cdot X^{n-1} \cdot R(X^{-1})$

$$+ \widehat{\lambda}^2 \cdot X^{n-1} \cdot O(X^{-1}) \ + \ \widehat{\lambda}^3 \cdot X^{N-1} \cdot \widetilde{f}_{\gamma,\lambda,-}(X^{-1})$$

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

$$\mathbf{A}_{\widehat{\lambda}} \ := \ \mathbf{g}_1^{\widehat{f}_{\widehat{\lambda}}(\mathbf{s})}.$$

The evaluation challenge

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

13 $\mathcal{P}$ sends the $\mathbb{F}_p$-elements

$$\beta_L := L(\alpha) \ , \ \beta_R := R(\alpha) \ , \ \beta_O := O(\alpha) \ , \ \beta_{L,R} := L \odot R(\alpha)$$

$$\widetilde{\beta}_{\gamma,\lambda,-} := \widetilde{f}_{\gamma,\lambda,-}(\alpha) \ , \ \widetilde{\beta}_{\gamma,\lambda,+} := \widetilde{f}_{\gamma,\lambda,+}(\alpha) \ , \ \beta_{\mathrm{id}} := P_{\mathrm{id}}(\alpha) \ , \ \beta_{\boldsymbol{\sigma}} := P_{\boldsymbol{\sigma}}(\alpha)$$

$$\beta_{\widehat{F}} := \widehat{F}(\alpha) = \alpha^N \cdot F(\alpha^{-1)} \ , \ \nu_{\gamma,\widehat{F}} := \widehat{F}(\gamma^{-1} \cdot \alpha) = (\gamma^{-1} \cdot \alpha)^N \cdot F(\alpha^{-1} \cdot \gamma)$$

$$\nu_{\gamma,\widehat{R}} := \widehat{R}(\gamma^{-1} \cdot \alpha) = (\gamma^{-1} \cdot \alpha)^N \cdot R(\gamma \cdot \alpha^{-1}) \ , \ \beta_{\mathtt{pub}} := P_{\mathtt{pub}}(\alpha).$$

14 The hashing algorithm $\mathtt{Hash}_{\mathrm{FS}}$ generates a challenge λ_1.

15 $\mathcal{P}$ computes the polynomial $f_{\lambda_1}(X) :=$

$$L(X) + \lambda_1 \cdot R(X) + \lambda_1^2 \cdot O(X) + \lambda_1^3 \cdot \widetilde{f}_{\gamma,\lambda,-}(X) + \lambda_1^4 \cdot P_{\mathrm{id}}(X) + \lambda_1^5 \cdot P_{\boldsymbol{\sigma}}(X)$$

$$+\lambda_1^6 \cdot \widetilde{f}_{\gamma,\lambda,+}(X) + \lambda_1^7 \cdot P_{\mathtt{pub}}(X) + \lambda_1^8 \cdot L \odot R(X) + \lambda_1^9 \cdot [\widehat{F}^{\mathtt{RShift}}(X)]$$

16 $\mathcal{P}$ computes the polynomial $f_{\gamma,\lambda,\alpha}(X) :=$

$$\beta_L \cdot \widehat{R}(\gamma^{-1} \cdot \alpha) \ + \ \lambda \cdot [\beta_L + \beta_R - \beta_O] \cdot \widehat{\chi}_{\mathcal{I}_{\mathcal{A}}}(X) \ + \ \lambda^2 \cdot [\beta_{L,R} - \beta_O] \cdot \widehat{\chi}_{\mathcal{I}_{\mathcal{M}}}(X)$$

$$+ \lambda^3 \cdot \big[[\beta_L + \beta_R \cdot \alpha^n + \ \beta_O \cdot \alpha^{2n}] + \delta_1 \cdot \beta_{\mathtt{id}} + \delta_2 \cdot \frac{\alpha^N - 1}{\alpha - 1}\big] \cdot \widehat{F}(\gamma^{-1} \cdot \alpha)$$

$$- \lambda^3 \cdot \big[[\beta_L + \beta_R \cdot \alpha^n + \beta_O \cdot \alpha^{2n}] + \delta_1 \cdot \beta_{\boldsymbol{\sigma}} + \delta_2 \cdot \frac{\alpha^N - 1}{\alpha - 1}\big] \cdot \widehat{F}^{\mathtt{RShift}}(X)$$

$$+ \lambda^4 \cdot [\beta_L - \beta_{\mathtt{pub}}] \cdot \widehat{\chi}_{\mathcal{I}_{\mathtt{pub}}}(X).$$

Batched divisibility

We use the following notations for brevity:

(i). $h_1(X) \ := \ L \odot R(X) - \gamma_{L,R} \ , \ e_1(X) \ := \ X - \gamma \ ,$

(ii). $h_2(X) \ := \ \Big[\widetilde{f}_{\gamma,\lambda,-}(X) + \gamma_{L,R} \cdot X^N + (\gamma^{-1} \cdot \alpha)^{N+1} \cdot \widetilde{f}_{\gamma,\lambda,+}(X)\Big] \ - \ f_{\gamma,\lambda,\alpha}(X)$

$e_2(X) \ := \ \gamma \cdot X - \alpha \ ,$

(iii). $h_3(X) \ := \ f_{\lambda_1}(X) - f_{\lambda_1}(\alpha) \ , \ e_3(X) \ := \ X - \alpha$

(iv). $h_4(X) \ := \ \widehat{f}_{\widehat{\lambda}}(X) \ - \ \widehat{f}_{\widehat{\lambda}}(\alpha^{-1}) \ , \ e_4(X) \ := \ X - \alpha^{-1}$

(v). $h_5(X) \ := \ \widehat{F}^{\mathtt{RShift}}(X) \ , \ e_5(X) \ := \ X.$

(vi). $h_6(X) := \widehat{F}^{\texttt{RShift}}(X) - \left[(\alpha^{-1}\cdot\gamma)\cdot\nu_{\gamma,\widehat{F}} + (\gamma^{-1}\cdot\alpha)^N - 1\right]$, $e_6(X) := \gamma\cdot X - \alpha$

(vii). $h_7(X) := R(X) - (\gamma\cdot\alpha^{-1})^N\cdot\nu_{\gamma,\widehat{R}}$, $e_7(X) := \alpha\cdot X - \gamma$.

17 The hashing algorithm $\texttt{Hash}_{\text{FS}}$ generates a challenge $\widetilde{\lambda}$.

18 The Prover $\mathcal{P}$ computes

$$h_{\widetilde{\lambda}}(X) := \sum_{i=1}^{7}\widetilde{\lambda}^{i-1}\cdot e_i(X)^{-1}\cdot h_i(X)$$

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

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

19 The hashing algorithm $\texttt{Hash}_{\text{FS}}$ generates a challenge $\widetilde{\alpha}$.

20 $\mathcal{P}$ sends the $\mathbb{F}_p$-elements $\beta_i := h_i(\widetilde{\alpha}) \quad (i = 1,\cdots,7)$.

21 The hashing algorithm $\texttt{Hash}_{\text{FS}}$ generates a challenge $\widetilde{\lambda}_1$.

22 $\mathcal{P}$ computes the polynomial

$$\widetilde{q}(X) := (X-\widetilde{\alpha})^{-1}\cdot\left[[h_{\widetilde{\lambda}}(X) - h_{\widetilde{\lambda}}(\widetilde{\alpha})] + \sum_{i=1}^{7}\widetilde{\lambda}_1^i\cdot[h_i(X)-\beta_i]\right]$$

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

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

The verification

23 The Verifier $\mathcal{V}$ computes the $\mathbb{F}_p$-element $\beta_{\texttt{Wires}} := \beta_L + \alpha^n\cdot\beta_R + \alpha^{2n}\cdot\beta_O$ and the $\mathbb{G}_1$-element

$$\mathbf{A}_{\lambda,\alpha} := \left[\mathbf{g}_1^{\beta_L\cdot\nu_{\gamma,\widehat{R}}}\right]\cdot\left[(\widehat{\mathbf{C}}_{\mathcal{A}})^{\lambda\cdot[\beta_L+\beta_R-\beta_0]}\right]\cdot\left[(\widehat{\mathbf{C}}_{\mathcal{M}})^{\lambda^2\cdot[\beta_{L,R}-\beta_0]}\right]\cdot$$

$$\cdot\left[(\mathbf{g}_1)^{\lambda^3\cdot\nu_{\gamma,\widehat{F}}\cdot[\beta_{\texttt{Wires}}+\delta_1\cdot\beta_{\text{id}}+\delta_2\cdot\frac{\alpha^N-1}{\alpha-1}]}\right]\cdot\left[(\mathbf{A}_{\widehat{F},1})^{-\lambda^3\cdot[\beta_{\texttt{Wires}}+\delta_1\cdot\beta_{\boldsymbol{\sigma}}+\delta_2\cdot\frac{\alpha^N-1}{\alpha-1}]}\right]\cdot\left[(\widehat{\mathbf{C}}_{\texttt{pub}})^{\lambda^4\cdot[\beta_L-\beta_{\texttt{pub}}]}\right]$$

24 $\mathcal{V}$ computes the $\mathbb{F}_p$-element

$$\beta_{\lambda_1} := \beta_L + \lambda_1\cdot\beta_R + \lambda_1^2\cdot\beta_O + \lambda_1^3\cdot\widetilde{\beta}_{\gamma,\lambda,-} + \lambda_1^4\cdot\beta_{\text{id}} + \lambda_1^5\cdot\beta_{\boldsymbol{\sigma}}$$

$$+ \lambda_1^6\cdot\widetilde{\beta}_{\gamma,\lambda,+} + \lambda_1^7\cdot\beta_{\texttt{pub}} + \lambda_1^8\cdot\beta_{L,R} + \lambda_1^9\cdot[\alpha^{-1}\cdot\beta_{\widehat{F}} + \alpha^N - 1]$$

25 $\mathcal{V}$ computes the $\mathbb{G}_1$ element

$$\mathbf{A}_{\lambda_1} := \mathbf{A}_L\cdot\mathbf{A}_R^{\lambda_1}\cdot\mathbf{A}_O^{\lambda_1^2}\cdot\widetilde{\mathbf{A}}_{\gamma,\lambda,-}^{\lambda_1^3}\cdot\mathbf{C}_{\text{id}}^{\lambda_1^4}\cdot\mathbf{C}_{\boldsymbol{\sigma}}^{\lambda_1^5}\cdot\widetilde{\mathbf{A}}_{\gamma,\lambda,+}^{\lambda_1^6}\cdot\mathbf{A}_{\texttt{pub}}^{\lambda_1^7}\cdot\mathbf{A}_{L,R}^{\lambda_1^8}\cdot\mathbf{A}_{\widehat{F},1}^{\lambda_1^9}$$

26 $\mathcal{V}$ computes the $\mathbb{F}_p$-element

$$\beta_{\widehat{\lambda}} \;:=\; \alpha^{1-n} \cdot \beta_L + \widehat{\lambda} \cdot \alpha^{1-n} \cdot \beta_R + \widehat{\lambda}^2 \cdot \alpha^{1-n} \cdot \beta_O + \widehat{\lambda}^3 \cdot \alpha^{1-N} \cdot \widetilde{\beta}_{\gamma,\lambda,-}$$

Batched divisibility checks:

27 $\mathcal{V}$ computes the linear polynomials $e_1(X)$, $\cdots$, $e_7(X)$ and the $\mathbb{G}_1$ elements $\mathbf{a}_1$, $\cdots$, $\mathbf{a}_7$ given by:

$$e_1(X) := X - \gamma \;\;,\;\; e_2(X) := \gamma \cdot X - \alpha \;\;,\;\; e_3(X) := X - \alpha \;\;,\;\; e_4(X) := X - \alpha^{-1}$$

$$e_5(X) := X \;\;,\;\; e_6(X) := \gamma \cdot X - \alpha \;\;,\;\; e_7(X) := \alpha \cdot X - \gamma$$

$$\mathbf{a}_1 := \mathbf{A}_{L,R} \cdot \mathbf{g}_1^{-\gamma_{L,R}} \quad , \quad \mathbf{a}_2 := \big[\widetilde{\mathbf{A}}_{\gamma,\lambda,-} \cdot (g_1^{\mathbf{s}^N})^{\gamma_{L,R}} \cdot \widetilde{\mathbf{A}}_{\gamma,\lambda,-}^{(\gamma^{-1}\cdot\alpha)^{N+1}}\big] \cdot \mathbf{A}_{\lambda,\alpha}^{-1}$$

$$\mathbf{a}_3 := \mathbf{A}_{\lambda_1} \cdot \mathbf{g}_1^{-\beta_{\lambda_1}} \quad , \quad \mathbf{a}_4 := \mathbf{A}_{\widehat{\lambda}} \cdot \mathbf{g}_1^{-\beta_{\widehat{\lambda}}} \quad , \quad \mathbf{a}_5 := \mathbf{A}_{\widehat{F},1}$$

$$\mathbf{a}_6 := \mathbf{A}_{\widehat{F},1} \cdot \mathbf{g}_1^{-[(\alpha^{-1}\cdot\gamma)\cdot\nu_{\gamma,\widehat{F}}+(\gamma^{-1}\cdot\alpha)^N-1]} \quad , \quad \mathbf{a}_7 := \mathbf{A}_R \cdot \mathbf{g}_1^{-(\gamma\cdot\alpha^{-1})^N\cdot\nu_{\gamma,\widehat{R}}}$$

28 $\mathcal{V}$ computes the $\mathbb{F}_p$-element

$$\widetilde{\beta} \;:=\; \big[\sum_{j=1}^{7} \widetilde{\lambda}^{j-1} \cdot \beta_j \cdot e_j(\widetilde{\alpha})^{-1}\big] + \sum_{i=1}^{7} \widetilde{\lambda}_1^i \cdot \beta_i$$

The pairing check

29 $\mathcal{V}$ verifies the equation:

$$\widetilde{\mathbf{Q}}^{\mathbf{s}-\widetilde{\alpha}} \;\stackrel{?}{=}\; \mathbf{B}_{\widetilde{\lambda}} \cdot \big[\prod_{i=1}^{7} (\mathbf{a}_i)^{\widetilde{\lambda}_1^i}\big] \cdot \mathbf{g}_1^{-\widetilde{\beta}}$$

via the pairing check

$$\mathbf{e}\Big(\widetilde{\mathbf{Q}} \,,\, \mathbf{g}_2^{\mathbf{s}}\Big) \;\stackrel{?}{=}\; \mathbf{e}\Big(\mathbf{B}_{\widetilde{\lambda}} \cdot \big[\prod_{i=1}^{7} (\mathbf{a}_i)^{\widetilde{\lambda}_1^i}\big] \cdot \mathbf{g}_1^{-\widetilde{\beta}} \cdot \big(\widetilde{\mathbf{Q}}\big)^{\widetilde{\alpha}^{-1}} \,,\, \mathbf{g}_2\Big).$$

□

This version has 10 $\mathbb{G}_1$ elements and 20 $\mathbb{F}_p$-elements. The proof generation runtime is dominated by

- The 10 MSMs in $\mathbb{G}_1$ with a total MSM length of $22 \cdot |\text{Circuit}|$.

- The computation of the sum of six polynomial products in Step 7 (to a lesser extent than the MSMs).

We note that it is possible to reduce the total MSM length (and hence, the Prover time) at the cost of more $\mathbb{G}_1$ elements in the proof.

4 The underlying subprotocols

In this section, we describe the subprotocols underpinning the Snark. Unlike in the case of the Snark, we have made no particular effort to make these modular subprotocols efficient, beyond ensuring that the proof sizes and verification times are constant and the Prover times quasi-linear. The purpose of this section is merely expository.

The first four protocols here are neither new nor original and have only been included for the sake of clarity and because we invoke them repeatedly for the subsequent protocols. We note that - as is usually the case with Snarks - we have not invoked the subprotocols in the Snark in a modular fashion, because that would increase the proof size as well as the proof generation runtime.

We start out with the simple protocol for the knowledge of the exponent, which is an argument of knowledge for the following relation.

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

Protocol 4.1. *Proof of knowledge of the exponent with base* $\mathbf{g}_1$ (PoKE*):
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 $[\mathbf{g}_1,\ \mathbf{g}_1^{\mathbf{s}},\ \cdots,\ \mathbf{g}_1^{\mathbf{s}^M}]\ ,\ [\mathbf{g}_2,\ \mathbf{g}_2^{\mathbf{s}}]$
Inputs: Element $\mathbf{a} \in \mathbb{G}_1$
Claim: The Prover knows a polynomial $f(X) \in \mathbb{F}_p[X]$ such that $\mathbf{g}_1^{f(\mathbf{s})} = \mathbf{a}$.
1. The hashing algorithm $\mathtt{Hash}_{\mathrm{FS}}$ generates a challenge $\alpha \in \mathbb{F}_p^*$.
2. $\mathcal{P}$ sends $\beta := f(\alpha)$ and the $\mathbb{G}_1$-element $\mathbf{Q} := \mathbf{g}_1^{[f(\mathbf{s})-\beta]/[\mathbf{s}-\alpha]}$.
3. $\mathcal{V}$ verifies the equation
$$\mathbf{e}(\mathbf{Q}\ ,\ \mathbf{g}_2^{\mathbf{s}-\alpha}) \stackrel{?}{=} \mathbf{e}(\mathbf{a} \cdot \mathbf{g}_1^{-\beta}\ ,\ \mathbf{g}_2).$$
□

$$\mathcal{R}_{\mathtt{Prod}}[\mathbf{g}_1,\ (\mathbf{a}_1, \mathbf{a}_2),\ \mathbf{a}_{1,2}] = \left\{ \begin{array}{l} \big((\mathbf{a}_1,\ \mathbf{a}_2,\ \mathbf{a}_{1,2} \in \mathbb{G}_1)\ ,\ f_1(X),\ f_2(X) \in\ \mathbb{F}_p[X]\big) : \\ \mathbf{g}_1^{f_1(\mathbf{s})} = \mathbf{a}_1\ ,\ \mathbf{g}_1^{f_2(\mathbf{s})} = \mathbf{a}_2\ ,\ \mathbf{g}_1^{f_1(\mathbf{s})\cdot f_2(\mathbf{s})} = \mathbf{a}_{1,2} \end{array} \right\}$$

Protocol 4.2. *Proof of product* (PoProd):
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 $[\mathbf{g}_1,\ \mathbf{g}_1^{\mathbf{s}},\ \cdots,\ \mathbf{g}_1^{\mathbf{s}^M}]\ ,\ [\mathbf{g}_2,\ \mathbf{g}_2^{\mathbf{s}}]$
Inputs: Elements $\mathbf{a}_1,\ \mathbf{a}_2,\ \mathbf{a}_{1,2} \in \mathbb{G}_1$
Claim: The Prover knows polynomials $f_1(X),\ f_2(X)\ \in\ \mathbb{F}_p[X]$ such that
$$\mathbf{a}_1 = \mathbf{g}_1^{f_1(\mathbf{s})}\ \ ,\ \ \mathbf{a}_2 = \mathbf{g}_1^{f_2(\mathbf{s})}\ \ ,\ \ \mathbf{a}_{1,2} = \mathbf{g}_1^{f_1(\mathbf{s})\cdot f_2(\mathbf{s})}.$$
1. The hashing algorithm $\mathtt{Hash}_{\mathrm{FS}}$ generates a challenge $\alpha \in \mathbb{F}_p^*$.
2. $\mathcal{P}$ sends the $\mathbb{F}_p$-elements $\beta_1 := f_1(\alpha)$, $\beta_2 := f_2(\alpha)$ and the $\mathbb{G}_1$-elements
$$\mathbf{Q}_1 := \mathbf{g}_1^{[f_1(\mathbf{s})-\beta_1]/[\mathbf{s}-\alpha]}\ ,\ \mathbf{Q}_2 := \mathbf{g}_1^{[f_2(\mathbf{s})-\beta_2]/[\mathbf{s}-\alpha]}\ ,\ \mathbf{Q}_{1,2} := \mathbf{g}_1^{[f_1(\mathbf{s})\cdot f_2(\mathbf{s})-\beta_1\cdot\beta_2]/[\mathbf{s}-\alpha]}.$$

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

$$\mathbf{e}(\mathbf{Q}_1\ ,\ \mathbf{g}_2^{\mathbf{s}-\alpha}) \stackrel{?}{=} \mathbf{e}(\mathbf{a}_1 \cdot \mathbf{g}_1^{-\beta_1}\ ,\ \mathbf{g}_2)\ ,\ \mathbf{e}(\mathbf{Q}_2\ ,\ \mathbf{g}_2^{\mathbf{s}-\alpha}) \stackrel{?}{=} \mathbf{e}(\mathbf{a}_1 \cdot \mathbf{g}_1^{-\beta_2}\ ,\ \mathbf{g}_2),$$

$$\mathbf{e}(\mathbf{Q}_{1,2}\ ,\ \mathbf{g}_2^{\mathbf{s}-\alpha})\ \stackrel{?}{=}\ \mathbf{e}(\mathbf{a}_{1,2} \cdot \mathbf{g}_1^{-\beta_1 \cdot \beta_2}\ ,\ \mathbf{g}_2).$$

□

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

Protocol 4.3. *Proof of knowledge of the exponent* (PoKE):
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 $[\mathbf{g}_1,\ \mathbf{g}_1^{\mathbf{s}},\ \cdots,\ \mathbf{g}_1^{\mathbf{s}^M}]\ ,\ [\mathbf{g}_2,\ \mathbf{g}_2^{\mathbf{s}}]$

Inputs: Elements $\mathbf{a}, \mathbf{b} \in \mathbb{G}_1$
Claim: The Prover knows polynomials $e(X), f(X) \in \mathbb{F}_p[X]$ such that

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

1. The Prover $\mathcal{P}$ sends a proof of $\mathtt{PoKE}^*[\mathbf{g}_1,\ \mathbf{a}]$.
2. $\mathcal{P}$ sends $\widetilde{\mathbf{g}}_1 := \mathbf{g}_1^{f(\mathbf{s})} \in \mathbb{G}_1$ and a proof of $\mathtt{PoKE}^*[\mathbf{g}_1,\ \widetilde{\mathbf{g}}_1]$.
3. $\mathcal{P}$ sends a proof of $\mathtt{PoProd}[\mathbf{g}_1,\ (\widetilde{\mathbf{g}}_1,\ \mathbf{a}),\ \mathbf{b}]$.
4. The Verifier $\mathcal{V}$ accepts iff the $\mathtt{PoKE}^*$s and the PoProd are valid. □

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

Protocol 4.4. *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 $[\mathbf{g}_1,\ \mathbf{g}_1^{\mathbf{s}},\ \cdots,\ \mathbf{g}_1^{\mathbf{s}^M}]\ ,\ [\mathbf{g}_2,\ \mathbf{g}_2^{\mathbf{s}}]$

Inputs: $\mathbb{G}_1$ elements $\mathbf{a}$, $\mathbf{a}_\gamma$; $\mathbb{F}_p$-element γ
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 $\mathtt{Hash}_{\mathrm{FS}}$ 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

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

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

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

□

4.1 Proof of reverse

We describe the subprotocol that allows a Prover to show that given two committed polynomials $f(X),\ h(X)$ and a public integer n, we have $h(X) = X^n \cdot f(X^{-1})$. It hinges on the simple observation that if, for a randomly generated challenge α, the equation

$$h(\alpha^{-1}) = \alpha^{-n}\cdot f(\alpha)$$

holds, then with overwhelming probability, $h(X) = X^n \cdot f(X^{-1})$. We note that this protocol is possible if and only if $\deg(f) \leq n$, since otherwise the rational function $X^n \cdot f(X^{-1})$ is not a polynomial and hence, computing $\mathbf{g}_1^{\mathbf{s}^n\cdot f(\mathbf{s}^{-1})}$ is infeasible.

$$\mathcal{R}_{\texttt{Rev}}[\mathbf{g}_1,\ n,\ (\mathbf{a}_1,\mathbf{a}_2)] = \{(\mathbf{a}_1,\mathbf{a}_2\in\mathbb{G}_1,\ n\in\mathbb{Z}),\ f(X)\in\mathbb{F}_p[X]) : \mathbf{g}_1^{f(\mathbf{s})} = \mathbf{a}_1\ ,\ \mathbf{g}_1^{\mathbf{s}^n\cdot f(\mathbf{s}^{-1})} = \mathbf{a}_2\}$$

Protocol 4.5. *Proof of reverse* (PoRev):
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 $[\mathbf{g}_1,\ \mathbf{g}_1^{\mathbf{s}},\ \cdots,\ \mathbf{g}_1^{\mathbf{s}^M}]\ ,\ [\mathbf{g}_2,\ \mathbf{g}_2^{\mathbf{s}}]$

Inputs: Elements $\mathbf{a},\ \widehat{\mathbf{a}}\ \in\mathbb{G}_1$; a public integer n
Claim: The Prover knows polynomial $f(X)\ \in\mathbb{F}_p[X]$ such that

$$\mathbf{a} = \mathbf{g}_1^{f(\mathbf{s})}\ \ ,\ \ \widehat{\mathbf{a}} = \mathbf{g}_1^{\mathbf{s}^n\cdot f(\mathbf{s}^{-1})}.$$

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

2. 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 $\beta := f(\alpha)$.

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

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

via the (batchable) pairing checks

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

□

4.2 Protocol for the degree upper bound

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

Protocol 4.6. *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 $[\mathbf{g}_1,\ \mathbf{g}_1^{\mathbf{s}},\ \cdots,\ \mathbf{g}_1^{\mathbf{s}^M}]\ ,\ [\mathbf{g}_2,\ \mathbf{g}_2^{\mathbf{s}}]$

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

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

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

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

via the (batchable) pairing checks

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

□

4.3 The 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:

$$f_{j,1} \odot f_{j,2}(X) = f_{j,1,2}(X)\ \text{ for }\ j = 1, \cdots, k\ ,$$

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

$$f_\lambda(X) := \sum_{j=1}^{k} \lambda^{j-1} \cdot f_{j,1}(\gamma \cdot X) \cdot X^N \cdot f_{j,2}(X^{-1})$$

has coefficient $\sum\limits_{j=1}^{k} \lambda^{j-1} \cdot f_{j,1,2}(\gamma)$ at the position X^N. This boils down to expressing the difference

$$f_\lambda(X)\ \ -\ \ \big[\sum_{j=1}^{k} \lambda^{j-1} \cdot f_{j,1,2}(\gamma)\big] \cdot X^N$$

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}.

$$\mathcal{R}_{\texttt{HadProd}}[\mathbf{g}_1,\ (\mathbf{a}_1,\mathbf{a}_2),\ \mathbf{a}_{1,2}] = \left\{ \begin{array}{l} \big((\mathbf{a}_1,\ \mathbf{a}_2,\ \mathbf{a}_{1,2} \in \mathbb{G}_1)\ ,\ f_1(X),\ f_2(X) \in\ \mathbb{F}_p[X]\big) : \\ \mathbf{g}_1^{f_1(\mathbf{s})} = \mathbf{a}_1\ ,\ \mathbf{g}_1^{f_2(\mathbf{s})} = \mathbf{a}_2\ ,\ \mathbf{g}_1^{f_1 \odot f_2(\mathbf{s})} = \mathbf{a}_{1,2} \end{array} \right\}$$

Protocol 4.7. *Proof of the Hadamard Product* (`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 such that $\mathcal{V}$ stores the element $\mathbf{g}_1^{\mathbf{s}^N}$
Inputs: Elements $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_{1,2}\ \in\ \mathbb{G}_1$;
Claim: The Prover knows polynomials $f_1(X)$, $f_2(X)$ such that:

$$\mathbf{g}_1^{f_1(\mathbf{s})} = \mathbf{a}_1\ \ ,\ \ \mathbf{g}_1^{f_2(\mathbf{s})} = \mathbf{a}_2\ \ ,\ \ \mathbf{g}_1^{f_1 \odot f_2(\mathbf{s})} = \mathbf{a}_{1,2}$$

($f_1 \odot f_2$ denotes the Hadamard product)

Proof generation algorithm

1. The hashing algorithm $\texttt{Hash}_{\text{FS}}$ generates a challenge γ.
2. $\mathcal{P}$ sends the $\mathbb{F}_p$ element $\gamma_{1,2} :=\ f_1 \odot f_2(\gamma)$
3. $\mathcal{P}$ computes the product

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

The low degree part

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

$$f_{\gamma,-}(X)\ := f_\gamma(X)\ (\text{mod}\ X^N)$$

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

$$\mathbf{a}_{\gamma,-}\ :=\ \mathbf{g}_1^{f_{\gamma,-}(\mathbf{s})}.$$

Degree upper bound on the low degree part

5. $\mathcal{P}$ computes the residue

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

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

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

High degree part

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

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

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

$$\mathbf{a}_{\gamma,+} \ := \ \mathbf{g}_1^{f_{\gamma,+}(\mathbf{s})}.$$

The evaluation challenge

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

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

$$\beta_{\gamma,1} := \ f_1(\gamma \cdot X) \ , \ \widehat{\beta}_2 := f_2(\alpha^{-1}) \ , \ \beta_{\gamma,-} := \ f_{\gamma,-}(\alpha) \ , \ \beta_{\gamma,+} := \ f_{\gamma,+}(\alpha)$$

The batched divisibility (BatchDiv) subprotocol

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

(i). $h_1(X) := \ f_1(X) - \beta_{\gamma,1} \ , \ e_1(X) := \ X - \gamma^{-1} \cdot \alpha$

(ii). $h_2(X) := \ f_2(X) - \widehat{\beta}_2 \ , \ e_2(X) := \ X - \alpha^{-1}$

(iii) $h_3(X) := \ f_{\gamma,-}(X) - \beta_{\gamma,-} \ , \ e_3(X) := \ X - \alpha$

(iv) $h_4(X) := \ f_{\gamma,+}(X) - \beta_{\gamma,+} \ , \ e_4(X) := \ X - \alpha$

(v) $h_5(X) := \ \widehat{f}_{\gamma,-}(X) - \alpha^{1-N} \cdot \beta_{\gamma,+} \quad e_5(X) := \ X - \alpha^{-1}$

(vi) $h_6(X) := \ f_1 \odot f_2(X) - \gamma_{1,2} \ , \ e_6(X) := \ X - \gamma$

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

11. $\mathcal{P}$ sends a proof for the protocol $\texttt{BatchDiv}$ on the tuple $[\mathbf{g}_1, \ (\mathbf{b}_i)_{i=1}^6, \ (e_i(X))_{i=1}^6]$

12. The Verifier $\mathcal{V}$ computes the following $\mathbf{G}_1$ elements:

$$\mathbf{b}_1 := \ \mathbf{a}_1 \cdot \mathbf{g}_1^{-\beta_{\gamma,1}} \ , \ \mathbf{b}_2 := \ \mathbf{a}_2 \cdot \mathbf{g}_1^{-\widehat{\beta}_2} \ , \ \mathbf{b}_3 := \ \mathbf{a}_{\gamma,-} \cdot \mathbf{g}_1^{-\beta_{\gamma,-}} \ , \ \mathbf{b}_4 := \ \mathbf{a}_{\gamma,+} \cdot \mathbf{g}_1^{-\beta_{\gamma,+}}$$

$$\mathbf{b}_5 := \ \mathbf{b}_{\gamma,-} \cdot \mathbf{g}_1^{-\alpha^{1-N} \cdot \beta_{\gamma,+}} \ , \ \mathbf{b}_6 := \ \mathbf{a}_{1,2} \cdot \mathbf{g}_1^{-\gamma_{1,2}}.$$

13. $\mathcal{V}$ verifies the subprotocol $\texttt{BatchDiv}$.

14. $\mathcal{V}$ verifies the equation $\beta_{\gamma,1} \cdot \alpha^N \cdot \widehat{\beta}_2 \ \stackrel{?}{=} \ \beta_{\gamma,-} + \alpha^N \cdot \gamma_{1,2} + \alpha^{N+1} \cdot \beta_{\gamma,+}$. □

4.4 Proof of cyclic right shift

We describe a protocol to show that for a public integer N and committed polynomials $f(X)$, $f_1(X)$, the polynomials are of degree $\leq N-1$ and the latter is a cyclic right shift of the first in the sense that:

$$f_1(X) \ = \ \mathtt{Coef}(f\ ,\ N-1) + \sum_{j=0}^{N-2} \mathtt{Coef}(f\ ,\ j) \cdot X^{j+1} \ \equiv \ X \cdot f(X) \ (\text{mod } X^N - 1)$$

The two polynomials are linked by the equation

$$X \cdot f(X) - f_1(X) \ = \ \mathtt{Coef}(f\ ,\ N-1) \cdot [X^N - 1].$$

and hence, no expensive polynomial multiplication or division is necessary.

$$\mathcal{R}_{\mathtt{RShift}}[\mathbf{g}_1,\ (\mathbf{a},\mathbf{b})] = \left\{ \begin{array}{l} \big((\mathbf{a},\ \mathbf{b} \ \in \ \mathbb{G}_1) \\ f(X) \ \in \ \mathbb{F}_p[X] \text{ with } \deg(f) \leq N \\ \mathbf{a} = \mathbf{g}_1^{f(\mathbf{s})} \ \ , \ \ \mathbf{b} = \mathbf{g}_1^{f^{\mathtt{RShift}}(\mathbf{s})} \end{array} \right\}$$

Protocol 4.8. *Proof of cyclic right shift* (`PoRShift`)
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 such that $\mathcal{V}$ stores the element $\mathbf{g}_1^{\mathbf{s}^N}$

Inputs: Elements $a, b \in \mathbb{G}_1$.
Claim: The Prover knows a polynomial $f(X)$ of degree $\leq N-1$ such that

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

where

$$f^{\mathtt{RShift}}(X) \ := \ X \cdot f(X) \ (\text{mod } X^N - 1).$$

1. The Prover $\mathcal{P}$ sends the $\mathbb{F}_p$-element

$$c_{N-1} \ := \ \mathtt{Coef}(f\ ,\ N-1).$$

2. $\mathcal{P}$ sends proofs of $\mathtt{PoDegUp}[\mathbf{g}_1,\ \mathbf{a},\ N-1]$, $\mathtt{PoDegUp}[\mathbf{g}_1,\ \mathbf{b},\ N-1]$.
(degree upper bounds)
3. The Verifier $\mathcal{V}$ verifies the proofs of degree upper bounds and the equation

$$\mathbf{e}(\mathbf{a}\ ,\ \mathbf{g}_2^{\mathbf{s}}) \cdot \mathbf{e}(\mathbf{b}^{-1}\ ,\ \mathbf{g}_1) \stackrel{?}{=} \mathbf{e}((\mathbf{g}_1^{\mathbf{s}^N - 1}\ ,\ \mathbf{g}_2^{c_{N-1}}).$$

□

4.5 The permutation protocol

We now describe the permutation protocol that the Snark hinges upon. For a committed permutation $\boldsymbol{\sigma} : [0,\ N-1] \longrightarrow [0,\ N-1]$ and two committed polynomials $f(X)$, $\widetilde{f}(X)$, the Prover succinctly shows that the polynomials are of degree $\leq N-1$ and are linked by the permutation $\boldsymbol{\sigma}$ in the following sense:

$$\texttt{Coef}(\widetilde{f}\ ,\ j)\ =\ \texttt{Coef}(f\ ,\ \boldsymbol{\sigma}(j))\ \ \forall\ j.$$

The permutation $\boldsymbol{\sigma}$ is committed via the KZG10 commitment to the polynomial

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

$\texttt{PoPerm}$ (*Proof of permutation*)

$$\mathcal{R}_{\texttt{Perm}}[\mathbf{g}_1,\ (\mathbf{a}, \mathbf{C}_{\boldsymbol{\sigma}}),\ \mathbf{b})] = \left\{ \begin{array}{l} \big((\mathbf{a},\ \mathbf{C}_{\boldsymbol{\sigma}},\ \mathbf{b}\ \in\ \mathbb{G}_1) \\ f(X)\ \in\ \mathbb{F}_p[X] \text{ with } \deg(f) \leq N \\ \text{Permutation } \boldsymbol{\sigma}:\ [0,\ N-1]\ \longrightarrow\ [0,\ N-1]\big): \\ \\ \mathbf{a} = \mathbf{g}_1^{f(\mathbf{s})}\ \ ,\ \ \mathbf{b} = \mathbf{g}_1^{f^{\boldsymbol{\sigma}}(\mathbf{s})}\ \ ,\ \ \mathbf{C}_{\boldsymbol{\sigma}} = \mathbf{g}_1^{\sum_{k=0}^{N-1} \boldsymbol{\sigma}(k)\cdot \mathbf{s}^k} \end{array} \right\}$$

Protocol 4.9. *Proof of permutation* ($\texttt{PoPerm}$)

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 such that $\mathcal{V}$ stores the element $\mathbf{g}_1^{\mathbf{s}^N}$

Commitments to the polynomials

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

for a permutation $\boldsymbol{\sigma}$ of $[0,\ N-1]$.

The Prover stores the polynomial $P_{\boldsymbol{\sigma}}(X)$. The Verifier stores the [KZG10] commitments

$$\mathbf{g}_1^{\mathbf{s}^N}\ \ ,\ \ \mathbf{C}_{\boldsymbol{\sigma}} := \mathbf{g}_1^{P_{\boldsymbol{\sigma}}(\mathbf{s})}\ \ ,\ \ \mathbf{C}_{\text{id}} := \mathbf{g}_1^{P_{\text{id}}(\mathbf{s})}\ \ ,\ \ \mathbf{C}_1 := \mathbf{g}_1^{\sum_{i-0}^{N-1} \mathbf{s}^i}$$

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

Claim: The Prover knows a polynomial $f(X)$ of degree $\leq N-1$ such that

$$\mathbf{a} = \mathbf{g}_1^{f(\mathbf{s})}\ \ ,\ \ \mathbf{b} = \mathbf{g}_1^{f^{\boldsymbol{\sigma}}(\mathbf{s})}\quad \text{where } f^{\boldsymbol{\sigma}}(X) := \sum_{k=0}^{N-1} \texttt{Coef}(f\ ,\ \boldsymbol{\sigma}(k)) \cdot X^k.$$

1. $\mathcal{P}$ sends a proof of $\texttt{PoDegUp}[\mathbf{g}_1,\ \mathbf{a},\ N-1]$. (proof of degree upper bound)
2. The hashing algorithm $\texttt{Hash}_{\text{FS}}$ generates two challenges $\delta_1,\ \delta_2$.
3. $\mathcal{P}$ computes the polynomials

$$f_{\delta_1,\delta_2}(X) := f(X) + \delta_1 \cdot \Big(\sum_{i=0}^{N-1} i \cdot X^i\Big) + \delta_2 \cdot \sum_{i=0}^{N-1} X^i\ \ ,\ \ \widetilde{f}_{\delta_1,\delta_2}(X) := f^{\boldsymbol{\sigma}}(X) + \delta_1 \cdot P_{\boldsymbol{\sigma}}(X) + \delta_2 \cdot \sum_{i=0}^{N-1} X^i$$

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

$$\mathbf{a}_{\delta_1,\delta_2} := \mathbf{g}_1^{f_{\delta_1,\delta_2}(\mathbf{s})} = \mathbf{a} \cdot \mathbf{C}_{\text{id}}^{\delta_1} \cdot \mathbf{C}_1^{\delta_2}\ \ ,\ \ \mathbf{b}_{\delta_1,\delta_2} := \mathbf{g}_1^{\widetilde{f}_{\delta_1,\delta_2}(\mathbf{s})} = \mathbf{b} \cdot \mathbf{C}_{\boldsymbol{\sigma}}^{\delta_1} \cdot \mathbf{C}_1^{\delta_2}.$$

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

$$F_{\delta_1,\delta_2}(X) := \sum_{i=0}^{N-1} \prod_{j=0}^{i} \frac{\mathtt{Coef}(f^{\boldsymbol{\sigma}}\ ,\ j) + \delta_1 \cdot \boldsymbol{\sigma}(j) + \delta_2}{\mathtt{Coef}(f\ ,\ j) + \delta_1 \cdot j + \delta_2} \cdot X^i$$

and the right shift

$$F^{\mathtt{RShift}}_{\delta_1,\delta_2}(X) := X \cdot F_{\delta_1,\delta_2}(X) \ (\mathrm{mod}\ X^N - 1).$$

Note that

$$\mathtt{Coef}(F_{\delta_1,\delta_2}\ ,\ 0) = \frac{\mathtt{Coef}(f^{\boldsymbol{\sigma}}\ ,\ 0) + \delta_1 \cdot \boldsymbol{\sigma}(0) + \delta_2}{\mathtt{Coef}(f\ ,\ 0) + \delta_1 \cdot 0 + \delta_2}$$

and for $i \geq 1$,

$$\mathtt{Coef}(F_{\delta_1,\delta_2}\ ,\ i+1) = \mathtt{Coef}(F_{\delta_1,\delta_2}\ ,\ i) \cdot \frac{\mathtt{Coef}(f^{\boldsymbol{\sigma}}\ ,\ i) + \delta_1 \cdot \boldsymbol{\sigma}(i) + \delta_2}{\mathtt{Coef}(f\ ,\ i) + \delta_1 \cdot i + \delta_2}$$

In particular,

$$\mathtt{Coef}(F_{\delta_1,\delta_2},\ N-1) = \mathtt{Coef}(F^{\mathtt{RShift}}_{\delta_1,\delta_2}\ ,0) = \prod_{j=0}^{N-1} \frac{\mathtt{Coef}(f^{\boldsymbol{\sigma}}\ ,\ j) + \delta_1 \cdot \boldsymbol{\sigma}(j) + \delta_2}{\mathtt{Coef}(f\ ,\ j) + \delta_1 \cdot j + \delta_2} = 1.$$

5. $\mathcal{P}$ sends the $\mathbb{G}_1$-elements

$$\mathbf{a}^{\vee} := \mathbf{g}_1^{F_{\delta_1,\delta_2}(\mathbf{s})} \quad , \quad \mathbf{b}^{\vee} := \mathbf{g}_1^{F^{\mathtt{RShift}}_{\delta_1,\delta_2}(\mathbf{s})}.$$

and a proof of $\mathtt{PoRShift}[\mathbf{g}_1,\ (\mathbf{a}^{\vee},\ \mathbf{b}^{\vee})]$. (proof of cyclic right shift)

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

$$H(X)\ :=\ f_{\delta_1,\delta_2}(X)\ \odot\ F_{\delta_1,\delta_2}(X)\ =\ \widetilde{f}_{\delta_1,\delta_2}(X)\ \odot\ F^{\mathtt{RShift}}_{\delta_1,\delta_2}(X).$$

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

$$\mathbf{A}\ :=\ \mathbf{g}_1^{H(\mathbf{s})}.$$

7. $\mathcal{P}$ sends proofs of $\mathtt{PoHadProd}[\mathbf{g}_1,\ (\mathbf{a}_{\delta_1,\delta_2},\ \mathbf{a}^{\vee}),\ \mathbf{A}]$ and $\mathtt{PoHadProd}[\mathbf{g}_1,\ (\mathbf{b}_{\delta_1,\delta_2},\ \mathbf{b}^{\vee}),\ \mathbf{A}]$ (Hadamard product protocols)

8. $\mathcal{P}$ computes the polynomial $q_0(X)$ such that

$$F^{\mathtt{RShift}}_{\delta_1,\delta_2}(X)\ =\ q_0(X) \cdot X + 1$$

and sends the $\mathbb{G}_1$-element $\mathbf{Q}_0\ :=\ \mathbf{g}_1^{q_0(\mathbf{s})}$.

9. The Verifier $\mathcal{V}$ computes

$$\mathbf{a}_{\delta_1,\delta_2} = \mathbf{a} \cdot \mathbf{C}_{\mathrm{id}}^{\delta_1} \cdot \mathbf{C}_1^{\delta_2} \quad , \quad \mathbf{b}_{\delta_1,\delta_2} = \mathbf{a} \cdot \mathbf{C}_{\boldsymbol{\sigma}}^{\delta_1} \cdot \mathbf{C}_1^{\delta_2}.$$

10. $\mathcal{V}$ verifies the subprotocol proofs $\texttt{PoRShift}[\mathbf{g}_1,\ (\mathbf{a}^\vee,\ \mathbf{b}^\vee)]$, $\texttt{PoHadProd}[\mathbf{g}_1,\ (\mathbf{a}_{\delta_1,\delta_2},\ \mathbf{a}^\vee),\ \ \mathbf{A}]$, $\texttt{PoHadProd}[\mathbf{g}_1,\ (\mathbf{b}_{\delta_1,\delta_2},\ \mathbf{b}^\vee),\ \ \mathbf{A}]$ and the equation

$$\mathbf{Q}_0^{\mathbf{s}} \stackrel{?}{=} \mathbf{b}^\vee \cdot \mathbf{g}_1^{-1}$$

via the pairing check

$$\mathbf{e}(\mathbf{Q}_0,\ ,\ \mathbf{g}_1^{\mathbf{s}}) \stackrel{?}{=} \mathbf{e}(\mathbf{b}^\vee \cdot \mathbf{g}_1^{-1}\ ,\ \mathbf{g}_2).$$

□

5 Lookups

Let $T(X)$ be a polynomial such that the Verifier has access to the commitment $\mathbf{T} := \mathbf{g}_1^{T(\mathbf{s})}$. We discuss a protocol that allows a Prover to succinctly demonstrate show that for a committed polynomial $f(X)$ and an index set $\mathcal{I}$, we have the set containment:

$$\{\texttt{Coef}(f\ ,\ i)\ :\ i \in \mathcal{I}\} \subseteq \{\texttt{Coef}(T\ ,\ j)\ :\ j \in [0,\ \deg(T)]\}$$

Note that it suffices to show that

$$\sum_{i \in \mathcal{I}} [X + \texttt{Coef}(f\ ,\ i)]^{-1}$$

can be expressed as a linear combination

$$\sum_{j \in \mathcal{J}} m_j \cdot [X + \texttt{Coef}(T\ ,\ j)]^{-1}$$

for some subset $\mathcal{J} \subseteq [0,\ \deg(T)]$ and a sequence $m_j \in \mathbb{F}_p$ $(j \in \mathcal{J})$. This is because setting

$$f_{\Pi,\mathcal{I}}(X) := \prod_{i \in \mathcal{I}} X + \texttt{Coef}(f\ ,i)\ \ ,\ \ T_{\Pi,\mathcal{J}} := \prod_{j \in \mathcal{J}} X + \texttt{Coef}(T\ ,j)$$

yields

$$\frac{f'_{\Pi,\mathcal{I}}(X)}{f_{\Pi,\mathcal{I}}(X)} = \sum_{i \in \mathcal{I}} [X + \texttt{Coef}(f\ ,\ i)]^{-1}\ \ ,\ \ \frac{T'_{\Pi,\mathcal{J}}(X)}{T_{\Pi,\mathcal{J}}(X)} = \sum_{j \in \mathcal{J}} [X + \texttt{Coef}(T\ ,\ j)]^{-1},$$

where $h'(X)$ denotes the derivative of a polynomial $h(X)$. Thus, this would imply that the coefficient multiset of $f_{\mathcal{I}}(X)$ has at its underlying set the coefficient set $\{\texttt{Coef}(T\ ,\ j)\ :\ j \in [0,\ \deg(T)]\}$ with each $\texttt{Coef}(T\ ,\ j)$ occurring with multiplicity precisely m_j.

Note that the polynomial $T(X)$ is part of the preprocessed circuit and is known to have no repeated coefficients. Were that not the case, the Prover would need to verifiably send a commitment to the polynomial $T_{\Pi,\mathcal{J}} := \prod_{j \in \mathcal{J}} X + \texttt{Coef}(T\ ,j)$ and show that it is co-prime to its derivative. However, this is unnecessary for the lookup protocol since this claim is proven and verified during the circuit generation.

To this end, the Prover $\mathcal{P}$ constructs the subset $\mathcal{J} \subseteq [0,\ \deg(T)]$ such that the set $\{\texttt{Coef}(T\ ,\ j)\ :\ j \in \mathcal{J}\}$ is the underlying set of the coefficient multiset of $f_{\mathcal{I}}(X)$. $\mathcal{P}$ sends a commitment to the characteristic polynomial $\chi_{\mathcal{J}}(X)$ and shows that this is a commitment to the characteristic polynomial of some index set. In other words, he shows that this is a polynomial with all of its coefficients in $\{0, 1\}$. He then verifiably sends a commitment to the polynomial

$$T_{\mathcal{J}}(X) := \sum_{j \in \mathcal{J}} \texttt{Coef}(T\ ,\ j) \cdot X^j,$$

which is the Hadamard product of $T(X)$ and $\chi_{\mathcal{J}}(X)$. The Prover computes the multiplicities m_j with which the $\texttt{Coef}(T\ ,\ j)$ occur in $f_{\mathcal{I}}(X)$. He sends a commitment to the polynomial $\texttt{Mul}_{\mathcal{J}}(X) := \sum_{j\in\mathcal{J}} m_j \cdot X^j$.

Now, for a randomly generated challenge α, the Prover needs to show that

$$\sum_{i\in\mathcal{I}}[\alpha + \texttt{Coef}(f\ ,\ i)]^{-1} \ = \ \sum_{j\in\mathcal{J}} m_j \cdot [\alpha + \texttt{Coef}(T\ ,\ j)]^{-1},$$

whence it will follow that $\{\texttt{Coef}(T\ ,\ j)\ :\ j\in\mathcal{J}\}$ is the underlying set of $\{\texttt{Coef}(f\ ,\ j)\ :\ j\in\mathcal{I}\}$ with each $\texttt{Coef}(T\ ,\ j)$ occurring with multiplicity $m_j = \texttt{Coef}(M\ ,\ j)$.

In response to a challenge α generated by the Fiat-Shamir heuristic, the Prover verifiably sends commitments to the polynomials

$$f_{\mathcal{I},\alpha,\text{inv}}(X) := \sum_{i\in\mathcal{I}}[\texttt{Coef}(f\ ,\ i)\ +\ \alpha]^{-1}\cdot X^i \ \ ,\ \ T_{\mathcal{J},\alpha,\text{inv}}(X) := \sum_{j\in\mathcal{J}}[\texttt{Coef}(T\ ,\ j)\ +\ \alpha]^{-1}\cdot X^j.$$

It is straightforward to do so, since they are the unique polynomials that satisfy the equations

$$f_{\mathcal{I},\alpha,\text{inv}}(X)\ \odot\ [f(X) + \alpha\cdot\sum_{i=0}^{N-1} X^i] = \chi_{\mathcal{I}}(X)\ \ ,\ \ T_{\mathcal{J},\alpha,\text{inv}}(X)\ \odot\ [T(X) + \alpha\cdot\sum_{i=0}^{N-1} X^i] = \chi_{\mathcal{J}}(X)$$

Lastly, the Prover shows that the dot product of $\texttt{Mul}_{\mathcal{J}}(X)$ and $T_{\mathcal{J},\alpha,\text{inv}}(X)$ coincides with the evaluation of $f_{\mathcal{I},\alpha,\text{inv}}(X)$ at $X = 1$.

Protocol 5.1. *Proof of coefficient subset containment*

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 such that $\mathcal{V}$ stores the element $\mathbf{g}_1^{\mathbf{s}^N}$

Common preprocessed input: Index set $\mathcal{I}$; polynomial $T(X)$ that is known to have no repeated coefficients

Verifier's preprocessed input: The commitments

$$\mathbf{C}_{\mathcal{I}} := \mathbf{g}_1^{\chi_{\mathcal{I}}(\mathbf{s})}\ \ ,\ \ \mathbf{T} := \mathbf{g}_1^{T(\mathbf{s})}\ \ ,\ \ \mathbf{C}_{\text{id}} := \mathbf{g}_1^{\sum_{i=0}^{N-1}\mathbf{s}^k}.$$

Inputs: An element $\mathbf{a}\ \in\ \mathbb{G}_1$

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

$$\mathbf{a} = \mathbf{g}_1^{f(\mathbf{s})}\ \ ,\ \ \{\texttt{Coef}(f\ ,\ i)\ :\ i\in\mathcal{I}\}\ \subseteq\ \{\texttt{Coef}(T\ ,\ k)\ :\ k\in[0,\ \deg(T)]\}.$$

1. The Prover computes $f_{\mathcal{I}}(X) := f(X)\odot\chi_{\mathcal{I}}(X)$ and sends the $\mathbb{G}_1$-element

$$\mathbf{a}_{\mathcal{I}} := \mathbf{g}_1^{f_{\mathcal{I}}(\mathbf{s})}$$

2. $\mathcal{P}$ chooses an index set $\mathcal{J}$ such that the sets

$$\{\texttt{Coef}(f\ ,\ i)\ :\ i\in\mathcal{I}\}\ \ ,\ \ \{\texttt{Coef}(T\ ,\ j)\ :\ j\in\mathcal{J}\}$$

coincide and computes

$$T_{\mathcal{J}}(X) := T(X) \odot \chi_{\mathcal{J}}(X).$$

3. $\mathcal{P}$ sends the $\mathbb{G}_1$-elements

$$\mathbf{C}_{\mathcal{J}} := \mathbf{g}_1^{\chi_{\mathcal{J}}(\mathbf{s})} \ , \ \mathbf{T}_{\mathcal{J}} := \mathbf{g_1}^{T_{\mathcal{J}}(\mathbf{s})}$$

4. $\mathcal{P}$ computes the multiplicities m_j $(j \in \mathcal{J})$ with which the elements $\mathtt{Coef}(T_{\mathcal{J}}\ ,\ j)$ occur in the multiset

$$\mathtt{Coef}(f\ ,\ i)\ :\ i \in \mathcal{I}$$

and computes the polynomial

$$\mathtt{Mul}_{\mathcal{J}}(X) := \sum_{j \in \mathcal{J}} m_j \cdot X^j.$$

5. $\mathcal{P}$ sends the $\mathbb{G}_1$-element $\mathbf{M}_{\mathcal{J}} := \mathbf{g}_1^{\mathtt{Mul}_{\mathcal{J}}(\mathbf{s})}$.

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

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

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

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

$$\mathbf{a}_{\mathcal{I},\alpha,\mathrm{inv}} := \mathbf{g}_1^{f_{\mathcal{I},\alpha,\mathrm{inv}}(\mathbf{s})}$$

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

$$T_{\mathcal{J},\alpha,\mathrm{inv}}(X) := \sum_{j \in \mathcal{J}} [\mathtt{Coef}(T\ ,\ j)\ +\ \alpha]^{-1} \cdot X^j$$

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

$$\mathbf{T}_{\mathcal{J},\alpha,\mathrm{inv}} := \mathbf{g}_1^{T_{\mathcal{J},\alpha,\mathrm{inv}}(\mathbf{s})}$$

9. $\mathcal{P}$ sends the elements

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

10. $\mathcal{P}$ sends (batched) proofs of the following Hadamard and dot products:

- $\mathtt{PoHadProd}[\mathbf{g}_1,\ (\mathbf{a},\ \mathbf{C}_{\mathcal{I}}),\ \mathbf{a}_{\mathcal{I}}]$
- $\mathtt{PoHadProd}[\mathbf{g}_1,\ (\mathbf{a}_{\mathcal{I},\alpha,\mathrm{inv}},\ \mathbf{a} \cdot \mathbf{C}_{\mathrm{id}}^{\alpha}),\ \mathbf{C}_{\mathcal{I}}]$.
- $\mathtt{PoHadProd}[\mathbf{g}_1,\ (\mathbf{T},\ \mathbf{C}_{\mathcal{J}}),\ \mathbf{T}_{\mathcal{J}}]$
- $\mathtt{PoHadProd}[\mathbf{g}_1,\ (\mathbf{C}_{\mathcal{J}},\ \mathbf{C}_{\mathrm{id}} \cdot \mathbf{C}_{\mathcal{J}}^{-1}),\ \mathbf{g}_1]$.
- $\mathtt{PoHadProd}[\mathbf{g}_1,\ (\mathbf{T}_{\mathcal{J},\alpha,\mathrm{inv}},\ \mathbf{T} \cdot \mathbf{C}_{\mathrm{id}}^{\alpha}),\ \mathbf{C}_{\mathcal{J}}]$
- $\mathtt{PoDotProd}[\mathbf{g}_1,\ (\mathbf{M}_{\mathcal{J}},\ \mathbf{T}_{\mathcal{J},\alpha,\mathrm{inv}}),\ \beta_{\mathcal{I},\alpha,\mathrm{inv}}]$

11. $\mathcal{V}$ verifies the Hadamard and dot product subprotocols.

12. V verifies the equation

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

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

We note that prior to us becoming aware of the work of [Hab22], our protocol required the Prover to show that $\frac{f_{\prod}(X)}{\gcd(f_{\prod}(X),\, f'_{\prod}(X))}$ divides $\frac{T_{\prod}(X)}{\gcd(T_{\prod}(X),\, T'_{\prod}(X))}$. This entailed subprotocols for the derivatives in addition to expensive polynomial divisions, which made the protocol inefficient in practice. While the protocol from [Hab22] is meant for the Lagrange basis with roots of unity, we noticed that the underlying trick could be combined with the Hadamard product protocol to yield the analogous protocol in the monomial basis setting.

5.1 Subsequences

The lookup argument can also be extended to succinctly show that the coefficients of a polynomial at an index set form a subsequence in a table. As before, let $f(X), T(X)$ be committed polynomials of degree $\leq N-1$ and let $\mathcal{I}, \mathcal{J}$ be index sets to which the Verifier stores commitments. A Prover can show that the subsequences

$$\big[\texttt{Coef}(f\ ,\ i)\big]_{i\in\mathcal{I}}\ ,\ \ \big[\texttt{Coef}(T\ ,\ j)\big]_{j\in\mathcal{J}}$$

coincide. We note that this property can be rephrased as follows:

The Prover can construct a permutation $\boldsymbol{\sigma} :\ [0,\ N-1] \longrightarrow [0,\ N-1]$ such that:

1. $[f(X) \odot \chi_{\mathcal{J}}(X)]^{\boldsymbol{\sigma}} = T(X) \odot \chi_{\mathcal{I}}(X)$, i.e.

$$\texttt{Coef}\big(T \odot \chi_{\mathcal{J}}\ ,\ k\big)\ =\ \texttt{Coef}\big(f \odot \chi_{\mathcal{I}}\ ,\ \boldsymbol{\sigma}(k)\big)\ \ \forall\ \ k\ \in\ [0,\ N-1].$$

2. The permutation $\boldsymbol{\sigma}$ is increasing on the set $\mathcal{I}$ in the sense that:

$$\forall\ i_1, i_2 \in \mathcal{I},\quad i_1 < i_2\ \iff\ \boldsymbol{\sigma}(i_2) - \boldsymbol{\sigma}(i_1)\ \in\ [1,\ N-1]\ \subseteq\ \mathbb{F}_p.$$

The Prover constructs this permutation and sends a commitment to the polynomial

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

He uses the preceding protocol (i.e. the protocol for coefficient set containment) to show that the set of the coefficients of this committed polynomial is the interval $[0,\ N-1]$, which implies that it is indeed a commitment to some permutation of $[0,\ N-1]$. The Prover uses the protocol for the permutation and the Hadamard product protocol to show that this committed permutation $\boldsymbol{\sigma}$ satisfies the equation

$$[f(X) \odot \chi_{\mathcal{J}}(X)]^{\boldsymbol{\sigma}} = T(X) \odot \chi_{\mathcal{J}}(X).$$

It then remains to show that this permutation $\boldsymbol{\sigma}$ is increasing on the set $\mathcal{I}$, or equivalently, that for any $i_1,\ i_2\ \in\ \mathcal{I}$ with $i_1 < i_2$,

$$\texttt{Coef}(S_{\boldsymbol{\sigma}}(X)\ ,\ i_2)\ -\ \texttt{Coef}(S_{\boldsymbol{\sigma}}(X)\ ,\ i_1)\ \in\ [1,\ N-1]\ \subseteq\ \mathbb{F}_p.$$

To that end, the Prover constructs and sends the commitment to the polynomial $\widetilde{S}_{\boldsymbol{\sigma}}(X)$ given by

$$\texttt{Coef}(\widetilde{S}_{\boldsymbol{\sigma}}(X)\ ,\ k)\ :=\ \boldsymbol{\sigma}(m_k)\quad\text{where}\quad m_k := \max(I \cap [0,\ k]).$$

Note that this polynomial is created by replacing the 0's in the coefficients of $S_{\boldsymbol{\sigma}}(X)$ so that there are "jumps" precisely at the indices $\mathcal{I}$. The coefficients of this polynomial are in non-decreasing order if and only if $\boldsymbol{\sigma}$ is increasing on $\mathcal{I}$. The Prover uses the Hadamard product protocol and the protocol for coefficient set containment to show that this committed polynomial fulfills the following properties:

1. $\widetilde{S}_{\boldsymbol{\sigma}}(X) \odot \chi_{\mathcal{I}}(X) = S_{\boldsymbol{\sigma}}(X) \odot \chi_{\mathcal{I}}(X)$
2. The polynomial

$$\big[(1-X)\cdot\widetilde{S}_{\boldsymbol{\sigma}}(X)\big] \odot \frac{X^N-1}{X-1}\ =\ \sum_{k=0}^{N-1}\big[\texttt{Coef}(\widetilde{S}_{\boldsymbol{\sigma}}\ ,\ k+1) - \texttt{Coef}(\widetilde{S}_{\boldsymbol{\sigma}}\ ,\ k)\big]\cdot X^i$$

has all of its coefficients in the interval $[0,\ N-1]$.

This shows that sequence of coefficients of $\widetilde{S}_{\boldsymbol{\sigma}}(X)$ is *non-decreasing*. Since the coefficients of $S_{\boldsymbol{\sigma}}(X) \odot \chi_{\mathcal{I}}(X)$ form a subsequence of this sequence, it follows that the coefficients of $S_{\boldsymbol{\sigma}}(X) \odot \chi_{\mathcal{I}}(X)$ appear in increasing order, i.e. $\boldsymbol{\sigma}$ is increasing on the index set $\mathcal{I}$.

6 Univariate custom gates of high degree

We describe a protocol that allows for univariate custom gates of high degree. For committed polynomials $f(X),\ \widetilde{f}(X)$, a public univariate polynomial $P(X)$ and an index set $\mathcal{I}$, we describe a protocol that allows a Prover to succinctly demonstrate that the coefficients of $f(X)$, $\widetilde{f}(X)$ are linked as follows:

$$P\big(\texttt{Coef}(f\ ,\ i)\big) = \texttt{Coef}(\widetilde{f}\ ,\ i)\ \ ;\forall\ i\ \in\mathcal{I},\ .$$

In other words, there exist $c_i \in \mathbb{F}_p$ $(i \in \mathcal{I})$ such that

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

In addition to requiring that $P(X)$ is univariate, we also need the product $\deg(P)\cdot\big|\mathcal{I}\big|$ to be O(Circuit size). This is so that the Prover can compute the polynomial $h_\lambda(X)$ defined below, without incurring a high computational cost. The basic idea (lemma 2.1) is that for rational functions $f_i(X) \in \mathbb{F}_p(X)\ \ (i \in \mathcal{I})$, if the randomized sum

$$f_\lambda(X)\ :=\ \sum_{i\in\mathcal{I}} \lambda^i\cdot f_i(X)$$

is a polynomial for some randomly generated λ, then with overwhelming probability, the rational functions $f_i(X)$ are all polynomials. Thus, to show that the relation

$$P(\texttt{Coef}(f\ ,\ i)) = \texttt{Coef}(\widetilde{f}\ ,\ i)\ \forall\ i\ \in\ \mathcal{I}$$

holds, it suffices to verifiably send a KZG10 commitment to the polynomial

$$h_\lambda(X)\ :=\ \sum_{i\in\mathcal{I}} \frac{P(X) - \texttt{Coef}(\widetilde{f}\ ,\ i)}{X - \texttt{Coef}(f\ ,\ i)}\cdot\lambda^i$$

for some randomly generated challenge λ. The Prover first sends a commitment to this polynomial.

In response to a challenge α generated after the commitment has been sent, the Prover sends the commitment to the polynomial

$$h_\alpha^\vee(X) := \sum_{i \in \mathcal{I}} \frac{P(\alpha) - \texttt{Coef}(\widetilde{f}\ ,\ i)}{\alpha - \texttt{Coef}(f\ ,\ i)} \cdot X^i.$$

The Hadamard product protocol allows the Prover to succinctly show that this $\mathbb{G}_1$ element is indeed a KZG10 commitment to this polynomial. This is the unique polynomial that fulfills both of the Hadamard product equations

$$h_\alpha^\vee(X)\ \odot [\alpha \cdot \chi_{\mathcal{I}}(X) - f(X)]\ \stackrel{?}{=}\ P(\alpha) \cdot \chi_{\mathcal{I}}(X) - [\widetilde{f}(X) \odot \chi_{\mathcal{I}}(X)]$$

$$h_\alpha^\vee(X)\ \odot\ \chi_{\mathcal{I}}(X)\ \stackrel{?}{=}\ h_\alpha^\vee(X).$$

Now, the Prover shows that the committed polynomials $h_\alpha^\vee(X)$ and $h_\lambda(X)$ satisfy the equations

$$h_\alpha^\vee(\lambda) = h_\lambda(\alpha).$$

This implies that with overwhelming probability,

$$h_\lambda(\alpha) = \sum_{i \in \mathcal{I}} \frac{P(\alpha) - \texttt{Coef}(\widetilde{f}\ ,\ i)}{\alpha - \texttt{Coef}(f\ ,\ i)} \cdot \lambda^i.$$

Since α was randomly and uniformly generated after $\mathbf{b}_\lambda$ was sent, it follows that with overwhelming probability,

$$h_\lambda(X) = \sum_{i \in \mathcal{I}} \frac{P(X) - \texttt{Coef}(\widetilde{f}\ ,\ i)}{X - \texttt{Coef}(f\ ,\ i)} \cdot \lambda^i.$$

Since λ was randomly and uniformly generated, lemma 2.1 implies that overwhelming probability, the rational functions

$$\frac{P(X) - \texttt{Coef}(\widetilde{f}\ ,\ i)}{X - \texttt{Coef}(f\ ,\ i)}$$

are all polynomials, whence it follows that with overwhelming probability,

$$P(\texttt{Coef}(f\ , i)) = \texttt{Coef}(\widetilde{f}\ ,\ i)\ \forall\ i\ \in\ \mathcal{I}.$$

Protocol 6.1. *Proof of coefficient relation* (`CoefRel`)

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 such that $\mathcal{V}$ stores the element $\mathbf{g}_1^{\mathbf{s}^N}$

A commitment $\mathbf{C}_{\mathcal{I}} := \mathbf{g}_1^{\chi_{\mathcal{I}}(\mathbf{s})}$ to an index set $\mathcal{I}$. A public polynomial $P(X)$ and the commitment

$$\mathbf{C}_P := \mathbf{g}_1^{P(\mathbf{s})}.$$

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

Claim: The Prover knows polynomials $f(X)$, $\widetilde{f}(X)$ such that

$$\mathbf{a} = \mathbf{g}_1^{f(\mathbf{s})}\ \ ,\ \ \widetilde{\mathbf{a}} = \mathbf{g}_1^{\widetilde{f}(\mathbf{s})}\ \ ,\ \ P(\texttt{Coef}(f\ , i)) = \texttt{Coef}(\widetilde{f}\ , i)\ \forall\ i\ \in\ \mathcal{I}.$$

1. $\mathcal{P}$ computes
$$f_{\mathcal{I}}(X) := f(X) \odot \chi_{\mathcal{I}}(X) \ , \ \widetilde{f}_{\mathcal{I}}(X) := \widetilde{f}(X) \odot \chi_{\mathcal{I}}(X)$$
and sends the $\mathbb{G}_1$-elements
$$\mathbf{a}_{\mathcal{I}} := \mathbf{g}_1^{f_{\mathcal{I}}(\mathbf{s})} \ , \ \widetilde{\mathbf{a}}_{\mathcal{I}} := \mathbf{g}_1^{\widetilde{f}_{\mathcal{I}}(\mathbf{s})}.$$

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

3. $\mathcal{P}$ computes the polynomial
$$h_\lambda(X) \ := \ \sum_{i \in \mathcal{I}} \frac{P(X) - \mathtt{Coef}(\widetilde{f} \ , \ i)}{X - \mathtt{Coef}(f \ , \ i)} \cdot \lambda^i$$
and sends the $\mathbb{G}_1$ element
$$\mathbf{b}_\lambda := \mathbf{g}_1^{h_\lambda(\mathbf{s})}$$

4. The hashing algorithm $\mathtt{Hash}_{\text{FS}}$ generates a challenge $\alpha \in \mathbb{F}_p$.

5. $\mathcal{P}$ sends the $\mathbb{F}_p$-elements $\beta_\lambda \ := \ h_\lambda(\alpha) \ , \ \beta_P := P(\alpha)$.

6. $\mathcal{P}$ computes
$$h_\alpha^\vee(X) \ := \ \sum_{i \in \mathcal{I}} \frac{\beta_P - \mathtt{Coef}(\widetilde{f} \ , \ i)}{\alpha - \mathtt{Coef}(f \ , \ i)} \cdot X^i$$
and sends
$$\mathbf{b}^\vee \ := \ \mathbf{g}_1^{h_\alpha^\vee(\mathbf{s})}.$$

7. $\mathcal{P}$ sends the $\mathbb{G}_1$-elements
$$\mathbf{Q}_1 \ := \ \mathbf{g}_1^{\frac{h_\lambda(\mathbf{s}) - \beta_\lambda}{\mathbf{s} - \alpha}} \ , \ \mathbf{Q}_2 \ := \ \mathbf{g}_1^{\frac{P(\mathbf{s}) - \beta_P}{\mathbf{s} - \alpha}} \ , \ \mathbf{Q}_3 \ := \ \mathbf{g}_1^{\frac{h_\alpha^\vee(X)(\mathbf{s}) - \beta_\lambda}{\mathbf{s} - \lambda}}$$

8. $\mathcal{P}$ sends (batched) proofs of the following Hadamard product protocols:

- $\mathtt{PoHadProd}[\mathbf{g}_1, \ (\mathbf{a}, \ \mathbf{C}_{\mathcal{I}}), \ \mathbf{a}_{\mathcal{I}}]$

- $\mathtt{PoHadProd}[\mathbf{g}_1, \ (\widetilde{\mathbf{a}}, \ \mathbf{C}_{\mathcal{I}}), \ \widetilde{\mathbf{a}}_{\mathcal{I}}]$.

- $\mathtt{PoHadProd}[\mathbf{g}_1, \ (\mathbf{b}^\vee, \ \mathbf{C}_{\mathcal{I}}^\alpha \cdot \ \mathbf{a}_{\mathcal{I}}^{-1}), \ \mathbf{C}_{\mathcal{I}}^{\beta_P} \cdot \ \widetilde{\mathbf{a}}_{\mathcal{I}}^{-1}]$

- $\mathtt{PoHadProd}[\mathbf{g}_1, \ (\mathbf{b}^\vee, \ \mathbf{C}_{\mathcal{I}}), \ \mathbf{b}^\vee]$.

9. $\mathcal{V}$ verifies the the $\mathtt{PoHadProd}$s (the Hadamard product subprotocols) and the batchable equations
$$\mathbf{Q}_1^{\mathbf{s}-\alpha} \stackrel{?}{=} \mathbf{b}_\lambda \cdot \mathbf{g}_1^{-\beta_\lambda} \ , \ \mathbf{Q}_2^{\mathbf{s}-\alpha} \stackrel{?}{=} \mathbf{C}_P \cdot \mathbf{g}_1^{-\beta_P} \ , \ \mathbf{Q}_3^{\mathbf{s}-\lambda} \stackrel{?}{=} \mathbf{b}^\vee \cdot \mathbf{g}_1^{-\beta_\lambda}$$
via pairing checks. □

When the protocol (in the batched version rather than this modular version) is merged with the Snark, the Hadamard products and the polynomial openings do not contribute any extra $\mathbb{G}_1$ elements to the proof. But the proof does need commitments to the polynomials $f(X) \odot \chi_{\mathcal{I}}(X)$, $h_\lambda(X)$ and $h_\alpha^\vee(X)$.

Multiple univariate gates: The protocol naturally extends to multiple univariate polynomials $P_1(X), \ \cdots \ , \ P_k(X)$ and pairwise disjoint index sets $\mathcal{I}_1, \cdots, \mathcal{I}_k$ with the caveat that we need the sum

$$\sum_{j=1}^{k} \deg(P_j) \cdot |\mathcal{I}_j|.$$

to be small enough for the Prover to compute the polynomial

$$\sum_{j=1}^{k} \sum_{i \in \mathcal{I}_j} \frac{P_i(X) - \texttt{Coef}(\widetilde{f}\ ,\ i)}{X - \texttt{Coef}(f\ ,\ i)} \cdot \lambda^i$$

for a randomly and uniformly generated challenge $\lambda \in \mathbb{F}_p$. The Verifier needs to store commitments to the polynomials $P_j(X)$ and the characteristic polynomials $\chi_{\mathcal{I}_j}(X)$.

The proof is similar to the case $k = 1$ we described above. The Prover sends a commitment to

$$h_\lambda(X) \ := \ \sum_{j=1}^{k} \sum_{i \in \mathcal{I}_j} \frac{P_i(X) - \texttt{Coef}(\widetilde{f}\ ,\ i)}{X - \texttt{Coef}(f\ ,\ i)} \cdot \lambda^i.$$

In response to a challenge α, he sends a commitment to the polynomial

$$h^\vee_\alpha(X) \ := \ \sum_{j=1}^{k} \sum_{i \in \mathcal{I}_j} \frac{P_i(\alpha) - \texttt{Coef}(\widetilde{f}\ ,\ i)}{\alpha - \texttt{Coef}(f\ ,\ i)} \cdot \lambda^i$$

and shows that this committed polynomial satisfies the Hadamard products

$$h^\vee_\alpha(X) \ \odot \ \Big[\alpha \cdot \sum_{j=1}^{k} \chi_{\mathcal{I}_j}(X) \ - \ f(X)\Big] \ = \ \Big[\sum_{j=1}^{k} P_j(\alpha) \cdot \chi_{\mathcal{I}_j}(X)\Big] \ - \ [\widetilde{f}(X) \odot \sum_{j=1}^{k} \chi_{\mathcal{I}_j}(X)]$$

$$h^\vee_\alpha(X) \ \odot \ \Big[\sum_{j=1}^{k} \chi_{\mathcal{I}_j}(X)\Big].$$

Lastly, he shows that $h^\vee_\alpha(\lambda) = h_\lambda(\alpha)$

7 Acknowledgements

We thank Kapil Shenvi Pause for the Rust implementation of the bare-bones version of the scheme and for helpful feedback on previous drafts. We thank Mark Blunden, Anish Mohammed and Assimakis Kattis for helpful feedback on previous drafts. We thank Roman Melnikov for helpful conversations. We thank Remco Bloemen for communicating to us a protocol (https://xn–2-umb.com/22/ntt-argument/) linking monomial and Lagrange bases in settings where the scalar field has a large smooth order subgroup. We thank Sergio Juarez and Naman Kumar for help with the benchmarks and with the implementation.

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* (blogpost), https://xn–2-umb.com/22/ntt-argument/

[BGG17] S. Bowe, A. Gabizon, M. Green, *A multi-party protocol for constructing the public parameters of the Pinocchio zk-SNARK*

[CBBZ22] B. Chen, B. Bünz, D. Boneh, Z. Zhang, *HyperPLONK: PLONK with Linear-Time Prover and High-Degree Custom Gates*, https://eprint.iacr.org/2022/1355

[Ch10] J.H.Cheon, *Discrete Logarithm Problems with Auxiliary Inputs*

[CHMMVW] A. Chiesa,Y. Hu, M. Maller, P. Mishra, N. Vesely and N.P. Ward. Marlin: Preprocessing zk- SNARKs with universal and updatable SRS. EUROCRYPT2020, PartI, volume 12105 of LNCS

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

[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

[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.

[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] Ulrich 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.

[Lee21] J. Lee, *Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments.* In Theory of Cryptography Conference, pages 1–34. Springer, 2021.

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

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

[Sh20] V. Shoup *Arithmetic Software Libraries*, https://www.shoup.net/papers/akl-chapter.pdf

[Ngu05] L. Nguyen, *Accumulators from bilinear pairings and applications*, CT-RSA, 3376:275–292, 2005

[PST13] C. Papamanthou, E. Shi and R. Tamassia, *Signatures of correct computation*, in: Theory of Cryptography 2013, Lecture Notes in Comput. Sci. 7785, Springer, Heidelberg (2013), 222–242.

[PK22] Jim Posen and Assimakis A. Kattis. Caulk+: Table-independent lookup arguments. https://eprint.iacr.org/2022/957.

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

[ZBK+22] A. Zapico, V. Buterin, D. Khovratovich, M. Maller, A. Nitulescu, M. Simkin. *Caulk: Lookup arguments in sublinear time.* https://eprint.iacr.org/2022/621.

A Deferred proofs

A.1 The batched divisibility protocol

Proposition A.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 $\mathbf{B}_{\widetilde{\lambda}}$, $\widetilde{\mathbf{Q}}$ and the $\mathbb{F}_p$-elements $\beta_1, \cdots, \beta_k$.

Since the challenge $\widetilde{\alpha}$ was randomly and uniformly generated after the elements $\mathbf{B}_{\widetilde{\lambda}}$ and $\beta_1, \cdots, \beta_k$ were sent, the equation

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

implies that with overwhelming probability, an extractor $\mathcal{E}_{\mathtt{PPT}}$ can simulate the extractor $\mathcal{E}_{\mathtt{multi-PC}}$ to extract a polynomial $h^*(X)$ such that

$$\mathbf{B}_{\widetilde{\lambda}} \cdot [\prod_{i=1}^{k} (\mathbf{a}_i)^{\widetilde{\lambda}_1^i}] = \mathbf{g}_1^{h^*(\mathbf{s})} \ , \ h^*(\alpha) = \widetilde{\beta}.$$

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

$$\widetilde{\beta} := [\sum_{j=1}^{k} \widetilde{\lambda}^{j-1} \cdot \beta_j \cdot e_j(\widetilde{\alpha})^{-1}] + \sum_{i=1}^{k} \widetilde{\lambda}_1^i \cdot \beta_i.$$

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

$$h^*(\alpha) = [\sum_{j=1}^{k} \widetilde{\lambda}^{j-1} \cdot \beta_j \cdot e_j(\widetilde{\alpha})^{-1}] + \sum_{i=1}^{k} \widetilde{\lambda}_1^i \cdot \beta_i.$$

Since the challenge $\widetilde{\lambda}_1$ was randomly and uniformly generated after the $\mathbf{G}_1$ element $\mathbf{B}_{\widetilde{\lambda}}$ and the $\mathbb{F}_p$-elements $\beta_1, \cdots, \beta_k$ were sent, it follows that with overwhelming probability, $\mathcal{E}_{\mathtt{PPT}}$ can simulate the extractor $\mathcal{E}_{\mathtt{multi-PC}}$ to extract polynomials $f_\lambda^*(X)$ and $f_1^*(X), \cdots, f_k^*(X)$ such that

$$\mathbf{B}_{\widetilde{\lambda}} = \mathbf{g}_1^{f_\lambda^*(\mathbf{s})} \ , \ f_\lambda^*(\widetilde{\alpha}) = \widetilde{\beta} \ , \ \mathbf{a}_i = \mathbf{g}_1^{f_i^*(\mathbf{s})} \ , \ f_i^*(\widetilde{\alpha}) = \beta_i \ \ (i = 1, \cdots, k).$$

Thus, with overwhelming probability, the extracted polynomials $f_\lambda^*(X)$ and $f_i^*(X)$ $(i = 1, \cdots, 7)$ satisfy the equation

$$f_\lambda^*(\widetilde{\alpha}) = [\sum_{j=1}^{k} \widetilde{\lambda}^{j-1} \cdot \beta_j \cdot e_j(\widetilde{\alpha})^{-1}] + \sum_{i=1}^{k} \widetilde{\lambda}_1^i \cdot \beta_i = [\sum_{j=1}^{k} \widetilde{\lambda}^{j-1} \cdot \beta_j \cdot e_j(\widetilde{\alpha})^{-1}] + \sum_{i=1}^{k} \widetilde{\lambda}_1^i \cdot f_i^*(\widetilde{\alpha}).$$

Since the challenge $\widetilde{\alpha}$ was 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 $f_\lambda^*(X)$, $f_i^*(X)$ $(i = 1, \cdots, k)$ satisfy the equation

$$f_\lambda^*(X) = [\sum_{j=1}^{k} \widetilde{\lambda}^{j-1} \cdot f_i^*(X) \cdot e_j(X)^{-1}] + \sum_{i=1}^{k} \widetilde{\lambda}_1^i \cdot f_i^*(X).$$

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

$$\sum_{j=1}^{k} \widetilde{\lambda}^{j-1} \cdot f_j^*(X) \cdot e_j(X)^{-1} \in \mathbb{F}_p(X)$$

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

A.2 Security proof of the SNARK

Theorem A.2. *The protocol 3.1 is secure in the algebraic group model.*

Proof. (Sketch) The completeness is straightforward. It suffices to prove soundness.

Suppose a PPT algorithm $\mathcal{A}_{\mathtt{PPT}}$ outputs an accepting transcript consisting of the 10 $\mathbb{G}_1$-elements

$$\mathbf{A}_L,\ \mathbf{A}_R,\ \mathbf{A}_O,\ \mathbf{A}_{L,R},\ \mathbf{A}_{\widehat{F},1},\ \widetilde{\mathbf{A}}_{\gamma,\lambda,-},\ \widetilde{\mathbf{A}}_{\gamma,\lambda,+},\ \mathbf{A}_{\widehat{\lambda}},\ \mathbf{B}_{\widetilde{\lambda}},\ \widetilde{\mathbf{Q}}$$

and 20 $\mathbb{F}_p$-elements

$$\gamma_{_{L,R}},\ \beta_L,\ \beta_R,\ \beta_O,\ \beta_{L,R},\ \widetilde{\beta}_{\gamma,\lambda,-},\ \beta_{\mathrm{id}},\ \beta_{\boldsymbol{\sigma}},\ \widetilde{\beta}_{\gamma,\lambda,+},\ \beta_{\mathtt{pub}},\ \beta_{\widehat{F}},\ \nu_{\gamma,\widehat{F}},\ \nu_{\gamma,\widehat{R}},\ \beta_1,\cdots,\beta_7.$$

As in the protocol, the challenges

$$\delta_1,\ \delta_2,\ \gamma,\ \lambda,\ \ \widehat{\lambda},\ \alpha,\ \lambda_1,\ \widetilde{\lambda},\ \widetilde{\alpha},\ \widetilde{\lambda}_1$$

are computed by hashing the transcript at each stage, where the intermediary transcripts $\mathcal{T}_i$ are given by:

1. $\mathcal{T}_0 := \mathbf{C}_{\mathcal{A}} \,||\, \mathbf{C}_{\mathcal{M}} \,||\, \mathbf{C}_{\mathrm{id}} \,||\mathbf{C}_{\boldsymbol{\sigma}} \,||\, \mathbf{C}_{\mathtt{pub}} \,||\, \mathbf{A}_{\mathtt{pub}} \,||\, \mathbf{A}_L \,||\, \mathbf{A}_R \,||\, \mathbf{A}_O \,||\, \mathbf{A}_{L,R}.$

$\delta_1,\ \delta_2 := \mathtt{Hash}_{\mathrm{FS}}(\mathcal{T}_0, 0),\ \mathtt{Hash}_{\mathrm{FS}}(\mathcal{T}_0, 1)$

2. $\mathcal{T}_1 := \mathcal{T}_0 \,||\, \mathbf{A}_{F,1}$

$\gamma := \mathtt{Hash}_{\mathrm{FS}}(\mathcal{T}_1)$

3. $\mathcal{T}_2 := \mathcal{T}_1 \,||\, \gamma_{_{L,R}}$

$\lambda := \mathtt{Hash}_{\mathrm{FS}}(\mathcal{T}_2)$

4. $\mathcal{T}_3 := \mathcal{T}_2 \,||\, \mathbf{A}_{\gamma,\lambda,-} \,||\, \mathbf{A}_{\gamma,\lambda,+}$

$\widehat{\lambda} := \mathtt{Hash}_{\mathrm{FS}}(\mathcal{T}_3)$

5. $\mathcal{T}_4 := \mathcal{T}_3 \,||\, \mathbf{A}_{\widehat{\lambda}}$

$\alpha := \mathtt{Hash}_{\mathrm{FS}}(\mathcal{T}_4)$

6. $\mathcal{T}_5 := \mathcal{T}_4 \,||\, \beta_L \,||\, \beta_R \,||\, \beta_O \,||\, \beta_{L,R} \,||\, \widetilde{\beta}_{\gamma,\lambda,-} \,||\, \beta_{\mathrm{id}} \,||\, \beta_{\boldsymbol{\sigma}} \,||\, \widetilde{\beta}_{\gamma,\lambda,+} \,||\, \beta_{\mathtt{pub}} \,||\, \beta_{\widehat{F}} \,||\, \nu_{\gamma,\widehat{F}} \,||\, \nu_{\gamma,\widehat{R}}$

$\widetilde{\lambda} := \mathtt{Hash}_{\mathrm{FS}}(\mathcal{T}_5)$

7. $\mathcal{T}_6 := \mathcal{T}_5 \,||\, \mathbf{B}_{\widetilde{\lambda}}$

$\widetilde{\alpha} := \mathtt{Hash}_{\mathrm{FS}}(\mathcal{T}_6)$

8. $\mathcal{T}_7 := \mathcal{T}_6 \,||\, \beta_1 \,||\, \beta_2 \,||\, \beta_3 \,||\, \beta_4 \,||\, \beta_5 \,||\, \beta_6 \,||\, \beta_7$

$\widetilde{\lambda}_1 := \mathtt{Hash}_{\mathrm{FS}}(\mathcal{T}_7)$

9. $\mathcal{T}_8 := \mathcal{T}_7 \,||\, \widetilde{\mathbf{Q}}$

As in the protocol, we denote:

1. $e_1(X) := X - \gamma\ ,\ \mathbf{a}_1 := \mathbf{A}_{L,R} \cdot \mathbf{g}_1^{-\gamma_{_{L,R}}}$
2. $e_2(X) := \gamma \cdot X - \alpha\ ,\ \mathbf{a}_2 := \big[\widetilde{\mathbf{A}}_{\gamma,\lambda,-} \cdot (\mathbf{g}_1^{\mathbf{s}^N})^{\gamma_{_{L,R}}} \cdot \widetilde{\mathbf{A}}_{\gamma,\lambda,+}^{(\gamma^{-1}\cdot\alpha)^{N+1}}\big] \cdot \mathbf{A}_{\lambda,\alpha}^{-1}$ where

$$\beta_{\mathtt{Wires}} := \beta_L + \alpha^n \cdot \beta_R + \alpha^{2n} \cdot \beta_O$$

and

$$\mathbf{A}_{\lambda,\alpha} := \big[\mathbf{g}_1^{\beta_L\cdot\nu_{\gamma,\widehat{R}}}\big] \cdot \big[(\widehat{\mathbf{C}}_{\mathcal{A}})^{\lambda\cdot[\beta_L+\beta_R-\beta_0]}\big] \cdot \big[(\widehat{\mathbf{C}}_{\mathcal{M}})^{\lambda^2\cdot[\beta_{L,R}-\beta_0]}\big].$$

$$\cdot\big[(\mathbf{g}_1)^{\lambda^3\cdot\nu_{\gamma,\widehat{F}}\cdot[\beta_{\mathtt{Wires}}+\delta_1\cdot\beta_{\mathrm{id}}+\delta_2\cdot\frac{\alpha^N-1}{\alpha-1}]}\big] \cdot \big[(\mathbf{A}_{\widehat{F},1})^{-\lambda^3\cdot[\beta_{\mathtt{Wires}}+\delta_1\cdot\beta_{\boldsymbol{\sigma}}+\delta_2\cdot\frac{\alpha^N-1}{\alpha-1}]}\big] \cdot \big[(\widehat{\mathbf{C}}_{\mathtt{pub}})^{\lambda^4\cdot[\beta_L-\beta_{\mathtt{pub}}]}\big]$$

3. $e_3(X) := X - \alpha$, $\mathbf{a}_3 := \mathbf{A}_{\lambda_1} \cdot \mathbf{g}_1^{-\beta_{\lambda_1}}$,

where $\mathbf{A}_{\lambda_1} := \mathbf{A}_L \cdot \mathbf{A}_R^{\lambda_1} \cdot \mathbf{A}_O^{\lambda_1^2} \cdot \widetilde{\mathbf{A}}_{\gamma,\lambda,-}^{\lambda_1^3} \cdot \mathbf{C}_{\mathtt{id}}^{\lambda_1^4} \cdot \mathbf{C}_{\boldsymbol{\sigma}}^{\lambda_1^5} \cdot \widetilde{\mathbf{A}}_{\gamma,\lambda,+}^{\lambda_1^6} \cdot \mathbf{A}_{\mathtt{pub}}^{\lambda_1^7} \cdot \mathbf{A}_{L,R}^{\lambda_1^8} \cdot \mathbf{A}_{\widehat{F},1}^{\lambda_1^9}$

$$\beta_{\lambda_1} := \beta_L + \lambda_1 \cdot \beta_R + \lambda_1^2 \cdot \beta_O + \lambda_1^3 \cdot \widetilde{\beta}_{\gamma,\lambda,-} + \lambda_1^4 \cdot \beta_{\mathrm{id}} + \lambda_1^5 \cdot \beta_{\boldsymbol{\sigma}} + \lambda_1^6 \cdot \widetilde{\beta}_{\gamma,\lambda,+} + \lambda_1^7 \cdot \beta_{\mathtt{pub}} + \lambda_1^8 \cdot \beta_{L,R} + \lambda_1^9 \cdot [\alpha^{-1} \cdot \beta_{\widehat{F}} + \alpha^N - 1].$$

4. $e_4(X) := X - \alpha^{-1}$, $\mathbf{a}_4 := \mathbf{A}_{\widehat{\lambda}} \cdot \mathbf{g}_1^{-\beta_{\widehat{\lambda}}}$
5. $e_5(X) := X$, $\mathbf{a}_5 := \mathbf{A}_{\widehat{F},1}$
6. $e_6(X) := \gamma \cdot X - \alpha$, $\mathbf{a}_6 := \mathbf{A}_{\widehat{F},1}^{[\gamma^{-1} \cdot \alpha]} \cdot \mathbf{g}_1^{[1-\mathbf{s}^N]-\beta_{\gamma,\widehat{F}}}$
7. $e_7(X) := \alpha \cdot X - \gamma$, $\mathbf{a}_7 := \mathbf{A}_R \cdot \mathbf{g}_1^{-(\gamma \cdot \alpha^{-1})^N \cdot \nu_{\gamma,\widehat{R}}}$

The Verifier $\mathcal{V}$ independently computes

$$\widetilde{\beta} := [\sum_{j=1}^{7} \widetilde{\lambda}^{j-1} \cdot \beta_j \cdot e_j(\widetilde{\alpha})^{-1}] + \sum_{i=1}^{7} \widetilde{\lambda}_1^i \cdot \beta_i.$$

The pairing check implies that with overwhelming probability,

$$\widetilde{\mathbf{Q}}^{\mathbf{s}-\widetilde{\alpha}} \stackrel{?}{=} \mathbf{B}_{\widetilde{\lambda}} \cdot [\prod_{i=1}^{7} (\mathbf{a}_i)^{\widetilde{\lambda}_1^i}] \cdot \mathbf{g}_1^{-\widetilde{\beta}} = \mathbf{B}_{\widetilde{\lambda}} \cdot \mathbf{g}_1^{-[\sum_{j=1}^{7} \widetilde{\lambda}^{j-1} \cdot \beta_j \cdot e_j(\widetilde{\alpha})^{-1}]} \cdot [\prod_{i=1}^{7} (\mathbf{a}_i \cdot \mathbf{g}_1^{-\beta_i})^{\widetilde{\lambda}_1^i}].$$

The challenge $\widetilde{\lambda}_1$ was randomly and uniformly generated after the $\mathbb{G}_1$-element $\mathbf{B}_{\widetilde{\lambda}}$ and the $\mathbb{F}_p$-elements $\beta_1, \cdots, \beta_7$ had been sent and the elements $\mathbf{a}_1, \cdots, \mathbf{a}_7$ were computable from the Prover's intermediate transcript. Hence, with overwhelming probability, an extractor $\mathcal{E}_{\mathtt{PPT}}$ can simulate the extractor $\mathcal{E}_{\mathtt{multi-PC}}$ to extract polynomials $h_{\widetilde{\lambda}}(X)$ and $h_i^*(X)$ $(i = 1, \cdots, 7)$ such that

$$\mathbf{a}_i := \mathbf{g}_1^{h_i^*(\mathbf{s})} \quad , \quad \mathbf{B}_{\widetilde{\lambda}} = \mathbf{g}_1^{h_{\widetilde{\lambda}}^*(\mathbf{s})} \quad , \quad h_i^*(\widetilde{\alpha}) = \beta_i$$

$$h_{\widetilde{\lambda}}^*(\widetilde{\alpha}) = \sum_{j=1}^{7} \widetilde{\lambda}^{j-1} \cdot \beta_j \cdot e_j(\widetilde{\alpha})^{-1} = \sum_{j=1}^{7} \widetilde{\lambda}^{j-1} \cdot h_i^*(\widetilde{\alpha}) \cdot e_j(\widetilde{\alpha})^{-1}.$$

Since the challenge $\widetilde{\alpha}$ was randomly and uniformly generated after the elements $\mathbf{B}_{\widetilde{\lambda}}$ was sent, the Schwartz-Zippel lemma implies that with overwhelming probability, the extracted polynomials $h_{\widetilde{\lambda}}^*(X)$, $h_j^*(X)$ $(j = 1, \cdots, 7)$ satisfy the equation

$$h_{\widetilde{\lambda}}^*(X) = \sum_{j=1}^{7} \widetilde{\lambda}^{j-1} \cdot h_j^*(X) \cdot e_j(X)^{-1}.$$

Thus, in particular, the rational function $\sum_{j=1}^{7} \widetilde{\lambda}^{j-1} \cdot h_j^*(X) \cdot e_j(X)^{-1}$ is a polynomial. The challenge $\widetilde{\lambda}$ was randomly and uniformly generated after the elements $\mathbf{a}_1, \cdots, \mathbf{a}_7 \in \mathbb{G}_1$ and the linear polynomials $e_1(X), \cdots, e_7(X)$ had been communicated to the Verifier. More precisely, $\widetilde{\lambda}$ was randomly and uniformly generated after the Prover's intermediate transcript had enough information for the Verifier to independently compute the the elements $\mathbf{a}_1, \cdots, \mathbf{a}_7 \in \mathbb{G}_1$ and the linear polynomials $e_1(X), \cdots, e_7(X)$. Hence, lemma 2.1 implies that with overwhelming

probability, each of the rational functions $h_j^*(X) \cdot e_j(X)^{-1}$ is a polynomial, i.e. $h_j^*(X)$ is divisible by $e_j(X)$ for $j = 1, \cdots, 7$.

Thus, it follows that with overwhelming probability, $\mathcal{E}_{\mathtt{PPT}}$ can simulate the extractor $\mathcal{E}_{\mathtt{multi-PC}}$ to extract polynomials

$$P_{L,R}^*(X),\ \widetilde{f}_{\gamma,\lambda,-}^*(X),\ \widetilde{f}_{\gamma,\lambda,+}^*(X),\ f_{\lambda_1}^*(X),\ \widehat{f}_{\widehat{\lambda}}^*(X),\ \widehat{F}_1^*(X),\ R^*(X)$$

such that

1. $\mathbf{A}_{L,R} = \mathbf{g}_1^{P_{L,R}^*(\mathbf{s})}$, $P_{L,R}^*(\gamma) = \gamma_{L,R}$
2. $\widetilde{\mathbf{A}}_{\gamma,\lambda,-} = \mathbf{g}_1^{\widetilde{f}_{\gamma,\lambda,-}^*(\mathbf{s})}$, $\widetilde{\mathbf{A}}_{\gamma,\lambda,+} = \mathbf{g}_1^{\widetilde{f}_{\gamma,\lambda,+}^*(\mathbf{s})}$
3. $\mathbf{A}_{\lambda_1} = \mathbf{g}_1^{f_{\lambda_1}^*(\mathbf{s})}$, $f_{\lambda_1}^*(\alpha) = \beta_{\lambda_1}$.
4. $\mathbf{A}_{\widehat{\lambda}} = \mathbf{g}_1^{\widehat{f}_{\widehat{\lambda}}^*(\mathbf{s})}$, $\widehat{f}_{\widehat{\lambda}}^*(\alpha^{-1}) = \beta_{\widehat{\lambda}}$
5. $\mathbf{A}_{\widehat{F},1} = \mathbf{g}_1^{\widehat{F}_1^*(\mathbf{s})}$, $\widehat{F}_1^*(0) = 0$
6. $(\gamma^{-1} \cdot \alpha) \cdot [\widehat{F}_1^*(\gamma^{-1} \cdot \alpha) + 1 - (\gamma^{-1} \cdot \alpha)^N] = \nu_{\gamma,\widehat{F}}$
7. $\mathbf{A}_R = \mathbf{g}_1^{R^*(\mathbf{s})}$, $R^*(\gamma \cdot \alpha^{-1}) = (\gamma \cdot \alpha^{-1})^N \cdot \nu_{\gamma,\widehat{R}}$.

We initially fixate on the implications of Statement 3 in this list. Since the challenge λ_1 was randomly and uniformly generated after the elements

$$\beta_L,\ \beta_R,\ \beta_O,\ \beta_{L,R},\ \widetilde{\beta}_{\gamma,\lambda,-},\ \beta_{\mathrm{id}},\ \beta_{\boldsymbol{\sigma}},\ \widetilde{\beta}_{\gamma,\lambda,+},\ \beta_{\mathtt{pub}},\ \beta_{\widehat{F}},\ \nu_{\gamma,\widehat{F}},\ \nu_{\gamma,\widehat{R}}$$

had been sent, it follows that with overwhelming probability, $\mathcal{E}_{\mathtt{PPT}}$ can simulate the extractor $\mathcal{E}_{\mathtt{multi-PC}}$ to extract polynomials

$$L^*(X),\ R^*(X),\ O^*(X),\ \widehat{F}_1^*(X),\ \widetilde{f}_{\gamma,\lambda,-}^*(X),\ \widetilde{f}_{\gamma,\lambda,+}^*(X)$$

such that:

1. $\mathbf{A}_L = \mathbf{g}_1^{L^*(\mathbf{s})}$, $\beta_L = L^*(\alpha)$
2. $\mathbf{A}_R = \mathbf{g}_1^{R^*(\mathbf{s})}$, $\beta_R = R^*(\alpha)$
3. $\mathbf{A}_O = \mathbf{g}_1^{O^*(\mathbf{s})}$, $\beta_O = O^*(\alpha)$
4. $\mathbf{A}_{L,R} = \mathbf{g}_1^{P_{L,R}^*(\mathbf{s})}$, $\beta_{L,R} = P_{L,R}^*(\alpha)$
5. $\widetilde{\mathbf{A}}_{\gamma,\lambda,-} = \mathbf{g}_1^{\widetilde{f}_{\gamma,\lambda,-}^*(\mathbf{s})}$, $\widetilde{\beta}_{\gamma,\lambda,-} = \widetilde{f}_{\gamma,\lambda,-}^*(\alpha)$
6. $\widetilde{\mathbf{A}}_{\gamma,\lambda,+} = \mathbf{g}_1^{\widetilde{f}_{\gamma,\lambda,+}^*(\mathbf{s})}$, $\widetilde{\beta}_{\gamma,\lambda,+} = \widetilde{f}_{\gamma,\lambda,+}^*(\alpha)$
7. $\beta_{\mathrm{id}} = P_{\mathrm{id}}(\alpha)$
8. $\beta_{\boldsymbol{\sigma}} = P_{\boldsymbol{\sigma}}(\alpha)$
9. $\beta_{\mathtt{pub}} = P_{\mathtt{pub}}(\alpha)$
10. $\mathbf{A}_{\widehat{F},1} := \mathbf{g}_1^{\widehat{F}_1^*(\mathbf{s})}$, $\widehat{F}_1^*(\alpha) := \alpha^{-1} \cdot [\beta_F + \alpha^N - 1]$.

We define

$$\overline{\widehat{F}}_1^*(X) := \widehat{F}_1^*(X) \pmod{X^N} \ , \ F_1^*(X) := X^N \cdot \overline{\widehat{F}}_1^*(X^{-1}) = \sum_{i=0}^{N} \mathtt{Coef}(\widehat{F}_1^* \ , \ i) \cdot X^{N-i}$$

$$\widehat{F}^*(X) := X \cdot [\widehat{F}_1^*(X) + 1 - X^N] \ , \ \overline{\widehat{F}}^*(X) := \widehat{F}^*(X) \pmod{X^N}$$

$$F^*(X) := X^N \cdot \overline{\widehat{F}}^*(X^{-1}) = \sum_{i=0}^{N} \mathtt{Coef}(\widehat{F}^* \ , \ i) \cdot X^{N-i}$$

By construction, the polynomials $F^*(X)$, $F_1^*(X)$ satisfy the equations

$$F_1^*(X) = X \cdot F^*(X) \pmod{X^N} \ , \ \mathtt{Coef}(F^* \ , \ N-1) = \mathtt{Coef}(F^* \ , \ N-1) = 1.$$

We define

$$\mathtt{Wires}^*(X) \ := \ L^*(X) + X^n \cdot R^*(X) + X^{2n} \cdot O^*(X).$$

$$\overline{R}^*(X) := R^*(X) \pmod{X^N} \ , \ \widehat{R}^*(X) \ := \ X^N \cdot \overline{R}^*(X^{-1}) = \sum_{i=0}^{N-1} \mathtt{Coef}(R \ , \ i) \cdot X^{N-i}.$$

$$\widetilde{f}^*_{\gamma,\lambda}(X) \ := \ \widetilde{f}^*_{\gamma,\lambda,-} + \gamma_{L,R} \cdot X^N + \widetilde{f}^*_{\gamma,\lambda,+} \cdot X^{N+1},$$

where $L^*(X)$, $R^*(X)$, $O^*(X)$, $\widetilde{f}^*_{\gamma,\lambda,-}(X)$, $\widetilde{f}^*_{\gamma,\lambda,+}(X)$ are the aformentioned polynomials extracted by $\mathcal{E}_{\mathtt{PPT}}$.

As in the protocol, set

$$\beta_{\mathtt{Wires}} \ := \ \beta_L + \alpha^n \cdot \beta_R + \alpha^{2n} \cdot \beta_O = L^*(\alpha) + \alpha^n \cdot R^*(\alpha) + \alpha^{2n} \cdot O^*(\alpha) = \mathtt{Wires}^*(\alpha)$$

and

$$\mathbf{A}_{\lambda,\alpha} \ := \ \mathbf{g}_1^{\beta_L \cdot \nu_{\gamma,\widehat{R}}} \cdot (\widehat{\mathbf{C}}_{\mathcal{A}})^{\lambda \cdot [\beta_L + \beta_R - \beta_0]} \cdot (\widehat{\mathbf{C}}_{\mathcal{M}})^{\lambda^2 \cdot [\beta_{L,R} - \beta_0]}.$$

$$\cdot (\mathbf{g}_1)^{\lambda^3 \cdot \nu_{\gamma,\widehat{F}} \cdot [\beta_{\mathtt{Wires}} + \delta_1 \cdot \beta_{\mathrm{id}} + \delta_2 \cdot \frac{\alpha^N - 1}{\alpha - 1}]} \cdot (\mathbf{A}_{\widehat{F},1})^{-\lambda^3 \cdot [\beta_{\mathtt{Wires}} + \delta_1 \cdot \beta_{\boldsymbol{\sigma}} + \delta_2 \cdot \frac{\alpha^N - 1}{\alpha - 1}]} \cdot (\widehat{\mathbf{C}}_{\mathtt{pub}})^{\lambda^4 \cdot [\beta_{\mathtt{Wires}} - \beta_{\mathtt{pub}}]}$$

Now, the extracted polynomial

$$f^*_{\gamma,\lambda,\alpha}(X) :=$$

$$L^*(\alpha) \cdot \widehat{R}^*(\gamma^{-1} \cdot \alpha) + \lambda \cdot [L^*(\alpha) + R^*(\alpha) - O^*(\alpha)] \cdot \widehat{\chi}_{\mathcal{I}_{\mathcal{A}}}(X) + \lambda^2 \cdot [P^*_{L,R}(\alpha) - O^*(\alpha)] \cdot \widehat{\chi}_{\mathcal{I}_{\mathcal{M}}}(X)$$

$$+\lambda^3 \cdot \widehat{F}^*(\gamma^{-1} \cdot \alpha) \cdot [\beta_{\mathtt{Wires}} + \delta_1 \cdot \beta_{\mathrm{id}} + \delta_2 \cdot \frac{\alpha^N - 1}{\alpha - 1}] - \lambda^3 \cdot [\beta_{\mathtt{Wires}} + \delta_1 \cdot \beta_{\boldsymbol{\sigma}} + \delta_2 \cdot \frac{\alpha^N - 1}{\alpha - 1}]$$

$$+\lambda^4 \cdot [\beta_{\mathtt{Wires}} - P_{\mathtt{pub}}(\alpha)] \cdot \widehat{\chi}_{\mathcal{I}_{\mathtt{pub}}}(X) \ ,$$

with overwhelming probability, satisfies the equation

$$\mathbf{A}_{\lambda,\alpha} \ = \ \mathbf{g}_1^{f^*_{\gamma,\lambda,\alpha}(\mathbf{s})}$$

and the congruence

$$\widetilde{f}^*_{\gamma,\lambda,-}(X) + \gamma_{L,R} \cdot X^N + \widetilde{f}^*_{\gamma,\lambda,+}(X) \cdot X^{N+1} \equiv f^*_{\gamma,\lambda,\alpha}(X) \pmod{\gamma \cdot X - \alpha}.$$

It follows that with overwhelming probability, the polynomial $\widetilde{f}^*_{\gamma,\lambda}(X)$ defined by

$$\widetilde{f}^*_{\gamma,\lambda}(X) \ := \ \widetilde{f}^*_{\gamma,\lambda,-}(X) + \gamma_{L,R} \cdot X^N + \widetilde{f}^*_{\gamma,\lambda,+}(X) \cdot X^{N+1}$$

is congruent to the polynomial

$$\mathfrak{f}^*_{\gamma,\lambda}(X) \;:=\; L^*(\gamma\cdot X)\cdot\widehat{R}^*(X)+\lambda\cdot[L^*(\gamma\cdot X)+R^*(\gamma\cdot X)-O^*(\gamma\cdot X)]\cdot\widehat{\chi}_{\mathcal{I}_{\mathcal{A}}}(X)$$

$$+\lambda^2\cdot[P^*_{L,R}(\gamma\cdot X)-O^*(\gamma\cdot X)]\cdot\widehat{\chi}_{\mathcal{I}_{\mathcal{M}}}(X)$$

$$+\lambda^3\cdot\big[\mathtt{Wires}^*(\gamma\cdot X)+\delta_1\cdot P_{\mathtt{id}}(\gamma\cdot X)+\delta_2\cdot\sum_{i=0}^{N-1}(\gamma\cdot X)^i\big]\cdot X\cdot[\widehat{F}^*_1(X)-X^N+1]\big]$$

$$-\lambda^3\cdot\big[\mathtt{Wires}^*(\gamma\cdot X)+\delta_1\cdot P_{\boldsymbol{\sigma}}(\gamma\cdot X)+\delta_2\cdot\sum_{i=0}^{N-1}(\gamma\cdot X)^i\big]\cdot\widehat{F}^*_1(X)$$

$$+\lambda^4\cdot[\mathtt{Wires}^*(\gamma\cdot X)-P_{\mathtt{pub}}(\gamma\cdot X)]\cdot\widehat{\chi}_{\mathcal{I}_{\mathtt{pub}}}(X)$$

modulo $(\gamma\cdot X-\alpha)$.

The challenge α was randomly and uniformly generated after the $\mathbb{G}_1$ elements $\widetilde{\mathbf{A}}_{\gamma,\lambda,-},\ \widetilde{\mathbf{A}}_{\gamma,\lambda,+}$ were sent and after the challenge γ was generated. Hence, this forces the equality

$$\widetilde{f}^*_{\gamma,\lambda}(X)\;=\;\mathfrak{f}^*_{\gamma,\lambda}(X)$$

with overwhelming probability. In particular, the coefficient $\mathtt{Coef}(\widetilde{f}^*_{\gamma,\lambda},\ N)$ of the extracted polynomial $\widetilde{f}^*_{\gamma,\lambda}(X)$ at the position X^N coincides with the sum

$$\mathtt{HP}_1(\gamma)+\lambda\cdot\mathtt{HP}_2(\gamma)+\lambda^2\cdot\mathtt{HP}_3(\gamma)+\lambda^3\cdot[\mathtt{HP}_4(\gamma)-\mathtt{HP}_5(\gamma)]+\lambda^4\cdot\mathtt{HP}_6(\gamma)$$

where the $\mathtt{HP}_i(X)$ are the following Hadamard products:

1. $\mathtt{HP}_1(X)\;:=\;L^*(X)\odot R^*(X)$.
2. $\mathtt{HP}_2(X)\;:=\;[L^*(X)+R^*(X)-O^*(X)]\odot\chi_{\mathcal{I}_{\mathcal{A}}}(X)$.
3. $\mathtt{HP}_3(X)\;:=\;[P^*_{L,R}(X)-O^*(X)]\odot\chi_{\mathcal{I}_{\mathcal{M}}}(X)$.
4. $\mathtt{HP}_4(X)\;:=\;[\mathtt{Wires}^*(X)+\delta_1\cdot P_{\mathrm{id}}(X)+\delta_2\cdot\frac{X^N-1}{X-1}]\odot F^*(X)$
5. $\mathtt{HP}_5(X)\;:=\;[\mathtt{Wires}^*(X)+\delta_1\cdot P_{\boldsymbol{\sigma}}(X)+\delta_2\cdot\frac{X^N-1}{X-1}]\odot F^*_1(X)$
6. $\mathtt{HP}_6(X)\;:=\;[\mathtt{Wires}^*(X)-P_{\mathtt{pub}}(X)]\odot\chi_{\mathcal{I}_{\mathtt{pub}}}(X)$.

and $L^*(X),\ R^*(X),\ O^*(X),\ P^*_{L,R}(X),\ \mathtt{Wires}^*(X)$ are the aforementioned polynomials extracted by $\mathcal{E}_{\mathtt{PPT}}$. We now argue that with overwhelming probability, the following claims are valid regarding these extracted polynomials.

Claim 1: $\mathtt{Coef}(\widetilde{f}^*_{\gamma,\lambda},\ N)\;=\;\gamma_{L,R}\;=\;P^*_{L,R}(\gamma)$.

Claim 2: The polynomials $L^*(X)$, $R^*(X)$, $O^*(X)$ are of degrees $\leq\ n-1$.

We first explain why the validity of these two claims would imply soundness. The challenge λ was randomly and uniformly generated after the $\mathbb{G}_1$ elements

$$\mathbf{A}_L\ ,\ \mathbf{A}_R\ ,\ \mathbf{A}_0\ ,\ \mathbf{A}_{L,R}\ ,\ \mathbf{A}_{\widehat{F},1}\ ,\ \gamma_{L,R}$$

had been sent. Hence, if Claims 1 and Claims 2 are both valid, the Schwartz-Zippel lemma will then imply that with overwhelming probability, the extracted polynomials $L^*(X)$, $R^*(X)$, $P^*_{L,R}(X)$, $O^*(X)$, $F^*_1(X)$ are such that the following evaluations of Hadamard products are valid:

- $L^*(X) \odot R^*(X)$ is valued $P^*_{L,R}(\gamma)$ at $X = \gamma$.

- $[L^*(X) + R^*(X) - O^*(X)] \odot \chi_{\mathcal{I}_{\mathcal{A}}}(X)$ is valued 0 at $X = \gamma$.

- $[P^*_{L,R}(X) - O^*(X)] \odot \chi_{\mathcal{I}_{\mathcal{M}}}(X)$ is valued 0 at $X = \gamma$.

- $[\mathtt{Wires}^*(X) + \delta_1 \cdot P_{\mathrm{id}}(X) + \delta_2 \cdot \frac{X^N-1}{X-1}] \odot F^*(X) \; - \; [\mathtt{Wires}^*(X) + \delta_1 \cdot P_{\boldsymbol{\sigma}}(X) + \delta_2 \cdot \frac{X^N-1}{X-1}] \odot F^*_1(X)$ is valued 0 at $X = \gamma$.

- $[\mathtt{Wires}^*(X) - P_{\mathtt{pub}}(X)] \odot \chi_{\mathcal{I}_{\mathtt{pub}}}(X)$ is valued 0 at $X = \gamma$.

Since the challenge γ was randomly and uniformly generated after $\mathbf{A}_L, \mathbf{A}_R, \mathbf{A}_0, \mathbf{A}_{L,R}, \mathbf{A}_{\widehat{F},1}$ had been sent, the Schwartz-Zippel lemma will then imply that with overwhelming probability, all of the following Hadamard product equations hold.

- $L^*(X) \odot R^*(X) \; = \; P^*_{L,R}(X)$

- $[L^*(X) + R^*(X) - O^*(X)] \odot \chi_{\mathcal{I}_{\mathcal{A}}}(X) \; = \; 0.$

- $[P^*_{L,R}(X) - O^*(X)] \odot \chi_{\mathcal{I}_{\mathcal{M}}}(X) \; = \; 0.$

- $\big[\mathtt{Wires}^*(X) + \delta_1 \cdot P_{\mathrm{id}}(X) + \delta_2 \cdot \frac{X^N-1}{X-1}\big] \odot F^*(X) \; = \; \big[\mathtt{Wires}^*(X) + \delta_1 \cdot P_{\boldsymbol{\sigma}}(X) + \delta_2 \cdot \frac{X^N-1}{X-1}\big] \odot F^*_1(X)$

- $\big[\mathtt{Wires}^*(X) - P_{\mathtt{pub}}(X)\big] \odot \chi_{\mathcal{I}_{\mathtt{pub}}}(X) \; = \; 0.$

Furthermore, the polynomials $F^*(X)$, $F^*_1(X)$ satisfy the equations

$$F^*_1(X) \; = \; F^{*\mathtt{RShift}}(X) \; := \; X \cdot F^*(X) \; (\mathrm{mod} \; X^N)$$

$$\mathtt{Coef}(F^* \; , \; N-1) = \mathtt{Coef}(F^*_1 \; , \; 0) = 1.$$

Thus, it will then follow that

$$\prod_{i=0}^{N-1} \mathtt{Coef}(\mathtt{Wires}^*(X) \; , \; i) + \delta_1 \cdot i + \delta_2 \; = \; \prod_{i=0}^{N-1} \mathtt{Coef}(\mathtt{Wires}^*(X) \; , \; i) + \delta_1 \cdot \boldsymbol{\sigma}(i) + \delta_2 \; ,$$

whence it will follow that with overwhelming probability, the polynomial $\mathtt{Wires}^*(X)$ is preserved under the action of $\boldsymbol{\sigma}$, which would complete the proof of soundness. We now prove claims 1 and 2.

Proof of Claims 1 and 2: Since

$$\widetilde{f}^*_{\gamma,\lambda}(X) \; = \; \widetilde{f}^*_{\gamma,\lambda,-}(X) \; + \; \gamma_{L,R} \cdot X^N \; + \; \widetilde{f}^*_{\gamma,\lambda,+}(X) \cdot X^{N+1},$$

it follows that with overwhelming probability, the extracted polynomials $\widetilde{f}^*_{\gamma,\lambda}(X)$, $\widetilde{f}^*_{\gamma,\lambda,-}(X)$ satisfy the equation

$$\mathtt{Coef}(\widetilde{f}^*_{\gamma,\lambda} \; , \; N) \; = \; \gamma_{L,R} + \mathtt{Coef}(\widetilde{f}^*_{\gamma,\lambda,-} \; , \; N).$$

It remains to argue that the polynomial $\widetilde{f}^*_{\gamma,\lambda,-}(X)$ is of degree $\leq N-1$ and the three polynomials $L^*(X)$, $R^*(X)$, $O^*(X)$ are all of degrees $\leq n-1$. As noted above, the pairing

check implies that with overwhelming probability, $\mathcal{E}_{\texttt{PPT}}$ can simulate the extractor $\mathcal{E}_{\texttt{multi-PC}}$ to extract a polynomial $\widehat{f}^*_{\widehat{\lambda}}(X)$ such that

$$\mathbf{A}_{\widehat{\lambda}} \;=\; \mathbf{g}_1^{\widehat{f}^*_{\widehat{\lambda}}(\mathbf{s})}$$

and

$$\widehat{f}^*_{\widehat{\lambda}}(\alpha^{-1}) \;=\; \beta_{\widehat{\lambda}} := \alpha^{1-n}\cdot\beta_L + \widehat{\lambda}\cdot\alpha^{1-n}\cdot\beta_R + \widehat{\lambda}^2\cdot\alpha^{1-n}\cdot\beta_O + \widehat{\lambda}^3\cdot\alpha^{1-N}\cdot\beta_{\gamma,\lambda,-}$$

$$= \alpha^{1-n}\cdot L^*(\alpha) + \widehat{\lambda}\cdot\alpha^{1-n}\cdot R^*(\alpha) + \widehat{\lambda}^2\cdot\alpha^{1-n}\cdot O^*(\alpha) + \widehat{\lambda}^3\cdot\alpha^{1-N}\cdot\widetilde{f}^*_{\gamma,\lambda,-}(\alpha).$$

Since the challenge α was randomly and uniformly generated after the element $\mathbf{A}_{\widehat{\lambda}}$ had been sent, the Schwartz-Zippel lemma implies that with overwhelming probability, the extracted polynomial $\widehat{f}^*_{\widehat{\lambda}}(X)$ is given by

$$\widehat{f}^*_{\widehat{\lambda}}(X) \;=\; X^{n-1}\cdot\big[L^*(X^{-1}) + \widehat{\lambda}\cdot R^*(X^{-1}) + \widehat{\lambda}^2\cdot O^*(X^{-1})\big] + \widehat{\lambda}^3\cdot X^{N-1}\cdot\widetilde{f}^*_{\gamma,\lambda,-}(X^{-1})$$

In particular, it follows that the expression on the right is a polynomial in $\mathbb{F}_p[X]$ rather than merely a rational function. Since the challenge $\widehat{\lambda}$ was randomly and uniformly generated after the elements $\mathbf{A}_L,\ \mathbf{A}_R,\ \mathbf{A}_O,\ \widetilde{\mathbf{A}}_{\gamma,\lambda,-}$ had been sent, lemma 2.1 implies that with overwhelming probability, all four of the rational functions

$$X^{n-1}\cdot L^*(X^{-1})\ ,\ X^{n-1}\cdot R^*(X^{-1})\ ,\ X^{n-1}\cdot O^*(X^{-1})\ ,\ X^{N-1}\cdot\widetilde{f}^*_{\gamma,\lambda,-}(X^{-1})$$

are polynomials. Thus, with overwhelming probability, the extracted polyonmials $L^*(X)$, $R^*(X)$, $O^*(X)$, $\widetilde{f}^*_{\gamma,\lambda,-}$ satisfy the degree upper bounds

$$\deg(L^*),\ \deg(R^*),\ \deg(O^*) \;\leq\; n-1\ \ ,\ \ \deg(\widetilde{f}^*_{\gamma,\lambda,-}) \;\leq\; N-1,$$

which completes the proof. □

B Subprotocol proofs

B.0.1 The degree upper bound protocol

Proposition B.1. *The protocol* PoDegUp *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 equations

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

verified via the (batchable) pairing checks

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

imply that with overwhelming probability, an extractor $\mathcal{E}_{\texttt{PPT}}$ can simulate the extractor $\mathcal{E}_{\texttt{multi-PC}}$ to extract polynomials $f^*(X)$, $\widehat{f}^*(X)$ such that

$$\mathbf{a} = \mathbf{g}_1^{f^*(\mathbf{s})}\ \ ,\ \ \widehat{\mathbf{a}} = \mathbf{g}_1^{\widehat{f}^*(\mathbf{s})}\ \ ,\ \ f^*(\alpha) = \beta\ \ ,\ \ \widehat{f}^*(\alpha^{-1}) = \alpha^{-n}\cdot\beta.$$

Since the challenge α was randomly and uniformly generated, the Schwartz-Zippel lemma implies that with overwhelming probability, the extracted polynomials $f^*(X)$, $\widehat{f}^*(X)$ satisfy the equation

$$\widehat{f}^*(X) = X^n \cdot f^*(X^{-1}).$$

Thus, with overwhelming probability, the rational function $X^n \cdot f^*(X^{-1})$ is a polynomial and hence, $\deg(f^*) \leq n$. □

B.0.2 The Hadamard product protocol

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

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

The subprotocol `BatchDiv` that with overwhelming probability, an extractor $\mathcal{E}_{\mathtt{PPT}}$ can simulate the extractor of `BatchDiv` to extract polynomials $f_1^*(X)$, $f_2^*(X)$, $f_{1,2}^*(X)$, $f_{\gamma,-}^*(X)$, $f_{\gamma,+}^*(X)$, $\widehat{f}_{\gamma,-}^*(X)$ such that:

- $\mathbf{g}_1^{f_1^*(\mathbf{s})} = \mathbf{a}_1, \quad f_1^*(\gamma \cdot \alpha) = \beta_{\gamma,1}$
- $\mathbf{g}_1^{f_2^*(\mathbf{s})} = \mathbf{a}_2, \quad f_2^*(\alpha^{-1}) = \widehat{\beta}_2$
- $\mathbf{g}_1^{f_{\gamma,-}^*(\mathbf{s})} = \mathbf{a}_{\gamma,-}, \quad f_{\gamma,-}^*(\alpha) = \beta_{\gamma,-}^*$
- $\mathbf{g}_1^{f_{\gamma,+}^*(\mathbf{s})} = \mathbf{a}_{\gamma,+}, \quad f_{\gamma,+}^*(\alpha) = \beta_{\gamma,+}^*$
- $\mathbf{g}_1^{\widehat{f}_{\gamma,-}^*(\mathbf{s})} = \widehat{\mathbf{a}}_{\gamma,-}, \quad \widehat{f}_{\gamma,-}^*(\alpha^{-1}) = \alpha^{1-N} \cdot f_{\gamma,-}^*(\alpha^{-1})$
- $f_{1,2}^*(\gamma) = \gamma_{1,2}$

Thus, the equation

$$\beta_{\gamma,1} \cdot \alpha^N \cdot \widehat{\beta}_2 \stackrel{?}{=} \beta_{\gamma,-} + \alpha^N \cdot \gamma_{1,2} + \alpha^{N+1} \cdot \beta_{\gamma,+}$$

may be rephrased as

$$f_1^*(\gamma \cdot \alpha) \cdot \alpha^N \cdot f_2^*(\alpha^{-1}) = f_{\gamma,-}^*(\alpha) + \alpha^N \cdot f_{1,2}^*(\gamma) + \alpha^{N+1} \cdot f_{\gamma,+}^*(X).$$

Since the challenge α was randomly and uniformly generated after the elements $\mathbf{a}_{\gamma,-}$, $\mathbf{a}_{\gamma,+}$, $\widehat{\mathbf{a}}_{\gamma,-}$ were sent, the Schwart-Zippel lemma implies that with overwhelming probability, the extracted polynomials satisfy the equation

$$f_1^*(\gamma \cdot X) \cdot X^N \cdot f_2^*(X^{-1}) = f_{\gamma,-}^*(X) + X^N \cdot f_{1,2}^*(\gamma) + X^{N+1} \cdot f_{\gamma,+}^*(X).$$

Since $\widehat{f}_{\gamma,-}^*(\alpha^{-1}) = \alpha^{1-N} \cdot f_{\gamma,-}^*(\alpha^{-1})$, the Schwartz-Zippel lemma implies that with overwhelming probability, the extracted polynomials satisfy the equation $\widehat{f}_{\gamma,-}^*(X) = X^{N-1} \cdot f_{\gamma,-}^*(X^{-1})$. In particular, it follows that with overwhelming probability, the rational function $X^{N-1} \cdot f_{\gamma,-}^*(X^{-1})$ is a polynomial, which implies that $\deg(f_{\gamma,-}^*) \leq N-1$.

Thus, the twisted product $f_1^*(\gamma \cdot X) \cdot X^N \cdot f_2^*(X^{-1})$ has coefficient $f_{1,2}^*(\gamma)$ at the position X^N. Since the challenge γ was randomly and uniformly generated, it follows that with overwhelming probability, $f_{1,2}^*(X) = f_1^* \odot f_2^*(X)$. □

B.0.3 Proof of cyclic right shift

Proposition B.3. *The protocol* `PoRShift` *is secure in the algebraic group model.*

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

The subprotocols for the degree upper bounds imply that with overwhelming probability, an extractor $\mathcal{E}_{\mathtt{PPT}}$ can extract polynomials $f^*(X),\ f^*(X)$ such that

$$\mathbf{a} = \mathbf{g}_1^{f^*(\mathbf{s})}\ \ ,\ \ \mathbf{b} = \mathbf{g}_1^{f_1^*(\mathbf{s})}\ \ ,\ \ \max\big(\deg(f^*),\ \deg(f_1^*)\big)\ \leq\ N-1.$$

The pairing check implies that with overwhelming probability, the extracted polynomials $f^*(X)$, $f_1^*(X)$ satisfy the equations

$$c_{N-1}\cdot(X^N-1) = X\cdot f^*(X) - f_1^*(X)$$

and hence,

$$f_1(X) \equiv X\cdot f(X)\ (\mathrm{mod}\ X^N-1),$$

whence it follows that with overwhelming probability, $f_1^*(X)$ is the cyclic right shift of $f(X)$. □

B.0.4 The permutation argument

Proposition B.4. *The protocol* `PoPerm` *is secure in the algebraic group model.*

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

The subprotocols $\mathtt{PoRShift}[\mathbf{g}_1,\ (\mathbf{a}^\vee,\ \mathbf{b}^\vee)]$, $\mathtt{PoHadProd}[\mathbf{g}_1,\ (\mathbf{a}_{\delta_1,\delta_2},\ \mathbf{a}^\vee),\ \ \mathbf{A}]$ and $\mathtt{PoHadProd}[\mathbf{g}_1,\ (\mathbf{b}_{\delta_1,\delta_2},\ \mathbf{b}^\vee),\ \ \mathbf{A}]$ imply that with overwhelming probability, an extractor $\mathcal{E}_{\mathtt{PPT}}$ can extract polynomials $f^*(X),\ \widetilde{f}^*(X),\ F^*(X)$ such that

$$\mathbf{a} = \mathbf{g}_1^{f^*(\mathbf{s})}\ \ ,\ \ \mathbf{b} = \mathbf{g}_1^{\widetilde{f}^*(\mathbf{s})}\ \ ,\ \ \deg(F)\leq N-1$$

$$[f^*(X)+\delta_1\cdot P_{\mathrm{id}}(X)+\delta_2\cdot\sum_{i=0}^{N-1}X^i]\ \odot\ F^*(X)\ =\ [\widetilde{f}^*(X)+\delta_1\cdot P_{\boldsymbol{\sigma}}(X)+\delta_1\cdot\sum_{i=0}^{N-1}X^i]\ \odot\ F^{*^{\mathtt{RShift}}}(X).$$

Furthermore, the pairing check

$$\mathbf{e}(\mathbf{Q}_0,\ ,\ \mathbf{g}_1^{\mathbf{s}})\ \stackrel{?}{=}\ \mathbf{e}(\mathbf{b}^\vee\cdot\mathbf{g}_1^{-1}\ ,\ \mathbf{g}_2)$$

implies that with overwhelming probability, the extracted polynomial $F^*(X)$ and its cyclic right $F^{*^{\mathtt{RShift}}} := X\cdot F^*(X)\ (\mathrm{mod}\ X^N-1)$ satisfy the equations

$$\mathtt{Coef}(F^{*^{\mathtt{RShift}}}\ ,\ 0)\ =\ \mathtt{Coef}(F^*\ ,\ N-1)=1.$$

Thus, with overwhelming probability, the extracted polynomials $f(X),\ \widetilde{f}^*(X)$ satisfy the equation

$$\prod_{i=0}^{N-1}\frac{\mathtt{Coef}(f^*\ ,\ i)+\delta_1\cdot i+\delta_2}{\mathtt{Coef}(\widetilde{f}^*\ ,\ i)+\delta_1\cdot\boldsymbol{\sigma}(i)+\delta_2}=1.$$

Since the challenges $\delta_1,\ \delta_2$ were randomly and uniformly generated, it follows that with overwhelming probability,

$$\mathtt{Coef}(\widetilde{f}^*\ ,\ i)\ =\ \mathtt{Coef}(f^*\ ,\ \boldsymbol{\sigma}(i))\ \ \forall\, i,$$

which completes the proof of soundness. □

C The lookup protocol

Proposition C.1. *The protocol 5.1 is secure in the algebraic group model.*

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

The three Hadamard product subprotocols

$$\mathtt{PoHadProd}[\mathbf{g}_1,\ (\mathbf{a},\ \mathbf{C}_{\mathcal{I}}),\ \mathbf{a}_{\mathcal{I}}],\ \mathtt{PoHadProd}[\mathbf{g}_1,\ (\mathbf{a},\ \mathbf{C}_{\mathcal{J}}),\ \mathbf{T}_{\mathcal{J}}],\ \mathtt{PoHadProd}[\mathbf{g}_1,\ (\mathbf{C}_{\mathcal{J}},\ \mathbf{C}_{\mathrm{id}}\cdot\mathbf{C}_{\mathcal{J}}^{-1}),\ \mathbf{g}_1]$$

imply that with overwhelming probability, an extractor $\mathcal{E}_{\mathtt{PPT}}$ can extract an index set $\mathcal{J}^*$ (with characteristic polynomial $\chi_{\mathcal{J}^*}(X)$) and a polynomial $f^*(X)$ such that

$$\mathbf{a} = \mathbf{g}_1^{f^*(\mathbf{s})}\ ,\ \ \mathbf{a}_{\mathcal{I}} = \mathbf{g}_1^{f^*\odot\chi_{\mathcal{I}}(\mathbf{s})}\ ,\ \ \mathbf{C}_{\mathcal{J}} = \mathbf{g}_1^{\chi_{\mathcal{J}^*}(\mathbf{s})}\ ,\ \ \mathbf{T}_{\mathcal{J}} = \mathbf{g}_1^{T\odot\chi^*_{\mathcal{J}}(\mathbf{s})}.$$

Furthermore, the subprotocols

$$\mathtt{PoHadProd}[\mathbf{g}_1,\ (\mathbf{a}_{\mathcal{I},\alpha,\mathrm{inv}},\ \mathbf{a}\cdot\mathbf{C}_{\mathrm{id}}^{\alpha}),\ \mathbf{C}_{\mathcal{I}}]\ \ ,\ \ \mathtt{PoHadProd}[\mathbf{g}_1,\ (\mathbf{T}_{\mathcal{J},\alpha,\mathrm{inv}},\ \mathbf{T}\cdot\mathbf{C}_{\mathrm{id}}^{\alpha}),\ \mathbf{C}_{\mathcal{J}}]$$

imply that with overwhelming probability, $\mathcal{E}_{\mathtt{PPT}}$ can extract polynomials $f^*_{\mathcal{I},\alpha,\mathrm{inv}}(X)$, $T^*_{\mathcal{J}^*,\alpha,\mathrm{inv}}(X)$ such that

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

$$T^*_{\mathcal{J}^*,\alpha,\mathrm{inv}}(X)\ \odot\ \big[T(X)+\alpha\cdot P_{\mathrm{id}}(X)\big] = \chi_{\mathcal{J}^*}(X)\ \ ,\ \ f^*_{\mathcal{I},\alpha,\mathrm{inv}}(X)\ \odot\ \big[f^*(X)+\alpha\cdot P_{\mathrm{id}}(X)\big] = \chi_{\mathcal{I}}(X).$$

Thus, with overwhelming probability, the extracted polynomials $T^*_{\mathcal{J}^*}(X)$, $f^*_{\mathcal{I}}(X)$ satisfy the equations

$$\mathtt{Coef}\big(T^*_{\mathcal{J}^*,\alpha,\mathrm{inv}}\ ,\ k\big)\ =\ \begin{cases}\big[\alpha+\mathtt{Coef}\big(T\ ,\ k\big)\big]^{-1}\ \text{if}\ k\in\mathcal{J}^*\\ 0\ \text{if}\ k\notin\mathcal{J}^*\end{cases}$$

$$\mathtt{Coef}\big(f^*_{\mathcal{I},\alpha,\mathrm{inv}}\ ,\ k\big)\ =\ \begin{cases}\big[\alpha+\mathtt{Coef}\big(f^*\ ,\ k\big)\big]^{-1}\ \text{if}\ k\in\mathcal{I}\\ 0\ \text{if}\ k\notin\mathcal{I}\end{cases}\ .$$

The pairing check

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

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

$$\beta_{\mathcal{I},\alpha,\mathrm{inv}}\ =\ f^*_{\mathcal{I},\alpha,\mathrm{inv}}(1)\ =\ \sum_{i\in\mathcal{I}}\big[\alpha+\mathtt{Coef}\big(f^*\ ,\ k\big)\big]^{-1}.$$

Furthermore, the subprotocol $\mathtt{PoDotProd}[\mathbf{g}_1,\ (\mathbf{M}_{\mathcal{J}},\ \mathbf{T}_{\mathcal{J},\alpha,\mathrm{inv}}),\ \beta_{\mathcal{I},\alpha,\mathrm{inv}}]$ implies that with overwhelming probability, $\mathcal{E}_{\mathtt{PPT}}$ can extract a polynomial $\mathtt{Mul}_{\mathcal{J}^*}(X)$ such that

$$\mathbf{M}_{\mathcal{J}} = \mathbf{g}_1^{\mathtt{Mul}_{\mathcal{J}^*}(\mathbf{s})}\ \ ,$$

$$f^*_{\mathcal{I},\alpha,\mathrm{inv}}(\alpha)\ =\ \mathtt{Mul}_{\mathcal{J}^*}(X)\circ T^*_{\mathcal{J}^*,\alpha,\mathrm{inv}}(X)\ =\ \sum_{j\in\mathcal{J}^*}\mathtt{Coef}\big(\mathtt{Mul}_{\mathcal{J}^*}\ ,\ j\big)\cdot\big[\alpha+\mathtt{Coef}\big(T\ ,\ j\big)\big]^{-1}\ \ ,$$

where $\circ$ denotes the dot product.

Since the challenge α was randomly and uniformly generated after the element $\mathbf{M}_{\mathcal{J}}\in\mathbb{G}_1$ was sent, the Schwartz-Zippel lemma implies that with overwhelming probability, the extracted polynomials $f^*(X)$, $\mathtt{Mul}_{\mathcal{J}^*}(X)$ satisfy the equality

$$\sum_{i\in\mathcal{I}}\big[X+\mathtt{Coef}\big(f^*\ ,\ i\big)\big]^{-1}\ =\ \sum_{j\in\mathcal{J}^*}\mathtt{Coef}\big(\mathtt{Mul}_{\mathcal{J}^*}\ ,\ j\big)\cdot\big[X+\mathtt{Coef}\big(T\ ,\ j\big)\big]^{-1}$$

of rational functions. It now follows that with overwhelming probability, the multiset of the coefficients of $f^*(X)$ at the index set $\mathcal{I}$ has underlying set $\{\mathtt{Coef}\big(T\ ,\ j\big)\ :\ j\in\mathcal{J}^*\}$ with each $\mathtt{Coef}\big(T\ ,\ j\big)$ occurring with multiplicity $\mathtt{Coef}\big(\mathtt{Mul}_{\mathcal{J}^*}\ ,\ j\big)$. □

D The univariate custom gate protocol

Proposition D.1. *The protocol 6.1 is secure in the algebraic group model.*

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

The equations

$$\mathbf{Q}_1^{\mathbf{s}-\alpha}\stackrel{?}{=}\mathbf{b}_\lambda\cdot\mathbf{g}_1^{-\beta_\lambda}\ \ ,\ \ \mathbf{Q}_2^{\mathbf{s}-\alpha}\stackrel{?}{=}\mathbf{C}_P\cdot\mathbf{g}_1^{-\beta_P}\ \ ,\ \ \mathbf{Q}_3^{\mathbf{s}-\lambda}\stackrel{?}{=}\mathbf{b}^\vee\cdot\mathbf{g}_1^{-\beta_\lambda}$$

verified via pairing checks imply that with overwhelming probability, $P(\alpha)=\beta_P$ and an extractor $\mathcal{E}_{\mathtt{PPT}}$ can extract polynomials $h_\lambda^*(X),\ h_\alpha^{\vee,*}(X)$ such that

$$\mathbf{b}_\lambda=\mathbf{g}_1^{h_\lambda(\mathbf{s})}\ \ ,\ \ \mathbf{b}^\vee=\mathbf{g}_1^{h_\alpha^{\vee,*}(\mathbf{s})}\ \ ,\ \ h_\lambda^*(\alpha)=\beta_\lambda=h_\alpha^{\vee,*}(\lambda).$$

The subprotocols $\mathtt{PoHadProd}[\mathbf{g}_1,\ (\mathbf{a},\ \mathbf{C}_\mathcal{I}),\ \mathbf{a}_\mathcal{I}]$, $\mathtt{PoHadProd}[\mathbf{g}_1,\ (\widetilde{\mathbf{a}},\ \mathbf{C}_\mathcal{I}),\ \widetilde{\mathbf{a}}_\mathcal{I}]$ imply that with overwhelming probability, $\mathcal{E}_{\mathtt{PPT}}$ can extract polynomials $f^*(X)$, $\widetilde{f}^*(X)$ such that the Hadamard products $f_\mathcal{I}^*\ :=\ f^*(X)\odot\chi_\mathcal{I}(X),\ \widetilde{f}_\mathcal{I}^*\ :=\ \widetilde{f}^*(X)\odot\chi_\mathcal{I}(X)$ satisfy the equations

$$\mathbf{a}_\mathcal{I}:=\mathbf{g}_1^{f_\mathcal{I}^*(\mathbf{s})}\ \ ,\ \ \widetilde{\mathbf{a}}_\mathcal{I}:=\mathbf{g}_1^{\widetilde{f}_\mathcal{I}^*(\mathbf{s})}.$$

Furthermore, the subprotocols $\mathtt{PoHadProd}[\mathbf{g}_1,\ (\mathbf{a}_{\delta_1,\delta_2},\ \mathbf{a}^\vee),\ \mathbf{A}]$, $\mathtt{PoHadProd}[\mathbf{g}_1,\ (\mathbf{b}_{\delta_1,\delta_2},\ \mathbf{b}^\vee),\ \mathbf{A}]$ imply that with overwhelming probability, the extracted polynomials $h_\alpha^{\vee,*}(X)$, $f^*(X)$ satisfy the following equations:

$$h_\alpha^{\vee,*}(X)\ \odot\ [\alpha\cdot\chi_\mathcal{I}(X)-f^*(X)]\ =\ P(\alpha)\cdot\chi_\mathcal{I}(X)-\widetilde{f}^*(X)\odot\chi_\mathcal{I}(X)$$

$$h_\alpha^{\vee,*}(X)\ \odot\ \chi_\mathcal{I}(X)\ =\ h_\alpha^{\vee,*}(X).$$

Thus, with overwhelming probability,

$$h_\alpha^{*,\vee}(X)\ =\ \sum_{i\in\mathcal{I}}\frac{P(\alpha)-\mathtt{Coef}(\widetilde{f}^*\ ,\ i)}{\alpha-\mathtt{Coef}(f^*\ ,\ i)}\cdot X^i.$$

In particular,

$$h_\lambda^*(\alpha)=\beta_\lambda=h_\alpha^{\vee,*}(\lambda)=\sum_{i\in\mathcal{I}}\frac{P(\alpha)-\mathtt{Coef}(\widetilde{f}^*\ ,\ i)}{\alpha-\mathtt{Coef}(f^*\ ,\ i)}\cdot\lambda^i.$$

Since the challenge α was randomly and uniformly generated after the element $\mathbf{b}_\lambda$ was sent, it follows that with overwhelming probability, the extracted polynomials $h_\lambda^*(X)$, $f^*(X)$ and $\widetilde{f}^*(X)$ satisfy the equation

$$h_\lambda^*(X)\ =\ \sum_{i\in\mathcal{I}}\frac{P(X)-\mathtt{Coef}(\widetilde{f}^*\ ,\ i)}{X-\mathtt{Coef}(f^*\ ,\ i)}\cdot\lambda^i.$$

Thus, the sum

$$\sum_{i\in\mathcal{I}} \frac{P(X) - \texttt{Coef}(\widetilde{f^*}\ ,\ i)}{X - \texttt{Coef}(f^*\ ,\ i)} \cdot \lambda^i.$$

is a polynomial. Since the challenge λ was randomly and uniformly generated, lemma 2.1 implies that with overwhelming probability, each of the rational functions

$$\frac{P(X) - \texttt{Coef}(\widetilde{f^*}\ ,\ i)}{X - \texttt{Coef}(f^*\ ,\ i)} \qquad (i \in \mathcal{I})$$

is a polynomial. This, in turn, implies that with overwhelming probability,

$$P(\texttt{Coef}(f^*\ ,\ i)) \ =\ \texttt{Coef}(\widetilde{f^*}\ ,\ i)\ \ \forall\, i\ \in\ \mathcal{I},$$

which completes the proof. □

E List of protocols and the relations they are arguments of knowledge for

1. PoKE* (*Proof of knowledge of the exponent with base* $\mathbf{g}_1$)

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

2. PoKE (*Proof of knowledge of the exponent*)

$$\mathcal{R}_{\texttt{KE}}[a,\ b] = \{((\mathbf{a}, \mathbf{b}) \in \mathbb{G}_1^2),\ f(X) \in \mathbb{F}_p[X]) : \mathbf{a}^{f(\mathbf{s})} = \mathbf{b}\}$$

3. PoProd (*Proof of product*)

$$\mathcal{R}_{\texttt{Prod}}[\mathbf{g}_1,\ (\mathbf{a}_1, \mathbf{a}_2),\ \mathbf{a}_{1,2}] = \left\{ \begin{array}{l} (\mathbf{a}_1, \mathbf{a}_2 \in \mathbb{G}_1),\ f_1(X), f_2(X) \in \mathbb{F}_p[X]) : \\ \mathbf{g}_1^{f_1(\mathbf{s})} = \mathbf{a}_1\ ,\ \mathbf{g}_1^{f_2(\mathbf{s})} = \mathbf{a}_2\ ,\ \mathbf{g}_1^{f_1(\mathbf{s})\cdot f_2(\mathbf{s})} = \mathbf{a}_{1,2} \end{array} \right\}$$

4. PoTwist (*Proof of twist*)

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

5. PoRev (*Proof of reverse polynomial*)

$$\mathcal{R}_{\texttt{Rev}}[\mathbf{g}_1,\ (\mathbf{a}_1,\ n),\ \mathbf{a}_2] = \{(\mathbf{a}_1, \mathbf{a}_2 \in \mathbb{G}_1,\ n \in \mathbb{Z}),\ f(X) \in \mathbb{F}_p[X]) : \mathbf{g}_1^{f(\mathbf{s})} = \mathbf{a}_1\ ,\ \mathbf{g}_1^{\mathbf{s}^{\deg(f)}\cdot f(\mathbf{s}^{-1})} = \mathbf{a}_2\}$$

6. PoDegUp (*Proof of degree upper bound*)

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

($\circ$ denotes the dot product of polynomials)

7. PoHadProd (*Proof of Hadamard product*)

$$\mathcal{R}_{\texttt{HadProd}}[\mathbf{g}_1,\ (\mathbf{a}_1, \mathbf{a}_2),\ \mathbf{a}_{1,2}] = \left\{ \begin{array}{l} \big((\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_{1,2} \in \mathbb{G}_1),\ f_1(X), f_2(X) \in \mathbb{F}_p[X]\big) : \\ \mathbf{g}_1^{f_1(\mathbf{s})} = \mathbf{a}_1\ ,\ \mathbf{g}_1^{f_2(\mathbf{s})} = \mathbf{a}_2\ ,\ \mathbf{g}_1^{f_1 \odot f_2(\mathbf{s})} = \mathbf{a}_{1,2} \end{array} \right\}$$

($\odot$ denotes the Hadamard product of polynomials).

8. PoDotProd (*Proof of dot product*)

$$\mathcal{R}_{\texttt{DotProd}}[\mathbf{g}_1,\ (\mathbf{a}_1, \mathbf{a}_2),\ \alpha_{1,2}] = \left\{ \begin{array}{l} (\mathbf{a}_1, \mathbf{a}_2 \in \mathbb{G}_1),\ f_1(X),\ f_2(X)\ \in\ \mathbb{F}_p[X]) : \\ \mathbf{g}_1^{f_1(\mathbf{s})} = \mathbf{a}_1\ ,\ \mathbf{g}_1^{f_2(\mathbf{s})} = \mathbf{a}_2\ ,\ f_1 \odot f_2(1) = \alpha_{1,2} \end{array} \right\}$$

9. `PoRShift` (*Proof of cyclic right shift*)

$$\mathcal{R}_{\texttt{RShift}}[\mathbf{g}_1,\ (\mathbf{a},\mathbf{b})] = \left\{ \begin{array}{l} \big((\mathbf{a},\ \mathbf{b}\ \in\ \mathbb{G}_1) \\ f(X)\ \in\ \mathbb{F}_p[X] \text{ with } \deg(f) \leq N\big) : \\ \mathbf{a} = \mathbf{g}_1^{f(\mathbf{s})}\ \ ,\ \ \mathbf{b} = \mathbf{g}_1^{f^{\texttt{RShift}}(\mathbf{s})} \end{array} \right\}$$

$f^{\texttt{RShift}}(X)$ is the polynomial $X \cdot f(X) \pmod{X^N - 1}$.

10. `PoPerm` (*Proof of permutation*)

$$\mathcal{R}_{\texttt{Perm}}[\mathbf{g}_1,\ (\mathbf{a},\mathbf{C}_{\boldsymbol{\sigma}}),\ \mathbf{b})] = \left\{ \begin{array}{l} \big((\mathbf{a},\ \mathbf{C}_{\boldsymbol{\sigma}},\ \mathbf{b}\ \in\ \mathbb{G}_1)] \\ f(X)\ \in\ \mathbb{F}_p[X] \text{ with } \deg(f) \leq N-1 \\ \text{Permutation } \boldsymbol{\sigma} :\ [0,\ N-1]\ \longrightarrow\ [0,\ N-1]\big) : \\ \\ \mathbf{a} = \mathbf{g}_1^{f(\mathbf{s})}\ \ ,\ \ \mathbf{b} = \mathbf{g}_1^{f^{\boldsymbol{\sigma}}(\mathbf{s})}\ \ ,\ \ \mathbf{C}_{\boldsymbol{\sigma}} = \mathbf{g}_1^{\sum_{i=0}^{n} \boldsymbol{\sigma}(i)\cdot\mathbf{s}^i} \end{array} \right\}$$

11. `PoCoefSet` (*Protocol for coeffcient subset containment*)

$$\mathcal{R}_{\texttt{CoefSet}}[\mathbf{g}_1,\ (\mathbf{a},\ \mathbf{T}),\ \mathbf{C}_{\mathcal{I}}] = \left\{ \begin{array}{l} \big((\mathbf{a},\ \mathbf{T},\ \mathbf{C}_{\mathcal{I}}\ \in\ \mathbb{G}_1), \\ f(X),\ T(X)\ \in\ \mathbb{F}_p[X]\ \ ,\ \ \mathcal{I}\ \subseteq\ [0,\ N-1]\big) : \\ \mathbf{a} = \mathbf{g}_1^{f(\mathbf{s})}\ \ ,\ \ \mathbf{T} = \mathbf{g}_1^{T(\mathbf{s})}\ \ ,\ \ \mathbf{C}_{\mathcal{I}} = \mathbf{g}_1^{\chi_{\mathcal{I}}(\mathbf{s})} \\ \{\texttt{Coef}(f\ ,\ i) :\ i \in \mathcal{I}\}\ \subseteq\ \{\texttt{Coef}(T\ ,\ i) :\ j \in [0,\ N-1]\} \end{array} \right\}$$

12. `PoSubseq` (*Protocol for subsequences*)

$$\mathcal{R}_{\texttt{Subseq}}[\mathbf{g}_1,\ (\mathbf{a},\ \widetilde{\mathbf{a}}),\ (\mathbf{C}_{\mathcal{I}}.\ \mathbf{C}_{\widetilde{\mathcal{I}}})] = \left\{ \begin{array}{l} \big((\mathbf{a},\ \widetilde{\mathbf{a}},\ \mathbf{C}_{\mathcal{I}},\ \mathbf{C}_{\widetilde{\mathcal{I}}}\ \in\ \mathbb{G}_1), \\ f(X),\ \widetilde{f}(X)\ \in\ \mathbb{F}_p[X]\ \ ,\ \ \mathcal{I},\ \widetilde{\mathcal{I}}\ \subseteq\ [0,\ N-1] \\ \text{Permutation } \boldsymbol{\sigma} :\ [0,\ N-1]\ \longrightarrow [0,\ N-1]\big) : \\ \mathbf{a} = \mathbf{g}_1^{f(\mathbf{s})}\ \ ,\ \ \mathbf{b} = \mathbf{g}_1^{\widetilde{f}(\mathbf{s})}\ \ ,\ \ \mathbf{C}_{\mathcal{I}} = \mathbf{g}_1^{\chi_{\mathcal{I}}(\mathbf{s})}\ \ ,\ \ \mathbf{C}_{\widetilde{\mathcal{I}}} = \mathbf{g}_1^{\chi_{\widetilde{\mathcal{I}}}(\mathbf{s})} \\ [f(X) \odot \chi_{\mathcal{I}}(X)]^{\boldsymbol{\sigma}}\ =\ [\widetilde{f}(X) \odot \chi_{\widetilde{\mathcal{I}}}(X)] \\ \forall\ i,j \in \mathcal{I},\quad i < j\ \iff \boldsymbol{\sigma}(i) < \boldsymbol{\sigma}(j) \end{array} \right\}$$

13. `PoCoefRel` (*Protocol for coefficient relation*)

$$\mathcal{R}_{\texttt{CoefRel}}[\mathbf{g}_1,\ (\mathbf{a},\ \widetilde{\mathbf{a}},\ \mathtt{C}_{\mathcal{I}}),\ P(X)] = \left\{ \begin{array}{l} \big((\mathbf{a},\ \widetilde{\mathbf{a}},\ \mathtt{C}_{\mathcal{I}}\ \in\ \mathbb{G}_1,\ P(X) \in \mathbb{F}_p[X]), \\ f(X),\ \widetilde{f}(X)\ \in \mathbb{F}_p\ ,\ \mathcal{I} \subseteq [0,\ \deg(f)]\big) : \\ \mathbf{g}_1^{f(\mathbf{s})} = \mathbf{a}\ ,\ \mathbf{g}_1^{\widetilde{f}(\mathbf{s})} = \widetilde{\mathbf{a}}\ ,\ \mathbf{g}_1^{\chi_{\mathcal{I}}(\mathbf{s})} = \mathtt{C}_{\mathcal{I}} \\ P(\texttt{Coef}(f\ ,\ i)) = \texttt{Coef}(\widetilde{f}\ ,\ i)\ \forall\ i \in \mathcal{I} \end{array} \right\}$$

F Benchmarks for polynomial products over FFT unfriendly fields (NTL library)

We provide some benchmarks for the polynomial multiplication in Step 7 of the SNARK. We used the NTL library to multiply two randomly generated polynomials of the same degree. The benchmarks were performed on a 6-core Intel machine with 2.60 GHz clock speed.

The Ed25519 base field (2-adicity 2):

Degree	Time (seconds)
10^5	0.118007
10^6	1.26678
10^7	15.5655

The secp256k1 base field (2-adicity 1):

Degree	Time (seconds)
10^5	0.122884
10^6	1.27445
10^7	15.4692

The BN254 base field (2-adicity 1):

Degree	Time (seconds)
10^5	0.116782
10^6	1.30053
10^7	15.8205

The BLS12-381 base field (2-adicity 1):

Degree	Time (seconds)
10^5	0.181763
10^6	1.95427
10^7	23.97