Zero Knowledge for WHIR

Zero-Knowledge WHIR as a PCS

The following scheme describes a Zero-Knowledge variant of WHIR as a PCS. The scheme uses a single run of non-ZK WHIR as a subprotocol. The scheme is zero knowledge in the bounded query model.

Given a multilinear witness polynomial $f(x_1, \cdots, x_\mu)$, the protocol works as follows:

1. Prover (P) masks f by adding a new variable y to create a new randomized *multilinear* polynomial

$$\hat{f}(x_1, \cdots, x_\mu, y) := f(x_1, \cdots, x_\mu) + y \cdot \mathsf{msk}(x_1, \cdots, x_\mu)$$

   where $\mathsf{msk}(\vec{x})$ should be random with at least as many non-zero coefficients as the *total* query bound. Furthermore, it must also not evaluate to zero on the NTT evaluation domain w.h.p. (Looking ahead, the oracle corresponding to $\hat{f}$ will only be queried on NTT evaluation domain and never as part of queries to the Sumcheck protocol.)
2. P additionally samples a random $\mu + 1$-variate *multilinear polynomial* $g(x_1, \cdots, x_\mu, y)$.
3. **Commitment**: P commits to f by sending two FRI oracles $([[\hat{f}]], [[g]])$, to the verifier V.
4. **Opening**: Verifier sends the evaluation point $\vec{a} := (a_1, \cdots, a_\mu)$ to the prover who opens f by computing the following:
   - Honest Prover computes $F = f(a_1, \cdots, a_\mu) = \hat{f}(a_1, \cdots, a_\mu, 0)$.
   - Honest Prover computes $G = g(a_1, \cdots, a_\mu, 0)$. (Notice y is set to zero.)
   - P $\longrightarrow$ V: Prover sends the pair (F, G) to the verifier.
5. **Proof Verification**: After receiving (F, G), the protocol starts verification as follows:
   - V $\longrightarrow$ P: Verifier sends a random challenge ρ to the prover. Verifier expects P to prove the claim

$$\rho \cdot F + G = \rho \cdot f(\vec{a}) + g(\vec{a}, 0)$$

   - Prover and Verifier proceed with non-ZK WHIR opening proof of the previous claim as follows:
   - Verifier:
     - Creates the **virtual oracle** $L := \rho \cdot [[\hat{f}]] + [[g]]$ and sets it as the first WHIR FRI oracle.
     - Verifier creates the WHIR weight polynomial $w(z, x_1, \cdots, x_\mu, y) := z \cdot \mathsf{eq}(\vec{x}, y;\ \vec{a}, 0)$.
     - Verifier runs the standard WHIR Sumcheck rounds receiving new folded FRI oracles as usual. If the prover is honest, the folded FRI oracles will consist of the folding of the polynomial

$$h(\vec{x}, y) := \rho \cdot \hat{f}(x_1, \cdots, x_\mu, y) + g(x_1, \cdots, x_\mu, y)$$

       And an honest prover's Sumcheck rounds will prove the claim:

$$\rho \cdot F + G = \rho \cdot \hat{f}(\vec{a}, 0) + g(\vec{a}, 0) = \sum_{\vec{x} \in \{0,1\}^{\mu+1}} \left[\rho \cdot \hat{f}(\vec{x}) + g(\vec{x})\right] \mathsf{eq}(\vec{x};\ \vec{a}, 0)$$

Notice that since $f(\vec{a}) = \hat{f}(\vec{a}, 0)$, the Sumcheck opening proof is indeed a proof of evaluation of f and is therefore binding, in spite of the mask!

- **OOD Queries**: The verifier should make its out-of-domain (OOD) queries as usual.
- **Shift Queries**: At the end of the first Sumcheck round and on receiving the first *folded* oracle corresponding to $h(\vec{x}, y)$, the Verifier should compute it's shift-query response using the virtual oracle L. For subsequent rounds, the protocol should follow normal WHIR operations.
- **Termination Check**: The verifier should do termination check as normal WHIR terminal check, i.e., locally compute the Sumcheck over final $w(z, \vec{x}, y)$, with z replaced by the last round of FRI oracle $h(\vec{r}, y)$ now sent in clear. Verifier should also check that the final FRI oracle is not a constant, i.e., it must have a non-zero y-coefficient. (Degree of y should also be one.)

  Note that at the μ-th Sumcheck round, an honest prover would have sent

  $$h(\vec{r}, y) \cdot \mathsf{eq}(\vec{r}, y; \vec{a}, 0) = (\rho \cdot [f(\vec{r}) + y \cdot \mathsf{msk}(\vec{r})] + g(\vec{r}, y)) \cdot \mathsf{eq}(\vec{r}, y; \vec{a}, 0)$$

  and the corresponding FRI oracle, sent in clear to the Verifier will be

  $$\rho \cdot [f(\vec{r}) + y \cdot \mathsf{msk}(\vec{r})] + g(\vec{r}, y)$$

  where $\vec{r}$ is the randomness sent during Sumcheck. Since the verifier had set $\vec{w}(z, \vec{x}, y) = z \cdot \mathsf{eq}(\vec{x}, y;\ \vec{a}, 0)$ at the start of the protocol, the result of locally computing

  $$\sum_{y \in \{0,1\}} h(\vec{r}, y) \cdot \mathsf{eq}(\vec{r}, y; \vec{a}, 0)) = h(\vec{r}, 0) = \rho \cdot f(\vec{r}) + g(\vec{r}, 0)$$

- Honest Prover
  - On receiving verifier's challenge ρ, the prover starts WHIR PCS evaluation proof for $(\vec{a}, 0)$ on the polynomial $h(\vec{x}, y) = \rho \cdot \hat{f}(x_1, \cdots, x_\mu, y) + g(x_1, \cdots, x_\mu, y)$. However, unlike a standard proof, the Prover should not compute the first proof oracle.
  - **OOD Queries**: For an out-of-domain (OOD) query q_o, the prover should respond with the evaluation of $h\left(\mathsf{pow2}(q_o), q_o^{2^\mu}\right)$ (i.e., without setting y to zero) to the verifier. Notice that out-of-domain queries are meant for $h(\vec{x})$, which by construction is blinded by g, and therefore doesn't leak information about f.

Zero-Knowledge WHIR for Σ-IOP

The following Zero-Knowledge variant of WHIR is applicable to Σ-IOP.

Given:

- A private *multilinear* witness polynomial $f(x_1, \cdots, x_\mu)$

- A public WHIR weight polynomial w of the form $w(z, x_1, \cdots, x_\mu) = z \cdot W(x_1, \cdots, x_\mu)$, and
- A Σ-IOP claim $F = \sum_{\vec{b} \in \{0,1\}^\mu} f(\vec{b}) \cdot W(\vec{b})$

The following protocol proves the above claim in zero-knowledge using non-ZK WHIR as a subprotocol. The protocol proceeds as follows:

1. As in the case of PCS, the Prover (P) masks f by adding a new variable y to create a new randomized multilinear polynomial with same requirements as in the case of PCS:

$$\hat{f}(x_1, \cdots, x_\mu, y) := f(x_1, \cdots, x_\mu) + y \cdot \mathsf{msk}(x_1, \cdots, x_\mu)$$

1. P additionally samples a random $\mu + 1$-variate multilinear polynomial $g(x_1, \cdots, x_\mu, y)$.
2. **Commitment**: P commits to the claim $F = \sum_{\vec{b} \in \{0,1\}^\mu} f(\vec{b}) \cdot W(\vec{b})$ as follows:
   - Prover computes a new weight polynomial

$$\hat{w}(z, \vec{x}, y) := z \cdot W(\vec{x}) \cdot \mathsf{eq}(0, y) = z \cdot (1 - y) \cdot W(\vec{x})$$

   - Prover computes

$$G = \sum_{\vec{b} \in \{0,1\}^\mu} g(\vec{b}, 0) \cdot W(\vec{b})$$

   - Prover computes the FRI oracles $[[\hat{f}]]$ and $[[g]]$
   - P $\longrightarrow$ V: Prover sends the quadruple $(F, G, [[\hat{f}]], [[g]])$ to the verifier.
3. **Proof Verification**: After receiving $(F, G, [[\hat{f}]], [[g]])$, the protocol starts verification as follows:
   - V $\longrightarrow$ P: Verifier sends a random challenge ρ to the prover. Verifier expects P to prove the claim

$$\rho \cdot F + G = \sum_{\vec{b} \in \{0,1\}^\mu} [\rho \cdot f(\vec{b}) + g(\vec{b}, 0)] \cdot W(\vec{b})$$

   - Prover and Verifier proceed with non-ZK WHIR opening proof of the previous claim as follows:
   - Verifier:
     - Creates the **virtual oracle** $L := \rho \cdot [[\hat{f}]] + [[g]]$ and sets it as the first WHIR FRI oracle.
     - Verifier creates a new WHIR weight polynomial

$$\hat{w}(z, x_1, \cdots, x_\mu, y) := z \cdot (1 - y) \cdot W(x_1, \cdots, x_\mu)$$

     and runs non-ZK WHIR as usual, expecting P to prove the Σ-IOP claim on $\rho \cdot F + G$.
     - Rest of the protocol should follow the same algorithm for OOD-queries, Shift-queries, and Termination checks as in the case of WHIR PCS.
   - Honest Prover should use the weight polynomial computed during the commitment phase, and follow rest of the protocol as in case of WHIR PCS.

Sampling Randomness

Sampling msk polynomial

Assuming desired round-by-round soundness error of $2^{-\lambda}$ and the corresponding first-round query complexity $q_\lambda(\rho) = \frac{\lambda}{1-\log_2(1+\rho)}$ targeting unique-decoding radius (for the *virtual oracles only*), the number of non-zero coefficients should be set to

$$C(n) := c \cdot n \cdot \frac{\lambda}{1 - \log_2(1+\rho)}; \quad 0 < \rho \leq \frac{1}{2}$$

Where n is the number of evaluation points on which the PCS will be opened ($n = 1$ for Σ-IOP) and c is a small constant ($c \approx 10$ should be sufficient, but c must be strictly greater than the degree of the Sumcheck round polynomial, which in case of WHIR is 2). This number takes into account those coefficients of msk that *may contribute* to the last round of Sumcheck polynomial when y is a free variable. Apart from the last round, coefficients of msk do not contribute any information to Sumcheck round polynomial $h(\vec{x}, y)$.

Sampling Sumcheck blinding polynomial $g(\vec{x}, y)$

Since $f(\vec{x})$ is multilinear, from Sumcheck perspective, it's sufficient for to sample $g(\vec{x}, y)$ such that g is a linear polynomial in each x_i. That is, for zero-knowledge Sumcheck, the following blinding polynomial is sufficient:

$$g(x_1, \cdots, x_\mu;\ y) = (a_{\mu+1} \cdot y + b_{\mu+1}) + \sum_{i=1}^{\mu} (a_i \cdot x_i + b_i)$$

where for all $i = 1, 2, \cdots, \mu+1$, a_i and b_i are sampled uniformly at random from $\mathbb{F}_q$.

However, if the query complexity $q_\lambda(\rho)$ of WHIR is greater than $2 \cdot (\mu + 2)$, i.e., the verifier will open the blinding polynomial at more than $2 \cdot (\mu + 2)$ points, the above scheme will reveal all the coefficients (a_i, b_i) making the scheme useless. Because of this, it's advisable to sample at least $\ell = q_\lambda(\rho) - 2\mu$ distinct random *monomials* (with individual degree 1) and coefficients $c_i \xleftarrow{\$} \mathbb{F}$

$$\begin{aligned} \pi_1(\vec{x};\ y) &\xleftarrow{\$} \mathsf{mon}(x_1, \cdots, x_\mu;\ y) \\ &\vdots \\ \pi_\ell(\vec{x};\ y) &\xleftarrow{\$} \mathsf{mon}(x_1, \cdots, x_\mu;\ y) \end{aligned}$$

to compute

$$g(x_1, \cdots, x_\mu;\ y) = (a_{\mu+1} \cdot y + b_{\mu+1}) + \sum_{i=1}^{\mu} (a_i \cdot x_i + b_i) + \sum_{j=1}^{\ell} c_j \cdot \pi_j(\vec{x};\ y)$$

(Recall that a monomial is a polynomial with a single term. For example, $\pi(x_1, \cdots, x_9;\ y) := x_1 x_5$ is a degree one monomial while $x_1 x_5 + 4$ is not!)

NOTE: $g(\vec{x};\ y)$ is an extremely sparse polynomial with at most $\mathsf{q} + \mu + 2$ non zero coefficients, and it will be more time and memory efficient to compute the FRI oracle for g using direct evaluation instead of using NTT.

Soundness

Since WHIR has round-by-round soundness, if zk-WHIR verifier accepts, that means the virtual oracle $L := \rho \cdot [[\hat{f}]] + [[g]]$ must have been close to a RS codeword. Therefore, by proximity gaps, with high probability, $\hat{f}$ must be a low degree polynomial in its univariate basis. Therefore, to compute soundness, we need to compute the following probability over the random choices of the WHIR verifier, (namely ρ, Sumcheck round randomness, and STIR and OOD query round randomness):

$$\mathbf{Pr}_{\rho}[F \neq \hat{f}(\vec{a}, 0) \mid \text{whir accepts}]$$

Let $\mathsf{s}_{\mathsf{whir}}$ be the overall soundness for a non-ZK WHIR verifier. For a given ρ, let $H(\rho) := G + \rho \cdot F$ and $h_\rho(\vec{x}, y) := g(\vec{x}, y) + \rho \cdot f(\vec{x}, y)$. Then

$$\begin{aligned}
&\mathbf{Pr}_{\rho}[F \neq \hat{f}(\vec{a}, 0) \mid \text{whir accepts}] \\
&= \mathbf{Pr}_{\rho}[F \neq \hat{f}(\vec{a}, 0) \wedge H(\rho) = h_\rho(\vec{a}, 0) \mid \text{whir accepts}] \\
&\quad + \mathbf{Pr}_{\rho}[F \neq \hat{f}(\vec{a}, 0) \wedge H(\rho) \neq h_\rho(\vec{a}, 0) \mid \text{whir accepts}] \\
&\leq \mathbf{Pr}_{\rho}[F \neq \hat{f}(\vec{a}, 0) \wedge H(\rho) = h_\rho(\vec{a}, 0)] + \mathsf{s}_{\mathsf{whir}}
\end{aligned}$$

The last inequality follows from the fact that WHIR has perfect completeness, so $\mathbf{Pr}[H(\rho) = h_\rho(\vec{a}, 0) \mid \text{whir accepts}] = \frac{1}{1+\mathsf{s}_{\mathsf{whir}}}$ and $\mathbf{Pr}[H(\rho) \neq h_\rho(\vec{a}, 0) \mid \text{whir accepts}] = \frac{\mathsf{s}_{\mathsf{whir}}}{1+\mathsf{s}_{\mathsf{whir}}}$.

Furthermore,

$$\begin{aligned}
&\mathbf{Pr}_{\rho}[F \neq \hat{f}(\vec{a}, 0) \wedge H(\rho) = h_\rho(\vec{a}, 0)] \\
&= \mathbf{Pr}_{\rho}[H(\rho) = h_\rho(\vec{a}, 0) | F \neq \hat{f}(\vec{a}, 0)] \cdot \mathbf{Pr}_{\rho}[F \neq \hat{f}(\vec{a}, 0)] \\
&= \mathbf{Pr}_{\rho}[H(\rho) = h_\rho(\vec{a}, 0) | F \neq \hat{f}(\vec{a}, 0)] \\
&= \mathbf{Pr}_{\rho}\left[\rho = \frac{g(\vec{a}, 0) - G}{F - \hat{f}(\vec{a}, 0)} | F \neq \hat{f}(\vec{a}, 0)\right] \\
&= \frac{1}{|\mathbb{F}_q|}
\end{aligned}$$

Therefore:

$$\mathbf{Pr}_{\rho}[F \neq \hat{f}(\vec{a}, 0) \mid \text{whir accepts}] \leq \frac{1}{|\mathbb{F}_q|} + \mathsf{s}_{\mathsf{whir}}$$

NOTE: This only establishes a bound for $\hat{f}(\vec{a}, 0)$ and not on $f(\vec{a})$. Since a polynomial commitment only needs to be binding, the above scheme is sufficient, because given a commitment $\mathcal{C}$ and an evaluation point $\vec{a}$, the

above scheme cannot be opened as two different values F and F' for the same $\mathcal{C}$.

However, for a randomized commitment scheme where the same polynomial $f(\vec{x})$ can have multiple random commitments, a stronger notion of binding is possible. Let $\mathcal{C}_1, \cdots, \mathcal{C}_t \xleftarrow{\$} \mathsf{commit}(f)$ be the randomized commitments for the same polynomial $f(x)$ let $\vec{a}$ be an arbitrarily chosen evaluation point. Then the randomized commitment scheme is strongly binding if opening any $\mathcal{C}_i$ at $\vec{a}$ must return the same value with w.h.p., i.e., $\forall \vec{a} \in \mathbb{F}_q^n : \mathbf{Pr}_{i \neq j}[\mathsf{open}(\mathcal{C}_i, \vec{a}) \neq \mathsf{open}(\mathcal{C}_j, \vec{a})] < \mathsf{negl}(\mathsf{n})$.

The zk-WHIR protocol described above can be trivially modified to achieve this stronger notion of binding: Instead of setting $y = 0$ deterministically, if y is chosen uniformly at random by the verifier (and the weight polynomial computed accordingly), the resulting scheme is guaranteed to open to the same value except with probability $1/|\mathbb{F}_q|$. To achieve this, steps-4 and 5 should be modified as follows:

4. **Opening**: Verifier sends the evaluation point $\vec{a} := (a_1, \cdots, a_\mu)$ to the prover who opens f by computing the following:
   - Honest Prover computes $F = f(a_1, \cdots, a_\mu)$
   - Honest Prover computes $F' := \mathsf{msk}(a_1, \cdots, a_\mu)$
   - Honest Prover writes $g(x_1, \cdots, x_\mu, y) := g_1(x_1, \cdots, x_\mu) + y \cdot g_1(x_1, \cdots, x_\mu)$. Since $g(x_1, \cdots, x_\mu, y)$ is expected to be multilinear for a honest prover, this form of decomposition is always possible.
   - Honest Prover computes $G := g_1(a_1, \cdots, a_\mu)$ and $G' := g_2(a_1, \cdots, a_\mu)$
   - P $\longrightarrow$ V: Prover sends *four field elements* (F, F', G, G') to the verifier.
5. **Proof Verification**: After receiving (F, F', G, G'), the protocol starts verification as follows:
   - V $\longrightarrow$ P: Verifier sends *two random challenges* ρ and γ to the prover. Verifier expects P to prove the claim.

     $$\rho \cdot (F + \gamma \cdot F') + (G + \gamma \cdot G') = \rho \cdot \hat{f}(\vec{a}, \gamma) + g(\vec{a}, \gamma)$$

   - Prover and Verifier proceed with non-ZK WHIR opening proof of the previous claim as follows:
   - Verifier:
     - Creates the **virtual oracle** $L := \rho \cdot [[\hat{f}]] + [[g]]$ and sets it as the first WHIR FRI oracle.
     - Verifier creates the WHIR weight polynomial $w(z, x_1, \cdots, x_\mu, y) := z \cdot \mathsf{eq}(\vec{x}, y;\ \vec{a}, \gamma)$.
     - Verifier runs the standard WHIR Sumcheck rounds receiving new folded FRI oracles as usual. If the prover is honest, the folded FRI oracles will consist of the folding of the polynomial

       $$h(\vec{x}, y) := \rho \cdot \hat{f}(x_1, \cdots, x_\mu, y) + g(x_1, \cdots, x_\mu, y)$$

       And an honest prover's Sumcheck rounds will prove the claim:

$$\rho \cdot F + G + \gamma \cdot (\rho \cdot F' + G') = \rho \cdot \hat{f}(\vec{a}, \gamma) + g(\vec{a}, \gamma) = \sum_{\vec{x} \in \{0,1\}^{\mu+1}} \left[\rho \cdot \hat{f}(\vec{x}) + g(\vec{x})\right] \mathsf{eq}(\vec{x};\ \vec{a}, \gamma)$$

Notice that since γ is chosen randomly by the verifier, the probability with which two different randomized commitments can open to different values is $O(\frac{1}{|\mathbb{F}_q|})$, i.e.,

$$\begin{aligned}
&\mathbf{Pr}_{\gamma_i \in \mathbb{F}_q^\times}\left[F_1 + \gamma_1 \cdot F_1' = F_2 + \gamma_2 \cdot F_2' \quad | \quad F_1 \neq F_2\right] \\
&= \sum_{\gamma \in \mathbb{F}_q^\times} \mathbf{Pr}_{\gamma_1}\left[F_1 + \gamma_1 \cdot F_1' = F_2 + \gamma \cdot F_2' \mid (F_1 \neq F_2) \wedge (\gamma_2 = \gamma)\right] \cdot \mathbf{Pr}[\gamma_2 = \gamma] \\
&= \sum_{\gamma \in \mathbb{F}_q^\times} \mathbf{Pr}_{\gamma_1}\left[\frac{1}{\gamma_1} = \frac{F_1'}{(F_2 + \gamma \cdot F_2' - F_1)} \mid (F_1 \neq F_2) \wedge (\gamma_2 = \gamma)\right] \cdot \mathbf{Pr}[\gamma_2 = \gamma] \\
&= \sum_{\gamma \in \mathbb{F}_q^\times} \frac{1}{|\mathbb{F}_q| - 1} \cdot \mathbf{Pr}[\gamma_2 = \gamma] \\
&= \sum_{\gamma \in \mathbb{F}_q^\times} \left(\frac{1}{|\mathbb{F}_q| - 1}\right)^2 \\
&= \frac{1}{|\mathbb{F}_q| - 1}
\end{aligned}$$