Libra-PCS

1. MLE 多项式

当然一个 MLE 多项式也可以采用「系数式」来表示，即 Coefficients form，表示如下：

$$\tilde{f}(X_0, X_1, \ldots, X_{n-1}) = \sum_{i_0=0}^{1}\sum_{i_1=0}^{1}\cdots\sum_{i_{n-1}=0}^{1} f_{i_0 i_1 \cdots i_{n-1}} X_0^{i_0} X_1^{i_1} \cdots X_{n-1}^{i_{n-1}} \tag{1}$$

对于上图三维 MLE 多项式的例子，我们可以将它写为：

$$\tilde{f}(X_0, X_1, X_2) = f_0 + f_1 X_0 + f_2 X_1 + f_3 X_2 + f_4 X_0 X_1 + f_5 X_0 X_2 + f_6 X_1 X_2 + f_7 X_0 X_1 X_2 \tag{2}$$

其中 $(f_0, f_1, \ldots, f_7)$ 为 MLE 多项式的系数向量。注意因为 MLE 多项式属于多元多项式（Multivariate Polynomial），任何表示方式都需要事先确定多项式中的项的排序顺序，本文以及后续讨论我们都基于 Lexicographic Order。

对于 MLE 多项式的「点值式」表示，我们可以定义为：

$$\tilde{f}(X_0, X_1, \ldots, X_{n-1}) = \sum_{i_0=0}^{1}\sum_{i_1=0}^{1}\cdots\sum_{i_{n-1}=0}^{1} a_{i_0 i_1 \cdots i_{n-1}} \cdot eq(i_0, i_1, \ldots, i_{n-1}, X_0, X_1, \ldots, X_{n-1}) \tag{3}$$

其中 eq 为 一组关于 n 维 Boolean HyperCube $\{0,1\}^n$ 的 Lagrange Polynomial：

$$eq(i_0, i_1, \ldots, i_{n-1}, X_0, X_1, \ldots, X_{n-1}) = \prod_{j=0}^{n-1}\Big((1-i_j)\cdot(1-X_j) + i_j \cdot X_j\Big) \tag{4}$$

MLE 多项式在「点值式」和「系数式」之间存在 $N\log(N)$ 的转换算法，这里不再深入讨论。

2. MLE 多项式的除法

对于 Univariate 多项式 $f(X) \in \mathbb{F}_p[X]$，如果 $f(X)$ 在 $X = u$ 处的运算值为 v，那么我们可以有下面的等式：

$$f(X) = q(X)\cdot(X-u) + v \tag{5}$$

其中 $q(X)$ 是 $f(X)$ 除以 $(X-u)$ 的商多项式，v 是余数。

我们可以简单推导下上面的这个等式。当我们把 $X = u$ 代入到等式，可以得到 $f(u) = v$。这说明多项式的求值问题等价于为多项式的除法余数。那么，我们可以让 $f(X)$ 减去这个余数，那么得到的多项式 $g(X) = f(X) - v$，显然可以被 $(X-u)$ 整除，即存在一个商多项式，记为 $q(X)$。

那么对于一个 Multivariate 多项式 $f(X_0, X_1, \ldots, X_{n-1})$，论文 [PST13] 给出了一个类似的除法关系等式：

$$f(X_0, X_1, \ldots, X_{n-1}) - f(u_0, u_1, \ldots, u_{n-1}) = \sum_{k=0}^{n-1} q_k(X_0, X_1, \ldots, X_{n-1})\cdot(X_k - u_k) \tag{6}$$

如果 $f(X_0, X_1, \ldots, X_{n-1})$ 是一个 MLE 多项式，那么它可以被简化为下面的等式：

$$\begin{aligned}\tilde{f}(X_0, X_1, \ldots, X_{n-1}) - \tilde{f}(u_0, u_1, \ldots, u_{n-1}) = &\ \tilde{q}_{n-1}(X_0, X_1, \ldots, X_{n-2})\cdot(X_{n-1} - u_{n-1}) \\ &+ \tilde{q}_{n-2}(X_0, X_1, \ldots, X_{n-3})\cdot(X_{n-2} - u_{n-2}) \\ &+ \cdots \\ &+ \tilde{q}_1(X_0)\cdot(X_1 - u_1) \\ &+ \tilde{q}_0\cdot(X_0 - u_0)\end{aligned} \tag{7}$$

这是因为 MLE 多项式 $\tilde{f}(X_0,X_1,\ldots,X_{n-1})$ 中每一个未知数 X_k 的最高次数为 1，对于 $f(X_0,X_1,\ldots,X_k)$，它除以 (X_k-u_k) 这个因式之后，余数多项式中将不再含有未知数 X_k ，所以当 $f(X_0,X_1,\ldots,X_{n-1})$ 按顺序除以 $(X_{n-1}-u_{n-1})$ 到 (X_0-u_0) 这些因式，我们得到的商多项式和余数多项式中的未知数数量将依次减少，直到最后得到一个常数的商多项式 $\tilde{q}_0$ ，当然在 n 次除法结束之后，会出现一个常数的余数多项式，而它正好是 MLE 多项式在 $(u_0,u_1,\ldots,u_{n-1})$ 处的求值。这被称为 Ruffini's 法则 [Ruffini]。

我们假设这个最后的求值为 v ，即

$$\tilde{f}(u_0,u_1,\ldots,u_{n-1})=v \tag{8}$$

3. Libra-PCS 的构造

类比 KZG10 的构造，Libra-PCS 也需要一个结构化的 SRS。需要注意的是，Libra 论文基于 KOE 安全假设，而本文介绍的是基于 AGM 的安全假设，所以方案的 SRS 具有更小的尺寸，同时也可以在 AGM 假设下证明方案的正确性和 Extractibility 性质。

它由一个 Trusted Setup 来产生：

$$SRS=\begin{pmatrix}[1]_1,[\tau_0]_1,[\tau_1]_1,[\tau_0\tau_1]_1,[\tau_2]_1,[\tau_0\tau_2]_1,[\tau_1\tau_2]_1,[\tau_0\tau_1\tau_2]_1,\ldots,[\tau_0\tau_1\cdots\tau_{n-1}]_1,[\xi]_1\\ [1]_2,\ [\tau_0]_2,\ [\tau_1]_2,\ [\tau_2]_2,\ \ldots,\ [\tau_{n-1}]_2,\ [\xi]_2\end{pmatrix} \tag{9}$$

有了这个 SRS，我们可以将一个 n 元 MLE 多项式 $\tilde{f}(X_0,X_1,\ldots,X_{n-1})$ 进行承诺计算。即将其长度为 $N=2^n$ 的系数向量用 SRS 作为 Basis 进行线性组合，得到 $\mathbb{G}_1$ 上的一个元素。

$$\begin{aligned}\tilde{f}(X_0,X_1,\ldots,X_{n-1})=&\ f_0+f_1X_0+f_2X_1+f_3X_0X_1+f_4X_2+f_5X_0X_2+f_6X_1X_2\\&+f_7X_0X_1X_2+\cdots+f_{N-1}X_0X_1\cdots X_{n-1}\end{aligned} \tag{10}$$

其中 $\vec{f}=(f_0,f_1,f_2,\ldots,f_{N-1})$ 是 $\tilde{f}(X_0,X_1,\ldots,X_{n-1})$ 的系数向量。那么 Commitment 的计算方式如下：

$$\begin{aligned}F=\mathsf{Commit}(\tilde{f}(X_0,X_1,\ldots,X_{n-1});\rho)=&\ f_0\cdot[1]_1+f_1\cdot[\tau_0]_1+f_2\cdot[\tau_1]_1+f_3\cdot[\tau_0\tau_1]_1\\&+f_4\cdot[\tau_2]_1+f_5\cdot[\tau_0\tau_2]_1+f_6\cdot[\tau_1\tau_2]_1+f_7\cdot[\tau_0\tau_1\tau_2]_1\\&+\cdots+f_{N-1}\cdot[\tau_0\tau_1\cdots\tau_{n-1}]_1+\rho\cdot[\xi]_1\end{aligned} \tag{11}$$

这里 ρ 是一个随机数，用来隐藏 $\vec{f}$ 的信息。

然后如果 Prover 要证明 $\tilde{f}(u_0,u_1,\ldots,u_{n-1})=v$，他需要承诺 $\tilde{q}_0,\tilde{q}_1,\ldots,\tilde{q}_{n-1}$ 这些商多项式。

$$\begin{aligned}Q_0&=\mathsf{Commit}(\tilde{q}_0;\eta_0)=\tilde{q}_0\cdot[1]_1+\eta_0\cdot[\xi]_1\\Q_1&=\mathsf{Commit}(\tilde{q}_1;\eta_1)=\tilde{q}_{1,0}\cdot[1]_1+\tilde{q}_{1,1}\cdot[\tau_0]_1+\eta_1\cdot[\xi]_1\\Q_2&=\mathsf{Commit}(\tilde{q}_2;\eta_2)=\tilde{q}_{2,0}\cdot[1]_1+\tilde{q}_{2,1}\cdot[\tau_0]_1+\tilde{q}_{2,2}\cdot[\tau_1]_1+\tilde{q}_{2,3}\cdot[\tau_0\tau_1]_1+\eta_2\cdot[\xi]_1\\\cdots&=\cdots\\Q_{n-1}&=\mathsf{Commit}(\tilde{q}_{n-1};\eta_{n-1})=\tilde{q}_{n-1,0}\cdot[1]_1+\tilde{q}_{n-1,1}\cdot[\tau_0]_1+\tilde{q}_{n-1,2}\cdot[\tau_1]_1+\cdots\\&\qquad+\tilde{q}_{n-1,2^{n-1}-1}\cdot[\tau_0\tau_1\cdots\tau_{n-2}]_1+\eta_{n-1}\cdot[\xi]_1\end{aligned} \tag{12}$$

为了保证 Verifier 可以验证这些 Commitment，并且允许每一个 Commitment 都带有一个 Blinding Factor。但是增加了这些 Blinding Factor 后，需要修改下 MLE 多项式的除法等式：

$$\begin{aligned}&\ (q_0+\eta_0\xi)(X_0-u_0)+(q_1+\eta_1\xi)(X_1-u_1)+\cdots+(q_{n-1}+\eta_{n-1}\xi)(X_{n-1}-u_{n-1})\\=&\ q_0(X_0-u_0)+q_1(X_0)(X_1-u_1)+\cdots+q_{n-1}(X_0,\ldots,X_{n-2})(X_{n-1}-u_{n-1})\\&+\eta_0\xi(X_0-u_0)+\eta_1\xi(X_1-u_1)+\cdots+\eta_{n-1}\xi(X_{n-1}-u_{n-1})\\=&\ f(\vec{X})-f(\vec{u})+(\eta_0(X_0-u_0)+\eta_1(X_1-u_1)+\cdots+\eta_{n-1}(X_{n-1}-u_{n-1}))\cdot\xi\\=&\ f(\vec{X})+\rho\xi-f(\vec{u})+(\eta_0(X_0-u_0)+\eta_1(X_1-u_1)+\cdots+\eta_{n-1}(X_{n-1}-u_{n-1})-\rho)\cdot\xi\end{aligned} \tag{13}$$

因此 Prover 还需要计算一个额外的 Commitment，搜集所有的 Blinding Factor，即上面等式右边标红色的部分：

$$\begin{aligned}R=&\ \rho\cdot[1]_1+(-\eta_0\cdot[\tau_0]_1+(\eta_0\cdot u_0)[1]_1)+(-\eta_1\cdot[\tau_1]_1+(\eta_1\cdot u_1)[1]_1)\\&+\cdots+(-\eta_{n-1}\cdot[\tau_{n-1}]_1+(\eta_{n-1}\cdot u_{n-1})[1]_1)\end{aligned} \tag{14}$$

这里的 R 显然也是一个 $\mathbb{G}_1$ 上的元素。

所以当 Prover 发送 $(Q_0, Q_1, \ldots, Q_{n-1}, R)$ 给 Verifier 之后，Verifier 可以通过下面的 Pairing 等式来验证：

$$e(F - v[1]_1, [1]_2) \overset{?}{=} e(R, [\xi]_2) + \sum_{i=0}^{n-1} e(Q_i, [\tau_i]_2 - u_i[1]_2) \tag{15}$$

4. 支持 MLE Evaluation 形式

上面的 Libra-PCS 中，Prover 计算 Commitment 的时候，需要先得到 MLE 多项式的「系数形式」 Coefficient Form，然后才能计算 Commitment。但是在很多 Sumcheck 协议中，采用的是 MLE 的 Evaluation Form，即点值式。也就是说，n 元 MLE 多项式用它在 n-维 Boolean Hypercube 上的运算值来表示：

$$\tilde{f}(X_0, X_1, \ldots, X_{n-1}) = \sum_{i_0=0}^{1}\sum_{i_1=0}^{1}\cdots\sum_{i_{n-1}=0}^{1} a_{i_0 i_1 \cdots i_{n-1}} \cdot eq(i_0, i_1, \ldots, i_{n-1}, X_0, X_1, \ldots, X_{n-1}) \tag{16}$$

如果我们需要将 MLE 多项式的 Evaluation form 转换到 Coefficient form，那么这个转换算法需要 $O(n \cdot 2^n)$ 的时间复杂度。那么能不能直接支持 MLE 的 Evaluation form 呢？

MLE 的 Evaluation form 会带来两个困难问题，第一个是如何根据 SRS 来计算一个 Evaluation Form 的承诺，第二个难点是如何计算 MLE 多项式的 n 次除法，得到 n 个商多项式 $q_0, q_1, \ldots, q_{n-1}$。

我们先看看第一个问题，如何针对 Evaluation Form 计算 Commitment。这里有两种方式。比较直接的方式是，在计算 SRS 的时候，直接产生 MLE 多项式的 Multilinear Basis 的 Commitment，而不是 Monomial Basis 的 Commitment。

比如，假设 n=3，那么我们产生一组新的 SRS 参数，用来计算 三元 MLE 多项式的 Evaluation Form 的 Commitment。

$$SRS^{(3)} = ([eq_0(\tau_0,\tau_1,\tau_2)]_1, [eq_1(\tau_0,\tau_1,\tau_2)]_1, [eq_2(\tau_0,\tau_1,\tau_2)]_1, \ldots, [eq_7(\tau_0,\tau_1,\tau_2)]_1, [\xi]_1) \tag{17}$$

为了简化表示，我们用 $eq_i(\vec{X})$ 来表示 $eq(i_0, i_1, \ldots, i_{n-1}, X_0, X_1, \ldots, X_{n-1})$。假设我们有一个多项式 $f(X_0, X_1, X_2)$：假设我们有一个多项式 $f(X_0, X_1, X_2)$：

$$f(X_0,X_1,X_2) = a_0 \cdot eq_0(X_0,X_1,X_2) + a_1 \cdot eq_1(X_0,X_1,X_2) + \cdots + a_7 \cdot eq_7(X_0,X_1,X_2) \tag{18}$$

于是我们计算它的 Commitment 如下：

$$\mathsf{cm}(f) = a_0 \cdot [eq_0(\tau_0,\tau_1,\tau_2)]_1 + a_1 \cdot [eq_1(\tau_0,\tau_1,\tau_2)]_1 + \cdots + a_7 \cdot [eq_7(\tau_0,\tau_1,\tau_2)]_1 + \rho \cdot [\xi]_1 \tag{19}$$

不过，我们需要意识到 $SRS^{(3)}$ 并不够，因为 $eq_i(\vec{X})$ 这种 Multilinear Basis 是和 Domain 的大小相关。我们要能发现，如果我们要承诺一个 二元多项式的 Evaluation Form，那么我们不能使用 $SRS^{(3)}$ 来计算。而需要再产生一组针对 二元 MLE 多项式的 SRS 参数，记为 $SRS^{(2)}$

$$SRS^{(2)} = ([eq_0(\tau_0,\tau_1)]_1, [eq_1(\tau_0,\tau_1)]_1, \ldots, [eq_3(\tau_0,\tau_1)]_1) \tag{20}$$

同样，我们还需要一组针对一元多项式的 SRS 参数，记为 $SRS^{(1)}$：

$$SRS^{(1)} = ([eq_0(\tau_0)]_1, [eq_1(\tau_0)]_1) \tag{21}$$

最后对于常数多项式，我们只需要 $[1]_1$ 作为 Basis来计算承诺即可。我们把所有这些基于 k-MLE 的 $SRS^{(k)}$ 合并在一起，记为 SRS^*：

$$SRS^* = \begin{pmatrix} [eq_0(\tau_0,\tau_1,\tau_2)]_1, [eq_1(\tau_0,\tau_1,\tau_2)]_1, [eq_2(\tau_0,\tau_1,\tau_2)]_1, \ldots, [eq_7(\tau_0,\tau_1,\tau_2)]_1, \\ [eq_0(\tau_0,\tau_1)]_1, [eq_1(\tau_0,\tau_1)]_1, \ldots, [eq_3(\tau_0,\tau_1)]_1, \\ [eq_0(\tau_0)]_1, [eq_1(\tau_0)]_1, \\ [1]_1, [\xi]_1 \\ [1]_2,\ [\tau_0]_2,\ [\tau_1]_2,\ [\tau_2]_2,\ \ldots,\ [\tau_{n-1}]_2,\ [\xi]_2 \end{pmatrix} \tag{22}$$

接下来，假如我们得到了 $f(X_0, X_1, X_2)$ 的三个商多项式 $q_0, q_1(X_0), q_2(X_0, X_1)$。那么我们通过它们的 Evaluation Form 来计算所对应的 Commitment：

$$
\begin{aligned}
\mathsf{cm}(q_0) &= q_0 \cdot [1]_1 + \eta_0 \cdot [\xi]_1 \\
\mathsf{cm}(q_1) &= q_{1,0} \cdot [eq_0(\tau_0)]_1 + q_{1,1} \cdot [eq_1(\tau_0)]_1 + \eta_1 \cdot [\xi]_1 \\
\mathsf{cm}(q_2) &= q_{2,0} \cdot [eq_0(\tau_0, \tau_1)]_1 + q_{2,1} \cdot [eq_1(\tau_0, \tau_1)]_1 + q_{2,2} \cdot [eq_2(\tau_0, \tau_1)]_1 + q_{2,3} \cdot [eq_3(\tau_0, \tau_1)]_1 + \eta_2 \cdot [\xi]_1
\end{aligned} \tag{23}
$$

接下来，我们考虑第二个困难问题，如何计算 MLE 多项式的 n 次除法，得到 n 个商多项式 $q_0, q_1, \ldots, q_{n-1}$ 的 Evaluation Form。我们熟知的多项式长除法只适用于多项式的 Coefficient Form，而 Libra 论文 [XZZPS19] 则给出了一个 $O(n)$ 的算法支持 MLE 多项式的 Evaluation Form 的除法。

我们先考虑一个简单的情况，假设一个二元 MLE 多项式 $f(X_0, X_1)$ 的 Evaluation Form：

$$
f(X_0, X_1) = a_0 \cdot (1 - X_0)(1 - X_1) + a_1 \cdot X_0(1 - X_1) + a_2 \cdot (1 - X_0)X_1 + a_3 \cdot X_0X_1 \tag{24}
$$

第一步，我们计算 $f(X_0, X_1)/(X_1 - u_1)$，

我们把 $f(X_0, X_1)$ 的 Evaluation Form 按照 X_1 的次数展开，得到：

$$
\begin{aligned}
f(X_0, X_1) &= (a_0(1 - X_0) + a_1X_0)(1 - X_1) + (a_2(1 - X_0) + a_3X_0)X_1 \\
&= (a_0(1 - X_0) + a_1X_0) + (a_2(1 - X_0) + a_3X_0 - (a_0(1 - X_0) + a_1X_0)) \cdot X_1
\end{aligned} \tag{25}
$$

那么商多项式 $q_1(X_0)$ 就等于：

$$
q_1(X_0) = (a_2(1 - X_0) + a_3X_0) - (a_0(1 - X_0) + a_1X_0) \tag{26}
$$

我们重新整理下 $q_1(X_0)$ 的项，可以得到：

$$
q_1(X_0) = (a_2 - a_0)(1 - X_0) + (a_3 - a_1)X_0 \tag{27}
$$

上面的等式恰好是 $q_1(X_0)$ 的 Evaluation Form，要计算它并不难，只要将 $f(X_0, X_1)$ 的 Evaluation 向量的后半段减去前半段，即可得到 $q_1(X_0)$ 的 Evaluation 向量。

接下来看看 $f(X_0, X_1)/(X_1 - u_1)$ 除完之后的余数多项式，这是一个关于 X_0 的一元多项式，记为 $f'(X_0)$：

$$
\begin{aligned}
f'(X_0) &= f(X_0, X_1) - q_1(X_0)(X_1 - u_1) \\
&= (a_0(1 - X_0) + a_1X_0) + u_1 \cdot \Big((a_2(1 - X_0) + a_3X_0) - (a_0(1 - X_0) + a_1X_0)\Big) \\
&= (1 - u_1) \cdot (a_0(1 - X_0) + a_1X_0) + u_1 \cdot (a_2(1 - X_0) + a_3X_0) \\
&= ((1 - u_1) \cdot a_0 + u_1 \cdot a_2) \cdot (1 - X_0) + ((1 - u_1) \cdot a_1 + u_1 \cdot a_3) \cdot X_0
\end{aligned} \tag{28}
$$

经过整理之后，我们看到余数多项式 $f'(X_0)$ 的 Evaluation 向量，恰好是 $f(X_0, X_1)$ 的 Evaluation 向量前半段和后半段的合并，即：

$$
\mathsf{evaluations}(f) = (a_0, a_1, a_2, a_3) \tag{29}
$$

那么

$$
\mathsf{evaluations}(f') = (\mathsf{fold}(u_1, a_0, a_2),\ \mathsf{fold}(u_1, a_1, a_3)) \tag{30}
$$

这里折叠函数 fold 的定义是：

$$
\mathsf{fold}(u, a, b) = (1 - u) \cdot a + u \cdot b \tag{31}
$$

因此，我们可以得到一个清晰的算法，每次将 Evaluation 向量拆成两半，用高一半减去低一半，得到商多项式的 Evaluation 向量；将高低两半进行折叠，得到余数多项式的 Evaluation 向量。然后继续递归下去，就可以得到所有的商多项式。下面我们给出这个算法的 Python 实现：

```python
def decompose_by_div(evaluations, point) -> tuple[list, int]:
    e = evaluations.copy()
    k = log_2(len(e))
    quotients = []
    half = pow_2(k - 1)
    for i in range(k):
        q = [0] * half  # init quotient MLE (evalations)
        for j in range(half):
            q[j] = e[j + half] - e[j]  # compute quotient MLE
            e[j] = e[j] * (1 - point[k-i-1]) + e[j + half] * point[k-i-1]  # fold by point[k-i-1]
        quotients.insert(0, q)
        half >>= 1
    return quotients, e[0] # e[0] = f(point)
```

References

- [PST13] Papamanthou, Charalampos, Elaine Shi, and Roberto Tamassia. "Signatures of correct computation." Theory of Cryptography Conference. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. https://eprint.iacr.org/2011/587
- [XZZPS19] Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. "Libra: Succinct Zero-Knowledge Proofs with Optimal Prover Computation." Cryptology ePrint Archive (2019). https://eprint.iacr.org/2019/317
- [Ruffini] Ruffini's rule. (https://en.wikipedia.org/wiki/Ruffini%27s_rule)