Combining PLONK with Plookup

Author 1
Affiliation 1

Author 2
Affiliation 2

Author 3
Affiliation 3

May 2021

Abstract

In 2019, Gabizon, Williamson, and Ciobotaru introduced Plonk – a fast and flexible zk-SNARK using an updatable and universal structured reference string. Plonk uses a *grand product argument* to check permutations of wire values and exploits convenient interactions between multiplicative subgroups and Lagrange bases. The next year, Gabizon and Williamson used similar techniques to develop Plookup – a zk-SNARK that can verify that each element in a list of queries can be found in a public lookup table. These two zk-SNARK constructions were presented seperately, but are similar enough to be combined into a single protocol. Here we demonstrate a combined Plonk and Plookup protocol, noting implementation details and justifying deviations from the original protocols required to combine them effectively.

1 Introduction

Plonk uses constraints of the form:

$$(q_L)_i \cdot a_i + (q_R)_i \cdot b_i + (q_O)_i \cdot c_i + (q_M)_i \cdot a_i b_i + (q_C)_i = 0$$

where the vectors $\mathbf{q_L}, \mathbf{q_R}, \mathbf{q_O}, \mathbf{q_M}$, and $\mathbf{q_C}$ are *selectors* which can scale their term in the constraint (or "turn off" that term by using a selector value of zero). The vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ are *wires* whose values at index i need to satisfy the ith constraint.

A variable often needs to be used in multiple constraints, for instance the "out" value of constraint i may need to be the "left" value in constraint j, i.e. $c_i = a_j$. These *copy constraints* are enforced by a *grand product argument* constructed using additional vectors representing permutations of variables.

2 Terminology and Notation

We use notation similar to GWC19, so that $\mathbb{F}$ is a field of prime order, $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_t$ are groups of order r, and e is an efficiently computable non-degenerate pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$. There are generators for these groups, g_1, g_2, and g_t, such that $e(g_1, g_2) = g_t$.

A *wire* is a vector of field elements providing the values of variables in each gate. Here four wires are used, labeled "left", "right", and "out". However, any number of wires can be used. In our implementation we use these same three wires to carry data for lookup queries so that our lookup

table can represent a function of up to three inputs. A *selector* is also a vector of field elements which serve as coefficients to the variables in each gate. A value of 0 at index i in the multiplication selector means the multiplication term in gate i is not being used (i.e. it is some other kind of gate).

3 Implementation Details

3.1 Options for Hiding

In GWC19, hiding of the wire polynomials and the permutation polynomial is accomplished by adding a distinct random multiple of the target polynomial Z_H to each polynomial that needs to be hidden. The random factors are either degree 1 or 2 and so the hidden polynomials have slightly higher degree than the unhidden polynomials.

We found an alternative approach using randomized "dummy" values added to each wire. This approach provides the same hiding as the method in GWC19, but is more useful in a context where padding the wires is usually necessary. The dummy values can also be used to pad the circuit.

[[elaborate]]

3.2 Connecting Plookup to a Plonk Circuit

For our needs, the Plookup and Plonk portions of the protocol needed to be able to refer to the same variables, so that the output variable of a Plonk gate could be an input to a Plookup gate and vice-versa. A simple way to achieve this is to use the same wires for both Plonk and Plookup gates. Then a *selector* can be used to select only Plookup gates when necessary.

3.3 Padding for Sorting

In GW20, the vectors $f \in \mathbb{F}^{n-1}$ and $t \in \mathbb{F}^n$, containing the compressed lookup queries and the compressed lookup table rows respectively, are concatenated and sorted into a vector $s \in \mathbb{F}^{2n-1}$. Then s is divided into two equal size parts, $h_1 \in \mathbb{F}^n$ and $h_2 \in \mathbb{F}^n$, so that the last (and largest) element of h_1 is the same as the first (and smallest) element of h_2. Then f, t, h_1, and h_2 are interpolated into polynomials and used to create the plookup permutation polynomial.

Both f and t may need to be padded in order to get them to be the correct size.

However, if either f or t contains many repeats of the same value (say a certain query is repeated many times, or the lookup table is small and needs to be padded with many copies of a dummy row), and that value is very small or very large, it can happen that h_1 or h_2 consists entirely of the repeated element. When this vector is interpolated the resulting polynomial will be constant.

[[Is this bad, apart from Dusk's code throwing an error? Will Kate commitments fail? Security broken?]]

To prevent this, care must be taken to pad f and t with a field element that is neither the largest nor the smallest of either.

3.4 Constructing a Linearization Polynomial

3.5 The Query Polynomial

The vector f contains the compressed queries from the wires. But f is supposed to be length $n-1$ while each wire is supposed to be of length n. Some circuit row needs to be excluded when defining

4 Combined Protocol

4.1 Polynomials defining a circuit including Plookup

- The selectors $q_M(X), q_L(X), q_R(X), q_O(X), q_C(X)$ which define the arithmetization of a Plonk circuit
- The additional selector $q_K(X)$ which selects Plookup gates
- $S_{ID1}(X) = X, S_{ID2}(X) = k_1(X), S_{ID2}(X) = k_2X$: the identity permutation applied to **a**, **b**, **c**, with constants $k_1, k_2 \in \mathbb{F}$ chosen such that $H, k_1 \cdot H, k_2 \cdot H$ are distinct cosets in $mathbbF*$.
- $S_{\sigma_1}(X)$, $S_{\sigma_2}(X)$, $S_{\sigma_3}(X)$: The copy permutation applied to **a**, **b** and **c**.
- n, the number of gates for a given circuit (the sum total of arithmetic and lookup gates)

4.2 Commitments

4.3 Preprocessing

4.4 Prover algorithm

Round 1

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

$$a(X) = (b_1X + b_2)Z_H(X) + \sum_{i=0}^{n-1} w_i \mathbf{L}_i(X)$$

$$b(X) = (b_3X + b_4)Z_H(X) + \sum_{i=0}^{n-1} w_{n+i} \mathbf{L}_i(X)$$

$$c(X) = (b_5X + b_6)Z_H(X) + \sum_{i=0}^{n-1} w_{2n+i} \mathbf{L}_i(X)$$

3. Compute $[a]_1$, $[b]_1$, and $[c]_1$ and add to transcript.
4. Generate table compression factor ζ from the transcript.
5. Compute compressed table $\{t_i\}_{i=0}^{n}$ as $t_i = x_{t_i} + \zeta y_{t_i} + \zeta^2 z_{t_i}$ and sort.
6. Compute table polynomial $t(X) = \sum_{i=0}^{n-1} t_i \mathbf{L}_i(X)$.

7. Compute compressed queries $\{f_i\}_{i=0}^{n}$ as $f_i = x_{f_i} + \zeta y_{f_i} + \zeta^2 z_{f_i}$.
8. Compute query polynomial $f(X) = \sum_{i=0}^{n-1} f_i \mathbf{L}_i(X)$.
9. Compute $[f]_1 = [f(x)]_1$.

Output $([a]_1, [b]_1, [c]_1, [f]_1)$.

Round 2

1. Compute permutation challenges $(\beta, \gamma, \delta, \varepsilon) \in \mathbb{F}_p$ from the transcript.
2. Compute permutation polynomial $z(X)$:

$$\begin{aligned} z(X) = {} & (b_7 X^2 + b_8 X + b_9) Z_H(X) + \mathbf{L}_0(X) \\ & + \sum_{i=0}^{n-2} \left(\mathbf{L}_{i+1}(X) \prod_{j=1}^{i} \frac{(w_j + \beta\omega^{j-1} + \gamma)(w_{n+j} + \beta k_1 \omega^{j-1} + \gamma)(w_{2n+j} + \beta k_2 \omega^{j-1} + \gamma)}{(w_j + \beta\sigma(j) + \gamma)(w_{n+j} + \beta\sigma(n+j) + \gamma)(w_{2n+j} + \beta\sigma(2n+j) + \gamma)} \right) \end{aligned}$$

3. Compute $\{s_i\}_{i=0}^{2n-1}$, the sorted concatenation of (f, t).
4. Compute the Plookup polynomial $p(X)$:

$$p(X) = \mathbf{L}_0(X) + \sum_{i=1}^{n-2} \left(\mathbf{L}_i(X) \prod_{j=0}^{i-1} \frac{(\varepsilon + f_j)(1+\delta)(\varepsilon(1+\delta) + t_j + \delta t_{j+1})}{(\varepsilon(1+\delta) + h_{1j} + \delta h_{1j+1})(\varepsilon(1+\delta) + h_{2j} + \delta h_{2j+1})} \right) + \mathbf{L}_{n-1}(X)$$

5. Compute $[z]_1 = [z(x)]_1$, $[p]_1 = [p(x)]_1$, $[h_1]_1 = [h_1(x)]_1$, $[h_2]_1 = [h_2(x)]_1$.

Output $([z]_1, [p]_1, [h_1]_1, [h_2]_1)$.

Round 3

1. Compute challenges $\alpha_0, \alpha_1 \in \mathbb{F}_p$ from the transcript.
2. Compute the quotient polynomial $Q(X)$:

$$Q(X) = \frac{1}{Z_H(X)} \left(\begin{array}{l} (a(X)b(X)q_M(X) + a(X)q_L(X) + b(X)q_R(X) + c(X)q_O(X) + PI(X) + q_C(X)) \\ +(a(X) + \beta X + \gamma)(b(X) + \beta k_1 X + \gamma)(c(X) + \beta k_2 X + \gamma) z(X) \alpha_0 \\ -(a(X) + \beta S_{\sigma 1}(X) + \gamma)(b(X) + \beta S_{\sigma 2}(X) + \gamma)(c(X) + \beta S_{\sigma 3}(X) + \gamma) z(X\omega) \alpha_0 \\ +(z(X) - 1)\mathbf{L}_0(X)\alpha_0^2 \\ +q_{Lookup}(X)(a(X) + \zeta b(X) + \zeta^2 c(X) - f(X))\alpha_1 \\ +(p(X) - 1)\mathbf{L}_0(X)\alpha_1^2 \\ +(X - \omega^{-1}) p(X)(1+\delta)(\varepsilon + f(X))(\varepsilon(1+\delta) + t(X) + \delta t(X\omega))\alpha_1^3 \\ -(X - \omega^{-1}) p(X\omega)(\varepsilon(1+\delta) + h_1(X) + \delta h_1(X\omega))(\varepsilon(1+\delta) + h_2(X) + \delta h_2(X\omega))\alpha_1^3 \\ +\mathbf{L}_{n-1}(X)(h_1(X) - h_2(X\omega))\alpha_1^4 \\ +\mathbf{L}_{n-1}(X)(p(X) - 1)\alpha_1^5 \end{array} \right)$$

3. Split $Q(X)$ into degree $\leq n+2$ polynomials $Q_{\text{lo}}(X)$, $Q_{\text{mid}}(X)$, $Q_{\text{hi}}(X)$

Output $([Q_{\text{lo}}]_1, [Q_{\text{mid}}]_1, [Q_{\text{hi}}]_1)$.

Round 4

1. Compute evaluation challenge $\mathfrak{z}$ from the transcript.
2. Compute opening evaluations:

$$\begin{aligned}
&\bar{a} = a(\mathfrak{z}), \bar{b} = b(\mathfrak{z}), \bar{c} = c(\mathfrak{z}), \bar{S}_{\sigma 1} = S_{\sigma 1}(\mathfrak{z}), \bar{S}_{\sigma 2} = S_{\sigma 2}(\mathfrak{z}),\\
&\bar{Q} = Q(\mathfrak{z}), \bar{z}_\omega = z(\mathfrak{z}\omega), \bar{p}_\omega = p(\mathfrak{z}\omega), \bar{f} = f(\mathfrak{z}), \bar{t} = t(\mathfrak{z}), \bar{t}_\omega = t(\mathfrak{z}\omega),\\
&\bar{h}_1 = h_1(\mathfrak{z}), \bar{h}_{1_\omega} = h_1(\mathfrak{z}\omega), \bar{h}_{2_\omega} = h_2(\mathfrak{z}\omega)
\end{aligned}$$

3. Compute linearization polynomial $r(X)$:

$$\begin{aligned}
r(X) = &\\
&\bar{a}\bar{b} \cdot q_M(X) + \bar{a} \cdot q_L(X) + \bar{b} \cdot q_R(X) + \bar{c} \cdot q_O(X) + q_C(X)\\
&+ ((\bar{a} + \beta\mathfrak{z} + \gamma)(\bar{b} + \beta k_1\mathfrak{z} + \gamma)(\bar{c} + \beta k_2\mathfrak{z} + \gamma)z(X))\alpha_0\\
&- ((\bar{a} + \beta\bar{s}_{\sigma 1} + \gamma)(\bar{b} + \beta\bar{s}_{\sigma 2} + \gamma)\beta\bar{z}_\omega \cdot S_{\sigma 3}(X))\alpha_0\\
&+ z(X)\mathbf{L}_0(\mathfrak{z})\alpha_0^2\\
&- q_{Lookup}(X)\bar{f}\alpha_1\\
&+ p(X)\mathbf{L}_0(\mathfrak{z})\alpha_1^2\\
&+ (\mathfrak{z} - \omega^{-1})p(X)(1+\delta)(\varepsilon + \bar{f})(\varepsilon(1+\delta) + \bar{t} + \delta\bar{t}_\omega)\alpha_1^3\\
&- (\mathfrak{z} - \omega^{-1})\bar{p}_\omega(\varepsilon(1+\delta) + \bar{h}_1 + \delta\bar{h}_{1_\omega})h_2(X)\alpha_1^3\\
&+ \mathbf{L}_{n-1}(\mathfrak{z})h_1(X)\alpha_1^4\\
&+ \mathbf{L}_{n-1}(\mathfrak{z})p(X)\alpha_1^5
\end{aligned}$$

4. Compute $\bar{r} = r(\mathfrak{z})$.

Output $\bar{r}$.

Round 5

1. Compute opening challenge $v \in \mathbb{F}_p$ from the transcript.
2. Compute opening proof polynomial $W_{\mathfrak{z}}(X)$:

$$W_{\mathfrak{z}}(X) = \frac{1}{X - \mathfrak{z}} \begin{pmatrix}
(Q_{\text{lo}}(X) + \mathfrak{z}^{n+2}Q_{\text{mid}}(X) + \mathfrak{z}^{2n+4}Q_{\text{hi}}(X) - \bar{Q})\\
+v(r(X) - \bar{r})\\
+v^2(a(X) - \bar{a})\\
+v^3(b(X) - \bar{b})\\
+v^4(c(X) - \bar{c})\\
+v^5(S_{\sigma 1}(X) - \bar{s}_{\sigma 1})\\
+v^6(S_{\sigma 2}(X) - \bar{s}_{\sigma 2})\\
+v^7(f(X) - \bar{f})\\
+v^8(h_1(X) - \bar{h}_1)
\end{pmatrix}$$

3. Compute opening proof polynomial $W_{\mathfrak{z}\omega}(X)$:

$$W_{\mathfrak{z}\omega}(X) = \frac{1}{X - \mathfrak{z}\omega} \begin{pmatrix} z(X) - \bar{z}_\omega \\ +v'(h_1(X) - \bar{h}_{1\omega}) \\ +v'^2(h_2(X) - \bar{h}_{2\omega}) \\ +v'^3(p(X) - \bar{p}_\omega) \end{pmatrix}$$

4. Compute $[W_{\mathfrak{z}}]_1$ and $[W_{\mathfrak{z}\omega}]_1$.

Output $([W_{\mathfrak{z}}]_1, [W_{\mathfrak{z}\omega}]_1)$.

The full proof is:

$$\pi_{\text{SNARK}} = \begin{pmatrix} [a]_1, [b]_1, [c]_1, [f]_1, [z]_1, [p]_1, [h_1]_1, [h_2]_1, [Q_{\text{lo}}]_1, [Q_{\text{mid}}]_1, [Q_{\text{hi}}]_1, [W_{\mathfrak{z}}]_1, [W_{\mathfrak{z}\omega}]_1, \\ \bar{a}, \bar{b}, \bar{c}, \bar{s}_{\sigma 1}, \bar{s}_{\sigma 2}, \bar{r}, \bar{z}_\omega, \bar{p}_\omega, \bar{h}_1, \bar{h}_{1_\omega}, \bar{h}_{2_\omega}, \bar{f} \end{pmatrix}$$

4.5 Verifier Algorithm

Verifier preprocessed input:

$[q_M]_1 := q_M(x) \cdot [1]_1, [q_L]_1 := q_L(x) \cdot [1]_1, [q_R]_1 := q_R(x) \cdot [1]_1, [q_O]_1 := q_O(x) \cdot [1]_1, [q_{\text{Lookup}}]_1 := q_{\text{Lookup}}(x) \cdot [1]_1, [s_{\sigma 1}]_1 := s_{\sigma 1}(x) \cdot [1]_1, [s_{\sigma 2}]_1 := s_{\sigma 2}(x) \cdot [1]_1, [s_{\sigma 3}]_1 := s_{\sigma 1}(3) \cdot [1]_1, [t]_1 := t(x) \cdot [1]_1, x \cdot [1]_2$

1. Validate $([a]_1, [b]_1, [c]_1, [z]_1, [p]_1, [h_1]_1, [h_2]_1, [Q_{\text{lo}}]_1, [Q_{\text{mid}}]_1, [Q_{\text{hi}}]_1, [W_{\mathfrak{z}}]_1, [W_{\mathfrak{z}\omega}]_1) \in \mathbb{G}_1^{11}$.
2. Validate $(\bar{a}, \bar{b}, \bar{c}, \bar{z}, \bar{s}_{\sigma 1}, \bar{s}_{\sigma 2}, \bar{r}, \bar{z}_\omega, \bar{p}_\omega, \bar{h}_1, \bar{h}_{1_\omega}, \bar{h}_{2_\omega}, \bar{f}) \in \mathbb{F}_p^{13}$.
3. Validate $(w_i)_{i \in [l]} \in \mathbb{F}_p^l$.
4. Compute challenges $\beta, \gamma, \delta, \varepsilon, \alpha_0, \alpha_1, \mathfrak{z}, v, v', u \in \mathbb{F}_p$.
5. Compute zero polynomial evaluation $Z_H(\mathfrak{z}) = \mathfrak{z}^n - 1$.
6. Compute Lagrange polynomial evaluations $\mathbf{L}_0(\mathfrak{z}) = \frac{\mathfrak{z}^n - 1}{n(\mathfrak{z} - 1)}$ and $\mathbf{L}_{n-1}(\mathfrak{z}) = \frac{\mathfrak{z}^n - 1}{n(\mathfrak{z}\omega - 1)}$.
7. Compute public input polynomial evaluation $PI(\mathfrak{z}) = \sum_{i \in l} w_i \mathbf{L}_i(\mathfrak{z})$, public table polynomial evaluations $\bar{t} = t(\mathfrak{z})$ and $\bar{t}_\omega = t(\mathfrak{z}\omega)$, and public selector polynomial evaluation $\bar{q}_{\text{Lookup}} = q_{\text{Lookup}}(\mathfrak{z})$.
8. Compute the quotient polynomial evaluation:

$$\bar{Q} = \frac{1}{Z_H(\mathfrak{z})} \begin{pmatrix} \bar{r} + PI(\mathfrak{z}) - ((\bar{a} + \beta \bar{s}_{\sigma 1} + \gamma)(\bar{b} + \beta \bar{s}_{\sigma 2} + \gamma)(\bar{c} + \gamma)\bar{z}_\omega)\alpha_0 \\ -\mathbf{L}_0(\mathfrak{z})\alpha_0^2 + (\bar{q}_{\text{Lookup}})\,(\bar{a} + \zeta \bar{b} + \zeta^2 \bar{c})\alpha_1 - \mathbf{L}_0(\mathfrak{z})\alpha_1^2 \\ -(\mathfrak{z} - \omega^{-1})\bar{p}_\omega(\varepsilon(1+\delta) + \bar{h}_1 + \delta \bar{h}_{1_\omega})(\varepsilon(1+\delta) + \delta \bar{h}_{2_\omega})\alpha_1^3 \\ -\mathbf{L}_{n-1}(\mathfrak{z})\bar{h}_{2_\omega}\alpha_1^4 - \mathbf{L}_{n-1}(\mathfrak{z})\alpha_1^5 \end{pmatrix}$$

9. Compute the first part of the batched polynomial commitment

$[D]_1 := v \cdot [r]_1 + u \cdot \big([z]_1 + v'[h_1]_1 + v'^2[h_2]_1 + v'^3[p]_1\big)$:

$$
\begin{aligned}
[D]_1 = {} & \bar{a}\bar{b}v \cdot [q_M]_1 + \bar{a}v \cdot [q_L]_1 + \bar{b}v \cdot [q_R]_1 + \bar{c}v \cdot [q_O]_1 + v \cdot [q_C]_1 - \bar{f}v\alpha_1 \cdot [q_{\text{Lookup}}]_1 \\
& + \big((\bar{a} + \beta\mathfrak{z} + \gamma)(\bar{b} + \beta k_1\mathfrak{z} + \gamma)(\bar{c} + \beta k_2\mathfrak{z} + \gamma)\alpha v + \mathbf{L}_0(\mathfrak{z})\alpha^2 v + u\big) \cdot [z]_1 \\
& - (\bar{a} + \beta\bar{s}_{\sigma 1} + \gamma)(\bar{b} + \beta\bar{s}_{\sigma 2} + \gamma)\alpha v\beta\bar{z}_\omega[s_{\sigma 3}]_1 \\
& + \big(\mathbf{L}_{n-1}(\mathfrak{z})\alpha_1^4 v + uv'\big) \cdot [h_1]_1 \\
& - \big((\mathfrak{z} - \omega^{-1})\bar{p}_\omega(\varepsilon(1+\delta) + \bar{h}_1 + \delta\bar{h}_{1_\omega})\alpha_1^3 v + uv'^2\big) \cdot [h_2]_1 \\
& + \big((\mathfrak{z} - \omega^{-1})(1+\delta)(\varepsilon + \bar{f})(\varepsilon(1+\delta) + \bar{t} + \delta\bar{t}_\omega)\alpha_1^3 v + \mathbf{L}_0(\mathfrak{z})\alpha_1^2 v + \mathbf{L}_{n-1}(\mathfrak{z})\alpha_1^5 v + uv'^3\big) \cdot [p]_1
\end{aligned}
$$

10. Compute full batched polynomial commitment $[F]_1$:

$$
\begin{aligned}
[F]_1 := {} & [Q_{\text{lo}}]_1 + \mathfrak{z}^{n+2} \cdot [Q_{\text{mid}}]_1 + \mathfrak{z}^{2n+4} \cdot [Q_{\text{hi}}]_1 \\
& + [D]_1 + v^2 \cdot [a]_1 + v^3 \cdot [b]_1 + v^4 \cdot [c]_1 + v^5 \cdot [s_{\sigma 1}]_1 + v^6 \cdot [s_{\sigma 2}]_1 \\
& + v^7 \cdot [f]_1 + v^8 \cdot [h_1]_1
\end{aligned}
$$

11. Compute group-encoded batch evaluation $[E]_1$:

$$
[E]_1 := \begin{pmatrix} \bar{Q} + v\bar{r} + v^2\bar{a} + v^3\bar{b} + v^4\bar{c} + v^5\bar{s}_{\sigma 1} + v^6\bar{s}_{\sigma 2} + v^7\bar{f} + v^8\bar{h}_1 \\ +u(\bar{z}_\omega + v'\bar{h}_{1_\omega} + v'^2\bar{h}_{2_\omega} + v'^3\bar{p}_\omega) \end{pmatrix} \cdot [1]_1
$$

12. Batch validate all evaluations:

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