Introduction to WHIR

This blog introduces WHIR, a recent protocol for both low-degree test and polynomial commitment scheme (PCS), and the PCS can be either univariate or multivariate. WHIR claims to have a much faster verifier than its prior works: variants of FRI, STIR and BaseFold.

This blog assumes the reader is familiar with Reed-Solomon code and FRI. Please read this blog for the basic concepts about FRI. The reader is also assumed to be familiar with the Sum-check protocol.

FRI is widely used for building hash-based zkSNARKs. Particularly, FRI is used to build the core component of zkSNARKs called polynomial commitment scheme (PCS). Due to its close relation to univariate polynomials, it is natural to leverage FRI to build univariate PCS (UPCS). However, recently there is a trend to construct zkSNARKs over multilinear PCS (MPCS), because the sum-check protocol is recognized as a highly efficient component for building zkSNARKs, and MPCS is the key ingredient for sum-check-based zkSNARKs. FRI is, unfortunately, unable to be transformed into a MPCS, not with considerable overhead, until the arrival of BaseFold.

BaseFold is the first construction that modifies FRI into a MPCS with almost no additional overhead on the prover side. By cleverly combining FRI and sum-check, BaseFold produces a MPCS whose cost is the sum of that of FRI and sum-check. As sum-check is significantly cheaper than FRI, this overhead is far smaller than any previous transformations from FRI into MPCS.

However, BaseFold is not as efficient as (some variants of) FRI on the verifier side, because BaseFold could not benefit from some techniques that FRI uses to reduce the number of queries. One of the most important techniques is the DEEP technique, which, unfortunately, cannot be directly applied to BaseFold because it involves "quotienting" the polynomial, breaking the linearity BaseFold relies on to keep the consistence between the FRI side and the sum-check side.

Finally, here comes WHIR, which combines the insights from STIR (a variant of FRI that heavily exploits the DEEP technique) and BaseFold to produce a PCS (which can be configured to be either MPCS and UPCS) that has a faster verifier (and proof size) than both STIR and BaseFold. This blog is trying to capture the technical details of WHIR.

The Problem

To explain the mechanism of WHIR, we start from the very beginning problem that a PCS is designed to solve: given a polynomial $f(X)$, we want to produce a short commitment to $f(X)$, denoted by cm, such that anyone holding cm can ask the prover to evaluate $f(X)$ at any point z, and the prover must answer with the correct answer $f(z)$, together with a proof. The commitment ensures that the prover cannot generate a valid proof for any wrong answer.

The committing algorithm of WHIR is the same as that of FRI. The parties agree on a domain L beforehand, such that $|L|$ is several times larger than the polynomial size N (which is roughly the polynomial degree in univariate case, or 2^μ where μ is the number of variables in multivariate case). The prover computes the codeword of f with respect to this domain, denoted by $f(L) := \{f(x) \mid x \in L\}$. Then the prover computes the Merkle root of $f(L)$ and outputs it as the commitment.

The codeword is a vector of size $|L|$, and all valid codewords form a set of size $|\mathbb{F}|^N$, which we call a Reed-Solomon code, denoted by $RS(\mathbb{F}, L, \rho)$, where $\rho = \frac{N}{|L|}$ is called the code rate. From the point view of the verifier, after receiving the Merkle root, the committed vector may be a valid codeword in $RS(\mathbb{F}, L, \rho)$, in case the prover is honest, or a completely arbitrary malicious vector picked by the prover.

Now the verifier asks the prover to evaluate $f(X)$ at a point z. The prover answers with $y = f(z)$, then the prover must provide a proof that two statements simultaneously hold:
1. The committed vector is indeed a valid codeword, i.e., $f(L)$ for some $f(X)$.
2. The committed polynomial $f(X)$ indeed satisfies $f(z) = y$.

The proving protocol of WHIR is also similar to that of FRI in the overall structure: instead of directly proving the statement, we reduce the original statement into a smaller one, i.e., the prover provides cm' for another polynomial $f'(X)$, and the prover instead proves that $f'(z') = y'$ for some different z' and y'. To prove this new statement, they continue this reduction until f becomes small enough such that the prover may directly send the polynomial to the verifier.

Obviously, the key to the above structure is the reduction: we need to ensure that the original statement holds if and only if the new statement holds. This reduction is where the difference between FRI, STIR and WHIR lies.

The Equivalent Problem of WHIR

Unlike the previous protocols, the first step of WHIR is to define a new code that captures the two statements simultaneously. This new code is called the Constrained Reed-Solomon code. Formally, let $CRS[\mathbb{F}, L, \rho, w, \sigma]$ be defined as the set

$$\left\{ f(X) \in RS[\mathbb{F}, L, \rho] \mid \sum_{b \in \{0,1\}^\mu} w\left(\hat{f}(b), b\right) = \sigma \right\},$$

where $\hat{f}$ is the multilinear version of $\boldsymbol{f}$, i.e., the multilinear polynomial with the same coefficients of f, w is a weight function, and σ is a constant. Here, by choosing w appropriately, we can capture the second statement, i.e., $f(z) = y$ using the additional constraint $\sum_{b \in \{0,1\}^\mu} w\left(\hat{f}(b), b\right) = \sigma$.

For example, in the multivariate case, let $w\left(Z, \vec{X}\right) = Z \cdot eq\left(\vec{z}, \vec{X}\right)$, the sum

$$\sum_{b \in \{0,1\}^\mu} w\left(\hat{f}(b), b\right) = \sum_{b \in \{0,1\}^\mu} \hat{f}(b) \cdot eq(\vec{z}, b)$$

is exactly equal to the evaluation $f(\vec{z})$.

Using this new definition, we can alternatively describe the goal of the PCS as: prove that the given Merkle root is a commitment to $f(L)$ which is a codeword in the code $CRS[\mathbb{F}, L, \rho, w, \sigma]$, where w and σ are selected according to the evaluation point z and the claimed result y.

From now on, we can forget about the original definition of PCS (either MPCS or UPCS) and the evaluation point and results, knowing that the above definition is descriptive enough to capture both MPCS and UPCS for arbitrary z or y.

Reduction of WHIR

We are now left with a single problem: prove that a given committed vector is a member of $CRS[\mathbb{F}, L, \rho, w, \sigma]$ for any w and σ.

As mentioned above, the idea is to reduce the original problem into a smaller instance (but still of the same type). In our situation, we want the prover to send a new commitment cm' to the verifier, such that for a new domain L', new parameters ρ', w' and σ', the original commitment is valid if and only if cm' is commitment to some $f'(L') \in CRS[\mathbb{F}, L', \rho', w', \sigma']$, and most importantly, all these new stuffs should be several times smaller than the original ones.

The reduction starts in the same way as FRI. Let $f(X)$ be the original polynomial with $N = 2^\mu$ coefficients. Then the even and odd split of $f(X)$ are

$$f_e(X) = \frac{f(X) + f(-X)}{2} \quad \text{and} \quad f_o(X) = \frac{f(X) - f(-X)}{2X}$$

which satisfy $f(X) = f_e(X^2) + X \cdot f_o(X^2)$.

The folding of f by random scalar α is defined as $\mathsf{Fold}(f, \alpha) := f_e(X) + \alpha \cdot f_o(X)$. Given a vector of k scalars $\vec{\alpha} = (\alpha_0, ..., \alpha_{k-1})$ we define the folding of f by $\vec{\alpha}$ as folding f sequentially by α_i for $i = 0, ..., k-1$, i.e., $\mathsf{Fold}(f, \vec{\alpha}) := \mathsf{Fold}(\mathsf{Fold}(f, \alpha_0), \alpha_1)...)$.

Note that when we fold the univariate polynomial f, we are correspondingly evaluating the multilinear version $\hat{f}$ by its first variable. Therefore, $\mathsf{Fold}(f, \alpha)(X) = \hat{f}(\alpha, X)$, and similarly for multiple foldings $\mathsf{Fold}(f, \vec{\alpha})(X) = \hat{f}(\vec{\alpha}, X)$.

We are now ready to describe the concrete steps for producing the smaller instance.

1. Sum-check

The parties run the sum-check protocol with the polynomial $w\left(\hat{f}(X), X\right)$ and the target sum σ, but only execute the first k rounds. During these sum-check rounds, the prover sends k short polynomials to the verifier, and the verifier sends k random challenges $\alpha_0, ..., \alpha_{k-1}$ to the prover. At the end of these k sum-check rounds, the original statement about the sum is reduced to

$$\sum_{b \in \{0,1\}^{\mu-k}} w\left(\hat{f}(\vec{\alpha}, b), (\vec{\alpha}, b)\right) = \sigma^{(1)}$$

for some new target sum $\sigma^{(1)}$.

Note that now we have a new polynomial $\hat{f}(\vec{\alpha}, X)$, and a new weight function $w(Z, \vec{\alpha}, X)$. This new polynomial is exactly the folded polynomial mentioned above, and the result of this reduction, so let us denote it by $\hat{f}'(X) := \hat{f}(\vec{\alpha}, X)$. However, the new w and σ are still not yet the final results of the reduction, so we denote them respectively by $w^{(1)}(Z, X) := w(Z, \vec{\alpha}, X)$ and $\sigma^{(1)}$ temporarily.

2. Commit to new polynomial

Since $\hat{f}'(X)$ is already the end result of this reduction, the prover computes its commitment as an ordinary polynomial, i.e., evaluate the univariate version $f'(X)$ over the new domain $L' = L^2 = \{x^2 | x \in L\}$, and sends its Merkle root to the verifier. Note that L is a (coset of) multiplicative subgroup with order being a power of two, so the size of L' is exactly half of L.

3. Out of domain query

This step is from the DEEP technique. The verifier samples a random z_0 from $\mathbb{F}$, with overwhelming probability z_0 is not in L or L', and sends it to the prover. The prover answers with $y_0 = f'(z_0)$.

4. Shift queries

The verifier samples t random elements z_i inside $L^{2^k} = \left\{x^{2^k} | x \in L\right\}$, for i from 1 to t. The prover answers with $y_i = f'(z_i)$. The parameter t here is decided according to the security parameter. The correctness of y_i against the Merkle root of f is checked as follows.

Let $r_{ij} \in L$ for j from 0 to $2^k - 1$ be the 2^k-th roots of z_i, which are equal to one of the roots r multiplied by all 2^k-th roots of unity, which can then be viewed as the tensor product of k vectors $(1, \omega), (1, \omega^2), ..., \left(1, \omega^{2^{k-1}}\right)$, where ω is a primitive 2^k-th root of unity. The prover opens f at every r_{ij}. Then y_i should be equal to the result of:

1. First interpolate $f(r_{ij})$ over the k-dimensional hypercube $r \cdot (1, \omega) \times (1, \omega^2) \times ... \times \left(1, \omega^{2^{k-1}}\right)$.
2. Then evaluate the polynomial at $\vec{\alpha}$.

Alternatively, the prover doesn't send y_i in the first place, only sends $f(r_{ij})$, and the verifier computes y_i by itself.

5. Final reduction

The correctness of the query results with respect to the new polynomial f' are captured by equations

$$y_i = f'(z_i) = \hat{f}'(\vec{z}_i) = \sum_{b \in \{0,1\}^{\mu - k}} \hat{f}(b) \cdot eq(\vec{z}_i, b)$$

where $\vec{z}_i := \left(z_i, z_i^2, z_i^4, ..., z_i^{2^{\mu-k-1}}\right)$, for every i from 0 to t. These equations can also be captured by the weight function. Let $w^{(i+2)}(Z, X) := Z \cdot eq(\vec{z}_i, X)$ and $\sigma^{(i+2)} := y_i$ for i from 0 to t. To obtain the final weight function and the target sum, the verifier samples a random γ, and both parties compute

$$w' := \sum_{i=0}^{t+2} \gamma^i w^{(i)} \quad \text{and} \quad \sigma' := \sum_{i=0}^{t+2} \gamma^i \sigma^{(i)}.$$

This finishes one iteration of WHIR.