Bulletproofs: applications

July 31, 2025

Distributed Lab

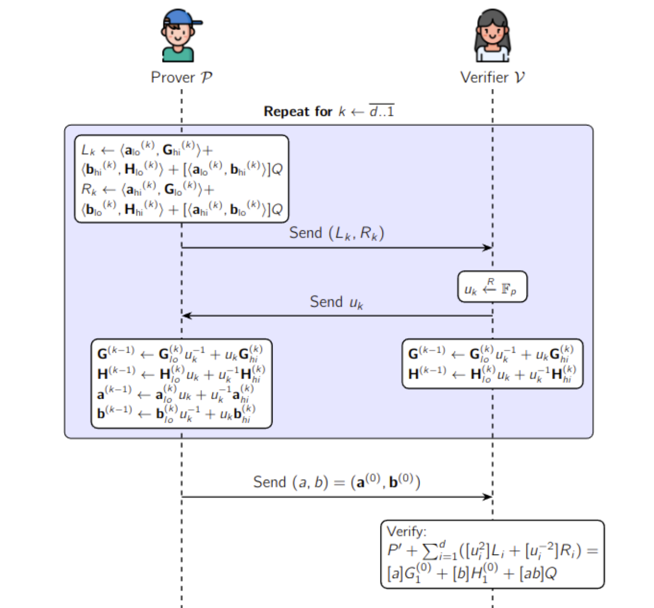
zkdl-camp.github.io

github.com/ZKDL-Camp

Plan

Introduction

Inner-product argument: illustration



Recap: inner-product argument

- **Goal:** Prove $\langle \mathrm{a}, \mathrm{b} \rangle = c$ with logarithmic proof size
- Commitment: $P' = \langle \mathrm{a}, \mathrm{G} \rangle + \langle \mathrm{b}, \mathrm{H} \rangle + [\langle \mathrm{a}, \mathrm{b} \rangle]Q$
- Protocol recursively compresses vectors at each step
- Final check: $P' + \sum([u_i^2]L_i + [u_i^{-2}]R_i) = [a]G + [b]H + [ab]Q$

Key properties

Proof size is $O(\log_2 n)$, prover and verifier both run in $O(n)$. The protocol doesn't need a *trusted setup*. Protocol is *knowledge sound* and *perfect complete* but not *zero-knowledge*.

Idea

We could provide *zero-knowledge* directly to *inner-product argument* construction or use **zk-mul** protocol for outer construction.

Recap: zkmul

Consider relation $R_{mul} = \{(\perp; l(x), r(x), t(x)) | t(x) = l(x)r(x)\}$ where $l(x) = a + s_L x, r(x) = b + s_R x, t(x) = l(x)r(x)$. Protocol **zk-mul** is defined as follows:

- Prover computes and sends to $\mathcal{V}$ commitments to $l(x), r(x), t(x)$:

$$A = [a]G + [b]H + [\alpha]B \qquad T_0 = [ab]G + [\tau_0]B$$
$$S = [s_L]G + [s_R]H + [\beta]B \qquad T_1 = [s_L + s_R]G + [\tau_1]B$$
$$T_2 = [s_L s_R]G + [\tau_2]B$$

- Verifier draws random challenge $u \in \mathbb{F}_p$ and sends it to prover
- Prover evaluates and sends to Verifier $(l_u, r_u, t_u, \alpha_u, \tau_u)$:

$$l_u = l(u), r_u = r(u), t_u = l_u \cdot r_u, \alpha_u = \alpha + \beta u, \tau_u = \tau_0 + \tau_1 u + \tau_2 u^2$$

- Verifier checks: $A + [u]S \stackrel{?}{=} [l_u]G + [r_u]H + [\alpha_u]B$, $[t_u]G + [\tau_u]B \stackrel{?}{=} T_0 + [u]T_1 + [u^2]T_2,\ t_u \stackrel{?}{=} l_u r_u$

What's next?


IPA
zk-mul
Range proofs &
Arithmetic circuits
@Petirep
+ JAKE-CLARK.TUMBLR
imgflip.com

IPA polynomial commitment scheme

Recap: Polynomial commitments

- **Polynomial commitment scheme**

$$\mathcal{C} = (\text{Setup}, \text{Commit}, \text{Open}, \text{VerifyOpen})$$

  allows to commit to a polynomial $f(x) = \sum_{i=0}^{n-1} a_i x^i$ and prove its evaluation at some point.

- **Applications:** SNARKs compiled with IOP + polynomial commitment scheme framework (e.g., Halo, Nova, Spartan, Plonk)
- **Desirable properties:** sublinear size, efficient, no trusted setup

Example

One famouos example is the **KZG** polynomial commitment scheme, which uses bilinear pairings and requires a trusted setup.

IPA polynomial commitment

Let $f(x) = \sum_{i=0}^{n-1} a_i x^i$ be a polynomial of degree $n - 1 = 2^d - 1$.

The **non-hiding IPA polynomial commitment scheme** $\mathcal{C}_{ip} = (\text{Setup}, \text{Commit}, \text{Open}, \text{VerifyOpen})$ is defined as follows:

- Setup returns independent generators $\mathrm{G} = (G_1, \ldots, G_n)$.
- Commit returns $\text{Com}(f) = \langle \mathrm{f}, \mathrm{G} \rangle$ where $\mathrm{f} = (a_0, \ldots, a_{n-1})$
- Open given evaluation point $u \in \mathbb{F}_p$ computes $\mathrm{u^n} = (1, u, u^2, \ldots, u^{n-1})$, obtains $f(u) = \langle \mathrm{f}, \mathrm{u^n} \rangle$ and runs *inner-product argument* Π_{ip} non-interactively setting

$$\mathrm{a} = \mathrm{f}, \mathrm{b} = \mathrm{u^n}, P = \text{Com}(f), c = f(u)$$

  to produce an evaluation proof π_{ip}
- VerifyOpen given evaluation point $u \in \mathbb{F}_p$ and commitment $\text{Com}(f)$ validates proof π_{ip} running the non-interactive verifier $\mathcal{V}$ of *inner-product* argument.

Range proofs

Range proofs: motivation

- **Goal:** Prove $v \in [0, 2^n)$ without revealing v:

$$\mathcal{R}_{rp} = \{(G, B, V, n; v, \gamma) | V = [v]G + [\gamma]B, v \in [0, 2^n)\}$$

- **Applications:** Confidential transactions (e.g., Monero, Mimblewimble), other privacy-preserving protocols.
- **Idea**: Prove $v = \sum_{i=0}^{n-1} v_i 2^i$ and $\forall i \in 0..n-1 : v_i \in \{0, 1\}$

Naive approach

One could prove $v = \sum_{i=0}^{n-1} v_i 2^i$ and $\forall i \in 0..n-1 : v_i \in \{0, 1\}$ using Σ-protocols, but this would be inefficient due to linear proof.

Compiling range proof into inner-product

Firstly, write v in base-2 representation: $v = \sum_{i=0}^{\lfloor \log_2 v \rfloor} 2^i v_i$ and $\mathrm{a}_L = (v_0, v_1, \ldots, v_{n-1})$ be the vector of bits padded with zeroes to length n, define $\mathrm{a}_R = \mathrm{a}_L - 1^n$ so the range validation that v lays in $[0, 2^n)$ implies two checks:

- The following inner-product equality holds: $\langle \mathrm{a}_L, 2^n \rangle = v$
- Each bit v_i must be either 0 or 1:

$$\mathrm{a}_L - \mathrm{a}_R - 1^n = 0^n$$
$$\mathrm{a}_L \circ \mathrm{a}_R = 0^n$$

Example

Let $\mathrm{a}_L = (1, 0, 1, 0)$, $\mathrm{a}_R = (0, -1, 0, -1)$, then $\mathrm{a}_L \circ \mathrm{a}_R = (0, 0, 0, 0)$

Compiling range proof into inner-product

This two checks imply verification that some vector is zero vector, for that we use some challenge $y \in \mathbb{F}_p$ and check inner-product equalities

$$\langle \mathrm{a}_L \circ \mathrm{a}_R, \mathrm{y}^n \rangle = 0 \text{ and } \langle \mathrm{a}_L - \mathrm{a}_R - 1^n, \mathrm{y}^n \rangle = 0$$

This checks are sound because the prover doesn't know challenge y in advance. So we must combine three inner-product checks:

1. $\langle \mathrm{a}_L, 2^n \rangle = v$
2. $\langle \mathrm{a}_L, \mathrm{a}_R \circ \mathrm{y}^n \rangle = 0$
3. $\langle \mathrm{a}_L - \mathrm{a}_R - 1^n, \mathrm{y}^n \rangle = 0$

into one soundly summing up with powers of other challenge $z \in \mathbb{F}_p$:

$$z^2 \cdot \langle \mathrm{a}_L, 2^n \rangle + z \cdot \langle \mathrm{a}_L - \mathrm{a}_R - 1^n, \mathrm{y}^n \rangle + \langle \mathrm{a}_L, \mathrm{a}_R \circ \mathrm{y}^n \rangle = z^2 v$$

Compiling range proof into inner-product

Using some dark linear algebra wizardry we could combine the three inner-product checks into a single one inner-product check:

$$\langle \mathrm{a}_L - z \cdot 1^n, z^2 \cdot 2^n + z \cdot \mathrm{y}^n + \mathrm{a}_R \circ \mathrm{y}^n \rangle = z^2 v + \delta(y, z)$$

Where $\delta(y, z)$ could easily be computed by verifier:

$$\delta(y, z) = (z - z^2)\langle 1^n, \mathrm{y}^n \rangle - z^3 \langle 1^n, 2^n \rangle$$

Now it's time to bring out **zk-mul** for inner-products!

Firstly, construct the blinding polynomials for a_L and a_R:

$$\mathrm{a}'_L \leftarrow \mathrm{a}_L + \mathrm{s}_L x \quad \mathrm{a}'_R \leftarrow \mathrm{a}_R + \mathrm{s}_R x$$

Compute polynomials $\mathrm{l}(x) = \mathrm{l}_0 + \mathrm{l}_1 x, \quad \mathrm{r}(x) = \mathrm{r}_0 + \mathrm{r}_1 x$:

$$\mathrm{l}(x) = \mathrm{a}'_L - z \cdot 1^n = (\mathrm{a}_L + \mathrm{s}_L x) - z \cdot 1^n = \mathrm{a}_L - z \cdot 1^n + \mathrm{s}_L x$$

$$\mathrm{r}(x) = z^2 \cdot 2^n + z \cdot \mathrm{y}^n + \mathrm{a}'_R \circ \mathrm{y}^n = z^2 \cdot 2^n + z \cdot \mathrm{y}^n + (\mathrm{a}_R + \mathrm{s}_R x) \circ \mathrm{y}^n$$

$$= z^2 \cdot 2^n + z \cdot \mathrm{y}^n + \mathrm{a}_R \circ \mathrm{y}^n + \mathrm{s}_R \circ \mathrm{y}^n x$$

Compiling range proof into inner-product

$$t(x) = \langle \mathrm{l}(x), \mathrm{r}(x) \rangle = t_0 + t_1 x + t_2 x^2$$

Now $\mathcal{P}$ needs to apply **zk-mul** for proving:

$$t_0 = \langle \mathrm{a}_L - z \cdot 1^n, z^2 \cdot 2^n + z \cdot \mathrm{y}^n + \mathrm{a}_R \circ \mathrm{y}^n \rangle = z^2 v + \delta(y, z)$$

Note: $\mathcal{V}$ could compute commitment $Com(t_0)$ using $V = Com(v)$

Remark

We couldn't apply raw **zk-mul** as l_0 depends on verifier-provided challenges, instead $\mathcal{P}$ firstly commits to $\mathrm{a}_L, \mathrm{a}_R$ and blinders $\mathrm{s}_L, \mathrm{s}_R$, obtaints challenges y, z from $\mathcal{V}$ and computes rest of the commitments.
During verification phase $\mathcal{V}$ should adjust commitments to $\mathrm{l}(x), \mathrm{r}(x)$ by himself using homomorphic proterties of Pedersen commitment scheme.

Range proofs: building the protocol

- Setup returns independent generators $\mathsf{G}, \mathsf{H} \in \mathbb{G}^n$
- Prover does bit decomposition of v: $\mathsf{a}_L \leftarrow \mathsf{v}, \mathsf{a}_R \leftarrow \mathsf{a}_L - 1^n$, choses blinding terms $\mathsf{s}_L, \mathsf{s}_R \in \mathbb{F}_p^n, \alpha, \beta \in \mathbb{F}_p$, sends commitments:

$$A = \langle \mathsf{a}_L, \mathsf{G} \rangle + \langle \mathsf{a}_R, \mathsf{H} \rangle + [\alpha]B \quad S = \langle \mathsf{s}_L, \mathsf{G} \rangle + \langle \mathsf{s}_R, \mathsf{H} \rangle + [\beta]B$$

- Verifier $\mathcal{V}$ samples challenges $y, z \xleftarrow{R} \mathbb{F}_p$ and sends them to $\mathcal{P}$
- Prover $\mathcal{P}$ reconstructs polynomials $\mathsf{l}(x), \mathsf{r}(x), \mathsf{t}(x)$:

$$\mathsf{l}(x) = \mathsf{a}_L - z \cdot 1^n + \mathsf{s}_L x$$
$$\mathsf{r}(x) = z^2 \cdot 2^n + z \cdot \mathsf{y}^n + \mathsf{a}_R \circ \mathsf{y}^n + \mathsf{s}_R \circ \mathsf{y}^n x$$
$$t(x) = \langle \mathsf{l}(x), \mathsf{r}(x) \rangle = t_0 + t_1 x + t_2 x^2$$

$$t_0 = \langle \mathsf{a}_L - z \cdot 1^n, z^2 \cdot 2^n + z \cdot \mathsf{y}^n + \mathsf{a}_R \circ \mathsf{y}^n \rangle = z^2 v + \delta(y, z)$$
$$t_1 = \langle \mathsf{a}_L - z \cdot 1^n, \mathsf{y}^n \circ \mathsf{s}_R \rangle + \langle \mathsf{y}^n \circ (\mathsf{a}_R + z \cdot 1^n) + z^2 \cdot 2^n, \mathsf{s}_L \rangle$$
$$t_2 = \langle \mathsf{s}_L, \mathsf{y}^n \circ \mathsf{s}_R \rangle$$

Range proofs: proving

- Prover $\mathcal{P}$ draws blinding factors $\tau_1, \tau_2 \xleftarrow{R} \mathbb{F}_p$ and sends to $\mathcal{V}$ commitments for coefficients of $t(x)$:

$$T_1 = [t_1]G + [\tau_1]B$$
$$T_2 = [t_2]G + [\tau_2]B$$

Note: prover does not have to send commitment to t_0 as it's the inner-product we want to prove and it could be computed from high-level commitment V.

- Verifier $\mathcal{V}$ samples and sends to $\mathcal{P}$ evaluation point $u \xleftarrow{R} \mathbb{F}_p$
- Prover $\mathcal{P}$ evaluates polynomials at u:

$$\mathsf{l}_u = \mathsf{l}(u) \qquad \alpha_u = \alpha + \beta u$$
$$\mathsf{r}_u = \mathsf{r}(u) \qquad \tau_u = z^2\gamma + \tau_1 u + \tau_2 u^2$$
$$t_u = t(u) = t_0 + t_1 u + t_2 u^2$$

and sends $(\mathsf{l}_u, \mathsf{r}_u, t_u, \alpha_u, \tau_u)$ to $\mathcal{V}$.

Range proofs: verification

- Verifier $\mathcal{V}$ checks:

$$A + [u]S + \langle -z \cdot 1^n, \mathsf{G}\rangle + \langle z \cdot \mathsf{y}^n + z^2 \cdot 2^n, \mathsf{y}^{-n} \circ \mathsf{H}\rangle \stackrel{?}{=} \langle \mathsf{l}_u, \mathsf{G}\rangle + \langle \mathsf{r}_u, \mathsf{y}^{-n} \circ \mathsf{H}\rangle + [\alpha_u]B$$

$$[t_u]G + [\tau_u]B \stackrel{?}{=} [z^2]V + [\delta(y,z)]G + [u]T_1 + [u^2]T_2$$

$$t_u \stackrel{?}{=} \langle \mathsf{l}_u \mathsf{r}_u \rangle$$

Remark

To provide logarithmic size-proof instead of sending $\mathsf{l}_u, \mathsf{r}_u$ parties could run an inner-product argument **IPA** on inputs $(\mathsf{G}, \mathsf{y}^{-n} \circ \mathsf{H}, P, t_u; \mathsf{l}_u, \mathsf{r}_u)$ where:

$$P = A + [u]S + \langle -z \cdot 1^n, \mathsf{G}\rangle + \langle z \cdot \mathsf{y}^n + z^2 \cdot 2^n, \mathsf{y}^{-n} \circ \mathsf{H}\rangle - [\alpha_u]B$$

Range proofs: efficiency & extensions

Theorem

*The **range proof protocol** Π_{rp} has perfect completeness, computational extended witness emulation, perfect honest-verifier zero-knowledge*

Note that protocol is efficient as it has logarithmic proof size.

Remark

The **range proof protocol** could be extended to support proving multiple range proofs at once with some efficiency improvements.

Range proofs & subset-sum NP-complete problem

One of the most famous *NP-complete* problems is the **subset-sum problem**: given a set of numbers presented as vector s and number $v \in \mathbb{N}$, does a some subset sums up to v. It turns out that we could use our **range-proof** protocol for this problem. One could simply replace first inner-product check $\langle \mathsf{a}_L, 2^n \rangle = v$ with $\langle \mathsf{a}_L, \mathsf{s} \rangle = v$ where a_L is the secret vector of bits that encode positions of s that sum up to v.

Example

Let $\mathsf{s} = (6, 8, 2, 3)$ and $v = 14$. Then setting $\mathsf{a}_L = (1, 1, 0, 0)$ we could use Π_{rp} to prove that there exists a subset of s that sums up to $v = 14$ without disclosing that subset.

Therefore, **bulletproofs range proof** protocol is capable to prove a knowledge of witness to any *NP*-problem as they all could be reduced to the **subset-sum problem**

Arithmetic circuits

Bulletproofs for arithmetic circuits

- **Goal:** Prove that a circuit computes correctly without revealing inputs or intermediate values (*circuit satisfiability problem*).
- **Approach:** Use inner-product argument to prove correctness of arithmetic circuits
- **Applications:** Privacy-preserving smart contracts, confidential computations, zero-knowledge proofs for complex computations

Bulletproofs arithmetization slightly differs from the classic R1CS, however it could be transformed vice-versa easily. Also **bulletproofs** arithmetization is more convenient and human-friendly for encoding most of the arithmetic circuits than the R1CS.

Arithmetic circuits: variables

There is two types of variables in *bulletproofs* constraint system:

- **High-level variables** $\mathrm{v} \in \mathbb{F}_p^m$ are the private witness inputs to the circuit, typically provided with Pedersen commitments $\mathrm{V} \in \mathbb{G}^m$.
- **Low-level variables** $\mathrm{a}_L, \mathrm{a}_R, \mathrm{a}_O \in \mathbb{F}_p^n$ are the intermediate witness values of computation.

We will define circuit as a set of multiplication constraints operating with *low-level* variables and set of linear constraints which links *low-level* variables between each other and *high-level* variables as well.

Arithmetic circuits: constraints

Multiplication constraints are defined with one vector equation:

$$a_L \circ a_R = a_O$$

Linear constraints are defined via:

$$W_L \cdot a_L + W_R \cdot a_R + W_O \cdot a_O = W_V \cdot v + c$$

Where a_L, a_R, a_O – vectors of left and right inputs for multiplication gates and output values (all of them are low-level variables). $W_L, W_R, W_O \in \mathbb{F}_p^{q\times n}, W_V \in \mathbb{F}_p^{q\times m}$ – public matrices of weights for linear constraints(obviously known to verifier). $c \in \mathbb{F}_p^q$ – public vector of constants. Typically they encode wiring of the circuit and other linear relations between variables.

Arithmetic circuits: example

Example

Consider the following elliptic curve membership circuit. Here witness (v_1, v_2) should satisfy elliptic curve equation:

$$y^2 = x^3 + ax + b$$

The arithmetization for this circuit is as follows:

Low-level variables:

$$\mathrm{a}_L = \begin{bmatrix} x \\ x \\ y \end{bmatrix}, \quad \mathrm{a}_R = \begin{bmatrix} x \\ x^2 \\ y \end{bmatrix}, \quad \mathrm{a}_O = \begin{bmatrix} x^2 \\ x^3 \\ y^2 \end{bmatrix}$$

High-level variables:

$$\mathrm{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

Arithmetic circuits: example

Example

Multiplication constraints:

$$\mathrm{a}_L \circ \mathrm{a}_R = \mathrm{a}_O \Rightarrow \begin{bmatrix} x \cdot x = x^2 \\ x \cdot x^2 = x^3 \\ y \cdot y = y^2 \end{bmatrix}$$

Linear constraints:

$$\begin{aligned} \mathrm{a}_L^{(1)} &= v_1 & \mathrm{a}_R^{(1)} &= v_1 \\ \mathrm{a}_L^{(2)} - \mathrm{a}_L^{(1)} &= 0 & \mathrm{a}_R^{(2)} - \mathrm{a}_O^{(1)} &= 0 \\ \mathrm{a}_L^{(3)} &= v_2 & \mathrm{a}_R^{(3)} &= v_2 \\ \mathrm{a}_O^{(3)} - \mathrm{a}_O^{(2)} - a \cdot \mathrm{a}_L^{(1)} &= b \end{aligned}$$

$$\mathrm{W}_L \cdot \mathrm{a}_L + \mathrm{W}_R \cdot \mathrm{a}_R + \mathrm{W}_O \cdot \mathrm{a}_O = \mathrm{W}_V \cdot \mathrm{v} + \mathrm{c}$$

Arithmetic circuits: example

Example

$$W_L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ -a & 0 & 0 \end{bmatrix}, \quad W_R = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad W_O = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 1 \end{bmatrix},$$

$$W_V = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad c = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ b \end{bmatrix}$$

Bulletproofs for circuits: relation

Consider the relation:

$$\mathcal{R}_{sat} = \left\{ \begin{array}{l} (G, B, \mathsf{V}, \mathsf{W}_L, \mathsf{W}_R, \mathsf{W}_O, \mathsf{W}_V, \mathsf{c}; \mathsf{a}_L, \mathsf{a}_R, \mathsf{a}_O, \mathsf{v}, \mathsf{r})| \\ \forall i = 1..m : V_i = [v_i]G + [r_i]B \wedge \\ \mathsf{a}_L \circ \mathsf{a}_R = \mathsf{a}_O \wedge \\ \mathsf{W}_L \cdot \mathsf{a}_L + \mathsf{W}_R \cdot \mathsf{a}_R + \mathsf{W}_O \cdot \mathsf{a}_O = \mathsf{W}_V \cdot \mathsf{v} + \mathsf{c} \end{array} \right\}$$

Where $\mathsf{a}_L, \mathsf{a}_R, \mathsf{a}_O \in \mathbb{F}_p^n$, $\mathsf{v}, \mathsf{r} \in \mathbb{F}_p^m$, $\mathsf{W}_L, \mathsf{W}_R, \mathsf{W}_O \in \mathbb{F}_p^{q \times n}$, $\mathsf{W}_V \in \mathbb{F}_p^{q \times m}$, $\mathsf{c} \in \mathbb{F}_p^q$.

Note

Informally this relation states that there exists a valid witness v that satisfies all constraints of the circuit. For the verifier witness is presented only as commitments vector V.

Arithmetic circuits: compiling into inner-product

We could use similar to *range-proofs* technique to compile constraints of the circuit into inner-product relation. For multiplicative constraints take random $y \in \mathbb{F}_p$ and apply zero check:

$$\langle \mathrm{a}_L \circ \mathrm{a}_R - \mathrm{a}_O, \mathrm{y}^n \rangle = 0$$

Same for linear constraints, but for different randomness $z \in \mathbb{F}_p$:

$$\langle \mathrm{W}_L \cdot \mathrm{a}_L + \mathrm{W}_R \cdot \mathrm{a}_R + \mathrm{W}_O \cdot \mathrm{a}_O - \mathrm{W}_V \cdot \mathrm{v} - \mathrm{c}, \mathrm{z}^q \rangle = 0$$

Combine this two checks to one using the same randomness z:

$$\langle \mathrm{a}_L \circ \mathrm{a}_R - \mathrm{a}_O, \mathrm{y}^n \rangle + \langle z \cdot \mathrm{z}^q, \mathrm{W}_L \cdot \mathrm{a}_L + \mathrm{W}_R \cdot \mathrm{a}_R + \mathrm{W}_O \cdot \mathrm{a}_O - \mathrm{W}_V \cdot \mathrm{v} - \mathrm{c} \rangle = 0$$

This check is sound as typically a prover could not control values of y, z before he commits to $\mathrm{a_L}, \mathrm{a_R}, \mathrm{a_O}$ and v.

Arithmetic circuits: compiling into inner-product

Denote $w_c = \langle z \cdot \mathbf{z}^q, \mathbf{c}\rangle$ and flattened linear constraints(still public and easily computed by verifier):

$$\mathbf{w}_L = \mathbf{W}_L^T \cdot (z \cdot \mathbf{z}^q) \quad \mathbf{w}_R = \mathbf{W}_R^T \cdot (z \cdot \mathbf{z}^q)$$
$$\mathbf{w}_O = \mathbf{W}_O^T \cdot (z \cdot \mathbf{z}^q) \quad \mathbf{w}_V = \mathbf{W}_V^T \cdot (z \cdot \mathbf{z}^q)$$

Again doing some linear algebra witchcraft we could separate $\mathbf{a}_L, \mathbf{a}_O$ to be on the left side of the inner-product and $\mathbf{a}_R$ to be on the right:

$$w_c + \langle \mathbf{w}_V, \mathbf{v}\rangle + \delta(y, z) =$$
$$\langle \mathbf{a}_L + \mathbf{y}^{-n} \circ \mathbf{w}_R, \mathbf{y}^n \circ \mathbf{a}_R + \mathbf{w}_L\rangle + \langle \mathbf{a}_O, -\mathbf{y}^n + \mathbf{w}_O\rangle$$

Where $\delta(y, z) = \langle \mathbf{y}^{-n} \circ \mathbf{w}_R, \mathbf{w}_L\rangle$ – easily computable by $\mathcal{V}$.

Here we have a sum of 2 separate inner-products, we could express it as second-degree coefficient of the following polynomial:

$$\langle \mathbf{a}x + \mathbf{c}x^2, \mathbf{d} + \mathbf{b}x\rangle = s_1 x + s_2 x^2 + s_3 x^3 =$$
$$x \cdot \langle \mathbf{a}, \mathbf{d}\rangle + x^2 \cdot (\langle \mathbf{a}, \mathbf{b}\rangle + \langle \mathbf{c}, \mathbf{d}\rangle) + x^3 \cdot \langle \mathbf{c}, \mathbf{b}\rangle$$

Arithmetic circuits: compiling into inner-product

$$\mathbf{a} \leftarrow \mathbf{a}_L + \mathbf{y}^{-n} \circ \mathbf{w}_R \quad \mathbf{b} \leftarrow \mathbf{y}^n \circ \mathbf{a}_R + \mathbf{w}_L$$
$$\mathbf{c} \leftarrow \mathbf{a}_O \quad \mathbf{d} \leftarrow -\mathbf{y}^n + \mathbf{w}_O$$

Desired sum of inner products is the second-degree coefficient s_2:

$$w_c + \langle \mathbf{w}_V, \mathbf{v} \rangle + \delta(y, z) = s_2$$

To obtain final polynomials $\mathbf{l}(x), \mathbf{r}(x)$ we must firstly blind $\mathbf{a}_L, \mathbf{a}_R$:

$$\mathbf{a}_L \leftarrow \mathbf{a}_L + \mathbf{s}_L x^2 \quad \mathbf{a}_R \leftarrow \mathbf{a}_R + \mathbf{s}_R x^2$$

And finally compute polynomials $\mathbf{l}(x), \mathbf{r}(x)$ as follows:

$$\mathbf{l}(x) = \mathbf{s}_L \cdot x^3 + \mathbf{a}_O \cdot x^2 + (\mathbf{a}_L + \mathbf{y}^{-n} \circ \mathbf{w}_R) \cdot x$$
$$\mathbf{r}(x) = \mathbf{y}^n \circ \mathbf{s}_R \cdot x^3 + (\mathbf{y}^n \circ \mathbf{a}_R + \mathbf{w}_L) \cdot x - \mathbf{y}^n + \mathbf{w}_O$$

$$t(x) = \langle \mathbf{l}(x), \mathbf{r}(x) \rangle = \sum_{i=0}^{6} t_i x_i$$

Where $t_2 = w_c + \langle \mathbf{w}_V, \mathbf{v} \rangle + \delta(y, z)$ – desired sum of inner-products.

Arithmetic circuits: witness commitments

Here we could again apply modified **zk-mul** to prove that t_2 is a valid sum of inner-products:

- Setup: returns vectors of independent generators $\mathsf{G}, \mathsf{H} \in \mathbb{G}^n$.
- Prover $\mathcal{P}$ choses blinding factors $\alpha, \beta, \gamma \in \mathbb{F}_p, \mathsf{s}_L, \mathsf{s}_R \in \mathbb{F}_p^n$ and sends the following commitments to $\mathcal{V}$:

$$\begin{aligned} A_I &= \langle \mathsf{a}_L, \mathsf{G} \rangle + \langle \mathsf{a}_R, \mathsf{H} \rangle + [\alpha]B \\ A_O &= \langle \mathsf{a}_O, \mathsf{G} \rangle + [\gamma]B \\ S &= \langle \mathsf{s}_L, \mathsf{G} \rangle + \langle \mathsf{s}_R, \mathsf{H} \rangle + [\beta]B \end{aligned}$$

- Verifier samples challenges $y, z \xleftarrow{R} \mathbb{F}_p$ and sends them to $\mathcal{P}$.

Arithmetic circuits: product commitments

- Using challenges y, z prover forms polynomials $\mathsf{l}(x), \mathsf{r}(x), t(x)$:

$$\mathsf{l}(x) = \mathsf{s}_L \cdot x^3 + \mathsf{a}_O \cdot x^2 + (\mathsf{a}_L + \mathsf{y}^{-n} \circ \mathsf{w}_R) \cdot x$$
$$\mathsf{r}(x) = \mathsf{y}^n \circ \mathsf{s}_R \cdot x^3 + (\mathsf{y}^n \circ \mathsf{a}_R + \mathsf{w}_L) \cdot x - \mathsf{y}^n + \mathsf{w}_O$$
$$t(x) = \langle \mathsf{l}(x), \mathsf{r}(x) \rangle = t_1 x + t_2 x^2 + t_3 x^3 + t_4 x^4 + t_5 x^5 + t_6 x^6$$

$\mathcal{P}$ choses random blinding factors $\tau_1, \tau_3, \tau_4, \tau_5, \tau_6 \in \mathbb{F}_p$ and sends to $\mathcal{V}$ commitments to its coefficients:

$$T_1 = [t_1]G + [\tau_1]B \quad T_3 = [t_3]G + [\tau_3]B \quad T_4 = [t_4]G + [\tau_4]B$$
$$T_5 = [t_5]G + [\tau_5]B \quad T_6 = [t_6]G + [\tau_6]B$$

Note: Prover does not send separate commitment to t_2 as the verifier could derive it from V and the circuit public parameters:

$$t_2 = w_c + \langle \mathsf{w}_V, \mathsf{v} \rangle + \delta(y, z)$$
$$T_2 = \langle \mathsf{w}_V, \mathsf{V} \rangle + [\delta(y, z) + w_c]G$$

Arithmetic circuits: evaluating polynomials

- Verifier samples and sends to $\mathcal{P}$ random evaluation point $u \xleftarrow{R} \mathbb{F}_p$.
- Prover evaluates polynomials at u:

$$
\begin{aligned}
\mathsf{l}_u &= \mathsf{l}(u) \\
\mathsf{r}_u &= \mathsf{r}(u) \\
t_u &= \langle \mathsf{l}_u, \mathsf{r}_u \rangle = t(u) \\
\tau_u &= \tau_1 \cdot u + \langle \mathsf{w}_V, \mathsf{r} \rangle u^2 + \tau_3 \cdot u^3 + \tau_4 \cdot u^4 + \tau_5 \cdot u^5 + \tau_6 \cdot u^6 \\
\alpha_u &= \alpha u + \gamma u^2 + \beta u^3
\end{aligned}
$$

and sends $(\mathsf{l}_u, \mathsf{r}_u, t_u, \alpha_u, \tau_u)$ to $\mathcal{V}$.

Arithmetic circuits: verification

- Verifier performs checks:

$$[u]A_I + [u^2]A_O + [u^3]S - \langle \mathbf{1}, \mathrm{H}\rangle + u \cdot (\langle \mathrm{y}^{-n} \circ \mathrm{w}_L, \mathrm{G}\rangle + \langle \mathrm{y}^{-n} \circ \mathrm{w}_R, \mathrm{H}\rangle) + \langle \mathrm{y}^{-n} \circ \mathrm{w}_O, \mathrm{H}\rangle \stackrel{?}{=} \langle \mathrm{l}_u, \mathrm{G}\rangle + \langle \mathrm{r}_u, \mathrm{y}^{-n} \circ \mathrm{H}\rangle + [\alpha_u]B$$

$$[t_u]G + [\tau_u]B \stackrel{?}{=} [u]T_1 + u^2 \cdot (\langle \mathrm{w}_V, \mathrm{V}\rangle + [\delta(y,z) + w_c]G) + [u^3]T_3 + [u^4]T_4 + [u^5]T_5 + [u^6]T_6$$

$$t_u \stackrel{?}{=} \langle \mathrm{l}_u, \mathrm{r}_u\rangle$$

Remark

To provide logarithmic proof instead of sending $\mathrm{l}_u, \mathrm{r}_u$ parties could run **IPA** on inputs $(\mathrm{G}, \mathrm{y}^{-n} \circ \mathrm{H}, P, t_u; \mathrm{l}_u, \mathrm{r}_u)$ where:

$$P = [u]A_I + [u^2]A_O + [u^3]S - \langle \mathbf{1}, \mathrm{H}\rangle + u \cdot (\langle \mathrm{y}^{-n} \circ \mathrm{w}_L, \mathrm{G}\rangle + \langle \mathrm{y}^{-n} \circ \mathrm{w}_R, \mathrm{H}\rangle) + \langle \mathrm{y}^{-n} \circ \mathrm{w}_O, \mathrm{H}\rangle - [\alpha_u]B$$

Arithmetic circuits: efficiency & extensions

Theorem

*The **arithmetic circuits protocol** has perfect completeness, computational extended witness emulation, perfect honest-verifier zero-knowledge*

The protocol is efficient as it has logarithmic proof size.

Remark

The **arithmetic circuits protocol** protocol could be slightly modified to provide intermediate random challenges inside the circuit. For example it would allow proving *permutation check*: $\{a, b\} = \{c, d\} \iff (a - x) \cdot (b - x) = (c - x) \cdot (d - x)$ for some random challenge x.

Questions?