The Missing Protocol PH23-PCS (Part 1)

In the paper "Improving logarithmic derivative lookups using GKR" ([PH23]), the authors presented an idea to convert MLE into a Univariate Polynomial. Although the paper didn't provide a complete protocol description, this protocol demonstrates advantages in certain aspects, such as supporting Shift Arguments with arbitrary offsets.

The main advantage of this scheme is its ability to support Shift Arguments with arbitrary offsets (see Appendix A.2 of the paper). Additionally, when interfacing with KZG10, the proof of this PCS Adaptor only includes a constant number of $\mathbb{G}_1$ elements and a logarithmic number of $\mathbb{F}_r$ elements. This is superior to Gemini-PCS and Zeromorph-PCS (KT23), which require a logarithmic number of $\mathbb{G}_1$ elements.

The approach of this protocol is similar to Virgo-PCS in that they both view MLE polynomial operations as a summation and use the Univariate Sumcheck protocol to complete the "sum proof". However, PH23-PCS also requires the Prover to prove the value of the MLE Lagrange Polynomial at the evaluation point, thus reducing the burden on the Verifier; while Virgo-PCS uses the GKR protocol to achieve this. Another difference is that Virgo-PCS requires the MLE polynomial to be represented in Coefficient Form, so Virgo-PCS uses the GKR circuit to prove the correctness of the computation of converting the MLE polynomial from Evaluation Form to Coefficient Form.

This article series completes the description of PH23-PCS in the [PH23] paper and provides a simplified protocol for PH23-KZG10 to help everyone understand the basic idea of this protocol.

This article first introduces the basic principles of PH23-PCS-Adaptor in detail, and then provides a simple protocol implementation of PH23-KZG10.

1. Principle Overview

Before explaining how the Prover proves the Evaluation of an MLE polynomial $\tilde{f}(\vec{X})$, let's recall the definition of an MLE polynomial:

$$\tilde{f}(X_0, X_1, \ldots, X_{n-1}) = \sum_{i=0}^{N-1} a_i \cdot \tilde{eq}(\mathsf{bits}(i), (X_0, X_1, \ldots, X_{n-1})) \tag{1}$$

Here $N = 2^n$. When calculating the value of $\tilde{f}(\vec{X})$ at $\vec{X} = (u_0, u_1, \ldots, u_{n-1})$, we need to calculate $\sum_{i=0}^{N-1} a_i \cdot \tilde{eq}(\mathsf{bits}(i), \vec{u})$. To facilitate explanation, we introduce a new vector $\vec{c}$, where each element c_i = $\tilde{eq}(\mathsf{bits}(i), \vec{u})$.

If $n = 3$ and $N = 8$, then all values of $\vec{c}$ can be enumerated:

$$\begin{array}{ccccc}
c_0 & = & (1-u_0) & (1-u_1) & (1-u_2) \\
c_1 & = & u_0 & (1-u_1) & (1-u_2) \\
c_2 & = & (1-u_0) & u_1 & (1-u_2) \\
c_3 & = & u_0 & u_1 & (1-u_2) \\
c_4 & = & (1-u_0) & (1-u_1) & u_2 \\
c_5 & = & u_0 & (1-u_1) & u_2 \\
c_6 & = & (1-u_0) & u_1 & u_2 \\
c_7 & = & u_0 & u_1 & u_2
\end{array} \tag{2}$$

It can be seen that the elements of $\vec{c}$ are defined with certain patterns. For example, c_i is the product of s values, and these values also have certain patterns. Here $(1-u_i)$ represents binary 0, while u_i represents binary 1. For instance, c_7 is the product of three numbers, u_0, u_1, u_2, which represents `(111)`, exactly the binary representation of 7. Another example is c_5, which is the product of three numbers, $u_0, (1-u_1), u_2$, representing `(101)`, which is the binary representation of 5.

The key idea of PH23 is whether the Prover can first commit to the vector $\vec{c}$, and then prove that each element of $\vec{c}$ is correctly defined according to the binary pattern above. If possible, the Prover can then prove an Inner Product relationship, i.e., prove $\langle\vec{a},\vec{c}\rangle = v$, which is equivalent to proving $\tilde{f}(\vec{X}) = v$.

Therefore, the proof protocol of PH23 is divided into two parts:

1. Prove the Well-Formedness of vector $\vec{c}$.
2. Prove $\langle\vec{a},\vec{c}\rangle = v$.

2. Well-Formedness of $\vec{c}$

Continuing with the example of $\vec{c}$ where $s=3$,

$$\begin{array}{ccccc}
c_0 & = & (1-u_0) & (1-u_1) & (1-u_2) \\
c_1 & = & u_0 & (1-u_1) & (1-u_2) \\
c_2 & = & (1-u_0) & u_1 & (1-u_2) \\
c_3 & = & u_0 & u_1 & (1-u_2) \\
c_4 & = & (1-u_0) & (1-u_1) & u_2 \\
c_5 & = & u_0 & (1-u_1) & u_2 \\
c_6 & = & (1-u_0) & u_1 & u_2 \\
c_7 & = & u_0 & u_1 & u_2
\end{array} \tag{3}$$

We observe that

$$\frac{c_0}{c_4} = \frac{1-u_2}{u_2} \tag{4}$$

Thus, if c_0 is correct, we can prove that c_4 is correct by proving the following constraint equation:

$$c_0 \cdot u_2 - c_4 \cdot (1-u_2) = 0 \tag{5}$$

Next, we observe

$$\frac{c_0}{c_2} = \frac{c_4}{c_6} = \frac{1-u_1}{u_1} \tag{6}$$

From this, we can infer that if c_0 is correct, then the following two constraint equations ensure that c_2 and c_6 are correct:

$$\begin{aligned}
c_0 \cdot u_1 - c_2 \cdot (1-u_1) = 0 \\
c_4 \cdot u_1 - c_6 \cdot (1-u_1) = 0
\end{aligned} \tag{7}$$

Next, we can prove that c_1, c_3, c_5, c_7 are correct because they can be derived from c_0, c_2, c_4, c_6, which have been proven correct in the previous step:

$$\begin{aligned} c_0 \cdot u_0 - c_1 \cdot (1-u_0) &= 0 \\ c_2 \cdot u_0 - c_3 \cdot (1-u_0) &= 0 \\ c_4 \cdot u_0 - c_5 \cdot (1-u_0) &= 0 \\ c_6 \cdot u_0 - c_7 \cdot (1-u_0) &= 0 \end{aligned} \tag{8}$$

The final conclusion is that through the above $1+2+4$ constraint equations, we can prove that $c_1, c_2, c_3, c_4, c_5, c_6, c_7$ are all correct, assuming c_0 is known to be correct. The inference relationship between the elements of vector $\vec{c}$ is shown in the following diagram:

$$\begin{array}{llll} c_4 & & & \\ c_2 & c_6 & & \\ c_1 & c_3 & c_5 & c_7 \\ \cdots & & & \end{array} \tag{9}$$

Assume H is a multiplicative subgroup of size 8 in the finite field $\mathbb{F}_p$, $H = \{1, \omega, \omega^2, \omega^3, \omega^4, \omega^5, \omega^6, \omega^7\}$, where $\omega \in \mathbb{F}_p$ is an 8th root of unity. And let $\{L_i(X)\}_{i=0}^{N-1}$ denote the Lagrange Basis polynomials on H.

Then we can introduce $c(X)$ as the polynomial encoding of $\vec{c}$ according to the Lagrange Basis:

$$c(X) = \sum_{i=0}^{N-1} c_i \cdot L_i(X) \tag{10}$$

It's easy to verify that $c(\omega^i) = c_i$, where $i = 0, 1, 2, \ldots, N-1$.

Furthermore, the four constraint equations proving c_1, c_3, c_5, c_7 can be combined into one polynomial constraint equation:

$$(X-\omega)(X-\omega^3)(X-\omega^5)(X-\omega^7) \cdot \big(c(X)u_0 - c(\omega \cdot X)(1-u_0)\big) = 0, \quad X \in H \tag{11}$$

We can substitute $X = \omega^2$, and the above constraint equation corresponds to:

$$c(\omega^2) \cdot u_0 - c(\omega^3) \cdot (1-u_0) = c_2 \cdot u_0 - c_3 \cdot (1-u_0) = 0 \tag{12}$$

By substituting $X = \omega, X = \omega^4, X = \omega^6$ respectively, we can obtain the constraint equations proving the correctness of c_1, c_5, c_7.

The polynomial $(X-\omega)(X-\omega^3)(X-\omega^5)(X-\omega^7)$ looks like a Selector polynomial, filtering out X values that don't satisfy the condition.

Using this method, we can use $n = \log N$ polynomial constraints to prove the Well-Formedness of $\vec{c}$.

For the example of $N = 8$, we need to introduce 3 Selector polynomials $s_0(X), s_1(X), s_2(X)$,

$$s_i(X) = \frac{v_H(X)}{v_{H_i}(X)}, \qquad i = 0, 1, 2 \tag{13}$$

where $v_H(X)$ and $v_{H_i}(X)$ are the Vanishing polynomials of Domain H and H_i respectively. And H_i is a subgroup of H, satisfying the following Group Tower relationship:

$$\{1\} = H_0 \subset H_1 \subset H_2 \subset H_3 = H \tag{14}$$

They are defined as follows:

$$
\begin{aligned}
H = H_3 &= (1, \omega, \omega^2, \omega^3, \omega^4, \omega^5, \omega^6, \omega^7) \\
H_2 &= (1, \omega^2, \omega^4, \omega^6) \\
H_1 &= (1, \omega^4) \\
H_0 &= (1)
\end{aligned} \tag{15}
$$

Naturally, the representations of Selector polynomials $s_0(X), s_1(X), s_2(X)$ are as follows:

$$
\begin{aligned}
s_0(X) &= (X-\omega)(X-\omega^2)(X-\omega^3)(X+1)(X+\omega)(X+\omega^2)(X+\omega^3) \\
s_1(X) &= (X-\omega)(X-\omega^2)(X-\omega^3)(X+\omega)(X+\omega^2)(X+\omega^3) \\
s_2(X) &= (X-\omega)(X-\omega^3)(X+\omega)(X+\omega^3)
\end{aligned} \tag{16}
$$

Polynomial Constraint Equations

The constraint equation ensuring the correctness of c_0 can be expressed as the following polynomial constraint:

$$
s_0(X) \cdot \big(c(X) - (1-u_0)(1-u_1)(1-u_2)\big) = 0, \quad X \in H \tag{17}
$$

The constraint equation ensuring the correctness of c_4 can be expressed as the following polynomial constraint:

$$
s_0(X) \cdot \big(c(X)u_2 - c(\omega^4 \cdot X)(1-u_2)\big) = 0, \quad X \in H \tag{18}
$$

The following are the constraint equations ensuring the correctness of c_2, c_6:

$$
s_1(X) \cdot \big(c(X)u_1 - c(\omega^2 \cdot X)(1-u_1)\big) = 0, \quad X \in H \tag{19}
$$

Finally, the constraint equation ensuring the correctness of c_1, c_3, c_5, c_7:

$$
s_2(X) \cdot \big(c(X)u_0 - c(\omega \cdot X)(1-u_0)\big) = 0, \quad X \in H \tag{20}
$$

3. Proving Inner Product

The second part of the proof is to prove $\langle \vec{a}, \vec{c} \rangle = v$. Assuming $a(X)$ is the encoding of vector $\vec{a}$, i.e., $a(X) \mid_H = \vec{a}$, then $a(X)$ is committed as $[a(\tau)]_1$, along with the commitment of $c(X)$, $[c(\tau)]_1$, we can use the Univariate Sumcheck protocol to prove the inner product.

Univariate Sumcheck

Let's first look at a theorem (Remark 5.6 in [BCRSVW19], Sec.3 in [RZ21], Sec.5.1 in [CHMMVW19]): For any $P(X) \in \mathbb{F}[X]$, a multiplicative subgroup $H \subset \mathbb{F}$, $P(X)$ can be decomposed as:

$$
P(X) = q(X) \cdot v_H(X) + X \cdot g(X) + (v/N) \tag{21}
$$

Here v is the sum of $P(X)$ over H, i.e.,

$$
\sum_{\omega \in H} P(\omega) = v \tag{22}
$$

Therefore, we can use this theorem to prove the inner product of two vectors. If $a(X) \cdot c(X)$ can be expressed as the following equation,

$$
a(X) \cdot c(X) = q(X) \cdot v_H(X) + X \cdot g(X) + (v/N), \quad \deg(g) < N-1 \tag{23}
$$

then $\langle\vec{a},\vec{c}\rangle = v$.

The Prover can send commitments of $q(X)$ and $g(X)$, then the Verifier challenges with ζ, the Prover sends the evaluations of related polynomials at $X=\zeta$, and then the Verifier verifies if the above equation holds:

$$a(\zeta)\cdot c(\zeta) \overset{?}{=} q(\zeta)\cdot v_H(\zeta) + \zeta\cdot g(\zeta) + (v/N) \tag{24}$$

The Prover and Verifier then use a univariate polynomial commitment scheme, such as KZG10, to prove the correctness of $a(\zeta), c(\zeta), q(\zeta), g(\zeta)$.

At the same time, the KZG10 protocol can also prove the Degree Bound of $g(X)$, i.e., $\deg(g(X)) < N-1$.

Grand Sum

In fact, we can also use the Grand Sum protocol to prove the inner product of two vectors.

One problem with Univariate Sumcheck is that it includes a Degree Bound constraint, i.e., $\deg(g(X)) < N-1$. This requires additional processing for the underlying KZG10 protocol, which increases the complexity of the protocol and introduces too many Pairing operations. The Grand Sum protocol avoids introducing Degree Bound constraints.

The idea of the Grand Sum protocol comes from the Grand Product Argument in Plonk [GWC19], and this protocol was first proposed by [BG12].

Suppose a polynomial $f(X)$ encodes the values of vector $\vec{f}$, then we construct an auxiliary vector $\vec{z}$, satisfying:

$$\begin{aligned}
z_0 &= a_0\cdot c_0 \\
z_1 &= z_0 + a_1\cdot c_1 \\
z_2 &= z_1 + a_2\cdot c_2 \\
\cdots \\
z_{N-1} &= z_{N-2} + a_{N-1}\cdot c_{N-1}
\end{aligned} \tag{25}$$

Or expressed more concisely with a recursive formula:

$$\begin{aligned}
z_0 &= a_0\cdot c_0 \\
z_i &= z_{i-1} + a_i\cdot c_i, \quad i = 1,\ldots,N-1
\end{aligned} \tag{26}$$

We can encode $\vec{z}$ using polynomial $z(X)$, i.e.,

$$z(X) = \sum_{i=0}^{N-1} z_i\cdot L_i(X) \tag{27}$$

where $L_i(X)$ are the Lagrange Basis polynomials for H defined above.

Then we can use the following three polynomial constraints to represent the recursive formula of z_i, thus ensuring the correctness of $z(X)$:

$$\begin{aligned}
L_0(X)\cdot\big(z(X) - a(X)\cdot c_0\big) &= 0 \\
(X-1)\cdot\big(z(X) - z(\omega^{-1}\cdot X) - a(X)\cdot c(X)\big) &= 0 \\
L_{N-1}(X)\cdot\big(z(X) - v\big) &= 0
\end{aligned} \tag{28}$$

Note that we use $z(\omega^{-1}\cdot X)$ to represent z_{i-1}. And the third polynomial constraint ensures that the sum of the result equals the final polynomial operation result v.

Using a Univariate PCS protocol, such as KZG10 or FRI, we can prove the correctness of these polynomial constraint equations.

4. Interfacing with KZG10

To prove the polynomial constraints listed above, we can implement a proof of these polynomial equations based on the KZG10 protocol. In summary, we have two types of polynomial constraints. The first type is constraints about the correctness of $c(X)$, and the second type is constraints about the correctness of $z(X)$.

$$
\begin{aligned}
p_0(X) =& s_0(X)\cdot\Big(c(X) - (1-u_0)(1-u_1)\cdots(1-u_{n-1})\Big)\\
p_1(X) =& s_0(X)\cdot\Big(c(X)u_{n-1} - c(\omega^{2^{n-1}}\cdot X)(1-u_{n-1})\Big)\\
p_2(X) =& s_1(X)\cdot\Big(c(X)u_{n-2} - c(\omega^{2^{n-2}}\cdot X)(1-u_{n-2})\Big)\\
\cdots\quad & \cdots\\
p_n(X) =& s_{n-1}(X)\cdot\Big(c(X)u_0 - c(\omega\cdot X)(1-u_0)\Big)
\end{aligned}
\tag{29}
$$

The second type of polynomials are

$$
\begin{aligned}
h_0(X) =& L_0(X)\cdot\big(z(X) - a_0\cdot c_0\big)\\
h_1(X) =& (X-1)\cdot\big(z(X) - z(\omega^{-1}\cdot X) - a(X)\cdot c(X))\\
h_2(X) =& L_{N-1}(X)\cdot\big(z(X) - v\big)
\end{aligned}
\tag{30}
$$

We can use an α provided by the Verifier to aggregate these polynomials into a large polynomial, denoted as $h(X)$:

$$
\begin{split}
h(X) = p_0(X) + \alpha\cdot p_1(X) + \alpha^2\cdot p_2(X) + \cdots + \alpha^n\cdot p_n(X)\\
+ \alpha^{n+1}\cdot h_0(X) + \alpha^{n+2}\cdot h_1(X) + \alpha^{n+3}\cdot h_2(X)
\end{split}
\tag{31}
$$

If $\vec{c}$ and $\vec{z}$ are correct, then the values of $h(X)$ at $X\in H$ are all zero, and then we know that $h(X)$ is zero everywhere on H, so it must contain the Vanishing polynomial $v_H(X)$ of H as a factor, i.e., there exists a Quotient polynomial $t(X)$, such that

$$
h(X) = t(X)\cdot v_H(X) \tag{32}
$$

Then the Verifier challenges a random point ζ, requiring the Prover to calculate and send the values of $a(X)$, $c(X)$ and $z(X)$ $t(X)$ at ζ, as well as the value of $z(X)$ at $X=\zeta\cdot\omega^{-1}$, and the values of $c(X)$ at $X=\zeta\cdot\omega,\zeta\cdot\omega^2,\ldots,\zeta\cdot\omega^{2^{n-1}}$. And provide KZG10 Evaluation proofs π_{kzg10} for these evaluations.

$$
\begin{aligned}
\pi_e =& \Big(a(\zeta), c(\zeta), c(\zeta\cdot\omega), c(\zeta\cdot\omega^2),\ldots, c(\zeta\cdot\omega^{2^{n-1}}), z(\zeta), z(\zeta\cdot\omega^{-1}), t(\zeta)\Big)\\
\pi_{kzg10} =& \Big(\pi_{a(\zeta)}, \pi_{c(\zeta)}, \pi_{c(\zeta\cdot\omega)}, \pi_{c(\zeta\cdot\omega^2)},\ldots, \pi_{c(\zeta\cdot\omega^{2^{n-1}})}, \pi_{z(\zeta)}, \pi_{z(\zeta\cdot\omega^{-1})}, \pi_{t(\zeta)}\Big)
\end{aligned}
\tag{33}
$$

The Verifier first verifies the correctness of π_{kzg10}, then the Verifier calculates the values of the following public polynomials at $X=\zeta$:

$$
s_0(\zeta) = \frac{v_H(\zeta)}{v_{H_0}(\zeta)}\quad s_1(\zeta) = \frac{v_H(\zeta)}{v_{H_1}(\zeta)}\quad s_2(\zeta) = \frac{v_H(\zeta)}{v_{H_2}(\zeta)}\quad\cdots\quad s_{n-1}(\zeta) = \frac{v_H(\zeta)}{v_{H_{n-1}}(\zeta)} \tag{34}
$$

And

$$
\begin{aligned}
L_0(\zeta) &= \frac{v_H(\zeta)}{N(\zeta-1)} \\
L_{N-1}(\zeta) &= \frac{v_H(\zeta)}{N(\omega\cdot\zeta-1)}
\end{aligned} \tag{35}
$$

Then calculate the following values using all the polynomial operation values included in π_e:

$$
\begin{aligned}
p_0(\zeta) &= s_0(\zeta)\cdot\big(c(\zeta)-(1-u_0)(1-u_1)\cdots(1-u_{n-1})\big) \\
p_1(\zeta) &= s_0(\zeta)\cdot\big(c(\zeta)u_{n-1}-c(\omega^{2^{n-1}}\cdot\zeta)(1-u_{n-1})\big) \\
p_2(\zeta) &= s_1(\zeta)\cdot\big(c(\zeta)u_1-c(\omega^{2^{n-2}}\cdot\zeta)(1-u_1)\big) \\
&\cdots \\
p_n(\zeta) &= s_{n-1}(\zeta)\cdot\big(c(\zeta)u_0-c(\omega\cdot\zeta)(1-u_0)\big) \\
h_1(\zeta) &= L_0(\zeta)\cdot\big(z(\zeta)-a_0\cdot c_0\big) \\
h_2(\zeta) &= (\zeta-1)\cdot\big(z(\zeta)-z(\omega^{-1}\cdot\zeta)-a(\zeta)\cdot c(\zeta)\big) \\
h_3(\zeta) &= L_{N-1}(\zeta)\cdot\big(z(\zeta)-v\big)
\end{aligned} \tag{36}
$$

Use α to aggregate these polynomial evaluations to get $h(\zeta)$:

$$
\begin{aligned}
h(\zeta) = p_0(\zeta) &+ \alpha\cdot p_1(\zeta)+\alpha^2\cdot p_2(\zeta)+\cdots+\alpha^n\cdot p_n(\zeta) \\
&+\alpha^{n+1}\cdot h_0(\zeta)+\alpha^{n+2}\cdot h_1(\zeta)+\alpha^{n+3}\cdot h_2(\zeta)
\end{aligned} \tag{37}
$$

Finally, check if the following equation holds:

$$
h(\zeta) \overset{?}{=} t(\zeta)\cdot v_H(\zeta) \tag{38}
$$

where $v_H(\zeta)$ is the value of the Vanishing polynomial of H at ζ, calculated by the Verifier itself.

5. PH23+KZG10 Protocol Flow (Simplified Version)

In this article, we don't consider the performance of the protocol, but only show a simple and direct implementation of PH23+KZG10. The protocol is mainly divided into three parts:

- Commit of MLE polynomial
- Proof process of MLE polynomial's Evaluation Argument
- Verification process of MLE polynomial's Evaluation Argument

Commit

If we want to commit to an MLE polynomial with n unknowns, $\tilde{f}(X_0,X_1,\ldots,X_{n-1})$, defined as follows:

$$
\tilde{f}(X_0,X_1,\ldots,X_{n-1}) = \sum_{i=0}^{N-1} a_i\cdot\tilde{eq}(\mathsf{bits}(i),(X_0,X_1,\ldots,X_{n-1})) \tag{39}
$$

Here $\vec{a}=(a_0,a_1,\ldots,a_{N-1})$ represents the values of this MLE polynomial on the $\{0,1\}^n$ Boolean HyperCube.

First, the Prover constructs a univariate polynomial, denoted as $a(X)$, such that its values on Domain H are exactly $\vec{a}$, and the size of H is exactly N. Moreover, H is generally a multiplication-friendly subgroup, which can be generated by an N-th Root of Unity (denoted as ω) as the generator:

$$H = (1, \omega, \omega^2, \omega^3, \ldots, \omega^{N-1}) \tag{40}$$

Then $a(X)$ is defined as follows:

$$a(X) = a_0 \cdot L_0(X) + a_1 \cdot L_1(X) + \cdots + a_{N-1} \cdot L_{N-1}(X) \tag{41}$$

Here $\{L_i\}_{i=0}^{N-1}$ represents the Lagrange polynomials based on H, defined as follows:

$$L_i(X) = \frac{\omega_i \cdot v_H(X)}{N \cdot (X - \omega_i)}, \qquad i = 0, 1, \ldots, N-1 \tag{42}$$

Here $v_H(X)$ is the Vanishing polynomial on H, defined as follows:

$$v_H(X) = (X-1)(X-\omega)(X-\omega^2)\cdots(X-\omega^{N-1}) \tag{43}$$

Then the Prover calculates the coefficient form of $a(X)$, uses the SRS of KZG10 to calculate the commitment of $a(X)$, denoted as C_a

$$C_a = b_0 \cdot [1]_1 + b_1 \cdot [\tau]_1 + b_2 \cdot [\tau^2]_1 + \cdots + b_{N-1} \cdot [\tau^{N-1}]_1 \tag{44}$$

Here $(b_0, b_1, \ldots, b_{N-1})$ is the coefficient form of $a(X)$:

$$a(X) = b_0 + b_1 X + b_2 X^2 + \cdots + b_{N-1} X^{N-1} \tag{45}$$

Evaluation Proof Protocol

The so-called Evaluation Argument refers to the Prover proving to the Verifier that an MLE polynomial $\tilde{f}$ corresponding to a polynomial commitment C_f with n unknowns takes a value at a public point $(u_0, u_1, \ldots, u_{n-1})$:

$$\tilde{f}(u_0, u_1, u_2, \ldots, u_{n-1}) = v \tag{46}$$

Here, we also assume that the value of the polynomial at the public point is also a public value, denoted as v.

Common Input

1. $C_a = [a(\tau)]_1$: A univariate polynomial commitment for vector $\vec{a} = (a_0, a_1, \ldots, a_{N-1})$. Where $\vec{a}$ equals the values of MLE polynomial $\tilde{f}(X_0, X_1, \ldots, X_{n-1})$ on the n-dimensional Boolean Hypercube, which is also the values of $a(X)$ on H.
2. $\vec{u} = (u_0, u_1, \ldots, u_{n-1})$: The evaluation point of MLE polynomial $\tilde{f}(X_0, X_1, \ldots, X_{n-1})$
3. v: The value of MLE polynomial $\tilde{f}(X_0, X_1, \ldots, X_{n-1})$ at $\vec{u}$.

Round 1.

Prover:

1. Construct polynomial $c(X)$, where $c_i = \widetilde{eq}(\mathsf{bits}(i), \vec{u})$

$$c(X) = \sum_{i=0}^{N-1} c_i \cdot L_i(X) \tag{47}$$

2. Calculate the commitment of $c(X)$, $C_c = [c(\tau)]_1$, and send C_c

$$C_c = [c(\tau)]_1 = \mathsf{KZG10.commit}(\vec{c}) \tag{48}$$

Round 2.

Verifier: Send random number $\alpha \leftarrow_\$ \mathbb{F}_p$

Prover:

1. Construct Selector polynomials $s_0(X), s_1(X), \ldots, s_{n-1}(X)$

$$s_i(X) = \frac{v_H(X)}{z_{H_i}(X)} = \frac{X^{2^n} - 1}{X^{2^i} - 1}, \qquad i = 0, 1, \ldots, n-1 \tag{49}$$

2. Construct constraint polynomials $p_0(X), \ldots, p_{n-1}(X)$ for $\vec{c}$

$$p_0(X) = s_0(X) \cdot \Big(c(X) - (1-u_0)(1-u_1)\ldots(1-u_{n-1})\Big) \tag{50}$$

$$p_k(X) = s_{k-1}(X) \cdot \Big(u_{n-k} \cdot c(X) - (1-u_{n-k}) \cdot c(\omega^{2^{n-k}} \cdot X)\Big), \qquad k = 1 \ldots n \tag{51}$$

3. Construct accumulation polynomial $z(X)$, satisfying

$$\begin{aligned} z(1) &= a_0 \cdot c_0 \\ z(\omega_i) - z(\omega_{i-1}) &= a(\omega_i) \cdot c(\omega_i), \quad i = 1, \ldots, N-1 \\ z(\omega^{N-1}) &= v \end{aligned} \tag{52}$$

4. Construct constraint polynomials $h_0(X), h_1(X), h_2(X)$, satisfying

$$\begin{aligned} h_0(X) &= L_0(X) \cdot \big(z(X) - c_0 \cdot a(X)\big) \\ h_1(X) &= (X-1) \cdot \big(z(X) - z(\omega^{-1} \cdot X) - a(X) \cdot c(X)\big) \\ h_2(X) &= L_{N-1}(X) \cdot \big(z(X) - v\big) \end{aligned} \tag{53}$$

5. Construct aggregation polynomial $h(X)$:

$$\begin{aligned} h(X) = p_0(X) + \alpha \cdot p_1(X) + \alpha^2 \cdot p_2(X) + \cdots + \alpha^n \cdot p_n(X) \\ + \alpha^{n+1} \cdot h_0(X) + \alpha^{n+2} \cdot h_1(X) + \alpha^{n+3} \cdot h_2(X) \end{aligned} \tag{54}$$

6. Calculate Quotient polynomial $t(X)$, satisfying

$$t(X) \cdot v_H(X) = h(X) \tag{55}$$

7. Calculate polynomial commitments $C_t = [t(\tau)]_1$, $C_z = [z(\tau)]_1$, and send C_t and C_z

Round 3.

Verifier: Send random evaluation point $\zeta \leftarrow_\$ \mathbb{F}_p$

Prover:

1. Calculate the values of $s_i(X)$ at ζ:

$$s_0(\zeta), s_1(\zeta), \ldots, s_{n-1}(\zeta) \tag{56}$$

2. Define a new Domain D, containing $n+1$ elements:

$$D = \{\omega, \omega^2, \omega^4, \ldots, \omega^{2^{n-1}}, 1\} \tag{57}$$

Its coset $D' = \zeta D$ is the set of evaluation points for $c(X)$ that the Prover needs to calculate.

$$D' = \zeta D = \{\zeta\omega, \zeta\omega^2, \zeta\omega^4, \ldots, \zeta\omega^{2^{n-1}}, \zeta\} \tag{58}$$

3. Calculate $c(\zeta \cdot \omega), c(\zeta \cdot \omega^2), c(\zeta \cdot \omega^4), \ldots, c(\zeta \cdot \omega^{2^{n-1}})$, $c(\zeta)$
4. Calculate $z(\zeta), z(\omega^{-1} \cdot \zeta)$, $t(\zeta), a(\zeta)$
5. Send these polynomial values:

$$\big(a(\zeta), c(\zeta), c(\zeta \cdot \omega), c(\zeta \cdot \omega^2), \ldots, c(\zeta \cdot \omega^{2^{n-1}}), z(\zeta), z(\zeta \cdot \omega^{-1}), t(\zeta)\big) \tag{59}$$

6. Send KZG10 evaluation proofs

$$\pi_{kzg10} = \big(\pi_{a(\zeta)}, \pi_{c(\zeta)}, \pi_{c(\zeta\cdot\omega)}, \pi_{c(\zeta\cdot\omega^2)}, \ldots, \pi_{c(\zeta\cdot\omega^{2^{n-1}})}, \pi_{z(\zeta)}, \pi_{z(\zeta\cdot\omega^{-1})}, \pi_{t(\zeta)}\big) \tag{60}$$

Verification

The proof π contains the following elements:

$$\pi = \begin{pmatrix} C_t, C_z, C_c, a(\zeta), z(\zeta), z(\zeta \cdot \omega^{-1}), t(\zeta), \\ c(\zeta), c(\zeta \cdot \omega), c(\zeta \cdot \omega^2), \ldots, c(\zeta \cdot \omega^{2^{n-1}}), \\ \pi_{a(\zeta)}, \pi_{z(\zeta)}, \pi_{z(\zeta\cdot\omega^{-1})}, \pi_{t(\zeta)}, \\ \pi_{c(\zeta)}, \pi_{c(\zeta\cdot\omega)}, \pi_{c(\zeta\cdot\omega^2)}, \ldots, \pi_{c(\zeta\cdot\omega^{2^{n-1}})}, \end{pmatrix} \tag{61}$$

The Verifier first calculates $s_0(\zeta), \ldots, s_{n-1}(\zeta)$, $v_H(\zeta)$, $L_0(\zeta), L_{N-1}(\zeta)$.

The Verifier needs to verify the following equations to complete the verification process of the evaluation proofs for polynomials $a(X)$, $c(X)$, $t(X)$, $z(X)$:

$$\begin{aligned} \mathsf{KZG.Verify}(C_a, \zeta, a(\zeta), \pi_{a(\zeta)}) &\overset{?}{=} 1 \\ \mathsf{KZG.Verify}(C_t, \zeta, t(\zeta), \pi_{t(\zeta)}) &\overset{?}{=} 1 \\ \mathsf{KZG.Verify}(C_z, \zeta, z(\zeta), \pi_{z(\zeta)}) &\overset{?}{=} 1 \\ \mathsf{KZG.Verify}(C_z, \zeta\omega^{-1}, z(\zeta\omega^{-1}), \pi_{z(\zeta\omega^{-1})}) &\overset{?}{=} 1 \\ \mathsf{KZG.Verify}(C_c, \zeta, c(\zeta), \pi_{c(\zeta)}) &\overset{?}{=} 1 \\ \mathsf{KZG.Verify}(C_c, \zeta\omega, c(\zeta\omega), \pi_{c(\zeta\omega)}) &\overset{?}{=} 1 \\ \mathsf{KZG.Verify}(C_c, \zeta\omega^2, c(\zeta\omega^2), \pi_{c(\zeta\omega^2)}) &\overset{?}{=} 1 \\ &\cdots \\ \mathsf{KZG.Verify}(C_c, \zeta\omega^{2^{n-1}}, c(\zeta\omega^{2^{n-1}}), \pi_{c(\zeta\omega^{2^{n-1}})}) &\overset{?}{=} 1 \end{aligned} \tag{62}$$

The Verifier uses the polynomial values at $X = \zeta$ to verify the following constraint equation:

$$t(\zeta) \cdot v_H(\zeta) \overset{?}{=} \Big(\sum_{i=0}^{n} \alpha^i \cdot p_i(\zeta)\Big) + \alpha^{n+1} \cdot h_0(\zeta) + \alpha^{n+2} \cdot h_1(\zeta) + \alpha^{n+3} \cdot h_2(\zeta) \tag{63}$$

Here $p_0(\zeta), \ldots, p_n(\zeta), h_0(\zeta), h_1(\zeta), h_2(\zeta)$ are defined as follows:

$$
\begin{aligned}
p_0(\zeta) &= s_0(\zeta) \cdot \big(c(\zeta) - (1-u_0)(1-u_1)\cdots(1-u_{n-1})\big) \\
p_1(\zeta) &= s_0(\zeta) \cdot \big(c(\zeta)u_{n-1} - c(\omega^{2^{n-1}} \cdot \zeta)(1-u_{n-1})\big) \\
p_2(\zeta) &= s_1(\zeta) \cdot \big(c(\zeta)u_1 - c(\omega^{2^{n-2}} \cdot \zeta)(1-u_1)\big) \\
&\cdots \\
p_n(\zeta) &= s_{n-1}(\zeta) \cdot \big(c(\zeta)u_0 - c(\omega \cdot \zeta)(1-u_0)\big) \\
h_0(\zeta) &= L_0(\zeta) \cdot \big(z(\zeta) - a_0 \cdot c_0\big) \\
h_1(\zeta) &= (\zeta - 1) \cdot \big(z(\zeta) - z(\zeta\omega^{-1}) - a(\zeta) \cdot c(\zeta)\big) \\
h_2(\zeta) &= L_{N-1}(\zeta) \cdot \big(z(\zeta) - v\big)
\end{aligned} \tag{64}
$$

Summary

The PH23 PCS Adaptor maps the Evaluations of MLE polynomials to the Evaluations of a univariate polynomial, and then uses "sum proof" to ensure the correctness of MLE polynomial operations.

We will optimize and improve this protocol in subsequent articles.

References

- [PH23] Papini, Shahar, and Ulrich Haböck. "Improving logarithmic derivative lookups using GKR." Cryptology ePrint Archive (2023). https://eprint.iacr.org/2023/1284
- [KT23] Kohrita, Tohru, and Patrick Towa. "Zeromorph: Zero-knowledge multilinear-evaluation proofs from homomorphic univariate commitments." Cryptology ePrint Archive (2023). https://eprint.iacr.org/2023/917
- [ZXZS20] Jiaheng Zhang, Tiancheng Xie, Yupeng Zhang, and Dawn Song. "Transparent polynomial delegation and its applications to zero knowledge proof." 2020 IEEE Symposium on Security and Privacy (SP). IEEE, 2020. https://eprint.iacr.org/2019/1482
- [KZG10] Kate, Aniket, Gregory M. Zaverucha, and Ian Goldberg. "Constant-size commitments to polynomials and their applications." Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16. Springer Berlin Heidelberg, 2010.
- [CHMMVW19] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas Ward. "Marlin: Preprocessing zkSNARKs with universal and updatable SRS." Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part I 39. Springer International Publishing, 2020.
- [BCRSVW19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. "Aurora: Transparent succinct arguments for R1CS." Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part I 38. Springer International Publishing, 2019.
- [RZ21] Carla Ràfols and Arantxa Zapico. An algebraic framework for universal and updatable SNARKs. In Tal Malkin and Chris Peikert, editors, CRYPTO 2021, Part I, volume 12825 of LNCS, pages 774–804, Virtual Event, August 2021. Springer, Heidelberg.