

Symplex: Improved Pairing-Based zkSNARK using Partial Fraction Techniques

Charanjit S. Jutla and Arnab Roy

¹ csjutla@us.ibm.com

² arnabr@gmail.com

Abstract. We present *Symplex*, a pairing-based zkSNARK for R1CS that preserves the syntax of Groth16: a $2\mathbb{G}_1+1\mathbb{G}_2$ proof, and a verifier with three pairings and one public-input multi-scalar multiplication (MSM), while *strictly reducing prover cost*. For constraint count n , wire count m , public-input count ℓ , and $\kappa = \min\{n, m+1\}$, *Symplex*’s prover uses four FFTs of size n rather than the six of our coset-Lagrange Groth16 comparator, and its larger $\mathbb{G}_1$ -MSM has width $m+n-\ell+4$ rather than the adaptively based Groth16 width $2\kappa+m+n-\ell+3$.

The matched Groth16 prover derives three length- n vectors from the R1CS instance, converts each between coefficient and evaluation form, and commits the quotient with a size- n coset-Lagrange column, resulting in six FFTs in total. Its usual monomial quotient column has one fewer CRS element but requires a final inverse FFT.

Symplex instead uses the partial-fraction techniques of Jutla, Nema, Roy (EuroCrypt 2026) allowing the output-wire selector polynomials to be precomputed into the per-wire CRS bundles during setup. So, only the left- and right-wire vectors require FFT-based processing at proving time. The verifier is unchanged (three pairings, one public-input MSM). On a BLS12-381 prototype of *Symplex* that shares sparse R1CS, FFT, and pairing code with Groth16, a Circom circuit chaining SHA-256 ten times gives a $1.84\times$ prover speedup with online verification at ≈ 2.6 ms for both schemes. The uniform algebraic extraction argument, perfect completeness, and perfect zero-knowledge are machine-checked in Lean 4.

Polymath (Lipmaa, CRYPTO 2024) and PARI (Dellepere, Mishra, and Shirzad, USENIX Security 2026) shrink the proof using a squared R1CS arithmetization. Polymath proves soundness in the AGMOS model, and PARI does not provide zero-knowledge in the stated construction. *Symplex* keeps Groth16’s proof shape and standard R1CS arithmetization, and is proved knowledge-sound in the standard Generic Group Model.

Keywords: zkSNARK · Groth16 · Partial Fractions · Cauchy Matrices

1 Introduction

Zero-knowledge succinct non-interactive arguments of knowledge (zkSNARKs) [Mic00, GGPR13, PHGR13, Gro16] over pairing-friendly groups are a standard

tool in practical cryptography. Among them, Groth16 [Gro16] remains the standard proof system for rank-one constraint systems (R1CS), the arithmetization derived from the quadratic arithmetic programs (QAPs) of [GGPR13], with circuit-specific trusted setup. Its proof is three group elements ($2\mathbb{G}_1 + 1\mathbb{G}_2$), its online verifier consists of three pairings and one public-input multi-scalar multiplication (MSM), and it offers perfect zero-knowledge. Groth16 has anchored production deployments for nearly a decade, from the shielded payments of Zcash Sapling [HBHW24] and Loopring’s zk-rollup [DWF⁺20] to zkLogin blockchain authentication [BCJ⁺24]. Other approaches to zkSNARKS which can be transparent and post-quantum-friendly are based on interactive oracle proofs [BCS16] and STARKs [BBHR18]. These offer setup-freeness but at the cost of larger proofs and longer verification time. Our work is in the pairing-friendly group setting.

The prover in Groth16, however, pays for its minimal verifier. The Groth16 prover in `arkworks` [ark22] and `snarkjs` [ide24] performs three forward and three inverse size- n FFTs to evaluate the critical QAP quotient $h(X)$ of [GGPR13, Gro16], followed by a final inverse FFT for its usual monomial $t(x)/\delta$ column. More specifically, it switches the three wire-indexed evaluations $\mathbf{ua}$, $\mathbf{va}$, $\mathbf{wa}$ between coefficient and evaluation form, and assembles a $\mathbb{G}_1$ -MSM of width $3m + (n - \ell) + 4$ using Pippenger’s algorithm [Pip80]. For the matched comparison in this paper, we give Groth16 a size- n coset-Lagrange $t(x)/\delta$ column: this adds one $\mathbb{G}_1$ basis element, makes the corresponding width $3m + (n - \ell) + 5$, and removes the final inverse FFT, leaving six FFTs total [CJL⁺26]. These costs motivate several recent lines of work, but their objectives and arithmetizations differ. Section 2 gives a dedicated account of the partial-fraction antecedent, short-proof systems over squaring arithmetizations, and dynamic Groth16 variants. It also states the parameter conventions needed to compare them without conflating unlike prover costs.

Our contribution: Symplex. We present **Symplex**,³ a pairing-based zkSNARK for R1CS that matches Groth16 on proof structure ($2\mathbb{G}_1 + 1\mathbb{G}_2$), arithmetization, perfect zero-knowledge, and verifier cost (three online pairings and one public-input MSM, without any field-element opening and without any public-input interpolation), while reducing prover cost. **Symplex**’s prover performs four FFTs of size n instead of the matched Groth16 comparator’s six (Section 6.2) and a $\mathbb{G}_1$ -MSM of width $n + (m - \ell) + 4$ in place of Groth16’s adaptively based width $2\kappa + m + (n - \ell) + 3$, where $\kappa = \min\{n, m + 1\}$ (Table 2). Knowledge-soundness is proved in the standard GGM [Sho97] (Section 7.3), without relying on AGMOS and, like Groth16, without random oracle or Fiat–Shamir transform. The uniform algebraic extraction argument, perfect completeness, and perfect honest-verifier zero-knowledge have additionally been machine-checked in Lean 4 (Section 9).

³ The name hints informally at a link to *symplectic* structure: the Cauchy matrix $\mathbf{C}$ of Section 6.1 is a skew-symmetric bilinear form and is a core engine of our construction. We do not further develop this viewpoint here.

Technical idea. An R1CS instance is specified by matrices L, R, O (representing resp. left-wire, right-wire and output-wire incidence graphs of constraints) and accepts a wire vector $\mathbf{a} = (a_0, \dots, a_m)$ with $a_0 = 1$ iff, at every constraint $\alpha \in \{1, \dots, n\}$,

$$\ell_\alpha r_\alpha = o_\alpha, \quad \ell_\alpha := L_{\alpha,*} \mathbf{a}, \quad r_\alpha := R_{\alpha,*} \mathbf{a}, \quad o_\alpha := O_{\alpha,*} \mathbf{a}.$$

Thus satisfiability is an *entry-wise* (Hadamard) product identity over the n constraints (over some finite field $\mathbb{F}$).

What Groth16 does. Groth16 [Gro16] rewrites this point-wise identity as a single polynomial equation (using Lagrange interpolation) over an auxiliary set of points $\{\rho_1, \dots, \rho_n\}$: one builds polynomials u_i, v_i, w_i of degree $< n$ with $u_i(\rho_j) = L_{j,i}$, $v_i(\rho_j) = R_{j,i}$, $w_i(\rho_j) = O_{j,i}$, sets $t(X) = \prod_j (X - \rho_j)$, and proves

$$\left(\sum_i a_i u_i(X) \right) \left(\sum_i a_i v_i(X) \right) = \sum_i a_i w_i(X) + h(X) t(X)$$

for some quotient h . The last term is zero if one can work in the quotient ring $\mathbb{F}[X]/(t(X))$. Since this quotient representation is not easy to prove in the exponent, Groth16 explicitly calculates h and *linearly*⁴ computes and adds $h(X) * t(X)$ from one proof element. The prover must therefore work with three length- n vectors (the evaluations of the left, right, and output legs) and move each between coefficient and evaluation form so that multiplication and division by $t(X)$ can be done point-wise. This repeated coefficient-evaluation conversion on all three legs drives six size- n FFTs. A prover using the usual monomial quotient column needs one further inverse FFT; our matched Groth16 comparator instead uses a coset-Lagrange quotient column so that its total remains six (Remark 2).

What Symplex does. Symplex keeps the Hadamard-product structure explicit in a *partial-fraction* basis anchored at poles $\{\alpha_1, \dots, \alpha_n\}$, one for each constraint, rather than an FFT evaluation point-set. For each wire ι , define formal rational functions

$$u_\iota(X) := \sum_\alpha \frac{L_{\alpha,\iota}}{X-\alpha}, \quad v_\iota(X) := \sum_\alpha \frac{R_{\alpha,\iota}}{X-\alpha}, \quad w_\iota(X) := \sum_\alpha \frac{O_{\alpha,\iota}}{(X-\alpha)^2}.$$

For an assignment $\mathbf{a}$, the left and right combinations $U(X) = \sum_\iota a_\iota u_\iota(X)$ and $V(X) = \sum_\iota a_\iota v_\iota(X)$ are sums of simple poles. Multiplying simple poles obeys the bilinear law

$$\frac{1}{X-\alpha} \cdot \frac{1}{X-\beta} = \begin{cases} \frac{1}{(X-\alpha)^2} & \text{if } \alpha = \beta, \\ \frac{1}{\alpha-\beta} \left(\frac{1}{X-\alpha} - \frac{1}{X-\beta} \right) & \text{if } \alpha \neq \beta, \end{cases}$$

so the product UV splits into a *diagonal* squared-pole part $\sum_\alpha \ell_\alpha r_\alpha / (X-\alpha)^2$, which is exactly the Hadamard left-hand side, and an *off-diagonal* simple-pole

⁴ This operation is linear as $t(X)$ is already given in the common-reference string.

residue $H_{\text{lin}}(X)$. The output leg is placed directly in the same squared-pole subspace: the CRS publishes the wire bundles built from $w_i(X)$, so the right-hand side $\sum_i a_i w_i(X)$ is available without evaluating a third Hadamard vector at proof time. When $\ell_\alpha r_\alpha = o_\alpha$ for every α , the squared poles on both sides match and cancel; the verifier then checks only the remaining Cauchy residue H_{lin} , which the prover forms from $\ell = L\mathbf{a}$ and $\mathbf{r} = R\mathbf{a}$ by two matrix-vector products in the Cauchy basis (four FFTs in total). The pairing-based proof template is that of [JNR26, App. D.1], adapted from an inner-product setting to R1CS. Section 3 gives the full construction and CRS layout.

Prototype benchmarks. We implemented **Symplex** and Groth16 on shared sparse-R1CS, FFT, fixed-base scalar multiplication, MSM, and pairing code over BLS12-381, using wire-indexed prover queries for both schemes (Section 8). On a Circom circuit chaining SHA-256 ten times (587 766 non-linear constraints, $m = 619\,552$ wires, $\ell = 512$ public inputs, padded to $n = 2^{20}$), single-threaded proving on an Apple M1 Pro takes 7.56 s for **Symplex** and 13.90 s for Groth16, a $1.84\times$ speedup. Verification time is unchanged within run-to-run noise (2.61 ms versus 2.67 ms). The wire-indexed **Symplex** CRS has 21.3% fewer $\mathbb{G}_1$ elements than the implemented Groth16 CRS and only one additional $\mathbb{G}_2$ element. Trusted setup is also slightly faster (26.43 s versus 28.40 s). Table 3 gives the complete matched results.

Organization. Section 2 situates **Symplex** among Groth16, partial-fraction, squaring-only, and dynamic-proof systems, with explicit comparison conventions. Section 3 gives a technical overview, and Section 4 fixes the notation and cryptographic preliminaries. Sections 5 and 6 present the construction and its FFT, MSM, CRS, and setup costs. Section 7 proves completeness, zero-knowledge, and knowledge soundness. Sections 8 and 9 report, respectively, the BLS12-381 implementation and the Lean 4 verification.

2 Related Work and Comparison

Groth16 and partial fractions. Groth16 [Gro16] is a circuit-specific preprocessing zkSNARK for the QAP encoding of R1CS. Our baseline uses the same adaptive basis indexing as **Symplex** and a coset-Lagrange quotient column. Table 2 gives its costs, and Remark 2 distinguishes its six FFTs from the usual monomial implementation’s seven. Jutla, Nema, and Roy [JNR26] give fast partial-fraction algorithms and a pairing-based Hadamard proof. We adapt their Hadamard template to R1CS with wire queries and blinding, retaining Groth16’s proof and verifier shape while obtaining four FFTs and GGM knowledge soundness.

Squaring-only arithmetizations. An SAP [DFGK14, Gro16, GM17] enforces:

$$\left(\sum_i a_i u_i(X)\right)^2 \equiv \sum_i a_i w_i(X) \pmod{t(X)}$$

. SR1CS is the matrix relation $(Az) \odot (Az) = Bz$, used by PARI [DMS26]. Conversion between these and the R1CS/QAP arithmetization of our baseline may add variables and constraints.

Polymath [Lip24] is a zkSNARK for SAP: it has a different trapdoor layout, public-input handling, and proof equation, and uses a Fiat–Shamir KZG-style opening [KZG10]. Its $3\mathbb{G}_1 + 1\mathbb{F}_p$ construction is perfect zero-knowledge and soundness is analyzed in AGMOS [LPS23]. PARI uses an SR1CS polynomial IOP and an equi-efficient polynomial commitment linking the same coefficient vector across bases. Its Fiat–Shamir proof is $2\mathbb{G}_1 + 2\mathbb{F}_p$. The interactive protocol is analyzed in the AGM, while its stated PARI construction is not zero-knowledge. Groth16 and Symplex do not use random oracle or Fiat–Shamir transform.

	Symplex (this work)	Groth16 [Gro16]	Polymath [Lip24]	PARI [DMS26]
Native relation	R1CS	R1CS/QAP	SAP	SR1CS
Proof size	$2\mathbb{G}_1 + 1\mathbb{G}_2$	$2\mathbb{G}_1 + 1\mathbb{G}_2$	$3\mathbb{G}_1 + 1\mathbb{F}_p$	$2\mathbb{G}_1 + 2\mathbb{F}_p$
BLS12-381 (bits)	1536	1536	1408	1280
BLS24-509 (bits)	3072	3072	1792	1536
Prover field stage	4 FFTs (size n)	6 FFTs (size n)	$\Theta(\tilde{n} \log \tilde{n})$ ops [†]	$\Theta(\tilde{n} \log \tilde{n})$ ops [†]
Prover $\mathbb{G}_1$ -MSMs	2, widths $\kappa+2, n+(m-\ell)+4$	2, widths $\kappa+2, 2\kappa+m+(n-\ell)+3$	1, width $\tilde{m}+13\tilde{n}$	4, widths $\tilde{n}, \tilde{n}, \tilde{n}, k-\ell$
Prover $\mathbb{G}_2$ -MSMs	1, width $\kappa+2$	1, width $\kappa+2$	0	0
Online pairings	$3^\ddagger$	$3^\ddagger$	2	3
Verifier $\mathbb{G}_1$ -MSMs	1, width $\ell+1$	1, width $\ell+1$	1, width 2	1, width $3^\S$
Verifier $\mathbb{G}_2$ scalar muls	0	0	1	$1^\S$
Verifier extra field work	—	—	$\frac{3}{2}\ell \log \ell$	$\Theta(\ell)$
Fiat–Shamir/ROM	no	no	yes	yes
ZK in stated construction	perfect	perfect	perfect	none
Security analysis	GGM	GGM	AGMOS	AGM (interactive)

Table 1: Detailed comparison under circuit-specific trusted setup. The cost columns use each scheme’s native relation and therefore are not a same-instance benchmark.

[†]Polymath and PARI report asymptotic field-operation counts rather than an exact FFT schedule. Standard quotient evaluation uses 4–6 FFTs of size $2\tilde{n}$ for Polymath and 4–5 of that size for PARI. [‡]One pairing is setup-precomputable, leaving three online. [§]PARI also uses fixed-base source-group linear combinations beyond the displayed variable-base MSM.

Here (n, m, ℓ) denotes R1CS constraints, wires, and public inputs, and $\kappa = \min\{n, m + 1\}$ depicts the adaptive choice between constraint- and wire-indexed prover queries for both Symplex and Groth16. $(\tilde{n}, \tilde{m})$ denotes the SAP/SR1CS size after conversion, which may be up to $(2n, 2m)$, and k is PARI’s SR1CS variable count. Table 1 reports native costs, while the exact same-R1CS FFT and MSM comparison in Table 2 applies only to Symplex and Groth16. PARI’s BLS24-509 entry is obtained from $2|\mathbb{G}_1| + 2|\mathbb{F}_p|$ rather than reported by its authors.

Dynamic zkSNARKs. Dynamic zkSNARKs [WPSP26b, ZOZ26, WPSP26a] address efficient updates between nearby valid relation pairs, while Symplex only targets fresh proofs. Dynark [ZOZ26] alludes to a four-FFT fresh Groth16 prover,

but the construction is not described in the conference paper or linked full ePrint version.

3 Technical Overview

This section outlines **Symplex** informally. We recall the partial-fraction primitives of [JNR26], relate R1CS to a Hadamard-product identity, adapt the pairing-based Hadamard proof of [JNR26, App. D.1] to a SNARK, and summarize the construction. Partial fractions avoid the switching between coefficient and evaluation representations that costs Groth16 three forward and three inverse FFTs: in the basis $\{1/(X - \alpha)\}_\alpha$ the pairing-friendly product law is already point-wise on the diagonal.

3.1 Partial fraction techniques

Fix a field $\mathbb{F} = \mathbb{F}_p$ and a formal variable X . The starting point of [JNR26] is the elementary identity, for distinct $a_1, \dots, a_n \in \mathbb{F}$,

$$\prod_{i=1}^n \frac{1}{X - a_i} = \sum_{i=1}^n \frac{c_i}{X - a_i}, \quad c_i = \prod_{j \neq i} \frac{1}{a_i - a_j}, \quad (1)$$

and, more generally, the corresponding decomposition for any product of distinct simple linear fractions. Equivalently, the subalgebra $R = \mathbb{F}[(X - a_1)^{-1}, \dots, (X - a_n)^{-1}]$ of the rational-function field $\mathbb{F}(X)$ is governed by the bilinear relations

$$\frac{1}{X - a_i} \cdot \frac{1}{X - a_j} = \frac{1}{a_i - a_j} \left(\frac{1}{X - a_i} - \frac{1}{X - a_j} \right) \quad (i \neq j), \quad (2)$$

which collapse every *off-diagonal* product of simple poles back into a linear combination of simple poles, while the *diagonal* product $1/(X - a_i) \cdot 1/(X - a_i) = 1/(X - a_i)^2$ produces a squared pole. The Hadamard-product perspective of Section 3.3 is essentially a restatement of this diagonal-vs-off-diagonal split.

Lifted to a bilinear group system $(\mathbb{F}_p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ with secret trapdoor $x \xleftarrow{\$} \mathbb{F}_p$, the partial-fraction basis becomes the CRS skeleton:

$$\left\{ \left[\frac{1}{x - a_i} \right]_1, \left[\frac{1}{x - a_i} \right]_2, \left[\frac{1}{(x - a_i)^2} \right]_1 \right\}_{i=1}^n. \quad (3)$$

The hardness of recovering the trapdoor x from (3) is the intuitive basis of **Symplex**'s knowledge-soundness in the GGM.

3.2 Hadamard products, R1CS, and QAPs

Recall from Section 4.4 that an R1CS instance with constraint matrices $L, R, O \in \mathbb{F}_p^{n \times (m+1)}$ accepts a wire vector $\mathbf{a}$ iff the *Hadamard* (entry-wise) identity

$$(L\mathbf{a}) \odot (R\mathbf{a}) = O\mathbf{a} \quad (4)$$

holds in $\mathbb{F}_p^n$. Writing $\ell_\alpha = L_{\alpha,*}\mathbf{a}$, $r_\alpha = R_{\alpha,*}\mathbf{a}$, $o_\alpha = O_{\alpha,*}\mathbf{a}$ for the per-constraint left, right, and output values, this is exactly $\ell_\alpha r_\alpha = o_\alpha$ for every constraint $\alpha \in [n]$. R1CS satisfiability is therefore a Hadamard-product equation, and any SNARK for R1CS ultimately proves one such identity.

The QAP [GGPR13] encoding underlying Groth16 [Gro16] converts this point-wise equation into a polynomial identity by Lagrange interpolation against an auxiliary evaluation domain $\{\rho_1, \dots, \rho_n\}$: pick polynomials $u_i, v_i, w_i \in \mathbb{F}_p[X]_{<n}$ with $u_i(\rho_j) = L_{j,i}$, $v_i(\rho_j) = R_{j,i}$, $w_i(\rho_j) = O_{j,i}$, set $t(X) = \prod_j (X - \rho_j)$, and observe that (4) is equivalent to

$$\left(\sum_i a_i u_i(X)\right) \left(\sum_i a_i v_i(X)\right) = \sum_i a_i w_i(X) + h(X)t(X) \quad (5)$$

for some quotient $h(X)$ of degree $\leq n-2$. The QAP move trades the n point-wise constraints in (4) for a single polynomial identity (5), and divisibility by $t(X)$ then establishes the Hadamard equation. The Groth16 prover computes this identity by FFT-ing $\mathbf{u}\mathbf{a}$, $\mathbf{v}\mathbf{a}$, $\mathbf{w}\mathbf{a}$ between the coefficient and evaluation bases so that point-wise multiplication and division by $t(X)$ can be carried out in the evaluation domain [Gro16, §3.2]. These three forward and three inverse transforms give the six FFTs of our coset-Lagrange Groth16 comparator (Remark 2); the usual monomial quotient column requires one additional inverse transform.

3.3 The partial-fraction Hadamard product proof of [JNR26]

Appendix D.1 of [JNR26] gives a pairing-based proof of a Hadamard-product equation with vectors committed in the partial-fraction basis rather than the monomial basis. For committed vectors $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{F}_p^n$ with

$$\text{com}_{\mathbf{u}} = \left[\sum_{i=1}^n \frac{u_i}{x+i}\right]_1, \quad \text{com}_{\mathbf{v}} = \left[\sum_{i=1}^n \frac{v_i}{x+i}\right]_2, \quad \text{com}_{\mathbf{w}} = \left[\sum_{i=1}^n \frac{w_i}{x+i}\right]_1,$$

to prove that $\mathbf{u} \odot \mathbf{v} = \mathbf{w}$ the prover commits to the rational function

$$H(X) := \left(\sum_i \frac{u_i}{X+i}\right) \left(\sum_i \frac{v_i}{X+i}\right) - \left(\sum_i \frac{w_i}{X+i}\right) \left(\sum_i \frac{1}{X+i}\right). \quad (6)$$

The bilinear law (2) expands the first product into diagonal terms $u_i v_i / (X+i)^2$ plus off-diagonal terms that all decompose into simple poles. The second product similarly produces diagonal terms $w_i / (X+i)^2$ plus off-diagonal simple poles. *Whenever $u_i v_i = w_i$ for every i , the squared-pole diagonals cancel exactly*, and $H(X)$ collapses to a sum of simple poles, $H(X) = \sum_i h_i / (X+i)$. Conversely, if any $u_i v_i \neq w_i$, the residual squared pole $\sum_i (u_i v_i - w_i) / (X+i)^2$ remains and H cannot be expressed as a simple-pole combination.

The prover commits to H via $\text{com}_h = [\sum_i h_i / (x+i)]_1$, and the verifier checks

$$e(\text{com}_{\mathbf{u}}, \text{com}_{\mathbf{v}}) \stackrel{?}{=} e(\text{com}_{\mathbf{w}}, \text{com}_{\mathbf{1}}) \cdot e(\text{com}_h, [1]_2), \quad (7)$$

where $\text{com}_{\mathbf{1}} = [\sum_i 1 / (x+i)]_2$ is a fixed public parameter. The check is a partial-fraction instantiation of $UV = W \cdot \mathbf{1} + H$, with the squared-pole diagonals on the left and right canceled iff the Hadamard equation $\mathbf{u} \odot \mathbf{v} = \mathbf{w}$ holds.

Two points carry over to the SNARK. First, $1/(X+i)^2$ signals an unsatisfied Hadamard constraint at i : the squared-pole diagonal of UV vanishes only if $u_i v_i = w_i$ for every i . Second, no FFT is needed: the prover forms $H(X)$ from $\mathbf{u}, \mathbf{v}, \mathbf{w}$ by a Cauchy-type matrix-vector product (Lemma 1), without introducing a quotient $h(X)$ outside this basis.

3.4 From the Hadamard product proof to a zkSNARK

To go from the Hadamard product proof of Section 3.3 to a SNARK for R1CS, three additional ingredients are needed.

(i) *R1CS structure in the CRS.* The Hadamard proof above commits to vectors $\mathbf{u}, \mathbf{v}, \mathbf{w}$ chosen by the prover. For R1CS we instead check (4) for $\ell = L\mathbf{a}$, $\mathbf{r} = R\mathbf{a}$, $\mathbf{o} = O\mathbf{a}$ with public matrices L, R, O . We replace the per-constraint basis $\{1/(X+i)\}_i$ by a per-wire basis. Index each multiplication constraint by a distinct *pole* $\alpha \in \mathbb{F}_p^*$, and define the partial-fraction selectors

$$u_\iota(X) := \sum_{\alpha} \frac{L_{\alpha, \iota}}{X - \alpha}, \quad v_\iota(X) := \sum_{\alpha} \frac{R_{\alpha, \iota}}{X - \alpha}, \quad w_\iota(X) := \sum_{\alpha} \frac{O_{\alpha, \iota}}{(X - \alpha)^2}. \quad (8)$$

Note the *squared pole* choice for w_ι : $w_\iota(X)$ lives in the same diagonal subspace $\text{span}\{1/(X - \alpha)^2\}_\alpha$ in which the unsatisfied Hadamard residual of Section 3.3 would emerge. R1CS satisfaction $\ell_\alpha r_\alpha = o_\alpha$ is equivalent to

$$\underbrace{\sum_{\alpha} \frac{\ell_\alpha r_\alpha}{(X - \alpha)^2}}_{\text{diagonal of } UV} = \sum_{\iota} a_\iota w_\iota(X), \quad (9)$$

i.e. the squared-pole diagonal of UV , with $U = \sum_{\iota} a_\iota u_\iota$ and $V = \sum_{\iota} a_\iota v_\iota$, coincides with the wire combination of the squared-pole CRS elements w_ι . Subtracting the diagonal from UV leaves the *Cauchy residue*:

$$H_{\text{lin}}(X) := U(X)V(X) - \sum_{\alpha} \frac{\ell_\alpha r_\alpha}{(X - \alpha)^2},$$

a sum of simple poles, exactly as in (6).

(ii) *Wire-bundled CRS.* Following Groth16, we bundle the three wire-indexed selectors into a single CRS element per wire,

$$W_\iota(X) := w_\iota(X) + y u_\iota(X) + z v_\iota(X), \quad (10)$$

where $(y, z) \xleftarrow{\$} \mathbb{F}_p$ are setup trapdoors playing the role of Groth16's (β, α) . The trapdoors y, z couple the u, v, w legs per wire as in Groth16's QAP argument.

(iii) *Zero-knowledge and public/private split.* The blinding randomizers $r\delta, s\delta$ that Groth16 adds to A and B for zero-knowledge are replaced here by two *blind poles* $\xi_1, \xi_2 \in \mathbb{F}_p$, disjoint from all gate poles, with which the prover blinds U and V by $r/(X - \xi_1)$ and $s/(X - \xi_2)$ (one-sided blinding per side. The cross term $rs/((X - \xi_1)(X - \xi_2))$ is decomposed into simple poles by (2) and absorbed into C). The public/private wire split via the γ, δ trapdoors is unchanged from Groth16.

The result is the construction of Figure 1 of Section 5, with the same proof shape and same verifier equation template as Groth16, but with the Cauchy residue H_{lin} playing the role of Groth16's $h(X)$.

3.5 Putting it all together

Stitching everything together we roughly arrive at the following blue-print:

CRS. For each constraint pole α publish $[1/(x - \alpha)]_1, [1/(x - \alpha)]_2$; for each wire ι publish the bundled $[W_\iota(x)/\gamma]_1$ (public wires) or $[W_\iota(x)/\delta]_1$ (private wires); and for the simple-pole basis of H_{lin} publish $[1/(\delta(x - \alpha))]_1$. Additionally, the blind-pole compensators $[W_{\xi_1}(x)/\delta]_1, [W_{\xi_2}(x)/\delta]_1$, and $[y]_2, [z]_1, [\gamma]_2, [\delta]_2$.

Prove. $A(X) := z + U(X)$, $B(X) := y + V(X)$, and

$$C(X) := \frac{1}{\delta} \left(H_{\text{lin}}(X) + \sum_{\iota > \ell} a_\iota W_\iota(X) + r W_{\xi_1}(X) + s W_{\xi_2}(X) \right).$$

Send $\pi = ([A(x)]_1, [C(x)]_1, [B(x)]_2)$.

Verify.

$$e([A]_1, [B]_2) \stackrel{?}{=} e([z]_1, [y]_2) \cdot e\left(\sum_{\iota \leq \ell} a_\iota [W_\iota(x)/\gamma]_1, [\gamma]_2\right) \cdot e([C]_1, [\delta]_2).$$

Expanding $(z + U)(y + V) - zy = UV + yU + zV$ and substituting $UV = H_{\text{lin}} + \sum_\iota a_\iota w_\iota$ (the Hadamard cancellation (9) plus the simple-pole residue) yields the prover's $\delta C + \sum_{\iota \leq \ell} a_\iota W_\iota$ on the left (Section 6). Completeness (Section 7) follows from Section 3.3, with R1CS encoded in the CRS and zero-knowledge from blind poles.

3.6 Why point-wise products are native to partial fractions

In the monomial basis $\{1, X, X^2, \dots\}$, polynomial multiplication is the convolution of coefficient vectors, whereas *point-wise* (Hadamard) multiplication corresponds to multiplication of *evaluation* vectors at a chosen domain. The two operations are conjugate via the DFT, and any protocol that wants to express a Hadamard relation as a polynomial identity in the monomial basis must convert between the two via FFTs: that is the source of Groth16's 6-FFT cost (Remark 2).

In the partial-fraction basis $\{1/(X - \alpha_i)\}_i$ the product law is (2), with two effects:

- **Diagonal entries** ($\alpha_i = \alpha_j$) produce a *new* basis element $1/(X - \alpha_i)^2$. This is the partial-fraction analogue of Hadamard multiplication: the diagonal of the outer product $(\mathbf{u} \mathbf{u}^\top)$ emerges at the squared pole. The R1CS satisfaction equation (4) is exactly the cancellation of this squared-pole diagonal against the w -side, as in (9).
- **Off-diagonal entries** ($\alpha_i \neq \alpha_j$) decompose back into the *same* simple-pole basis with Cauchy coefficients $1/(\alpha_i - \alpha_j)$. This is the “convolution”-style residue H_{lin} , captured by the Cauchy matrix $\mathbf{C}$ of Lemma 1.

The partial-fraction basis acts as both an evaluation basis (each ℓ_α is $L_{\alpha,*}\mathbf{a}$ at the corresponding constraint) and a coefficient basis with a bilinear product law. The group element $[1/(x - \alpha)]_1$ serves both as a coefficient as well as a Lagrange basis. Groth16’s conversion from coefficients to evaluations, then to point-wise products, and back to coefficients is unnecessary in **Symplex**: the diagonal of UV appears as squared poles via (2), and H_{lin} is one Cauchy matrix–vector product per side.

3.7 Quantitative consequence: four FFTs instead of six

Fix the gate poles α_i to be n -th roots of unity ω^{i-1} in $\mathbb{F}_p$. Then by [CJL⁺26, Thm. 3] (recalled as Lemma 2 of Section 6.2), the Cauchy matrix $\mathbf{C}$ diagonalises up to a sparse correction:

$$\mathbf{C} \mathbf{v} = \text{DFT}^{-1}(\hat{\mathbf{J}}(\text{DFT} \mathbf{v})) - D_\omega \mathbf{v}, \quad \hat{\mathbf{J}} \text{ sparse, } D_\omega \text{ diagonal,}$$

Here, using indices $i, j \in \{0, \dots, n-1\}$, J is the Cauchy-like matrix with $J_{ij} = 1/(\omega^i - \omega^j)$ for $i \neq j$ and $J_{ii} = (n-1)/(2\omega^i)$. The diagonal correction is $D_\omega := \text{diag}((n-1)/(2\omega^i))_{i=0}^{n-1}$, so $\mathbf{C} = J - D_\omega$ has the required zero diagonal. Its conjugate representation $\hat{\mathbf{J}} = \text{DFT} J \text{DFT}^{-1}$ is sparse, with only $\hat{J}_{i,i-1} = n-i$ nonzero. Thus, multiplying by J requires one forward FFT, a sparse weighted shift, and one inverse FFT. Including the diagonal correction, each Cauchy action costs $O(n)$ scalar work. **Symplex**’s prover needs $\mathbf{C}$ applied twice (once to ℓ , once to $\mathbf{r}$) to assemble the Cauchy residue $\mathbf{h} = \ell \odot (\mathbf{C}\mathbf{r}) + \mathbf{r} \odot (\mathbf{C}\ell)$ (Corollary 1). The two Hadamard products and the D_ω corrections are $O(n)$, thus totaling 4 **FFTs**.

Groth16’s prover, by contrast, applies forward and inverse DFTs to all three of $\mathbf{u}\mathbf{a}$, $\mathbf{v}\mathbf{a}$, $\mathbf{w}\mathbf{a}$ to evaluate the QAP quotient $h(X)t(X) = uv - w$ on the auxiliary domain and divide by $t(X)$ [CJL⁺26, Alg. 1, 2], costing 3 forward plus 3 inverse FFTs = 6 FFTs when the quotient is committed directly in the coset-Lagrange basis used by our comparator. The usual monomial quotient column requires a final inverse coset FFT and has one fewer $\mathbb{G}_1$ CRS element. The two-FFT difference is contributed by the cost of the third Hadamard-leg evaluation $\mathbf{w}\mathbf{a}$, which **Symplex** avoids: by (8) the “ w ” role is precomputed into the squared-pole part $w_\iota(X)$ of the per-wire CRS bundle W_ι , and is not used by the prover at proof time. The coefficient-to-evaluation switching that Groth16 spends two FFTs on is the operation that the partial-fraction basis performs at setup.

Section 6.3 describes a further consequence: the **Symplex** setup itself is FFT-free, since each u_i, v_i, w_i is already a sum of simple/squared poles indexed by the constraint matrix entries, so the evaluation-to-coefficient inverse-FFT step that Groth16 implementations (e.g. Arkworks/SnarkJS) carry out at **Setup** time is not required. The same point-wise structure therefore yields both the four-FFT prover and the FFT-free setup of **Symplex**.

4 Preliminaries

This section outlines notation and recalls the cryptographic background used in the rest of the paper: bilinear groups, the syntax of non-interactive arguments, the generic group model, rank-1 constraint systems.

4.1 Notation and bilinear groups

We work over a finite field $\mathbb{F}_p$ of prime order p with $\log p = \text{poly}(\lambda)$. A *bilinear group system* is a tuple $(\mathbb{F}_p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$ of three cyclic groups of order p together with a non-degenerate bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, where g generates $\mathbb{G}_1$ and h generates $\mathbb{G}_2$. We use the bracket [EHK⁺13] notation $[x]_1 := g^x \in \mathbb{G}_1$, $[y]_2 := h^y \in \mathbb{G}_2$, and $[z]_T := e(g, h)^z \in \mathbb{G}_T$, so that pairing equations take the multiplicative form $[x]_1 \cdot [y]_2 = [xy]_T$.

4.2 Non-interactive zero-knowledge arguments of knowledge

Throughout, **Symplex** is presented as a quadruple (**Setup**, **Prove**, **Verify**, **Sim**) for a relation R encoded by an R1CS instance: the CRS σ is output at **setup**, honest proofs are arguments π accepted by **Verify**, and **Sim** establishes perfect zero-knowledge. We prove perfect completeness and perfect zero-knowledge in Section 7, and computational knowledge soundness in the generic group model (Section 7.3). The formal syntax of non-interactive arguments and the definitions of completeness, zero-knowledge, knowledge soundness, and preprocessing SNARK succinctness are recalled in Appendix A.

4.3 The generic group model

We use the generic group model (GGM) of Shoup [Sho97], extended with a bilinear-pairing oracle. For each $s \in \{1, 2, T\}$, the experiment samples an independent random injective encoding $\text{enc}_s : \mathbb{F}_p \rightarrow \{0, 1\}^\kappa$, where $2^\kappa \geq p$. The adversary receives only the encodings of the public group elements and accesses the groups through oracles that, on valid handles, compute

$$\begin{aligned} \text{enc}_s(a), \text{enc}_s(b) &\mapsto \text{enc}_s(ca + c'b) \quad (c, c' \in \mathbb{F}_p), \\ \text{enc}_1(a), \text{enc}_2(b) &\mapsto \text{enc}_T(ab). \end{aligned}$$

The first oracle implements group operations and scalar multiplication in $\mathbb{G}_s$, while the second implements the pairing. Equality is equality of handles, and

malformed handles are rejected. In the symmetric setting $\mathbb{G}_1 = \mathbb{G}_2$, a single source-group encoding is used and the pairing oracle accepts two source-group handles.

Thus a generic adversary can manipulate group elements and test relations among them, but cannot inspect their representations or otherwise access their discrete logarithms. Moreover, every source-group handle in an oracle transcript has a coefficient vector expressing its exponent as an $\mathbb{F}_p$ -linear combination of the source-group inputs. The GGM simulator can recover these vectors by tracking the oracle calls. These are the symbolic representations used by the knowledge extractor in Section 7.3. Distinct formal expressions can be assigned the same handle only when their evaluations collide, whose probability is bounded by the standard Schwartz–Zippel and union-bound argument.

4.4 Rank-1 constraint systems

A relation R over $\mathbb{F}_p$ is presented by a *rank-1 constraint system* (R1CS) with n multiplication constraints and $m + 1$ wire indices. Index 0 is reserved for the constant wire $a_0 = 1$, indices $I_{\text{pub}} = \{1, \dots, \ell\}$ are the public statement wires and $I_{\text{priv}} = \{\ell + 1, \dots, m\}$ are the witness wires. The relation is specified by three matrices $L, R, O \in \mathbb{F}_p^{n \times (m+1)}$, indexed by constraint $\alpha \in \{1, \dots, n\}$ and wire $\iota \in \{0, \dots, m\}$. An assignment $\mathbf{a} = (a_0, \dots, a_m)$ with $a_0 = 1$ satisfies R iff

$$(L\mathbf{a}) \odot (R\mathbf{a}) = O\mathbf{a}, \quad \text{i.e.,} \quad \left(\sum_{\iota} L_{\alpha,\iota} a_{\iota}\right) \cdot \left(\sum_{\iota} R_{\alpha,\iota} a_{\iota}\right) = \sum_{\iota} O_{\alpha,\iota} a_{\iota} \quad \forall \alpha.$$

Following standard usage, we write $\ell_{\alpha} := L_{\alpha,*}\mathbf{a}$, $r_{\alpha} := R_{\alpha,*}\mathbf{a}$, $o_{\alpha} := O_{\alpha,*}\mathbf{a}$ for the per-constraint left, right, and output values. The satisfaction condition is then $\ell_{\alpha} r_{\alpha} = o_{\alpha}$ at every constraint. For a matrix M , write $\text{nnz}(M)$ for its number of nonzero entries. The *instance sparsity* of the R1CS is

$$\text{nnz}(L, R, O) := \text{nnz}(L) + \text{nnz}(R) + \text{nnz}(O),$$

the total number of nonzero entries across the three constraint matrices.

4.5 Quadratic arithmetic programs and Groth16

The R1CS is encoded into a *quadratic arithmetic program* (QAP) $(\ell, \{u_i(X), v_i(X), w_i(X)\}_{i=0}^m, t(X))$ over $\mathbb{F}_p$ as follows. Pick distinct evaluation points $\rho_1, \dots, \rho_n \in \mathbb{F}_p$ and let $t(X) = \prod_{j=1}^n (X - \rho_j)$. Define u_i, v_i, w_i as the unique polynomials of degree less than n whose values at ρ_j are the j -th rows of L, R, O at column i . Then $\mathbf{a}$ satisfies R if and only if

$$\sum_{i=0}^m a_i u_i(X) \cdot \sum_{i=0}^m a_i v_i(X) = \sum_{i=0}^m a_i w_i(X) + h(X) t(X)$$

for some quotient polynomial $h(X)$ of degree at most $n - 2$. Groth16 [Gro16] for QAPs is recalled in Appendix B (Figure 2).

5 The Symplex Construction

We describe **Symplex** for the R1CS relation R of Section 4.4. Fix a bilinear group system $(\mathbb{F}_p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$, and associate each constraint with a distinct public *constraint pole* $\alpha \in \mathbb{F}_p$. Two additional public *blind poles* ξ_1, ξ_2 are distinct from one another and from all constraint poles.

For every wire $\iota \in \{0, \dots, m\}$, define

$$u_\iota(X) := \sum_\alpha \frac{L_{\alpha, \iota}}{X - \alpha}, \quad v_\iota(X) := \sum_\alpha \frac{R_{\alpha, \iota}}{X - \alpha}, \quad w_\iota(X) := \sum_\alpha \frac{O_{\alpha, \iota}}{(X - \alpha)^2},$$

and, for independently sampled $y, z \in \mathbb{F}_p^*$, set

$$\begin{aligned} W_\iota(X) &:= w_\iota(X) + y u_\iota(X) + z v_\iota(X), \\ W_{\xi_1}(X) &:= \frac{y}{X - \xi_1}, \quad W_{\xi_2}(X) := \frac{z}{X - \xi_2}. \end{aligned} \tag{11}$$

Thus W_ι encodes the role of wire ι on the output, left, and right legs of every constraint. It is the partial-fraction analogue of Groth16's bundled term $\beta u_\iota(x) + \alpha v_\iota(x) + w_\iota(x)$.

The Symplex zkSNARK (R1CS over $\mathbb{F}_p$; proof $\pi = ([A]_1, [C]_1, [B]_2)$).

Setup(R) $\rightarrow (\sigma, \tau)$:

Sample $x, y, z, \gamma, \delta \xleftarrow{\$} \mathbb{F}_p^*$ with x distinct from every constraint and blind pole. Let $\tau = (x, y, z, \gamma, \delta)$ and, using (11), publish $\sigma = ([\sigma_1]_1, [\sigma_2]_2)$, where

$$\sigma_1 = \left(\begin{array}{c} z, \left\{ \frac{1}{x-\alpha} \right\}_\alpha, \frac{1}{x-\xi_1}, \\ \left\{ \frac{W_\ell(x)}{\gamma} \right\}_{\ell=0}^\ell, \left\{ \frac{W_\ell(x)}{\delta} \right\}_{\ell=\ell+1}^m, \frac{W_{\xi_1}(x)}{\delta}, \frac{W_{\xi_2}(x)}{\delta}, \\ \left\{ \frac{1}{\delta(x-\alpha)} \right\}_\alpha, \frac{1}{\delta(x-\xi_1)}, \frac{1}{\delta(x-\xi_2)} \end{array} \right),$$

$$\sigma_2 = \left(y, \gamma, \delta, \left\{ \frac{1}{x-\alpha} \right\}_\alpha, \frac{1}{x-\xi_2} \right).$$

Prove($R, \sigma, a_1, \dots, a_m$) $\rightarrow \pi$:

Set $a_0 = 1$, $\ell_\alpha = L_{\alpha,*} \mathbf{a}$, and $r_\alpha = R_{\alpha,*} \mathbf{a}$. Sample $r, s \xleftarrow{\$} \mathbb{F}_p$ and define

$$\begin{aligned} U(X) &:= \sum_\alpha \frac{\ell_\alpha}{X-\alpha} + \frac{r}{X-\xi_1}, & A(X) &:= z + U(X), \\ V(X) &:= \sum_\alpha \frac{r_\alpha}{X-\alpha} + \frac{s}{X-\xi_2}, & B(X) &:= y + V(X), \\ H_{\text{lin}}(X) &:= U(X)V(X) - \sum_\alpha \frac{\ell_\alpha r_\alpha}{(X-\alpha)^2}. \end{aligned}$$

$$C(X) := \frac{1}{\delta} \left(H_{\text{lin}}(X) + \sum_{\ell=\ell+1}^m a_\ell W_\ell(X) + r W_{\xi_1}(X) + s W_{\xi_2}(X) \right).$$

Output $\pi = ([A(x)]_1, [C(x)]_1, [B(x)]_2)$. The partial-fraction coefficients needed to assemble $[H_{\text{lin}}(x)/\delta]_1$ are given in Corollary 1 and Remark 1.

Verify($R, \sigma, a_1, \dots, a_\ell, \pi$):

Set $a_0 = 1$ and check

$$e([A]_1, [B]_2) \stackrel{?}{=} e([z]_1, [y]_2) \cdot e\left(\sum_{\ell=0}^\ell a_\ell \left[\frac{W_\ell(x)}{\gamma}\right]_1, [\gamma]_2\right) \cdot e([C]_1, [\delta]_2).$$

Sim($R, \tau, a_1, \dots, a_\ell$) $\rightarrow \pi$:

Set $a_0 = 1$. Sample $\tilde{A}, \tilde{B} \xleftarrow{\$} \mathbb{F}_p$, set

$$\tilde{C} := \frac{1}{\delta} \left(\tilde{A}\tilde{B} - zy - \sum_{\ell=0}^\ell a_\ell W_\ell(x) \right),$$

and output $\pi = ([\tilde{A}]_1, [\tilde{C}]_1, [\tilde{B}]_2)$.

Fig. 1: The asymmetric Symplex zkSNARK. In exponent form, verification checks $AB = zy + \sum_{\ell \leq \ell} a_\ell W_\ell(x) + \delta C$; the γ in the public-input pairing cancels the $1/\gamma$ in each public-wire bundle.

6 Complexity

This section analyzes the cost of Symplex relative to Groth16. We derive the Cauchy structure of the linear residue H_{lin} , bound the prover's FFT and MSM counts, and compare CRS size and setup cost.

6.1 Cauchy structure of the linear residue

We now show that H_{lin} has a closed-form coefficient vector in the partial-fraction basis $\{1/(X - \alpha_i)\}$, governed by a Cauchy matrix. Index the constraints by $i = 1, \dots, n$ with poles $\alpha_1, \dots, \alpha_n \in \mathbb{F}_p$, and write $\boldsymbol{\ell} = L\mathbf{a}, \mathbf{r} = R\mathbf{a} \in \mathbb{F}_p^n$ for the per-constraint left and right values, $\ell_i = L_{\alpha_i, *}\mathbf{a}$ and $r_i = R_{\alpha_i, *}\mathbf{a}$. Set

$$\mathbf{p}(X) := (p_1(X), \dots, p_n(X))^\top, \quad p_i(X) := \frac{1}{X - \alpha_i}.$$

Ignoring the blind poles for a moment, $U_0 = \boldsymbol{\ell}^\top \mathbf{p}$ and $V_0 = \mathbf{r}^\top \mathbf{p}$, and

$$U_0 V_0 = \boldsymbol{\ell}^\top (\mathbf{p} \mathbf{p}^\top) \mathbf{r}, \quad H_{\text{lin}}^{(0)} := U_0 V_0 - \sum_i \ell_i r_i \frac{1}{(X - \alpha_i)^2} = \boldsymbol{\ell}^\top \text{OffDiag}(\mathbf{p} \mathbf{p}^\top) \mathbf{r},$$

where $\text{OffDiag}(\mathbf{p} \mathbf{p}^\top)$ is the matrix obtained by zeroing the diagonal (i, i) -entries $1/(X - \alpha_i)^2$ of $\mathbf{p} \mathbf{p}^\top$.

Lemma 1 (Cauchy decomposition). *Let $D_{\mathbf{p}} := \text{diag}(p_1, \dots, p_n)$ and let $\mathbf{C} \in \mathbb{F}_p^{n \times n}$ be the Cauchy matrix*

$$C_{ij} := \begin{cases} 1/(\alpha_i - \alpha_j), & i \neq j, \\ 0, & i = j. \end{cases}$$

(We use $\mathbf{C}$ throughout this section to disambiguate from the prover's $\mathbb{G}_1$ -element $C(X)$.) Then $\mathbf{C}^\top = -\mathbf{C}$ and

$$\text{OffDiag}(\mathbf{p} \mathbf{p}^\top) = D_{\mathbf{p}} \mathbf{C} - \mathbf{C} D_{\mathbf{p}}.$$

Proof. Both sides have zero diagonal: for the right-hand side, the (i, i) -entry is $p_i C_{ii} - C_{ii} p_i = 0$. For $i \neq j$,

$$(D_{\mathbf{p}} \mathbf{C} - \mathbf{C} D_{\mathbf{p}})_{ij} = p_i C_{ij} - C_{ij} p_j = \frac{p_i - p_j}{\alpha_i - \alpha_j}.$$

Computing the numerator,

$$p_i - p_j = \frac{1}{X - \alpha_i} - \frac{1}{X - \alpha_j} = \frac{\alpha_i - \alpha_j}{(X - \alpha_i)(X - \alpha_j)},$$

so $(D_{\mathbf{p}} \mathbf{C} - \mathbf{C} D_{\mathbf{p}})_{ij} = 1/((X - \alpha_i)(X - \alpha_j)) = (\mathbf{p} \mathbf{p}^\top)_{ij}$, matching $\text{OffDiag}(\mathbf{p} \mathbf{p}^\top)_{ij}$. $\square$

Corollary 1 (Coefficient vector of $H_{\text{lin}}^{(0)}$). $H_{\text{lin}}^{(0)} = \mathbf{h}^\top \mathbf{p}$, where

$$\mathbf{h} = \boldsymbol{\ell} \odot (\mathbf{C} \mathbf{r}) + \mathbf{r} \odot (\mathbf{C} \boldsymbol{\ell}).$$

Proof. By Lemma 1, $H_{\text{lin}}^{(0)} = \boldsymbol{\ell}^\top D_{\mathbf{p}} \mathbf{C} \mathbf{r} - \boldsymbol{\ell}^\top \mathbf{C} D_{\mathbf{p}} \mathbf{r}$. Using $\mathbf{x}^\top D_{\mathbf{p}} \mathbf{y} = \sum_i x_i y_i p_i$ and $\mathbf{C}^\top = -\mathbf{C}$,

$$\boldsymbol{\ell}^\top D_{\mathbf{p}} \mathbf{C} \mathbf{r} = \sum_i \ell_i (\mathbf{C} \mathbf{r})_i p_i = (\boldsymbol{\ell} \odot (\mathbf{C} \mathbf{r}))^\top \mathbf{p},$$

$$\boldsymbol{\ell}^\top \mathbf{C} D_{\mathbf{p}} \mathbf{r} = \sum_i (\mathbf{C}^\top \boldsymbol{\ell})_i r_i p_i = -(\mathbf{r} \odot (\mathbf{C} \boldsymbol{\ell}))^\top \mathbf{p},$$

and subtracting gives the formula. $\square$

In particular $\mathbf{h}$ is symmetric in $(\ell, \mathbf{r})$, consistent with $UV = VU$. The blind-pole contributions to H_{lin} (the cross terms involving ξ_1, ξ_2) require $O(n)$ coefficient corrections and introduce two additional pole terms, but don't require additional FFTs. We ignore them in the upcoming FFT analysis, but briefly discuss the details in the next remark.

Remark 1 (Closed form including blind poles). Setting $p_{\xi_i} := 1/(X - \xi_i)$, the partial-fraction coefficients of H_{lin} in the basis $\{p_1, \dots, p_n, p_{\xi_1}, p_{\xi_2}\}$ are obtained by adding the three blind-pole cross products of UV to Corollary 1 via the identity $\frac{1}{(X-\rho)(X-\sigma)} = \frac{1}{\rho-\sigma}(p_\rho - p_\sigma)$:

$$\begin{aligned} h_i &= \ell_i (\mathbf{C}\mathbf{r})_i + r_i (\mathbf{C}\ell)_i + \frac{r r_i}{\alpha_i - \xi_1} + \frac{s \ell_i}{\alpha_i - \xi_2}, \quad i = 1, \dots, n, \\ h_{\xi_1} &= -r \sum_{j=1}^n \frac{r_j}{\alpha_j - \xi_1} + \frac{rs}{\xi_1 - \xi_2}, \\ h_{\xi_2} &= -s \sum_{j=1}^n \frac{\ell_j}{\alpha_j - \xi_2} - \frac{rs}{\xi_1 - \xi_2}. \end{aligned}$$

The blind-pole correction to each h_i and the closed forms for h_{ξ_1}, h_{ξ_2} are $O(n)$ scalar work, and don't contribute to the FFT count of Theorem 1.

6.2 Prover complexity: four FFTs

We choose the gate poles to be n -th roots of unity, $\alpha_i = \omega^{i-1}$ for a primitive n -th root of unity $\omega \in \mathbb{F}_p$ (so $n \mid p-1$). Circuits with fewer than n gates are padded. Under this choice, the Cauchy matrix $\mathbf{C}$ acts on a vector via two FFTs, by the explicit reduction of [CJL⁺26, Thm. 3].

Lemma 2 (Conjugate form of the Cauchy matrix [CJL⁺26, Thm. 3]).

Let ω be a primitive n -th root of unity in $\mathbb{F}_p$, and let $\text{DFT} \in \mathbb{F}_p^{n \times n}$ be the discrete Fourier transform matrix with entries $\text{DFT}_{kj} := \omega^{jk}$ (so $(\text{DFT}^{-1})_{jk} = n^{-1}\omega^{-jk}$). Define

$$J_{ij} := \begin{cases} 1/(\omega^i - \omega^j), & i \neq j, \\ (n-1)/(2\omega^i), & i = j \end{cases} \quad (i, j \in \{0, \dots, n-1\}).$$

Then

$$\hat{J} := \text{DFT} \cdot J \cdot \text{DFT}^{-1}$$

is the sparse matrix with $\hat{J}_{i,i-1} = n-i$ for $1 \leq i \leq n-1$ and $\hat{J}_{ij} = 0$ otherwise. Consequently for any $\mathbf{v} \in \mathbb{F}_p^n$,

$$J\mathbf{v} = \text{DFT}^{-1}(\hat{J}(\text{DFT}\mathbf{v})),$$

computable in 1 forward FFT, 1 inverse FFT, and $O(n)$ scalar operations.

The Cauchy matrix $\mathbf{C}$ used in Corollary 1 differs from J only on the diagonal ($\mathbf{C}$ has zero, J has $(n-1)/(2\omega^i)$). Hence $\mathbf{C} = J - D_\omega$ with $D_\omega := \text{diag}((n-1)/(2\omega^{i-1}))_{i=1}^n$, and

$$\mathbf{C}\mathbf{v} = \text{DFT}^{-1}(\hat{J}(\text{DFT}\mathbf{v})) - D_\omega \mathbf{v}$$

also costs 2 FFTs plus $O(n)$.

Theorem 1 (Prover FFT count). *With gate poles at the n -th roots of unity, the FFT portion of the Symplex prover is the computation of $\mathbf{h} = \boldsymbol{\ell} \odot (\mathbf{C}\mathbf{r}) + \mathbf{r} \odot (\mathbf{C}\boldsymbol{\ell})$ (Corollary 1), which requires 2 forward FFTs and 2 inverse FFTs (4 FFTs total), plus $O(n)$ scalar operations. In the constraint-indexed presentation of Figure 1, the prover performs two $\mathbb{G}_1$ -MSMs of widths $n+2$ and $n+(m-\ell)+4$, respectively, and one $\mathbb{G}_2$ -MSM of width $n+2$, in addition to the $O(\text{nnz}(L) + \text{nnz}(R))$ scalar work needed to compute $\boldsymbol{\ell} = L\mathbf{a}$ and $\mathbf{r} = R\mathbf{a}$. The public change of basis in Section 6.4 reduces the first $\mathbb{G}_1$ width and the $\mathbb{G}_2$ width to $\min\{n, m+1\} + 2$ without changing the FFT count.*

Proof. Apply Lemma 2 once to compute $\mathbf{C}\mathbf{r}$ and once to compute $\mathbf{C}\boldsymbol{\ell}$, costing 2 forward and 2 inverse FFTs in total. The two Hadamard products and the diagonal subtractions D_ω are $O(n)$. Once $\mathbf{h}$ is known, add the blind-pole corrections of Remark 1 to obtain

$$H_{\text{lin}}(X) = \sum_{i=1}^n h_i^{\text{full}} p_i(X) + h_{\xi_1} p_{\xi_1}(X) + h_{\xi_2} p_{\xi_2}(X).$$

where h_i^{full} denotes the corrected gate-pole coefficient h_i from that remark. Thus $[H_{\text{lin}}(x)/\delta]_1$ is assembled from the $n+2$ published δ -cleared pole elements. The remaining private-wire and blind-compensation terms are included in the same C -MSM, while $[A(x)]_1$ and $[B(x)]_2$ are assembled by their respective MSMs. $\square$

Remark 2 (Comparison with Groth16). The dominant FFT cost of the Groth16 prover is computing $h(x)$, whose coset evaluation requires 3 forward FFTs + 3 inverse FFTs. The usual monomial $t(x)/\delta$ column in ARKWORKS/SNARKJS requires one additional inverse coset FFT to return h to coefficient form. Our matched implementation instead publishes a size- n coset-Lagrange $t(x)/\delta$ column, removing that final transform and giving the six FFTs used throughout our comparison; the column has one more $\mathbb{G}_1$ element than the monomial version. The derivative approach of [CJL⁺26, Algorithm 2] likewise achieves a six-FFT comparison point.

Symplex saves 2 FFTs by absorbing the “ w ”-role of Groth16 into the squared-pole part $w_\ell(X)$ of the per-wire CRS elements W_ℓ . The prover still uses the private-wire bundles $W_\ell(x)$ in the C -MSM, but does not separately form the output-leg vector $O\mathbf{a}$ or apply an FFT to it. Like [CJL⁺26, Alg. 2], Symplex needs only an n -th root of unity, not $2n$ -th like the coset approach.

Remark 3 (Generic gate poles). For circuits whose gate poles are *not* chosen at roots of unity, the displacement-structure identity $D_a \mathbf{C} - \mathbf{C} D_b = \mathbf{1}\mathbf{1}^\top - I$ for the Cauchy matrix yields $\mathbf{C}\mathbf{v}$ in $O(n \log^2 n)$ via the displacement-based algorithms of [GO94]. The roots-of-unity specialization above is the fastest case.

6.3 CRS size and setup cost

Two further advantages over Groth16 follow from the same features that drive the prover-side savings.

CRS size. Counting elements in the constraint-indexed CRS of Figure 1,

$$|\sigma_1|_{\text{Sym}} = 2n + m + 7, \quad |\sigma_2|_{\text{Sym}} = n + 4.$$

The adaptive CRS of Section 6.4, used in our wire-indexed implementation, instead has

$$|\sigma_1|_{\text{Sym, adaptive}} = n + m + 7 + \kappa, \quad |\sigma_2|_{\text{Sym, adaptive}} = 4 + \kappa, \quad \kappa = \min\{n, m + 1\}.$$

For Groth16 there are two reference variants. The small-CRS presentation of Figure 2 publishes only the monomial bases $\{[x^i]_*\}_{i=0}^{n-1}$, giving

$$|\sigma_1|_{\text{G16, small}} = m + 2n + 3, \quad |\sigma_2|_{\text{G16, small}} = n + 3,$$

at the cost of two additional prover FFTs to evaluate $\sum a_i u_i(X), \sum a_i v_i(X)$ on the fly. The wire-indexed variant used in our implementation publishes the bases $\{[u_i(x)]_1, [v_i(x)]_1\}_{i=0}^m$ and $\{[v_i(x)]_2\}_{i=0}^m$ to absorb those two FFTs into the CRS, together with the size- n coset-Lagrange $t(x)/\delta$ column described in Section 8, giving

$$|\sigma_1|_{\text{G16, wire}} = 3m + n + 6, \quad |\sigma_2|_{\text{G16, wire}} = m + 4.$$

The usual monomial $t(x)/\delta$ column has one fewer $\mathbb{G}_1$ element but requires the final inverse FFT described above. Against the implemented Groth16 variant, the adaptive **Symplex** CRS saves $2m - \kappa - 1$ elements in σ_1 ; the difference $|\sigma_2|_{\text{G16, wire}} - |\sigma_2|_{\text{Sym, adaptive}}$ is $m - \kappa$. In the benchmark branch $m + 1 < n$, where $\kappa = m + 1$, this means that **Symplex** saves $m - 2 = 619\,550$ $\mathbb{G}_1$ elements (21.3%) and uses one additional $\mathbb{G}_2$ element, exactly as reported in Table 3. Against the small-CRS variant of Figure 2, the constraint-indexed **Symplex** CRS and Groth16 CRS are within a fixed $O(1)$ of each other, with **Symplex** paying 4 extra elements in σ_1 and 1 in σ_2 to encode the blind-pole machinery $([1/(x - \xi_i)]_*, [W_{\xi_i}/\delta]_1, [1/(\delta(x - \xi_i))]_1)$. The FFT savings of Remark 2 are unchanged.

FFT-free setup. The **Symplex** setup is sparse-arithmetic only. After a single batched inversion that produces $\{1/(x - \alpha)\}_\alpha \cup \{1/(x - \xi_1), 1/(x - \xi_2)\}$ in $O(n)$ field operations, the per-wire bundle scalars $W_\iota(x) = w_\iota(x) + y u_\iota(x) + z v_\iota(x)$ and the blind-pole compensators $W_{\xi_i}(x)$ are assembled in $O(\text{nnz}(L, R, O))$ field multiplications (Section 4.4), one per nonzero entry of L, R , or O , after which $O(m + n)$ group exponentiations lift the result into σ_1, σ_2 . No polynomial-coefficient or FFT step is required at any point. This is a direct consequence of the partial-fraction basis being intrinsically Lagrange-style: each u_ι, v_ι is already a sum of simple poles indexed by the constraint matrix entries, so the evaluation-to-coefficient conversion that Groth16's monomial basis $\{[x^i]_*\}$ would otherwise require, and is typically performed via an inverse FFT in common implementation, is not required.

6.4 Group-operation count and the role of the blind poles

A derived wire-indexed prover key. The unscaled constraint-pole queries used to form A and B admit a public linear change of basis. Define:

$$[u_\iota(x)]_1 := \sum_\alpha L_{\alpha, \iota} \left[\frac{1}{x - \alpha} \right]_1, \quad [v_\iota(x)]_2 := \sum_\alpha R_{\alpha, \iota} \left[\frac{1}{x - \alpha} \right]_2 \quad (0 \leq \iota \leq m).$$

Then, simply by exchanging the order of summation,

$$\sum_{\alpha} \ell_{\alpha} \left[\frac{1}{x-\alpha} \right]_1 = \sum_{\iota=0}^m a_{\iota} [u_{\iota}(x)]_1, \quad \sum_{\alpha} r_{\alpha} \left[\frac{1}{x-\alpha} \right]_2 = \sum_{\iota=0}^m a_{\iota} [v_{\iota}(x)]_2.$$

Thus the A - and B -MSMs may use the $m+1$ assignment scalars a_{ι} , giving width $m+3$ including the constant and blind terms, instead of the n constraint scalars $\ell_{\alpha}, r_{\alpha}$, which give width $n+2$. The prover still computes ℓ and r for H_{lin} . This optimization changes only the bases and scalars of the two source-group MSMs.

The derived queries require no trapdoor: they can be computed once from the original pole queries in $O(\text{nnz}(L))$ operations in $\mathbb{G}_1$ and $O(\text{nnz}(R))$ operations in $\mathbb{G}_2$, and then cached as part of the prover key. Equivalently, setup may publish the wire queries in place of the unscaled pole queries. Since they are public linear combinations of the original CRS, this transformation reveals no additional information. Let

$$\kappa := \min\{n, m+1\}.$$

Choosing the smaller of the constraint- and wire-indexed forms makes both the A - and B -MSM widths $\kappa+2$. If the corresponding unused unscaled basis is omitted, the transformed CRS counts are

$$|\sigma_1|_{\text{Sym, adaptive}} = n + m + 7 + \kappa, \quad |\sigma_2|_{\text{Sym, adaptive}} = 4 + \kappa.$$

The C -MSM and its δ -divided simple-pole queries are unchanged.

We now tally the prover's and verifier's group operations and compare with Groth16. The three proof elements $([A(x)]_1, [C(x)]_1, [B(x)]_2)$ are each assembled by a single multiexponentiation over CRS elements:

- $[A(x)]_1$ uses either its constraint-indexed form $[z]_1 + \sum_{\alpha} \ell_{\alpha} [1/(x-\alpha)]_1 + r[1/(x-\xi_1)]_1$ or its wire-indexed form $[z]_1 + \sum_{\iota} a_{\iota} [u_{\iota}(x)]_1 + r[1/(x-\xi_1)]_1$: one $\mathbb{G}_1$ -MSM of width $\kappa+2$ after choosing the smaller basis.
- $[B(x)]_2$ analogously uses either the constraint-indexed r_{α} form or the wire-indexed $a_{\iota} [v_{\iota}(x)]_2$ form: one $\mathbb{G}_2$ -MSM of width $\kappa+2$.
- $[C(x)]_1$ assembles $H_{\text{lin}}(x)/\delta$ (Remark 1) together with the private-wire bundles $\{[W_{\iota}(x)/\delta]_1\}_{\iota=\ell+1}^m$ and the two blind-pole compensators $[W_{\xi_i}(x)/\delta]_1$: one $\mathbb{G}_1$ -MSM of width $n + (m - \ell) + 4$.

The verifier reconstructs $\sum_{\iota=0}^{\ell} a_{\iota} [W_{\iota}(x)/\gamma]_1$ as one $\mathbb{G}_1$ -MSM of width $\ell+1$ and runs four pairings, three of them online if $e([z]_1, [y]_2)$ is precomputed into the verification key.

Groth16 admits the same adaptive indexing. Let $\{A_{\alpha}(X)\}_{\alpha}$ be the Lagrange basis on its n QAP points. Since

$$[u_i(x)]_1 = \sum_{\alpha} L_{\alpha,i} [A_{\alpha}(x)]_1, \quad [v_i(x)]_j = \sum_{\alpha} R_{\alpha,i} [A_{\alpha}(x)]_j \quad (j \in \{1, 2\}),$$

the corresponding wire-indexed sums can instead be written as the constraint-indexed sums $\sum_{\alpha} \ell_{\alpha} [A_{\alpha}(x)]_1$ and $\sum_{\alpha} r_{\alpha} [A_{\alpha}(x)]_j$. The Lagrange queries are public linear combinations of a monomial QAP CRS, so this is again a public change of basis. Groth16 can therefore use width $\kappa+2$ for both $[A]_1$ and $[B]_2$ (and for the auxiliary $[B_1]_1 \in \mathbb{G}_1$).

For $[C]_1$, expanding $sA + rB_1 - rs\delta$ contributes two u/v blocks of width κ plus the three constant terms. Together with the $m - \ell$ private-wire queries and the n coset-Lagrange quotient queries used by the six-FFT comparator, this gives a combined C -MSM width of

$$(m - \ell) + n + 2\kappa + 3 = 2\kappa + m + (n - \ell) + 3.$$

Equivalently, the prover may form $[B_1]_1$ separately and reuse the already computed $[A]_1$ and $[B_1]_1$ through variable-base scalar multiplications when assembling C . Table 2 retains the combined fixed-base-MSM accounting for a direct comparison.

	Symplex	Groth16
Prover $\mathbb{G}_1$ -MSM widths	$\kappa + 2, n + (m - \ell) + 4$	$\kappa + 2, 2\kappa + m + (n - \ell) + 3$
Prover $\mathbb{G}_2$ -MSM width	$\kappa + 2$	$\kappa + 2$
Total prover $\mathbb{G}_1$ basis elements	$\kappa + n + (m - \ell) + 6$	$3\kappa + m + (n - \ell) + 5$
Total prover $\mathbb{G}_2$ basis elements	$\kappa + 2$	$\kappa + 2$
Prover FFTs	4	6
Verifier $\mathbb{G}_1$ -MSM width	$\ell + 1$	$\ell + 1$
Verifier pairings (online)	3	3
Proof size	$2\mathbb{G}_1 + 1\mathbb{G}_2$	$2\mathbb{G}_1 + 1\mathbb{G}_2$

Table 2: Prover and verifier costs, with the n gate/QAP points chosen as roots of unity (cf. Remark 2 and Section 8 for the matched Groth16 FFT count). Here $\kappa = \min\{n, m + 1\}$. **Symplex** uses the smaller of its constraint-pole and derived wire-indexed bases; **Groth16** uses the smaller of its wire-indexed and constraint-Lagrange bases, including for the two expanded u/v blocks in C . MSM widths are partition-independent: Groth16 may instead reuse separately computed A and B_1 through variable-base scalar multiplications, as described above. The C -MSM remains indexed jointly by constraint poles and private wires. The Groth16 counts include its size- n coset-Lagrange quotient column; a monomial column reduces its C -MSM by one but adds a seventh FFT.

The cost reduction follows from two sources. Firstly, the FFT count drops from 6 to 4 (Remark 2). Secondly, the adaptive queries prevent either scheme's A/B -MSMs from paying the larger of the constraint and wire dimensions. After granting the same optimization to Groth16, the gap in combined total $\mathbb{G}_1$ MSM width is

$$3\kappa + m + (n - \ell) + 5 - (\kappa + n + (m - \ell) + 6) = 2\kappa - 1,$$

while the $\mathbb{G}_2$ widths are equal. Thus **Symplex**'s C -MSM is narrower by $2\kappa - 1$: this is $2m + 1$ in the wire-indexed branch and $2n - 1$ in the constraint-indexed branch. The blind-pole construction that lets **Symplex** avoid the wide $\{[u_i(x)]_1\} \cup \{[v_i(x)]_1\}$ contribution from $sA + rB_1 - rs\delta$ is analyzed in Remark 4.

Remark 4 (Blind poles vs. multiplicative randomization). The blind-pole terms $r/(X - \xi_1), s/(X - \xi_2)$ in U, V play the role of Groth16's multiplicative randomizers $r\delta, s\delta$. One alternative is to use the latter directly: drop the blind-pole terms

from U, V and instead set

$$A := z + U_0(x) + r\delta, \quad B := y + V_0(x) + s\delta,$$

so the corresponding proof elements are $[A]_1 \in \mathbb{G}_1$ and $[B]_2 \in \mathbb{G}_2$, where $U_0(X) := \sum_{\alpha} \ell_{\alpha}/(X - \alpha)$ and $V_0(X) := \sum_{\alpha} r_{\alpha}/(X - \alpha)$. The verifier equation is unchanged in form,

$$e([A]_1, [B]_2) = e([z]_1, [y]_2) \cdot e\left(\sum_{\iota=0}^{\ell} a_{\iota}[W_{\iota}(x)/\gamma]_1, [\gamma]_2\right) \cdot e([C]_1, [\delta]_2),$$

which is the Groth16 verification identity under the substitution $\alpha \mapsto z$ and $\beta \mapsto y$. Expanding the LHS and grouping public/private contributions (using $U_0V_0 = \sum_{\iota} a_{\iota}w_{\iota}(x) + H_{\text{lin}}^{(0)}(x)$, as in Theorem 2) yields

$$C = \frac{1}{\delta} \left[\sum_{\iota=\ell+1}^m a_{\iota}W_{\iota}(x) + H_{\text{lin}}^{(0)}(x) \right] + sA + rB_1 - rs\delta,$$

where $B_1 := y + V_0(x) + s\delta$ and $[B_1]_1 \in \mathbb{G}_1$ is the $\mathbb{G}_1$ -form of $[B]_2$. Forming $[B_1]_1$ requires publishing $[y]_1 \in \sigma_1$ and a *second* $\mathbb{G}_1$ -MSM of width $n + 2$ over the basis $\{[1/(x - \alpha)]_1\}_{\alpha} \cup \{[y]_1, [\delta]_1\}$ that the blind-pole prover does not need. This matches Groth16's prover, which must likewise form a $\mathbb{G}_1$ -form of $[B]_2$ in order to assemble $[C]_1$.

The blind-pole CRS elements $[W_{\xi_1}(x)/\delta]_1$ and $[W_{\xi_2}(x)/\delta]_1$ of Figure 1 replace the $sA + rB_1 - rs\delta$ block: both halves of the cross-randomization in the verifier identity are absorbed into a precomputed pair of group elements with disjoint pole support from the constraint poles, so the prover combines them in the *same* $\mathbb{G}_1$ -MSM as H_{lin}/δ and the private wires (cf. the closed form of $\mathbf{h}$ in Remark 1). The gadget is partial-fraction-native: encoding the cross term $r\delta y$ as a single rational $y/(X - \xi_1)$ at a fresh pole has no analog in Groth16's monomial basis, where every degree- $(n - 1)$ expression in u_i, v_i, w_i already entangles all constraint positions.

With the blind-pole construction, the cross-randomization inside $[C]_1$ contributes only the two CRS elements $[W_{\xi_i}(x)/\delta]_1$ to the prover MSM (and the corresponding $O(1)$ corrections to H_{lin} in Remark 1), versus the $2\kappa + 3$ elements from the two adaptively indexed u/v blocks and $\{[\alpha]_1, [\beta]_1, [\delta]_1\}$ that multiplicative randomization adds to Groth16's combined C -MSM. Together with Remark 2, this yields the $2\kappa - 1$ width gap in Table 2 for the coset-Lagrange comparator.

7 Proofs

We prove perfect completeness, perfect zero-knowledge, and knowledge-soundness for the Symplex construction of Figure 1.

7.1 Completeness

Theorem 2 (Perfect completeness). *For every $\mathbf{a} \in \mathbb{F}_p^{m+1}$ with $a_0 = 1$ satisfying $(La) \odot (Ra) = Oa$, an honestly generated proof is accepted by Verify.*

Proof. Using $A = z + U$ and $B = y + V$, we have

$$A(X)B(X) - zy = U(X)V(X) + yU(X) + zV(X).$$

The definitions of H_{lin} , U , and V , together with $\ell_\alpha r_\alpha = \sum_{\iota=0}^m O_{\alpha,\iota} a_\iota$, give

$$A(X)B(X) - zy = H_{\text{lin}}(X) + \sum_{\iota=0}^m a_\iota W_\iota(X) + rW_{\xi_1}(X) + sW_{\xi_2}(X).$$

By the definition of $C(X)$, the right-hand side equals $\sum_{\iota=0}^\ell a_\iota W_\iota(X) + \delta C(X)$. Evaluating at the setup point x and applying bilinearity yields the verifier equation of Figure 1. $\square$

7.2 Zero-knowledge

Theorem 3 (Perfect zero-knowledge). *Fix an R1CS instance R , an output (σ, τ) of $\text{Setup}(R)$ with $\tau = (x, y, z, \gamma, \delta)$, a public statement $(a_1, \dots, a_\ell)$, and any satisfying extension $\mathbf{a} = (1, a_1, \dots, a_m)$. The distributions*

$$\text{Prove}(R, \sigma, a_1, \dots, a_m) \quad \text{and} \quad \text{Sim}(R, \tau, a_1, \dots, a_\ell)$$

are identical.

Proof. For $r, s \xleftarrow{\$} \mathbb{F}_p$, the honest proof has

$$A(x) = z + \sum_\alpha \frac{\ell_\alpha}{x-\alpha} + \frac{r}{x-\xi_1}, \quad B(x) = y + \sum_\alpha \frac{r_\alpha}{x-\alpha} + \frac{s}{x-\xi_2}.$$

Since $x \neq \xi_1, \xi_2$, the map $(r, s) \mapsto (A(x), B(x))$ is an affine bijection of $\mathbb{F}_p^2$. Hence $(A(x), B(x))$ is uniform, exactly as the simulator's $(\tilde{A}, \tilde{B})$. By Theorem 2, the remaining exponent is uniquely determined by

$$C(x) = \frac{A(x)B(x) - zy - \sum_{\iota=0}^\ell a_\iota W_\iota(x)}{\delta},$$

which is exactly the simulator's definition of $\tilde{C}$. Thus the honest and simulated proofs have identical distributions. $\square$

7.3 Knowledge Soundness

Coefficients used by the extractor. Consider a generic prover that outputs a public statement $(a_1, \dots, a_\ell) \in \mathbb{F}_p^\ell$ and a proof $\pi = ([A]_1, [C]_1, [B]_2)$. The GGM simulator tracks the coefficient of every source-group input in these three outputs (Section 4.3). Only the following coefficients are needed:

$$\begin{aligned} a_z &: \text{coefficient of the } z \text{ slot in } A, \\ b_y, b_\gamma, b_\delta &: \text{coefficients of the } y, \gamma, \delta \text{ slots in } B, \\ a_\alpha, b_\alpha &: \text{coefficients of } 1/(x-\alpha) \text{ in } A, B, \\ \mu_\iota^A, \mu_\iota^C &: \text{coefficients of } W_\iota(x)/\delta \text{ in } A, C, \text{ for } \iota > \ell. \end{aligned}$$

Extend the private-wire coefficients by zeros to vectors $\boldsymbol{\mu}_A, \boldsymbol{\mu}_C \in \mathbb{F}_p^{m+1}$, and set

$$\mathbf{a}_{\text{pub}} := (1, a_1, \dots, a_\ell, 0, \dots, 0).$$

Formal acceptance. Replace x, y, z, γ, δ by indeterminates X, Y, Z, Γ, Δ , respectively, in the tracked CRS combinations, and write the resulting proof exponents as A^*, B^*, C^* . Also set

$$W_\ell^*(X) := w_\ell(X) + Y u_\ell(X) + Z v_\ell(X).$$

The verifier equation lifts from its evaluation at the sampled setup values to the Laurent-polynomial identity:

$$A^* B^* = ZY + \sum_{\ell=0}^{\ell} a_\ell W_\ell^* + \Delta C^* \quad (12)$$

with coefficients in $\mathbb{F}_p(X)$.

Lemma 3 (The required coefficient identities). *For every constraint α , identity (12) implies*

$$a_z b_y = 1, \quad b_\gamma = 0, \quad (13)$$

$$L_{\alpha,*} \mu_A = 0, \quad R_{\alpha,*} \mu_A = 0, \quad O_{\alpha,*} \mu_A = 0, \quad (14)$$

$$b_y a_\alpha = L_{\alpha,*}(\mathbf{a}_{\text{pub}} + \mu_C), \quad (15)$$

$$a_z b_\alpha = R_{\alpha,*}(\mathbf{a}_{\text{pub}} + \mu_C), \quad (16)$$

$$a_\alpha b_\alpha = O_{\alpha,*}(\mathbf{a}_{\text{pub}} + \mu_C). \quad (17)$$

Proof. Write $p_\alpha(X) = 1/(X - \alpha)$. We compare Laurent monomials in (12). Whenever p_α occurs, “coefficient” means the indicated principal-part coefficient at $X = \alpha$.

Step 1. Normalization. Only $(a_z Z)(b_y Y)$ can produce YZ on the left of (12), while its coefficient on the right is one. Likewise, only $(a_z Z)(b_\gamma \Gamma)$ can produce $Z\Gamma$, which is absent from the right. Hence

$$a_z b_y = 1, \quad a_z b_\gamma = 0.$$

Thus $a_z, b_y \neq 0$ and $b_\gamma = 0$, proving (13).

Step 2. Put the private bundles in A^* in the R1CS kernel. Their relevant part is:

$$\frac{1}{\Delta} \sum_{\ell > \ell} \mu_\ell^A W_\ell^* = \frac{1}{\Delta} \sum_{\ell > \ell} \mu_\ell^A (w_\ell + Y u_\ell + Z v_\ell).$$

The right side of (12) has no negative power of Δ . Comparing its three relevant coefficients with those of $A^* B^*$ therefore gives:

$$Y^2 \Delta^{-1} p_\alpha : \quad b_y L_{\alpha,*} \mu_A = 0,$$

$$YZ \Delta^{-1} p_\alpha : \quad b_y R_{\alpha,*} \mu_A = 0,$$

$$Y \Delta^{-1} p_\alpha^2 : \quad b_y O_{\alpha,*} \mu_A + b_\alpha L_{\alpha,*} \mu_A = 0.$$

The first two equations come from the $Y u_\ell$ and $Z v_\ell$ parts multiplied by $b_y Y$. In the last equation, the two summands come from $(w_\ell / \Delta)(b_y Y)$ and $(Y u_\ell / \Delta)(b_\alpha p_\alpha)$, respectively. Since $b_y \neq 0$, the first equation gives $L_{\alpha,*} \mu_A = 0$, the second gives $R_{\alpha,*} \mu_A = 0$, and substituting the first into the third gives $O_{\alpha,*} \mu_A = 0$. This proves (14).

Step 3. Recover the three R1CS rows. After Step 1, public-bundle terms from A^* vanish because they can contribute to these coefficients only through the $b_\gamma \Gamma$ term of B^* . The private bundles in A^* do still contribute: multiplication by the $b_\delta \Delta$ term of B^* gives

$$\left(\frac{1}{\Delta} \sum_{\iota > \ell} \mu_\iota^A W_\iota^* \right) (b_\delta \Delta) = b_\delta \sum_{\iota > \ell} \mu_\iota^A W_\iota^*.$$

Thus, before using (14), comparison of Yp_α , Zp_α , and p_α^2 gives:

coefficient	$A^* B^*$	right side of (12)
Yp_α	$b_y a_\alpha + b_\delta L_{\alpha,*} \mu_A$	$L_{\alpha,*}(\mathbf{a}_{\text{pub}} + \mu_C)$
Zp_α	$a_z b_\alpha + b_\delta R_{\alpha,*} \mu_A$	$R_{\alpha,*}(\mathbf{a}_{\text{pub}} + \mu_C)$
p_α^2	$a_\alpha b_\alpha + b_\delta O_{\alpha,*} \mu_A$	$O_{\alpha,*}(\mathbf{a}_{\text{pub}} + \mu_C)$

The public bundles on the right contribute the rows applied to $\mathbf{a}_{\text{pub}}$, and ΔC^* contributes the rows applied to μ_C . Blind-pole terms do not contribute to the three displayed coefficients. Equation (14) now sets the three terms multiplied by b_δ to zero. The rows of the table therefore reduce exactly to (15)–(17). $\square$

Theorem 4 (Knowledge soundness in the GGM). *For every polynomial-time generic prover, there is an efficient extractor that outputs a witness extending every accepted public statement on a collision-free GGM transcript. Consequently, its failure probability is at most the GGM collision bound of Section 4.3.*

Proof. The extractor returns the private-wire coefficients of C :

$$\chi_\iota := \mu_\iota^C, \quad \iota = \ell + 1, \dots, m,$$

and forms the corresponding full assignment

$$\hat{\mathbf{a}} := \mathbf{a}_{\text{pub}} + \mu_C.$$

By (15)–(17), for every constraint α ,

$$L_{\alpha,*} \hat{\mathbf{a}} = b_y a_\alpha, \quad R_{\alpha,*} \hat{\mathbf{a}} = a_z b_\alpha, \quad O_{\alpha,*} \hat{\mathbf{a}} = a_\alpha b_\alpha.$$

Using $a_z b_y = 1$,

$$(L_{\alpha,*} \hat{\mathbf{a}})(R_{\alpha,*} \hat{\mathbf{a}}) = (a_z b_y) a_\alpha b_\alpha = O_{\alpha,*} \hat{\mathbf{a}}.$$

Thus $\hat{\mathbf{a}}$ satisfies the R1CS, so $(\chi_{\ell+1}, \dots, \chi_m)$ is a witness for the claimed public statement. $\square$

Why a simple-pole univariate variant does not work. A tempting alternative is to eliminate the independent trapdoors, including y and z , by representing them as

additional simple poles in the same variable X .⁵ That univariate variant allows a prover to choose per-constraint vectors $\tilde{\ell}, \tilde{r}, \tilde{o}$ satisfying $\tilde{\ell} \odot \tilde{r} = \tilde{o}$, even when they did not arise as $L\mathbf{a}, R\mathbf{a}, O\mathbf{a}$ for any common wire vector, and to absorb the discrepancies into C using terms of the form

$$\frac{Y(\tilde{U} - U_{\mathbf{a}})}{\Delta} \quad \text{and} \quad \frac{Z(\tilde{V} - V_{\mathbf{a}})}{\Delta}.$$

Because Y and Z are now independent trapdoor indeterminates, these terms cannot be reduced to the unweighted simple-pole slots $1/(\Delta(X - \alpha))$ in the CRS. At a constraint pole, the relevant Y -, Z -, and squared-pole terms are available only through the tied bundle

$$\frac{W_{\iota}^*}{\Delta} = \frac{w_{\iota} + Y u_{\iota} + Z v_{\iota}}{\Delta}.$$

Consequently, in the present construction, the same bundle coefficients μ_C must account for all three constraint vectors. This is indeed what (15)–(17) formalize: they identify those vectors, up to the nonzero normalization $a_z b_y = 1$, with $L(\mathbf{a}_{\text{pub}} + \mu_C)$, $R(\mathbf{a}_{\text{pub}} + \mu_C)$, and $O(\mathbf{a}_{\text{pub}} + \mu_C)$. Thus the correction term required by the attack lies outside the CRS span unless its fake vectors already come from a common satisfying assignment. Appendix C shows that a different univariate specialization, using hierarchically separated high-order anchor poles, does preserve the required coefficient independence. There, we also show why the above counterexample extends to any scheme that uses poles of order less than n .

8 Implementation and Empirical Evaluation

We implemented Symplex and Groth16 on shared sparse-R1CS, FFT, fixed-base lifting, MSM, and pairing code, using wire-indexed prover queries for both schemes. Our benchmark is a Circom circuit that chains SHA-256 ten times: the first block hashes a 512-bit input, and each of the next nine blocks hashes the previous 256-bit output concatenated with 256 zero bits. The final hash is cross-checked against an independent ten-fold SHA-256 computation in Rust. The circuit has 587 766 non-linear constraints, $m = 619\,552$ wires, and $\ell = 512$ public inputs. Its R1CS contains 625 280 rows after Circom’s linear constraints are included and is padded to $n = 2^{20} = 1\,048\,576$ for the FFT prover.

All numbers below are from one single-threaded release-mode invocation over BLS12-381 on an Apple M1 Pro at 3.2 GHz. The setup algorithms publish the wire basis directly. In particular, Symplex publishes $\{[u_i(x)]_1, [v_i(x)]_2\}_{i=0}^m$ in place of its unscaled constraint-pole queries. The public post-setup derivation of Section 6.4 is therefore not included in the setup timing. Because $m + 1 < n$, the

⁵ An earlier version of our paper had this univariate construction. We are thankful to anonymous Asiacrypt 2026 reviewers for discovering the attack mentioned here. We also independently found the attacks using AI tools, fixed the construction and proved its security using Lean, as described in Section 9.

adaptive choice has $\kappa = m + 1 = 619\,553$. Groth16 uses its usual wire-indexed u_i, v_i queries and a size- n coset-Lagrange $t(x)/\delta$ column, which avoids the final coset-IFFT required by the usual monomial-basis column. The invocation sets up, proves, and verifies both schemes on the same instance.

	Symplex (wire)	Groth16 (wire)
$ \sigma_1 $ ($\mathbb{G}_1$ elements)	2 287 688	2 907 238
$ \sigma_2 $ ($\mathbb{G}_2$ elements)	619 557	619 556
Setup wall-clock	26.43 s	28.40 s
Prove wall-clock	7.56 s	13.90 s
Verify wall-clock	2.61 ms	2.67 ms
Prover $\mathbb{G}_1$ -MSM widths	{619 555, 1 667 620}	{619 555, 2 906 725}
Prover $\mathbb{G}_2$ -MSM width	619 555	619 555
Prover size- n FFTs	4	6
Verifier $\mathbb{G}_1$ -MSM width	$\ell + 1 = 513$	$\ell + 1 = 513$
Verifier online pairings	3	3

Table 3: Wire-indexed comparison on the chained SHA-256 ten-iteration Circom R1CS, single-threaded over BLS12-381. The prover ratio $7.56/13.90 = 0.544$ is a $1.84\times$ speedup for **Symplex**.

Prover. Wire indexing makes the first $\mathbb{G}_1$ -MSM and the $\mathbb{G}_2$ -MSM identical in width for the two schemes: both use the $m + 1$ assignment scalars together with two constant/blinding terms, for width $m + 3 = 619\,555$. The remaining difference is therefore concentrated in the C -MSM and the FFT stage. **Symplex**’s C -MSM has width $n + (m - \ell) + 4 = 1\,667\,620$, whereas the implemented coset-Lagrange Groth16 prover has width 2 906 725, about 1.239 million additional $\mathbb{G}_1$ bases. **Symplex** also performs four size- n FFTs instead of six. These two savings give the observed $1.84\times$ prover speedup.

CRS and setup. Publishing the wire queries in place of the unscaled pole queries reduces the **Symplex** CRS to $|\sigma_1| = n + m + 7 + \kappa = 2\,287\,688$ and $|\sigma_2| = 4 + \kappa = 619\,557$. Its $\mathbb{G}_1$ CRS is therefore 619 550 elements (21.3%) smaller than the implemented Groth16 CRS, while the two $\mathbb{G}_2$ CRSes differ by only one element. In this matched run, **Symplex** setup is also slightly faster (26.43 s versus 28.40 s).

Verification and scope. The change of prover basis does not modify either proof or verification: both schemes retain a $2\mathbb{G}_1 + 1\mathbb{G}_2$ proof, a width-513 public-input MSM, and three online pairings. The measured verification times remain within run-to-run noise. These are prototype measurements from a single machine and a single run. The exact MSM widths, FFT counts, and CRS sizes are deterministic, while the wall-clock figures should be read as an implementation-level comparison rather than a cross-platform performance claim.

9 Machine-Checked Verification in Lean

We formalized the information-theoretic core of `Symplex` in Lean 4 with Mathlib [dMU21, The20]. The development models the construction of Figure 1: the gate and blind poles share the formal variable X , while y, z, γ, δ are independent indeterminates. It represents partial fractions by their graded coordinates rather than by Mathlib’s rational-function type. Multiplication is therefore an executable table implementing the same-pole and distinct-pole identities of Section 6.1, and formal acceptance is coefficient-wise equality in the resulting graded Laurent representation. The mechanization establishes three top-level results:

Result	Machine-checked statement
Perfect completeness	The evaluated protocol is perfectly complete at every valid setup output.
Perfect HVZK	Over a finite field, honest proofs and simulated proofs have identical distributions at every fixed valid trapdoor.
Knowledge soundness	Every algebraic prover strategy satisfying the formal verification identity yields an R1CS witness, uniformly for every relation.

Completeness and zero-knowledge. For completeness, Lean verifies the partial-fraction recombination formula for H_{lin} and establishes the honest verification identity by comparing its four nonzero graded components, corresponding to the monomials yz , y , z , and 1 . The evaluated completeness theorem is proved directly in the field. Perfect HVZK is formalized as equality of probability mass functions: the map from the honest coins (r, s) to (A, B) is constructed as an explicit affine bijection of $\mathbb{F}_p^2$, and the simulator’s value of C is identified with the honest value using completeness. Thus the distributional claim is exact and does not rely on an informal conditioning argument.

Knowledge-soundness. The soundness proof mechanizes the coefficient argument of Lemma 3. Lean proves a single theorem for arbitrary R1CS matrices by comparing a fixed set of coefficients in the formal verification identity. These coefficients are indexed jointly by monomials in Y, Z, Γ, Δ and by partial-fraction pole coordinates. The resulting coefficient equations first imply $a_z b_y = 1$ and $b_\gamma = 0$, then show that the coefficients of the private-wire CRS elements occurring in A lie in the simultaneous kernel of L, R, O , and finally recover the L -, R -, and O -row relations from the coefficients of Yp_α , Zp_α , and p_α^2 . The result is one extractor theorem quantified over arbitrary fields and arbitrary R1CS matrices. Although $a_z b_y = 1$ permits the conventional normalization $a_z = b_y = 1$ by inversely rescaling A and B , the Lean proof retains the original proof elements and uses the relation $a_z b_y = 1$ directly. Its extracted private-wire vector initially contains a correction formed from the δ -coefficient of B and the private-wire coefficients of A . Lean proves that this correction lies in the simultaneous kernel of L, R , and O , so removing it does not change any constraint. This gives the simpler extractor from the coefficients of C used in Theorem 4.

Validation and axiomatic dependencies. The partial-fraction structure constants are checked internally by evaluation lemmas and externally against an independently transcribed symbolic model. Compile-time regression tests cover honest proofs, extraction, malformed witnesses, degenerate instances, and the known single-trapdoor forgery. Definitional-equality checks ensure that the top-level security statements are connected to the intended protocol definitions rather than to simplified surrogate definitions. The three results use only Lean’s standard axioms `propext`, `Classical.choice`, and `Quot.sound`. The Lean sources, build instructions, and validation artifacts are planned to be released soon.

Scope. The mechanized soundness theorem starts from an algebraic strategy that satisfies the verification equation as a formal identity. It does not formalize the reduction from a Type-III generic-group adversary to those source-group coefficients, the collision bound, or the Schwartz–Zippel step that lifts acceptance at the sampled setup values to formal acceptance. These are the probabilistic steps used in Theorem 4. The compilation from exponent identities to the pairing equation, and the circuit-to-R1CS reduction, are likewise outside the Lean development. Consequently, the formalization checks the algebraic core on which the paper’s security reductions rely while leaving those standard cryptographic reductions explicit in the paper. The same framework also verifies a high-order univariate specialization of Appendix C. Its additional ingredient is a hierarchical-independence lemma stated there.

Acknowledgments. During the preparation of this work, the authors used generative AI software tools (predominantly Claude) for grammar, spell checking, formatting, checking the correctness of the results, and prototyping the construction. The authors take full responsibility for the correctness and appropriateness of all material in this submission, including any AI-assisted material.

References

- ark22. arkworks contributors. arkworks-rs: A Rust ecosystem for zkSNARK programming. <https://arkworks.rs>, 2022.
- BBHR18. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018.
- BCJ⁺24. Foteini Baldimtsi, Konstantinos Kryptos Chalkias, Yan Ji, Jonas Lindstrøm, Deepak Maram, Ben Riva, Arnab Roy, Mahdi Sedaghat, and Joy Wang. zkLogin: Privacy-preserving blockchain authentication with existing credentials. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 3182–3196. ACM Press, October 2024.
- BCS16. Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Berlin, Heidelberg, October / November 2016.

- CJL⁺26. Kostas Kryptos Chalkias, Charanjit S. Jutla, Jonas Lindstrøm, Varun Madathil, and Arnab Roy. A linear operator framework for polynomial divisions in cryptography. In *Proceedings of the 21st ACM Asia Conference on Computer and Communications Security (ASIACCS 2026)*, Bangalore, India, 2026. ACM.
- DFGK14. George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 532–550. Springer, Berlin, Heidelberg, December 2014.
- DMS26. Michel Dellepère, Pratyush Mishra, and Alireza Shirzad. Garuda and Pari: Faster and smaller SNARKs via equi-efficient polynomial commitments. In *35th USENIX Security Symposium (USENIX Security 26)*, pages 5791–5810, Baltimore, MD, USA, August 2026. USENIX Association.
- dMU21. Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *CADE 28*, volume 12699 of *LNCS*, pages 625–635. Springer, 2021.
- DWF⁺20. Brecht Devos, Daniel Wang, Matthew Finestone, et al. Loopring 3.0 design doc. https://github.com/Loopring/protocols/blob/master/packages/loopring_v3/DESIGN.md, 2020.
- EHK⁺13. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Berlin, Heidelberg, August 2013.
- GGPR13. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Berlin, Heidelberg, May 2013.
- GM17. Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 581–612. Springer, Cham, August 2017.
- GO94. Israel Gohberg and Vadim Olshevsky. Complexity of multiplication with vectors for structured matrices. *Linear Algebra and its Applications*, 202:163–192, 1994.
- Gro16. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Berlin, Heidelberg, May 2016.
- HBHW24. Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash protocol specification, version 2024.5.1 [NU6]. <https://zips.z.cash/protocol/protocol.pdf>, 2024.
- ide24. iden3. snarkjs: a JavaScript implementation of zkSNARK schemes. <https://github.com/iden3/snarkjs>, 2024.
- JNR26. Charanjit S. Jutla, Rohit Nema, and Arnab Roy. Partial fraction techniques for cryptography. In Joan Daemen and Emmanuel Thomé, editors, *EUROCRYPT 2026, Part V*, volume 16545 of *LNCS*, pages 423–453. Springer, Cham, May 2026.
- KZG10. Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe,

- editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Berlin, Heidelberg, December 2010.
- Lip24. Helger Lipmaa. Polymath: Groth16 is not the limit. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 170–206. Springer, Cham, August 2024.
- LPS23. Helger Lipmaa, Roberto Parisella, and Janno Siim. Algebraic group model with oblivious sampling. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023, Part IV*, volume 14372 of *LNCS*, pages 363–392. Springer, Cham, November / December 2023.
- Mic00. Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.
- PHGR13. Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013.
- Pip80. Nicholas Pippenger. On the evaluation of powers and related problems. *SIAM J. Comput.*, 9(2):230–250, 1980.
- Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT’97*, volume 1233 of *LNCS*, pages 256–266. Springer, Berlin, Heidelberg, May 1997.
- The20. The mathlib Community. The lean mathematical library. In *CPP 2020*, pages 367–381. ACM, 2020.
- WPSP26a. Weijie Wang, Charalampos Papamanthou, Shravan Srinivasan, and Dimitrios Papadopoulos. Designated-verifier dynamic zk-SNARKs with applications to dynamic proofs of index. In *Proceedings of the 2026 ACM SIGSAC Conference on Computer and Communications Security (CCS ’26)*. ACM, 2026. To appear.
- WPSP26b. Weijie Wang, Charalampos Papamanthou, Shravan Srinivasan, and Dimitrios Papadopoulos. Dynamic zk-SNARKs (with applications to sparse zk-SNARKs and IVC). In Joan Daemen and Emmanuel Thomé, editors, *EUROCRYPT 2026, Part VII*, volume 16547 of *LNCS*, pages 329–358. Springer, Cham, May 2026.
- ZOZ26. Tianyu Zhang, Yupeng Ouyang, and Yupeng Zhang. Dynark: Making Groth16 dynamic. In Joan Daemen and Emmanuel Thomé, editors, *EUROCRYPT 2026, Part VII*, volume 16547 of *LNCS*, pages 178–208. Springer, Cham, May 2026.

A Non-interactive zero-knowledge arguments of knowledge

For a polynomial-time decidable binary relation R , we call ϕ the *statement* and w the *witness* when $(\phi, w) \in R$. Let $\mathcal{R}_\lambda$ be the set of relations output by a relation generator $\mathcal{R}$ on input 1^λ ; the generator may also output auxiliary information z for the adversary.

A *publicly verifiable non-interactive argument* for $\mathcal{R}$ is a quadruple of PPT algorithms (**Setup**, **Prove**, **Verify**, **Sim**) with the following interfaces:

- **Setup**(R) $\rightarrow (\sigma, \tau)$: On input a security parameter λ and a relation $R \in \mathcal{R}_\lambda$, output a common reference string σ and a simulation trapdoor τ for R .

- $\text{Prove}(R, \sigma, \phi, w) \rightarrow \pi$: On input σ and $(\phi, w) \in R$, output an argument π .
- $\text{Verify}(R, \sigma, \phi, \pi)$: On input σ , a statement ϕ , and an argument π , output 0 (reject) or 1 (accept).
- $\text{Sim}(R, \tau, \phi) \rightarrow \pi$: On input τ and ϕ , output an argument π .

Definition 1. We say $(\text{Setup}, \text{Prove}, \text{Verify}, \text{Sim})$ is a perfect non-interactive zero-knowledge argument of knowledge for $\mathcal{R}$ if it has perfect completeness, perfect zero-knowledge and computational knowledge soundness as defined below.

Perfect completeness. For all $\lambda \in \mathbb{N}$, $R \in \mathcal{R}_\lambda$, $(\phi, w) \in R$,

$$\Pr[(\sigma, \tau) \leftarrow \text{Setup}(R); \pi \leftarrow \text{Prove}(R, \sigma, \phi, w) : \text{Verify}(R, \sigma, \phi, \pi) = 1] = 1.$$

Perfect zero-knowledge. We say $(\text{Setup}, \text{Prove}, \text{Verify}, \text{Sim})$ is perfectly zero-knowledge if for all $\lambda \in \mathbb{N}$, $(R, z) \leftarrow \mathcal{R}(1^\lambda)$, $(\phi, w) \in R$, and all adversaries $\mathcal{A}$,

$$\begin{aligned} & \Pr[(\sigma, \tau) \leftarrow \text{Setup}(R); \pi \leftarrow \text{Prove}(R, \sigma, \phi, w) : \mathcal{A}(R, z, \sigma, \tau, \pi) = 1] \\ &= \Pr[(\sigma, \tau) \leftarrow \text{Setup}(R); \pi \leftarrow \text{Sim}(R, \tau, \phi) : \mathcal{A}(R, z, \sigma, \tau, \pi) = 1]. \end{aligned}$$

Computational knowledge soundness. An argument of knowledge admits an extractor that recovers a witness from any accepting adversary with access to its internal state. Formally, for every non-uniform PPT adversary $\mathcal{A}$ there is a non-uniform PPT extractor $\mathcal{X}_\mathcal{A}$ such that

$$\begin{aligned} & \Pr[(R, z) \leftarrow \mathcal{R}(1^\lambda); (\sigma, \tau) \leftarrow \text{Setup}(R); ((\phi, \pi); w) \leftarrow (\mathcal{A} \parallel \mathcal{X}_\mathcal{A})(R, z, \sigma) : \\ & (\phi, w) \notin R \text{ and } \text{Verify}(R, \sigma, \phi, \pi) = 1] \approx 0. \end{aligned}$$

Succinctness: SNARGs and SNARKs. A non-interactive argument where the verifier runs in polynomial time in $\lambda + |\phi|$ and the proof size is polynomial in λ is called a *preprocessing succinct non-interactive argument* (SNARG) if it is sound, and a *preprocessing succinct argument of knowledge* (SNARK) if it is knowledge-sound. If the CRS is also polynomial in λ , the scheme is *fully succinct*.

Public verifiability and designated-verifier proofs. One may split σ into prover and verifier parts σ_P, σ_V . The argument is *publicly verifiable* when σ_V is derivable from σ_P ; otherwise it is *designated-verifier*. In the latter setting, soundness and knowledge soundness are often stated with the adversary given only σ_P .

B Groth16 specification

Groth16 [Gro16] (*QAP version, for reference*).

Setup(R) $\rightarrow (\sigma, \tau)$: Pick $\alpha, \beta, \gamma, \delta, x \xleftarrow{\$} \mathbb{F}_p^*$. Let $\tau = (\alpha, \beta, \gamma, \delta, x)$ and compute $\sigma = ([\sigma_1]_1, [\sigma_2]_2)$ with

$$\sigma_1 = \left(\begin{array}{c} \alpha, \beta, \delta, \{x^i\}_{i=0}^{n-1}, \\ \left\{ \frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma} \right\}_{i=0}^{\ell}, \\ \left\{ \frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta} \right\}_{i=\ell+1}^m, \\ \left\{ \frac{x^i t(x)}{\delta} \right\}_{i=0}^{n-2} \end{array} \right), \quad \sigma_2 = (\beta, \gamma, \delta, \{x^i\}_{i=0}^{n-1}).$$

Prove($R, \sigma, a_1, \dots, a_m$) $\rightarrow \pi$: Pick $r, s \xleftarrow{\$} \mathbb{F}_p$ and output $\pi = ([A]_1, [C]_1, [B]_2)$ with

$$\begin{aligned} A &:= \alpha + \sum_{i=0}^m a_i u_i(x) + r\delta, \\ B &:= \beta + \sum_{i=0}^m a_i v_i(x) + s\delta, \\ C &:= \frac{1}{\delta} \sum_{i=\ell+1}^m a_i (\beta u_i(x) + \alpha v_i(x) + w_i(x)) + \frac{h(x)t(x)}{\delta} + As + Br - rs\delta. \end{aligned}$$

Verify($R, \sigma, a_1, \dots, a_\ell, \pi$): Check

$$[A]_1 \cdot [B]_2 \stackrel{?}{=} [\alpha]_1 \cdot [\beta]_2 + \sum_{i=0}^{\ell} a_i \left[\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma} \right]_1 \cdot [\gamma]_2 + [C]_1 \cdot [\delta]_2.$$

Sim($R, \tau, a_1, \dots, a_\ell$) $\rightarrow \pi$: Pick $A, B \xleftarrow{\$} \mathbb{F}_p$ and output $\pi = ([A]_1, [C]_1, [B]_2)$ with

$$C := \frac{1}{\delta} (AB - \alpha\beta - \sum_{i=0}^{\ell} a_i (\beta u_i(x) + \alpha v_i(x) + w_i(x))).$$

Fig. 2: The Groth16 zkSNARK for QAPs.

C A Single-Trapdoor Univariate Variant

The simple-pole univariate construction discussed after Theorem 4 is unsound because its auxiliary rational functions are linearly dependent on the low-order partial-fraction span available to the prover. This does not rule out every univariate construction.

However, the following exercise demonstrates that at least small pole orders will not suffice, and later we will show how sufficiently large order poles can indeed yield a secure scheme. To start with, let $y = \frac{1}{f(X)}$, where $f(X)$ is a degree t polynomial in X (irreducible over $\mathbb{F}$ or otherwise). Now, consider the term $y u_\ell(\tau)$ in the wire-bundle W_ℓ (see (11)). Since, $u_\ell(X)$ includes terms such as $p_{\alpha_g} = \frac{1}{X - \alpha_g}$ (for various gates g), we need to check if $y \cdot p_{\alpha_g}$ for various g collectively is independent of y and $\{p_{\alpha_g}\}_g$. Let $f(X) = \sum_{j=0}^t f_j * X^j$. Now, note that $f(X) - f(\alpha_g)$ has α_g as a root, and hence is divisible by $(X - \alpha_g)$. In other words, there exist a polynomial $h_g(X)$ of degree $t - 1$ such that $f(X) =$

$h_g(X)(X - \alpha_g) + f(\alpha_g)$. Then, we have:

$$\begin{aligned} y \cdot p_{\alpha_g} &= \frac{1}{f(X)} \cdot \frac{1}{X - \alpha_g} = \frac{-1}{f(\alpha_g)} \cdot \left(\frac{h_g(X)}{f(X)} - \frac{1}{X - \alpha_g} \right) \\ &= \frac{-1}{f(\alpha_g)} \cdot \left(\frac{h_g(X)}{f(X)} - p_{\alpha_g} \right) \end{aligned}$$

Since $h_g(X)$ is of degree at most $t - 1$, at most t such $h_g(X)$ can be linearly independent. Thus, if t is no more than the number of gates, we are guaranteed a linear dependence.

In this section we show that the independent parameters y, z, γ, δ of Figure 1 can instead be derived from one secret evaluation point using rational functions with sufficiently large, separated pole orders. The resulting scheme has a single-element trapdoor, while retaining the prover, verifier, proof size, and number of CRS elements of Symplex.

C.1 Construction

Let P contain the n constraint poles and the two blind poles, and write $q = n + 2$. Choose four additional public points $\eta_y, \eta_z, \eta_\gamma, \eta_\delta$, distinct from P and from one another, together with positive integers satisfying:

$$\begin{aligned} N_\delta &> 3q, \\ N_\gamma &> 2N_\delta + 3q, \\ N_z &> 2N_\gamma + 2N_\delta + 3q, \\ N_y &> 2N_z + 2N_\gamma + 2N_\delta + 3q. \end{aligned} \tag{18}$$

These orders are $O(n)$ and may be chosen recursively as the least integers satisfying the four inequalities.

Setup now samples only

$$x \xleftarrow{\$} \mathbb{F}_p \setminus (P \cup \{\eta_y, \eta_z, \eta_\gamma, \eta_\delta\})$$

and derives

$$y = (x - \eta_y)^{-N_y}, \quad z = (x - \eta_z)^{-N_z}, \quad \gamma = (x - \eta_\gamma)^{-N_\gamma}, \quad \delta = (x - \eta_\delta)^{-N_\delta}. \tag{19}$$

It then generates exactly the CRS of Figure 1 using these derived values. Thus the trapdoor is the single scalar x . The construction requires four additional field exponentiations during setup but publishes no additional group elements. Proving, verification, and simulation are unchanged. In particular, the simulator can derive all four parameters from the trapdoor x .

The use of high orders is essential. If, for example, $y = 1/f(X)$ for a polynomial f of degree t , then at every constraint pole α ,

$$\frac{y}{X - \alpha} = -\frac{1}{f(\alpha)} \left(\frac{h_\alpha(X)}{f(X)} - \frac{1}{X - \alpha} \right), \quad f(X) = h_\alpha(X)(X - \alpha) + f(\alpha).$$

The polynomials h_α have degree at most $t - 1$, so low-degree auxiliary functions create linear dependencies when sufficiently many constraint poles are present. The order hierarchy in (18) ensures that the four derived functions cannot participate in cancellations of this bounded degree.

C.2 Hierarchical independence

Introduce the formal rational functions

$$Y = (X - \eta_y)^{-N_y}, \quad Z = (X - \eta_z)^{-N_z}, \quad \Gamma = (X - \eta_\gamma)^{-N_\gamma}, \quad \Delta = (X - \eta_\delta)^{-N_\delta},$$

and the inner partial-fraction family

$$\mathcal{T} = \{1\} \cup \{(X - \rho)^{-k} : \rho \in P, 1 \leq k \leq 3\}.$$

Every term that can occur after expanding the formal verification identity lies in the span of

$$\mathcal{F} = \{Y^{e_y} Z^{e_z} \Gamma^{e_\gamma} \Delta^{e_\delta} t : (e_y, e_z) \in \{0, 1, 2\}^2, (e_\gamma, e_\delta) \in \{-1, 0, 1\}^2, t \in \mathcal{T}\}. \quad (20)$$

The exponent range is deliberately conservative: it contains all products of a $\mathbb{G}_1$ CRS row with a $\mathbb{G}_2$ CRS row, as well as every term on the right of the verifier equation. Products of two inner partial fractions reduce to $\mathcal{T}$ by the identities of Section 6.1.

Lemma 4 (Hierarchical independence). *Under (18), the family $\mathcal{F}$ is linearly independent over $\mathbb{F}_p$.*

Proof. Suppose that a nonzero linear combination of $\mathcal{F}$ vanishes. Group its terms by e_y , and let e_y^* be the largest exponent with a nonzero group. After factoring $Y^{e_y^*}$, the remaining rational function has numerator degree at most:

$$2N_z + 2N_\gamma + 2N_\delta + 3q < N_y.$$

It therefore cannot have a zero of order N_y at η_y . Hence the e_y^* group has a pole of strictly greater order at η_y than all groups with smaller e_y and cannot be cancelled by them unless that group itself vanishes.

Apply the same argument successively at η_z , η_γ , and η_δ . At each step the next inequality in (18) bounds the maximum vanishing order contributed by all terms with smaller exponents. This reduces the alleged relation to one among the elements of $\mathcal{T}$. Those elements are linearly independent: after multiplication by the common denominator $\prod_{\rho \in P} (X - \rho)^3$, their coefficients are recovered from the principal parts at the distinct points ρ . Thus every original coefficient is zero, contradicting the assumed nontrivial relation. $\square$

The lemma is a valuation-separation statement: the four pole orders preserve, within the one-variable function field, the grading supplied by independent indeterminates in Equation (12).

C.3 Security

Theorem 5 (Security of the univariate variant). *The construction of Section C.1 is perfectly complete and perfectly zero-knowledge. It is knowledge-sound in the GGM with the same collision term as Theorem 4 and an additional Schwartz–Zippel term $O(n/p)$.*

Proof. Completeness is an identity for every nonzero choice of y, z, γ, δ , and the values in (19) are nonzero on every setup output. The zero-knowledge proof is also unchanged: for fixed x , the two affine maps from the coins to A and B have nonzero slopes because x is distinct from the blind poles.

For soundness, substitute the four rational functions Y, Z, Γ, Δ above into the formal verifier equation. Both sides lie in the span of $\mathcal{F}$. By Lemma 4, equality as rational functions forces equality of the coefficient of each member of $\mathcal{F}$, and hence exactly the coefficient identities of Lemma 3. The extractor of Theorem 4 therefore applies without modification. After clearing denominators, the specialized identity has degree $O(n)$ because all four orders are $O(n)$, giving the stated Schwartz–Zippel term. $\square$

The simple-pole attack fails here because the normalization identity $a_z b_y = 1$ remains enforceable: a nonzero residual multiple of ZY cannot belong to the span of the remaining CRS terms, since Lemma 4 makes ZY linearly independent of those terms. More generally, distinct monomials in Y, Z, Γ, Δ remain linearly independent from the gate-pole basis throughout the finite family (20).

C.4 AI-Assisted Discovery and Lean Verification

This variant emerged from an iterative interaction between formal verification and AI-assisted mathematical exploration. The Lean formalization first reduced soundness of the scheme with independent parameters y, z, γ, δ to a fixed finite collection of coefficient identities. That reduction isolated the exact requirement for a univariate specialization: it need not preserve arbitrary algebraic independence, but only injectivity on the finite family (20) appearing in the verifier equation.

Generative-AI tools were then used during the exploratory phase to examine failed low-order encodings, identify candidate choices with separated pole orders, and refine possible degree bounds and valuation arguments. Lean provided formal validation: the candidate definitions were connected to the existing formal verifier, the order conditions were made explicit, and the hierarchical-independence argument and its reduction to the verified coefficient identities were machine-checked. The known low-order forgery served throughout as a regression test of whether a proposed specialization enforced the normalization identity $a_z b_y = 1$. These tools supported exploration of the candidate space, while the proof assistant checked the resulting logical argument. Neither generated output nor finite testing is treated as evidence in place of the proof. The authors selected and audited the resulting construction and remain responsible for all claims. As noted in Section 9, the Lean sources and validation artifacts are planned to be released soon.