PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge

Ariel Gabizon
Protocol Labs

Zachary J. Williamson
Aztec Protocol

Oana Ciobotaru

August 24, 2019

Abstract

zk-SNARK constructions that utilize an updatable universal structured reference string remove one of the main obstacles in deploying zk-SNARKs[GKM$^+$]. The important work of Maller et al. [MBKM19] presented Sonic - the first potentially practical zk-SNARK with fully succinct verification for general arithmetic circuits with such an SRS. However, the version of Sonic enabling fully succinct verification still requires relatively high proof construction overheads. We present a universal SNARK construction with fully succinct verification, and significantly lower prover running time (roughly 7.5-20 less group exponentiations than [MBKM19] in the fully succinct verifier mode depending on circuit structure).

Similarly to [MBKM19, BCC$^+$16] we rely on a permutation argument based on Bayer and Groth [BG12]. However, we focus on "Evaluations on a subgroup rather than coefficients of monomials"; which enables simplifying both the permutation argument and the artihmetization step.

1 Introduction

Due to real-world deployments of zk-SNARKs, it has become of significant interest to have the structured reference string (SRS) be constructible in a "universal and updatable" fashion. Meaning that the same SRS can be used for statements about all circuits of a certain bounded size; and that at any point in time the SRS can be updated by a new party, such that the honesty of only one party from all updaters up to that point is required for soundness. For brevity, let us call a zk-SNARK with such a setup process *universal*.

For the purpose of this introduction, let us say a zk-SNARK for circuit satisfiability is *fully succinct* if

1. The preprocessing[1] phase/SRS generation run time is quasilinear in circuit size.
2. The prover run time is quasilinear in circuit size.
3. The proof length is logarithmic[2] in circuit size.
4. The verifier run time is polylogarithmic in circuit size.[3]

Maller et al. [MBKM19] constructed for the first time a universal fully succinct zk-SNARK for circuit satisfiability, called Sonic.

[MBKM19] also give a version of Sonic with dramatically improved prover run time, at the expense of efficient verification only in a certain amortized sense.

1.1 Our results

In this work we give a universal fully-succinct zk-SNARK with significantly improved prover run time compared to fully-succinct Sonic.

At a high level our improvements stem from a more direct arithmetization of a circuit as compared to the [BCC+16]-inspired arithmetization of [MBKM19]. This is combined with a permutation argument over univariate evaluations on a multiplicative subgroup rather than over coefficients of a bivariate polynomial as in [BCC+16, MBKM19].

In a nutshell, one reason multipicative subgroups are useful is that [BCC+16, MBKM19] use a permutation argument based on Bayer and Groth [BG12]. Ultimately, in the "grand product argument", this reduces to checking relations between coefficients of polynomials at "neighbouring monomials".

We observe that if we think of the points $x, \mathbf{g}\cdot x$ as neighbours, where $\mathbf{g}$ is a generator of a multiplicative subgroup of a field $\mathbb{F}$, it is very convenient to check relations between different polynomials at such pairs of points.

A related convenience is that multiplicative subgroups interact well with Lagrange bases. For example, suppose $H \subset \mathbb{F}$ is a multiplicative subgroup of order $n+1$, and $x \in H$. The polynomial L_x of degree at most n that vanishes on $H \setminus \{x\}$ and has $f(x) = 1$, has a very sparse representation of the form

$$L_x(X) = \frac{c_x(X^{n+1} - 1)}{(X - x)},$$

for a constant c_x. This is beneficial when constructing an efficiently verifiable [BG12]-style permutation argument in terms of polynomial identities.

[1] We use the term SNARK in this paper for what is sometimes called a "SNARK with preprocessing"(see e.g. [GGPR13]) where one allows a one-time verifier computation that is polynomial rather than polylogarithmic in the circuit size. In return, the SNARK is expected to work for all *non-uniform circuits*, rather than only statements about uniform computation.

[2] From a theoretical point of view, polylogarithmic proof length is more natural; but logarithmic nicely captures recent constructions with a constant number of group elements, and sometimes is a good indication of the "practicality barrier".

[3] In many definitions, only proof size is required to be polylogarithmic. For example, in the terminology of [GGPR13], additionally requiring polylogarithmic verifier run time means the SNARK is *unsubtle*.

1.2 Efficiency Analysis

We compare the performance of this work to the state of the art, both for non-universal SNARKs and universal SNARKs. At the time of publication, the only fully succinct universal SNARK construction is (the fully-succinct version of) the Sonic protocol [MBKM19]. This protocol requires the prover compute $273n$ $\mathbb{G}_1$ group exponentiations, where n is the number of multiplication gates. In fully-succinct Sonic, every wire can only be used in three linear relationships, requiring the addition of 'dummy' multiplication gates to accomodate wires used in more than three addition gates. This increase in the multiplication gate count is factored into the prover computation estimate (see [MBKM19] for full details).

Our universal SNARK requires the prover to compute 6 polynomial commitments, combined with two opening proofs to evaluate the polynomial commitments at a random challenge point. The combined degree of the polynomials is $12(n+a)$, where n is the number of multiplication gates and a is the number of addition gates. Currently, the most efficient fully-succinct SNARK construction available is Groth's 2016 construction [Gro16], which requires a unique, non-updateable CRS per circuit. Proof construction times are dominated by $3n+m$ $\mathbb{G}_1$ and n $\mathbb{G}_2$ group exponentiations. If we assume that one $\mathbb{G}_2$ exponentiation is equivalent to three $\mathbb{G}_1$ exponentiations, this yields $6n+m$ equivalent $\mathbb{G}_1$ group exponentiations.

Performing a direct comparison between these SNARK arithmetizations requires some admittedly subjective assumptions. When evaluating common circuits, we found that the number of addition gates is 2x the number of multiplication gates, however circuits that are optimized under the assumption that addition gates are 'free' (as is common in R1CS based systems like [Gro16]) will give worse estimates.

At one extreme, for a circuit containing no addition gates and only fan-in-2 multiplication gates, our universal SNARK proofs require ≈ 1.5 times more prover work than [Gro16], and ≈ 22 times less work than Sonic. If $a=2n$, the ratios change to ≈ 3 times more prover work than [Gro16], and ≈ 7.5 times less work than Sonic. If $a=5n$, this changes to ≈ 4 times more work than [Gro16], and ≈ 3.8 times less work than Sonic.

Table 1: Prover comparison. m = number of wires, n = number of multiplication gates, a = number of addition gates

	size $\leq d$ SRS	**size $= n$ CRS/SRS**	**prover work**	**proof length**	**succinct**	**universal**
Groth'16	-	$3n+m$ $\mathbb{G}_1$	$3n+m-\ell$ $\mathbb{G}_1$ exp, n $\mathbb{G}_2$ exp	2 $\mathbb{G}_1$, 1 $\mathbb{G}_2$	✓	✗
Sonic (helped)	$12d$ $\mathbb{G}_1$, $12d$ $\mathbb{G}_2$	$12n$ $\mathbb{G}_1$	$18n$ $\mathbb{G}_1$ exp	4 $\mathbb{G}_1$, 2 $\mathbb{F}$	✗	✓
Sonic (succinct)	$4d$ $\mathbb{G}_1$, $4d$ $\mathbb{G}_2$	$36n$ $\mathbb{G}_1$	$273n$ $\mathbb{G}_1$ exp	20 $\mathbb{G}_1$, 16 $\mathbb{F}$	✓	✓
Auroralight	$2d$ $\mathbb{G}_1$, $2d$ $\mathbb{G}_2$	$2n$ $\mathbb{G}_1$	$8n$ $\mathbb{G}_1$ exp	6 $\mathbb{G}_1$, 4 $\mathbb{F}$	✗	✓
This work	d $\mathbb{G}_1$, 1 $\mathbb{G}_2$	$3n+3a$ $\mathbb{G}_1$, 1 $\mathbb{G}_2$	$12n+12a$ $\mathbb{G}_1$ exp	8 $\mathbb{G}_1$, 10 $\mathbb{F}$	✓	✓

Verifier computation per proof is shown in table 2. Only two bilinear pairing operations are required, due to the simple structure of the committed prover polynomials. In addition, the $\mathbb{G}_2$ elements in each pairing are fixed, enabling optimizations that reduce pairing computation time by $\approx 30\%$ [CS10].

Table 2: Verifier comparison per proof, P=pairing, ℓ=num of pub inputs. For non-succinct protocols, additional helper work is specified

	verifier work	**elem. from helper**	**extra verifier work in helper mode**
Groth'16	$3P$, ℓ $\mathbb{G}_1$ exp	-	-
Sonic (helped)	$10P$	3 $\mathbb{G}_1$, 2 $\mathbb{F}$	$4P$
Sonic (succinct)	$13P$	-	-
Auroralight	$5P$, 6 $\mathbb{G}_1$ exp	8 $\mathbb{G}_1$, 10 $\mathbb{F}$	$12P$
This work	$2P$, 18 $\mathbb{G}_1$ exp	-	-

2 Preliminaries

2.1 Terminology and Conventions

We assume our field $\mathbb{F}$ is of prime order. We denote by $\mathbb{F}_{<d}[X]$ the set of univariate polynomials over $\mathbb{F}$ of degree smaller than d. We assume all algorithms described receive as an implicit parameter the security parameter λ.

Whenever we use the term "efficient", we mean an algorithm running in time $\mathsf{poly}(\lambda)$. Furthermore, we assume an "object generator" $\mathcal{O}$ that is run with input λ before all protocols, and returns all fields and groups used. Specifically, in our protocol $\mathcal{O}(\lambda) = (\mathbb{F}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, e, g_1, g_2, g_t)$ where

- $\mathbb{F}$ is a prime field of super-polynomial size $r = \lambda^{\omega(1)}$.
- $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$ are all groups of size r, and e is an efficiently computable non-degenerate pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$.
- g_1, g_2 are uniformly chosen generators such that $e(g_1, g_2) = g_t$.

We usually let the λ parameter be implicit, i.e. write $\mathbb{F}$ instead of $\mathbb{F}(\lambda)$. We write $\mathbb{G}_1$ and $\mathbb{G}_2$ additively. We use the notations $[x]_1 := x \cdot g_1$ and $[x]_2 := x \cdot g_2$.

We often denote by $[n]$ the integers $\{1, \ldots, n\}$.

universal SRS-based public-coin protocols We describe public-coin (meaning the verifier messages are uniformly chosen) interactive protocols between a prover and verifier; when deriving results for non-interactive protocols, we implicitly assume we can get a proof length equal to the total communcation of the prover, using the Fiat-Shamir transform/a random oracle. Using this reduction between interactive and non-interactive protocols, we can refer to the "proof length" of an interactive protocol.

We allow our protocols to have access to a structured reference string (SRS) that can be derived in deterministic $\mathsf{poly}(\lambda)$-time from an "SRS of monomials" of the form $\left\{[x^i]_1\right\}_{a\le i\le b}, \left\{[x^i]_2\right\}_{c\le i\le d}$, for uniform $x \in \mathbb{F}$, and some integers a, b, c, d with absolute value bounded by $\mathsf{poly}(\lambda)$. It then follows from Bowe et al. [BGM17] that the required SRS can be derived in a universal and updatable setup requiring only one honest participant; in the sense that an adversary controlling all but one of the participants in the setup does not gain more than a $\mathsf{negl}(\lambda)$ advantage in its probability of producing a proof of any statement.

For notational simplicity, we sometimes use the SRS srs as an implicit parameter in protocols, and do not explicitly write it.

2.2 Analysis in the AGM model

For security analysis we will use the Algebraic Group Model of Fuchsbauer, Kiltz and Loss[FKL18]. In our protocols, by an *algebraic adversary* $\mathcal{A}$ in an SRS-based protocol we mean a $\mathsf{poly}(\lambda)$-time algorithm which satisfies the following.

- For $i \in \{1, 2\}$, whenever $\mathcal{A}$ outputs an element $A \in \mathbb{G}_i$, it also outputs a vector v over $\mathbb{F}$ such that $A = < v, \mathsf{srs_i} >$.

Idealized verifier checks for algebraic adversaries We introduce some terminology to capture the advantage of analysis in the AGM.

First we say our srs *has degree* Q if all elements of $\mathsf{srs_i}$ are of the form $[f(x)]_i$ for $f \in \mathbb{F}_{<Q}[X]$ and uniform $x \in \mathbb{F}$. In the following discussion let us assume we are executing a protocol with a degree Q SRS, and denote by $f_{i,j}$ the corresponding polynomial for the j'th element of $\mathsf{srs_i}$.

Denote by a, b the vectors of $\mathbb{F}$-elements whose encodings in $\mathbb{G}_1, \mathbb{G}_2$ an algebraic adversary $\mathcal{A}$ outputs during a protocol execution; e.g., the j'th $\mathbb{G}_1$ element output by $\mathcal{A}$ is $[a_j]_1$.

By a "real pairing check" we mean a check of the form

$$(a \cdot T_1) \cdot (T_2 \cdot b) = 0$$

for some matrices T_1, T_2 over $\mathbb{F}$. Note that such a check can indeed be done efficiently given the encoded elements and the pairing function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$.

Given such a "real pairing check", and the adversary $\mathcal{A}$ and protocol execution during which the elements were output, define the corresponding "ideal check" as follows. Since $\mathcal{A}$ is algebraic when he outputs $[a_j]_i$ he also outputs a vector v such that, from linearity, $a_j = \sum v_\ell f_{i,\ell}(x) = R_{i,j}(x)$ for $R_{i,j}(X) := \sum v_\ell f_{i,\ell}(X)$. Denote, for $i \in \{1, 2\}$ the vector of polynomials $R_i = (R_{i,j})_j$. The corresponding ideal check, checks as a polynomial identity whether

$$(R_1 \cdot T_1) \cdot (T_2 \cdot R_2) \equiv 0$$

The following lemma is inspired by [FKL18]'s analysis of [Gro16], and tells us that for soundness analysis against algebraic adversaries it suffices to look at ideal checks. Before stating the lemma we define the Q-DLOG assumption similarly to [FKL18].

Definition 2.1. *Fix integer Q. The Q-DLOG* assumption for $(\mathbb{G}_1, \mathbb{G}_2)$ *states that given*

$$[1]_1, [x]_1, \ldots, \left[x^Q\right]_1, [1]_2, [x]_2, \ldots, \left[x^Q\right]_2$$

for uniformly chosen $x \in \mathbb{F}$, the probability of an efficient $\mathcal{A}$ outputting x is $\mathsf{negl}(\lambda)$.

Lemma 2.2. *Assume the Q-DLOG for $(\mathbb{G}_1, \mathbb{G}_2)$. Given an algebraic adversary $\mathcal{A}$ participating in a protocol with a degree Q SRS, the probability of any real pairing check passing is larger by at most an additive $\mathsf{negl}(\lambda)$ factor than the probability the corresponding ideal check holds.*

Proof. Let γ be the difference between the satisfiability of the real and ideal check. We describe an adversary $\mathcal{A}^*$ for the Q-DLOG problem that succeeds with probability γ; this implies $\gamma = \mathsf{negl}(\lambda)$. $\mathcal{A}^*$ receives the challenge

$$[1]_1, [x]_1, \ldots, \left[x^Q\right]_1, [1]_2, [x]_2, \ldots, \left[x^Q\right]_2$$

and constructs using group operations the correct SRS for the protocol. Now $\mathcal{A}^*$ runs the protocol with $\mathcal{A}$, simulating the verifier role. Note that as $\mathcal{A}^*$ receives from $\mathcal{A}$ the vectors of coefficients v, he can compute the polynomials $\{R_{i,j}\}$ and check if we are in the case that the real check passed but ideal check failed. In case we are in this event, $\mathcal{A}^*$ computes

$$R := (R_1 \cdot T_1)(T_2 \cdot R_2).$$

We have that $R \in \mathbb{F}_{<2Q}[X]$ is a non-zero polynomial for which $R(x) = 0$. Thus $\mathcal{A}^*$ can factor R and find x. □

Knowlege soundness in the Algebraic Group Model We say a protocol $\mathscr{P}$ between a prover $\mathbf{P}$ and verifier $\mathbf{V}$ for a relation $\mathcal{R}$ has *Knowledge Soundness in the Algebraic Group Model* if there exists an efficient E such that the probability of any algebraic adversary $\mathcal{A}$ winning the following game is $\mathsf{negl}(\lambda)$.

1. $\mathcal{A}$ chooses input x and plays the role of $\mathbf{P}$ in $\mathscr{P}$ with input x.

2. E given access to all of $\mathcal{A}$'s messages during the protocol (including the coefficients of the linear combinations) outputs ω.

3. $\mathcal{A}$ wins if

   (a) $\mathbf{V}$ outputs acc at the end of the protocol, and

   (b) $(\mathsf{x}, \omega) \notin \mathcal{R}$.

3 A batched version of the [KZG10] scheme

Crucial to the efficiency of our protocol is a batched version of the [KZG10] polynomial commitment scheme similar to Appendix C of [MBKM19], allowing to query multiple committed polynomials at multiple points. We begin by defining polynomial commitment schemes in a manner conducive to our protocol. Specifically, we defining the open procedure in a batched setting allowing multiple polynomials and evaluation points.

Definition 3.1. *A d-polynomial commitment scheme consists of*

- $\mathsf{gen}(d)$ - *a randomized algorithm that outputs an SRS* srs.
- $\mathsf{com}(f, \mathsf{srs})$ - *that given a polynomial* $f \in \mathbb{F}_{<d}[X]$ *returns a commitment* cm *to* f.
- *A public coin protocol* open *between parties* P_{PC} *and* V_{PC}. P_{PC} *is given* $f_1, \ldots, f_t \in \mathbb{F}_{<d}[X]$. P_{PC} *and* V_{PC} *are both given integer* $t = \mathsf{poly}(\lambda)$, $\mathsf{cm}_1, \ldots, \mathsf{cm}_t$ *- the alleged commitments to* $f_1, \ldots, f_t$, $z_1, \ldots, z_t \in \mathbb{F}$ *and* $s_1, \ldots, s_t \in \mathbb{F}$ *- the alleged correct openings* $f_1(z_1), \ldots, f_t(z_t)$. *At the end of the protocol* V_{PC} *outputs* acc *or* rej.

such that

- ***Completeness:*** *Fix integer* t, $z_1, \ldots, z_t \in \mathbb{F}$, $f_1, \ldots, f_t \in \mathbb{F}_{<d}[X]$. *Suppose that for each* $i \in [t]$, $\mathsf{cm}_i = \mathsf{com}(f_i, \mathsf{srs})$. *Then if* open *is run correctly with values* $t, \{\mathsf{cm}_i, z_i, s_i = f_i(z)\}_{i \in [t]}$, V_{PC} *outputs* acc *with probability one.*
- ***Knowledge soundness in the algebraic group model:*** *There exists an efficient E such that for any algebraic adversary* $\mathcal{A}$ *the probability of* $\mathcal{A}$ *winning the following game is* $\mathsf{negl}(\lambda)$ *over the randomness of* $\mathcal{A}$ *and* gen.
  1. *Given* srs, $\mathcal{A}$ *outputs* $t, \mathsf{cm}_1, \ldots, \mathsf{cm}_t$.
  2. *E, given access to the messages of* $\mathcal{A}$ *during the previous step, outputs* $f_1, \ldots, f_t \in \mathbb{F}_{<d}[X]$.
  3. $\mathcal{A}$ *outputs* $z_1, \ldots, z_t \in \mathbb{F}$, $s_1, \ldots, s_t \in \mathbb{F}$.
  4. $\mathcal{A}$ *takes the part of* P_{PC} *in the protocol* open *with inputs* $\mathsf{cm}_1, \ldots, \mathsf{cm}_t, z_1, \ldots, z_t, s_1, \ldots, s_t$.
  5. $\mathcal{A}$ *wins if*
     - V_{PC} *outputs* acc *at the end of the protocol.*
     - *For some* $i \in [t]$, $s_i \neq f_i(z)$.

We describe the following scheme based on [KZG10, MBKM19].

1. $\mathsf{gen}(d)$ - choose uniform $x \in \mathbb{F}$. Output $\mathsf{srs} = ([1]_1, [x]_1, \ldots, [x^{d-1}]_1, [1]_2, [x]_2)$.
2. $\mathsf{com}(f, \mathsf{srs}) := [f(x)]_1$.
3. We first describe the open protocol in the case $z_1 = \ldots = z_t = z$.
   $\mathsf{open}(\{cm_i\}, \{z_i\}, \{s_i\})$:

(a) V_{PC} sends random $\gamma \in \mathbb{F}$.

(b) P_{PC} computes the polynomial

$$h(X) := \sum_{i=1}^{t} \gamma^i \cdot \frac{f_i(X) - f_i(z)}{X - z}$$

and using srs computes and sends $W := [h(x)]_1$.

(c) V_{PC} computes the elements

$$F := \sum_{i \in [t]} \gamma^i \cdot \mathsf{cm}_i, v := \left[\sum_{i \in [t]} \gamma^i \cdot s_i \right]_1$$

(d) V_{PC} outputs acc if and only if

$$e(F - v, [1]_2) \cdot e(-W, [x - z]_2) = 1.$$

We argue knowledge soundness for the above protocol. More precisely, we argue the existence of an efficient E such that an algebraic adversary $\mathcal{A}$ can only win the KS game w.p. $\mathsf{negl}(\lambda)$ when restricting itself to choosing $z = z_1 = \ldots = z_t$.

Let $\mathcal{A}$ be such an algebraic adversary.

$\mathcal{A}$ begins by outputting $\mathsf{cm}_1, \ldots, \mathsf{cm}_t$. Each cm_i is a linear combination $\sum_{j=0}^{d-1} a_{i,j} \left[x^j\right]_1$. E, who is given the coefficients $\{a_{i,j}\}$, simply outputs the polynomials

$$f_i(X) := \sum_{j=0}^{d-1} a_{i,j} \cdot X^j.$$

$\mathcal{A}$ now outputs $z, s_1, \ldots, s_t \in \mathbb{F}$. Assume that for some $i \in [t]$, $f_i(z) \neq s_i$. We show that for any strategy of $\mathcal{A}$ from this point, $\mathrm{V}_{\mathsf{poly}}$ outputs acc w.p $\mathsf{negl}(\lambda)$.

In the first step of open, $\mathrm{V}_{\mathsf{poly}}$ chooses a random $\gamma \in \mathbb{F}$. Define

$$f(X) := \sum_{i \in [t]} \gamma^i \cdot f_i(X), s := \sum_{i \in [t]} \gamma^i \cdot s_i.$$

We have that e.w.p. $t/|\mathbb{F}|$, $f(z) \neq s$. Now $\mathcal{A}$ outputs $W = H(x)$ for some $H \in \mathbb{F}_{<d}[X]$. According to Lemma 2.2, it suffices to upper bound the probability that the ideal check corresponding to the real pairing check in the protocol passes. It has the form

$$f(X) - s \equiv H(X)(X - z).$$

The check passing implies that $f(X) - s$ is divisible by $(X - z)$, which implies $f(z) = s$. Thus the ideal check can only pass w.p. $\mathsf{negl}(\lambda)$ over the randomness of $\mathrm{V}_{\mathsf{poly}}$, which implies the same thing for the real check according to Lemma 2.2.

The open protocol for multiple evaluation points simply consists of running in parallel the open protocol for each evaluation point and the polynomials evaluated at that point. And then applying a generic method for batch randomized evaluation of pairing equations. For notational simplicity we describe the open protocol explicitly only in the case of two distinct evaluation points among $z_1, \ldots, z_t$ (this also happens to be our case in the main protocol). For this, let us denote the distinct evaluation points by z, z' and by t, t' the number of polynomials and by f_i, f'_i the i'th polynomial to be evaluated at z, z' respectively.

$\underline{\mathsf{open}(\{cm_i, cm'_i\}, \{z, z'\}, \{s_i, s'_i\})}$:

(a) V_{PC} sends random $\gamma\,\gamma', \in \mathbb{F}$.

(b) P_{PC} computes the polynomials

$$h(X) := \sum_{i=1}^{t} \gamma^i \cdot \frac{f_i(X) - f_i(z)}{X - z}$$

$$h'(X) := \sum_{i=1}^{t'} \gamma'^i \cdot \frac{f'_i(X) - f'_i(z')}{X - z'}$$

and using srs computes and sends $W := [h(x)]_1, W' := [h'(x)]_1$.

(c) V_{PC} chooses random $r, r' \in \mathbb{F}$.

(d) $\mathrm{V}_{\mathsf{poly}}$ computes the element

$$F := r \cdot \left(\sum_{i \in [t]} \gamma^i \cdot \mathsf{cm}_i - \left[\sum_{i \in [t]} \gamma^i \cdot s_i \right]_1 \right) + r' \cdot \left(\sum_{i \in [t']} \gamma'^i \cdot \mathsf{cm}'_i - \left[\sum_{i \in [t']} \gamma'^i \cdot s'_i \right]_1 \right)$$

V_{PC} computes outputs acc if and only if

$$e\left(F + rz \cdot W + r'z' \cdot W', [1]_2\right) \cdot e(-r \cdot W - r' \cdot W', [x]_2) = 1.$$

We summarize the efficiency properties of this batched version of the [KZG10] scheme.

Lemma 3.2. *Fix positive integer d. There is a d-polynomial commitment scheme $\mathscr{S}$ such that*

(a) For $n \leq d$ and $f \in \mathbb{F}_{<n}[X]$, computing $\mathsf{com}(f)$ requires n $\mathbb{G}_1$-exponentiations.

(b) Given $\mathbf{z} := (z_1, \ldots, z_t) \in \mathbb{F}^t, f_1, \ldots, f_t \in \mathbb{F}_{<d}[X]$, denote by t^* the number of distinct values in $\mathbf{z}$; and for $i \in [t^*]$, $d_i := 1 + \max\{\deg(f_i)\}_{i \in S_i}$ where S_i is the set of indices j such that z_j equals the i'th distinct point in $\mathbf{z}$. Let $\mathsf{cm}_i = \mathsf{com}(f_i)$. Then open $(\{cm_i, f_i, z_i, s_i\})$ requires

i. $\sum_{i \in [t^*]} d_i$ $\mathbb{G}_1$-exponentiations of $\mathrm{P}_{\mathsf{poly}}$.

ii. $t + t^*$ $\mathbb{G}_1$-exponentiations and 2 pairings of $\mathrm{V}_{\mathsf{poly}}$.

4 Idealised low-degree protocols

We define a limited type of protocol between a prover and a verifier to cleanly capture and abstract the use of a polynomial commitment scheme such as [KZG10]. In this protocol, the prover sends low-degree polynomials to a third trusted party $\mathcal{I}$. The verifier may then ask $\mathcal{I}$ whether certain identites hold between the prover's polynomials, and additional predefined polynomials known to the verifier.

Definition 4.1. *Fix a positive integers d, D, t, ℓ. A (d, D, t, ℓ)-*polynomial protocol *is a multiround protocol between a prover $\mathrm{P_{poly}}$, verifier $\mathrm{V_{poly}}$ and trusted party $\mathcal{I}$ that proceeds as follows.*

1. *The protocol definition includes a set of* preprocessed polynomials $g_1, \ldots, g_\ell \in \mathbb{F}_{<d}[X]$.
2. *The messages of $\mathrm{P_{poly}}$ are sent to $\mathcal{I}$ and are of the form f for $f \in \mathbb{F}_{<d}[X]$. If $\mathrm{P_{poly}}$ sends a message not of this form, the protocol is aborted.*
3. *The messages of $\mathrm{V_{poly}}$ to $\mathrm{P_{poly}}$ are arbitrary (but we will concentrate on public coin protocols where the messages are simply random coins).*
4. *At the end of the protocol, suppose $f_1, \ldots, f_t$ are the polynomials that were sent from $\mathrm{P_{poly}}$ to $\mathcal{I}$. $\mathrm{V_{poly}}$ may ask $\mathcal{I}$ if certain polynomial identities holds between $\{f_1, \ldots, f_t, g_1, \ldots, g_\ell\}$. Where each identity is of the form*
$$F(X) := G(h_1(v_1(X)), \ldots, h_M(v_M(X))) \equiv 0,$$
for some $h_i \in \{f_1, \ldots, f_t, g_1, \ldots, g_\ell\}$, $G \in \mathbb{F}[X_1, \ldots, X_M]$, $v_1, \ldots, v_M \in \mathbb{F}_{<d}[X]$ such that $F \in \mathbb{F}_{<D}[X]$ for every choice of $f_1, \ldots, f_t$ made by $\mathrm{P_{poly}}$ when following the protocol correctly.
5. *After receiving the answers from $\mathcal{I}$ regarding the identities, $\mathrm{V_{poly}}$ outputs* acc *if all identities hold, and outputs* rej *otherwise.*

Remark 4.2. *A more expressive model would be to have $\mathrm{P_{poly}}$ send messages (f, n) for $n \leq d$ to $\mathcal{I}$ instead of just f; and have $\mathcal{I}$ enforce $f \in \mathbb{F}_{<n}[X]$. We avoid doing this as this extra power is not needed for our protocol, and results in reduced efficiency as it translates to needing to use a polynomial commitment scheme with the ability to dynamically enforce a smaller than d degree bound (as the [MBKM19]-variant of [KZG10] is able to do).*

We define polynomial protocols for relations in the natural way.

Definition 4.3. *Given a relation $\mathcal{R}$, a* polynomial protocol for $\mathcal{R}$ *is a polynomial protocol with the following additional properties.*

1. *At the beginning of the protocol, $\mathrm{P_{poly}}$ and $\mathrm{V_{poly}}$ are both additionally given an input x. The description of $\mathrm{P_{poly}}$ assumes possession of ω such that $(\mathsf{x}, \omega) \in \mathcal{R}$.*

2. ***Completeness:*** *If* $\mathrm{P}_{\mathsf{poly}}$ *follows the protocol correctly using a witness* ω *for* x*,* $\mathrm{V}_{\mathsf{poly}}$ *accepts with probability one.*

3. ***Knowledge Soundness:*** *There exists an efficient* E*, that given access to the messages of* $\mathrm{P}_{\mathsf{poly}}$ *to* $\mathcal{I}$ *outputs* ω *such that, for any strategy of* $\mathrm{P}_{\mathsf{poly}}$*, the probability of the following event is* $\mathsf{negl}(\lambda)$*.*

   (a) $\mathrm{V}_{\mathsf{poly}}$ *outputs* acc *at the end of the protocol, and*

   (b) $(\mathsf{x}, \omega) \notin \mathcal{R}$*.*

Remark 4.4. *We intentionally do not define a zero-knowlege property for idealized protocols, as acheiving ZK will depend on how much information on the polynomials sent to* $\mathcal{I}$ *is leaked in the final "compiled" protocol. This in turn depends on specific details of the polynomial commitment scheme used for compilation.*

4.1 Polynomial protocols on ranges

In our protocol $\mathrm{V}_{\mathsf{poly}}$ actually needs to check if certain polynomial equations hold on a certain range of input values, rather than as a polynomial identity. Motivated by this, for a subset $S \subset \mathbb{F}$, we define an *S-ranged (d, D, t, ℓ)-polynomial protocol* identically to a (d, D, t, ℓ)-polynomial protocol, except that the verifier asks if his identities hold on all points of S, rather than identically. We then define ranged polynomial protocols for relations in the exact same way as in Definition 4.3.

We show that converting a ranged protocol to a polynomial protocol only incurs one additional prover polynomial.

Lemma 4.5. *Let* $\mathscr{P}$ *be an* S*-ranged* (d, D, t, ℓ)*-polynomial protocol for* $\mathcal{R}$*. Then we can construct a* $(\max\{d, |S|, D - |S|\}, D, t+1, \ell+1)$*-polynomial protocol* $\mathscr{P}^*$ *for* $\mathcal{R}$*.*

For the lemma, we use the following simple claim.

Claim 4.6. *Fix* $F_1, \ldots, F_k \in \mathbb{F}_{<n}[X]$*. Fix* $Z \in \mathbb{F}_{<n}[X]$*. Suppose that for some* $i \in [k]$*,* $Z \nmid F_i$*. Then e.w.p* $1/|\mathbb{F}|$ *over uniform* $a_1, \ldots, a_k \in \mathbb{F}$*,* Z *doesn't divide*

$$F := \sum_{j=1}^{k} a_j \cdot F_j.$$

Proof. $Z|F$ is equivalent to $F \bmod Z = 0$. Denoting $R := F_i \bmod Z$, we have that $R \neq 0$; i.e. R isn't the zero polynomial. And we have

$$F = \sum_{j=1, j\neq i}^{k} a_j \cdot F_j + a_i \cdot R \pmod{Z}$$

Thus, for any fixing of $\{a_j\}_{j\neq i}$ there is at most one value $a_i \in \mathbb{F}$ such that $F \bmod Z = 0$. The claim follows. □

Proof. (Of Lemma 4.5) Let $\mathscr{P}$ be the S-ranged (d, D, t, ℓ)-polynomial protocol. We construct the protocol $\mathscr{P}^*$. The set of preprocessed polynomials in $\mathscr{P}^*$ are the same as in $\mathscr{P}$ with the addition of $Z_S(X) := \prod_{a\in S}(X - a)$. $\mathscr{P}^*$ proceeds exactly as $\mathscr{P}$ until the point where $\mathrm{V}_{\mathsf{poly}}$ asks about identites on S. Suppose that the k identities the verifier asks about are $F_1(X), \ldots, F_k(X)$ (where each F_i is of total degree at most D and of the form described in Definition 4.1). $\mathscr{P}^*$ now proceeds as follows:

- $\mathrm{V}_{\mathsf{poly}}$ sends uniform $a_1, \ldots, a_k \in \mathbb{F}$ to $\mathrm{P}_{\mathsf{poly}}$.
- $\mathrm{P}_{\mathsf{poly}}$ computes the polynomial $T := \frac{\sum_{i\in[k]} a_i \cdot F_i}{Z_S}$.
- $\mathrm{P}_{\mathsf{poly}}$ sends T to $\mathcal{I}$.
- $\mathrm{V}_{\mathsf{poly}}$ queries the identity

$$\sum_{i\in[k]} a_i \cdot F_i(X) \equiv T \cdot Z_S$$

It follows from Claim 4.6 that e.w.p. $1/|\mathbb{F}|$ over $\mathrm{V}_{\mathsf{poly}}$'s choice of $a_1, \ldots, a_k$, the existence of an appropriate $T \in \mathbb{F}[X]$ is equivalent to $F_1, \ldots, F_\ell$ vanishing on S. This in turn is equivalent to $\mathrm{V}_{\mathsf{poly}}$ outputting acc in the analogous execution of $\mathscr{P}$. □

4.2 From polynomial protocols to protocols against algebraic adversaries

We wish to use the polynomial commitment scheme of Section 3 to compile a polynomial protocol into one with knowledge soundness in the algebraic group model (in the sense defined in Section 2.2).

For the purpose of capturing the efficiency of the transformation, we first define somewhat technical measures of the (d, D, t, ℓ)-polynomial protocol $\mathscr{P}$.

For $i \in [t]$, let d_i be one plus the maximal degree of f_i sent by an honest prover in $\mathscr{P}$. Assume only one identity is checked by $\mathrm{V}_{\mathsf{poly}}$ in $\mathscr{P}$. And let $v_1, \ldots, v_M$ be the "inner" polynomials used in this identity.

For $i \in [M]$, let d'_i be the "matching" d_j. That is $d'_i = d_j$ if $h_i = f_j$, and $d'_i = d_{t+j}$ if $h_i = g_j$.

Let $t^* = t^*(\mathscr{P})$ be the number of distinct polynomials amongst $v_1, \ldots, v_M$. Let $S_1 \cup \ldots \cup S_{t^*} = [M]$ be a partition of $[M]$ according to the distinct values. For $j \in [t^*]$, let $e_j := \max\{d'_i\}_{i\in S_j}$

Finally, define $\mathsf{e}(\mathscr{P}) := \sum_{i\in[t]} d_i + \sum_{j\in[t^*]} e_j$.

Lemma 4.7. *Let $\mathscr{P}$ be a public coin (d, D, t, ℓ)-polynomial protocol for a relation $\mathcal{R}$ where only one identity is checked by $\mathrm{V}_{\mathsf{poly}}$. Then we can construct a protocol $\mathscr{P}^*$ for $\mathcal{R}$ with knowledge soundness in the Algebraic Group Model under $2d$-DLOG such that*

1. *The prover $\mathbf{P}$ in $\mathscr{P}^*$ requires $\mathsf{e}(\mathscr{P})$ $\mathbb{G}_1$-exponentiations.*
2. *The total prover communication consists of $t + t^*(\mathscr{P})$ $\mathbb{G}_1$ elements and M $\mathbb{F}$-elements.*

Proof. Let $\mathscr{S} = (\mathsf{gen}, \mathsf{com}, \mathsf{open})$ be the d-polynomial commitment scheme described in Lemma 3.2. The SRS of $\mathscr{P}^*$ includes $\mathsf{srs} = \mathsf{gen}(d)$, with the addition of $\{\mathsf{com}(g_1), \ldots, \mathsf{com}(g_\ell)\}$.

Given $\mathscr{P}$ we describe $\mathscr{P}^*$. **P** and **V** behave identically to $\mathrm{P}_{\mathsf{poly}}$ and $\mathrm{V}_{\mathsf{poly}}$, except in the following two cases.

- Whenever $\mathrm{P}_{\mathsf{poly}}$ sends a polynomial $f_i \in \mathbb{F}_{<d}[X]$ to $\mathcal{I}$ in $\mathscr{P}$, **P** sends $\mathsf{cm}_i = \mathsf{com}(f_i)$ to **V**.
- Let $v_1^*, \ldots, v_{t^*}^*$ be the distinct polynomials amongst $v_1, \ldots, v_M$. When $\mathrm{V}_{\mathsf{poly}}$ asks about the identity

$$F(X) := G(h_1(v_1(X)), \ldots, h_M(v_M(X))) \equiv 0,$$

  1. **V** chooses random $x \in \mathbb{F}$, computes $v_1^*(x), \ldots, v_{t^*}^*(x)$, and sends x to **P**.
  2. **P** replies with $\{s_i\}_{i\in[M]}$, which are the alleged values $h_1(v_1(x)), \ldots, h_M(v_M(x))$.
  3. **V** engages in the protocol open with **P** to verify the correctness of $\{s_i\}$
  4. **V** outputs acc if and only if

$$G(s_1, \ldots, s_M) = 0.$$

The efficiency claims about $\mathscr{P}^*$ follow directly from Lemma 3.2.

To prove the claim about knowledge soundness in the AGM we must describe the extractor E for the protocol $\mathscr{P}^*$. For this purpose, let $E_{\mathscr{P}}$ be the extractor of the protocol $\mathscr{P}$ as guaranteed to exist from Definition 4.3, and $E_{\mathscr{S}}$ be the extractor for the Knowledge Soundness game of $\mathscr{S}$ as in Definition 3.1.

Now assume an algebraic adversary $\mathcal{A}$ is taking the role of **P** in $\mathscr{P}^*$.

1. E sends the commitments $\mathsf{cm}_1, \ldots, \mathsf{cm}_t$ to $E_{\mathscr{S}}$ and receives in return $f_1, \ldots, f_t \in \mathbb{F}_{<d}[X]$.
2. E plays the role of $\mathcal{I}$ in interaction with $E_{\mathscr{P}}$, sending him the polynomials $f_1, \ldots, f_t$.
3. When $E_{\mathscr{P}}$ outputs ω, E also outputs ω.

Now let us define two events (over the randomness of **V**,$\mathcal{A}$ and gen):

1. We think of an adversary $\mathcal{A}_{\mathscr{P}}$ participating in $\mathscr{P}$, and using the polynomials $f_1, \ldots, f_t$ as their messages to $\mathcal{I}$. We define A to be the event that the identity F held, but $(\mathsf{x}, \omega) \notin \mathcal{R}$. By the KS of $\mathscr{P}$, $\Pr(A) = \mathsf{negl}(\lambda)$.
2. We let B be the event that for some $i \in [M], h_i(v_i(x)) \neq s_i$, and at the same time V_{PC} has output acc when open was run as a subroutine in Step 3. By the KS of $\mathscr{S}$, $\Pr(B) = \mathsf{negl}(\lambda)$.

Now look at the event C that **V** outputs acc, but E failed in the sense that $(\mathsf{x}, \omega) \notin \mathcal{R}$. We split C into two events.

1. A or B also happened - this has $\mathsf{negl}(\lambda)$ probability.
2. C happened but not A or B. This means F is not the zero polynomial, but $F(x) = 0$; which happens w.p. $\mathsf{negl}(\lambda)$.

□

Reducing the number of field elements   We describe an optimization by Mary Maller, to reduce the number of $\mathbb{F}$-elements in the proof from M. To describe the general method, we must define another technical measure of a polynomial protocol. We assume again (mainly for simplicity) that $\mathsf{V_{poly}}$ checks only one identity F. Now define $\mathsf{r}(\mathscr{P})$ to be the minimal size of a subset $S \subset [M]$ such that

- $([M] \setminus S) \subset S_i$ for one of the subsets S_i of the partition described before Lemma 4.7.
- The polynomial G such that

$$F(X) := G(h_1(v_1(X)), \ldots, h_M(v_M(X)))$$

  is has degree zero or one as a polynomial in the variables $\{X_j\}_{j \in [M] \setminus S}$ whose coefficients are polynomials in $\{X_j\}_{j \in S}$.

Assume $\mathscr{P}$ is such that $\mathsf{r} := \mathsf{r}(\mathscr{P}) < M$

We claim that the reduction of Lemma 4.7 can be changed such that only r $\mathbb{F}$-elements are sent by $\mathbf{P}$.

1. $\mathbf{P}$ now sends only $\{s_i = h_i(v_i(x))\}_{i \in S}$.
2. Now let L be the restriction $G|_{X_i = s_i, i \in S}$. $\mathbf{V}$ and $\mathbf{P}$ use $\{\mathsf{com}(f_i\}_{i \in [S]}$, and the linearity of com, to compute the commitment to the corresponding restriction F_L of F.
3. $\mathbf{V}$ computes the unique value $s = F_L(v_i^*(x))$, that will imply $F(x) = 0$, assuming correctness of $\{s_i\}_{i \in S}$.
4. Now $\mathbf{P}$ and $\mathbf{V}$ engage in the protocol open to verify the correctness of $\{s_i\}_{i \in S} \cup \{s\}$.

5 Polynomial protocols for identifying permutations

At the heart our universal SNARK is a "permutation check" inspired by the permutation argument originally presented by Bayer and Groth [BG12] and its variants in [BCC+16, MBKM19]. Again, our main advantage over [MBKM19] is getting a simpler protocol by working with *univariate* polynomials and multiplicative subgroups.

Degree bounds: We use two integer parameters $n \leq d$. Intuitively, n is the degree of the honest prover's polynomials, and d is the bound we actually enforce on malicious provers. Accordingly, we assume degree bound n while analyzing prover efficiency and describing "official" protocol inputs; but allow degree bound d while analyzing soundness.

We assume the existence of a multiplicative subgroup $H \subset \mathbb{F}$ of order $n+1$ with generator $\mathbf{g}$, and denote by $H^\star$ the subset of H of the form $\{\mathbf{g}, \mathbf{g}^2, \ldots, \mathbf{g}^n\}$.

For $i \in [n+1]$, we denote by $L_i(X)$ the element of $\mathbb{F}_{<n+1}[X]$ with $L_i(\mathbf{g}^i) = 1$ and $L_i(a) = 0$ for $a \in H$ different from $\mathbf{g}^i$, i.e. $\{L_i\}_{i \in [n+1]}$ is a Lagrange basis for H.[4]

One thing to note is that the $\{L_i\}$ can "reduce point checks to range checks". More precisely, the following claim follows directly from the definition of $\{L_i\}$.

Claim 5.1. *Fix $i \in [n]$, and $Z, Z^* \in \mathbb{F}[X]$. Then $L_i(a)(Z(a) - Z^*(a)) = 0$ for each $a \in H^\star$ if and only if $Z(\mathbf{g}^i) = Z^*(\mathbf{g}^i)$.*

For $f, g \in \mathbb{F}_{<d}[X]$ and a permutation $\sigma : [n] \to [n]$, we write $g = \sigma(f)$ if for each $a \in H^\star$, $g(\mathbf{g}^i) = f(\mathbf{g}^{\sigma(i)})$. [5]

We present a ranged polynomial protocol enabling $\mathrm{P}_{\mathsf{poly}}$ to prove that $g = \sigma(f)$.

Preprocessed polynomials: The polynomial $\mathsf{S}_{\mathsf{ID}} \in \mathbb{F}_{<n}[X]$ defined by $\mathsf{S}_{\mathsf{ID}}(\mathbf{g}^i) = i$ for each $i \in [n]$ and $\mathsf{S}_\sigma \in \mathbb{F}_{<n}[X]$ defined by $\mathsf{S}_\sigma(\mathbf{g}^i) = \sigma(i)$ for each $i \in [n]$.

Inputs: $f, g \in \mathbb{F}_{<n}[X]$

Protocol:

1. $\mathrm{V}_{\mathsf{poly}}$ chooses random $\beta, \gamma \in \mathbb{F}$ and sends them to $\mathrm{P}_{\mathsf{poly}}$.
2. Let $f' := f + \beta \cdot \mathsf{S}_{\mathsf{ID}} + \gamma, g' := g + \beta \cdot \mathsf{S}_\sigma + \gamma$. That is, for $i \in [n]$
$$f'(\mathbf{g}^i) = f(\mathbf{g}^i) + \beta \cdot \sigma(i) + \gamma, g'(\mathbf{g}^i) = g(\mathbf{g}^i) + \beta \cdot \sigma(i) + \gamma$$
3. $\mathrm{P}_{\mathsf{poly}}$ computes $Z, Z^* \in \mathbb{F}_{<n+1}[X]$, such that $Z(\mathbf{g}) = Z^*(\mathbf{g}) = 1$; and for $i \in \{2, \ldots, n+1\}$
$$Z(\mathbf{g}^i) = \prod_{1 \le j < i} f'(\mathbf{g}^j), Z^*(\mathbf{g}^i) = \prod_{1 \le j < i} g'(\mathbf{g}^j)$$
4. $\mathrm{P}_{\mathsf{poly}}$ sends $\mathsf{com}(Z, n+1), \mathsf{com}(Z^*, n+1)$ to $\mathrm{V}_{\mathsf{poly}}$.
5. $\mathrm{V}_{\mathsf{poly}}$ checks if for all $a \in H^\star$
   (a) $L_1(a)Z(a) = L_1(a)Z^*(a)$
   (b) $Z(a)f'(a) = Z(a \cdot \mathbf{g})$.
   (c) $Z^*(a)g'(a) = Z^*(a \cdot \mathbf{g})$
   (d) $L_n(a)Z(a \cdot \mathbf{g}) = L_n(a)Z^*(a \cdot \mathbf{g})$.

   and outputs acc iff all checks hold.

[4]It would be more natural to work with a basis for $H^\star$, but it is more efficient to work with a basis for H, and Claim 5.1 implies this is also fine.

[5]Note that according to this definition there are multiple g with $g = \sigma(f)$. Intuitively, we think of $\sigma(f)$ as the unique such $g \in \mathbb{F}_{<n}[X]$, but do not define this formally to avoid needing to enforce this degree bound for efficiency reasons.

Lemma 5.2. *Fix $f, g \in \mathbb{F}_{<d}[X]$. For any strategy of* $\mathrm{P}_{\mathsf{poly}}$*, the probability of* $\mathrm{V}_{\mathsf{poly}}$ *outputting* acc *in the above protocol when $g \neq \sigma(f)$ is* $\mathsf{negl}(\lambda)$*.*

Proof. Suppose that $g \neq \sigma(f)$. By claim A.1, e.w.p $\mathsf{negl}(\lambda)$ over the choice of $\beta, \gamma \in \mathbb{F}$,

$$a := \prod_{i \in [n]} f'(\mathbf{g}^i) \neq b := \prod_{i \in [n]} g'(\mathbf{g}^i).$$

Assume such β, γ were chosen. We show $\mathrm{V}_{\mathsf{poly}}$ rejects; specifically, that if the first three identites $\mathrm{V}_{\mathsf{poly}}$ checks are satisifed, the fourth must not hold.

From the first check we know that $Z(\mathbf{g}) = Z^*(\mathbf{g}) = c$ for some $c \in \mathbb{F}$. From the second and third checks we can show, inductively, that for each $i \in \{2, \ldots, n+1\}$

$$Z(\mathbf{g}^i) = c \cdot \prod_{1 \leq j < i} f'(\mathbf{g}^j), Z^*(\mathbf{g}^i) = c \cdot \prod_{1 \leq j < i} g'(\mathbf{g}^j).$$

In particular,

$$c \cdot a = Z(\mathbf{g}^{n+1}) \neq Z^*(\mathbf{g}^{n+1}) = c \cdot b.$$

Hence the fourth check, being equivalent to $Z(\mathbf{g}^{n+1}) \stackrel{?}{=} Z^*(\mathbf{g}^{n+1})$, must fail. □

5.1 Checking "extended" permutations

In our protocol, we in fact need to check a permutation "across" the values of several polynomials. Let us define this setting formally. Suppose we now have mutliple polynomials $f_1, \ldots, f_k \in \mathbb{F}_{<d}[X]$ and a permutation $\sigma : [kn] \to [kn]$. For $(g_1, \ldots, g_k) \in (\mathbb{F}_{<d}[X])^k$, we say that $(g_1, \ldots, g_k) = \sigma(f_1, \ldots, f_k)$ if the following holds.

Define the sequences $(f_{(1)}, \ldots, f_{(kn)}), (g_{(1)}, \ldots, g_{(kn)}) \in \mathbb{F}^{kn}$ by

$$f_{((j-1)\cdot n+i)} := f_j(\mathbf{g}^i), g_{((j-1)\cdot n+i)} := g_j(\mathbf{g}^i),$$

for each $j \in [k], i \in [n]$. Then we have $g_{(\ell)} = f_{(\sigma(\ell))}$ for each $\ell \in [kn]$.

Preprocessed polynomials: The polynomials $\mathsf{S}_{\mathsf{ID}1}, \ldots, \mathsf{S}_{\mathsf{ID}k} \in \mathbb{F}_{<n}[X]$ defined by $\mathsf{S}_{\mathsf{ID}j}(\mathbf{g}^i) = (j-1) \cdot n + i$ for each $i \in [n]$.

In fact, only $\mathsf{S}_{\mathsf{ID}} = \mathsf{S}_{\mathsf{ID}1}$ is actually included in the set of preprocessed polynomials, as $\mathsf{S}_{\mathsf{ID}j}(x)$ can be computed as $\mathsf{S}_{\mathsf{ID}j}(x) = \mathsf{S}_{\mathsf{ID}}(x) + (j-1) \cdot n$.

For each $j \in [k]$ $\mathsf{S}_{\sigma j} \in \mathbb{F}_{<n}[X]$, defined by $\mathsf{S}_{\sigma j}(\mathbf{g}^i) = \sigma((j-1) \cdot n + i)$ for each $i \in [n]$.

Inputs: $f_1, \ldots, f_k, g_1, \ldots, g_k \in \mathbb{F}_{<n}[X]$

Protocol:

1. $\mathrm{V}_{\mathsf{poly}}$ chooses random $\beta, \gamma \in \mathbb{F}$ and sends to $\mathrm{P}_{\mathsf{poly}}$.

2. Let $f'_j := f_j + \beta \cdot \mathsf{S}_{\mathsf{ID}j} + \gamma$, and $g'_j := g_j + \beta \cdot \mathsf{S}_{\sigma j} + \gamma$. That is, for $j \in [k], i \in [n]$

$$f'_j(\mathbf{g}^i) = f_j(\mathbf{g}^i) + \beta \cdot \sigma((j-1) \cdot n + i) + \gamma, g'_j(\mathbf{g}^i) = g_j(\mathbf{g}^i) + \beta \cdot \sigma((j-1) \cdot n + i) + \gamma$$

3. Define $f', g' \in \mathbb{F}_{<kn}[X]$ by

$$f'(X) := \prod_{j \in [k]} f'_j(X), g'(X) := \prod_{j \in [k]} g'_j(X)$$

4. $\mathrm{P}_{\mathsf{poly}}$ computes $Z, Z^* \in \mathbb{F}_{<n+1}[X]$, such that $Z(\mathbf{g}) = Z^*(\mathbf{g}) = 1$; and for $i \in \{2, \ldots, n+1\}$

$$Z(\mathbf{g}^i) = \prod_{1 \leq \ell < i} f'(\mathbf{g}^j), Z^*(\mathbf{g}^i) = \prod_{1 \leq \ell < i} g'(\mathbf{g}^j).$$

5. $\mathrm{P}_{\mathsf{poly}}$ sends $\mathsf{com}(Z), \mathsf{com}(Z^*)$ to $\mathrm{V}_{\mathsf{poly}}$.

6. $\mathrm{V}_{\mathsf{poly}}$ checks if for all $a \in H^\star$
   (a) $L_1(a)Z(a) = L_1(a)Z^*(a)$
   (b) $Z(a)f'(a) = Z(a \cdot \mathbf{g})$
   (c) $Z^*(a)g'(a) = Z^*(a \cdot \mathbf{g})$
   (d) $L_n(a)Z(a \cdot \mathbf{g}) = L_n(a)Z^*(a \cdot \mathbf{g})$

   and outputs acc iff all checks hold.

Lemma 5.3. *Fix any $f_1, \ldots, f_k, g_1, \ldots g_k \in \mathbb{F}_{<d}[X]$ and permutation σ on $[kn]$ as inputs to the above protocol $\mathscr{P}_k$. Suppose that $(g_1, \ldots, g_k) \neq \sigma(f_1, \ldots, f_k)$. Then, for any strategy of $\mathrm{P}_{\mathsf{poly}}$, the probability of $\mathrm{V}_{\mathsf{poly}}$ outputting acc is $\mathsf{negl}(\lambda)$.*

Proof. $(g_1, \ldots, g_k) \neq \sigma(f_1, \ldots, f_k)$ implies that with high probability over $\beta, \gamma \in \mathbb{F}$ the product F of the values $\left\{f'_j(\mathbf{g}^i)\right\}_{j \in [k], i \in [n]}$ is different from the product G of the values $\left\{g'_j(\mathbf{g}^i)\right\}_{j \in [k], i \in [n]}$. Note now that

$$F = \prod_{i \in [n]} f'(\mathbf{g}^i), G = \prod_{i \in [n]} g'(\mathbf{g}^i),$$

and that the next steps of the protocol are identical to those in the previous protocol, and as analyzed there - exactly check if these products are equal. □

5.2 Checking "extended copy constraints" using a permutation

We finally come to the actual primitive that will be used in our main protocol. Let $\mathcal{T} = \{T_1, \ldots, T_s\}$ be a partition of $[kn]$ into disjoint blocks. Fix $f_1, \ldots, f_k \in \mathbb{F}_{<n}[X]$. We say that $f_1, \ldots, f_k$ *copy-satisfy* $\mathcal{T}$ if, when defining $(f_{(1)}, \ldots, f_{(kn)}) \in \mathbb{F}^{kn}$ as above, we have $f_{(\ell)} = f_{(\ell')}$ whenever ℓ, ℓ' belong to the same block of $\mathcal{T}$.

We claim that the above protocol for extended permutations can be directly used for checking whether $f_1, \ldots, f_k$ satisfy $\mathcal{T}$: Define a permutation $\sigma(\mathcal{T})$ on $[kn]$ such that for each block T_i of $\mathcal{T}$, $\sigma(\mathcal{T})$ contains a cycle going over all elements of T_i. Then, $(f_1, \ldots, f_k)$ copy-satisfy $\mathcal{T}$ if and only if $(f_1, \ldots, f_k) = \sigma(f_1, \ldots, f_k)$.

6 Constraint systems

Fix integers m, n with $n \leq m \leq 2n$ We present a type of constraint system that captures fan-in two arithmetic circuits of unlimited fan-out with n gates and m wires (but is more general).

The constraint system $\mathscr{C} = (\mathcal{V}, \mathcal{Q})$ is defined as follows.

- $\mathcal{V}$ is of the form $\mathcal{V} = (\mathbf{a}, \mathbf{b}, \mathbf{c})$, where $\mathbf{a}, \mathbf{b}, \mathbf{c} \in [m]^n$. We think of $\mathbf{a}, \mathbf{b}, \mathbf{c}$ as the left, right and output sequence of $\mathscr{C}$ respectively.
- $\mathcal{Q} = (\mathbf{q_L}, \mathbf{q_R}, \mathbf{q_O}, \mathbf{q_M}, \mathbf{q_C}) \in (\mathbb{F}^n)^5$ where we think of $\mathbf{q_L}, \mathbf{q_R}, \mathbf{q_O}, \mathbf{q_M}, \mathbf{q_C} \in \mathbb{F}^n$ as "selector vectors".

We say $\mathbf{x} \in \mathbb{F}^m$ *satisfies* $\mathscr{C}$ if for each $i \in [n]$,

$$(\mathbf{q_L})_i \cdot \mathbf{x}_{\mathbf{a}_i} + (\mathbf{q_R})_i \cdot \mathbf{x}_{\mathbf{b}_i} + (\mathbf{q_O})_i \cdot \mathbf{x}_{\mathbf{c}_i} + (\mathbf{q_M})_i \cdot (\mathbf{x}_{\mathbf{a}_i} \mathbf{x}_{\mathbf{b}_i}) + (\mathbf{q_C})_i = 0.$$

To define a relation based on $\mathscr{C}$, we extend it to include a positive integer $\ell \leq m$, and subset $\mathcal{I} \subset [m]$ of "public inputs". Assume without loss of generality that $\mathcal{I} = \{1, \ldots, \ell\}$.

Now we can define the relation $\mathcal{R}_{\mathscr{C}}$ as the set of pairs (x, ω) with $\mathsf{x} \in \mathbb{F}^\ell, \omega \in \mathbb{F}^{m-\ell}$ such that $\mathbf{x} := (\mathsf{x}, \omega)$ satisfies $\mathscr{C}$.

We proceed to show some useful instantiations of this type of constraints.

Arithmetic circuits: A fan-in 2 circuit of n gates, each being either an addition or multiplication gate, can be captured in such a constraint system as follows.

1. m is set to be the number of wires, and each wire is associated with an index $i \in [m]$.

   For each $i \in [n]$,

2. Set $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i$ to be the index of left/right/output wire of the i'th gate respectively.
3. Set $(\mathbf{q_L})_i = 0, (\mathbf{q_R})_i = 0, (\mathbf{q_M})_i = 1, (\mathbf{q_O})_i = -1$ when the i'th gate is a multiplication gate.

4. Set $(\mathbf{q_L})_i = 1, (\mathbf{q_R})_i = 1, (\mathbf{q_M})_i = 0, (\mathbf{q_O})_i = -1$ when the i'th gate is an addition gate. (Note that we can get "linear combination gates" by using other non-zero values for $(\mathbf{q_L})_i, (\mathbf{q_R})_i$.)

5. Always set $(\mathbf{q_C})_i = 0$.

Booleanity constraints: A common occurrence in proof systems is the need to enforce $\mathbf{x}_j \in \{0,1\}$ for some $j \in [m]$. This is equivalent in our system to setting, for some $i \in [n]$,

$$\mathbf{a}_i = \mathbf{b}_i = j, (\mathbf{q_L})_i = -1, (\mathbf{q_M})_i = 1, (\mathbf{q_R})_i = (\mathbf{q_O})_i = (\mathbf{q_C})_i = 0.$$

Enforcing public inputs: It is quite convenient and direct to enforce values of public inputs $\mathbf{x}_1, \ldots \mathbf{x}_\ell$: To enforce the constraint $\mathbf{x}_j = \mathsf{x}_j$ we set for some $i \in [n]$

$$\mathbf{a}_i = j, (\mathbf{q_L})_i = 1, (\mathbf{q_M})_i = (\mathbf{q_R})_i = (\mathbf{q_O})_i = 0, (\mathbf{q_C})_i = -\mathsf{x}_j.$$

7 Main protocol

Let $\mathscr{C} = (\mathcal{V}, \mathcal{Q})$ be a constraint system of the form described in Section 6. We present our main protocol for the relation $\mathcal{R}_{\mathscr{C}}$. It will be convenient to first predefine the following notion of the *partition of* $\mathscr{C}$, denoted $\mathcal{T}_{\mathscr{C}}$, as follows.

Suppose $\mathcal{V} = (\mathbf{a}, \mathbf{b}, \mathbf{c})$; think of $\mathcal{V}$ as a vector V in $[m]^{3n}$. For $i \in [m]$, let $T_i \subset [3n]$ be the set of indices $j \in [3n]$ such that $V_j = i$. Now define

$$\mathcal{T}_{\mathscr{C}} := \{T_i\}_{i \in [m]}$$

We make a final definition before presenting the protocol. We say $\mathscr{C}$ is *prepared for* ℓ *public inputs* if for $i \in [\ell]$

$$\mathbf{a}_i = i, (\mathbf{q_L})_i = 1, (\mathbf{q_M})_i = (\mathbf{q_R})_i = (\mathbf{q_O})_i = 0, (\mathbf{q_C})_i = 0.$$

Recall that $H = \{\mathbf{g}, \ldots, \mathbf{g}^n\}$. We present an H-ranged polynomial protocol for $\mathcal{R}_{\mathscr{C}}$.

Preprocessing: Let $\sigma = \sigma(\mathcal{T}_{\mathscr{C}})$.

The polynomials $\mathsf{S}_{\mathsf{ID}1}, \mathsf{S}_{\mathsf{ID}2}, \mathsf{S}_{\mathsf{ID}3}, \mathsf{S}_{\sigma 1}, \mathsf{S}_{\sigma 2}, \mathsf{S}_{\sigma 3} \in \mathbb{F}_{<n}[X]$ as defined in the protocol of subsection 5.1 .

Overloading notation, the polynomials $\mathbf{q_L}, \mathbf{q_R}, \mathbf{q_O}, \mathbf{q_M}, \mathbf{q_C} \in \mathbb{F}_{<n}[X]$ defined for each $i \in [n]$ by

$$\mathbf{q_L}(\mathbf{g}^i) := (\mathbf{q_L})_i, \mathbf{q_R}(\mathbf{g}^i) := (\mathbf{q_R})_i, \mathbf{q_O}(\mathbf{g}^i) := (\mathbf{q_O})_i, \mathbf{q_M}(\mathbf{g}^i) := (\mathbf{q_M})_i, \mathbf{q_C}(\mathbf{g}^i) := (\mathbf{q_C})_i$$

Protocol:

1. Let $\mathbf{x} \in \mathbb{F}^m$ be $\mathrm{P}_{\mathsf{poly}}$'s assignment consistent with the public input x. $\mathrm{P}_{\mathsf{poly}}$ computes the three polynomials $f_L, f_R, f_O \in \mathbb{F}_{<n}[X]$, where for $i \in [n]$

$$f_L(i) = \mathbf{x}_{\mathbf{a}_i}, f_R(i) = \mathbf{x}_{\mathbf{b}_i}, f_O(i) = \mathbf{x}_{\mathbf{c}_i}.$$

   $\mathrm{P}_{\mathsf{poly}}$ sends f_L, f_R, f_O to $\mathcal{I}$.

2. $\mathrm{P}_{\mathsf{poly}}$ and $\mathrm{V}_{\mathsf{poly}}$ run the extended permutation check protocol using the permutation σ between (f_L, f_R, f_O) and itself. As explained in Section 5.2, this exactly checks whether (f_L, f_R, f_O) copy-satisfies $\mathcal{T}_{\mathscr{C}}$.

3. $\mathrm{V}_{\mathsf{poly}}$ computes the "Public input polynomial"

$$\mathsf{PI}(X) := \sum_{i \in [\ell]} -x_i \cdot L_i(X)$$

4. $\mathrm{V}_{\mathsf{poly}}$ now checks the identity

$$\mathbf{q_L} \cdot f_L + \mathbf{q_R} \cdot f_R + \mathbf{q_O} \cdot f_O + \mathbf{q_M} \cdot f_L \cdot f_R + (\mathbf{q_C} + \mathsf{PI}) = 0,$$

   on H.

Theorem 7.1. *The above protocol is an H-ranged polynomial protocol for the relation $\mathcal{R}_{\mathscr{C}}$.*

Proof. Our main task is to describe and prove correctness of an extractor E. E simply uses the values of f_L, f_R, f_O to construct an assignment in the natural way - e.g. if $\mathbf{a}_i = j$ for some $i \in [n]$, let $\mathbf{x}_j = f_L(\mathbf{g}^i)$. Finally, E defines and outputs $\omega := (\mathbf{x}_{\ell+1}, \ldots, \mathbf{x}_m)$. Now, let us look at the event C where $(\mathsf{x}, \omega) \notin \mathcal{R}$ but $\mathrm{V}_{\mathsf{poly}}$ outputs acc. We split C into the two subevents, where (f_L, f_R, f_O) doesn't copy-satisfy $\sigma(\mathscr{C})$, and where it does. The first subevent has probability $\mathsf{negl}(\lambda)$ according to the correctness of Lemma 5.3 and its use for copy-satisfiability checks as explained in Section 5.2.

On the other hand, if (f_L, f_R, f_O) copy-satisfies $\sigma(\mathscr{C})$ and the identity checked by $\mathrm{V}_{\mathsf{poly}}$ holds, it must be the case that $(\mathsf{x}, \omega) \in \mathcal{R}_{\mathscr{C}}$. □

Now, using Lemma 4.5 and Lemma 4.7 we get

Corollary 7.2. *Let $\mathscr{C}$ be a constraint system of the form described in Section 6 with parameter n. There is a protocol for the relation $\mathcal{R}_{\mathscr{C}}$ with Knowledge Soundness in the Algebraic Group Model such that*

1. *The prover* $\mathbf{P}$ *requires* $12n + 3$ $\mathbb{G}_1$*-exponentiations.*

2. *The total prover communication consists of* 8 $\mathbb{G}_1$*-elements and* 10 $\mathbb{F}$*-elements.*

Proof. We bound the quantities $\mathsf{e}(\mathscr{P}), t^*(\mathscr{P}), \mathsf{r}(\mathscr{P})$ from Section 4.2; where $\mathscr{P}$ is the polynomial protocol derived from the protocol of Theorem 7.1 using Lemma 4.5. The result then follows from Lemma 4.7 and the discussion after. (For extra clarity, a full self-contained description of the final protocol is given in Section 8.)

We commit to polynomials $f_L, f_R, f_O \in \mathbb{F}_{<n}[X], Z, Z^* \in \mathbb{F}_{<n+1}[X]$ and a polynomial $T \in \mathbb{F}_{<3n}[X]$ resulting from division by Z_{H^*}. This requires $8n+2$ $\mathbb{G}_1$-exponentiations. Then, we need to open at random $x \in \mathbb{F}$: $f_L(x), f_R(x), f_O(x), T(x), \mathsf{S}_{\mathsf{ID}}(x), \mathsf{S}_{\sigma 1}(x), \mathsf{S}_{\sigma 2}(x), \mathsf{S}_{\sigma 3}(x)$ and at $x \cdot \mathbf{g}$: $Z(x \cdot \mathbf{g}), Z^*(x \cdot \mathbf{g})$.

Note that fixing these 10 values, our identity becomes a linear polynomial L which is a linear combination of $\mathbf{q_L}, \mathbf{q_R}, \mathbf{q_O}, \mathbf{q_M}, \mathbf{q_C}, Z, Z^*$. This implies $\mathsf{r}(\mathscr{P}) \leq 10$.

It follows that

- $\mathsf{e}(\mathscr{P}) = 12n + 3$ - as we add to the $8n+2$ cost of commitments, the maximal degree among the polynomial evaluated at x which is $3n$ plus the maximum degree among polynomials evaluated at $x/\mathbf{g}$ which is $n+1$.
- $t^*(\mathscr{P}) = 2$ - as we have two distinct evaluation points.
- $\mathsf{r}(\mathscr{P}) \leq 10$.

□

8 The final protocol, rolled out

For the reader's convenience we present the full final protocol.

Adding zero-knowledge was not explicitly discussed so far, but is implemented here: All that is needed is essentially adding random multiples of Z_{H^*} to the witness based-polynomials. This does not ruin satisfiability, but creates a situation where the values are either completely uniform or determined by verifier equations.

We explicitly define the multiplicative subgroup H as containing the $(n+1)$'th roots of unity in $\mathbb{F}_p$, where ω is a primitive $(n+1)$'th root of unity and a generator of H. i.e: $H = \{\omega, \ldots, \omega^{n+1}\}$.

We note that, when H is defined using roots of unity, the 3 identity permutation polynomials, $\mathsf{S}_{\mathsf{ID}1}, \mathsf{S}_{\mathsf{ID}2}, \mathsf{S}_{\mathsf{ID}3}$ can be represented solely via $\mathsf{S}_{\mathsf{ID}1}$, as $\mathsf{S}_{\mathsf{ID}j}(X) = (j-1) \cdot n + \mathsf{S}_{\mathsf{ID}1}(X)$.

Finally, in the following protocol we use H to refer to a hash function, where $H : \{0,1\}^* \rightarrow \{0,1\}^\ell$ is an efficienly computable hash function that takes arbitrary length inputs and returns ℓ-bit outputs

8.1 Polynomial identities that define a specific circuit

The following polynomials, along with integer n, uniquely define a universal SNARK circuit:

- $\mathsf{q_M}(X), \mathsf{q_L}(X), \mathsf{q_R}(X), \mathsf{q_O}(X), \mathsf{q_C}(X)$, the 'selector' polynomials that define the circuit's arithmetisation
- $\mathsf{S_{ID1}}(X)$: the identity permutation applied to $\mathbf{a}, \mathbf{b}, \mathbf{c}$
- $\mathsf{S}_{\sigma 1}(X), \mathsf{S}_{\sigma 2}(X), \mathsf{S}_{\sigma 3}(X)$: the copy permutation applied to $\mathbf{a}, \mathbf{b}, \mathbf{c}$
- n, the total number of arithmetic gates for a given circuit. This is used by $\mathcal{V}\mathsf{snark}(\lambda)$ to compute the 'zero' polynomial $\mathsf{Z_{H^\star}}(X) = \prod_{i=1}^{n}(X - \omega^i) = \frac{X^{n+1}-1}{(X-\omega^{n+1})}$

8.2 Commitments to wire values

For the following protocol we describe a proof relation for a universal SNARK circuit containing n arithmetic gates. The witnesses to the proof are the wire value witnesses $(w_i)_{i=1}^{3n}$. The commitments $[a]_1, [b]_1, [c]_1$ are computationally binding Kate polynomial commitments to the wire value witnesses, utilizing a structured reference string containing the group elements $(x \cdot [1]_1, \ldots, x^{n+2} \cdot [1]_1)$.

8.3 The un-rolled universal SNARK proof relation

$$\mathcal{R}\mathsf{snark}(\lambda) = \left\{ \begin{array}{c} (x, w, crs) = ((w_i)_{i \in [\ell]}), ((w_i)_{i=1, i \notin [\ell]}^{3n}), \\ (q_{Mi}, q_{Li}, q_{Ri}, q_{Oi}, q_{Ci})_{i=1}^{n}, n, \sigma(x)) \\ \text{For all } i \in \{1, \ldots, 3n\} : w_i \in \mathbb{F}_p, \text{ and for all } i \in \{1, \ldots, n\} : \\ w_i w_{n+i} q_{Mi} + w_i q_{Li} + w_{n+i} q_{Ri} + w_{2n+i} q_{Oi} + q_{Ci} = 0 \\ \text{and for all } i \in \{1, \ldots, 3n\} : w_i = w_{\sigma(i)} \end{array} \right\}$$

8.4 The protocol

Common input:

$$\begin{array}{l} n, (x \cdot [1]_1, \ldots, x^{3n+5} \cdot [1]_1), (q_{Mi}, q_{Li}, q_{Ri}, q_{Oi}, q_{Ci})_{i=1}^{n}, \sigma(X), \\ \mathsf{q_M}(X) = \sum_{i=1}^{n} q_{Mi} \mathsf{L}_i(X), \\ \mathsf{q_L}(X) = \sum_{i=1}^{n} q_{Li} \mathsf{L}_i(X), \\ \mathsf{q_R}(X) = \sum_{i=1}^{n} q_{Ri} \mathsf{L}_i(X), \\ \mathsf{q_O}(X) = \sum_{i=1}^{n} q_{Oi} \mathsf{L}_i(X), \\ \mathsf{q_C}(X) = \sum_{i=1}^{n} q_{Ci} \mathsf{L}_i(X), \\ \mathsf{S_{ID}}(X) = \sum_{i=1}^{n} i \mathsf{L}_i(X), \\ \mathsf{S}_{\sigma 1}(X) = \sum_{i=1}^{n} \sigma(i) \mathsf{L}_i(X), \\ \mathsf{S}_{\sigma 2}(X) = \sum_{i=1}^{n} \sigma(n+i) \mathsf{L}_i(X), \\ \mathsf{S}_{\sigma 3}(X) = \sum_{i=1}^{n} \sigma(2n+i) \mathsf{L}_i(X) \end{array}$$

Prover input: $((w_i)_{i=1}^{3n})$

Verifier input:

$$\begin{array}{l} [q_\mathsf{M}]_1 := \mathsf{q_M}(x) \cdot [1]_1, [q_\mathsf{L}]_1 := \mathsf{q_L}(x) \cdot [1]_1, [q_\mathsf{R}]_1 := \mathsf{q_R}(X) \cdot [1]_1, [q_\mathsf{O}]_1 := \mathsf{q_O}(X) \cdot [1]_1, \\ [s_{\mathsf{ID1}}]_1 := \mathsf{S_{ID1}} \cdot [1]_1, [s_{\sigma 1}]_1 := \mathsf{S}_{\sigma 1} \cdot [1]_1, [s_{\sigma 2}]_1 := \mathsf{S}_{\sigma 2} \cdot [1]_1, [s_{\sigma 3}]_1 := \mathsf{S}_{\sigma 3} \cdot [1]_1 \\ x \cdot [1]_2 \end{array}$$

$\mathcal{P}\mathsf{snark}(\lambda)((w_i)_{i=1}^{3n})$:

Generate random blinding scalars $(b_1, \ldots, b_{12}) \in \mathbb{F}_p$

Compute wire polynomials $\mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X)$:

$$\mathsf{a}(X) = (b_1 X + b_2)\mathsf{Z}_{\mathsf{H}^\star}(X) + \sum_{i=1}^{n} w_i \mathsf{L}_i(X)$$

$$\mathsf{b}(X) = (b_3 X + b_4)\mathsf{Z}_{\mathsf{H}^\star}(X) + \sum_{i=1}^{n} w_{n+i} \mathsf{L}_i(X)$$

$$\mathsf{c}(X) = (b_5 X + b_6)\mathsf{Z}_{\mathsf{H}^\star}(X) + \sum_{i=1}^{n} w_{2n+i} \mathsf{L}_i(X)$$

Compute $[a]_1 := \mathsf{a}(x) \cdot [1]_1, [b]_1 := \mathsf{b}(x) \cdot [1]_1, [c]_1 := \mathsf{c}(x) \cdot [1]_1$

First output of $\mathcal{P}\mathsf{snark}(\lambda)$ is $((w_i)_{i \in [\ell]}, [a]_1, [b]_1, [c]_1)$

Compute permutation challenges $(\beta, \gamma) \in \mathbb{F}_p$:

$$\beta = H([a]_1, [b]_1, [c]_1, (w_i)_{i \in [\ell]}), \gamma = H([a]_1, [b]_1, [c]_1, (w_i)_{i \in \ell}, \beta)$$

Compute permutation polynomials $\mathsf{z}_1(X), \mathsf{z}_2(X)$:

$$\mathsf{z}_1(X) = (b_7 X^2 + b_8 X + b_9)\mathsf{Z}_{\mathsf{H}^\star}(X)$$
$$+\mathsf{L}_1(X) + \textstyle\sum_{i=1}^{n} \left(\mathsf{L}_{i+1}(X) \prod_{j=1}^{i} (w_j + j\beta + \gamma)(w_{n+j} + (n+j)\beta + \gamma)(w_{2n+j} + (2n+j)\beta + \gamma)\right)$$

$$\mathsf{z}_2(X) = (b_{10} X^2 + b_{11} X + b_{12})\mathsf{Z}_{\mathsf{H}^\star}(X)$$
$$+\mathsf{L}_1(X) + \textstyle\sum_{i=1}^{n} \left(\mathsf{L}_{i+1}(X) \prod_{j=1}^{i} (w_j + \sigma(j)\beta + \gamma)(w_{n+j} + \sigma(n+j)\beta + \gamma)(w_{2n+j} + \sigma(2n+j)\beta + \gamma)\right)$$

Compute $[z_1]_1 := \mathsf{z}_1(x) \cdot [1]_1$

Compute $[z_2]_1 := \mathsf{z}_2(x) \cdot [1]_1$

Second output of $\mathcal{P}\mathsf{snark}(\lambda)$ is $([z_1]_1, [z_2]_1)$

Compute quotient challenge $\alpha \in \mathbb{F}_p$:

$$\alpha = H([a]_1, [b]_1, [c]_1, [z_1]_1, [z_2]_1)$$

Compute quotient polynomial $\mathsf{t}(X)$:

$$\begin{aligned}
&\mathsf{t}(X) = \\
&\quad (\mathsf{a}(X)\mathsf{b}(X)\mathsf{q_M}(X) + \mathsf{a}(X)\mathsf{q_L}(X) + \mathsf{b}(X)\mathsf{q_R}(X) + \mathsf{c}(X)\mathsf{q_O}(X) + \mathsf{PI}(X) + \mathsf{q_C}(X)) \tfrac{\alpha}{\mathsf{Z}_{\mathsf{H}^\star}(X)} \\
&+ \big((\mathsf{a}(X) + \beta \mathsf{S}_{\mathsf{ID}1}(X) + \gamma)(\mathsf{b}(X) + \beta(n + \mathsf{S}_{\mathsf{ID}1}(X)) + \gamma)(\mathsf{c}(X) + \beta(2n + \mathsf{S}_{\mathsf{ID}1}(X)) + \gamma)\mathsf{z}_1(X) - \mathsf{z}_1(X\omega^{-1})\big) \tfrac{\alpha^2}{\mathsf{Z}_{\mathsf{H}^\star}(X)} \\
&+ \big((\mathsf{a}(X) + \beta \mathsf{S}_{\sigma 1}(X) + \gamma)(\mathsf{b}(X) + \beta \mathsf{S}_{\sigma 2}(X) + \gamma)(\mathsf{c}(X) + \beta \mathsf{S}_{\sigma 2}(X) + \gamma)\mathsf{z}_2(X) - \mathsf{z}_2(X\omega^{-1})\big) \tfrac{\alpha^3}{\mathsf{Z}_{\mathsf{H}^\star}(X)} \\
&+ \left(\mathsf{z}_1(X\omega^{-1}) - \mathsf{z}_2(X\omega^{-1})\right) \mathsf{L}_n(X) \tfrac{\alpha^4}{\mathsf{Z}_{\mathsf{H}^\star}(X)} \\
&+ (\mathsf{z}_1(X) - \mathsf{z}_2(X)) \, \mathsf{L}_1(X) \tfrac{\alpha^5}{\mathsf{Z}_{\mathsf{H}^\star}(X)}
\end{aligned}$$

Compute $[t]_1 := \mathsf{t}(x) \cdot [1]_1$

Third output of $\mathcal{P}\mathsf{snark}(\lambda)$ is $[t]_1$

Compute evaluation challenge $z \in \mathbb{F}_p$:

$$z = H([a]_1, [b]_1, [c]_1, [z_1]_1, [z_2]_1, [t]_1)$$

Compute opening evaluations:

$$\bar{a} = \mathsf{a}(z), \bar{b} = \mathsf{b}(z), \bar{c} = \mathsf{c}(z), \bar{\mathsf{s}}_{\mathsf{ID1}} = \mathsf{S}_{\mathsf{ID}1}(z), \bar{\mathsf{s}}_{\sigma 1} = \mathsf{S}_{\sigma 1}(z), \bar{\mathsf{s}}_{\sigma 2} = \mathsf{S}_{\sigma 2}(z), \bar{\mathsf{s}}_{\sigma 3} = \mathsf{S}_{\sigma 3}(z), \bar{t} = \mathsf{t}(z)$$
$$\bar{z_1} = \mathsf{z}_1(z\omega^{-1}), \bar{z_2} = \mathsf{z}_2(z\omega^{-1})$$

Compute linearisation polynomial $\mathsf{r}(X)$:

$$\begin{aligned}
&\mathsf{r}(X) = \\
&\quad \left(\bar{a}\bar{b} \cdot \mathsf{q_M}(X) + \bar{a} \cdot \mathsf{q_L}(X) + \bar{b} \cdot \mathsf{q_R}(X) + \bar{c} \cdot \mathsf{q_O}(X) + \mathsf{q_C}(X)\right) \alpha \\
&+ \left((\bar{a} + \beta\bar{\mathsf{s}}_{\mathsf{ID1}} + \gamma)(\bar{b} + \beta(n + \bar{\mathsf{s}}_{\mathsf{ID1}}) + \gamma)(\bar{c} + \beta(2n + \bar{\mathsf{s}}_{\mathsf{ID1}}) + \gamma) \cdot \mathsf{z}_1(X)\right) \alpha^2 \\
&+ \left((\bar{a} + \beta\bar{\mathsf{s}}_{\sigma 1} + \gamma)(\bar{b} + \beta\bar{\mathsf{s}}_{\sigma 2} + \gamma)(\bar{c} + \beta\bar{\mathsf{s}}_{\sigma 3} + \gamma) \cdot \mathsf{z}_2(X)\right) \alpha^3 \\
&+ \left(\mathsf{z}_1(X) - \mathsf{z}_2(X)\right) \mathsf{L}_1(z)\alpha^5
\end{aligned}$$

Compute linearisation evaluation $\bar{r} = \mathsf{r}(z)$
Compute opening challenge $v \in \mathbb{F}_p$:

$$v = H([a]_1, [b]_1, [c]_1, [z_1]_1, [z_2]_1, [t]_1, \bar{a}, \bar{b}, \bar{c}, \bar{\mathsf{s}}_{\mathsf{ID1}}, \bar{\mathsf{s}}_{\sigma 1}, \bar{\mathsf{s}}_{\sigma 2}, \bar{\mathsf{s}}_{\sigma 3}, \bar{z_1}, \bar{z_2}, \bar{t}, \bar{r})$$

Compute opening polynomial $\mathsf{W_z}(X)$:

$$\mathsf{W_z}(X) = \frac{1}{X - z} \left(\begin{array}{l} (\mathsf{t}(X) - \bar{t}) \\ +v(\mathsf{r}(X) - \bar{r}) \\ +v^2(\mathsf{a}(X) - \bar{a}) \\ +v^3(\mathsf{b}(X) - \bar{b}) \\ +v^4(\mathsf{c}(X) - \bar{c}) \\ +v^5(\mathsf{S}_{\mathsf{ID}1}(X) - \bar{\mathsf{s}}_{\mathsf{ID1}}) \\ +v^6(\mathsf{S}_{\sigma 1}(X) - \bar{\mathsf{s}}_{\sigma 1}) \\ +v^7(\mathsf{S}_{\sigma 2}(X) - \bar{\mathsf{s}}_{\sigma 2}) \\ +v^8(\mathsf{S}_{\sigma 3}(X) - \bar{\mathsf{s}}_{\sigma 3}) \end{array} \right)$$

Compute opening polynomial $\mathsf{W}_{\mathsf{z}\omega^{-1}}(X)$:

$$\mathsf{W}_{\mathsf{z}\omega^{-1}}(X) = \frac{1}{X - z\omega^{-1}} \left(v^9(\mathsf{z}_1(X) - \bar{z_1}) + v^{10}(\mathsf{z}_2(X) - \bar{z_2})\right)$$

Compute $[W_z]_1 := \mathsf{W_z}(x) \cdot [1]_1, [W_{z\omega^{-1}}] := \mathsf{W}_{\mathsf{z}\omega^{-1}}(x) \cdot [1]_1$
Return

$$\pi_{\mathsf{SNARK}} = \left(\begin{array}{c} [a]_1, [b]_1, [c]_1, [z_1]_1, [z_2]_1, [W_z]_1, [W_{z\omega^{-1}}]_1, \\ \bar{a}, \bar{b}, \bar{c}, \bar{\mathsf{s}}_{\mathsf{ID1}}, \bar{\mathsf{s}}_{\sigma 1}, \bar{\mathsf{s}}_{\sigma 2}, \bar{\mathsf{s}}_{\sigma 3}, \bar{r}, \bar{z_1}, \bar{z_2} \end{array} \right)$$

$\mathcal{V}\mathsf{snark}(\lambda)((w_i)_{i\in[\ell]}, \pi_{\mathsf{SNARK}})$**:**

Validate $([a]_1, [b]_1, [c]_1, [z_1]_1, [z_2]_1, [t]_1, [W_z]_1, [W_{z\omega^{-1}}]_1) \in \mathbb{G}_1$

Validate $(\bar{a}, \bar{b}, \bar{c}, \bar{\mathsf{s}}_{\mathsf{ID1}}, \bar{\mathsf{s}}_{\sigma 1}, \bar{\mathsf{s}}_{\sigma 2}, \bar{\mathsf{s}}_{\sigma 3}, \bar{r}, \bar{z_1}, \bar{z_2}) \in \mathbb{F}_p^{10}$

Validate $(w_i)_{i\in[\ell]} \in \mathbb{F}_p^{\ell}$

Compute permutation challenges $(\beta, \gamma) \in \mathbb{F}_p$:

$$\beta = H([a]_1, [b]_1, [c]_1, (w_i)_{i\in[\ell]}), \gamma = H([a]_1, [b]_1, [c]_1, (w_i)_{i\in[\ell]}, \beta)$$

Compute quotient challenge $\alpha \in \mathbb{F}_p$:

$$\alpha = H([a]_1, [b]_1, [c]_1, [z_1]_1, [z_2]_1)$$

Compute evaluation challenge $z \in \mathbb{F}_p$:

$$z = H([a]_1, [b]_1, [c]_1, [z_1]_1, [z_2]_1, [t]_1)$$

Compute zero polynomial evaluation $\mathsf{Z}_{\mathsf{H}^\star}(z) = \frac{z^{n+2}-1}{(z-\omega^{n+1})(z-\omega^{n+2})}$

Compute Lagrange polynomial evaluation $\mathsf{L}_1(z) = \frac{z^{n+2}-1}{(z-\omega)\prod_{j=2}^{n+2}(\omega-\omega^j)}$

Compute Lagrange polynomial evaluation $\mathsf{L}_n(z) = \frac{z^{n+2}-1}{(z-\omega^n)\prod_{j=1,j\neq n}^{n+2}(\omega^n-\omega^j)}$

Compute public input polynomial evaluation $\mathsf{PI}(z) = \sum_{i\in\ell} w_i \mathsf{L}_i(z)$

Compute quotient polynomial evaluation $\bar{t} = \frac{\bar{r}+\mathsf{PI}(z)\alpha - \bar{z_1}\alpha^2 - \bar{z_2}\alpha^3 + (\bar{z_1}-\bar{z_2})\mathsf{L}_n(z)\alpha^4}{\mathsf{Z}_{\mathsf{H}^\star}(z)}$

Compute opening challenge $v \in \mathbb{F}_p$:

$$v = H([a]_1, [b]_1, [c]_1, [z_1]_1, [z_2]_1, [t]_1, \bar{a}, \bar{b}, \bar{c}, \bar{\mathsf{s}}_{\mathsf{ID1}}, \bar{\mathsf{s}}_{\sigma 1}, \bar{\mathsf{s}}_{\sigma 2}, \bar{\mathsf{s}}_{\sigma 3}, \bar{z_1}, \bar{z_2}, \bar{t}, \bar{r})$$

Compute multipoint evaluation separation challenge $u \in \mathbb{F}_p$:

$$u = H([a]_1, [b]_1, [c]_1, [z_1]_1, [z_2]_1, [t]_1, [W_z]_1, [W_{z\omega^{-1}}]_1, \bar{a}, \bar{b}, \bar{c}, \bar{\mathsf{s}}_{\mathsf{ID1}}, \bar{\mathsf{s}}_{\sigma 1}, \bar{\mathsf{s}}_{\sigma 2}, \bar{\mathsf{s}}_{\sigma 3}, \bar{z_1}, \bar{z_2}, \bar{t}, \bar{r})$$

Compute partial opening commitment $[D]_1 := v \cdot [r]_1 + v^9 u \cdot [z_1]_1 + v^{10} u \cdot [z_2]_1$:

$$\begin{aligned}[D]_1 := \; & \bar{a}\bar{b}\alpha v \cdot [q_{\mathsf{M}}]_1 + \bar{a}\alpha v \cdot [q_{\mathsf{L}}]_1 + \bar{b}\alpha v \cdot [q_{\mathsf{R}}]_1 + \bar{c}\alpha v \cdot [q_{\mathsf{O}}]_1 + \alpha v \cdot [q_{\mathsf{C}}]_1 \\ & + \big((\bar{a} + \beta \bar{\mathsf{s}}_{\mathsf{ID1}} + \gamma)(\bar{b} + \beta(n + \bar{\mathsf{s}}_{\mathsf{ID1}}) + \gamma)(\bar{c} + \beta(2n + \bar{\mathsf{s}}_{\mathsf{ID1}}) + \gamma)\alpha^2 v + \mathsf{L}_1(z)\alpha^5 v + v^9\big) \cdot [z_1]_1 \\ & + \big((\bar{a} + \beta \bar{\mathsf{s}}_{\sigma 1} + \gamma)(\bar{b} + \beta \bar{\mathsf{s}}_{\sigma 2} + \gamma)(\bar{c} + \beta \bar{\mathsf{s}}_{\sigma 3} + \gamma) r^3 v - \mathsf{L}_1(z)\alpha^5 v + v^{10}\big) \cdot [z_2]_1\end{aligned}$$

Compute batch opening commitment $[F]_1$:

$$\begin{aligned}[F]_1 := \; & [t]_1 + [D]_1 + v^2 \cdot [a]_1 + v^3 \cdot [b]_1 + v^4 \cdot [c]_1 + v^5 \cdot [s_{\mathsf{ID1}}]_1 \\ & + v^6 \cdot [s_{\sigma 1}]_1 + v^7 \cdot [s_{\sigma 2}]_1 + v^8 \cdot [s_{\sigma 3}]_1\end{aligned}$$

Compute batch evaluation commitment $[E]_1$:

$$[E]_1 := \big(\bar{t} + v\bar{r} + v^2\bar{a} + v^3\bar{b} + v^4\bar{c} + v^5\bar{\mathsf{s}}_{\mathsf{ID1}} + v^6\bar{\mathsf{s}}_{\sigma 1} + v^7\bar{\mathsf{s}}_{\sigma 2} + v^8\bar{\mathsf{s}}_{\sigma 3} + v^9 u\bar{z_1} + v^{10} u\bar{z_2}\big) \cdot [1]_1$$

Validate

$$e([W_z]_1 + u \cdot [W_{z\omega^{-1}}]_1, x \cdot [1]_2) \stackrel{?}{=} e(z \cdot [W_z]_1 + z\omega^{-1} u \cdot [W_{z\omega^{-1}}]_1 + [F]_1 + [E]_1, [1]_2)$$

Acknowledgements

We thank Mary Maller for telling us about the field element reduction method discussed in the end of Section 4.

References

[BCC+16] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, pages 327–357, 2016.

[BG12] S. Bayer and J. Groth. Efficient zero-knowledge argument for correctness of a shuffle. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 263–280, 2012.

[BGM17] S. Bowe, A. Gabizon, and I. Miers. Scalable multi-party computation for zk-snark parameters in the random beacon model. Cryptology ePrint Archive, Report 2017/1050, 2017. https://eprint.iacr.org/2017/1050.

[CS10] C. Costello and D. Stebila. Fixed argument pairings. In *International Conference on Cryptology and Information Security in Latin America*, pages 92–108. Springer, 2010.

[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 19-23, 2018, Proceedings, Part II*, pages 33–62, 2018.

[GGPR13] R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct nizks without pcps. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 626–645, 2013.

[GKM+] J. Groth, M. Kohlweiss, M. Maller, S. Meiklejohn, and I. Miers. Updatable and universal common reference strings with applications to zk-snarks. *IACR Cryptology ePrint Archive*, 2018.

[Gro16] J. Groth. On the size of pairing-based non-interactive arguments. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, pages 305–326, 2016.

[KZG10] A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, pages 177–194, 2010.

[MBKM19] M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero-knowledge snarks from linear-size universal and updateable structured reference strings. *IACR Cryptology ePrint Archive*, 2019:99, 2019.

A Claims for permutation argument:

Fix a permutation σ of $[n]$, and $a_1, \ldots, a_n, b_1, \ldots, b_n \in \mathbb{F}$.

Claim A.1. *If the following holds with non-negligible probability over random $\beta, \gamma \in \mathbb{F}$*

$$\prod_{i \in [n]} (a_i + \beta \cdot i + \gamma) = \prod_{i \in [n]} (b_i + \beta \cdot \sigma(i) + \gamma)$$

then $\forall i \in [n]$, $b_i = a_{\sigma(i)}$.

We will prove claim A.1 using the following lemmas. For completeness, we include below a variant of the well-known Schwartz-Zippel lemma that we use in this paper.

Lemma A.2. *(Schwartz-Zippel). Let $P(X_1, \ldots, X_n)$ be a non-zero multivariate polynomial of degree d over $\mathbb{Z}_p$, then the probability of $P(\alpha_1, \ldots, \alpha_n) = 0 \leftarrow \mathbb{Z}_p$ for randomly chosen $\alpha_1, \ldots, \alpha_n$ is at most d/p.*

The Schwartz-Zippel lemma is used in polynomial equality testing. Given two multivariate polynomials $P_1(X_1, \ldots, X_n)$ and $P_2(X_1, \ldots, X_n)$ we can test whether $P_1(\alpha_1, \ldots, \alpha_n) - P_2(\alpha_1, \ldots, \alpha_n) = 0$ for random $\alpha_1, \ldots, \alpha_n \leftarrow \mathbb{Z}_p$. If the two polynomials are identical, this will always be true, whereas if the two polynomials are different then the equality holds with probability at most $max(d_1, d_2)/p$, where d_1 and d_2 are the degrees of the polynomials P_1 and P_2.

Lemma A.3. *If the following holds with non-negligible probability over random $\gamma \in \mathbb{F}$,*

$$\prod_{i=1}^{n} (a_i + \gamma) = \prod_{i=1}^{n} (b_i + \gamma) \tag{1}$$

then the entries in the tuple $(a_1, \ldots, a_n)$ equal the entries in the tuple $(b_1, \ldots, b_n)$, but not necessarily in that order.

Proof. Let $P_a(X) = \prod_{i=1}^{n}(X + a_i)$ and $P_b(X) = \prod_{i=1}^{n}(X + b_i)$. The roots of P_a are $(-a_1, \ldots, -a_n)$ and the roots of P_b are $(-b_1, \ldots, -b_n)$. By the Schwartz-Zippel lemma, if polynomials $P_a(X)$ and $P_b(X)$ are not equal, equality 1 holds with probability $\frac{n}{|\mathbb{F}|}$ which is negligible for any polynomial degree n used in our snark construction. Thus, equality 1 holds with non-negligible probability only when the two polynomials $P_a(X)$ and $P_b(X)$ are equal. This implies all the roots of $P_a(X)$ must be roots of $P_b(X)$ and the other way around, which, in turn, implies the conclusion of the lemma. As a note, the conclusion of the lemma can be written in an equivalent but more formal way: there exist a permutation σ of $[n]$ such that $b_i = a_{\sigma(i)}, \forall i \in [n]$. □

Corollary A.4. *Fix a permutation σ of $[n]$, and $A_1, \ldots, A_n, B_1, \ldots, B_n \in \mathbb{F}$. If the following holds with non-negligible probability over random $\beta \in \mathbb{F}$,*

$$\prod_{i\in[n]} (A_i + \beta \cdot i) = \prod_{i\in[n]} (B_i + \beta \cdot \sigma(i))$$

then the values in the tuple $(\frac{B_1}{\sigma(1)}, \ldots, \frac{B_n}{\sigma(n)})$ are the same as the values in the tuple $(A_1, \ldots, \frac{A_n}{n})$, but not necessarily in this order.

Proof. The roots of $P_A(Y) = \prod_{i\in[n]}(i \cdot Y + A_i)$ are $(-A_1, \ldots, -\frac{A_n}{n})$, the roots of $P_B(Y) = \prod_{i\in[n]}(\sigma(i) \cdot Y + B_i)$ are $(-\frac{B_1}{\sigma(1)}, \ldots, -\frac{B_n}{\sigma(n)})$. Together with lemma A.3, we obtain the desired conclusion. □

Proof. for Claim A.1

Lets denote by $B_i = b_i + \gamma$ and $A_i = a_i + \gamma$, $\forall i \in [n]$. Then, according to A.4, the values in the tuple $(\frac{B_1}{\sigma(1)}, \ldots, \frac{B_n}{\sigma(n)})$ are the same as the values in the tuple $(A_1, \ldots, \frac{A_n}{n})$, $\forall i \in [n]$. Assume there exists $i_0 \in [n]$ such that $B_{i_0} \neq A_{\sigma(i_0)}$. This implies there exists $j \in [n]$, with $j \neq \sigma(i_0)$ such that $\frac{B_{i_0}}{\sigma(i_0)} = \frac{A_j}{j}$. Expending, we obtain $j \cdot (b_{i_0} + \gamma) = \sigma(i_0) \cdot (a_j + \gamma)$ and this holds with non-negligible probability over the choice of γ. Using the Schwartz-Zippel lemma as in A.3, we obtain that $\sigma(i_0) = j$ which contradicts our assumption. Hence, we have proven that $\forall i \in [n], B_i = A_{\sigma(i)}$, which, in turn, implies $b_i = a_{\sigma(i)}$, $\forall i \in [n]$. □