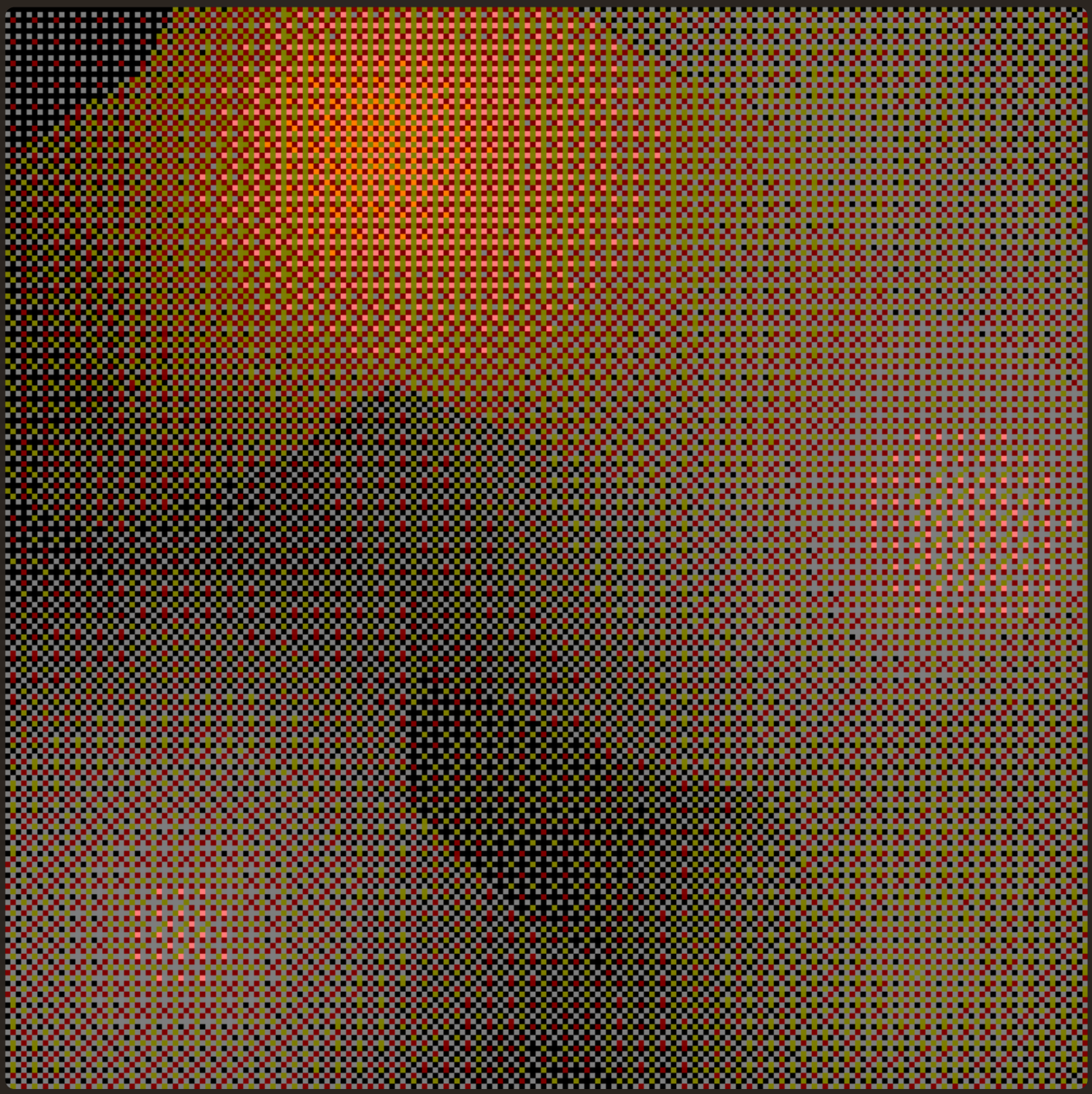
ZKSECURITY

Generalized Bulletproofs

MAGIC GRANTS • JUNE 1ST, 2026 • TECHNICAL REPORT



Introduction

From the 1st of June to the 12th of June (10 working days), two consultants from zkSecurity reviewed the `generalized-bulletproofs` crate inside the `monero-oxide` project, commit `f3660043e042b5608fc9c8ce1383cee238da9d0a`. This code is slated for use in the upcoming Monero 'fullchain membership proofs' (FCMP++) and a soundness/privacy critical component of the whole system. We reviewed the codebase as a general library, meaning, we did not discount findings simply because the current application does not trigger the issue.

The code was found to be well-written and easy to understand. The code was reviewed along with its associated paper, for context and because of some concerns about the formal writeup of the extractor (proof of knowledge soundness), we include here a writeup / motivation for the 'generalized bulletproofs' protocol as well as a formal proof of (computational) knowledge soundness.

Notation

Let $\mathbb{F}$ denote the scalar field and $\mathbb{G}$ a prime-order group (in practice a subgroup of an elliptic curve), written additively. Generators are denoted $G, H \in \mathbb{G}$ and $\mathbf{G}, \mathbf{H} \in \mathbb{G}^n$; commitments are capitalized ($V_{(k)}$, $A_C^{(i)}$) and blinding factors written as Greek letters (γ). We write $a \leftarrow S$ to denote sampling an element from the set S uniformly. An (honest) vector Pedersen commitment to $\mathbf{v}$ with randomness γ is $\langle \mathbf{v}, \mathbf{G} \rangle + \gamma \cdot H$, and a scalar commitment to v is $v \cdot G + \gamma \cdot H$.

Bold font denotes vectors, $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{F}^n$, with $\mathbf{1}$ and $\mathbf{0}$ the all-ones and all-zeros vectors, respectively; the witness wires $\mathbf{a_L}, \mathbf{a_R}, \mathbf{a_O}$ (left, right, output) lie in $\mathbb{F}^n$, where n and q are the number of multiplication gates and linear constraints. We denote by $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_i a_i b_i$ the inner product, and by $\mathbf{a} \circ \mathbf{b} = (a_1 b_1, \ldots, a_n b_n)$ the Hadamard (entry-wise) product; $(\mathbf{a})^{-1}$ is the entry-wise inverse, so $\mathbf{a} \circ (\mathbf{a})^{-1} = \mathbf{1}$. For $y \in \mathbb{F}^*$ we write $\mathbf{y} = (1, y, \ldots, y^{n-1})$ for the vector of powers and abbreviate $(\mathbf{y})^{-1} = (\mathbf{y})^{-1}$; likewise $\mathbf{z} = (1, z, \ldots, z^{q-1})$.

Generalized Bulletproofs

Bulletproofs provide a Non-Interactive (perfectly) Zero-Knowledge Argument-of-Knowledge (NIZKAoK) for knowledge of $\mathbf{a_L}, \mathbf{a_R}, \mathbf{a_O}, \mathbf{v}$ such that the following relation holds:

$$\mathscr{R}_{\mathsf{BP}} = \left\{ \begin{array}{l} \left(\mathbf{a_L}, \mathbf{a_R}, \mathbf{a_O}, \mathbf{v}, \{\gamma_k\}_{k=1}^m\right) : \\ \mathbf{0} = W_L \cdot \mathbf{a_L} + W_R \cdot \mathbf{a_R} + W_O \cdot \mathbf{a_O} + W_V \cdot \mathbf{v} + \mathbf{c} \\ \mathbf{0} = \mathbf{a_L} \circ \mathbf{a_R} - \mathbf{a_O} \\ \forall k.\ V_{(k)} = v_k \cdot G + \gamma_k \cdot H \end{array} \right\}$$

Where $\mathbf{a_L}, \mathbf{a_R}, \mathbf{a_O} \in \mathbb{F}^n$ and $\mathbf{v}$ are the witnesses, with $\mathbf{v}$ being the opening of a number of (scalar) Pedersen commitments $V_{(1)}, \ldots, V_{(m)} \in \mathbb{G}$. Compared to the standard bulletproofs description, to follow the implementation, we have moved $W_V \cdot \mathbf{v}$ and $\mathbf{c}$ over, so the relation is a single linear combination equal to $\mathbf{0}$. Generalized bulletproofs extends this with c "pre-committed" vectors $\mathbf{a_C}^{(1)}, \ldots, \mathbf{a_C}^{(c)}$, each the opening of a Pedersen commitment of some higher dimension:

$$\mathscr{R}_{\mathsf{GBP}} = \left\{ \begin{array}{l} \left(\mathbf{a_L}, \mathbf{a_R}, \mathbf{a_O}, \mathbf{v}, \{\mathbf{a_C}^{(i)}, \mathsf{aux}^{(i)}, \gamma^{(i)}\}_{i=1}^c, \{\gamma_k\}_{k=1}^m\right) : \\ \mathbf{0} = W_L \cdot \mathbf{a_L} + W_R \cdot \mathbf{a_R} + W_O \cdot \mathbf{a_O} + \sum_{i=1}^c W_C^{(i)} \cdot \mathbf{a_C}^{(i)} + W_V \cdot \mathbf{v} + \mathbf{c} \\ \mathbf{0} = \mathbf{a_L} \circ \mathbf{a_R} - \mathbf{a_O} \\ \forall i.\ A_C^{(i)} = \left\langle \mathbf{a_C}^{(i)}, \mathbf{G} \right\rangle + \left\langle \mathsf{aux}^{(i)}, \mathbf{H} \right\rangle + \gamma^{(i)} \cdot H \\ \forall k.\ V_{(k)} = v_k \cdot G + \gamma_k \cdot H \end{array} \right\}$$

For *vector commitments* $A_C^{(1)}, \ldots, A_C^{(c)} \in \mathbb{G}$ and *scalar commitments* $V_{(1)}, \ldots, V_{(m)} \in \mathbb{G}$. Each committed vector $\mathbf{a_C}^{(i)}$ comes with its own public weight matrix $W_C^{(i)}$ and public commitment $A_C^{(i)}$. The "$\langle \mathbf{aux}^{(i)}, \mathbf{H} \rangle$" term is an artifact of the proof system: it lets a prover who only knows an opening which also depends on the $\mathbf{H}$ generators complete a proof, however these "wires" cannot be "accessed" from within the circuit. To build context for the following code-level review, let's walk through the Bulletproof arithmetization, which turns this R1CS-equivalent system into an inner product relation, between uniformly random vectors.

§01

GBP Arith 1: R1CS $\rightarrow \Sigma$ IP

To simplify the claim:

$$\mathbf{0} = \mathbf{a_L} \circ \mathbf{a_R} - \mathbf{a_O}$$

$$\mathbf{0} = W_L \cdot \mathbf{a_L} + W_R \cdot \mathbf{a_R} + W_O \cdot \mathbf{a_O} + \sum_{i=1}^{c} W_C^{(i)} \cdot \mathbf{a_C}^{(i)} + W_V \cdot \mathbf{v} + \mathbf{c}$$

The verifier samples $y \leftarrow \mathbb{F}^*$ and $z \leftarrow \mathbb{F}$, and defines $\mathbf{y} = (1, \ldots, y^{n-1})$ and $\mathbf{z} = (1, \ldots, z^{q-1})$. We use this to compute linear combinations of every row of each expression:

$$0 = \langle \mathbf{y}, \mathbf{a_L} \circ \mathbf{a_R} - \mathbf{a_O} \rangle$$

$$0 = \left\langle \mathbf{z}, W_L \cdot \mathbf{a_L} + W_R \cdot \mathbf{a_R} + W_O \cdot \mathbf{a_O} + \sum_{i=1}^{c} W_C^{(i)} \cdot \mathbf{a_C}^{(i)} + W_V \cdot \mathbf{v} + \mathbf{c} \right\rangle$$

In case either of the old relations did not hold, this new relation only holds with probability $\leq \max(n, q)/|\mathbb{F}|$. Finally, we can compute a linear combination of each of these equations to end up with a single equation which must equal zero, here, we "reuse" the challenge z to "shift" the second equation:

$$\begin{aligned} 0 = &\langle \mathbf{y}, \mathbf{a_L} \circ \mathbf{a_R} - \mathbf{a_O} \rangle \\ &+ z \cdot \left\langle \mathbf{z}, W_L \cdot \mathbf{a_L} + W_R \cdot \mathbf{a_R} + W_O \cdot \mathbf{a_O} + \sum_{i=1}^{c} W_C^{(i)} \cdot \mathbf{a_C}^{(i)} + W_V \cdot \mathbf{v} + \mathbf{c} \right\rangle \end{aligned}$$

In summary, we went from an equation over vectors to single field elements using the challenges y and z. Moving $z\cdot$ in and splitting the inner products, we can rewrite the second part of the expression as:

$$\begin{aligned} 0 = &\langle z\mathbf{z} \cdot W_L, \mathbf{a_L} \rangle + \langle z\mathbf{z} \cdot W_R, \mathbf{a_R} \rangle + \langle z\mathbf{z} \cdot W_O, \mathbf{a_O} \rangle \\ &+ \sum_{i=1}^{c} \left\langle z\mathbf{z} \cdot W_C^{(i)}, \mathbf{a_C}^{(i)} \right\rangle + \langle z\mathbf{z} \cdot W_V, \mathbf{v} \rangle + \langle z\mathbf{z}, \mathbf{c} \rangle \end{aligned}$$

Let us define:

$$\begin{aligned} \mathbf{w_L} &= z \cdot \mathbf{z} \cdot W_L \in \mathbb{F}^n \\ \mathbf{w_R} &= z \cdot \mathbf{z} \cdot W_R \in \mathbb{F}^n \\ \mathbf{w_V} &= z \cdot \mathbf{z} \cdot W_V \in \mathbb{F}^n \\ \mathbf{w_C}^{(i)} &= z \cdot \mathbf{z} \cdot W_C^{(i)} \in \mathbb{F}^n \quad (i = 1, \ldots, c) \\ \mathbf{w_O} &= z \cdot \mathbf{z} \cdot W_O \in \mathbb{F}^n \\ w_c &= \langle z \cdot \mathbf{z}, \mathbf{c} \rangle \in \mathbb{F} \end{aligned}$$

Note that the verifier can compute all the vectors itself, since the matrices are public.

We are now left to “deal” with:

$$0 = \langle \mathbf{y}, \mathbf{a_L} \circ \mathbf{a_R} - \mathbf{a_O} \rangle + \langle \mathbf{w_L}, \mathbf{a_L} \rangle + \langle \mathbf{w_R}, \mathbf{a_R} \rangle + \langle \mathbf{w_O}, \mathbf{a_O} \rangle + \sum_{i=1}^{c} \left\langle \mathbf{w_C}^{(i)}, \mathbf{a_C}^{(i)} \right\rangle + \langle \mathbf{w_V}, \mathbf{v} \rangle + w_c \in \mathbb{F}$$

Splitting the first inner product, we get the expression that we want to look at going forward:

$$0 = \langle \mathbf{y}, \mathbf{a_L} \circ \mathbf{a_R} \rangle - \langle \mathbf{y}, \mathbf{a_O} \rangle + \langle \mathbf{w_L}, \mathbf{a_L} \rangle + \langle \mathbf{w_R}, \mathbf{a_R} \rangle + \langle \mathbf{w_O}, \mathbf{a_O} \rangle + \sum_{i=1}^{c} \left\langle \mathbf{w_C}^{(i)}, \mathbf{a_C}^{(i)} \right\rangle + \langle \mathbf{w_V}, \mathbf{v} \rangle + w_c \in \mathbb{F}$$

The next issue at hand is that the expression above has *many* inner products. Since we need to do an expensive folding argument for every inner product we would like to avoid this: we want to combine all these inner products into a single one. The trick is going to be to interpret these as committed vector polynomials, enabling us to evaluate many parallel polynomials of low degree. More on that now...

§02

GBP Arith 2: Vector Polynomials

An “n” dimensional “vector polynomial” consists of n polynomials “in parallel”:

$$\mathbf{f}(X) = (f_1(X), \ldots, f_n(X)) = \sum_{i=0}^{d} \mathbf{c_i} \cdot X^i \in \mathbb{F}[X]^n$$

In other words, a polynomial with *coefficients* in $\mathbf{c_i} \in \mathbb{F}^n$ and evaluation points from $\mathbb{F}$, such that plugging an x into $\mathbf{f}(X)$ evaluates to $\mathbf{f}(x) \in \mathbb{F}^n$. This notion is useful, because (vector) Pedersen commitments allow us to commit to a vector polynomial efficiently (independently of the dimension n):

$$\begin{aligned} \mathsf{Com}_{\mathbf{G},H} : (\mathbb{F}^n)[X]^{\leq d} \times \mathbb{F}^{d+1} &\to \mathbb{G}^{d+1} \\ \mathsf{Com}_{\mathbf{G},H}(\mathbf{f}(X); \mathbf{r}) &\mapsto (\langle \mathbf{c_0}, \mathbf{G} \rangle + r_0 \cdot H, \ldots, \langle \mathbf{c_d}, \mathbf{G} \rangle + r_d \cdot H) \in \mathbb{G}^{d+1} \end{aligned}$$

As well as homomorphically evaluate every coordinate of the vector polynomial at some public point x:

$$\sum_{i=0}^{d} x^i \cdot (\langle \mathbf{c_i}, \mathbf{G} \rangle + r_i \cdot H) = \langle \mathbf{f}(x), \mathbf{G} \rangle + r' \cdot H \in \mathbb{G}$$

Pedersen vector commitments support one further homomorphic operation we will rely on: multiplying the committed vector by a *public* vector, i.e. a Hadamard product with a public factor, simply by a change of basis. Writing the commitment to $\mathbf{v}$ in basis $\mathbf{G}$ explicitly as $\langle \mathbf{v}, \mathbf{G} \rangle + r \cdot H$, a public Hadamard factor $\mathbf{s}$ is applied to the opening by re-committing in the basis $(\mathbf{s})^{-1} \circ \mathbf{G}$:

$$\begin{aligned} \mathsf{Com}_{(\mathbf{s})^{-1} \circ \mathbf{G},\, H}(\mathbf{v} \circ \mathbf{s}; r) &= \mathsf{Com}_{\mathbf{G},\, H}(\mathbf{v}; r) \\ \left\langle \mathbf{v} \circ \mathbf{s}, (\mathbf{s})^{-1} \circ \mathbf{G} \right\rangle + r \cdot H &= \langle \mathbf{v}, \mathbf{G} \rangle + r \cdot H \end{aligned}$$

In other words, a commitment to $\mathbf{v}$ in basis $\mathbf{G}$ is also a commitment to $\mathbf{v} \circ \mathbf{s}$ in the basis $(\mathbf{s})^{-1} \circ \mathbf{G}$ with the same blinding r: to apply a public Hadamard factor $\mathbf{s}$ to the opening, one re-commits in the basis $(\mathbf{s})^{-1} \circ \mathbf{G}$. This holds because $\mathbf{s} \circ (\mathbf{s})^{-1} = \mathbf{1}$ leaves each generator’s contribution unchanged and the blinding term $r \cdot H$ is untouched.

The “inner product” between two vector polynomials is defined in the intuitive way (for any module over any ring): taking the *coordinate-wise product of polynomials and summing*:

$$\langle \mathbf{f}(X), \mathbf{g}(X) \rangle = \sum_i f_i(X) \cdot g_i(X) \in \mathbb{F}[X]$$

Note that $\forall x.\ \langle \mathbf{f}, \mathbf{g} \rangle(x) = \langle \mathbf{f}(x), \mathbf{g}(x) \rangle$ and $\deg(\langle \mathbf{f}, \mathbf{g} \rangle) = \deg(\mathbf{f}) + \deg(\mathbf{g})$.

§03

GBP Arith 3: $\sum$ IP $\rightarrow$ Single IP

Viewing the vectors as vector polynomials is useful because it allows us to batch check independent inner products. Suppose we have two inner products:

$$\Delta = \langle \mathbf{a}, \mathbf{b} \rangle + \langle \mathbf{c}, \mathbf{d} \rangle$$

We define the vector polynomials (“left/right”):

$$\mathbf{f_L}(X) = \mathbf{a} \cdot X + \mathbf{c} \cdot X^2$$
$$\mathbf{f_R}(X) = \mathbf{b} \cdot X + \mathbf{d}$$

And consider the inner product, then the desired inner product Δ lands in the square term:

$$\langle \mathbf{f_L}, \mathbf{f_R} \rangle(X) = \delta_0 + \delta_1 \cdot X + \Delta \cdot X^2 + \delta_3 \cdot X^3 \in \mathbb{F}[X]$$

Where $\delta_0, \delta_1, \delta_3$ are some cross-term garbage we don’t care about. More generally: we define a “left polynomial” where powers *increase* for every left term in the series of inner products and a “right polynomial” where the powers *decrease* for every right term, then the terms will “align” at the “middle power”. i.e. in general, suppose we have:

$$\Delta = \sum_{i=1}^{t} \langle \mathbf{L_i}, \mathbf{R_i} \rangle$$

Then we define:

$$\mathbf{f_L}(X) = \sum_{i=1}^{t} \mathbf{L_i} \cdot X^i$$
$$\mathbf{f_R}(X) = \sum_{i=1}^{t} \mathbf{R_i} \cdot X^{t-i}$$

In which case, the t‘th coefficient of $\langle \mathbf{f_L}, \mathbf{f_R} \rangle(X)$ is Δ. This observation suggests the following approach to reduce a sum of multiple inner products, given commitments to every vector, to a single inner product as follows:

1. Prover sends commitments to the coefficients $\{\delta_i\}_{i \in 0, \ldots, 2 \cdot t - 1}$, where we are interested in $\delta_t = \Delta$, which is usually implicit (e.g. fixed to 0). Then both parties locally define:

$$\mathbf{f_L}(X) = \sum_{i=1}^{t} \mathbf{L_i} \cdot X^i \in \mathbb{F}[X]^n$$
$$\mathbf{f_R}(X) = \sum_{i=1}^{t} \mathbf{R_i} \cdot X^{t-i} \in \mathbb{F}[X]^n$$
$$g(X) = \sum_{i=0}^{2 \cdot t - 1} \delta_i \cdot X^i \in \mathbb{F}[X]$$

1. Verifier samples $x \leftarrow \mathbb{F}$
2. Both sides compute commitments to the vectors:

$$\mathbf{f_L}(x), \mathbf{f_R}(x) \in \mathbb{F}^n$$

And a commitment to the field element $g(x) \in \mathbb{F}$, using the homomorphic property of the Pedersen commitments. We now have just a single inner product claim about Pedersen commitments:

$$\langle \mathbf{f_L}(x), \mathbf{f_R}(x) \rangle = g(x)$$

§04

GBP Arith 4: Combining

Recall the expression we were trying to verify, before our digression about batching inner products and vector polynomials:

$$0 = \langle \mathbf{y}, \mathbf{a_L} \circ \mathbf{a_R} \rangle - \langle \mathbf{y}, \mathbf{a_O} \rangle + \langle \mathbf{w_L}, \mathbf{a_L} \rangle + \langle \mathbf{w_R}, \mathbf{a_R} \rangle + \langle \mathbf{w_O}, \mathbf{a_O} \rangle + \sum_{i=1}^{c} \left\langle \mathbf{w_C}^{(i)}, \mathbf{a_C}^{(i)} \right\rangle + \langle \mathbf{w_V}, \mathbf{v} \rangle + w_c \in \mathbb{F}$$

To apply the techniques above, we want to:

- **Eliminate Pairwise Products.** We haven't yet described a technique for proving Hadamard (pairwise) products of secret (committed) vectors, so we get to get rid of $\mathbf{a_L} \circ \mathbf{a_R}$; we want each secret to appear on a different side of an inner product.
- **Collect Common Terms.** Reduce the number of inner products, for efficiency, by collecting common terms. This reduces the degree of the vector polynomials, meaning: less committed coefficients and less cross-terms.

Eliminate Secret-Secret Hadamard Product

Note that the left side of the inner product $\langle \mathbf{y}, \mathbf{a_L} \circ \mathbf{a_R} \rangle$ is public, so let us move one secret to each side, using $\langle \mathbf{y}, \mathbf{a_L} \circ \mathbf{a_R} \rangle = \langle \mathbf{a_L}, \mathbf{y} \circ \mathbf{a_R} \rangle$ to get rid of the Hadamard product between secret values:

$$0 = \langle \mathbf{y}, \mathbf{a_L} \circ \mathbf{a_R} \rangle - \langle \mathbf{y}, \mathbf{a_O} \rangle + \langle \mathbf{w_L}, \mathbf{a_L} \rangle + \langle \mathbf{w_R}, \mathbf{a_R} \rangle + \langle \mathbf{w_O}, \mathbf{a_O} \rangle + \sum_{i=1}^{c} \left\langle \mathbf{w_C}^{(i)}, \mathbf{a_C}^{(i)} \right\rangle + \langle \mathbf{w_V}, \mathbf{v} \rangle + w_c \in \mathbb{F}$$

Becomes

$$0 = \langle \mathbf{a_L}, \mathbf{y} \circ \mathbf{a_R} \rangle - \langle \mathbf{y}, \mathbf{a_O} \rangle + \langle \mathbf{w_L}, \mathbf{a_L} \rangle + \langle \mathbf{w_R}, \mathbf{a_R} \rangle + \langle \mathbf{w_O}, \mathbf{a_O} \rangle + \sum_{i=1}^{c} \left\langle \mathbf{w_C}^{(i)}, \mathbf{a_C}^{(i)} \right\rangle + \langle \mathbf{w_V}, \mathbf{v} \rangle + w_c \in \mathbb{F}$$

Note that we *know* how to deal with a Hadamard product between a secret and a public value.

Combine the $\mathbf{a_O}$ terms

$$0 = \langle \mathbf{a_L}, \mathbf{y} \circ \mathbf{a_R} \rangle - \langle \mathbf{y}, \mathbf{a_O} \rangle + \langle \mathbf{w_L}, \mathbf{a_L} \rangle + \langle \mathbf{w_R}, \mathbf{a_R} \rangle + \langle \mathbf{w_O}, \mathbf{a_O} \rangle + \sum_{i=1}^{c} \left\langle \mathbf{w_C}^{(i)}, \mathbf{a_C}^{(i)} \right\rangle + \langle \mathbf{w_V}, \mathbf{v} \rangle + w_c \in \mathbb{F}$$

Becomes

$$0 = \langle \mathbf{a_L}, \mathbf{y} \circ \mathbf{a_R} \rangle + \langle \mathbf{w_L}, \mathbf{a_L} \rangle + \langle \mathbf{w_R}, \mathbf{a_R} \rangle + \langle \mathbf{w_O} - \mathbf{y}, \mathbf{a_O} \rangle + \sum_{i=1}^{c} \left\langle \mathbf{w_C}^{(i)}, \mathbf{a_C}^{(i)} \right\rangle + \langle \mathbf{w_V}, \mathbf{v} \rangle + w_c \in \mathbb{F}$$

Rewrite $\mathbf{a_R}$ Inner Product

Using $\langle \mathbf{a_R}, \mathbf{w_R} \rangle = \left\langle \mathbf{w_R} \circ (\mathbf{y})^{-1}, \mathbf{y} \circ \mathbf{a_R} \right\rangle$ rewrite:

$$0 = \langle \mathbf{a_L}, \mathbf{y} \circ \mathbf{a_R} \rangle + \langle \mathbf{w_L}, \mathbf{a_L} \rangle + \langle \mathbf{w_R}, \mathbf{a_R} \rangle + \langle \mathbf{w_O} - \mathbf{y}, \mathbf{a_O} \rangle + \sum_{i=1}^{c} \left\langle \mathbf{w_C}^{(i)}, \mathbf{a_C}^{(i)} \right\rangle + \langle \mathbf{w_V}, \mathbf{v} \rangle + w_c \in \mathbb{F}$$

Becomes

$$0 = \langle \mathbf{a_L}, \mathbf{y} \circ \mathbf{a_R} \rangle + \langle \mathbf{w_L}, \mathbf{a_L} \rangle + \left\langle \mathbf{w_R} \circ (\mathbf{y})^{-1}, \mathbf{y} \circ \mathbf{a_R} \right\rangle + \langle \mathbf{w_O} - \mathbf{y}, \mathbf{a_O} \rangle + \sum_{i=1}^{c} \left\langle \mathbf{w_C}^{(i)}, \mathbf{a_C}^{(i)} \right\rangle + \langle \mathbf{w_V}, \mathbf{v} \rangle + w_c \in \mathbb{F}$$

Collect $\mathbf{y} \circ \mathbf{a_R}$ Terms

This was our motivation for the previous step:

$$0 = \langle \mathbf{a_L}, \mathbf{y} \circ \mathbf{a_R} \rangle + \langle \mathbf{w_L}, \mathbf{a_L} \rangle + \left\langle \mathbf{w_R} \circ (\mathbf{y})^{-1}, \mathbf{y} \circ \mathbf{a_R} \right\rangle + \langle \mathbf{w_O} - \mathbf{y}, \mathbf{a_O} \rangle + \sum_{i=1}^{c} \left\langle \mathbf{w_C}^{(i)}, \mathbf{a_C}^{(i)} \right\rangle + \langle \mathbf{w_V}, \mathbf{v} \rangle + w_c \in \mathbb{F}$$

Becomes

$$0 = \left\langle \mathbf{a_L} + \mathbf{w_R} \circ (\mathbf{y})^{-1}, \mathbf{y} \circ \mathbf{a_R} \right\rangle + \langle \mathbf{w_L}, \mathbf{a_L} \rangle + \langle \mathbf{w_O} - \mathbf{y}, \mathbf{a_O} \rangle + \sum_{i=1}^{c} \left\langle \mathbf{w_C}^{(i)}, \mathbf{a_C}^{(i)} \right\rangle + \langle \mathbf{w_V}, \mathbf{v} \rangle + w_c \in \mathbb{F}$$

Add $\delta(y, z)$ to Both Sides

Add the public scalar $\delta(y, z) = \left\langle (\mathbf{y})^{-1} \circ \mathbf{w_R}, \mathbf{w_L} \right\rangle$ to both sides (note that 0 on the left becomes $\delta(y, z)$):

$$0 = \left\langle \mathbf{a_L} + \mathbf{w_R} \circ (\mathbf{y})^{-1}, \mathbf{y} \circ \mathbf{a_R} \right\rangle + \langle \mathbf{w_L}, \mathbf{a_L} \rangle + \langle \mathbf{w_O} - \mathbf{y}, \mathbf{a_O} \rangle + \sum_{i=1}^{c} \left\langle \mathbf{w_C}^{(i)}, \mathbf{a_C}^{(i)} \right\rangle + \langle \mathbf{w_V}, \mathbf{v} \rangle + w_c \in \mathbb{F}$$

Becomes

$$\delta(y,z) = \left\langle \mathbf{a_L} + \mathbf{w_R} \circ (\mathbf{y})^{-1}, \mathbf{y} \circ \mathbf{a_R} \right\rangle + \langle \mathbf{w_L}, \mathbf{a_L} \rangle + \left\langle (\mathbf{y})^{-1} \circ \mathbf{w_R}, \mathbf{w_L} \right\rangle + \langle \mathbf{w_O} - \mathbf{y}, \mathbf{a_O} \rangle + \sum_{i=1}^{c} \left\langle \mathbf{w_C}^{(i)}, \mathbf{a_C}^{(i)} \right\rangle + \langle \mathbf{w_V}, \mathbf{v} \rangle + w_c \in \mathbb{F}$$

Note that $\delta(y,z)$ does not depend on the witness! (the verifier can compute it)

Combine $\mathbf{w_L}$ Terms

$$\delta(y,z) = \left\langle \mathbf{a_L} + \mathbf{w_R} \circ (\mathbf{y})^{-1}, \mathbf{y} \circ \mathbf{a_R} \right\rangle + \langle \mathbf{w_L}, \mathbf{a_L} \rangle + \left\langle (\mathbf{y})^{-1} \circ \mathbf{w_R}, \mathbf{w_L} \right\rangle + \langle \mathbf{w_O} - \mathbf{y}, \mathbf{a_O} \rangle + \sum_{i=1}^{c} \left\langle \mathbf{w_C}^{(i)}, \mathbf{a_C}^{(i)} \right\rangle + \langle \mathbf{w_V}, \mathbf{v} \rangle + w_c \in \mathbb{F}$$

Becomes

$$\delta(y,z) = \left\langle \mathbf{a_L} + \mathbf{w_R} \circ (\mathbf{y})^{-1}, \mathbf{y} \circ \mathbf{a_R} \right\rangle + \left\langle (\mathbf{y})^{-1} \circ \mathbf{w_R} + \mathbf{a_L}, \mathbf{w_L} \right\rangle + \langle \mathbf{w_O} - \mathbf{y}, \mathbf{a_O} \rangle + \sum_{i=1}^{c} \left\langle \mathbf{w_C}^{(i)}, \mathbf{a_C}^{(i)} \right\rangle + \langle \mathbf{w_V}, \mathbf{v} \rangle + w_c \in \mathbb{F}$$

Combine $\mathbf{a_L} + \mathbf{w_R} \circ (\mathbf{y})^{-1}$ Terms

$$\delta(y,z) = \left\langle \mathbf{a_L} + \mathbf{w_R} \circ (\mathbf{y})^{-1}, \mathbf{y} \circ \mathbf{a_R} \right\rangle + \left\langle (\mathbf{y})^{-1} \circ \mathbf{w_R} + \mathbf{a_L}, \mathbf{w_L} \right\rangle + \langle \mathbf{w_O} - \mathbf{y}, \mathbf{a_O} \rangle + \sum_{i=1}^{c} \left\langle \mathbf{w_C}^{(i)}, \mathbf{a_C}^{(i)} \right\rangle + \langle \mathbf{w_V}, \mathbf{v} \rangle + w_c \in \mathbb{F}$$

Becomes

$$\delta(y,z) = \left\langle \mathbf{a_L} + \mathbf{w_R} \circ (\mathbf{y})^{-1}, \mathbf{y} \circ \mathbf{a_R} + \mathbf{w_L} \right\rangle + \langle \mathbf{w_O} - \mathbf{y}, \mathbf{a_O} \rangle + \sum_{i=1}^{c} \left\langle \mathbf{w_C}^{(i)}, \mathbf{a_C}^{(i)} \right\rangle + \langle \mathbf{w_V}, \mathbf{v} \rangle + w_c \in \mathbb{F}$$

So we are left with $2 + c$ inner products: two coming from the $\mathbf{a_L}, \mathbf{a_R}, \mathbf{a_O}$ terms and one for each of the c committed vectors $\mathbf{a_C}^{(i)}$. All these are combined into a single inner product using the previous technique based on vector polynomials.

Note that during verification everything except the $2 + c$ inner products is known to the verifier: $\delta(y,z)$ (on the left) and w_c are public, and $\langle \mathbf{w_V}, \mathbf{v} \rangle$ is a commitment to a single field element, formed homomorphically from the $V_{(k)}$. Those inner products are folded into a single one next.

§05

Arithmetization

Public inputs: generators $\mathbf{G}, \mathbf{H} \in \mathbb{G}^n$ and $G, H \in \mathbb{G}$; weight matrices W_L, W_R, W_O, $\{W_C^{(k)}\}_{k=1}^{c}$, W_V with constant $\mathbf{c}$; vector commitments $\{A_C^{(k)}\}_{k=1}^{c}$ and scalar commitments $\{V_{(j)}\}_{j=1}^{m}$. Write $n' = 2c + 2$ for the target monomial.

- **Prover $\rightarrow$ Verifier.**
  - Sample $\alpha, \beta, \rho \leftarrow \mathbb{F}$ and $\mathbf{s_L}, \mathbf{s_R} \leftarrow \mathbb{F}^n$.
  - $A_{L|R} = \langle \mathbf{a_L}, \mathbf{G} \rangle + \langle \mathbf{a_R}, \mathbf{H} \rangle + \alpha H$

- $A_O = \langle \mathbf{a_O}, \mathbf{G} \rangle + \beta H$
- $S = \langle \mathbf{s_L}, \mathbf{G} \rangle + \langle \mathbf{s_R}, \mathbf{H} \rangle + \rho H$
- Send $(A_{L|R}, A_O, S)$.

- **Verifier $\rightarrow$ Prover.**
  - Sample $y, z \leftarrow \mathbb{F}^*$.
  - Send (y, z).
  - Both parties compute:
    - The challenge vectors $\mathbf{y}, (\mathbf{y})^{-1}, \mathbf{z}$
    - The weights $\mathbf{w_L}, \mathbf{w_R}, \mathbf{w_O}, \{\mathbf{w_C}^{(k)}\}, \mathbf{w_V}, w_c$
    - $\delta(y, z) = \left\langle (\mathbf{y})^{-1} \circ \mathbf{w_R}, \mathbf{w_L} \right\rangle$

- **Prover $\rightarrow$ Verifier.**
  - Recall that the inner products to batch are:

$$\begin{aligned} t_{n'} &= \left\langle \mathbf{a_L} + \mathbf{w_R} \circ (\mathbf{y})^{-1}, \mathbf{y} \circ \mathbf{a_R} + \mathbf{w_L} \right\rangle + \langle \mathbf{a_O}, \mathbf{w_O} - \mathbf{y} \rangle + \sum_{k=1}^{c} \left\langle \mathbf{a_C}^{(k)}, \mathbf{w_C}^{(k)} \right\rangle \\ &= \delta(y, z) - w_c - \langle \mathbf{w_V}, \mathbf{v} \rangle. \end{aligned}$$

  - Build $\mathbf{f_L}(X), \mathbf{f_R}(X) \in \mathbb{F}[X]^n$ (powers as in GBP Arith 3) with nonzero coefficients:
    - $\mathbf{f_L}(X)$:
      - $\mathbf{a_L} + \mathbf{w_R} \circ (\mathbf{y})^{-1}$ at $X^{n'/2}$
      - $\mathbf{a_C}^{(k)}$ at $X^{n'-k}$ for $k = 1, \ldots, c$
      - $\mathbf{a_O}$ at $X^{n'}$
      - $\mathbf{s_L}$ at $X^{n'+1}$
    - $\mathbf{f_R}(X)$:
      - $\mathbf{w_O} - \mathbf{y}$ at X^0
      - $\mathbf{w_C}^{(k)}$ at X^k for $k = 1, \ldots, c$
      - $\mathbf{y} \circ \mathbf{a_R} + \mathbf{w_L}$ at $X^{n'/2}$
      - $\mathbf{y} \circ \mathsf{aux}^{(k)}$ at $X^{n'-k}$ for $k = 1, \ldots, c$
      - $\mathbf{y} \circ \mathbf{s_R}$ at $X^{n'+1}$
    - Corresponding to generators in $\mathbf{G}$ and $\mathbf{H}$ respectively.
  - $t(X) = \langle \mathbf{f_L}(X), \mathbf{f_R}(X) \rangle = \sum_{i=n'/2}^{2(n'+1)} t_i X^i$; the target coefficient is $t_{n'}$.
  - Sample $\tau_i \leftarrow \mathbb{F}$ for every $i \neq n'$ and set $T_i = t_i G + \tau_i H$.
  - Send $\{T_i\}_{i \neq n'}$.

- **Verifier $\rightarrow$ Prover.**
  - Sample $x \leftarrow \mathbb{F}^*$.
  - Send x.

- **Prover $\rightarrow$ Verifier.**

- Opening of $t(x)$ (value $\hat{t}$, blinding τ_x):

$$\hat{t} = \langle \mathbf{f_L}(x), \mathbf{f_R}(x) \rangle = t(x)$$
$$\tau_x = \sum_{i \neq n'} \tau_i x^i - x^{n'} \langle \mathbf{w_V}, \gamma_V \rangle$$

- Opening of P (vectors $\mathbf{f_L}(x), \mathbf{f_R}(x)$, blinding μ):

$$\mu = \alpha \cdot x^{n'/2} + \beta \cdot x^{n'} + \rho \cdot x^{n'+1} + \sum_{k=1}^{c} \gamma^{(k)} \cdot x^{n'-k}$$

  With γ_V the blinding factors of the $V_{(j)}$ and $\gamma^{(k)}$ the blinding factor of $A_C^{(k)}$.

- Send $(\tau_x, \mu, \hat{t}, \mathbf{f_L}(x) \in \mathbb{F}^n, \mathbf{f_R}(x) \in \mathbb{F}^n)$.

- **Verifier.**
  - Set $\mathbf{H}^{(\mathbf{y})^{-1}} = (\mathbf{y})^{-1} \circ \mathbf{H}$.
  - Define:

$$\widetilde{W}_L = \left\langle \mathbf{w_L}, \mathbf{H}^{(\mathbf{y})^{-1}} \right\rangle \in \mathbb{G} \qquad \widetilde{W}_R = \left\langle (\mathbf{y})^{-1} \circ \mathbf{w_R}, \mathbf{G} \right\rangle \in \mathbb{G}$$
$$\widetilde{W}_O = \left\langle \mathbf{w_O} - \mathbf{y}, \mathbf{H}^{(\mathbf{y})^{-1}} \right\rangle \in \mathbb{G} \qquad \widetilde{W}_k = \left\langle \mathbf{w_C}^{(k)}, \mathbf{H}^{(\mathbf{y})^{-1}} \right\rangle \in \mathbb{G}$$

  - Compute:

$$P = \widetilde{W}_O + \sum_{k=1}^{c} x^k \widetilde{W}_k + x^{n'/2}\left(A_{L|R} + \widetilde{W}_L + \widetilde{W}_R\right) + \sum_{k=1}^{c} x^{n'-k} A_C^{(k)} + x^{n'} A_O + x^{n'+1} S$$

  - Check:

$$\hat{t}\, G + \tau_x H = x^{n'}\left((\delta(y,z) - w_c) G - \textstyle\sum_{j=1}^{m} (\mathbf{w_V})_j\, V_{(j)}\right) + \sum_{i \neq n'} x^i T_i$$

  - Check:

$$\left((\mathbf{G}, \mathbf{H}^{(\mathbf{y})^{-1}}, H, P, \mu, \hat{t}),\ (\mathbf{f_L}(x), \mathbf{f_R}(x))\right) \in \mathscr{R}_{\mathsf{IP}}$$

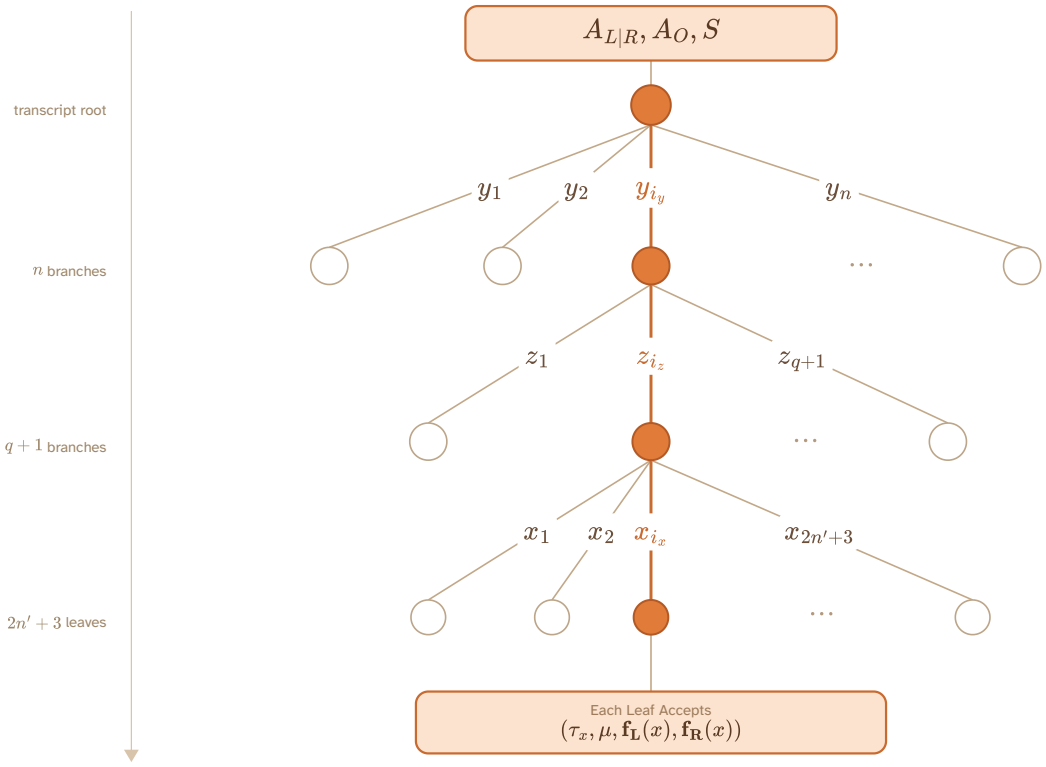
Computational Special Soundness

We prove that the arithmetization ('Generalized Bulletproofs') is an argument of knowledge for $\mathscr{R}_{\mathsf{GBP}}$. The theorem below establishes $(n, q+1, 2n'+3)$-computational special soundness. By Attema, Fehr and Klooß (TCC 2022) the Fiat-Shamir compiled argument then has tight security: under the hardness of discrete log over $\mathbb{G}$ it is knowledge sound with knowledge error at most $(Q_{\mathsf{ro}} + 1) \cdot (n + q + 4c + 5)/(|\mathbb{F}| - 1)$, where Q_{ro} is the number of random-oracle queries made by the adversary.

Theorem. *Assume W_V has full column rank m, i.e. a left inverse. The full arithmetization is $(n, q+1, 2n'+3)$-computationally special sound for $\mathscr{R}_{\mathsf{GBP}}$: from any $(n, q+1, 2n'+3)$-tree of accepting transcripts a deterministic extractor returns either a witness for $\mathscr{R}_{\mathsf{GBP}}$, or a non-trivial discrete-log relation among the generators $\Gamma = \mathbf{G} \,\|\, \mathbf{H} \,\|\, G \,\|\, H \in \mathbb{G}^{2n+2}$, i.e. a non-zero $\mathbf{v}$ with $\langle \mathbf{v}, \Gamma \rangle = 0$.*

The Tree of Transcripts. A $(n, q+1, 2n'+3)$-tree fixes the first message $(A_{L|R}, A_O, S)$ at the root and branches on the three challenges, recording each verifier message at its node: n distinct children on y; under each, $q+1$ distinct children on z, each with its $\{T_i\}$; under each, $2n'+3$ distinct **leaves** on x, each with its opening $(\tau_x, \mu, \mathbf{f_L}(x), \mathbf{f_R}(x))$.

Every root-to-leaf path is an accepting transcript. Index a (y, z)-node by (i_y, i_z) and its x-leaves by i_x, so (i_y, i_z, i_x) indexes a full transcript. The three arities are forced by the three interpolations below: $2n' + 3 = \deg t + 1$ for x; $q + 1$ for the q constraint rows together with the constant Hadamard term; n for the Hadamard coordinates.



The $(n, q + 1, 2n' + 3)$ tree of accepting transcripts: each edge is a verifier challenge (y, then z, then x), and every root-to-leaf path is an accepting transcript (one highlighted).

Step 1: x-Vandermonde Read-off. Fix a node (i_y, i_z). The verifier's P is the value at the leaf's challenge of a polynomial $\sum_{p=0}^{2(n'+1)} x^p P_p$ whose coefficients $P_p \in \mathbb{G}$ are:

$$\begin{aligned} P_0 &= \widetilde{W}_O, & P_k &= \widetilde{W}_k \ (1 \le k \le c), & P_{n'/2} &= A_{L|R} + \widetilde{W}_L + \widetilde{W}_R, \\ P_{n'-k} &= A_C^{(k)}, & P_{n'} &= A_O, & P_{n'+1} &= S, \end{aligned}$$

With $P_p = \mathbf{0}$ for $p > n' + 1$.

The $2n' + 3$ leaves of the node share (y, z) and use distinct x, so the P-opening check at leaf i_x:

$$\sum_p x_{i_x}^p \, P_p = \langle \mathbf{f_L}(x_{i_x}), \mathbf{G} \rangle + \left\langle \mathbf{f_R}(x_{i_x}), \mathbf{H}^{(\mathbf{y})^{-1}} \right\rangle + \mu_{i_x} H,$$

Is a Vandermonde system in x; inverting it writes each P_p as a representation in the $(\mathbf{G}, \mathbf{H}^{(\mathbf{y})^{-1}}, H)$ basis. When the node's collision candidates (Step 5) all vanish, the $\mathbf{G}$-block of P_p at every degree equals the honest $\mathbf{f_L}$ coefficient and the $\mathbf{H}^{(\mathbf{y})^{-1}}$-block at every degree $\neq n'$ equals the honest $\mathbf{f_R}$ coefficient: the public degrees 0 and k open purely on $\mathbf{H}^{(\mathbf{y})^{-1}}$ to $\mathbf{w_O} - \mathbf{y}$ and $\mathbf{w_C}^{(k)}$; degree $n'/2$ gives $\mathbf{a_L} + \mathbf{w_R} \circ (\mathbf{y})^{-1}$ on $\mathbf{G}$ and $\mathbf{y} \circ \mathbf{a_R} + \mathbf{w_L}$ on $\mathbf{H}^{(\mathbf{y})^{-1}}$; degrees $n' - k, n', n' + 1$ give the wires $\mathbf{a_C}^{(k)}, \mathbf{a_O}, \mathbf{s_L}$ on $\mathbf{G}$ and $\mathbf{y} \circ \mathsf{aux}^{(k)}, \mathbf{y} \circ \mathbf{s_R}$ on $\mathbf{H}^{(\mathbf{y})^{-1}}$. The lone degree-$n'$ $\mathbf{H}^{(\mathbf{y})^{-1}}$-block — the $\mathbf{H}$-part of A_O — is unconstrained; the relation tolerates this slack, and since its convolution partner is the degree-0 coefficient of $\mathbf{f_L}$, which is $\mathbf{0}$, it never contributes downstream.

Step 2: The Per-Node Identity $(\star)$. With the leaf openings pinned to the honest $\mathbf{f_L}, \mathbf{f_R}$, the n'-coefficient of $\langle \mathbf{f_L}(X), \mathbf{f_R}(X) \rangle$ — the committed target $t_{n'}$, recovered from the leaves by inverting the same x-Vandermonde at degree n' — expands *without invoking any constraint* into $\delta(y,z)$, the z-aggregated linear $R1CS$ rows, and the y-weighted Hadamard term $\langle \mathbf{y}, \mathbf{a_L} \circ \mathbf{a_R} - \mathbf{a_O} \rangle$. The first verifier check independently pins the same coefficient to $t_{n'} = \delta(y,z) - w_c - \langle \mathbf{w_V}, \mathbf{v} \rangle$; here the check couples $\langle \mathbf{w_V}, \mathbf{v} \rangle$ to the aggregate $\sum_j (\mathbf{w_V})_j V_{(j)}$ through its degree-n' coefficient, the two agreeing once the corresponding (G,H)-collision candidate of Step 5 vanishes. Equating the two expansions and cancelling $\delta(y,z)$ gives, at every node (i_y, i_z):

$$(\star) \qquad \langle \mathbf{y}, \mathbf{a_L} \circ \mathbf{a_R} - \mathbf{a_O} \rangle + \sum_{\ell=1}^{q} z^\ell \Big(W_L \mathbf{a_L} + W_R \mathbf{a_R} + W_O \mathbf{a_O} + \textstyle\sum_k W_C^{(k)} \mathbf{a_C}^{(k)} + W_V \mathbf{v} + \mathbf{c} \Big)_\ell = 0$$

Step 3: Deaggregation over z then y. It suffices to know $(\star)$ on one reference row and one reference column. Fix a reference y-child: as z ranges over the $q+1$ children of that node, $(\star)$ is a degree-q polynomial in z — constant term $\langle \mathbf{y}, \mathbf{a_L} \circ \mathbf{a_R} - \mathbf{a_O} \rangle$, row ℓ at z^ℓ — vanishing at $q+1$ distinct points, so a Vandermonde in z forces every $R1CS$ row to zero. With the rows gone, $(\star)$ on the reference column (z fixed to a reference child, y ranging over its n children) collapses to $\langle \mathbf{y}, \mathbf{a_L} \circ \mathbf{a_R} - \mathbf{a_O} \rangle = 0$ at n distinct y; a Vandermonde in y forces each coordinate. This yields the first two relation clauses:

$$W_L \mathbf{a_L} + W_R \mathbf{a_R} + W_O \mathbf{a_O} + \sum_k W_C^{(k)} \mathbf{a_C}^{(k)} + W_V \mathbf{v} + \mathbf{c} = \mathbf{0}$$

$$\mathbf{a_L} \circ \mathbf{a_R} - \mathbf{a_O} = \mathbf{0}$$

(The $q+1$ arity is the constant term plus q rows; the L-shape avoids needing $(\star)$ on the full $n \cdot (q+1)$ grid.)

Step 4: Commitment Openings. Two clauses remain, read straight off Step 1. The degree-$(n'-k)$ coefficient is $A_C^{(k)}$, whose $\mathbf{G}$-, $\mathbf{H}$- and H-blocks give the *vector-commitment* clause:

$$A_C^{(k)} = \Big\langle \mathbf{a_C}^{(k)}, \mathbf{G} \Big\rangle + \Big\langle \mathsf{aux}^{(k)}, \mathbf{H} \Big\rangle + \gamma^{(k)} H$$

with the $\big\langle \mathsf{aux}^{(k)}, \mathbf{H} \big\rangle$ slack retained — precisely the term the improved protocol removes. For the *scalar* commitments, the first check's degree-n' coefficient opens the aggregate $\sum_j (\mathbf{w_V})_j V_{(j)}$ as a (G,H)-combination at each of the $q+1$ z-children; inverting the Vandermonde in z and applying the left inverse of W_V recovers each $V_{(k)} = v_k G + \gamma_k H$.

Step 5: Witness or Relation. The extractor reads a candidate witness $\mathbf{W}$ off the Step-1 representations at a fixed reference (y,z) (the first y- and z-children): $\mathbf{a_L}$ from the $\mathbf{G}$-block at degree $n'/2$ minus the public $\mathbf{w_R} \circ (\mathbf{y})^{-1}$; $\mathbf{a_O}$ at degree n'; $\mathbf{a_C}^{(k)}$ and $\gamma^{(k)}$ from the $\mathbf{G}$- and H-blocks at degree $n'-k$; $\mathbf{a_R}$ from the $\mathbf{H}^{(\mathbf{y})^{-1}}$-block at degree $n'/2$, minus the public $\mathbf{w_L}$ and rescaled by $(\mathbf{y})^{-1}$; $\mathsf{aux}^{(k)}$ from the $\mathbf{H}^{(\mathbf{y})^{-1}}$-block at degree $n'-k$, rescaled by $(\mathbf{y})^{-1}$; and $\mathbf{v}, \{\gamma_k\}$ from the degree-n' first-check coefficients via the left inverse of W_V. It is computed by Lagrange-form Vandermonde inversion and Gaussian elimination — no discrete-log search. It also forms a list of candidate discrete-log relations: differences of two recovered representations of the same group element — the public coefficients $\widetilde{W}_O, \widetilde{W}_k$; the challenge-independent commitments $A_{L|R}, A_O, A_C^{(k)}, S$ across distinct nodes; the degree-n' (G,H)-candidate of Step 2; and the high-degree coefficients, which open $\mathbf{0}$ directly. Each candidate $\mathbf{c}$ satisfies $\langle \mathbf{c}, \Gamma \rangle = 0$ by construction, being a difference of two openings of the same commitment; it is *non-trivial* exactly when $\mathbf{c} \neq \mathbf{0}$.

The extractor outputs $\mathbf{W}$ if $\mathbf{W} \in \mathscr{R}_{\mathsf{GBP}}$, and otherwise the first non-zero candidate relation; we argue the contrapositive. If every candidate vanishes, the Step-1 read-offs hold and the degree-n' identity of Step 2 holds at every node, so $(\star)$ holds on the reference row and column; Step 3 deaggregates it into the first two clauses, and Step 4 supplies the vector- and scalar-commitment clauses. All four clauses hold, so $\mathbf{W} \in \mathscr{R}_{\mathsf{GBP}}$, contradicting the assumption. Hence a failing $\mathbf{W}$ yields a non-zero candidate, which the extractor returns as the discrete-log relation.

§06

Reduce $\mathscr{R}_{\mathsf{IP}} \longrightarrow \mathscr{R}_{\mathsf{IPA},n}$

The $\hat{t}$ check above ties $\hat{t}$ to the committed polynomial $t(X)$; the remaining verifier check is membership in a single inner-product relation $\mathscr{R}_{\mathsf{IP}}$, with statement $(\mathbf{G}, \mathbf{H}^{(\mathbf{y})^{-1}}, H, P, \mu, \hat{t})$ and witness $(\mathbf{a}, \mathbf{b})$:

$$\mathscr{R}_{\mathsf{IP}} = \left\{ \left((\mathbf{G}, \mathbf{H}^{(\mathbf{y})^{-1}}, H, P, \mu, \hat{t}),\ (\mathbf{a}, \mathbf{b})\right) : P = \langle \mathbf{a}, \mathbf{G} \rangle + \left\langle \mathbf{b}, \mathbf{H}^{(\mathbf{y})^{-1}} \right\rangle + \mu H \ \ \wedge \ \ \hat{t} = \langle \mathbf{a}, \mathbf{b} \rangle \right\}$$

Here $\mathbf{a}, \mathbf{b} \in \mathbb{F}^n$ and $\mathbf{H}^{(\mathbf{y})^{-1}} = (\mathbf{y})^{-1} \circ \mathbf{H}$. The generators $\mathbf{G}, \mathbf{H}^{(\mathbf{y})^{-1}}, H$ are public; the verifier derives the remaining statement components $P, \mu, \hat{t}$ from the protocol transcript, and the prover holds the witness $(\mathbf{a}, \mathbf{b}) = (\mathbf{f}_{\mathbf{L}}(x), \mathbf{f}_{\mathbf{R}}(x))$. We abbreviate the generators $\mathbf{g} = \mathbf{G}$ and $\mathbf{h} = \mathbf{H}^{(\mathbf{y})^{-1}}$.

To recursively fold this relation, we first want to reduce it to:

$$\mathscr{R}_{\mathsf{IPA},N} = \left\{ \left((\mathbf{g}, \mathbf{h}, u, P),\ (\mathbf{a}, \mathbf{b})\right) : \mathbf{a}, \mathbf{b} \in \mathbb{F}^N \ \ \wedge \ \ P = \langle \mathbf{a}, \mathbf{g} \rangle + \langle \mathbf{b}, \mathbf{h} \rangle + \langle \mathbf{a}, \mathbf{b} \rangle \cdot u \right\}$$

The verifier samples a challenge $\xi_0 \leftarrow \mathbb{F}^*$; both parties set $u = \xi_0 \cdot G$ (a generator independent of $\mathbf{g}, \mathbf{h}$) and update the commitment:

$$P \ \leftarrow \ P - \mu H + \hat{t} \cdot u$$

Subtracting μH removes the blinding term from the $\mathscr{R}_{\mathsf{IP}}$ commitment, and adding $\hat{t} \cdot u$ encodes the claimed inner product in the u component. Since $\hat{t} = \langle \mathbf{a}, \mathbf{b} \rangle$, the updated statement is an instance of $\mathscr{R}_{\mathsf{IPA},n}$.

§07

Inner-Product Argument

The argument proves membership in $\mathscr{R}_{\mathsf{IPA},N}$ on length-N vectors, initially $N = n$. In each round, the prover and verifier combine the left and right halves of each vector into vectors of half the length. After $\lceil \log_2 n \rceil$ rounds, the prover has sent $2\lceil \log_2 n \rceil$ group elements and sends two final scalars.

Folding

Split each vector in half:

$$\begin{aligned} \mathbf{a} &= (\mathbf{a}_1 \parallel \mathbf{a}_2) \qquad & \mathbf{b} &= (\mathbf{b}_1 \parallel \mathbf{b}_2) \\ \mathbf{g} &= (\mathbf{g}_1 \parallel \mathbf{g}_2) \qquad & \mathbf{h} &= (\mathbf{h}_1 \parallel \mathbf{h}_2) \end{aligned}$$

The prover sends the two cross terms:

$$\begin{aligned} L &= \langle \mathbf{a}_1, \mathbf{g}_2 \rangle + \langle \mathbf{b}_2, \mathbf{h}_1 \rangle + \langle \mathbf{a}_1, \mathbf{b}_2 \rangle \cdot u \\ R &= \langle \mathbf{a}_2, \mathbf{g}_1 \rangle + \langle \mathbf{b}_1, \mathbf{h}_2 \rangle + \langle \mathbf{a}_2, \mathbf{b}_1 \rangle \cdot u \end{aligned}$$

The verifier replies with a challenge $\xi \leftarrow \mathbb{F}^*$, and both sides fold the generators and statement to half length:

$$\begin{aligned} \mathbf{g}' &= \xi^{-1} \cdot \mathbf{g}_1 + \xi \cdot \mathbf{g}_2 \\ \mathbf{h}' &= \xi \cdot \mathbf{h}_1 + \xi^{-1} \cdot \mathbf{h}_2 \\ P' &= \xi^2 \cdot L + P + \xi^{-2} \cdot R \end{aligned}$$

While the prover folds the witness the same way:

$$\mathbf{a}' = \xi \cdot \mathbf{a}_1 + \xi^{-1} \cdot \mathbf{a}_2$$
$$\mathbf{b}' = \xi^{-1} \cdot \mathbf{b}_1 + \xi \cdot \mathbf{b}_2$$

A simple calculation shows the relation is preserved, $P' = \langle \mathbf{a}', \mathbf{g}' \rangle + \langle \mathbf{b}', \mathbf{h}' \rangle + \langle \mathbf{a}', \mathbf{b}' \rangle \cdot u$. The folded statement and witness then form a smaller instance of the same relation, $\big((\mathbf{g}', \mathbf{h}', u, P'), (\mathbf{a}', \mathbf{b}')\big) \in \mathscr{R}_{\mathsf{IPA},N/2}$.

Base Case

After $\lceil \log_2 n \rceil$ rounds the vectors have length one, leaving the base relation $\mathscr{R}_{\mathsf{IPA},1}$. The prover sends the two scalars $\mathbf{a}, \mathbf{b} \in \mathbb{F}$, and the verifier accepts iff:

$$P = \mathbf{a} \cdot \mathbf{g} + \mathbf{b} \cdot \mathbf{h} + (\mathbf{a} \cdot \mathbf{b}) \cdot u$$

The verifier never folds the generators one round at a time: the round challenges $\xi_1, \ldots, \xi_{\lceil \log_2 n \rceil}$ assign each original generator a fixed coefficient (a product of the $\xi_j^{\pm 1}$), so the whole check collapses into a single multi-exponentiation, batched with the rest of verification.

Improvements

We describe two improvements to the protocol above: a cheaper arithmetization, and an inversion-free inner-product argument with a tighter and simpler proof of computational special soundness.

§08

Arithmetization

Recall that the full arithmetization spreads $t(X)$ across degree $2(n'+1) = 4c+6$, so the prover commits to all $3c+5$ of its coefficients. Most of that degree is wasted: it exists only to park the two halves of $A_{L|R}$ (and of S) on a shared monomial, so that they can be jointly committed; this is a bad 'optimization', that ends up hurting efficiency in the GBP setting. If we give every vector its own monomial and $t(X)$ collapses to degree $2(c+2)$, the prover commits to $2c+4$ coefficients, a saving of $c+1$. We pay for this by splitting $A_{L|R}$ into (A_L, A_R) and S into (S_L, S_R): two extra commitments, leaving a net saving of $c-1$ group elements. For 4 vector commitments the saving is 96 bytes: 3 fewer group elements.

The split buys something else: it tightens the relation. A vector commitment opens as:

$$A_C^{(k)} = \left\langle \mathbf{a_C}^{(k)}, \mathbf{G} \right\rangle + \left\langle \mathsf{aux}^{(k)}, \mathbf{H} \right\rangle + \gamma^{(k)} H$$

The $\left\langle \mathsf{aux}^{(k)}, \mathbf{H} \right\rangle$ term is slack the circuit cannot reach; the full arithmetization tolerates it, carrying it along in the $\mathbf{y} \circ \mathsf{aux}^{(k)}$ terms of $\mathbf{f_R}(X)$. However, the improved protocol below can ensure that all those terms are zero. The verifier evaluates the $\mathbf{G}$- and $\mathbf{H}$-sides separately and recombines them with a binding challenge r, drawn after the evaluation point x and before the opening. The improved protocol then proves the tighter relation $\mathscr{R}_{\mathsf{GBP}'}$, identical to the generalized Bulletproofs relation except that the vector commitments carry no $\mathbf{H}$ component:

$$\mathscr{R}_{\mathsf{GBP}'} = \left\{ \begin{array}{l} \big(\mathbf{a_L}, \mathbf{a_R}, \mathbf{a_O}, \mathbf{v}, \{\mathbf{a_C}^{(i)}, \gamma^{(i)}\}_{i=1}^{c}, \{\gamma_k\}_{k=1}^{m}\big) : \\ \mathbf{0} = W_L \cdot \mathbf{a_L} + W_R \cdot \mathbf{a_R} + W_O \cdot \mathbf{a_O} + \sum_{i=1}^{c} W_C^{(i)} \cdot \mathbf{a_C}^{(i)} + W_V \cdot \mathbf{v} + \mathbf{c} \\ \mathbf{0} = \mathbf{a_L} \circ \mathbf{a_R} - \mathbf{a_O} \\ \forall i.\ A_C^{(i)} = \left\langle \mathbf{a_C}^{(i)}, \mathbf{G} \right\rangle + \gamma^{(i)} \cdot H \\ \forall k.\ V_{(k)} = v_k \cdot G + \gamma_k \cdot H \end{array} \right\}$$

Public inputs are as in the arithmetization above; the target monomial is now X^{c+1}.

- **Prover $\to$ Verifier.**
  - Sample $\alpha_L, \alpha_R, \beta, \rho_L, \rho_R \leftarrow \mathbb{F}$ and $\mathbf{s_L}, \mathbf{s_R} \leftarrow \mathbb{F}^n$.
  - Commit:

$$\begin{aligned} S_L &= \langle \mathbf{s_L}, \mathbf{G} \rangle + \rho_L H \qquad & S_R &= \langle \mathbf{s_R}, \mathbf{H} \rangle + \rho_R H \\ A_L &= \langle \mathbf{a_L}, \mathbf{G} \rangle + \alpha_L H \qquad & A_R &= \langle \mathbf{a_R}, \mathbf{H} \rangle + \alpha_R H \\ A_O &= \langle \mathbf{a_O}, \mathbf{G} \rangle + \beta H \end{aligned}$$

  - Send $(A_L, A_R, A_O, S_L, S_R)$.
- **Verifier $\to$ Prover.**
  - Sample $y \leftarrow \mathbb{F}^*$ and $z \leftarrow \mathbb{F}$.
  - Send (y, z).
  - Both parties compute:
    - The challenge vectors $\mathbf{y}, (\mathbf{y})^{-1}, \mathbf{z}$
    - The weights $\mathbf{w_L}, \mathbf{w_R}, \mathbf{w_O}, \{\mathbf{w_C}^{(k)}\}, \mathbf{w_V}, w_c$
    - $\delta(y, z) = \left\langle (\mathbf{y})^{-1} \circ \mathbf{w_R}, \mathbf{w_L} \right\rangle$
- **Prover $\to$ Verifier.**
  - Recall that the inner products to batch are:

$$\begin{aligned} t_{c+1} &= \left\langle \mathbf{a_L} + \mathbf{w_R} \circ (\mathbf{y})^{-1}, \mathbf{y} \circ \mathbf{a_R} + \mathbf{w_L} \right\rangle + \langle \mathbf{a_O}, \mathbf{w_O} - \mathbf{y} \rangle + \sum_{k=1}^{c} \left\langle \mathbf{a_C}^{(k)}, \mathbf{w_C}^{(k)} \right\rangle \\ &= \delta(y, z) - w_c - \langle \mathbf{w_V}, \mathbf{v} \rangle. \end{aligned}$$

  - Build $\mathbf{f_L}(X), \mathbf{f_R}(X) \in \mathbb{F}[X]^n$ with nonzero coefficients:
    - $\mathbf{f_L}(X)$:
      - $\mathbf{a_L} + \mathbf{w_R} \circ (\mathbf{y})^{-1}$ at X^0
      - $\mathbf{a_O}$ at X^1
      - $\mathbf{a_C}^{(k)}$ at X^{k+1} for $k = 1, \ldots, c$
      - $\mathbf{s_L} + z^{q+1} \cdot \mathbf{1}$ at X^{c+2}
    - $\mathbf{f_R}(X)$:
      - $\mathbf{w_C}^{(k)}$ at X^{c-k} for $k = 1, \ldots, c$
      - $\mathbf{w_O} - \mathbf{y}$ at X^c
      - $\mathbf{y} \circ \mathbf{a_R} + \mathbf{w_L}$ at X^{c+1}
      - $\mathbf{y} \circ \mathbf{s_R}$ at X^{c+2}
    - Corresponding to generators in $\mathbf{G}$ and $\mathbf{H}$ respectively.
  - $t(X) = \langle \mathbf{f_L}(X), \mathbf{f_R}(X) \rangle = \sum_{i=0}^{2(c+2)} t_i X^i$; the target coefficient is t_{c+1}.
  - Sample $\tau_i \leftarrow \mathbb{F}$ for every $i \neq c+1$ and set $T_i = t_i G + \tau_i H$.
  - Send $\{T_i\}_{i \neq c+1}$.

- **Verifier $\to$ Prover.**
  - Sample $x \leftarrow \mathbb{F}$, then $r \leftarrow \mathbb{F}^*$.
  - Send (x, r).
- **Prover $\to$ Verifier.**
  - Opening of $t(x)$ (value $\hat{t}$, blinding τ_x):

$$\begin{aligned} \hat{t} &= \langle \mathbf{f_L}(x), \mathbf{f_R}(x) \rangle = t(x) \\ \tau_x &= \sum_{i \neq c+1} \tau_i x^i - x^{c+1} \langle \mathbf{w_V}, \gamma_V \rangle \end{aligned}$$

  - Opening of P (vectors $\mathbf{f_L}(x), \mathbf{f_R}(x)$, blinding $\mu = \mu_L + r\,\mu_R$):

$$\begin{aligned} \mu_L &= \alpha_L + \beta x + \sum_{k=1}^{c} \gamma^{(k)} x^{k+1} + \rho_L x^{c+2} \\ \mu_R &= \alpha_R x^{c+1} + \rho_R x^{c+2} \end{aligned}$$

    with γ_V the blinding factors of the $V_{(j)}$ and $\gamma^{(k)}$ the blinding factor of $A_C^{(k)}$.
  - Send $(\tau_x, \hat{t}, \mu, \mathbf{f_L}(x) \in \mathbb{F}^n, \mathbf{f_R}(x) \in \mathbb{F}^n)$.
- **Verifier.**
  - Set $\mathbf{H}^{(\mathbf{y})^{-1}} = (\mathbf{y})^{-1} \circ \mathbf{H}$.
  - Define:

$$\begin{aligned} \widetilde{W}_L &= \left\langle \mathbf{w_L}, \mathbf{H}^{(\mathbf{y})^{-1}} \right\rangle \in \mathbb{G} & \widetilde{W}_R &= \left\langle (\mathbf{y})^{-1} \circ \mathbf{w_R}, \mathbf{G} \right\rangle \in \mathbb{G} \\ \widetilde{W}_O &= \left\langle \mathbf{w_O} - \mathbf{y}, \mathbf{H}^{(\mathbf{y})^{-1}} \right\rangle \in \mathbb{G} & \widetilde{W}_k &= \left\langle \mathbf{w_C}^{(k)}, \mathbf{H}^{(\mathbf{y})^{-1}} \right\rangle \in \mathbb{G} \end{aligned}$$

  - Compute:

$$P = \underbrace{x^0 \left(A_L + \widetilde{W}_R \right) + x^1 A_O + \sum\nolimits_{k=1}^{c} x^{k+1} A_C^{(k)} + x^{c+2} \left(S_L + z^{q+1} \cdot \langle \mathbf{1}, \mathbf{G} \rangle \right)}_{\langle \mathbf{f_L}(x), \mathbf{G} \rangle + \mu_L H}$$

$$+ \underbrace{r \left(\sum\nolimits_{k=1}^{c} x^{c-k} \widetilde{W}_k + x^c \widetilde{W}_O + x^{c+1} \left(A_R + \widetilde{W}_L \right) + x^{c+2} S_R \right)}_{\left\langle \mathbf{f_R}(x), \mathbf{H}^{(\mathbf{y})^{-1}} \right\rangle + \mu_R H}$$

  - Check:

$$\hat{t}\, G + \tau_x H = x^{c+1} \left((\delta(y,z) - w_c) G - \sum\nolimits_{j=1}^{m} (\mathbf{w_V})_j \, V_{(j)} \right) + \sum_{i \neq c+1} x^i T_i$$

  - Check:

$$\left((\mathbf{G},\ r \cdot \mathbf{H}^{(\mathbf{y})^{-1}},\ H,\ P,\ \mu,\ \hat{t}),\ (\mathbf{f_L}(x), \mathbf{f_R}(x)) \right) \in \mathscr{R}_{\mathsf{IP}}$$

Computational Special Soundness

We prove that the protocol is an argument of knowledge for $\mathscr{R}_{\mathsf{GBP}'}$. The theorem below establishes $(n, q+2, 2c+5, 3)$-computational special soundness; by Attema, Fehr and Klooß (TCC 2022) the Fiat-Shamir compiled argument then has tight security: under the hardness of discrete log over $\mathbb{G}$ it is knowledge sound with knowledge error at most $(Q_{\mathsf{ro}}+1)\cdot(n+q+2c+6)/(|\mathbb{F}|-1)$, where Q_{ro} is the number of random-oracle queries.

Theorem. *Assume W_V has full column rank m, i.e. a left inverse. The improved arithmetization is $(n, q+2, 2c+5, 3)$-computationally special sound for $\mathscr{R}_{\mathsf{GBP}'}$: from any $(n, q+2, 2c+5, 3)$-tree of accepting transcripts a deterministic extractor returns either a witness for $\mathscr{R}_{\mathsf{GBP}'}$, or a non-trivial discrete-log relation among the generators $\Gamma = \mathbf{G} \parallel \mathbf{H} \parallel G \parallel H \in \mathbb{G}^{2n+2}$, i.e. a non-zero $\mathbf{v}$ with $\langle \mathbf{v}, \Gamma \rangle = 0$.*

The Tree of Accepting Transcripts. A $(n, q+2, 2c+5, 3)$-tree fixes the first message $(A_L, A_R, A_O, S_L, S_R)$ at the root and branches on the four challenges, recording each verifier message at its node: n distinct children on y; under each, $q+2$ distinct children on z, each with its $\{T_i\}$; under each, $2c+5$ distinct children on x; under each, 3 distinct **leaves** on r. The opening $(\tau_x, \mu, \mathbf{a}, \mathbf{b}) = (\tau_x, \mu_L + r\,\mu_R, \mathbf{f_L}(x), \mathbf{f_R}(x))$ is the never-sent final message decorating the leaves; drawn after r, it may differ across the r-children. Every root-to-leaf path is an accepting transcript. Index a (y, z, x)-node by (i_y, i_z, i_x) and its r-leaves by $e \in \{0, 1, 2\}$, with challenges r_e and leaf openings $(\tau_{x,e}, \mu_e, \mathbf{a}_e, \mathbf{b}_e)$. Write P_L, P_R for the two bracketed sums of the verifier's P (so $P = P_L + rP_R$) and $\mathbf{H}^{(\mathbf{y})^{-1}} = (\mathbf{y})^{-1} \circ \mathbf{H}$. The four arities are forced by the four interpolations below: $2c+5 = \deg t + 1$ for x; $q+2$ for the constraints together with the fresh offset degree; n for the Hadamard rows; 3 for the quadratic in r.

Step 1: Interpolation in r. Fix a node (i_y, i_z, i_x). The second verifier check at leaf e reads:

$$P_L + r_e P_R = \langle \mathbf{a}_e, \mathbf{G} \rangle + r_e \left\langle \mathbf{b}_e, \mathbf{H}^{(\mathbf{y})^{-1}} \right\rangle + \mu_e H$$

with μ_e the single blinding supplied at the leaf. The leaves $e = 0, 1$ (with $r_0 \neq r_1$) form a 2×2 system whose unique solution pins both sides to four **quadrant** vectors and recovers the two blinders:

$$P_L = \langle \mathbf{a}^\star, \mathbf{G} \rangle + \left\langle \boldsymbol{\beta}, \mathbf{H}^{(\mathbf{y})^{-1}} \right\rangle + \mu_L H$$
$$P_R = \langle \boldsymbol{\alpha}, \mathbf{G} \rangle + \left\langle \mathbf{b}^\star, \mathbf{H}^{(\mathbf{y})^{-1}} \right\rangle + \mu_R H$$

Where the quadrants are the interpolants:

$$\mathbf{a}^\star = \frac{r_1 \mathbf{a}_0 - r_0 \mathbf{a}_1}{r_1 - r_0} \qquad \boldsymbol{\beta} = \frac{r_0 r_1 (\mathbf{b}_0 - \mathbf{b}_1)}{r_1 - r_0}$$
$$\boldsymbol{\alpha} = \frac{\mathbf{a}_1 - \mathbf{a}_0}{r_1 - r_0} \qquad \mathbf{b}^\star = \frac{r_1 \mathbf{b}_1 - r_0 \mathbf{b}_0}{r_1 - r_0}$$

And the blinding factors by the same interpolation, $\mu_L = \frac{r_1\mu_0 - r_0\mu_1}{r_1 - r_0}$ and $\mu_R = \frac{\mu_1 - \mu_0}{r_1 - r_0}$. Here $\boldsymbol{\beta}$ is a $\mathbf{H}^{(\mathbf{y})^{-1}}$-mass sitting on the $\mathbf{G}$-side P_L, and $\boldsymbol{\alpha}$ a $\mathbf{G}$-mass on the $\mathbf{H}$-side P_R. The opening is sent *after* r, so nothing yet forces $\boldsymbol{\beta} = \boldsymbol{\alpha} = \mathbf{0}$; this is the slack the offset must remove.

Step 2: The $\hat{t}$-Quadratic Forces the Cross Terms to Vanish. On this family the e-th leaf satisfies $\mathbf{a}_e = \mathbf{a}^\star + r_e \boldsymbol{\alpha}$ and $\mathbf{b}_e = r_e^{-1} \boldsymbol{\beta} + \mathbf{b}^\star$ (leaves $0, 1$ by construction; leaf 2 either lies on the family — vectors and blinding alike — since its candidate below vanishes, or yields a relation). Hence:

$$\langle \mathbf{a}_e, \mathbf{b}_e \rangle = \langle \mathbf{a}^\star, \boldsymbol{\beta} \rangle r_e^{-1} + \big(\langle \mathbf{a}^\star, \mathbf{b}^\star \rangle + \langle \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle \big) + \langle \boldsymbol{\alpha}, \mathbf{b}^\star \rangle r_e$$

The *first* check at leaf e reads $\langle \mathbf{a}_e, \mathbf{b}_e \rangle\, G + \tau_{x,e}\, H = R$, with right-hand side $R = \sum_{i \neq c+1} x^i T_i + x^{c+1}\big((\delta(y,z) - w_c) G - \sum_j (\mathbf{w_V})_j V_{(j)} \big)$ determined before r is drawn. Subtracting the leaf-0 instance, $\big(\langle \mathbf{a}_e, \mathbf{b}_e \rangle - \langle \mathbf{a}_0, \mathbf{b}_0 \rangle \big) G + (\tau_{x,e} - \tau_{x,0}) H = 0$ is a discrete-log relation on (G, H); unless it is trivial — both

$\langle \mathbf{a}_e, \mathbf{b}_e \rangle$ and $\tau_{x,e}$ constant across the r-children — the extractor returns it. With $\langle \mathbf{a}_e, \mathbf{b}_e \rangle$ pinned to a single value, a map $r \mapsto Ar^{-1} + C + Br$ constant at the three distinct non-zero r_0, r_1, r_2 has $A = B = 0$; hence at every node:

$$\langle \mathbf{a}^\star, \boldsymbol{\beta} \rangle = 0, \qquad \langle \boldsymbol{\alpha}, \mathbf{b}^\star \rangle = 0, \qquad \langle \mathbf{a}_e, \mathbf{b}_e \rangle = \langle \mathbf{a}^\star, \mathbf{b}^\star \rangle + \langle \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle$$

Forcing both the r^{-1}- and r-terms to vanish requires three points, which fixes the r-arity at 3.

Step 3: x-Vandermonde Extraction. Fix y and z. Across the $2c + 5$ accepting transcripts that share them and use distinct x, each quadrant is the evaluation at x_{i_x} of a vector polynomial of degree $\leq 2c + 4$ whose coefficients are the per-degree representations of $P_L(X)$ and $P_R(X)$. Inverting the Vandermonde in x recovers, for each degree p, a representation in three blocks:

$$[\operatorname{coeff}_p P_L] = \langle \mathbf{g}_p, \mathbf{G} \rangle + \left\langle \mathbf{h}_p, \mathbf{H}^{(\mathbf{y})^{-1}} \right\rangle + \eta_p H$$

And likewise for P_R. Matching the layout of $\mathbf{f_L}$, the $\mathbf{G}$-blocks at degrees $0, 1, k+1, c+2$ are $\mathbf{a_L} + \mathbf{w_R} \circ (\mathbf{y})^{-1}$, $\mathbf{a_O}$, $\mathbf{a_C}^{(k)}$, and $\mathbf{s_L} + z^{q+1} \cdot \mathbf{1}$ (the mask *with* offset); the $\mathbf{H}^{(\mathbf{y})^{-1}}$-blocks are the residual components $\boldsymbol{\beta}$ at each degree.

Step 4: The Offset Eliminates the Residual Components. The residual $\mathbf{H}^{(\mathbf{y})^{-1}}$-block of each P_L-coefficient is the *same* fixed vector $\mathbf{w}_p$ for every (y, z) (up to the $\mathbf{y}$-rescaling), a property of the fixed commitments $A_L, A_O, A_C^{(k)}, S_L$. We show every $\mathbf{w}_p = \mathbf{0}$ by descending induction on p, from $c + 2$ down.

Step 2 gives $\langle \mathbf{a}^\star(x), \boldsymbol{\beta}(x) \rangle = 0$ at every x-child; interpolating across the $2c + 5$ points, every convolution coefficient of the two degree-$(c+2)$ quadrant polynomials vanishes. Take the coefficient at convolution degree $p_0 + c + 2$. With the higher-degree residual components already zero by the induction hypothesis, the only nonzero term pairs the $\mathbf{G}$-block of the mask slot $c + 2$ with $\mathbf{w}_{p_0}$; that block is $\mathbf{u} + z^{q+1} \cdot \mathbf{1}$, with $\mathbf{u}$ the fixed representation of S_L, so the coefficient is:

$$\langle \mathbf{u}, \mathbf{y} \circ \mathbf{w}_{p_0} \rangle + z^{q+1} \sum_t y^t (\mathbf{w}_{p_0})_t = 0$$

Fix y. The summand $D = \langle \mathbf{u}, \mathbf{y} \circ \mathbf{w}_{p_0} \rangle$ and the coefficient $B = \sum_t y^t (\mathbf{w}_{p_0})_t$ are independent of z: the weights live at $z^1, \ldots, z^q$, and $\mathbf{u}$ is fixed before z. The term $z^{q+1} B$ carries the fresh power, so $D + z^{q+1} B$ is a polynomial vanishing at the $q + 2$ distinct z, and full-degree Vandermonde forces $B = 0$. (Pairwise differences would *not* suffice: distinct z can share a $(q+1)$-th power, which is why the z-arity is $q + 2$, not $q + 1$.) Finally $\sum_t y^t (\mathbf{w}_{p_0})_t = 0$ across the n distinct y is a Vandermonde in y, so $\mathbf{w}_{p_0} = \mathbf{0}$ coordinate-wise.

Since every $\mathbf{w}_p = \mathbf{0}$, the residual quadrant $\boldsymbol{\beta}$ vanishes at every node, and each leaf's inner product equals the honest convolution $\langle \mathbf{a}^\star, \mathbf{b}^\star \rangle$.

Step 5: Witness or Relation. The extractor reads a candidate witness $\mathbf{W}$ off the Step-3 representations at a fixed reference (y, z) (the first y- and z-children): $\mathbf{a_L}$ from the $\mathbf{G}$-block of P_L at degree 0, minus the public $\mathbf{w_R} \circ (\mathbf{y})^{-1}$; $\mathbf{a_O}$ at degree 1; $\mathbf{a_C}^{(k)}$ and its blinder $\gamma^{(k)}$ from the $\mathbf{G}$- and H-blocks of P_L at degree $k + 1$; $\mathbf{a_R}$ from the $\mathbf{H}^{(\mathbf{y})^{-1}}$-block of P_R at degree $c + 1$, minus the public $\mathbf{w_L}$ and rescaled by $(\mathbf{y})^{-1}$; and $\mathbf{v}, \{\gamma_k\}$ from the first check's degree-$(c+1)$ coefficient across the first $q + 1$ z-children, via the precomputed left inverse of W_V. It also forms a list of candidate discrete-log relations: differences of two recovered representations of the same group element, e.g. the representation of A_L at (y_{i_y}, z_{i_z}) minus the one at the reference (y_0, z_0), the high-degree coefficients, and the per-node consistency checks — the third r-child's opening and blinding against the family interpolated from the first two (Step 1), and the agreement of $\left(\langle \mathbf{a}_e, \mathbf{b}_e \rangle, \tau_{x,e}\right)$ across the r-children (Step 2). Each candidate $\mathbf{c}$ satisfies $\langle \mathbf{c}, \Gamma \rangle = 0$ by construction, being the difference of two representations of the same commitment; it is therefore a discrete-log relation, *non-trivial* exactly when $\mathbf{c} \neq \mathbf{0}$.

The extractor outputs $\mathbf{W}$ if $\mathbf{W} \in \mathscr{R}_{\mathsf{GBP}'}$, and otherwise the first non-zero candidate relation; we argue the contrapositive. If every candidate vanishes, the Step-3 representations and the per-node identities of Step 2 hold, and the residual components are pinned to the single family $\mathbf{w}_p$, so Step 4 forces every $\mathbf{w}_p$ to zero. With the leaf openings now honest, cancelling δ in the first-check identity $t_{c+1} = \delta(y, z) - w_c - \langle \mathbf{w_V}, \mathbf{v} \rangle$ gives, for each (y, z):

$$\langle \mathbf{y}, \mathbf{a_L} \circ \mathbf{a_R} - \mathbf{a_O} \rangle + \sum_{\ell=1}^{q} z^\ell \big(W_L \mathbf{a_L} + W_R \mathbf{a_R} + W_O \mathbf{a_O} + \textstyle\sum_k W_C^{(k)} \mathbf{a_C}^{(k)} + W_V \mathbf{v} + \mathbf{c} \big)_\ell = 0$$

Deaggregating over the n distinct y and the $q+1$ distinct z (two Vandermondes) splits this into the first two relation clauses:

$$W_L \mathbf{a_L} + W_R \mathbf{a_R} + W_O \mathbf{a_O} + \textstyle\sum_k W_C^{(k)} \mathbf{a_C}^{(k)} + W_V \mathbf{v} + \mathbf{c} = \mathbf{0}, \qquad \mathbf{a_L} \circ \mathbf{a_R} - \mathbf{a_O} = \mathbf{0}$$

The vector-commitment clause is *tight*: since the residual components vanish, the degree-$(k+1)$ $\mathbf{G}$-block representation gives $A_C^{(k)} = \langle \mathbf{a_C}^{(k)}, \mathbf{G} \rangle + \gamma^{(k)} H$, with no $\mathbf{H}$-component. The scalar clause $V_{(k)} = v_k G + \gamma_k H$ follows from the degree-$(c+1)$ recovery and the W_V left inverse. All four clauses hold, so $\mathbf{W} \in \mathscr{R}_{\mathsf{GBP}'}$, contradicting the assumption. Hence a failing $\mathbf{W}$ yields a non-zero candidate, which the extractor returns as the discrete-log relation.

§09

Inner-Product Argument

The improved arithmetization reduces to the same inner-product relation $\mathscr{R}_{\mathsf{IP}}$, so it can reuse the inner-product argument above. Folding in the value yields the same relation $\mathscr{R}_{\mathsf{IPA},N}$, and the base case is unchanged; we change only the folding round, replacing the Laurent fold with a *polynomial* fold. This removes every field inversion, so the round challenges range over all of $\mathbb{F}$ rather than $\mathbb{F}^*$.

Folding

Split each vector in half:

$$\begin{aligned} \mathbf{a} &= (\mathbf{a}_1 \parallel \mathbf{a}_2) & \mathbf{b} &= (\mathbf{b}_1 \parallel \mathbf{b}_2) \\ \mathbf{g} &= (\mathbf{g}_1 \parallel \mathbf{g}_2) & \mathbf{h} &= (\mathbf{h}_1 \parallel \mathbf{h}_2) \end{aligned}$$

The prover sends the two cross terms:

$$\begin{aligned} L &= \langle \mathbf{a}_1, \mathbf{g}_2 \rangle + \langle \mathbf{b}_2, \mathbf{h}_1 \rangle + \langle \mathbf{a}_1, \mathbf{b}_2 \rangle \cdot u \\ R &= \langle \mathbf{a}_2, \mathbf{g}_1 \rangle + \langle \mathbf{b}_1, \mathbf{h}_2 \rangle + \langle \mathbf{a}_2, \mathbf{b}_1 \rangle \cdot u \end{aligned}$$

The verifier replies with a challenge $\xi \leftarrow \mathbb{F}$, and both sides fold the generators and statement to half length:

$$\begin{aligned} \mathbf{g}' &= \mathbf{g}_1 + \xi \cdot \mathbf{g}_2 \\ \mathbf{h}' &= \xi \cdot \mathbf{h}_1 + \mathbf{h}_2 \\ P' &= \xi \cdot P + \xi^2 \cdot L + R \end{aligned}$$

While the prover folds the witness the same way:

$$\begin{aligned} \mathbf{a}' &= \xi \cdot \mathbf{a}_1 + \mathbf{a}_2 \\ \mathbf{b}' &= \mathbf{b}_1 + \xi \cdot \mathbf{b}_2 \end{aligned}$$

A simple calculation shows the relation is preserved, $P' = \langle \mathbf{a}', \mathbf{g}' \rangle + \langle \mathbf{b}', \mathbf{h}' \rangle + \langle \mathbf{a}', \mathbf{b}' \rangle \cdot u$. The folded statement and witness then form a smaller instance of the same relation, $\big((\mathbf{g}', \mathbf{h}', u, P'), (\mathbf{a}', \mathbf{b}') \big) \in \mathscr{R}_{\mathsf{IPA},N/2}$. Observe that, unlike the original version, the field element ξ may take any value in $\mathbb{F}$.

Base Case

After $\lceil \log_2 n \rceil$ rounds the vectors have length one, leaving the base relation $\mathscr{R}_{\mathsf{IPA},1}$. The prover sends the two scalars $\mathbf{a}, \mathbf{b} \in \mathbb{F}$, and the verifier accepts iff:

$$P = \mathbf{a} \cdot \mathbf{g} + \mathbf{b} \cdot \mathbf{h} + (\mathbf{a} \cdot \mathbf{b}) \cdot u$$

§10

Findings

Below are listed the findings found during the engagement. High severity findings can be seen as so-called "priority 0" issues that need fixing (potentially urgently). Medium severity findings are most often serious findings that have less impact (or are harder to exploit) than high-severity findings. Low severity findings are most often exploitable in contrived scenarios, if at all, but still warrant reflection. Findings marked as informational are general comments that did not fit any of the other criteria.

LOW

#00 Repeated `LinComb` Additions Can Grow the Sparse Weight Vectors Exponentially
generalized-bulletproofs/src/lincomb.rs

#01 Commitment Has Secret-Dependent Memory Access Patterns
generalized-bulletproofs/src/arithmetic_circuit_proof.rs

#02 Secret Data Not Zeroized After Vector Reallocation
generalized-bulletproofs/src/lib.rs

#03 Scalar Commitment Are Not Extractable for Rank-Deficient `W_V`
generalized-bulletproofs/src/arithmetic_circuit_proof.rs

INFORMATIONAL

#04 Generator Validation for `g` Is Skipped in Release Builds
generalized-bulletproofs/src/lib.rs

#05 Inner Product Prover Accepts Generator Vectors With Non-Canonical Length
generalized-bulletproofs/src/inner_product.rs

#06 Vector Commitment Tail Coordinates Are Unconstrained Unless Explicitly Restricted
crypto/generalized-bulletproofs/src/lib.rs

• LOW · #00

Repeated `LinComb` Additions Can Grow the Sparse Weight Vectors Exponentially

generalized-bulletproofs/src/lincomb.rs

Description. `LinComb` stores its weights in sparse vectors (`WL` , `WR` , `WO` , `WV`) and a sparse map (`WCG`). Addition concatenates the operand's entries onto these vectors rather than coalescing terms that share an index.

```
fn add(mut self, constraint: &Self) -> Self {
  self.reconcile_for_merging(constraint);

  self.WL.extend(&constraint.WL);
  self.WR.extend(&constraint.WR);
  self.WO.extend(&constraint.WO);
  for (i, sparse_vec) in &constraint.WCG {
    self
      .WCG
      .entry(*i)
      .and_modify(|existing| existing.extend(sparse_vec))
      .or_insert_with(|| sparse_vec.clone());
  }
  self.WV.extend(&constraint.WV);
  self.c += constraint.c;
  self
}
```

Observe that the size of the result is the sum of the sizes of the operands. Adding a combination to itself doubles its length, so a loop that repeatedly accumulates a combination into itself grows the stored representation exponentially in the number of iterations.

```
for _ in 0 .. n {
  a = a + &a;
}
```

After `n` iterations the `WL` / `WR` / `WO` / `WV` vectors hold on the order of 2^n entries, even though the combination is mathematically equivalent to a far smaller one.

Impact. This does not affect correctness or soundness; duplicated entries are accumulated additively when the constraint is evaluated, so the result is unchanged. It's a potential denial of service concern *if* the library is used in a setting where the adversary can cause the verifier to try and verify a circuit of this form.

Recommendation. If linear combinations are built from programmatic or untrusted inputs, coalesce terms sharing an index during `add` / `sub` , or document that the sparse representation is not reduced and that callers should make sure that the constructed circuit does not trigger this behavior.

LOW · #01

Commitment Has Secret-Dependent Memory Access Patterns

generalized-bulletproofs/src/arithmetic_circuit_proof.rs

Description. The prover forms its commitments A_I, A_O, and S with the constant-time `multiexp` over witness-derived scalars (`aL` , `aR` , `aO` , the blinding vectors `sL` , `sR` , and the masks `alpha` , `beta` , `rho`).

```
let AI = {
  let alg = witness.aL.0.iter().enumerate().map(|(i, aL)| (*aL, self.generators.g_bold(i)));
  let arh = witness.aR.0.iter().enumerate().map(|(i, aR)| (*aR, self.generators.h_bold(i)));
  let ah = core::iter::once((alpha, self.generators.h()));
  let mut AI_terms = alg.chain(arh).chain(ah).collect::<Vec<_>>();
  let AI = multiexp(&AI_terms);
  AI_terms.zeroize();
  AI
};
```

`multiexp` is the constant-time variant, but it is not constant time with respect to the scalars in the sense needed to hide the witness. It uses Pippenger's algorithm, whose bucket accumulation indexes memory using bits of the secret scalars. Memory accesses are not constant time in general, so the access pattern leaks information about the scalars and cache attacks such as flush-and-reload apply. A scan over all buckets would close this, but that is not what the algorithm does.

The issue is in the upstream `multiexp` dependency used by the `generalized-bulletproofs` library. The upstream project has already fixed the issue, but the fixed version has not yet been published to `crates.io` .

Impact. This is not a verifier-soundness issue and is not exploitable from the proof bytes. It is a local side-channel risk: an attacker who can observe the prover's memory access pattern, e.g. a co-tenant on the same machine, can learn information about the witness scalars. Whether this matters depends on whether the proofs are used in a witness-hiding setting and on the prover's execution environment.

Recommendation. Document that proof generation assumes no local microarchitectural side-channel adversary and that witness privacy is out of scope for that environment, or update to the fixed upstream `multiexp` version once it is available.

LOW · #02

Secret Data Not Zeroized After Vector Reallocation

generalized-bulletproofs/src/lib.rs

Description. `PedersenVectorCommitment::commit` builds the input to `multiexp` in a temporary `terms` vector. The vector contains `self.mask` and the committed witness values from `self.g_values`.

```
impl<C: Ciphersuite> PedersenVectorCommitment<C> {
  /// Commit to the vectors of values.
  ///
  /// This function returns `None` if the amount of generators is less than the amount of values
  /// within the relevant vector.
  pub fn commit(&self, g_bold: &[C::G], h: C::G) -> Option<C::G>
  where
    C::G: ConditionallySelectable,
  {
    if g_bold.len() < self.g_values.len() {
      None?;
    }

    let mut terms = vec![(self.mask, h)];
    for pair in self.g_values.iter().copied().zip(g_bold.iter().copied()) {
      terms.push(pair);
    }
    let res = multiexp(&terms);
    terms.zeroize();
    Some(res)
  }
}
```

The `terms` vector is initialized with one element and then extended in the loop. Since the vector does not reserve space for all entries upfront, pushing witness pairs can force one or more reallocations. During a reallocation, Rust allocates a larger buffer, copies the existing elements into the new buffer, and releases the previous buffer. The old allocation is not zeroized before being released back to the allocator.

As a result, stale copies of `self.mask` and earlier `self.g_values` entries may remain in freed heap memory even though the final `terms` allocation is zeroized after `multiexp` returns. The explicit `terms.zeroize()` call is therefore incomplete for copies created by vector growth.

Impact. This does not affect the correctness, soundness, or hiding properties of the commitment scheme. It is a local secret-handling issue: witness scalars may remain in process memory longer than intended and could be recovered only if an attacker can read freed heap memory, inspect a memory dump, or otherwise obtain process memory. Zeroization is a best effort with Rust zeroize library, and the compiler or runtime may introduce other copies outside this function, but the vector reallocation pattern is a concrete and avoidable source of stale witness data.

Recommendation. It is recommended to pre-allocate enough space for `terms` before pushing witness values, so the vector does not reallocate during construction.

```
pub fn commit(&self, g_bold: &[C::G], h: C::G) -> Option<C::G>
where
  C::G: ConditionallySelectable,
{
  if g_bold.len() < self.g_values.len() {
    None?;
  }

  let mut terms = Vec::with_capacity(self.g_values.len() + 1);
  terms.push((self.mask, h));
  for pair in self.g_values.iter().copied().zip(g_bold.iter().copied()) {
    terms.push(pair);
  }
```

```
    let res = multiexp(&terms);
    terms.zeroize();
    Some(res)
}
```

• LOW · #03

Scalar Commitment Are Not Extractable for Rank-Deficient `W_V`

generalized-bulletproofs/src/arithmetic_circuit_proof.rs

Description. The fixed Generalized Bulletproofs relation treats the scalar-commitment weight matrix `W_V` as having full column rank, so each scalar commitment $V_k = v_k \cdot G + \gamma_k \cdot H$ is individually bound by the constraints. `ArithmeticCircuitStatement::new` does not enforce this; it only checks that the indices referenced by each constraint are in bounds.

```
for constraint in &constraints {
  if Some(generators.len()) <= constraint.highest_a_index {
    Err(AcStatementError::ConstrainedNonExistentTerm)?;
  }
  if Some(C.len()) <= constraint.highest_c_index {
    Err(AcStatementError::ConstrainedNonExistentVectorCommitment)?;
  }
  if Some(V.len()) <= constraint.highest_v_index {
    Err(AcStatementError::ConstrainedNonExistentCommitment)?;
  }
}
```

Both the prover and the verifier access the scalar commitments only through the `z`-aggregated combination of the `W_V` rows, so the proof binds the single combination $\mathbf{z}^\top W_V \cdot \mathbf{V}$ rather than the individual commitments V_k.

Observe that if `W_V` is not full column rank, any commitment in the right kernel of `W_V` is invisible to the verifier. The only code that checks each witness opening against its public commitment is gated behind `#[cfg(debug_assertions)]`, so a release build does not catch a bogus opening for such a commitment.

This is a known restriction inherited from the original Bulletproofs relation, not specific to this implementation: a rank-deficient `W_V` binds only a linear combination of the commitments, so the individual openings cannot be extracted.

Consider two public commitments $V_1 = v_1 G + \gamma_1 H$ and $V_2 = v_2 G + \gamma_2 H$, with a single constraint asserting their values sum to some target, $v_1 + v_2 = v_3$. The weight row over (V_1, V_2) is $[1, 1]$: rank one for two commitments, so $Q = 1 < m = 2$. The proof binds only $v_1 + v_2$ and $\gamma_1 + \gamma_2$, which is the opening of the sum $V_1 + V_2$; it says nothing about V_1 and V_2 individually.

A prover therefore need not know either opening. Pick V_1 to be an arbitrary point with no known opening, then set $V_2 = (v_3 G + \gamma_3 H) - V_1$ for a chosen opening (v_3, γ_3). Now $V_1 + V_2 = v_3 G + \gamma_3 H$, so the prover proves $v_1 + v_2 = v_3$ from the opening of the sum alone, while knowing no opening of V_1 or V_2. The right kernel of `W_V` is spanned by $(1, -1)$: the prover can slide value and mask freely between the two commitments, including pushing V_1 onto a point it cannot open.

The extreme case is a self-cancelling column. The degenerate constraint `V(0) - V(0) = 0` zeroes the entire column for that commitment, dropping it from every verifier equation; since the per-opening check is `#[cfg(debug_assertions)]`-gated, a release build then accepts a proof whose witness does not open the public commitment at all.

Impact. This is a defense-in-depth issue at the statement-construction boundary; it does not affect a caller using fixed, full-rank constraints. An integration that passes externally supplied Pedersen commitments in `V` and relies on the proof to establish knowledge of each opening, without itself ensuring `W_V` has full column rank, does not get that guarantee: the extractor cannot recover the openings of commitments outside the column span of `W_V`.

In particular, such a proof cannot stand in for a Schnorr-style proof of knowledge of a commitment opening, for example to spend a commitment by proving the prover can open it. When `W_V` is rank-deficient the proof establishes knowledge of an aggregate opening only, not of the individual commitments the application believes were proven.

Recommendation. If `ArithmeticCircuitStatement::new` does not enforce that `w_v` has full column rank, this limitation must be documented clearly: the proof binds only the aggregated combination $W_V \cdot \mathbf{V}$, not the individual scalar commitment openings. Callers that rely on per-commitment binding, for example to prove knowledge of a commitment opening, must ensure `w_v` has full column rank themselves.

INFORMATIONAL · #04

Generator Validation for `g` Is Skipped in Release Builds

generalized-bulletproofs/src/lib.rs

Description. `Generators::new` uses a local `add_generator` helper to reject identity points and track duplicate generators.

```
let mut set = HashSet::new();
let mut add_generator = |generator: &C::G| {
  if bool::from(generator.is_identity()) {
    return Err(GeneratorsError::IdentityPoint);
  }
  let bytes = generator.to_bytes();
  Ok(!set.insert(bytes.as_ref().to_vec()))
};

debug_assert!(!add_generator(&g)?, "g was prior present in empty set");
if add_generator(&h)? {
  Err(GeneratorsError::DuplicatedGenerator)?;
}
```

The call to `add_generator(&g)?` is placed inside `debug_assert!`. In release builds, `debug_assert!` expressions are compiled out, so this call is not evaluated. As a result, `g` is not checked for being the identity and is not inserted into the duplicate-detection set.

The following checks still validate `h`, `g_bold`, and `h_bold`, but they no longer compare those generators against `g` in release builds because `g` was never inserted into `set`. This makes the production validation weaker than the validation performed in debug builds.

Impact. Invalid generator sets involving `g` can be accepted in release builds even though they would be rejected in debug builds. The practical impact depends on how the library is integrated and whether generator inputs can be influenced by an untrusted party. At minimum, this creates a configuration validation mismatch.

Recommendation. Evaluate `add_generator(&g)?` unconditionally and return a normal error instead of relying on `debug_assert!`.

```
if add_generator(&g)? {
  Err(GeneratorsError::DuplicatedGenerator)?;
}
```

The duplicate and identity checks are best-effort validation. They do not prove that the accepted bases are independently generated, or that no party knows discrete-log relations between them. It is recommended to document `Generators::new` as accepting only already-trusted independent bases, with these checks serving as additional defensive validation.

INFORMATIONAL · #05

Inner Product Prover Accepts Generator Vectors With Non-Canonical Length

generalized-bulletproofs/src/inner_product.rs

Description. `IpStatement::prove` checks that the prover has at least enough generators for the witness length rounded up to the next power of two.

```
// Ensure we have the necessary amount of generators
if witness.a.len() > ((usize::MAX >> 1) + 1) {
  Err(IpProveError::IncorrectAmountOfGenerators)?;
}
if generators.g_bold_slice().len() < witness.a.len().next_power_of_two() {
  Err(IpProveError::IncorrectAmountOfGenerators)?;
}
```

This permits `generators.g_bold_slice().len()` to be larger than `witness.a.len().next_power_of_two()`. However, the folding loop later treats `g_bold.len()` as the inner product dimension, while it splits the witness vectors using `a.len().next_power_of_two() / 2`.

```
// This interprets `g_bold.len()` as `n`
while g_bold.len() > 1 {
  // Split a, b, g_bold, h_bold as needed for lines 20-24
  let split_at = a.len().next_power_of_two() / 2;
  let (a1, a2) = a.split(split_at);
  let (b1, b2) = b.split(split_at);

  let (g_bold1, g_bold2) = g_bold.split();
  let (h_bold1, h_bold2) = h_bold.split();

  let n_hat = g_bold1.len();

  // Sanity
  debug_assert_eq!(a1.len(), n_hat);
  debug_assert!(a2.len() <= n_hat);
  debug_assert_eq!(b1.len(), n_hat);
  debug_assert!(b2.len() <= n_hat);
  debug_assert_eq!(g_bold1.len(), n_hat);
  debug_assert_eq!(g_bold2.len(), n_hat);
  debug_assert_eq!(h_bold1.len(), n_hat);
  debug_assert_eq!(h_bold2.len(), n_hat);

  [...]
}
```

The debug assertions show that the intended invariant is stronger than the initial check: the generator vector length should match the padded witness length. If extra generators are accepted, the witness vectors and generator vectors can be folded under different dimensions. In release builds, the debug assertions are not enforced.

Impact. This is an input validation issue in the prover. Under the expected canonical parameters, where the generator vector length equals the padded witness length, there is no impact. If `prove` is called with an oversized generator vector, it can operate on inconsistent dimensions and may produce an invalid proof or fail later in the folding process instead of returning `IpProveError::IncorrectAmountOfGenerators` at the boundary.

Recommendation. It is recommended to require the generator vector length to be exactly `witness.a.len().next_power_of_two()` before entering the folding loop.

```
if generators.g_bold_slice().len() != witness.a.len().next_power_of_two() {
  Err(IpProveError::IncorrectAmountOfGenerators)?;
```

```
}
```

INFORMATIONAL · #07

Vector Commitment Tail Coordinates Are Unconstrained Unless Explicitly Restricted

crypto/generalized-bulletproofs/src/lib.rs

Description. In the generalized Bulletproof construction, vector commitments are expressed over the same `G_bold` and `H_bold` generator vectors used by the proof. The length of these generator vectors is tied to the proof dimension, which is derived from the number of gates and padded to a power of two for the inner product argument.

This proof dimension can be larger than the logical vector length used by an application. For example, an application may intend to commit to a vector of length 3, while the generalized Bulletproof instance uses 4 generators after power-of-two padding.

The vector commitment API accepts an opening as long as there are enough generators for the provided values.

```
pub fn commit(&self, g_bold: &[C::G], h: C::G) -> Option<C::G>
where
  C::G: ConditionallySelectable,
{
  if g_bold.len() < self.g_values.len() {
    None?;
  }

  let mut terms = vec![(self.mask, h)];
  for pair in self.g_values.iter().copied().zip(g_bold.iter().copied()) {
    terms.push(pair);
  }
  let res = multiexp(&terms);
  terms.zeroize();
  Some(res)
}
```

The generalized Bulletproof circuit then constrains only the coordinates that appear in the arithmetic constraints. This is expected for the `H_bold` part of the vector commitment, which represents auxiliary opening data that is not accessible from the circuit. However, the same issue can affect the `G_bold` part when the proof dimension is larger than the application's intended vector length: tail coordinates past the logical vector length are not forced to be zero by default.

As a result, an application that reasons about only a fixed prefix of the vector commitment must ensure that every unused `G_bold` coordinate is either absent from the opening or explicitly constrained to zero. Otherwise, a prover can use non-zero values in the padded tail while still satisfying all constraints over the intended prefix.

Impact. This is an integration-level soundness concern. The proof remains valid for the generalized Bulletproof relation that was actually encoded, but that relation may be wider than the application intended. If an application treats a vector commitment as an exact-length commitment and does not constrain padded tail coordinates, the commitment can contain additional non-zero data outside the checked prefix.

Recommendation. It is recommended to document that generalized Bulletproof vector commitments are over the proof dimension, not necessarily the application-level vector length, and that callers are responsible for constraining unused tail coordinates.

Is is also recommended to make the logical vector length explicit wherever vector commitments are used in generalized Bulletproof statements. For exact-length vectors, add constraints forcing every padded `G_bold` coordinate beyond the logical length to be zero, or provide an API helper that automatically inserts these constraints.