缺失的协议 PH23-PCS（一）

在 Improving logarithmic derivative lookups using GKR （[PH23]） 论文中， 作者给出了一个将 MLE 转成 Univariate Polynomial 的思路，虽然论文没有给出完整的协议描述，但是这个协议也展示了在某些特性方面的优势，比如可以支持任意偏移的 Shift Argument。

这个方案的主要优势是能支持任意的 Shift Argument（见论文 Appendix A.2）。其次，如果对接 KZG10，那么这个 PCS Adaptor 的证明仅仅包含常数个 $\mathbb{G}_1$ 元素，和对数个 $\mathbb{F}_r$ 元素。这要优于 Gemini-PCS 与 Zeromorph-PCS（KT23），后者需要对数个 $\mathbb{G}_1$ 元素。

这个协议的思路与 Virgo-PCS 有类似之处，都是将 MLE 的多项式运算看成是一个求和，并且都是利用 Univariate Sumcheck 协议来完成「求和证明」。但是 PH23-PCS 还要求 Prover 证明 MLE Lagrange Polynomial 在求值点上的值，从而减轻 Verifier 的负担；而 Virgo-PCS 则利用了 GKR 协议来做到这一点。另一个不同之处是，Virgo-PCS 要求 MLE 多项式需要是一个 Coefficient Form 的表示，因此 Virgo-PCS 利用 GKR 电路来完成 MLE 多项式由 Evaluation Form 到 Coefficient Form 的转换的计算正确性的证明。

这个文章系列补全了 [PH23] 论文中关于 PH23-PCS 的描述，并给出了 PH23-KZG10 的一个简化协议来帮助大家理解这个协议的基本思路。

本文先详细介绍 PH23-PCS-Adaptor 的基础原理，然后给出 PH23-KZG10 的一个简单协议实现。

1. 原理概述

在解释 Prover 如何证明一个 MLE 多项式 $\tilde{f}(\vec{X})$ 的 Evaluation 之前，我们先回忆下 MLE 多项式的定义：

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

这里 $N = 2^n$ 。 当要计算 $\tilde{f}(\vec{X})$ 在 $\vec{X} = (u_0, u_1, \ldots, u_{n-1})$ 处的值时，我们需要计算 $\sum_{i=0}^{N-1} a_i \cdot \widetilde{eq}(\mathsf{bits}(i), \vec{u})$。为了方便解释，我们引入一个新的向量 $\vec{c}$ ，其中的每一个元素 c_i = $\widetilde{eq}(\mathsf{bits}(i), \vec{u})$。

如果 $n = 3$ and $N = 8$, 那么 $\vec{c}$ 的值可以全部枚举出：

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

可以看出，$\vec{c}$ 的元素定义具有一定的规律。比如 c_i 是 s 个数值的连乘，这些数值也具有一定的规律。其中 $(1-u_i)$ 表示二进制的 0 ，而 u_i 表示二进制的 1 。比如 c_7 是由三个数相乘，分别为 u_0, u_1, u_2 ，这表示 `(111)`，这恰好是 7 的二进制表示。又比如 c_5 是由三个数相乘，分别为 $u_0, (1-u_1), u_2$ ，这表示 `(101)`，正是 5 的二进制表示。

而 PH23 的关键思路是，是否可以让 Prover 先承诺向量 $\vec{c}$，然后证明 $\vec{c}$ 的每一个元素都是按照上面的二进制规律去正确定义的。如果可以，那么 Prover 再证明一个 Inner Product 关系，即可以证明 $\langle \vec{a}, \vec{c} \rangle = v$ ，这正是等价于证明 $\tilde{f}(\vec{X}) = v$ 。

因此 PH23 的证明协议分为两部分：

1. 证明 $\vec{c}$ 向量的 Well-Formedness。
2. 证明 $\langle \vec{a}, \vec{c} \rangle = v$ 。

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

继续 $s = 3$ 的 $\vec{c}$ 的例子，

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

观察到

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

于是，如果 c_0 是正确的，那么我们可以通过证明下面的约束等式

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

来证明 c_4 是正确的。接下来，观察

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

由此我们可以推断，如果 c_0 是正确的，那么下面两个约束等式保证了 c_2, c_6 是正确的：

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

接下来，我们可以证明 c_1, c_3, c_5, c_7 是正确的，因为它们可以由 c_0, c_2, c_4, c_6 推导出来，而 c_0, c_2, c_4, c_6 已经在上一步中证明正确：

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

最后结论，通过上面的 $1+2+4$ 个约束等式，我们在已知 c_0 正确的前提下，可以证明 $c_1, c_2, c_3, c_4, c_5, c_6, c_7$ 都是正确的。向量 $\vec{c}$ 的元素之间的推理关系如下图所示：

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

假设 H 为大小为 8 的有限域 $\mathbb{F}_p$ 的乘法子群，$H = \{1, \omega, \omega^2, \omega^3, \omega^4, \omega^5, \omega^6, \omega^7\}$，其中 $\omega \in \mathbb{F}_p$ 是 8 次单位根。并且将 $\{L_i(X)\}_{i=0}^{N-1}$ 记为 H 上的 Lagrange Basis 多项式。

然后我们可以引入 $c(X)$ 作为 $\vec{c}$ 按照 Lagrange Basis 编码的多项式：

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

容易验证，$c(\omega^i) = c_i$，其中 $i = 0, 1, 2, \ldots, N-1$。

进一步，上面证明 c_1, c_3, c_5, c_7 的四个约束等式可以合并为一个多项式约束等式：

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

我们可以代入 $X = \omega^2$，上面的约束等式则对应于：

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

分别代入 $X = \omega, X = \omega^4, X = \omega^6$，我们可以得到证明 c_1, c_5, c_7 正确性的约束等式。

而 $(X - \omega)(X - \omega^3)(X - \omega^5)(X - \omega^7)$ 这个多项式像是一个 Selector 多项式，过滤掉不满足的 X 值。

利用这个方法，我们可以利用 $n = \log N$ 个多项式约束来证明 $\vec{c}$ 的 Well-Formedness。

对于 $N = 8$ 的例子，我们需要引入 3 个 Selector 多项式 $s_0(X), s_1(X), s_2(X)$，

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

其中 $v_H(X)$ 与 $v_{H_i}(X)$ 分别是 Domain H 与 H_i 的 Vanishing 多项式。而 H_i 是 H 的子群，并且满足下面的 Group Tower 关系：

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

他们的定义如下：

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

自然地，Selector 多项式 $s_0(X), s_1(X), s_2(X)$ 的表示如下：

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

多项式约束等式

保证 c_0 正确的约束等式可以表示为下面的多项式约束：

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

保证 c_4 正确的约束等式可以表示为下面的多项式约束：

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

下面是保证 c_2, c_6 正确的约束等式：

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

最后，保证 c_1, c_3, c_5, c_7 正确的约束等式：

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

3. 证明 Inner Product

证明的第二个部分是证明 $\langle \vec{a}, \vec{c} \rangle = v$。假设 $a(X)$ 是向量 $\vec{a}$ 的编码，即 $a(X) \mid_H = \vec{a}$，那么 $a(X)$ 被承诺为 $[a(\tau)]_1$，加上 $c(X)$ 的承诺，$[c(\tau)]_1$，我们可以使用 Univariate Sumcheck 协议来证明内积。

Univariate Sumcheck

这里我们先看一个定理(Remark 5.6 in [BCRSVW19], Sec.3 in [RZ21], Sec.5.1 in [CHMMVW19])：对任意的 $P(X) \in \mathbb{F}[X]$，一个乘法子群 $H \subset \mathbb{F}$，$P(X)$ 可以分解为：

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

这里 v 是 $P(X)$ 在 H 上的求和，即

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

因此，我们可以使用这个定理来证明两个向量的内积。如果 $a(X) \cdot c(X)$ 可以表示为下面的等式，

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

那么 $\langle \vec{a}, \vec{c} \rangle = v$。

Prover 可以发送 $q(X)$ 与 $g(X)$ 的承诺，然后 Verifier 通过挑战 ζ，Prover 发送相关多项式在 $X = \zeta$ 的取值，然后 Verifier 验证上面等式是否成立：

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

Prover 和 Verifier 再通过一个一元多项式承诺方案，比如 KZG10 来证明 $a(\zeta), c(\zeta), q(\zeta), g(\zeta)$ 的正确性。

同时通过 KZG10 协议还可以证明 $g(X)$ 的 Degree Bound，即 $\deg(g(X)) < N - 1$。

Grand Sum

其实，我们还可以使用 Grand Sum 协议来证明两个向量的内积。

Univariate Sumcheck 的一个问题是它包含了一个 Degree Bound 约束，即 $\deg(g(X)) < N - 1$。这对于协议下层的KZG10 协议来说，需要进行额外的处理，这会使得协议的复杂度增加，也会引入过多的 Pairing 运算。而 Grand Sum 协议则避免了引入 Degree Bound 约束。

Grand Sum 协议的思路来自于 Plonk [GWC19] 中的 Grand Product Argument，而这个协议最早由 [BG12] 提出。

假设有一个多项式 $f(X)$ 编码了向量 $\vec{f}$ 的值，那么我们构造一个辅助的向量 $\vec{z}$，满足：

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

或者用更简洁的递推公式表示：

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

我们可以通过多项式 $z(X)$ 来编码 $\vec{z}$，即

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

其中 $L_i(X)$ 是上面定义的关于 H 的 Lagrange Basis 多项式。

然后我们可以利用下面三个多项式约束来表示 z_i 的递推公式，从而保证 $z(X)$ 的正确性：

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

注意这里我们用 $z(\omega^{-1} \cdot X)$ 来表示 z_{i-1}。并且第三个多项式约束，保证了求和的结果等于最终的多项式运算结果 v。

利用一个 Univariate PCS 协议，比如 KZG10 或者 FRI，我们可以证明上述这些多项式约束等式的正确性。

4. 对接 KZG10

对于证明上面列出的多项式约束，我们可以基于 KZG10 协议来实现对这些多项式等式的证明。总结下，我们有两类的多项式约束。第一类是关于 $c(X)$ 的正确性的约束，第二类是关于 $z(X)$ 的正确性的约束。

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

第二类多项式为

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

我们可以利用一个 Verifier 提供的 α 将这些多项式聚合为一个大的多项式，记为 $h(X)$：

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

如果 $\vec{c}$ 与 $\vec{z}$ 是正确的，那么 $h(X)$ 在 $X \in H$ 上的取值都为零，然后我们知道 $h(X)$ 在 H 上处处为零，因此它必然包含 H 的 Vanishing 多项式 $v_H(X)$ 作为因式，即存在一个 Quotient 多项式 $t(X)$，使得

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

然后 Verifier 挑战一个随机点 ζ，要求 Prover 计算并发送 $a(X)$，$c(X)$ 以及 $z(X)$ $t(X)$ 在 ζ 处的取值，以及 $z(X)$ 在 $X = \zeta \cdot \omega^{-1}$ 处的取值，还有 $c(X)$ 在 $X = \zeta \cdot \omega, \zeta \cdot \omega^2, \ldots, \zeta \cdot \omega^{2^{n-1}}$ 处的取值。并提供这些求值的 KZG10 Evaluation 证明 π_{kzg10}。

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

Verifier 首先验证 π_{kzg10} 的正确性， 然后 Verifier 自行计算下面的公开多项式在 $X = \zeta$ 处的取值：

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

还有

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

然后通过 π_e 中包含的所有的多项式运算值计算下面的值：

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

用 α 来聚合这些多项式求值，得到 $h(\zeta)$：

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

最后检查下面的等式是否成立：

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

其中 $v_H(\zeta)$ 是 H 的 Vanishing 多项式在 ζ 处的取值，由 Verifier 自行计算。

5. PH23+KZG10 协议流程（简单版）

本文我们不考虑协议的性能，仅仅展示 PH23+KZG10 的一个简单直接的实现方式。协议主要分为三部分：

- MLE 多项式的 Commit
- MLE 多项式的 Evaluation Argument 的证明流程
- MLE 多项式的 Evaluation Argument 的验证流程

Commit

如果要承诺一个有 n 个未知数的 MLE 多项式，$\tilde{f}(X_0, X_1, \ldots, X_{n-1})$，定义如下：

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

这里 $\vec{a} = (a_0, a_1, \ldots, a_{N-1})$ 表示该 MLE 多项式在 $\{0,1\}^n$ Boolean HyperCube 上的取值。

首先 Prover 构造一个一元多项式，记为 $a(X)$，使其在 Domain H 上的取值恰好为 $\vec{a}$，并且 H 的大小恰好为 N。并且，H 一般是一个 FFT 友好的乘法子群，它可以由一个 N 次单位根（N-th Root of Unity）来作为生成元（记为 ω）产生：

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

那么 $a(X)$ 定义如下：

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

这里 $\{L_i\}_{i=0}^{N-1}$ 表示基于 H 的 Lagrange 多项式，其定义如下：

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

这里 $v_H(X)$ 为 H 上的 Vanishing 多项式，定义如下：

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

然后 Prover 计算得到 $a(X)$ 的系数形式，利用 KZG10 的 SRS 计算 $a(X)$ 的承诺，记为 C_a

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

这里 $(b_0, b_1, \ldots, b_{N-1})$ 为 $a(X)$ 的系数形式：

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

Evaluation 证明协议

所谓的 Evaluation Argument 是指 Prover 向 Verifier 证明一个多项式承诺 C_f 所对应的具有 n 个未知数的 MLE 多项式 $\tilde{f}$ 在一个公开点 $(u_0, u_1, \ldots, u_{n-1})$ 的取值：

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

这里，我们同样假设多项式在公开点的取值也是一个公开的值，记为 v。

公共输入

1. $C_a = [a(\tau)]_1$: 对于向量 $\vec{a} = (a_0, a_1, \ldots, a_{N-1})$ 的一元多项式承诺。其中 $\vec{a}$ 等于 MLE 多项式 $\tilde{f}(X_0, X_1, \ldots, X_{n-1})$ 在 n-维 Boolean Hypercube 上的取值，也是 $a(X)$ 在 H 上的取值。
2. $\vec{u} = (u_0, u_1, \ldots, u_{n-1})$: MLE 多项式 $\tilde{f}(X_0, X_1, \ldots, X_{n-1})$ 的求值点
3. v: MLE 多项式 $\tilde{f}(X_0, X_1, \ldots, X_{n-1})$ 在 $\vec{u}$ 处的取值。

Round 1.

Prover:

1. 构造多项式 $c(X)$, 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. 计算 $c(X)$ 的承诺 $C_c = [c(\tau)]_1$，并发送 C_c

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

Round 2.

Verifier: 发送随机数 $\alpha \leftarrow_{\$} \mathbb{F}_p$

Prover:

1. 构造 Selector 多项式 $s_0(X), s_1(X), \ldots, s_{n-1}(X)$

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

2. 构造 $\vec{c}$ 的约束多项式 $p_0(X), \ldots, p_{n-1}(X)$

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

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

3. 构造累加多项式 $z(X)$ ，满足

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

4. 构造约束多项式 $h_0(X), h_1(X), h_2(X)$，满足

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

5. 构造聚合多项式 $h(X)$:

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

6. 计算 Quotient 多项式 $t(X)$，满足

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

7. 计算多项式承诺 $C_t = [t(\tau)]_1$， $C_z = [z(\tau)]_1$，并发送 C_t 和 C_z

Round 3.

Verifier: 发送随机求值点 $\zeta \leftarrow_{\$} \mathbb{F}_p$

Prover:

1. 计算 $s_i(X)$ 在 ζ 处的取值：

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

2. 定义一个新的 Domain D，包含 $n+1$ 个元素：

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

它的陪集 $D' = \zeta D$ 是 Prover 要计算 $c(X)$ 的求值点的集合。

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

3. 计算 $c(\zeta \cdot \omega), c(\zeta \cdot \omega^2), c(\zeta \cdot \omega^4), \ldots, c(\zeta \cdot \omega^{2^{n-1}}), c(\zeta)$
4. 计算 $z(\zeta), z(\omega^{-1} \cdot \zeta)$ ， $t(\zeta), a(\zeta)$
5. 发送这些多项式的取值：

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

6. 发送 KZG10 的求值证明

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

Verification

证明 π 包含下面的元素：

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

Verifier 先计算出 $s_0(\zeta), \ldots, s_{n-1}(\zeta)$ ， $v_H(\zeta)$ ， $L_0(\zeta), L_{N-1}(\zeta)$ 。

Verifier 需要验证下面的等式，来完成对多项式 $a(X), c(X), t(X), z(X)$ 多项式的求值证明的验证过程：

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

Verifier 用多项式在 $X = \zeta$ 处的取值来验证下面的约束等式：

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

这里 $p_0(\zeta), \ldots, p_n(\zeta), h_0(\zeta), h_1(\zeta), h_2(\zeta)$ 的定义如下：

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

总结

PH23 PCS Adaptor 是将 MLE 多项式的 Evaluations 映射到一个一元多项式的 Evaluations，然后再利用「求和证明」来保证 MLE 多项式运算的正确性。

我们将在后续文章优化和改进这个协议。

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.