Pairing-Based SNARKs. Pinocchio And Groth16

October 22, 2024

Distributed Lab

zkdl-camp.github.io

github.com/ZKDL-Camp

Plan

Recap

Recap. R1CS

Each **constraint** in the Rank-1 Constraint System must be in the form:

$$\langle \boldsymbol{a}, \boldsymbol{w} \rangle \times \langle \boldsymbol{b}, \boldsymbol{w} \rangle = \langle \boldsymbol{c}, \boldsymbol{w} \rangle$$

Where $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ is a dot product.

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle := \boldsymbol{u}^\top \boldsymbol{v} = \sum_{i=1}^{n} u_i v_i$$

Thus

$$\left(\sum_{i=1}^{n} a_i w_i \right) \times \left(\sum_{j=1}^{n} b_j w_j \right) = \sum_{k=1}^{n} c_k w_k$$

That is, actually, a quadratic equation with multiple variables.

Recap. R1CS

Consider the simplest program:

```
def example(a: F, b: F, c: F) -> F:
    if a:
        return b * c
    else:
        return b + c
```

Recap. R1CS

$$r = x_1 \times (x_2 \times x_3) + (1 - x_1) \times (x_2 + x_3)$$

Thus, the next constraints can be build:

$$x_1 \times x_1 = x_1 \quad \text{(binary check)} \tag{1}$$
$$x_2 \times x_3 = \text{mult} \tag{2}$$
$$x_1 \times \text{mult} = \text{selectMult} \tag{3}$$
$$(1 - x_1) \times (x_2 + x_3) = r - \text{selectMult} \tag{4}$$

The witness vector: $\boldsymbol{w} = (1, r, x_1, x_2, x_3, \text{mult}, \text{selectMult})$.

The coefficients vectors:

$$\begin{aligned}
\boldsymbol{a}_1 &= (0,0,1,0,0,0,0), & \boldsymbol{b}_1 &= (0,0,1,0,0,0,0), & \boldsymbol{c}_1 &= (0,0,1,0,0,0,0) \\
\boldsymbol{a}_2 &= (0,0,0,1,0,0,0), & \boldsymbol{b}_2 &= (0,0,0,0,1,0,0), & \boldsymbol{c}_2 &= (0,0,0,0,0,1,0) \\
\boldsymbol{a}_3 &= (0,0,1,0,0,0,0), & \boldsymbol{b}_3 &= (0,0,0,0,0,1,0), & \boldsymbol{c}_3 &= (0,0,0,0,0,0,1) \\
\boldsymbol{a}_4 &= (1,0,-1,0,0,0,0), & \boldsymbol{b}_4 &= (0,0,0,1,1,0,0), & \boldsymbol{c}_4 &= (0,1,0,0,0,0,-1)
\end{aligned}$$

Recap. QAP

R1CS provides us with the following constraint vectors:

$$\boldsymbol{a}_1, \boldsymbol{a}_2, \dots, \boldsymbol{a}_m, \quad \boldsymbol{b}_1, \boldsymbol{b}_2, \dots, \boldsymbol{b}_m, \quad \boldsymbol{c}_1, \boldsymbol{c}_2, \dots, \boldsymbol{c}_m,$$

Of course, they form corresponding matrices:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}, \text{ same goes for } B \text{ and } C$$

An example of a single “if“ statement:

$$\begin{aligned} \boldsymbol{a}_1 &= (0,0,1,0,0,0,0) \\ \boldsymbol{a}_2 &= (0,0,0,1,0,0,0) \\ \boldsymbol{a}_3 &= (0,0,1,0,0,0,0) \\ \boldsymbol{a}_4 &= (1,0,-1,0,0,0,0) \end{aligned} \qquad \begin{array}{c|ccccccc} & & & 3 & & & & \\ \hline & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 4 & 1 & 0 & \boxed{-1} & 0 & 0 & 0 & 0 \end{array}$$

Recap. QAP

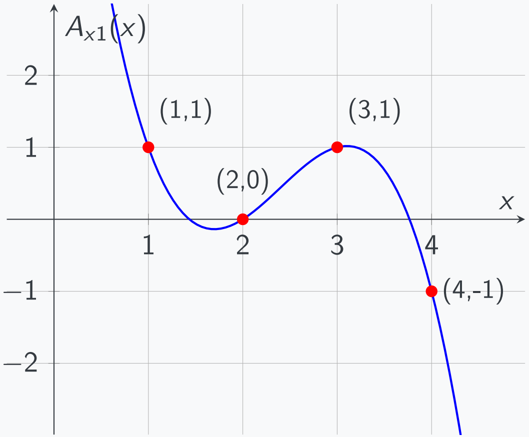


Illustration: The Lagrange inteprolation polynomial for points $\{(1,1),(2,0),(3,1),(4,-1)\}$ visualized over $\mathbb{R}$.

Recap. QAP

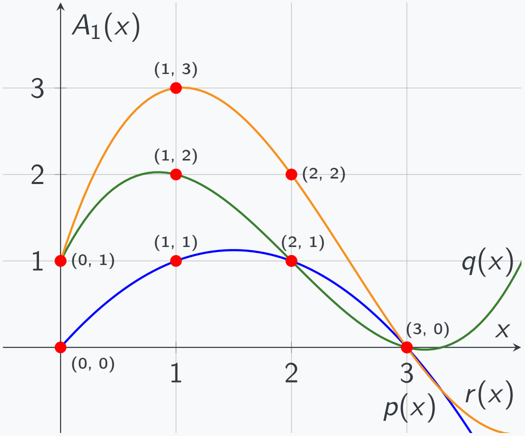


Figure: Addition of two polynomials

Now, using coefficients encoded with polynomials, we can build a constraint number $x \in \{1, \dots, m\}$ in the next way:

$$\begin{aligned}
&(w_1 A_1(x) + w_2 A_2(x) + \dots + w_n A_n(x)) \times \\
\times &(w_1 B_1(x) + w_2 B_2(x) + \dots + w_n B_n(x)) = \\
= &(w_1 C_1(x) + w_2 C_2(x) + \dots + w_n C_n(x))
\end{aligned}$$

Or written more concisely:

$$\left(\sum_{i=1}^{n} w_i A_i(x)\right) \times \left(\sum_{i=1}^{n} w_i B_i(x)\right) = \left(\sum_{i=1}^{n} w_i C_i(x)\right)$$

$$A(x) \times B(x) = C(x)$$

Recap. QAP

Now, we can define a **master polynomial** $M(x)$, that has zeros at all elements from the set $\Omega = \{1, \dots, m\}$

$$M(x) = A(x) \times B(x) - C(x)$$

It means, that $M(x)$ can be divided by **vanishing polynomial** $Z_\Omega(x)$ without a remainder!

$$Z_\Omega(x) = \prod_{i=1}^{m}(x - i), \qquad H(x) = \frac{M(x)}{Z_\Omega(x)} \text{ is a polynomial}$$

Encrypted Verification

Current Point

We've managed to encode into **a single polynomial** an entire computation (a program), of any size, independent of how much data it consumes.

Now, we need to figure our the protocol, how a prover can succinctly proof the knowledge of a correct witness for some circuit to a verifier, additionally, make it zero-knowledge and non-interactive.

Where the knowledge of the correct witness is a knowledge of the quotient polynomial $H(x)$.

$$M(x) = H(x) \times Z_\Omega(x)$$

Remark

Further, for brevity, we will denote $Z_\Omega(x)$ as $Z(x)$.

Notation Preliminaries: Groups

In this section, we will use:

✓ Group of points on elliptic curve denoted as $\mathbb{G}$ of prime order q with a generator g.

✓ The symmetric pairing function $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $(\mathbb{G}_T, \times)$ is a target group (typically, just a scalar from extension $\mathbb{F}_{p^k}$).

Recall

The core property of the pairing function e is the **bilinearity**:

$$e(g^{\alpha}, g^{\beta}) = e(g^{\alpha\beta}, g) = e(g, g^{\alpha\beta}) = e(g, g)^{\alpha\beta}.$$

Here, g^{α} is the same as "scalar multiplication of a generator by a scalar $\alpha \in \mathbb{Z}_q$".

Notation Preliminaries: QAP

Recall that the core equation to be proven:

$$\left(\sum_{i=1}^{n} w_i A_i(x)\right) \times \left(\sum_{i=1}^{n} w_i B_i(x)\right) - \left(\sum_{i=1}^{n} w_i C_i(x)\right) = Z(x)H(x)$$

Here, we will change notation a bit: instead of A and B, we are going to use L and R, while C becomes O.

So equation becomes:

$$\underbrace{\left(\sum_{i=1}^{n} w_i L_i(x)\right)}_{\text{left wires encoding}} \times \underbrace{\left(\sum_{i=1}^{n} w_i R_i(x)\right)}_{\text{right wires encoding}} - \underbrace{\left(\sum_{i=1}^{n} w_i O_i(x)\right)}_{\text{output encodings}} = Z(x)H(x)$$

Naive Proof

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 $Z(x)$ and QAP polynomials $\{L_i(x)\}_{i\in[n]}, \{R_i(x)\}_{i\in[n]}, \{O_i(x)\}_{i\in[n]}$, where n is number of witness elements.

Prover
✓ Provides witness $\boldsymbol{w}$ to a Verifier.

Verifier
✓ Checks $(\sum_{i=1}^{n} w_i L_i(x)) \times (\sum_{i=1}^{n} w_i R_i(x)) = (\sum_{i=1}^{n} w_i O_i(x))$

Naive Proof

- ✗ Succint
- ✓ Non-Interactive
- ✗ Zero-Knowledge

The verifier could actually just run a program that represents a circuit `C` on witness data $\boldsymbol{w}$.

We, definitely, need to encrypt the witness data $\boldsymbol{w}$ somehow...

Let's define the *encryption* operation as follows:

$$\mathsf{Enc} : \mathbb{Z}_q \to \mathbb{G}, \quad \mathsf{Enc}(x) := g^x$$

Essentially, $\mathsf{Enc}(p(\tau))$ is the **KZG Commitment** $\mathsf{com}(p)$.

Example

Consider the polynomial: $p(x) = x^2 - 5x + 2$, the encryption of $p(\tau)$ can be found as follows:

$$\mathsf{Enc}(p(\tau)) = g^{p(\tau)} = g^{\left(\tau^2 - 5\tau + 2\right)} = \left(g^{\tau^2}\right)^1 \cdot \left(g^{\tau^1}\right)^{-5} \cdot \left(g^{\tau^0}\right)^2$$

Question

KZG Commitment requires encrypted powers of τ: $\{g^{\tau^i}\}_{i \in [d]}$. But where the prover can take them?

Trusted Setup

Trusted Party Setup

✓ 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 $\mathsf{vp} \leftarrow \mathsf{com}(Z)$.

This way, we can find the KZG commitment for each polynomial. For example:

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

Now, we can calculate the following KZG commitments (or, synonymously, encryptions):

$$g^{L(\tau)}, g^{R(\tau)}, g^{O(\tau)}, g^{H(\tau)}, g^{Z(\tau)}$$

But how can we verify $H(x)Z(x) = L(x)R(x) - O(x)$ in the encrypted space?

Well, first notice that, according to the **Schwarz-Zippel Lemma**, *with overwhelming probability* the check is equivalent to:

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

So, we can check this equality as follows:

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

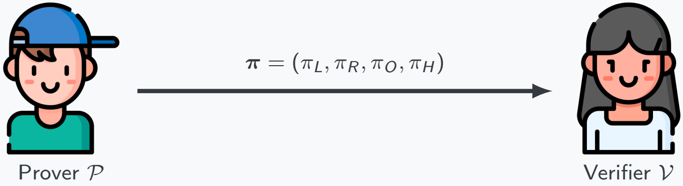
Trusted Setup: $\tau \xleftarrow{R} \mathbb{F}$, $\{g^{\tau^i}\}_{i\in[d]}$, $g^{Z(\tau)}$, **delete**(τ).

✓ $H(x) = \frac{L(x)\times R(x) - O(x)}{Z(x)}$.

✓ KZG commitments:

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

✓ $e(\pi_L, \pi_R) \stackrel{?}{=} e(\mathrm{com}(Z), \pi_H) \cdot e(\pi_O, g)$.



- ✓ Succint
- ✓ Non-Interactive
- ✓ Zero-Knowledge

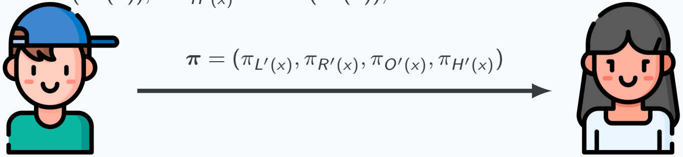
- ✗ Does it work?

Why it doesn't work??

Trusted Setup: $\tau \xleftarrow{R} \mathbb{F}$, $\{g^{\tau^i}\}_{i\in[d]}$, $g^{Z(\tau)}$, **delete**(τ).

- ✓ $H'(x) \xleftarrow{R} \mathbb{F}[x]$, $M'(x) = Z(x) \times H'(x)$.
- ✓ Finds $L'(x), R'(x), O'(x)$ such that: $L'(x) \times R'(x) - O'(x) = M'(x)$
- ✓ KZG commitments:
  $\pi_{L'(x)} \leftarrow \text{com}(L'(x))$, $\pi_{R'(x)} \leftarrow \text{com}(R'(x))$,
  $\pi_{O'(x)} \leftarrow \text{com}(O'(x))$, $\pi_{H'(x)} \leftarrow \text{com}(H'(x))$,

✓ $e(\pi_{L'(x)}, \pi_{R'(x)}) \overset{?}{=} e(\text{com}(Z), \pi_H) \cdot e(\pi_{O'(x)}, g)$.


$\boldsymbol{\pi} = (\pi_{L'(x)}, \pi_{R'(x)}, \pi_{O'(x)}, \pi_{H'(x)})$

Prover $\mathcal{P}$ Verifier $\mathcal{V}$

Problem

Prover isn't forced to use the values from the trusted setup.

Proof Of Exponent

Trusted Setup: $\tau, \alpha \xleftarrow{R} \mathbb{F}, \quad \{\{g^{\tau^i}, g^{\alpha\tau^i}\}_{i\in[d]}\}, \quad \{g^{Z(\tau)}, g^{\alpha}\}, \quad$ **delete**(τ, α).

✓ $H(x) = \frac{L(x)\times R(x) - O(x)}{Z(x)}$.

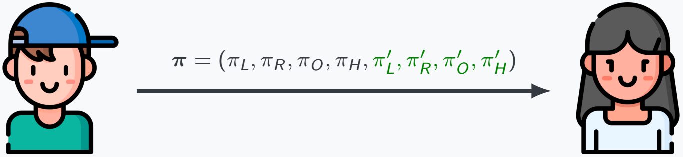
✓ KZG commitments:

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

✓ $e(\pi_L, \pi_R) \overset{?}{=} e(\text{com}(Z), \pi_H) \cdot e(\pi_O, g)$.

✓ Proof of Exponent:

$$e(\pi_L, g^{\alpha}) = e(\pi'_L, g),$$
$$e(\pi_R, g^{\alpha}) = e(\pi'_R, g),$$
$$e(\pi_O, g^{\alpha}) = e(\pi'_O, g),$$
$$e(\pi_H, g^{\alpha}) = e(\pi'_H, g).$$


$\boldsymbol{\pi} = (\pi_L, \pi_R, \pi_O, \pi_H, \pi'_L, \pi'_R, \pi'_O, \pi'_H)$
Prover $\mathcal{P}$
Verifier $\mathcal{V}$

Including PoE

- ✓ Succint
- ✓ Non-Interactive
- ✓ Zero-Knowledge
- ✗ Sound

Problem

There is no guarantee that the same witness w was used to calculate all the commitments $\pi_L, \pi_R, \pi_O, \pi_H$.

Make It Sound

Additional Optimization

Recal that:

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

Here public data is:

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

Moreover, it's defined only by the circuit and trusted setup, thus, it can calculated before proof generation as a part of the trusted setup.

Additional Optimization

Updated Trusted Setup:

$$\begin{aligned}
&\{g^{\tau^i}\}_{i\in[d]}, && \{g^{\alpha\tau^i}\}_{i\in[d]},\\
&\{g^{L_i(\tau)}\}_{i\in[n]}, && \{g^{\alpha L_i(\tau)}\}_{i\in[n]},\\
&\{g^{R_i(\tau)}\}_{i\in[n]}, && \{g^{\alpha R_i(\tau)}\}_{i\in[n]},\\
&\{g^{O_i(\tau)}\}_{i\in[n]}, && \{g^{\alpha O_i(\tau)}\}_{i\in[n]}
\end{aligned}$$

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:

$$\pi_L \triangleq g^{L(\tau)} = g^{\sum_{i=0}^{n} w_i L_i(\tau)} = \prod_{i=0}^{n} (g^{L_i(\tau)})^{w_i},$$

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

Witness Consistency Check

To prove that the same 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 w.

We can do so by proving the next simple statement:

$$g^{L(\tau)+R(\tau)+O(\tau)} = \prod_{i=1}^{n} \left(g^{L_i(\tau)+R_i(\tau)+O_i(\tau)}\right)^{w_i}$$

And we already know how to do that — POE!

Witness Consistency Check

Let's introduce one more coefficient...

$$\beta \xleftarrow{R} \mathbb{F}$$

Extended trusted setup contains additional values:

$$g^{\beta}, \quad \{g^{\beta(L_i(\tau)+R_i(\tau)+O_i(\tau))}\}_{i\in[n]}$$

Prover needs to calculate π_β:

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

And easy check for verifier:

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

One more time... that doesn't work

If the witness is consistent, the following condition must hold:

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

But, what if $L_i \equiv R_i$. Let's call them q, thus:

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

The adversary can choose $w_{L,i}$, $w_{R,i}$ and $w_{O,i}$ such that:

$$w_i := w_{O,i} \quad \text{and} \quad w_{L,i} = 2w_{O,i} - w_{R,i}$$

Example

$$w = w_O = 5, \quad w_L = 7, \quad w_R = 3$$

$$(7+3)q + 5O(\tau) = 5(2q + O(\tau))$$

$$10q + 5O(\tau) = 10q + 5O(\tau)$$

More coefficients!

The main problem is that if $R_i \equiv L_i$ then $\beta R_i \equiv \beta L_i$.

Let's fix that by introducing a separate β coefficients for L, R and O.

$$(\beta_L, \beta_R, \beta_O) \xleftarrow{R} \mathbb{F}^3$$

Therefore, $\beta_R R_i \neq \beta_L L_i$ even if $R_i \equiv L_i$, so the previous hack doesn't work.

So, finally, the trusted setup is updated with:

$$g^{\beta_L}, g^{\beta_R}, g^{\beta_O}, \{g^{(\beta_L L_i(\tau) + \beta_R R_i(\tau) + \beta_O O_i(\tau))}\}_{i \in [n]}$$

Verification:

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

Even more coefficients!

As the adversary has an access to the public g^{β_L}, g^{β_R}, g^{β_O} he still can cheat verifier by calculating modified π_β.

Example

Consider a constraint $w_1 \times w_1 = w_2$. Let's try to assign 2 and 5 for w_1 in a single constraint. As $2 \times 5 = 10$, the w_2 should contains value 10.

$$\mathrm{w} = (w_1, w_2) = (2, 10)$$

The next QAP can be built:

$$L(x) = 2L_1(x) + 10L_2(x)$$
$$R(x) = 2R_1(x) + 3 + 10R_2(x)$$
$$O(x) = 2O_1(x) + 10O_2(x)$$

Compute π_β as:

$$(g^{(\beta_L \mathbf{L_1}(\tau) + \beta_R \mathbf{R_1}(\tau) + \beta_O \mathbf{O_1}(\tau))})^2 \cdot (g^{\beta_R})^3 \cdot (g^{(\beta_L \mathbf{L_2}(\tau) + \beta_R \mathbf{R_2}(\tau) + \beta_O \mathbf{O_2}(\tau))})^{10}$$

Even more coefficients!

To prevent this, let's introduce... one more coefficient!

$$\gamma \xleftarrow{R} \mathbb{F}$$

So, finally... the trusted setup is updated with:

$$g^{\gamma}, g^{\beta_L}, g^{\beta_R}, g^{\beta_O}, \{g^{(\beta_L L_i(\tau)+\beta_R R_i(\tau)+\beta_O O_i(\tau))}\}_{i\in[n]}$$

Proving process isn't changed, unlike verification:

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

That makes it unfeasible to cheat.

Sound SNARK Protocol

Trusted Setup:

$\tau, \alpha, \beta_L, \beta_R, \beta_O, \gamma \xleftarrow{R} \mathbb{F}$, $\{\{g^{\tau^i}, g^{\alpha\tau^i}\}_{i\in[d]}$, $\{g^{(\beta_L L_i(\tau)+\beta_R R_i(\tau)+\beta_O O_i(\tau))}\}_{i\in[n]}\}$,
$\{g^{Z(\tau)}, g^{\alpha}, g^{\beta_L}, g^{\beta_R}, g^{\beta_O}, g^{\beta_L\gamma}, g^{\beta_R\gamma}, g^{\beta_O\gamma}, g^{\gamma}\}$, **delete**$(\tau, \alpha, \beta_L, \beta_R, \beta_O, \gamma)$.

- ✓ $H(x) = \frac{L(x)\times R(x) - O(x)}{Z(x)}$.
- ✓ KZG commitments:

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

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

- ✓ $e(\pi_L, \pi_R) \overset{?}{=}$ $e(\mathsf{com}(Z), \pi_H)\cdot e(\pi_O, g)$.
- ✓ Proof of Exponent:

$$\begin{aligned} e(\pi_L, g^{\alpha}) &= e(\pi'_L, g), \\ e(\pi_R, g^{\alpha}) &= e(\pi'_R, g), \\ e(\pi_O, g^{\alpha}) &= e(\pi'_O, g), \\ e(\pi_H, g^{\alpha}) &= e(\pi'_H, g). \end{aligned}$$

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

Prover $\mathcal{P}$

$\boldsymbol{\pi} = (\pi_L, \pi_R, \pi_O, \pi_H, \pi'_L, \pi'_R, \pi'_O, \pi'_H, \pi_\beta)$ $\longrightarrow$

Verifier $\mathcal{V}$

Make it Zero-Knowledge

Why we need modifications?

Question

- According to the security assumptions, given any element of our proof $\boldsymbol{\pi}$, it is impossible to restore the witness $\boldsymbol{w}$.
- Well, that is true, but does it guarantee that no other information about $\boldsymbol{w}$ is **not leaked**?

It does not! For example, given $\boldsymbol{\pi}$, I can check if $L(x) = 10R(x)$. How? Simply check

$$\pi_L \overset{?}{=} \pi_R^{10}.$$

This works since $\pi_L = g^{L(\tau)}$ and $\pi_R = g^{R(\tau)}$, so

$$\pi_L = \pi_R^{10} \Leftrightarrow g^{L(\tau)} = g^{10R(\tau)} \Leftrightarrow L(\tau) = 10R(\tau) \Leftrightarrow L(x) = 10R(x)$$

How do we fix that? Analysis

Analysis

- Currently, proof consists of four ingredients: $\pi_L, \pi_R, \pi_O, \pi_H$, and their corresponding PoEs.
- But, current construction creates some "connection" between these ingredients. For example, $\pi_L^{101} = \pi_R$ reveals whether $101L(x) = R(x)$ or $\pi_L\pi_O = \pi_R$ whether $L(x) + O(x) = R(x)$.
- Therefore, the prover must add some "noise" to the proof, so that the verifier can't use the proof to extract any information about the witness. The randomness of prover is kept secret.
- Simultaneously, this noise would still make the same proof checkable by already defined verification equations.

Question

So how to we represent such "noise" with preserving "homomorphic" properties of the proof?

How do we fix that? Doing stuff

Idea #1

Let prover pick random values $\delta_R, \delta_O, \delta_L, \delta_H \xleftarrow{R} \mathbb{F}$ and calculate the "distorted" values:

$$\pi_L \leftarrow g^{L(\tau)+\delta_L}, \quad \pi_R \leftarrow g^{R(\tau)+\delta_R}, \quad \text{same for } \pi_O, \pi_H$$

Problem (informally): To make previous verification mechanism work, δ_H must be picked "carefully". However, it's impossible to do so with this construction.

Idea #2

Let prover pick random values $\delta_R, \delta_O, \delta_L \xleftarrow{R} \mathbb{F}$ and calculate:

$$\pi_L \leftarrow g^{L(\tau)+\delta_L Z(\tau)}, \quad \pi_R \leftarrow g^{R(\tau)+\delta_R Z(\tau)}, \quad \pi_O \leftarrow g^{O(\tau)+\delta_O Z(\tau)}$$

Making Idea Practical

This expression can be easily evaluated. For example,

$$\pi_L = g^{L(\tau)} \cdot \left(g^{Z(\tau)}\right)^{\delta_L} = \mathsf{com}(L) \cdot \mathsf{com}(Z)^{\delta_L}$$

Problem

However, the verifier can't check the equation $e(\pi_L, \pi_R) = e(\mathsf{com}(Z), \pi_H) \cdot e(\pi_O, g)$ anymore.

Idea #3

Our "noise" must be controlled! This can be achieved by calculating π_H as $g^{H(\tau)+\Delta_H}$ where Δ_H depends on $\delta_L, \delta_R, \delta_O$ (and possibly other public parameters).

Making Idea Practical

Check $e(\pi_L, \pi_R) = e(\text{com}(Z), \pi_H) \cdot e(\pi_O, g)$ is now equivalent to:

$$(L(x)+\delta_L Z(x))(R(x)+\delta_R Z(x)) = (H(x)+\Delta_H)Z(x)+(O(x)+\delta_O Z(x)),$$

which, by expanding, gives us the following equation:

$$\begin{aligned} \cancel{L(x)R(x)} + \delta_R L(x)Z(x) + \delta_L Z(x)R(x) + \delta_L\delta_R Z(x)^2 \\ = \cancel{H(x)Z(x) + O(x)} + \Delta_H Z(x) + \delta_O Z(x) \end{aligned}$$

where we can cancel out $L(x)R(x)$ and $H(x)Z(x) + O(x)$ terms since they are equal based on initial construction. This way, we get the following expression for Δ_H:

$$\boxed{\Delta_H = \delta_O + \delta_R L(x) + \delta_L R(x) + \delta_L\delta_R Z(x)}$$

Witness consistency proof

Finally, let us not forget about π_β! Previously, we had:

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

Now, we change:

- $L(\tau) \mapsto L(\tau) + \delta_L Z(\tau)$
- $R(\tau) \mapsto R(\tau) + \delta_R Z(\tau)$
- $O(\tau) \mapsto O(\tau) + \delta_O Z(\tau)$

Therefore, our new π_β becomes:

$$\pi_\beta = \left(g^{\beta_L Z(\tau)}\right)^{\delta_L} \left(g^{\beta_R Z(\tau)}\right)^{\delta_R} \left(g^{\beta_O Z(\tau)}\right)^{\delta_O} g^{\beta_L L(\tau)+\beta_R R(\tau)+\beta_O O(\tau)}$$

Overall protocol

Trusted Setup:

$$\tau, \alpha, \beta_L, \beta_R, \beta_O, \gamma \xleftarrow{R} \mathbb{F}, \quad \{\{g^{\tau^i}, g^{\alpha\tau^i}\}_{i\in[d]}, \quad \{g^{\beta_L L_i(\tau)}, g^{\beta_R R_i(\tau)}, g^{\beta_O O_i(\tau)}\}_{i\in[n]}\},$$
$$\{g^{Z(\tau)}, g^{\alpha}, g^{\beta_L}, g^{\beta_R}, g^{\beta_O}, g^{\beta_L\gamma}, g^{\beta_R\gamma}, g^{\beta_O\gamma}, g^{\gamma}\}, \quad \textbf{delete}(\tau, \alpha, \beta_L, \beta_R, \beta_O, \gamma).$$

✓ $H(x) = \frac{L(x)\times R(x) - O(x)}{Z(x)}$.

✓ Sample $\delta_L, \delta_R, \delta_O \xleftarrow{R} \mathbb{F}$, compute:

$$\begin{aligned} \pi_L &\leftarrow g^{L(\tau)}(g^{Z(\tau)})^{\delta_L}, \pi'_L \leftarrow g^{\alpha L(\tau)}(g^{\alpha Z(\tau)})^{\delta_L}, \\ \pi_R &\leftarrow g^{R(\tau)}(g^{Z(\tau)})^{\delta_R}, \pi'_R \leftarrow g^{\alpha R(\tau)}(g^{\alpha Z(\tau)})^{\delta_R}, \\ \pi_O &\leftarrow g^{O(\tau)}(g^{Z(\tau)})^{\delta_O}, \pi'_O \leftarrow g^{\alpha O(\tau)}(g^{\alpha Z(\tau)})^{\delta_O}, \\ \pi_H &\leftarrow g^{H(\tau)}(g^{\delta_O})(g^{R(\tau)})^{\delta_L}(g^{L(\tau)})^{\delta_R}(g^{Z(\tau)})^{\delta_L\delta_R} \\ \pi_\beta &= \dots \end{aligned}$$

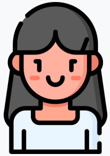
✓ $e(\pi_L, \pi_R) \overset{?}{=\!=} e(\mathbf{com}(Z), \pi_H) \cdot e(\pi_O, g)$.

✓ Proof of Exponent:

$$\begin{aligned} e(\pi_L, g^{\alpha}) &= e(\pi'_L, g), \\ e(\pi_R, g^{\alpha}) &= e(\pi'_R, g), \\ e(\pi_O, g^{\alpha}) &= e(\pi'_O, g), \\ e(\pi_H, g^{\alpha}) &= e(\pi'_H, g). \end{aligned}$$

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

$$\boldsymbol{\pi} = (\pi_L, \pi_R, \pi_O, \pi_H, \pi'_L, \pi'_R, \pi'_O, \pi'_H, \pi_\beta) \longrightarrow$$



Prover $\mathcal{P}$ Verifier $\mathcal{V}$

Real Protocols

Complexity of the Basic Protocol

Overall Complexity

Suppose circuit consists of n gates. Then, the complexity of the basic protocol is as follows:

- **Proof Size:** $O(1)$ — constant number of group elements.
- **Setup Time:** $O(n)$ — calculating powers of τ, evaluations at τ.
- **Prover Time:** $O(n \log n)$ — using FFT and wise choice of Ω.
- **Verifier Time:** $O(1)$ — constant number of pairings.

However, $O(1)$ is not very descriptive for proof and verifier complexities, so let us provide a more detailed analysis.

- **Proof Size:** $9\ \mathbb{G}$ group elements.
- **Verifier Time:** 15 pairings.

We can do better!

Pinocchio Protocol

Idea

In toxic waste, include $\rho_L, \rho_R \xleftarrow{R} \mathbb{F}$, set $\rho_O \leftarrow \rho_L \rho_R$, and define the following generators:

$$g_L \leftarrow g^{\rho_L}, \quad g_R \leftarrow g^{\rho_R}, \quad g_O \leftarrow g^{\rho_O}$$

Reason

Such choice of generators reduce 15 pairings to **11 pairings**. Additionally, we have only **8 group elements** in the proof.

Groth16 Protocol

Idea: Generic Group Model

Use **Generic Group Model** (GGM) technique. Simply put, GGM allows the adversary to only make **oracle requests** to compute the group operations. For example, having a set $\{g^{\alpha R_i(\tau)}\}_{i \in [d]}$, adversary can compute only linear combinations of these values. In the particular case of Groth16, instead of considering $L_i(x)$, $R_i(x)$, and $O_i(x)$ separately, we construct their linear combinations as $Q_i(x) := \beta L_i(x) + \alpha R_i(x) + O_i(x)$, where α and β are toxic parameters.

Groth16 Protocol: Setup Procedure

The proving key is formed as follows:

$$\mathsf{pp} \leftarrow \left(g_1^{\alpha}, g_1^{\beta}, g_1^{\delta}, \left\{g_1^{\tau^i}, \frac{\beta L_i(\tau) + \alpha R_i(\tau) + O_i(\tau)}{\gamma}, \frac{\tau^i Z(\tau)}{\delta}\right\}_{i \in [n]}, \right.$$
$$\left. g_2^{\beta}, g_2^{\delta}, g_2^{\gamma}, \{g_2^{\tau^i}\}_{i \in [d]}\right)$$

Groth16 Protocol: Proving Procedure

Sample random $\delta_L, \delta_R \xleftarrow{R} \mathbb{F}$ and compute the following values:

$$\pi_L \leftarrow g_1^{\alpha+\sum_{i=1}^n w_i L_i(\tau)+\delta_L\delta}, \quad \pi_R \leftarrow g_2^{\beta+\sum_{i=0}^n w_i R_i(\tau)+\delta_R\delta},$$
$$\pi_O \leftarrow g_1^{\frac{Q_{\text{mid}}(\tau)+H(\tau)Z(\tau)}{\delta}+L\delta_R+R\delta_L-\delta_L\delta_R\delta},$$

where by Q_{mid} we denoted the following expression:

$$Q_{\text{mid}}(\tau) = \sum_{i\in\mathcal{I}_{\text{mid}}} w_i Q_i(\tau) = \sum_{i\in\mathcal{I}_{\text{mid}}} w_i(\beta L_i(\tau) + \alpha R_i(\tau) + O_i(\tau))$$

Groth16 Protocol: Verification Procedure

The verifier first calculates the following value:

$$\pi_{\text{io}} \leftarrow g_1^{\sum_{i \in \mathcal{I}_{\text{io}}} w_i(\beta L_i(\tau)+\alpha R_i(\tau)+O_i(\tau))/\gamma},$$

and then checks the following single condition:

$$e(\pi_L, \pi_R) = e(g_1^{\alpha}, g_2^{\beta}) e(\pi_{\text{io}}, g_2^{\gamma}) e(\pi_O, g_2^{\delta})$$

Note

$e(g_1^{\alpha}, g_2^{\beta})$ can be additionally hard-coded in the verifier, thus reducing the number of pairings to 3. Finally, the proof's size is now reduced to 3 group elements: two from $\mathbb{G}_1$, and one from $\mathbb{G}_2$.

Thank you for your attention

♥

zkdl-camp.github.io

github.com/ZKDL-Camp