0.1 Probabilistically Checkable Proofs

Before going further we should get acquainted with one more concept from the computational complexity theory, that have an important application in zk-SNARK and provides the theoretical backbone.

A Probabilistically Checkable Proof (PCP) is a type of proof system where the verifier can efficiently check the correctness of a proof by examining only a small, random portion of it, rather than verifying it entirely.

Definition 0.1. A language $\mathcal{L} \subseteq \Sigma^*$ (for some given alphabet Σ) is in the class $\mathsf{PCP}(r, q)$ (**probabilistically checkable proofs**), where r is the *randomness complexity* and q is the *query complexity*, if for a given pair of algorithms $(\mathcal{P}, \mathcal{V})$:

- *Syntax:* $\mathcal{P}$ calculates a proof (bit string) $\pi \in \Sigma^*$ in polynomial time $\mathsf{poly}(|x|)$ of the common input x. The prover $\mathcal{P}$ and verifier $\mathcal{V}$ interact, where the verifier has an oracle access to π (meaning, he queries it at any position).
- *Complexity:* $\mathcal{V}$ uses at most r random bits to decide which part of the proof to query and the verifier queries at most q bits of the proof.

Such pair of algorithms $(\mathcal{P}, \mathcal{V})$ should satisfy the following properties (with a security parameter $\lambda \in \mathbb{N}$):

- **Completeness**: If $x \in \mathcal{L}$, then $\Pr[\mathcal{V}^{\pi}(x) = 1] = 1$.
- **Soundness**: If $x \notin \mathcal{L}$, then for any possible (malicious) proof π^*,

$$\Pr\left[\mathcal{V}^{\pi^*}(x) = 1\right] = \mathsf{negl}(\lambda).$$

This allows a verification of huge statements with high confidence while using limited computational resources. See Figure 0.2.

Theorem 0.2. PCP theorem (PCP characterization theorem)
Any decision problem in NP has a PCP verifier that uses logarithmic randomness $O(\log n)$ and a constant number of queries $O(1)$, independent of n.

$$\mathsf{NP} = \mathsf{PCP}(O(\log n), O(1))$$

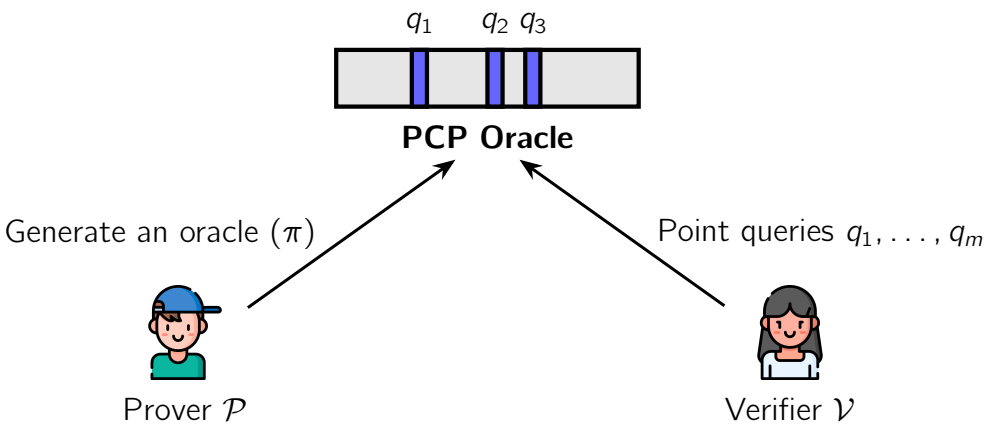

Figure 0.1: Illustration of a Probabilistically Checkable Proof (PCP) system. The prover $\mathcal{P}$ generates a PCP oracle π that is queried by the verifier $\mathcal{V}$ at specific points $q_1, \ldots, q_m$.

However, despite the fact that PCP is a very powerful tool, it is not used directly in zk-SNARKs. We need to extend it a bit to make it more suitable for our purposes.

Three main extensions of PCPs that are frequently used in SNARKs are:

- **IPCP** (**Interactive PCP**): The prover commits to the PCP oracle and then, based on the interaction between the prover and verifier, the verifier queries the oracle and decides whether to accept the proof.
- **IOP** (**Interactive Oracle Proof**): The prover and verifier interact and on each round, the prover commits to a new oracle. The verifier queries the oracle and decides whether to accept the proof.
- **LPCP** (**Linear PCP**): The prover commits to a linear function and the verifier queries the function at specific points.

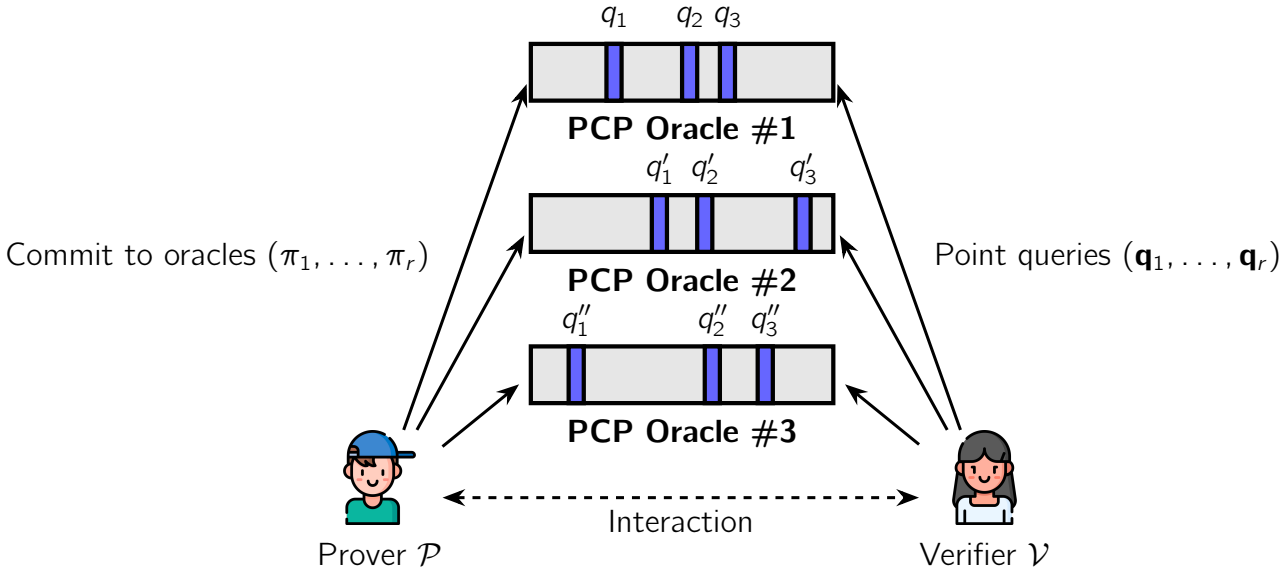

Figure 0.2: Illustration of an Interactive Oracle Proof (IOP). On each round i ($1 \leq i \leq r$), $\mathcal{V}$ sends a message m_i, and $\mathcal{P}$ commits to a new oracle π_i, which $\mathcal{V}$ can query at $\mathbf{q}_i = (q_{i,1}, \ldots, q_{i,m})$.

While IOPs will be later used for PLONK and zk-STARKs, we will focus on Linear PCPs in the context of Groth16 zk-SNARK. Let us define it below.

Definition 0.3 (Linear PCP)**.** A **Linear PCP** is a PCP where the prover commits to a linear function $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_k)$ and the verifier queries the function at specific points $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_r$. Then, the prover responds with the values of the function at these points:

$$\langle \boldsymbol{\pi}_1, \mathbf{q}_1 \rangle, \langle \boldsymbol{\pi}_2, \mathbf{q}_2 \rangle, \ldots, \langle \boldsymbol{\pi}_r, \mathbf{q}_r \rangle.$$

Example (QAP as a Linear PCP)**.** Recall that key QAP equation is:

$$\left(\sum_{i=1}^{n} w_i L_i(X)\right)\left(\sum_{i=1}^{n} w_i R_i(X)\right) - \left(\sum_{i=1}^{n} w_i O_i(X)\right) = Z(X)H(X).$$

Now, the notation might be confusing, but firstly, we denote vectors of polynomials:

$$\begin{aligned} \boldsymbol{L}(X) &= (L_1(X), \ldots, L_n(X)), \\ \boldsymbol{R}(X) &= (R_1(X), \ldots, R_n(X)), \\ \boldsymbol{O}(X) &= (O_1(X), \ldots, O_n(X)). \end{aligned}$$

Now, consider the following **linear PCP for QAP**:

1. $\mathcal{P}$ commits to an extended witness $\mathbf{w}$ and coefficients $\mathbf{h} = (h_1, \ldots, h_n)$ of $H(x)$.
2. $\mathcal{V}$ samples $\gamma \xleftarrow{R} \mathbb{F}$ and sends query $\boldsymbol{\gamma} = (\gamma, \gamma^2, \ldots, \gamma^n)$ to $\mathcal{P}$.
3. $\mathcal{P}$ reveals the following values:

$$\pi_1 \leftarrow \langle \mathbf{w}, \boldsymbol{L}(\gamma) \rangle, \quad \pi_2 \leftarrow \langle \mathbf{w}, \boldsymbol{R}(\gamma) \rangle, \quad \pi_3 \leftarrow \langle \mathbf{w}, \boldsymbol{O}(\gamma) \rangle, \quad \pi_4 \leftarrow Z(\gamma) \cdot \langle \mathbf{h}, \boldsymbol{\gamma} \rangle.$$

4. $\mathcal{V}$ checks whether $\pi_1\pi_2 - \pi_3 = \pi_4$.

Of course, the above example cannot be used as it is: at the very least, we have not specified how the prover commits to the extended witness $\mathbf{w}$ and coefficients $\mathbf{h}$ and then ensures consistency of $\pi_1, \ldots, \pi_4$ with these commitments. For that reason, we need some more tools to make it work which we learned in the previous lectures.

0.1.1 PCP application in QAP

When constructing a Quadratic Arithmetic Program (QAP) for a circuit C, we represented the whole circuit's computation using the following relation:

$$L(X)R(X) - O(X) = Z(X)H(X),$$

where by $L(X)$, $R(X)$, $O(X)$ we denote the polynomials that represent the left, right and output wires of the circuit, respectively. $Z(X)$ is the target polynomial, while $H(X) := M(X)/Z(X)$ for master polynomial $M(X) = L(X)R(X) - O(X)$ is the quotient polynomial.

We effectively managed to transform all the circuit's constraints, and computations in the short form. It still allows one to verify that each computational step is preserved by verifying the polynomial evaluation in specific (random) points, instead of recomputing everything. However,

it is not quite clear why such a check is safe and how it can be used in a PCP. In other words, why checking that $L(s)R(s) - O(s) = Z(s)H(s)$ for randomly selected s is enough to verify the circuit C?

Soundness justification. Why is it safe to use such a check? As it was said early, we perform all the computations in some finite field $\mathbb{F}$. The polynomials $L(X)$, $R(X)$ and $O(X)$ are interpolated polynomials using $|C|$ (number of gates) points, so

$$\deg(L) \leq |C|, \quad \deg(R) \leq |C|, \quad \deg(O) \leq |C|$$

Thus, using properties of polynomials' degrees, we can estimate the degree of polynomial $M(X) = L(X)R(X) - O(X)$.

$$\deg(M) \leq \max\{\deg(L) + \deg(R), \deg(O)\} \leq 2|C|$$

Now, using the Schwartz-Zippel Lemma (see **??**), we can deduce that if an adversary $\mathcal{A}$ does not know a valid witness $\mathbf{w}$, resolving the circuit C, he can compute a polynomial $\widetilde{M}(X) \leftarrow \mathcal{A}(\cdot)$ that satisfies a verifier $\mathcal{V}$ with probability less than $2|C|/|\mathbb{F}|$. To put it formally, we can write:

$$\Pr_{s \xleftarrow{R} \mathbb{F}}[\widetilde{M}(s) = M(s)] \leq \frac{2|C|}{|\mathbb{F}|}$$

This probability becomes negligible as $|\mathbb{F}|$ grows large (which is typically the case), giving us soundness. In the same time, the verifier accepts the $M(X)$ generated using a valid witness with probability 1 giving us the completeness, so, we can categorize QAP as PCP.

We will modify the form of our proof with the next modifications, but still preserve the PCP properties.

In the following sections, we will introduce tools needed to succintly verify the equality above using the PCP properties. Since the overall proof is very complex from the very first glance, we will break it down into smaller parts and explain each of them in detail. Let us start with the first step.

0.1.2 Attempt #1: Encrypted Verification

Now, assume we have the cyclic group $\mathbb{G}$ of prime order q with a generator g. Typically, this is the group of points on an elliptic curve. Assume for simplicity that $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a symmetric pairing function, where $(\mathbb{G}_T, \times)$ is a target group.

Now, suppose during the setup phase, we have a trusted party that generated a random value $\tau \xleftarrow{R} \mathbb{F}$ and public parameters $g^\tau, g^{\tau^2}, \ldots, g^{\tau^d}$ for $d = 2|C|$ — maximum degree of used polynomials (later we will use notation $\{g^{\tau^i}\}_{i \in [d]}$ for brevity). Then, party **deleted** τ. This way, we can now find the KZG commitment for each polynomial. Indeed, for example,

$$\mathsf{com}(L) \triangleq g^{L(\tau)} = g^{\sum_{i=0}^{d} L_i \tau^i} = \prod_{i=0}^{d} (g^{\tau^i})^{L_i},$$

and the same goes for $g^{R(\tau)}, g^{O(\tau)}, g^{H(\tau)}, g^{Z(\tau)}$. Now, given these give points, how can we verify that the polynomial $M(X) = L(X)R(X) - O(X)$ is correct? Well, first notice that the check is equivalent to

$$L(\tau)R(\tau) = Z(\tau)H(\tau) + O(\tau).$$

Notice that we transferred $O(\tau)$ to the right side of the equation to further avoid finding the inverse. Now, we can check this equality using encrypted values as follows:

$$e(\mathsf{com}(L), \mathsf{com}(R)) = e(\mathsf{com}(Z), \mathsf{com}(H)) \cdot e(\mathsf{com}(O), g),$$

Remark. One might ask: why is the above equation correct? Well, let us see:

$$\begin{aligned}
& e(\mathsf{com}(L), \mathsf{com}(R)) = e(\mathsf{com}(Z), \mathsf{com}(H)) \cdot e(\mathsf{com}(O), g) && \textit{Initial statement} \\
\Leftrightarrow\ & e(g^{L(\tau)}, g^{R(\tau)}) = e(g^{Z(\tau)}, g^{H(\tau)}) \cdot e(g^{O(\tau)}, g) && \textit{KZG Commitment Definition} \\
\Leftrightarrow\ & e(g, g)^{L(\tau)R(\tau)} = e(g, g)^{Z(\tau)H(\tau)} e(g, g)^{O(\tau)} && \textit{Pairing bilinearity} \\
\Leftrightarrow\ & e(g, g)^{L(\tau)R(\tau)} = e(g, g)^{Z(\tau)H(\tau)+O(\tau)} && \textit{Exponent product rule} \\
\Leftrightarrow\ & L(\tau)R(\tau) = Z(\tau)H(\tau) + O(\tau) && \textit{QAP Check} \\
\Leftrightarrow\ & L(X)R(X) \equiv Z(X)H(X) + O(X) && \textit{Schwarz-Zippel Lemma}
\end{aligned}$$

So, sounds like we are done. Let us summarize what we have done so far:

Attempt #1: Non-sound SNARK Protocol

Suppose we are given a circuit $\mathcal{C}$ with a maximum degree d of polynomials used underneath. Thus, all parties additionally know the target polynomial $Z(X)$.

Setup(1^λ)

The *trusted party* conducts the following steps:

- Picks a random value $\tau \xleftarrow{R} \mathbb{F}$.
- Calculates the public parameters $\{g^{\tau^i}\}_{i\in[d]}$.
- **Deletes** τ (toxic waste).
- **Outputs** prover parameters $\mathsf{pp} \leftarrow \{g^{\tau^i}\}_{i\in[d]}$ and verifier parameters $\mathsf{vp} \leftarrow \mathsf{com}(Z)$.

Prove($\mathsf{pp}, \mathbf{x}, \mathbf{w}$)

The prover $\mathcal{P}$ conducts the following steps:

- Runs the circuit with the statement $\mathbf{x}$ and witness $\mathbf{w}$, obtains the intermediate constraint values, and calculates the polynomials $L(X)$, $R(X)$, $O(X)$ through Lagrange Interpolation.
- Calculates $H(X) \leftarrow (L(X)R(X) - O(X))/Z(X)$.
- Calculates the KZG commitments as follows:

  $$\pi_L \leftarrow \mathsf{com}(L), \pi_R \leftarrow \mathsf{com}(R), \pi_O \leftarrow \mathsf{com}(O), \pi_H \leftarrow \mathsf{com}(H),$$

  using powers of τ from the prover parameters pp.
- Publishes $\boldsymbol{\pi} = (\pi_L, \pi_R, \pi_O, \pi_H)$ as a proof.

Verify($\mathsf{vp}, \mathbf{x}, \boldsymbol{\pi}$)

Upon receiving $\boldsymbol{\pi} = (\pi_L, \pi_R, \pi_O, \pi_H)$, using $\mathsf{com}(Z)$ from the verifier parameters vp, the verifier $\mathcal{V}$ checks:

$$e(\pi_L, \pi_R) = e(\mathsf{com}(Z), \pi_H) \cdot e(\pi_O, g).$$

This sounds like an end to the story. However, there is a problem with this approach: there is no guarantee that commitments π_L, π_R, π_O, π_H were indeed obtained through exponentiating the base g by the corresponding values $L(\tau)$, $R(\tau)$, $O(\tau)$, and $H(\tau)$. In other words, how can the verifier know that prover indeed knows, say, polynomial $L(X)$ such that $\pi_L = g^{L(\tau)}$?

0.1.3 Attempt #2: Including Proof of Exponent

In this section, we introduce the **Proof of Exponent assumption** (PoE) which makes KZG knowledge sound. Let us define it below.

Definition 0.4 (Proof of Exponent for KZG Commitment)**.** A **Proof of Exponent** (PoE) is a protocol that allows the prover $\mathcal{P}$ to convince the verifier $\mathcal{V}$ that he obtained a value $\mathsf{com}(f)$ through exponentiating a base g by $f(\tau)$. The protocol works as follows:

1. **Setup:** Proper parameters pp now contain not only $\{g^{\tau^i}\}_{i\in[d]}$, but also $\{g^{\alpha\tau^i}\}_{i\in[d]}$ for randomly selected $\tau, \alpha \xleftarrow{R} \mathbb{F}$ and further **deleted** values.
2. **Commit:** $\mathcal{P}$ commits to two values: $\mathsf{com}(f) = g^{f(\tau)}$ and $\mathsf{com}'(f) = g^{\alpha f(\tau)}$. The latter can be computed using pp as follows:
$$\mathsf{com}'(f) = \prod_{i=0}^{d} (g^{\alpha\tau^i})^{f_i}$$
3. **Verify:** $\mathcal{V}$ additionally checks $e(\mathsf{com}(f), g^\alpha) = e(\mathsf{com}'(f), g)$.

The informal reason why it makes KZG commitment sound is following: suppose we have an adversary prover $\mathcal{P}^*$ that published commitment c without knowing underlying polynomial $f(X)$. Now, there is no way for him to cheat the verifier. Indeed, what $\mathcal{P}^*$ needs to do is calculating c^α, but he does now know α since, similarly to τ, it was deleted as a part of the toxic waste. Besides, he cannot obtain α for the same reason he cannot obtain τ.

For that reason, we modify the SNARK protocol to include not only commitments to polynomials but also PoE for each of them. Let us see how it looks like.

Attempt #2: SNARK with PoE included

Suppose we are given a circuit C with a maximum degree d of polynomials used underneath. Thus, all parties additionally know the target polynomial $Z(X)$.

Setup(1^λ)

The *trusted party* conducts the following steps:

- Picks a random value $\tau, \alpha \xleftarrow{R} \mathbb{F}$.
- Calculates the public parameters $\{g^{\tau^i}\}_{i\in[d]}$, $\{g^{\alpha\tau^i}\}_{i\in[d]}$.
- **Deletes** τ, α (toxic waste).
- **Outputs** proper parameters $\mathsf{pp} \leftarrow \{\{g^{\tau^i}\}_{i\in[d]}, \{g^{\alpha\tau^i}\}_{i\in[d]}\}$, and verification parameters $\mathsf{vp} \leftarrow \{g^{Z(\tau)}, g^\alpha\}$.

Prove($\mathsf{pp}, \mathbf{x}, \mathbf{w}$)

The prover $\mathcal{P}$ conducts the following steps:

- Runs the circuit with the statement $\mathbf{x}$ and witness $\mathbf{w}$, obtains the intermediate constraint values, and calculates the polynomials $L(X), R(X), O(X)$ through Lagrange Interpolation.
- Calculates $H(X) \leftarrow (L(X)R(X) - O(X))/Z(X)$.
- Calculates the sound KZG commitments as follows:

$$\begin{aligned} \pi_L &\leftarrow g^{L(\tau)}, \quad & \pi'_L &\leftarrow g^{\alpha L(\tau)} \\ \pi_R &\leftarrow g^{R(\tau)}, \quad & \pi'_R &\leftarrow g^{\alpha R(\tau)} \\ \pi_O &\leftarrow g^{O(\tau)}, \quad & \pi'_O &\leftarrow g^{\alpha O(\tau)} \\ \pi_H &\leftarrow g^{H(\tau)}, \quad & \pi'_H &\leftarrow g^{\alpha H(\tau)}. \end{aligned}$$

  using powers $\{g^{\tau^i}\}_{i\in[d]}$ and $\{g^{\alpha\tau^i}\}_{i\in[d]}$ from the proper parameters pp.
- Publishes $\boldsymbol{\pi} = (\pi_L, \pi'_L, \pi_R, \pi'_R, \pi_O, \pi'_O, \pi_H, \pi'_H)$ as a proof.

Verify($\mathsf{vp}, \mathbf{x}, \boldsymbol{\pi}$)

Upon receiving $\boldsymbol{\pi} = (\pi_L, \pi'_L, \pi_R, \pi'_R, \pi_O, \pi'_O, \pi_H, \pi'_H)$, the verifier $\mathcal{V}$ checks:

$$\begin{gathered} e(\pi_L, \pi_R) = e(\mathsf{com}(Z), \pi_H) \cdot e(\pi_O, g), \\ e(\pi_L, g^\alpha) = e(\pi'_L, g), \quad e(\pi_R, g^\alpha) = e(\pi'_R, g), \\ e(\pi_O, g^\alpha) = e(\pi'_O, g), \quad e(\pi_H, g^\alpha) = e(\pi'_H, g). \end{gathered}$$

The provided protocol is secure under the PoE assumption. However, it is still not fully sound. Currently, there is no guarantee that when evaluating π_L, π_R, π_O we used the same extended witness $\mathbf{w}$. In other words, the prover can still cheat by using different extended witnesses for each polynomial (faking the proof system is still hard in this situation, but we want to make sure to eliminate all possible weaknesses). Let us see how to fix this!

0.1.4 Attempt #3: Making SNARK Sound

Besides fixing the issue with consistent use of witness $\mathbf{w}$, we additionally include one more optimization we have not included so far.

Optimization. Left/Right/Output Polynomial Preprocessing. Recall that:

$$L(X) = \sum_{i=0}^{n} w_i L_i(X), \quad R(X) = \sum_{i=0}^{n} w_i R_i(X), \quad O(X) = \sum_{i=0}^{n} w_i O_i(X).$$

while $M(X) = L(X)R(X) - O(X)$ is only known to the prover $\mathcal{P}$ since it contains the extended witness ($\mathbf{w}$) coefficients. However, the set of polynomials

$$\{L_i(X)\}_{i\in[n]}, \{R_i(X)\}_{i\in[n]}, \{O_i(X)\}_{i\in[n]}$$

are known in advance. Meaning, we can precompute the values of $\{g^{L_i(\tau)}\}_{i\in[n]}$, $\{g^{\alpha L_i(\tau)}\}_{i\in[n]}$, $\{g^{R_i(\tau)}\}_{i\in[n]}$, $\{g^{\alpha R_i(\tau)}\}_{i\in[n]}$, $\{g^{O_i(\tau)}\}_{i\in[n]}$, $\{g^{\alpha O_i(\tau)}\}_{i\in[n]}$ and use them in the prover parameters pp.

How? Suppose the prover $\mathcal{P}$ knows the extended witness $\mathbf{w}$. Consider the polynomial $L(X) = \sum_{i=0}^{n} w_i L_i(X)$. $\mathcal{P}$ can compute the KZG commitment π_L and its PoE π'_L as follows:

$$\pi_L \triangleq g^{L(\tau)} = g^{\sum_{i=0}^{n} w_i L_i(\tau)} = \prod_{i=0}^{n} (g^{L_i(\tau)})^{w_i}, \quad \pi'_L \triangleq g^{\alpha L(\tau)} = g^{\alpha \sum_{i=0}^{n} w_i L_i(\tau)} = \prod_{i=0}^{n} (g^{\alpha L_i(\tau)})^{w_i}.$$

Fix. Witness consistency check.

Introducing new term with β. To prove that the same $\mathbf{w}$ is used in all commitments, we need some "checksum" term that will somehow combine all polynomials $L(X)$, $R(X)$, and $O(X)$ with the witness $\mathbf{w}$. Moreover, we will need to compare this term with proofs π_L, π_R, and π_O. The best candidate for this is the following group element for some other random $\beta \xleftarrow{R} \mathbb{F}$:

$$\pi_\beta = g^{L(\tau)+R(\tau)+O(\tau)} = \prod_{i=1}^{n} (g^{L_i(\tau)+R_i(\tau)+O_i(\tau)})^{w_i}, \ \pi'_\beta = \prod_{i=1}^{n} (g^{\beta(L_i(\tau)+R_i(\tau)+O_i(\tau))})^{w_i}$$

This way, we get a term that includes all three polynomials $L(X)$, $R(X)$, and $O(X)$, and all coefficients of the extended witness $\mathbf{w}$. Moreover, verifier $\mathcal{V}$ can compare π_β with three other commitments π_L, π_R, π_O to ensure that all of them are consistent. This is done through the following check:

$$e(\pi_L \pi_R \pi_O, g^\beta) = e(\pi'_\beta, g).$$

Again, this approach still has a weakness (yeah-yeah, I am also tired of this). Indeed, this check is complete (meaning, if $\mathbf{w}$ is used consistently across π_L, π_R, π_O, then the check will pass), but it is still not sound with an overwhelming probability. Indeed, suppose $\mathbf{w}$ is used inconsistently, meaning, we have extended witnesses $\mathbf{w}_L$, $\mathbf{w}_R$, $\mathbf{w}_O$, and $\mathbf{w}_\beta$, each for corresponding polynomials. If the witness is consistent, the following condition must hold:

$$(w_{L,i} L_i(\tau) + w_{R,i} R_i(\tau) + w_{O,i} O_i(\tau))\beta = w_{\beta,i}\beta(L_i(\tau) + R_i(\tau) + O_i(\tau)) \quad \forall i \in [n]$$

Assume otherwise. Consider a simple situation where it happens that $L_i \equiv R_i$, meaning $L_i(\tau)$ and $R_i(\tau)$ are the same (call them q). Then,

$$(w_{L,i} + w_{R,i})q + w_{O,i} O_i(\tau) = w_{\beta,i}(2q + O_i(\tau)) \quad \forall i \in [n]$$

For arbitrarily chosen $w_{R,i}$ and $w_{O,i}$, the adversary prover $\mathcal{P}^*$ can set $w_{\beta,i} := w_{O,i}$ and $w_{L,i} = 2w_{O,i} - w_{R,i}$. It is easy to verify that the above equation would hold, meaning that $\mathbf{w}$ is not the same across all polynomials.

One might ask: well, situation when $L_i \equiv R_i$ is very rare! Indeed, but there also might be situations where $L_i \equiv 5R_i$, $R_i \equiv 100O_i$, or $L_i \equiv 235O_i$ — all of them would lead to the same issue. One easy fix is to make β different for each polynomial $L(X)$, $R(X)$, $O(X)$. What is the solution?

Introducing separate β_L, β_R, β_O. Our proposal is to make β different for each polynomial $L(X)$, $R(X)$, $O(X)$, making it much harder for the adversary to find inconsistent witnesses. That being said, during the setup phase, we choose arbitrary $\beta_L, \beta_R, \beta_O \xleftarrow{R} \mathbb{F}$ and publish $\{g^{\beta_L L_i(\tau) + \beta_R R_i(\tau) + \gamma_O O_i(\tau)}\}_{i \in [n]}$ as a part of the prover parameters pp. Then, the prover $\mathcal{P}$ calculates the following additional commitment:

$$\pi_\beta \leftarrow \prod_{i=1}^{n} (g^{\beta_L L_i(\tau) + \beta_R R_i(\tau) + \beta_O O_i(\tau)})^{w_i}$$

and then publishes π_β as a part of the proof. The verifier $\mathcal{V}$ checks the following condition:

$$e(\pi_L, g^{\beta_L}) \cdot e(\pi_R, g^{\beta_R}) \cdot e(\pi_O, g^{\beta_O}) = e(\pi_\beta, g).$$

Even that is not the end of the story! We also need to make sure that g^{β_L}, g^{β_R}, and g^{β_O} are incompatible with $g^{\sum_{i=0}^{n} (\beta_L L(\tau) + \beta_R R(\tau) + \beta_O O(\tau)) w_i}$ (for reasons we drop here due to already long explanation). For that reason, we multiply β_L, β_R, β_O by some random factor $\gamma \xleftarrow{R} \mathbb{F}$. This way, the proving part remains the same, but the check becomes:

$$e(\pi_L, g^{\gamma\beta_L}) \cdot e(\pi_R, g^{\gamma\beta_R}) \cdot e(\pi_O, g^{\gamma\beta_O}) = e(\pi_\beta, g^{\gamma}).$$

Oof, that was a long fix! Let us come back to the new version of the SNARK protocol (not yet zero-knowledge)!

Attempt #3: Sound SNARK Protocol

Suppose we are given a circuit C with a maximum degree d of polynomials used underneath. Thus, all parties additionally know the target polynomial $Z(X)$.

Setup(1^λ)

The *trusted party* conducts the following steps:

- Picks random values $\tau, \alpha, \beta_L, \beta_R, \beta_O, \gamma \xleftarrow{R} \mathbb{F}$.
- **Outputs** prover parameters pp and verification parameters vp:

$$\begin{aligned}\mathsf{pp} \leftarrow \Big\{ &\{g^{\tau^i}\}_{i\in[d]}, \{g^{L_i(\tau)}, g^{\alpha L_i(\tau)}, g^{R_i(\tau)}, g^{\alpha R_i(\tau)}, \\ &g^{O_i(\tau)}, g^{\alpha O_i(\tau)}, g^{\beta_L L(\tau)+\beta_R R(\tau)+\beta_O O(\tau)}\}_{i\in[n]}\Big\} \\ \mathsf{vp} \leftarrow \Big\{ &g^{Z(\tau)}, g^{\alpha}, g^{\beta_L}, g^{\beta_R}, g^{\beta_O}, g^{\beta_L\gamma}, g^{\beta_R\gamma}, g^{\beta_O\gamma}, g^{\gamma}\Big\}\end{aligned}$$

- **Deletes** aforementioned random scalars (toxic waste).

Prove(pp, $\mathbf{x}$, $\mathbf{w}$)

The prover $\mathcal{P}$ conducts the following steps:

- Runs the circuit to get $\mathbf{w}$ and $L(X), R(X), O(X)$.
- Calculates $H(X) \leftarrow (L(X)R(X) - O(X))/Z(X)$.
- Calculates the sound KZG commitments as follows:

$$\begin{aligned} \pi_L &\leftarrow g^{L(\tau)}, \quad & \pi'_L &\leftarrow g^{\alpha L(\tau)}, \\ \pi_R &\leftarrow g^{R(\tau)}, \quad & \pi'_R &\leftarrow g^{\alpha R(\tau)}, \\ \pi_O &\leftarrow g^{O(\tau)}, \quad & \pi'_O &\leftarrow g^{\alpha O(\tau)}, \\ \pi_H &\leftarrow g^{H(\tau)}, \quad & \pi'_H &\leftarrow g^{\alpha H(\tau)}. \end{aligned}$$

- Calculates the additional commitment π_β as follows:

$$\pi_\beta \leftarrow g^{\beta_L L(\tau)+\beta_R R(\tau)+\beta_O O(\tau)}$$

- Publishes $\boldsymbol{\pi} = (\pi_L, \pi'_L, \pi_R, \pi'_R, \pi_O, \pi'_O, \pi_H, \pi'_H, \pi_\beta)$ as a proof.

Verify(vp, $\mathbf{x}$, $\boldsymbol{\pi}$)

Upon receiving $\boldsymbol{\pi} = (\pi_L, \pi'_L, \pi_R, \pi'_R, \pi_O, \pi'_O, \pi_H, \pi'_H, \pi_\beta)$, the verifier $\mathcal{V}$ checks:

$$\begin{aligned} e(\pi_L, \pi_R) &= e(\mathsf{com}(Z), \pi_H) \cdot e(\pi_O, g), \\ e(\pi_L, g^\alpha) = e(\pi'_L, g), \quad & e(\pi_R, g^\alpha) = e(\pi'_R, g), \\ e(\pi_O, g^\alpha) = e(\pi'_O, g), \quad & e(\pi_H, g^\alpha) = e(\pi'_H, g), \\ e(\pi_L, g^{\gamma\beta_L}) \cdot e(\pi_R, g^{\gamma\beta_R}) &\cdot e(\pi_O, g^{\gamma\beta_O}) = e(\pi_\beta, g^\gamma). \end{aligned}$$

0.1.5 Attempt #4: Splitting the Extended Witness

Now, recall that the actual circuit C is defined for some statement $\mathbf{x}$ and witness $\mathbf{w}$. According to the Circuit Satisfability Problem, the prover $\mathcal{P}$ wants to convince the verifier $\mathcal{V}$ that he knows the witness $\mathbf{w}$ such that the circuit $C(\mathbf{x}, \mathbf{w}) = 0$. Up until now, we have been using the extended witness $\widetilde{\mathbf{w}}$ to represent the trace of computation. However, we can split the witness into two parts: the first part $\mathbf{w}_{\text{mid}}$ — intermediate witness that contains the private witness $\mathbf{w}$ and intermediate variables values, and the second part $\mathbf{w}_{\text{io}}$ — input/output witness that contains public statement information (e.g., $\mathbf{x}$). Suppose for vector $\widetilde{\mathbf{w}} = (\widetilde{w}_1, \ldots, \widetilde{w}_n)$ we pick out the set of indeces $\mathcal{I}_{\text{mid}} \subset [n]$ to represent the intermediate witness and $\mathcal{I}_{\text{io}} \subset [n]$ to represent the input/output witness (of course, $\mathcal{I}_{\text{mid}} \cap \mathcal{I}_{\text{io}} = \emptyset$ and $\mathcal{I}_{\text{mid}} \cup \mathcal{I}_{\text{io}} = [n]$).

Now, how do we split the proofs $\pi_L, \pi_R, \ldots$ into two parts? Well, consider for instance π_L:

$$\pi_L = g^{\sum_{i=0}^{n} w_i L_i(\tau)}.$$

We split this expression as follows:

$$\pi_L = \underbrace{g^{\sum_{i \in \mathcal{I}_{\text{mid}}} w_i L_i(\tau)}}_{\text{new } \pi_L} \times \underbrace{g^{\sum_{i \in \mathcal{I}_{\text{io}}} w_i L_i(\tau)}}_{\pi_{L,\text{io}}}.$$

This way, the prover $\mathcal{P}$ first calculates the new commitment π_L using only the intermediate witness $\mathbf{w}_{\text{mid}}$ and then the verifier can compute the "public" portion of the proof $\pi_{L,\text{io}}$ using the input/output witness $\mathbf{w}_{\text{io}}$ (which is typically quite easy to do since $|\mathcal{I}_{\text{io}}|$ is typically much smaller than $|\mathcal{I}_{\text{mid}}|$). The same goes for other parts of the proof $\boldsymbol{\pi}$.

0.1.6 Attempt #5: Making SNARK Zero-Knowledge

0.2 Real Protocols

0.2.1 Pinocchio Protocol

0.2.2 Groth16 Protocol