

Low-Space Quantum Discrete Logarithms on Genus-Two Jacobians

Yan Huang¹, Yulin Chen¹, and Fangguo Zhang²

¹ School of Mathematics and Statistics, Hunan University of Science and Technology, Xiangtan 411201, China.

² School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510006, China.

Abstract. Generic divisor-addition formulas use little arithmetic workspace. However, these formulas fail on exceptional inputs. We prove a randomized reduction that allows quantum phase computation to use these formulas. The reduction uses two independent public group seeds over a finite abelian group of odd order. For each fixed input, the two children at each node of a complete balanced addition tree are independent and uniform. The density of exceptional pairs then bounds the total failure probability and the error of the randomized quantum channel.

We apply the reduction to an odd prime-order subgroup of size r in a genus-two Jacobian over $\mathbb{F}_q$. Here q is an odd prime. The curve model is a monic quintic with zero quartic coefficient. The construction uses an integer lift of weighted Mumford arithmetic, a terminal polynomial for phase computation, and an independent shifted Legendre symbol. These components give useful-sample probability $1 - O(L/q)$ on the exact domain $\mathbb{Z}_r^2$ when $r = \Theta(q^2)$. The notation $\mathbb{Z}_r$ denotes $\mathbb{Z}/r\mathbb{Z}$, and L is the number of tree leaves. This bound assumes exact preparation and Fourier readout. Explicit height certificates, streaming CRT reconstruction, and clean modular primitives give a sampler that uses $10n + o(n)$ logical qubits. This count includes both scalar registers, and $n = \lceil \log_2 q \rceil$ is the field bit length. The model allows intermediate measurements, reset, and classical feedforward.

For the Gaudry–Schost instance over $\mathbb{F}_{2^{127}-1}$, a 16-bit window gives an analytic allocation of 1,923 qubits, compared with 2,172 and 3,063 qubits for the matched Huang et al. and Chen et al. module schedules. These schedules retain the same scalar inputs. The bound for a capped run is below 2^{60} Toffoli gates. A 32-bit window uses 1,618 qubits, but its Toffoli bound is about $2^{70.43}$ and its classical tables contain at most 2^{36} entries. Rotation synthesis is charged separately. These estimates describe an explicit space–time tradeoff.

Keywords: quantum discrete logarithms; genus-two Jacobians; reversible divisor arithmetic; Legendre symbol; output compression.

1 Introduction

Shor’s algorithm factors integers and solves the discrete logarithm problem in finite cyclic groups in polynomial time [38]. Its implementation requires reversible group arithmetic. This arithmetic determines much of the quantum gate count and the number of logical qubits. Resource estimates therefore help us assess the cost of quantum attacks on public-key cryptography.

Several works have reduced the quantum space needed to factor RSA moduli. We use B for the bit length of an RSA modulus. Beauregard’s circuit uses $2B + 3$ logical qubits [2]. Häner, Roetteler, and Svore gave a $2B + 2$ -qubit implementation with Toffoli-based modular multiplication [21]. Chevignard, Fouque, and Schrottenloher later reduced the asymptotic width to $B/2 + o(B)$ for balanced RSA moduli [8]. Their construction combines residue arithmetic, output compression, and the Ekerå–Håstad short-discrete-logarithm algorithm [17]. It computes a small part of the modular-exponentiation output with sublinear workspace. The algorithm uses repeated quantum samples and classical postprocessing. Gidney further optimized the residue-arithmetic approach and reported an estimate of 2,043 logical qubits for a 3,072-bit RSA modulus [19].

Researchers have also reduced the space needed for elliptic-curve discrete logarithms over prime fields. Here n denotes the bit length of the base-field prime. Proos and Zalka proposed a register-sharing method with an asymptotic allocation of $5n + o(n)$ logical qubits [33]. Roetteler et al. [35] and Häner et al. [22] gave explicit arithmetic circuits and detailed resource estimates. Chevignard, Fouque, and Schrottenloher then

developed an output-compression approach with $3.12n + o(n)$ logical qubits [10]. This reduction requires a larger quantum gate count.

Arithmetic optimization provides another route to lower space use. Luo et al. [30] gave a prime-field elliptic-curve algorithm with $3n + 6\lceil \log_2 n \rceil + O(1)$ logical qubits. Their construction uses a compact reversible inversion circuit, register sharing, intermediate measurements, and classical feedforward. These developments show that both the arithmetic implementation and the information retained for period finding affect quantum space.

Genus-two Jacobians provide a natural setting for this problem. A reduced divisor of degree two is specified by four field coefficients. A quantum circuit must also store temporary values for divisor addition and normalization. Huang et al. [24] estimated these resources over prime fields. Chen et al. [7] later gave explicit circuits over Mersenne prime fields. These works provide concrete arithmetic baselines. They also motivate the question studied here: how much quantum space can we save if we compute only the information needed for Fourier sampling?

Shor’s algorithm and output compression. Let C be a genus-two curve over an odd prime field $\mathbb{F}_q$. Let $G = \langle P \rangle \subseteq \text{Jac}(C)(\mathbb{F}_q)$ have known odd prime order r . We write $\mathbb{Z}_r = \mathbb{Z}/r\mathbb{Z}$ and use $0, \dots, r-1$ as register representatives. The notation $[a]D$ denotes scalar multiplication of a divisor class D . Given $Q = [k]P$, we seek $k \in \mathbb{Z}_r$. The group-valued function

$$F(a, b) = [a]P + [b]Q, \quad (a, b) \in \mathbb{Z}_r^2,$$

satisfies $F(a+k, b-1) = F(a, b)$. Quantum Fourier sampling uses this periodicity to recover k . A standard evaluation circuit stores a canonical representation of $F(a, b)$ before the Fourier stage.

Output compression replaces this representation with a smaller value that preserves enough information about the period. May and Schlieper [31] studied this approach for period finding. Cheignard, Fouque, and Schrottenloher (CFS) [10] developed a concrete construction for elliptic curves over prime fields. Their construction lifts projective-coordinate formulas to the integers and evaluates them through a residue number system. It then uses a Legendre symbol to compress the output to one bit. This method avoids field inversion during the compressed evaluation. It reduces quantum storage at the cost of more gates. The space saving comes from computing the compressed value directly. A circuit that first computes a full group element still reaches the peak width of that evaluator, even if it later compresses the result.

The genus-two case. An extension to genus two requires additional arguments. Cantor’s algorithm provides complete divisor arithmetic [5]. Earlier work also gives explicit weighted coordinates and inversion-free addition laws [28,13,23,32]. Our construction uses a generic weighted Mumford formula. This formula can fail when a denominator vanishes or a reduced divisor has lower degree. A quantum circuit must account for these exceptional inputs.

Two further issues arise. First, integer lifting enlarges the intermediate values. Their size determines how many auxiliary moduli the residue computation needs. Second, a phase that is invariant under coordinate scaling need not produce useful Fourier samples. We must therefore control exceptional additions, bound integer growth, and prove the sampling property of the chosen phase.

Contribution. We give a compressed discrete-logarithm sampler for curves with a square-free monic quintic model $C : y^2 = f(x)$ whose quartic coefficient is zero. Section 2 states the model assumptions. We write $n = \lceil \log_2 q \rceil$ and $m_r = \lceil \log_2 r \rceil$. For $r = \Theta(q^2)$ and window width $w = \Theta(\log_2 n)$, our sampler uses at most $10n + o(n)$ logical qubits. This bound includes both m_r -bit scalar registers. We count peak live logical qubits in a model that allows intermediate measurements, reset, and classical feedforward. The classical preprocessing and quantum arithmetic have polynomial cost under this window choice. Theorem 6.1 states the full result. The construction has two main components within the output-compression and residue-arithmetic framework of CFS [10]. Binary Jacobi evaluation [37] and measured tree evaluation [27] supply the phase and cleanup methods.

First, we randomize the leaves of a complete balanced addition tree using two independent uniform public group seeds. For each fixed scalar input and each internal node, the two true child sums are independent and uniform. This property turns the density of exceptional pairs into a failure bound that holds uniformly over inputs. For a genus-two tree with L leaves, the bound is $(L-1)(14q+3)/r$. We also obtain an error bound

for the randomized quantum channel. This bound averages over the fresh seeds and retains their classical records. The input and any reference system must be independent of those seeds. The masking argument applies to finite abelian groups of odd order and does not depend on the genus-two coordinate system. It builds on affine randomization [6] and the random translations used in earlier quantum discrete-logarithm analyses [33,35].

Second, we give an integer lift of the weighted Mumford formulas and a reversible schedule for the compressed phase. An independent uniform shift $\gamma \in \mathbb{F}_q$ defines the shifted Legendre phase. A terminal polynomial of degree at most 19 evaluates this phase without field inversion on generic inputs. We evaluate the polynomial modulo small auxiliary primes and use streaming Chinese remainder reconstruction to obtain its value modulo q . Explicit height bounds determine the required prime product. A shared evaluation schedule and its inverse clear the temporary registers. These bounds and cleanup certificates make the quantum-space estimate explicit. The integer lift and its reversible implementation are our arithmetic contributions beyond the existing coordinate formulas.

We prove the useful-sample bound for uniform inputs on the exact domain $\mathbb{Z}_r^2$. With exact preparation and Fourier readout, the probability is $1 - O(L/q)$ when $r = \Theta(q^2)$. Each run uses fresh tree seeds and an independent Legendre shift. The finite bound also includes preparation, Fourier, and rotation-synthesis errors. Every accepted candidate is checked classically by $[k']P = Q$.

Resource estimates. We evaluate the construction on the Gaudry–Schost instance over $\mathbb{F}_{2^{127}-1}$. A 16-bit window gives an upper bound of 1,923 logical qubits and a capped-run bound below 2^{60} Toffoli gates. The classical leaf tables contain at most 2^{21} entries. For comparison, the matched schedules derived from the addition modules of Huang et al. and Chen et al. use 2,172 and 3,063 qubits, respectively. All three schedules retain the same 500 scalar-input qubits. These comparisons refer to the specified module allocations; they do not establish optimality over all circuit designs.

A 32-bit window further reduces the allocation to 1,618 qubits. Its Toffoli bound is about $2^{70.43}$, and its classical tables contain at most 2^{36} entries. The quantum gate estimates include reversible table selection, reconstruction, cleanup, preparation, and Fourier readout. Rotation synthesis is charged separately. These estimates show an explicit tradeoff: the construction saves logical qubits, but it requires substantially more gates and classical storage. Section 6.2 gives the detailed comparisons.

Organization of the paper. Section 2 introduces the curve model and reversible arithmetic. Section 3 recalls compressed Fourier sampling on the exact input domain and fixes the resource conventions. Section 4 constructs the shifted Legendre phase and its terminal polynomial. Section 5 gives the residue tree, height bounds, reconstruction, and masking theorem. Section 6 combines these results into the sampler and evaluates its resources. Section 7 discusses the remaining questions. Appendices A–E contain proof details, implementation certificates, and comparison data. The accompanying source archive contains the verification programs and their recorded results.

2 Hyperelliptic Curves and Arithmetic

This section starts with reversible arithmetic over the base field. It then specifies the curve and subgroup used in the discrete logarithm problem. Finally, it specifies the canonical Mumford encoding and recalls complete divisor arithmetic and its standard reversible implementation.

2.1 Arithmetic over the base field

Modular multiplication, modular squaring, and modular inversion are comparatively costly operations in quantum finite-field arithmetic. We briefly describe their algorithms and summarize the quantum resources required to implement them. Let q be an odd prime, and set $n = \lceil \log_2 q \rceil$. Each element of $\mathbb{F}_q$ is represented by its canonical residue in $\{0, \dots, q-1\}$, using an n -qubit register. The resource estimates count peak live logical qubits, including all source, target, and workspace registers, and Toffoli gates for the complete computation and workspace cleanup.

Multiplication. We implement multiplication by the clean accumulation map

$$|x, y, z, 0\rangle \mapsto |x, y, z + xy \bmod q, 0\rangle, \quad 0 \leq x, y, z < q.$$

The target is distinct from the sources and may hold any canonical residue. Both sources are preserved and all workspace returns to zero. The Horner method scans the bits of y from most to least significant, alternating controlled additions of x with modular doubling of the accumulator. Luo et al. [30, Appendix B] implement each controlled addition with $11n + O(1)$ Toffolis and each doubling with $6n + O(1)$. Their zero-target multiplier uses n additions and $n - 1$ doublings, giving $17n^2 + O(n)$ Toffolis and $3n + O(1)$ logical qubits. To accumulate into an arbitrary initial target z , we first replace it by $2^{-(n-1)}z \bmod q$. The subsequent scan returns

$$2^{n-1}(2^{-(n-1)}z \bmod q) + x \sum_{j=0}^{n-1} y_j 2^j \equiv z + xy \pmod{q}.$$

The $n - 1$ initial halvings have the same cost as modular doubling. Our complete multiplier therefore uses $3n + O(1)$ logical qubits and

$$n(11n + O(1)) + 2(n - 1)(6n + O(1)) = 23n^2 + O(n)$$

Toffoli gates. The fixed-modulus arithmetic uses the measurement-assisted adder of Gidney [18]; borrowed source qubits are restored after each block, with the prescribed measurement-dependent phase corrections.

For a Mersenne prime $q = 2^n - 1$, the congruence $2^n \equiv 1 \pmod{q}$ makes modular doubling and halving cyclic bit rotations. The all-zero word represents zero, and the all-one word q is excluded. Rotations preserve this canonical encoding and use no Toffolis. Combining them with the Luo controlled adder reduces the complete multiplication cost to $11n^2 + O(n)$ Toffolis, with the same $3n + O(1)$ width. A coherent implementation of the same rotation method uses $4n + 4$ logical qubits and $16n^2 - 5n$ Toffolis; see (45). The cyclic-shift approach also appears in [11, 7].

An alternative computes the $2n$ -bit integer product and reduces it by folding the high half into the low half. Writing $xy = c_0 + 2^n c_1$, the reduction is

$$s = c_0 + c_1, \quad xy \bmod q = s - q \mathbf{1}_{s \geq q}, \quad 0 \leq s < 2q,$$

where $\mathbf{1}_E$ denotes the indicator of E . One conditional subtraction suffices, including when $s = q$. After the reduced product is added to the target, the fold and integer product are uncomputed. This complete construction uses $7n + 4$ logical qubits and $8n^2 + 28n - 15$ Toffolis (Appendix E.2). It follows the Mersenne reduction used in [25, Appendix A]. For $n \geq 4$, its gate bound is smaller than that of the coherent rotation multiplier, at the cost of $3n$ additional qubits.

Squaring. The corresponding clean accumulation map for squaring is

$$|x, z, 0\rangle \mapsto |x, z + x^2 \bmod q, 0\rangle, \quad 0 \leq x, z < q.$$

The source is preserved, the target is a distinct canonical-residue register, and all workspace returns to zero. Squaring can be implemented by multiplication with equal operands. For the Luo Horner circuit, CNOT gates first supply a separate copy of the source bits. After accumulation, the same CNOTs clear that copy. The original source, copied source, and target occupy three n -qubit words. Consequently, this construction uses $3n + O(1)$ logical qubits and $23n^2 + O(n)$ Toffolis over a general prime field, or $11n^2 + O(n)$ over a Mersenne prime field. Preparing and clearing the copy costs $2n$ CNOTs and no Toffolis.

Dedicated squaring also exploits the repeated partial products in

$$x^2 = \sum_{j=0}^{n-1} x_j 2^{2j} + \sum_{0 \leq i < j < n} x_i x_j 2^{i+j+1}, \quad x = \sum_{j=0}^{n-1} x_j 2^j.$$

The diagonal terms require no nonlinear bit products, and each off-diagonal pair occurs only once. This identity underlies the integer squarers in [12, 15]. For the coherent Mersenne rotation circuit, the equality $x_j^2 = x_j$ saves six Toffolis per controlled update and removes the need for a second source word. Its complete square-accumulate map uses $3n + 4$ logical qubits and $16n^2 - 11n$ Toffolis. The product-and-fold squarer uses $6n + 4$ logical qubits and $8n^2 + 24n - 15$ Toffolis, including reduction, target accumulation, and reversal of both temporary computations. As with multiplication, folding has a smaller gate bound for $n \geq 4$ and uses $3n$ more qubits. The constructions and their cleanup are described in Section 5.3 and Appendix E.2.

Inversion. Define $\text{inv}_0(0) = 0$ and $\text{inv}_0(x) = x^{-1}$ for $x \neq 0$. Normalization uses the clean inverse map

$$|x\rangle|z\rangle|0\rangle \mapsto |x\rangle|z \oplus \text{inv}_0(x)\rangle|0\rangle, \quad 0 \leq x < q, \quad z \in \{0, 1\}^n.$$

Here $\oplus$ is bitwise XOR, so the target may be any n -bit word. The extended Euclidean algorithm computes the inverse by updating the remainders of (x, q) together with their Bézout coefficients. The terminal gcd is one for $x \neq 0$, and its coefficient of x gives $x^{-1} \bmod q$. Luo et al. [30, Sections 3, 6.1–6.2] share registers between shrinking remainders and growing coefficients, using length registers to locate the active data. Their forward circuit replaces x by x^{-1} and retains the Euclidean state needed for reversal. It uses at most $2n + 6\lceil \log_2 n \rceil + 19$ logical qubits and $229n^2 + O(n \log_2 n)$ Toffolis.

Our clean interface adds an n -qubit target and one zero-input flag. We compute $e = [x = 0]$, replace a zero source by one, run the forward Euclidean circuit, and XOR its inverse into z when $e = 0$. Running the inverse logical circuit then restores the source and clears the Euclidean state. Finally, the zero-input substitution is undone and the flag is erased. The forward and reverse calls have the same cost; the zero tests and controlled copy add $5n - 6$ Toffolis. Thus the complete inverse uses

$$3n + 6\lceil \log_2 n \rceil + 20 \quad \text{logical qubits}, \quad 458n^2 + O(n \log_2 n) \quad \text{Toffoli gates}.$$

These bounds include source restoration, the arbitrary XOR target, and workspace cleanup. The measurement-assisted blocks are implemented with their phase corrections, including in the inverse logical circuit.

For $q = 2^n - 1$, a coherent binary Euclidean implementation gives a different tradeoff. Each of $2n - 1$ fixed rounds uses parity and comparison tests, controlled swaps and subtractions, and cyclic right rotations to halve the remainder and its modular coefficient. Two recorded bits per round allow all updates to be reversed after the inverse has been XORed into the target. This zero-completed construction uses $10n + 2$ logical qubits and $104n^2 - 80n + 14$ Toffolis, with no intermediate measurements (Appendix A). Its larger width buys a smaller quadratic gate coefficient. An independent binary construction for general odd primes is given in Appendix A.

Resource summary. Table 1 summarizes these complete interfaces. Multiplication and squaring update a canonical target by modular addition; inversion updates a binary target by XOR. All rows preserve their sources and clear all workspace. The Horner rows and the Luo inverse adapter allow intermediate measurements, reset, and classical feedforward; the other rows are coherent.

Table 1. Resource upper bounds for clean base-field arithmetic, including all input and target registers and workspace cleanup. The Horner bounds include compensation for a nonzero initial target; their squaring implementations include a separate source copy.

Operation	Implementation	Logical qubits	Toffoli gates
General odd prime q , with $n = \lceil \log_2 q \rceil$			
Multiplication	Luo Horner	$3n + O(1)$	$23n^2 + O(n)$
Squaring	Luo Horner	$3n + O(1)$	$23n^2 + O(n)$
Inversion	Luo adapter	$3n + 6\lceil \log_2 n \rceil + 20$	$458n^2 + O(n \log_2 n)$
Mersenne prime $q = 2^n - 1$			
Multiplication	Luo Horner	$3n + O(1)$	$11n^2 + O(n)$
Squaring	Luo Horner	$3n + O(1)$	$11n^2 + O(n)$
Multiplication	Coherent rotations	$4n + 4$	$16n^2 - 5n$
Squaring	Coherent rotations	$3n + 4$	$16n^2 - 11n$
Multiplication	Product and fold	$7n + 4$	$8n^2 + 28n - 15$
Squaring	Product and fold	$6n + 4$	$8n^2 + 24n - 15$
Inversion	Binary rotations	$10n + 2$	$104n^2 - 80n + 14$

All three operations have linear qubit bounds and quadratic Toffoli bounds, with different leading constants. The larger gate coefficient of clean inversion motivates the inversion-free polynomial map used by the residue

sampler in Section 5. The Mersenne bounds apply only when the active modulus has that form; a Mersenne base field does not make the auxiliary CRT primes Mersenne. The finite sampler estimates in Section 6 use the explicit coherent primitive constants of Section 5.3. Additional outer controls are accounted for separately.

2.2 Curve model, encodings, and reference arithmetic

The curve C is the smooth projective model of

$$C: y^2 = f(x),$$

where the supplied model has

$$f(x) = x^5 + c_3x^3 + c_2x^2 + c_1x + c_0 \in \mathbb{F}_q[x] \quad \text{square-free.} \quad (1)$$

We assume that the model is a monic quintic with zero quartic coefficient. Not every genus-two curve over $\mathbb{F}_q$ has such a model. When $q \neq 5$, a translation removes the quartic coefficient of an existing monic quintic. In characteristic five, that coefficient must already vanish. The symbol ∞ denotes the unique point at infinity. We define

$$J = \text{Jac}(C)(\mathbb{F}_q), \quad G = \langle P \rangle \subseteq J, \quad |G| = r, \quad (2)$$

where r is a known odd prime. The input includes $Q = [k]P$. The task is to find $k \in \mathbb{Z}_r$. The field and subgroup bit lengths are

$$n = \lceil \log_2 q \rceil, \quad m_r = \lceil \log_2 r \rceil, \quad N_{\text{in}} = 2m_r \quad \text{for the exact domain } \mathbb{Z}_r^2. \quad (3)$$

Every class has a unique reduced Mumford representation (u, v) , with u monic,

$$\deg v < \deg u \leq 2, \quad u \mid v^2 - f.$$

The identity is $(1, 0)$ [5]. For degree two, we write

$$u(x) = x^2 + u_1x + u_0, \quad v(x) = v_1x + v_0. \quad (4)$$

Thus u_1, u_0, v_1, v_0 are field elements, whereas u, v are polynomials. For several divisors we write $u_j^{(i)}, v_j^{(i)}: i$ identifies the divisor and j identifies the coefficient. The Weil bounds give $(\sqrt{q} - 1)^4 \leq |J| \leq (\sqrt{q} + 1)^4$. In particular, $m_r = 2n + O(1)$ when the cofactor $|J|/r$ is bounded. For smaller subgroups, we retain m_r as a separate size parameter.

The standard Cantor construction gives a reference addition algorithm over $\mathbb{F}_q$ [5]. We write the two reduced pairs as $(u^{(1)}, v^{(1)})$ and $(u^{(2)}, v^{(2)})$. This notation distinguishes input labels from coefficient subscripts. The algorithm computes the following monic gcd:

$$d = \gcd(u^{(1)}, u^{(2)}, v^{(1)} + v^{(2)})$$

The algorithm also computes Bezout coefficients a, b, c satisfying

$$au^{(1)} + bu^{(2)} + c(v^{(1)} + v^{(2)}) = d.$$

The composition step is

$$\tilde{u} = u^{(1)}u^{(2)}/d^2, \quad (5)$$

$$\tilde{v} = \frac{au^{(1)}v^{(2)} + bu^{(2)}v^{(1)} + c(v^{(1)}v^{(2)} + f)}{d} \pmod{\tilde{u}}. \quad (6)$$

All divisions in these formulas are exact polynomial divisions. The algorithm reduces $\tilde{v}$ modulo $\tilde{u}$. While $\deg \tilde{u} > 2$, it replaces the pair by

$$\left(\text{monic} \left(\frac{f - \tilde{v}^2}{\tilde{u}} \right), -\tilde{v} \bmod \text{monic} \left(\frac{f - \tilde{v}^2}{\tilde{u}} \right) \right).$$

The right-hand side uses the old pair. The result is the unique reduced representative of the sum. Gcd computation handles shared support, inverse pairs, and doubling. The algorithm also includes the identity $(1, 0)$. We use this algorithm as a correctness reference. The clean inversion circuit in Section 2.1 supplies the inverses needed for normalization.

The reference algorithm includes gcds, degree decisions, and normalization in $\mathbb{F}_q$. These operations do not directly give a fixed polynomial circuit over $\mathbb{Z}$ with bounded degree. This distinction matters when we compare the algorithm with the complete elliptic-curve polynomial formulas in [9, Section 2.2]. The residue interface in Section 5 requires its own treatment of all reachable cases.

Canonical Mumford encoding. We fix an injective encoding $\text{Enc} : G \rightarrow \{0, 1\}^\ell$. One explicit Mumford encoding has $\ell = 2 + 4n$. It uses a two-bit degree tag d and four n -bit slots in the following order. All coefficients have canonical representatives in $\{0, \dots, q - 1\}$.

d coefficient slots (u_1, u_0, v_1, v_0) represented pair		
2	(u_1, u_0, v_1, v_0)	$(x^2 + u_1x + u_0, v_1x + v_0)$
1	$(0, u_0, 0, v_0)$	$(x + u_0, v_0)$
0	$(0, 0, 0, 0)$	$(1, 0)$

The degree tag d determines the implicit monic coefficient. The tag 3 is unused. The same convention encodes every class of J ; Enc denotes its restriction to G .

This encoding fixes the register representation for the complete divisor arithmetic below. Weighted coordinates for the Legendre construction are specified in Section 4.

The standard compute-copy-uncompute construction [3] turns Cantor's algorithm [5] into a clean reversible reference routine. For fixed genus two, fixed coefficient arrays and degree tags bound the number of field operations independently of n . The base-field primitives above therefore give $O(n)$ qubits and $O(n^2)$ Toffoli gates. For a classically specified divisor B , a compute-swap-uncompute schedule also implements

$$|c\rangle|D\rangle|0\rangle \mapsto |c\rangle|D + [c]B\rangle|0\rangle, \quad c \in \{0, 1\}. \quad (7)$$

Appendix A records the fixed-length Cantor implementation and cleanup. The finite resource estimates for the sampler use the weighted polynomial evaluator specified below. The Legendre sampler uses weighted polynomial formulas, whose exceptional inputs are handled by the masking argument in Section 5.5.

3 Fourier Sampling with a Compressed Phase

We use the Fourier-sampling framework of Shor [38] and the output-compression approach of May and Schlieper [31] and CFS [10]. This section fixes the conventions and records the sampling and resource formulas used by the genus-two construction. We count peak logical qubits, including the two scalar registers, and allow intermediate measurements, reset, and classical feedforward. Classical records do not contribute to this width. Arithmetic subroutines must implement their stated clean coherent maps, including on inputs entangled with other registers. We account for single-qubit rotation synthesis with a separate distribution-error budget ε_{syn} . All gate bounds below are analytical bounds.

Our exact-domain analysis uses

$$F(a, b) = [a]P + [b]Q = [a + kb]P, \quad (a, b) \in \mathbb{Z}_r^2. \quad (8)$$

Here (a, b) are scalar inputs, while (u, v) below are Fourier outcome labels in $\mathbb{Z}_r^2$. In this section, these outcome labels are not the Mumford polynomials $u(x), v(x)$ of (4). We define $\omega_r = e^{2\pi i/r}$ and use the convention

$$\mathcal{F}_r^{-1}|a\rangle = \frac{1}{\sqrt{r}} \sum_{u \in \mathbb{Z}_r} \omega_r^{-au} |u\rangle.$$

Each register uses m_r qubits, with valid basis states $0, \dots, r - 1$. The transform on this subspace is extended to a unitary on the entire binary register. Preparing a uniform state on this subspace and implementing the

transform are separate resource items. Hadamards on m_r qubits alone do not prepare that state when r is not a power of two.

For $h : G \rightarrow \{0, 1\}$, we define $g(D) = (-1)^{h(D)}$. A clean evaluation of $h \circ F$ is a reversible circuit satisfying

$$|x\rangle|z\rangle|0\rangle \mapsto |x\rangle|z \oplus h(F(x))\rangle|0\rangle.$$

Initializing the target to $|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$ induces the phase $g(F(x))$. All input-dependent workspace must be erased before Fourier processing. Otherwise it changes the off-diagonal entries of the input density operator.

We identify $g([t]P)$ with $g(t)$. We define its normalized Fourier coefficient and mean by

$$\widehat{g}(u) = \frac{1}{r} \sum_{t \in \mathbb{Z}_r} g(t) \omega_r^{-ut}, \quad \mu_g = \frac{1}{r} \sum_{t \in \mathbb{Z}_r} g(t). \quad (9)$$

Parseval's identity gives $\sum_u |\widehat{g}(u)|^2 = 1$.

The following exact-domain identity specializes the standard Fourier-sampling and output-compression framework [38,31,10] to the clean sign phase used here.

Theorem 3.1 (Exact-domain phase compression). *Let r be an odd prime, let $G = \langle P \rangle$ have order r , and let $Q = [k]P$ for $k \in \mathbb{Z}_r$. Let $g : G \rightarrow \{-1, 1\}$, with $g([t]P)$ identified with $g(t)$ and with $\widehat{g}$ and μ_g as in (9). Assume that the phase $g(a + kb)$ is evaluated cleanly, with all workspace reset before the Fourier stage. Starting from*

$$|\psi_0\rangle = \frac{1}{r} \sum_{a,b \in \mathbb{Z}_r} |a, b\rangle, \quad (10)$$

apply this phase and then measure after $\mathcal{F}_r^{-1} \otimes \mathcal{F}_r^{-1}$. The resulting distribution is

$$p_g(u, v) = \begin{cases} |\widehat{g}(u)|^2, & v = ku \pmod{r}, \\ 0, & \text{otherwise.} \end{cases} \quad (11)$$

The event $u \neq 0$ has probability

$$p_{\text{use}} = 1 - \mu_g^2 = \text{Var}_{t \leftarrow \mathbb{Z}_r}(g(t)). \quad (12)$$

Every sample in this event yields $k = vu^{-1} \pmod{r}$.

Proof. Clean phase evaluation leaves the scalar registers in the pure state $r^{-1} \sum_{a,b} g(a + kb) |a, b\rangle$. Under the stated inverse-Fourier convention, the amplitude at (u, v) is

$$\begin{aligned} A(u, v) &= \frac{1}{r^2} \sum_{a,b \in \mathbb{Z}_r} g(a + kb) \omega_r^{-ua - vb} \\ &= \frac{1}{r^2} \left(\sum_{t \in \mathbb{Z}_r} g(t) \omega_r^{-ut} \right) \left(\sum_{b \in \mathbb{Z}_r} \omega_r^{(uk - v)b} \right). \end{aligned}$$

Here the substitution $t = a + kb$ is a bijection for each fixed b . Character orthogonality makes the second factor equal to r when $v = ku \pmod{r}$ and to zero otherwise. Thus $A(u, ku) = \widehat{g}(u)$, which proves (11). Parseval's identity confirms that these probabilities sum to one.

The only supported label with $u = 0$ is $(0, 0)$, and its probability is $|\widehat{g}(0)|^2 = \mu_g^2$. Hence the useful probability is $1 - \mu_g^2$. Since $g(t)^2 = 1$, its variance is $\mathbb{E}_t g(t)^2 - (\mathbb{E}_t g(t))^2 = 1 - \mu_g^2$, proving (12). Finally, every nonzero $u \in \mathbb{Z}_r$ is invertible because r is prime, and the support relation $v = ku$ gives $k = vu^{-1} \pmod{r}$. $\square$

Corollary 3.2 (Variance criterion). *Under the assumptions of Theorem 3.1, suppose each sign occurs on at least an η fraction of G , where $0 < \eta \leq 1/2$. Then $p_{\text{use}} \geq 4\eta(1 - \eta)$, and independent executions have expected waiting time at most $1/(4\eta(1 - \eta))$.*

Proof. The sign-frequency assumption gives $|\mu_g| \leq 1 - 2\eta$. Equation (12) yields the probability bound, and the waiting-time bound follows from independent repetition. $\square$

A nonconstant phase need not be efficient. If its minority sign occurs once, the useful-sample probability is $p_{\text{use}} = 4(r-1)/r^2$. This probability is exponentially small in m_r . For $g_T(t) = g(t+T)$, the Fourier coefficients satisfy $\hat{g}_T(u) = \omega_r^{uT} \hat{g}(u)$. Translation leaves both the distribution and the variance unchanged.

The low-space construction in Section 4 uses the field-shifted Legendre phase g_γ of (16). Its useful-sample probability is established using shifted-character correlations and coordinate-fiber bounds.

The preparation and terminal Fourier readout below instantiate Kitaev's eigenvalue-measurement method [26] and the semiclassical feedback of Griffiths and Niu [20], using Cuccaro ripple-carry arithmetic [14]. The explicit constants refer to the modular-shift schedule in Appendix D.1.

Theorem 3.3 (Space and error of exact-domain sampling). *Let $r \geq 3$ be an integer, write $m_r = \lceil \log_2 r \rceil$, and let $0 < \varepsilon_F < 1$ and $J_{\text{prep}} \geq 1$, with J_{prep} an integer. Set*

$$t = m_r + \lceil \log_2(4 + 2/\varepsilon_F) \rceil. \quad (13)$$

Allow intermediate measurements, reset, and classical feedforward. The explicit schedule in Appendix D.1 implements uniform preparation on $\mathbb{Z}_r^2$ and terminal Fourier readout with the following bounds. These bounds exclude phase-oracle resources and rotation-synthesis costs.

The preparation and readout stages use at most $3m_r + 5$ peak logical qubits, including both m_r -qubit scalar registers and all quantum workspace. Classical records do not contribute to this width. With at most J_{prep} trials per register, successful preparation produces exactly the state in (10); its abort probability is at most $2^{1-J_{\text{prep}}}$. Preparation uses fewer than $16m_r - 8$ Toffolis in expectation and at most $4J_{\text{prep}}(2m_r - 1)$ under the trial cap.

Fourier readout uses at most $10(2m_r - 1)t$ Toffolis and at most $t(t-1)$ individual classically controlled dyadic feedback rotations across both registers. This count includes all semiclassical feedback. Combining the angles within each round reduces the number of single-qubit rotations to at most $2(t-1)$. For every joint state supported on $\mathbb{Z}_r^2$, its decoded joint labels have total variation distance at most ε_F from ideal measurement after $(\mathcal{F}_r^{-1})^{\otimes 2}$, before rotation-synthesis errors. Consequently, conditional on successful preparation and clean phase evaluation, the sampler approximates (11) within this error budget.

Preparation measures only the range predicate $x < r$. Acceptance therefore leaves the exact uniform state, with probability $r/2^{m_r} > 1/2$ per trial. Readout estimates the eigenphase of the cyclic shift with one recycled control and classical feedback, then rounds it to a Fourier label [26,20]. The chosen t bounds each incorrect-label probability by $\varepsilon_F/2$, giving the two-register error bound. Cuccaro-based modular shifts supply the stated gate count [14]; the allocation and error estimate are recorded in Appendix D.1. Rotation synthesis adds its separately budgeted distribution error ε_{syn} .

4 Output Compression Using the Legendre Symbol

The shifted Legendre phase is evaluated in Lange's weighted Mumford coordinates [28], with the second scaling parameter fixed to one. The nonleading coefficients of u have weight two, and the coefficients of v have weight three. With a scale coordinate Z of weight one, we use the degree-two chart $Z \neq 0$ in $\mathbb{P}(2, 2, 3, 3, 1)$, with coordinates $[U_1 : U_0 : V_1 : V_0 : Z]$. The affine decoding is

$$u_1 = \frac{U_1}{Z^2}, \quad u_0 = \frac{U_0}{Z^2}, \quad v_1 = \frac{V_1}{Z^3}, \quad v_0 = \frac{V_0}{Z^3}. \quad (14)$$

Thus the equivalence relation is

$$(U_1, U_0, V_1, V_0, Z) \sim (\lambda^2 U_1, \lambda^2 U_0, \lambda^3 V_1, \lambda^3 V_0, \lambda Z), \quad \lambda \in \mathbb{F}_q^\times. \quad (15)$$

Each affine degree-two pair has the representative $[u_1 : u_0 : v_1 : v_0 : 1]$. Conversely, rescaling a point by Z^{-1} gives this unique representative with last coordinate one. Thus (14) identifies the valid part of this chart with the degree-two Mumford chart. This representation permits evaluation of the shifted quadratic character without division by Z .

4.1 Field-shifted Legendre phases

The symbol χ_q denotes the quadratic character of $\mathbb{F}_q$, with $\chi_q(0) = 0$. We sample a classical seed $\gamma \in \mathbb{F}_q$ uniformly and define the phase by

$$g_\gamma(D) = \begin{cases} -1, & \deg u = 2 \text{ and } \chi_q(u_0 + \gamma) = -1, \\ +1, & \text{otherwise.} \end{cases} \quad (16)$$

On the weighted chart (14), the character satisfies

$$\chi_q(u_0 + \gamma) = \chi_q(U_0 + \gamma Z^2). \quad (17)$$

The identity $U_0 + \gamma Z^2 = (u_0 + \gamma)Z^2$ explains this equality. The rescaling in (15) multiplies its argument by λ^2 . The classical seed γ is fixed under projective rescaling. Thus the shift requires no division by Z . It is a shift of a field coefficient, not a translation of the group element. The convention at zero makes g_γ a binary phase on every class.

The field seed γ is distinct from the subgroup order r . The success estimate below follows from the shifted-character correlation and the sizes of the u_0 fibers.

Example 4.1 (Why the unshifted phase can fail). We take a prime $q \neq 2, 5$. We define the curve $C : y^2 = x^5 + 1$ and the point $A = (0, 1)$. The associated class is $P = [A - \infty]$. Then P has order five, but g_0 from (16) is identically $+1$ on $\langle P \rangle$. More generally, a square or zero shift γ gives useful-sample probability zero, whereas a nonsquare shift gives probability $24/25$.

Proof. The identity $(y - 1)(y + 1) = x^5$ gives $\text{div}(y - 1) = 5A - 5\infty$. Thus $[5]P = 0$. The class P is nonzero. Otherwise, a function with divisor $A - \infty$ would define a degree-one map from a genus-two curve to $\mathbb{P}^1$. The reduced pairs are

$$0 = (1, 0), \quad \pm P = (x, \pm 1), \quad \pm[2]P = (x^2, \pm 1).$$

The first three classes have degree below two. The other two have $u_0 = 0$. Hence all five phases equal $+1$ at $\gamma = 0$. For a general shift, the first three phases remain $+1$. The last two are $+1$ for square or zero γ , and -1 for nonsquare γ . Thus the mean phase is respectively 1 or $1/5$, and Theorem 3.1 gives the stated useful probabilities. $\square$

Scale invariance alone therefore does not ensure positive sampling variance. Uniform field shifts give an average guarantee while allowing zero-variance realizations. This order-five example illustrates failure of a fixed unshifted phase; the regime $r = \Theta(q^2)$ considered below concerns families of much larger subgroups.

We define the completed binary character by

$$s_q(x) = \chi_q(x) + \mathbf{1}_{x=0}. \quad (18)$$

Thus χ_q takes values in $\{-1, 0, 1\}$, whereas s_q takes values in $\{-1, 1\}$. In particular, $s_q(0) = 1$. We have $s_q(x) = -1$ precisely for nonsquares and $s_q(x) = +1$ otherwise. For uniform $\gamma \in \mathbb{F}_q$, its mean satisfies

$$\mathbb{E}_\gamma s_q(a + \gamma) = \frac{1}{q}.$$

For distinct $a, b \in \mathbb{F}_q$, its correlation satisfies

$$\mathbb{E}_\gamma [s_q(a + \gamma)s_q(b + \gamma)] = \frac{-1 + \chi_q(a - b) + \chi_q(b - a)}{q} \leq \frac{1}{q}. \quad (19)$$

For $a = b$ the correlation equals one.

The mean follows from $\sum_x \chi_q(x) = 0$. For $a \neq b$, the standard quadratic character sum [29, Chapter 5] is $\sum_\gamma \chi_q((\gamma + a)(\gamma + b)) = -1$. An elementary proof uses a change of variables to express the quadratic as a nonzero multiple of $t^2 - 1$. The proof then counts solutions of $z^2 = t^2 - 1$. The factorization $(t - z)(t + z) = 1$ gives $q - 1$ solutions because q is odd. The character sum gives the same count as $q + \sum_t \chi_q(t^2 - 1)$. The two zero-indicator terms contribute $\chi_q(a - b) + \chi_q(b - a)$. Their sum is at most two, so the inequality follows.

For any subgroup $G \subseteq J$, the symbol e denotes the number of its classes with $\deg u < 2$. We also define

$$m_a = \#\{D \in G : \deg u = 2, u_0 = a\}.$$

The reduced Mumford representation [5] gives the elementary bounds

$$\sum_a m_a = r - e, \quad e \leq 2q + 1, \quad m_a \leq 4q \quad (a \in \mathbb{F}_q). \quad (20)$$

The lower-degree classes consist of the identity and at most two affine points for each $x \in \mathbb{F}_q$. Thus $e \leq 2q + 1$. For fixed $u_0 = a$ there are q possible monic polynomials $u = x^2 + u_1x + a$. For a square-free u , the algebra $\mathbb{F}_q[x]/(u)$ is either a quadratic field or a product of two copies of $\mathbb{F}_q$. The equation $v^2 = f \pmod{u}$ has at most four solutions in either case. For $u = (x - t)^2$, a solution satisfies $v(t)^2 = f(t)$ and $2v(t)v'(t) = f'(t)$. If $f(t) = 0$, no solution exists because square-freeness gives $f'(t) \neq 0$. Otherwise, $v(t)$ has at most two choices. Each choice determines $v'(t)$ uniquely. Hence at most four reduced pairs occur for each u , and $m_a \leq 4q$ follows.

Theorem 4.2 (Shift-averaged useful probability). *We assume that G has odd prime order r . The exact-domain sampling algorithm uses (16) with a fresh uniform γ in each run. For the counts e, m_a in (20), the mean useful-sample probability satisfies*

$$\mathbb{E}_\gamma[1 - \mu_{g_\gamma}^2] \geq \left(1 - \frac{1}{q}\right) \left(1 - \frac{e^2 + \sum_a m_a^2}{r^2}\right) =: V_G, \quad (21)$$

$$\mathbb{E}_\gamma[1 - \mu_{g_\gamma}^2] \geq \max \left\{ 0, \left(1 - \frac{1}{q}\right) \left(1 - \frac{e^2 + 4q(r - e)}{r^2}\right) \right\}. \quad (22)$$

In particular, if $r = \Theta(q^2)$, then the mean useful-sample probability is $1 - O(q^{-1})$. Under this condition, the expected number of independent runs is $1 + O(q^{-1})$.

Proof. The phase mean is

$$\mu_{g_\gamma} = \frac{e + \sum_a m_a s_q(a + \gamma)}{r}.$$

We expand the square and use $\sum_a m_a = r - e$. Equation (19) then gives

$$\begin{aligned} \mathbb{E}_\gamma \mu_{g_\gamma}^2 &\leq \frac{e^2 + \sum_a m_a^2}{r^2} + \frac{r^2 - e^2 - \sum_a m_a^2}{qr^2} \\ &= \frac{1}{q} + \left(1 - \frac{1}{q}\right) \frac{e^2 + \sum_a m_a^2}{r^2}. \end{aligned}$$

Theorem 3.1 identifies the useful probability with the variance in (12). The bound $\sum_a m_a^2 \leq (\max_a m_a)(r - e) \leq 4q(r - e)$ then gives the second inequality. For $r = \Theta(q^2)$, $e = O(q)$ and $\sum_a m_a^2 = O(qr)$ give the stated asymptotic probability. Independent resampling gives a geometric waiting time. $\square$

Example 4.1 attains the bound (21). Indeed, $e = 3$, $m_0 = 2$, and all other m_a vanish. There are $(q + 1)/2$ square or zero shifts and $(q - 1)/2$ nonsquare shifts, so

$$\mathbb{E}_\gamma[1 - \mu_{g_\gamma}^2] = \frac{q - 1}{2q} \frac{24}{25} = \frac{12(q - 1)}{25q} = V_G.$$

For $q = 13$, the average is $144/325 \approx 0.4431$: seven shifts give probability zero and six give probability $24/25$. This equality illustrates the distinction between a shift-averaged bound and a guarantee for a specified shift.

When $q \equiv 3 \pmod{4}$, the same calculation gives the exact shift-averaged useful-sample probability

$$\mathbb{E}_\gamma[1 - \mu_{g_\gamma}^2] = 1 - \frac{\left(e + \frac{r - e}{q}\right)^2 + (1 + 1/q) \sum_a \left(m_a - \frac{r - e}{q}\right)^2}{r^2}. \quad (23)$$

Here $\chi_q(a-b) + \chi_q(b-a) = 0$ for $a \neq b$. Equation (19) therefore gives the off-diagonal correlation $-1/q$. The squared mean therefore satisfies

$$r^2 \mathbb{E}_\gamma[\mu_{g_\gamma}^2] = e^2 + \frac{2e(r-e)}{q} + \left(1 + \frac{1}{q}\right) \sum_a m_a^2 - \frac{(r-e)^2}{q}.$$

The identity $\sum_a (m_a - (r-e)/q)^2 = \sum_a m_a^2 - (r-e)^2/q$ and (12) give the result. This identity applies to the field $q = 2^{127} - 1$ used below. It separates mean phase bias from variation in the fiber sizes. Its numerical evaluation requires the actual fiber counts. The uniform bound (22) requires no such enumeration.

The precise bound V_G is nonnegative because $e^2 + \sum_a m_a^2 \leq r^2$. It is a lower bound on the mean success, not on every fixed shift. The following bound gives a certificate without subgroup enumeration:

$$\mathbb{E}_\gamma[1 - \mu_{g_\gamma}^2] \geq \max \left\{ 0, \left(1 - \frac{1}{q}\right) \left(1 - \frac{(2q+1)^2}{r^2} - \frac{4q}{r}\right) \right\}.$$

For the subgroup in Example 4.1, any nonsquare γ changes the two degree-two phases to -1 and gives success $24/25$. A nonsquare shift need not be favorable on an arbitrary subgroup.

An exact character implementation may compute an integer X_γ satisfying

$$X_\gamma \bmod q = \begin{cases} U_0 + \gamma Z^2, & \deg u = 2 \text{ in a valid chart,} \\ 0, & \deg u < 2. \end{cases} \quad (24)$$

The seed is classical and its canonical integer lift has at most n bits. The arithmetic interface must include exceptional-case masking and this terminal polynomial. A variance theorem alone does not specify that program. The masked construction in Section 5 uses a different total polynomial whose disagreement with the ideal phase is bounded explicitly.

4.2 A phase-only terminal node

Only the shifted Legendre phase uses the terminal node considered here. The node therefore need not produce the five coordinates of the sum. This observation gives a fully specified optimization at the terminal node. Internal nodes still require all five output coordinates. We retain the assumption of a monic degree-five model with vanishing x^4 coefficient from Section 2. The two input tuples are $\mathbf{U}^{(i)} = (U_1^{(i)}, U_0^{(i)}, V_1^{(i)}, V_0^{(i)}, Z_i)$, $i = 1, 2$. For $j = 0, 1$, we define the scaled differences and sum by

$$\begin{aligned} \Delta U_j &= U_j^{(2)} Z_1^2 - U_j^{(1)} Z_2^2, \\ \Delta V_j &= V_j^{(2)} Z_1^3 - V_j^{(1)} Z_2^3, \\ \Sigma U_1 &= U_1^{(2)} Z_1^2 + U_1^{(1)} Z_2^2. \end{aligned} \quad (25)$$

The symbols Δ and Σ here denote these two-input differences and sums, not differentiation or an unspecified summation. For example, when $Z_1 = Z_2 = 1$, they reduce to $\Delta U_j = u_j^{(2)} - u_j^{(1)}$ and $\Sigma U_1 = u_1^{(2)} + u_1^{(1)}$. In the arithmetic listings, $\mathbf{dU}$, $\mathbf{dV}$, and $\mathbf{sU}$ are the corresponding code names. We also define

$$\begin{aligned} \mathcal{C}_* &= U_1^{(1)} U_0^{(2)} - U_0^{(1)} U_1^{(2)}, \\ \mathcal{R}_* &= (\Delta U_0)^2 - \mathcal{C}_* \Delta U_1, \\ \mathcal{E}_* &= \Delta V_1 \Delta U_0 - \Delta V_0 \Delta U_1, \\ \mathcal{L}_* &= \Delta U_0 \Delta V_0 - \mathcal{C}_* \Delta V_1. \end{aligned} \quad (26)$$

We set $\mathcal{K} = Z_1 Z_2$ and $\mathbf{A}_{\text{sc}} = \mathcal{K}^2$. The subscript “sc” identifies this scalar scale factor. It distinguishes the factor from the addition map $\mathcal{A}$ in (36). We define the terminal polynomials by

$$\begin{aligned} N_* &= \mathbf{A}_{\text{sc}} \mathcal{L}_*^2 + 2U_1^{(1)} Z_2^2 \mathcal{L}_* \mathcal{E}_* + (U_1^{(1)} U_1^{(2)} - \Delta U_0) \mathcal{E}_*^2 \\ &\quad - 2V_1^{(1)} Z_2^3 \mathcal{R}_* \mathcal{E}_* + \Sigma U_1 \mathbf{A}_{\text{sc}} \mathcal{R}_*^2, \\ D_* &= \mathbf{A}_{\text{sc}} \mathcal{E}_*^2, \quad \Psi_\gamma = N_* + \gamma D_*. \end{aligned} \quad (27)$$

Here N_* and D_* are the numerator and denominator for the output coefficient $u_0^{(3)}$ on the generic branch. The scalar polynomial Ψ_γ is the shifted phase argument. The group-valued map F in (8) has a different role. Every expression is an integer polynomial. The terminal program uses no division. The generic interpretation requires the nonvanishing condition in Proposition 4.3.

Proposition 4.3 (Square-factor elimination at a generic terminal node). *We assume that the degree-two inputs are valid and satisfy $Z_1 Z_2 \mathcal{R}_* \mathcal{E}_* \neq 0$. The symbol $u_0^{(3)}$ denotes the constant coefficient of the monic quadratic representing their sum. The following identities hold:*

$$u_0^{(3)} = N_*/D_*, \quad s_q(\Psi_\gamma) = s_q(u_0^{(3)} + \gamma) \quad (\gamma \in \mathbb{F}_q). \quad (28)$$

The identity includes $u_0^{(3)} + \gamma = 0$. In particular, the phase is unchanged by independently rescaling either weighted input.

Proof. The following interpolation proof also establishes the compact addition map (36). We first take normalized inputs

$$(u^{(1)}, v^{(1)}) = (x^2 + ax + c, sx + u), \quad (u^{(2)}, v^{(2)}) = (x^2 + bx + d, tx + v).$$

The letters in this display denote field elements. We define

$$\begin{aligned} R &= (d - c)^2 - (ad - cb)(b - a), \\ E &= (t - s)(d - c) - (v - u)(b - a), \\ L &= (d - c)(v - u) - (ad - cb)(t - s), \\ B_1 &= sR - aL - (ab + c)E, \quad B_0 = uR - cL - bcE. \end{aligned}$$

Here $R = \text{Res}_x(u^{(1)}, u^{(2)})$. When $R \neq 0$, the two quadratics are coprime. Thus a unique polynomial of degree at most three has the prescribed remainders $v^{(i)}$ modulo $u^{(i)}$. It is

$$\ell(x) = \frac{-Ex^3 - [L + (a + b)E]x^2 + B_1x + B_0}{R}.$$

The numerator reduces to $Rv^{(i)}$ modulo each input quadratic. This calculation verifies interpolation without choosing roots or extending the base field. The additional assumption $E \neq 0$ means that the interpolant has degree three.

Since the inputs are valid, $u^{(1)}u^{(2)}$ divides $\ell^2 - f$. Its quotient has degree two and leading coefficient E^2/R^2 . We write the associated monic quadratic as $x^2 + \alpha x + \beta$. The leading coefficient of f is one and its x^4 coefficient is zero. We compare the coefficients of x^5 and x^4 in the following identity:

$$R^2(\ell^2 - f) = E^2 u^{(1)} u^{(2)} (x^2 + \alpha x + \beta)$$

The comparison gives

$$\begin{aligned} A &= 2LE + (a + b)E^2 - R^2, \quad T = LE - R^2, \\ N &= L^2 + 2aLE + (ab - d + c)E^2 - 2sRE + (a + b)R^2, \\ \alpha &= A/E^2, \quad \beta = N/E^2. \end{aligned}$$

The formulas in (27) and (35) reduce to these polynomials for normalized inputs. Cantor reduction gives $v^{(3)} = -\ell \bmod (x^2 + \alpha x + \beta)$. The identities $x^2 \equiv -\alpha x - \beta$ and $x^3 \equiv (\alpha^2 - \beta)x + \alpha\beta$ give

$$v_1^{(3)} = \frac{AT - E^2(B_1E + N)}{RE^3}, \quad v_0^{(3)} = \frac{NT - B_0E^3}{RE^3}.$$

We choose the output scale RE and clear the decoding denominators. These steps give the normalized compact five-coordinate output. The construction therefore performs the usual composition and reduction on this generic branch. It also applies when the input supports do not split over the base field.

For arbitrary nonzero input scales, we apply the weighted rescalings Z_1^{-1} and Z_2^{-1} . These rescalings give the normalized inputs above. Every monomial of N_* and D_* has weighted bidegree $(12, 12)$, so their quotient is unchanged. The five compact output polynomials have bidegrees $(20, 20)$, $(20, 20)$, $(30, 30)$, $(30, 30)$, $(10, 10)$, respectively. Their common output rescaling therefore cancels under the decoding rule (14). This cancellation proves both the compact addition formula and $u_0^{(3)} = N_*/D_*$ for the original weighted inputs. The rescaling uses only the assumed nonzero scales. It requires no additional generic condition.

Finally, $D_* = (Z_1 Z_2 \mathcal{E}_*)^2$ is a nonzero square. Hence $\Psi_\gamma = D_*(u_0^{(3)} + \gamma)$ has the same completed quadratic character as $u_0^{(3)} + \gamma$. If this last argument is zero, both completed characters equal one. The phase identity thus includes the zero case and is invariant under independent weighted rescaling of either input. $\square$

For a nonzero polynomial $P(z) = \sum_\alpha c_\alpha z^\alpha$, we define

$$\deg P = \max_{c_\alpha \neq 0} |\alpha|, \quad \|P\|_1 = \sum_\alpha |c_\alpha|. \quad (29)$$

Here the multi-index α runs over the exponents of the input coordinate variables. The quantity $|\alpha|$ is their sum. Thus $\|P\|_1$ is a coefficient norm, not a norm of a state or a divisor. Classical parameters such as γ are fixed coefficients unless stated otherwise. The unexpanded program gives the ordinary degree bounds $\deg N_* \leq 19$, $\deg D_* \leq 18$. It also gives the coefficient-norm bounds $\|N_*\|_1 \leq 640$, $\|D_*\|_1 \leq 64$. These bounds follow by propagating the degree and coefficient norm through the arithmetic program. Exact sparse expansion gives $\|N_*\|_1 = 232$ and $\deg N_* = 19$. The supplied symbolic certificate contains 115 numerator monomials and 34 denominator monomials. It verifies (66) as a polynomial identity over $\mathbb{Z}$. This verification is stronger than a check at finitely many field points. Consequently a classical integer lift of γ gives

$$\deg \Psi_\gamma \leq 19, \quad \|\Psi_\gamma\|_1 \leq 232 + 64|\gamma|. \quad (30)$$

Alternatively, we can treat γ as one additional input variable. The resulting polynomial has degree 19 and coefficient norm 296. The terminal polynomial is therefore compatible with integer reduction at auxiliary moduli. This compatibility does not extend to the generic-branch test. The terminal polynomial also does not supply formulas for nongeneric inputs.

To turn the reconstructed polynomial value into a phase, we use the binary Jacobi algorithm of Shallit and Sorenson [37], as used for quantum character evaluation by CFS [10, Sections 4.2–4.3]. We fix a worst-case schedule of $2n + 1$ rounds and use the completion $s_q(0) = 1$ from (18). The character argument Y serves as one working operand. Recording the branches, applying the sign phase, and reversing the computation implements

$$|Y\rangle|0\rangle \mapsto s_q(Y)|Y\rangle|0\rangle \quad (0 \leq Y < q). \quad (31)$$

The input and all workspace are restored. With the ripple and dirty-ancilla decompositions specified in Appendix D.1, the total logical width, including the n -qubit input, satisfies

$$W_\chi(n) \leq 6n + 12. \quad (32)$$

One forward macro costs at most $36n$ Toffolis. Including all $2n + 1$ forward macros and their complete reversal gives

$$C_\chi(n) \leq 72n(2n + 1). \quad (33)$$

A clean character phase uses this bound once. At $n = 127$, it is 2,331,720 Toffolis. These resource allowances are used in the sampler's stage allocation; the fixed schedule, zero-input case, and cleanup are detailed in the appendix.

5 Addition of Divisor Classes Using Residues in a Binary Tree

We follow CFS [9, Section 5] and compute the final character argument in two steps. We first evaluate the tree modulo small auxiliary primes. We then reconstruct the result with a guarded CRT calculation. Our construction uses a compact weighted genus-two addition node. Public random masks justify its generic

formulas on the exact uniform sampling domain. This section gives a sampler with bounded error. Appendix C separately specifies the stronger exact residue interface for arbitrary inputs.

The input size is $N_{\text{in}} = 2m_r$, as in (3). We set $s_w = \lceil m_r/w \rceil$ and write $a = \sum_{j=0}^{s_w-1} a^{(j)} 2^{jw}$ and $b = \sum_{j=0}^{s_w-1} b^{(j)} 2^{jw}$. The window values are $a^{(j)} = \lfloor a/2^{jw} \rfloor \bmod 2^w$ and similarly for $b^{(j)}$. The highest window may have fewer than w bits. We place the P -windows before the Q -windows. We pad the $L_0 = 2s_w$ summands to $L = 2^{\lceil \log_2 L_0 \rceil} = 2^h$ leaves. For $x = (a, b)$, we define the group-valued leaves by

$$A_i(x) = \begin{cases} [a^{(i)} 2^{i w}] P, & 0 \leq i < s_w, \\ [b^{(i-s_w)} 2^{(i-s_w)w}] Q, & s_w \leq i < 2s_w, \\ 0_G, & 2s_w \leq i < L. \end{cases} \quad (34)$$

The symbol 0_G denotes the identity of G . The leaf sum satisfies $\sum_i A_i(x) = [a]P + [b]Q = F(x)$. One window selects each nonpadding leaf from a public table. The padded leaves are fixed. The total table size is at most $L2^w$. Zero windows and padded leaves also receive masks. A classical computation prepares all tables and masks. Both m_r -bit scalar registers remain live throughout the quantum arithmetic.

5.1 A lower-degree weighted addition map

We use the recovery rule (14) to reduce the degree of the denominator-cleared internal node. We set $\mathcal{K} = Z_1 Z_2$ and retain $\mathcal{R}_*, \mathcal{E}_*, \mathcal{L}_*$ from (26) and N_* from (27). In particular, $\Sigma U_1 = U_1^{(2)} Z_1^2 + U_1^{(1)} Z_2^2$ is the scaled sum of the two linear coefficients, as defined in (25). We define the following local polynomial quantities:

$$\begin{aligned} \mathcal{J}_* &= \mathcal{K}^2 \mathcal{L}_* + \Sigma U_1 \mathcal{E}_*, \\ A_* &= 2\mathcal{K}^2 \mathcal{L}_* \mathcal{E}_* + \Sigma U_1 \mathcal{E}_*^2 - \mathcal{K}^4 \mathcal{R}_*^2, \\ T_* &= \mathcal{L}_* \mathcal{E}_* - \mathcal{K}^2 \mathcal{R}_*^2, \\ B_1 &= V_1^{(1)} Z_2^3 \mathcal{R}_* - U_1^{(1)} Z_2^2 \mathcal{L}_* - (U_1^{(1)} U_1^{(2)} + U_0^{(1)} Z_2^2) \mathcal{E}_*, \\ B_0 &= V_0^{(1)} Z_2^3 \mathcal{R}_* - U_0^{(1)} Z_2^2 \mathcal{L}_* - U_0^{(1)} U_1^{(2)} \mathcal{E}_*. \end{aligned} \quad (35)$$

The polynomial $\mathcal{J}_*$ is used in the interpolation. The group $J = \text{Jac}(C)(\mathbb{F}_q)$ is the Jacobian. The symbol N_* denotes a polynomial numerator. The symbol N_{in} denotes the number of scalar input qubits. The compact polynomial map $\mathcal{A} : \mathbb{Z}^5 \times \mathbb{Z}^5 \rightarrow \mathbb{Z}^5$ has output

$$\begin{aligned} U'_1 &= A_* \mathcal{R}_*^2, & U'_0 &= N_* \mathcal{R}_*^2, \\ V'_1 &= \mathcal{R}_*^2 (A_* T_* - \mathcal{E}_*^2 (B_1 \mathcal{E}_* + N_*)), \\ V'_0 &= \mathcal{R}_*^2 (N_* T_* - B_0 \mathcal{E}_*^3), & Z' &= \mathcal{K} \mathcal{R}_* \mathcal{E}_*. \end{aligned} \quad (36)$$

Primes on coordinates label the output of one addition. They do not denote derivatives. The evaluator computes all five expressions over $\mathbb{Z}$, or reduces them modulo the active modulus. These expressions represent group addition over $\mathbb{F}_q$ on the generic domain $\mathcal{K} \mathcal{R}_* \mathcal{E}_* \neq 0$.

The proof of Proposition 4.3 shows that this map represents group addition on its generic domain. The proof uses interpolation, composition, coefficient comparison, and one Cantor reduction.

Choice of arithmetic formula. Earlier papers report classical field-operation counts under specified coordinate and genericity conditions. Lange's extended weighted addition uses 47 field multiplications and 7 squarings [28, Section 3.1]. Hisil and Costello's plain projective addition uses 41 multiplications, 7 squarings, and 22 additions [23, Table 2]. These counts describe forward field evaluation. They do not include reversible cleanup or CRT reconstruction. Ozbay Gurler and Hisil give a complete inversion-free algorithm [32, Sections 3–5]. Their algorithm distinguishes input degrees, equal input polynomials, common support, and a degree-one output. Their cost analysis also includes the field operations used to select a case. Their Jacobian coordinates have coefficient weights (2, 4, 3, 5) and two scales. Our five-coordinate chart instead specializes Lange's weights (2, 2, 3, 3).

We choose (36) because the same integer polynomial map defines every nonterminal addition node at every auxiliary modulus. Its degree, coefficient norms, and height recurrence give the required lower bound

on the product of the CRT moduli. Proposition 5.1 certifies its evaluation and cleanup with eight local words. The terminal polynomial gives the required phase argument without a field inversion. A reversible implementation of the cited complete algorithm would also need compatible branch selection, integer-height bounds, and a clean register schedule. Independent case selection modulo the auxiliary primes would not necessarily evaluate one common integer expression. Our masks give a proved sampling guarantee for the chosen generic formula. The resource bounds below apply to this construction.

5.2 Weighted height and growth along the tree

The ordinary total degrees of the five outputs in (36) are bounded by (32, 31, 48, 47, 15). Their weighted bihomogeneity gives a more useful bound for repeated additions. Under the separate input rescalings with weights (2, 2, 3, 3, 1), their bidegrees are

$$(20, 20), (20, 20), (30, 30), (30, 30), (10, 10).$$

Thus the output scale changes by $\lambda_1^{10}\lambda_2^{10}$, as required by the same recovery weights. A direct coefficient-norm certificate gives the componentwise upper bound

$$(\|U'_1\|_1, \|U'_0\|_1, \|V'_1\|_1, \|V'_0\|_1, \|Z'\|_1) \leq (20480, 40960, 6291456, 6029312, 64). \quad (37)$$

For an integer tuple $\mathbf{U} = (U_1, U_0, V_1, V_0, Z)$, we define

$$\mathcal{H}(\mathbf{U}) = \max\{1, |U_1|^{1/2}, |U_0|^{1/2}, |V_1|^{1/3}, |V_0|^{1/3}, |Z|\}. \quad (38)$$

This function measures the size of the chosen integer tuple. It is not a projectively invariant height of a divisor class. If $H_U = \mathcal{H}(\mathbf{U})$, then $|U_j| \leq H_U^2$, $|V_j| \leq H_U^3$, and $|Z| \leq H_U$. The symbol $\mathcal{H}$ denotes this weighted height throughout the tree argument. The symbol H in (41) is an integer bit bound. Bihomogeneity and (37) give the following bound. The bound also holds for exceptional tuples that do not represent group elements:

$$\mathcal{H}(\mathcal{A}(\mathbf{U}, \mathbf{V})) \leq 2^8 \mathcal{H}(\mathbf{U})^{10} \mathcal{H}(\mathbf{V})^{10}. \quad (39)$$

We justify the constant and exponents as follows. We write $H_U = \mathcal{H}(\mathbf{U})$ and $H_V = \mathcal{H}(\mathbf{V})$ for the two child tuples. An output coordinate of weight $\omega \in \{2, 3, 1\}$ has weighted bidegree $(10\omega, 10\omega)$. If its coefficient norm is c_ω , its absolute value is at most $c_\omega H_U^{10\omega} H_V^{10\omega}$. The ω -th root of this bound is $c_\omega^{1/\omega} H_U^{10} H_V^{10}$. The five constants from (37) satisfy

$$\max\{20480^{1/2}, 40960^{1/2}, 6291456^{1/3}, 6029312^{1/3}, 64\} < 2^8.$$

The maximum of the five coordinate bounds and 1 proves (39). Thus the exponents 10 apply to the weighted heights of the two children. The balanced-tree growth factor 20 is their sum after taking logarithms.

Normalized leaf tuples have $Z = 1$ and four coordinates in $[0, q)$. Every fixed dummy leaf must satisfy the same conditions. For $q < 2^n$, these tuples have weighted log-height at most $n/2$. A leaf has tree height $j = 0$. Its parent has height $j = 1$. We define a bound t_j for twice the logarithmic weighted height by

$$t_0 = n, \quad t_j = 20t_{j-1} + 16 = n20^j + 16 \frac{20^j - 1}{19}. \quad (40)$$

If both children have $\log_2 \mathcal{H} \leq t_{j-1}/2$, then (39) gives $\log_2 \mathcal{H}_{\text{parent}} \leq 8 + 10t_{j-1} = t_j/2$. This inequality proves the bound by induction. The terminal polynomial $\Psi_\gamma = N_* + \gamma D_*$ has weighted bidegree (12, 12). Its coefficient norm is at most $232 + 64|\gamma| < 2^{n+9}$ for the canonical seed $0 \leq \gamma < q$. Thus a tree with 2^h leaves, with the direct phase polynomial at the final node, has the certified signed-integer bound

$$|\Psi_\gamma| < 2^H, \quad H = n + 10 + 12t_{h-1} \quad (h \geq 1). \quad (41)$$

In particular $H = O(nL^{\log_2 20})$ for $L = 2^h$. The index $h - 1$ occurs because the terminal polynomial receives the two subtrees immediately below the root. Its bidegree (12, 12) gives $\log_2 |\Psi_\gamma| < n + 9 + 12t_{h-1}$ whenever the output is nonzero. The extra bit in H is a conservative reserve. The smaller terminal polynomial does not change the internal-node exponent $\log_2 20$.

For finite allocations, the normalized leaf scale gives a sharper bound for each coordinate. For a nonzero integer polynomial P in ten variables, we define

$$B_P(\mathbf{v}) = \lceil \log_2 \|P\|_1 \rceil + \max_{\alpha \in \text{supp } P} \alpha \cdot (\mathbf{v}, \mathbf{v}).$$

We collect equal monomials before computing the norm and support. These quantities depend on the integer polynomial. They do not depend on its evaluation schedule. We start with $\mathbf{v}_0 = (n, n, n, n, 0)$ and set $\mathbf{v}_j = (B_{A_i}(\mathbf{v}_{j-1}))_{i=1}^5$. The triangle inequality gives $|U_i| \leq 2^{v_{j,i}}$ at height j , including exceptional tuples. Thus a valid replacement in the finite CRT allocation is

$$H_{\text{cw}} = 1 + \max\{B_{N_*}(\mathbf{v}_{h-1}), n + B_{D_*}(\mathbf{v}_{h-1})\}, \quad |\Psi_\gamma| < 2^{H_{\text{cw}}}. \quad (42)$$

The strict inequality follows from $0 \leq \gamma < 2^n$. We write $a = B_{N_*}(\mathbf{v}_{h-1})$ and $d = B_{D_*}(\mathbf{v}_{h-1})$. The triangle inequality then gives $|\Psi_\gamma| \leq 2^a + \gamma 2^d < 2^a + 2^{n+d} \leq 2^{1+\max\{a, n+d\}}$. The repeated vector $(\mathbf{v}, \mathbf{v})$ bounds the two child tuples separately. Its entries and H_{cw} bound integer heights. They do not specify quantum register allocations. The initial zero in $\mathbf{v}_0$ records the normalized leaf scale $Z = 1$. Appendix D.3 supplies the exact coefficient and support certificate. This certificate improves the finite allocations. The general tree-growth exponent remains $\log_2 20$.

5.3 An explicit low-workspace local evaluator

For an active auxiliary modulus p , a residue word uses $b_p = \lceil \log_2 p \rceil$ qubits. In uniform allocation bounds, b is an upper bound on all these word lengths. We use $C_M(b)$, $C_S(b)$, and $C_A(b)$ for the gate costs of clean modular multiply-accumulate, square-accumulate, and addition/subtraction primitives. These primitives preserve their source words and return their scratch to zero. $A_{\text{prim}}(b)$ is the largest number of additional scratch qubits used by one selected primitive. This allocation includes fixed-constant multiplication where required. It excludes the source and target words. In particular, multiplication implements $|x\rangle|y\rangle|z\rangle \mapsto |x\rangle|y\rangle|z + xy \bmod p\rangle$ for canonical x, y, z . Squaring allows its two sources to be the same word, but its target remains distinct. A zero-target product oracle is a different interface.

Clean modular primitive bounds. For every odd modulus $p < 2^b$, $b \geq 2$, the preceding interfaces have the following explicit Toffoli and workspace bounds:

$$\begin{aligned} A_{\text{prim}}(b) &\leq b + 4, \\ C_M(b) &\leq 24b^2 - 17b + 4, \quad C_S(b) \leq 24b^2 - 23b + 4, \\ C_A(b) &\leq 10b - 5, \quad C_\gamma(b) \leq 10b^2 - 5b. \end{aligned} \quad (43)$$

The cost C_γ applies to multiplication by a classical constant and accumulation into a distinct target. The circuit preserves every source. The target may initially hold any canonical residue. The circuit returns all additional qubits to zero. The same bounds hold for inverse circuits. An inverse circuit reverses the accumulation, for example $z \leftarrow z - xy \bmod p$. It does not compute x^{-1} . Only Toffoli gates are counted in these inequalities.

The scratch allocation is explicitly

register	qubits	
reusable clean operand	$b + 1$	
target extension	1	$(b + 1) + 1 + 1 + 1 = b + 4.$
wrap/underflow flag	1	
ripple carry	1	

The same allocation is reused for successive additions and Horner rounds. Thus the corresponding total primitive widths are at most $3b + 4$ for addition, $4b + 4$ for multiplication, and $3b + 4$ for squaring or classical-constant multiplication. These totals include all preserved sources and the separate target. Appendix E.1 explains why the source column “Ancillas” in the Chen inventory is not A_{prim} . The circuit loads a controlled classical constant with CNOT gates only. It loads a variable word controlled by one preserved bit with at

most b Toffolis. Five uncontrolled ripple additions then implement a modular update, including wrap-flag erasure. Since a ripple addition modulo 2^{b+1} uses $2b - 1$ Toffolis [14], a controlled variable update costs at most $16b - 5$ Toffolis. In-place modular doubling costs

$$D_b \leq 4b - 2. \quad (44)$$

Its inverse is modular halving because p is odd.

To accumulate xy into z , the circuit first halves z exactly $b - 1$ times. It then processes the bits of y from most to least significant. It adds $y_j x$ and doubles the target between consecutive bits. If z_0 is the original target and $\tilde{z} = 2^{-(b-1)} z_0 \bmod p$, the final value is

$$2^{b-1} \tilde{z} + x \sum_{j=0}^{b-1} y_j 2^j \equiv z_0 + xy \pmod{p}.$$

The initial compensation costs at most $(b - 1)(4b - 2) = 4b^2 - 6b + 2$ Toffolis, separately from the same cost for the interleaved doublings. Without it a nonzero initial target would be multiplied by 2^{b-1} . Consequently the multiplication ledger is

$$\underbrace{b(16b - 5)}_{\text{controlled updates}} + \underbrace{(b - 1)(4b - 2)}_{\text{initial halvings}} + \underbrace{(b - 1)(4b - 2)}_{\text{interleaved doublings}} = 24b^2 - 17b + 4.$$

For squaring, $g = x_j$ controls the j th update. The coincident source bit satisfies $x_j^2 = x_j$, so each load/unload pair saves two Toffolis. Three such pairs save six per round. The replacement of $16b - 5$ by $16b - 11$ gives $24b^2 - 23b + 4$. For a classical γ , a classical computation prepares $k_j = 2^j \gamma \bmod p$. The circuit accumulates the b controlled constants $x_j k_j$. It loads them with CNOT gates. This gives $C_\gamma \leq b(10b - 5)$ without Horner compensation. Appendix D.1 supplies the gate decomposition, flag proof, and finite ledger. After residue output, the large-modulus CRT update can reuse the tree workspace. The maximum of the separate stage bounds therefore gives the peak allocation. Every specified gate is a permutation on the full binary space. The modular specification applies to canonical residues.

The Mersenne rotation optimization [7, Section 3.2] specializes these resource bounds. If the active modulus is $p = 2^b - 1$, the same canonical arbitrary-target interfaces and scratch allocation have the bounds

$$\begin{aligned} A_{\text{prim}}^{\text{Mer}}(b) &\leq b + 4, & D_b^{\text{Mer}} &= 0, \\ C_M^{\text{Mer}}(b) &\leq 16b^2 - 5b, & C_S^{\text{Mer}}(b) &\leq 16b^2 - 11b, \\ C_A^{\text{Mer}}(b) &\leq 10b - 5, & C_\gamma^{\text{Mer}}(b) &\leq 10b^2 - 5b. \end{aligned} \quad (45)$$

The cost D_b^{Mer} is the Toffoli count for doubling or halving. These operations may still have Clifford or routing costs.

For $z = \sum_{j=0}^{b-1} z_j 2^j$, cyclic left rotation gives

$$D(z) = 2z - z_{b-1}(2^b - 1) \equiv 2z \pmod{p}.$$

The canonical set excludes the all-one word p . Rotation preserves that exclusion and fixes zero. Its output is therefore canonical. The reverse rotation D^{-1} is modular halving. This is the Mersenne shift optimization of [7, Section 3.2 and Fig. 4], applied here with an explicit check that the canonical domain is preserved. Since $D^b = I$, the initial $b - 1$ halvings can be combined into one left rotation, $D^{-(b-1)} = D$. The remaining $b - 1$ rotations between updates also use no Toffolis. The circuit uses the clean modular adder, including its canonical reduction and wrap erasure. The b updates then cost $b(16b - 5)$, or $b(16b - 11)$ for squaring. The addition and constant-accumulation bounds are unchanged, as is the reusable scratch allocation. This unconditional accumulation schedule requires no controlled rotation.

This specialization applies when the active modulus has Mersenne form. The form of the curve's base field alone does not suffice. In a padded physical word of width b , a Mersenne modulus $2^d - 1$ with $d \leq b$ uses rotations of its lowest d bits. The zero high bits remain untouched. The construction also works for composite $2^d - 1$ as ring arithmetic, although the prime schedule below uses only primes. Further reduction and product-circuit options are discussed in Appendix E.2.

Proposition 5.1 (A cached clean evaluator). *The polynomial map (36) has a clean arbitrary-target accumulation with ten preserved input words, five output words, eight local words, and primitive scratch. Its cost is*

$$C_{\text{node}} \leq 274C_M + 262C_S. \quad (46)$$

The scalar terminal uses the same eight-word local allocation and has cost $130C_M + 138C_S + C_\gamma$. A measured five-coordinate node can be phase-corrected with the ten inputs, one scalar correction word, and the same eight local words at cost

$$C_{\text{ghost}} \leq 638C_M + 776C_S. \quad (47)$$

All local words and primitive scratch are returned to zero.

Proof. The evaluator first computes $\Delta U_0, \Delta U_1, \Delta V_0, \Delta V_1, C_*$ in five words. It then computes $\mathcal{R}_*, \mathcal{E}_*, \mathcal{L}_*$ in the remaining three. It erases C_* first and then erases the four differences. Each reversal uses fresh scratch. The evaluator computes $\mathcal{K}^2$ in a freed word. It now holds four cached quantities at cost $43C_M + 50C_S$. It retains these caches while processing A_*, N_*, B_1 , and B_0 . It shares A_* and N_* across their output contributions. It clears each quantity before the next stage. The evaluator distributes the terms of N_* and B_i . This reduces the additional workspace to the four remaining words. The stage costs, including reversal, are

stage	M	S
prepare and clear the four caches	86	100
A_* contributions	28	36
N_* contributions	95	82
B_1 contribution	33	24
B_0 contribution	27	20
Z' contribution	5	0

The terminal adds N_* and then γD_* before clearing the caches. For ghost correction, the evaluator computes each completed coordinate into one zero scalar word. It applies the recorded binary parity and erases the word before processing the next coordinate. This scalar interface requires a separate count because it cannot assume output sharing. Appendix D.2 gives all associations, workspace allocations, and the resulting ghost count. $\square$

Thus a uniform local-move bound may use $G_* = 2 \max\{Q_w, 707A_b\}$ whenever $C_M, C_S, C_\gamma \leq A_b$. Here Q_w and A_b are the lookup and primitive bounds defined in Section 6.1. The scalar ghost correction determines this conservative constant. This bound cannot use twice the vector-evaluation cost. The executable certificate checks the complete word traces as polynomial identities over $\mathbb{Z}$ and verifies arbitrary-target reversal and recorded-phase correction.

Measured erasure uses the phase-corrected uncomputation method of CFS and Kornerup et al. [10,27]. We assume that a clean computation has produced a binary word $f(x)$ of length d while retaining its logical input x . A measurement of that word in the Hadamard basis gives a uniform classical outcome $\mu \in \{0, 1\}^d$. It applies the diagonal phase $(-1)^{\langle \mu, f(x) \rangle}$ to the retained state. If this phase is corrected coherently, the measured word can be reset without changing the logical state, including its entanglement with any other register.

We write the state before measurement as

$$\sum_x |x\rangle |\eta_x\rangle |f(x)\rangle, \quad \sum_x \|\eta_x\|^2 = 1,$$

where the possibly unnormalized vectors η_x include all other registers. The unnormalized state conditioned on μ is

$$2^{-d/2} \sum_x (-1)^{\langle \mu, f(x) \rangle} |x\rangle |\eta_x\rangle. \quad (48)$$

Orthogonality of the x register gives probability 2^{-d} for every μ . To correct the phase, the circuit recomputes $f(x)$ into clean workspace. It applies Z to the output positions selected by μ and uncomputes the word. These operations multiply the state by the same sign and restore all workspace. The two signs cancel for each x . The claim therefore holds for arbitrary logical states. The inputs to a correction must be available when it

is performed. The pebbling schedule includes their recomputation cost. Uniformity also holds conditional on earlier measurement outcomes. The complete classical transcript therefore reveals no logical input. After all corrections, the resulting logical channel equals the required unitary channel. It can therefore be invoked again on an arbitrary XOR target to erase the computed residue. This use of a fresh invocation does not require reversing a measurement.

A recorded five-coordinate ghost is corrected by the separate scalar schedule in Appendix D.2. Given the ten node-input words, its cost $638C_M + 776C_S$ includes coordinate recomputation and erasure within the eight local words and the reserved scalar correction word. The tree schedule separately accounts for reconstructing these inputs.

We use the explicit measurement-assisted schedule in Appendix C.1, following [10,27], with at most $8(h+1)L$ moves and $h+2$ vector pebbles. All residue words have the same physical width b . Each ordinary pebble is allocated five residue words, including the terminal pebble. The circuit implements a clean XOR oracle on an arbitrary external b -bit target as follows. It copies the computed scalar root to that target with CNOT gates. It then reverses the local evaluation. These two steps form the fused terminal action. This action costs twice the terminal evaluation cost. The schedule then removes all ghosts and corrects every recorded phase. Each Hadamard-basis erasure has uniform outcomes. The transcript is therefore independent of the logical input. The resulting channel is the desired XOR unitary. A fresh invocation erases its output. This cleanup does not require an inverse of a measurement process. The allocation includes the external target, local evaluation, and the scalar phase-correction word. Its size is

$$[5(h+2) + 10]b + A_{\text{prim}}(b) \quad (49)$$

qubits, besides the two scalar input registers. This allocation conservatively reserves $h+2$ full vector pebbles throughout. The move cost includes the terminal computation and its reversal.

5.4 Streaming reconstruction for the residue sampler

The residue-tree oracle computes the same integer polynomial modulo successive auxiliary primes. We reconstruct its value modulo q by explicit Chinese remaindering [4], as in the residue approach of CFS [10, Section 2.3]. The reconstruction uses a short fixed-point sum and an n -bit modular accumulator. We assume that $X \in \mathbb{Z}$ satisfies a certified bound $|X| \leq B$, with $B \geq 1$. For the masked sampler, X is the integer value of Ψ_γ in (27). We may take $B = 2^H$ from (41) or (42). We choose distinct auxiliary primes $p_1, \dots, p_K$, with $K \geq 1$. None equals q . We set

$$M = \prod_{i=1}^K p_i > 4B, \quad M_i = M/p_i, \quad a_i = M_i^{-1} \pmod{p_i}.$$

All residues below are canonical. Given $x_i = X \bmod p_i$, we define

$$t_i = a_i x_i \bmod p_i, \quad 0 \leq t_i < p_i.$$

Since $\sum_i t_i M_i \equiv X \pmod{M}$, there is an integer η such that

$$\sum_i t_i M_i = X + \eta M, \quad \sum_i \frac{t_i}{p_i} = \eta + \frac{X}{M}. \quad (50)$$

The bounds $|X/M| < 1/4$ and $0 \leq \sum_i t_i/p_i < K$ imply $0 \leq \eta \leq K$.

For the centered representative $|X| < M/4$, rounding recovers the integer quotient with the following guard precision. We define

$$\tau = \lceil \log_2(8K) \rceil, \quad A_{\text{fp}} = \sum_{i=1}^K \left\lfloor \frac{2^\tau t_i}{p_i} \right\rfloor.$$

The quotient satisfies

$$\eta = \left\lfloor \frac{A_{\text{fp}}}{2^\tau} + \frac{1}{2} \right\rfloor, \quad (51)$$

and the reconstructed residue therefore satisfies

$$X \bmod q = \left(\sum_i t_i (M_i \bmod q) - \eta (M \bmod q) \right) \bmod q. \quad (52)$$

An accumulator of

$$a_{\text{fp}} = \tau + \lceil \log_2 K \rceil + 1 \quad (53)$$

bits holds the full nonnegative sum and permits addition of $2^{\tau-1}$ during rounding, both without overflow. The quotient has at most $\lceil \log_2(K+1) \rceil$ bits. Both sizes are $O(\log_2(K+1))$.

For each summand, the definition of the floor function gives

$$0 \leq \frac{t_i}{p_i} - 2^{-\tau} \left\lfloor \frac{2^\tau t_i}{p_i} \right\rfloor < 2^{-\tau}.$$

The total error therefore satisfies $0 \leq \sum_i t_i/p_i - 2^{-\tau} A_{\text{fp}} < K 2^{-\tau} \leq 1/8$. Together with (50), this yields

$$-\frac{3}{8} < \frac{A_{\text{fp}}}{2^\tau} - \eta < \frac{1}{4},$$

which proves (51) by rounding to the nearest integer. The reduction of $X = \sum_i t_i M_i - \eta M$ modulo q gives (52). Each summand of A_{fp} is below 2^τ . Thus $A_{\text{fp}} + 2^{\tau-1} < (K + 1/2) 2^\tau$, and the width in (53) therefore suffices. The bound $0 \leq \eta \leq K$ gives the stated quotient length.

This reconstruction requires only pairwise coprimality of the auxiliary moduli. The condition $M > 4B$ separates the exact sum from the half-integer rounding boundaries and leaves room for the truncation error. The classical computation prepares M and the required reduced constants. Section 6.1 gives their streaming generation. The stored product M has $O(\log_2 M)$ bits. The prime bound in Section 6.1 gives polynomial classical storage in n .

We next implement the fixed-point sum and quotient correction reversibly. This implementation uses only odd auxiliary primes. The circuit maintains an n -bit accumulator $Y \in \mathbb{Z}_q$ and the a_{fp} -bit accumulator A_{fp} . Both accumulators are initially zero. For each p_i , the circuit computes x_i with the clean residue-tree oracle. It forms t_i and adds its contributions to the two accumulators. It then erases the digit and residue. The circuit reuses the workspace for the next modulus.

We describe the fixed-point contribution without the index i . We set $y_0 = t$, $y_j = 2y_{j-1} \bmod p$ in canonical residues. We define $d_j = \lfloor 2y_{j-1}/p \rfloor \in \{0, 1\}$. Since p is odd,

$$y_j = 2y_{j-1} - p d_j, \quad d_j = y_j \bmod 2.$$

Iteration of this recurrence gives

$$2^\tau t = p \sum_{j=1}^{\tau} d_j 2^{\tau-j} + y_\tau, \quad 0 \leq y_\tau < p,$$

so the weighted digit sum is exactly $\lfloor 2^\tau t/p \rfloor$. After the j th modular doubling, the lowest bit of y_j controls addition of $2^{\tau-j}$ to A_{fp} . These controls are coherent. The circuit does not measure the digits. It finally applies the τ inverse doublings to the residue word alone. These operations restore t and retain its contribution in A_{fp} . The circuit needs no separate τ -bit quotient output or division history. All operations extend linearly to superpositions and leave their local scratch clean.

For a general odd modulus, modular doubling uses a zero extension bit and an initially zero underflow flag in addition to the adder workspace. The circuit extends the b -bit word by a zero high bit. It rotates the $b+1$ bits left once and obtains $2t$. It subtracts p and copies the resulting high bit into a flag e . It then adds p back controlled by e . The output is the canonical residue $y = 2t \bmod p$. Its extension bit is zero. Since $e = 1 - \lfloor 2t/p \rfloor$ and $y \bmod 2 = \lfloor 2t/p \rfloor$, the update $e \oplus = 1 \oplus (y \bmod 2)$ clears the flag. The inverse map is modular halving, explicitly $y \mapsto (y + p(y \bmod 2))/2$ on canonical residues. For an active Mersenne modulus $p = 2^d - 1$, the rotations used in (45) replace these doublings and halvings. Their 2τ rotations have zero Toffoli cost. The controlled additions to A_{fp} retain their cost.

For the other contribution, the circuit performs $Y \leftarrow Y + t_i(M_i \bmod q) \pmod{q}$. Once all moduli have been processed, (51) permits the correction without a separate η register. The circuit temporarily adds $2^{\tau-1}$ to A_{fp} . Its bits at positions τ and above then represent η , with positions numbered from zero. The bit at position $\tau + k$ controls subtraction of the precomputed constant $2^k(M \bmod q) \bmod q$ from Y . These subtractions implement $Y \leftarrow Y - \eta(M \bmod q) \pmod{q}$. The circuit undoes the constant addition to A_{fp} . By (52), Y now holds $X \bmod q$.

The circuit clears the residue-tree workspace before performing arithmetic on Y . Its $n + 1$ -bit constant-adder operand therefore reuses that space. The circuit retains A_{fp} through the character phase. It then undoes the quotient correction and the per-modulus updates. These steps return both accumulators to zero. When a residue oracle uses measured erasure, each cleanup uses a fresh phase-corrected XOR invocation as in the measured-erasure calculation in Section 5.3. Subtraction undoes the modular additions. Appendix C records this clean interface and the tree schedule that supplies its residues.

5.5 Public masks and sampling error bounds

The window summands in (34) are structured functions of the scalar inputs and may be zero. Their child sums therefore need not satisfy the generic addition conditions with high probability. Public masks make the two sums in each child pair independent and uniform for every fixed input. We can therefore count exceptional pairs to obtain a uniform arithmetic error bound. The resulting translation of F preserves the Fourier probabilities on $\mathbb{Z}_r^2$.

Proposition 5.2 (Two-seed masks for balanced trees). *We consider any finite abelian group G of odd order. For a complete balanced tree with $L = 2^h \geq 2$ ordered leaves, we choose independent uniform $U, V \in G$. We replace each leaf $A_i(x)$ by $A_i(x) + R_i$, where*

$$R_i = U + [i]V, \quad T_R = [L]U + \left\lfloor \frac{L(L-1)}{2} \right\rfloor V. \quad (54)$$

For every fixed x , the two true child sums at every internal node are independent uniform elements of G . The true group sum at the root is $F(x) + T_R$. The seeds attain the minimum entropy $2 \log_2 |G|$ among schemes that require an exact independent uniform root pair for every fixed input.

Proof. A subtree occupies a consecutive leaf interval. For adjacent child intervals of sizes s_1, s_2 starting at j , their mask sums are the image of (U, V) under

$$M_v = \begin{pmatrix} s_1 & s_1 j + s_1(s_1 - 1)/2 \\ s_2 & s_2(j + s_1) + s_2(s_2 - 1)/2 \end{pmatrix}, \quad \det M_v = s_1 s_2 (s_1 + s_2)/2.$$

In a balanced tree $s_1 = s_2 = 2^k$, so the determinant is 2^{3k} . Multiplication by this integer is invertible on an odd-order group. The adjugate formula therefore makes M_v a bijection on G^2 . Addition of fixed unmasked sums preserves uniformity. The sum of the masks is T_R . An independent uniform root pair has entropy $2 \log_2 |G|$. It is a deterministic function of the seed at fixed x . The seed entropy must therefore satisfy the stated lower bound. Our independent uniform seeds have entropy $H(U, V) = 2 \log_2 |G|$, so they attain this bound. $\square$

For an ordered binary tree, the same argument applies when $\gcd(\exp G, s_1 s_2 (s_1 + s_2)/2) = 1$ at each node. Prime order $|G| > L$ suffices. This affine randomization [6] uses invertibility for sums over child intervals. Appendix B treats general trees and independent leaf masks. Appendix B discusses zero-sum masks.

We write $R = (U, V)$ for the public group seeds. In the genus-two sampler, a classical computation prepares the masked window tables. It uses the known generator to sample U, V . A degree-two entry has $Z = 1$. A lower-degree entry receives the dummy tuple $(0, 0, 0, 0, 1)$. Every nonterminal node evaluates the fixed compact polynomial map (36). The root evaluates $\Psi_\gamma = N_* + \gamma D_*$ with an independent uniform field seed γ . The resulting completed character is $\tilde{g}_{R, \gamma}(x)$. This integer program is defined on all tuples. Every auxiliary ring uses the same polynomial reduced modulo its modulus. The circuit needs no quantum mask, degree-tag, or exceptional-branch register.

The genericity conditions are

$$\deg u^{(1)} = \deg u^{(2)} = 2, \quad Z_1 Z_2 \mathcal{R}_* \mathcal{E}_* \neq 0. \quad (55)$$

These conditions ensure that the map represents group addition for every reached valid weighted representative. Bihomogeneity makes the condition invariant under nonzero scale choices. Polynomial evaluation remains defined when the conditions fail.

Lemma 5.3 (Frequency of an exceptional generic addition). *We define $e = \#\{D \in G : \deg u_D < 2\}$. For independent uniform $D, E \in G$, the probability that their canonical degree-two representatives fail the generic condition (55) is at most*

$$\min \left\{ 1, \frac{3e + 8q}{r} \right\} \leq \min \left\{ 1, \frac{14q + 3}{r} \right\}. \quad (56)$$

Here an input of degree less than two is counted as a failure.

Proof. The two input degree failures have combined probability at most $2e/r$. We fix a monic quadratic u_D . At most $2q$ monic quadratics u_E share a factor with it. If u_D splits, each of its at most two roots allows at most q choices of the other root of u_E . If u_D is irreducible, only $u_E = u_D$ is possible. For each such u_E , there are at most four valid polynomials v_E , by the Mumford counting argument in Section 4.1. Thus common support contributes at most $8q/r$, also after restricting to a subgroup. When the inputs have degree two and coprime u -polynomials, the interpolation polynomial has degree at most three. Its cubic coefficient is nonzero exactly when Cantor reduction produces a degree-two output. If this coefficient vanishes, the monic quintic curve equation gives a degree-one reduced output. Thus the remaining generic denominator failure is contained in $\{\deg u_{D+E} < 2\}$. The sum $D + E$ is uniform, so this event has probability e/r . The union bound and $e \leq 2q + 1$ give the result. $\square$

Theorem 5.4 (Input-uniform masked arithmetic). *We use e from Lemma 5.3 and define*

$$\varepsilon_{\text{tree}} = \min \left\{ 1, \frac{(L-1)(3e + 8q)}{r} \right\}.$$

For every fixed input x , the probability over the two fresh group seeds that at least one true child pair is exceptional is at most $\varepsilon_{\text{tree}}$. Consequently, for every seed-independent input distribution, its bad-input mass ε_R satisfies

$$\mathbb{E}_R \varepsilon_R \leq \varepsilon_{\text{tree}}. \quad (57)$$

Outside the bad set, $\tilde{g}_{R,\gamma}(x) = g_\gamma(F(x) + T_R)$ for every γ . The randomized clean phase channel differs from this translated ideal channel by at most $\min\{1, 2\sqrt{\varepsilon_{\text{tree}}}\}$ in half-diamond norm, even with the classical seeds retained. The half-diamond distance between channels $\mathcal{E}$ and $\mathcal{F}$ is $\frac{1}{2}\|\mathcal{E} - \mathcal{F}\|_\diamond$. This distance is the maximum output trace distance over input states with an arbitrary reference system.

Proof. We define each bad event using true group sums, even if a lower polynomial node is bad. Proposition 5.2 and Lemma 5.3 bound the probability at each node. The union bound over $L - 1$ nodes then gives the pointwise bound. This argument requires no independence between the events. Outside the bad set, the generic correctness proof applies inductively. The terminal square-factor identity then gives the claimed phase.

For a normalized input with an arbitrary reference, we write $|\psi\rangle = \sum_x |x\rangle |\eta_x\rangle$. The squared norm of the difference of its two phase outputs is at most $4 \sum_{x \text{ bad}} \|\eta_x\|^2$. Its seed average is at most $4\varepsilon_{\text{tree}}$, uniformly in the input. The coupled isometries draw the same classical seeds. Their difference therefore has operator norm at most $2\sqrt{\varepsilon_{\text{tree}}}$. The standard isometry comparison and contractivity give the half-diamond bound. Theorem B.1 details the channel construction. $\square$

The channel bound averages fresh masks drawn independently of the input and its reference. The Legendre sampling analysis below uses the exact uniform domain $\mathbb{Z}_r^2$.

The masks change only classical lookup constants. The canonical leaf and dummy bounds, degrees, and arithmetic schedule remain unchanged.

To bound the success probability, we use V_G for the shift-averaged ideal lower bound in (21). For every fixed mask, translation by T_R multiplies the Fourier amplitudes by unit phases and leaves their probabilities

unchanged. We write $\tilde{p}_{\text{use}}$ for the success probability of the masked arithmetic with exact preparation and readout, averaged over R, γ . For each mask, the translated ideal failure probability is $\mu_{g_\gamma}^2$ by Theorem 3.1. The failure projector is the complement of the outcomes $u \neq 0, v = ku \bmod r$, where $Q = [k]P$. We apply the exact Fourier transform and then this projector to the phase-state difference in the preceding proof. Minkowski's inequality over the seeds then gives

$$\begin{aligned} \sqrt{1 - \tilde{p}_{\text{use}}} &\leq \sqrt{\mathbb{E}_\gamma \mu_{g_\gamma}^2} + 2\sqrt{\mathbb{E}_R \varepsilon_R} \\ &\leq \sqrt{1 - V_G} + 2\sqrt{\varepsilon_{\text{tree}}}. \end{aligned} \quad (58)$$

This argument allows correlation between T_R and the arithmetic bad set. With at most J_{prep} preparation trials per register, the abort probability is at most $2^{1-J_{\text{prep}}}$ by Theorem 3.3. We use ε_F and ε_{syn} for the total distribution error budgets for Fourier readout and rotation synthesis. We square (58) and add these losses to obtain

$$p_{\text{run}} \geq \max \left\{ 0, 1 - \left(\sqrt{1 - V_G} + 2\sqrt{\varepsilon_{\text{tree}}} \right)^2 - 2^{1-J_{\text{prep}}} - \varepsilon_F - \varepsilon_{\text{syn}} \right\}. \quad (59)$$

Candidate labels $u \neq 0$ yield $k' = vu^{-1} \bmod r$. The algorithm accepts a candidate only after the classical check $[k']P = Q$. Each run uses fresh independent masks and a fresh field shift. Thus for any positive lower bound p_* in (59), the expected number of runs is at most $1/p_*$. A sequence of N independent runs fails with probability at most $(1 - p_*)^N$.

For $r = \Theta(q^2)$ and $w = \Theta(\log_2 n)$, the bounds are $1 - V_G = O(q^{-1})$ and $\varepsilon_{\text{tree}} = O(L/q)$. The square of (58) therefore gives $1 - \tilde{p}_{\text{use}} = O(L/q)$. Equation (59) yields

$$p_{\text{run}} \geq 1 - O(L/q) - \varepsilon_F - \varepsilon_{\text{syn}} - 2^{1-J_{\text{prep}}}. \quad (60)$$

Appendix D.4 gives the asymptotic details. The complete width allocation and finite resource estimates are collected in Section 6.1.

6 Detailed Estimates

Throughout this section, $n = \lceil \log_2 q \rceil$ is the field bit length. The quantity $m_r = \lceil \log_2 r \rceil$ is the subgroup-order bit length.

We use W for the peak logical width and C_{Tof} for the number of ordinary three-qubit Toffoli gates. Clifford gates, measurements, and classical feed-forward are separate resources. The symbol T below denotes the Clifford+ T gate. It does not denote a Toffoli gate. The earlier bounds denoted by G become Toffoli bounds only after the count includes every local primitive, control, and cleanup operation in that gate basis.

Unless stated otherwise, the estimates cover one sampling run. They include its live inputs, arithmetic, preparation, and Fourier processing. We consider independent identically distributed runs with success probability $p > 0$. Each run has a fixed Toffoli count C_{Tof} . The expected count to obtain a verified logarithm is C_{Tof}/p . A worst-case per-run count and a conditional success lower bound give the corresponding upper bound. Serial repetitions reuse the same qubits.

6.1 Sampler bounds and finite resource estimates

Complete width. We assume the curve model and subgroup hypotheses of Section 2. We also use the computational model of Section 3. In particular, q and r are odd primes and $N_{\text{in}} = 2m_r$. Both scalar registers remain live during arithmetic. The preprocessing cost includes the classical lookup tables.

The construction uses a complete balanced window tree with $L = 2^h \geq 2$ leaves. Its distinct odd auxiliary primes $p_i < 2^b$ differ from q and satisfy $M = \prod_{i=1}^K p_i > 4 \cdot 2^{H_{\text{cw}}}$. The root bit bound H_{cw} is given in (42). The residue words have uniform width b with $w - 1 \leq 8b$. The guarded CRT construction in Section 5.4 uses $a_{\text{fp}} = 2\lceil \log_2 K \rceil + 4$ fixed-point bits.

With the clean primitives in (43), residue reconstruction retains both scalar inputs, the n -bit CRT output, and the fixed-point sum. Its tree workspace is at most $[5(h+2) + 11]b + 4$ by (49) and $A_{\text{prim}}(b) \leq b + 4$. The large-modulus update starts after the tree workspace has been cleared. It uses at most $N_{\text{in}} + 2n + a_{\text{fp}} + 2b + 20$ qubits. The character width $W_\chi(n) \leq 6n + 12$ from (32) already includes its n -bit argument. The fixed-point

sum remains live until inverse reconstruction. Theorem 3.3 supplies the preparation and Fourier width. The stages reuse clean workspace, so

$$W_{\text{sample}} \leq \max \left\{ \begin{array}{l} N_{\text{in}} + n + a_{\text{fp}} + [5(h+2) + 11]b + 4, \\ N_{\text{in}} + 2n + a_{\text{fp}} + 2b + 20, \\ N_{\text{in}} + a_{\text{fp}} + 6n + 12, \\ 3m_r + 5 \end{array} \right\}. \quad (61)$$

The first entry bounds the reconstruction-stage allocation W_{recon} . The remaining entries cover the large-modulus update, the completed character, and preparation/Fourier readout.

For $r = \Theta(q^2)$ and $w = \Theta(\log_2 n)$, the table size and the integer bit bound are polynomial in n . The auxiliary primes have bit size $b = O(\log_2 n)$ and also satisfy $w - 1 \leq 8b$. This choice gives $a_{\text{fp}} = O(\log_2 n)$ and a tree allocation $O(\log_2^2 n)$ beyond the n -bit reconstructed value. The CRT update uses at most another $n + O(\log_2 n)$ bits. The exact character circuit uses $6n + O(1)$, including that value. The input width is $N_{\text{in}} = 4n + O(1)$, and the preparation/Fourier width is $3m_r + 5 = 6n + O(1)$. The character stage therefore gives the largest leading term in (61). The resulting bound is

$$W_{\text{sample}} \leq 10n + o(n). \quad (62)$$

Classical preprocessing and reversible arithmetic have polynomial cost under this window choice. The height growth $L^{\log_2 20}$ gives a large factor in this cost. Appendix D.4 gives the details. The finite estimates below evaluate each stage separately.

Theorem 6.1 (Masked residue sampler). *We assume the curve and subgroup hypotheses of Section 2 and use the computational model of Section 3. The construction in Sections 4 and 5 then has the following properties. Its weighted Mumford addition map has ordinary polynomial degree at most 48 and satisfies*

$$\mathcal{H}(\mathcal{A}(U, V)) \leq 2^8 \mathcal{H}(U)^{10} \mathcal{H}(V)^{10},$$

where $\mathcal{H}$ is the tuple size defined in (38). At a generic terminal node, the polynomial $\Psi_\gamma = N_* + \gamma D_*$ has degree at most 19. At such a node, the completed character of its value equals that of $u_0 + \gamma$. The circuit evaluates this phase without field inversion. Generic correctness requires the nonvanishing condition in Proposition 4.3. Polynomial evaluation itself is defined on every tuple.

The sampler uses the exact input domain $\mathbb{Z}_r^2$, a complete balanced tree with $L = 2^h \geq 2$ leaves, and the two-seed affine masks in (54). Each run samples two fresh independent group seeds and an independent uniform $\gamma \in \mathbb{F}_q$. The ideal useful-sample bound is V_G from (21). The tree exception bound is $\varepsilon_{\text{tree}}$ from Theorem 5.4. The preparation trial cap is J_{prep} per register. The distribution error budgets are ε_{F} , ε_{syn} for Fourier readout and rotation synthesis, respectively. The probability of a classically verified logarithm in one run is at least

$$\max \left\{ 0, 1 - \left(\sqrt{1 - V_G} + 2\sqrt{\varepsilon_{\text{tree}}} \right)^2 - 2^{1-J_{\text{prep}}} - \varepsilon_{\text{F}} - \varepsilon_{\text{syn}} \right\}.$$

The probability is averaged over the classical seeds for the exact uniform input state. Every accepted candidate is checked by $[k']P = Q$.

The masked leaf tables are computed classically using complete divisor arithmetic. Each lower-degree table value receives a fixed dummy tuple. The circuit then evaluates a fixed integer polynomial at each node. Thus the quantum circuit needs no mask register or coherent record of exceptional branches. The stated probability estimate justifies this choice. It counts every relevant lower-degree or singular generic addition as a bad event.

For the auxiliary moduli and fixed-point allocation specified above, the complete logical width is bounded by (61). For $r = \Theta(q^2)$ and window width $w = \Theta(\log_2 n)$, it satisfies (62). Under this parameter choice, the classical preprocessing and arithmetic costs are polynomial. With exact preparation and readout, the useful-sample probability is $1 - O(L/q)$.

Proof. Proposition 5.2 makes the true child pairs uniform for each fixed scalar input. Theorem 5.4 therefore bounds the error caused by exceptional additions. The full-run success follows from (59). Its asymptotic form is (60). Exact preparation and readout remove the preparation, Fourier, and synthesis losses. The preceding stage-allocation argument proves (61). Its asymptotic form gives (62). Appendix D.4 gives the detailed size and prime-product estimates. $\square$

A 127-bit field instance. We now apply Theorem 6.1 to the Gaudry–Schost curve used by μ Kummer [34]. Its parameters satisfy

$$\begin{aligned} q &= 2^{127} - 1, & \#J(\mathbb{F}_q) &= 16r, \\ r &= 2^{250} - 0\text{x}334\text{D}69820\text{C}75294\text{D}2\text{C}27\text{F}\text{C}9\text{F}9\text{A}154\text{F}\text{F}47730\text{B}4\text{B}840\text{C}05\text{B}\text{D}. \end{aligned} \quad (63)$$

The cited construction gives the primality of r and a nonzero divisor of order r . To obtain the zero-quartic model assumed by our formulas, we substitute $X = x + a$, where $a = -f_4/5$ in the published monic model. The resulting curve and generator are

$$C' : y^2 = x^5 + c_3x^3 + c_2x^2 + c_1x + c_0, \quad P = (x^2 + u_1x + u_0, v_1x + v_0),$$

with the hexadecimal constants in Table 2.

Table 2. An isomorphic zero-quartic Gaudry–Schost model. Constants are canonical residues modulo q , written in hexadecimal with a 0x prefix.

Parameter	Value
a	0x5FAD7267DA7A005DC143AD1352526C82
c_0	0x3AF79604317A895BB28139B4DD66D56
c_1	0x4761178EC206F6DB2E90B2BF4913BE47
c_2	0x432B709D2D5801982F927BB531F0F535
c_3	0x00D26A12F16AE9D00045BB92C5382A8E
u_0	0x384CB21DBD5E1F600F3FE0E336182F14
u_1	0x3CB88102BCDD5A7AAA402188B6783790
v_0	0x4F4AB1A434DFCBE8FD26C203D18D54F4
v_1	0x444569AF177A9C1C721736D8F288C942

This instance has $n = 127$ and $m_r = \lceil \log_2 r \rceil = 250$. The live input width is $N_{\text{in}} = 500$ for the exact domain $\mathbb{Z}_r^2$. For a tree with $L = 32$ leaves, the masking union bound evaluates to

$$\varepsilon_{\text{tree}} \leq \frac{31(14q + 3)}{r} < 4.082 \times 10^{-35} < 2^{-114}.$$

The corresponding $2\sqrt{\varepsilon_{\text{tree}}}$ state-distance bound is less than 1.278×10^{-17} .

The accompanying `check_gs_parameters.py` verifies square-freeness, membership, $P \neq 0_G$, and $[r]P = 0_G$ by complete Cantor arithmetic. Lucas–Lehmer and a supplied recursive Pocklington certificate prove primality of q and r . These checks, together with exact integer Weil bounds, certify $\text{ord}(P) = r$ and $\# \text{Jac}(C')(\mathbb{F}_q) = 16r$. Appendix D.3 gives the details. The identification with the published model relies on the cited source. These certificates verify classical parameters.

For $L = 32$, the coarse fiber bound in Section 4 and the square of (58) give $1 - \tilde{p}_{\text{use}}$ below $1.795 \times 10^{-34} < 2^{-112}$. We choose $J_{\text{prep}} = 42$ and $\varepsilon_{\text{F}} = \varepsilon_{\text{syn}} = 2^{-40}$. The preparation, Fourier, and synthesis losses sum to $2^{-41} + 2 \cdot 2^{-40}$. Equation (59) therefore gives

$$1 - p_{\text{run}} < 1.795 \times 10^{-34} + 2^{-41} + 2 \cdot 2^{-40} < 2.274 \times 10^{-12} < 2^{-38}.$$

The measured Fourier schedule uses 292 classical precision bits. The finite estimates below include the preparation, Fourier-readout, and character costs. Rotation synthesis is charged separately through the stated accuracy budget.

Finite widths. We use the root bit bound H_{cw} from (42). One explicit choice uses all odd primes at most 2^b , with q excluded. Dusart’s unconditional estimates [16] give the following bounds, where $\ln$ denotes the natural logarithm:

$$\frac{2^b}{\ln 2} \left(1 - \frac{0.2}{(b \ln 2)^2} \right) - 1 - n > H_{\text{cw}} + 2 \implies M > 4 \cdot 2^{H_{\text{cw}}}$$

when $2^b \geq 3594641$. The number of primes satisfies

$$K \leq K_{\max} := \left\lceil \frac{2^b}{b \ln 2 - 1.1} \right\rceil.$$

We reserve the fixed-point space using $K_{\max}$. We set $\ell_K = \lceil \log_2 K_{\max} \rceil$, $\tau = \ell_K + 3$, and $a_{\text{fp}} = 2\ell_K + 4$. The subtraction of $1 + n$ above allows removal of 2 and q . These estimates certify the allocation without requiring a prime list for the numerical bound.

We substitute $A_{\text{prim}}(b) \leq b + 4$ and these register sizes into (61). The resulting allocations are

w	L	b	$K_{\max}$	a_{fp}	W_{recon}
8	64	31	105,333,026	58	2,270
16	32	27	7,619,525	50	1,923
24	32	27	7,619,525	50	1,923
32	16	23	565,180	44	1,618

Each reconstruction width exceeds the other stage widths. For $w = 16$, the four allocations are 1923, 878, 1324, 755. For $w = 32$, they are 1618, 864, 1318, 755. Window 24 has the same quantum width as window 16, but its lookup table is larger.

Finite Toffoli count. We define

$$A_b = 24b^2 - 17b + 4, \quad Q_w = 4w2^w, \quad G_* = 2 \max\{Q_w, 707A_b\}.$$

The clean local node uses $274C_M + 262C_S$. The terminal uses $130C_M + 138C_S + C_\gamma$. Coordinatewise ghost correction uses $638C_M + 776C_S$. Thus $707A_b$ bounds each of the first two costs, and $1414A_b$ bounds ghost correction. A lookup equality ladder uses at most $w - 1 \leq 8b$ clean selector bits while the eight local words are free. Its computation and erasure cost less than $4w$ Toffolis per entry. The selected output is copied with CNOTs. Thus G_* covers local evaluation, erasure, and ghost correction.

One reconstruction direction has the bound

$$G_R \leq K_{\max} [16(h+1)LG_* + 40b(b+1) + \tau\{8b-4+20(a_{\text{fp}}+1)\} + 20b(n+1)] \\ + 40(a_{\text{fp}}+1) + 20(\ell_K+1)(n+1).$$

The terms involving the n -bit target include the large-modulus operations. Reconstruction runs once before the character phase and once after it. For the stated preparation cap and Fourier precision, the full-run budget is

$$C_{\text{Tof}} \leq 2G_R + C_\chi(n) + 4J_{\text{prep}}(2m_r - 1) + 10 \cdot 292(2m_r - 1).$$

Here $C_\chi(n)$ already includes the character computation and its reversal. For $w = 16$, the two reconstruction directions cost at most 1,128,039,642,229,582,960 Toffolis. The character, preparation cap, and Fourier readout contribute at most 2,331,720, 83,832, and 1,457,080, respectively. Their sum gives

$$C_{\text{Tof}} \leq 1,128,039,642,233,455,592 < 2^{59.969}.$$

The same formula with $w = 32$ gives

$$C_{\text{Tof}} \leq 1,590,840,273,492,881,652,552 < 2^{70.431}.$$

Both counts cover one capped run. Both runs use the same preparation and readout budgets. The smaller tree at $w = 32$ reduces the masking error, so the bound $p_{\text{run}} > 1 - 2^{-38}$ holds for both windows. Their expected Toffoli costs per verified logarithm are at most the corresponding run bounds divided by $1 - 2^{-38}$. Single-qubit rotation synthesis is charged separately through its accuracy budget. These analytic counts do not include classical preprocessing. The estimates use generic primitives for every auxiliary prime. Appendix E.2 gives a refinement for a certified explicit CRT list.

Classical storage and table preparation. The prime list and the product M depend on the field and window parameters. They do not depend on the fresh masks. A classical preprocessing pass enumerates the primes, forms M , and records $M \bmod q$. The controller needs no stored copy of each full M_i . For the active prime p_i , it computes

$$a_i = \left(\frac{M \bmod p_i^2}{p_i} \right)^{-1} \bmod p_i, \quad M_i \bmod q = (M \bmod q)(p_i^{-1} \bmod q) \bmod q.$$

The first division is exact because $p_i \mid M$. The first inverse exists because the primes are distinct. The second inverse exists because $p_i \neq q$. The controller retains these reduced constants for the active modulus and recomputes them in the reverse pass. It stores M and the prime list in $O(Kb)$ bits. The controller computes these constants by modular reduction of the stored product M . This procedure avoids storing K integers of size $\log_2 M$.

Each run uses fresh group seeds and therefore prepares fresh masked leaf tables. For a nonpadding leaf, the classical computation starts at R_i and repeatedly adds its fixed window base using complete Cantor arithmetic. It stores the four canonical field coefficients for each degree-two result. It stores four zeros for a lower-degree result, in accordance with the dummy tuple in Section 5.5. Every entry has the fixed scale $Z = 1$. Thus at most $4nL2^w$ bits store the coefficient tables, before metadata. This construction uses $O(L2^w)$ complete classical additions, besides preparation of the window bases and masks. For $w = 16$, the entry bound is 2^{21} and the packed coefficient storage bound is 133,169,152 bytes. For $w = 32$, the corresponding bounds are 2^{36} entries and 4,363,686,772,736 bytes.

During each residue lookup, the controller reduces the stored coefficients modulo the active p_i and emits the corresponding classically controlled X/CNOT pattern. It reuses the same canonical tables across all auxiliary moduli and through phase correction and cleanup. It needs no separate table for every modulus. The controller can emit each lookup pattern when the schedule reaches it, so it also needs no stored list of all elementary gates. These policies use $O(nL2^w + Kb)$ classical bits. The gate estimates include reversible selection, but the classical preparation and generation of constants remain separate costs.

6.2 Matched comparison with earlier circuits

Chen et al. [7] report the controlled-addition allocation and modular-operation inventory

$$W_{\text{add}} = 20n + 2\lceil \log_2 n \rceil + 10, \quad C_{\text{add}} = 836n^2 + 18n - 340.$$

Appendix E.1 retains their primitive table. It distinguishes their printed polynomial from the expansion of the inventory. The appendix checks the reported inventory without changing its circuit interfaces.

A controlled addition includes one control qubit in W_{add} . A sequential, unwindowed full-output schedule retains two m_r -bit scalar registers and performs $2m_r$ controlled additions. Its arithmetic count is $2m_r C_{\text{add}}$, and its allocation is

$$W_{\text{add}} + 2m_r - 1 \quad \text{logical qubits.} \tag{64}$$

The subtraction shares the module's control with one live scalar bit. The addition workspace is reused. The peak also includes the preparation and Fourier stages.

The width in [7, Sections 5.1–5.2] describes the addition module. Its gate total multiplies that module's count by $2m_r$. Equation (64) therefore adds the remaining live inputs. A circuit with scalar-register recycling would require its own allocation and sampling analysis. The comparison here uses the same two live input registers for all rows. The substitution $2m_r = 4n$ is illustrative. It is not an identity for every genus-two subgroup.

For the instance in Section 6.1, Chen's reported module width is $20 \cdot 127 + 2\lceil \log_2 127 \rceil + 10 = 2,564$ qubits. Its matched allocation is $2564 + 500 - 1 = 3063$ qubits. The inventory gives 6,742,895,000 Toffolis for 500 additions. The preparation and Fourier stages use at most 755 qubits and do not increase this peak. The comparison retains the source's reported module width and generic-addition domain. Section 2.1 gives a clean XOR adapter for the later inversion circuit of Luo et al. [30]. Replacing Chen's fourteen inversion calls also requires the appropriate controlled interfaces and a new allocation of live registers. The reported Chen row retains its original circuit; it is not an estimate for this replacement. The p_{521} multiplication results in

[25, Appendix A, Table 6] also use different interfaces, field size, and Clifford+ T objectives. Appendix E.2 gives the space and gate adapters for such a replacement. A T -count and a Toffoli count measure different resources.

Comparison with Huang et al. (2020). Huang et al. [24] gave a reversible genus-two divisor-addition construction over prime fields before the Mersenne-field implementation of Chen et al. Section 5.1, Eq. (3), of their paper reports the controlled divisor-addition width

$$W_{\text{H,add}} = 13n + 2\lfloor \log_2 n \rfloor + 10.$$

The logarithm is rounded down, as in the original formula. The allocation includes one control qubit, the inversion workspace, and six additional n -bit words. Their Algorithm 1 and Figure 2 retain both scalar registers through the addition sequence and Fourier readout. The same live-input convention used above therefore gives the arithmetic-stage allocation

$$W_{\text{H,matched}} = W_{\text{H,add}} + 2m_r - 1 = 13n + 2\lfloor \log_2 n \rfloor + 2m_r + 9.$$

The subtraction of one shares the addition module’s control with one live scalar-input bit. At $n = 127$ and $m_r = 250$, $\lfloor \log_2 127 \rfloor = 6$, and hence

$$W_{\text{H,add}} = 13 \cdot 127 + 2 \cdot 6 + 10 = 1,673, \quad W_{\text{H,matched}} = 1,673 + 500 - 1 = 2,172.$$

The preparation and Fourier allocation of 755 qubits uses our common implementation. Retaining the four-word divisor output during readout adds 508 qubits. The resulting 1,263 qubits stay below the arithmetic peak. Our $w = 16$ and $w = 32$ allocations are thus 11.46% and 25.51% lower, respectively. Table 3 includes this earlier comparison.

Table 3. Finite resources on the same 127-bit instance with 500 live input qubits. The earlier rows use reported generic arithmetic allocations and operation inventories. Their gate entries cover arithmetic only. The residue rows bound complete capped sampling runs and exclude rotation synthesis. The fourth column counts classical residue lookup entries. The earlier circuits use fixed classical divisors. The fifth column states their published domain or the verified-logarithm probability proved here. All widths are schedule-specific bounds.

Construction	Qubits	Toffoli gates	Classical lookup entries	Domain / success probability
Huang et al. [24]	2172	$\approx 3.055 \cdot 10^{10}$	—	Published generic domain
Chen et al. [7]	3063	$6.743 \cdot 10^9$	—	Published generic domain
Residue, $w = 16$	1923	$< 2^{59.969}$	$\leq 2^{21}$	$> 1 - 2^{-38}$
Residue, $w = 32$	1618	$< 2^{70.431}$	$\leq 2^{36}$	$> 1 - 2^{-38}$

The residue widths are 37.22% and 47.18% below the Chen-derived allocation. They are 11.46% and 25.51% below the Huang-derived allocation. These percentages compare the stated live-register schedules. The earlier arithmetic inventories do not establish an end-to-end verified-logarithm probability under the present budgets.

The smaller residue allocations have much larger declared gate costs. For $w = 16$, the Toffoli upper bound is about $1.67 \cdot 10^8$ times Chen’s reported arithmetic count. For $w = 32$, the corresponding ratio is about $2.36 \cdot 10^{11}$. These ratios compare analytic estimates; they are not measured running-time ratios. The construction gives an explicit logical-space tradeoff, with large quantum and classical time costs. Appendix E.1 records the scope of each construction.

Comparison across curve families. For bounded-cofactor genus-two groups, the generic classical discrete-logarithm work factor is $2^{n+O(1)}$. For bounded-cofactor elliptic-curve groups over an n_E -bit prime field, it is $2^{n_E/2+O(1)}$. An approximate comparison at equal generic security therefore uses $n_E \simeq 2n$. This relation aligns the parameters. It does not certify the security of every curve or subgroup. Mersenne-specific circuits require $q = 2^n - 1$ to be prime. Their formulas do not define implementations at arbitrary n .

6.3 Verification and scope

The accompanying exact-integer checks expand the terminal and compact polynomials. Table 4 records their exact degrees, coefficient norms, and weighted bidegrees. The checks verify the terminal square-factor identities as polynomial identities. The verification does not rely only on evaluations at finitely many points. The five compact-output norms are componentwise below the conservative bounds in (37). These checks therefore preserve the height-growth bound.

Table 4. Exact sparse-polynomial certificates. Coefficient norms are sums of absolute integer coefficients; bidegrees use weights (2, 2, 3, 3, 1) on each input tuple.

Polynomial	Monomials	Degree	Coefficient norm	Weighted bidegree
N_*	115	19	232	(12, 12)
D_*	34	18	64	(12, 12)
U'_1	743	32	7512	(20, 20)
U'_0	837	31	9816	(20, 20)
V'_1	6623	48	577524	(30, 30)
V'_0	7603	47	621964	(30, 30)
Z'	44	15	60	(10, 10)

An independent Cantor implementation compares all 1,296 and 4,356 ordered reduced input pairs on the specified curves over $\mathbb{F}_5$ and $\mathbb{F}_7$. It also compares 2,000 sampled pairs over each of $\mathbb{F}_{11}$ and $\mathbb{F}_{13}$. All generic outputs agree, including independently scaled inputs. The shared-expression certificate records the eight-word local workspace. It also records the separate node, terminal, and scalar-correction costs. Appendix D.3 describes the verification scope. The accompanying source archive supplies the runnable checks.

A complete small-field example. We use the curve $y^2 = x^5 + 2x^3 + x^2 + 2x + 1$ over $\mathbb{F}_3$. The Mumford pair $P = (x^2, x + 1)$ generates a subgroup of order 11. We set $Q = [2]P$ and use a two-bit window with four leaves. The program `check_complete_sampler.py` checks all 121 inputs, 121 group-seed pairs, and three field shifts. These choices give 43,923 phase cases. The check connects the masked tables, integer tree, streamed CRT, clean Jacobi phase, measured cleanup, and Fourier output. The averaged useful-sample probabilities are 256/363 for the ideal translated phase and 102544/5314683 for the polynomial implementation. The general bound has $\varepsilon_{\text{tree}} = 1$ and gives a zero success lower bound at this size. The example therefore checks integration. Integer and rational checks are exact. Numerical Fourier checks use tolerance 2×10^{-12} .

Together, these checks validate the stated polynomial identities, local coherent interfaces, and analytic resource counts. The complete cryptographic-size sampler still requires compilation. The arithmetic channel bound applies to seed-independent inputs. The Legendre sampling theorem uses the exact uniform domain. The large Toffoli bound and classical tables are the main costs of the resulting logical-space reduction.

7 Conclusion

Two public group seeds make the two children at each node independent and uniform over a finite abelian group of odd order. This result applies to a complete balanced addition tree. It gives a bound on exceptional arithmetic that holds uniformly over inputs. It also gives a comparison of randomized quantum channels with the seeds retained. The determinant criterion identifies the other tree shapes for which the construction works.

For the stated genus-two curve model, the reduction uses an integer lift of weighted Mumford arithmetic and a shifted Legendre phase. The resulting useful-sample probability on the exact domain is $1 - O(L/q)$ when $r = \Theta(q^2)$. Height certificates, streaming reconstruction, shared evaluation, and clean scalar phase correction give a complete sampler allocation of $10n + o(n)$ logical qubits. This allocation includes both scalar inputs in the measurement and reset model.

On the 127-bit instance, a 16-bit window gives an analytic allocation of 1,923 qubits. This allocation is 11.46% below the matched Huang-derived allocation and 37.22% below the Chen-derived allocation. The capped-run bound is below 2^{60} Toffoli gates. A 32-bit window uses 1,618 qubits, with a Toffoli bound of about $2^{70.43}$ and at most 2^{36} table entries. Rotation synthesis is charged separately. These estimates quantify the time and classical-storage costs of the logical-space reduction. Polynomial identities, local gate networks, register traces, and the small-field sampler check support the construction. A complete cryptographic-size compilation remains open.

The main ways to lower total cost are to reduce internal height growth and use cheaper lookup and phase-correction schedules. Special-modulus arithmetic offers another local improvement. The Mersenne rotation method of Chen et al. [7] removes the doubling/halving Toffolis on the canonical residue domain. The clean adapter proved here makes the folding approach of [25, Appendix A] usable in our circuit model. Its benefit must be assessed together with the actual CRT moduli and the extra live product words. This refinement does not change the unconditional prime-product argument or the reported generic finite allocations.

References

1. A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, *Elementary gates for quantum computation*, Physical Review A **52** (1995), 3457–3467. doi:10.1103/PhysRevA.52.3457.
2. S. Beauregard, *Circuit for Shor’s algorithm using $2n + 3$ qubits*, Quantum Information & Computation **3** (2003), no. 2, 175–185. doi:10.26421/QIC3.2-8. arXiv:quant-ph/0205095.
3. C. H. Bennett, *Time/space trade-offs for reversible computation*, SIAM Journal on Computing **18** (1989), no. 4, 766–776. doi:10.1137/0218053.
4. D. J. Bernstein and J. P. Sorenson, *Modular exponentiation via the explicit Chinese remainder theorem*, Mathematics of Computation **76** (2007), no. 257, 443–454. doi:10.1090/S0025-5718-06-01849-7.
5. D. G. Cantor, *Computing in the Jacobian of a hyperelliptic curve*, Mathematics of Computation **48** (1987), no. 177, 95–101.
6. J. L. Carter and M. N. Wegman, *Universal classes of hash functions*, Journal of Computer and System Sciences **18** (1979), no. 2, 143–154. doi:10.1016/0022-0000(79)90044-8.
7. C. Chen, P. Guan, Y. Huang, and F. Zhang, *Quantum circuits for hyperelliptic curve discrete logarithms over the Mersenne prime fields*, Quantum Information Processing **22** (2023), article 274. doi:10.1007/s11128-023-04017-x.
8. C. Chevnard, P.-A. Fouque, and A. Schrottenloher, *Reducing the number of qubits in quantum factoring*, in Advances in Cryptology—CRYPTO 2025, Part II, Lecture Notes in Computer Science, vol. 16001, Springer, 2025, 384–415. doi:10.1007/978-3-032-01878-6_13. Full version: ePrint 2024/222.
9. C. Chevnard, P.-A. Fouque, and A. Schrottenloher, *Reducing the number of qubits in quantum discrete logarithms on elliptic curves*, February 2026 full version, Cryptology ePrint Archive, paper 2026/280, Theorem 3 and Appendix A, pp. 14, 33–34. ePrint version history.
10. C. Chevnard, P.-A. Fouque, and A. Schrottenloher, *Reducing the number of qubits in quantum discrete logarithms on elliptic curves*, in Advances in Cryptology—EUROCRYPT 2026, Lecture Notes in Computer Science, vol. 16541, Springer, 2026, 371–401. doi:10.1007/978-3-032-25291-3_13. Full version: ePrint 2026/280, last revised 4 June 2026.
11. S.-M. Cho, A. Kim, D. Choi, B.-S. Choi, and S.-H. Seo, *Quantum modular multiplication*, IEEE Access **8** (2020), 213244–213252. doi:10.1109/ACCESS.2020.3039167.
12. S.-M. Cho, C. Lee, and S.-H. Seo, *Quantum circuit designs of efficient squaring*, Quantum Information Processing **24** (2025), article 35. doi:10.1007/s11128-025-04647-3.
13. C. Costello and K. Lauter, *Group law computations on Jacobians of hyperelliptic curves*, in Selected Areas in Cryptography—SAC 2011, Lecture Notes in Computer Science, vol. 7118, Springer, 2012, 92–117. doi:10.1007/978-3-642-28496-0_6.
14. S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton, *A new quantum ripple-carry addition circuit*, 2004. arXiv:quant-ph/0410184.
15. L. M. Donaire, G. Ortega, E. M. Garzón, E. Filatovas, and F. Orts, *Low-qubit quantum circuits for efficient integer squaring*, The Journal of Supercomputing **82** (2026), article 164. doi:10.1007/s11227-026-08255-5.
16. P. Dusart, *Estimates of some functions over primes without $R.H.$* , arXiv:1002.0442 (2010). arXiv:1002.0442.
17. M. Ekerå and J. Håstad, *Quantum algorithms for computing short discrete logarithms and factoring RSA integers*, in: Post-Quantum Cryptography (PQCrypto 2017), Lecture Notes in Computer Science **10346**, 347–363 (2017). doi:10.1007/978-3-319-59879-6_20. arXiv:1702.00249.
18. C. Gidney, *A classical-quantum adder with constant workspace and linear gates*, arXiv:2507.23079 (2025). arXiv:2507.23079v1.
19. C. Gidney, *How to factor 2048 bit RSA integers with less than a million noisy qubits*, arXiv:2505.15917 (2025). arXiv:2505.15917.
20. R. B. Griffiths and C.-S. Niu, *Semiclassical Fourier transform for quantum computation*, Physical Review Letters **76** (1996), 3228–3231. doi:10.1103/PhysRevLett.76.3228.

21. T. Häner, M. Roetteler, and K. M. Svore, *Factoring using $2n + 2$ qubits with Toffoli based modular multiplication*, Quantum Information & Computation **17** (2017), nos. 7–8, 673–684. doi:10.26421/QIC17.7-8-7. arXiv:1611.07995.
22. T. Häner, S. Jaques, M. Naehrig, M. Roetteler, and M. Soeken, *Improved quantum circuits for elliptic curve discrete logarithms*, in Post-Quantum Cryptography—PQCrypto 2020, Lecture Notes in Computer Science, vol. 12100, Springer, 2020, 425–444. arXiv:2001.09580.
23. H. Hisil and C. Costello, *Jacobian coordinates on genus 2 curves*, Journal of Cryptology **30** (2017), 572–600. doi:10.1007/s00145-016-9227-7.
24. Y. Huang, Z. Su, F. Zhang, Y. Ding, and R. Cheng, *Quantum algorithm for solving hyperelliptic curve discrete logarithm problem*, Quantum Information Processing **19** (2020), article 62. doi:10.1007/s11128-019-2562-5.
25. Y. Huang, F. Zhang, Z. Hu, Z. Zhou, and L. Qu, *Improved quantum circuits for elliptic curve discrete logarithm problems on Ed25519*, Quantum Information Processing **24** (2025), article 311. doi:10.1007/s11128-025-04916-1.
26. A. Yu. Kitaev, *Quantum measurements and the Abelian stabilizer problem*, arXiv:quant-ph/9511026 (1995). arXiv:quant-ph/9511026.
27. N. Kornerup, J. Sadun, and D. Soloveichik, *Tight bounds on the spooky pebble game: Recycling qubits with measurements*, Quantum **9** (2025), 1636. arXiv:2110.08973.
28. T. Lange, *Weighted coordinates on genus 2 hyperelliptic curves*, Cryptology ePrint Archive, Report 2002/153, 2002. ePrint 2002/153.
29. R. Lidl and H. Niederreiter, *Finite Fields*, 2nd ed., Encyclopedia of Mathematics and its Applications, vol. 20, Cambridge University Press, 1996. doi:10.1017/CBO9780511525926.
30. H. Luo, Z. Yang, J. Luo, Z. Wang, Y. Su, X. Sun, L. Li, and T. Li, *Quantum algorithm for elliptic curve discrete logarithms with space-efficient point addition*, arXiv:2607.13816, version 3, 5 September 2026. arXiv:2607.13816v3.
31. A. May and L. Schlieper, *Quantum period finding is compression robust*, IACR Transactions on Symmetric Cryptology **2022** (2022), no. 1, 183–211. Preprint: arXiv:1905.10074.
32. E. Ozbay Gurler and H. Hisil, *Complete group law for genus 2 Jacobians on Jacobian coordinates*, Journal of Cryptographic Engineering **14** (2024), 87–101. doi:10.1007/s13389-024-00351-7.
33. J. Proos and C. Zalka, *Shor’s discrete logarithm quantum algorithm for elliptic curves*, Quantum Information & Computation **3** (2003), no. 4, 317–344. arXiv:quant-ph/0301141.
34. J. Renes, P. Schwabe, B. Smith, and L. Batina. μ Kummer: Efficient hyperelliptic signatures and key exchange on microcontrollers. In *Cryptographic Hardware and Embedded Systems—CHES 2016*, LNCS 9813, pages 301–320. Springer, 2016. Author version dated 26 January 2017, <https://cryptojedi.org/papers/mukummer-20170126.pdf>.
35. M. Roetteler, M. Naehrig, K. M. Svore, and K. Lauter, *Quantum resource estimates for computing elliptic curve discrete logarithms*, in Advances in Cryptology—ASIACRYPT 2017, Part II, Lecture Notes in Computer Science, vol. 10625, Springer, 2017, 241–270. arXiv:1706.06752.
36. P. Selinger, *Quantum circuits of T -depth one*, Physical Review A **87** (2013), 042302. arXiv:1210.0974.
37. J. Shallit and J. Sorenson, *A binary algorithm for the Jacobi symbol*, ACM SIGSAM Bulletin **27** (1993), no. 1, 4–11. doi:10.1145/152379.152384.
38. P. W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM Journal on Computing **26** (1997), no. 5, 1484–1509. arXiv:quant-ph/9508027.

A Complete Arithmetic and Integer Identities

A fixed binary schedule for clean inversion. For completeness, we instantiate a fixed-length binary Euclidean schedule using standard reversible inversion methods [22]. It gives $O(n)$ workspace and $O(n^2)$ Toffoli gates. For the canonical residue $[z]_q \in \{0, \dots, q-1\}$, we write

$$H_q(z) = \frac{z + (z \bmod 2)q}{2}, \quad 0 \leq z < q.$$

This formula defines modular halving with a canonical output.

The circuit initializes $(a, b, c, d) = (x, q, 1, 0)$. It then performs $T = 2n + 1$ fixed steps. At each step, it records the selected row in two bits and updates all entries simultaneously:

condition	(a, b, c, d) after the update
$a = 0$	(a, b, c, d)
$a > 0$ even	$(a/2, b, H_q(c), d)$
a odd, $a \geq b$	$((a-b)/2, b, H_q([c-d]_q), d)$
a odd, $a < b$	$((b-a)/2, a, H_q([d-c]_q), c).$

Throughout the computation, b is positive and odd. The invariants are $a \equiv cx \pmod{q}$, $b \equiv dx \pmod{q}$, and $\gcd(a, b) = \gcd(x, q)$. Every active step either makes $a = 0$ or reduces the positive integer ab by a factor of at least two. Since initially $ab < q^2 < 2^{2n}$, termination occurs within T steps. For $x \neq 0$, the terminal state has $b = 1$. Thus the terminal coefficient satisfies $d = x^{-1} \pmod{q}$. For $x = 0$, every step leaves the state unchanged and $d = 0$.

The retained labels make each row reversible. The last two rows recover their previous integer pair as $(2a + b, b)$ and $(b, 2a + b)$. Their previous coefficient pairs are, respectively, $([2c + d]_q, d)$ and $(d, [2c + d]_q)$. Modular halving is a permutation. After halving, the circuit recovers the original parity by testing whether the output is at least $(q+1)/2$. Thus additions, comparisons, and reversible shifts implement each row with $O(n)$ gates and reusable scratch. The four evolving words and $2T$ history bits occupy $O(n)$ qubits. The circuit XORs the terminal d into the arbitrary target z . It then reverses every row and erases each label after restoring the preceding state. Finally, it clears the known initialization constants. These steps restore x and all workspace. Both asymptotic bounds follow.

A coherent binary inverse for Mersenne primes. We now specialize the fixed binary schedule to $q = 2^n - 1$ prime, with $n \geq 2$. The circuit again initializes $(a, b, c, d) = (x, q, 1, 0)$. It uses the input register itself for a and restores that register at exit. Each of $2n - 1$ rounds records two bits and performs the following operations in order:

1. Record $t = a \bmod 2$ and $h = t[a < b]$.
2. If $h = 1$, swap (a, b) and swap (c, d) .
3. If $t = 1$, set $a \leftarrow a - b$ and $c \leftarrow [c - d]_q$.
4. $a \leftarrow \text{rotr}_n(a)$, $c \leftarrow \text{rotr}_n(c)$.

The record t refers to the parity before either swap. The value of a before step 4 is always a nonnegative even integer. If $t = 0$, the integer a was already even. If $t = 1$, step 2 places the larger odd integer in a , and step 3 subtracts the smaller odd integer. Thus the first rotation is exactly integer halving. For a canonical coefficient c , the second rotation satisfies

$$\text{rotr}_n(c) = \frac{c + (c \bmod 2)(2^n - 1)}{2} = H_q(c).$$

Both rotations use only CNOT gates. The swap-based arrangement follows the general binary-EEA approach of [22, Section 4.2 and Fig. 7].

The congruence and gcd invariants from the preceding construction remain valid. The new schedule also applies H_q after a becomes zero. For $x \neq 0$, the invariant $a \equiv cx \pmod{q}$ then gives $c = 0$, so this extra operation has no effect. For $x = 0$, every round leaves $a = 0$, $b = q$, and $d = 0$. The changing coefficient c does not affect the answer. The sum of the binary lengths of a and b starts at most $2n$. Each round with $a > 0$ reduces this sum by at least one. Integer halving reduces the length of a positive even a by one. In an

odd round, the half-difference has a smaller length than the larger original integer. If a stays positive for $2n - 1$ rounds, the length sum is at most one, whereas two positive integers have a length sum of at least two. The circuit therefore ends with $a = 0$ and $d = \text{inv}_0(x)$ for every canonical input.

The circuit XORs d into the arbitrary binary target z with CNOT gates. It then reverses all rounds. Each inverse round first undoes the two rotations. It next undoes the two subtractions and the two swaps using the retained bits. It finally recomputes the comparison and parity on the restored words to clear h and t . The circuit clears the initialization constants after it restores $(a, b, c, d) = (x, q, 1, 0)$. Thus it preserves x and clears every workspace bit, including on inputs entangled with a reference.

One clean $(n + 1)$ -bit operand, one target extension bit, one carry, and one flag provide $n + 4$ scratch qubits. The circuit reuses them for the following three stages. To record h , it copies a into the operand, subtracts b , toggles h with the controls t and the sign bit, then undoes the subtraction and the copy. It borrows the extension bit as the zero high bit of b during this comparison. The difference lies in $(-2^n, 2^n)$, so its high bit records $a < b$. The two ripple operations and one Toffoli cost $4n - 1$. To subtract tb from the extended integer a , the circuit loads and unloads tb with $2n$ Toffolis. The inverse ripple operation costs $2n - 1$. The extension bit returns to zero because $t = 1$ implies $a \geq b$ after the swap. The controlled canonical subtraction on c costs $16n - 5$, by Appendix D.1. The two controlled word swaps cost $2n$ Toffolis. Recording t and performing the two rotations require no Toffolis. Hence one round costs at most

$$(4n - 1) + 2n + (4n - 1) + (16n - 5) = 26n - 7.$$

The forward and reverse passes each have $2n - 1$ rounds, so the complete gate bound is $2(2n - 1)(26n - 7) = 104n^2 - 80n + 14$. The four evolving words, target, retained bits, and scratch give the total width

$$4n + n + (4n - 2) + (n + 4) = 10n + 2.$$

These bounds use coherent reversible gates and do not require measurement or reset. They describe a width–gate tradeoff. Compared with the Luo adapter in Section 2.1, this circuit has a larger leading width coefficient and a smaller quadratic Toffoli coefficient.

Complete canonical evaluation. The Cantor formulas (5)–(6) apply to all reduced input pairs [5]. They cover common support, inverse pairs, doubling, and the identity. We compute their Bézout coefficients from

$$g = e_1 u^{(1)} + e_2 u^{(2)} = \gcd(u^{(1)}, u^{(2)}), \quad d = c_1 g + c_2 (v^{(1)} + v^{(2)}),$$

where the second extended gcd makes $d = \gcd(g, v^{(1)} + v^{(2)})$ monic. The coefficients $(a, b, c) = (c_1 e_1, c_1 e_2, c_2)$ then give the required relation. Both Mumford pairs are valid, so the displayed composition quotients are exact. The composed polynomial $\tilde{u}$ has degree at most four. The algorithm reduces $\tilde{v}$ modulo it. If $\deg \tilde{u} = 3$, then $\deg \tilde{v} \leq 2$. Thus $(f - \tilde{v}^2)/\tilde{u}$ has degree two. If $\deg \tilde{u} = 4$, this quotient has degree at most two. Its numerator is nonzero because f has odd degree. The algorithm therefore needs at most one Cantor reduction, including when the degree drops.

The circuit uses fixed coefficient arrays, degree tags, and coherent activity flags. It pads the two Euclidean algorithms and all divisions to their constant maximum lengths. It normalizes nonzero leading coefficients with the clean inversion circuit in Section 2.1. Inactive branches use $\text{inv}_0(0) = 0$. The circuit copies the canonical result and reverses every intermediate computation, including the flags [3]. This construction gives the complete $O(n)$ -qubit, $O(n^2)$ -Toffoli reference described in Section 2.2. Its finite constants need not match those of the generic polynomial map.

Integer equations of the weighted chart. For the model (1), we define

$$\begin{aligned} R_1(a, b) &= a^4 - 3a^2b + b^2 + c_3(a^2 - b) - c_2a + c_1, \\ R_0(a, b) &= a^3b - 2ab^2 + c_3ab - c_2b + c_0, \quad \mathcal{R}_j = Z^8 R_j(U_1/Z^2, U_0/Z^2). \end{aligned}$$

We fix integer lifts of the curve coefficients. The expressions above are then integer polynomials. For $Z \neq 0$, the degree-two validity equations are

$$2V_1V_0Z^2 - U_1V_1^2 = \mathcal{R}_1, \quad V_0^2Z^2 - U_0V_1^2 = \mathcal{R}_0. \tag{65}$$

The remainder of v^2 modulo u has coefficients $2v_1v_0 - u_1v_1^2$ and $v_0^2 - u_0v_1^2$. We substitute (14) and multiply by Z^8 to obtain (65). Every term has weighted degree eight. The degree-one tag sets $U_1 = V_1 = 0$ and imposes

$$V_0^2 Z^4 = c_0 Z^{10} - c_1 U_0 Z^8 + c_2 U_0^2 Z^6 - c_3 U_0^3 Z^4 - U_0^5.$$

The degree-zero tag represents $(1, 0)$ and may use $(0, 0, 0, 0, 1)$. These tagged charts do not define a complete projective model at $Z = 0$. We make no claim that polynomial evaluation there represents a group operation.

A denominator-clearing certificate for the terminal. Proposition 4.3 derives the generic addition and terminal formulas by interpolation and Cantor reduction. The integer identities below give a separate check of the smaller terminal formula. They relate it to coefficient comparison with a common interpolation denominator. We use the quantities in (25)–(26). We set $\Sigma U_0 = U_0^{(2)} Z_1^2 + U_0^{(1)} Z_2^2$ and define

$$\begin{aligned} j &= U_1^{(2)} Z_1^2 \mathcal{E}_* + Z_2^2 (Z_1^2 \mathcal{L}_* + U_1^{(1)} \mathcal{E}_*), \\ C_2 &= 2j, \quad C_3 = 2A_{\text{sc}} \mathcal{E}_*, \quad \mathcal{W} = -2K A_{\text{sc}} \mathcal{R}_*, \\ \Lambda &= \mathcal{W} Z_1^4, \quad \mathcal{B}_2 = C_2 Z_1^4, \quad \mathcal{B}_3 = C_3 Z_1^4, \\ \mathcal{B}_1 &= V_1^{(1)} \mathcal{W} Z_1 + C_2 U_1^{(1)} Z_1^2 - C_3 ((U_1^{(1)})^2 - U_0^{(1)} Z_1^2), \\ a_5 &= 2\mathcal{B}_2 \mathcal{B}_3 - \Lambda^2, \quad a_4 = \mathcal{B}_2^2 + 2\mathcal{B}_1 \mathcal{B}_3, \\ N_\alpha &= A_{\text{sc}} a_5 - \mathcal{B}_3^2 \Sigma U_1, \quad D_\beta = A_{\text{sc}}^2 \mathcal{B}_3^2, \\ N_\beta &= A_{\text{sc}}^2 a_4 - A_{\text{sc}} \mathcal{B}_3^2 (\Sigma U_0 + U_1^{(1)} U_1^{(2)}) - \Sigma U_1 N_\alpha. \end{aligned}$$

On the generic branch, $\mathcal{B}_2/\Lambda$ and $\mathcal{B}_3/\Lambda$ are the two highest interpolation coefficients. Comparison of the x^5, x^4 coefficients gives $u_0^{(3)} = N_\beta/D_\beta$. We substitute these expressions and collect the factors. This gives the following identities over $\mathbb{Z}$:

$$N_\beta = (2Z_1^7 Z_2^3)^2 N_*, \quad D_\beta = (2Z_1^7 Z_2^3)^2 D_*. \quad (66)$$

These identities hold for every integer input tuple. They need no genericity assumption. The executable sparse certificate checks that all coefficients agree. Their interpretation as group operations requires $Z_1 Z_2 \mathcal{R}_* \mathcal{E}_* \neq 0$. Under this condition, $D_* = (Z_1 Z_2 \mathcal{E}_*)^2$ is a nonzero square. This remains true when the shifted phase argument is zero.

B Sampling Reductions and Their Scope

Theorem B.1 (General masked reduction). *We assume that a finite abelian group G and an ordered addition tree have the uniform child-pair property of Proposition 5.2. The quantity δ_v bounds the exceptional-pair density at node v . We also assume that a total clean diagonal implementation agrees, outside these exceptions, with $U_{R,\lambda}|x\rangle = g_\lambda(F(x) + T_R)|x\rangle$, where $|g_\lambda| = 1$ and λ is independent of R . The input, together with any reference system, is independent of the fresh seeds. The implemented and ideal randomized channels then satisfy $\frac{1}{2}\|\mathcal{E} - \mathcal{E}\|_\diamond \leq \min\{1, 2\sqrt{\sum_v \delta_v}\}$, even when the seeds are retained.*

Proof. The set $\mathcal{B}_R$ consists of inputs with an exceptional true child pair. Uniform child pairs and the union bound give $\Pr_R(x \in \mathcal{B}_R) \leq \sum_v \delta_v$ for every x . For $\kappa = (R, \lambda)$ of probability p_κ , we couple the dilations as follows:

$$W|\psi\rangle = \sum_{\kappa} \sqrt{p_\kappa} |\kappa\rangle_C |\kappa\rangle_E U_\kappa |\psi\rangle, \quad \widetilde{W} : U_\kappa \mapsto \widetilde{U}_\kappa.$$

We write $|\psi\rangle = \sum_x |x\rangle |\eta_x\rangle$. The dilations then satisfy $\|(\widetilde{W} - W)\psi\|^2 \leq 4\mathbb{E}_R \sum_{x \in \mathcal{B}_R} \|\eta_x\|^2 \leq 4\sum_v \delta_v$. We trace out E and retain the classical seed C . The isometry comparison proves the channel bound. A failure projector and Minkowski's inequality also give $\tilde{p}_{\text{fail}} \leq \min\{1, (\sqrt{p_0} + 2\sqrt{\sum_v \delta_v})^2\}$ for ideal failure p_0 . $\square$

For arbitrary tree shapes, the determinant criterion in Section 5.5 gives the required masks. Independent leaf masks always suffice. The theorem does not cover conditioning on a fixed seed. It also does not cover an input chosen after the seed has been observed.

Elliptic specialization. We consider a short Weierstrass curve in characteristic greater than three and an odd prime-order subgroup of order r . Generic finite addition fails on exactly $4r - 3$ ordered pairs. Of these pairs, r start with O . Each other first input D excludes $O, D, -D$ as the second input. The whole-tree exception probability is therefore at most $\min\{1, (L-1)(4r-3)/r^2\}$. For $g_\gamma(O) = 1$ and $g_\gamma(D) = s_q(x(D) + \gamma)$ otherwise, Equation (19) and the two-element x -fibres give $p_0 \leq q^{-1} + (1 - q^{-1})(2r - 1)/r^2$. The theorem gives success $1 - O(L/q)$ when $r = \Theta(q)$. This result does not provide a new finite allocation for an elliptic-curve circuit.

Zero-sum masks. The masks $R_1, \dots, R_{L-1}$ are independent and uniform, and $R_L = -\sum_{i < L} R_i$. The child sums at every proper node are then independent and uniform. The reason is that the two disjoint child sets have a nonempty complement. At the root, the pair is $(U, F(x) - U)$, where U is uniform. Independence holds after averaging over uniform $x \in \mathbb{Z}_r^2$. It generally fails for fixed x . This variant therefore lacks the pointwise channel guarantee.

Collision bounds and the comparison with CFS. We consider d output labels, a collision matrix C , and Fourier amplitudes w with $\sum_z w_z = 0$. The distinct-output condition $|C_{zz'} - 1/2| \leq \varepsilon$ implies

$$|p_{\text{bit}} - p/2| \leq \varepsilon(\|w\|_1^2 - \|w\|_2^2) \leq \varepsilon(d-1)p, \quad p = \|w\|_2^2.$$

We obtain this bound by subtracting $(I + \mathbf{1}\mathbf{1}^\top)/2$ and bounding each off-diagonal term in absolute value.

A four-input Fourier circuit shows why these entrywise collision bounds need not give $|p_{\text{bit}} - p/2| \leq \varepsilon p$. The circuit prepares the uniform state on $\mathbb{Z}_4$, evaluates the injective map

$$f(0) = 1, \quad f(1) = 3, \quad f(2) = 2, \quad f(3) = 4,$$

and applies $\mathcal{F}_4^{-1}$ to the input. At Fourier label 2, the amplitudes in output-label order 1, 2, 3, 4 are $w = (1, 1, -1, -1)/4$. Every nonzero Fourier label has zero summed amplitude because $\sum_{x=0}^3 e^{-2\pi i xy/4} = 0$ for $y = 1, 2, 3$. The circuit therefore satisfies the cancellation condition. For each binary hash h , the compressed circuit evaluates $h \circ f$ into a bit initialized to $|0\rangle$ and applies the same Fourier transform. We trace out the output in both circuits when computing the input probabilities.

We select uniformly among 17 indexed binary hashes. The family contains all 16 hashes on the four output labels and one extra copy of the hash with sign vector $v = (1, 1, -1, -1)$. Distinct-label collision probabilities are 8/17 or 9/17, so $\varepsilon = 1/34$. At Fourier label 2, the uncompressed and compressed probabilities are

$$p = \frac{1}{4}, \quad p_{\text{bit}} = \frac{1}{17} \left(16 \frac{p}{2} + \frac{1}{2} \right) = \frac{5}{34} > \frac{9}{68} = (1/2 + \varepsilon)p.$$

This example concerns the distinct-output interpretation of the collision hypothesis in Theorem 3 of the February 2026 full version of CFS [9, Theorem 3, pp. 14, 33–34]. The printed hypothesis includes equal outputs. Taken literally, it forces $\varepsilon \geq 1/2$. Its small- ε application therefore requires the distinct-output interpretation. Under the distinct-output interpretation used with that work's Proposition 1, the last inequality in the proof of Theorem 3 does not follow from the entrywise collision bound. The four-input example demonstrates this point within a Fourier-sampling circuit. The example does not determine the success of that work's translated-Legendre family or its ECDLP algorithm. A relative estimate independent of the number of output labels needs a stronger condition. One sufficient condition is an operator norm bound on the collision error restricted to the zero-sum subspace.

C Exact Reconstruction and Measured Tree Evaluation

The exact interface. An exact integer addition tree uses integer leaf tables of fixed dimension and polynomial maps. The tables and maps are the same for every auxiliary modulus. On every valid input, the root modulo q represents the required group sum and phase argument. The polynomials and their height bound must include all tags, exceptional branches, and conversions. This requirement is stronger than generic correctness. The masked tree evaluates a total integer polynomial on every input. Its result represents the group operation on the good set in Theorem 5.4. Separate branching modulo each auxiliary prime would not preserve the

common integer lift. In both settings, integer evaluation commutes with reduction. The guarded calculation in Section 5.4 reconstructs the certified integer, even when that integer has no group interpretation.

We assume that the leaf coordinates are bounded by $B_0 \geq 1$. We also assume that each output polynomial has degree at most an integer $\Delta \geq 1$ and coefficient norm at most $A \geq 1$. The recurrence $B_j = AB_{j-1}^\Delta$ then bounds level j . For $\Delta > 1$, this recurrence gives $\log_2 B_h \leq \Delta^h \log_2 B_0 + (\Delta^h - 1) \log_2 A / (\Delta - 1)$. For $\Delta = 1$, the bound is $\log_2 B_0 + h \log_2 A$. The weighted and coordinatewise bounds in Section 5.2 are sharper for our fixed maps.

Clean streaming reconstruction. The reconstruction in Section 5.4 preserves the scalar inputs. It clears each digit and residue before advancing to the next modulus. For the final correction, the circuit temporarily adds $2^{\tau-1}$ to A_{fp} . Its high bits control subtraction of $\eta(M \bmod q)$ from Y . The circuit then undoes the constant addition. It therefore has no separate copy of η to erase. After the character phase, the circuit undoes the correction and processes the moduli in reverse order. It recomputes each digit and subtracts its contributions from Y and A_{fp} . The odd-modulus digit procedure applies in both passes. Each call to the measured root oracle completes its own phase corrections. A fresh clean XOR call erases the oracle output. Both accumulators and all temporary registers therefore return to zero. The logical input retains the character phase.

C.1 Word allocation for the measured tree schedule

We give the finite schedule for (49). We instantiate the balanced-tree measured-pebbling schedule of [10,27] with five-word genus-two vertices, a scalar terminal, and the local phase-correction costs certified above. A pebble holds five residue words. A ghost is a classical parity record for a measured vertex. Records for the same vertex combine by XOR. The scalar inputs remain available throughout the schedule.

The procedure F_j computes a height- j subtree root and leaves ghosts below it. The procedure F_0 loads a leaf. For $j \geq 1$, the schedule computes the left root with F_{j-1} . It retains that root while computing the right root. It then evaluates the parent and erases both children by Hadamard measurement. When the schedule recomputes a vertex with an existing ghost, it applies the recorded parity to the computed vector. It clears the record before retaining the vector. This correction cancels the old phase. The vertex therefore does not hold a pebble and a ghost at the same time.

We count a clean vertex evaluation, a Hadamard erasure, or a complete ghost correction as one move. The procedure F_j uses $2^{j+2} - 3$ moves and at most $j + 2$ vector pebbles for $j \geq 1$. The procedure F_0 uses one move. At the full root, the schedule runs F_{h-1} on both children. It computes the scalar terminal, copies the result to the arbitrary external XOR target, and uncomputes the scalar. We count this combined terminal action as one move, which includes both evaluations. The schedule then erases both children. These steps use $4L - 3$ moves and leave only ghosts.

The schedule clears ghosts in decreasing order of height. For a height- j ghost with $j \geq 1$, it computes the two children with F_{j-1} . It corrects the recorded parent phase with the scalar schedule in Appendix D.2 and erases both children. These steps use $4 \cdot 2^j - 3$ moves. They create ghosts only at smaller heights. The schedule corrects a leaf ghost by lookup-parity-unlookup. Each correction starts and ends without vector pebbles. Thus, even if all records require correction, the total number of moves is

$$(4L - 3) + L + \sum_{j=1}^{h-1} \frac{L}{2^j} (4 \cdot 2^j - 3) = (4h - 2)L + 3 \leq 8(h + 1)L.$$

The peak number of vector pebbles is at most $h + 2$. The external target, eight local words, and one scalar correction word account for the additional $10b$ in (49). Each move has gate cost at most $G_* = 2 \max\{Q_w, 707A_b\}$. This bound includes the full scalar ghost cost $638C_M + 776C_S$. It also includes the combined terminal action and the leaf correction. Hadamards, recorded parity phases, and resets use no Toffoli gates.

The schedule eventually removes every ghost. The measured-erasure calculation in Section 5.3 gives uniform measurement outcomes conditional on all earlier outcomes. The transcript is independent of the input. Each correction cancels its recorded phase, including on states entangled with a reference. The complete logical action is $|x\rangle|O\rangle \mapsto |x\rangle|O \oplus R_p(x)\rangle$ with clean workspace. This statement covers general quantum

inputs and is stronger than basis-state correctness. The schedule therefore gives the interface required by both reconstruction passes.

D Implementation Certificates and Reproducibility

D.1 Modular primitives and sampling interfaces

Clean modular updates. The five-addition schedule below implements a controlled variable update modulo p with $c = gx$. The circuit uses a $(b + 1)$ -bit operand, one target extension bit, one carry, and one wrap flag. These registers require $b + 4$ scratch qubits. A ripple addition modulo 2^{b+1} uses $2b - 1$ Toffolis [14]. The circuit first applies ripple MAJ through the lowest $b - 1$ positions. It then writes the last carry into the highest target bit with one Toffoli and reverses the carry operations with UMA. The count is $2(b - 1) + 1$, including $b = 2$.

Each of the three occurrences of gx requires at most $2b$ Toffolis for loading and unloading. The constants p and fp require only X/CNOT gates. Thus the controlled and uncontrolled updates cost $16b - 5$ and $10b - 5$, respectively. If $g = x_j$, the identity $x_j^2 = x_j$ saves two Toffolis per occurrence. The total is then $16b - 11$. For cleanup, we define $w = [z + c \geq p]$ and $z' = (z + c) \bmod p$. The first tested high bit is $1 - w$. The second is $[z' < c] = w$. The final flag flip clears the record. These identities also hold for $c = 0$ and padded words. The tested differences lie in $[-p, p) \subset (-2^b, 2^b)$, so their high bit gives their sign. Here A denotes the extended target, and $f = 0$ initially. The five additions modulo 2^{b+1} are

$$\begin{aligned} 1 \quad & A \leftarrow A + c; \\ 2 \quad & A \leftarrow A - p; \quad f \oplus = \text{msb}(A); \\ 3 \quad & A \leftarrow A + fp; \\ 4 \quad & A \leftarrow A - c; \quad f \oplus = \text{msb}(A); \\ 5 \quad & A \leftarrow A + c; \quad f \oplus = 1. \end{aligned} \tag{67}$$

The target satisfies $A = z'$ after step 3. It satisfies $A = z - wp$ as a signed integer after step 4. Thus the second high bit is w , and the flag is one before the final X gate. The circuit loads and unloads every operand before it reuses that operand's source or changes its control. In particular, the circuit unloads fp before the next flag update. Each adder restores its own ripple carry.

The doubling circuit rotates the extended target left and subtracts p . It then copies the high bit into f and adds fp . The result $z' = 2z \bmod p$ has high bit zero. Since p is odd, $z'_0 = [2z \geq p]$. The update $f \oplus = 1 \oplus z'_0$ therefore clears the flag. The two additions cost $4b - 2$ Toffolis. The inverse circuit performs halving. Compensated Horner accumulation in Section 5.3 therefore uses b controlled updates and $2(b - 1)$ doublings. This gives the upper bounds $C_M = 24b^2 - 17b + 4$ and $C_S = 24b^2 - 23b + 4$. The expansion $\gamma x = \sum_j x_j (2^j \gamma \bmod p)$ gives $C_\gamma \leq b(10b - 5)$. All targets are distinct from the preserved sources. The permutation identities also hold coherently by linearity.

The initial modular-halving cost. On canonical residues the inverse doubling map is

$$H_p(y) = \begin{cases} y/2, & y \text{ even}, \\ (y + p)/2, & y \text{ odd}. \end{cases}$$

This map divides by two modulo the odd modulus. It does not require a variable field inversion. One explicit inverse circuit starts with $A = y$ in $b + 1$ bits and $f = 0$. It then performs

$$\begin{aligned} f \oplus &= 1 \oplus A_0; \quad A \leftarrow A - fp; \quad f \oplus = \text{msb}(A); \\ A &\leftarrow A + p; \quad A \leftarrow \text{rotr}_{b+1}(A). \end{aligned}$$

The arithmetic before the rotation is modulo 2^{b+1} . The circuit unloads fp before it changes f . If y is even, the subtraction of p produces a negative difference. In this case, the high-bit test clears f , and the next addition restores y . If y is odd, the first flag is zero. In this case, the two arithmetic operations leave $y + p$ before rotation. Both resulting integers are even and below $2p$. Thus the right rotation gives $H_p(y) < p$ and clears the extension bit. The two ripple operations cost $2(2b - 1) = 4b - 2$ Toffolis. The X/CNOT gates

and the fixed rotation require no Toffolis. The circuit reuses the same scratch for $b - 1$ applications. These applications therefore cost $4b^2 - 6b + 2$ gates and require only $b + 4$ scratch qubits. Equation (43) includes this cost. For example, at $(p, b) = (13, 4)$, the target 7 follows $7 \mapsto 10 \mapsto 5 \mapsto 9$. The Horner scaling restores this target because $2^3 \cdot 9 \equiv 7 \pmod{13}$. The initial three halvings cost at most $3 \cdot 14 = 42$ Toffolis. At $p = 2^b - 1$, the rotation identity used in (45) shows that the entire initial compensation requires no Toffolis.

Preparation and readout. We record the circuit allocation and error estimate for Theorem 3.3. The construction combines cyclic-shift eigenvalue measurement [26], semiclassical feedback [20], and the modular updates above. We use $m_r = \lceil \log_2 r \rceil$, the trial cap J_{prep} , and t from (13).

For a classical constant $0 \leq c_0 < r$ and quantum control d , set $c = dc_0$. Apply (67) with $p = r$ to an m_r -bit target, extended by one bit. The reusable operand word has $m_r + 1$ bits. Together with the target extension, ripple carry, and wrap flag, it requires $m_r + 4$ scratch qubits. Constants are loaded and unloaded with X/CNOT gates. Each of the five additions modulo 2^{m_r+1} uses at most $2m_r - 1$ Toffolis [14]. The flag and extension bit are clean at exit by the identities established above. Thus a controlled constant shift costs at most $5(2m_r - 1)$ Toffolis.

For each input register, apply Hadamards and measure the cleanly computed predicate $x < r$. Acceptance leaves the exact uniform state on $\mathbb{Z}_r$. After rejection, reset the register and retry. Each trial succeeds with probability $r/2^{m_r} > 1/2$. The probability that either register exhausts its trial cap is therefore at most $2^{1-J_{\text{prep}}}$. The comparator uses a subtraction of r , a copy of the high bit to the predicate flag, and the inverse subtraction. These two additions cost $2(2m_r - 1)$ Toffolis. The expected number of trials for both registers is less than four, and the capped number is at most $2J_{\text{prep}}$. This gives the stated preparation costs.

Define

$$T_r|x\rangle = |x - 1 \bmod r\rangle, \quad |\phi_j\rangle = r^{-1/2} \sum_{x=0}^{r-1} \omega_r^{jx} |x\rangle.$$

The identity $T_r|\phi_j\rangle = \omega_r^j|\phi_j\rangle$ identifies Fourier labels with eigenphases of the cyclic shift. Standard t -bit phase estimation is performed with one recycled control and semiclassical feedback [26,20]. Each controlled power is a shift by the classical constant $-2^h \bmod r$, for $0 \leq h < t$. The phase bits are retained classically. The two scalar inputs contribute $2m_r$ qubits, the shift workspace contributes $m_r + 4$, and the recycled control contributes one. Hence the total allocation is $3m_r + 5$. There are $2t$ controlled shifts, giving at most $10(2m_r - 1)t$ Toffolis.

For each register, the successive readout rounds use at most $0, 1, \dots, t - 1$ dyadic feedback rotations. Counting each individual feedback gate gives $2 \sum_{j=0}^{t-1} j = t(t - 1)$ rotations across both registers, including all semiclassical feedback. The angles in each noninitial round may instead be accumulated classically and applied as one rotation, giving at most $2(t - 1)$ rotations in total. Phase outcomes and retry counters require only classical storage; rotation synthesis is accounted for separately.

On eigenstate ϕ_j , the outcome $z \in \{0, \dots, 2^t - 1\}$ has probability

$$p_j(z) = \left| 2^{-t} \sum_{a=0}^{2^t-1} e^{2\pi i a(j/r - z/2^t)} \right|^2.$$

Decode by rounding $rz/2^t$ to the nearest integer modulo r . For positive circular distance $d_z = \text{dist}(2^t j/r, z)$, the geometric-sum estimate gives $p_j(z) \leq 1/(4d_z^2)$. An incorrect label requires $d_z \geq D := 2^{t-1}/r$. The precision in (13) ensures $D \geq 2 + 1/\varepsilon_F > 2$. Summing the two circular tails yields

$$\Pr(\text{incorrect label}) \leq \frac{1}{2} \sum_{a=\lfloor D \rfloor}^{\infty} a^{-2} \leq \frac{1}{2(D-2)} \leq \frac{\varepsilon_F}{2}.$$

The measurement effects are diagonal in the Fourier basis. Thus, for an arbitrary joint state of the two scalar registers, including an entangled state, decoding applies a classical noisy channel to the ideal joint Fourier labels. Coupling each decoded label to its ideal value and taking a union bound gives total variation distance at most ε_F .

Finite Jacobi decomposition. The binary Jacobi method [37,10] gives the clean completed-character phase in (31). The following fixed schedule and gate decomposition justify (32) and (33).

The circuit initializes $(a, b, j) = (Y, q, 0)$, with a occupying the input register. For odd $d > 0$, we define $e_2(d) = [d \bmod 8 \in \{3, 5\}]$. We also define $e_{\text{rec}}(a, b) = [a \equiv b \equiv 3 \pmod{4}]$. The circuit executes $T = 2n + 1$ macros. Each macro records a two-bit branch code and performs the following update. The sign correction uses the old operands.

Condition	New (a, b)	$j \leftarrow j \oplus \delta$
$a = 0$	(a, b)	$\delta = 0$
$a > 0$ even	$(a/2, b)$	$\delta = e_2(b)$
a odd, $a \geq b$	$((a - b)/2, b)$	$\delta = e_2(b)$
a odd, $a < b$	$((b - a)/2, a)$	$\delta = e_{\text{rec}}(a, b) \oplus e_2(a)$

The denominator b stays positive and odd. Whenever the new first operand is positive, the product ab decreases by at least a factor two. Since initially $ab < 2^{2n}$, the fixed schedule terminates at $a = 0$. Quadratic reciprocity and the supplementary law for 2 give

$$\left(\frac{a}{b}\right) = (-1)^\delta \left(\frac{a'}{b'}\right)$$

on every active branch, with $(0/1) = 1$. For $Y > 0$, the preserved gcd is one. The terminal values therefore satisfy $b = 1$ and $(-1)^j = \chi_q(Y)$. For $Y = 0$, all macros are stationary and $j = 0$.

The recorded updates have reversible extensions to all bit strings. The circuit halves an even word by a cyclic right rotation. On the third branch, it subtracts b modulo 2^n and rotates the word. On the fourth branch, it subtracts a from b , exchanges the words, and rotates the first word. The legal differences are nonnegative and even. The inverse macro restores the old operands and erases its sign correction. It also recomputes the branch code by XOR to clear the record. The circuit applies Z to j after the forward schedule. It then reverses every macro and clears the initialization of b .

The two operands use $2n$ bits and the trace uses $2T = 4n + 2$ bits. A sign bit, two comparison flags, a ripple carry, a branch-enable bit, and one decomposition bit suffice for the remaining operations. Zero tests use a dirty-ancilla multiply-controlled NOT. These tests borrow $n - 2$ bits from the other operand and restore them exactly. They therefore require no additional linear workspace [1, Lemma 7.2]. Ripple comparisons and controlled arithmetic use $O(n)$ Toffolis per macro and constant clean scratch [14]. The allocation is $6n + 8$, which is below the stated conservative allowance. The $O(n)$ forward and reverse macros give the gate bound.

For the finite gate decomposition, we encode the stationary, halving, subtraction, and exchange branches by 00, 01, 10, 11. We define $z = [a = 0]$ and $\ell = [a < b]$. The code bits are a_0 and $(\neg a_0 \wedge \neg z) \oplus (a_0 \wedge \ell)$. The circuit computes these flags and copies the code. It then erases the flags before it changes the operands. The allocation consists of $2n$ operand bits, $4n + 2$ record bits, one sign bit, two flags, one carry, one enable, and one decomposition bit. The total is $6n + 8$. A zero test uses $4n - 8$ Toffolis for $n \geq 3$. It borrows and restores $n - 2$ bits of the other operand [1, Lemma 7.2]. A comparator costs $2n - 1$ Toffolis. A ripple subtraction controlled gate by gate costs at most $3(2n - 3) + (5n - 7) = 11n - 16$ [14]. Hence one forward macro costs

$$\underbrace{12n - 18}_{\text{flags}} + \underbrace{11}_{\text{code/sign}} + \underbrace{22n - 32}_{\text{subtractions}} + \underbrace{2n - 1}_{\text{exchange/rotation}} + \underbrace{6}_{\text{enables}} = 36n - 34.$$

For $n = 2$, a controlled adder costs five Toffolis. At this width, a four-Toffoli comparator stores $(1 - a_0)b_0$ in a carry. It XORs $(1 - a_1)b_1$ and $(1 - (a_1 \oplus b_1))c$ into its target and then erases the carry. With a one-Toffoli zero test, the macro costs at most $10 + 11 + 19 = 40 \leq 36n$. Missing high bits in the sign rule are zero. The forward and reverse schedules each use $2n + 1$ macros. They give $C_\chi \leq 72n(2n + 1)$ with $W_\chi \leq 6n + 12$. The main text uses these bounds.

D.2 Eight-word evaluation and scalar phase correction

Here $(a, c, s, u, z_1, b, d, t, v, z_2)$ denotes the two input tuples. The four cache names are $R = \mathcal{R}_*$, $E = \mathcal{E}_*$, $L = \mathcal{L}_*$, and $Q = \mathcal{K}^2$. These local names refer to (26). In particular, the local name for the third cache does not denote the tree's leaf count.

Reproducible association rule. The quantity $\rho(e)$ counts temporary words beyond an accumulation target. The vector $\mathbf{c}(e) = (M, S)$ counts its multiplication and squaring calls. An available source has rank and cost zero. A sum uses the maximum rank and the sum of the costs. We set $\epsilon_e = 0$ for a source and 1 otherwise. To accumulate a product, the circuit computes its operands that are not available sources. It then multiplies into the target and clears those operands. Thus

$$\begin{aligned}\rho(e f) &= \min\{\max(\epsilon_e + \epsilon_e \rho(e), \epsilon_e + \epsilon_f + \epsilon_f \rho(f)), \\ &\quad \max(\epsilon_f + \epsilon_f \rho(f), \epsilon_e + \epsilon_f + \epsilon_e \rho(e))\}, \\ \mathbf{c}(e f) &= 2\epsilon_e \mathbf{c}(e) + 2\epsilon_f \mathbf{c}(f) + (1, 0).\end{aligned}$$

A square has rank $\epsilon_e(1 + \rho(e))$ and cost $2\epsilon_e \mathbf{c}(e) + (0, 1)$. For an integer coefficient, the circuit repeats only the final primitive with the coefficient's sign. It then clears the operands. These recurrences include every reversal.

Caches and associations. The circuit computes $\Delta U_0, \Delta U_1, \Delta V_0, \Delta V_1, C_*$ in five words. Their (M, S) costs are $(2, 4), (2, 4), (6, 8), (6, 8), (2, 0)$, respectively. Their extra ranks are 1, 1, 2, 2, 0. The circuit computes R, E, L in the remaining three words at cost $(5, 1)$. It erases C_* , the two ΔU_i , and then the two ΔV_i . Each reversal uses a schedule compiled for the scratch that is free at that step. The circuit then computes $Q = (z_1 z_2)^2$ at cost $(2, 1)$. Preparation costs $(43, 50)$, peaks at eight words, and leaves four caches. Direct reversal of the initial preparation could overwrite a live cache. The stated erasure order avoids this problem.

The following terms of N_* specify the order of multiplication. The ranks exclude the target and the four caches. The costs include cleanup.

term	ρ	M	S
$Q(L^2)$	1	1	2
$2(z_2^2)(a(EL))$	2	8	2
$(ab)(E^2)$	2	3	2
$-(z_1^2)(d(E^2))$	2	3	6
$(z_2^2)(c(E^2))$	2	3	6
$-2(z_2(z_2^2))(s(RE))$	3	10	4
$(z_1^2)((bQ)(R^2))$	3	7	6
$(z_2^2)((aQ)(R^2))$	3	7	6
total	3	42	34

We expand $B_1 = (z_2(z_2^2))(sR) - (z_2^2)(aL) - a(bE) - (z_2^2)(cE)$ and $B_0 = (z_2(z_2^2))(uR) - (z_2^2)(cL) - c(bE)$ as the displayed sums. Their costs are $(\rho, M, S) = (2, 14, 8)$ and $(2, 11, 6)$, respectively. The displayed A_* has $(2, 10, 14)$, and $T_* = LE - Q(R^2)$ has $(1, 2, 2)$. For any additional cached word W , the output associations are

expression	ρ	M	S
$(R^2)W$	1	1	2
$(WT_*)(R^2)$	2	3	2
$(R^2)(W(E^2))$	2	3	6
$(W(R^2))(E(E^2))$	3	5	8
$(z_1 z_2)(RE)$	2	5	0

These rules give all stages of Proposition 5.1. The circuit first caches and clears A_*, T_* , then N_*, T_* , and finally B_1 and B_0 separately. With N_* cached, the T_* word and two product temporaries bring the allocation to eight local words. The circuit reads no vector output as a source. For the terminal, it accumulates N_* directly and computes $D_* = Q(E^2)$ in a fifth word. It then accumulates γD_* and clears the fifth word. Cache preparation and reversal are included in the totals. These totals are $274C_M + 262C_S$ for the vector and $130C_M + 138C_S + C_\gamma$ for the terminal.

Independent scalar correction. The scalar-correction circuit retains the four caches and uses one initially zero scalar target. For each coordinate, it computes the completed value, applies the recorded binary parity, and

reverses the computation. The forward costs are

coordinate	U'_1	U'_0	V'_1	V'_0	Z'
maximum local words	7	7	8	8	6
M	21	56	76	118	5
S	30	78	136	94	0

The circuit uses the factored expression for U'_1 . For U'_0, V'_1, Z' , the association search expands the expressions in the original coordinates and the four cache sources. It collects monomials and lists all nonempty bipartitions of each monomial's distinct variable-power factors. The search applies the recurrence above. It minimizes rank first and total cost second. For a power, the search also lists all splits into smaller positive powers. It includes the half-power square when the power is even. For V'_0 , the circuit caches N_*, T_* , accumulates their contribution, and clears them. It then caches B_0 , accumulates its contribution, and clears it. This procedure defines a finite association certificate. The accompanying code supplies the full traces. The scalar values have no garbage when the circuit applies their parities. The sum of the costs, including reversals, is

$$(86, 100) + 2(276, 338) = (638, 776),$$

and this equality certifies the constant 707 in $G_* = 2 \max\{Q_w, 707A_b\}$. Twice the shared vector cost would not certify this interface.

Reproducing the complete example. The command `python checks/check_complete_sampler.py` runs the example in Section 6.3. The exhaustive integer bound permits 25 auxiliary primes from 5 to 103, with $M > 4B$. The fixed-point register uses 14 bits. A separate word-trace witness runs two fresh residue-XOR calls on 121 logical inputs entangled with a two-dimensional reference. Both calls clear every ghost record. The fresh second call restores the arbitrary XOR targets. An explicit Hadamard-basis calculation checks all 32,768 outcomes of one vector erasure. Each outcome has probability $1/32,768$. The word-trace ghost phases restore the conditional reference state within 2×10^{-12} . The example checks interfaces and modular word traces; it does not compile the whole sampler into elementary gates.

D.3 Height, parameters, and verification scope

We reproduce the exact sparse expansions in Table 4 directly from (27) and (36). The checker represents each polynomial by a dictionary of integer coefficients. It adds dictionaries term by term and multiplies them by convolution. It then combines equal exponent vectors and removes zero coefficients. The checker computes $\|P\|_1 = \sum_{\alpha} |c_{\alpha}|$ only after this combination. The logarithmic ceilings for $(\mathcal{A}_1, \dots, \mathcal{A}_5, N_*, D_*)$ are (13, 14, 20, 20, 6, 8, 6). The checker applies the support maximum that defines B_P in Section 5.2. It starts from $(n, n, n, n, 0)$. The following exact integer certificate results at $n = 127$:

L	$B_{N_*}(\mathbf{v}_{h-1})$	$B_{D_*}(\mathbf{v}_{h-1})$	H_{cw}
16	7,001,372	7,000,856	7,001,373
32	140,029,734	140,029,216	140,029,735
64	2,800,596,976	2,800,596,456	2,800,596,977

The coefficient triangle inequality proves the coordinate bounds for all integer tuples, including tuples with zero scales. The leaf bounds are non-strict because $Z = 1 = 2^0$. The condition $0 \leq \gamma < 2^n$ makes the terminal bound in (42) strict. The independent height checker also verifies the CRT prime margin. It uses rational bounds for $\ln 2 = 2 \sum_{k \geq 0} ((2k+1)3^{2k+1})^{-1}$. After m terms, the remainder is at most $2 \cdot 3^{-(2m+1)} / ((2m+1)(1-1/9))$. The choice $m = 100$ suffices.

Classical parameter certificates. The Gaudry–Schost parameters come from the published instance used in μ Kummer [34]. The supplied certificates independently verify that q, r are prime, $\text{ord}(P) = r$, and $\# \text{Jac}(C')(\mathbb{F}_q) = 16r$.

The Lucas–Lehmer test certifies q . The supplied recursive Pocklington certificate certifies r . Exact arithmetic verifies square-freeness, membership, $P \neq 0_G$, and $[r]P = 0_G$. Thus $\text{ord}(P) = r$. For $s = \lfloor \sqrt{q} \rfloor + 1$,

the genus-two Weil interval is contained in $[B_-, B_+]$, where $B_{\pm} = q^2 + 6q + 1 \pm 4(q+1)s$. Exact division gives $\lceil B_-/r \rceil = \lfloor B_+/r \rfloor = 16$. The group order is divisible by r , so this identity proves the assertion. The certificate contains 35 recursively certified primes. Its leaves are verified by trial division.

The source archive supplies 28 registered Python checks and their result files. The registered `check_review_revision.py` verifies the reported rounding convention, feedback-rotation counts, and the exact field-shift average in Example 4.1. The archive also includes the MATLAB reference example. Sparse-polynomial propagation checks all vector, terminal, and scalar-correction identities. It also checks that the inputs remain unchanged and the scratch returns to zero. The modular-word tests include 120 arbitrary-target cases for each vector and terminal interface. They also include 120 recorded-phase cases. The gate-integration checker composes the explicit X/CNOT/Toffoli networks in 24 targeted cases at $(p, b) = (3, 2), (9, 5)$. These cases cover arbitrary targets, reversals, exceptional tuples, and scalar interfaces that physically omit four vector-output words. Separate programs check the Jacobi circuit, finite-field arithmetic, sampling identities, pebbling, and the exact resource ledger. We checked the MATLAB example through a Python transcription. We did not execute MATLAB. These reproducible checks supplement the proofs. They do not compile the complete sampler at cryptographic size or provide a fault-tolerant layout.

Toffoli counts alone do not determine Clifford+ T depth. The chosen decomposition [36] requires a separate schedule.

The source archive also includes `check_mersenne_revision.py` and `check_clean_fold.py`, with machine-readable results. These programs check the explicit doubling and halving flag schedules and the canonical rotations. They check compensated Horner accumulation and its inverse on arbitrary targets. They also check the folding boundary and its inverse, the sliced schoolbook integer product and its reversal, and the associated cost-polynomial identities. The programs check small domains exhaustively. They sample larger Mersenne widths, including 127 and 521, with a fixed seed. These tests operate on arithmetic words and treat ripple additions as atomic operations. They verify the stated word-level schedules. The registered `check_folded_gate_adapter.py` separately expands our complete product-and-fold accumulators into NOT, CNOT, and Toffoli gates. It checks 4,976 multiplication and squaring cases, together with 1,300 standalone folds. The checks cover arbitrary canonical targets, preserved sources, all scratch bits, and the inverse interface. They also count the actual gates in every stage. At $b = 27$, the multiplication uses 193 qubits and 6,573 Toffolis; the square uses 166 qubits and 6,465 Toffolis. These checks certify our local adapter. They do not compile the cited external multiplier or the complete residue sampler.

The registered `check_mersenne_inverse.py` builds the complete NOT/CNOT/Toffoli network from Appendix A. It exhausts canonical inputs and all binary targets for $n = 2, 3, 5$, and samples $n = 7, 13$. It also checks each round's workspace cleanup, recorded bits, and reverse action. Independent word-level checks cover larger widths, including 127 and 521. The constructed networks reproduce the finite width and Toffoli formulas. These checks concern this binary construction; they do not compile the Luo or Chen inverter.

D.4 Asymptotic success and space bounds

We give the asymptotic consequences in Theorem 6.1. The relations $\sum_a m_a = r - e$ and $m_a \leq 4q$ imply

$$1 - V_G \leq \frac{1}{q} + \frac{e^2}{r^2} + \frac{4q}{r} = O(q^{-1}) \quad (r = \Theta(q^2)).$$

The tree error also satisfies $\varepsilon_{\text{tree}} = O(L/q)$. For $a, b \geq 0$, the inequality $(\sqrt{a} + 2\sqrt{b})^2 \leq 2a + 8b$ holds. We therefore square (58) to obtain $1 - \tilde{p}_{\text{use}} = O(L/q)$. We then add the preparation, Fourier, and synthesis losses as in (59).

The weighted certificate also bounds the coordinatewise certificate. We set $\omega = (2, 2, 3, 3, 1)$. The five coefficient-log ceilings $(13, 14, 20, 20, 6)$ are componentwise at most 8ω . Weighted bihomogeneity therefore gives $v_{j,i} \leq \omega_i t_j / 2$ by induction. The induction starts from $\mathbf{v}_0 = (n, n, n, n, 0)$. The terminal ceilings 8, 6 and bidegrees (12, 12) give

$$H_{\text{cw}} \leq 1 + \max\{8, n + 6\} + 12t_{h-1} \leq n + 10 + 12t_{h-1} = H.$$

Thus, for $w = \Theta(\log_2 n)$, we have $L = O(n/\log_2 n)$, $h = O(\log_2 n)$, and $H_{\text{cw}} = O(nL^{\log_2 20})$. We choose all odd auxiliary primes up to 2^b , excluding q . We take $b = O(\log_2(H+n) + w) = O(\log_2 n)$ large enough to satisfy

the prime-product estimate in Section 6.1 and $w - 1 \leq 8b$. Then K is polynomial in n , and $a_{\text{fp}} = O(\log_2 n)$. The four stage allocations in (61) are, respectively,

$$5n + O(\log_2^2 n), \quad 6n + O(\log_2 n), \quad 10n + O(\log_2 n), \quad 6n + O(1).$$

Their maximum proves $W_{\text{sample}} \leq 10n + o(n)$. The $L2^w$ table entries, the auxiliary-prime list, the integer constants, and the finite evaluation schedules all have polynomial size under this choice. The polynomial exponents need not be small. This asymptotic statement does not change the finite Toffoli estimate. Preparation, readout, and rotation synthesis also depend on their stated accuracy budgets.

E Comparison Details and Notation

E.1 The reported Chen arithmetic inventory

For $q = 2^n - 1$ prime, the following table reproduces the local primitive costs of [7, Table 1]. Here $\ell_n = \lceil \log_2 n \rceil$. The table uses the source’s named primitives. Extra controls require separate accounting. The peak width of sequential operations depends on which registers are live at the same time. It is not the sum of the rows.

Primitive	Total qubits	Ancillas	Toffolis
Addition/subtraction	$3n + 1$	$n + 1$	$6n - 4$
Constant addition/subtraction	$3n + 1$	$2n + 1$	$6n - 4$
Doubling/halving	$n + 1$	1	0
Multiplication	$4n + 2$	$2n + 2$	$n(6n - 4)$
Constant multiplication	$4n + 2$	$3n + 2$	$n(6n - 4)$
Squaring	$3n + 3$	$2n + 3$	$n(6n - 4)$
Inversion	$7n + 2\ell_n + 9$	$5n + 2\ell_n + 9$	$2n(8n - 3)$

Ancillas versus clean scratch. The column “Ancillas” above retains the source’s terminology. It does not mean A_{prim} from Section 5.3. For example, the multiplication row gives $(4n + 2) - (2n + 2) = 2n$, which is the width of its two inputs. Its out-of-place interface also has an n -bit result. The squaring row similarly gives $(3n + 3) - (2n + 3) = n$. Thus these entries cannot be counted as additional clean scratch if all input and output words are also counted separately. A product output is nonzero in general at exit. Our convention therefore excludes it from scratch. The prose definition of ancillas in [7, Section 5.1] does not by itself resolve this interface distinction. Our multiplication allocation is $3n + (n + 4) = 4n + 4$ when $b = n$. This allocation excludes the target from scratch. The reported Chen total is $4n + 2$. This arithmetic comparison does not establish equivalent interfaces or an n -qubit saving. A sequential peak-width bound still requires a proof of which registers are live at each step.

Canonical zero and controls. The end-around-carry idea in [7, Section 3.1, Fig. 2] uses $2^n \equiv 1 \pmod{q}$. A literal implementation adds the overflow back to the low word. It leaves the all-one word when $x + y = q$ and there is no binary overflow. One example is $q = 7, x = 3, y = 4$. Similarly, bitwise complementation represents modular negation on nonzero residues but sends zero to the all-one word. These observations concern the representation contract. The word q represents zero modulo q , but it lies outside our canonical set $\{0, \dots, q - 1\}$. A direct replacement must certify zero normalization, coherent cleanup, and any additional controls. We therefore retain the source’s $6n - 4$ addition entry as reported. Our Mersenne specialization uses the independently proved canonical $10n - 5$ adder. This local interface check alone gives no conclusion about the source’s complete implementation.

Each outer transform uses $(A, M, S, I) = (40, 42, 2, 4)$. The middle transform uses $(5, 10, 4, 6)$. The total is $(85, 94, 8, 14)$, which gives

$$85(6n - 4) + 102n(6n - 4) + 28n(8n - 3) = 836n^2 + 18n - 340.$$

The nominal $4n$ additions give $3344n^3 + 72n^2 - 1360n$. The source prints the quadratic term with a minus sign. Our actual-subgroup comparison instead uses $2m_r$ additions. This calculation corrects the expansion of the published inventory. It does not provide a new gate synthesis. We also retain the reported width $20n + 2\ell_n + 10$. The source labels its inversion quantity as total width in the table and as auxiliary space in the allocation discussion. Any subtraction of registers requires a new proof of their live ranges. The comparison retains the source’s generic arithmetic domain.

The reported Huang arithmetic inventory. Table 1 of [24] gives 26 multiplications, 5 squarings, 5 inversions, and 28 additions for one divisor addition. We apply the approximate primitive costs in its Section 5.1 to the actual $2m_r$ additions. The arithmetic gate estimate is

$$2m_r \left[31(16n^2 \log_2 n - 26.3n^2) + 160n^2 \log_2 n + 28(16n \log_2 n - 26.9n) \right].$$

This estimate is approximately 3.055×10^{10} Toffoli gates for the same instance. It covers the reported divisor-addition sequence. The preparation, Fourier, and rotation costs are separate.

Section 4.1 of [24] treats exceptional inputs by a random initial translation and approximate bounds on invalid scalar pairs. Our construction uses masks at every tree level. Each true child pair is independent and uniform for every fixed input, and Theorem 5.4 bounds the resulting arithmetic error. The table retains the generic-addition domain and the reported primitive costs of the earlier construction.

When $2m_r = 4n$, the preceding inventory expands to $2624n^3 \log_2 n - 3261.2n^3 + 1792n^2 \log_2 n - 3012.8n^2$. The last line of [24, Eq. (2)] prints $-2209.2n^3$ instead; our comparison uses the operation sum in the first line of that equation.

Scope of the constructions.

Construction	Established scope
Canonical group circuit	Complete reversible base-field arithmetic on every divisor class.
Generic terminal	Exact polynomial and local reversible costs on the stated generic domain.
Masked residue sampler	Total integer program, seed-averaged channel bound, and exact-domain sampling; large gate and table costs remain.
Exact residue interface	Guarded integer reconstruction is unconditional; an everywhere-correct group polynomial program is a stronger requirement.
Huang and Chen comparators	Reported generic arithmetic allocations and operation inventories; no optimality or verified-success claim is added.
Mersenne specialization	Canonical rotations and clean folding for the active modulus; integer-product costs and CRT selection are accounted for separately.

E.2 Mersenne folding and product-interface adapters

The optimization in the Ed25519 paper's appendix. The relevant Mersenne case of [25, Appendix A, pp. 17–19, Fig. 17] is $p_{521} = 2^{521} - 1$. Its reduction computes a residue in an output word. It preserves the high and low parts of the integer product. The same algebra applies to any $p = 2^b - 1$. If $c = c_0 + 2^b c_1$, then $c \equiv c_0 + c_1 \pmod{p}$. This congruence does not assert equality of the two integers. The following construction supplies the canonical boundary convention and cleanup required by our interfaces.

Lemma E.1 (Resource certificate for canonical folding). *We assume $p = 2^b - 1$ and $b \geq 2$. For preserved words satisfying $0 \leq c_0 \leq p$ and $0 \leq c_1 < p$, a clean zero-target circuit computes*

$$|c_0, c_1, 0\rangle \mapsto |c_0, c_1, r\rangle, \quad r = c_0 + c_1 - p[c_0 + c_1 \geq p] \in \{0, \dots, p-1\},$$

with $C_{\text{fold}} \leq 10b - 5$ Toffolis and $A_{\text{fold}} \leq b + 4$ additional clean qubits, excluding the two input words and the result word. The comparison includes equality.

Proof. We set $s = c_0 + c_1 < 2p$ and $w = [s \geq p]$. The circuit copies c_0 into the initially zero result word by CNOTs. It extends this word to A with one zero high bit and initializes $f = 0$. The circuit then applies the following additions modulo 2^{b+1} :

$$\begin{aligned} 1 \quad & A \leftarrow A + c_1; \\ 2 \quad & A \leftarrow A - p; \quad f \oplus = \text{msb}(A); \\ 3 \quad & A \leftarrow A + fp; \\ 4 \quad & A \leftarrow A - c_0; \quad f \oplus = \text{msb}(A); \\ 5 \quad & A \leftarrow A + c_0; \quad f \oplus = 1. \end{aligned}$$

The first tested signed value $s - p$ lies in $[-p, p)$. Its high bit is therefore $1 - w$. Step 3 leaves $A = r$. At the second test, the target satisfies $A = r - c_0 = c_1 - wp \in [-p, p)$. Its high bit is w because $c_1 < p$. Thus the

flag is one before the final X gate and zero afterwards. The restored target is $r < p < 2^b$, so its extension bit is also zero. In particular, $(c_0, c_1) = (p, 0)$ gives the canonical zero.

The circuit uses the same $(b+1)$ -bit operand, extension bit, wrap flag, and carry as in (43). The circuit loads c_0, c_1 by CNOTs, p by X gates, and fp by CNOTs. It unloads fp before it changes f . The five ripple additions cost at most $5(2b-1)$ Toffolis. They restore their carry and operand. The reversible circuit includes the initial CNOT copy. Its complete reversal therefore clears the result on the stated image. $\square$

For a canonical product $c = xy$ with $0 \leq x, y < p$, we split the product at bit b . The parts satisfy $c_0 \leq p$ and $c_1 < p$ because $c < p^2 < 2^b p$. Thus the lemma applies. The algebraic reduction needs at most one subtraction of p . The five additions above implement the reduction and restore the scratch. The circuit preserves the product bits until it can uncompute them from x, y . The construction does not assume irreversible compression of an arbitrary $2b$ -bit word into b bits.

A fused integer-product implementation. We assume that U_{int} is an independently certified clean integer-product unitary with interface

$$|x, y, 0^{2b}, 0^{A_{\text{int}}}\rangle \mapsto |x, y, xy, 0^{A_{\text{int}}}\rangle$$

and complete Toffoli count T_{int} . The unitary preserves the sources. The scratch A_{int} excludes the sources and the $2b$ -bit product word C . The fused circuit retains C and computes a temporary residue R by Lemma E.1. It updates an arbitrary canonical z and then reverses both computations:

$$U_{\text{int}}; \quad \text{Fold}(C, R); \quad z \leftarrow z + R \bmod p; \quad \text{Fold}^\dagger; \quad U_{\text{int}}^\dagger.$$

This gives the independently derived bounds

$$\begin{aligned} C_{\text{acc}}^{\text{fold}} &\leq 2T_{\text{int}} + 2C_{\text{fold}} + C_A \leq 2T_{\text{int}} + 30b - 15, \\ A_{\text{acc}}^{\text{fold}} &\leq 3b + \max\{A_{\text{int}}, b + 4\}. \end{aligned}$$

The $3b$ temporary allocation consists of C and R . It excludes x, y, z . The integer-product, folding, and addition stages run in sequence and reuse their scratch. These formulas show the time-space tradeoff of the appendix's reduction strategy. For example, its gate bound improves on the rotation-based Horner bound when $T_{\text{int}} < 8b^2 - \frac{35}{2}b + \frac{15}{2}$. The extra live product words must also fit the global schedule. This comparison concerns certified upper bounds. It does not claim that either circuit is optimal.

A fully specified schoolbook instance. A schoolbook multiplier gives an explicit instance of the fused bound. It does not require a convolution multiplier. The circuit initializes a $2b$ -bit integer-product word $C = 0$. For $j = 0, \dots, b-1$, the circuit loads $y_j x$ into a clean $(b+1)$ -bit operand. It adds this operand modulo 2^{b+1} to the slice of C at bit positions $j, \dots, j+b$. It then unloads the operand. Before this round, the product word satisfies

$$C = x(y \bmod 2^j), \quad 0 \leq C \leq x(2^j - 1) < 2^{b+j}.$$

Thus the higher bits of C are zero, and the slice value is below 2^b . The addition of $y_j x < 2^b$ cannot overflow the slice's $b+1$ bits. The round therefore adds exactly $2^j y_j x$ to C and leaves the lower positions unchanged. Induction gives $C = xy$, including zero inputs. The final slice ends at position $2b-1$.

Each round uses at most $2b$ Toffolis to load and unload the operand. The clean ripple addition uses $2b-1$ Toffolis. The operand and one carry give

$$T_{\text{int}} \leq 4b^2 - b, \quad A_{\text{int}} \leq b + 2.$$

For a dedicated integer squaring circuit, a CNOT loads the repeated bit in $x_j x_j$. This saves two Toffolis per round and gives $T_{\text{int},S} \leq 4b^2 - 3b$ without a source-copy word. The explicit loading construction justifies the use of the same physical source for both inputs. An unspecified multiplication oracle does not justify this identification. The resulting canonical folded accumulators satisfy

$$\begin{aligned} C_M^{\text{fold, sb}}(b) &\leq 8b^2 + 28b - 15, \\ C_S^{\text{fold, sb}}(b) &\leq 8b^2 + 24b - 15, \\ A_{\text{acc}}^{\text{fold, sb}}(b) &\leq 4b + 4. \end{aligned}$$

Their total widths are at most $7b + 4$ for multiplication and $6b + 4$ for squaring. For integer $b \geq 4$, both gate bounds are smaller than the corresponding rotation-based Horner bounds. They require $3b$ more scratch. The sampler’s stated low-space allocation retains the $b + 4$ scratch cap. The folded schoolbook choice therefore requires a larger scratch term in the sampler’s peak-width formula. The residue words themselves retain the same size.

Adapting a clean zero-target product as a black box. The product interface in [25, Fig. 7, p. 9] computes a product into a zero result word. More generally, we assume that a clean unitary U_P computes $R = xy \bmod p$ with certified cost C_P and scratch A_P . The scratch excludes x, y, R . The circuit

$$U_P; \quad z \leftarrow z + R \bmod p; \quad U_P^\dagger$$

implements the arbitrary-target interface with

$$C_{\text{acc}} \leq 2C_P + C_A, \quad A_{\text{acc}} \leq b + \max\{A_P, A_A\},$$

where A_A excludes the adder’s source and target. For the clean product $U_{\text{int}}; \text{Fold}; U_{\text{int}}^\dagger$, we have $C_P \leq 2T_{\text{int}} + 10b - 5$ and $A_P \leq 2b + \max\{A_{\text{int}}, b + 4\}$. The black-box adapter would therefore cost at most $4T_{\text{int}} + 30b - 15$. The fused schedule above retains C . It saves two integer-product calls at the same stated peak width. The constant and square variants in [25, Figs. 8–9] explicitly load a constant or make a source copy. The allocation must include these words when it uses those variants. The two input wires cannot automatically share the same physical source.

Propagation to the local evaluator. The rotation-based implementation preserves the $b + 4$ scratch cap. Its costs therefore substitute directly into Proposition 5.1 and give

operation at active $p = 2^b - 1$	Toffoli upper bound
shared node	$8576b^2 - 4252b$
scalar terminal	$4298b^2 - 2173b$
scalar ghost correction	$22624b^2 - 11726b$

These bounds use the same $274M + 262S$, $130M + 138S + C_\gamma$, and $638M + 776S$ ledgers. The folded integer-product option replaces the scratch cap by $A_{\text{acc}}^{\text{fold}}$. The cap must be larger if a square or constant variant requires more scratch. We must then recompute the stage maximum in (49). A smaller multiplication count alone does not imply a smaller sampler allocation.

Relation to the reported resource tables. Table 6 of [25] reports multiplication results at p_{521} . Its separate objectives are low T -count, low depth, and low width. We use the folding idea from its Appendix A. We do not substitute its convolution-based numerical estimates for T_{int} without certifying the complete integer-product interface, the precision and synthesis model, and cleanup. A reported T -count or circuit depth does not give an ordinary-Toffoli count. A reported total qubit count does not give A_P or A_{int} .

The main Ed25519 construction uses $p = 2^{255} - 19$ and intermediate representatives modulo $2p = 2^{256} - 38$ [25, Section 3.1]. This case differs from the true Mersenne case in its appendix. On the full residue set modulo $2p$, doubling is not injective because $2(x + p) \equiv 2x \pmod{2p}$. A clean in-place doubling/halving interface therefore needs a restricted encoding or extra information that distinguishes these inputs. The odd-modulus canonical proofs above do not require that representation assumption. They apply to the actual arithmetic modulus. A Mersenne value of q or the p_{521} example does not make every auxiliary prime in the CRT schedule a Mersenne prime.

Using special moduli in an explicit CRT list. The finite allocations and total in Section 6.1 use the generic bound for every auxiliary prime. We retain the common word width b and the quantities $K_{\text{max}}, \ell_K, \tau, a_{\text{fp}}, A_b, Q_w$ defined there. Those allocations remain valid when $q = 2^{127} - 1$. The residue tree operates modulo p_i , so the form of q does not change its generic allocation. We consider an explicit list that satisfies the section’s width, count, and product certificates. These conditions are $p_i < 2^b$, $K \leq K_{\text{max}}$, distinct odd $p_i \neq q$, and $\prod_i p_i > 4 \cdot 2^{H_{\text{cw}}}$. We define $d_i = \lceil \log_2 p_i \rceil$ and set

$$(A_i, D_i) = \begin{cases} (16d_i^2 - 5d_i, 0), & p_i = 2^{d_i} - 1, \\ (A_b, 4b - 2), & \text{otherwise.} \end{cases}$$

We use $G_{*,i} = 2 \max\{Q_w, 707A_i\}$ and retain the physical b -bit allocation for lookup and local words. For a Mersenne modulus, all arithmetic primitives act on the lowest d_i bits of each word. For this modulus, they leave the high zero bits unchanged. The corresponding refinement is

$$G_R \leq \sum_{i=1}^K [16(h+1)LG_{*,i} + 40b(b+1) + \tau\{2D_i + 20(a_{\text{fp}} + 1)\} + 20b(n+1)] \\ + 40(a_{\text{fp}} + 1) + 20(\ell_K + 1)(n+1).$$

All unchanged terms remain conservative. In particular, the n -bit CRT accumulator is not an auxiliary residue word. This refinement can improve the bound for a certified list. It does not assume that sufficiently many Mersenne primes exist. A replacement of the prime list by pairwise coprime composite Mersenne moduli would require a new product-size and width certificate. The stated finite and asymptotic bounds do not use such a replacement.

E.3 Index of principal notation

The following index records the symbols used in several sections. Each temporary circuit name is defined where it is used.

Symbols	Meaning and defining location
$q, n; G, r, m_r$	Field and subgroup sizes and bit lengths, (2)–(3).
N_{in}, F	Live scalar width and group-valued map, (3), (8).
Enc, ℓ	Canonical Mumford encoding and its bit length; Section 2.2.
(U_1, U_0, V_1, V_0, Z)	Weighted tuple, with recovery (14) and scaling (15).
s_q, γ, g_γ	Completed character, field seed, and shifted phase, (18), (16).
e, m_a, V_G	Lower-degree count, coordinate fibers, and useful-sample bound, (20) and (21).
$K, \mathcal{C}_*, \mathcal{R}_*, \mathcal{E}_*, \mathcal{L}_*$	Local scale and interpolation quantities, (26).
N_*, D_*, Ψ_γ	Terminal numerator, square denominator, and integer phase argument, (27).
$A_*, T_*, B_j, \mathcal{A}$	Compact-node auxiliaries and map, (35)–(36).
$\mathcal{H}, t_j, H, H_{\text{cw}}$	Tuple height, level recurrence, and terminal bit bounds, Section 5.2.
w, s_w, L_0, L, h	Window width, number per scalar, and tree sizes, Section 5.
$a^{(j)}, b^{(j)}, A_i$	Window digits and group-valued leaves, (34).
R_i, U, V, T_R	Affine masks, two group seeds, and total translation, (54).
$\varepsilon_R, \varepsilon_{\text{tree}}$	Bad-input mass and its seed-averaged tree bound, Theorem 5.4.
$p_i, K, b; M, M_i, a_i, t_i$	Auxiliary moduli, widths, and CRT data, Section 5.4.
$\tau, A_{\text{fp}}, a_{\text{fp}}, \eta$	Precision, accumulator value and width, and rounding quotient, (51)–(53).
$C_M, C_S, C_A, C_\gamma, A_{\text{prim}}$	Modular gate and scratch bounds, (43).
$C_M^{\text{Mer}}, C_S^{\text{Mer}}, D_b^{\text{Mer}}$	Canonical Mersenne primitive costs, (45).
$C_{\text{fold}}, A_{\text{fold}}, T_{\text{int}}, A_{\text{int}}$	Clean folding and certified integer-product resources, Appendix E.2.
Q_w, A_b, G_*	Lookup, primitive, and local-move cost bounds, Section 6.1.
W_χ, C_χ	Completed-character width and gates, (32).
$W_{\text{recon}}, W_{\text{sample}}$	Reconstruction and sampler peaks, (61) in Section 6.1.
p_{run}	Full-run success, (59) in Section 5.5.