Notes on Libra-PCS

1. MLE Polynomials

Of course, an MLE polynomial can also be represented in "coefficient form", which is expressed as follows:

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

For the example of a three-dimensional MLE polynomial shown above, we can write it as:

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

where $(f_0, f_1, \ldots, f_7)$ is the coefficient vector of the MLE polynomial. Note that because MLE polynomials belong to multivariate polynomials, any representation needs to determine the ordering of terms in the polynomial in advance. In this article and subsequent discussions, we will base on the Lexicographic Order.

For the "point-value form" representation of MLE polynomials, we can define it as:

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

where eq is a set of Lagrange Polynomials for the n-dimensional Boolean HyperCube $\{0,1\}^n$:

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

There exists an $N\log(N)$ conversion algorithm between the "point-value form" and "coefficient form" of MLE polynomials, which we won't discuss in depth here.

2. Division of MLE Polynomials

For a univariate polynomial $f(X) \in \mathbb{F}_p[X]$, if the value of $f(X)$ at $X = u$ is v, then we have the following equation:

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

where $q(X)$ is the quotient polynomial of $f(X)$ divided by $(X-u)$, and v is the remainder.

We can derive this equation simply. When we substitute $X = u$ into the equation, we get $f(u) = v$. This shows that the problem of polynomial evaluation is equivalent to finding the remainder of polynomial division. Then, we can subtract this remainder from $f(X)$, and the resulting polynomial $g(X) = f(X) - v$ can obviously be divided by $(X-u)$, meaning there exists a quotient polynomial, denoted as $q(X)$.

For a multivariate polynomial $f(X_0, X_1, \ldots, X_{n-1})$, the paper [PST13] provides a similar division relation equation:

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

If $f(X_0, X_1, \ldots, X_{n-1})$ is an MLE polynomial, it can be simplified to the following equation:

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

This is because in the MLE polynomial $\tilde{f}(X_0, X_1, \ldots, X_{n-1})$, the highest degree of each unknown X_k is 1. For $f(X_0, X_1, \ldots, X_k)$, after dividing by the factor $(X_k - u_k)$, the remainder polynomial will no longer contain the unknown X_k. So when $f(X_0, X_1, \ldots, X_{n-1})$ is sequentially divided by factors from $(X_{n-1} - u_{n-1})$ to $(X_0 - u_0)$, the number of unknowns in the resulting quotient polynomials and remainder polynomials will decrease successively, until we finally get a constant quotient polynomial $\tilde{q}_0$. Of course, after n divisions, a constant remainder polynomial will appear, which is exactly the evaluation of the MLE polynomial at $(u_0, u_1, \ldots, u_{n-1})$. This is known as Ruffini's rule [Ruffini].

Let's assume this final evaluation is v, i.e.,

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

3. Construction of Libra-PCS

Similar to the construction of KZG10, Libra-PCS also requires a structured SRS. It should be noted that the Libra paper is based on the KOE security assumption, while the scheme introduced in this article is based on the AGM security assumption, so the scheme's SRS has a smaller size and can also prove the correctness and Extractibility properties of the scheme under the AGM assumption.

It is generated by a Trusted Setup:

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

With this SRS, we can calculate the commitment of an n-variable MLE polynomial $\tilde{f}(X_0, X_1, \ldots, X_{n-1})$. That is, use the SRS as a basis for linear combination with its coefficient vector of length $N = 2^n$ to obtain an element on $\mathbb{G}_1$.

$$\begin{aligned} \tilde{f}(X_0, X_1, \ldots, X_{n-1}) = f_0 + f_1 X_0 + f_2 X_1 + f_3 X_0 X_1 + f_4 X_2 + f_5 X_0 X_2 + f_6 X_1 X_2 \\ + f_7 X_0 X_1 X_2 + \cdots + f_{N-1} X_0 X_1 \cdots X_{n-1} \end{aligned} \tag{10}$$

where $\vec{f} = (f_0, f_1, f_2, \ldots, f_{N-1})$ is the coefficient vector of $\tilde{f}(X_0, X_1, \ldots, X_{n-1})$. The commitment is calculated as follows:

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

Here ρ is a random number used to hide the information of $\vec{f}$.

Then if the Prover wants to prove $\tilde{f}(u_0, u_1, \ldots, u_{n-1}) = v$, they need to commit to the quotient polynomials $\tilde{q}_0, \tilde{q}_1, \ldots, \tilde{q}_{n-1}$.

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

To ensure that the Verifier can verify these Commitments, and to allow each Commitment to have a Blinding Factor, we need to modify the division equation of the MLE polynomial after adding these Blinding Factors:

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

Therefore, the Prover needs to calculate an additional Commitment, collecting all the Blinding Factors, which is the red part on the right side of the equation above:

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

Here R is clearly also an element on $\mathbb{G}_1$.

So after the Prover sends $(Q_0, Q_1, \ldots, Q_{n-1}, R)$ to the Verifier, the Verifier can verify through the following Pairing equation:

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

4. Supporting MLE Evaluation Form

In the Libra-PCS described above, when the Prover calculates the Commitment, they need to first obtain the "coefficient form" of the MLE polynomial before calculating the Commitment. However, in many Sumcheck protocols, the Evaluation Form of MLE is used, that is, the point-value form. In other words, the n-variable MLE polynomial is represented by its evaluation values on the n-dimensional Boolean Hypercube:

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

If we need to convert the Evaluation form of the MLE polynomial to the Coefficient form, this conversion algorithm requires $O(n \cdot 2^n)$ time complexity. So is it possible to directly support the Evaluation form of MLE?

The Evaluation form of MLE brings two difficult problems. The first is how to calculate a commitment for an Evaluation Form based on SRS, and the second is how to calculate n divisions of the MLE polynomial to obtain n quotient polynomials $q_0, q_1, \ldots, q_{n-1}$.

Let's look at the first problem: how to calculate the Commitment for the Evaluation Form. There are two ways to do this. A more direct way is to produce the Commitment of the Multilinear Basis of the MLE polynomial directly when calculating the SRS, rather than the Commitment of the Monomial Basis.

For example, suppose n=3, then we produce a new set of SRS parameters to calculate the Commitment of the Evaluation Form of a three-variable MLE polynomial.

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

To simplify the notation, we use $eq_i(\vec{X})$ to represent $eq(i_0, i_1, \ldots, i_{n-1}, X_0, X_1, \ldots, X_{n-1})$. Suppose we have a polynomial $f(X_0, X_1, X_2)$:

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

So we calculate its Commitment as follows:

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

However, we need to realize that $SRS^{(3)}$ is not enough, because the Multilinear Basis like $eq_i(\vec{X})$ is related to the size of the Domain. We should realize that if we want to commit to the Evaluation Form of a bivariate polynomial, we cannot use $SRS^{(3)}$ to calculate. Instead, we need to produce another set of SRS parameters for bivariate MLE polynomials, denoted as $SRS^{(2)}$

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

Similarly, we also need a set of SRS parameters for univariate polynomials, denoted as $SRS^{(1)}$:

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

Finally, for constant polynomials, we only need $[1]_1$ as a Basis to calculate the commitment. We combine all these $SRS^{(k)}$ based on k-MLE together, denoted as SRS^*:

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

Next, suppose we have obtained three quotient polynomials $q_0, q_1(X_0), q_2(X_0, X_1)$ of $f(X_0, X_1, X_2)$. Then we calculate the corresponding Commitments through their Evaluation Form:

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

Next, let's consider the second difficult problem: how to calculate n divisions of the MLE polynomial to obtain n quotient polynomials $q_0, q_1, \ldots, q_{n-1}$ in Evaluation Form. The familiar polynomial long division is only applicable to the Coefficient Form of polynomials, while the Libra paper [XZZPS19] provides an $O(n)$ algorithm that supports division of MLE polynomials in Evaluation Form.

Let's first consider a simple case, assuming a bivariate MLE polynomial $f(X_0, X_1)$ in Evaluation Form:

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

The first step is to calculate $f(X_0, X_1)/(X_1 - u_1)$,

We expand the Evaluation Form of $f(X_0, X_1)$ according to the degree of X_1:

$$\begin{aligned} f(X_0, X_1) &= (a_0(1 - X_0) + a_1 X_0)(1 - X_1) + (a_2(1 - X_0) + a_3 X_0)X_1 \\ &= (a_0(1 - X_0) + a_1 X_0) + (a_2(1 - X_0) + a_3 X_0 - (a_0(1 - X_0) + a_1 X_0)) \cdot X_1 \end{aligned} \tag{25}$$

Then the quotient polynomial $q_1(X_0)$ is equal to:

$$q_1(X_0) = (a_2(1 - X_0) + a_3 X_0) - (a_0(1 - X_0) + a_1 X_0) \tag{26}$$

We rearrange the terms of $q_1(X_0)$ and can get:

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

The above equation is exactly the Evaluation Form of $q_1(X_0)$, which is not difficult to calculate. We just need to subtract the first half of the Evaluation vector of $f(X_0, X_1)$ from the second half to obtain the Evaluation vector of $q_1(X_0)$.

Next, let's look at the remainder polynomial after dividing $f(X_0, X_1)/(X_1 - u_1)$, which is a univariate polynomial in terms of X_0, denoted as $f'(X_0)$:

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

After reorganization, we see that the Evaluation vector of the remainder polynomial $f'(X_0)$ is exactly the combination of the first half and the second half of the Evaluation vector of $f(X_0, X_1)$, that is:

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

Then

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

Here the definition of the fold function fold is:

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

Therefore, we can obtain a clear algorithm. Each time, split the Evaluation vector into two halves, subtract the lower half from the higher half to get the Evaluation vector of the quotient polynomial; fold the high and low halves to get the Evaluation vector of the remainder polynomial. Then continue recursively to get all the quotient polynomials. Below we give the Python implementation of this algorithm:

```
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)