

dusk-plonk: an implementation of $\mathcal{P}\text{lonk}$ with custom gates

Dusk Team

August 6, 2024

Abstract

`dusk-plonk` is an implementation of $\mathcal{P}\text{lonk}$ made by the Dusk team, which deviates from the original paper to use custom gates [1]. This document serves the purpose of describing the main differences between the implementation and the original protocol. Note that it is not self-explanatory, so refer to the original paper [2] for further details.

The protocol

We now describe the protocol as done in the original paper, adding in red the changes applied in `dusk-plonk`.

Common preprocessed input:

$$\begin{aligned}
& n, (x \cdot [1]_1, \dots, x^{n+5} \cdot [1]_1), \\
& (q_{Mi}, q_{Li}, q_{Ri}, q_{Oi}, \textcolor{red}{q_{Fi}}, q_{Ci}, \textcolor{red}{q_{arithi}}, \textcolor{red}{q_{rangei}}, \textcolor{red}{q_{logici}}, \textcolor{red}{q_{fixedi}}, \textcolor{red}{q_{vari}})_{i=1}^n, \sigma^*, \\
& q_M(X) = \sum_{i=1}^n q_{Mi} L_i(X), \\
& q_L(X) = \sum_{i=1}^n q_{Li} L_i(X), \\
& q_R(X) = \sum_{i=1}^n q_{Ri} L_i(X), \\
& q_O(X) = \sum_{i=1}^n q_{Oi} L_i(X), \\
& \textcolor{red}{q_F(X)} = \sum_{i=1}^n \textcolor{red}{q_{Fi}} L_i(X), \\
& q_C(X) = \sum_{i=1}^n q_{Ci} L_i(X), \\
& \textcolor{red}{q_{arith}(X)} = \sum_{i=1}^n \textcolor{red}{q_{arithi}} L_i(X), \\
& \textcolor{red}{q_{range}(X)} = \sum_{i=1}^n \textcolor{red}{q_{rangei}} L_i(X), \\
& \textcolor{red}{q_{logic}(X)} = \sum_{i=1}^n \textcolor{red}{q_{logici}} L_i(X), \\
& \textcolor{red}{q_{fixed}(X)} = \sum_{i=1}^n \textcolor{red}{q_{fixedi}} L_i(X), \\
& \textcolor{red}{q_{var}(X)} = \sum_{i=1}^n \textcolor{red}{q_{vari}} L_i(X), \\
& S_{\sigma_1}(X) = \sum_{i=1}^n \sigma^*(i) L_i(X), \\
& S_{\sigma_2}(X) = \sum_{i=1}^n \sigma^*(n+i) L_i(X), \\
& S_{\sigma_3}(X) = \sum_{i=1}^n \sigma^*(2n+i) L_i(X), \\
& \textcolor{red}{S_{\sigma_4}(X)} = \sum_{i=1}^n \textcolor{red}{\sigma^*(3n+i)} L_i(X)
\end{aligned}$$

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

Prover algorithm:

Prover input: $(w_i)_{i \in [4n]}$

Round 1:

Generate random blinding scalars $(b_1, \dots, b_{11}) \in \mathbb{F}$

Compute wire polynomials $a(X), b(X), c(X), \textcolor{red}{d(X)}$:

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

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

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

$$\textcolor{red}{d(X)} = (b_7 X + b_8) Z_H(X) + \sum_{i=1}^n \textcolor{red}{w_{3n+i}} L_i(X)$$

Compute $[a]_1 := [a(x)]_1, [b]_1 := [b(x)]_1, [c]_1 := [c(x)]_1, [d]_1 := [d(x)]_1$

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

Round 2:

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

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

Compute permutation polynomial $z(X)$:

$$z(X) = (b_9 X^2 + b_{10} X + b_{11}) Z_H(X) + L_1(X) + \sum_{i=1}^{n-1} \left(L_{i+1}(X) \prod_{j=1}^i \frac{(w_j + \beta \omega^j + \gamma)(w_{n+j} + \beta k_1 \omega^j + \gamma)(w_{2n+j} + \beta k_2 \omega^j + \gamma)(w_{3n+j} + \beta k_3 \omega^j + \gamma)}{(w_j + \sigma^*(j)\beta + \gamma)(w_{n+j} + \sigma^*(n+j)\beta + \gamma)(w_{2n+j} + \sigma^*(2n+j)\beta + \gamma)(w_{3n+j} + \sigma^*(3n+j)\beta + \gamma)} \right)$$

Compute $[z]_1 := [z(x)]_1$

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

Round 3:

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

$$\alpha = \mathcal{H}(\text{transcript})$$

Compute custom challenges $s \in \mathbb{F}$:

$$s_{\text{range}} = \mathcal{H}(\text{transcript})$$

$$s_{\text{logic}} = \mathcal{H}(\text{transcript})$$

$$s_{\text{fixed}} = \mathcal{H}(\text{transcript})$$

$$s_{\text{var}} = \mathcal{H}(\text{transcript})$$

Compute the quotient selector polynomials:

$$t_{\text{arith}}(X) = a(X)b(X)q_M(X) + a(X)q_L(X) + b(X)q_R(X) + c(X)q_O(X) + d(X)q_F(X) + q_C(X)$$

$$t_{\text{range}}(X) = TBC$$

$$t_{\text{logic}}(X) = TBC$$

$$t_{\text{fixed}}(X) = TBC$$

$$t_{\text{var}}(X) = TBC$$

Compute quotient polynomial $t(X)$:

$$\begin{aligned}
t(X) = & t_{\text{arith}}(X) * q_{\text{arith}}(X) \\
& + t_{\text{range}}(X) * q_{\text{range}}(X) * s_{\text{range}} \\
& + t_{\text{logic}}(X) * q_{\text{logic}}(X) * s_{\text{logic}} \\
& + t_{\text{fixed}}(X) * q_{\text{fixed}}(X) * s_{\text{fixed}} \\
& + t_{\text{var}}(X) * q_{\text{var}}(X) * s_{\text{var}} \\
& + \text{PI}(X) \frac{1}{Z_H(X)} \\
& + ((a(X) + \beta X + \gamma)(b(X) + \beta k_1 X + \gamma)(c(X) + \beta k_2 X + \gamma)(d(X) + \beta k_3 X + \gamma)z(X)) \frac{\alpha}{Z_H(X)} \\
& - ((a(X) + \beta S_{\sigma 1}(X) + \gamma)(b(X) + \beta S_{\sigma 2}(X) + \gamma)(c(X) + \beta S_{\sigma 3}(X) + \gamma)(d(X) + \beta S_{\sigma 4}(X) + \gamma)z(X\omega)) \frac{\alpha}{Z_H(X)} \\
& + (z(X) - 1) L_1(X) \frac{\alpha^2}{Z_H(X)}
\end{aligned}$$

Split $t(X)$ into degree $< n$ polynomials $t'_{\text{lo}}(X)$, $t'_{\text{mid}}(X)$, $t'_{\text{hi}}(X)$ and $t'_{\text{fourth}}(X)$ of degree at most $n + 5$, such that

$$t(X) = t'_{\text{lo}}(X) + X^n t'_{\text{mid}}(X) + X^{2n} t'_{\text{hi}}(X) + X^{3n} t'_{\text{fourth}}(X)$$

Now choose random scalars $b_{12}, b_{13}, b_{14} \in \mathbb{F}$ and define

$$\begin{aligned}
t_{\text{lo}}(X) &:= t'_{\text{lo}}(X) + b_{12} X^n, t_{\text{mid}}(X) := t'_{\text{mid}}(X) - b_{12} + b_{13} X^n, t_{\text{hi}}(X) := t'_{\text{hi}}(X) - b_{13} + b_{14} X^n \\
t_{\text{fourth}}(X) &:= t'_{\text{fourth}}(X) - b_{14}
\end{aligned}$$

Note that we have $t(X) = t_{\text{lo}}(X) + X^n t_{\text{mid}}(X) + X^{2n} t_{\text{hi}}(X) + X^{3n} t_{\text{fourth}}(X)$.

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

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

Round 4:

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

$$\mathfrak{z} = \mathcal{H}(\text{transcript})$$

Compute opening evaluations:

$$\begin{aligned}
\bar{a} &= a(\mathfrak{z}), \bar{b} = b(\mathfrak{z}), \bar{c} = c(\mathfrak{z}), \bar{d} = d(\mathfrak{z}), \bar{s}_{\sigma 1} = S_{\sigma 1}(\mathfrak{z}), \bar{s}_{\sigma 2} = S_{\sigma 2}(\mathfrak{z}), \\
\bar{s}_{\sigma 3} &= S_{\sigma 3}(\mathfrak{z}), \bar{z}_{\omega} = z(\mathfrak{z}\omega), \bar{a}_{\omega} = a(\mathfrak{z}\omega), \bar{b}_{\omega} = b(\mathfrak{z}\omega), \bar{d}_{\omega} = d(\mathfrak{z}\omega), \bar{q}_{\text{arith}} = q_{\text{arith}}(\mathfrak{z}), \\
\bar{q}_{\text{C}} &= q_{\text{C}}(\mathfrak{z}), \bar{q}_{\text{L}} = q_{\text{L}}(\mathfrak{z}), \bar{q}_{\text{R}} = q_{\text{R}}(\mathfrak{z})
\end{aligned}$$

Fourth output of \mathbf{P} is $(\bar{a}, \bar{b}, \bar{c}, \bar{d}, \bar{s}_{\sigma 1}, \bar{s}_{\sigma 2}, \bar{s}_{\sigma 3}, \bar{z}_{\omega}, \bar{a}_{\omega}, \bar{b}_{\omega}, \bar{d}_{\omega}, \bar{q}_{\text{arith}}, \bar{q}_{\text{C}}, \bar{q}_{\text{L}}, \bar{q}_{\text{R}})$

Round 5:

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

$$v = \mathcal{H}(\text{transcript})$$

Compute the linearisation selector polynomials:

$$r_{\text{arith}}(X) = \bar{a}\bar{b} \cdot q_M(X) + \bar{a} \cdot q_L(X) + \bar{b} \cdot q_R(X) + \bar{c} \cdot q_O(X) + \bar{d} \cdot q_F(X) + q_C(X)$$

$$r_{\text{range}}(X) = TBC$$

$$r_{\text{logic}}(X) = TBC$$

$$r_{\text{fixed}}(X) = TBC$$

$$r_{\text{var}}(X) = TBC$$

Compute linearisation polynomial $r(X)$:

$$\begin{aligned} r(X) = & [r_{\text{arith}}(X) * \bar{q}_{\text{arith}} \\ & + r_{\text{range}}(X) * q_{\text{range}}(X) \\ & + r_{\text{logic}}(X) * q_{\text{logic}}(X) \\ & + r_{\text{fixed}}(X) * q_{\text{fixed}}(X) \\ & + r_{\text{var}}(X) * q_{\text{var}}(X) \\ & + \text{Pl}(\mathfrak{z})] \\ & + \alpha[(\bar{a} + \beta\mathfrak{z} + \gamma)(\bar{b} + \beta k_1\mathfrak{z} + \gamma)(\bar{c} + \beta k_2\mathfrak{z} + \gamma)(\bar{d} + \beta k_3\mathfrak{z} + \gamma) \cdot z(X) \\ & - (\bar{a} + \beta \bar{s}_{\sigma 1} + \gamma)(\bar{b} + \beta \bar{s}_{\sigma 2} + \gamma)(\bar{c} + \beta \bar{s}_{\sigma 3} + \gamma) \bar{z}_{\omega} \beta S_{\sigma 4}(X)] \\ & + \alpha^2 [(z(X) - 1)L_1(\mathfrak{z})] \\ & - Z_H(\mathfrak{z}) \cdot (t_{lo}(X) + \mathfrak{z}^n t_{mid}(X) + \mathfrak{z}^{2n} t_{hi}(X) + \mathfrak{z}^{3n} t_{fourth}(X)) \end{aligned}$$

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

$$W_{\mathfrak{z}}(X) = \frac{1}{X - \mathfrak{z}} \begin{pmatrix} r(X) \\ +v(a(X) - \bar{a}) \\ +v^2(b(X) - \bar{b}) \\ +v^3(c(X) - \bar{c}) \\ +v^4(d(X) - \bar{d}) \\ +v^5(S_{\sigma 1}(X) - \bar{s}_{\sigma 1}) \\ +v^6(S_{\sigma 2}(X) - \bar{s}_{\sigma 2}) \\ +v^7(S_{\sigma 3}(X) - \bar{s}_{\sigma 3}) \end{pmatrix}$$

Compute shifted opening challenge $v_{\omega} \in \mathbb{F}$:

$$v_{\omega} = \mathcal{H}(\text{transcript})$$

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

$$W_{\mathfrak{z}\omega}(X) = \frac{1}{X - \mathfrak{z}\omega} \begin{pmatrix} (z(X) - \bar{z}_\omega) \\ + v_\omega(\mathbf{a}(X) - \bar{a}_\omega) \\ + v_\omega^2(\mathbf{b}(X) - \bar{b}_\omega) \\ + v_\omega^3(\mathbf{d}(X) - \bar{d}_\omega) \end{pmatrix}$$

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

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

Return

$$\pi_{\text{SNARK}} = \left([a]_1, [b]_1, [c]_1, [\mathbf{d}]_1, [z]_1, [t_{lo}]_1, [t_{mid}]_1, [t_{hi}]_1, [\mathbf{t}_{fourth}]_1, [W_{\mathfrak{z}}]_1, [W_{\mathfrak{z}\omega}]_1, \right. \\ \left. \bar{a}, \bar{b}, \bar{c}, \bar{d}, \bar{a}_\omega, \bar{b}_\omega, \bar{d}_\omega, \bar{q}_{arith}, \bar{q}_L, \bar{q}_R, \bar{q}_C, \bar{s}_{\sigma 1}, \bar{s}_{\sigma 2}, \bar{s}_{\sigma 3}, \bar{z}_\omega \right)$$

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

$$u = \mathcal{H}(\text{transcript})$$

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

Verifier algorithm

Verifier preprocessed input:

$$[q_M]_1 := \mathbf{q}_M(x) \cdot [1]_1, [q_L]_1 := \mathbf{q}_L(x) \cdot [1]_1, [q_R]_1 := \mathbf{q}_R(x) \cdot [1]_1, [q_O]_1 := \mathbf{q}_O(x) \cdot [1]_1, \\ [\mathbf{q}_F]_1 := \mathbf{q}_F(x) \cdot [1]_1, [q_C]_1 := \mathbf{q}_C(x) \cdot [1]_1, [s_{\sigma 1}]_1 := S_{\sigma 1}(x) \cdot [1]_1, [s_{\sigma 2}]_1 := S_{\sigma 2}(x) \cdot [1]_1, \\ [s_{\sigma 3}]_1 := S_{\sigma 3}(x) \cdot [1]_1, [\mathbf{s}_{\sigma 4}]_1 := \mathbf{S}_{\sigma 4}(x) \cdot [1]_1, x \cdot [1]_2$$

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

1. Validate $([a]_1, [b]_1, [c]_1, [\mathbf{d}]_1, [z]_1, [t_{lo}]_1, [t_{mid}]_1, [t_{hi}]_1, [\mathbf{t}_{fourth}]_1, [W_{\mathfrak{z}}]_1, [W_{\mathfrak{z}\omega}]_1) \in \mathbb{G}_1^9$.
2. Validate $(\bar{a}, \bar{b}, \bar{c}, \bar{d}, \bar{a}_\omega, \bar{b}_\omega, \bar{d}_\omega, \bar{q}_{arith}, \bar{q}_L, \bar{q}_R, \bar{q}_C, \bar{s}_{\sigma 1}, \bar{s}_{\sigma 2}, \bar{s}_{\sigma 3}, \bar{z}_\omega) \in \mathbb{F}^6$.
3. Validate $(w_i)_{i \in [\ell]} \in \mathbb{F}^\ell$.
4. Compute challenges $\beta, \gamma, \alpha, \mathbf{s}_{range}, \mathbf{s}_{logic}, \mathbf{s}_{fixed}, \mathbf{s}_{var}, \mathfrak{z}, v, v_\omega, u \in \mathbb{F}$ as in prover description, from the common inputs, public input, and elements of π_{SNARK} .
5. Compute zero polynomial evaluation $Z_H(\mathfrak{z}) = \mathfrak{z}^n - 1$.
6. Compute Lagrange polynomial evaluation $L_1(\mathfrak{z}) = \frac{\omega(\mathfrak{z}^n - 1)}{n(\mathfrak{z} - \omega)}$.

7. Compute public input polynomial evaluation $\text{PI}(\mathfrak{z}) = \sum_{i \in [\ell]} w_i \text{L}_i(\mathfrak{z})$.
8. To save a verifier scalar multiplication, we split r into its constant and non-constant terms. Compute r 's constant term:

$$r_0 := \text{PI}(\mathfrak{z}) - \text{L}_1(\mathfrak{z})\alpha^2 - \alpha(\bar{a} + \beta\bar{s}_{\sigma 1} + \gamma)(\bar{b} + \beta\bar{s}_{\sigma 2} + \gamma)(\bar{c} + \beta\bar{s}_{\sigma 3} + \gamma)(\bar{d} + \gamma)\bar{z}_{\omega},$$

and let $r'(X) := r(X) - r_0$.

9. Compute first part of batched polynomial commitment $[D]_1 := [r']_1 + u \cdot [z]_1$:

$$\begin{aligned} & \bar{a}\bar{b} \cdot \bar{q}_{\text{arith}} \cdot [q_M]_1 + \bar{a} \cdot \bar{q}_{\text{arith}} \cdot [q_L]_1 + \bar{b} \cdot \bar{q}_{\text{arith}} \cdot [q_R]_1 + \bar{c} \cdot \bar{q}_{\text{arith}} \cdot [q_O]_1 \\ & + \bar{d} \cdot \bar{q}_{\text{arith}} \cdot [q_F]_1 + \bar{q}_{\text{arith}} \cdot [q_C]_1 \\ & + \text{range} : TBC \\ & + \text{logic} : TBC \\ [D]_1 := & + \text{fixed} : TBC \\ & + \text{var} : TBC \\ & + ((\bar{a} + \beta\mathfrak{z} + \gamma)(\bar{b} + \beta k_{1\mathfrak{z}} + \gamma)(\bar{c} + \beta k_{2\mathfrak{z}} + \gamma)(\bar{d} + \beta k_{3\mathfrak{z}} + \gamma)\alpha + \text{L}_1(\mathfrak{z})\alpha^2 + u) \cdot [z]_1 \\ & - (\bar{a} + \beta\bar{s}_{\sigma 1} + \gamma)(\bar{b} + \beta\bar{s}_{\sigma 2} + \gamma)(\bar{c} + \beta\bar{s}_{\sigma 3} + \gamma)\alpha\beta\bar{z}_{\omega} \cdot [s_{\sigma 4}]_1 \\ & - Z_H(\mathfrak{z})([t_{lo}]_1 + \mathfrak{z}^n \cdot [t_{mid}]_1 + \mathfrak{z}^{2n} \cdot [t_{hi}]_1 + \mathfrak{z}^{3n} \cdot [t_{fourth}]_1) \end{aligned}$$

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

$$\begin{aligned} [F]_1 := & [D]_1 + v \cdot [a]_1 + v^2 \cdot [b]_1 + v^3 \cdot [c]_1 + v^4 \cdot [d]_1 + v^5 \cdot [s_{\sigma 1}]_1 + v^6 \cdot [s_{\sigma 2}]_1 \\ & + v^7 \cdot [s_{\sigma 3}]_1 + (uv_{\omega}) \cdot [a]_1 + (uv_{\omega}^2) \cdot [b]_1 + (uv_{\omega}^3) \cdot [d]_1 \end{aligned}$$

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

$$[E]_1 := \begin{pmatrix} -r_0 + v\bar{a} + v^2\bar{b} + v^3\bar{c} + v^4\bar{d} \\ + v^5\bar{s}_{\sigma 1} + v^6\bar{s}_{\sigma 2} + v^7\bar{s}_{\sigma 3} + u\bar{z}_{\omega} \\ + (uv_{\omega})\bar{a}_{\omega} + (uv_{\omega}^2)\bar{b}_{\omega} + (uv_{\omega}^3)\bar{d}_{\omega} \end{pmatrix} \cdot [1]_1$$

12. Batch validate all evaluations:

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

References

- [1] GABIZON, A., AND WILLIAMSON, Z. J. Proposal: The turbo-plonk program syntax for specifying snark programs. <https://docs.zkproof.org/pages/standards/accepted-workshop3/proposal-turbo-plonk.pdf>.
- [2] GABIZON, A., WILLIAMSON, Z. J., AND CIOBOTARU, O. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Paper 2019/953, 2019. <https://eprint.iacr.org/2019/953>.