Notes on Basefold (Part III): MLE Evaluation Argument

- Yu Guo yu.guo@secbit.io
- Jade Xie jade@secbit.io

Assume we have an MLE polynomial $\tilde{f}(\vec{X}) \in \mathbb{F}[\vec{X}]^{\leq 1}$, an evaluation point $\mathbf{u} \in \mathbb{F}^d$, and the result of the polynomial's operation at the evaluation point $v = \tilde{f}(\mathbf{u})$. We aim to construct a Polynomial Evaluation Argument based on the Basefold-IOPP protocol.

Based on FRI, we can utilize the DEEP Method to construct a PCS protocol, which verifies the existence of a Low Degree Polynomial $q(X) = (f(X) - f(u))/(X - u)$. However, the FRI protocol can only handle the case of Univariate Polynomials. For MLE Polynomials, we generally need to use the following extended polynomial remainder theorem to reduce it to a quotient polynomial existence problem:

$$\tilde{f}(X_0, X_1, \ldots, X_{n-1}) - v = \sum_{i=0}^{n-1} (X_k - u_k) \cdot q_i(X_0, X_1, \ldots, X_{k-1}) \tag{1}$$

For example, Virgo adopts this scheme for the MLE PCS protocol. However, Basefold [ZCF23] presents a refreshing approach to implementing the MLE Evaluation Argument by combining the Basefold-IOPP protocol and the Sumcheck protocol. We can use the Sumcheck protocol's summation and proof as the main framework of the protocol, then use Basefold-IOPP to ensure the Low degree of the MLE polynomial. Additionally, the evaluation of the MLE polynomial at random points generated after multiple rounds of folding in the Basefold-IOPP protocol compensates for the correctness proof of the evaluation required in the last round of the Sumcheck protocol.

Final Output of the Basefold-IOPP Protocol

For demonstration purposes, we use an MLE polynomial $\tilde{f}(X_0, X_1, X_2)$ with $d = 3$, defined as follows:

$$\tilde{f}(\vec{X}) = f_0 + f_1 \cdot X_0 + f_2 \cdot X_1 + f_3 \cdot X_0X_1 + f_4 \cdot X_2 + f_5 \cdot X_0X_2 + f_6 \cdot X_1X_2 + f_7 \cdot X_0X_1X_2 \tag{2}$$

Here, $\mathbf{f}$ is the coefficient vector of $\tilde{f}$, defined above as the Coefficients Form of the MLE polynomial.

Let's revisit the folding process of the Basefold-IOPP protocol. First, let $f^{(3)}(X_0, X_1, X_2)$ correspond to the codeword after encoding $\tilde{f}$ with C_3. In each folding round, the Verifier sends a random challenge α_i to the Prover, who then generates $f^{(i)}$ by folding $f^{(i+1)}$, and sends the encoded codeword (as an Oracle) to the Verifier.

After one folding round of the Basefold-IOPP protocol, the polynomial corresponding to the codeword obtained by both parties, $f^{(2)}(X_0, X_1)$, is:

$$\begin{aligned} f^{(2)}(X_0, X_1) &= f^{(3)}(X_0, X_1, \alpha_2) \\ &= (f_0 + \alpha_2 \cdot f_4) + (f_1 + \alpha_2 \cdot f_5) \cdot X_0 + (f_2 + \alpha_2 \cdot f_6) \cdot X_1 + (f_3 + \alpha_2 \cdot f_7) \cdot X_0X_1 \end{aligned} \tag{3}$$

After another folding round, the polynomial corresponding to the codeword, $f^{(1)}(X_0)$, is:

$$\begin{aligned} f^{(1)}(X_0) &= f^{(2)}(X_0, \alpha_1) \\ &= (f_0 + \alpha_2 \cdot f_4 + \alpha_1 \cdot f_2 + \alpha_2\alpha_1 \cdot f_6) + (f_1 + \alpha_2 \cdot f_5 + \alpha_1 \cdot f_3 + \alpha_2\alpha_1 \cdot f_7) \cdot X_0 \end{aligned} \tag{4}$$

Finally, the polynomial corresponding to the codeword, the **constant polynomial** $f^{(0)}$, is as follows:

$$\begin{aligned} f^{(0)} &= f^{(1)}(\alpha_0) \\ &= (f_0 + \alpha_2 \cdot f_4 + \alpha_1 \cdot f_2 + \alpha_2\alpha_1 \cdot f_6) + (f_1 + \alpha_2 \cdot f_5 + \alpha_1 \cdot f_3 + \alpha_2\alpha_1 \cdot f_7) \cdot \alpha_0 \\ &= f_0 + f_1\alpha_0 + f_2\alpha_1 + f_3\alpha_0\alpha_1 + f_4\alpha_2 + f_5\alpha_0\alpha_2 + f_6\alpha_1\alpha_2 + f_7\alpha_0\alpha_1\alpha_2 \end{aligned} \tag{5}$$

Upon closer examination, $f^{(0)}$ is exactly the evaluation of $\tilde{f}(X_0, X_1, X_2)$ at $(\alpha_0, \alpha_1, \alpha_2)$.

Thus, in the final round of the Basefold-IOPP protocol, the Prover sends the folded constant polynomial to the Verifier. The Verifier then uses the Query-phase to verify the correctness of each folding step. Meanwhile, the Verifier also obtains the evaluation result $v = \tilde{f}(\vec{X})$ at the random point $\vec{X} = (\alpha_0, \alpha_1, \alpha_2)$. From another perspective, the Basefold-IOPP protocol not only completes the Proximity proof but also provides an additional evaluation of $\tilde{f}(X_0, X_1, X_2)$ at a random point. More precisely, this is a proof of a vector inner product:

$$\langle \mathbf{f}, (1, \alpha_0) \otimes (1, \alpha_1) \otimes (1, \alpha_2) \rangle = v \tag{6}$$

However, this MLE evaluation point is not a pre-negotiated public input but is instead composed of random challenges generated during the execution of the Basefold-IOPP protocol.

In summary, in the final round of the IOPP protocol, the Prover folds to produce a constant polynomial, denoted as $f^{(0)}$, which is precisely $\tilde{f}$ evaluated at the random point $(\alpha_0, \alpha_1, \alpha_2)$. What we need is to prove the correctness of $\tilde{f}$ at a **public point**, such as (u_0, u_1, u_2).

$$\tilde{f}(u_0, u_1, u_2) = f_0 + f_1 \cdot u_0 + f_2 \cdot u_1 + f_3 \cdot u_0u_1 + f_4 \cdot u_2 + f_5 \cdot u_0u_2 + f_6 \cdot u_1u_2 + f_7 \cdot u_0u_1u_2 \tag{7}$$

The subsequent question becomes: How can we use the evaluation of $\tilde{f}$ at a **random point** to prove the correctness of its evaluation at another **public point**?

Leveraging Sumcheck

We can revisit $\tilde{f}(u_0, u_1, u_2)$ by utilizing the properties of MLE. According to the definition of MLE, we have the following equation:

$$\mathsf{EQ}_1: \qquad \tilde{f}(X_0, X_1, X_2) = \sum_{\mathbf{b}\in\{0,1\}^3} \tilde{f}(b_0, b_1, b_2) \cdot \tilde{eq}_{(b_0,b_1,b_2)}(X_0, X_1, X_2) \tag{8}$$

Here, $\tilde{eq}$ is the Lagrange Polynomial of MLE, defined as:

$$\tilde{eq}_{(b_0,b_1,\ldots,b_{n-1})}(X_0, X_1, \ldots, X_{n-1})\big) = \prod_{i=0}^{n-1} \big((1-b_i)(1-X_i) + b_i \cdot X_i\big) \tag{9}$$

It is easily verified that when $(X_0, X_1, X_2) = \mathbf{b}'$, $\mathbf{b}' \in \{0,1\}^3$, the left side of equation EQ_1 equals $\tilde{f}(\mathbf{b}')$. Observing each summand $\tilde{eq}_{\mathbf{b}}(\mathbf{b}')$, only when the vectors $\mathbf{b} = \mathbf{b}'$ are completely identical does $\tilde{eq}_{\mathbf{b}}(\mathbf{b}') = 1$; otherwise, they all equal 0. Therefore, the right side of the equation reduces to $\tilde{f}(\mathbf{b}') \cdot \tilde{eq}_{\mathbf{b}}(\mathbf{b}') = \tilde{f}(\mathbf{b}')$, ensuring the equality holds.

This means that both sides of equation EQ_1 agree on their values over the 3-dimensional Boolean HyperCube. Based on the uniqueness property of MLE, we can conclude that EQ_1 holds for any $\mathbf{X} \in F^3$.

Although the above is fundamental knowledge of MLE, note that equation EQ_1 introduces a new perspective: we can transform the evaluation problem of $\tilde{f}(u_0, u_1, u_2)$ into a Sumcheck problem:

$$\tilde{f}(u_0, u_1, u_2) = v = \sum_{\mathbf{b}\in\{0,1\}^3} \tilde{f}(\mathbf{b}) \cdot \tilde{eq}_{\mathbf{b}}(u_0, u_1, u_2) \tag{10}$$

It is immediately apparent that we can use the Sumcheck protocol to perform a "sum proof" on the above equation. An important function of the Sumcheck protocol is to transform a "summation problem" into two MLE polynomial evaluation problems (specifically for $\tilde{f}$ and $\tilde{eq}$) at a random point. However, in general, this approach is meaningless because, in the last round of Sumcheck, we need to prove the evaluation of $\tilde{f}$ at a new point, leading to a circular proof situation: we originally prove the evaluation of a polynomial at one point, then use the Sumcheck protocol to transform this proof into an evaluation at another point. This makes the Sumcheck protocol process redundant and useless. Typically, the protocol would rely on another MLE Evaluation Argument protocol, where the Prover sends the final evaluation and its correctness proof, thereby concluding the Sumcheck protocol. But here, we cannot rely on another MLE PCS protocol because our goal is to construct an MLE Evaluation Argument protocol, not to depend on an existing MLE PCS.

However, this problem is not difficult to solve because the previously introduced Basefold-IOPP protocol provides a byproduct: a proof of $\tilde{f}$'s evaluation at a **random point**. If the Sumcheck protocol and the Basefold-IOPP protocol share random challenges, the value provided by Basefold-IOPP in the final step can compensate for the correctness proof of the evaluation required in the last round of Sumcheck. Another remaining issue is how to handle the evaluation proof of $\tilde{eq}$. This is simpler because $\tilde{eq}$ is a public polynomial, allowing the Verifier to compute it independently without requiring the Prover to send an evaluation proof. Moreover, computing the evaluation of $\tilde{eq}$ only has a complexity of $O(\log N)$, which does not increase the Verifier's burden or affect the Verifier's succinctness. At this point, the Sumcheck protocol's role is no longer redundant but becomes crucial: **it transforms the problem of proving a polynomial's evaluation at a public point into a proof of its evaluation at a random point**.

Next, we need to synchronize the execution of the Basefold-IOPP protocol and the Sumcheck protocol, allowing both protocols to share a set of random challenges. This way, in the final step of both protocols, we can complete the collaboration, thereby ultimately proving the correctness of $\tilde{f}(u_0, u_1, u_2) = v$.

Following this approach, we attempt to outline the protocol flow when $s = 3$.

Public Inputs

1. Commitment of $\tilde{f}$, $\pi_3 = [\tilde{f}] = \mathsf{Enc}_3(\mathbf{f})$
2. Evaluation point $\mathbf{u} = (u_0, u_1, u_2)$
3. Operation value v

Witness

- Coefficient vector of MLE $\tilde{f}$, $\mathbf{f} = (f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7)$
- Evaluation vector of $\tilde{f}$, $\mathbf{a} = (a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7)$

First Round: Prover sends the univariate polynomial $h^{(2)}(X)$

$$h^{(2)}(X) = \sum_{b_0,b_1\in\{0,1\}^2} f(b_0, b_1, X) \cdot \tilde{eq}\big((b_0, b_1, X), (u_0, u_1, u_2)\big) \tag{11}$$

Since the right side is a degree 2 Univariate Polynomial, the Prover can compute the coefficients of $h^{(2)}(X)$:

$$
\begin{aligned}
h_0^{(2)} &= \sum_{i=0}^{3} a_i \cdot e_i \\
h_1^{(2)} &= \sum_{i=0}^{3} a_{i+4} \cdot e_i + a_i \cdot e_{i+4} \\
h_2^{(2)} &= \sum_{i=0}^{3} a_{i+4} \cdot e_{i+4}
\end{aligned} \tag{12}
$$

Here, $\mathbf{e}$ is the evaluation vector of $\tilde{eq}(\mathbf{X}, \mathbf{u})$. It is easy to verify that $h^{(2)}(X) = h_0^{(2)} + h_1^{(2)} \cdot X + h_2^{(2)} \cdot X^2$.

Second Round: Verifier sends challenge $\alpha_2 \leftarrow \mathbb{F}_p$

Third Round: Prover and Verifier simultaneously execute the Basefold-IOPP protocol and the Sumcheck protocol:

- Prover computes $\tilde{f}^{(2)}(X_1, X_2) = f(\alpha_0, X_1, X_2)$

$$
\tilde{f}^{(2)}(X_0, X_1) = (f_0 + f_4\alpha_2) + (f_1 + f_5\alpha_2) \cdot X_0 + (f_2 + f_6\alpha_2) \cdot X_1 + (f_3 + f_7\alpha_2) \cdot X_0X_1 \tag{13}
$$

- Prover sends the folded vector encoding: $\pi_2 = \mathrm{Enc}_2[\tilde{f}^{(2)}]$
- Prover computes $h^{(2)}(\alpha_2)$ as the summation value for the next round of the Sumcheck protocol
- Prover computes and sends $h^{(1)}(X)$

$$
h^{(1)}(X) = \sum_{b_0 \in \{0,1\}} f(b_0, X, \alpha_2) \cdot \tilde{eq}((b_0, X, \alpha_2), (u_0, u_1, u_2)) \tag{14}
$$

Since the right side is also a degree 2 Univariate Polynomial in X, the Prover can calculate the coefficients of $h^{(1)}(X)$: $(h_0^{(1)}, h_1^{(1)}, h_2^{(1)})$.

Fourth Round: Verifier sends challenge $\alpha_1 \leftarrow \mathbb{F}_p$

Fifth Round: Prover continues executing the Basefold-IOPP protocol and the Sumcheck protocol:

- Prover computes $\tilde{f}^{(1)}(X_2)$

$$
\tilde{f}^{(1)}(X_2) = f(X_0, \alpha_1, \alpha_2) = (f_0 + f_4\alpha_2 + f_2\alpha_1 + f_6\alpha_1\alpha_2) + (f_1 + f_5\alpha_2 + f_3\alpha_1 + f_7\alpha_1\alpha_2) \cdot X_0 \tag{15}
$$

- Prover sends the folded vector encoding $\pi_1 = \mathrm{Enc}_1[\tilde{f}^{(1)}]$
- Prover computes the summation value for the next round of the Sumcheck protocol: $h^{(1)}(\alpha_1)$
- Prover computes and sends $h^{(0)}(X)$

$$
h^{(0)}(X) = \sum_{b_0 \in \{0,1\}} f(X, \alpha_1, \alpha_2) \cdot \tilde{eq}((b_0, \alpha_1, \alpha_2), (u_0, u_1, u_2)) \tag{16}
$$

Assume the coefficients of $h^{(0)}(X)$ are represented as $(h_0^{(0)}, h_1^{(0)}, h_2^{(0)})$.

Sixth Round: Verifier sends challenge $\alpha_0 \leftarrow \mathbb{F}_p$

Seventh Round: Prover continues executing the Basefold-IOPP protocol:

- Prover computes $\tilde{f}^{(0)}$, a **constant polynomial**

$$
\tilde{f}^{(0)} = f(\alpha_0, \alpha_1, \alpha_2) = f_0 + f_1\alpha_0 + f_2\alpha_1 + f_3\alpha_0\alpha_1 + f_4\alpha_2 + f_5\alpha_0\alpha_2 + f_6\alpha_1\alpha_2 + f_7\alpha_0\alpha_1\alpha_2 \tag{17}
$$

- Prover sends the folded vector encoding $\pi_0 = \mathsf{Enc}_0[\tilde{f}^{(0)}]$

Eighth Round: Verifier validates the following equations:

1. Verifier verifies several rounds of Basefold Queries, $Q = \{q_i\}$
2. Verifier checks the correctness of each folding step in Sumcheck:

$$
\begin{aligned}
h^{(2)}(0) + h^{(2)}(1) &\overset{?}{=} v \\
h^{(1)}(0) + h^{(1)}(1) &\overset{?}{=} h^{(2)}(\alpha_2) \\
h^{(0)}(0) + h^{(0)}(1) &\overset{?}{=} h^{(1)}(\alpha_1)
\end{aligned} \tag{18}
$$

3. Verifier checks whether the final encoding π_0 is correct

$$
\pi_0 \overset{?}{=} \mathsf{Enc}_0\left(\frac{h^{(0)}(\alpha_0)}{\tilde{eq}((\alpha_0, \alpha_1, \alpha_2), (u_0, u_1, u_2))}\right) \tag{19}
$$

An Alternative Folding Method

In the Basefold paper, the PCS protocol requires that the MLE polynomial $\tilde{f}(X_0, X_1, \ldots, X_{d-1})$ be converted to the Coefficients Form beforehand, and then follow the protocol described in the previous section for the proof. However, in many Sumcheck-based SNARK systems, the Sumcheck protocol finally produces the Evaluation Form of the MLE polynomial, that is, the evaluation of the MLE polynomial on a Boolean HyperCube. Therefore, if we directly connect the SNARK protocol to Basefold-PCS, we would still need to convert the MLE polynomial from the Evaluation Form to the Coefficients Form. This conversion algorithm has a complexity of $O(N\log(N))$, where $N = 2^d$, and d is the number of variables in the MLE polynomial. Below is the definition of the Evaluation Form of $\tilde{f}(X_0, X_1, \ldots, X_{d-1})$:

$$\tilde{f}(X_0, X_1, \ldots, X_{d-1}) = \sum_{\mathbf{b}\in\{0,1\}^d} a_{\mathbf{b}} \cdot eq_{\mathbf{b}}(X_0, X_1, \ldots, X_{d-1}) \tag{20}$$

As noted in the FRI-Binius paper, we do not need to convert the MLE polynomial from the Evaluation Form to the Coefficients Form. Instead, we can directly construct the PCS protocol in the Evaluation Form. In the previous section, the reason we needed the Coefficients Form of $\tilde{f}(X_0, X_1, \ldots, X_{d-1})$ was that the Basefold-IOPP protocol's Commit method required the coefficients of $\tilde{f}(X_0, X_1, \ldots, X_{d-1})$. If we follow the folding method of the Basefold-IOPP protocol described below, we can directly construct the PCS protocol in the Evaluation Form.

Now, let's consider the folding method of the Evaluation Form of $\tilde{f}(X_0, X_1, X_2)$. First, expand the definition of the Evaluation Form of $\tilde{f}(X_0, X_1, X_2)$:

$$\begin{aligned}\tilde{f}^{(3)}(X_0, X_1, X_2) &= \tilde{f}(X_0, X_1, X_2)\\ &= a_0 \cdot eq_{(0,0,0)}(X_0, X_1, X_2) + a_1 \cdot eq_{(1,0,0)}(X_0, X_1, X_2) + a_2 \cdot eq_{(0,1,0)}(X_0, X_1, X_2) + a_3 \cdot eq_{(1,1,0)}(X_0, X_1, X_2)\\ &+ a_4 \cdot eq_{(0,0,1)}(X_0, X_1, X_2) + a_5 \cdot eq_{(1,0,1)}(X_0, X_1, X_2) + a_6 \cdot eq_{(0,1,1)}(X_0, X_1, X_2) + a_7 \cdot eq_{(1,1,1)}(X_0, X_1, X_2)\end{aligned} \tag{21}$$

We can fold the above Evaluation Form, ultimately obtaining the evaluation $\tilde{f}^{(3)}(\alpha_0, \alpha_1, \alpha_2)$. For example, we can fold the Evaluation Form of $\tilde{f}(X_0, X_1, X_2)$ with respect to α_2 to obtain:

$$\begin{aligned}\tilde{f}^{(2)}(X_0, X_1) &= \tilde{f}^{(3)}(X_0, X_1, \alpha_2)\\ &= a_0 \cdot eq_{(0,0,0)}(X_0, X_1, \alpha_2) + a_1 \cdot eq_{(1,0,0)}(X_0, X_1, \alpha_2) + a_2 \cdot eq_{(0,1,0)}(X_0, X_1, \alpha_2) + a_3 \cdot eq_{(1,1,0)}(X_0, X_1, \alpha_2)\\ &+ a_4 \cdot eq_{(0,0,1)}(X_0, X_1, \alpha_2) + a_5 \cdot eq_{(1,0,1)}(X_0, X_1, \alpha_2) + a_6 \cdot eq_{(0,1,1)}(X_0, X_1, \alpha_2) + a_7 \cdot eq_{(1,1,1)}(X_0, X_1, \alpha_2)\end{aligned} \tag{22}$$

Next, we need to decompose $eq_{\mathbf{b}}(X_0, X_1\alpha_2)$:

$$eq_{(b_0,b_1,b_2)}(X_0, X_1, \alpha_2) = eq_{(b_0,b_1)}(X_0, X_1) \cdot \Big((1-b_2)\cdot(1-\alpha_2) + b_2 \cdot \alpha_2\Big) \tag{23}$$

When $b_2 = 0$, the right side simplifies to $(1-\alpha_2)\cdot eq_{(b_0,b_1)}(X_0, X_1)$; when $b_2 = 1$, it simplifies to $\alpha_2 \cdot eq_{(b_0,b_1)}(X_0, X_1)$. Then, continue simplifying $\tilde{f}^{(2)}(X_0, X_1)$:

$$\begin{aligned}\tilde{f}^{(2)}(X_0, X_1) &= ((1-\alpha_2)\cdot a_0 + \alpha_2 \cdot a_4)\cdot eq_{(0,0)}(X_0, X_1) + ((1-\alpha_2)\cdot a_1 + \alpha_2 \cdot a_5)\cdot eq_{(0,1)}(X_0, X_1)\\ &+ ((1-\alpha_2)\cdot a_2 + \alpha_2 \cdot a_6)\cdot eq_{(1,0)}(X_0, X_1) + ((1-\alpha_2)\cdot a_3 + \alpha_2 \cdot a_7)\cdot eq_{(1,1)}(X_0, X_1)\end{aligned} \tag{24}$$

This is equivalent to folding (linear combination) the vectors (a_0, a_1, a_2, a_3) and (a_4, a_5, a_6, a_7) with $(1-\alpha_2, \alpha_2)$. Compared with the folding method in the Basefold-IOPP protocol, it uses $(1, \alpha_2)$ to fold (linear combination) the vectors (f_0, f_1, f_2, f_3) and (f_4, f_5, f_6, f_7). If we continue to fold in the new way, we can eventually get the same folding result as the Basefold-IOPP protocol.

$$\begin{aligned}\tilde{f}^{(1)}(X_0) &= \tilde{f}^{(2)}(X_0, \alpha_1)\\ &= ((1-\alpha_2)\cdot a_0 + \alpha_2 \cdot a_4)\cdot eq_{(0,0)}(X_0, \alpha_1) + ((1-\alpha_2)\cdot a_1 + \alpha_2 \cdot a_5)\cdot eq_{(1,0)}(X_0, \alpha_1)\\ &+ ((1-\alpha_2)\cdot a_2 + \alpha_2 \cdot a_6)\cdot eq_{(0,1)}(X_0, \alpha_1) + ((1-\alpha_2)\cdot a_3 + \alpha_2 \cdot a_7)\cdot eq_{(1,1)}(X_0, \alpha_1)\\ &= ((1-\alpha_2)\cdot a_0 + \alpha_2 \cdot a_4)\cdot(1-\alpha_1)\cdot eq_{(0)}(X_0) + ((1-\alpha_2)\cdot a_1 + \alpha_2 \cdot a_5)\cdot(1-\alpha_1)\cdot eq_{(1)}(X_0)\\ &+ ((1-\alpha_2)\cdot a_2 + \alpha_2 \cdot a_6)\cdot\alpha_1\cdot eq_{(0)}(X_0) + ((1-\alpha_2)\cdot a_3 + \alpha_2 \cdot a_7)\cdot\alpha_1\cdot eq_{(1)}(X_0)\\ &= \Big((1-\alpha_1)\cdot((1-\alpha_2)\cdot a_0 + \alpha_2 \cdot a_4) + \alpha_1\cdot((1-\alpha_2)\cdot a_2 + \alpha_2 \cdot a_6)\Big)\cdot eq_{(0)}(X_0)\\ &+ \Big((1-\alpha_1)\cdot((1-\alpha_2)\cdot a_1 + \alpha_2 \cdot a_5) + \alpha_1\cdot((1-\alpha_2)\cdot a_3 + \alpha_2 \cdot a_7)\Big)\cdot eq_{(1)}(X_0)\end{aligned} \tag{25}$$

Continuing the folding process, we eventually obtain:

$$\begin{aligned}\tilde{f}^{(0)} &= \tilde{f}^{(1)}(\alpha_0)\\ &= \Big((1-\alpha_0)\cdot((1-\alpha_1)\cdot((1-\alpha_2)\cdot a_0 + \alpha_2 \cdot a_4) + \alpha_1\cdot((1-\alpha_2)\cdot a_2 + \alpha_2 \cdot a_6))\\ &\quad +\alpha_0\cdot((1-\alpha_1)\cdot((1-\alpha_2)\cdot a_1 + \alpha_2 \cdot a_5) + \alpha_1\cdot((1-\alpha_2)\cdot a_3 + \alpha_2 \cdot a_7))\Big)\\ &= \tilde{f}^{(3)}(\alpha_0, \alpha_1, \alpha_2)\end{aligned} \tag{26}$$

Can we improve the folding method of the Basefold-IOPP protocol so that it can directly fold the Evaluation Form of the MLE polynomial?

Basefold-IOPP Protocol Based on the Evaluation Form

Next, we attempt to outline the Basefold-IOPP protocol for the Evaluation Form of $\tilde{f}(X_0, X_1, X_2)$ when $d = 3$. First, we rewrite the folding function from the previous article:

$$\mathsf{fold}^*_\alpha(y_0, y_1) = (1-\alpha) \cdot \frac{t_j \cdot y_1 - t'_j \cdot y_0}{t_j - t'_j} + \alpha \cdot \frac{y_0 - y_1}{t_j - t'_j} \tag{27}$$

From earlier discussions, we know that the folding function has the following homomorphic property: after folding, the codeword is equivalent to the original message being folded and then encoded, expressed as:

$$\mathsf{fold}_\alpha(\mathsf{Enc}_i(\mathbf{m})) = \mathsf{Enc}_i(\mathsf{fold}_\alpha(\mathbf{m})) \tag{28}$$

Similarly, we can prove that the new folding function satisfies the above property. We can attempt to fold an element in the codeword π_i using the new folding function:

$$\begin{aligned}\mathsf{fold}^*_\alpha(\pi_i[j], \pi_i[n_{i-1}+j]) &= \frac{1-\alpha}{t_j - t'_j} \cdot \Big(t_j \cdot (\mathbf{m}_l G_{i-1}[j] + t'_j \cdot \mathbf{m}_r G_{i-1}[j]) - t'_j \cdot (\mathbf{m}_l G_{i-1}[j] + t_j \cdot \mathbf{m}_r G_{i-1}[j])\Big) \\ &+ \frac{\alpha}{t_j - t'_j} \cdot \Big(\mathbf{m}_l G_{i-1}[j] + t_j \cdot \mathbf{m}_r G_{i-1}[j] - \mathbf{m}_l G_{i-1}[j] - t'_j \cdot \mathbf{m}_r G_{i-1}[j]\Big) \\ &= (1-\alpha) \cdot \mathbf{m}_l G_{i-1}[j] + \alpha \cdot \mathbf{m}_r G_{i-1}[j]\end{aligned} \tag{29}$$

The above derivation demonstrates that the new folding function also satisfies the homomorphic property concerning the encoding function:

$$\mathsf{fold}^*_\alpha(\mathsf{Enc}_i(\mathbf{m})) = \mathsf{Enc}_i(\mathsf{fold}^*_\alpha(\mathbf{m})) \tag{30}$$

Additionally, we can prove that the new folding process does not affect the reliability of the Basefold-IOPP protocol, i.e., the Minimum Hamming Weight of the folded codeword remains above a lower bound.

Thus, we have an important conclusion: whether we use fold_α or fold^*_α, both can be used to construct a Proximity-Proof protocol without significant differences.

Next, we outline the Commit-phase protocol flow for better understanding:

Public Inputs

- Codeword of the MLE polynomial $\tilde{f}$, $\pi_3 = \mathsf{Enc}_3(\mathbf{a}) = \mathbf{a}G_3$

Witness

- Evaluation vector of the MLE polynomial $\tilde{f}$, $\mathbf{a} = (a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7)$

First Round: Verifier sends a random number α_2

Second Round: Prover computes $\pi_2 = \mathsf{fold}^*_\alpha(\pi_3)$ and sends it to the Verifier

The process of computing π_2 is as follows:

$$\pi_2[j] = \mathsf{fold}^*_\alpha(\pi_3[j], \pi_3[j+4]), \quad j \in \{0,1,2,3\} \tag{31}$$

The computed $\pi_2 = \mathsf{Enc}_2(f^{(2)})$, meaning that π_2 is the codeword of $f^{(2)}$, where $f^{(2)}$ is the folded result of f with respect to α_2:

$$\begin{aligned}f^{(2)}(X_0, X_1) &= ((1-\alpha_2)a_0 + \alpha_2 a_4) \cdot eq_{(0,0)}(X_0, X_1) + ((1-\alpha_2)a_1 + \alpha_2 a_5) \cdot eq_{(1,0)}(X_0, X_1) \\ &+ ((1-\alpha_2)a_2 + \alpha_2 a_6) \cdot eq_{(0,1)}(X_0, X_1) + ((1-\alpha_2)a_3 + \alpha_2 a_7) \cdot eq_{(1,1)}(X_0, X_1) \\ &= f(X_0, X_1, \alpha_2)\end{aligned} \tag{32}$$

Third Round: Verifier sends a random number α_1

Fourth Round: Prover computes $\pi_1 = \mathsf{fold}^*_\alpha(\pi_2)$ and sends it to the Verifier

The process of computing π_1 is as follows:

$$\pi_1[j] = \mathsf{fold}^*_\alpha(\pi_2[j], \pi_2[j+2]), \quad j \in \{0,1\} \tag{33}$$

The computed $\pi_1 = \mathsf{Enc}_1(f^{(1)})$, where $f^{(1)}$ is the folded result of $f^{(2)}$ with respect to α_1:

$$\begin{aligned}f^{(1)}(X_0) &= \Big((1-\alpha_1)((1-\alpha_2)a_0 + \alpha_2 a_4) + \alpha_1((1-\alpha_2)a_1 + \alpha_2 a_5)\Big) \cdot eq_{(0)}(X_0) \\ &+ \Big((1-\alpha_1)((1-\alpha_2)a_2 + \alpha_2 a_6) + \alpha_1((1-\alpha_2)a_3 + \alpha_2 a_7)\Big) \cdot eq_{(1)}(X_0) \\ &= f(X_0, \alpha_1, \alpha_2)\end{aligned} \tag{34}$$

Fifth Round: Verifier sends a random number α_0

Sixth Round: Prover computes $\pi_0 = \mathsf{fold}_\alpha(\pi_1)$ and sends it to the Verifier

The process of computing π_0 is as follows:

$$\pi_0[j] = \mathsf{fold}^*_\alpha(\pi_1[j], \pi_1[j+1]), \quad j = 0 \tag{35}$$

Similarly, $\pi_0 = \mathsf{Enc}_0(f^{(0)})$, meaning π_0 is the codeword of $f^{(0)}$, where $f^{(0)}$ is the folded result of f with respect to $(\alpha_0, \alpha_1, \alpha_2)$:

$$f^{(0)} = f(\alpha_0, \alpha_1, \alpha_2) \tag{36}$$

Then, we can improve the Evaluation Argument protocol.

Basefold Evaluation Argument Protocol Based on the Evaluation Form

Public Inputs

1. Commitment of $\tilde{f}$, $\pi_3 = \mathsf{Enc}_3(\mathbf{a})$
2. Evaluation point $\mathbf{u}$
3. Operation value $v = \tilde{f}(\mathbf{u})$

Witness

- Evaluation vector of MLE $\tilde{f}$, $\mathbf{a} = (a_0, a_1, \ldots, a_7)$

Such that

$$\tilde{f}(X_0, X_1, X_2) = \sum_{\mathbf{b} \in \{0,1\}^3} a_{\mathbf{b}} \cdot eq_{\mathbf{b}}(X_0, X_1, X_2) \tag{37}$$

First Round: Prover sends the evaluations of $h^{(2)}(X)$ at points $(0, 1, 2)$

$$h^{(2)}(X) = \sum_{b_1, b_2 \in \{0,1\}^2} f(X, b_1, b_2) \cdot \tilde{eq}((X, b_1, b_2), \mathbf{u}) \tag{38}$$

Since the right side is a degree 2 Univariate Polynomial, the Prover can compute the evaluations of $h^{(2)}(X)$ at $X = 0, 1, 2$:

$$\begin{aligned} h^{(2)}(0) &= a_0 \cdot e_0 + a_1 \cdot e_1 + a_2 \cdot e_2 + a_3 \cdot e_3 \\ h^{(2)}(1) &= a_4 \cdot e_4 + a_5 \cdot e_5 + a_6 \cdot e_6 + a_7 \cdot e_7 \\ h^{(2)}(2) &= \sum_{i=0}^{3} (2 \cdot a_{i+4} - a_i) \cdot (2 \cdot e_{i+4} - e_i) \end{aligned} \tag{39}$$

Here, $\mathbf{e}$ is the evaluation vector of $\tilde{eq}(\mathbf{X}, \mathbf{u})$.

Second Round: Verifier sends challenge $\alpha_2 \leftarrow \mathbb{F}_p$

Third Round: Prover simultaneously executes the Basefold-IOPP protocol and the Sumcheck protocol:

- Prover sends the folded vector encoding: $\pi_2 = \mathsf{fold}^*_{\alpha_2}(\pi_3)$
- Prover computes $h^{(2)}(\alpha_2)$ as the summation value for the next round of the Sumcheck protocol
- Prover computes the evaluations vector of $f^{(2)}(X_0, X_1)$: $\mathbf{a}^{(2)} = \mathsf{fold}^*_{\alpha_2}(\mathbf{a})$
- Prover computes and sends $h^{(1)}(X)$

$$h^{(1)}(X) = \sum_{b_0 \in \{0,1\}} f(b_0, X, \alpha_2) \cdot \tilde{eq}((b_0, X, \alpha_2), (u_2, u_1, u_0)) \tag{40}$$

Since the right side is also a degree 2 Univariate Polynomial in X, the Prover can compute the evaluations of $h^{(1)}(X)$ at $X = 0, 1, 2$: $(h^{(1)}(0), h^{(1)}(1), h^{(1)}(2))$.

Fourth Round: Verifier sends challenge $\alpha_1 \leftarrow \mathbb{F}_p$

Fifth Round: Prover continues executing the Basefold-IOPP protocol and the Sumcheck protocol:

- Prover sends the folded vector encoding $\pi_1 = \mathsf{fold}^*_{\alpha_1}(\pi_2)$
- Prover computes the summation value for the next round of the Sumcheck protocol: $h^{(1)}(\alpha_1)$
- Prover computes the evaluations vector of $\mathbf{a}^{(1)} = \mathsf{fold}^*_{\alpha_1}(\mathbf{a}^{(2)})$
- Prover computes and sends the evaluations of $h^{(0)}(X)$ at $X = 0, 1, 2$: $(h^{(0)}(0), h^{(0)}(1), h^{(0)}(2))$

$$h^{(0)}(X) = f(X_0, \alpha_1, \alpha_2) \cdot \tilde{eq}((X_0, \alpha_1, \alpha_2), (u_0, u_1, u_2)) \tag{41}$$

Sixth Round: Verifier sends challenge $\alpha_0 \leftarrow F$

Seventh Round: Prover continues executing the Basefold-IOPP protocol:

- Prover sends the folded vector encoding $\pi_0 = \mathsf{fold}^*_{\alpha_0}(\pi_1)$

Eighth Round: Verifier validates the following equations:

1. Verifier sends several rounds of Query, $Q = \{q_i\}$
2. Verifier checks the correctness of each folding step in Sumcheck:

$$
\begin{aligned}
h^{(2)}(0) + h^{(2)}(1) &\overset{?}{=} v \\
h^{(1)}(0) + h^{(1)}(1) &\overset{?}{=} h^{(2)}(\alpha_2) \\
h^{(0)}(0) + h^{(0)}(1) &\overset{?}{=} h^{(1)}(\alpha_1)
\end{aligned}
\tag{42}
$$

3. Verifier checks whether the final encoding π_0 is correct

$$
\pi_0 \overset{?}{=} \mathsf{enc}_0 \left(\frac{h^{(0)}(\alpha_0)}{\tilde{eq}((\alpha_0, \alpha_1, \alpha_2), \mathbf{u})} \right)
\tag{43}
$$

At this point, we have obtained a Basefold Evaluation Argument protocol based on the Evaluation Form. It can be directly connected to Sumcheck-based zkSNARKs or similar Jolt-style zkVMs.

References

- [ZCF23] Hadas Zeilberger, Binyi Chen, and Ben Fisch. "BaseFold: efficient field-agnostic polynomial commitment schemes from foldable codes." Annual International Cryptology Conference. Cham: Springer Nature Switzerland, 2024.
- Jiaheng Zhang, Tiancheng Xie, Yupeng Zhang, and Dawn Song. "Transparent polynomial delegation and its applications to zero knowledge proof." In *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 859-876. IEEE, 2020.