UltraGroth. Lookup Checks Enabled in Groth16

September 4, 2025

Distributed Lab

zkdl-camp.github.io
github.com/ZKDL-Camp

Introduction

Motivation

We typically want to check **inclusion** $\{z_i\}_{i\in[n]} \subseteq \{t_j\}_{j\in[v]}$, where $\{z_i\}_{i\in[n]}$ is part of the witness while $\{t_j\}_{j\in[v]}$ is the lookup table.

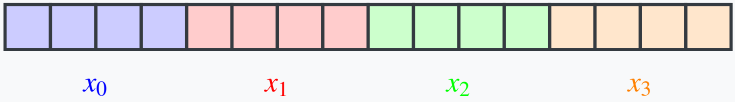
Example usage: effective range-checks (Bionetta, Rarimo circuits, non-native ZK verifications etc.)

$x =$

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}

16 constraints

$x =$



4 constraints + one-time 2^4 commitment

Generally: for n-bit range-check, the circuit's complexity reduces from $\mathcal{O}(n)$ to $\mathcal{O}(2^w + \frac{n}{w})$, which yields $\mathcal{O}(n/\log n)$ assymptotic.

Logup Check

Theorem (Some stuff from ZKDL Camp)

The inclusion check $\{z_i\}_{i \in [n]} \subseteq \{t_i\}_{i \in [v]}$ *is satisfied if and only if there exists the set of multiplicities* $\{\mu_i\}_{i \in [v]}$ *where* $\mu_i = \#\{j \in [n] : z_j = t_i\}$ *such that for* $\gamma \leftarrow\$ \mathbb{F}$*:*

$$\sum_{i \in [n]} \frac{1}{\gamma + z_i} = \sum_{i \in [v]} \frac{\mu_i}{\gamma + t_i}$$

Naive approach: Define `signal gamma` and implement this check in-circuit. This costs exactly $n + 2v$ constraints.

Problem: We *cannot define random* signals in Circom since Groth16, compared to $\mathcal{P}\text{lon}\mathcal{K}$ or sumcheck-based approaches, is not compiled from interactive protocol using Fiat-Shamir heuristic.

Other implications

UltraGroth, though, is not about lookup checks only.

Assume that you need to implement multiplication of two matrices $A, B \in \mathbb{F}^{n \times n}$. **Naive way** is to compute $C = AB$ by definition:

$$C_{i,j} = \sum_{k=1}^{n} A_{i,k} B_{k,j} \qquad \texttt{// Costs } n \texttt{ constraints per } C_{i,j}$$

As we have n^2 elements in C, we thus need n^3 constraints.

We can instead apply the **Freiveld's protocol**. Sample random $\gamma \leftarrow\$ \mathbb{F}^n$, compute C off-circuit and then verify:

$$AB\gamma = C\gamma \qquad \texttt{// Costs } 3n^2 \texttt{ constraints}$$

Example: Attention layer implementation in zkML.

Plan

1. We recap the Groth16 construction.
2. We identify how to make it interactive.
3. We specify how Fiat-Shamir transformation should be applied.
4. We show how it can be practically implemented.

Groth16 Recap

R1CS

Recall that we can encode any NP statement in the form of m equations of form $\langle \boldsymbol{\ell}_j, \mathbf{z} \rangle \cdot \langle \boldsymbol{r}_j, \mathbf{z} \rangle = \langle \boldsymbol{o}_j, \mathbf{z} \rangle$ for $j \in [m]$ and $\mathbf{z} \in \mathbb{F}^n$. Such way of representing the statement is called **R1CS arithmetization**.

This equality is rewritten more succinctly in the matrix form:

$$L\mathbf{z} \odot R\mathbf{z} = O\mathbf{z}$$

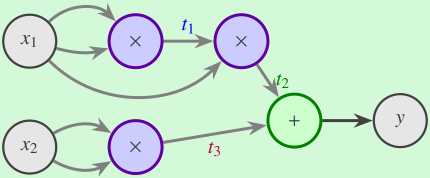
✓ As of now, this is one of the most optimal arithmetization systems available (compared to PlonK and AIR).

✓ All linear operations over elements of $\mathbf{z}$ cost **0 constraints**, compared to PlonK.

Example

Suppose the program computes the expression $y = x_1^3 + x_2^2$.

Circuit Diagram



Constraints

$$t_1 = x_1 \cdot x_1$$
$$t_2 = t_1 \cdot x_1$$
$$t_3 = x_2 \cdot x_2$$
$$y = t_2 + t_3$$

In this case, the witness looks as $z = (1, x_1, x_2, t_1, t_2, t_3, y)$, and (for simplicity, consider only L, R):

$$L = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}, \quad R = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

QAP

We interpolate the columns of each of the matrices, thus getting $3n$ polynomials $\{(\ell_i(X), r_i(X), o_i(X))\}_{i\in[n]} \subseteq \mathbb{F}^{\leq m}[X]$:

$$\ell_i(\omega^j) = L_{i,j}, \quad r_i(\omega^j) = R_{i,j}, \quad o_i(\omega^j) = O_{i,j}, \quad i \in [n],\ j \in [m]$$

Now, the same R1CS check can be encoded over polynomial space:

$$\sum_{i\in[n]} z_i \ell_i(X) \cdot \sum_{i\in[n]} z_i r_i(X) = \sum_{i\in[n]} z_i o_i(X) + t_\Omega(X)h(X),$$

where $h(X)$ is computed by a prover and $t_\Omega(X) \triangleq \prod_{h\in\Omega}(X - h)$ is the *vanishing polynomial* over *evaluation domain* $\Omega = \{\omega^j\}_{j\in[m]}$. The corresponding relation:

$$\mathcal{R}_{\text{QAP}} = \left\{ \begin{matrix} \mathbb{x} = \{z_i\}_{i\in\mathcal{I}_X} \\ \mathbb{w} = \{z_i\}_{i\in\mathcal{I}_W} \end{matrix} \middle| \begin{matrix} \sum_{i\in[n]} z_i \ell_i(X) \cdot \sum_{i\in[n]} z_i r_i(X) = \sum_{i\in[n]} z_i o_i(X) + t_\Omega(X)h(X) \\ \text{for some } h(X) \in \mathbb{F}[X] \end{matrix} \right\}$$

Linear non-interactive proofs

Recall that Groth16 is compiled from *Linear non-interactive proofs*.

Definition (Linear Non-Interactive Proof)

The **Linear Non-Interactive Proof** consists of the following procedures:

- $\mathsf{Setup}(1^{\lambda}, \mathcal{R}) \to (\sigma, \tau)$. The setup returns $\sigma \in \mathbb{F}^m$ and $\tau \in \mathbb{F}^n$.
- $\mathsf{Prove}(\sigma, \mathbb{x}, \mathbb{w}) \to \pi$. $\mathcal{P}$ chooses the matrix $\Pi \in \mathbb{F}^{k \times m}$ and computes the proof as $\pi \leftarrow \Pi\sigma$.
- $\mathsf{Verify}(\sigma, \mathbb{x}, \pi) \to \{0, 1\}$. The verifier gets the arithmetic circuit $t : \mathbb{F}^{m+k} \to \mathbb{F}^{\eta}$ of degree d and verifies whether $t(\sigma, \pi) = 0$.

Groth16 is essentially a Linear NIP where $d = 2$ and σ is given by:

$$\sigma = \left(\alpha, \beta, \gamma, \delta, \{\tau^i\}_{i \in [n]}, \left\{\frac{\zeta_i(\tau)}{\gamma}\right\}_{i \in [m]}, \left\{\frac{\tau^i t_{\Omega}(\tau)}{\delta}\right\}_{i \in [n]}\right),$$

with $\zeta_i(X) := \beta \ell_i(X) + \alpha r_i(X) + o_i(X)$ and $\tau = (\alpha, \beta, \gamma, \delta, \tau)$.

Groth16 Construction

Fix bilinear group $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ with pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

- Setup$(1^\lambda, \mathcal{R}_{\mathsf{QAP}}) \to (\mathsf{pp}, \mathsf{vp})$. See previous slide.
- Prove$(\mathsf{pp}, \mathbb{x}, \mathbb{w}) \to \pi$. Sample random $r, s \leftarrow\!\$\ \mathbb{F}$ and output $\pi \leftarrow (g_1^{a(\tau)}, g_1^{c(\tau)}, g_2^{b(\tau)})$ where:

$$a(X) = \alpha + \sum_{i\in[n]} z_i \ell_i(X) + r\delta,\ \ b(X) = \beta + \sum_{i\in[n]} z_i r_i(X) + s\delta,$$

$$c(X) = \delta^{-1}\left(\sum_{i\in\mathcal{I}_W} z_i \zeta_i(X) + h(X)t_\Omega(X)\right) + a(X)s + b(X)r - rs\delta$$

- Verify$(\mathsf{vp}, \mathbb{x}, \pi) \to \{0,1\}$. Parse $\pi = (\pi_A, \pi_C, \pi_B)$ and accept the proof if and only if

$$e(\pi_A, \pi_B) = e(g_1^\alpha, g_2^\beta) \cdot e(g_1^{\iota(\tau)}, g_2^\gamma) \cdot e(\pi_C, g_2^\delta),$$

where $\iota(X) := \gamma^{-1} \sum_{i\in\mathcal{I}_X} z_i \zeta_i(X)$ is the input commitment.

UltraGroth

Desired Interactive Protocol

We would like to have the following interactive protocol (IP) between the prover $\mathcal{P}$ and verifier $\mathcal{V}$.

Input: Relation $\mathcal{R}_{\text{QAP}}$ and public statement $\mathbb{x}_0$.

Round 0: $\mathcal{P}$ runs the circuit without imposing lookup check and gets witness $\mathbb{w}_0$. $\mathcal{V}$ sends the random challenge $\mathbb{x}_1 \leftarrow\$ \mathbb{F}$.

Round 1: $\mathcal{P}$ computes the second part of the witness $\mathbb{w}_1$, corresponding to the lookup check $\sum_{i\in[n]} \frac{1}{\mathbb{x}_1+z_i} = \sum_{i\in[v]} \frac{\mu_i}{\mathbb{x}_1+t_i}$. The verifier $\mathcal{V}$ sends $\mathbb{w}_1$ and $h(X)$ to prover.

Check: $\mathcal{V}$ checks $\ell(X)r(X) = o(X) + t_\Omega(X)h(X)$.

Compiling IP into NIZK. Apply Fiat-Shamir transformation: sample challenge as $\mathbb{x}_1 = \mathcal{H}(\sigma, \mathbb{x}_0, \mathbb{w}_0)$.

Problem. We cannot practically "hash" the witness part $\mathbb{w}_0$.

2-round UltraGroth Construction

Split public indexing set $\mathcal{I}_X$ into two parts: $\mathcal{I}_X^{\langle 0 \rangle}$ and $\mathcal{I}_X^{\langle 1 \rangle}$. Similarly, split the witness indexing set $\mathcal{I}_W$ into $\mathcal{I}_W^{\langle 0 \rangle}$ and $\mathcal{I}_W^{\langle 1 \rangle}$.

Input: Relation $\mathcal{R}_{\text{QAP}}$ and public statement $\mathbb{x}_0$.

Round 0: $\mathcal{P}$ runs circuit without lookup check and gets witness $\mathbb{w}_0$. She samples $r_0 \leftarrow\$ \mathbb{F}$, and computes $\pi_C^{\langle 0 \rangle} \leftarrow g_1^{c_0(\tau)}$ as:

$$c_0(X) = \delta_0^{-1} \sum_{j \in \mathcal{I}_W^{\langle 0 \rangle}} z_j \zeta_j(X) + r_0 \delta$$

Round 1: $\mathcal{P}$ samples the challenge $\mathbb{x}_1 \leftarrow \mathcal{H}(\sigma, \pi_C^{\langle 0 \rangle})$, samples $r, s \leftarrow\$ \mathbb{F}$ and computes $\pi_C^{\langle 1 \rangle} \leftarrow g_1^{c_1(\tau)}$ as:

$$c_1(X) = \delta^{-1} \left(\sum_{j \in \mathcal{I}_W^{\langle 1 \rangle}} z_j \zeta_j(X) + h(X) t_\Omega(X) \right) + a(X)s + b(X)r - r_0 \delta_0 - rs\delta$$

2-round UltraGroth Construction: The rest

Then, parts $\pi_A \leftarrow g_1^{a(\tau)}$ and $\pi_B \leftarrow g_1^{b(\tau)}$ are computed as usual via:

$$a(X) = \alpha + \sum_{i \in [n]} z_i \ell_i(X) + r\delta,\ \ b(X) = \beta + \sum_{i \in [n]} z_i r_i(X) + s\delta.$$

Note: $\delta_0 c_0(X) + \delta c_1(X)$ is exactly $\delta c(X)$ is the original Groth16. Thus, $\mathcal{V}$ checks:

$$e(\pi_A, \pi_B) = e(g_1^{\alpha}, g_2^{\beta}) \cdot e(g_1^{\iota(\tau)}, g_2^{\gamma}) \cdot e(\pi_C^{\langle 0 \rangle}, g_2^{\delta_0}) \cdot e(\pi_C^{\langle 1 \rangle}, g_2^{\delta}),$$

where $\iota(X) = \gamma^{-1} \sum_{i \in \mathcal{I}_X} z_i f_i(X)$ as before and $\mathbb{x}_1 = \mathcal{H}(\sigma, \pi_C^{\langle 0 \rangle})$.

Conclusion

UltraGroth protocol's verifier is only **4 pairings**, **1 hashing operation**, and $\mathcal{O}(|\mathbb{x}|)$ exponentiations over $\mathbb{G}_1$.

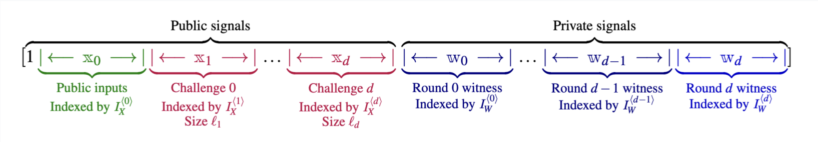
Multi-round UltraGroth

Definition (dQAP)

We define the $(d+1)$**-round quadratic arithmetic program** (or d**QAP**, for short), as follows:

$$\mathcal{R}_{\mathsf{dQAP}} = \left\{ \begin{array}{l} \mathbb{x}_i = \{z_j\}_{j \in \mathcal{I}_X^{\langle i \rangle}} \\ \mathbb{w}_i = \{z_j\}_{j \in \mathcal{I}_W^{\langle i \rangle}} \\ \text{for } i \in [d+1] \end{array} \middle| \begin{array}{c} \ell(X) \cdot r(X) = o(X) + t_\Omega(X) h(X) \\ \ell(X) = \sum_{i \in [n]} z_i \ell_i(X), \\ r(X) = \sum_{i \in [n]} z_i r_i(X), \\ o(X) = \sum_{i \in [n]} z_i o_i(X), \\ \text{for some } h(X) \in \mathbb{F}[X] \end{array} \right\},$$

where $\{\mathcal{I}_X^{\langle i \rangle}\}_{i \in [d+1]}$ and $\{\mathcal{I}_W^{\langle i \rangle}\}_{i \in [d+1]}$ partition $[n]$.



Strategy

Definition (Strategy)

Define **strategy** for $\mathcal{R}_{\text{dQAP}}$ as the collection of functions $S = \{S_i\}_{i \in [d]}$ each of which computes the witness for the given round given previous witnesses and challenges and the current challenge, sampled by the verifier. In other words,

$$\mathbb{w}_i = S_i(\mathbb{x}_0, \dots, \mathbb{x}_i, \mathbb{w}_0, \dots, \mathbb{w}_{i-1})$$

Example

0QAP represents the regular QAP with the strategy $S = \{S_0\}$ that consists of the witness generator: $\mathbb{w} = S_0(\mathbb{x})$. In turn, 1QAP represents the lookup Groth16 version where $\mathbb{w}_0 = S_0(\mathbb{x}_0)$ computes the witness without lookups while $\mathbb{w}_1 = S_1(\mathbb{x}_0, \mathbb{x}_1, \mathbb{w}_0)$ computes lookup constraints.

d-round UltraGroth

Initialize accumulator $a_0 := \mathcal{H}(\sigma)$.

On each round $i \in [d]$:

- Sample $r_i \leftarrow\$ \mathbb{F}$.
- Compute witness $\mathbb{w}_i \leftarrow S_i(\mathbb{x}_0, \ldots, \mathbb{x}_i, \mathbb{w}_0, \ldots, \mathbb{w}_{i-1})$.
- Compute $\pi_C^{\langle i \rangle}$ with $c_i(X) := \delta_i^{-1} \sum_{j \in \mathcal{I}_W^{\langle i \rangle}} z_j \zeta_j(X) + r_i \delta_d$.
- Update accumulator $a_{i+1} \leftarrow \mathcal{H}(a_i, \pi_C^{\langle i \rangle})$.
- If $i < d$, for each $j \in \mathcal{I}_X^{\langle i+1 \rangle}$, set $z_j \leftarrow \mathcal{H}(a_{i+1}, g_1^j)$.

d-round UltraGroth: Last Round

During the last round:

- Compute $h(X)$ similar to Groth16.
- Sample $r, s \leftarrow\$ \mathbb{F}$ and compute $\pi_A \leftarrow g_1^{a(\tau)}$, $\pi_B \leftarrow g_2^{b(\tau)}$, and the last proof piece $\pi_C^{\langle d \rangle} \leftarrow g_1^{c_d(\tau)}$ where:

$$a(X) = \alpha + \sum_{i \in [n]} z_i \ell_i(X) + r\delta_d, \;\; b(X) = \beta + \sum_{i \in [n]} z_i r_i(X) + s\delta_d,$$

$$c_d(X) = \delta_d^{-1} \left(\sum_{i \in \mathcal{I}_W^{\langle d \rangle}} z_i f_i(X) + h(X) t_\Omega(X) \right) + a(X)s + b(X)r - \sum_{i \in [d]} r_i \delta_i - rs\delta_d$$

- **Output** proof $\pi = (\pi_A, \pi_B, \{\pi_C^{\langle i \rangle}\}_{i \in [d+1]}) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1^{d+1}$.

Verification: $e(\pi_A, \pi_B) = e(g_1^{\alpha}, g_2^{\beta}) \cdot e(g_1^{i(\tau)}, g_2^{\gamma}) \cdot \prod_{i \in [d+1]} e(\pi_C^{\langle i \rangle}, g_2^{\delta_i})$.

UltraGroth Efficiency

Groth16 performance over the circuit of size n and statement size ℓ.

- **Prover work**: MSM of size $\mathcal{O}(n)$ over $\mathbb{G}_1$ and $\mathbb{G}_2$.
- **Proof size:** $2\mathbb{G}_1 + \mathbb{G}_2$.
- **Verifier work:** 3 pairings + $\mathcal{O}(\ell)$ $\mathbb{G}_1$ exps.

UltraGroth performance over $\mathcal{R}_{\mathsf{dQAP}}$ in turn:

- **Prover work**: MSM of size $\mathcal{O}(n/\log n)$ over $\mathbb{G}_1$ and $\mathbb{G}_2$.
- **Proof size:** $(d+2)\mathbb{G}_1 + \mathbb{G}_2$.
- **Verifier work:** $(d+3)$ pairings + $\mathcal{O}(\ell)$ $\mathbb{G}_1$ exps + $\sum_{i \in [d+1] \setminus \{0\}} |\mathbb{x}_i|$ hashing operations.
- Allowed interactiveness for potentially more complex protocols.

Any Questions?

♥

zkdl-camp.github.io

github.com/ZKDL-Camp