18 Bulletproofs

"Just the place for a Snark!" the Bellman cried,
As he landed his crew with care;
Supporting each man on the top of the tide
By a finger entwined in his hair.

– Lewis Carroll, "The Hunting of the Snark"

18.1 Introduction

Bulletproofs is a zero-knowledge proof protocol with logarithmically sized proofs without a trusted setup. Originally, **bulletproofs** was developed to provide efficient range proofs in application to confidential transactions, but it applies also to arbitrary arithmetic circuit (possibly encoded in R1CS). In the heart of protocol lays **inner-product argument** which we describe in details. Technically, the protocol is built in an interactive fashion (like Σ-protocols from Section 12), but one could make it non-interactive with a Fiat-Shamir transform. One key feature that differs it from Σ-protocols is the number of challenges from a verifier $\mathcal{V}$ – in Σ-protocols there is only one challenge, while **bulletproofs** implies a logarithmic in circuit size number of queries.

Also, **bulletproofs**' **inner-product argument** could be used to build various polynomial commitment schemes – crucial building block of proving systems built with *IOP* framework (*Halo, Nova, etc*).

The main advantages of **bulletproofs** are an absence of a trusted setup and security against eavesdropping that relies on the *discrete-logarithm* assumption without any other auxiliary structures like bilinear pairings. Also it has quite fast prover for small circuits making it practically useful for client-side proving. However, the main disadvantage of **bulletproofs** is that it isn't a classic *SNARK* due to linear in circuit size verification time, however still very efficient for small circuits.

18.2 Notation

Let $\mathbb{G}$ - cyclic group of prime order p written additively, $\mathbf{G} = (G_1, \ldots, G_n), \mathbf{H} = (H_1, \ldots, H_n) \in \mathbb{G}^n$ - vectors of independent generators. We denote by $\langle \mathbf{a}, \mathbf{b} \rangle$ - inner product of vectors $\mathbf{a} = (a_1, \ldots, a_n), \mathbf{b} = (b_1, \ldots, b_n) \in \mathbb{F}_p^n$ and $\langle \mathbf{a}, \mathbf{G} \rangle = \sum_{i=1}^{n} [a_i] G_i \in \mathbb{G}$ - inner product of vector $\mathbf{a}$ with vector of generators $\mathbf{G}$. Denote by $\mathbf{k}^n$ vector of k's first n powers: $\mathbf{k}^n = (1, k, k^2, \ldots, k^{n-1})$, for example $\mathbf{0}^n, \mathbf{1}^n$ represents vectors of zeros and ones respectively, while $\mathbf{2}^n = (1, 2, 4, \ldots, 2^{n-1})$

18.3 Zero-knowledge multiplication

Let $a, b, c \in \mathbb{F}_p$. Here we build a zero-knowledge protocol for relation $\mathcal{R}_{abc} = \{(\perp; c, a, b) | c = ab\}$. We use well-known Σ-protocol framework for that,

but firstly we make very useful generalization that could allow us to prove much larger class of relations.

Consider the first-degree polynomials $l(x) = a + s_L x, r(x) = b + s_R x \in \mathbb{F}_p[x]$. Let $t(x) = l(x)r(x)$ and relation

$$\mathcal{R}_{mul} = \{(\perp; l(x), r(x), t(x)) | t(x) = l(x)r(x)\}$$

Firstly, observe that proving $t(x) = l(x)r(x)$ may be reduced to evaluation check at some challenge point $u \in \mathbb{F}_p$: $t(u) = l(u)r(u)$, due to the *Schwartz-Zippel lemma*(Definition 5.14):

$$\mathsf{Pr}[l(u)r(u) = t(u) | l(x)r(x) \neq t(x)] \leq \frac{max(\deg(l(x)r(x)), \deg(t(x)))}{p} = \frac{2}{p}$$

is typically negligible function from security level which makes this check *sound*.

18.3.1 Naïve polynomial multiplication protocol

Let's describe naïve unoptimized version of **polynomial multiplication** protocol $\Pi'_{mul} = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ for relation $\mathcal{R}_{mul}$. During Setup parties agree on group elements $G, B \in \mathbb{G}$. After that parties involve in the following protocol:

- Prover $\mathcal{P}$ computes:

$$t(x) = l(x)r(x) = (a + s_L x)(b + s_R x) = ab + (as_R + bs_L) + s_L s_R x^2$$

- Prover $\mathcal{P}$ draws blinding factors $\alpha_0, \alpha_1, \beta_0, \beta_1, \tau_0, \tau_1, \tau_2 \leftarrow\!\!\$\ \mathbb{F}_p$ forming blinding polynomials

$$\alpha(x) = \alpha_0 + \alpha_1 x, \beta(x) = \beta_0 + \beta_1 x, \tau(x) = \tau_0 + \tau_1 x + \tau_2 x^2$$

and sends to $\mathcal{V}$ Pedersen commitments (Section 10) for each coefficient of $l(x), r(x), t(x)$:

$$\begin{aligned} L_0 &= [a]G + [\alpha_0]B \quad & R_0 &= [b]G + [\beta_0]B \\ L_1 &= [s_L]G + [\alpha_1]B \quad & R_1 &= [s_R]G + [\beta_1]B \\ & & T_0 &= [ab]G + [\tau_0]B \\ & & T_1 &= [as_R + bs_L]G + [\tau_1]B \\ & & T_2 &= [s_L s_R]G + [\tau_2]B \end{aligned} \tag{1}$$

> **Remark.** Each commitment could be also seen as a Pedersen commitment to a reciprocal blinding polynomial coefficient as well.

- Verifier $\mathcal{V}$ samples and sends to $\mathcal{P}$ random evaluation point $u \leftarrow\!\!\$\ \mathbb{F}_p$

- Prover $\mathcal{P}$ evaluates $l(x), r(x), t(x)$ and $\alpha(x), \beta(x), \tau(x)$ at u:

$$\begin{aligned} l_u &= a + s_L u \quad & \alpha_u &= \alpha_0 + \alpha_1 u \\ r_u &= b + s_R u \quad & \beta_u &= \beta_0 + \beta_1 u \\ t_u &= l_u r_u & \tau_u &= \tau_0 + \tau_1 u + \tau_2 u^2 \end{aligned} \tag{2}$$

  and sends $(l_u, r_u, t_u, \alpha_u, \beta_u, \tau_u)$ to $\mathcal{V}$.
- Verifier $\mathcal{V}$ performs checks:

$$\begin{aligned} [l_u]G + [\alpha_u]B &\stackrel{?}{=} L_0 + [u]L_1 \\ [r_u]G + [\beta_u]B &\stackrel{?}{=} R_0 + [u]R_1 \\ [t_u]G + [\tau_u]B &\stackrel{?}{=} T_0 + [u]T_1 + [u^2]T_2 \\ t_u &\stackrel{?}{=} l_u r_u \end{aligned} \tag{3}$$

Theorem 18.1. Naïve **polynomial multiplication** protocol Π'_{mul} has *perfect completeness, 3-special soundness, perfect honest-verifier zero-knowledge*

Proof idea. *Perfect completeness* holds due to:

$$\begin{aligned} [l_u]G + [\alpha_u]B &= [a + s_L u]G + [\alpha_0 + \alpha_1 u]B \\ L_0 + [u]L_1 &= [a]G + [\alpha_0]B + [s_L u]G + [\alpha_1 u]B = \\ &= [a + s_L u]G + [\alpha_0 + \alpha_1 u]B \\ [r_u]G + [\beta_u]B &= [b + s_R u]G + [\beta_0 + \beta_1 u]B \\ R_0 + [u]R_1 &= [b]G + [\beta_0]B + [s_R u]G + [\beta_1 u]B = \\ &= [b + s_R u]G + [\beta_0 + \beta_1 u]B \\ [t_u]G + [\tau_u]B &= [l_u r_u]G + [\tau_0 + \tau_1 u + \tau_2 u^2]B \\ T_0 + [u]T_1 + [u^2]T_2 &= [ab]G + [\tau_0]B + [u(as_R + bs_L)]G + \\ &+ [u]\tau_1 B + [u^2]s_L s_R G + [u^2]\tau_2 B \\ &= [ab + (as_R + bs_L)u + s_L s_R u^2]G + [\tau_0 + \tau_1 u + \tau_2 u^2]B \\ &= [l_u r_u]G + [\tau_0 + \tau_1 u + \tau_2 u^2]B \end{aligned}$$

Proving *honest-verifier zero-knowledge* is a bit complicated due to proper building of a simulator and proving indistinguishability of distributions, so we briefly describe the idea behind it: each commitment sent in the first phase by $\mathcal{P}$ is a Pedersen commitment which is hiding by design, every second phase response of $\mathcal{P}$ is an evaluation of some first or second degree polynomial at chosen point so there's not enough information for interpolation and polynomial reconstruction, moreover it could be easily simulated.

To prove *3-special soundness* we need to build a knowledge extractor $\mathcal{E}$ which extracts knowledge of witness polynomials $l(x), r(x), t(x)$ such that $l(x)r(x) = t(x)$ using 3 accepting transcripts:

1. $\mathcal{E}$ runs $\mathcal{P}$ to the end and rewinds back the second phase of $\mathcal{P}$, getting three non-equal challenges $u_1, u_2, u_3 \in \mathbb{F}_p$ and three prover responses $(l_{u_i}, r_{u_i}, t_{u_i})_{i=1}^3$
2. $\mathcal{E}$ solves the following systems of linear equations:

$$\begin{cases} l_{u_1} = a + s_L u_1 \\ l_{u_2} = a + s_L u_2 \end{cases} \qquad \begin{cases} r_{u_1} = b + s_R u_1 \\ r_{u_2} = b + s_R u_2 \end{cases} \qquad \begin{cases} t_{u_1} = t_0 + t_1 u_1 + t_2 u_1^2 \\ t_{u_2} = t_0 + t_1 u_2 + t_2 u_2^2 \\ t_{u_3} = t_0 + t_1 u_3 + t_2 u_3^2 \end{cases}$$

   and gets the coefficients of witness polynomials: $(a, s_L, b, s_R, t_0, t_1, t_2)$
3. To prove that $l(x)r(x) = t(x)$ we apply Schwartz-Zippel lemma which asserts polynomial equality with high probability since for random challenges u_i for honest prover we have $l(u_i)r(u_i) = t(u_i)$ $\square$.

18.3.2 Optimized polynomial multiplication protocol

We could optimize our **polynomial multiplication protocol** furthermore. Note that we could simply apply vector Pedersen commitment for constant and linear terms using one more group element $H \in \mathbb{G}$.

Definition 18.2. The **polynomial multiplication protocol** $\Pi_{mul} = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ for the relation

$$\mathcal{R}_{mul} = \{(\perp; l(x), r(x), t(x)) | t(x) = l(x)r(x)\}$$

with prover $\mathcal{P}$ and verifier $\mathcal{V}$ is defined as follows:

- Setup returns triple of group generators with unknown discrete log relations $G, H, B \in \mathbb{G}$
- Parties $\mathcal{P}, \mathcal{V}$ run the following protocol:
  - Prover $\mathcal{P}$ computes:

$$t(x) = l(x)r(x) = (a + s_L x)(b + s_R x) = ab + (as_R + bs_L) + s_L s_R x^2$$

  - Prover $\mathcal{P}$ draws blinding factors $\alpha, \beta, \tau_0, \tau_1, \tau_2 \leftarrow\$ \mathbb{F}_p$ and sends to $\mathcal{V}$ the following commitments for coefficients of $l(x), r(x), t(x)$:

$$\begin{aligned} A &= [a]G + [b]H + [\alpha]B \\ S &= [s_L]G + [s_R]H + [\beta]B \end{aligned}$$

$$\begin{aligned} T_0 &= [ab]G + [\tau_0]B \\ T_1 &= [as_R + bs_L]G + [\tau_1]B \\ T_2 &= [s_L s_R]G + [\tau_2]B \end{aligned} \tag{4}$$

  - Verifier $\mathcal{V}$ samples and sends to $\mathcal{P}$ random evaluation point $u \leftarrow\$ \mathbb{F}_p$

– Prover $\mathcal{P}$ evaluates polynomials at u:

$$\begin{aligned} l_u &= a + s_L u \quad & \alpha_u &= \alpha + \beta u \\ r_u &= b + s_R u \quad & \tau_u &= \tau_0 + \tau_1 u + \tau_2 u^2 \\ t_u &= l_u r_u \end{aligned} \tag{5}$$

and sends $(l_u, r_u, t_u, \alpha_u, \tau_u)$ to $\mathcal{V}$.

– Verifier $\mathcal{V}$ performs checks:

$$\begin{aligned} A + [u]S &\stackrel{?}{=} [l_u]G + [r_u]H + [\alpha_u]B \\ [t_u]G + [\tau_u]B &\stackrel{?}{=} T_0 + [u]T_1 + [u^2]T_2 \\ t_u &\stackrel{?}{=} l_u r_u \end{aligned} \tag{6}$$

Theorem 18.3. The **polynomial multiplication** protocol Π_{mul} has *perfect completeness, special soundness, perfect honest-verifier zero-knowledge*

Proof. We left to a reader proof of the theorem in the sake of brevity because it's very similar to the proof of Definition 18.1 □

18.3.3 Zero-knowledge multiplication protocol

Finally, we could easily build the protocol for the zk-multiplication relation where each witness element presented in statement as a Pedersen commitment:

$$\mathcal{R}_{abc} = \left\{ \begin{array}{l} (G, H, B, A, T_0; a, b, \alpha, \tau_0) | \\ A = [a]G + [b]H + [\alpha]B \wedge \\ T_0 = [ab]G + [\tau_0]B \end{array} \right\}$$

Where $G, H, B \in \mathbb{G}$ are generators with unknown discrete log relations, $A \in \mathbb{G}$ is a Pedersen commitment to a, b and $T_0 \in \mathbb{G}$ is a Pedersen commitment to their product ab.

Definition 18.4. The **multiplication protocol** $\Pi_{abc} = (\mathcal{P}, \mathcal{V})$ for the relation $\mathcal{R}_{abc}$ with prover $\mathcal{P}$ and verifier $\mathcal{V}$ is defined as follows:

- Prover $\mathcal{P}$ draws random $s_L, s_R \leftarrow\!\!\$\ \mathbb{F}_p$ and defines polynomials:

$$l(x) = a + s_L x, \quad r(x) = b + s_R x, \quad t(x) = l(x)r(x)$$

- Parties run Π_{mul} on inputs $(l(x), r(x), t(x))$ along with provided a-priori setup $G, H, B \in \mathbb{G}$ and commitments A, T_0

The protocol obviously has *perfect completeness, special soundness, perfect honest-verifier zero-knowledge* due to the Definition 18.3.

Remark. Now curious reader may wonder why build so overwhelmingly complicated protocol for simple multiplication and not just use classic *Chaum-Pedersen protocol* for DH-triplets? Indeed, it definitely could establish that for given group elements $[a]G, [b]G, [c]G$ equality $c = ab$ holds, but unfortunately commitments $[a]G, [b]G, [c]G$ do not have a *perfect hiding* property (though preserving *computational binding* property) so an adversary could potentially learn a, b, c values especially if they are small or have non-uniform distribution.

Also, there's a folklore version of very similar protocol for establishing product relationship between Pedersen committed values described in [**?**, section 12].

18.3.4 Zero-knowledge inner-product protocol

We could extend our Π_{mul} protocol even further to provide zero-knowledge proof for the inner-product of vectors: $\langle \mathbf{a}, \mathbf{b} \rangle = v$. The main trick is to substitute polynomials $l(x), r(x) \in \mathbb{F}_p[x]$ by vector polynomials $\mathbf{l}(x), \mathbf{r}(x) \in \mathbb{F}_p^n[x]$ where constant terms are equal to $\mathbf{a}$ and $\mathbf{b}$ respectively, taking inner-product $\langle \mathbf{l}(x), \mathbf{r}(x) \rangle$ results in polynomial with scalar coefficients where constant term is equal to $\langle \mathbf{a}, \mathbf{b} \rangle$.

Example 18.1. Let $\mathbf{a} = (a_1, a_2)$ and $\mathbf{b} = (b_1, b_2)$ be vectors in $\mathbb{F}_p^2$. Consider vector polynomials with vector coefficients:

$$\mathbf{l}(x) = \mathbf{a} + \mathbf{s}_L x = (a_1, a_2) + (s_{L,1}, s_{L,2})x$$

$$\mathbf{r}(x) = \mathbf{b} + \mathbf{s}_R x = (b_1, b_2) + (s_{R,1}, s_{R,2})x$$

where $\mathbf{s}_L = (s_{L,1}, s_{L,2})$, $\mathbf{s}_R = (s_{R,1}, s_{R,2})$.
Their inner-product is a degree two scalar polynomial:

$$\begin{aligned} t(x) &= \langle \mathbf{l}(x), \mathbf{r}(x) \rangle \\ &= \langle \mathbf{a} + \mathbf{s}_L x,\ \mathbf{b} + \mathbf{s}_R x \rangle \\ &= \langle \mathbf{a}, \mathbf{b} \rangle + \langle \mathbf{a}, \mathbf{s}_R \rangle x + \langle \mathbf{s}_L, \mathbf{b} \rangle x + \langle \mathbf{s}_L, \mathbf{s}_R \rangle x^2 \end{aligned}$$

So the constant term is the inner-product $\langle \mathbf{a}, \mathbf{b} \rangle$.

Definition 18.5. The **zero-knowledge inner-product protocol** $\Pi_{zkip} = (\mathcal{P}, \mathcal{V})$ for the relation

$$\mathcal{R}_{zkip} = \left\{ (\mathbf{G}, \mathbf{H}, G, B, A, V; \mathbf{a}, \mathbf{b}, \alpha, \gamma) | A = \langle \mathbf{a}, \mathbf{G} \rangle + \langle \mathbf{b}, \mathbf{H} \rangle + [\alpha]B, \atop V = [\langle \mathbf{a}, \mathbf{b} \rangle]G + [\gamma]B \right\}$$

where $\mathbf{G}, \mathbf{H} \in \mathbb{G}^n, G, B \in \mathbb{G}$ – independent group generators with prover $\mathcal{P}$ and verifier $\mathcal{V}$ is defined as follows:

- Prover $\mathcal{P}$ choses blinding vectors $\mathbf{s}_L, \mathbf{s}_R \in \mathbb{F}_p^n$ and computes polynomi-

als:

$$\begin{aligned}\mathbf{l}(x) &= \mathbf{a} + \mathbf{s}_L x \\ \mathbf{r}(x) &= \mathbf{b} + \mathbf{s}_R x \\ t(x) &= \langle \mathbf{l}(x), \mathbf{r}(x) \rangle = \langle \mathbf{a}, \mathbf{b} \rangle + (\langle \mathbf{a}, \mathbf{s}_R \rangle + \langle \mathbf{s}_L, \mathbf{b} \rangle)x + \langle \mathbf{s}_L, \mathbf{s}_R \rangle x^2\end{aligned}$$

- Prover $\mathcal{P}$ draws blinding factors $\beta, \tau_1, \tau_2 \leftarrow\!\!\$\ \mathbb{F}_p$ and sends to $\mathcal{V}$ the following commitments for coefficients of $\mathbf{l}(x), \mathbf{r}(x), \mathbf{t}(x)$:

$$\begin{aligned}S &= \langle \mathbf{s}_L, \mathbf{G} \rangle + \langle \mathbf{s}_R, \mathbf{H} \rangle + [\beta]B \\ T_1 &= [\langle \mathbf{a}, \mathbf{s}_R \rangle + \langle \mathbf{s}_L, \mathbf{b} \rangle]G + [\tau_1]B \\ T_2 &= [\langle \mathbf{s}_L \mathbf{s}_R \rangle]G + [\tau_2]B\end{aligned} \tag{7}$$

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

$$\begin{aligned}\mathbf{l}_u &= \mathbf{a} + \mathbf{s}_L u \quad & \alpha_u &= \alpha + \beta u \\ \mathbf{r}_u &= \mathbf{b} + \mathbf{s}_R u \quad & \tau_u &= \tau_0 + \tau_1 u + \tau_2 u^2 \\ t_u &= \langle \mathbf{l}_u \mathbf{r}_u \rangle\end{aligned} \tag{8}$$

  and sends $(\mathbf{l}_u, \mathbf{r}_u, t_u, \alpha_u, \tau_u)$ to $\mathcal{V}$.
- Verifier $\mathcal{V}$ performs checks:

$$\begin{aligned}A + [u]S &\stackrel{?}{=} \langle \mathbf{l}_u, \mathbf{G} \rangle + \langle \mathbf{r}_u, \mathbf{H} \rangle + [\alpha_u]B \\ [t_u]G + [\tau_u]B &\stackrel{?}{=} V + [u]T_1 + [u^2]T_2 \\ t_u &\stackrel{?}{=} \langle \mathbf{l}_u \mathbf{r}_u \rangle\end{aligned} \tag{9}$$

The protocol also has *perfect completeness, special soundness, perfect honest-verifier zero-knowledge* due to the Definition 18.3, however building the extractor needs some vector equations we omit for the sake of brevity. Also note that transcript size is linear in size of vectors $\mathbf{l}_u, \mathbf{r}_u$ which is extremely inefficient when vectors are large. So in the next section we present so called **inner-product argument** which is summoned to reduce conversational complexity to logarithmic in vector length making the last check $t_u \stackrel{?}{=} \langle \mathbf{l}_u \mathbf{r}_u \rangle$ quite efficient.

18.4 Inner-product argument

Here we describe the further generalization of Π_{mul} – efficient protocol for the **inner-product argument** - core component of the **bulletproofs** protocol. After that we will apply it to range proofs and arithmetic circuits. We have already seen that inner-products are the main ingredients for R1CS language because

any R1CS relation could be seen as a batch of inner-products though it's not the most efficient representation and we'll see how to amortize all the constraints into inner-products more efficiently.

The **inner-product argument** allows to prove that two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}_p^n$ satisfy the relation:

$$\mathcal{R}_{ip} = \{(\mathbf{G}, \mathbf{H}, P, c; \mathbf{a}, \mathbf{b}) | P = \langle \mathbf{a}, \mathbf{G} \rangle + \langle \mathbf{b}, \mathbf{H} \rangle \wedge \langle \mathbf{a}, \mathbf{b} \rangle = c\}$$

We refer to $P \in \mathbb{G}$ as a binding Pedersen vector commitment to $\mathbf{a}, \mathbf{b}$.

One way to prove the relation is to use Π_{zkip}, but as we've seen it's not efficient due to linear in n size of the proof. We want to build an argument system for the relation $\mathcal{R}_{ip}$ with logarithmic in n size of the proof.

Firtsly, let's combine statements $P = \langle \mathbf{a}, \mathbf{G} \rangle + \langle \mathbf{b}, \mathbf{H} \rangle \wedge \langle \mathbf{a}, \mathbf{b} \rangle = c$ into a single statement by multiplying the second one by a random $r \in \mathbb{F}_p$ and some orthogonal generator $B \in \mathbb{G}$, summing up:

$$\mathcal{R}'_{ip} = \{(\mathbf{G}, \mathbf{H}, Q, P'; \mathbf{a}, \mathbf{b}) | P' = \langle \mathbf{a}, \mathbf{G} \rangle + \langle \mathbf{b}, \mathbf{H} \rangle + [\langle \mathbf{a}, \mathbf{b} \rangle] Q\}$$

Where $P' = P + [cr]B, Q = [r]B$. Intuitively, if prover $\mathcal{P}$ can prove $\mathcal{R}'_{ip}$ for all $r \in \mathbb{F}_p$, then it can prove $\mathcal{R}_{ip}$ for any valid witness. We use such transformation to compress each vector in half and arrive to the same form of commitment

$$P' = \langle \mathbf{a}, \mathbf{G} \rangle + \langle \mathbf{b}, \mathbf{H} \rangle + [\langle \mathbf{a}, \mathbf{b} \rangle] Q$$

Definition 18.6. The **inner-product** protocol $\Pi_{ip} = (\mathcal{P}, \mathcal{V})$ for relation $\mathcal{R}_{ip} = \{(\mathbf{G}, \mathbf{H}, P, c; \mathbf{a}, \mathbf{b}) | P = \langle \mathbf{a}, \mathbf{G} \rangle + \langle \mathbf{b}, \mathbf{H} \rangle \wedge \langle \mathbf{a}, \mathbf{b} \rangle = c\}$ with prover $\mathcal{P}$, verifier $\mathcal{V}$ is defined as follows:

- Parties $\mathcal{P}, \mathcal{V}$ agree on some group element $B \in \mathbb{G}$ with unknown discrete log
- Verifier $\mathcal{V}$ samples random $r \leftarrow\!\!\$\ \mathbb{F}_p$ and sends it to prover $\mathcal{P}$
- Parties $\mathcal{P}, \mathcal{V}$ compute $Q \leftarrow [r]B$ and $P' = P + [c]Q$
- Parties run protocol Π'_{ip} for relation $\mathcal{R}'_{ip}$ on input $(\mathbf{G}, \mathbf{H}, Q, P'; \mathbf{a}, \mathbf{b})$

18.4.1 Inner-product compression

Here we describe **inner-product compression** algorithm – main building block of the interactive **inner-product** protocol. Firstly, assuming that $n = 2^d$ define by $\mathbf{G_{lo}} = (G_1, \ldots, G_{n/2}), \mathbf{G_{hi}} = (G_{n/2+1}, \ldots, G_n) \in \mathbb{G}^{n/2}$ – lower and higher halves of vector $\mathbf{G}$ and $\mathbf{a_{lo}} = (a_1, \ldots, a_{n/2}), \mathbf{a_{hi}} = (a_{n/2+1}, \ldots, a_n) \in \mathbb{F}_p^{n/2}$ – lower and higher halves of $\mathbf{a} \in \mathbb{F}_p^n$.

Let $u_k \in \mathbb{F}_p$ - be some scalar, define compressed vectors:

$$\begin{aligned}
\mathbf{a}^{(k-1)} &= \mathbf{a_{lo}} \cdot u_k + u_k^{-1} \cdot \mathbf{a_{hi}} \\
\mathbf{b}^{(k-1)} &= \mathbf{b_{lo}} \cdot u_k^{-1} + u_k \cdot \mathbf{b_{hi}} \\
\mathbf{G}^{(k-1)} &= \mathbf{G_{lo}} \cdot u_k^{-1} + u_k \cdot \mathbf{G_{hi}} \\
\mathbf{H}^{(k-1)} &= \mathbf{H_{lo}} \cdot u_k + u_k^{-1} \cdot \mathbf{H_{hi}}
\end{aligned}$$

Define $P_k \leftarrow P' = \langle \mathbf{a}, \mathbf{G} \rangle + \langle \mathbf{b}, \mathbf{H} \rangle + [\langle \mathbf{a}, \mathbf{b} \rangle]Q$ – current commitment to vectors $\mathbf{a}, \mathbf{b}$ and define P_{k-1} using compressed vectors to have the same form as P_k, but in new basis $(\mathbf{G}^{(k-1)}, \mathbf{H}^{(k-1)})$:

$$P_{k-1} = \langle \mathbf{a}^{(k-1)}, \mathbf{G}^{(k-1)} \rangle + \langle \mathbf{b}^{(k-1)}, \mathbf{H}^{(k-1)} \rangle + [\langle \mathbf{a}^{(k-1)}, \mathbf{b}^{(k-1)} \rangle]Q \tag{10}$$

Or alternatively, expressing P_{k-1} in old basis $(\mathbf{G}^{(k)}, \mathbf{H}^{(k)})$ we get:

$$\begin{aligned}
P_{k-1} = \langle u_k^{-1} \cdot \mathbf{a}^{(k-1)}, \mathbf{G_{lo}}^{(k)} \rangle + \langle u_k \cdot \mathbf{a}^{(k-1)}, \mathbf{G_{hi}}^{(k)} \rangle + \langle u_k \cdot \mathbf{b}^{(k-1)}, \mathbf{H_{lo}}^{(k)} \rangle \\
+ \langle u_k^{-1} \cdot \mathbf{b}^{(k-1)}, \mathbf{H_{hi}}^{(k)} \rangle + [\langle \mathbf{a}^{(k-1)}, \mathbf{b}^{(k-1)} \rangle]Q
\end{aligned} \tag{11}$$

Substituting compressed vectors and applying bilinearity property of inner product we get:

$$\begin{aligned}
P_{k-1} = &\langle \mathbf{a_{lo}}, \mathbf{G_{lo}} \rangle + \langle \mathbf{a_{hi}}, \mathbf{G_{hi}} \rangle \quad &+u_k^2 \langle \mathbf{a_{lo}}, \mathbf{G_{hi}} \rangle + u_k^{-2} \langle \mathbf{a_{hi}}, \mathbf{G_{lo}} \rangle + \\
&\langle \mathbf{b_{lo}}, \mathbf{H_{lo}} \rangle + \langle \mathbf{b_{hi}}, \mathbf{H_{hi}} \rangle \quad &+u_k^2 \langle \mathbf{b_{hi}}, \mathbf{H_{lo}} \rangle + u_k^{-2} \langle \mathbf{b_{lo}}, \mathbf{H_{hi}} \rangle + \\
&[\langle \mathbf{a_{lo}}, \mathbf{b_{lo}} \rangle + \langle \mathbf{a_{hi}}, \mathbf{b_{hi}} \rangle]Q \quad &+[u_k^2 \langle \mathbf{a_{lo}}, \mathbf{b_{hi}} \rangle + u_k^{-2} \langle \mathbf{a_{hi}}, \mathbf{b_{lo}} \rangle]Q
\end{aligned}$$

Note that $\langle \mathbf{a_{lo}}, \mathbf{G_{lo}} \rangle + \langle \mathbf{a_{hi}}, \mathbf{G_{hi}} \rangle = \langle \mathbf{a}, \mathbf{G} \rangle$ so that the first two columns of P_{k-1} definition precisecly contains $P_k = P'$:

$$P_k = \langle \mathbf{a_{lo}}, \mathbf{G_{lo}} \rangle + \langle \mathbf{a_{hi}}, \mathbf{G_{hi}} \rangle + \langle \mathbf{b_{lo}}, \mathbf{H_{lo}} \rangle + \langle \mathbf{b_{hi}}, \mathbf{H_{hi}} \rangle + [\langle \mathbf{a_{lo}}, \mathbf{b_{lo}} \rangle + \langle \mathbf{a_{hi}}, \mathbf{b_{hi}} \rangle]Q$$

Define cross-terms L_k, R_k of P_{k-1} such that:

$$\begin{aligned}
P_{k-1} &= P_k + [u_k^2]L_k + [u_k^{-2}]R_k \\
L_k &= \langle \mathbf{a_{lo}}, \mathbf{G_{hi}} \rangle + \langle \mathbf{b_{hi}}, \mathbf{H_{lo}} \rangle + [\langle \mathbf{a_{lo}}, \mathbf{b_{hi}} \rangle]Q \\
R_k &= \langle \mathbf{a_{hi}}, \mathbf{G_{lo}} \rangle + \langle \mathbf{b_{lo}}, \mathbf{H_{hi}} \rangle + [\langle \mathbf{a_{hi}}, \mathbf{b_{lo}} \rangle]Q
\end{aligned}$$

The first equation $P_{k-1} = P_k + [u_k^2]L_k + [u_k^{-2}]R_k$ could be used as a check for asserting correctness of next commitment P_{k-1} given cross-terms L_k, R_k, half-sized vectors $\mathbf{a}^{(k-1)}, \mathbf{b}^{(k-1)}$ from which P_{k-1} was computed (10) using updated basis $\mathbf{G}^{(k-1)}, \mathbf{H}^{(k-1)}$ and current commitment value P_k.

But we wish not send $\mathbf{a}^{(k-1)}, \mathbf{b}^{(k-1)}$ directly as this's inefficient due to still linear sizes, instead we apply recursion to compress this vectors to just one

element. Here we come up with some kind of statement compression algorithm reducing size of all vectors in half per compression step. Repeating compression algorithm k times we end up with sending vectors $\mathbf{a}^{(0)}, \mathbf{b}^{(0)}$ each of length one and P_0 containing all accumulated cross-terms:

$$P_0 = [a_1^{(0)}]G_1^{(0)} + [b_1^{(0)}]H_1^{(0)} + [a_1^{(0)}b_1^{(0)}]Q$$
$$P_0 = P_k + \sum_{i=1}^{k}([u_i^2]L_i + [u_i^{-2}]R_i)$$

Recalling that $P_k = P'$, the final compressed statement asserting inner-product value will have the following form:

$$P' + \sum_{i=1}^{k}([u_i^2]L_i + [u_i^{-2}]R_i) = [a_1^{(0)}]G_1^{(0)} + [b_1^{(0)}]H_1^{(0)} + [a_1^{(0)}b_1^{(0)}]Q \tag{12}$$

Remark. We take $n = 2^d$ without loss of generality, since one could always pad the vectors with zeroes to make their length a power of two and inner-product compression would shrink the size of the vectors by a factor of two per compression step down to one element.

18.4.2 Proving $\mathcal{R}'_{ip}$

Let's describe the **inner-product** protocol Π'_{ip} for relation $\mathcal{R}'_{ip}$.

Definition 18.7. The **inner-product** protocol $\Pi'_{ip} = (\mathcal{P}, \mathcal{V})$ for relation $\mathcal{R}'_{ip} = \{(\mathbf{G}, \mathbf{H}, Q, P'; \mathbf{a}, \mathbf{b}) | P' = \langle \mathbf{a}, \mathbf{G} \rangle + \langle \mathbf{b}, \mathbf{H} \rangle + [\langle \mathbf{a}, \mathbf{b} \rangle]Q\}$, where all vectors have length $n = 2^d$ with prover $\mathcal{P}$, verifier $\mathcal{V}$ is defined as follows:

- Prover $\mathcal{P}$ sets

$$(k, \mathbf{a}^{(k)}, \mathbf{b}^{(k)}, \mathbf{G}^{(k)}, \mathbf{H}^{(k)}, P_k) \leftarrow (d, \mathbf{a}, \mathbf{b}, \mathbf{G}, \mathbf{H}, P')$$

- Verifier $\mathcal{V}$ sets

$$(k, \mathbf{G}^{(k)}, \mathbf{H}^{(k)}, P_k) \leftarrow (d, \mathbf{G}, \mathbf{H}, P')$$

- While $k > 0$ then:
  - Prover $\mathcal{P}$ computes

$$L_k = \langle \mathbf{a_{lo}}^{(k)}, \mathbf{G_{hi}}^{(k)} \rangle + \langle \mathbf{b_{hi}}^{(k)}, \mathbf{H_{lo}}^{(k)} \rangle + [\langle \mathbf{a_{lo}}^{(k)}, \mathbf{b_{hi}}^{(k)} \rangle]Q$$
$$R_k = \langle \mathbf{a_{hi}}^{(k)}, \mathbf{G_{lo}}^{(k)} \rangle + \langle \mathbf{b_{lo}}^{(k)}, \mathbf{H_{hi}}^{(k)} \rangle + [\langle \mathbf{a_{hi}}^{(k)}, \mathbf{b_{lo}}^{(k)} \rangle]Q$$

  - $\mathcal{P}$ sends (L_k, R_k) to $\mathcal{V}$
  - $\mathcal{V}$ draws challenge $u_k \xleftarrow{\$} \mathbb{F}_p$ and sends it to $\mathcal{P}$

– Both $\mathcal{P}$ and $\mathcal{V}$ compute:

$$\mathbf{G}^{(k-1)} = \mathbf{G_{lo}}^{(k)} \cdot u_k^{-1} + u_k \cdot \mathbf{G_{hi}}^{(k)}$$
$$\mathbf{H}^{(k-1)} = \mathbf{H_{lo}}^{(k)} \cdot u_k + u_k^{-1} \cdot \mathbf{H_{hi}}^{(k)}$$

– $\mathcal{P}$ computes:

$$\mathbf{a}^{(k-1)} = \mathbf{a_{lo}}^{(k)} \cdot u_k + u_k^{-1} \cdot \mathbf{a_{hi}}^{(k)}$$
$$\mathbf{b}^{(k-1)} = \mathbf{b_{lo}}^{(k)} \cdot u_k^{-1} + u_k \cdot \mathbf{b_{hi}}^{(k)}$$

- Prover $\mathcal{P}$ sends $(a, b) \leftarrow (\mathbf{a}_1^{(0)}, \mathbf{b}_1^{(0)})$ to verifier $\mathcal{V}$
- Verifier performs final check:

$$P' + \sum_{i=1}^{d}([u_i^2]L_i + [u_i^{-2}]R_i) = [a]G_1^{(0)} + [b]H_1^{(0)} + [ab]Q$$

outputs **accept** if equality holds and **reject** otherwise.

This protocol is illustrated in Figure 18.1.

As we can see, overall communication complexity of Π_{ip} is $2\log_2 n$ group elements plus 2 field elements so we come up with logarithmic proof size for our inner-product relation $\mathcal{R}_{ip}$. Now one might ask whether Π_{ip} has any desired properties such as completeness, soundness and zero-knowledge. It appears that the first two holds under some generalizations needed for security proofs but not *zero-knowledge* (indeed, if $n = 1$ then $\mathcal{P}$ sends witness pair a, b directly). We'll compile efficient **inner-product argument** Π_{ip} with zero-knowledge Π_{zkip} to achieve efficient zero-knowledge proofs for range proofs and arithmetic circuits.

Theorem 18.8 (Inner-Product Argument)**.** The argument system Π_{ip} for relation $\mathcal{R}_{ip}$ has *perfect completeness and statistical witness-extended emulation* for either extracting a non-trivial discrete logarithm relation between $\mathbf{G}, \mathbf{H}, Q$ or extracting valid witness $\mathbf{a}, \mathbf{b}$.

Proof idea. *Perfect completeness* of Pi_{ip} follows because Pi_{ip} converts instance of $\mathcal{R}_{ip}$ to instance of $\mathcal{P}'_{ip}$ and Π'_{ip} is trivially complete by construction due to Equation (12). Notation *statistical witness-extended emulation* generalizes *special soundness* in the way applicable for multi-stage complex argument systems where each step of the protocol could be rewound to extract part of the witness so that more accurate definition of protocol security is achieved despite *special soundness* implies building the whole knowledge extractor which might has non-polynomial running time for multi-stage protocols.

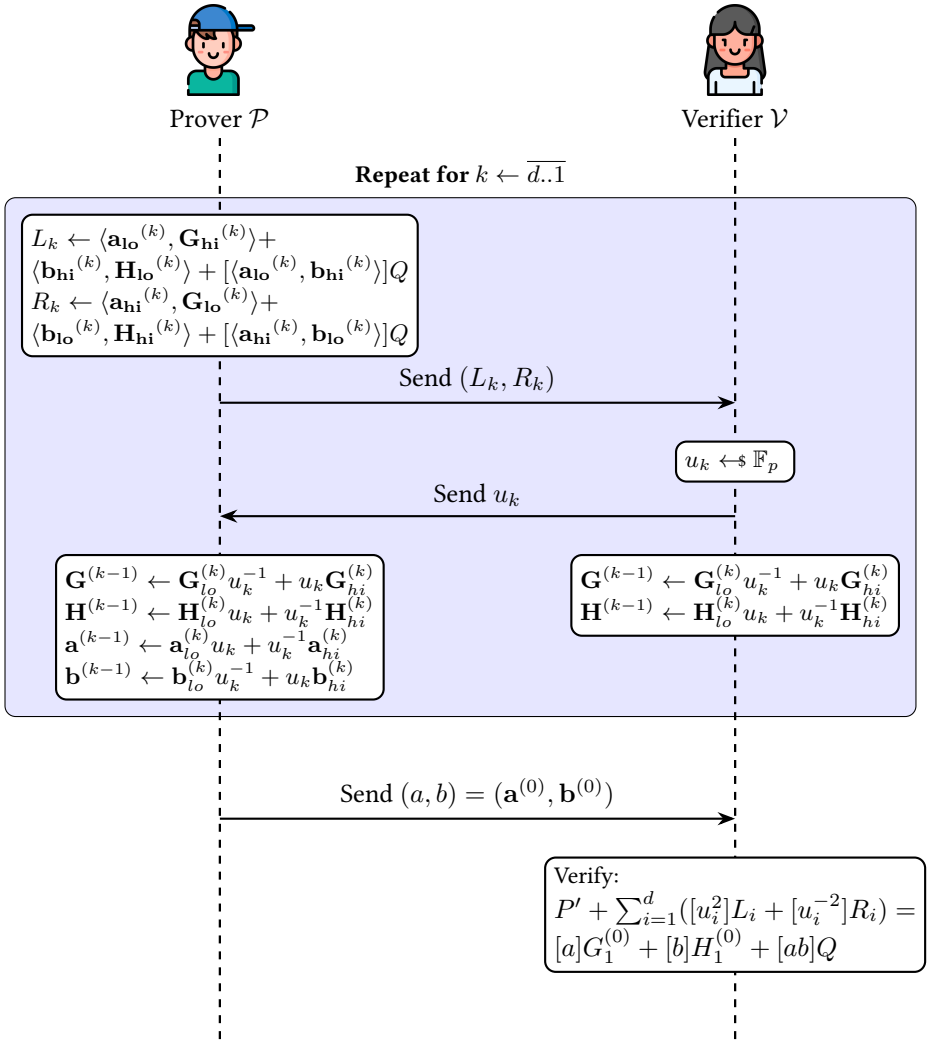

Figure 18.1: Interactive inner-product protocol Π'_{ip} between prover $\mathcal{P}$ and verifier $\mathcal{V}$ for relation $\mathcal{R}'_{ip}$

Here we briefly describe a knowledge extractor $\mathcal{E}'_{ip}$ for a witness $(\mathbf{a}, \mathbf{b})$ or non-trivial discrete logarithm relation for $(\mathbf{G}, \mathbf{H}, Q)$ for Π'_{ip}.

1. $\mathcal{E}'_{ip}$ runs $\mathcal{P}$ to the last stage to obtain $(L, R) \leftarrow (L_1, R_1)$.
2. $\mathcal{E}'_{ip}$ rewinds the Π'_{ip} final stage four times to obtain four challenges x_1, x_2, x_3, x_4 and responses $(\mathbf{a}_1^{(0)}, \mathbf{b}_1^{(0)}), (\mathbf{a}_2^{(0)}, \mathbf{b}_2^{(0)}), (\mathbf{a}_3^{(0)}, \mathbf{b}_3^{(0)}), (\mathbf{a}_4^{(0)}, \mathbf{b}_4^{(0)})$

so that the following equation holds for $i \in \{1,2,3,4\}$ (11):

$$[x_i^2]L + P_1 + [x_i^{-2}]R = \langle x_i^{-1} \cdot \mathbf{a}_i^{(0)}, \mathbf{G_{lo}}^{(1)} \rangle + \langle x_i \cdot \mathbf{a}_i^{(0)}, \mathbf{G_{hi}}^{(1)} \rangle + \langle x_i \cdot \mathbf{b}_i^{(0)}, \mathbf{H_{lo}}^{(1)} \rangle + \langle x_i^{-1} \cdot \mathbf{b}_i^{(0)}, \mathbf{H_{hi}}^{(1)} \rangle + [\langle \mathbf{a}_i^{(0)}, \mathbf{b}_i^{(0)} \rangle] Q \tag{13}$$

3. $\mathcal{E}'_{ip}$ choses $(v_1, v_2, v_3) \in \mathbb{F}_p^3$ as a solution for the system of linear equations:

$$\begin{cases} x_1^2 v_1 + x_2^2 v_2 + x_3^2 v_3 = 0 \\ v_1 + v_2 + v_3 = 1 \\ x_1^{-2} v_1 + x_2^{-2} v_2 + x_3^{-2} v_3 = 0 \end{cases} \tag{14}$$

4. $\mathcal{E}'_{ip}$ takes linear combination of (13) with coefficients v_i for $i \in \{1,2,3\}$:

$$\sum_{i=1}^{3} [v_i x_i^2] L + [v_i] P_1 + [v_i x_i^{-2}] R = \sum_{i=1}^{3} \langle v_i x_i^{-1} \cdot \mathbf{a}_i^{(0)}, \mathbf{G_{lo}}^{(1)} \rangle + \langle v_i x_i \cdot \mathbf{a}_i^{(0)}, \mathbf{G_{hi}}^{(1)} \rangle + \langle v_i x_i \cdot \mathbf{b}_i^{(0)}, \mathbf{H_{lo}}^{(1)} \rangle + \langle v_i x_i^{-1} \cdot \mathbf{b}_i^{(0)}, \mathbf{H_{hi}}^{(1)} \rangle + [v_i \langle \mathbf{a}_i^{(0)}, \mathbf{b}_i^{(0)} \rangle] Q$$

Simplifying equation above we get:

$$P_1 = \langle \sum_{i=1}^{3} v_i x_i^{-1} \cdot \mathbf{a}_i^{(0)} + v_i x_i \cdot \mathbf{a}_i^{(0)}, \mathbf{G}^{(1)} \rangle + \langle \sum_{i=1}^{3} v_i x_i \cdot \mathbf{b}_i^{(0)} + v_i x_i^{-1} \cdot \mathbf{b}_i^{(0)}, \mathbf{H}^{(1)} \rangle + [\sum_{i=1}^{3} v_i \langle \mathbf{a}_i^{(0)}, \mathbf{b}_i^{(0)} \rangle] Q \tag{15}$$

On the other hand we have an expression for P_1 from (10):

$$P_1 = \langle \mathbf{a}^{(1)}, \mathbf{G}^{(1)} \rangle + \langle \mathbf{b}^{(1)}, \mathbf{H}^{(1)} \rangle + [\langle \mathbf{a}^{(1)}, \mathbf{b}^{(1)} \rangle] Q \tag{16}$$

5. Asserting equality (15)=(16) the extractor $\mathcal{E}'_{ip}$ sets:

$$\mathbf{a}^{(1)} = \sum_{i=1}^{3} v_i x_i^{-1} \cdot \mathbf{a}_i^{(0)} + v_i x_i \cdot \mathbf{a}_i^{(0)} \tag{17}$$

$$\mathbf{b}^{(1)} = \sum_{i=1}^{3} v_i x_i \cdot \mathbf{b}_i^{(0)} + v_i x_i^{-1} \cdot \mathbf{b}_i^{(0)} \tag{18}$$

We need the fourth rewinding to assert equality of inner product:

$$\langle \mathbf{a}^{(1)}, \mathbf{b}^{(1)} \rangle = \sum_{i=1}^{3} v_i \langle \mathbf{a}_i^{(0)}, \mathbf{b}_i^{(0)} \rangle$$

We won't describe it fully since it takes some unwieldy technical details and refer a reader to the original *bulletproofs* paper [**?**], where the full proof of extraction is described in Theorem 1.

6. The extractor $\mathcal{E}'_{ip}$ recursively extracts $\mathbf{a}^{(k+1)}, \mathbf{b}^{(k+1)}$ from $\mathbf{a}^{(k)}, \mathbf{b}^{(k)}$ using the method described in steps 2-5 until it reaches the final witness $\mathbf{a}^{(d)}, \mathbf{b}^{(d)} = \mathbf{a}, \mathbf{b}$ for which the relation $\mathcal{R}'_{ip}$ holds:

$$P' = \langle \mathbf{a}, \mathbf{G} \rangle + \langle \mathbf{b}, \mathbf{H} \rangle + [\langle \mathbf{a}, \mathbf{b} \rangle] Q$$

Building the extractor $\mathcal{E}_{ip}$ for Π_{ip} is quite simple relatively to what we've done by now:

1. $\mathcal{E}_{ip}$ runs Π_{ip} to the end and applies the extractor $\mathcal{E}'_{ip}$ for Pi'_{ip} to extract the witness $\mathbf{a}, \mathbf{b}$ such that the following holds for $\mathcal{V}$'s challenge $r_1 \in \mathbb{F}_p$:

$$P + [r_1 c]B = \langle \mathbf{a}, \mathbf{G} \rangle + \langle \mathbf{b}, \mathbf{H} \rangle + [r_1 \cdot \langle \mathbf{a}, \mathbf{b} \rangle] B \tag{19}$$

2. $\mathcal{E}_{ip}$ rewinds the prover $\mathcal{P}$ to obtain another challenge from $\mathcal{V}$: $r_2 \in \mathbb{F}_p$ and yield another witness $\mathbf{a}', \mathbf{b}'$ from $\mathcal{E}'_{ip}$:

$$P + [r_2 c]B = \langle \mathbf{a}', \mathbf{G} \rangle + \langle \mathbf{b}', \mathbf{H} \rangle + [r_2 \cdot \langle \mathbf{a}', \mathbf{b}' \rangle] B \tag{20}$$

3. $\mathcal{E}_{ip}$ substitute (19) from (20) to get:

$$[c(r_1 - r_2)]B = \langle \mathbf{a} - \mathbf{a}', \mathbf{G} \rangle + \langle \mathbf{b} - \mathbf{b}', \mathbf{H} \rangle + [r_1 \cdot \langle \mathbf{a}, \mathbf{b} \rangle - r_2 \cdot \langle \mathbf{a}', \mathbf{b}' \rangle] B \tag{21}$$

   Unless $a = a'$ and $b = b'$ we get a non-trivial discrete log relation between $\mathbf{G}, \mathbf{H}$ and B, otherwise if equality of witnesses holds we get:

$$[(r_1 - r_2)c]B = [(r_1 - r_2)\langle \mathbf{a}, \mathbf{b} \rangle] B \tag{22}$$

   Hence $c = \langle \mathbf{a}, \mathbf{b} \rangle$.

To formally finalize a proof of *witness-extended emulation* we also need to apply so-called *the forking lemma*, we again refer a reader to the original *bulletproofs* paper [**?**] □.

18.4.3 Zero-knowledge extension of inner-product argument

The main approach to make *inner-product argument zero-knowledge* is to use Π_{zkip} protocol which original bulletproofs [**?**] does for *range proofs*(Section 18.6) and *arithmetic circuits satisfiability*(Section 18.7). However, there exists an alternative elegant construction based on tweaking *inner-product argument* itself described in [**?**, Appendix E.2].

The key idea is to bring a blinding factor $\delta \in \mathbb{F}_p$ with a verifier-provided random element $D \in \mathbb{G}$ to the commitment P_k:

$$P_k = \langle \mathbf{a}^{(k)}, \mathbf{G}^{(k)} \rangle + \langle \mathbf{b}^{(k)}, \mathbf{H}^{(k)} \rangle + [\langle \mathbf{a}^{(k)}, \mathbf{b}^{(k)} \rangle] Q + [\delta^{(k)}] D$$

To get L_k, R_k prover draws $\delta_L^{(k)}, \delta_R^{(k)} \leftarrow\!\!\$\ \mathbb{F}_p$ and sets:

$$\begin{aligned}
P_{k-1} &= P_k + [u_k^2] L_k + [u_k^{-2}] R_k \\
L_k &= \langle \mathbf{a_{lo}}^{(k)}, \mathbf{G_{hi}}^{(k)} \rangle + \langle \mathbf{b_{hi}}^{(k)}, \mathbf{H_{lo}}^{(k)} \rangle + [\langle \mathbf{a_{lo}}^{(k)}, \mathbf{b_{hi}}^{(k)} \rangle] Q + [\delta_L^{(k)}] D \\
R_k &= \langle \mathbf{a_{hi}}^{(k)}, \mathbf{G_{lo}}^{(k)} \rangle + \langle \mathbf{b_{lo}}^{(k)}, \mathbf{H_{hi}}^{(k)} \rangle + [\langle \mathbf{a_{hi}}^{(k)}, \mathbf{b_{lo}}^{(k)} \rangle] Q + [\delta_R^{(k)}] D
\end{aligned}$$

Then $\mathcal{P}$ updates next-round δ using verifier-provided challenge u_k:

$$\delta^{(k-1)} = \delta^{(k)} + u_k^2 \delta_L^{(k)} + u_k^{-2} \delta_R^{(k)}$$

On the last step prover must prove that he possesses $a = a^{(0)}, b = b^{(0)}, \delta = \delta^{(0)}$ such that for $G = G^{(0)}, H = H^{(0)}$ equality holds:

$$P_0 = P_k + \sum_{i=1}^{k} ([u_i^2]L_i + [u_i^{-2}]R_i) = [a]G + [b]H + [a \cdot b]Q + [\delta]D$$

$P_0 = P_k + \sum_{i=1}^{k} ([u_i^2]L_i + [u_i^{-2}]R_i)$ is a Pedersen commitment to $a, b, a \cdot b$ with blinding factor δ and could easily be computed by verifier. One could prove knowledge of an opening (a, b, δ) to that commitment using protocol similar to Π_{zkip}.

1. $\mathcal{P}$ draws blinders $u, v, r, s \leftarrow\!\!\$\ \mathbb{F}_p$ and sends to $\mathcal{V}$ commitments:

$$A = [r]G + [s]H + [as + br]Q + [u]D$$
$$T = [rs]Q + [v]D$$

2. $\mathcal{V}$ draws challenge $c \leftarrow\!\!\$\ \mathbb{F}_p$ and sends it to $\mathcal{P}$
3. $\mathcal{P}$ evaluates and sends to $\mathcal{V}$:

$$a' = a + rc$$
$$b' = b + sc$$
$$\delta' = \delta + uc + vc^2$$

4. $\mathcal{V}$ performs check:

$$[a']G + [b']H + [a'b']Q + [\delta']D \stackrel{?}{=} P_0 + [c]A + [c^2]T$$

Protocol is *knowledge-sound* as value δ could easily be extracted from three accepting transcripts. To argue *zero-knowledge* we stress that an adversary couldn't learn anything from transcript and each prover's message (L_k, R_k) could be easily simulated by random element, at the base of recursion simulator simulates zero-knowledge proof of opening (a, b, δ) to commitment P_0.

18.5 Inner-product based polynomial commitment scheme

Here we describe one of the main theoretical applications of the *inner-product argument* – **polynomial commitment scheme** that relies only on *discrete logarithm* assumption, while studied before *KZG* commitment scheme needs bilinear pairings.

Definition 18.9. The inner-product non-hiding polynomial commitment scheme $\mathcal{C}_{ip} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{VerifyOpen})$ is defined as follows. Let $f(x) = \sum_{i=0}^{n-1} a_i x^i \in \mathbb{F}_p[x]$ be a polynomial of degree $n-1$.

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

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

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

Remark. As the second vector $\mathbf{b} = \mathbf{u^n}$ is known to the verifier, the prover don't have to commit to it using vector $\mathbf{H}$, so the parties might adjust all the steps eliminating vector $\mathbf{H}$ and $\mathbf{b}$ vector compression as well. The full scheme is described here. Note that described scheme is not zero-knowledge as classic *inner-product* argument is not zero-knowledge. But we could make it zero-knowledge using Π_{zkip} protocol or using alternative construction from Section 18.4.3.

18.6 Range proofs

Let $G, B \in \mathbb{G}$ – independent group generators. Let's consider the relation:

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

This relation is often called the **range proof** relation. It asserts that committed value v lays in the interval $[0, 2^n)$. Range proofs have very significant applications in various privacy *blockchain* protocols since them usually imply proving that transaction inputs or outputs are valid, e.g. have positive value or satisfy other relations between them.

For the first view it seems very inconspicuous why **inner-product argument** is useful for proving the range proof relation, but we'll show it ab initio.

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

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

We already know how to prove the second one inner product equality – simply

by taking evaluation point $u \leftarrow 2$ in *inner-product based polynomial commitment scheme*.

The first relation is a bit more tricky to check algebraically, but still we'll manage to do that, note that binary check for v_i takes form $v_i(v_i - 1) = 0$, or in vector form:

$$\begin{aligned} \mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n \Leftrightarrow \mathbf{a}_L - \mathbf{a}_R - \mathbf{1}^n = \mathbf{0}^n \\ \mathbf{a}_L \circ \mathbf{a}_R = \mathbf{0}^n \end{aligned}$$

> **Example 18.2.** Let $\mathbf{a}_L = (1, 0, 1, 0)$, $\mathbf{a}_R = (0, -1, 0, -1)$, then $\mathbf{a}_L \circ \mathbf{a}_R = (0, 0, 0, 0)$

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

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

This checks are sound because the prover doesn't know challenge y in advance.

Note that $\langle \mathbf{a}_L \circ \mathbf{a}_R, \mathbf{y}^n \rangle = \langle \mathbf{a}_L, \mathbf{a}_R \circ \mathbf{y}^n \rangle$ so the prover could commit to $\mathbf{a_L}, \mathbf{a_R}$ and verifier will adjust commitment for $\mathbf{a}_R$ using modified generators $\mathbf{H} \circ \mathbf{y}^{-n}$.

Here we came up with three inner-product checks:

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

Here we could soundly combine all three checks into one using random linear combination with some verifier-provided challenge $z \in \mathbb{F}_p$:

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

> **Remark.** Naïve check $\langle \mathbf{a}_L, \mathbf{2}^n \rangle + \langle \mathbf{a}_L, \mathbf{a}_R \circ \mathbf{y}^n \rangle + \langle \mathbf{a}_L - \mathbf{a}_R - \mathbf{1}^n, \mathbf{y}^n \rangle = v$ is not sound as Prover could adjust vectors to be non-zero but still satisfy the check.

Now simplify this expression having only one inner-product check:

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

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

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

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

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

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

Now adding to both sides $\langle z^2 \cdot \mathbf{2}^n + z \cdot \mathbf{y}^n, -z \cdot \mathbf{1}^n \rangle$:

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

$$= z^2 v + \langle z \cdot \mathbf{1}^n, \mathbf{y}^n \rangle + \boxed{\langle z^2 \cdot \mathbf{2}^n + z \cdot \mathbf{y}^n, -z \cdot \mathbf{1}^n \rangle}$$

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

$$= z^2 v + \boxed{\langle z \cdot \mathbf{1}^n, \mathbf{y}^n \rangle + \langle z^2 \cdot \mathbf{2}^n + z \cdot \mathbf{y}^n, -z \cdot \mathbf{1}^n \rangle}$$

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

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

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

Now we have only one inner-product check left:

$$\langle \mathbf{a}_L - z \cdot \mathbf{1}^n, z^2 \cdot \mathbf{2}^n + z \cdot \mathbf{y}^n + \mathbf{a}_R \circ \mathbf{y}^n \rangle = z^2 v + \delta(y, z) \tag{23}$$

We will use a technique presented in Π_{zkip} to provide zero-knowledge and **inner-product argument** to achieve logarithmic size-proof. One key problem is that the verifier must adjust commitments to compensate auxiliary terms.

Firstly, construct the blinding polynomials for $\mathbf{a}_L$ and $\mathbf{a}_R$ with substitution:

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

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

$$\begin{aligned}
\mathbf{l}(x) &= \mathbf{a}'_L - z \cdot \mathbf{1}^n = (\mathbf{a}_L + \mathbf{s}_L x) - z \cdot \mathbf{1}^n = \mathbf{a}_L - z \cdot \mathbf{1}^n + \mathbf{s}_L x \\
\mathbf{r}(x) &= z^2 \cdot \mathbf{2}^n + z \cdot \mathbf{y}^n + \mathbf{a}'_R \circ \mathbf{y}^n = z^2 \cdot \mathbf{2}^n + z \cdot \mathbf{y}^n + (\mathbf{a}_R + \mathbf{s}_R x) \circ \mathbf{y}^n \\
&= z^2 \cdot \mathbf{2}^n + z \cdot \mathbf{y}^n + \mathbf{a}_R \circ \mathbf{y}^n + \mathbf{s}_R \circ \mathbf{y}^n x
\end{aligned}$$

So that $\langle \mathbf{l}_0, \mathbf{r}_0 \rangle = z^2 v + \delta(y, z)$ – inner product that we want to prove using a bit modified Π_{zkip}.

Definition 18.10. The **range proof protocol** $\Pi_{rp} = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ for the relation $\mathcal{R}_{rp} = \{(G, B, V, n; v, \gamma) | V = [v]G + [\gamma]B, v \in [0, 2^n)\}$ with prover $\mathcal{P}$ and verifier $\mathcal{V}$ is defined as follows:

- Setup returns vectors of group generators with unknown discrete log relations $\mathbf{G}, \mathbf{H} \in \mathbb{G}^n$
- Prover does bit decomposition of v to obtain vectors $\mathbf{a}_L \leftarrow \mathbf{v}, \mathbf{b_L} \leftarrow \mathbf{a}_L - \mathbf{1}^n$ and choses blinding terms $\mathbf{s}_L, \mathbf{s}_R \in \mathbb{F}_p^n, \alpha, \beta \in \mathbb{F}_p$ computing

and sending commitments:

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

- Verifier $\mathcal{V}$ samples challenges $y, z \leftarrow\!\!\$\ \mathbb{F}_p$ and sends them to $\mathcal{P}$
- Prover $\mathcal{P}$ combines the three inner products into one using provided challenges y, z (23) and reconstructs vector polynomials $\mathbf{l}(x), \mathbf{r}(x), \mathbf{t}(x)$:

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

Where

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

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

$$\begin{aligned}
T_1 &= [t_1]G + [\tau_1]B \\
T_2 &= [t_2]G + [\tau_2]B
\end{aligned} \tag{24}$$

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 $\mathcal{P}$ random evaluation point $u \leftarrow\!\!\$\ \mathbb{F}_p$
- Prover $\mathcal{P}$ evaluates polynomials at u:

$$\begin{aligned}
\mathbf{l}_u &= \mathbf{l}(u) & \alpha_u &= \alpha + \beta u \\
\mathbf{r}_u &= \mathbf{r}(u) & \tau_u &= z^2 \gamma + \tau_1 u + \tau_2 u^2 \\
t_u &= t(u) = z^2 v + \delta(y, z) + t_1 u + t_2 u^2
\end{aligned} \tag{25}$$

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

- Verifier $\mathcal{V}$ performs checks:

$$\begin{aligned}
A + [u]S + \langle -z \cdot \mathbf{1}^n, \mathbf{G} \rangle + \langle z \cdot \mathbf{y}^n + z^2 \cdot \mathbf{2}^n, \mathbf{y}^{-n} \circ \mathbf{H} \rangle & \\
\stackrel{?}{=} \langle \mathbf{l}_u, \mathbf{G} \rangle + \langle \mathbf{r}_u, \mathbf{y}^{-n} \circ \mathbf{H} \rangle + [\alpha_u]B & \\
[t_u]G + [\tau_u]B \stackrel{?}{=} [z^2]V + [\delta(y, z)]G + [u]T_1 + [u^2]T_2 & \\
t_u \stackrel{?}{=} \langle \mathbf{l}_u \mathbf{r}_u \rangle &
\end{aligned} \tag{26}$$

Remark. The last two steps of Π_{rp} could be substituted with an inner-product argument Π_{ip} to provide logarithmic size-proof with the following steps:

- After $\mathcal{P}$ evaluates polynomials at u he sends $(t_u, \alpha_u, \tau_u$ to $\mathcal{V})$ and computes commitment:

$$P = \langle \mathbf{l}_u, \mathbf{G} \rangle + \langle \mathbf{r}_u, \mathbf{y}^{-n} \circ \mathbf{H} \rangle$$

- $\mathcal{V}$ performs check $[t_u]G + [\tau_u]B \stackrel{?}{=} [z^2]V + [\delta(y,z)]G + [u]T_1 + [u^2]T_2$, halts if it fails and reconstructs commitment P otherwise:

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

- Parties run inner-product argument Π_{ip} on $(\mathbf{G}, \mathbf{y}^{-n} \circ \mathbf{H}, P, t_u; \mathbf{l}_u, \mathbf{r}_u)$

Theorem 18.11. The **range proof protocol** Π_{rp} has *perfect completeness, computational extended witness emulation, perfect honest-verifier zero-knowledge*

Proof idea. *Perfect completeness*
Expand LHS of the polynomial correctness check:

$$\begin{aligned}
&A + [u]S + \langle -z \cdot \mathbf{1}^n, \mathbf{G} \rangle + \langle z \cdot \mathbf{y}^n + z^2 \cdot \mathbf{2}^n, \mathbf{y}^{-n} \circ \mathbf{H} \rangle = \\
&\langle \mathbf{a}_L, \mathbf{G} \rangle + \langle \mathbf{b_L}, \mathbf{H} \rangle + [\alpha]B + u \cdot \langle \mathbf{s}_L, \mathbf{G} \rangle + u \cdot \langle \mathbf{s}_R, \mathbf{H} \rangle + [u\beta]B + \\
&\langle -z \cdot \mathbf{1}^n, \mathbf{G} \rangle + \langle z \cdot \mathbf{y}^n + z^2 \cdot \mathbf{2}^n, \mathbf{y}^{-n} \circ \mathbf{H} \rangle = \\
&\langle \mathbf{a}_L, \mathbf{G} \rangle + \langle \mathbf{b_L}, \mathbf{H} \rangle + u \cdot \langle \mathbf{s}_L, \mathbf{G} \rangle + u \cdot \langle \mathbf{s}_R, \mathbf{H} \rangle + \\
&\langle -z \cdot \mathbf{1}^n, \mathbf{G} \rangle + \langle z \cdot \mathbf{1}^n, \mathbf{H} \rangle + \langle z^2 \cdot \mathbf{2}^n, \mathbf{y}^{-n} \circ \mathbf{H} \rangle + [\alpha + u\beta]B
\end{aligned}$$

Expand RHS of the polynomial correctness check:

$$\begin{aligned}
&\langle \mathbf{l}_u, \mathbf{G} \rangle + \langle \mathbf{r}_u, \mathbf{y}^{-n} \circ \mathbf{H} \rangle + [\alpha_u]B = \\
&\langle \mathbf{a}_L - z \cdot \mathbf{1}^n + \mathbf{s}_L \cdot u, \mathbf{G} \rangle + \langle z^2 \cdot \mathbf{2}^n + z \cdot \mathbf{y}^n + \mathbf{a}_R \circ \mathbf{y}^n + \mathbf{s}_R \circ \mathbf{y}^n \cdot u, \mathbf{y}^{-n} \circ \mathbf{H} \rangle \\
&+ [\alpha + u\beta]B = \\
&\langle \mathbf{a}_L, \mathbf{G} \rangle + \langle -z \cdot \mathbf{1}^n, \mathbf{G} \rangle + u \cdot \langle \mathbf{s}_L, \mathbf{G} \rangle + \langle z^2 \cdot \mathbf{2}^n, \mathbf{y}^{-n} \circ \mathbf{H} \rangle + \langle z \cdot \mathbf{1}^n, \mathbf{H} \rangle + \\
&\langle \mathbf{a}_R, \mathbf{H} \rangle + u \cdot \langle \mathbf{s}_R, \mathbf{H} \rangle + [\alpha + u\beta]B
\end{aligned}$$

We could see that *LHS* is equal to *RHS* so the first check pass.
Taking the t_0 correctness check:

$$\begin{aligned}
[t_u]G + [\tau_u]B &\stackrel{?}{=} [z^2]V + \delta(y,z) + [u]T_1 + [u^2]T_2 \\
[t_u]G + \cancel{[z^2\gamma + \tau_1 u + \tau_2 u^2]B} &\stackrel{?}{=} [z^2 v]G + \cancel{[z^2\gamma]B} + [\delta(y,z)]G + \\
&\quad [ut_1]G + \cancel{[u\tau_1]B} + [t_2 u^2]G + \cancel{[\tau_2 u^2]B} \\
[t_u]G &\stackrel{?}{=} [z^2 v + \delta(y,z) + t_1 u + t_2 u^2]G
\end{aligned}$$

Which holds since $t_u = t(u) = z^2 v + \delta(y,z) + t_1 u + t_2 u^2$. The third inner-product also holds as $t_u = \langle \mathbf{l}_u \mathbf{r}_u \rangle$.

Perfect honest-verifier zero-knowledge follows from the zero-knowledge construction of Π_{zkip} protocol as Verifier learns no information about $\mathbf{a}_L, \mathbf{a}_R$.

Computational extended witness emulation implies building extractor that combines extractors for two subprotocols: $\mathcal{E}_{ip}$ extracts witness $(\mathbf{l}_u, \mathbf{r}_u)$ from Π_{ip} than the extractor $\mathcal{E}_{zkip}$ extracts high-level witness $(\mathbf{a}_L, \mathbf{a}_R)$ from Π_{zkip} □

The proof size of **range-proof** protocol is $2\log_2 n + 4$ group $\mathbb{G}$ elements and 5 field $\mathbb{F}_p$ elements.

Remark. Range proofs could be efficiently aggregated: e.g. one could prove the relation using slightly modified range proof protocol Π_{rp}

$$\mathcal{R}_{rpm} = \{(G, B, \vec{V}, n; \vec{v}, \vec{\gamma}) | \forall i \in 1..m : V_i = [v_i]G + [\gamma_i]B, v_i \in [0, 2^n)\}$$

Where $\vec{v} = (v_1, v_2, \ldots, v_m)$ and $\vec{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_m)$ – respectively secrets and blinding factors. Detail explanation of aggregation protocol could be found in original bulletproofs paper [**?**]

Example 18.3. One of the most famous *NP-complete* problems is the **subset-sum problem**: given a set of numbers presented as vector $\mathbf{s}$ 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 $\langle \mathbf{a}_L, \mathbf{2}^n \rangle = v$ with $\langle \mathbf{a}_L, \mathbf{s} \rangle = v$ where $\mathbf{a}_L$ is the secret vector of bits that encode positions of $\mathbf{s}$ that sum up to v.
For example take $\mathbf{s} = (6, 8, 2, 3)$ and $v = 14$. Then setting $\mathbf{a}_L = (1, 1, 0, 0)$ we could use Π_{rp} to prove that there exists a subset of $\mathbf{s}$ 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**

18.7 Arithmetic circuits proofs

Bulletproofs presents not only range proofs, but also efficient proofs for arithmetic circuits satisfiability. As we could see before, inner-product relation is quite powerful tool and could be used to prove a knowledge of witness to any NP-problem. But here we present a more convenient way to compile arithmetic circuits into inner-product relation.

18.7.1 Arithmetization

Bulletproofs arithmetization slightly differs from the classic R1CS we studied before at Section 13, 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.

There are two types of variables in *bulletproofs* constraint system: *low-level* and *high-level*. Typically *high-level* variables are provided with Pedersen commitments $\mathbf{V} \in \mathbb{G}^m$ as the private witness inputs $\mathbf{v} \in \mathbb{F}_p^m$ to the circuit, while *low-level* variables $\mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O \in \mathbb{F}_p^n$ 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.

Multiplication constraints are defined with one vector equation:

$$\mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O$$

Linear constraints are defined via:

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

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

Example 18.4. 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:

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

High-level variables:

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

Multiplication constraints:

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

Linear constraints:

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

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

Where:

$$
\mathbf{W}_L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ -a & 0 & 0 \end{bmatrix}, \quad \mathbf{W}_R = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{W}_O = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 1 \end{bmatrix},
$$

$$
\mathbf{W}_V = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ b \end{bmatrix}
$$

Classic R1CS arithmetization:
We can also represent the same circuit using the classic R1CS formalism:

$$
(A \cdot \mathbf{w}) \circ (B \cdot \mathbf{w}) = (C \cdot \mathbf{w})
$$

where $\mathbf{w}$ is extended witness vector $\mathbf{w} = (1, x, y, x^2, x^3, y^2)$
The R1CS constraints for the circuit $y^2 = x^3 + ax + b$ are:

$$
\begin{aligned}
&\text{Constraint 1:} \quad x \times x = x^2 \\
&\text{Constraint 2:} \quad x^2 \times x = x^3 \\
&\text{Constraint 3:} \quad y \times y = y^2 \\
&\text{Constraint 4:} \quad x^3 + ax + b = y^2
\end{aligned}
$$

Explicitly, the R1CS matrices are:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & a & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

18.7.2 Proving a circuit satisfiability

In this section we build an argument system for the following arithmetic circuit satisfiability relation:

$$\mathcal{R}_{sat} = \left\{ \begin{array}{l} (G, B, \mathbf{V}, \mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O, \mathbf{W}_V, \mathbf{c}; \mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O, \mathbf{v}, \mathbf{r}) | \\ \forall i = 1..m : V_i = [v_i]G + [r_i]B \wedge \\ \mathbf{a}_L \circ \mathbf{a}_R = \mathbf{a}_O \wedge \\ \mathbf{W}_L \cdot \mathbf{a}_L + \mathbf{W}_R \cdot \mathbf{a}_R + \mathbf{W}_O \cdot \mathbf{a}_O = \mathbf{W}_V \cdot \mathbf{v} + \mathbf{c} \end{array} \right\} \tag{27}$$

Where $\mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O \in \mathbb{F}_p^n$, $\mathbf{v}, \mathbf{r} \in \mathbb{F}_p^m$, $\mathbf{W}_L, \mathbf{W}_R, \mathbf{W}_O \in \mathbb{F}_p^{q \times n}$, $\mathbf{W}_V \in \mathbb{F}_p^{q \times m}$, $\mathbf{c} \in \mathbb{F}_p^q$. Informally this relation states that there exists a valid witness $\mathbf{v}$ that satisfies all constraints of the circuit. For the verifier witness is presented only as commitments vector $\mathbf{V}$.

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:

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

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

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

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

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

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

Then split the second inner product and factor-out public terms to RHS:

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

Applying conjugation rule(if A – linear operator and A^T – its transpose(conjugate) then $\langle \mathbf{a}, A\mathbf{b} \rangle = \langle A^T \mathbf{a}, \mathbf{b} \rangle$) to the second inner product we get:

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

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

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

Now we could finally rewrite the equation as:

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

Then rearrange and combine terms so that $\mathbf{a}_L, \mathbf{a}_O$ be on the left side and $\mathbf{a}_R$ on the right side:

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

Add $\delta(y, z) = \langle \mathbf{y}^{-n} \circ \mathbf{w}_R, \mathbf{w}_L \rangle$ to both sides:

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

Now it seems we are stuck as there is two separate inner products so one could not simply linearly blind witness parts, multiply corresponding polynomials and

obtain desired inner product in constant term just as we did in the *range-proof* case. But fortunately we could take polynomials of higher degree and obtain desired sum of inner-products as some coefficient of the product of polynomials:

$$\langle \mathbf{a}x+\mathbf{c}x^2, \mathbf{d}+\mathbf{b}x\rangle = s_1x+s_2x^2+s_3x^3 = x\cdot\langle \mathbf{a},\mathbf{d}\rangle+x^2\cdot(\langle \mathbf{a},\mathbf{b}\rangle+\langle \mathbf{c},\mathbf{d}\rangle)+x^3\cdot\langle \mathbf{c},\mathbf{b}\rangle \tag{28}$$

So take:

$$\begin{aligned} &\mathbf{a} \leftarrow \mathbf{a}_L + \mathbf{y}^{-n} \circ \mathbf{w}_R \quad && \mathbf{b} \leftarrow \mathbf{y}^n \circ \mathbf{a}_R + \mathbf{w}_L \\ &\mathbf{c} \leftarrow \mathbf{a}_O && \mathbf{d} \leftarrow -\mathbf{y}^n + \mathbf{w}_O \end{aligned} \tag{29}$$

So than we could get desired sum of inner products as the second-degree coefficient s_2:

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

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

$$\mathbf{a}_L \leftarrow \mathbf{a}_L + \mathbf{s}_L x^2 \quad \mathbf{a}_R \leftarrow \mathbf{a}_R + \mathbf{s}_R x^2 \tag{30}$$

> **Remark.** We multiplied blinders $\mathbf{s}_L, \mathbf{s}_R$ with the second power of challenge x^2 because we want the blinding terms to do not interfere with other parts of inner-product.
> $\mathbf{a}_O$ does not need separate blinding as it's located on the left side of the inner-product (28) along with $\mathbf{a}_L$, which is already blinded by $\mathbf{s}_L$.

Now we could compute polynomials $\mathbf{l}(x), \mathbf{r}(x)$ from (28) using assignments from (29) and blinders from (30):

$$\begin{aligned} \mathbf{l}(x) &= (\mathbf{a}_L + \mathbf{s}_L \cdot x^2) \cdot x + \mathbf{y}^{-n} \circ \mathbf{w}_R \cdot x + \mathbf{a}_O \cdot x^2 \\ &= \mathbf{a}_L \cdot x + \mathbf{s}_L \cdot x^3 + \mathbf{y}^{-n} \circ \mathbf{w}_R \cdot x + \mathbf{a}_O \cdot x^2 \\ &= \mathbf{s}_L \cdot x^3 + \mathbf{a}_O \cdot x^2 + (\mathbf{a}_L + \mathbf{y}^{-n} \circ \mathbf{w}_R) \cdot x \\ \mathbf{r}(x) &= \mathbf{y}^n \circ (\mathbf{a}_R + \mathbf{s}_R \cdot x^2) \cdot x + \mathbf{w}_L \cdot x - \mathbf{y}^n + \mathbf{w}_O \\ &= \mathbf{y}^n \circ \mathbf{a}_R \cdot x + \mathbf{y}^n \circ \mathbf{s}_R \cdot x^3 + \mathbf{w}_L \cdot x - \mathbf{y}^n + \mathbf{w}_O \\ &= \mathbf{y}^n \circ \mathbf{s}_R \cdot x^3 + (\mathbf{y}^n \circ \mathbf{a}_R + \mathbf{w}_L) \cdot x - \mathbf{y}^n + \mathbf{w}_O \end{aligned} \tag{31}$$

Now we could compute the polynomial $t(x) = \langle \mathbf{l}(x), \mathbf{r}(x) \rangle$:

$$t(x) = \langle \mathbf{l}(x), \mathbf{r}(x) \rangle = t_1 x + t_2 x^2 + t_3 x^3 + t_4 x^4 + t_5 x^5 + t_6 x^6 = \sum_{i=1}^{6} t_i x^i$$

$$\begin{aligned}
t_1 &= \langle \mathbf{a}_L + \mathbf{y}^{-n} \circ \mathbf{w}_R, -\mathbf{y}^n + \mathbf{w}_O \rangle \\
t_2 &= \langle \mathbf{a}_L + \mathbf{y}^{-n} \circ \mathbf{w}_R, \mathbf{y}^n \circ \mathbf{a}_R + \mathbf{w}_L \rangle + \langle \mathbf{a}_O, -\mathbf{y}^n + \mathbf{w}_O \rangle \\
&= \langle \mathbf{a}_L \circ \mathbf{a}_R, \mathbf{y}^n \rangle - \langle \mathbf{a}_O, \mathbf{y}^n \rangle + \langle \mathbf{w}_L, \mathbf{a}_L \rangle + \langle \mathbf{w}_R, \mathbf{a}_R \rangle + \langle \mathbf{w}_O, \mathbf{a}_O \rangle + \delta(y, z) \\
&= w_c + \langle \mathbf{w}_V, \mathbf{v} \rangle + \delta(y, z) \\
t_3 &= \langle \mathbf{s}_L, -\mathbf{y}^n + \mathbf{w}_O \rangle + \langle \mathbf{a}_O, \mathbf{y}^n \circ \mathbf{a}_R + \mathbf{w}_L \rangle \\
t_4 &= \langle \mathbf{s}_L, \mathbf{y}^n \circ \mathbf{a}_R + \mathbf{w}_L \rangle + \langle \mathbf{a}_L + \mathbf{y}^{-n} \circ \mathbf{w}_R, \mathbf{y}^n \circ \mathbf{s}_R \rangle \\
t_5 &= \langle \mathbf{a}_O, \mathbf{y}^n \circ \mathbf{s}_R \rangle \\
t_6 &= \langle \mathbf{s}_L, \mathbf{y}^n \circ \mathbf{s}_R \rangle
\end{aligned} \tag{32}$$

Our proving strategy is the same as in the *range-proof* case:

- Prove that $\mathbf{l}(x), \mathbf{r}(x)$ are correct using binding commitments to their coefficients.
- Prove that t_2 is correct (as it's the inner-product we want to prove) using evaluation at challenge point u.
- Apply inner-product argument to compress the proof of evaluation of $t(x)$ at challenge point u.

Definition 18.12. The **arithmetic circuit satisfiability** protocol $\Pi_{sat} = (\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ for the relation $\mathcal{R}_{sat}$ with prover $\mathcal{P}$ and verifier $\mathcal{V}$ is defined as follows:

- Setup: returns vector of group generators with unknown discrete log relations $\mathbf{G}, \mathbf{H} \in \mathbb{G}^n$.
- Prover $\mathcal{P}$ choses blinding factors $\alpha, \beta, \gamma \in \mathbb{F}_p, \mathbf{s}_L, \mathbf{s}_R \in \mathbb{F}_p^n$ and sends the following commitments to $\mathcal{V}$:

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

- Verifier samples challenges $y, z \leftarrow\!\!\$\ \mathbb{F}_p$ and sends them to $\mathcal{P}$.
- Using provided challenges y, z prover forms polynomials $\mathbf{l}(x), \mathbf{r}(x)$:

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

  computes

$$t(x) = \langle \mathbf{l}(x), \mathbf{r}(x) \rangle = t_1 x + t_2 x^2 + t_3 x^3 + t_4 x^4 + t_5 x^5 + t_6 x^6$$

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:

$$
\begin{aligned}
T_1 &= [t_1]G + [\tau_1]B \\
T_3 &= [t_3]G + [\tau_3]B \\
T_4 &= [t_4]G + [\tau_4]B \\
T_5 &= [t_5]G + [\tau_5]B \\
T_6 &= [t_6]G + [\tau_6]B
\end{aligned}
$$

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

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

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

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

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

- Verifier performs checks:

$$
\begin{aligned}
&[u]A_I + [u^2]A_O + [u^3]S - \langle \mathbf{1}, \mathbf{H} \rangle + \\
&u \cdot (\langle \mathbf{y}^{-n} \circ \mathbf{w}_L, \mathbf{G} \rangle + \langle \mathbf{y}^{-n} \circ \mathbf{w}_R, \mathbf{H} \rangle) + \langle \mathbf{y}^{-n} \circ \mathbf{w}_O, \mathbf{H} \rangle \\
&\stackrel{?}{=} \langle \mathbf{l}_u, \mathbf{G} \rangle + \langle \mathbf{r}_u, \mathbf{y}^{-n} \circ \mathbf{H} \rangle + [\alpha_u]B \\
[t_u]G + [\tau_u]B &\stackrel{?}{=} [u]T_1 + u^2 \cdot (\langle \mathbf{w}_V, \mathbf{V} \rangle + [\delta(y, z) + w_c]G) + \\
&[u^3]T_3 + [u^4]T_4 + [u^5]T_5 + [u^6]T_6 \\
t_u &\stackrel{?}{=} \langle \mathbf{l}_u, \mathbf{r}_u \rangle
\end{aligned}
$$

Remark. As in the *range proof* protocol case the last two steps of Π_{sat} could be substituted with an inner-product argument Π_{ip} to provide logarithmic size-proof with the following steps:

- After $\mathcal{P}$ evaluates polynomials at u he sends $(t_u, \alpha_u, \tau_u$ to $\mathcal{V})$ and

computes commitment:

$$P = \langle \mathbf{l}_u, \mathbf{G} \rangle + \langle \mathbf{r}_u, \mathbf{y}^{-n} \circ \mathbf{H} \rangle$$

- $\mathcal{V}$ performs check:

$$[t_u]G + [\tau_u]B \stackrel{?}{=} \sum_{k=1, k\neq 2}^{6} [u^k]T_k + u^2 \cdot (\langle \mathbf{w}_V, \mathbf{V} \rangle + [\delta(y,z) + w_c]G)$$

halts if it fails and reconstructs commitment P otherwise:

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

- Parties run inner-product argument Π_{ip} on $(\mathbf{G}, \mathbf{y}^{-n} \circ \mathbf{H}, P, t_u; \mathbf{l}_u, \mathbf{r}_u)$

Theorem 18.13. The **arithmetic circuit satisfiability protocol** Π_{sat} has *perfect completeness, computational extended witness emulation, perfect honest-verifier zero-knowledge*

Proof idea. *Perfect completeness*

- **Polynomial correctness check**:

$$\begin{aligned} LHS &= [u]A_I + [u^2]A_O + [u^3]S - \langle \mathbf{1}, \mathbf{H} \rangle + \\ &\quad u \cdot (\langle \mathbf{y}^{-n} \circ \mathbf{w}_L, \mathbf{G} \rangle + \langle \mathbf{y}^{-n} \circ \mathbf{w}_R, \mathbf{H} \rangle) + \langle \mathbf{y}^{-n} \circ \mathbf{w}_O, \mathbf{H} \rangle \\ &\quad = u \cdot \langle \mathbf{a}_L, \mathbf{G} \rangle + u \cdot \langle \mathbf{a}_R, \mathbf{H} \rangle + [\alpha u]B + u^2 \cdot \langle \mathbf{a}_O, \mathbf{G} \rangle + [\gamma u^2]B + \\ &\quad + u^3 \cdot \langle \mathbf{s}_L, \mathbf{G} \rangle + u^3 \cdot \langle \mathbf{s}_R, \mathbf{H} \rangle + [\beta u^3]B - \langle \mathbf{1}, \mathbf{H} \rangle + \\ &\quad + u \cdot \langle \mathbf{y}^{-n} \circ \mathbf{w}_L, \mathbf{G} \rangle + u \cdot \langle \mathbf{y}^{-n} \circ \mathbf{w}_R, \mathbf{H} \rangle + \langle \mathbf{y}^{-n} \circ \mathbf{w}_O, \mathbf{H} \rangle \\ RHS &= \langle \mathbf{l}_u, \mathbf{G} \rangle + \langle \mathbf{r}_u, \mathbf{y}^{-n} \circ \mathbf{H} \rangle + [\alpha_u]B \\ &\quad = u^3 \cdot \langle \mathbf{s}_L, \mathbf{G} \rangle + u^2 \cdot \langle \mathbf{a}_O, \mathbf{G} \rangle + u \cdot \langle \mathbf{a}_L + \mathbf{y}^{-n} \circ \mathbf{w}_R, \mathbf{G} \rangle + \\ &\quad + u^3 \cdot \langle \mathbf{y}^n \circ \mathbf{s}_R, \mathbf{y}^{-n} \circ \mathbf{H} \rangle + u \cdot \langle \mathbf{y}^n \circ \mathbf{a}_R, \mathbf{y}^{-n} \circ \mathbf{H} \rangle + u \cdot \langle \mathbf{w}_L, \mathbf{y}^n \circ \mathbf{H} \rangle \\ &\quad - \langle \mathbf{y}^n, \mathbf{y}^{-n} \circ \mathbf{H} \rangle + \langle \mathbf{w}_O, \mathbf{y}^{-n} \circ \mathbf{H} \rangle + [\alpha u + \gamma u^2 + \beta u^3]B \\ &\quad = u^3 \cdot \langle \mathbf{s}_L, \mathbf{G} \rangle + u^2 \cdot \langle \mathbf{a}_O, \mathbf{G} \rangle + u \cdot \langle \mathbf{a}_L, \mathbf{G} \rangle + u \cdot \langle \mathbf{y}^{-n} \circ \mathbf{w}_R, \mathbf{G} \rangle + \\ &\quad + u^3 \cdot \langle \mathbf{s}_R, \mathbf{H} \rangle + u \cdot \langle \mathbf{a}_R, \mathbf{H} \rangle + u \cdot \langle \mathbf{w}_L, \mathbf{y}^n \circ \mathbf{H} \rangle \\ &\quad - \langle \mathbf{1}^n, \mathbf{H} \rangle + \langle \mathbf{w}_O, \mathbf{y}^{-n} \circ \mathbf{H} \rangle + [\alpha u]B + [\gamma u^2]B + [\beta u^3]B \end{aligned}$$

As we see $LHS = RHS$ so the check holds.

- t_2 **correctness check**:

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

Note that

$$\begin{aligned}\langle \mathbf{w}_V, \mathbf{V} \rangle &= \sum_{i=1}^{m} [w_{V,i}] V_i = \sum_{i=1}^{n} ([w_{i,k} \cdot v_i] G + [w_{V,i} \cdot r_i] B) \\ &= [\sum_{i=1}^{m} w_{V,i} \cdot v_i] G + [\sum_{i=1}^{m} w_{V,i} \cdot r_i] B \\ &= [\langle \mathbf{w}_V, \mathbf{v} \rangle] G + [\langle \mathbf{w}_V, \mathbf{r} \rangle] B\end{aligned}$$

so simplifying the LHS we get:

$$\begin{aligned}&[t_u] G + [\tau_u] B = \\ &[t_1 u + (w_c + \langle \mathbf{w}_V, \mathbf{v} \rangle + \delta(y,z)) u^2 + t_3 u^3 + t_4 u^4 + t_5 u^5 + t_6 u^6] G + \\ &[\tau_1 u + \langle \mathbf{w}_V, \mathbf{r} \rangle u^2 + \tau_3 u^3 + \tau_4 u^4 + \tau_5 u^5 + \tau_6 u^6] B = \\ &[u] T_1 + u^2 \cdot (\langle \mathbf{w}_V, \mathbf{V} \rangle + [\delta(y,z) + w_c] G) + \\ &+ [u^3] T_3 + [u^4] T_4 + [u^5] T_5 + [u^6] T_6 = RHS \quad \square\end{aligned}$$

Perfect honest-verifier zero-knowledge follows from the zero-knowledge construction of Π_{zkip} protocol as Verifier learns no information about $\mathbf{a}_L, \mathbf{a}_R, \mathbf{v}$.

Computational extended witness emulation implies building extractor that combines extractors for two subprotocols: $\mathcal{E}_{ip}$ extracts witness $(\mathbf{l}_u, \mathbf{r}_u)$ from Π_{ip} than the extractor built on top of $\mathcal{E}_{zkip}$ extracts high-level witness $(\mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O, \mathbf{v})$ from Π_{zkip} $\square$

The proof size of **arithmetic circuit satisfiability** protocol is $2 \log_2 n + 8$ group $\mathbb{G}$ elements and 5 field $\mathbb{F}_p$ elements.

Usually the number of multipliers n is not a power of 2 so that the intermediate witness vectors $\mathbf{a}_L, \mathbf{a}_R, \mathbf{a}_O$ must be padded with zeroes to the next power of 2 in order to apply the *inner-product argument* efficiently.

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

Acknowledgements

This section was heavily inspired by:

- dalek's crate bulletproofs
- bulletproofs whitepaper
- RareSkills explanation