# HyperKZG

在 Gemini-PCS [BCHO22] 中，一个系数式的 MLE 多项式，对应到一个一元多项式，

$$\tilde{f}(X_0, X_1, \ldots, X_{n-1}) = f_0 + f_1X_0 + f_2X_1 + f_3X_0X_1 + \cdots + f_{2^n-1}X_0X_1\cdots X_{n-1} \tag{1}$$

对应到一个一元多项式：

$$f(X) = f_0 + f_1X + f_2X^2 + \cdots + f_{2^n-1}X^{2^n-1} \tag{2}$$

当我们确定一个公开的求值点 $\vec{u} = (u_0, u_1, \ldots, u_{n-1})$，MLE 多项式 $\tilde{f}$ 在 $\vec{u}$ 处的值可以表示为下面的 Tensor Product：

$$\tilde{f}(u_0, u_1, \ldots, u_{n-1}) = \langle \vec{f}, \otimes_{i=0}^{n-1}(1, u_i)\rangle \tag{3}$$

接下来，Gemini-PCS 利用 Split-and-fold 的思路，把上面的等式转换成多个一元多项式的求值的正确性，这可以利用 KZG10 来完成证明。

不过通常 MLE 多项式默认是以点值式的形式表示，

$$\tilde{f}(X_0, X_1, \ldots, X_{n-1}) = \sum_{i=0}^{2^n-1} a_i \cdot \tilde{eq}(\mathsf{bits}(i), (X_0, X_1, \ldots, X_{n-1})) \tag{4}$$

为了使用 Gemini-PCS，Prover 需要先将 MLE 的点值式转换成上述的系数式，即通过 $\vec{a}$ 向量计算得到 $\vec{f}$ 向量。这个转换算法类似 FFT 运算，时间复杂度为 $O(N\log N)$，这里 $N = 2^n$。

HyperKZG 的思路是，仍然利用 Gemini-PCS 的 Split-and-fold 的核心思路，但是不需要进行类似 FFT 的多项式转换。这初听起来似乎不可思议，但这里的关键点在于，MLE 多项式本质上是多维空间中的线性多项式，不管它是 Evaluation-form 还是 Coefficient-form，其运算过程实际上都是一个线性运算。而同时 Gemini-PCS 所采用的 Split-and-fold 的思路也是将高维空间不断降维的映射过程，因此这个 Split-and-fold 过程可以移植到 MLE 的 Evaluation-form 上，一样可以达到折叠的效果，但是却完美避开了多项式 Basis 转换的复杂计算。

## 1. Gemini-PCS 的原理回顾

Gemini [BCHO22] 给出了一种 MLE 多项式到一元多项式的映射方法。下面是一个 MLE 的定义：

$$\tilde{f}(X_0, X_1, \ldots, X_{n-1}) = f_0 + f_1X_0 + f_2X_1 + f_3X_0X_1 + \cdots + f_{2^n-1}X_0X_1\cdots X_{n-1} \tag{5}$$

如果我们用长度为 $N = 2^n$ 的系数向量 $\vec{f}$ 来表示 $\tilde{f}$，那么我们可以定义一个 Univariate 多项式 $f(X)$，它具有和 $\tilde{f}(X_0, \ldots, X_{n-1})$ 相同的系数：

$$f(X) = f_0 + f_1X + f_2X^2 + \cdots + f_{2^n-1}X^{2^n-1} \tag{6}$$

我们用 $X = -X$ 代入上面的等式，可以得到：

$$f(-X) = f_0 - f_1X + f_2X^2 - \cdots - f_{2^n-1}X^{2^n-1} \tag{7}$$

那么分别把上面两个等式相加与相减，我们可以得到：

$$f(X) + f(-X) = 2(f_0 + f_2X^2 + \cdots + f_{2^n-2}X^{2^n-2}) \tag{8}$$

$$f(X) - f(-X) = 2X(f_1 + f_3X^2 + \cdots + f_{2^n-1}X^{2^n-2}) \tag{9}$$

再观察下 $\tilde{f}(X_0, X_1, \ldots, X_{n-1})$ 的「部分运算」 Partial Evaluation，即只实例化一个未知数 $X_0 = u_0$，即：

$$\begin{aligned}
\tilde{f}({\color{red}u_0}, X_1, \ldots, X_{n-1}) &= f_0 + f_1{\color{red}u_0} + f_2X_1 + f_3{\color{red}u_0}X_1 + f_4X_2 + f_5{\color{red}u_0}X_2 + f_6X_1X_2 + f_7{\color{red}u_0}X_1X_2 + \cdots + f_{2^n-1}{\color{red}u_0}X_1\cdots X_{n-1} \\
&= (f_0 + f_1{\color{red}u_0}) + (f_2 + f_3{\color{red}u_0})X_1 + (f_4 + f_5{\color{red}u_0})X_2 + (f_6 + f_7{\color{red}u_0})X_1X_2 + \cdots + (f_{2^n-2} + f_{2^n-1}{\color{red}u_0})X_1\cdots X_{n-1} \\
&= \tilde{f}^{(1)}(X_1, X_2, \ldots, X_{n-1})
\end{aligned}$$

这里 $\tilde{f}^{(1)}(X_1, X_2, \ldots, X_{n-1})$ 的系数向量为

$$(f_0 + f_1{\color{red}u_0}), (f_2 + f_3{\color{red}u_0}), (f_4 + f_5{\color{red}u_0}), (f_6 + f_7{\color{red}u_0}), \ldots, (f_{2^n-2} + f_{2^n-1}{\color{red}u_0}) \tag{11}$$

它正好和下面一个 UniPoly 的系数向量完全相同：

$$f^{(1)}(X) = (f_0 + f_1{\color{red}u_0}) + (f_2 + f_3{\color{red}u_0})X + (f_4 + f_5{\color{red}u_0})X^2 + \cdots + (f_{2^n-2} + f_{2^n-1}{\color{red}u_0})X^{2^{n-1}-1} \tag{12}$$

而一元多项式 $f^{(1)}(X)$ 加上 $f(X), f(-X)$，三者恰好满足下面的关系：

$$\begin{aligned}
f^{(1)}(X^2) &= (f_0 + f_1{\color{red}u_0}) + (f_2 + f_3{\color{red}u_0})X^2 + (f_4 + f_5{\color{red}u_0})X^4 + \cdots + (f_{2^n-2} + f_{2^n-1}{\color{red}u_0})X^{2^n-1} \\
&= \frac{1}{2}(f(X) + f(-X)) + u_0 \cdot \frac{1}{2X}(f(X) - f(-X))
\end{aligned} \tag{13}$$

所以说，如果我们要证明 $\tilde{f}^{(1)}(X_1, \ldots, X_{n-1})$ 为 $\tilde{f}(X_0, X_1, \ldots, X_{n-1})$ 的 Partial Evaluation，我们只需要证明上面的等式成立即可。

同理可推，如果我们要证明 $\tilde{f}(X_0, X_1, \ldots, X_{n-1}) = v$，那么我们可以引入若干个中间结果，即 Partial Evaluated 的 MLE 多项式，以及他们同构映射到的一元多项式

$$
\begin{aligned}
\tilde{f}^{(0)}(X_0, X_1, \ldots, X_{n-1}) &\mapsto f^{(0)}(X) \\
\tilde{f}^{(1)}(u_0, X_1, \ldots, X_{n-1}) &\mapsto f^{(1)}(X) \\
\tilde{f}^{(2)}(u_0, u_1, \ldots, X_{n-1}) &\mapsto f^{(2)}(X) \\
&\vdots \\
\tilde{f}^{(n-1)}(u_0, u_1, \ldots, u_{n-2}, X_{n-1}) &\mapsto f^{(n-1)}(X) \\
\tilde{f}^{(n)}(u_0, u_1, \ldots, u_{n-2}, u_{n-1}) &\mapsto f^{(n)}(X)
\end{aligned} \tag{14}
$$

其中最后一个 $f^{(n)}(X)$ 是一个常数多项式，它恰好为 $\tilde{f}(u_0, u_1, \ldots, u_{n-1})$ 完全的运算结果，即 $f^{(n)}(X) = v$。

并且，这些引入的一元多项式 $f^{(0)}(X), \ldots, f^{(n-1)}(X)$ ，它们相邻两项之间都满足下面的关系：

$$
f^{(i+1)}(X^2) = \frac{f^{(i)}(X) + f^{(i)}(-X)}{2} + u_i \cdot \frac{f^{(i)}(X) - f^{(i)}(-X)}{2X} \tag{15}
$$

其中 $\tilde{f}^{(n)}(u_0, u_1, \ldots, u_{n-2}) = v$ 为最终的 MLE 运算结果，而 $h^{(0)}(X)$ 为与 $\tilde{f}$ 同构的 UniPoly: $h^{(0)}(X) = f(X)$。

回到我们的证明目标：$\tilde{f}^{(n)}(u_0, u_1, \ldots, u_{n-1}) = v$，我们把这个运算过程的证明拆分为下面几个步骤：

- 构造一个 Univariate 多项式 $f(X)$，使其系数向量等于 $\tilde{f}$ 的系数，并构造多项式承诺 $\mathsf{cm}(f(X))$
- 多项式 $\tilde{f}$ 运算过程总共包含 $n$ 步的部分运算，每一次中间部分运算都会产生一个新的 MLE 多项式：$\tilde{f}^{(1)}, \tilde{f}^{(2)}, \ldots, \tilde{f}^{(n-1)}$
- 证明这些中间 MLE 所对应的 Univariate 多项式满足一个递推关系，这通过 Verifier 提供的随机数 $\beta$ 来随机抽样检查：

$$
f^{(i+1)}(\beta^2) = \frac{f^{(i)}(\beta) + f^{(i)}(-\beta)}{2} + u_i \cdot \frac{f^{(i)}(\beta) - f^{(i)}(-\beta)}{2\beta} \tag{16}
$$

- 证明 $f^{(n)}(\beta^2) = v$
- 证明所有的 Univariate 多项式 $\{f^{(i)}(X)\}$ 在 $X = \beta, X = -\beta, X = \beta^2$ 处的运算求值都正确

# 2. Evaluation-form 的线性折叠

如果我们在 PIOP 证明系统中采用的是 MLE 的 evaluation-form，那么我们采用需要一个类似 FFT 的转换运算，将其转换成 MLE 的 coefficient-form。这个转换运算的复杂度是 $O(N \log N)$。

在 Nova 的实现中，Setty 给出了 HyperKZG 的改进方案。它是利用了 Gemini PCS 方案背后的一个通用技术，这个技术与 $\tilde{f}$ 究竟是 Evaluation-form 还是 Coefficient-form 无关，只要他们能把计算过程拆分成多步的线性运算。

回顾上节介绍的 Gemini 论文中的这个 MLE 运算求值证明，它是将 $\tilde{f}(X_0, X_1, \ldots, X_{n-1})$ 的 Evaluation 过程分解为 $\log n$ 个步骤，然后把每一个中间 MLE 的系数向量都映射到一个 UniPoly 的系数，然后通过证明这些 UniPoly 之间的关系来保证 $\tilde{f}$ 运算的正确性。

对一个 MLE 多项式 $\tilde{f}(\vec{X})$ 的 Evaluation-form，让我们看下它的求值运算过程：

$$
\tilde{f}(X_0, X_1, \ldots, X_{n-1}) = a_0 E_0(X_0, X_1, \ldots, X_{n-1}) + a_1 E_1(X_0, X_1, \ldots, X_{n-1}) + \cdots + a_{2^n-1} E_{2^n-1}(X_0, X_1, \ldots, X_{n-1}) \tag{17}
$$

这里 $E_i(\vec{X})$ 为 Lagrange Polynomials，它的定义如下：

$$
E_i(X_0, X_1, \ldots, X_{n-1}) = \prod_{j=0}^{n-1} \Big(\mathsf{bits}(i)_j \cdot X_j + (1 - \mathsf{bits}(i)_j)(1 - X_j)\Big) \tag{18}
$$

其中 $\mathsf{bits}(i)_j$ 为 $i$ 的二进制表示的第 $j$ 位（注意，这里采用 Big-endian 的表示形式）。例如 $i = 5$，它的二进制表示为 $101$，那么 $\mathsf{bits}(5)_0 = 1$，$\mathsf{bits}(5)_1 = 0$，$\mathsf{bits}(5)_2 = 1$。

容易通过定义得知，$E_i(\vec{X})$ 满足下面的拆分性质（Tensor Structure）：

$$
E_{i\|j}(\vec{X}, \vec{Y}) = E_i(\vec{X}) \cdot E_i(\vec{Y}) \tag{19}
$$

其中 $k = i \,\|\, j$ 为 $k$ 的二进制位向量的分拆，例如，$i = 23$，它的二进制表示为 $10111$，可以拆分为 $10 \,\|\, 111$，或者记为 $23 = 2 \,\|\, 7$。根据拆分性质，我们可以得到：

$$
E_{i\|b}(X_0, (X_1, \ldots, X_{n-1})) = E_b(X_0) \cdot E_i(X_1, \ldots, X_{n-1}) \tag{20}
$$

那么我们观察下，如果 $\tilde{f}$ 在一次 Partial Evaluation 之后的样子，令 $X_0 = u_0$，$\vec{X} = (X_1, \ldots, X_{n-1})$：

$$
\begin{aligned}
\tilde{f}(u_0, \vec{X}) &= a_0 E_0(u_0, \vec{X}) + a_1 E_1(u_0, \vec{X}) + \cdots + a_{2^n-2} E_{2^n-2}(u_0, \vec{X}) + a_{2^n-1} E_{2^n-1}(u_0, \vec{X}) \\
&= \Big(a_0 E_0(u_0)\Big) \cdot E_0(\vec{X}) + \Big(a_1 E_1(u_0)\Big) \cdot E_0(\vec{X}) + \cdots + \Big(a_{2^n-1} E_0(u_0)\Big) \cdot E_{2^{n-1}-1}(\vec{X}) + \Big(a_{2^n-1} E_1(u_0)\Big) \cdot E_{2^{n-1}-1}(\vec{X}) \\
&= \Big(a_0(1-u_0)\Big) \cdot E_0(\vec{X}) + \Big(a_1 u_0\Big) \cdot E_0(\vec{X}) + \cdots + \Big(a_{2^n-2}(1-u_0)\Big) \cdot E_{2^{n-1}-1}(\vec{X}) + \Big(a_{2^n-1} u_0\Big) \cdot E_{2^{n-1}-1}(\vec{X}) \\
&= \Big(a_0(1-u_0)\Big) \cdot E_0(\vec{X}) + \Big(a_1 u_0\Big) \cdot E_0(\vec{X}) + \cdots + \Big(a_{2^n-2}(1-u_0)\Big) \cdot E_{2^{n-1}-1}(\vec{X}) + \Big(a_{2^n-1} u_0\Big) \cdot E_{2^{n-1}-1}(\vec{X}) \\
&= \Big(a_0(1-u_0) + a_1 u_0\Big) \cdot E_0(\vec{X}) + \cdots + \Big(a_{2^n-2}(1-u_0) + a_{2^n-1} u_0\Big) \cdot E_{2^{n-1}-1}(\vec{X}) \\
&= \tilde{h}^{(1)}(\vec{X})
\end{aligned} \tag{21}
$$

可以看出，$\tilde{h}^{(1)}(\vec{X})$ 的 Evaluation 点值向量为：

$$\Big(a_0(1-u_0)+a_1u_0\Big),\Big(a_2(1-u_0)+a_3u_0\Big),\ldots,\Big(a_{2^n-2}(1-u_0)+a_{2^n-1}u_0\Big) \tag{22}$$

对比下 $\tilde{f}(u_0,\vec{X})$ 的系数向量，我们会发现两者都是只有原先长度的一半，但是只是折半运算的方式不一样，前者为 $\Big((1-u)\cdot a+u\cdot b\Big)$，后者为 $\Big(a+u\cdot b\Big)$，这个新的折半运算方法并没有阻止我们采用 Gemini-PCS 的技术来保证 Split-and-fold 的正确性。 我们仍然可以引入 $n-1$ 个部分运算的 MLE 多项式，以及将他们的 Evaluations 映射到多个 UniPoly 的系数：

$$\begin{aligned}
\tilde{h}^{(0)}(X_0,X_1,\ldots,X_{n-1})&\mapsto h^{(0)}(X)\\
\tilde{h}^{(1)}(u_0,X_1,\ldots,X_{n-1})&\mapsto h^{(1)}(X)\\
\tilde{h}^{(2)}(u_0,u_1,\ldots,X_{n-1})&\mapsto h^{(2)}(X)\\
&\vdots\\
\tilde{h}^{(n-1)}(u_0,u_1,\ldots,u_{n-1})&\mapsto h^{(n-1)}(X)
\end{aligned} \tag{23}$$

其中 $\tilde{h}^{(0)}(\vec{X})=\tilde{f}(\vec{X})$，而 $\tilde{h}^{(n-1)}(u_0,u_1,\ldots,u_{n-1})=v$,

并且给出他们的 Evaluation form 之间满足一个「相似的」递推关系：

$$h^{(i+1)}(X^2)=(1-u_i)\cdot\frac{h^{(i)}(X)+h^{(i)}(-X)}{2}+u_i\cdot\frac{h^{(i)}(X)-h^{(i)}(-X)}{2X} \tag{24}$$

因此，Verifier 接下来可以发送一个唯一的随机挑战点 $X=\beta$，来检查 $\{h^{(i)}(\beta)\}$ 之间是否满足上面等式所定义的递推关系。这里就可以对接上 KZG10 这个针对 Univariate 多项式的 PCS 方案。完成剩下的证明。

# 3. 协议描述

本小节给出这个协议的流程描述。协议是证明一个 MLE 多项式 $\tilde{f}(X_0,X_1,\ldots,X_{n-1})$ 在一个给定的点 $(u_0,u_1,\ldots,u_{n-1})$ 处的运算求值结果等于 $v$ 。

## Witness 输入：

1. $\vec{a}=(a_0,a_1,\ldots,a_{2^n-1})$: 多项式 $f(X)$ 的系数向量。

$$f(X)=a_0+a_1X+a_2X^2+\cdots+a_{2^n-1}X^{2^n-1} \tag{25}$$

## 公共输入:

1. $C_f$: MLE 多项式 $\tilde{f}(X_0,X_1,\ldots,X_{n-1})$ 的同构多项式 $f(X)$ 的承诺,

$$C_f=\mathsf{KZG10.Commit}(f(X)) \tag{26}$$

2. $\vec{u}=(u_0,u_1,\ldots,u_{n-1})$: 运算求值点的坐标
3. $v=\tilde{f}(u_0,u_1,\ldots,u_{n-1})$: 运算求值的结果

## Round 1

1. Prover 计算 $h^{(1)}(X),h^{(2)}(X),\ldots,h^{(n-1)}(X)$

$$\begin{aligned}
h^{(1)}(X)&=((1-u_0)a_0+u_0a_1)+((1-u_0)a_2+u_0a_3)X+\cdots+((1-u_0)a_{2^n-2}+u_0a_{2^n-1})X^{2^{n-1}-1}\\
h^{(2)}(X)&=((1-u_1)a_0^{(1)}+u_1a_1^{(1)})+((1-u_1)a_2^{(1)}+u_1a_3^{(1)})X+\ldots+((1-u_1)a_{2^{n-1}-2}^{(1)}+u_1a_{2^{n-1}-1}^{(1)})X^{2^{n-2}-1}\\
&\vdots\\
h^{(n-1)}(X)&=((1-u_{n-1})a_0^{(n-2)}+u_{n-1}a_1^{(n-2)})+((1-u_{n-1})a_2^{(n-2)}+u_{n-1}a_3^{(n-2)})X
\end{aligned} \tag{27}$$

这里 $(a_0^{(1)},a_1^{(1)},\ldots,a_{2^{n-1}-1}^{(1)})$ 代表 $h^{(1)}(X)$ 的点值向量，同理，$(a_0^{(i)},\ldots,a_{2^{n-i}}^{(i)})$ 代表 $h^{(i)}(X)$ 的点值向量。

2. Prover 输出承诺 $\Big(C_{h^{(1)}},C_{h^{(2)}},\ldots,C_{h^{(n-1)}}\Big)$

## Round 2

1. Verifier 发送随机挑战数 $\beta\in\mathbb{F}$,
2. Prover 计算并发送 $\Big(h^{(0)}(\beta),h^{(0)}(-\beta)\Big)$, $\Big(h^{(1)}(\beta),h^{(1)}(-\beta),h^{(1)}(\beta^2)\Big)$, $\Big(h^{(2)}(\beta),h^{(2)}(-\beta),h^{(2)}(\beta^2)\Big)$, ..., $\Big(h^{(n-2)}(\beta),h^{(n-2)}(-\beta),h^{(n-2)}(\beta^2)\Big)$
3. Prover 证明上述的 Evaluation 的正确性，并发送证明：$\Big(\pi_{0,\beta},\pi_{0,-\beta}\Big)$, $\Big(\pi_{1,\beta},\pi_{1,-\beta},\pi_{1,\beta^2}\Big),\ldots,\Big(\pi_{n-1,\beta},\pi_{n-1,-\beta},\pi_{n-1,\beta^2}\Big)$

## Verification

1. 验证 $h^{(0)},\ldots,h^{(n-1)}$ 在 $X=\beta,X=-\beta$ 与 $X=\beta^2$ 处的取值是否满足递推式：

$$
\begin{aligned}
h^{(1)}(\beta^2) &\overset{?}{=} \frac{h^{(0)}(\beta)+h^{(0)}(-\beta)}{2} + u_0\cdot\frac{h^{(0)}(\beta)-h^{(0)}(-\beta)}{2\beta} \\
h^{(2)}(\beta^2) &\overset{?}{=} \frac{h^{(1)}(\beta)+h^{(1)}(-\beta)}{2} + u_1\cdot\frac{h^{(1)}(\beta)-h^{(1)}(-\beta)}{2\beta} \\
&\vdots \\
h^{(n-1)}(\beta^2) &\overset{?}{=} \frac{h^{(n-2)}(\beta)+h^{(n-2)}(-\beta)}{2} + u_{n-2}\cdot\frac{h^{(n-2)}(\beta)-h^{(n-2)}(-\beta)}{2\beta} \\
v &\overset{?}{=} \frac{h^{(n-1)}(\beta)+h^{(n-1)}(-\beta)}{2} + u_{n-1}\cdot\frac{h^{(n-1)}(\beta)-h^{(n-1)}(-\beta)}{2\beta}
\end{aligned} \tag{28}
$$

2. 根据 $C_f, C_{h^{(1)}}, C_{h^{(2)}}, \ldots, C_{h^{(n-1)}}$，验证多项式取值运算的正确性：

$$
\begin{array}{llllll}
\mathsf{KZG10.Verify} & (C_f, & \beta, & h^{(0)}(\beta), & \pi_{0,\beta}) & \overset{?}{=} 1 \\
\mathsf{KZG10.Verify} & (C_f, & -\beta, & h^{(0)}(-\beta), & \pi_{0,-\beta}) & \overset{?}{=} 1 \\
\mathsf{KZG10.Verify} & (C_{h^{(1)}}, & \beta, & h^{(1)}(\beta), & \pi_{1,\beta}) & \overset{?}{=} 1 \\
\mathsf{KZG10.Verify} & (C_{h^{(1)}}, & -\beta, & h^{(1)}(-\beta), & \pi_{1,-\beta}) & \overset{?}{=} 1 \\
\mathsf{KZG10.Verify} & (C_{h^{(1)}}, & \beta^2, & h^{(1)}(\beta^2), & \pi_{1,\beta^2}) & \overset{?}{=} 1 \\
 & & \vdots & & & \\
\mathsf{KZG10.Verify} & (C_{h^{(n-1)}}, & \beta, & h^{(n-1)}(\beta), & \pi_{n-1,\beta}) & \overset{?}{=} 1 \\
\mathsf{KZG10.Verify} & (C_{h^{(n-1)}}, & -\beta, & h^{(n-1)}(-\beta), & \pi_{n-1,-\beta}) & \overset{?}{=} 1 \\
\mathsf{KZG10.Verify} & (C_{h^{(n-1)}}, & \beta^2, & h^{(n-1)}(\beta^2), & \pi_{n-1,\beta^2}) & \overset{?}{=} 1
\end{array} \tag{29}
$$

## 4. 协议优化

在上述协议中，有几个点可以优化：

1. Prover 没有必要发送除 $h^{(0)}(\beta^2)$ 之外的其它所有 $X=\beta^2$ 的运算值，因为 Verifier 可以通过下面的递推关系计算出来。这样可以减少 Prover 的通信量，同时 Verifier 在计算的过程相当于同时进行了验证，因而可以省去递推式验证过程。

$$
h^{(i+1)}(\beta^2) = \frac{h^{(i)}(\beta)+h^{(i)}(-\beta)}{2} + u_i\cdot\frac{h^{(i)}(\beta)-h^{(i)}(-\beta)}{2\beta} \tag{30}
$$

2. Prover 可以通过一个随机数 $\gamma$ 把 $h^{(0)}(X),\ldots,h^{(n-1)}(X)$ 聚合在一起，得到 $h(X)$，然后证明 $h(X)$ 在 $X=\beta,-\beta,\beta^2$ 处的取值。这样可以避免让 Prover 发送 $3n-1$ 个独立的 KZG10 的 Evaluation 证明，而是只需要发送三个 Evaluataion 证明。

下面是优化过后的协议描述

### Round 1

Prover 计算 $h^{(1)}(X), h^{(2)}(X), \ldots, h^{(n-1)}(X)$

$$
\begin{aligned}
h^{(1)}(X) &= ((1-u_0)a_0+u_0a_1) + ((1-u_0)a_2+u_0a_3)X + \cdots + ((1-u_0)a_{2^n-2}+u_0a_{2^n-1})X^{2^{n-1}-1} \\
h^{(2)}(X) &= ((1-u_1)a_0^{(1)}+u_1a_1^{(1)}) + ((1-u_1)a_2^{(1)}+u_1a_3^{(1)})X + \ldots + ((1-u_1)a_{2^{n-1}-2}^{(1)}+u_1a_{2^{n-1}-1}^{(1)})X^{2^{n-2}-1} \\
&\vdots \\
h^{(n-1)}(X) &= ((1-u_{n-1})a_0^{(n-2)}+u_{n-1}a_1^{(n-2)}) + ((1-u_{n-1})a_2^{(n-2)}+u_{n-1}a_3^{(n-2)})X
\end{aligned} \tag{31}
$$

Prover 发送多项式承诺 $\left(C_{h^{(1)}}, C_{h^{(2)}}, \ldots, C_{h^{(n-1)}}\right)$

### Round 2

1. Verifier 发送随机挑战数 $\beta\in\mathbb{F}$,
2. Prover 计算并发送 $\left(h^{(0)}(\beta), h^{(0)}(-\beta), h^{(0)(\beta^2)}\right)$, $\left(h^{(1)}(\beta), h^{(1)}(-\beta)\right)$, $\left(h^{(2)}(\beta), h^{(2)}(-\beta)\right)$, …, $\left(h^{(n-1)}(\beta), h^{(n-1)}(-\beta)\right)$

### Round 3

1. Verifier 发送随机挑战数 $\gamma\in\mathbb{F}$,
2. Prover 计算聚合多项式 $h(X)$,

$$
h(X) = h^{(0)}(X) + \gamma\cdot h^{(1)}(X) + \cdots + \gamma^{n-1}\cdot h^{(n-1)}(X) \tag{32}
$$

3. Prover 计算 $v_\beta, v_{-\beta}, v_{\beta^2}$

$$
\begin{aligned}
v_\beta &= h^{(0)}(\beta) + \gamma\cdot h^{(1)}(\beta) + \cdots + \gamma^{n-1}\cdot h^{(n-1)}(\beta) \\
v_{-\beta} &= h^{(0)}(-\beta) + \gamma\cdot h^{(1)}(-\beta) + \cdots + \gamma^{n-1}\cdot h^{(n-1)}(-\beta) \\
v_{\beta^2} &= h^{(0)}(\beta^2) + \gamma\cdot h^{(1)}(\beta^2) + \cdots + \gamma^{n-1}\cdot h^{(n-1)}(\beta^2)
\end{aligned} \tag{33}
$$

4. Prover 计算 $h^*(x)$

$$
h^*(X) = h(\beta)\cdot L_\beta(X) + h(-\beta)\cdot L_{-\beta}(X) + h(\beta^2)\cdot L_{\beta^2}(X) \tag{34}
$$

这里假设 Domain $D=\{\beta,-\beta,\beta^2\}$，而 $\{L_\beta(X), L_{-\beta}(X), L_{\beta^2}(X)\}$ 为 $D$ 上的 Lagrange Polynomials。那么 $h^*(X)$ 满足下面的等式：存在一个商多项式 $q(X)$，使得

$$
\mathsf{EQ1}: h(X) - h^*(X) = q(X)\cdot(X-\beta)(X+\beta)(X-\beta^2) \tag{35}
$$

5. Prover 计算商多项式 $q(X)$ 并发送其多项式承诺 $C_q = \mathsf{cm}(q(X))$

## Round 4

1. Verifier 发送随机挑战数 $\zeta \in \mathbb{F}_p$
2. Prover 计算 EQ1 的线性化多项式 $r_\zeta(X)$，满足 $r_\zeta(\zeta)=0$,

$$
r_\zeta(X) = h(X) - h^*(\zeta) - q(X)\cdot(\zeta-\beta)(\zeta+\beta)(\zeta-\beta^2) \tag{36}
$$

3. Prover 发送线性化多项式的承诺 $C_r = [r(x)]_1$
4. Prover 计算商多项式 $w(X)$ 满足：

$$
w(X) = \frac{r_\zeta(X)}{X-\zeta} \tag{37}
$$

Prover 发送 $C_w = [w(x)]_1$

## Verification

1. 计算 $\left(h^{(1)}(\beta^2), h^{(2)}(\beta^2), \ldots, h^{(n-1)}(\beta^2)\right)$,

$$
\begin{aligned}
h^{(1)}(\beta^2) &= \frac{h^{(0)}(\beta)+h^{(0)}(-\beta)}{2} + u_0\cdot\frac{h^{(0)}(\beta)-h^{(0)}(-\beta)}{2\beta} \\
h^{(2)}(\beta^2) &= \frac{h^{(1)}(\beta)+h^{(1)}(-\beta)}{2} + u_1\cdot\frac{h^{(1)}(\beta)-h^{(1)}(-\beta)}{2\beta} \\
&\vdots \\
h^{(n-1)}(\beta^2) &= \frac{h^{(n-2)}(\beta)+h^{(n-2)}(-\beta)}{2} + u_{n-2}\cdot\frac{h^{(n-2)}(\beta)-h^{(n-2)}(-\beta)}{2\beta}
\end{aligned} \tag{38}
$$

2. 计算 $h(\beta), h(-\beta), h(\beta^2)$,

$$
\begin{aligned}
h(\beta) &= h^{(0)}(\beta) + \gamma\cdot h^{(1)}(\beta) + \cdots + \gamma^{n-1}\cdot h^{(n-1)}(\beta) \\
h(-\beta) &= h^{(0)}(-\beta) + \gamma\cdot h^{(1)}(-\beta) + \cdots + \gamma^{n-1}\cdot h^{(n-1)}(-\beta) \\
h(\beta^2) &= h^{(0)}(\beta^2) + \gamma\cdot h^{(1)}(\beta^2) + \cdots + \gamma^{n-1}\cdot h^{(n-1)}(\beta^2)
\end{aligned} \tag{39}
$$

3. 计算 $c(X)$ 在 $X=\zeta$ 处的取值 $c(\zeta)$,
4. 计算 $C_h = C_f + \gamma\cdot C_{h^{(1)}} + \gamma^2\cdot C_{h^{(2)}} + \cdots + \gamma^{n-1}\cdot C_{h^{(n-1)}}$
5. 计算 $C_r = [r_\zeta(x)]_1$ 的承诺：

$$
C_r = [r_\zeta(x)]_1 = C_h - c(\zeta)\cdot[1]_1 - (\zeta-\beta)(\zeta+\beta)(\zeta-\beta^2)\cdot C_q \tag{40}
$$

4. 验证 $C_h$ 与 $C_w$ 的关系：

$$
e(C_r + \zeta\cdot C_w, [1]_2) \stackrel{?}{=} e(C_w, [x]_2) \tag{41}
$$

# 5. 性能分析

Proof size: $(n+1)\cdot\mathbb{G}_1$ + $(2n+1)\cdot\mathbb{F}$

$$
\pi = \left(C_{h^{(1)}}, C_{h^{(2)}}, \ldots, C_{h^{(n-1)}}, C_q, C_w, \{h^{(i)}(\beta), h^{(i)}(-\beta)\}_{i=0}^{n-1}, h^{(0)}(\beta^2)\right) \tag{42}
$$

Verifier Cost: $(2n+2)\cdot\mathsf{EccMul}^{\mathbb{G}_1}$ + $(3n)\cdot\mathbb{F} + 2\cdot\mathsf{Pairing}$

1. $2n\cdot\mathbb{F}.\mathsf{Mult}$
2. 计算 $h(\beta), h(-\beta), h(\beta^2)$： $3n\cdot\mathbb{F}.\mathsf{Mult}$
3. 计算 $c(\zeta)$： $O(1)\cdot\mathbb{F}.\mathsf{Mult}$

4. 计算 $C_h$: $n \cdot \mathbb{G}_1$ Scalar Multiplication
5. 计算 $P$: $2 \cdot \mathbb{G}_1$ Scalar Multiplication

## References