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

Ariel Gabizon   Zachary J. Williamson   Oana Ciobotaru

Protocol Labs   Aztec Protocol

Plonk in two sips

1. All you need is a permutation check.
2. Permutations are easier to check on mutliplicative subgroups

Part 1: All you need is a permutation check

Our setting: want short proofs about fan-in 2 unlimited fan-out circuits, trusted setup depends only on circuit size.

example: Prove knowledge of $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ with

$$(\mathbf{a} + \mathbf{b}) \cdot \mathbf{c} = 7$$

Left values: $\mathbf{l}_1, \mathbf{l}_2$
Right values: $\mathbf{r}_1, \mathbf{r}_2$
Output values: $\mathbf{o}_1, \mathbf{o}_2$

Left values: $\mathbf{l}_1, \mathbf{l}_2$
Right values: $\mathbf{r}_1, \mathbf{r}_2$
Output values: $\mathbf{o}_1, \mathbf{o}_2$

Gate checks: $\mathbf{l}_1 + \mathbf{r}_1 = \mathbf{o}_1, \mathbf{l}_2 \cdot \mathbf{r}_2 = \mathbf{o}_2$
Wire/copy checks: $\mathbf{o}_1 = \mathbf{l}_2$
Public input checks: $\mathbf{o}_2 = 7$.

Left values: $\mathbf{l}_1, \mathbf{l}_2$
Right values: $\mathbf{r}_1, \mathbf{r}_2$
Output values: $\mathbf{o}_1, \mathbf{o}_2$

Gate checks: $\mathbf{l}_1 + \mathbf{r}_1 = \mathbf{o}_1, \mathbf{l}_2 \cdot \mathbf{r}_2 = \mathbf{o}_2$ **(easy)**
Wire/copy checks: $\mathbf{o}_1 = \mathbf{l}_2$ **(hard)**
Public input checks: $\mathbf{o}_2 = 7$ **(easy)**

Copy checks with permutations

similar to [Groth09,BCGGHJ17]

$$\mathbf{V} = (\mathbf{l}_1, \mathbf{l}_2, \mathbf{r}_1, \mathbf{r}_2, \mathbf{o}_1, \mathbf{o}_2)$$

Copy checks with permutations

similar to [Groth09,BCGGHJ17]

$$\mathbf{V} = (\mathbf{l}_1, \mathbf{l}_2, \mathbf{r}_1, \mathbf{r}_2, \mathbf{o}_1, \mathbf{o}_2)$$

$\mathbf{o}_1 = \mathbf{l}_2$ iff $\mathbf{V} = \boldsymbol{\sigma}(\mathbf{V})$
For permutation $\boldsymbol{\sigma} = (25)$

Part 2: Permutations are easier to check on mutliplicative subgroups

[Bayer-Groth12] - perm checks with products

example: Given $\mathbf{a}, \mathbf{b} \in \mathbb{F}^3$, want to check $(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3) = (\mathbf{a}_3, \mathbf{a}_1, \mathbf{a}_2)$

[Bayer-Groth12] - perm checks with products

example: Given $\mathbf{a}, \mathbf{b} \in \mathbb{F}^3$, want to check $(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3) = (\mathbf{a}_3, \mathbf{a}_1, \mathbf{a}_2)$

step 1: Choose random $\boldsymbol{\beta} \in \mathbb{F}$. Let

$$\mathbf{a}_1' = \mathbf{a}_1 + \boldsymbol{\beta}, \mathbf{a}_2' = \mathbf{a}_2 + 2\boldsymbol{\beta}, \mathbf{a}_3' = \mathbf{a}_3 + 3\boldsymbol{\beta}$$

$$\mathbf{b}_1' = \mathbf{b}_1 + 3\boldsymbol{\beta}, \mathbf{b}_2' = \mathbf{b}_2 + \boldsymbol{\beta}, \mathbf{b}_3' = \mathbf{b}_3 + 2\boldsymbol{\beta}$$

[Bayer-Groth12] - perm checks with products

example: Given $\mathbf{a}, \mathbf{b} \in \mathbb{F}^3$, want to check $(\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3) = (\mathbf{a}_3, \mathbf{a}_1, \mathbf{a}_2)$

step 1: Choose random $\boldsymbol{\beta} \in \mathbb{F}$. Let

$$\mathbf{a}'_1 = \mathbf{a}_1 + \boldsymbol{\beta}, \mathbf{a}'_2 = \mathbf{a}_2 + 2\boldsymbol{\beta}, \mathbf{a}'_3 = \mathbf{a}_3 + 3\boldsymbol{\beta}$$

$$\mathbf{b}'_1 = \mathbf{b}_1 + 3\boldsymbol{\beta}, \mathbf{b}'_2 = \mathbf{b}_2 + \boldsymbol{\beta}, \mathbf{b}'_3 = \mathbf{b}_3 + 2\boldsymbol{\beta}$$

If claim false - w.h.p as mutliset $\{\mathbf{a}'_1, \mathbf{a}'_2, \mathbf{a}'_3\} \neq \{\mathbf{b}'_1, \mathbf{b}'_2, \mathbf{b}'_3\}$:

[Bayer-Groth12] - reducing permutation checks to products

step 2: Choose random $\gamma \in \mathbb{F}$. Let

$$a_i'' = a_i' + \gamma, b_i'' = b_i + \gamma$$

[Bayer-Groth12] - reducing permutation checks to products

step 2: Choose random $\gamma \in \mathbb{F}$. Let

$$a''_i = a'_i + \gamma, b''_i = b_i + \gamma$$

If $\{a'_1, a'_2, a'_3\} \neq \{b'_1, b'_2, b'_3\}$ as multiset - w.h.p

$$a''_1 \cdot a''_2 \cdot a''_3 \neq b''_1 \cdot b''_2 \cdot b''_3.$$

Idealized Polynomials Protocols

Preprocessing: $\mathcal{V}$ chooses polynomials $\mathbf{g}_1, \ldots, \mathbf{g}_t \in \mathbb{F}_{<\mathbf{d}}[\mathbf{X}]$.

Protocol:

1. $\mathcal{P}$'s msgs are to ideal party $\mathbf{I}$. Must be $\mathbf{f}_\mathbf{i} \in \mathbb{F}_{<\mathbf{d}}[\mathbf{X}]$.
2. At protocol end $\mathcal{V}$ asks $\mathbf{I}$ if some identities hold between $\{\mathbf{f}_1, \ldots, \mathbf{f}_\ell, \mathbf{g}_1, \ldots, \mathbf{g}_t\}$. Outputs $\mathbf{acc}$ iff they do.

Idealized Polynomials Protocols

Preprocessing: $\mathcal{V}$ chooses polynomials $\mathbf{g}_1, \ldots, \mathbf{g}_\mathbf{t} \in \mathbb{F}_{<\mathbf{d}}[\mathbf{X}]$.

Protocol:

1. $\mathcal{P}$'s msgs are to ideal party $\mathbf{I}$. Must be $\mathbf{f}_\mathbf{i} \in \mathbb{F}_{<\mathbf{d}}[\mathbf{X}]$.
2. At end, $\mathcal{V}$ asks $\mathbf{I}$ if some identities hold between $\{\mathbf{f}_1, \ldots, \mathbf{f}_\ell, \mathbf{g}_1, \ldots, \mathbf{g}_\mathbf{t}\}$.

Using [KZG10], can compile to real protocol with each msg of $\mathcal{P}$ being 32-64 bytes according to your NFSPL.

$\mathbf{H}$-ranged Polynomials Protocols

Preprocessing: $\mathcal{V}$ chooses polynomials $\mathbf{g}_1, \ldots, \mathbf{g}_t \in \mathbb{F}_{<\mathbf{d}}[\mathbf{X}]$, $\mathbf{H} \subset \mathbb{F}$.

Protocol:

1. $\mathcal{P}$'s msgs are to ideal party $\mathbf{I}$. Must be $\mathbf{f}_\mathbf{i} \in \mathbb{F}_{<\mathbf{d}}[\mathbf{X}]$.
2. At end, $\mathcal{V}$ asks $\mathbf{I}$ if some identities hold between $\{\mathbf{f}_1, \ldots, \mathbf{f}_\ell, \mathbf{g}_1, \ldots, \mathbf{g}_t\}$ **on** $\mathbf{H}$.

$\mathbf{H}$-ranged protocol using polynomial protocol:

$\mathcal{V}$ wants to check identities $\mathbf{P}_1, \mathbf{P}_2$ on $\mathbf{H}$.

- After $\mathcal{P}$ finished sending $\{\mathbf{f_i}\}$, $\mathcal{V}$ sends random $\mathbf{a}_1, \mathbf{a}_2 \in \mathbb{F}$.
- $\mathcal{P}$ sends $\mathbf{T} \in \mathbb{F}_{<\mathbf{d}}[\mathbf{X}]$.
- $\mathcal{V}$ checks identity $\mathbf{a}_1 \cdot \mathbf{P}_1 + \mathbf{a}_2 \cdot \mathbf{P}_2 \equiv \mathbf{T} \cdot \mathbf{Z}_{\mathbf{H}}$.

Checking permutations with $\mathbf{H}$-ranged protocols

Permutation $\sigma : [n] \to [n]$. $H = \{\alpha, \alpha^2, \ldots, \alpha^n\}$.

$\mathcal{P}$ has sent $f \in \mathbb{F}_{<d}[X]$.

Wants to prove $f = \sigma(f)$:

$$\forall i \in [n], f(\alpha^i) = f(\alpha^{\sigma(i)})$$

Using [BG12] reduces to:

$\mathbf{H} = \left\{\alpha, \alpha^2, \ldots, \alpha^n\right\}$.

$\mathcal{P}$ has sent $f, g \in \mathbb{F}_{<d}[X]$.

Wants to prove:

$$\prod_{i \in [n]} f(\alpha^i) = \prod_{i \in [n]} g(\alpha^i)$$

Checking products with $\mathbb{H}$-ranged protocols

1. $\mathcal{P}$ computes Z_f with $Z_f(\alpha) = 1, Z_f(\alpha^i) = \prod_{j<i} f(\alpha^j)$, $i = 2..n+1$. Computes Z_g similarly.
2. Sends Z_f, Z_g to $\mathbf{I}$.

Checking products with $\mathbf{H}$-ranged protocols

1. $\mathcal{P}$ computes $\mathbf{Z_f}$ with $\mathbf{Z_f}(\alpha) = 1, \mathbf{Z_f}(\alpha^i) = \prod_{j<i} \mathbf{f}(\alpha^j)$. Computes $\mathbf{Z_g}$ similarly.
2. Sends $\mathbf{Z_f}, \mathbf{Z_g}$ to $\mathbf{I}$.
3. $\mathcal{V}$ checks following identities on $\mathbf{H}$.
   3.1 $\mathbf{L_1}(\mathbf{X})(\mathbf{Z_f}(\mathbf{X}) - \mathbf{Z_g}(\mathbf{X})) = 0$
   3.2 $\mathbf{Z_f}(\mathbf{X})\mathbf{f}(\mathbf{X}) = \mathbf{Z_f}(\alpha \cdot \mathbf{X})$
   3.3 $\mathbf{Z_g}(\mathbf{X})\mathbf{f}(\mathbf{X}) = \mathbf{Z_g}(\alpha \cdot \mathbf{X})$
   3.4 $\mathbf{L_n}(\mathbf{X})(\mathbf{Z_f}(\alpha \cdot \mathbf{X}) - \mathbf{Z_g}(\alpha \cdot \mathbf{X})) = 0$

The bottom line (on BLS-381 curve)

700 byte proofs with one trusted setup for all fan-in two circuits of $\mathbf{n}$ gates.

Prover does $12\mathbf{n}\ \mathbf{G}_1$ exp.

For batch of proofs on same circuit only $3\mathbf{n}\ \mathbf{G}_1$ exp and 240 bytes for each additional proof.

Bonus material: The KZG polynomial commitment scheme

SRS: $[1]$,$[x]$,. . . ,$\left[x^d\right]$, for random $x \in \mathbb{F}$.

$f(X) = \sum_{i=0}^{d} a_i X^i$

$\mathrm{cm}(f) := \sum_{i=0}^{d} a_i \left[x^i\right] = [f(x)]$

SRS: $[1]$,$[x]$,$\ldots$, $\left[x^{d}\right]$,
for random $x \in \mathbb{F}$.

$\mathsf{cm}(f) := [f(x)]$

$\mathsf{open}(f, i) := [h(x)]$, where $h(X) := \frac{f(X)-f(i)}{X-i}$

$\mathsf{cm}(\mathbf{f}) := [\mathbf{f}(\boldsymbol{x})]$

$\mathsf{open}(\mathbf{f}, \mathbf{i}) := [\mathbf{h}(\boldsymbol{x})]$, where $\mathbf{h}(\mathbf{X}) := \frac{\mathbf{f}(\mathbf{X}) - \mathbf{f}(\mathbf{i})}{\mathbf{X} - \mathbf{i}}$

$\mathsf{verify}(\mathsf{cm}, \boldsymbol{\pi}, \boldsymbol{z}, \mathbf{i}) :$

$$\boldsymbol{e}(\mathsf{cm} - [\boldsymbol{z}], [1]) \overset{?}{=} \boldsymbol{e}(\boldsymbol{\pi}, [\boldsymbol{x} - \mathbf{i}])$$

$\mathsf{cm}(\mathbf{f}) := [\mathbf{f}(x)]$

$\mathsf{open}(\mathbf{f}, \mathbf{i}) := [\mathbf{h}(x)]$, where $\mathbf{h}(\mathbf{X}) := \frac{\mathbf{f}(\mathbf{X})-\mathbf{f}(\mathbf{i})}{\mathbf{X}-\mathbf{i}}$

$\mathsf{verify}(\mathsf{cm}, \boldsymbol{\pi}, z, \mathbf{i})$:

$$e(\mathsf{cm} - [z], [1]) \overset{?}{=} e(\boldsymbol{\pi}, [x - \mathbf{i}])$$

Thm[KZG,MBKM]: *This works in the Algebraic Group*

Model.