

A Complete Logical Circuit with 16.5% Fewer Toffoli Gates for 256-Bit Elliptic-Curve Discrete Logarithms

Initio Intelligence

29 September 2026

Abstract

Häner et al. recently published a complete logical quantum circuit that runs Shor’s algorithm against the 256-bit elliptic-curve discrete-logarithm problem on secp256k1. It uses about 39.3 million Toffoli gates on 1,457 logical qubits and has a rigorous logical success probability of at least 0.553. We present a complete logical circuit for the same problem, in the same setting, that uses at most 32,771,245 Toffoli gates. This is 16.5% fewer. The new circuit fits in the same 1,457 logical qubits and has a certified logical success probability of at least 0.600, with confidence at least $1 - 2^{-129}$. The gate count is not an estimate. Every component, including the table lookups, the modular multiplications and all cleanup, has been built as a circuit, executed at full size and counted gate by gate. The saving comes from a redesigned table-lookup stage and a re-implemented modular multiplication. This report states the result, then gives the complete construction, its parameters, proofs, error certificate and verification in enough detail to reproduce it.

1 The result

We have built a complete logical circuit that solves the 256-bit elliptic-curve discrete-logarithm problem on secp256k1 with 16.5% fewer Toffoli gates than the logical count reported by Häner et al. [1] (Section 2.2), using no additional qubits. A Toffoli gate is a three-qubit controlled-controlled-NOT operation; it is the expensive gate in fault-tolerant quantum computers, so the Toffoli count is the standard measure of cost. A logical qubit is an error-corrected qubit, as opposed to the many physical qubits that encode it.

To our knowledge the circuit of [1] is the most complete published resource estimate for breaking secp256k1, the elliptic curve used by Bitcoin. Building on the point-addition circuits of Schrottenloher [2], it reaches a logical circuit with about 39 million Toffoli gates on 1,457 logical qubits. The authors then compile that circuit for a trapped-ion architecture and obtain a runtime of about 26 days on roughly 20,000 physical qubits. Our circuit replaces theirs at the logical level, in exactly the same setting, and performs the same computation. Table 1 summarizes the comparison.

Table 1. Headline comparison. Toffoli counts follow the reference paper’s own logical accounting convention; Section 2.2 states how each column’s figures were obtained.

	Reference circuit [1]	This work	Change
Toffoli gates, whole algorithm	39,255,168	32,771,245	−16.5%
Toffoli gates, excluding the initial lookup	38,993,024	32,669,922	−16.2%
Arithmetic Toffolis per point addition	1,196,000	1,131,363	−5.4%
Point additions	28	26	−2
Logical qubits	1,457	1,457	unchanged
Logical success probability, lower bound	0.553	0.600	+0.047

The lookup redesign alone accounts for 12.2 of the 16.5 percentage points (Section 6.4), largely because cheaper lookups allow wider windows and two fewer point additions; the multiplication accounts for the rest.

Three properties make this a result rather than a projection.

- **Everything is implemented.** The lookups, the arithmetic, the creation and erasure of the intermediate data that the multiplications need, and the final cleanup are real circuits, connected to one another and executed at full 256-bit size.
- **Everything is counted directly.** The total is the sum of the gates in the executed circuits. No saving is estimated by subtracting from the reference paper’s numbers.
- **Correctness is certified afresh.** The new circuit fails on a slightly different set of rare inputs than the reference circuit, so the reference’s error certificate does not carry over. The multiplication’s rare failure inputs were bounded statistically on two million fresh random inputs, and every other component was bounded analytically.

Section 2 defines what is being compared and Section 3 outlines the construction. Sections 4 to 6 give it in full: the lookup stage, the modular multiplication, and the remaining arithmetic together with the complete point addition. Section 7 derives the success certificate, Section 8 summarizes the verification and Section 9 states the scope, followed by a short conclusion. Six appendices give the gate accounting (Appendix A), the certificate arithmetic (Appendix B), the comparison schedule of the multiplication (Appendix C), the verification procedures (Appendix D), explicit circuits (Appendix E), and the error sets, input densities and ideal success anchor behind the certificate (Appendix F).

2 What is being compared

2.1 The reference circuit

Shor’s algorithm for the discrete logarithm comes down to computing $[k]P + [\ell]Q$ in superposition, where P is the base point, $Q = [d]P$ is the public point, and d is the unknown logarithm. The reference circuit does this with 28 windowed point additions. A point addition adds one curve point to another; a window is a group of consecutive bits of the secret scalars, used together to pick the point to add. In each step the circuit selects a precomputed point from a table of 2^{16} entries and adds it to a running total, the *accumulator*, held in quantum registers. A first, larger table lookup replaces the first addition, and the last three additions are replaced by classical postprocessing (these cover the 48 lowest bits of the Q scalar; Section 2.5). Each point addition contains three table lookups, one modular squaring, a few modular additions and two in-place modular multiplications, which are by far the most expensive part.

2.2 The numbers we compare against

The reference paper gives its logical cost as approximately 1,196,000 Toffoli gates of arithmetic per point addition, plus $3 \cdot 2^{16}$ for the three lookups. Over 28 additions this gives $28(1,196,000 + 3 \cdot 2^{16}) = 38,993,024$; adding the initial lookup of 2^{18} Toffolis gives 39,255,168 for the whole algorithm. These two figures are our benchmarks, and all our percentages are computed against them. The reference circuit executes a different number of gates depending on its input, and the 1,196,000 figure is the largest count the authors observed in 100 sample runs: an approximate maximum, not a proved one. Our counts charge every optional operation and hold for every input.

The reference paper also reports 40,420,330 Toffoli injections for its compiled circuit. That figure charges every conditional operation and a more expensive phase correction, so it is not

comparable and we do not use it. The reference uses 1,457 logical qubits and proves a logical success lower bound of 0.553 for a single run, assuming no physical errors. Its estimates of 40.7% and 63.3% for the complete physical machine include the physical error budget and are not comparable to logical bounds.

2.3 Notation

The field is $\mathbb{F}_p$ with

$$p = 2^{256} - c, \quad c = 2^{32} + 977, \quad n = 256,$$

and the curve is $E : y^2 = x^3 + 7$ with prime group order

$$r = p + 1 - 432420386565659656852420866390673177327.$$

Brackets denote scalar multiplication modulo r ; coordinate arithmetic is modulo p . Registers are little-endian. In gate lists we write CCX for a Toffoli gate and CX for a CNOT.

Table 2 lists the symbols used with a fixed meaning, with every local reuse in parentheses.

Table 2. Notation.

Symbol	Meaning
p, c, n	field prime $p = 2^n - c$, with $c = 2^{32} + 977$ and $n = 256$
r	prime group order (in Section 5 and Appendix F, r, s are the replay coefficients)
d	the unknown logarithm, $Q = [d]P$ (in Section 2.4, a register width)
τ	coordinate offset $(c - 1)/2$
$\mathcal{E}$	outer coordinate encoding $z \mapsto z + \tau$ (Section 4.3)
E	the curve; in Section 5, the replay encoding $z \mapsto (z + \mu) \bmod p$
K	2^{18} , bound on the points excluded from the initial table
b	random scale in $\mathbb{Z}_r^*$ (in Section 5, the replay sign control $1 - a$)
e_w	least bit position of a window
N, T_j, U_m	$N = 2^w$ for a window of width w ; entries of the centered and half tables (Section 4.1)
h_{top}	top address bit of a window (Lemma 4.1)
Z	accumulator translation (Section 4.2); in Section 5.4 and Appendices E and F, the corrected output word of a fused replay step
λ	slope $(y_1 - y_2)/(x_1 - x_2)$ of the point addition
a	magnitude-address width $w - 1$ of a lookup (in Section 5, the recorder's parity decision)
m	magnitude address of a lookup (in Section 5, the recorded swap decision; in Section 2.5 and Appendix E, a register length)
B	bank count of a lookup segment (in Section 4.1, the window's weighted base point)
L	number of rows of a unary lookup
$U(L)$	Toffoli count of the pruned unary loader over L rows
b_s, o	payload bits of a segment and retained first-segment bits (o is also the orientation bit)
s	phase split of the terminal phase circuit (in Lemma 4.1, the sign bit; in Section 5, a replay coefficient)
h	recorder bad-denominator fraction (in Sections 4.4 and 4.5, the row index, written g in Appendix E, or the loader depth $\lceil \log_2 L \rceil$)
$H(n')$	least cost of windows covering n' remaining scalar bits (Section 4.6)
σ	$1 - 2a$, the sign of a recorder step

Symbol	Meaning
μ	replay encoding constant $M/2 - c$ (in Section 4.5 and Appendix E, an X-measurement outcome; in the proof of Lemma 7.2, the expected failure count)
M, Λ, Δ	$2^n, 2^\kappa$ and $2^{n-\delta}$ (Δ in Appendices D and F only)
κ, δ	correction and comparison widths of the replay step (in Section 6, the outer widths)
R, S	encoded replay coefficients $E(r), E(s)$ (R or R_i is also the field mask of a point addition)
R_b, K_b	R and $-c$ when $b = 0$; $M - 1 - R$ and $c - 1$ when $b = 1$
$q_0, q_1, \hat{q}_1$	bits of the replay quotient $q = q_0 + 2q_1$ and the truncated estimate of q_1 (in Section 7.4, q is a bad field-pair fraction)
$H_b(q)$	short replay correction $qc - K_b$
T	number of recorder steps, 401
W_i, d_i, e_i	data width, scheduled comparison width and actual comparison width $\min(W_i, d_i)$ of recorder step i (in Appendix E, d_i is also a carry)
q_{rep}	replay failure bound of Theorem 7.1
N_s, h^+	certification sample size and upper confidence bound on h
Φ, Θ, ξ	observed failure count, $\lceil 7\xi/10 \rceil$ and confidence exponent of Lemma 7.2
D_h, D_{pair}	densities transferring denominator and field-pair bounds to point pairs
$p_{\text{add}}, p_{\text{sub}}, p_{\text{sq}}$	per-window failure bounds of the outer additions, subtractions and square subtraction
ϵ	error bound for the 26 point additions
P_0, Π	ideal success probability and success projector
$x_{\text{lo}}, x_{\text{hi}}$	low and high 128-bit halves of the square-subtraction source (Section 6.2)
κ_o, δ_o, ρ (App. E, F)	outer correction width 65, outer comparison width 32, local-slice carry padding 32
α, β, γ (App. F)	fixed field coefficients with $s = \alpha y, r = \beta y$, and the updated coefficient $\gamma = 2\alpha + (-1)^b \beta$ (in Lemma 4.2, α is a fixed nonzero scalar; in Theorem 7.3, α_x are amplitudes)
Z_* (App. D, F)	ideal encoded output $E(2s + (-1)^b r)$ of a fused replay step
η_i, ϕ_i (App. E)	packed history bits and wires of the block being filled at recorder step i
ν (App. F)	residue $2^{48}ad \bmod r$ estimated by the second scalar register
$\Sigma_{\text{hi}}, C_{65}$ (App. D, E)	high carry sum of the signed- c identity and the 65-bit buffer holding $c\Sigma_{\text{hi}}$
$X_{\text{lo}}, X_{\text{hi}}$ (App. E)	low and high parts of the squarer input at iteration i

2.4 Counting conventions and what the metric excludes

A Toffoli count includes every Toffoli gate in the circuit, including those that prepare temporary ANDs. A *temporary AND* is a fresh qubit holding the AND of two bits. It is prepared with one Toffoli and removed by an X-basis measurement followed by a classically controlled CZ; every counted circuit includes this correction explicitly.

All arithmetic is built from the measured ripple adder of [3], in which each carry is a temporary majority $ab \oplus ac \oplus bc$ prepared with one Toffoli and removed by measurement; we call this carry the measured majority. The primitive costs are:

- adding two w -bit registers modulo 2^w : $w - 1$ Toffolis;
- adding with a carry-out bit: w ;
- a strict comparison of two d -bit registers: d ;
- a controlled swap of two registers: one Toffoli per bit;
- a controlled constant load: no Toffolis (constants are loaded by CNOTs from the control bit), so a controlled constant addition costs the same as the plain addition.

A CCZ gate counts as one Toffoli, since conjugating its target with Hadamards turns it into a CCX. A controlled swap of two wires u, v with control s is decomposed as $CX(u, v)$, $CCX(s, v, u)$, $CX(u, v)$ and costs one Toffoli. Plain wire permutations cost nothing. CNOT, X, Z, CZ, measurements, classical Boolean processing and classical feed-forward are not counted.

All scratch starts in $|0\rangle$. Every workspace release is physically a computational-basis measurement followed by a reset, on every input, including inputs outside the good domain (the inputs on which every component is proved correct; Section 7); on a good input the released wire is zero.

The width is the maximum number of simultaneously live algorithm qubits. It includes the point accumulator, the 18 address bits, the compressed history, carry words, lookup banks and phase-correction workspace (Sections 4 and 5). Address bits unused by a narrower window stay clean and are still counted. Classical lookup tables, measurement transcripts and any environment introduced only to analyze a reset are not qubits. No factory, error-correction or routing qubits are included.

The semiclassical Fourier transform [4] uses single-qubit rotations and measurements, which we treat as ideal and do not count; the success bound of Section 7 assumes ideal rotations. An application that synthesizes them in a particular fault-tolerant gate set must add their cost and their approximation error. The reference count [1] uses the same boundary.

2.5 The computation implemented

As in [1], the scalar qubits are measured one at a time and reused, and the scalars are consumed in windows. The ideal controlled group operation is

$$|k\rangle|\ell\rangle|\mathcal{O}\rangle \mapsto |k\rangle|\ell\rangle|[k]P + [2^{48}\ell]Q\rangle, \quad 0 \leq k < 2^{257}, \quad 0 \leq \ell < 2^{209}.$$

The P scalar keeps 257 bits. The Q scalar keeps the 209 bits of a 257-bit scalar above its 48 omitted low bits, which are recovered classically (Appendix F).

The scalar control qubits are recycled by a windowed semiclassical inverse Fourier transform. For an m -bit scalar register, the control of group power 2^{m-1-t} is retired at step $t = 0, \dots, m-1$. Let $j_{<t} = \sum_{u<t} j_u 2^u$ collect the Fourier bits measured so far. Apply $\text{diag}(1, \exp[-2\pi i j_{<t}/2^{t+1}])$, then a Hadamard, then a Z measurement, which gives j_t . The rule is applied separately for $m = 257$ and $m = 209$; in the second register every group power carries the extra factor 2^{48} . Within a completed window, controls are retired in descending power order; the two registers commute, so their windows may be interleaved. These classically controlled single-qubit operations add no Toffolis.

The window optimization of Section 4.6 selects the following partition. The initial lookup uses the highest 17 P bits and the highest Q bit. The ordinary windows then cover the remaining 240 P bits and 208 Q bits, with widths, in descending bit order,

$$P : (18, 18, \underbrace{17, \dots, 17}_{12}), \quad Q : (18, 18, 18, 18, \underbrace{17, \dots, 17}_8).$$

The base of a window is $[b2^{e_w}]P$ or $[b2^{e_w}]Q$, where b is the random scale of Section 4.2 and e_w is the least bit position in the window. The lowest retained Q position is 48. This rule fixes every table weight.

3 Overview of the construction

The new circuit keeps the reference's curve, scalar precision (257 bits per scalar, with 48 low bits of the Q scalar dropped for postprocessing), classical postprocessing, affine point representation and overall structure. It replaces two stages and re-implements everything else so that a complete point addition can be executed and counted as a single circuit.

Table lookups (Section 4). Each window's point table is made symmetric, so only half of it is stored, and the sign is applied by negating the accumulator, which costs no Toffolis in our coordinate encoding. Each 256-bit coordinate is loaded in two segments. Within a segment the banks are loaded in parallel, all but the addressed one are released by measurement, and the phases these measurements leave are cancelled once per point addition by a classically compiled phase table. The lookup stage for one 16-bit window falls from 196,608 to 55,015 Toffolis. Cheaper lookups make wider windows economical: 26 windows of 17 and 18 bits replace 28 windows of 16 bits.

Modular multiplication (Section 5). Each in-place multiplication records the decision sequence of a binary GCD on one input, replays it on the other input, and erases it. It uses a shrinking register schedule, a public schedule of comparison widths, a compressed history, and a fused replay step that performs each step's doubling and signed addition as one integer operation plus one short modular correction. The replay step costs 345 Toffolis; a complete multiplication costs 534,279. The two multiplications account for 1,068,558 of the 1,131,363 arithmetic Toffolis per point addition.

Remaining arithmetic and the point addition (Section 6). The modular additions, subtractions and the square subtraction follow the reference's style, and the whole point addition runs in one shared qubit layout with a peak of 1,457 logical qubits.

Success probability (Section 7). Replay is bounded analytically, and the recorder's rare failures are bounded statistically on two million fresh random inputs. With an improved theorem converting the error bound into a success probability, this gives a certified lower bound of 0.600.

Throughout, the binding constraint is the 1,457-qubit budget. The two-segment lookup and the mixed history layout exist to meet it exactly.

4 The lookup stage

4.1 Signed half-tables

The centered table is antisymmetric, so storing half of it suffices and the top address bit supplies the sign. For a window of width w let $N = 2^w < r$ and let $B \neq \mathcal{O}$ be the window's weighted base point (Section 2.5). Define the centered table and the half table

$$T_j = [j - (N - 1)/2]B \quad (0 \leq j < N), \quad U_m = [m + 1/2]B \quad (0 \leq m < N/2),$$

where the fractions are inverses modulo the odd order r .

Lemma 4.1. $T_{N-1-j} = -T_j$; every table point is finite; and if h_{top} is the top address bit, setting $s = 1 \oplus h_{\text{top}}$ and complementing the remaining $w - 1$ bits when $s = 1$ yields a magnitude m with $T_j = (-1)^s U_m$.

Proof. Negating $j - (N - 1)/2$ gives $(N - 1 - j) - (N - 1)/2$. Twice a table scalar is a nonzero odd integer of magnitude below $N < r$, so no table point is $\mathcal{O}$. If $h_{\text{top}} = 1$ then $j = N/2 + m$ with

centered coefficient $m + 1/2$; if $h_{\text{top}} = 0$ then $m = N/2 - 1 - j$ and the coefficient is $-(m + 1/2)$. $\square$

The address fold uses only X and CNOT gates and no extra qubit. Writing $\mathcal{N}(A) = -A$, the identity $\mathcal{N}^s \circ \text{Add}_{U_m} \circ \mathcal{N}^s(A) = A + (-1)^s U_m$ shows that the expensive affine addition never needs to be controlled on the sign. The sign acts only through two conditional negations of the accumulator.

4.2 Randomization and the pair-density bound

The reference adds an independent random offset point to each table to avoid $\mathcal{O}$ and to obtain input-distribution bounds. Independent offsets would destroy the sign symmetry above, so we randomize differently. Draw one uniform scale $b \in \mathbb{Z}_r^*$ and replace both bases by $[b]P, [b]Q$; this preserves $[b]Q = [d]([b]P)$. Conditional on b , draw the initial accumulator translation Z uniformly from the points that do not place $\mathcal{O}$ in the initial 2^{18} -entry table; at most $K = 2^{18}$ points are excluded, and only Z is resampled on rejection. Each ordinary window also uses a fresh uniform field mask $R_i \in \mathbb{F}_p$ in its square subtraction.

Lemma 4.2. Fix the scalar input and an ordinary window on the ideal trajectory (Section 7.1). Let A be the (possibly sign-conjugated) accumulator and U the unsigned table point. Then for every pair (A_*, U_*) ,

$$\Pr[(A, U) = (A_*, U_*)] \leq \frac{1}{(r-1)(r-K)},$$

and a specified independent field mask contributes a further factor $1/p$.

Proof. $U = [\alpha b]B_0$ for a fixed nonzero α and base B_0 , so a specified U_* determines b . Conditional on b , $A = \pm Z + F(b)$ for a fixed F , so a specified A_* determines Z . The two probabilities are $1/(r-1)$ and at most $1/(r-K)$. $\square$

After randomization no accumulator and table pair is much more likely than under uniform sampling, which is what the error analysis needs. Windows share b and are correlated, but the error analysis never needs independence: it defines every failure event on the unconditional ideal trajectory and charges a run at its first failure. The output of the randomized algorithm is $[b]f(x) + C_{b,Z}$ with $C_{b,Z}$ independent of the scalar label; it also absorbs the known, input-independent translation from centering the tables. An output-only relabeling removes it in the analysis and need not be executed.

4.3 Coordinate encoding and the affine pipeline

Put $\tau = (c-1)/2$ and encode a coordinate z as the integer $\mathcal{E}(z) = z + \tau$. For $1 \leq z < p$ one has $\mathcal{E}(p-z) = 2^n - 1 - \mathcal{E}(z)$, so negating a finite point (whose y is nonzero in an odd-order group) is a bitwise NOT of the encoded y . The two conditional negations of Lemma 4.1 therefore cost 512 CNOTs and no Toffolis. Infinity receives an unused codeword and is charged as an exceptional case.

For the unsigned table point $U = (x_2, y_2)$ the five stored columns are

$$(x_2 + \tau) \bmod p, \quad (y_2 + \tau) \bmod p, \quad 3x_2 \bmod p, \quad (y_2 - \tau) \bmod p, \quad (x_2 - \tau) \bmod p.$$

Each point addition therefore makes five table reads, one per column, where the reference makes three; each read is much cheaper. Starting from the encoded, sign-conjugated accumulator $A' = (x_1, y_1)$ and a fresh mask R , the point addition executes

$$\begin{aligned} x &\leftarrow x - (x_2 + \tau), & y &\leftarrow y - (y_2 + \tau), \\ y &\leftarrow y/x, & x &\leftarrow x + R, \\ x &\leftarrow x + 3x_2, & x &\leftarrow x - y^2, \\ x &\leftarrow x - R, & y &\leftarrow y \cdot x, \\ y &\leftarrow y - (y_2 - \tau), & x &\leftarrow x - (x_2 - \tau), \\ x &\leftarrow \text{NOT}(x), \end{aligned}$$

with all operations except NOT modulo p . In order:

1. the first two subtractions give the coordinate differences;
2. the division gives the slope $\lambda = (y_1 - y_2)/(x_1 - x_2)$;
3. the masked square stage gives $x_1 + 2x_2 - \lambda^2 = x_2 - x_3$;
4. the multiplication gives $y_3 + y_2$;
5. the last two subtractions give $y_3 + \tau$ and the canonical residue of $\tau - x_3$, whose complement is $x_3 + \tau = \mathcal{E}(x_3)$.

The mask is removed by adding the classical constant $-R$, so the pipeline contains three additions and four subtractions.

The final NOT is correct only for one representative of $\tau - x_3$, and an encoded value fitting in n bits need not be below p . The good domain therefore excludes affine exceptions and imposes the *encoding-range condition*: $0 < y < p - \tau$ for the incoming point, its conjugate, the unsigned sum and the final result, and $\tau < x < p - \tau$ for the incoming point and the unsigned sum. A fixed x occurs on at most two points and a fixed y on at most three, and each state is a translate or negated translate of Z , so the encoding-range condition costs at most $32(\tau + 1)/(r - K)$ per window. The affine exceptional set is contained in $\{\mathcal{O}, U, -U, -[2]U\}$ and costs at most $4/(r - K)$.

4.4 Segmented banked lookups

Let the magnitude address have $a = w - 1$ bits. Four full 256-bit banks do not fit beside the 512-bit accumulator within the qubit budget, but four slightly narrower banks do. We therefore load each coordinate in two segments: a wide first segment with few banks, whose selected bits are retained, and a narrow second segment with many banks. The supported bank counts are $B = 2^k$ and $B = 3 \cdot 2^k$.

Bank counts of the form $3 \cdot 2^k$ give a finer trade-off between qubits and Toffolis than powers of two alone; they require dividing the address by three.

For $B = 2^k$ the address is split as $m = Bh + t$ and a binary controlled-swap tree selects bank t into bank zero. For $B = 3 \cdot 2^k$ the address is first divided by three, $m = 3q + r_3$, $q = 2^k h + t$; the row holds values with offsets $3t + r_3$, the remainder selects within each triple and the binary tree selects the triple. The reversible division by three applies a six-gate cell with two Toffolis (Appendix E) to the remainder pair (r_0, r_1) and each address bit x , from most to least significant. On valid remainders the cell maps $(r_3, x) \mapsto ((2r_3 + x) \bmod 3, \lfloor (2r_3 + x)/3 \rfloor)$, and it is a permutation of all eight basis states. The round trip costs $4a$ Toffolis and two remainder wires.

Rows are addressed by a pruned unary loader: a tree of temporary ANDs over the row address that XORs the addressed row's public word into the banks. For L rows and $h = \lceil \log_2 L \rceil$ it uses $\max(0, h - 1)$ scratch wires and $U(L)$ Toffolis, given by a recurrence over the pruned tree (Appendix E, with the gates); for a power of two, $U(L) = L - 2$. A selector with B banks uses $B - 1$ controlled word swaps, each costing one Toffoli per bit. A segment of b_s payload bits therefore costs

$$C_{\text{seg}} = U(L) + (B - 1)b_s + 4a \mathbf{1}_{B=3 \cdot 2^k}, \quad L = \lceil 2^a/B \rceil,$$

and, with A_{live} other occupied qubits, s_{live} sign bits and o retained first-segment bits, its peak occupancy is $A_{\text{live}} + a + s_{\text{live}} + o + Bb_s + 2 \cdot \mathbf{1}_{B=3 \cdot 2^k} + \max(0, h - 1)$. Unselected banks, remainder wires and unary scratch are measured or uncomputed and released before the next segment.

A dynamic program over the split point and bank counts (Appendix E) selects the plans in Table 3. The first segment always holds the low bits of the word and the second the remaining high bits.

Table 3. Lookup plans. One coordinate read costs 10,927, 20,509 and 39,628 Toffolis for widths 16, 17 and 18. The width-16 plan, for comparison with the reference's 16-bit windows only, reserves 16 controls and peaks at 1,456 qubits; under the convention of Section 2.4, with all 18 controls, it would occupy 1,458 and is infeasible. The selected widths 17 and 18 reserve all 18 controls and peak at 1,455 and 1,456 qubits, including the accumulator and all scratch.

Width	Segment	Bits	Banks	Rows L	$U(L)$	Selector	Divider	Toffolis
16	low	229	4	8,192	8,190	687	0	8,877
16	high	27	24	1,366	1,369	621	60	2,050
17	low	228	4	16,384	16,382	684	0	17,066
17	high	28	24	2,731	2,735	644	64	3,443
18	low	228	4	32,768	32,766	684	0	33,450
18	high	28	24	5,462	5,466	644	68	6,178
initial	low	473	3	87,382	87,388	946	72	88,406
initial	high	39	24	10,923	10,928	897	72	11,897

The initial lookup has an 18-bit address (the top 17 bits of the P scalar and the top bit of the Q scalar) and a 512-bit payload that becomes the accumulator, so nothing is reserved beside it. Its two segments cost 100,303 Toffolis and its phase correction 1,020, giving 101,323. Its peak is 1,455. In short, a wide low segment with few banks and a narrow high segment with many banks let a 17- or 18-bit table read fit beside the accumulator within the qubit budget.

4.5 Complete phase cleanup

Releasing the unselected banks by X-basis measurement is cheaper than uncomputing them, but each measurement leaves an address-dependent sign. This section shows that all these signs can be removed together, once per point addition. An X-basis measurement of a basis-valued data bit v with outcome μ contributes the amplitude factor $2^{-1/2}(-1)^{\mu v}$, whether or not the bank was selected.

Lemma 4.3. For a read of table D with bank-selection permutation π_t and measured masks μ_ℓ , the phase on magnitude address $m = Bh + t$ is $(-1)^{F_D(m)}$ with

$$F_D(m) = \bigoplus_{\ell} \langle \mu_\ell, D_{Bh+\pi_t(\ell)}^{\text{seg}} \rangle,$$

where padded entries are zero and unmeasured banks have mask zero; segments and reads combine by XOR. The phase can be corrected once after all five reads, provided the address has not yet been Fourier-transformed or discarded.

Proof. Conditional on a basis address, the loader prepares definite bank values, the selector permutes them, and the measurement factors multiply. Unary-prefix ANDs carry their own exact local phase corrections and the divider is uncomputed, so these are all the phase sources. Every intervening operation, including the address-controlled arithmetic, has block form $\sum_m |m\rangle\langle m| \otimes U_m$, so a phase diagonal in m commutes with it. Because the address blocks are orthogonal and preserved, every measurement transcript has the same normalization $2^{-N_X/2}$ for N_X bank measurements, so the relative phase ledger is complete. $\square$

Discarding the unused banks by measurement thus leaves only an address-dependent sign, known classically from the tables and the recorded outcomes, so one circuit removes it after all five reads. Omitting the unselected banks from F_D leaves every coordinate correct and the interference wrong (this is the negative control of Section 8).

The correction implements an arbitrary Boolean sign table on a bits. It decodes the s low bits into 2^s one-hot wires and streams the $a - s$ high bits through a unary-prefix iterator. Wherever the table is one, it applies a CZ between the live prefix and the one-hot wire (Appendix E). It costs

$$C_{\text{phase}}(a, s) = 2^s + 2^{a-s} - 4$$

Toffolis with $2^s + (a - s - 1)$ scratch wires, including measured cleanup of every AND. The selected splits $s = 7, 8, 8, 9$ give 380, 508, 764 and 1,020 Toffolis for $a = 15, 16, 17, 18$. The five-read totals are therefore $5 \cdot 10,927 + 380 = 55,015$, $5 \cdot 20,509 + 508 = 103,053$ and $5 \cdot 39,628 + 764 = 198,904$. In short, each unselected bank costs one measurement, and the phase debt of all five reads is paid once, for at most a few hundred Toffolis per ordinary window.

4.6 Window optimization

Let C_w be the total cost of a width- w point addition (arithmetic plus the five-read lookup cost with cleanup). For a remaining scalar length n' solve

$$H(0) = 0, \quad H(n') = \min_{3 \leq w \leq 18, w \leq n'} \{H(n' - w) + C_w\}$$

separately for the 240 remaining P bits and 208 remaining Q bits, with unreachable states at cost $+\infty$. With all 18 control wires reserved, the five-read lookup-and-cleanup costs for widths $w = 3, \dots, 18$ are

$$(10, 32, 74, 158, 322, 650, 1298, 2594, 3890, 6302, 9769, 16683, 30422, 55115, 103053, 198904),$$

and C_w is this cost plus the 1,131,363-Toffoli arithmetic of Section 6. The recursion selects twelve 17-bit and two 18-bit windows on the P side (cost 17,473,526) and eight 17-bit and four 18-bit windows on the Q side (cost 15,196,396): twenty 17-bit and six 18-bit windows in total. The same partition is optimal with the reference arithmetic charge. (The width-16 entry, 55,115, differs from Table 3 because it reserves all 18 controls.) Induction on the final window proves optimality within the stated width range and bank family; no global circuit optimality is claimed.

5 The modular multiplication

5.1 Record, replay, erase

The in-place multiplication $|x\rangle|y\rangle \mapsto |x\rangle|xy\rangle$ follows the record-and-replay structure of [2] as modified in [1]. A binary GCD walk on the denominator x produces a sequence of decisions, the *history*; replaying that history as linear updates on a coefficient pair computes xy or y/x ; running the recorder in reverse erases the history and restores x .

We use a shifted integer walk. For a nonzero denominator x , the walk starts at $(u, v) = (x, p)$ if x is odd and $(p, p + x)$ if x is even; both registers are then odd. Each step records a symbol (a, m) with $m \Rightarrow a$ and applies, with $\sigma = 1 - 2a$,

$$v \leftarrow (v + \sigma u)/2, \quad (u, v) \leftarrow (v, u) \text{ if } m = 1.$$

The canonical decisions are $a = u[1] \oplus v[1]$ (the second-lowest bits), which makes $v + \sigma u \equiv 2 \pmod{4}$ so the halved result is odd, and $m = a \wedge \mathbf{1}[u > v_{\text{new}}]$. The state $(1, 1)$ is absorbing: it produces $(a, m) = (0, 0)$ and remains $(1, 1)$. The circuit always runs all 401 steps, padding a converged walk without a stopping test.

Lemma 5.1 (replay contract). Let any sequence of exact, nonwrapping transitions of the above form connect the initial pair to $(1, 1)$. Multiplication initializes coefficients $(r, s) = (y, y)$, traverses the history backward, and at each step swaps if $m = 1$ and then sets $s \leftarrow 2s - \sigma r \pmod{p}$. The result is $(xy, 0)$ for odd x and $(0, xy)$ for even x . Division initializes $(y, 0)$ or $(0, y)$ respectively, traverses forward, sets $s \leftarrow (s + \sigma r)/2 \pmod{p}$ and then applies the recorded swap; the result is $(y/x, y/x)$.

Proof. By induction the coefficients equal y times the reconstructed integer pair modulo p ; the initial pair is $(x, p) \equiv (x, 0)$ or $(p, p + x) \equiv (0, x)$. Division is the inverse walk. $\square$

Because only reaching $(1, 1)$ matters, an approximate comparison that makes a different but still valid choice is harmless; the certificate of Section 7.3 nevertheless counts every disagreement with the full comparison as a failure.

5.2 The recorder

The recorder computes the history from the denominator with actual gates, compressing it as it goes, so that no second copy of the denominator exists during replay. Its frozen configuration has $T = 401$ steps. The data width of step i is

$$W_i = \max\left\{2, \min\left\{257, \left\lceil 256 - \frac{1000i}{1413} + 36.8 \right\rceil + 1\right\}\right\}, \quad 0 \leq i \leq 401,$$

evaluated in exact rational arithmetic (an equivalent integer form is $W_i = \max\{2, \min\{257, 257 + \lceil (259992 - 5000i)/7065 \rceil\}\}$, with $W_0 = 257$ and $W_{401} = 11$). Each data register carries two guard bits above it, so the initial work width is 259. Step i compares the highest $e_i = \min(W_i, d_i)$ data bits, where $d_0, \dots, d_{400}$ is the fixed public schedule of Appendix C. The width schedule follows the register-sharing estimate of [2]. Every d_i is at most 77, the value $40 + 2.3\sqrt{n}$ used by the reference; the schedule is shorter in the early steps, where the operands are wide and a coarse comparison rarely errs, rises to 77 in the middle of the walk and falls again as the registers shrink. A shorter comparison saves gates but enlarges the set of denominators on which the recorder makes a wrong swap decision; the certificate of Section 7 charges that set in full.

Initialization loads p into register U (holding u) with X gates and x into register V (holding v), and records the orientation bit $o = 1 \oplus x[0]$ into a clean wire that stays live until the eraser clears

it. Swap their low 256 bits **when** $o = 0$, omitting bit zero because both low bits are one on that branch (255 Toffolis; the negative control is implemented by X gates before and after). Then add p to V **when** $o = 1$ (the constant is loaded by CNOTs controlled by o), at the full work width of 259 bits (258 Toffolis). Thus odd x gives (x, p) and even x gives $(p, p + x)$ at a cost of $255 + 258 = 513$ Toffolis. Each subsequent step produces one symbol in clean wires:

1. compute $a = U[1] \oplus V[1]$ with CNOTs;
2. perform $V \leftarrow V + (-1)^a U$ with a measured ripple adder [3], subtracting U when $a = 1$ by complementing V under control of a before and after the addition. The adder's first carry is exactly $1 - a$, because both registers are odd, so it is set and cleared with X and CNOT gates. This costs W_i Toffolis at work width $W_i + 2$;
3. halve by cyclically moving the low numerator wire to the top of the register; no gate tests the pre-shift low bit or the top guard (`numerator_guard` and `halving_parity`, conditions of the audit predicate, Section 7.3);
4. compute the strict comparison $[U_{\text{high}} > V_{\text{high}}]$ of the top $e_i = \min(W_i, d_i)$ **data** bits after halving and before swapping, excluding the two guards (e_i Toffolis). AND it with a to obtain m (one Toffoli). Erase the comparison scratch by measurement with its phase correction. Swap controlled by m ; since both low bits are one, this takes $W_i - 1$ controlled swaps;
5. release every bit removed by the next width allocation by Z measurement and reset (audit condition `width_drop`).

A step therefore costs $W_i + (W_i - 1) + (e_i + 1) = 2W_i + e_i$ Toffolis. Over the steps $i = 0, \dots, 400$ the widths sum to $\sum W_i = 60,253$ and the comparison widths to $\sum e_i = 23,582$; with initialization and the 925-Toffoli packing pass of Section 5.3 the recorder costs

$$513 + 2(60,253) + 23,582 + 925 = 145,526.$$

(With a fixed 77-bit comparison at every step the sum $\sum \min(W_i, 77)$ would be 27,786 and the recorder 149,730; the schedule saves 4,204 Toffolis per pass.) After the last step, clear the two low bits by X gates and release both registers by Z measurement and reset; the audit requires their pre-clear values to be $(1, 1)$. The guard, parity, width and terminal zero conditions, and agreement of the shortened with the full comparison, are conditions of the classical audit predicate of Section 7.3, not gates or branches of the circuit.

5.3 History compression

Each symbol (a, m) with $m \Rightarrow a$ takes one of three values. The reference's compressor [2, Figure 1] packs three symbols (six wires) into five bits with five Toffolis; we verified its action on all 27 valid inputs and its full 64-state reversible action. We add a five-symbol block that packs ten wires into eight bits by reading the five symbols as a base-three number below $3^5 = 243$ and extracting its digits by four successive divisions by three with the cell of Section 4.4. The dividend widths are 8, 7, 5 and 4 bits (ranges below 243, 81, 27 and 9); a specialized first cell (Appendix E) brings the four divisions to 13, 10, 7 and 4 Toffolis, 34 in all, on twelve local wires. A CNOT from m to a converts the decoded ternary digit to the pair (a, m) and back.

The 401-symbol history is stored as 117 reference triples and ten five-symbol blocks: 665 stored bits, at most 669 live while one block is decoded, and $117 \cdot 5 + 10 \cdot 34 = 925$ Toffolis for each complete packing or unpacking pass. Each multiplication performs four such passes (pack on record, unpack and repack on replay, unpack on erase). With 133 triples and two raw symbols the compressed history would store 669 bits and peak at 670 for 665 Toffolis per pass; the mixed layout buys the one qubit that brings the replay peak to 1,457.

5.4 The fused replay step

Each replay step doubles one coefficient and adds or subtracts the other, modulo p . The reference does this as an approximate doubling followed by a modular addition. We fuse the two into one exact integer addition and one short correction of the low κ bits, which costs 345 Toffolis instead of 414. The fused step implements $s \leftarrow 2s \pm r$ with the sign control $b = 1 - a$, obtained by X gates on the decoded history wire before and after each step.

Put $M = 2^n$, $\Lambda = 2^\kappa$ and $\mu = M/2 - c$, and encode replay coefficients as $E(z) = (z + \mu) \bmod p$. This modular shift keeps identically zero coefficients, which occur systematically in GCD replay, away from the approximation's dangerous boundaries.

For $R = E(r)$ and $S = E(s)$ let $R_b = R$, $K_b = -c$ when $b = 0$ and $R_b = M - 1 - R$, $K_b = c - 1$ when $b = 1$; the conditional complement of R uses only CNOTs. Since $2\mu \equiv -c \pmod{p}$ and $M - 1 \equiv c - 1$,

$$E(2s + (-1)^b r) \equiv 2S + R_b - K_b \pmod{p}.$$

The circuit computes the exact integer $2S + R_b = qM + \ell$ with $q \in \{0, 1, 2\}$, $0 \leq \ell < M$, and adds the short correction $H_b(q) = qc - K_b$ to the low κ bits only:

$$Z = \Lambda \lfloor \ell / \Lambda \rfloor + ((\ell + H_b(q)) \bmod \Lambda),$$

where the six correction constants are $c, 2c, 3c$ for addition and $1 - c, 1, c + 1$ for subtraction ($q = 0, 1, 2$). With $q = q_0 + 2q_1$ and $q_0 q_1 = 0$, every bit of the six correction constants is an affine function of $1, q_0, q_1, b, bq_0, bq_1$; two Toffolis prepare bq_0, bq_1 , the masks are XORed into a clean κ -bit register, one addition of $\kappa - 1$ Toffolis is performed, and the loads are reversed.

The quotient is then recovered and cleared. The low bit is $q_0 = R_b[0] \oplus Z[0] \oplus (1 - b)$, an identity that holds even when a short carry is dropped, so it is cleared with Clifford gates. On value-correct inputs the high bit is $q_1 = (1 - q_0) \mathbf{1}[Z + K_b < R_b]$. It is measured in the X basis with outcome m' , and the unwanted phase $(-1)^{m'q_1}$ is corrected using the truncated comparison $\hat{q}_1 = (1 - q_0) \mathbf{1}[\lfloor Z/2^{n-\delta} \rfloor < \lfloor R_b/2^{n-\delta} \rfloor]$ with a δ -bit comparator and a controlled CZ; the comparator scratch is measurement-uncomputed with its own corrections. The only residual phase is $(-1)^{m'(q_1 \oplus \hat{q}_1)}$, which is charged to the error bound.

The forward step costs n (integer addition with a known-zero low bit), δ (quotient preparation and correction), 2 (table ANDs) and $\kappa - 1$ (short addition) Toffolis:

$$C_{\text{fused}} = n + \kappa + \delta + 1.$$

The inverse step is generated separately, not by reversing a list that contains measurements (Appendix E); its count is $(n - 1) + (\delta + 1) + 2 + (\kappa - 1) = n + \kappa + \delta + 1$ as well. At the reference widths $\kappa = 65$, $\delta = 32$ both directions cost 354. A matched reference step, a separate approximate doubling and conditionally inverted modular addition built from the same primitives with the same widths, costs $n + 2\kappa + \delta - 4 = 414$. That figure already includes its two obvious savings, the known-zero low bit and a Clifford final majority phase. The final circuit uses $\kappa = 61$, $\delta = 27$, for 345 Toffolis per step; Section 7 carries the enlarged error bound.

The interfaces, which convert a coordinate into the encoding E before replay and back afterwards, exploit $\mu \equiv -M/2 \pmod{p}$: flip the most significant bit, then subtract c in the low κ bits controlled by the new top bit ($\kappa - 1 = 60$ Toffolis); the inverse adds the constant before flipping back. Both are exact bit permutations; their disagreement with the intended field map occurs on exactly $c2^{n-\kappa-1}$ canonical inputs each. In short, one replay step is a single exact 256-bit addition plus a 61-bit correction, and its rare errors are charged analytically in Section 7.2.

5.5 Replay assembly

A replay decodes the history block by block, and for each of the 401 steps applies the recorded controlled swap (n Toffolis and $2n$ CNOTs), the sign conversion on the decoded wire, and one fused cell. Multiplication encodes y once and copies its encoded bits into the second register by CNOTs, replays backward, applies an orientation-controlled swap so the live result is in a fixed register, clears the known encoded zero μ with X gates, and decodes. Division encodes y , prepares the other register as μ with X gates, applies the orientation-controlled swap, replays forward, clears the duplicate coefficient by CNOTs, and decodes. Decoding a known encoded zero would produce the noncanonical integer p , so it is cleared as the constant μ ; a live zero scalar is a charged interface failure. The replay costs

$$401(345 + 256) + 256 + 2 \cdot 60 + 2 \cdot 925 = 243,227$$

Toffolis: fused cells, controlled swaps, the endpoint orientation swap, two interfaces and two codec passes.

5.6 Erasure

The eraser runs the recorder's walk backward to clear the history. It starts from the constants $(1, 1)$ at the final width, unpacks blocks in reverse, widens each register with zero wires, and for each step undoes the controlled swap, recomputes the comparison predicate (with the same width e_i) to clear m , doubles by rewiring, performs the opposite signed addition, and recomputes the parity predicate to clear a . It then reverses the initialization and clears p and the orientation bit, returning the restored x on possibly different physical wires. It is a separately generated inverse, with the adder's majority carries uncomputed by X measurement and explicit CZ corrections, so on every failure-free path it restores the denominator and workspace with no residual phase, including on superpositions within that subspace. It costs 145,526 Toffolis, like the recorder, and releases cleared symbols, guards and constants by the same measure/reset rule.

A complete multiplication or division thus costs $145,526 + 243,227 + 145,526 = 534,279$ Toffolis. Recording and erasure peak at 1,310 qubits with the other 256-bit operand and the 18 address bits live. The largest initialization overlap is two 259-bit integer words, a 259-bit controlled constant, 258 carry wires, that operand, orientation and controls:

$$3(259) + 258 + 256 + 1 + 18 = 1,310.$$

The largest recording step occupies 1,144 qubits (Appendix E), and erasure reverses these lifetimes.

Replay retains the 665-bit packed history and expands one block at a time, to at most 669 history wires including the sign control of the current cell. Its peak is

$$512 + 669 + 257 + 1 + 18 = 1,457$$

for the coefficient pair, expanded history, local scratch, orientation and controls. The 257 local scratch wires are the fused cell's $n - 1$ carries plus its two quotient wires (Appendix E, forward stage); the four expansion wires of the block being decoded are counted inside the 669, so the codec and the cell do not compete for space. Local scratch is recycled between steps, and no separate copy of the denominator survives recording. Interfaces use at most $2\kappa - 1 = 121$ scratch wires and cannot raise the peak. The kernel returns with 530 live qubits (the two coordinates and the 18 address bits).

6 The remaining arithmetic and the complete point addition

6.1 Modular additions and subtractions

The three additions and four subtractions of the pipeline use the reference’s approximate pseudo-Mersenne modular adder at $\kappa = 65$ and $\delta = 32$ (the outer widths, independent of the replay widths): an integer addition, a short correction of the low κ bits, and a δ -bit truncated phase comparison, with all temporary ANDs measured and phase-corrected. The integer addition with carry-out costs $n = 256$ Toffolis; the carry bit controls the addition of the constant c into the low κ bits ($\kappa - 1 = 64$), is measured, and its phase is repaired by a δ -bit comparison of the top bits of result and operand ($\delta - 1 = 31$, the final majority being applied as CZ gates). An addition of a classical or quantum operand thus costs $256 + 64 + 31 = 351$ Toffolis. A subtraction is an addition conjugated by the reference’s modular complement, which adds c into the low κ bits (64) and then applies a bitwise NOT, before and after; the two extra constant additions bring it to $351 + 128 = 479$. Classical constants, including the mask R and its negative, are conservatively prepared as 256-bit quantum operands with X gates and cleared afterwards. Each of these components peaks at 1,042 qubits in the global layout.

6.2 Modular square subtraction

The operation $|x\rangle|y\rangle \mapsto |x\rangle|(y - x^2) \bmod p\rangle$ (in the pipeline of Section 4.3 the source is the y register holding λ and the target is x) follows the reference’s construction. Write $x = x_{\text{lo}} + 2^{128}x_{\text{hi}}$ and $2^{256} \equiv c \pmod{p}$. Then

$$y - x^2 \equiv y - x_{\text{lo}}^2 + 2^{128}x_{\text{lo}}^2 + 2^{128}x_{\text{hi}}^2 - cx_{\text{hi}}^2 - 2^{128}(x_{\text{lo}} + x_{\text{hi}})^2,$$

so three integer squares are needed: the two halves and their 129-bit sum. Each square is computed into clean scratch, its contributions are added to y , and the square is uncomputed.

For an n -bit input the integer squarer uses $x^2 = -x + \sum_i (2x_i - 1)2^{2i+1}(h_i + x_i)$ with $h_i = \sum_{j>i} 2^{j-i-1}x_j$: it subtracts x from the low $n + 1$ bits of the clean output, then subtracts, for each i , a conditionally complemented window of h_i (Appendix E). It costs $\frac{1}{2}n(n + 3) - 1$ Toffolis: 8,383 for 128 bits and 8,513 for 129 bits, and the same again to uncompute on the image.

Each square’s contributions are added as shifted slices of the target, using the signed digits $c = 2^{32} + 2^{10} - 2^6 + 2^4 + 1$ for the weight $2^{256} \equiv c$ and for the term $-cz$. Adding $2^{128}z$ costs 1,146 for a half square and 1,668 for the negated sum square; adding $-cz$ costs 4,583. Appendix E gives the slice schedule, the exact identity for the discarded high bits and the per-slice costs. Table 4 gives the stage costs. In the low-half stage the 479 is the full-width subtraction of x_{lo}^2 . The sum stage extends the existing x_{hi} register by one clean high bit, adds x_{lo} into it, and later subtracts x_{lo} to restore it (128 Toffolis each).

Table 4. Square-subtraction stage costs.

Stage	Toffolis
Low half	$8,383 + 1,146 + 479 + 8,383 = 18,391$
High half	$8,383 + 1,146 + 4,583 + 8,383 = 22,495$
Sum	$128 + 8,513 + 1,668 + 8,513 + 128 = 18,950$
Total	$18,391 + 22,495 + 18,950 = 59,836$

The source is modified only while the sum square is computed, used and uncomputed; its appended high bit is zero before release. The peak, 1,042 qubits, is a half-square full-width

addition or subtraction: $512 + 18 + 256 + 256 = 1,042$ wires for the caller's field words, controls, square buffer and carries (Appendix E gives all lifetimes).

6.3 The point addition

The three additions, four subtractions and one square subtraction total $3 \cdot 351 + 4 \cdot 479 + 59,836 = 62,805$ Toffolis. With the two kernels, the arithmetic of one point addition costs $2 \times 534,279 + 62,805 = 1,131,363$ Toffolis, against the reference's 1,196,000.

One reusable allocation covers the two 256-bit coordinates, the 18 address bits, the lookup banks and scratch, the history and all arithmetic scratch. A window proceeds as follows.

1. Fold the address into sign and magnitude; conditionally complement the encoded y .
2. Read the first two columns (Section 4.3), subtract them, and release each payload by X measurement after use, recording the masks.
3. Divide y by x with the complete division kernel.
4. Add the mask R , read and add $3x_2$, subtract y^2 , add $-R$.
5. Multiply y by x with the complete multiplication kernel.
6. Read and subtract the last two columns, complement x .
7. Apply the single phase correction for all measured banks, undo the y complement and the address fold.

The stages are sequential, so their peaks are reused in time rather than added; the overall peak is 1,457 qubits, reached during replay (Table 5).

Table 5. Maximum simultaneously live qubits by stage, including the 18 address bits.

Stage	Peak logical qubits
Initial lookup, first / second payload segment	1,455 / 1,442
Width-17 lookup, first / second segment	1,455 / 1,443
Width-18 lookup, first / second segment	1,456 / 1,444
Recording and erasure	1,310
Replay, with the expanded history block and local scratch	1,457
Coordinate arithmetic and square subtraction	1,042
Terminal phase correction, widths 17 / 18 / initial	793 / 794 / 1,050
At a window boundary (coordinates and address)	530

Executed at full 256-bit size, the complete point addition contains $1,131,363 + 103,053 = 1,234,416$ Toffolis for a 17-bit window and $1,131,363 + 198,904 = 1,330,267$ for an 18-bit window.

6.4 Totals

Table 6 gives the Toffoli totals (Appendix A collects every count).

Table 6. Toffoli totals. The middle column keeps the reference arithmetic and changes only the lookups; the last column is the complete new circuit.

	Reference circuit [1]	Lookup redesign only	Complete new circuit
Point additions	28	26	26
Arithmetic, all additions	33,488,000	31,096,000	29,415,438
Lookups, all additions	5,505,024	3,254,484	3,254,484

	Reference circuit [1]	Lookup redesign only	Complete new circuit
Initial lookup	262,144	101,323	101,323
Whole algorithm	39,255,168	34,451,807	32,771,245
Reduction	–	12.2%	16.5%

The saving of $39,255,168 - 32,771,245 = 6,483,923$ Toffolis can be attributed as

$$\begin{aligned}
& 2(1,196,000) + 26(1,196,000 - 1,131,363) \\
& \quad + (5,505,024 - 3,254,484) + (262,144 - 101,323) \\
& = 2,392,000 + 1,680,562 + 2,250,540 + 160,821,
\end{aligned}$$

that is, two fewer point additions, cheaper arithmetic in the remaining 26, cheaper ordinary lookups, and a cheaper initial lookup. These are not four independent improvements: the cheaper lookups are what make the wider windows, and hence the two fewer additions, worthwhile. Figure 1 shows the same split graphically.

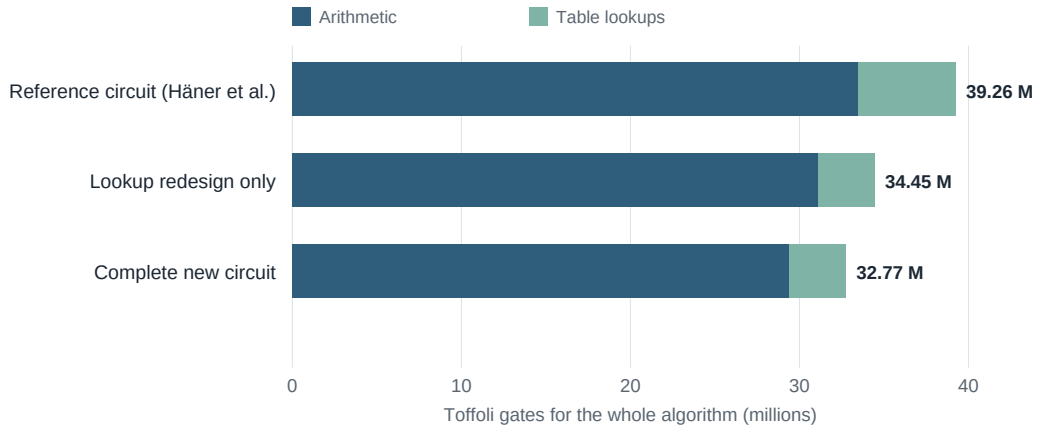


Figure 1. Toffoli count of the whole algorithm, split into arithmetic and table lookups.

7 Success probability

7.1 Failure model

A point addition fails if any component produces a wrong value, leaves an unwanted relative phase, or fails to return a temporary register to zero. Such a failure can spoil the interference that Shor’s algorithm relies on, so we bound the probability that any of the 26 additions fails on the *ideal trajectory*, the computation the circuit would perform if every earlier component were exact, and charge a run at its first failure. This union bound needs no independence between windows or between steps.

The reference certifies its multiplication statistically (one million inputs, 21 failures, per-kernel bound 0.000149304). Our multiplication fails on a different set of inputs, so we certify it afresh: an analytic bound for replay and its interfaces, and a statistical bound for the recorder.

7.2 Analytic bound for replay

Theorem 7.1. Let $p = 2^n - c$ with $p > 8c$, $4c \leq 2^\kappa$, $\kappa < n$, $1 \leq \delta \leq n$. Fix a valid history and a uniform field scalar y . A replay of T fused steps together with its two interfaces fails with probability at most

$$q_{\text{rep}}(T) = T \frac{(4c-1)\lceil p/2^\kappa \rceil + 2^{n-\delta+1} + 2c + 1}{p} + \frac{c 2^{n-\kappa}}{p}.$$

The proof (Appendix F) counts value errors and phase-only errors separately for register values that are fixed linear functions of one uniform y , treats identically zero coefficients, inverse cells and both live interfaces explicitly, and takes the union bound on the unconditioned ideal trajectories. At $\kappa = 61$, $\delta = 27$ and $T = 401$,

$$q_{\text{rep}} < 8.964912 \times 10^{-6}.$$

(At the reference widths 65/32 the bound would be 3.74×10^{-7} ; the cheaper widths trade this margin for 9 Toffolis per step.)

7.3 Statistical certificate for the recorder

The following deterministic predicate, the *audit predicate*, defines the recorder's conservative bad-denominator set. All integers in it are exact signed integers; there is no machine-word wrap before a guard test. Let h be the fraction of nonzero denominators for which it returns a failure. Outside that set the history is valid, so Lemma 5.1 and Theorem 7.1 apply, and since erasure is exact on the same domain, one multiplication or division fails with probability at most $h + q_{\text{rep}}$.

```
audit(x):                                     # 0 < x < p; W and d are public
  (U,V) = (x,p) if x is odd else (p,p+x)
  for i = 0,...,400:
    if U < 0 or V < 0 or max(U,V) >= 2^W[i]:
      return (entry_width, i)
    a = bit(U,1) XOR bit(V,1)
    N = V + (1-2*a)*U
    if N < 0 or N >= 2^(W[i]+1): return (numerator_guard, i)
    if N mod 2 != 0: return (halving_parity, i)
    Vnew = N/2
    e = min(W[i], d[i]); t = W[i]-e
    m = a AND (floor(U/2^t) > floor(Vnew/2^t))
    if m != (a AND (U > Vnew)): return (strict_comparison, i)
    (U,V) = (Vnew,U) if m == 1 else (U,Vnew)
    if max(U,V) >= 2^W[i+1]: return (width_drop, i)
  if (U,V) != (1,1): return (terminal, 401)
  return good
```

`numerator_guard` and `halving_parity` concern the wires moved in step 3 of Section 5.2, `width_drop` the wires released in step 5, and `terminal` the final release. `entry_width` holds at $i = 0$ because $p + x < 2^{257}$ and is implied at later steps by `width_drop` at the step before. `strict_comparison` has no circuit counterpart.

The audit is a classical certification procedure and stops at its **first** failure; the quantum circuit contains none of its tests, including the full strict comparison, never aborts, and postselects no outcome. On a denominator that fails the audit every released wire is still measured and reset,

and the denominator is excluded from the proved good domain even if the resulting computation happens to be correct. Rejecting a wrong shortened comparison is conservative, since some such choices still lead to a valid history under Lemma 5.1: the certificate bounds the stated bad set, not the exact fraction of wrong final quantum outputs.

Before drawing the certification sample, we froze the design (step count, width schedule, comparison rule, history layout, endpoint conventions and implementation), wrote an independent integer implementation of the audit predicate, and tested both against the gate-level circuit. The implication that the circuit behaves as required on every denominator the audit accepts follows from exact gate identities rather than from these tests. The measured adders, comparators and majority erasures of Appendix E are exact on every basis input with their stated phase corrections, and each audit condition other than `strict_comparison` constrains exactly the wires named above (Section 5.2), while `strict_comparison` makes every recorded swap bit the canonical one. By induction over the 401 steps, on an accepted denominator the recorder writes the exact history of Lemma 5.1, releases only zero wires and leaves no residual phase, and the separately generated eraser of Section 5.6 restores the denominator on the same domain (Appendix F, “First-failure bookkeeping”). The agreement of the two audit implementations on 4,610 full-size denominators and an 82-row full-width record-and-erase run, in which the circuit marks exactly the audit-bad rows, test the implementation (Appendix D). We then drew two million fresh nonzero field elements by rejection sampling from uniform random bytes (256-bit words, rejecting zero and values of at least p) and evaluated each with the audit predicate, an exact integer model of the circuit’s operations plus the full comparison. There were 94 failures: 54 `strict_comparison` events (wrong truncated comparisons) and 40 `terminal` events (walks that did not reach (1, 1) within 401 steps); no `entry_width`, `numerator_guard`, `halving_parity` or `width_drop` event occurred. A repetition of the experiment would give a different count; the reproducible statement is the bound below, not the count.

Lemma 7.2. For N_s independent trials with Φ failures and $\Theta = \lceil 7\xi/10 \rceil$, the true failure probability satisfies

$$h \leq h^+ = \min\left\{1, \frac{\Phi + \Theta + \lceil \sqrt{\Theta^2 + 2\Phi\Theta} \rceil}{N_s}\right\}$$

with confidence at least $1 - 2^{-\xi}$.

Proof. For true mean $\mu = N_s h$ and $k \leq \mu$ the Chernoff lower-tail bound gives $\Pr[\Phi \leq k] \leq \exp(-(\mu - k)^2/2\mu)$. The larger root of $(\mu - k)^2 = 2\mu\Theta$ is $\mu = k + \Theta + \sqrt{\Theta^2 + 2k\Theta}$, so if $h^+ < h$ the observation lies in a tail of probability at most $e^{-\Theta}$; rounding the square root up only enlarges h^+ ; and $\ln 2 < 7/10$ gives $e^{-\Theta} < 2^{-\xi}$. $\square$

With $N_s = 2 \times 10^6$, $\Phi = 94$, $\xi = 129$ we get $\Theta = 91$ and

$$h \leq h^+ = \frac{345}{2,000,000} = 0.0001725$$

with confidence at least $1 - 2^{-129}$, under the independent-uniform-byte sampling model. One complete multiplication or division therefore fails on uniform inputs with probability at most $h + q_{\text{rep}} < 0.000181465$.

7.4 Transfer to point-addition inputs

The inputs of the two kernels are coordinate differences, not uniform field elements. By Lemma 4.2 each point pair has probability at most $1/((r-1)(r-K))$. A bad denominator set of fraction h has at most $2(r-1)(p-1)h$ point-pair preimages, and a bad field-pair set of fraction q at most

$4p(p-1)q$ (Appendix F). The two kernels therefore contribute at most $2D_h h + 2D_{\text{pair}} q_{\text{rep}}$ per window with

$$D_h = \frac{2(p-1)}{r-K}, \quad D_{\text{pair}} = \frac{4p(p-1)}{(r-1)(r-K)},$$

both numerically indistinguishable from 2 and 4. Under the same uniform-mask hypotheses (the field mask R_i of Section 4.2 is uniform), the three additions, four subtractions and the square subtraction have analytic bounds $p_{\text{add}} \leq 1.746230 \times 10^{-9}$, $p_{\text{sub}} \leq 1.059380 \times 10^{-8}$ and $p_{\text{sq}} \leq 9.895304 \times 10^{-9}$ per window (Appendix F). With the affine exceptions and the encoding-range condition of Section 4.3, the 26-window error is

$$\epsilon \leq 26 \left[2D_h h + 2D_{\text{pair}} q_{\text{rep}} + p_{\text{add}} + p_{\text{sub}} + p_{\text{sq}} + \frac{4 + 32(\tau + 1)}{r - K} \right] \leq 0.019805279770$$

(Appendix B gives the arithmetic).

7.5 From error to success

Theorem 7.3. Let the ideal normalized state be $|\psi\rangle = \sum_x \alpha_x |x\rangle |f_x\rangle$ with normalized output blocks, and for a random implementation ω let $|\tilde{\psi}_\omega\rangle = \sum_x \alpha_x |x\rangle |g_{\omega,x}\rangle$ with the same amplitudes and normalized blocks. Suppose that for every x , outside a bad event of probability $p_x \leq \epsilon$, $|g_{\omega,x}\rangle = |f_x\rangle$ including phase. For a success projector Π on the scalar registers and $P_0 = \|\Pi\psi\|^2$,

$$\mathbb{E}_\omega \|\Pi\tilde{\psi}_\omega\|^2 \geq [\max\{0, (1 - \epsilon)\sqrt{P_0} - \epsilon\}]^2.$$

Proof. In each block $\mathbb{E}g_{\omega,x} = (1 - \epsilon)f_x + \zeta_x$ with $\zeta_x = (\epsilon - p_x)f_x + \mathbb{E}[\mathbf{1}_{\text{bad}}g_{\omega,x}]$, so $\|\zeta_x\| \leq \epsilon$, and by orthogonality of the labels $\|\sum_x \alpha_x |x\rangle \zeta_x\| \leq \epsilon$. Jensen's inequality applied to $\|\Pi\psi\|^2$ and the reverse triangle inequality applied to the mean state $(1 - \epsilon)\psi + \zeta$ give the bound. $\square$

The bound improves the reference's amplitude form $\sqrt{P_0} - 2\epsilon$ by $\epsilon(1 - \sqrt{P_0})$ and is asymptotically sharp (equally weighted labels, a rank-one success vector with signs of mean t , positive labels flipped with probability ϵ). Appendix F shows why the circuit satisfies the theorem's hypotheses despite its measurements and resets, and derives the ideal anchor $P_0 \geq (r-1)r^{-1}(8/\pi^2)^2 \approx 0.657$ for the operation with the shortened second register, including the classical recovery of the 48 omitted bits. With that anchor and ϵ above, exact rational evaluation with interval enclosures for π and downward rounding gives

$$P_{\text{success}} \geq 0.600176,$$

with confidence at least $1 - 2^{-129}$ (Appendix B gives the arithmetic). The reference bound is 0.553. The bound retains the reference's ideal-precision and postprocessing assumptions and the randomization of Section 4.2, averages over that randomization and over measurement transcripts, and excludes physical noise. Shortened comparisons can enlarge the conservative bad-denominator set relative to fixed 77-bit comparisons, but the larger fresh sample tightens its statistical upper bound. This explains how the cheaper construction can have a stronger certified success lower bound; it does not establish an ordering of true success probabilities.

8 Verification

Every count and bound in this report was recomputed by a second, independent implementation from the circuits and the recorded data. All circuits are executed at full width by a gate-level basis-and-phase simulation that streams the actual gates, tracks basis values and phases, samples measurement outcomes and applies the classically controlled corrections, and its outputs, residual

phases and released workspace are compared with independent integer or curve oracles; small instances are also checked by an independent state-vector simulation on every measurement transcript. The statistical certificate is a separate experiment on the frozen audit predicate (Section 7.3), and Appendix D describes every experiment. An *entangled reference* is a reference qubit entangled with the input and untouched by the circuit, whose correlations must survive. A *negative control* is a deliberately broken variant that must be detected. *Adaptive branches* are distinct measurement-dependent execution paths.

Table 7. Verification summary.

Component	Checks
Lookups	Every signed address of the 16-, 17- and 18-bit layouts and of the initial lookup executed at full size (65,536, 131,072 and 262,144 addresses respectively; the initial lookup also has 262,144); phases reconstructed from the public tables, selection permutation and recorded masks, independently of the simulation; negative control with unselected banks omitted; small instances checked by an independent state-vector simulation with an entangled reference
Fused replay step	Exhaustive on four small parameter sets, including all 256 measurement transcripts on the 56 good inputs of the three-bit case and all 160 adaptive branches; 4,391 full-width random and boundary inputs at the final widths 61/27, in both directions, with outputs, residual phases and scratch checked; omitted phase corrections detected
Replay contract	67,292 small-field and 12,560 full-width scalar cases over 2,512 denominators
Recorder and eraser	All nonzero denominators at $p = 29$ and $p = 61$ with multiple history layouts and three measurement seeds; scheduled small-field checks at $p = 29$ with two width profiles and three comparison schedules, including deliberately coarse ones; an 82-row full-width record-and-erase run with the final schedule and deliberate wrong comparisons; an unshrunk 512-step full-width run; the signed adder exhaustive at widths 3 to 5 on all transcripts; 1,728 basis and amplitude primitive cases; agreement of two independent implementations of the audit predicate on 4,610 full-size denominators
Complete kernels	All 812 scalar pairs at $p = 29$ in both directions for the small-field construction; 32 fresh random full-width pairs in both directions with the final 401-entry comparison schedule, restored denominator, product or quotient, zero phase, clean workspace and 530 remaining qubits; each final full-size kernel counted at exactly 534,279 Toffolis with a 1,457-wire peak
Coordinate arithmetic	Integer squarer exhaustive through 8 bits and amplitude-checked with an entangled reference at 2 and 3 bits; 528 addition and subtraction cases; 144 square-subtraction cases; the signed- c identity on 1,005 boundary and random cases
Complete point addition	Full-size executions with the final schedule at widths 17 and 18, eight input rows each, against an independent curve computation; peak, output, phase and cleanup agree
Certificate	The two million frozen inputs re-run by an independent implementation, reproducing the 94 failures and no other event; the first audit event and the step at which it occurs checked with independent big-integer arithmetic on 8,063 of the rows including every failure; 380 exact binomial coverage cases for Lemma 7.2; all bounds recomputed in exact rational arithmetic

No finite set of tests covers the full 256-bit input space. The tests support the implementation identities and their transcription; the coherent proofs and statistical certificate supply the general correctness and error bounds.

9 Scope

The claim is a logical one. We give a complete circuit that performs the reference computation with 16.5% fewer Toffoli gates, in the same number of logical qubits, with a higher certified success probability. It holds on any fault-tolerant architecture that implements the same logical gate set.

We do not claim a new physical runtime or physical qubit count. The reference paper obtains its 26-day estimate by compiling its circuit onto a specific trapped-ion architecture, with its own assumptions about scheduling, qubit transport and error correction. We have not repeated that compilation. Neither a runtime reduction nor a revised physical footprint follows from the Toffoli count alone; both require recompilation and scheduling.

10 Conclusion

We have replaced the logical circuit of the most complete published estimate for breaking secp256k1. The new circuit uses at most 32,771,245 Toffoli gates, 16.5% fewer, on the same 1,457 logical qubits. Its certified logical success probability is at least 0.600, compared with 0.553. Most of the saving comes from the lookup redesign, which makes wider windows economical and removes two point additions; the rest comes from the fused replay step and shortened comparisons in the multiplication. The circuit can be adopted as it stands at the logical level; compiling it onto a physical architecture is the natural next step.

A Gate accounting

Lookups. Per read: 10,927 (width 16 with 16 controls), 20,509 (17), 39,628 (18). The latter two reserve all 18 controls. Phase correction: 380, 508, 764. Five reads with correction: 55,015, 103,053, 198,904. Initial lookup: $88,406 + 11,897 + 1,020 = 101,323$.

Recorder and eraser. $513 + \sum_{i=0}^{400} (2W_i + e_i) + 925 = 513 + 120,506 + 23,582 + 925 = 145,526$ each, with $\sum W_i = 60,253$ and $\sum e_i = 23,582$ from the schedules of Section 5.2 and Appendix C.

Replay. $401(345 + 256) + 256 + 2 \cdot 60 + 2 \cdot 925 = 243,227$.

Per multiplication. $145,526 + 243,227 + 145,526 = 534,279$.

Per point addition. $2 \times 534,279 + 4 \times 479 + 3 \times 351 + 59,836 = 1,131,363$.

Whole algorithm.

$$20(1,131,363 + 103,053) + 6(1,131,363 + 198,904) = 32,669,922,$$

plus the initial lookup, 32,771,245. Against 38,993,024 and 39,255,168 the reductions are 16.22% and 16.52%. The window partition is twelve 17-bit and two 18-bit windows on the P side (240 bits) and eight 17-bit and four 18-bit windows on the Q side (208 bits).

B Certificate arithmetic

Replay. With $n = 256$, $c = 2^{32} + 977$, $\kappa = 61$, $\delta = 27$, $T = 401$,

$$q_{\text{rep}} = 401 \frac{(4c - 1)\lceil p/2^{61} \rceil + 2^{230} + 2c + 1}{p} + \frac{c 2^{195}}{p} < 8.964912 \times 10^{-6}.$$

Recorder. With $N_s = 2 \times 10^6$, $\Phi = 94$, $\xi = 129$, $\Theta = 91$, $h^+ = (94 + 91 + \lceil \sqrt{8281 + 17108} \rceil) / (2 \times 10^6) = 345 / (2 \times 10^6) = 0.0001725$.

Per kernel. $h + q_{\text{rep}} < 0.000181465$.

Per window and total. With $D_h = 2(p - 1)/(r - K)$, $D_{\text{pair}} = 4p(p - 1)/((r - 1)(r - K))$, $K = 2^{18}$, $\tau = (c - 1)/2$,

$$\epsilon \leq 26 \left[2D_h h + 2D_{\text{pair}} q_{\text{rep}} + p_{\text{add}} + p_{\text{sub}} + p_{\text{sq}} + \frac{4 + 32(\tau + 1)}{r - K} \right] = 0.019805279769 \dots$$

Success. $P_0 \geq 64(r - 1)/(r\pi^4)$; $[(1 - \epsilon)\sqrt{P_0} - \epsilon]^2 = 0.60017656 \dots$, reported as 0.600176 after rounding downward.

C The comparison schedule

The recorder compares the top $e_i = \min(W_i, d_i)$ bits of its two integers at step i , $0 \leq i \leq 400$, with W_i the data width of Section 5.2 and d_i the fixed public schedule below. A row headed i lists $d_i, d_{i+1}, \dots, d_{i+9}$. Every entry is at most 77, and $\sum_{i=0}^{400} e_i = 23,582$. The eraser uses the same e_i at the matching step. The schedule is part of the circuit specification: it was fixed before the certification sample of Section 7.3 was drawn, and any change to it requires a new certificate.

Table 8. The comparison-width schedule d_i .

i	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
0	20	23	23	25	24	24	27	27	28	28
10	28	31	32	31	33	34	36	43	34	37
20	36	38	37	41	39	40	42	41	42	45
30	45	52	46	49	47	47	50	52	51	53
40	55	55	53	59	54	56	59	60	59	58
50	61	61	61	59	63	60	61	60	61	61
60	66	65	62	62	61	63	65	63	61	63
70	61	63	63	64	61	64	63	61	64	64
80	64	62	62	66	62	65	62	65	66	61
90	63	61	64	62	64	62	61	69	66	68
100	67	66	63	67	63	62	64	62	64	64
110	67	62	63	64	65	67	63	73	65	64
120	66	67	63	66	63	63	62	63	67	64
130	66	63	66	64	64	66	67	64	71	65
140	65	64	66	69	64	66	65	66	67	69
150	65	67	66	66	68	74	66	66	66	64
160	67	67	65	67	67	66	68	69	68	69
170	69	66	70	76	68	64	68	66	69	68
180	66	68	65	69	68	66	67	67	72	71
190	69	68	65	70	67	67	69	67	69	68

i	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
200	68	70	70	69	70	67	70	73	70	68
210	70	71	71	71	72	70	72	71	69	70
220	73	71	67	71	67	69	67	70	69	73
230	72	71	68	67	72	69	71	72	74	72
240	71	72	75	70	73	73	69	69	74	69
250	74	71	70	72	73	71	71	70	72	71
260	69	71	71	73	71	73	70	70	72	72
270	70	77	72	70	71	70	71	72	71	73
280	71	71	71	69	69	73	72	73	71	76
290	72	74	72	72	77	74	73	71	71	74
300	72	71	71	73	72	72	72	74	77	73
310	74	71	75	74	74	74	72	73	72	72
320	71	70	69	69	68	67	67	66	65	64
330	64	63	62	62	61	60	60	59	58	57
340	57	56	55	55	54	53	52	52	51	50
350	50	49	48	47	47	46	45	45	44	43
360	43	42	41	40	40	39	38	38	37	36
370	35	35	34	33	33	32	31	30	30	29
380	28	28	27	26	26	25	24	23	23	22
390	21	21	20	19	18	18	17	16	16	15
400	14									

D Verification procedures

This appendix describes the experiments summarized in Section 8, in the order of its table. A basis-and-phase executor applies each X, CX, CCX, Z and CZ to the current bit values and sign. For an X measurement it records the outcome and the factor $(-1)^{\mu\nu}$ before reset, then executes the specified feed-forward. Final words, residual sign and released workspace are compared with an independent integer or curve oracle. This is full-width gate execution on selected basis inputs with tracked phases, not a state-vector simulation of all 2^{256} field values. On good inputs the executor asserts that every released wire is zero; on other inputs it records violated assertions and continues to execute the circuit’s measure-and-reset operations unchanged. Small-state tests retain amplitudes and an entangled reference (Section 8) and check every measurement branch and its norm.

D.1 Lookups

The 16-bit layout executed at full size is the 16-control plan of Table 3 (55,015 Toffolis), on all 65,536 signed addresses. The negative control omits the unselected banks from the phase correction: every coordinate stays correct and the interference is wrong, and on the original five-read layout it corrupted 32,742 of 65,536 signed addresses. The small entangled test uses a 3-bit address and an 8-entry single-bit table read with three banks. The address is entangled with an untouched reference qubit in a dense complex-amplitude simulation, every measurement record (at most 9 measurements) is enumerated, and the address-dependent phase of Lemma 4.3 must be reproduced and cancelled exactly.

D.2 Fused replay step

The exhaustive tests enumerate all $(R, S, b) \in [0, p)^2 \times \{0, 1\}$ for

$$(n, c, \kappa, \delta) \in \{(3, 1, 2, 2), (5, 3, 4, 3), (6, 3, 4, 4), (6, 5, 5, 4)\},$$

with three measurement seeds per triple. The oracle uses the exact integer semantics of Section 5.4: it computes q, ℓ by integer division of $2S + R_b$ by M , applies the low-word correction modulo 2^κ , and predicts the residual phase from the XOR of the true and estimated high quotient. The inverse is tested on every forward-good image and on every triple of its own input domain. The three-bit instance has $p = 7 < 8c$, so it lies outside the hypotheses of Theorem 7.1 and tests gate identities only. It has 98 basis inputs, of which 56 are forward-good; its adaptive forward circuit, which skips the comparison when the high-quotient outcome is 0, was run on a coherent superposition of the 56 good inputs. Enumerating all measurement strings and merging those that execute the same gates reduces the 256 raw strings to 160 adaptive branches, whose probabilities sum to 1 and whose amplitudes all match the ideal to within 2×10^{-13} . Removing the high-quotient phase repair must be detected.

The final-width suite uses 4,096 triples (R, S, b) drawn uniformly from $[0, p)^2 \times \{0, 1\}$ with a fixed pseudorandom seed, and 295 boundary triples. These take $R \in \{0, 1, c-1, c, c+1, \mu, p-2, p-1\}$ and both signs; for each pair, ideal outputs Z_* are selected from $\{0, 1, 3c-1, 3c, \Lambda-c, \Lambda-c+1, \Lambda-1, \mu, p-1\}$ and $(R_b + t) \bmod p$ for $t \in \{-\Delta-1, -\Delta, -c-1, -c, -1, 0, 1, c, \Delta-1, \Delta, \Delta+1\}$, with $\Delta = 2^{n-\delta}$, duplicates within each pair are removed, and $S = (Z_* - R_b + K_b)/2 \pmod{p}$. Each triple is run with three measurement seeds; the high-quotient outcome is fixed to 0, 1 and 0 for the three seeds, so both of its equally likely values are exercised, and all other outcomes are sampled from the seed. A triple passes if, for every seed, the output word, the residual sign and the released scratch equal the oracle's prediction, including any predicted value or phase error, and, if it is forward-good, the inverse cell returns the input with sign +1. At the final widths 61/27, 75 of these 4,391 inputs have a predicted value error and 13 further inputs a phase-only error; all match the oracle, and the other 4,303 invert exactly. These deliberately concentrated boundary cases are not samples for an error-rate estimate.

D.3 Replay contract

The small-field checks use $p \in \{7, 11, 13, 29, 59, 251\}$ with every denominator $1 \leq x < p$ and every scalar $0 \leq y < p$ (67,292 cases). Replaying the exact history must return $(xy, 0)$ or $(0, xy)$ for multiplication and $(y/x, y/x)$ for division. The same check is repeated with the swap comparison truncated to its leading 1, 2 and 3 bits. Of these 1,092 denominator and width combinations, 119 do not terminate and are recorded as such; every terminating one reproduces the exact history, so these reruns found no alternative valid history. The truncated reruns are not included in the 67,292 cases. At full width, 2,512 denominators ($1, 2, p-1, p-2$, the values 2^k and $p-2^k$ for $0 \leq k \leq 255$, and 2,000 pseudorandom values, deduplicated) are combined with five scalars each ($0, 1, 2, p-1$ and one pseudorandom value), giving 12,560 cases that use the exact untruncated history of each denominator, however long, and pass on the same condition. The longest exact history (510 steps) and the number of denominators whose exact walk exceeds 401 steps (219) are also recorded. The 401-step recorder cannot finish these denominators, but they are not the whole bad set of the implemented recorder, whose shortened comparisons and shrinking widths can fail on a shorter walk: for $x = 3$ the exact walk has 255 steps, yet the first audit event is a wrong strict comparison at step $i = 253$. The recorder's bad set is the one defined by the audit predicate of Section 7.3.

D.4 Recorder and eraser

Each test computes the first audit event and its step with the audit predicate of Section 7.3. On every audit-good row, a run passes if the packed history equals the exact integer history, the residual phase is zero, the denominator is restored, the other operand and the address are unchanged, and all workspace is clean. Exponents below denote repetitions.

The exhaustive small-field runs use $(n, c, T) = (5, 3, 11)$, so $p = 29$, with block sizes $(3, 3, 3, 2)$ and $(3, 3, 5)$, and $(n, c, T) = (6, 3, 15)$, so $p = 61$, with block sizes (3^5) and (5^3) ; they cover every nonzero denominator with full-width comparisons and measurement seeds 1, 2 and 3. The scheduled small-field tests use $p = 29$, four triple blocks, width profiles (6^{13}) and $(6^4, 5^4, 3^5)$, and comparison schedules $(1, 2, 5)^4$, $(5, 2, 1)^4$ and (6^{12}) . The unshrunk run uses $T = 512$, 170 triples and one pair, and keeps and compares all 257 data bits at every step. It covers 8 denominators (two pseudorandom and six structured values) with zero failures and zero phase; recording and erasure each cost 396,115 Toffolis and peak at 1,905 wires. It checks the exact construction independently of the shrinking schedule.

The 1,728 primitive cases cover the known-low-carry adder, the signed forward addition and the signed inverse addition at widths 3 and 4, on every valid basis input (U odd, V of the required parity, $a \in \{0, 1\}$) under every measurement string, each with amplitude error below 2×10^{-13} . For each primitive and width, a coherent test gives all valid inputs pseudorandom complex amplitudes, entangles them with an external reference bit and checks every measurement string, with total probability 1; omitting the phase corrections must be detected.

The two implementations of the audit predicate are compared on 4,610 full-size denominators, $1, 2, 3, p-1, p-2, \lfloor p/3 \rfloor + 3$, the values 2^i and $p - 2^i$ for $1 \leq i \leq 255$, and 4,096 pseudorandom values, deduplicated; both must return the same first event at the same step. The events were 4,204 good, 3 `strict_comparison` and 403 `width_drop`. The width drops all come from the structured families 2^i and $p - 2^i$, whose walks stay wide for longer than the schedule allows, and none from the pseudorandom values.

The 82-row full-width record-and-erase run with the final schedule combines 64 pseudorandom denominators, $1, 2, 3, p-1, p-2, \lfloor p/3 \rfloor + 3$, and twelve wrong-comparison cases: for $j = 180, \dots, 238$ and shifts $0, \dots, 3$ put $x = \lfloor p/3 \rfloor + 2^j + \text{shift}$, keep for each j the first shift whose first audit event is a wrong strict comparison, and use the first twelve such values. Its 67 audit-good rows must pass the checks above. For its 15 audit-bad rows the circuit records a nonzero released wire or a wrong terminal pair (a wrong shortened comparison leads to one of these later); the set of rows so marked must equal the set of audit-bad rows, and the circuit must run to completion without any abort or conditioning. This run tests the implementation at full width; the implication from audit acceptance to correct circuit behavior rests on the gate identities and the induction described in Section 7.3.

D.5 Complete kernels

The full-width kernel checks use 32 pairs (x, y) with $x \neq 0$, drawn uniformly with a fixed seed. Both directions are applied and compared with xy or y/x in exact modular arithmetic. The high-quotient outcome is forced to one, a simulator choice of one of two equally likely branches (Appendix F, “Fair X outcomes”), to exercise every optional branch.

D.6 Coordinate arithmetic

Every clean-target integer square through eight bits and its inverse is enumerated, and an entangled-reference test is run at two and three bits. The 528 coordinate cases comprise

512 pseudorandom pairs plus $\{0, 1, c, p - 1\}^2$; the 144 square-subtraction cases comprise 128 pseudorandom pairs plus the same boundary grid. They are compared with exact modular addition, subtraction or $y - x^2$. Every pseudorandom case must match exactly. The boundary-grid cases are run with one fixed measurement transcript per circuit and their mismatches recorded: 5 for addition, 3 for subtraction and 9 for square subtraction. Each addition and subtraction mismatch lies in the exceptional set counted in Appendix F: the internal addition leaves a sum in $[p, 2^n)$ unreduced or, for one addition pair, the top-bit phase comparison ties. Such a tie is visible only when the carry's X outcome is 1; the three subtraction pairs with $x = p - 1$, $y \in \{0, 1, c\}$ also tie and appear under other transcripts. The signed- c identity of Appendix E is checked on $s \in \{0, 1, 2^{256} - 1, 2^{224} - 1, 2^{224}\}$ and 1,000 pseudorandom 256-bit values; each must satisfy it exactly, with $0 \leq \Sigma_{hi} \leq c - 3$ and $0 \leq c \Sigma_{hi} < 2^{65}$.

D.7 Complete point addition

A random group multiple is chosen as the accumulator and a random signed-table address as the window value, and the result is compared with independent classical curve addition while all lookup and arithmetic phases are checked, at both selected window widths with the final schedule. In every tested case the output matched, the state carried no unwanted phase, all temporary registers returned to zero, and exactly 530 qubits remained live.

D.8 Certificate

Certification is a separate experiment using independent uniform rejection samples and the frozen audit predicate, with no boundary fixtures, schedule tuning or early stopping; Lemma 7.2 converts its two-million-sample result to a confidence bound. The 8,063 re-checked rows are all 94 failing samples plus every 251st sample by index $(0, 251, 502, \dots)$, which adds 7,969 good rows; for each, an independent big-integer implementation of the recurrence must give the same first audit event at the same step as the recorded observation. The 380 binomial coverage cases take $N_s \in \{8, 16, 32, 64, 128\}$, $\xi \in \{1, 2, 3, 4\}$ and a true failure probability $h = i/20$ for $i = 1, \dots, 19$; in each, the exact binomial probability that the bound h^+ of Lemma 7.2 falls below h must be at most $2^{-\xi}$. A further 10,000 checks confirm the integer square-root rounding used in h^+ : for every $v < 10,000$ the rounded root σ satisfies $\sigma^2 \geq v$ and, for $v > 0$, $(\sigma - 1)^2 < v$; at $v = 0$ it must be $\sigma = 0$. The certificate's own argument 25,389 gives $\sigma = 160$, since $159^2 = 25,281 < 25,389 \leq 25,600$.

E Explicit circuits

This appendix gives the gates, loop orders and measurement corrections that the main text describes in prose. Wires within a register are ordered from least to most significant, and slices $A[i : j]$ exclude the upper endpoint. Classical tables and loop bounds are part of the circuit specification, not quantum registers. Every allocation is a clean $|0\rangle$ wire, and a wire is released only after the stated cleanup. In gate lists, $CX(u \rightarrow v)$ maps $v \mapsto v \oplus u$, $CCX(u, v \rightarrow z)$ maps $z \mapsto z \oplus uv$, $CZ(u, v)$ multiplies an amplitude by $(-1)^{uv}$ and $Z(u)$ by $(-1)^u$. In the listings the arrow is written $\rightarrow$.

E.1 Measurement and erasure conventions

MX denotes a destructive X-basis measurement followed by reset of the measured wire. With outcome μ , its action on a stored Boolean value z is

$$|z\rangle \mapsto 2^{-1/2}(-1)^{\mu z}|0\rangle.$$

This fixes both the relative phase and the normalization.

Two AND-erasure rules follow. If $z = uv$ was computed into a clean wire, measure z and apply $\text{CZ}(u, v)$ when $\mu = 1$. If instead $z = u(1 \oplus v)$, apply both $\text{CZ}(u, v)$ and $\text{Z}(u)$ when $\mu = 1$, since $u(1 \oplus v) = u \oplus uv$. We call these EraseAND^+ and EraseAND^- . When the correction is applied, the controls must still hold the values used in the predicate. Neither rule uses a Toffoli.

All arithmetic uses the measured majority of [3]. For a clean target t put

$$\text{maj}(a, b, d) = ab \oplus ad \oplus bd.$$

The macro $\text{MAJ}(a, b, d \rightarrow t)$ CNOTs d into a and b , applies $\text{CCX}(a, b \rightarrow t)$, CNOTs d into t , and reverses the first two CNOTs. It uses one Toffoli and preserves its inputs, because $(a \oplus d)(b \oplus d) \oplus d = \text{maj}(a, b, d)$. When d is the known constant 0, every gate involving d is omitted. To erase $t = \text{maj}(a, b, d)$, measure t with outcome μ and apply

$$\text{eraseMAJ}(a, b, d \rightarrow t) : \quad (\text{CZ}(a, b) \text{CZ}(a, d) \text{CZ}(b, d))^\mu.$$

The case $d = 0$ uses only $\text{CZ}(a, b)^\mu$. Every branch is then the input-preserving map times $2^{-1/2}$, also on superposed inputs entangled with other registers.

Let X, Y be w -bit words. $\text{ADD}_w(X; Y)$ performs $Y \leftarrow X + Y \bmod 2^w$. (Here d_i is a carry.) With $d_{-1} = 0$ it computes the carries $d_i = \text{maj}(X_i, Y_i, d_{i-1})$ for $0 \leq i < w - 1$, then XORs X_i and, for $i > 0$, d_{i-1} into each Y_i . Carries are erased in descending order: before erasing d_i , undo those two XORs on Y_i , apply eraseMAJ , then restore the sum bit. Subtraction complements all target bits before and after, since $\sim(\sim Y + X) = Y - X \bmod 2^w$; it is emitted directly with measured cleanup, not by reversing a measured list. The primitive costs are:

- ADD_w : $w - 1$ Toffolis and $w - 1$ carry wires;
- IADD_w , which also computes d_{w-1} , erases only the lower carries and returns d_{w-1} as a live carry-out: w Toffolis;
- a w -bit strict comparison preserving X, Y , with carries $d_i = \text{maj}(Y_i, 1 - X_i, d_{i-1})$ and $d_{w-1} = [X < Y]$ (NOT gates around each majority supply the complement; the carries are erased with complemented-input corrections): w Toffolis;
- $\text{PCMP}_w(X, Y; \mu)$, which needs only the phase $(-1)^{\mu[X < Y]}$ and applies the last majority as its three diagonal CZ terms without building its target: $w - 1$ Toffolis;
- $\text{LCOR}_\kappa(Y, H)$, which adds a classically specified or coherently loaded κ -bit word $H \bmod 2^\kappa$ to $Y_0, \dots, Y_{\kappa-1}$ with ADD_κ , propagates no carry above bit $\kappa - 1$, and unloads the word: $\kappa - 1$ Toffolis. A constant or a linear XOR of control bits is loaded with NOT and CNOT gates only; nonlinear controls are supplied and charged separately.

E.2 Pruned unary row loading

Let L be the number of rows and $h = \lceil \log_2 L \rceil$. Let the row address bits be $x_0, \dots, x_{h-1}$, and let $\Omega_0, \dots, \Omega_{L-1}$ be the public row words (the concatenated bank words of a row). The output word O starts at zero. Allocate $\max(0, h - 1)$ prefix wires $z_0, \dots, z_{h-2}$. $\text{Emit}(v, \Omega)$ applies $\text{CX}(v \rightarrow O_j)$ for every j with $\Omega_j = 1$; it contains no nonlinear gate. In the recursion below, b is the recursion level and u the first row of the current range.

```

Visit(control v, level b, row start u):
    if u >= L: return
    if b < 0:
        Emit(v, Omega[u])
        return
    CCX(v, x[b] -> z[b])
    Visit(z[b], b-1, u + 2^b)
    CX(v -> z[b])
    Visit(z[b], b-1, u)
    EraseAND-minus(v, x[b] -> z[b])

```

For $h \geq 2$ the top level is

```

Visit(x[h-1], h-2, 2^(h-1))
X(x[h-1])
Visit(x[h-1], h-2, 0)
X(x[h-1])

```

For $h = 1$, emit row 1 controlled by x_0 if that row exists, then row 0 controlled by the temporarily complemented x_0 . For $h = 0$, write the only word with X gates; this case does not occur in our plans.

On entry, v indicates the fixed high prefix. The Toffoli gate sets $z_b = vx_b$ and selects the right child. The CNOT changes it to $v(1 \oplus x_b)$ and selects the left child. Neither subtree changes v or x_b , so EraseAND⁻ sees exactly its predicate. Induction on b proves both the loading map and the exact local phase cleanup. A row outside the table has no emission, so its output is zero. Only nodes whose whole starting range lies outside the table are pruned; a one-sided internal node still executes its Toffoli gate. No sparsity of the public words is used.

The Toffoli count $U(L)$ of Section 4.4 is fixed by the recurrence

$$V_L(u, b) = \begin{cases} 0, & u \geq L \text{ or } b < 0, \\ 1 + V_L(u, b-1) + V_L(u + 2^b, b-1), & \text{otherwise,} \end{cases}$$

$$U(L) = \begin{cases} 0, & h < 2, \\ V_L(0, h-2) + V_L(2^{h-1}, h-2), & h \geq 2. \end{cases}$$

For $L = 2^h$ with $h \geq 1$ this gives $U(L) = L - 2$. For other L it must be evaluated as written; for example $U(2731) = 2735$, not 2729. The six values used by the selected plans of Section 4.4 are:

L	16384	2731	32768	5462	87382	10923
$U(L)$	16382	2735	32766	5466	87388	10928

The stack uses $h - 1$ wires, because each recursion depth uses its own wire and reuses it for its two children in turn.

E.3 The reversible divide-by-three cell

The cell acts on three wires (r_0, r_1, x) , where $r_3 = r_0 + 2r_1$ is the remainder and x is an address bit. Its gates, from top to bottom, are

```

CX (r0 -> r1)
CCX(r0, x -> r1)
CX (r1 -> r0)
CX (x -> r0)
CX (r0 -> x)
CCX(r0, x -> r1)

```

Its complete truth table follows. The two rows with $r_3 = 3$ are unreachable from a clean start; they are listed to show that the cell is a permutation of all eight basis states.

Input r_3	Input x	Output r_3	Output x
0	0	0	0
0	1	1	0
1	0	2	0
1	1	0	1
2	0	1	1
2	1	2	1
3	0	3	1
3	1	3	0

On a valid remainder it maps $(r_3, x) \mapsto ((2r_3 + x) \bmod 3, \lfloor (2r_3 + x)/3 \rfloor)$. Start with both remainder wires at zero and apply the cell to the a address bits from most to least significant. After each bit, the consumed prefix holds its long-division quotient and the remainder wires hold its remainder. At the end the address wires hold $\lfloor m/3 \rfloor$ and the remainder is $m \bmod 3$. The forward pass costs $2a$ Toffolis. Reversing all cells, including their gate order, restores m and clears the remainder for another $2a$, giving the $4a$ of Section 4.4.

For $B = 3 \cdot 2^k$, the low k quotient bits are the binary bank address and the next $\lceil \log_2 L \rceil$ bits are the row address. Higher quotient bits are known zero and stay in the address allocation. All a bits take part in the division, not only the bank-selection bits.

E.4 One segment as a lifetime-ordered procedure

The magnitude address, the sign and the retained first-segment bits stay live throughout. Let D be the table being read and $D(i)_{[o, o+b_s)}$ bits o to $o + b_s - 1$ of entry i , with padded entries zero. A segment runs as follows.

1. For a ternary bank count, allocate two remainder wires and run the forward divider. For a binary bank count, use the address split $m = Bg + t$ directly.
2. Allocate B clean b_s -bit banks. Apply the unary loader to the row words $\Omega_g = \sum_{j=0}^{B-1} 2^{jb_s} D(Bg + j)_{[o, o+b_s)}$. Release the cleaned prefix stack.
3. For a ternary bank count, select the remainder position within every triple. Then select the addressed bank into bank zero with the binary swap tree of Section 4.4. The static swap permutation is kept as classical metadata for the phase formula; no quantum copy of it is made.
4. Measure every bit of banks 1 to $B - 1$ in the X basis, reset them, record their masks and release the wires. The selector is not reversed. Bank zero holds exactly the requested segment and stays live.
5. For a ternary bank count, reverse the divider and release its two zero remainder wires.

Each step preserves the selected data and the address on every basis label. The measurements leave only the phase of Lemma 4.3 and no entangled bank data. When the next segment starts,

$o + b_s$ selected bits are live, and no earlier unselected bank, prefix stack or remainder wire remains. After an ordinary word read, the 256-bit selected word is used by its arithmetic and then X-measured, so the whole word is free before the next kernel or column read. In the initial lookup all 512 selected bits stay live as the accumulator.

The peaks follow from the occupancy formula of Section 4.4, with terms in the same order. For a width-17 window the two segments give

$$513 + 16 + 1 + 4(228) + 13 = 1455, \quad 513 + 16 + 1 + 228 + 24(28) + 2 + 11 = 1443.$$

For width 18 they give

$$512 + 17 + 1 + 4(228) + 14 = 1456, \quad 512 + 17 + 1 + 228 + 24(28) + 2 + 12 = 1444.$$

For the initial lookup, whose 18 address wires are the only other live wires,

$$18 + 3(473) + 2 + 16 = 1455, \quad 18 + 473 + 24(39) + 2 + 13 = 1442.$$

The retained 228- or 473-bit prefix is never reused as workspace. The segment gate sums $U(L) + (B - 1)b_s + 4a\mathbf{1}_{3|B}$ are

$$\begin{aligned} 16382 + 3(228) &= 17066, & 2735 + 23(28) + 4(16) &= 3443, \\ 32766 + 3(228) &= 33450, & 5466 + 23(28) + 4(17) &= 6178, \\ 87388 + 2(473) + 4(18) &= 88406, & 10928 + 23(39) + 4(18) &= 11897. \end{aligned}$$

The width-16 example of Table 3 has $L = 8,192$ and $1,366$, with $U(L) = 8,190$ and $1,369$, and is not part of the selected plans.

E.5 The segmentation dynamic program

Fix a , the payload length n_{pay} , the number A_{live} of other occupied wires, the number s_{live} of sign wires and the qubit budget N_q . The bank family is

$$\mathcal{B} = \{2^k : 0 \leq k \leq 8\} \cup \{3 \cdot 2^k : 0 \leq k \leq 7\},$$

restricted to $B \leq 2^a$. Let $f(o)$ be the least total Toffoli count for the first o selected payload bits, with $f(0) = 0$ and all other entries infinite. For every state o , every $B \in \mathcal{B}$ and every $1 \leq b_s \leq n_{\text{pay}} - o$ that fits in the budget, update

$$f(o + b_s) \leftarrow \min \left\{ f(o + b_s), f(o) + U(\lceil 2^a/B \rceil) + (B - 1)b_s + 4a\mathbf{1}_{3|B} \right\}.$$

Feasibility bounds the segment width explicitly:

$$b_s \leq \left\lfloor \frac{N_q - A_{\text{live}} - a - s_{\text{live}} - o - 2 \cdot \mathbf{1}_{3|B} - \max(0, \lceil \log_2 \lceil 2^a/B \rceil \rceil - 1)}{B} \right\rfloor.$$

The transition cost depends on earlier segments only through o , because all earlier unselected banks have been freed. Induction on o shows that the program searches every ordered segmentation in this family and charges its retained prefix. Backtracking gives the plans of Section 4.4. This is an optimality statement within the family, not a lower bound for all lookup designs.

E.6 The terminal Boolean phase circuit

Let $m_0, \dots, m_{a-1}$ be the bits of the magnitude address and let s be the phase split of Section 4.5.

Low-address decoder. Allocate 2^s clean wires $\chi_0, \dots, \chi_{2^s-1}$, set $\chi_0 = 1$, and process $m_0, \dots, m_{s-1}$ in increasing order. Before bit i , the first 2^i wires indicate the prefixes of the i bits processed and the rest are zero. For each $0 \leq j < 2^i$ perform

$$\chi_{j+2^i} \leftarrow \chi_{j+2^i} \oplus \chi_j m_i, \quad \chi_j \leftarrow \chi_j \oplus \chi_{j+2^i}.$$

At $i = 0$ the first line is a CNOT, since its parent is the constant one; at later levels it is one Toffoli gate. At the end $\chi_j = [m_{\text{low}} = j]$. No tree-history wires are used. The Toffoli count is $\sum_{i=1}^{s-1} 2^i = 2^s - 2$. To erase the decoder, reverse the levels: first XOR the child into the parent, which restores the parent; at $i = 0$ undo the copy with a CNOT, otherwise erase the child with EraseAND^+ on the restored parent and m_i . Finally only $\chi_0 = 1$ remains; clear it with X and release all decoder wires.

Streamed high address. Keep the decoder and traverse the $a - s \geq 1$ high bits with the Visit recursion above, now over all 2^{a-s} rows with $a - s - 1$ prefix wires. At high leaf g , replace Emit by

$$\prod_{j: F(j+2^s g)=1} \text{CZ}(v, \chi_j),$$

where v is the live high-prefix indicator. For $a - s = 1$ use the high bit and its temporary complement directly. Erase the prefix stack with its local EraseAND^- corrections, then erase the decoder. For each basis address exactly one low wire and one high leaf indicator are one, so the product of the CZ phases is $(-1)^{F(m)}$. A constant-one F yields the global minus sign, not an omitted phase. The traversal costs $2^{a-s} - 2$ Toffolis, so the total is $C_{\text{phase}}(a, s) = 2^s + 2^{a-s} - 4$. The selected splits give

$$256 + 256 - 4 = 508, \quad 256 + 512 - 4 = 764, \quad 512 + 512 - 4 = 1020.$$

For ordinary reads all data banks are measured before this circuit runs; the accumulator and all 18 address wires stay live. In the initial lookup the selected point stays live and has zero selected-slot mask in F , so no copy of it is needed.

Branchwise cleanup. Let N_X be the number of data-bank X measurements and fix all their outcomes. Before the terminal correction, the data-bank part of the lookup acts as

$$|m\rangle|\psi\rangle \mapsto 2^{-N_X/2}(-1)^{F(m)}|m\rangle V_m|\psi\rangle,$$

where V_m is the intended consumed-column arithmetic, or the retained point preparation for the initial lookup. All released bank and divider wires are zero. This assumes only the arithmetic's own good-domain contract. The phase circuit multiplies by $(-1)^{F(m)}$, leaving an input-independent scalar times the intended map. By linearity this holds on arbitrary superpositions, including with an external reference register.

E.7 The fused integer stages

The listings below fix the wire order and every non-Clifford gate of the two integer stages of Section 5.4. The source word is the conditionally complemented R_b . The last argument of MAJ and eraseMAJ is the target; an argument 0 is a known constant, not a wire. Every release is of a wire that is zero on the good domain.

Forward stage, $2S + R_b$. Allocate clean wires **lo** and **hi** and carry wires $d_1, \dots, d_{n-1}$; d_0 is a formal zero. Let $A = (\text{lo}, S_0, \dots, S_{n-1})$.

```

for i = 1, ..., n-1:
    MAJ(Rb[i], A[i], d[i-1] -> d[i])
for i = 0, ..., n-1:
    CX(Rb[i] -> A[i])
    if i > 1: CX(d[i-1] -> A[i])
CCX(A[n], d[n-1] -> hi)
CX(d[n-1] -> A[n])
for i = n-1, ..., 1:
    CX(Rb[i] -> A[i])
    if i > 1: CX(d[i-1] -> A[i])
    eraseMAJ(Rb[i], A[i], d[i-1] -> d[i])
    if i > 1: CX(d[i-1] -> A[i])
    CX(Rb[i] -> A[i])
release d[1], ..., d[n-1]
return Z = A[0:n], q0 = A[n], q1 = hi

```

The first $n - 1$ Toffolis build the carries. The last one computes the high quotient bit from the top target bit and the final carry, and the former top target wire becomes q_0 . The returned word Z holds ℓ until the short correction updates it. No new n -bit output register is needed. The largest local overlap is $2n + 1$ input wires plus $n + 1$ scratch wires, or $3n + 2$.

Inverse stage. After q_0, q_1 are prepared and the short correction is undone, the low word holds $\ell' = 2^\kappa \lfloor Z/2^\kappa \rfloor + ((Z - H_b(q)) \bmod 2^\kappa)$. Set $A = (\ell'_0, \dots, \ell'_{n-1}, q_0)$, allocate $d_1, \dots, d_{n-1}$ and set $d_0 = 0$. By the parity identity for q_0 in Section 5.4, the low bits of ℓ' and R_b agree, so the subtraction has no initial borrow.

```

for i = 1, ..., n-1:
    X(A[i])
    MAJ(Rb[i], A[i], d[i-1] -> d[i])
    X(A[i])
for i = 0, ..., n-1:
    CX(Rb[i] -> A[i])
    if i > 1: CX(d[i-1] -> A[i])
CX(d[n-1] -> A[n])
m' = MX(q1)
if m' == 1: CZ(d[n-1], A[n])
for i = n-1, ..., 1:
    CX(Rb[i] -> A[i])
    if i > 1: CX(d[i-1] -> A[i])
    X(A[i])
    eraseMAJ(Rb[i], A[i], d[i-1] -> d[i])
    X(A[i])
    if i > 1: CX(d[i-1] -> A[i])
    CX(Rb[i] -> A[i])
release d[1], ..., d[n-1], A[0], q1
return S = A[1:n+1]

```

With $v = q_0 M + \ell' - R_b$, the stage outputs $(v \bmod 2M)/2$. If the borrow out of bit $n - 1$ is d_{n-1} , the updated top bit is $A_n = q_0 \oplus d_{n-1}$, and the full borrow is $d_{n-1}(1 - q_0) = d_{n-1}A_n$; this is the

displayed CZ. The measurement therefore leaves the phase $(-1)^{m'(q_1 \oplus [v < 0])}$ and no other, and on a forward-good image the exponent vanishes. The $n - 1$ borrow cells give the integer-stage count of the inverse cell.

Complete-cell recipe. Put $t_0 = bq_0$, $t_1 = bq_1$ and $h_{bq} = H_b(q) \bmod 2^\kappa$. The correction table is loaded into a clean κ -bit word by XORing, under the controls $1, q_0, q_1, b, t_0, t_1$ respectively, the masks

$$h_{00}, \quad h_{01} \oplus h_{00}, \quad h_{02} \oplus h_{00}, \quad h_{10} \oplus h_{00}, \quad h_{11} \oplus h_{10} \oplus h_{01} \oplus h_{00}, \quad h_{12} \oplus h_{10} \oplus h_{02} \oplus h_{00},$$

which is exact because $q_0q_1 = 0$. The forward cell complements R by CNOTs from b , runs the forward integer stage, computes t_0, t_1 with two Toffolis, loads the table, adds it into the low κ bits of Z with ADD_κ , unloads it, and erases t_0, t_1 with EraseAND^+ . It then measures q_1 with outcome m' . If $m' = 1$ it computes the raw comparison flag

$$t = \left[\left\lfloor Z/2^{n-\delta} \right\rfloor < \left\lfloor R_b/2^{n-\delta} \right\rfloor \right], \quad \hat{q}_1 = (1 - q_0)t.$$

It applies $Z(t) \text{CZ}(q_0, t)$, implementing the sign $(-1)^{(1-q_0)t}$ without a separate AND Toffoli, and erases the comparison; the resource count charges this branch at every cell. Finally it clears q_0 by three CNOTs and one NOT from $q_0 = R_b[0] \oplus Z[0] \oplus (1 - b)$, releases the quotient wires, and undoes the complements of R . The inverse cell complements R , prepares q_0 by the same parity expression, computes the δ -bit comparison, obtains q_1 with one Toffoli controlled by $1 - q_0$ and the flag, and erases the comparison. It then loads the same table, subtracts it in the low κ bits, unloads it, erases t_0, t_1 , runs the inverse integer stage, and undoes the complements of R . These two recipes fix the whole cell in either measurement branch.

E.8 The two history codecs

A symbol is $(a, m) \in \{(0, 0), (1, 0), (1, 1)\}$, with ternary digit $t = (a \oplus m) + 2m$. All codec gates are NOT, CNOT and Toffoli gates, so their phase is +1 everywhere, including on unused encodings, and reversing a codec's gate order gives its exact inverse.

Three-symbol compressor. Number the six input wires $(a_0, m_0, a_1, m_1, a_2, m_2)$ as $0, \dots, 5$. The compressor of [2], in this wire convention, is the following list, applied in order:

CX(1 -> 0)	CX(3 -> 2)	CX(5 -> 4)	CX(0 -> 2)
CX(5 -> 3)	X(4)	CCX(1, 3 -> 5)	CX(1 -> 4)
X(2)	CCX(3, 4 -> 5)	CCX(4, 5 -> 1)	CCX(2, 5 -> 0)
CCX(0, 1 -> 5)			

It has five Toffolis, six CNOTs and two NOTs. Wire 5 is zero on all 27 valid symbol triples; wires 0 to 4 are kept. Ordering the valid inputs by $j = t_0 + 3t_1 + 9t_2$, the five-bit outputs are:

j	+0	+1	+2	+3	+4	+5	+6	+7	+8
0	20	17	6	16	21	2	31	27	15
9	4	1	22	0	5	18	12	9	30
18	28	25	14	24	29	10	23	19	7

All are below 32 and distinct. Expansion appends a zero wire and reverses the list.

Five-symbol codec. The decoder maps an eight-bit integer $j = \sum_{i=0}^4 t_i 3^i$, $0 \leq j \leq 242$, to its five digits. Packing is the reversed decoder, preceded by one $\text{CX}(m_i \rightarrow a_i)$ per symbol; unpacking follows the decoder with the same five CNOTs. The decoder reuses the division cell of this

appendix; on the eight inputs $r_0 + 2r_1 + 4x$ it is the permutation $(0, 2, 5, 7, 1, 4, 6, 3)$ and costs two Toffolis.

To divide a λ -bit word G with $0 \leq G \leq g_{\max}$ by three, start with two clean remainder wires (r_0, r_1) . Let $g_1 = G_{\lambda-1}$ and $g_0 = G_{\lambda-2}$ be the two leading bits. If $\lfloor g_{\max}/2^{\lambda-2} \rfloor = 3$, apply $\text{CCX}(g_1, g_0 \rightarrow r_0)$, $\text{CX}(r_0 \rightarrow g_1)$, $\text{CX}(r_0 \rightarrow g_0)$; if this prefix is at most two, omit these three gates, since $g_1 g_0 = 0$ on that domain. Then swap r_0 with g_0 and r_1 with g_1 , three CNOTs per swap. Process $G_{\lambda-3}, \dots, G_0$, in that order, through the division cell with the same remainder wires. The remainder is $G \bmod 3$, and the word wires hold $\lfloor G/3 \rfloor$ with zero high bits free for reuse.

The decoder starts with address wires $0, \dots, 7$ and applies this procedure four times:

Pass	Address wires	Remainder wires	Input upper bound	Toffolis
0	$0, \dots, 7$	$(8, 9)$	242	$1 + 2(6) = 13$
1	$0, \dots, 6$	$(7, 10)$	80	$2(5) = 10$
2	$0, \dots, 4$	$(6, 5)$	26	$1 + 2(3) = 7$
3	$0, \dots, 3$	$(4, 11)$	8	$2(2) = 4$

The last digit ends on wires $(0, 1)$, and the four remainder pairs hold t_0, t_1, t_2, t_3 in order. All other wires are zero. The codec uses twelve wires, eight stored bits and four expansion bits, and exactly $13 + 10 + 7 + 4 = 34$ Toffolis per direction. The division recurrence proves it on all $3^5 = 243$ valid words. Each block of the history is packed as soon as its symbols exist, and a replay expands only the block it is reading and re-packs it before expanding the next.

Recorder occupancy. In recording step i , the adder phase holds U and V at work width $W_i + 2$ each, $W_i + 1$ carry wires, the other 256-bit operand, the orientation bit and 18 controls (together $3W_i + 280$), the η_i already-packed history bits, and the ϕ_i wires of the block being filled. Here $\eta_i = 5\lfloor i/3 \rfloor$ while the 117 triples are being filled ($i < 351$) and $\eta_i = 585 + 8\lfloor (i - 351)/5 \rfloor$ afterwards; $\phi_i = 6$ for a triple block and 12 for a five-symbol block. The comparator phase is the same with the e_i comparator carries in place of the adder carries. The maximum of $3W_i + 280 + \eta_i + \phi_i$ is 1,144, at $i = 54$ where $W_{54} = 256$, $\eta_{54} = 90$ and $\phi_{54} = 6$.

E.9 Integer square, signed slices and their workspace

This subsection fixes the bit endpoints and lifetimes used in Section 6.2. Slices $x_{[a,b)}$ include bit a and exclude bit b . For an m -bit source x and a clean $2m$ -bit target z , use the following procedure. For $m = 1$ the entire circuit is $\text{CX}(x_0, z_0)$; otherwise:

1. Subtract zero-extended x from $z_{[0,m+1)}$ modulo 2^{m+1} .
2. For $i = 0, \dots, m - 2$, take the target $V = z_{[2i+1,m+i+2)}$ and zero-extend $x_{[i+1,m)}$ to a word A of length $m + 1 - i$. Apply $\text{CX}(z_{m+i}, z_{m+i+1})$ to sign-extend the accumulated partial result. Complement every wire of A controlled by x_i , subtract A from V modulo its word length, undo those complements, and release the restored padding.
3. Apply $\text{CX}(x_{m-1}, z_{2m-1})$.

Step 1 costs m Toffolis and iteration i costs $m - i$, which gives the count $\frac{1}{2}n(n+3) - 1$ of Section 6.2 with $n = m$. The inverse on the image reverses this order and exchanges word additions and subtractions, each with its own measured carry cleanup. It is not the reversal of a list containing irreversible measurements, nor is the forward circuit specified as a square accumulator on a

dirty target. To verify the signed intermediate values, put $X_{\text{lo}} = x \bmod 2^i$ and $X_{\text{hi}} = \lfloor x/2^i \rfloor$. Immediately before iteration i the partial integer value is

$$P_i = X_{\text{lo}}^2 + 2^i X_{\text{hi}}(2X_{\text{lo}} - 2^i), \quad x^2 - P_i = 2^{2i} X_{\text{hi}}(X_{\text{hi}} + 1).$$

Substitution of $x = X_{\text{lo}} + 2^i X_{\text{hi}}$ proves the identity initially and after each update. The bounds on $X_{\text{lo}}, X_{\text{hi}}$ give $-2^{m+i} < P_i < 2^{m+i}$, so the displayed CNOT is valid sign extension in $m + i + 1$ active signed bits. The untouched low bits already equal the square because $X_{\text{hi}}(X_{\text{hi}} + 1)$ is even. The final difference is $x_{m-1}2^{2m-1}$, removed by the last CNOT. This proves the clean-target square identity including its measured phases.

Define $\text{Slice}_\sigma(s; o, w, t; y)$, which adds $\sigma 2^t s_{[o, o+w)}$ on its good domain. For negative sign, conjugate the following positive operation by the modular complement of Section 6.1. If $t + w + \rho < n$, zero-extend the source by ρ bits and add it into $y_{[t, t+w+\rho)}$. If $t + w = n$ and $w \geq \delta_o$, add into $y_{[t, n)}$ with carry-out, apply the low- κ_o correction, X-measure the carry and repair its phase with the top- δ_o comparison of the updated target and source. These are the two slice cases used; the case $t + w < n \leq t + w + \rho$ never occurs.

Let $\mathcal{T} = ((32, 1), (10, 1), (6, -1), (4, 1), (0, 1))$. For a square word of length $n_s \in \{256, 258\}$, addition of $\sigma 2^{128}s$ consists of

$$\text{Slice}_\sigma(s; 0, 128, 128; y), \quad \text{Slice}_{\sigma e}(s; 128, n_s - 128, d; y) \quad ((d, e) \in \mathcal{T}),$$

in that order. For a 256-bit word, addition of σcs uses the exact identity

$$cs = \sum_d \epsilon_d 2^d (s \bmod 2^{256-d}) + 2^{256} \sum_{d>0} \epsilon_d \lfloor s/2^{256-d} \rfloor,$$

over the signed digits $\epsilon_d 2^d$ of c , whose second sum is Σ_{hi} . It begins with $\text{Slice}_{\sigma e}(s; 0, 256 - d, d; y)$ for the five ordered terms. Prepare a clean 65-bit word $C_{65} = c\Sigma_{\text{hi}}$ by the following twenty exact word additions: for each $(d_1, e_1) \in \mathcal{T}$ with $d_1 > 0$, and then each $(d_2, e_2) \in \mathcal{T}$, add $e_1 e_2 s_{[256-d_1, 256)}$, zero-extended, into $(C_{65})_{[d_2, 65)}$. Here

$$\Sigma_{\text{hi}} = s_{[224, 256)} + s_{[246, 256)} - s_{[250, 256)} + s_{[252, 256)}, \quad 0 \leq \Sigma_{\text{hi}} \leq c - 3.$$

Temporary wrap modulo 2^{65} is harmless because $0 \leq c\Sigma_{\text{hi}} < c^2 < 2^{65}$. Apply $\text{Slice}_\sigma(C_{65}; 0, 65, 0; y)$, then reverse the twenty additions before releasing C_{65} . Each preparation costs $4 \sum_{d \in \{32, 10, 6, 4, 0\}} (64 - d) = 1,072$ Toffolis.

Slice costs. The low 128 bits of a 256-bit square z are added at offset 128 as a full-width modular addition (integer add with carry-out, constant correction and phase comparison: $128 + 64 + 31 = 223$). A local slice of the high half costs $128 + 32 - 1 = 159$ Toffolis (161 for the 130-bit high half of the 258-bit sum square), and a negative digit adds 128 for the complement conjugation. Adding $2^{128}z$ therefore costs $223 + 4 \cdot 159 + 287 = 1,146$ for a half square ($287 = 159 + 128$ for the one negative digit) and $351 + 4 \cdot 289 + 161 = 1,668$ for the negated sum square, in which four slices are negative ($289 = 161 + 128$). For $-cz$, each signed digit $\epsilon_d 2^d$ contributes the full-width modular addition of the low $256 - d$ bits of z at offset d ($(256 - d) + 95$ Toffolis, plus 128 for a negative sign); the five slices cost $447 + 469 + 345 + 475 + 479 = 2,215$. The buffer C_{65} is prepared by the twenty exact signed additions (1,072), added to y as a 97-bit padded slice with complement conjugation (224), and unprepared (1,072), so adding $-cz$ costs 4,583.

The following live-wire bounds include both 256-bit caller words and all 18 controls. A carry word is released before the next correction or comparison scratch is allocated. Source padding is retained only during the addition that uses it.

Table 9. Square-subtraction lifetimes. The 129-bit sum occupies the original high-half source plus one wire. The largest simultaneous allocation is 1,042; stage bounds are not added together.

Stage	Simultaneous allocation	Bound
Half-square compute or inverse	$530 + 256 + 2 + 128$	916
Half-square full-width addition or subtraction	$530 + 256 + 256$	1,042
Half-square local slice	$530 + 256 + 32 + 159$	977
Wrapped-slice correction	$530 + 256 + 1 + 65 + 64$	916
Prepare the $c\Sigma_{hi}$ buffer	$530 + 256 + 65 + 61 + 64$	976
Use that buffer, including complement	$530 + 256 + 65 + \max(32 + 96, 65 + 64)$	980
Form or restore the 129-bit source sum	$530 + 1 + 1 + 128$	660
Sum-square compute or inverse	$530 + 1 + 258 + 2 + 129$	920
Sum-square local slice	$530 + 1 + 258 + 32 + 161$	982
Sum-square wrap and correction	$530 + 1 + 258 + \max(128, 1 + 65 + 64)$	919

The summands in the table are as follows. 530 counts the caller’s two field words and the 18 controls. 256 is the square buffer (258 for the sum square). 2 counts the sign and padding wires. 128 and 129 are the carries of the squarer. 32 is local padding. 159 and 161 are the local-slice carries. 1 is a carry-out; in the sum-stage rows a further 1 is the extra wire of the 129-bit sum. 65 is the correction word and 64 its carries. 61 counts the zero-extension wires that pad the shortest, 4-bit source of the buffer preparation to the 65-bit word; the carries of those additions are the 64 already counted. 96 counts the carries of the 97-bit padded slice.

F Error sets, densities and the ideal anchor

F.1 Fused replay: complete exceptional-set proof

We prove Theorem 7.1 for the parameters and encoding of Section 5.4. Put $M = 2^n$, $\Lambda = 2^\kappa$, $\Delta = 2^{n-\delta}$ and $\mu = M/2 - c$. Fix a valid denominator history and draw one uniform $y \in \mathbb{F}_p$. At every ideal step there are fixed field coefficients α, β such that

$$r = \beta y, \quad s = \alpha y.$$

These two registers need not be independent. If a coefficient is nonzero, the corresponding register value is uniform on $\mathbb{F}_p$ as y varies. A zero coefficient must instead be treated directly.

Value errors. With the quotient and correction table of Section 5.4, let $A = \ell + H_b(q)$ be the integer before discarding the low-word carry and let $Z_* = (2S + R_b - K_b) \bmod p$ be the ideal canonical encoded result. The identity $Z_* = A \bmod p$ follows from $M \equiv c \pmod{p}$. Also $A \geq 0$: the sole negative correction is $1 - c$, for $b = 1, q = 0$, where $\ell = 2S + R_b \geq c$. For $b = 0$ and $q = 0, 1, 2$, the upper bounds on A are respectively $M + c - 1$, $M + 2c - 1$, $M - 3$; for $b = 1$ they are p , M , $p - 2$. Thus $A \leq p + 3c - 1$. The hypothesis $p > 8c$ gives $3c - 1 < p$, so $A < 2p$.

If $A \geq p$, then $Z_* < 3c$. If $A < p$, a value error requires a positive correction of at most $3c$ to cross a multiple of Λ , or the negative correction $1 - c$ to borrow across one. In terms of the ideal residue, all these events are contained in

$$Z_* \bmod \Lambda \in \mathcal{B} = [0, 3c) \cup [\Lambda - c + 1, \Lambda), \quad |\mathcal{B}| = 4c - 1,$$

with integer interval endpoints. The hypothesis $4c \leq \Lambda$ ensures that $\mathcal{B}$, of size $4c - 1$, lies inside one Λ -word. This containment counts value errors regardless of whether the associated phase happens to be correct.

The updated field coefficient is $\gamma = 2\alpha + (-1)^b\beta$. When $\gamma \neq 0$, $E(\gamma y)$ is uniform on the canonical residues, so at most $(4c - 1)\lceil p/\Lambda \rceil$ values of y belong to this set. When $\gamma = 0$, the ideal result is instead μ , and

$$\mu \bmod \Lambda = \Lambda - c \notin \mathcal{B}$$

because $\Lambda \mid M/2$ and $\Lambda \geq 4c$. This constant family has no value error; uniformity has not been assumed for it.

Phase-only errors. On a value-correct input, $A = Z_*$ and no low-word carry was discarded. Indeed, the circuit output is $Z = A - j\Lambda$ with $j \in \{-1, 0, 1\}$ and the ideal is $Z_* = A - kp$ with $k \in \{0, 1\}$. Equality forces $j\Lambda = kp$, which is impossible for $k = 1$ since p is odd; so $j = k = 0$. The integer identity then gives

$$Z_* + K_b - R_b = 2S - qp, \quad q_1 = (1 - q_0)[Z_* + K_b < R_b].$$

The circuit compares Z with R_b , the true predicate compares $Z + K_b$ with R_b , and $|K_b| \leq c$; hence a disagreement with the high- δ -bit comparison requires

$$|Z_* - R_b| \leq \Delta + c.$$

Numbers in distinct high-bit bins preserve their strict order; numbers in the same bin differ by less than Δ . Moving the threshold by K_b , whose magnitude is at most c , extends this interval by at most c . Modulo p , the difference is

$$Z_* - R_b \equiv \begin{cases} 2s, & b = 0, \\ 2s - (2c - 1), & b = 1. \end{cases}$$

If $\alpha \neq 0$, this expression is uniform. The interval of radius $\Delta + c$ contains at most $2\Delta + 2c + 1$ residue classes, so it contains at most that many values of y .

For the remaining family $\alpha = 0$, we have $S = \mu$ identically. If $b = 0$, the raw sum is $M - 2c + R$. Quotient zero would give $A = p + R \geq p$, whereas a value-correct input has $A = Z_* < p$ by the preceding paragraph. Thus every value-correct case has $q = 1$ and $1 - q_0 = 0$, so both high-quotient predicates vanish. If $b = 1$, the quotient is zero or one. Quotient one again makes that factor zero. For quotient zero on a value-correct input, $Z_* - R_b = M - 3c + 1 > 0$, so both strict comparisons are false. Hence a zero old target has no phase-only errors. These arguments also include $\alpha = \beta = 0$.

Combining the two parts, one cell's conservative exceptional fraction is

$$q_{\text{cell}} \leq \frac{(4c - 1)\lceil p/2^\kappa \rceil + 2^{n-\delta+1} + 2c + 1}{p}.$$

This charges a wrong phase predicate even on a measurement outcome that would conceal its effect.

Inverse cells. Call a forward input *good* if its value is correct and $q_1 = \hat{q}_1$, whatever the measurement outcome; every other input lies in a set charged by the bound above. The bands $Z_* \bmod \Lambda \in \mathcal{B}$ and $|Z_* - R_b| \leq \Delta + c$ contain every bad input but may also contain good ones, so the argument below uses goodness itself, not the bands. Fix the inverse cell's input family

(the forward output) with coefficient γ . The inverse cell receives (R, Z_*, b) with $Z_* = E(\gamma y)$. The ideal step $s \mapsto 2s + (-1)^b r$ is a bijection of $\mathbb{F}_p$ for each fixed r , so this input has a unique ideal preimage (R, S, b) . Suppose that preimage is good. Then Z_* is the forward output Z , and three facts hold. The Clifford parity gives the true q_0 . The inverse evaluates the same truncated comparison on the same pair (Z_*, R_b) as the forward cell, so it obtains $\hat{q}_1 = q_1$. No low-word carry was discarded, so $(\ell \bmod \Lambda) + H_b(q) \in [0, \Lambda)$ and subtracting $H_b(q)$ in the low κ bits restores ℓ without a borrow. The inverse integer stage then has $v = q_0 M + \ell - R_b = 2S - 2q_1 M$, so it returns S , and $[v < 0] = q_1$ because $0 \leq 2S < 2M$; its measured phase $(-1)^{m'(q_1 \oplus [v < 0])}$ is therefore trivial (Appendix E). Hence the inverse cell's bad inputs are contained in the images of the forward cell's bad inputs on the family (β, α) with $\alpha = (\gamma - (-1)^b \beta)/2$, and they are counted by the same bound, including the degenerate cases $\alpha = 0$ or $\gamma = 0$.

Interfaces and composition. For a canonical $z < M/2$, the encoder subtracts c from $z + M/2$ using only the low κ bits. It is exact precisely when $z \bmod \Lambda \geq c$. For $z \geq M/2$, flipping the high bit alone is exact. The encoder's exact bad set is therefore

$$\{z : 0 \leq z < M/2, z \bmod \Lambda < c\}, \quad \text{of size } c2^{n-\kappa-1}.$$

The decoder is the inverse bit permutation and has the same bad-set size on ideal encoded inputs. The live boundary coefficients are y and xy or y/x ; both are uniform for each fixed nonzero denominator. A known encoded zero scratch register is cleared as the constant μ , not decoded or assigned a uniform-input error count.

Finally union-bound the first possible deviations on the **unconditioned ideal trajectory**: T cell events and two live interfaces. Conditioning later coefficients on earlier survival would not preserve their uniformity and is unnecessary. No independence between steps is used. This gives

$$q_{\text{rep}}(T) \leq T \frac{(4c-1)[p/2^\kappa] + 2^{n-\delta+1} + 2c + 1}{p} + \frac{c2^{n-\kappa}}{p},$$

including wrong values, wrong measured phases and failures of good-domain cleanup. This proves Theorem 7.1. $\square$

F.2 Bad-set counts for the coordinate adder and subtractor

This section counts the inputs on which the outer operations of Section 6, with widths $\kappa_o = 65$, $\delta_o = 32$ and padding $\rho = 32$, give a wrong value, a noncanonical result or a wrong phase. The counts follow by counting explicit integer intervals, as in the analysis of [1].

Recall the adder of Section 6.1 on canonical inputs u, v . It adds u into v with a carry-out h , adds hc into the low κ_o bits of v , measures h in the X basis, and repairs the phase with a comparison of the top δ_o bits. Over ordered canonical pairs (u, v) let

$$N_{\text{add}} = \frac{c(2p - c - 1)}{2} + \frac{c(2^{n-\kappa_o} - 1)(p - 2c - 1)}{2} + \frac{(2^{n-\delta_o} - 1)(p - c)}{2}.$$

The three summands have the following meaning.

- The first counts sums $u + v \in [p, 2^n)$. Such a sum has no carry, so the result is not reduced. A given sum is reached by $2p - 1 - (u + v)$ pairs.
- The second counts overflows of the low-word correction. They occur at sums $2^n + j2^{\kappa_o} - 1 - b$ with $1 \leq j < 2^{n-\kappa_o}$ and $0 \leq b < c$.

- The third bounds missed carry phases. Here the canonical result z satisfies $z < u$, but the top δ_o bits of z and u agree. Partition $[c, p)$ into bins of width $2^{n-\delta_o}$. This gives at most $(2^{n-\delta_o} - 1)(p - c)/2$ such ordered pairs.

The complement adds c into the low κ_o bits and then inverts every bit, which returns $p - 1 - v$ on its good domain. It fails on

$$N_{\text{comp}} = c(2^{n-\kappa_o} - 1)$$

inputs. This count takes the c upper values of every full 2^{κ_o} -word interval below p ; the last, shortened interval contains none. A subtraction is a complement, an addition and a complement. The two complements each fail on N_{comp} target values, whatever the p values of the other operand, which gives

$$N_{\text{sub}} \leq N_{\text{add}} + 2pN_{\text{comp}}.$$

F.3 The square-subtraction bound

The square subtraction of Section 6.2 is built from exact integer squares and from slice additions into the target. A slice adds a shifted piece of a square into the target modulo p . A *local* slice ends at least ρ bits below the top of the word and is added into a window padded by ρ carry bits. A *wrapping* slice reaches the top bit; it uses a carry-out, the κ_o -bit correction and the δ_o -bit phase comparison, like the adder above. A *negative* slice is conjugated by two complements.

Fix x and let the initial target be uniform in $\mathbb{F}_p$ through the independent field mask. The bad target words are as follows.

- A local slice has fewer than $2^{n-\rho}$ bad full-word targets, from a lost high carry.
- A complement or a wrapping correction has at most $c2^{n-\kappa_o}$ bad targets.
- A wrapping phase comparison has at most $2^{n-\delta_o}$ tied-prefix targets and at most c further targets from a noncanonical representative.

The schedule contains 16 local slices, 9 wrapping slices and 13 negative slices, hence 26 complement sites. The exact integer squares and the exact preparation of the high correction word add no exceptions. So there are $26 + 9 = 35$ sites of the second kind and 9 of the third. Expressing the cardinalities as fractions of 2^n and dividing by p supplies the factor $2^n/p = 1 + c/p$:

$$p_{\text{sq}} \leq \left(1 + \frac{c}{p}\right) \left(\frac{16}{2^\rho} + \frac{35c}{2^{\kappa_o}} + \frac{9}{2^{\delta_o}} + \frac{9c}{2^n}\right).$$

This bound does not assume that the target is still uniform after earlier slices have succeeded. The argument is injectivity. For fixed x , every surviving good prefix applies a fixed residue translation to the initial masked target, or a complement followed by a translation. Distinct initial targets therefore have distinct residues after the prefix, and hence distinct stored words, even when intermediate words are noncanonical. A bad-word set $\mathcal{S}$ at the next stage thus has at most $|\mathcal{S}|$ preimages among the p masks. Its unconditional first-failure probability is at most $|\mathcal{S}|/p$. The union bound runs over the ideal masked trajectory, not over a distribution conditioned on earlier success.

The result is canonical on the good path. The last negative local slice ends with a complement. If its input word is w , the successful low-word addition gives $w + c < 2^n$, so $w < p$. The final bitwise NOT returns $2^n - 1 - (w + c) = p - 1 - w \in [0, p)$, as the affine interface requires.

F.4 Transfer from field inputs to curve points

The bounds so far are stated for field inputs; the actual inputs are derived from the point pair (A, U) of Lemma 4.2. Put

$$D_0 = (r - 1)(r - K), \quad K = 2^{18}.$$

By Lemma 4.2 each point pair has probability at most $1/D_0$, so it suffices to count how many point pairs map to one field input.

Denominators. Fix a nonzero denominator $u = x_1 - x_2$ and a second point. Then the first point has x-coordinate $x_2 + u$, and at most two points have that x-coordinate. A bad denominator set of fraction h among the $p - 1$ nonzero values therefore has at most $2(r - 1)(p - 1)h$ point-pair preimages. Its probability is at most $D_h h$ with

$$D_h = \frac{2(p - 1)}{r - K}.$$

Field pairs. A coordinate-difference pair with nonzero x-difference has at most four point-pair preimages. Write $A = (x + u, y + v)$ and $U = (x, y)$ with $u \neq 0$. Subtracting the two curve equations gives

$$2vy + v^2 = 3ux^2 + 3u^2x + u^3.$$

If $v \neq 0$, this expresses y as a quadratic polynomial in x . Substituting it into $y^2 = x^3 + 7$ gives a polynomial of degree four in x , with nonzero leading coefficient $(3u/2v)^2$. There are at most four roots, and each root fixes y . If $v = 0$, the equation is a quadratic in x with leading coefficient $3u \neq 0$. It has at most two roots, each with at most two y-values. Either way there are at most four preimages. A bad set of fraction q among the $p(p - 1)$ field pairs therefore has probability at most $D_{\text{pair}} q$ with

$$D_{\text{pair}} = \frac{4p(p - 1)}{(r - 1)(r - K)}.$$

The division kernel receives the pair $(x_1 - x_2, y_1 - y_2)$ directly. For the multiplication, first apply the point-pair change $(A, U) \mapsto (U, -(A + U))$, which is invertible on the nonexceptional domain. Then apply the field-pair change $(u, \lambda) \mapsto (u, u\lambda)$, which is a bijection for $u \neq 0$. The same four-preimage argument then applies. Each kernel has a denominator-only recorder term h and a replay term q_{rep} on arbitrary field pairs (Sections 7.2 and 7.3). The two kernels contribute at most

$$2D_h h + 2D_{\text{pair}} q_{\text{rep}}$$

per ordinary addition. No independence between the two kernels is needed.

Outer interfaces. A fixed x-coordinate belongs to at most two points and a fixed y-coordinate to at most three. The two x-coordinate subtractions therefore have multiplicity at most $2 \cdot 2 = 4$ each, and the two y-coordinate subtractions at most $3 \cdot 3 = 9$ each. The point-pair changes between inputs and outputs are bijections on the good affine domain, and fixed column offsets are residue permutations on the good representation domain. This gives a total multiplicity of $4 + 4 + 9 + 9 = 26$ for the subtractions.

The three additions involve the mask. In the first and third, fixing both input words fixes the mask and an x-difference. That leaves at most $2(r - 1) \leq 4p$ point/mask preimages per input pair, and we use the bound $4p$. In the middle addition, a prescribed $3x_2$ leaves at most two

choices of U , and each of the $r - 1$ finite choices of A then fixes the mask. That gives at most $2(r - 1)$ preimages. Each point/mask triple has probability at most $1/(pD_0)$. Hence

$$p_{\text{add}} \leq \frac{8p + 2(r - 1)}{pD_0} N_{\text{add}}, \quad p_{\text{sub}} \leq \frac{26}{D_0} (N_{\text{add}} + 2pN_{\text{comp}}).$$

Numerically,

$$p_{\text{add}} \approx 1.7462299598 \times 10^{-9}, \quad p_{\text{sub}} \approx 1.0593796008 \times 10^{-8}, \quad p_{\text{sq}} \approx 9.8953032821 \times 10^{-9}.$$

These are the three outer terms in the bound on ϵ of Section 7.4.

F.5 Why the success theorem applies

Theorem 7.3 needs normalized output blocks for every scalar label, equal to the ideal block outside a bad event. The circuit contains measurements and resets, so this form is not automatic. This section shows that the circuit has it, specifying what the arithmetic physically does on inputs charged as failures.

Deferred Fourier transforms. For the analysis, defer the Fourier transforms of the scalar registers and keep their computational-basis labels x ; this is an equivalent account of the semiclassical implementation and adds no qubits.

Labels never change. The address folds are fixed invertible changes of label, and they are undone on every path. Work bits never control a change of a scalar label, also on bad paths.

Fair X outcomes. Between measurements, the oracle applies computational-basis permutations and diagonal signs, with classical feed-forward. For a fixed label, fixed randomization and fixed previous outcomes, the work state is therefore one basis vector times a phase. An X measurement of a work bit with value v gives each outcome μ with probability $1/2$ and the phase $(-1)^{\mu v}$, as in Section 4.5. This concerns the implemented basis permutations, not their intended arithmetic, so it holds also on paths charged as bad: an overflow can give a wrong bit string, but the outcome is still fair.

Fixed-length transcripts. Let N_X be the number of work-bit X measurements that a run performs; it depends on classical branches. Pad the skipped measurements with unused fair bits up to a fixed public length $J \geq N_X$, and ignore their values. Induction over the operations shows that every padded transcript t has probability 2^{-J} for every fixed label. Since the labels are orthogonal and unchanged, the same holds for their superposition.

Resets without conditioning. Every workspace release is a Z measurement and reset. For the analysis, purify it as

$$|v\rangle_W |0\rangle_E \mapsto |0\rangle_W |v\rangle_E,$$

with a fresh environment bit for every discarded bit, and do not condition on the Z record. Each block then stays normalized, even when a reset value is nonzero. On the good domain all environment bits are zero.

Normalized blocks. Let ω be the independent classical randomization. For each padded transcript t the normalized conditional state is

$$|\tilde{\psi}_{\omega,t}\rangle = \sum_x \alpha_x |x\rangle |g_{\omega,t,x}\rangle, \quad \|g_{\omega,t,x}\| = 1,$$

where the work block includes all reset records. Outside the charged bad event, exact phase correction, clean scratch and restored addresses give $g_{\omega,t,x} = f_x \otimes |0\rangle_E$. Here an output-only

permutation removes the common point scale, translation and encoding. It commutes with the success test on the scalar registers and is not implemented.

Why averaging is needed. In one fixed circuit, a bad subspace of weight q can carry amplitude of norm $\sqrt{q}$. A bound linear in the error probability therefore cannot come from a single circuit. Theorem 7.3 obtains it by averaging the normalized blocks over ω and t before applying Π (in Section 7.5, ω may include the padded transcript), using no independence between the bad events of different labels or different arithmetic calls.

First-failure bookkeeping. Every local bad event is defined along the ideal trajectory, with the same public randomness and transcript. If none occurs, induction shows that the actual trajectory equals the ideal one, including phase and all discarded zeros. The union of their unconditional probabilities bounds the global bad event.

The argument relies on the basis-permutation and diagonal-sign structure throughout the oracle. Arbitrary quantum noise, an uncorrected relative phase on good inputs, conditioning on a Z record, or transcript weights that depend on the input would each need a different theorem. Correct outputs on basis-state tests alone do not establish these conditions.

F.6 The ideal anchor

This section proves $P_0 \geq 64(r-1)/(r\pi^4)$ for the ideal operation of Section 2 with the shortened second register. The Fourier transforms here are ideal.

Write $Q = [d]P$, where P generates a group of odd prime order $r < 2^{256}$. Put $M_P = 2^{257}$ and $M_Q = 2^{209}$. After the output-only offset and the common nonzero scale are removed, the ideal state before the Fourier transforms is

$$\frac{1}{\sqrt{M_P M_Q}} \sum_{k=0}^{M_P-1} \sum_{\ell=0}^{M_Q-1} |k\rangle |\ell\rangle | [k + 2^{48}d\ell]P \rangle.$$

Group Fourier decomposition. Define $|\chi_a\rangle = r^{-1/2} \sum_{z=0}^{r-1} e^{-2\pi i a z/r} |[z]P\rangle$. Then $|[z]P\rangle = r^{-1/2} \sum_a e^{2\pi i a z/r} |\chi_a\rangle$, and the state becomes

$$\frac{1}{\sqrt{r}} \sum_{a=0}^{r-1} \left(\frac{1}{\sqrt{M_P}} \sum_k e^{2\pi i a k/r} |k\rangle \right) \left(\frac{1}{\sqrt{M_Q}} \sum_{\ell} e^{2\pi i a 2^{48}d\ell/r} |\ell\rangle \right) |\chi_a\rangle.$$

The $|\chi_a\rangle$ are orthonormal. The scalar outcomes are therefore a uniform mixture over a of two independent phase estimations. A common point scale only permutes the label a .

Phase estimation. For a register of length $F \in \{M_P, M_Q\}$ and phase θ , the inverse Fourier transform gives outcome j with probability

$$p_F(j \mid \theta) = \frac{1}{F^2} \left| \sum_{v=0}^{F-1} e^{2\pi i v(\theta - j/F)} \right|^2 = \frac{\sin^2(\pi F \theta)}{F^2 \sin^2(\pi(\theta - j/F))},$$

with removable singularities read by continuity. Let $t = \{F\theta\}$. If $t \in (0, 1)$, the two outcomes just below and just above $F\theta$ have combined probability at least

$$\frac{\sin^2(\pi t)}{\pi^2} \left(\frac{1}{t^2} + \frac{1}{(1-t)^2} \right) \geq \frac{8}{\pi^2}.$$

The first inequality uses $|\sin u| \leq |u|$. For the second, put $u = t - 1/2$ and $v = u^2 \leq 1/4$. Then $\sin(\pi t) = \cos(\pi u)$ and $1/t^2 + 1/(1-t)^2 = 8(1+4v)/(1-4v)^2$, so it suffices to show $\cos(\pi u) \geq (1-4v)/\sqrt{1+4v}$. The cosine product and the inequality $\prod_i (1-z_i) \geq 1 - \sum_i z_i$ for nonnegative z_i give

$$\cos(\pi u) = (1-4v) \prod_{i=2}^{\infty} \left(1 - \frac{4v}{(2i-1)^2}\right) \geq (1-4v)(1-v) \geq \frac{1-4v}{\sqrt{1+4v}}.$$

The middle step uses $\sum_{i \geq 2} (2i-1)^{-2} = \pi^2/8 - 1 < 1/4$, since $\pi^2 < 10$. The last step uses $(1-v)^2(1+4v) - 1 = v(2-7v+4v^2) \geq 0$ on $[0, 1/4]$. If $t = 0$ the exact outcome has probability one, so the bound holds as well.

Recovering a . On either good outcome the circular phase error is at most $1/F$. Since $M_P = 2^{257} > 2r$, a good first outcome j identifies a exactly: round rj/M_P and reduce modulo r . Exclude $a = 0$, which has probability $1/r$.

Recovering d . The second register estimates the residue $\nu = 2^{48}ad \bmod r$. For a good second outcome j' , ν has an integer lift in $[I_-, I_+]$ with

$$I_- = \left\lceil \frac{r(j'-1)}{M_Q} \right\rceil, \quad I_+ = \left\lfloor \frac{r(j'+1)}{M_Q} \right\rfloor.$$

Endpoints below 0 or above r are allowed; the interval is reduced modulo r . It holds $\max\{0, I_+ - I_- + 1\}$ integers, and

$$\left\lfloor \frac{2r}{M_Q} \right\rfloor + 1 \leq 2^{48}.$$

For each candidate ν , form $d' = \nu(2^{48}a)^{-1} \bmod r$ and accept it only if $[d']P = Q$. This finite procedure always returns the true d on the two good Fourier events. It may need up to 2^{48} classical tests. Alternatively, the equation $[\nu]P = [2^{48}a]Q$ over the interval can be solved by a generic baby-step giant-step search with $O(\sqrt{I_+ - I_- + 1})$ group operations and stored elements, that is $O(2^{24})$ here. This is classical cost and is not part of the Toffoli count.

Conclusion. Given $a \neq 0$, the two Fourier events have joint probability at least $(8/\pi^2)^2$. With the recovery rule above,

$$P_0 \geq \frac{r-1}{r} \left(\frac{8}{\pi^2}\right)^2 = \frac{64(r-1)}{r\pi^4}.$$

No heuristic success estimate for lattice-style recovery is needed.

References

- [1] T. Häner, F. Tripier, J. Young, M. Naehrig, A. Maksymov, S. Alam, D. Maslov, M. Parrott, Y. de Sereville, J. Sullivan, M. Webster, N. Delfosse, J. Gamble and M. Roetteler. *Computing 256-bit elliptic curve discrete logarithms in 26 days on a fault-tolerant trapped-ion quantum computer with 20,000 qubits*. arXiv:2609.05625v1, September 2026. <https://arxiv.org/abs/2609.05625v1>.
- [2] A. Schrottenloher. *Optimized Point Addition Circuits for Elliptic Curve Discrete Logarithms*. arXiv:2606.02235v1, June 2026. <https://arxiv.org/abs/2606.02235v1>.
- [3] C. Gidney. *Halving the cost of quantum addition*. Quantum 2, 74 (2018). arXiv:1709.06648.
- [4] R. B. Griffiths and C.-S. Niu. *Semiclassical Fourier Transform for Quantum Computation*. Phys. Rev. Lett. 76, 3228 (1996).