# Public inputs in $\mathcal{P}\mathfrak{lon}\mathcal{K}$ in the COVID-19 era

Ariel Gabizon
Aztec

Zachary J. Williamson
Aztec

March 18, 2021

**Abstract**

In this note we give a variant of the $\mathcal{P}\mathfrak{lon}\mathcal{K}$ permutation argument[GWC19] that enables a publc input component. The implication of this is ultimately enabling the $\mathcal{P}\mathfrak{lon}\mathcal{K}$ verifier to perform two field multiplications per public input, instead of a group exponentiation as in the original proof system.

## 1 Introduction

In the $\mathcal{P}\mathfrak{lon}\mathcal{K}$ zk-SNARK [GWC19] a set of checks must be performed on committed polynomials $f_1, \ldots, f_d$ of the prover. One of these checks is the so-called "public input check" where we check that a set of $\ell$ values $f_1(\mathbf{g}), \ldots, f_1(\mathbf{g}^\ell)$ of the first polynomial are equal to values $\mathsf{x}_1, \ldots, \mathsf{x}_\ell$ given to the verifier.

In [GWC19] this requires

- the prover computing the polynomial $\mathsf{PI}(X) := \sum_{i=1}^{\ell} -\mathsf{x}_i \cdot L_i(X)$ on a set of $(d-1)n$ values using an FFT - requiring roughly $dn \log n + \ell \log n$ field multiplications.
- The verifier computing $\mathsf{PI}(\mathfrak{z})$ on some $\mathfrak{z} \in \mathbb{F}$, requiring roughly $\ell \log n$ multiplications.

We describe here an alternative method used in the barretenberg code base, where

- the prover doesn't require any computation beyond the permutation argument.
- The verifier only requires $2\ell$ field multiplications.

We achieve this by adding a public input componet in the permutation argument itself, allowing it to verify certain consistency with the public inputs at the same time as validating the permutation. The basic idea will be to "unbalance" the grand product from [GWC19] so that instead of accumulating to one, it will reach a randomized value dependent on the public input

We use the ranged polynomial protocol terminology from [GWC19] to describe our protocol. See Section 4 of [GWC19] for an explanation.

## 2 Polynomial protocols for identifying permutations with a public input

We assume $H \in \mathbb{F}$ is a multiplicative subgroup of order $n+1$ with generator $\mathbf{g}$. We let $H := \{1, \mathbf{g}, \mathbf{g}^2, \ldots, \mathbf{g}^{n-1}\}$

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

For $f \in \mathbb{F}_{<d}[X]$ and a permutation $\sigma : [n] \to [n]$, we write $g = \sigma(f)$ if for each $i \in [n]$, $g(\mathbf{g}^i) = f(\mathbf{g}^{\sigma(i)})$. [1]

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

We present a version of the protocol that also supports public inputs. That is for $i \in [\ell]$, it will also check that $f(\mathbf{g}^i) = \mathsf{x}_i$.

Before proceeding, given the permutation $\sigma$ on $[n]$, we define $\sigma' : [n] \to \mathbb{F}$ as follows. We predefine some subset $\{\zeta_1, \ldots, \zeta_\ell\}$ of distinct elements of $\mathbb{F}$ disjoint from $[n]$. We define $\sigma'(i) = \zeta_i$ for $i \in [\ell]$ and otherwise $\sigma'(i) = \sigma(i)$.

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: $f, g \in \mathbb{F}_{<n}[X]$. Public input $\mathsf{x} \in \mathbb{F}^\ell$.

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 $i \in [n]$
$$f'(\mathbf{g}^i) = f(\mathbf{g}^i) + \beta \cdot i + \gamma, g'(\mathbf{g}^i) = g(\mathbf{g}^i) + \beta \cdot \sigma'(i) + \gamma$$
3. $\mathrm{P}_{\mathsf{poly}}$ computes $Z \in \mathbb{F}_{<n+1}[X]$, such that $Z(1) = 1$; and for $i \in \{1, \ldots, n+1\}$
$$Z(\mathbf{g}^i) = \prod_{0 \le j < i} f'(\mathbf{g}^j)/g'(\mathbf{g}^j).$$
(If one of the product elements is undefined, which happens w.p. $\mathsf{negl}(\lambda)$ over $\gamma$, the protocol is aborted[2].)
4. $\mathrm{P}_{\mathsf{poly}}$ sends $Z$ to $\mathcal{I}$.

[1] 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.

[2] This abort ruins the perfect completeness of the protocol. If one wishes to preserve perfect completeness, the protocol can be altered such that if for some $i$, $g'(\mathbf{g}^i) = 0$, $\mathrm{P}_{\mathsf{poly}}$ proves this to $\mathrm{V}_{\mathsf{poly}}$, and $\mathrm{V}_{\mathsf{poly}}$ accepts in this case. This adds a $\mathsf{negl}(\lambda)$ factor to the soundness error.

5. $\mathsf{V}_{\mathsf{poly}}$ computes the term

$$\Delta_{\mathrm{PI}} := \frac{\prod_{i\in[\ell]}(\mathsf{x}_i + \beta\sigma(i) + \gamma)}{\prod_{i\in[\ell]}(\mathsf{x}_i + \beta\sigma'(i) + \gamma)}$$

6. $\mathsf{V}_{\mathsf{poly}}$ checks if for all $\mathbf{x} \in H$

   (a) $L_1(\mathbf{x})(Z(\mathbf{x}) - 1) = 0$.

   (b) $Z(\mathbf{x})f'(\mathbf{x}) = g'(\mathbf{x})Z(\mathbf{x} \cdot \mathbf{g})$.

   (c) $L_n(\mathbf{x})(Z(\mathbf{x} \cdot \mathbf{g}) - \Delta_{\mathrm{PI}}) = \mathsf{x}$.

   and outputs $\mathsf{acc}$ iff all checks hold.

## 2.1 Analysis

For simplicity we use the notation $f_i := f(\mathbf{g}^{i-1})$ below. Given fixed $f \in \mathbb{F}_{<d}[X]$ and $\mathsf{x} \in \mathbb{F}^{\ell}$, define

- $N := \prod_{i\in[n]}(f_i + \beta \cdot i + \gamma)$
- $D'_{\ell} := \prod_{i\in[\ell]}(f_i + \beta \cdot \sigma'(i) + \gamma)$
- $D_{n-\ell} := \prod_{i\in[\ell+1..n]}(f_i + \beta\sigma(i) + \gamma)$,
- $N_{\mathsf{x}} := \prod_{i\in\ell}(\mathsf{x}_i + \beta\sigma(i) + \gamma)$
- $D_{\mathsf{x}} := \prod_{i\in\ell}(\mathsf{x}_i + \beta\sigma'(i) + \gamma)$

The correctness of the protocol follows from

**Lemma 2.1.** *Assume that $\{\sigma'(i)\}_{i\in[\ell]}$ is disjoint from $\{\sigma(i)\}_{i\in[n]}$ and consists of distinct values.*

*If*

1. $f = \sigma(f)$ *and*
2. $f_i = \mathsf{x}_i$ *for some* $i \in [\ell]$

*then for any* $\beta, \gamma \in \mathbb{F}$

$$N/(D'_{\ell} \cdot D_{n-\ell}) = N_{\mathsf{x}}/D_{\mathsf{x}}$$

*Suppose that either*

1. $f \neq \sigma(f)$ *or*
2. $f_i \neq \mathsf{x}_i$ *for some* $i \in [\ell]$

*Then e.w.p* $2(n + \ell)/|\mathbb{F}|$ *over* $\beta, \gamma \in \mathbb{F}$.

$$N/(D'_{\ell} \cdot D_{n-\ell}) \neq N_{\mathsf{x}}/D_{\mathsf{x}}$$

*Proof.* Clearing denominators, and ignoring an $(n+\ell)/|\mathbb{F}|$ probability of denominators being zero, the equality above is equivalent to

$$N \cdot D_{\mathsf{x}} = N_{\mathsf{x}} \cdot D'_{\ell} \cdot D_{n-\ell}$$

which is the same as

$$\prod_{i\in[n]}(f_i+\beta\cdot i+\gamma)\prod_{i\in[\ell]}(\mathsf{x}_i+\beta\sigma'(i)+\gamma) = \prod_{i\in[\ell]}(\mathsf{x}_i+\beta\sigma(i)+\gamma)\prod_{i\in[\ell]}(f_i+\beta\sigma'(i)+\gamma)\prod_{i\in[\ell+1..n]}(f_i+\beta\sigma(i)+\gamma)$$

When $f = \sigma(f)$ and $f_i = \mathsf{x}_i$ for $i \in [\ell]$, inspection shows the linear factors on both sides are the same.

For the other direction, assume first $f_i \neq \mathsf{x}_i$ for some $i \in [\ell]$. Think of both sides as bi-variate polynomials in $\beta, \gamma$. Recall we have unique decomposition to irreducible factors in $\mathbb{F}[\beta, \gamma]$. Since both sides are decomposed to linear factors, this means that if we show a factor only appears on one side, the polynomials are different and thus disagree e.w.p $(n+\ell)/|\mathbb{F}|$. Then the factor $(\mathsf{x}_i + \beta\sigma'(i) + \gamma)$ appears on LHS but not RHS.

Now assume $f_i = \mathsf{x}_i$ for each $i \in [\ell]$ and for some $i \in [n]$, $f_i \neq f_{\sigma(i)}$. Then the factor $f_i + \beta\sigma(i) + \gamma$ appears in the RHS but not LHS.

□

**Lemma 2.2.** *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* negl$(\lambda)$.

*Proof.* Suppose that $g \neq \sigma(f)$. By claim A.1, by Lemma 2.1 e.w.p negl$(\lambda)$ over the choice of $\beta, \gamma \in \mathbb{F}$,

$$a := \prod_{i\in[n]} f'_i/g'_i \neq \Delta_{\mathrm{PI}}.$$

Assume $\beta, \gamma$ were chosen such that the above holds, and also such that $g'_i \neq 0$ for all $i \in [n]$. We show $\mathrm{V}_{\mathsf{poly}}$ rejects; specifically, that assuming both identities $\mathrm{V}_{\mathsf{poly}}$ checks hold leads to contradiction.

From the first check we know that $Z(1) = 1$. From the second check we can show inductively, that for each $i \in [n]$

$$Z(\mathbf{g}^i) = \prod_{0\leq j<i} \frac{f'(\mathbf{g}^j)}{g'(\mathbf{g}^j)}.$$

In particular, $Z(\mathbf{g}^n) = a$. From the last check we know that $Z(\mathbf{g}^n) = \Delta_{\mathrm{PI}} \neq a$; hence we've arrived at a contradiction. □

As in [GWC19], we give the version of the argument for multiple polynomials, though it is straightforward given the above.

## 2.2 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

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

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* $\mathsf{S}_{\mathsf{ID}j}(x) = \mathsf{S}_{\mathsf{ID}}(x) + (j-1)\cdot n$.

Let $(\zeta_1, \ldots, \zeta_\ell)$ be a set of $\ell$ distinct elements in $\mathbb{F}$ disjoint from $[kn]$.

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 $i \in [n]$; with the exception that for $i \in [\ell]$, we set $\mathsf{S}_{\sigma 1}(\mathbf{g}^i) = \zeta_i$.

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 $j \in [k], i \in [n]$

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

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 $Z \in \mathbb{F}_{<n+1}[X]$, such that $Z(1) = 1$; and for $i \in \{1, \ldots, n\}$

   $$Z(\mathbf{g}^i) = \prod_{0\le j<i} f'(\mathbf{g}^j)/g'(\mathbf{g}^j).$$

   (The case of one of the products being undefined is handled as in the previous protocol.)

5. $\mathrm{P}_{\mathsf{poly}}$ sends $Z$ to $\mathcal{I}$.
6. $\mathrm{V}_{\mathsf{poly}}$ checks if for all $a \in H$

(a) $L_1(a)(Z(a)-1)=0$.

(b) $Z(a)f'(a)=g'(a)Z(a\cdot\mathbf{g})$.

(c) $L_n(a)(Z(a)-\Delta_{\mathrm{PI}})=0$.

and outputs $\mathsf{acc}$ iff all checks hold.

**Lemma 2.3.** *Fix any $f_1,\ldots,f_k,g_1,\ldots g_k\in\mathbb{F}_{<d}[X]$ and permutation $\sigma$ 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 $\mathsf{acc}$ 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

$$F=\prod_{i\in[n]}f'(\mathbf{g}^{i-1}),G=\prod_{i\in[n]}g'(\mathbf{g}^{i-1}),$$

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. □

## 3 The final protocol, rolled out

For the reader's convenience we present the full $\mathcal{P}\mathfrak{lonK}$ zk-SNARK from [GWC19] when the method for handling public inputs is altered to the one above.

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 subet $H\subset\mathbb{F}$ as containing the $n$ first powers of an $n+1$'th root of unity in $\mathbb{F}_p$ $\omega$ . i.e: $H=\{1,\omega,\ldots,\omega^{n-1}\}$.

We assume that the number of gates in a circuit is no more than $n$.

We also include an optimisation suggested by Vitalik Buterin, to define the identity permutations through degree-1 polynomials. The identity permutations must map each wire value to a unique element $\in\mathbb{F}$. This can be done by defining $\mathsf{S_{ID}}_1(X)=X,\mathsf{S_{ID}}_2(X)=k_1X,\mathsf{S_{ID}}_3(X)=k_2X$, where $k_1,k_2$ are quadratic non-residues $\in\mathbb{F}$. This effectively maps each wire value to a root of unity in $H$, with right and output wires having an additional multiplicative factor of $k_1,k_2$ applied respectively. By representing the identity permutation via degree-1 polynomials, their evaluations can be directly computed by the verifier. This reduces the size of the proof by 1 $\mathbb{F}$ element, as well as reducing the number of Fast-Fourier-Transforms required by the prover.

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

### 3.1 Polynomials 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
- $\mathsf{S_{ID1}}(X) = X, \mathsf{S_{ID2}}(X) = k_1 X, \mathsf{S_{ID3}}(X) = k_2 X$: the identity permutation applied to $\mathbf{a}, \mathbf{b}, \mathbf{c}$. $k_1, k_2 \in \mathbb{F}$ are chosen such that $H, k1 \cdot H, k2 \cdot H$ are distinct cosets of $H$ in $\mathbb{F}^*$, and thus consist of $3n$ distinct elements. (For example, when $\omega$ is a quadratic residue in $\mathbb{F}$, take $k_1$ to be any quadratic non-residue, and $k_2$ to be a quadratic non-residue not contained in $k_1 \cdot H$.)
- $\mathsf{S}_{\sigma 1}(X), \mathsf{S}_{\sigma 2}(X), \mathsf{S}_{\sigma 3}(X)$: the copy permutation applied to $\mathbf{a}, \mathbf{b}, \mathbf{c}$. We the exception that $\mathsf{S}_{\sigma 1}(\mathbf{g}^i) = \zeta_i$ for $i \in [\ell]$, where $\{z_1, \ldots, z_\ell\}$ are distinct and disjoint from the values $H \cup k_1 \cdot H \cup k_2 \cdot H$.
- $n$, the total number of arithmetic gates for a given circuit. This is used by $\mathbf{V}$ to compute the vanishing polynomial $\mathsf{Z_H}(X) = X^n - 1$

### 3.2 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 $(x \cdot [1]_1, \ldots, x^n \cdot [1]_1)$.

### 3.3 The un-rolled universal SNARK proof relation

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

### 3.4 The protocol

We describe the protocol below as a non-interactive protocol using the Fiat-Shamir hueristic. For this purpose we always denote by transcript the concatenation of the common preprocessed input, and public input, and the proof elements written by the

prover up to a certain point in time. We use transcript for obtaining random challenges via Fiat-Shamir. One can alternatively, replace all points where we write below "compute challenges", by the verifier sending random field elements, to obtain the interactive protocol from which we derive the non-interactive one.

**Common preprocessed input:**

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

**Public input:** $\ell, (w_i)_{i \in [\ell]}$

**Prover algorithm:**

**Prover input:** $(w_i)_{i \in [3n]}$

**Round 1:**
Generate random blinding scalars $(b_1, \ldots, b_9) \in \mathbb{F}_p$
Compute wire polynomials $\mathsf{a}(X), \mathsf{b}(X), \mathsf{c}(X)$ :

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

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

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

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

First output of $\mathbf{P}$ is $([a]_1, [b]_1, [c]_1)$.

**Round 2:**
Compute permutation challenges $(\beta, \gamma) \in \mathbb{F}_p$ :

$$\beta = H(\mathsf{transcript}, 0), \gamma = H(\mathsf{transcript}, 1)$$

Compute permutation polynomial $\mathsf{z}(X)$ :

$$\mathsf{z}(X) = (b_7X^2 + b_8X + b_9)\mathsf{Z_H}(X)$$
$$+\mathsf{L}_1(X) + \sum_{i=1}^{n} \left(\mathsf{L}_{i+1}(X) \prod_{j=1}^{i} \frac{(w_j+\beta\omega^{j-1}+\gamma)(w_{n+j}+\beta k_1\omega^{j-1}+\gamma)(w_{2n+j}+\beta k_2\omega^{j-1}+\gamma)}{(w_j+\sigma(j)\beta+\gamma)(w_{n+j}+\sigma(n+j)\beta+\gamma)(w_{2n+j}+\sigma(2n+j)\beta+\gamma)}\right)$$

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

Second output of $\mathbf{P}$ is $([z]_1)$

**Round 3:**
Compute quotient challenge $\alpha \in \mathbb{F}_p$ :

$$\alpha = H(\mathsf{transcript})$$

Compute the element $\Delta_{\mathrm{PI}} := \frac{\prod_{i\in[\ell]}(w_i+\beta\sigma(i)+\gamma}{w_i+\beta\cdot\zeta_i+\gamma}$. Compute quotient polynomial $\mathsf{t}(X)$ :

$$\begin{aligned}
&\mathsf{t}(X) = \\
&\quad(\mathsf{a}(X)\mathsf{b}(X)\mathsf{q_M}(X) + \mathsf{a}(X)\mathsf{q_L}(X) + \mathsf{b}(X)\mathsf{q_R}(X) + \mathsf{c}(X)\mathsf{q_O}(X) + \mathsf{q_C}(X)) \tfrac{1}{\mathsf{Z_H}(X)} \\
&+((\mathsf{a}(X) + \beta X + \gamma)(\mathsf{b}(X) + \beta k_1 X + \gamma)(\mathsf{c}(X) + \beta k_2 X + \gamma)\mathsf{z}(X)) \tfrac{\alpha}{\mathsf{Z_H}(X)} \\
&-((\mathsf{a}(X) + \beta\mathsf{S}_{\sigma 1}(X) + \gamma)(\mathsf{b}(X) + \beta\mathsf{S}_{\sigma 2}(X) + \gamma)(\mathsf{c}(X) + \beta\mathsf{S}_{\sigma 3}(X) + \gamma)\mathsf{z}(X\omega)) \tfrac{\alpha}{\mathsf{Z_H}(X)} \\
&+(\mathsf{z}(X) - 1)\,\mathsf{L}_1(X)\tfrac{\alpha^2}{\mathsf{Z_H}(X)} \\
&+(\mathsf{z}(X\omega) - \Delta_{\mathrm{PI}})\mathsf{L}_n(X)\tfrac{\alpha^3}{\mathsf{Z_H}(X)}
\end{aligned}$$

Split $t(X)$ into degree$< n$ polynomials $t_{lo}(X), t_{mid}(X), t_{hi}(X)$, where

$$t(X) = t_{lo}(X) + X^n t_{mid}(X) + X^{2n} t_{hi}(X)$$

Compute $[t_{lo}]_1 := [\mathsf{t_{lo}}(x)]_1, [t_{mid}]_1 := [\mathsf{t_{mid}}(x)]_1, [t_{hi}]_1 := [\mathsf{t_{hi}}(x)]_1$

Third output of $\mathbf{P}$ is $([t_{lo}]_1, [t_{mid}]_1, [t_{hi}]_1)$

**Round 4:**
Compute evaluation challenge $\mathfrak{z} \in \mathbb{F}_p$ :

$$\mathfrak{z} = H(\mathsf{transcript})$$

Compute opening evaluations:

$$\bar{a} = \mathsf{a}(\mathfrak{z}), \bar{b} = \mathsf{b}(\mathfrak{z}), \bar{c} = \mathsf{c}(\mathfrak{z}), \bar{\mathsf{s}}_{\sigma 1} = \mathsf{S}_{\sigma 1}(\mathfrak{z}), \bar{\mathsf{s}}_{\sigma 2} = \mathsf{S}_{\sigma 2}(\mathfrak{z}),$$
$$\bar{t} = \mathsf{t}(\mathfrak{z}), \bar{z}_\omega = \mathsf{z}(\mathfrak{z}\omega)$$

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

$$\begin{aligned}
&\mathsf{r}(X) = \\
&\quad \big(\bar{a}\bar{b} \cdot \mathsf{q}_\mathsf{M}(X) + \bar{a} \cdot \mathsf{q}_\mathsf{L}(X) + \bar{b} \cdot \mathsf{q}_\mathsf{R}(X) + \bar{c} \cdot \mathsf{q}_\mathsf{O}(X) + \mathsf{q}_\mathsf{C}(X)\big) \\
&+ \big((\bar{a} + \beta\mathfrak{z} + \gamma)(\bar{b} + \beta k_1 \mathfrak{z} + \gamma)(\bar{c} + \beta k_2 z + \gamma) \cdot \mathsf{z}(X)\big)\, \alpha \\
&- \big((\bar{a} + \beta\bar{\mathsf{s}}_{\sigma 1} + \gamma)(\bar{b} + \beta\bar{\mathsf{s}}_{\sigma 2} + \gamma)\beta\bar{z}_\omega \cdot \mathsf{S}_{\sigma 3}(X)\big)\, \alpha \\
&+ (\mathsf{z}(X))\, \mathsf{L}_1(z)\alpha^2
\end{aligned}$$

Compute linearisation evaluation $\bar{r} = \mathsf{r}(\mathfrak{z})$

Fourth output of $\mathbf{P}$ is $(\bar{a}, \bar{b}, \bar{c}, \bar{\mathsf{s}}_{\sigma 1}, \bar{\mathsf{s}}_{\sigma 2}, \bar{z}_\omega, \bar{t}, \bar{r})$

**Round 5:**
Compute opening challenge $v \in \mathbb{F}_p$ :

$$v = H(\mathsf{transcript})$$

Compute opening proof polynomial $\mathsf{W}_\mathfrak{z}(X)$ :

$$\mathsf{W}_\mathfrak{z}(X) = \frac{1}{X - \mathfrak{z}} \begin{pmatrix} (\mathsf{t}_\mathsf{lo}(X) + \mathfrak{z}^n \mathsf{t}_\mathsf{mid}(X) + \mathfrak{z}^{2n} \mathsf{t}_\mathsf{hi}(X) - \bar{t}) \\ +v(\mathsf{r}(X) - \bar{r}) \\ +v^2(\mathsf{a}(X) - \bar{a}) \\ +v^3(\mathsf{b}(X) - \bar{b}) \\ +v^4(\mathsf{c}(X) - \bar{c}) \\ +v^5(\mathsf{S}_{\sigma 1}(X) - \bar{\mathsf{s}}_{\sigma 1}) \\ +v^6(\mathsf{S}_{\sigma 2}(X) - \bar{\mathsf{s}}_{\sigma 2}) \end{pmatrix}$$

Compute opening proof polynomial $\mathsf{W}_{\mathfrak{z}\omega}(X)$ :

$$\mathsf{W}_{\mathfrak{z}\omega}(X) = \frac{(\mathsf{z}(X) - \bar{z}_\omega)}{X - \mathfrak{z}\omega}$$

Compute $[W_\mathfrak{z}]_1 := [\mathsf{W}_\mathfrak{z}(x)]_1$ , $[W_{\mathfrak{z}\omega}]_1 := [\mathsf{W}_{\mathfrak{z}\omega}(x)]_1$

The fifth output of $\mathbf{P}$ is $([W_{\mathfrak{z}}]_1, [W_{\mathfrak{z}\omega}]_1)$

Return

$$\pi_{\mathsf{SNARK}} = \left( \begin{array}{c} [a]_1, [b]_1, [c]_1, [z]_1, [t_{lo}]_1, [t_{mid}]_1, [t_{hi}]_1, [W_{\mathfrak{z}}]_1, [W_{\mathfrak{z}\omega}]_1, \\ \bar{a}, \bar{b}, \bar{c}, \bar{\mathsf{s}}_{\sigma 1}, \bar{\mathsf{s}}_{\sigma 2}, \bar{r}, \bar{z}_{\omega} \end{array} \right)$$

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

$$u = H(\mathsf{transcript})$$

We now describe the verifier algorithm in a way that minimizes the number of $\mathbb{G}_1$ scalar multiplications.

**Verifier algorithm**

**Verifier preprocessed input:**

$[q_{\mathsf{M}}]_1 := \mathsf{q}_{\mathsf{M}}(x) \cdot [1]_1, [q_{\mathsf{L}}]_1 := \mathsf{q}_{\mathsf{L}}(x) \cdot [1]_1, [q_{\mathsf{R}}]_1 := \mathsf{q}_{\mathsf{R}}(x) \cdot [1]_1, [q_{\mathsf{O}}]_1 := \mathsf{q}_{\mathsf{O}}(x) \cdot [1]_1,$
$[s_{\sigma 1}]_1 := \mathsf{S}_{\sigma 1}(x) \cdot [1]_1, [s_{\sigma 2}]_1 := \mathsf{S}_{\sigma 2}(x) \cdot [1]_1, [s_{\sigma 3}]_1 := \mathsf{S}_{\sigma 3}(x) \cdot [1]_1$
$x \cdot [1]_2$

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

1. Validate $([a]_1, [b]_1, [c]_1, [z]_1, [t_{lo}]_1, [t_{mid}]_1, [t_{hi}]_1, [W_z]_1, [W_{z\omega}]_1) \in \mathbb{G}_1$.
2. Validate $(\bar{a}, \bar{b}, \bar{c}, \bar{\mathsf{s}}_{\sigma 1}, \bar{\mathsf{s}}_{\sigma 2}, \bar{r}, \bar{z}_{\omega}) \in \mathbb{F}_p^7$.
3. Validate $(w_i)_{i \in [\ell]} \in \mathbb{F}_p^{\ell}$.
4. Compute challenges $\beta, \gamma, \alpha, \mathfrak{z}, v, u \in \mathbb{F}_p$ as in prover description, from the common inputs, public input, and elements of $\pi_{\mathsf{SNARK}}$.
5. Compute the element $\Delta_{\mathrm{PI}}$ as in prover description.

$$\Delta_{\mathrm{PI}} := \frac{\prod_{i \in [\ell]} (w_i + \beta \sigma(i) + \gamma)}{\prod_{i \in [\ell]} (w_i + \beta \cdot \zeta_i + \gamma)}.$$

6. Compute zero polynomial evaluation $\mathsf{Z}_{\mathsf{H}}(\mathfrak{z}) = (\mathfrak{z}^n - 1)/(\mathfrak{z} - \mathbf{g}^n)$.
7. Compute Lagrange polynomial evaluations $\mathsf{L}_1(\mathfrak{z}) = \frac{\mathsf{Z}_{\mathsf{H}}(\mathfrak{z})}{n(\mathfrak{z}-1)}$; $\mathsf{L}_n(\mathfrak{z}) = \frac{\mathsf{Z}_{\mathsf{H}}(\mathfrak{z})}{n(\mathfrak{z}-\omega^{n-1})}$
8. Compute quotient polynomial evaluation

$$\bar{t} = \frac{\bar{r} - ((\bar{a} + \beta \bar{\mathsf{s}}_{\sigma 1} + \gamma)(\bar{b} + \beta \bar{\mathsf{s}}_{\sigma 2} + \gamma)(\bar{c} + \gamma)\bar{z}_{\omega})\alpha - \mathsf{L}_1(\mathfrak{z})\alpha^2 + (\bar{z}_{\omega} - \Delta_{\mathrm{PI}})\mathsf{L}_n(\mathfrak{z})\alpha^3}{\mathsf{Z}_{\mathsf{H}}(\mathfrak{z})}$$

.

9. Compute partial opening commitment $[D]_1 := v \cdot [r]_1 + u \cdot [z]_1$ :

$$\begin{array}{rl} & \bar{a}\bar{b}v \cdot [q_{\mathsf{M}}]_1 + \bar{a}v \cdot [q_{\mathsf{L}}]_1 + \bar{b}v \cdot [q_{\mathsf{R}}]_1 + \bar{c}v \cdot [q_{\mathsf{O}}]_1 + v \cdot [q_{\mathsf{C}}]_1 \\ [D]_1 := & + \big((\bar{a} + \beta\mathfrak{z} + \gamma)(\bar{b} + \beta k_{1}\mathfrak{z} + \gamma)(\bar{c} + \beta k_{2}\mathfrak{z} + \gamma)\alpha v + \mathsf{L}_1(\mathfrak{z})\alpha^2 v + u\big) \cdot [z]_1 \\ & -(\bar{a} + \beta\bar{\mathsf{s}}_{\sigma 1} + \gamma)(\bar{b} + \beta\bar{\mathsf{s}}_{\sigma 2} + \gamma)\alpha v \beta \bar{z}_{\omega} [s_{\sigma 3}]_1 \end{array}$$

10. Compute batch opening commitment $[F]_1$ :

$$\begin{array}{rl} [F]_1 := & [t_{lo}]_1 + \mathfrak{z}^n \cdot [t_{mid}]_1 + \mathfrak{z}^{2n} \cdot [t_{hi}]_1 \\ & + [D]_1 + v^2 \cdot [a]_1 + v^3 \cdot [b]_1 + v^4 \cdot [c]_1 + v^5 \cdot [s_{\sigma 1}]_1 + v^6 \cdot [s_{\sigma 2}]_1 \end{array}$$

11. Compute batch evaluation commitment $[E]_1$:

$$[E]_1 := \begin{pmatrix} \bar{t} + v\bar{r} + v^2\bar{a} + v^3\bar{b} + v^4\bar{c} \\ + v^5\bar{\mathsf{s}}_{\sigma 1} + v^6\bar{\mathsf{s}}_{\sigma 2} + u\bar{z}_1 \end{pmatrix} \cdot [1]_1$$

12. Validate

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

## Acknowledgements

We thank Mary Maller for telling us about the field element reduction method discussed in the end of Section **??**. We thank Vitalik Buterin for suggesting that we define the identity permutation using degree-1 polynomials, which reduces proof construction time, as well as reducing the proof size by 1 field element, and reduces the verification costs by 1 scalar multiplication. We thank Swastik Kopparty for pointing out an error in the permutation argument in a previous version of the paper, whose correction led to improved performance. We thank Justin Drake and Konstantin Panarin for discussions leading to corrections and simplifications of the permutation argument. We thank Sean Bowe, Alexander Vlasov and Kevaundray Wedderburn for comments and corrections. We thank the anonymous reviewers of SBC 2020 for their comments.

## 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. 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.

[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.

[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.

[GWC19] A. Gabizon, Z. J. Williamson, and O. Ciobotaru. PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *IACR Cryptology ePrint Archive*, 2019:953, 2019.

[KZG10] A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. 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 $\sigma$ 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 $\sigma$ of $[n]$ such that $b_i = a_{\sigma(i)}, \forall i \in [n]$. □

**Corollary A.4.** *Fix a permutation $\sigma$ 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 $\gamma$. 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 $b_i = a_{\sigma(i)}$, $\forall i \in [n]$. □