

A Qubit–Gate Frontier for Shor’s Algorithm on secp256k1: 24.9 to 28.2 Million Toffoli Gates on 1,380 to 2,022 Logical Qubits

Initio Intelligence

11 October 2026

Abstract

Almost all of the cost of Shor’s algorithm for elliptic-curve discrete logarithms lies in adding curve points whose coordinates are held in quantum registers. We give complete logical circuits for the algorithm on the curve secp256k1 at thirteen points between 1,380 and 2,022 logical qubits. They range from 28,248,654 Toffoli gates per run at 1,380 qubits to 24,867,156 at 2,014 qubits, and every one returns the discrete logarithm in a single run with a certified probability between 0.744 and 0.792. At 1,457 qubits, the qubit count of the logical estimate of Häner et al., one circuit uses 27,519,242 Toffoli gates with certified success at least 0.746924, and another uses 28,047,551 gates with certified success at least 0.789234. Against the approximately 39,255,168 Toffoli gates that we reconstruct from that estimate, these are reductions of 29.90% and 28.55%. The lowest cost per certified success, about 32.0 million Toffoli gates, is reached at 2,022 qubits. Most of the saving comes from field division and multiplication circuits driven by a Euclidean-type walk whose two registers are updated alternately on a fixed public schedule, so that the walk needs no data-dependent swaps or magnitude comparisons. The rest comes from recursive integer squaring with a cheap exact modular correction, segmented signed table lookups, and a last point addition that measures its output early and so omits its final multiplication. Extra qubits are spent where they save most: on replay cells without carry splits, on wider lookups, on table columns kept resident through a point addition so that fewer lookups are needed, and on measuring, rather than erasing, the history of the last division. Below 1,457 qubits a balanced carry split keeps the kernels within 1,380 qubits. We give the full construction: algorithms, fixed parameters, gate-count formulas, and proofs of every arithmetic error bound and of the success theorem. Correctness on a specified good domain is proved algebraically. Rare arithmetic failures are bounded analytically, except for the walk’s failure rate, which is bounded from 10^8 samples per walk schedule, with confidence $1 - 2^{-129}$ under an independent-uniform sampling model. With an alternative post-processing rule that classically tests the 32 nearest Fourier outcomes on each register (worst case about 2^{30} group operations), the certified success of the same circuits rises to between 0.915652 and 0.971502; all comparisons above use the standard rule. Clifford gates, the synthesis of Fourier rotations, the classical computation of the tables fed forward during the last point addition, and physical error correction are outside the Toffoli count.

1 Introduction

1.1 The result

Shor’s algorithm computes a discrete logarithm on an elliptic curve by preparing a superposition over pairs of scalars and computing, for each pair, a combination of two curve points; a quantum Fourier transform then reveals the logarithm. Nearly all of the work is the repeated *point addition*: adding a point read from a classical table to an accumulator point whose coordinates are quantum data. Each point addition in turn is dominated by two operations on 256-bit field elements, a modular division and a modular multiplication, which we call the *field kernels*.

We measure cost in *Toffoli gates* (controlled-controlled-NOT gates), the standard cost measure for fault-tolerant execution, and in *logical qubits*, the error-corrected qubits seen by the algorithm; their physical encoding is not part of this paper. We count every Toffoli in the circuit, including the worst-case charge for every gate that is conditioned on a measurement outcome.

We give complete logical circuits for Shor’s algorithm on secp256k1, the curve used by Bitcoin and Ethereum signatures, at thirteen points between **1,380 and 2,022 logical qubits** (Table 1 and Figure 1). They come in two families that differ only in the length of the walk that drives the field kernels. The *710-step* designs use the fewest gates: from **28,248,654 Toffoli gates** per run at 1,380 qubits down to **24,867,156** at 2,014 qubits, each with certified success probability of about 0.744 to 0.749. The *720-step* designs spend about 2% more gates on kernels that fail far less often, which raises the certified success to about 0.787 to 0.792 and gives the lowest cost per certified success: **32,001,879 Toffoli gates per certified success** at 2,022 qubits. Here *certified success* means the lower bound, proved in Section 8, on the probability that one run returns the discrete logarithm; one ingredient of that bound is statistical (Section 1.4). All success bounds hold for every fixed nonzero discrete logarithm; they average only over the circuit’s own randomization and measurement outcomes.

Table 1. Whole-algorithm logical resources for one run of Shor’s algorithm on secp256k1, for the published logical benchmark [1] and the thirteen designs of this paper. “Walk” is the length of the walk used in every field-kernel call. Peak qubits include the 18 address qubits and every lookup. Certified success and Toffolis per certified success use the standard recovery rule of [1]; the last column uses the wider rule $\mathcal{R}_{16}$ of Section 9. The rows in bold are the comparison at 1,457 qubits.

Peak qubits	Walk	Toffolis per run	Certified success	Toffolis per certified success	Success, wider rule
1,457	Häner et al. [1]	$\approx 39,255,168$	0.553	$\approx 70,985,838$	
1,380	710	28,248,654	0.744075	37,964,775	0.915652
1,388	720	28,723,852	0.786674	36,513,027	0.965264
1,457	710	27,519,242	0.746924	36,843,419	0.918972
1,457	720	28,047,551	0.789234	35,537,675	0.968243
1,502	710	27,052,231	0.749339	36,101,457	0.921786
1,510	720	27,464,489	0.792034	34,675,864	0.971502
1,713	710	26,065,837	0.746924	34,897,574	0.918972
1,713	720	26,592,549	0.789234	33,694,121	0.968243
1,721	720	26,536,460	0.789562	33,609,086	0.968624
1,758	710	25,667,373	0.749338	34,253,359	0.921786
1,766	720	26,123,794	0.792034	32,983,150	0.971501
2,014	710	24,867,156	0.749338	33,185,462	0.921786
2,022	720	25,346,594	0.792034	32,001,879	0.971501

The comparison at 1,457 qubits. The published logical benchmark of Häner et al. [1] uses 1,457 logical qubits. At that qubit count our 710-step design uses **27,519,242 Toffoli gates** with certified success at least **0.746924**, and our 720-step design uses **28,047,551** with at least **0.789234**. These are reductions of **29.90%** and **28.55%** in Toffolis per run, and of 48.1% and 49.9% in Toffolis per certified success. Every success bound of this work holds with statistical confidence $1 - 2^{-129}$ under the sampling model of Section 8.3, a statement about the sampled walk bound, not about physical error correction. The benchmark’s success figure, 0.553, is the lower bound stated in Section IV.B of [1], obtained there from the ideal-oracle bound 0.657 and a bound of 0.0335 on the failure probability of its 28 point additions (at confidence $1 - 2^{-128}$); its cost per certified success is $39,255,168/0.553$. These comparisons use the standard recovery rule of [1]; the wider rule of Section 9, which adds a classical search of at most about 2^{30} group operations per run, is reported alongside but is not part of them. The benchmark is the logical accounting of [1]: 28 point additions, approximately 1,196,000 arithmetic Toffolis per addition, three 2^{16} -entry lookups per addition, and an initial 2^{18} -entry lookup, so that $28(1,196,000 + 3 \cdot 2^{16}) + 2^{18} = 39,255,168$.

Its arithmetic figure is the largest count observed over sampled runs, whereas our counts charge every conditional operation on every input. The percentages therefore compare against that published logical estimate, not against an exact all-input count of the reference circuit. We do not compare with its separately compiled Toffoli-injection total, and we do not extrapolate its physical runtime to our circuits.

Away from 1,457 qubits. With 77 fewer qubits a balanced carry split in the field kernels gives a complete circuit at 1,380 qubits for 2.7% more gates. With 45 more qubits the kernels need no carry splits at all, and lookups planned for the larger budget become cheaper (27,052,231 Toffolis at 1,502 qubits). Beyond that, each further block of 256 qubits holds one table column for the whole of every point addition, which cuts the five table reads per addition to three (one block) or two (two blocks); the same space lets the last division’s history be measured instead of erased. This gives 25,667,373 Toffolis at 1,758 qubits and 24,867,156 at 2,014. The gain flattens there, because with two resident columns the lookups are only about 4% of the run.

1.2 Where the saving comes from

Table 2 shows how the Toffoli count falls as the ideas of this paper are added one at a time, all at 1,457 logical qubits. Table 3 then shows how the 710-step design changes when qubits are taken away or added. Each total is a sum of component counts given in Sections 3 to 7 and Appendix E.

Table 2. How the Toffoli count per run falls as each idea is added, at 1,457 logical qubits. Steps 1 to 4 use an ordinary last point addition. Steps 2 to 4 save a fixed amount in each of the 26 point additions (10,394, 1,774 and 351 Toffolis); step 5 changes only the last one. Step 6 trades gates for a higher success bound (0.789234 instead of 0.746924).

Step	What is added	Toffolis per run	Change
0	Published logical benchmark [1]	$\approx 39,255,168$	
1	Alternating-walk field kernels, segmented signed lookups, triangular squares	28,352,441	−10,902,727
2	Recursive integer squares (Section 4.3)	28,082,197	−270,244
3	Clean-target high correction in the square (Section 4.4)	28,036,073	−46,124
4	Mask folded into a lookup column (Section 2.3)	28,026,947	−9,126
5	Terminal window with measured square scratch (Section 5): 710-step design	27,519,242	−507,705
6	720-step walk in all 51 kernel calls: 720-step design	28,047,551	+528,309

Table 3. Trading qubits for gates, starting from the 710-step design at 1,457 qubits. The 1,380-qubit row moves down from that design. The remaining rows move up from it, each adding one change to the row above (the first 1,502-qubit row starts again from the 1,457-qubit design). “Change” is relative to the design it starts from. The 720-step walk gives the same sequence at 8 more qubits each (1,388, 1,510, 1,766 and 2,022). The designs at 1,713 and 1,721 qubits keep one resident column on kernels that still have carry splits.

Peak qubits	What changes (710-step walk throughout)	Toffolis per run	Change
1,457	The 710-step design of Table 2	27,519,242	
1,380	Balanced carry split in the replay cells (Section 6.2)	28,248,654	+729,412
1,502	No carry splits in the replay cells (Section 6.3)	27,186,008	−333,234
1,502	Lookups planned for the 1,502-qubit budget (Section 6.3)	27,052,231	−133,777
1,758	One table column kept resident (Section 6.4)	25,741,216	−1,311,015
1,758	Last division’s history measured, not erased (Section 6.5)	25,667,373	−73,843

Peak qubits	What changes (710-step walk throughout)	Toffolis per run	Change
2,014	A second resident column (Section 6.4)	24,867,156	−800,217

Against [1], part by part. The accounting of [1] gives Toffoli counts for only two parts of a run, so only these two can be compared directly with the 710-step design at 1,457 qubits:

- *Arithmetic.* [1] charges about 1,196,000 Toffolis per point addition for everything except the lookups, about 33,488,000 for its 28 additions. An ordinary point addition here costs 948,890: two kernel calls of 450,390 and 48,110 of outer arithmetic, of which 46,004 is the square subtraction. Over the run, the 25 ordinary and one terminal additions cost 24,200,132.
- *Lookups.* [1] charges $3 \cdot 2^{16} = 196,608$ Toffolis per point addition for three lookups on 16-bit windows, plus 2^{18} for the initial lookup: 5,767,168 per run. Here the five column reads of an ordinary addition, with their phase correction, cost 103,053 on a 17-bit window and 198,904 on an 18-bit window, and all lookups, phase tables and the terminal gadget cost 3,319,110 per run.

[1] does not state how its arithmetic figure divides between its in-place multiplications and its square subtraction; its per-component breakdown counts measurement layers, not Toffolis. A kernel-by-kernel or square-by-square comparison is therefore not possible from the published figures, and the arithmetic saving above combines the field kernels, the square, the terminal window and the use of 26 instead of 28 point additions.

The ideas, in plain terms, are as follows.

Field kernels. Following Schrottenloher [2] and Häner et al. [1], a division y/x is computed by running a binary-GCD-style *walk* on the denominator x , recording its sequence of decisions (the *history*), replaying the history as linear updates on the numerator (one *replay cell* per walk step), and finally running the walk backwards to erase the history. Our walk updates its two registers alternately on a fixed public schedule. The walk is longer than one that chooses which register to update, but each step needs only one sign bit and no data-dependent swap or magnitude comparison, and the replay needs no controlled swap of 256-bit registers. The register widths shrink on a public schedule as the walk proceeds, and recording, replay and erasure are interleaved so that the whole kernel fits in the qubit budget. Multiplication runs the same machinery in reverse. A 710-step walk gives the fewest gates; a 720-step walk costs 10,359 more Toffolis per kernel call at 1,457 qubits but fails far less often.

Lookups. Table reads use signed half-tables, so a w -bit window reads a table of 2^{w-1} points, and each 256-bit column is read in two segments chosen by a small optimization for the design’s qubit budget (Appendix B).

Squares. Each point addition subtracts a field square. We compute the three integer squares it needs by a Karatsuba-style recursion. The correction for the high part of the product, the word cH , is built exactly in a clean register from a 33-bit intermediate with seven short additions instead of twenty shifted additions.

Mask folding. Each point addition adds a fresh random mask to the x-coordinate register, so that the error analysis can treat that register as uniform. Folding the mask into a classical table column saves one modular addition.

Terminal window. The last point addition (the *terminal window*; all others are *ordinary*) produces an output that is never used again by the quantum computer. We measure its output

x-coordinate early; the slope register can then hold only one of two values, which a single selected bit distinguishes, so the slope can be measured and its phase repaired from a classically computed table. This removes the last multiplication and the uncomputation of the last square.

Carry splits. Each replay cell adds two 256-bit words, which needs 257 scratch wires. Where they do not fit in the qubit budget, the addition is split into two blocks joined by one measured carry, at a cost of 27 Toffolis per split cell. Splitting at bit 128 instead of 192 cuts the scratch wires of a split cell from 194 to 131 and lets the kernels run within 1,380 qubits; 45 extra qubits remove the splits altogether.

Resident columns. Each ordinary point addition reads five table columns, all affine functions of the table point’s two coordinates. Keeping the first column, $u + \tau$, live through the whole addition lets two other columns be formed from it by modular additions of 351 Toffolis each instead of reads of 20,509 to 39,628; keeping the second column, $v + \tau$, as well saves a third read.

Measured division history. In the last point addition, a copy of the denominator is kept, so the division’s history is no longer needed to restore it. The history is then measured instead of erased, and its phase, a known function of the denominator, joins the terminal phase table. This removes one walk pass.

1.3 Relation to earlier work

Roetteler, Naehrig, Svore and Lauter [10] gave precise circuit-level resource estimates (logical qubits, Toffoli gates and depth) for Shor’s algorithm on elliptic curves over prime fields of up to 521 bits, including 256-bit curves, with modular inversion built as a reversible version of Kaliski’s binary extended Euclidean algorithm for the Montgomery inverse. Häner, Jaques, Naehrig, Roetteler and Soeken [11] improved the arithmetic with windowing and a reformulated binary Euclidean inversion, and presented trade-offs between qubits, circuit depth and T-gate count. Litinski [12] studied the same problem in the active-volume architecture, whose cost function differs from a Toffoli count at a fixed number of qubits, and reduced the cost per key by reusing stored state across several keys and spreading the division over parallel instances. Häner et al. [1] is the most recent estimate and the like-for-like baseline of this paper. This work differs from them mainly in two constructions: the field kernels are driven by a walk on a fixed public schedule, with no magnitude comparisons and no data-dependent swaps; and the last point addition measures its output early and omits its final multiplication. It also presents its designs as a frontier that trades qubits for gates, each design with its own certified success bound.

The affine point-addition framework builds on [1,2], measured temporary-AND arithmetic on Gidney [3], and the reuse of scalar qubits on the semiclassical Fourier transform of Griffiths and Niu [4]. Our contribution is the complete construction and its resource and success analysis: the circuits and interleaved memory schedule of the kernels, the integration of recursive squaring and the clean-target correction, the terminal window, the balanced and budget-driven carry splits, resident table columns, the measured history of the last division, and the proofs and certificate that bound the probability of success of every design.

1.4 What is counted and what is assumed

The Toffoli count covers every Toffoli in the circuit. Single-qubit Fourier rotations, Clifford gates, measurements, resets and classical computation are required operations but are not Toffolis and are not counted; the Fourier rotations are treated as exact. The terminal window needs tables computed by a classical computer in the middle of the run; that computation and its latency are outside the count. The success bounds combine exact proofs with one statistical component, a sampled bound on the walk’s rare failures, whose confidence holds under the model that the

samples are independent and uniform. Every success bound outside Section 9 uses the standard recovery rule of [1], and no bound includes physical noise. Section 11 states these limits in full.

1.5 How the claims are established

The gate and qubit counts are exact: every component’s count is given by a formula in the text. Correctness on a specified *good domain* of inputs is proved algebraically. The probability of leaving the good domain is bounded analytically for every arithmetic step except the walk, whose failure rate is bounded statistically (Section 8). A general theorem then converts the total failure probability into a lower bound on the success probability (Theorem 8.3).

1.6 Organization

Section 2 fixes the curve, the scalar windows, the randomization and the point-addition pipeline. Section 3 builds the field kernels and Section 4 the modular additions, subtractions and the square. Section 5 describes the terminal window. Section 6 counts qubits and describes how the designs spend fewer or more of them, and Section 7 assembles the totals of all thirteen designs and the frontier between qubits and gates. Section 8 proves the success bounds under the standard recovery rule, and Section 9 gives the alternative, wider recovery rule and its success bounds. Section 10 summarizes the full-size executions of the circuits, and Section 11 states the scope of the claims. Appendices A to E give the explicit circuits, the lookups, the walk width schedules, the error-bound proofs and the complete gate and certificate arithmetic, and Appendix F lists every symbol used in the paper.

2 Setting, conventions and the point-addition pipeline

2.1 Field and scalar computation

Let

$$n = 256, \quad c = 2^{32} + 977, \quad p = 2^{256} - c, \quad \text{curve } y^2 = x^3 + 7 \text{ over } \mathbb{F}_p.$$

The group order is the prime

$$r = \text{FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141}_{16}.$$

For a generator P and a nonidentity public point $Q = [d]P$, the ideal controlled computation is

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

The 48 omitted low bits on the Q side are recovered classically (Section 8.7). This precision and recovery convention is used throughout the paper, except in Section 9, which widens the classical recovery.

Conventions. Registers are little-endian: $x[i]$ denotes bit i of x , and the slice $x_{[i,j]}$ holds bits i to $j - 1$. A classical constant is loaded into clean wires by X gates, or by CNOTs from a quantum control; these wires count toward the number of live qubits. An addition of width n_+ modulo 2^{n_+} costs $n_+ - 1$ Toffolis, an addition with carry-out costs n_+ , and a controlled swap costs one Toffoli per bit; the circuits for these, with measured carry erasure, are in Appendix A. Single-qubit Fourier rotations, Clifford gates, measurements, reset, classical memory, classical arithmetic and physical error correction are required, but they are not charged as Toffolis and lie outside the metric of this paper.

The scalar controls are reused through a windowed semiclassical inverse Fourier transform [4]. For a register of n_{ctl} bits, retire controls in order of group powers $2^{n_{\text{ctl}}-1-i}$, $i = 0, \dots, n_{\text{ctl}} - 1$. With $j_{<i} = \sum_{i'<i} j[i']2^{i'}$, apply $\text{diag}(1, e^{-2\pi i j_{<i}/2^{i+1}})$, a Hadamard gate, and a Z measurement producing bit $j[i]$ of the outcome j . Apply the rule separately for $n_{\text{ctl}} = 257$ and $n_{\text{ctl}} = 209$, with an extra group-power factor 2^{48} for the latter. Eighteen live controls, the width of the widest window below, suffice; they are the 18 *address qubits* included in every qubit count of this paper. The analysis assumes ideal Fourier rotations; a finite synthesis precision requires an additional error budget.

The scalar bits are consumed in *windows*: a window of width w uses w consecutive scalar bits as the address of a classical table of curve points, and adds the addressed point to an *accumulator*, a pair of 256-bit registers holding the running point. Each window after the first costs one point addition, and we also call that point addition an (ordinary or terminal) window. The initial lookup consumes the highest 17 P bits and the highest Q bit and writes the accumulator directly. Subsequent windows, in descending bit order within each scalar, are

$$\begin{aligned} P &: (18, 18, \underbrace{17, \dots, 17}_{12}), \\ Q &: (18, 18, 18, \underbrace{17, \dots, 17}_8, 18). \end{aligned}$$

Execute the P windows before the Q windows. Thus there are twenty 17-bit and six 18-bit additions, and the final addition has width 18. The Q windows cover precisely the bit interval $[48, 256)$. A window whose least bit is e has base $[b2^e]P$ or $[b2^e]Q$, where b is the random scale below. Changing the partition changes these weighted bases; it does not merely permute previously constructed tables.

2.2 Randomization and signed half-tables

Draw b uniformly from $\mathbb{Z}_r^*$ and use bases $[b]P, [b]Q$. Conditional on b , draw a translation $\mathcal{Z}$ uniformly among group points that make all 2^{18} entries of the initial lookup finite. At most $K = 2^{18}$ translations are excluded. Reject and redraw $\mathcal{Z}$, retaining b , until this condition holds. The initial lookup returns the encoded sum of the selected high-bit contributions, $\mathcal{Z}$, and the known offset needed to center all subsequent tables.

For a width- w window, $N = 2^w$ and weighted base Γ , define, for each window address $\mathbf{a}$,

$$\tilde{U}_{\mathbf{a}} = [\mathbf{a} - (N - 1)/2]\Gamma, \quad U_m = [m + 1/2]\Gamma, \quad 0 \leq m < N/2.$$

Fractions mean inverses modulo r . All U_m are finite because twice their scalar is a nonzero odd integer smaller than r . If $\mathbf{a}[w - 1]$ is the top address bit, put $s = 1 \oplus \mathbf{a}[w - 1]$ and complement each of the remaining $w - 1$ bits when $s = 1$. The resulting magnitude m satisfies $\tilde{U}_{\mathbf{a}} = (-1)^s U_m$. This fold and its inverse use only X and CNOT gates. Conjugating the accumulator by negation before and after adding U_m implements addition of $\tilde{U}_{\mathbf{a}}$.

The centering offset is independent of the partition: over windows tiling $[e_{\min}, e_{\max})$,

$$\sum_{\text{windows}} 2^e (2^w - 1)/2 = (2^{e_{\max}} - 2^{e_{\min}})/2.$$

The initial table adds the corresponding offset to compensate every centered digit. The ideal output differs from the desired group function only by a fixed translation and the invertible scale b ; neither changes the scalar Fourier distribution.

For any fixed scalar label and any ideal window trajectory, a specified unsigned table point U determines b , and a specified possibly sign-conjugated accumulator A then determines $\mathcal{Z}$. Consequently

$$\Pr[(A, U) = (A_*, U_*)] \leq D_0^{-1}, \quad D_0 = (r-1)(r-K).$$

A fresh independent mask $\rho \in \mathbb{F}_p$ in every window supplies a further factor $1/p$. This is an unconditional density bound. It is not conditioned on earlier arithmetic having succeeded.

2.3 Coordinate representation and addition pipeline

Set $\tau = (c-1)/2$ and represent a coordinate ζ by the integer $\mathcal{E}(\zeta) = \zeta + \tau$, on the range where this is a valid 256-bit word. For a nonzero coordinate, $\mathcal{E}(p-\zeta) = 2^{256} - 1 - \mathcal{E}(\zeta)$, so point negation complements the encoded y register. On representations outside the stated good range the circuit may behave arbitrarily; such inputs are included in the error bound.

For $U_m = (u_m, v_m)$ the five table columns, reduced modulo p , are

$$u_m + \tau, \quad v_m + \tau, \quad 3u_m + \rho, \quad v_m - \tau, \quad u_m - \tau.$$

Starting with $A' = (-1)^s A = (x_1, y_1)$ in encoded registers (X, Y) , the ordinary pipeline is

$$\begin{aligned} X &\leftarrow X - (u_m + \tau), & Y &\leftarrow Y - (v_m + \tau), \\ Y &\leftarrow Y/X, & X &\leftarrow X + (3u_m + \rho), \\ X &\leftarrow X - Y^2, & X &\leftarrow X + (-\rho \bmod p), \\ Y &\leftarrow YX, & Y &\leftarrow Y - (v_m - \tau), \\ X &\leftarrow X - (u_m - \tau), & X &\leftarrow \text{NOT}(X). \end{aligned}$$

Finally complement Y controlled by s , correct the accumulated lookup phase, undo the address fold, and retire the window’s controls. After division $Y = \lambda = (y_1 - v_m)/(x_1 - u_m)$; after mask removal $X = u_m - x_3$, where $(x_3, y_3) = A' + U_m$. Hence the final multiplication produces $y_3 + v_m$, and the last two subtractions and complement give $(\mathcal{E}(x_3), \mathcal{E}(y_3))$ before sign restoration.

The ordinary window has two additions, four direct subtractions, a square subtraction, a division and a multiplication. Folding ρ into the third table column saves a separate 351-Toffoli addition. Conditional on A', U_m , $3u_m + \rho$ is uniform, so the conservative mask-based arithmetic bound of the unfolded circuit remains valid; this is a distributional argument, not a claim that the bad sets of the folded and unfolded circuits are identical.

The affine exceptional set is contained in $\{\mathcal{O}, U_m, -U_m, -[2]U_m\}$ for A' . The good domain also requires $0 < y < p - \tau$ for the incoming point, its conjugate, the unsigned sum and the final result, and $\tau < x < p - \tau$ for the incoming point and unsigned sum. These exclusions have probability at most $[4 + 32(\tau + 1)]/(r - K)$ per window: a fixed x belongs to at most two curve points, a fixed y to at most three, and each state is a translate or negated translate of $\mathcal{Z}$.

3 The field kernels

Each point addition needs an in-place modular division $|x\rangle|y\rangle \mapsto |x\rangle|y/x\rangle$ and an in-place modular multiplication $|x\rangle|y\rangle \mapsto |x\rangle|xy\rangle$. We build both with the record-replay-erase structure of Schrottenloher [2] as used in [1]: a walk on the denominator x produces a sequence of decisions, the *history*; replaying the history as linear updates of a coefficient pair computes y/x or xy ; running the walk backward erases the history and restores x . Our walk differs from the binary GCD of [1, 2] in one respect: the register that is updated alternates on a fixed public pattern.

3.1 The walk

Let $0 < x < p$. Put $o = 1 - (x \bmod 2)$ and start from

$$(u_0, v_0) = (x + op, p),$$

two odd integers congruent to $(x, 0)$ modulo p . For $i = 0, \dots, T - 1$, the *target* t is u when i is even and v when i is odd; the other register is the *operand* w . With bit_1 the second-lowest bit of the two’s-complement representation, the step is

$$h_i = \text{bit}_1(t) \oplus \text{bit}_1(w), \quad n = t + (-1)^{h_i} w, \quad t \leftarrow n/2.$$

Since t and w are odd, $h_i = 0$ means $t \equiv w \pmod{4}$ and $h_i = 1$ means $t \equiv -w \pmod{4}$; either way $n \equiv 2 \pmod{4}$, so $n/2$ is again odd. Values may become negative, and $|n/2| \leq \max(|t|, |w|)$, so the larger magnitude never grows. The four states $(\pm 1, \pm 1)$ are absorbing: equal signs give $h_i = 0$ and $n = 2t$, opposite signs give $h_i = 1$ and $n = 2t$. A walk *converges* within T steps if $|u_T| = |v_T| = 1$; the signs $s_u = [u_T < 0]$ and $s_v = [v_T < 0]$ are recorded with the history. There is no comparison of magnitudes and no swap: the only data-dependent quantity in a step is the sign bit h_i .

Why the walk finishes. Because the operand is odd, a step preserves the greatest common divisor of the pair: $\gcd((t \pm w)/2, w) = \gcd(t \pm w, w) = \gcd(t, w)$. The starting pair has $\gcd(x + op, p) = \gcd(x, p) = 1$, since p is prime and $0 < x < p$, so every pair along the walk is coprime and odd. A step leaves its target unchanged only if $t = \pm w$, which for a coprime odd pair means $|t| = |w| = 1$; the absorbing states are therefore the only pairs at which the walk can come to rest. How fast it gets there is a heuristic matter. Each step removes exactly one factor of two from the updated register, while the addition of the operand adds back less than one bit on average, so the bit length of $|uw|$, which starts near $2n$, falls by a little over 0.8 bits per step on random inputs. A typical walk therefore finishes in about $2.4n$ steps, about 620 for $n = 256$, and only rare walks need about $2.8n$, near 720. The walk is not guaranteed to finish within a fixed T ; walks that do not are failures of the kind bounded in Section 8.3.

A *walk schedule* fixes the number of steps T and the register widths W_i of Section 3.2; both are fixed public constants, listed in Appendix C. We use two schedules, with $T = 710$ and $T = 720$ steps; every design uses one of them for all of its kernel calls, and we call it a 710-step or a 720-step design accordingly.

Lemma 3.1 (replay contract). For each step let $\mathcal{L}_i$ be the linear map of $\mathbb{F}_p^2$ that replaces the target coordinate t by $(t + (-1)^{h_i} w)/2$ and fixes the operand, and let $\mathcal{L} = \mathcal{L}_{T-1} \circ \dots \circ \mathcal{L}_0$. If the integer walk from (u_0, v_0) is exact and converges, with $(u_T, v_T) \in \{\pm 1\}^2$, then for every y

$$\mathcal{L}(y, 0) = \left(u_T \frac{y}{x}, v_T \frac{y}{x}\right), \quad \mathcal{L}^{-1}(u_T y, v_T y) = (xy, 0) \quad (\text{modulo } p).$$

Proof. The integer walk applies exactly the maps $\mathcal{L}_i$ to integer pairs, and 2 is invertible modulo p , so reducing modulo p gives $\mathcal{L}(x, 0) = (u_T, v_T)$. Both statements follow by linearity: $\mathcal{L}(y, 0) = (y/x) \mathcal{L}(x, 0)$ and $\mathcal{L}^{-1}(u_T y, v_T y) = y \mathcal{L}^{-1}(u_T, v_T) = (xy, 0)$. $\square$

The contract needs only an exact, convergent walk with the recorded signs; the sign rule matters only for convergence. Division therefore replays the history forward with *halving cells* $t \leftarrow (t + (-1)^{h_i} w)/2$, and multiplication replays it backward with *doubling cells* $t \leftarrow 2t - (-1)^{h_i} w$, the inverse map.

3.2 The recorder

The walk registers $\mathfrak{U}, \mathfrak{V}$ hold signed two’s-complement integers; at step i both have the public width W_i , so they represent $[-2^{W_i-1}, 2^{W_i-1})$.

Initialization. Compute $o = \text{NOT } x[0]$ into a clean wire (CX and X). Extend the register holding x by clean wires to $W_0 = 258$ bits; this is $\mathfrak{U}$. Load p into a clean 258-bit register $\mathfrak{V}$ with X gates. Add p to $\mathfrak{U}$ controlled by o (the constant is loaded by CNOTs from o); this 258-bit addition costs 257 Toffolis.

Step i . Let $(\mathfrak{T}_i, \mathfrak{D}_i)$ be $(\mathfrak{U}, \mathfrak{V})$ for even i and $(\mathfrak{V}, \mathfrak{U})$ for odd i .

1. Compute $h_i = \mathfrak{T}_i[1] \oplus \mathfrak{D}_i[1]$ into a clean wire with two CNOTs.
2. Sign-extend both registers by one guard wire (a CNOT from each top bit) and compute $\mathfrak{T}_i \leftarrow \mathfrak{T}_i + (-1)^{h_i} \mathfrak{D}_i$ at width $W_i + 1$: complement $\mathfrak{T}_i$ under control of h_i (CNOTs), add $\mathfrak{D}_i$, and undo the complement, using $\sim (\sim \mathfrak{t} + \mathfrak{w}) = \mathfrak{t} - \mathfrak{w}$. Both low bits are one, so the adder’s first carry is the known bit $1 \oplus h_i$ and is set with Clifford gates. This costs $W_i - 1$ Toffolis. Clear and release the operand’s guard wire.
3. Halve: the lowest wire of $\mathfrak{T}_i$ is now zero; release it by Z measurement and reset, and keep the remaining W_i wires, which hold $\mathfrak{n}/2$.
4. If $W_{i+1} < W_i$, the top $W_i - W_{i+1}$ wires of both registers are copies of the sign bit: clear each with a CNOT from bit $W_{i+1} - 1$ and release them.

A step costs $W_i - 1$ Toffolis, and one recording pass costs

$$C_{\text{rec}} = 257 + \sum_{i=0}^{T-1} (W_i - 1).$$

Terminal clearing. After step $T - 1$ both registers hold ± 1 at width W_T . Copy their sign bits into clean wires s_u, s_v ; CNOT each sign into bits $1, \dots, W_T - 1$ of its register and apply X to bit 0, which maps both $+1$ and -1 to zero; release both registers. The data kept after recording are $h_0, \dots, h_{T-1}, s_u, s_v$ and o : $T + 3$ wires.

The circuit contains no test. Whether each release is of a zero wire, and whether the walk converged, are conditions of the classical *walk predicate*, which defines the inputs on which the recorder is proved correct (Section 8.3):

```
walk_ok(x):                                # 0 < x < p; W_0..W_T public
  (u, v) = (x, p) if x is odd else (x + p, p)
  if max(|u|, |v|) >= 2^(W[0]-1): fail (entry overflow)
  for i = 0, ..., T-1:
    (t, w) = (u, v) if i is even else (v, u)
    h = bit1(t) XOR bit1(w); N = t + (-1)^h * w
    if N mod 4 != 2: fail (parity)
    if |N| >= 2^W[i]: fail (numerator overflow)
    t = N / 2 (written back into u or v)
    if max(|u|, |v|) >= 2^(W[i+1]-1): fail (width overflow)
  if |u| != 1 or |v| != 1: fail (walk not finished)
  accept
```

A parity failure cannot occur, and a numerator overflow is implied by the previous step’s width test, because $|\mathfrak{n}| \leq 2 \max(|\mathfrak{t}|, |\mathfrak{w}|)$. On real inputs only width overflows and unfinished walks occur; both are charged in full.

Width schedules. The widths W_i are fixed public constants of the circuit, chosen so that a width overflow is rare at every step and independently of the samples used to certify the walk (Section 8.3). Appendix C lists both schedules exactly. For $T = 710$, $W_0 = 258$, $W_{710} = 9$ and $\sum_{i < 710} W_i = 100,143$, so a recording pass costs $257 + 100,143 - 710 = 99,690$ Toffolis. For $T = 720$, $W_{720} = 8$ and $\sum_{i < 720} W_i = 101,750$, so a pass costs 101,287.

3.3 The eraser

The eraser is constructed directly as the inverse of the recorder, not by reversing a gate sequence that contains measurements. It rebuilds $(\pm 1, \pm 1)$ at width W_T from s_u, s_v (X on bit 0, CNOTs from the sign wire, then the sign wires are cleared by CNOTs from the rebuilt top bits and released). For $i = T - 1$ down to 0 it then widens both registers to W_i by sign extension, doubles the target by prepending a clean low wire, computes $\mathfrak{T}_i \leftarrow \mathfrak{T}_i - (-1)^{h_i} \mathfrak{D}_i$ at width $W_i + 1$ (same $W_i - 1$ Toffolis), clears and releases the top guard (a CNOT from the bit below it), and recomputes $\mathfrak{T}_i[1] \oplus \mathfrak{D}_i[1]$ into h_i , which clears it. Finally it subtracts $o \cdot p$ from $\mathfrak{U}$, clears o by a CNOT from the restored $x[0]$ and an X, releases the two zero high wires of $\mathfrak{U}$, and clears $\mathfrak{V} = p$ with X gates. Its cost equals C_{rec} . On every input accepted by the walk predicate it restores x and all workspace with no residual phase, because each step exactly inverts the corresponding recorder step on the same values.

3.4 The replay cell

Each replay step doubles or halves a 256-bit coefficient modulo p . An exact modular reduction after each doubling would need a full-width comparison and a conditional subtraction, about $2n$ Toffolis per cell on top of the integer addition. The cell below instead corrects only the low κ bits and checks only δ bits for the phase, $\kappa + \delta + 2$ Toffolis on top of the $n - 1$ of the integer addition; it is exact except on a small fraction of inputs, which Theorem 8.1 charges.

Replay coefficients are held in the *replay encoding* (distinct from the coordinate encoding $\mathcal{E}$ of Section 2.3)

$$E(\zeta) = (\zeta + \mu) \bmod p, \quad \mu = M/2 - c, \quad M = 2^n,$$

which keeps identically zero coefficients, common in GCD-style replay, away from the boundaries where the approximate arithmetic below can fail: the offset places $E(0) = \mu$ near the middle of the word, and it satisfies $2\mu + c = p$, which makes the doubling identity below exact.

The *doubling cell* computes $S \leftarrow E(2y_S + (-1)^\beta y_R)$ from $R = E(y_R)$, $S = E(y_S)$ and a sign control β ; in multiplication $\beta = 1 - h_i$, obtained by X gates on the history wire before and after the cell. Put $R_\beta = R$ and $\Theta_\beta = -c$ when $\beta = 0$, and $R_\beta = M - 1 - R$ and $\Theta_\beta = c - 1$ when $\beta = 1$; the conditional complement of R uses only CNOTs. Since $2\mu + c = M - c = p$ and $M - 1 \equiv c - 1$,

$$E(2y_S + (-1)^\beta y_R) \equiv 2S + R_\beta - \Theta_\beta \pmod{p}.$$

The cell computes the exact integer $2S + R_\beta = qM + \ell$ with $q \in \{0, 1, 2\}$ and $0 \leq \ell < M$, and adds the short correction $\Upsilon_\beta(q) = qc - \Theta_\beta$ to the low κ bits only, where the *correction width* κ is a public parameter:

$$Z = 2^\kappa \lfloor \ell / 2^\kappa \rfloor + ((\ell + \Upsilon_\beta(q)) \bmod 2^\kappa).$$

The six constants are $c, 2c, 3c$ for $\beta = 0$ and $1 - c, 1, c + 1$ for $\beta = 1$ ($q = 0, 1, 2$). With $q = q_0 + 2q_1$ and $q_0q_1 = 0$, every bit of the six constants is an affine function of $1, q_0, q_1, \beta, \beta q_0, \beta q_1$; two Toffolis prepare βq_0 and βq_1 , the constant is loaded into a clean κ -bit word by CNOTs, added with $\kappa - 1$ Toffolis and unloaded.

The quotient bits are then cleared. The low bit satisfies $q_0 = R_\beta[0] \oplus Z[0] \oplus (1 - \beta)$, also when a short carry is dropped, so it is cleared with Clifford gates. Call an input *value-correct* if the cell writes the exact encoded result, that is, the short correction loses no carry out of the low κ bits. On value-correct inputs the high bit is $q_1 = (1 - q_0)[Z + \Theta_\beta < R_\beta]$. It is measured in the X basis with outcome ϕ_1 , and the phase $(-1)^{\phi_1 q_1}$ is repaired with the truncated estimate $\hat{q}_1 = (1 - q_0)[\lfloor Z/2^{n-\delta} \rfloor < \lfloor R_\beta/2^{n-\delta} \rfloor]$, computed by a δ -bit comparator when $\phi_1 = 1$ (and charged on every run); δ is the public *comparison width*. The only residual phase is $(-1)^{\phi_1(q_1 \oplus \hat{q}_1)}$, which the certificate charges: a *phase-only error* is a value-correct input on which $\hat{q}_1 \neq q_1$, so the value is right but the sign may not be. The cell costs n (integer stage, including the Toffoli that forms q_1), 2 (the two ANDs), $\kappa - 1$ (short addition) and δ (comparison):

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

The *halving cell* used by division, $S \leftarrow E((y_S + (-1)^{h_i} y_R)/2)$, is the exact inverse of the doubling cell with $\beta = 1 - h_i$ and is constructed directly, with the same count. We use $\kappa = 60$, $\delta = 27$ (344 Toffolis) for the 710-step kernels and $\kappa = 62$, $\delta = 29$ (348) for the 720-step kernels. Appendix A lists both integer stages gate by gate.

The cell's target is fixed by the public schedule: at step i it acts on the first coefficient register if i is even and on the second if i is odd. No controlled swap of 256-bit registers is ever needed.

3.5 Split cells

A cell's integer stage needs $n - 1 = 255$ carry wires plus two quotient wires, 257 scratch wires in all. Where that does not fit in the qubit budget, the cell is built in *split* form at a public *split boundary* b_{sp} . The 256-bit integer addition is done as a low block (bits 1 to $b_{\text{sp}} - 1$) and a high block (bits b_{sp} to 255). After the low block's carries and sum bits are computed, the carry γ_{sp} into bit b_{sp} is copied into a fresh wire cy and the low carries are erased; the high block runs with carry-in cy and is erased; finally cy is X-measured with outcome ϕ_{sp} and the phase $(-1)^{\phi_{\text{sp}} \gamma_{\text{sp}}}$ is repaired using

$$\gamma_{\text{sp}} = [Z_{\text{lo}} < R_{\text{lo}}],$$

where Z_{lo} and R_{lo} are the low b_{sp} bits of the written sum and of the operand (exact because the block has no carry-in). The repair compares only the top $\delta' = 27$ bits of the low block (bits $b_{\text{sp}} - 27$ to $b_{\text{sp}} - 1$), conditioned on $\phi_{\text{sp}} = 1$, so it costs 27 Toffolis, and it is wrong only if those 27-bit windows tie while the full comparison differs. When the coefficient of the cell's target register is nonzero (Appendix D.1), so that the register is uniform, this has probability at most $2^{1-\delta'}$, charged per split cell, for either boundary used here; the hypothesis holds at every split cell of every design. The halving cell's split form uses the borrow predicate $[\sim R_{\text{lo}} < D_{\text{lo}}]$ in the same way, where D_{lo} is the low b_{sp} bits of the difference word that the halving cell writes (the word $I' \bmod 2M$ of Appendix A, before its final halving), the counterpart of Z_{lo} ; the predicate is exact because that subtraction has no initial borrow.

Which cells are split follows a fixed public rule. Each kernel call has a *kernel budget* n_{ker} : the design's peak qubit count less 256 for each extra 256-bit register held live through the call (Sections 6.4 and 6.5). *A cell is built in split form exactly when the number of live qubits at its start plus 257 exceeds n_{ker} .* With the interleaving of Section 3.6 (the live wires are listed in Section 6.1), the number of live qubits at the start of replay cell i , counted at a kernel budget of n_{ker} (that is, without any extra registers of Sections 6.4 and 6.5), is exactly

$$532 + i + 2W_{i+1} \quad (i < T - 64), \quad 533 + T \quad (\text{the last 64 cells}).$$

Its largest value is 1,245 for $T = 710$ (at cell 645) and 1,253 for $T = 720$ (at cell 655); adding 257 gives 1,502 and 1,510. At $n_{\text{ker}} = 1,457$, with $b_{\text{sp}} = 192$, the rule gives 242 split cells per kernel call for $T = 710$ and 273 for $T = 720$; at $n_{\text{ker}} = 1,465$ the 720-step kernel has 241. With $n_{\text{ker}} = 1,502$ no 710-step cell needs splitting, and the 720-step schedule needs 1,510.

The balanced split. The two carry blocks of a split cell are never live together. With $b_{\text{sp}} = 192$ its widest stage, the low block, needs 191 carry wires, the carry copy *cy* and the two quotient wires: 194 scratch wires (the 27-bit repair runs after the carries are released). Since $1,245 + 194 = 1,439$, the 710-step kernel cannot be split this way within budgets below about 1,440 qubits (1,447 for $T = 720$). Splitting at $b_{\text{sp}} = 128$ balances the two blocks (127 and 128 bits): the widest, the high block, needs 128 carry wires plus *cy* and the two quotient wires, 131 scratch wires in all, against 257 unsplit. As $1,245 + 131 = 1,376 \leq 1,380$ and $1,253 + 131 = 1,384 \leq 1,388$, every split cell then fits, and the same split rule fits the 710-step kernel within 1,380 qubits (528 split cells per call, the first at step 180) and the 720-step kernel within 1,388 (523 split cells, the first at step 196). The square stage, at 1,379 qubits, also fits within these budgets (Section 6.2). At 1,457 qubits the balanced split gives exactly the same cells and counts as $b_{\text{sp}} = 192$; the designs at 1,457 qubits and above use $b_{\text{sp}} = 192$, and those at 1,380 and 1,388 qubits use $b_{\text{sp}} = 128$. The repair still compares 27 bits, now bits 101 to 127, and its charge is unchanged (Appendix D.1).

3.6 Endpoints and assembly

Interfaces. Using $\mu \equiv -M/2 \pmod{p}$, the encoder flips the most significant bit and then subtracts c from the low κ bits, controlled by the new top bit ($\kappa - 1$ Toffolis); the decoder is the inverse. Each is an exact bit permutation that differs from the intended field map on exactly $c2^{n-\kappa-1}$ canonical inputs.

Conditional negation. Since $2\mu \equiv -c$, $E(-\zeta) \equiv -c - E(\zeta) \equiv \overline{E(\zeta)} - (2c - 1) \pmod{p}$, where the bar is the bitwise complement. A controlled negation complements the register by CNOTs from the control and subtracts the controlled constant $2c - 1$ in the low κ bits ($\kappa - 1$ Toffolis). It is charged $2c/2^\kappa + 2c/p$ per use (a lost low-word borrow or a non-canonical result).

Division ($y \leftarrow y/x$). Encode $R \leftarrow E(y)$ and set $S \leftarrow E(0) = \mu$ with X gates. Run recorder step i and then halving cell i , which uses the fresh h_i , for $i = 0, \dots, T - 65$; run the last 64 recorder steps and the terminal clearing, which releases the walk registers; then run the last 64 halving cells. By Lemma 3.1 the pair now holds $(E(\mathbf{u}_T y/x), E(\mathbf{v}_T y/x))$. Negate S controlled by $s_u \oplus s_v$ (computed in place by a CNOT and restored), so that $S = R$; clear S by CNOTs from R and release it; negate R controlled by s_u ; decode. Finally run the eraser, which restores x .

Multiplication ($y \leftarrow xy$). Run the recorder pass and its terminal clearing. Encode $R \leftarrow E(y)$, copy it into a clean S by CNOTs, negate R controlled by s_u and S controlled by s_v , so the pair is $(E(\mathbf{u}_T y), E(\mathbf{v}_T y))$. Run doubling cells $i = T - 1, \dots, T - 64$; rebuild the walk registers (eraser initialization); then for $i = T - 1, \dots, 0$ run doubling cell i (if $i < T - 64$) followed by eraser step i , which consumes h_i . By Lemma 3.1 the pair is $(E(xy), E(0))$; clear $S = \mu$ with X gates, release it, decode R , and finish the eraser, which restores x .

Interleaving the recorder (or eraser) with the replay keeps only part of the history and narrow walk registers live while the cells run, which is what makes budgets of 1,457 qubits and below reachable (Section 6). The 64-step offset at one end lets the last cells run after the walk registers have been released.

3.7 Kernel costs

A kernel has two walk passes, T cells, the split repairs, two interfaces and two conditional negations:

$$C_{\text{kernel}} = 2C_{\text{rec}} + T C_{\text{cell}} + 27 n_{\text{split}} + 4(\kappa - 1).$$

Table 4. Toffoli cost of one field-kernel call (a division or a multiplication; both cost the same) without split cells, by component, for the two walk schedules. Each split cell adds 27.

Kernel	T	κ, δ	Walk passes	Cells	Interfaces and nega- tions	Without splits
710-step	710	60, 27	$2 \times 99,690$	710×344	4×59	443,856
720-step	720	62, 29	$2 \times 101,287$	720×348	4×61	453,378

Table 5. Split cells and Toffoli cost of one kernel call at each kernel budget used by the designs. The 720-step kernel costs 9,522 more per call than the 710-step kernel without splits, and 10,359 more at 1,457 qubits; the difference does not interact with anything else in the window.

Kernel	Kernel budget			Toffolis per call
	n_{ker}	Split boundary	Split cells n_{split}	
710-step	1,380	128	528	458,112
710-step	1,457	192	242	450,390
710-step	$\geq 1,502$		0	443,856
720-step	1,388	128	523	467,499
720-step	1,457	192	273	460,749
720-step	1,465	192	241	459,885
720-step	$\geq 1,510$		0	453,378

4 Coordinate arithmetic and the square

4.1 Modular addition and subtraction

The modular additions and subtractions of a point addition, together with the square subtraction of Section 4.2, form its *outer arithmetic*: everything outside the field kernels and the lookups. The additions and subtractions use the pseudo-Mersenne structure of p with a correction width $\kappa_o = 65$ and a comparison width $\delta_o = 32$, independent of the replay widths.

Addition $y \leftarrow y + x \pmod{p}$. Add x into y with a carry-out γ (256 Toffolis); add $\gamma \cdot c$ into the low 65 bits of y (the constant is loaded by CNOTs from γ ; 64 Toffolis); X-measure γ with outcome ϕ_γ and repair the phase $(-1)^{\phi_\gamma \gamma}$ with a phase-only comparison of the top 32 bits of the result and of x , since on the good domain $\gamma = [y_{\text{new}} < x]$ (31 Toffolis). Total $256 + 64 + 31 = 351$.

Direct subtraction $y \leftarrow y - x \pmod{p}$. Complement y , add x with a carry-out, and complement y again; the carry-out γ is the borrow $\gamma = [x > y]$ and y now holds $y - x + \gamma 2^n$. Subtract $\gamma \cdot c$ from the low 65 bits, so that a borrow yields $y - x + p$ (64 Toffolis). X-measure γ and repair its phase with a phase-only comparison of the top 32 bits of $\bar{x}$ and of the result, using $\gamma = [\bar{x} < y_{\text{new}}]$ on the good domain (31 Toffolis). Total 351, with no complement conjugation of the modulus.

A classical constant, such as the mask $-\rho$, is prepared as a 256-bit quantum operand with X gates and cleared afterwards. Each point addition contains two additions (the third column and

the mask removal) and four subtractions (columns 0, 1, 3 and 4): 2,106 Toffolis. Their bad sets are counted in Appendix D.

4.2 Modular square subtraction

The operation $|x\rangle|y\rangle \mapsto |x\rangle|(y - x^2) \bmod p\rangle$ has the slope register as source and the x-coordinate register as target. Write $x = x_{\text{lo}} + 2^{128}x_{\text{hi}}$ and use $2^{256} \equiv c \pmod{p}$:

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

Three integer squares are needed: the two 128-bit halves and their 129-bit sum. The sum is formed in place by extending the x_{hi} register with one clean wire and adding x_{lo} (128 Toffolis), and removed the same way afterwards. Each square is computed into clean scratch, its contributions are added to y as shifted *slices*, and it is then uncomputed.

Slices. $\text{Slice}_{\pm}(z; i_{\text{src}}, n_{\text{sl}}, i_{\text{dst}})$ adds $\pm 2^{i_{\text{dst}}} z_{[i_{\text{src}}, i_{\text{src}} + n_{\text{sl}}]}$ to y . If $i_{\text{dst}} + n_{\text{sl}} + n_{\text{pad}} < n$ with $n_{\text{pad}} = 32$, it is a *local* slice: the source is zero-padded by n_{pad} bits and added (or subtracted) into $y_{[i_{\text{dst}}, i_{\text{dst}} + n_{\text{sl}} + n_{\text{pad}}]}$ at a cost of $n_{\text{sl}} + n_{\text{pad}} - 1$; it fails only if a carry escapes the padding. If $i_{\text{dst}} + n_{\text{sl}} = n$ it is a *wrapping* slice: an integer addition or direct subtraction into $y_{[i_{\text{dst}}, n]}$ with carry-out, the low-65-bit correction and the 32-bit phase repair, at a cost of $(n - i_{\text{dst}}) + 95$. No other case occurs.

Adding $\pm 2^{128}z$ for a square word z of length $n_z \in \{256, 258\}$: add the low 128 bits of z at offset 128 (wrapping, 223 Toffolis), then, because $2^{128} \cdot 2^{128}z_{\text{hi}} \equiv cz_{\text{hi}}$, add the high $n_z - 128$ bits at each offset i of the signed-digit form

$$c = 2^{32} + 2^{10} - 2^6 + 2^4 + 1,$$

with sign $\pm\sigma_i$, where $\sigma_i \in \{\pm 1\}$ is the digit at offset i . These five slices are local and cost $5 \cdot 159$ for $n_z = 256$ and $5 \cdot 161$ for $n_z = 258$. Total 1,018 or 1,028.

Adding $\pm cz$ for a 256-bit z uses the exact identity

$$cz = \sum_i \sigma_i 2^i (z \bmod 2^{256-i}) + 2^{256}H, \quad H = \sum_{i>0} \sigma_i \lfloor z/2^{256-i} \rfloor = z_{\gg 224} + z_{\gg 246} - z_{\gg 250} + z_{\gg 252},$$

with $0 \leq H \leq c - 3$. The five wrapping slices cost $319 + 341 + 345 + 347 + 351 = 1,703$. The overflow $2^{256}H \equiv cH$ is added as one local slice of a 65-bit word holding $cH < 2^{65}$ (96 Toffolis), which is prepared exactly in clean wires before and cleared after (Section 4.4).

4.3 The recursive integer square

The integer squarer maps an n_{sq} -bit x and a clean $2n_{\text{sq}}$ -bit z to $z = x^2$. For $n_{\text{sq}} \leq 40$ it uses the triangular method: with $x_{>i} = \sum_{i'>i} 2^{i'-i-1}x[i']$, $x^2 = -x + \sum_i (2x[i] - 1)2^{2i+1}(x_{>i} + x[i])$, implemented as one subtraction of x from the low $n_{\text{sq}} + 1$ bits followed, for each i , by a subtraction of a conditionally complemented window of $x_{>i}$ (Appendix A); it costs $\frac{1}{2}n_{\text{sq}}(n_{\text{sq}} + 3) - 1$ Toffolis.

For $n_{\text{sq}} > 40$ it recurses, in the manner of Karatsuba [5]. Put $n_0 = \lfloor n_{\text{sq}}/2 \rfloor$, $n_1 = n_{\text{sq}} - n_0$ and $x = x' + 2^{n_0}x''$.

1. Square x'' into $z_{[2n_0, 2n_{\text{sq}}]}$ and x' into $z_{[0, 2n_0]}$; the two products do not overlap, so $z = x'^2 + 2^{2n_0}x''^2$.
2. Form $x' + x''$ in place on x'' plus one clean wire (n_1 Toffolis).
3. Square $x' + x''$ into a clean register C of $2(n_1 + 1)$ bits.
4. Subtract x'^2 and x''^2 (read from z) from C , leaving $C = 2x'x'' \geq 0$ ($2n_1 + 1$ Toffolis each).

5. Add C into $z_{[n_0, 2n_{sq}]}$ ($2n_{sq} - n_0 - 1$ Toffolis); now $z = x^2$, which fits, so nothing wraps.
6. Restore x'' by subtracting x' from $x' + x''$ (n_1 Toffolis) and release the extra wire.

The cross-term registers C of every level are kept live until the inverse, which reverses these steps with its own measured cleanups and clears every retained register. The cost satisfies

$$C_{sq}(n_{sq}) = C_{sq}(n_1) + C_{sq}(n_0) + C_{sq}(n_1 + 1) + n_1 + 2(2n_1 + 1) + (2n_{sq} - n_0 - 1) + n_1,$$

with the triangular cost below the threshold. This gives $C_{sq}(64) = 2,000$, $C_{sq}(65) = 2,077$, $C_{sq}(128) = 6,654$ and $C_{sq}(129) = 6,774$, against 8,383 and 8,513 for triangular squares of 128 and 129 bits. Every step is an exact integer operation on registers wide enough to hold it, so the squarer is exact on every input and the slices downstream see bit-identical values. The retained cross terms raise the square stage’s peak to 1,379 qubits, still below the kernels’ 1,457.

4.4 The clean-target high correction

The 65-bit word cH is prepared exactly, in clean wires, from the source z in two steps. Write $H' = z_{\gg 246} - z_{\gg 250} + z_{\gg 252}$, so that $H = z_{\gg 224} + H'$. All three terms of H' come from the top ten bits of z : with $z_{top} = z_{\gg 246} < 2^{10}$ we have $H' = z_{top} - \lfloor z_{top}/2^4 \rfloor + \lfloor z_{top}/2^6 \rfloor$, and $0 \leq H' \leq 975$.

1. *Build H .* Copy z_{top} into a clean 10-bit register with CNOTs, subtract its top six bits and add its top four bits (two 10-bit additions, 9 Toffolis each), leaving H' . Copy the 32 bits $z_{\gg 224}$ into a clean 33-bit register and add H' into it (32 Toffolis), leaving $H < 2^{32} + 975$. Keep H' live.
2. *Build cH .* Use $c = 2^{32} + 977$ and $977 = 2^{10} - 2^6 + 2^4 + 1$. Copy H into the low bits of a clean 65-bit word; then, modulo 2^{42} , add $2^4 H$, subtract $2^6 H$ and add $2^{10} H$, truncating the last source to its low 32 bits (37, 35 and 31 Toffolis). Because $977H < 2^{42}$ the low 42 bits now hold $977H$ exactly. Finally add H into bits 32 to 64 (32 Toffolis), which gives $cH = 2^{32}H + 977H < 2^{65}$.

The forward preparation costs $18 + 32 + 37 + 35 + 31 + 32 = 185$ Toffolis, and its exact inverse, run after the slice has been added, clears the word, H and H' with another 185. The round trip costs 370 Toffolis, against 2,144 for the same round trip when cH is prepared directly by twenty shifted additions, one for each pair of signed digits. The prepared word is the same integer as with the direct preparation, and the slice that consumes it sees bit-identical inputs, so no error term changes. The ten extra wires of H' do not raise the peak of the square stage.

4.5 Cost of the square stage and of a point addition

Table 6. Toffoli cost of the square subtraction $y \leftarrow y - x^2$ by stage, with recursive integer squares and the clean-target high correction. The low stage is $2C_{sq}(128) + 1,018 + 351$; the high stage $2C_{sq}(128) + 1,018 + 1,703 + 96 + 370$; the sum stage $2 \cdot 128 + 2C_{sq}(129) + 1,028$. With triangular squares and the twenty-addition preparation of cH the same stages would cost 58,172.

Stage	Toffolis
Low half: square, $+2^{128}z$, $-z$, unsquare	14,677
High half: square, $+2^{128}z$, $-cz$, unsquare	16,495
Sum: form, square, $-2^{128}z$, unsquare, restore	14,832
Square subtraction	46,004

An ordinary point addition at 1,457 qubits therefore costs

$$2 \times 450,390 + 2 \times 351 + 4 \times 351 + 46,004 = 948,890$$

arithmetic Toffolis with the 710-step kernels, plus 103,053 or 198,904 for its five lookups: 1,051,943 for a 17-bit window and 1,147,794 for an 18-bit window. With the 720-step kernels the arithmetic is 969,608 and the windows cost 1,072,661 and 1,168,512. Section 7.2 gives the same breakdown at every other qubit budget.

5 The terminal window

The last point addition of a run is the only one whose output is never used again by the quantum computer; we call it the *terminal window*. After it, the accumulator holds the final point $[k_P]P + [2^{48}k_Q]Q$ (randomized), which only needs to be discarded or measured. We exploit this to skip its multiplication kernel.

5.1 The idea

Use the folded address of Section 2.2: magnitude m , sign s , unsigned table point $U_m = (u, v)$ and conjugated accumulator $A' = (-1)^s A$. The unsigned sum $A'_{\text{new}} = A' + U_m = (x_3, y_3)$ has

$$x_3 = \lambda^2 - x(A') - u, \quad y_3 = \lambda(u - x_3) - v,$$

and the true result is $A_{\text{new}} = A + (-1)^s U_m = (-1)^s A'_{\text{new}}$, so $x(A_{\text{new}}) = x_3$ and $y(A_{\text{new}}) = (-1)^s y_3$.

After the division, the mask additions and the square subtraction, the x-coordinate register holds $u - x_3$. The ordinary pipeline would now multiply to obtain y_3 . Instead, we perform the final x-column subtraction and the NOT, so that the register holds $\mathcal{E}(x_3)$, and *measure it*. Once x_3 is known classically, y_3 is one of the two square roots $\pm y_\star$ of $x_3^3 + 7$, so the slope can only be one of two values for each magnitude:

$$\lambda_0(m) = \frac{v - y_\star}{u - x_3}, \quad \lambda_1(m) = \frac{v + y_\star}{u - x_3} \pmod{p},$$

for $y_3 = -y_\star$ and $y_3 = +y_\star$ respectively, where $y_\star$ is the smaller root. Since $y_\star \neq 0$ (the group has odd order, so no point has $y = 0$) and $u \neq x_3$ on the good domain, $\lambda_0(m) \neq \lambda_1(m)$. Which one is present is a single bit η , and it can be read from a single bit of λ chosen per magnitude. The 256-bit slope register can then be measured in the X basis and its phase repaired by a classically compiled table on (m, η) .

5.2 The construction

The terminal window executes the ordinary pipeline of Section 2.3 up to and including the mask removal, and then:

1. *Final x-column.* Read column 4, subtract it from the x register, and apply NOT. The register holds $\mathcal{E}(x_3) = x_3 + \tau$.
2. *Measure.* Measure the 256 x wires in the Z basis, obtaining $\mathcal{E}(x_3)$ and hence $x_3 = \mathcal{E}(x_3) - \tau$; reset them with X gates according to the outcome and release them.
3. *Classical compilation.* Compute $y_\star$, the smaller square root of $x_3^3 + 7$ (as $p \equiv 3 \pmod{4}$), $y_\star = \pm(x_3^3 + 7)^{(p+1)/4}$, and, with one batch inversion [6] of the 2^{w-1} denominators $u_m - x_3$, the pairs $(\lambda_0(m), \lambda_1(m))$. For each m let $\iota(m)$ be the least bit position in which $\lambda_0(m)$ and $\lambda_1(m)$ differ and $\eta_0(m)$ the bit of $\lambda_0(m)$ there. If x_3 is not a valid encoding, $x_3^3 + 7$ is not a square, or $u_m = x_3$, use the fallback pair $(\lambda_0, \lambda_1) = (0, 1)$ for that entry (or for all entries).
4. *Selector.* Read the 9-bit word $\iota(m) | \eta_0(m) \cdot 2^8$ by a banked lookup on the magnitude (Appendix B): for the 18-bit last window, 128 banks of 1,024 rows ($1,022 + 127 \cdot 9 = 2,165$ Toffolis), or 96 banks (2,292 Toffolis) in the designs at 1,380 and 1,388 qubits.

5. *Extract η .* A binary tree of controlled swaps (the *slope multiplexer*), controlled by the 8 index bits of $\iota(m)$, moves bit $\iota(m)$ of the slope register to position 0 ($128+64+\dots+1 = 255$ controlled swaps); CNOT it into a clean wire η ; undo the tree (255). Then CNOT the selector’s bit 8 into η . Now $\eta = \lambda[\iota(m)] \oplus \eta_0(m) = [\lambda = \lambda_1(m)]$, that is, $\eta = 1$ exactly when $y_3 = +y_*$. Release the selector word by X measurement, recording its masks.
6. *Measure the slope.* Measure all 256 slope wires in the X basis, with outcome mask ϕ_λ , and release them.
7. *One phase table.* Apply the Boolean phase circuit of Appendix B on the w bits (m, η) , with the 2^w -entry table

$$F(m, \eta) = F_{\text{prefix}}(m) \oplus F_{\text{sel}}(m) \oplus \langle \phi_\lambda, \lambda_\eta(m) \rangle \oplus G_{\text{sq}}(\lambda_\eta(m)),$$

where F_{prefix} is the phase function F_Ω of the four column reads (Lemma B.1), F_{sel} that of the selector, and G_{sq} the phase of the measured square scratch (Section 5.4). Designs that measure the division history add the term F_{hist} of Section 6.5. For $w = 18$ this costs $C_{\text{phase}}(18, 9) = 1,020$ Toffolis.

8. *Output sign.* CNOT the address sign s into η , so that $\eta_{\text{out}} = 1$ exactly when $y(A_{\text{new}}) = +y_*$, and undo the address fold.

At the end only the 18 address wires and η_{out} are live. The output point is recorded as the classical x_3 and the bit η_{out} ; it is never used by the classical recovery.

5.3 Exceptional inputs

If $x_3 = u$ then $A'_{\text{new}} = \pm U_m$, so $A' = \mathcal{O}$ or $A' = -[2]U_m$; both belong to the affine exceptional set $\{\mathcal{O}, U, -U, -[2]U\}$ already charged for every point addition (Section 2.3). The case $y_3 = 0$ cannot occur. An infinite result $A'_{\text{new}} = \mathcal{O}$ is also in the affine exceptional set. A measured value that does not decode, or whose $x_3^3 + 7$ is not a square, cannot come from a good branch. The fallback of step 3 makes every measurement branch a complete circuit with the same gates and the same number of measurements; on bad branches the address is still restored and all workspace is still measured and released, and only the bad branch’s own output and phase are wrong. As a redundant safeguard, the certificate also charges $K/(r - K)$ for the event that the final point coincides with a table point of the last window, although the containment argument above already covers it.

5.4 Square scratch measured instead of uncomputed

In the terminal window the three integer squares of the square stage, and the cross-term registers retained by the recursive squarer, are deterministic functions of the slope: x_{lo}^2 , x_{hi}^2 , $(x_{\text{lo}} + x_{\text{hi}})^2$ and their cross terms, with $x = \lambda$. Instead of uncomputing them, each square buffer and its retained cross terms (1,764 bits in all) are X-measured immediately after their last use, and so are the 65-bit word cH , the 33-bit H and the 10-bit H' of the clean-target correction. Each measurement leaves the phase $(-1)^{\langle \text{mask}, \text{scr}(\lambda) \rangle}$ for a known function scr of the slope. Such a phase is diagonal in the computational basis and is carried unchanged through the later basis permutations; once x_3 is known it is a function of (m, η) , because $\lambda = \lambda_\eta(m)$ on the good domain, so it joins the final table as G_{sq} . This removes the three inverse squares ($2 \cdot 6,654 + 6,774 = 20,082$ Toffolis) and the inverse of the correction (185), and adds no error event.

5.5 Why measuring early is harmless

Lemma 5.1. Fix a good-domain input, the measured x_3 , the selector and lookup outcomes and the slope mask ϕ_λ . Then the terminal window acts on the scalar labels and

the accumulator as $2^{-J/2}$ times the ideal point addition followed by the output-only map $A_{\text{new}} \mapsto (x(A_{\text{new}}), \eta_{\text{out}}(A_{\text{new}}))$, where J is the number of X measurements, a public constant.

Proof. On a good basis state every intermediate register holds a definite value, and the only amplitude factors are the measurement factors $(-1)^{\langle \phi, \vartheta \rangle}$ of the X measurements (outcomes ϕ , measured values ϑ). Given x_3 , each measured value is a fixed function of (m, η) : the lookup words depend on m , the slope equals $\lambda_\eta(m)$, and the square scratch is a function of the slope. Their combined sign is therefore $(-1)^{F(m, \eta)}$, which step 7 removes exactly. All other operations are basis permutations that restore the address and leave (x_3, η_{out}) as the only output. The map is linear, so it holds on superpositions, including superpositions entangled with the scalar registers. $\square$

Averaging over the equally likely outcomes ϕ_λ therefore reproduces the channel of the ordinary last point addition followed by a measurement of the output point (with y replaced by its sign bit). By the deferred-measurement principle, measuring x_3 early and feeding it forward equals an x_3 -controlled unitary followed by a measurement at the end; the output measurement commutes with the scalar Fourier measurements and the classical recovery uses only the scalar outcomes. For the success theorem (Section 8.6) we do not condition on x_3 : the Z measurement and reset are purified into an environment record, the feed-forward is a control from that record, each X outcome is fair on every branch, and every branch performs the same public number of measurements. These are exactly the conditions under which the theorem applies (Appendix D.4).

5.6 Which window is last

The saving is larger when the last window is 18 bits wide, because the omitted y -column read costs 39,628 Toffolis instead of 20,509; every design therefore ends with an 18-bit window. The ordinary Q partition (18, 18, 18, 18, 17^8) is rearranged to (18, 18, 18, 17^8 , 18). Both tile the same bit interval [48, 256) with the same multiset of widths and retire bits in descending order. The centered digit offset of a partition is $(2^{256} - 2^{48})/2$ by telescoping, independently of the partition, so the initial offset is unchanged; only the public tables of the moved windows change, and they are compiled per run anyway.

5.7 Cost of the terminal window

Table 7. Toffoli cost of the 18-bit terminal point addition at 1,457 qubits by component. The *terminal gadget* (selector lookup, slope multiplexer and phase table) costs 3,695 Toffolis and peaks at 1,436 qubits. Section 7.2 gives the terminal window of every design.

Component	Toffolis
Four column reads (columns 0, 1, 2, 4)	158,512
Division kernel, 710 steps	450,390
Two additions and three subtractions	1,755
Square subtraction with measured scratch	25,737
Selector lookup	2,165
Slope multiplexer (forward and back)	510
Phase table on (m, η)	1,020
Terminal point addition	640,089
Same, 720-step division	650,448

Against an ordinary 18-bit window (1,147,794), the terminal window saves the multiplication (450,390), the column-3 read and subtraction (39,628 + 351), the ordinary window’s magnitude-

only phase correction (764) and the square inverses ($20,082 + 185$), and pays 3,695 for the gadget: 507,705 Toffolis.

5.8 The classical feed-forward

Between steps 2 and 4 a classical computer must compute one modular square root, one batch inversion over $2^{w-1} = 131,072$ denominators (about three field multiplications per entry) and the selector words; between steps 6 and 7 it must compile the $2^w = 262,144$ -bit phase table, each entry needing a 256-bit inner product and the integer squares of a candidate slope (and, in designs that measure the division history, one run of the walk on a candidate denominator). In all, this is one square root, one batch inversion of 2^{17} elements, 2^{17} selector words and 2^{18} inner products with their candidate squares. The quantum state must be held while this happens, once per run. This work is outside the Toffoli count (Section 11).

6 Qubit accounting and trading qubits for gates

6.1 The 1,457-qubit allocation

All stages share one allocation. Between point additions exactly 530 qubits are live: the two 256-bit coordinates and the 18 address bits. Within a point addition the stages are sequential, so the peak is the largest stage peak (Table 8).

Table 8. Peak number of live logical qubits in each stage, including all 18 address bits. The overall peak of 1,457 is reached only in the field kernels.

Stage	Peak logical qubits
Initial lookup, first / second segment	1,455 / 1,442
17-bit lookup, first / second segment	1,455 / 1,443
18-bit lookup, first / second segment	1,456 / 1,444
Field kernels (interleaved record-replay and replay-erase passes)	1,457
Square subtraction with recursive squares and retained cross terms	1,379
Modular additions and subtractions	1,042
Terminal gadget (18-bit last window)	1,436
Between point additions	530
After the terminal window (address bits and η_{out})	19

Lookups. The peak of a segment is $n_{\text{live}} + n_{\text{ad}} + n_{\text{sign}} + n_{\text{ret}} + Bn_{\text{seg}} + 2 \cdot \mathbf{1}_{B=3 \cdot 2^i} + \max(0, \lceil \log_2 L \rceil - 1)$, with n_{live} the other occupied wires, $n_{\text{ad}} = w - 1$ the magnitude address bits, n_{sign} the sign wire, n_{ret} the retained first-segment bits, B banks of n_{seg} bits and L rows (Appendix B). For a 17-bit window the first segment gives $513 + 16 + 1 + 4 \cdot 228 + 13 = 1,455$ and the second $513 + 16 + 1 + 228 + 24 \cdot 28 + 2 + 11 = 1,443$; for 18 bits, $512 + 17 + 1 + 4 \cdot 228 + 14 = 1,456$ and $512 + 17 + 1 + 228 + 24 \cdot 28 + 2 + 12 = 1,444$.

Kernels. During the interleaved passes the live wires are the 18 address bits, the two 256-bit coefficient registers, the two walk registers of width W_{i+1} , the orientation and sign bits, the history bits written so far (or not yet erased) and the 257 scratch wires of one cell: exactly $789 + i + 2W_{i+1}$ at an unsplit cell $i < T - 64$ and $790 + T$ in the last 64 cells (Section 3.5). The history grows by one wire per step while $2W_{i+1}$ shrinks by about $2 \cdot 0.35$ per step, so the

occupancy rises towards the end of the walk, up to 1,502 for $T = 710$ and 1,510 for $T = 720$. The split rule of Section 3.5 caps it at 1,457: a cell starts with at most 1,245 (or 1,253) live qubits, a split cell adds at most 194, and $1,253 + 194 = 1,447 \leq 1,457$. This costs 27 Toffolis for each of the 242 (or 273) split cells. The recording pass of the multiplication and the erasing pass of the division, which run without cells, peak at 1,306 qubits.

Square. The recursive squarer keeps its cross-term registers live: 130 bits at the top level and 66 or 68 at the next level for a 128-bit square. The stage peaks at 1,379 qubits.

Terminal gadget. The selector lookup with 128 banks of 9 bits dominates the 18-bit gadget’s peak of 1,436.

6.2 Fewer qubits: the balanced split

The kernels are the only stage that needs 1,457 qubits; the square stage peaks at 1,379 and the modular additions at 1,042. With the balanced split of Section 3.5 the kernels fit within 1,380 qubits (710-step walk, 528 split cells per call) or 1,388 (720-step walk, 523). Every lookup, including the initial lookup and the terminal selector, is then planned by the dynamic program of Appendix B with the smaller budget. Narrower banks make reads dearer: an ordinary window’s five reads and their cleanup cost 112,643 Toffolis for 17 bits (against 103,053 at 1,457) and 222,149 or 221,669 for 18 bits (198,904); the initial lookup costs 110,165 at 1,380 qubits and 109,410 at 1,388; and the selector of an 18-bit last window costs 2,292 (gadget 3,822, peak 1,151). Nothing else changes. The two designs cost 28,248,654 and 28,723,852 Toffolis per run.

6.3 More qubits: no carry splits and wider lookups

With a kernel budget of 1,502 qubits no 710-step cell needs splitting, which saves $2 \cdot 242 \cdot 27 = 13,068$ Toffolis per ordinary point addition; with 1,510 the same holds for the 720-step kernel, which saves $2 \cdot 273 \cdot 27 = 14,742$. The lookups, planned for the larger budget, use more banks: an ordinary window’s reads cost 98,868 (17 bits) and 190,819 (18 bits) at 1,502 qubits, and 96,723 and 185,749 at 1,510; the initial lookup costs 98,139 and 97,081. In every design of this paper every lookup is planned for the design’s own peak in this way.

6.4 Resident table columns

An ordinary point addition reads five table columns (Section 2.3), and each read releases its 256-bit word by X measurement right after use. Above 1,502 qubits the designs spend further space on keeping the first column read, $u + \tau$, live for the whole point addition in a *resident* register Col_0 , and, with 256 more qubits, the second, $v + \tau$, in Col_1 . Two other columns are affine functions of the resident one and can be formed from it by modular additions, which cost 351 Toffolis each against 20,509 to 39,628 for a read.

With one resident column the ordinary pipeline becomes

$$\begin{aligned} \text{Col}_0 &\leftarrow u_m + \tau \text{ (read, kept),} & X &\leftarrow X - \text{Col}_0, & Y &\leftarrow Y - (v_m + \tau), & Y &\leftarrow Y/X, \\ X &\leftarrow X + \text{Col}_0 \text{ (three times),} & X &\leftarrow X + ((\rho - 3\tau) \bmod p), \\ X &\leftarrow X - Y^2, & X &\leftarrow X + (-\rho \bmod p), & Y &\leftarrow YX, & Y &\leftarrow Y - (v_m - \tau), \\ X &\leftarrow X - \text{Col}_0, & X &\leftarrow X + 2\tau, & X &\leftarrow \text{NOT}(X), \end{aligned}$$

followed by the release of Col_0 by X measurement and the steps that end every window. Since $3(u_m + \tau) + \rho - 3\tau = 3u_m + \rho$ and $-(u_m + \tau) + 2\tau = -(u_m - \tau)$, the registers pass through exactly the values of the ordinary pipeline; the mask ρ is still fresh and uniform in every window, now added as a classical constant, and $2\tau = c - 1$. Columns 2 and 4 are no longer read: three

reads instead of five, for four more modular additions (1,404 Toffolis). With two resident columns the first y-column step becomes $Y \leftarrow Y - \text{Col}_1$ and the second becomes the pair $Y \leftarrow Y - \text{Col}_1$, $Y \leftarrow Y + 2\tau$, so column 3 is not read either: two reads, for five more additions (1,755 Toffolis).

Phase. Each resident register holds a fixed function of the magnitude address, so its release by X measurement leaves an address-dependent sign exactly like a measured lookup bank. Its outcome mask enters the window’s phase function (Lemma B.1) and is removed by the window’s single phase correction before the address is unfolded; no extra Toffoli is needed.

Qubits. Each resident column occupies 256 qubits through the whole window, so the kernels run at kernel budget n_{ker} equal to the peak less 256 per resident column, and every read after the first is planned with the resident wires counted as occupied. One resident column on kernels at 1,457, 1,465, 1,502 and 1,510 qubits gives designs at 1,713, 1,721, 1,758 and 1,766 qubits; two resident columns on split-free kernels give 2,014 and 2,022. An ordinary window’s reads then cost 56,758 (17 bits) and 108,715 (18 bits) at 1,758 qubits with one column, and 31,404 and 58,409 at 2,014 qubits with two.

The last window. The terminal window measures its slope and so has no $v - \tau$ step; it needs $v + \tau$ only for its first subtraction. In the two-column designs it therefore keeps only Col_0 , and the 256 qubits that Col_1 would occupy hold the denominator copy of Section 6.5. In the one-column designs that copy displaces Col_0 during the division: column 0 is read for the first subtraction and released, and read again into Col_0 after the division, where it supplies $3u_m + \rho$ and $u_m - \tau$ as above.

The additions of a resident register and the constant additions of 2τ carry small extra error charges (Section 8.4).

6.5 Measuring the last division’s history

The eraser of a division (Section 3.3) exists to restore the denominator, which the recorder consumes. In the terminal window the denominator register can instead be copied before the division and the history simply measured.

1. *Copy.* Before the terminal division, copy the x-coordinate register, which holds the denominator $x_1 - u_m$, into a clean 256-qubit register with 256 CNOTs.
2. *Record and replay.* Run the recorder and the halving cells exactly as in Section 3.6. The recorder consumes the original register, and its terminal clearing releases the walk registers as usual.
3. *Measure instead of erase.* After the division’s endpoint, X-measure the T history bits and the three bits o, s_u, s_v (713 or 723 wires) with outcome mask ϕ_h and release them. The copy now serves as the x-coordinate register for the rest of the window.

This removes the eraser, 99,690 Toffolis for $T = 710$ and 101,287 for $T = 720$, and adds none. Let $\mathbf{h}(x)$ be the vector of history and endpoint bits of the walk on x . On every denominator accepted by the walk predicate the measured bits equal $\mathbf{h}(x_1 - u_m)$, so the measurements leave the phase $(-1)^{\langle \phi_h, \mathbf{h}(x_1 - u_m) \rangle}$. Like the phase of the measured square scratch (Section 5.4), it is diagonal and is carried unchanged through the rest of the window. Once x_3 is measured, $x_3 = \lambda^2 - x_1 - u_m$ gives $x_1 - u_m = \lambda^2 - x_3 - 2u_m$, and $\lambda = \lambda_\eta(m)$ on the good domain, so the phase is a function of (m, η) . It joins the terminal phase table of step 7 of Section 5.2 as the extra term

$$F_{\text{hist}}(m, \eta) = \langle \phi_h, \mathbf{h}(\lambda_\eta(m)^2 - x_3 - 2u_m) \rangle.$$

The table keeps its 2^w entries and its cost; the classical compilation adds 2^w runs of the walk on 256-bit integers. The proof of Lemma 5.1 applies unchanged, with the history among the

measured values. A denominator rejected by the walk predicate is already charged in full by the kernel’s walk term, which the terminal division keeps; no error term is added, and none is removed.

The copy occupies 256 qubits while the division runs, so the terminal division has kernel budget n_{ker} equal to the peak less 256 (less 512 when Col_0 is also held). The division then costs $C_{\text{kernel}} - C_{\text{rec}}$: 344,166 without splits for $T = 710$, 350,700 with 242 split cells; 352,091, 358,598 and 359,462 for $T = 720$ with 0, 241 and 273. All designs from 1,713 qubits upward use it. In the one-column designs the second read of column 0 takes back part of the saving: at 1,758 qubits the run saves 73,843 Toffolis rather than 99,690.

7 Resource totals and the frontier

Every design runs the same sequence: the initial lookup, 25 ordinary point additions (twenty of 17 bits and five of 18 bits) and a terminal 18-bit point addition. Its Toffoli count is therefore

$$C_{\text{init}} + 20 C_{17} + 5 C_{18} + C_{\text{term}},$$

with the component counts given below.

7.1 The two designs at 1,457 qubits

For the 710-step design at 1,457 qubits,

$$101,323 + 20 \times 1,051,943 + 5 \times 1,147,794 + 640,089 = 27,519,242$$

Toffolis per run, at a peak of 1,457 logical qubits.

Table 9. The 710-step design at 1,457 qubits: Toffoli count per run, by component.

Component	Count	Toffolis each	Toffolis
Initial lookup ($88,406 + 11,897 + 1,020$)	1	101,323	101,323
Ordinary 17-bit point additions	20	1,051,943	21,038,860
Ordinary 18-bit point additions	5	1,147,794	5,738,970
Terminal 18-bit point addition	1	640,089	640,089
Total			27,519,242

Split by function: the 51 kernel calls cost $51 \times 450,390 = 22,969,890$ (83.5%); the outer arithmetic $25 \times 48,110 + 27,492 = 1,230,242$ (4.5%); lookups, phase tables and the terminal gadget the remaining 3,319,110 (12.1%). Here $48,110 = 2,106 + 46,004$ is the outer arithmetic of an ordinary point addition and $27,492 = 1,755 + 25,737$ that of the terminal one.

The 720-step design at 1,457 qubits differs only in its kernels, which cost 10,359 more Toffolis per call (273 split cells instead of 242):

$$101,323 + 20 \times 1,072,661 + 5 \times 1,168,512 + 650,448 = 27,519,242 + 51 \times 10,359 = 28,047,551.$$

Its error budget is 3.6 times smaller (Section 8.5), which raises the certified success from 0.746924 to 0.789234 and lowers the cost per certified success from 36,843,419 to 35,537,675 Toffolis.

7.2 Designs from 1,380 to 2,022 qubits

Tables 10 and 11 give the components of every design. The kernel budget and split cells follow Section 3.5, the lookups are planned for each design’s peak (Section 6.3), the resident columns follow Section 6.4, and the terminal division of the designs from 1,713 qubits upward measures its history (Section 6.5).

Table 10. Ordinary point additions of every design. “Resident” is the number of resident columns and n_{ker} the kernel budget. “Outer” is the square subtraction (46,004) plus the modular additions and subtractions: six (2,106), ten with one resident column (3,510) or eleven with two (3,861). “Lookups” includes the window’s phase correction. A window costs two kernel calls plus the outer arithmetic and the lookups.

Peak	Walk	Resident	n_{ker}	Split cells	Kernel call	Outer	Lookups, 17 bits	Lookups, 18 bits	Window, 17 bits	Window, 18 bits
1,380	710	0	1,380	528	458,112	48,110	112,643	222,149	1,076,977	1,186,483
1,388	720	0	1,388	523	467,499	48,110	112,643	221,669	1,095,751	1,204,777
1,457	710	0	1,457	242	450,390	48,110	103,053	198,904	1,051,943	1,147,794
1,457	720	0	1,457	273	460,749	48,110	103,053	198,904	1,072,661	1,168,512
1,502	710	0	1,502	0	443,856	48,110	98,868	190,819	1,034,690	1,126,641
1,510	720	0	1,510	0	453,378	48,110	96,723	185,749	1,051,589	1,140,615
1,713	710	1	1,457	242	450,390	49,514	58,676	112,779	1,008,970	1,063,073
1,713	720	1	1,457	273	460,749	49,514	58,676	112,779	1,029,688	1,083,791
1,721	720	1	1,465	241	459,885	49,514	58,412	111,939	1,027,696	1,081,223
1,758	710	1	1,502	0	443,856	49,514	56,758	108,715	993,984	1,045,941
1,766	720	1	1,510	0	453,378	49,514	55,890	106,677	1,012,160	1,062,947
2,014	710	2	1,502	0	443,856	49,865	31,404	58,409	968,981	995,986
2,022	720	2	1,510	0	453,378	49,865	31,278	58,219	987,899	1,014,840

Table 11. Terminal 18-bit point addition, initial lookup and total Toffolis per run of every design. “Outer” is the terminal outer arithmetic: the square subtraction with measured scratch (25,737) plus five modular additions and subtractions (1,755), or nine with a resident column (3,159). From 1,713 qubits upward the terminal division omits its eraser (Section 6.5); its split cells are those of the ordinary kernels. The total is the initial lookup plus twenty 17-bit and five 18-bit ordinary windows of Table 10 plus the terminal window.

Peak	Walk	Terminal division	Outer	Lookups	Gadget	Terminal window	Initial lookup	Toffolis per run
1,380	710	458,112	27,492	177,108	3,822	666,534	110,165	28,248,654
1,388	720	467,499	27,492	176,724	3,822	675,537	109,410	28,723,852
1,457	710	450,390	27,492	158,512	3,695	640,089	101,323	27,519,242
1,457	720	460,749	27,492	158,512	3,695	650,448	101,323	28,047,551
1,502	710	443,856	27,492	152,044	3,695	627,087	98,139	27,052,231
1,510	720	453,378	27,492	147,988	3,695	632,553	97,081	27,464,489
1,713	710	350,700	28,896	98,277	3,695	481,568	89,504	26,065,837
1,713	720	359,462	28,896	98,277	3,695	490,330	89,504	26,592,549
1,721	720	358,598	28,896	95,997	3,695	487,186	89,239	26,536,460
1,758	710	344,166	28,896	95,787	3,695	472,544	85,444	25,667,373
1,766	720	352,091	28,896	95,757	3,695	480,439	85,420	26,123,794
2,014	710	344,166	28,896	57,645	3,695	434,402	73,204	24,867,156
2,022	720	352,091	28,896	57,455	3,695	442,137	72,277	25,346,594

In every design the peak is reached in the kernels and equals the design's qubit count; the lookups are planned not to exceed it, and the square stage (1,379 qubits plus 256 per resident column), the modular additions and the terminal gadget stay below it.

7.3 The qubits-versus-gates frontier

Figure 1 plots all thirteen designs, by Toffolis per run and by Toffolis per certified success. Section 8.8 gives the certified success of each.

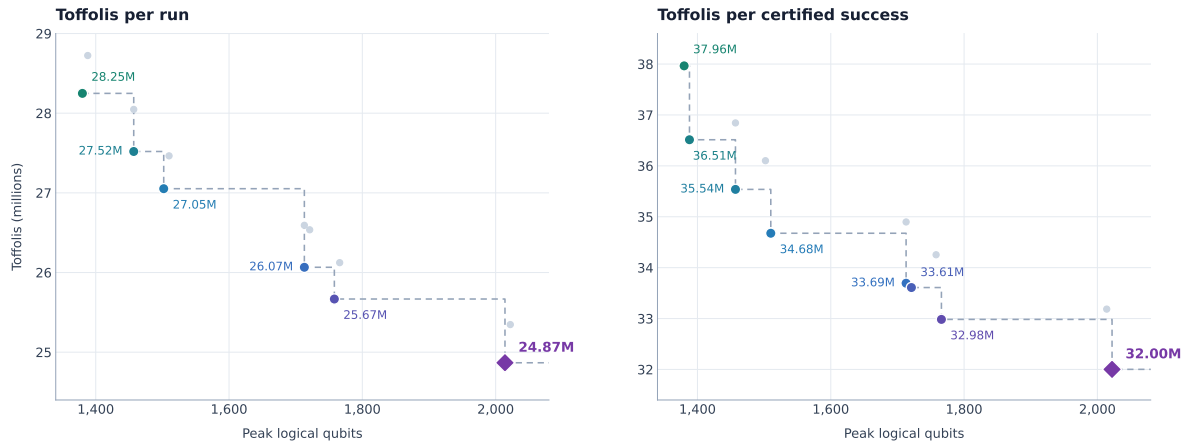


Figure 1. Toffoli gates per run (left) and per certified success under the standard recovery rule (right), against peak logical qubits, for the thirteen designs of Table 1. Colored points are the designs that no other design beats with fewer or equally many qubits; gray points are the other designs. The 710-step designs form the frontier per run and the 720-step designs (with the 1,380-qubit design) the frontier per certified success. The reference circuit of [1] (39.26 million per run and 70.99 million per certified success at 1,457 qubits) lies far above the plotted range.

Three observations follow. First, the two walk lengths give two parallel families: at every qubit count the 720-step design costs about 2% more gates per run and 3.5% to 4% fewer per certified success, because its error budget is 3.3 to 4.2 times smaller. Second, the curve is steep just above 1,457 qubits and again at 1,713 to 1,758: 45 qubits remove every carry split (467,011 Toffolis per run, with the lookup re-planning) and the first resident column with the measured history saves another 1.38 million. Third, the curve flattens beyond about 2,000 qubits: with two resident columns the lookups cost about a million Toffolis per run, and the 51 kernel calls, which no amount of extra space shortens in this construction, are 91% of the run. Below 1,457 qubits the balanced split costs about 9,500 to 9,800 Toffolis per qubit saved.

8 The success certificate

This section proves the success bounds of every design. The argument has four parts: an analytic bound for every approximate arithmetic step; a statistical bound for the one part that is not analytic, the walk's rare failures; a transfer from uniform field inputs to the inputs that actually occur; and a theorem that turns the total error into a success probability, using a lower bound on the success of the ideal algorithm.

8.1 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. We bound the probability that any of

the 26 point 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. Every local failure event is defined on that ideal trajectory with the same public randomness and measurement record, so the union bound applies without any independence between windows, between kernel calls or between steps. If no event occurs, induction over the operations shows that the actual trajectory equals the ideal one, including phases and all released zeros.

8.2 Replay

Theorem 8.1. Let $p = 2^n - c$ with $p > 8c$, $4c \leq 2^\kappa$, $\kappa < n$ and $1 \leq \delta \leq n$. Fix a history produced by an exact, convergent walk and let y be uniform on $\mathbb{F}_p$. A replay of T cells, n_{split} of them in split form, together with its two interfaces and two conditional negations, fails with probability at most

$$q_{\text{rep}} = T \frac{(4c - 1)[p/2^\kappa] + 2^{n-\delta+1} + 2c + 1}{p} + n_{\text{split}} 2^{1-\delta'} + \frac{c 2^{n-\kappa}}{p} + 2\left(\frac{2c}{2^\kappa} + \frac{2c}{p}\right).$$

The cell term is proved in Appendix D.1: at every ideal step the two coefficient registers are fixed linear functions $\alpha y, \alpha_{\text{op}} y$ of the one uniform y , so each register is uniform unless its coefficient is identically zero, and that degenerate family is treated directly. A value error requires the ideal encoded result to fall in a band of $4c - 1$ residues modulo 2^κ ; a phase-only error requires the result to lie within $2^{n-\delta} + c$ of R_β . The interface term counts the exact bad sets of the two interfaces. The split term charges $2^{1-\delta'}$ for each truncated carry repair, and the last term charges each conditional negation. The union is taken over the unconditioned ideal trajectory.

For the 710-step kernel at 1,457 qubits ($\kappa = 60$, $\delta = 27$, $n_{\text{split}} = 242$) this gives $q_{\text{rep}} < 2.478436 \times 10^{-5}$, and for the 720-step kernel ($\kappa = 62$, $\delta = 29$, $n_{\text{split}} = 273$) $q_{\text{rep}} < 9.437093 \times 10^{-6}$; Table 13 lists every kernel used. The cell term contributes about 2.98×10^{-8} per cell at $\kappa = 60, \delta = 27$ and 7.45×10^{-9} at $\kappa = 62, \delta = 29$, and the split term 1.49×10^{-8} per split cell at either split boundary.

8.3 The walk: a statistical certificate

What can fail. The walk of Section 3.1 has a fixed number of steps T and fixed register widths W_i , chosen in advance and the same for every input. For a rare denominator x the walk either has not finished after its T steps (its registers have not both reached ± 1), or a register value outgrows the width allotted to it at some step. On such an input the circuit would give a wrong result. We call this a *walk failure*. The walk predicate of Section 3.2 is the exact test for it: it rejects exactly the walks that have not finished and the walks with a width overflow.

Let q_{walk} be the fraction of nonzero denominators x that fail. On every other denominator the recorder writes the exact convergent history of Lemma 3.1, releases only zero wires and leaves no residual phase, and the eraser restores the denominator; this follows from exact gate identities by induction over the steps, not from tests. A kernel call on a uniform denominator and a uniform second input therefore fails with probability at most $q_{\text{walk}} + q_{\text{rep}}$.

How q_{walk} is bounded. We have no formula for q_{walk} , so we bound it by sampling. For each walk schedule we drew 100,000,000 (10^8) random denominators, each one a *sample*, ran the walk on each in exact integer arithmetic, and counted how many failed: 10,769 for the 710-step schedule and 2,138 for the 720-step schedule. Lemma 8.2 turns such a count into an upper bound on q_{walk} that holds with overwhelming confidence.

Lemma 8.2. For N_s independent uniform trials with n_{fail} failures and $\Xi = \lceil 7\xi/10 \rceil$, with confidence at least $1 - 2^{-\xi}$,

$$q_{\text{walk}} \leq q_{\text{walk}}^+ = \min\left\{1, \frac{n_{\text{fail}} + \Xi + \lceil \sqrt{\Xi^2 + 2n_{\text{fail}}\Xi} \rceil}{N_s}\right\}.$$

Proof. For true mean $\bar{n}_{\text{fail}} = N_s q_{\text{walk}}$ and a threshold $n_{\text{thr}} \leq \bar{n}_{\text{fail}}$ the Chernoff lower-tail bound gives $\Pr[n_{\text{fail}} \leq n_{\text{thr}}] \leq \exp(-(n_{\text{fail}} - n_{\text{thr}})^2/2\bar{n}_{\text{fail}})$. The larger root of $(\bar{n}_{\text{fail}} - n_{\text{thr}})^2 = 2\bar{n}_{\text{fail}}\Xi$ is $\bar{n}_{\text{fail}} = n_{\text{thr}} + \Xi + \sqrt{\Xi^2 + 2n_{\text{thr}}\Xi}$, so if $q_{\text{walk}}^+ < q_{\text{walk}}$ the observation lies in a tail of probability at most $e^{-\Xi}$; rounding the root up only enlarges q_{walk}^+ ; and $\ln 2 < 7/10$ gives $e^{-\Xi} < 2^{-\xi}$. $\square$

Sampling conditions. The walk length, the widths, the cell parameters and the failure test are fixed in advance, independently of the samples, and the $N_s = 10^8$ denominators are drawn uniformly from the nonzero residues modulo p , so Lemma 8.2 applies to the observed counts.

Table 12. Walk certification for the two schedules: number of sampled denominators, failures of each type (the two failures of the walk predicate that can occur), and the resulting upper bound q_{walk}^+ on the failure fraction, with $\xi = 129$ and $\Xi = 91$. No other kind of failure occurred.

Schedule	Samples	Width overflows	Walks not finished	Failures n_{fail}	q_{walk}^+
710 steps	10^8	5,838	4,931	10,769	$12,263/10^8 = 1.2263 \times 10^{-4}$
720 steps	10^8	1,283	855	2,138	$2,860/10^8 = 2.860 \times 10^{-5}$

For the 710-step schedule, $q_{\text{walk}}^+ = (10,769 + 91 + \lceil \sqrt{8,281 + 1,959,958} \rceil)/10^8$ with $\lceil \sqrt{1,968,239} \rceil = 1,403$; for the 720-step schedule $\lceil \sqrt{397,397} \rceil = 631$. Each bound holds with confidence at least $1 - 2^{-129}$ under the independent-uniform sampling model. The samples come from a deterministic pseudorandom generator, so every confidence statement in this paper holds under the model that its output behaves as independent uniform draws; no unconditional statistical confidence is claimed.

The 720-step schedule fails far less often: about 5 times fewer failures were observed, and its certified bound q_{walk}^+ is about 4.3 times lower. Its extra ten steps catch most slow walks, and its widths carry a larger margin.

8.4 Transfer to the actual inputs

The kernels receive coordinate differences, not uniform field elements. By the pair-density bound each pair (accumulator, table point) has probability at most $1/D_0$, $D_0 = (r-1)(r-K)$. A bad denominator set of fraction q_{walk} has at most $2(r-1)(p-1)q_{\text{walk}}$ point-pair preimages, and a bad set of field pairs of fraction q_{pair} at most $4p(p-1)q_{\text{pair}}$ (Appendix D.3). One kernel call therefore contributes at most

$$\epsilon_{\text{call}} = D_{\text{walk}} q_{\text{walk}}^+ + D_{\text{pair}} q_{\text{rep}}, \quad D_{\text{walk}} = \frac{2(p-1)}{r-K}, \quad D_{\text{pair}} = \frac{4p(p-1)}{(r-1)(r-K)},$$

with D_{walk} and D_{pair} numerically indistinguishable from 2 and 4. Table 13 gives q_{rep} and ϵ_{call} for every kernel used.

Table 13. Replay failure bound q_{rep} (Theorem 8.1) and error term ϵ_{call} per kernel call, for every kernel used by the designs. The walk term $D_{\text{walk}}q_{\text{walk}}^+$ is below 2.4527×10^{-4} for $T = 710$ and 5.721×10^{-5} for $T = 720$. As elsewhere in this section, every printed bound is rounded up.

Kernel	Split cells	q_{rep}	ϵ_{call}
710-step	0	2.117828×10^{-5}	3.299732×10^{-4}
710-step	242	2.478436×10^{-5}	3.443975×10^{-4}
710-step	528	2.904610×10^{-5}	3.614444×10^{-4}
720-step	0	5.369076×10^{-6}	7.867631×10^{-5}
720-step	241	8.960256×10^{-6}	9.304103×10^{-5}
720-step	273	9.437093×10^{-6}	9.494837×10^{-5}
720-step	523	1.316239×10^{-5}	1.098496×10^{-4}

The outer arithmetic has analytic bounds per window (Appendix D.2): $p_{\text{add}} \leq 1.746230 \times 10^{-9}$ for the additions, $p_{\text{sub}} \leq 9.080396 \times 10^{-9}$ for the four subtractions, $p_{\text{sq}} \leq 6.868505 \times 10^{-9}$ for the square subtraction, and $[4 + 32(\tau + 1)]/(r - K)$ for the affine and encoding-range exclusions. Their sum is $\epsilon_{\text{out}} = 1.769513 \times 10^{-8}$ per ordinary window. The recursive squarer and the clean-target correction are exact integer substitutions, so the square’s bound is unchanged. Folding the mask into the third column keeps the charge of the unfolded circuit: conditional on the accumulator and table point, $\rho' = (3u + \rho) \bmod p$ is uniform, so the fused addition sees exactly the input distribution of the unfolded circuit’s first mask addition, and the charge of the omitted middle addition is conservatively retained.

Resident columns. A window that keeps Col_0 (Section 6.4) adds three additions of Col_0 into X , the constant addition of $(\rho - 3\tau) \bmod p$ and the constant addition of 2τ , and replaces the read of $u - \tau$ by a subtraction of Col_0 . Fixing both inputs of an addition of Col_0 fixes u and x_1 , so at most four point pairs, and the three additions cost $3 \cdot 4N_{\text{add}}/D_0$, where N_{add} is the number of bad input pairs of one modular addition (Appendix D.2). The constant $(\rho - 3\tau)$ addition, with the mask and $x_1 + 2u$ fixed, has at most $2(r - 1)$ point-pair preimages, exactly the charge of the middle addition already retained in p_{add} . The subtraction of Col_0 has the same multiplicity as the subtraction it replaces, so p_{sub} is unchanged. The constant $2\tau = c - 1$ is added to a canonical target t : there is never a carry, the phase comparison is always right, and the value is wrong only for $t \in [p - c + 1, p)$; such a target fixes the x-coordinate of the output point, so this costs at most $(c - 1) \cdot 2/(r - K)$. Hence

$$\epsilon_{u\tau} = \frac{12N_{\text{add}}}{D_0} + \frac{2(c - 1)}{r - K} = 2.095476 \times 10^{-9}.$$

A window that also keeps Col_1 adds the constant 2τ on Y , whose target fixes the output’s y-coordinate (at most three points), and replaces the read of $v - \tau$ by a subtraction of the same multiplicity: $\epsilon_{v\tau} = 3(c - 1)/(r - K) < 1.2 \times 10^{-67}$. The measured division history adds no term (Section 6.5).

8.5 Error composition

Every term is attached to the step that incurs it.

Ordinary window: $2\epsilon_{\text{call}} + \epsilon_{\text{out}}$, plus $\epsilon_{u\tau}$ with one resident column and $\epsilon_{u\tau} + \epsilon_{v\tau}$ with two.

Terminal window: one kernel call, and the outer term without the column-3 subtraction, whose multiplicity is 9 of the 26 in p_{sub} : $\epsilon_{\text{call}} + \epsilon_{\text{out}} - \frac{9}{26}p_{\text{sub}}$, plus $\epsilon_{u\tau}$ in every design with resident columns, since its terminal window keeps or re-reads Col_0 . The gadget, the measured square

scratch and the measured history are exact on the good domain and add nothing. We add the redundant term $K/(r - K)$ of Section 5.3.

Whole run. Every design uses one walk schedule and one kernel budget, so all 51 kernel calls have the same ϵ_{call} (Table 13, with the split cells of Table 10), and

$$\epsilon = 51 \epsilon_{\text{call}} + 26 \epsilon_{\text{out}} - \frac{9}{26} p_{\text{sub}} + 25 \epsilon_{\text{res}} + \epsilon'_{\text{res}} + \frac{K}{r - K},$$

where $\epsilon_{\text{res}} = 0$, $\epsilon_{u\tau}$ or $\epsilon_{u\tau} + \epsilon_{v\tau}$ for zero, one or two resident columns and $\epsilon'_{\text{res}} = \epsilon_{u\tau}$ when the design has resident columns and 0 otherwise. For the 710-step design at 1,457 qubits,

$$\epsilon = 51\epsilon_{710} + 26\epsilon_{\text{out}} - \frac{9}{26}p_{\text{sub}} + \frac{K}{r - K} = 0.0175647261 \dots,$$

with $\epsilon_{710} < 3.443975 \times 10^{-4}$; for the 720-step design at 1,457 qubits, with $\epsilon_{720} < 9.494837 \times 10^{-5}$, $\epsilon = 0.0048428238 \dots$. Table 14 lists ϵ for every design. Each design relies on the certificate of one walk schedule only, so each holds with confidence at least $1 - 2^{-129}$, under the sampling model of Section 8.3. The kernel calls contribute all but about 5×10^{-7} of every ϵ .

8.6 From error to success

Theorem 8.3. Let the ideal normalized state be $|\psi\rangle = \sum_{\mathbf{k}} \varrho_{\mathbf{k}} |\mathbf{k}\rangle |\Phi_{\mathbf{k}}\rangle$ with normalized blocks, and for a random realization ω let $|\tilde{\psi}_{\omega}\rangle = \sum_{\mathbf{k}} \varrho_{\mathbf{k}} |\mathbf{k}\rangle |\Phi'_{\omega, \mathbf{k}}\rangle$ with the same amplitudes and normalized blocks. Suppose that for every label $\mathbf{k}$, outside a bad event of probability $\epsilon_{\mathbf{k}} \leq \epsilon$, $|\Phi'_{\omega, \mathbf{k}}\rangle = |\Phi_{\mathbf{k}}\rangle$ including phase. For a projector Π acting on the label 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 $\mathbb{E}\Phi'_{\omega, \mathbf{k}} = (1 - \epsilon)\Phi_{\mathbf{k}} + \Delta_{\mathbf{k}}$ with $\Delta_{\mathbf{k}} = (\epsilon - \epsilon_{\mathbf{k}})\Phi_{\mathbf{k}} + \mathbb{E}[\mathbf{1}_{\text{bad}}\Phi'_{\omega, \mathbf{k}}]$, so $\|\Delta_{\mathbf{k}}\| \leq \epsilon$, and by orthogonality of the labels $\|\Delta\| \leq \epsilon$ for $\Delta = \sum_{\mathbf{k}} \varrho_{\mathbf{k}} |\mathbf{k}\rangle \Delta_{\mathbf{k}}$. Jensen's inequality for the convex function $\|\Pi \cdot\|^2$ and the reverse triangle inequality applied to the mean state $(1 - \epsilon)\psi + \Delta$ give the bound. $\square$

Here the labels are the scalar values $\mathbf{k} = (k_P, k_Q)$ with the Fourier transforms deferred, ω collects the public randomization and the padded measurement record, and Π is the success event of the classical recovery below. Appendix D.4 shows that the circuit has the required block form despite its measurements, resets and feed-forward: every operation between measurements is a basis permutation with a diagonal sign, every X outcome is fair on every branch, resets are purified into an environment record on which nothing is conditioned, and the transcript is padded to a fixed length. Since the bound is non-decreasing in P_0 , any valid lower bound on P_0 may be used.

8.7 The ideal success probability

Write $M_P = 2^{257}$ and $M_Q = 2^{209}$. After the output-only offset and scale are removed, the ideal state before the Fourier transforms is $(M_P M_Q)^{-1/2} \sum_{k_P, k_Q} |k_P\rangle |k_Q\rangle |[(k_P + 2^{48} dk_Q)P]\rangle$. With the group Fourier states $|\chi_a\rangle = r^{-1/2} \sum_u e^{-2\pi i a u / r} |u\rangle P\rangle$ this becomes

$$\frac{1}{\sqrt{r}} \sum_{a=0}^{r-1} \left(\frac{1}{\sqrt{M_P}} \sum_{k_P} e^{2\pi i a k_P / r} |k_P\rangle \right) \left(\frac{1}{\sqrt{M_Q}} \sum_{k_Q} e^{2\pi i a 2^{48} dk_Q / r} |k_Q\rangle \right) |\chi_a\rangle,$$

so the scalar outcomes are a uniform mixture over the label a of two phase estimations, of the phases a/r and ν/r with $\nu = 2^{48} ad \bmod r$.

Classical recovery. From the first outcome j , round rj/M_P to obtain a . From the second outcome j' , take the integer interval $[\lceil r(j' - 1)/M_Q \rceil, \lfloor r(j' + 1)/M_Q \rfloor]$ reduced modulo r , which holds at most $\lfloor 2r/M_Q \rfloor + 1 \leq 2^{48}$ candidates for ν ; for each candidate form $d' = \nu(2^{48}a)^{-1} \bmod r$ and accept it if $[d']P = Q$. A baby-step giant-step search [7] over the interval does the same with 2^{24} baby steps and at most 2^{24} giant steps, about 2^{25} group operations. This is the recovery rule of [1]; it is used for every number in this paper outside Section 9. It always returns d when $a \neq 0$ and both outcomes are one of the two grid points nearest the true phase (a *good* outcome).

Theorem 8.4 (averaged anchor). For every nonzero logarithm d ,

$$P_0 \geq 2\bar{g} - 1 - \frac{6}{r}, \quad \bar{g} = \int_0^1 g(t) dt = \frac{2 \operatorname{Si}(2\pi)}{\pi}, \quad g(t) = \frac{\sin^2(\pi t)}{\pi^2} \left(\frac{1}{t^2} + \frac{1}{(1-t)^2} \right).$$

Proof. (1) For a register of length M_{reg} and phase θ , outcome j has probability

$$\sin^2(\pi M_{\text{reg}} \theta) / (M_{\text{reg}}^2 \sin^2(\pi(\theta - j/M_{\text{reg}})))$$

[8]. With $t = \{M_{\text{reg}}\theta\}$, $|\sin u| \leq |u|$ shows that the two outcomes nearest $M_{\text{reg}}\theta$ have combined probability at least $g(t)$, and $g(0) = 1$ covers $t = 0$. (2) In the first register $t_P = (2^{257}a \bmod r)/r$; in the second, since $2^{209} \cdot 2^{48} = 2^{257}$, $t_Q = (2^{257}ad \bmod r)/r$. As r is an odd prime and $d \neq 0$, both maps $a \mapsto t$ are bijections of the grid $\{0, 1/r, \dots, (r-1)/r\}$, so with a uniform each good event has probability at least $\bar{g}_r = r^{-1} \sum_{i < r} g(i/r)$. No independence of t_P and t_Q is claimed; indeed $t_Q = dt_P$ modulo one. (3) The union bound gives $\Pr[G_P \wedge G_Q] \geq 2\bar{g}_r - 1$, and removing $a = 0$ costs $1/r$. (4) Writing $f(t) = (\sin(\pi t)/(\pi t))^2 = 2 \int_0^1 (1-u) \cos(2\pi ut) du$ gives $|f'| \leq 2\pi/3$, so $g = f(t) + f(1-t)$ is $4\pi/3$ -Lipschitz and $|\bar{g}_r - \bar{g}| \leq 5/(2r)$. (5) By symmetry $\bar{g} = 2 \int_0^1 f$, and integration by parts gives $2 \operatorname{Si}(2\pi)/\pi$. $\square$

The average is over the label a , which is internal randomness of the algorithm, not over keys: the bound holds for every fixed nonzero d . The classical worst-case argument instead bounds each register by $\min_t g(t) = 8/\pi^2$ and gives $P_0 \geq 64(r-1)/(r\pi^4) \approx 0.657023$. Evaluating $\operatorname{Si}(2\pi)$ by its alternating series with a rational enclosure of π gives

$$0.902823333580280626795700377906 \leq \bar{g} \leq 0.902823333580280626795700377907,$$

so $P_0 \geq 0.8056466671605612 \dots$

8.8 The certified bounds

Substituting into Theorem 8.3, with exact rational arithmetic, an upper enclosure of π , a square root rounded down and the final value rounded down to six decimals, gives the bounds of Table 14. For the two designs at 1,457 qubits,

$$P_{\text{success}} \geq [(1-\epsilon)\sqrt{P_0} - \epsilon]^2 \geq 0.746924 \text{ (710-step)}, \quad P_{\text{success}} \geq 0.789234 \text{ (720-step)}.$$

Table 14. Error budget and certified success under the standard recovery rule for every design, with the averaged anchor of Theorem 8.4. “Toffolis per certified success” is the run cost divided by the unrounded success bound, rounded up: an upper bound on the expected gates per success under repeated fresh runs, not an estimate of the true expected cost. The error budget depends only on the walk, the split cells and the resident columns, so designs that share the walk and the split cells have nearly the same bound.

Peak qubits	Walk	ϵ	Certified success	Toffolis per certified success
1,380	710	0.0184341195	0.744075	37,964,775
1,388	720	0.0056027830	0.786674	36,513,027
1,457	710	0.0175647261	0.746924	36,843,419
1,457	720	0.0048428238	0.789234	35,537,675
1,502	710	0.0168290856	0.749339	36,101,457
1,510	720	0.0040129483	0.792034	34,675,864
1,713	710	0.0175647806	0.746924	34,897,574
1,713	720	0.0048428782	0.789234	33,694,121
1,721	720	0.0047456035	0.789562	33,609,086
1,758	710	0.0168291401	0.749338	34,253,359
1,766	720	0.0040130028	0.792034	32,983,150
2,014	710	0.0168291401	0.749338	33,185,462
2,022	720	0.0040130028	0.792034	32,001,879

Every bound holds with confidence at least $1 - 2^{-129}$ under the independent-uniform sampling model of Section 8.3, and under the further assumptions listed in Section 11. They are lower bounds on the success of a single run, averaged over the circuit’s randomization and measurement outcomes, for every fixed nonzero logarithm.

9 A wider classical recovery rule (alternative post-processing)

This section describes an alternative post-processing convention. Every other number in this paper, including the comparison with [1] in Tables 1 and 2, uses the standard recovery rule of [1] restated in Section 8.7. Here only the classical computation that follows the measurements changes: every gate, qubit, measurement and error term of the circuits is unchanged, and so is the error budget ϵ of each design.

9.1 Summary

The quantum part of a run ends with two measured integers, j and j' , each an estimate of a phase. The standard rule succeeds only when each of them is one of the two grid points nearest its true phase; this is why the ideal success probability is only about 0.806 (Theorem 8.4). When a measured value misses, it usually lands only a few grid points away, and a classical computer can test those nearby values as well. The rule $\mathcal{R}_k$ below tests the $2k$ nearest values on each register.

- **What improves.** The ideal success probability rises from 0.806 to at least $1 - 2/(\pi^2 k)$, up to a negligible finite-order term; for $k = 16$ the certified value is 0.987337. For $k \geq 2$, $\mathcal{R}_k$ tests every candidate that the standard rule tests, so the true success probability of the unchanged circuit can only increase: the gain is not merely a sharper bound.
- **Correctness.** Lemma 9.1 shows that $\mathcal{R}_k$ never returns a wrong key, and that it returns the key whenever both measured values lie within distance k of their true phases. Theorem 9.2 bounds the probability of that event for every fixed nonzero key, under the same

assumptions as Section 8 (ideal Fourier rotations, no physical noise). The constant in the bound is enclosed in exact rational arithmetic.

- **Gain for the designs.** With $k = 16$ the certified success of the two designs at 1,457 qubits rises from 0.746924 to **0.918972** (710-step) and from 0.789234 to **0.968243** (720-step), and Toffolis per certified success fall from 36,843,419 to 29,945,660 and from 35,537,675 to 28,967,458 (18.7% and 18.5%). Across all thirteen designs the certified success lies between 0.915652 and 0.971502, and the lowest cost per certified success is 26,090,112 Toffolis, at 2,022 qubits. The price is classical: in the worst case about $1.14 \times 10^9 \approx 2^{30}$ elliptic-curve group operations per run and a precomputed table of 2^{26} points (about 2.4 GB), roughly 20 minutes on one CPU core at an assumed one microsecond per group operation (Section 9.4).

9.2 The recovery rule $\mathcal{R}_k$

The notation is that of Section 8.7: $M_P = 2^{257}$, $M_Q = 2^{209}$, r is the prime group order, $Q = [d]P$, and the ideal outcomes are a uniform mixture over the label $a \in \mathbb{Z}_r$ of two phase estimations, of the phases a/r and ν/r with $\nu = 2^{48}ad \bmod r$. Here $k \geq 1$ is the width parameter of the rule. For a register of length M_{reg} and true phase θ , an outcome j is *k-good* if its circular distance from $M_{\text{reg}}\theta$ is at most k , that is, $|M_{\text{reg}}\theta - j - i'M_{\text{reg}}| \leq k$ for some integer i' . The good outcomes of the standard rule are 1-good.

Given the outcomes (j, j') , the rule $\mathcal{R}_k$ proceeds as follows.

1. **Candidates for a .** Let $\mathcal{C}(j)$ be the set of integers a' with $\lceil r(j - k)/M_P \rceil \leq a' \leq \lfloor r(j + k)/M_P \rfloor$, reduced modulo r , with 0 removed. The interval has length $2kr/M_P < k$, so $\mathcal{C}(j)$ has at most $n_a := \lfloor 2kr/M_P \rfloor + 1 = k$ elements, all distinct modulo r .
2. **Interval for ν .** Let $\nu_0 = \lceil r(j' - k)/M_Q \rceil$ and $n_\nu = \lfloor r(j' + k)/M_Q \rfloor - \nu_0 + 1$. Then $n_\nu \leq N_k := \lfloor 2kr/M_Q \rfloor + 1 \leq 2^{48}k$, with equality for every k used below. The candidates for ν are $\nu_0 + i \bmod r$ with $0 \leq i < n_\nu$. This interval does not depend on a .
3. **Shared baby-step table.** For a table size $\mathcal{T} < r/2$, store the points $[i_0]P$, $1 \leq i_0 < \mathcal{T}$, keyed by x-coordinate; the case $i_0 = 0$ (the identity) is tested directly. The table depends only on the fixed generator P , so it is built once and reused for every run. Building it, together with $[\mathcal{T}]P$, takes at most $\mathcal{T}$ group operations.
4. **One giant-step walk per candidate.** Take the candidates $a \in \mathcal{C}(j)$ in order of increasing distance from rj/M_P . For each, set $\mathcal{Y}_a = [2^{48}a]Q - [\nu_0]P$, and for $i_1 = 0, 1, \dots, \lceil n_\nu/\mathcal{T} \rceil - 1$ test whether $\mathcal{Y}_a - [i_1\mathcal{T}]P$ is a table point (an x-coordinate lookup followed by a comparison of the full point). If $\mathcal{Y}_a - [i_1\mathcal{T}]P = [i_0]P$ with $i = i_1\mathcal{T} + i_0 < n_\nu$, set $d' = (\nu_0 + i)(2^{48}a)^{-1} \bmod r$ and output d' if $[d']P = Q$. Otherwise move to the next candidate. If every candidate fails, output nothing.

For $k = 1$, $\mathcal{R}_1$ tests the same ν -interval as the standard rule, and the same value of a whenever the first outcome is good; it is the standard rule carried out by baby-step giant-step [7]. For $k \geq 2$ the half-width kr/M_P of the first interval exceeds $1/2$, so $\mathcal{C}(j)$ contains the rounded value $\text{round}(rj/M_P)$ used by the standard rule, and the ν -interval contains the standard one.

Lemma 9.1 (correctness and cost). (a) If $\mathcal{R}_k$ outputs d' , then $d' = d$. (b) If $a \neq 0$ and both j and j' are k -good, then $\mathcal{R}_k$ outputs d . (c) In the worst case, $\mathcal{R}_k$ uses at most

$$C_k(\mathcal{T}) = \mathcal{T} + n_a(\lceil N_k/\mathcal{T} \rceil - 1) + 2n_a + 2096$$

group additions or subtractions, plus $n_a \lceil N_k/\mathcal{T} \rceil$ table lookups.

Proof. (a) A test succeeds only if $[\nu_0 + i]P = [2^{48}a]Q = [2^{48}ad]P$. Since P has order r , this gives $\nu_0 + i \equiv 2^{48}ad \pmod{r}$, and as $a \not\equiv 0$ and r is odd, $d' \equiv d$. This holds whether or not the candidate a is the true label, so the final test $[d']P = Q$ is only a safeguard and always passes.

(b) The first outcome is k -good when $|M_P a/r - j - i' M_P| \leq k$ for some integer i' . Then $a - i'r$ satisfies $r(j - k)/M_P \leq a - i'r \leq r(j + k)/M_P$, so the true a lies in $\mathcal{C}(j)$. In the same way the true ν is congruent to $\nu_0 + i$ for exactly one $0 \leq i < n_\nu$, because $n_\nu \leq 2^{48}k < r$. Write $i = i_1\mathcal{T} + i_0$ with $0 \leq i_0 < \mathcal{T}$ and $i_1 < \lceil n_\nu/\mathcal{T} \rceil$. At step i_1 of the walk for the true a , $\mathcal{Y}_a - [i_1\mathcal{T}]P = [i_0]P$. Table entries are unique: $\mathcal{T} < r/2$ excludes both $[i_0]P = [i'_0]P$ and $[i_0]P = -[i'_0]P$ for $i_0 \neq i'_0$, so the x-coordinate lookup followed by the full comparison finds this i_0 . If an earlier candidate succeeds first, its output is also d by (a).

(c) The table costs at most $\mathcal{T}$ operations. A walk costs at most $\lceil N_k/\mathcal{T} \rceil - 1$ subtractions of $[\mathcal{T}]P$, plus one subtraction to form $\mathcal{Y}_a$. Candidates are visited outward from the first one, a_0 , alternately upward and downward; keeping one running point for each direction, each new $[2^{48}a]Q$ costs one addition of $\pm[2^{48}]Q$. The remaining work is the 48 doublings that give $[2^{48}]Q$ and three scalar multiplications, for $[2^{48}a_0]Q$, $[\nu_0]P$ and the final check, each at most $2 \cdot 256$ operations by double-and-add; $48 + 3 \cdot 512 \leq 2096$. $\square$

The search tries the most likely candidate first and stops at the first success, so its typical cost is below the worst case. The balanced table size $\mathcal{T} = \lceil \sqrt{n_a N_k} \rceil = 2^{24}k$ gives a worst case of about $2^{25}k$ operations. We quote a memory-capped table with $\mathcal{T} = 2^{26}$: each entry holds a 32-byte x-coordinate and a 4-byte index, about 2.4 GB in all.

Table 15. Worst-case number of group operations $C_k(\mathcal{T})$ of the rule $\mathcal{R}_k$ for the secp256k1 group order, for a table capped at 2^{26} points and for the balanced table size.

k	Candidates for a	ν -interval N_k	Worst case, $\mathcal{T} = 2^{26}$	Worst case, $\mathcal{T} = 2^{24}k$ (table size)
1 (standard)	1	2^{48}	3.36×10^7 ($\mathcal{T} = 2^{24}$)	3.36×10^7 (2^{24})
4	4	2^{50}	1.34×10^8	1.34×10^8 (2^{26})
8	8	2^{51}	3.36×10^8	2.68×10^8 (2^{27})
16	16	2^{52}	1.14×10^9	5.37×10^8 (2^{28})
32	32	2^{53}	4.36×10^9	1.07×10^9 (2^{29})

9.3 The ideal success probability

Let $f(u) = (\sin \pi u / (\pi u))^2$ with $f(0) = 1$, as in Theorem 8.4. Put

$$g_k(t) = \sum_{i=-k+1}^k f(t-i) \quad (0 \leq t \leq 1), \quad \bar{g}_k = \int_0^1 g_k(t) dt = \int_{-k}^k f(u) du = \frac{2 \operatorname{Si}(2\pi k)}{\pi}.$$

Since f is even, $g_1(t) = f(t) + f(1-t)$ is the function g of Theorem 8.4, and $\bar{g}_1 = \bar{g}$. The last equality follows by integration by parts, because $\sin^2(\pi k) = 0$.

Theorem 9.2. Let $k \geq 1$ with $2k \leq M_Q$, and let $P_0^{(k)}$ be the ideal success probability of $\mathcal{R}_k$. For every nonzero logarithm d ,

$$P_0^{(k)} \geq 2\bar{g}_k - 1 - \frac{5k+1}{r} \geq 1 - \frac{2}{\pi^2 k} - \frac{5k+1}{r}.$$

Proof. The steps follow the proof of Theorem 8.4.

- (1) *Window mass.* Take a register of length M_{reg} with phase θ , and write $M_{\text{reg}}\theta = n_\theta + t$ with n_θ an integer and $0 \leq t < 1$. For $0 < t < 1$, outcome $n_\theta + i$ has probability $\sin^2(\pi t)/(M_{\text{reg}}^2 \sin^2(\pi(t-i)/M_{\text{reg}}))$ [8]; since $|\sin u| \leq |u|$, this is at least $f(t-i)$. The outcomes $n_\theta + i$ with $-k < i \leq k$ are $2k \leq M_{\text{reg}}$ distinct residues modulo M_{reg} , and all are k -good, so the k -good outcomes have total probability at least $g_k(t)$. When $t = 0$ the exact outcome has probability $1 = g_k(0)$.
- (2) *Uniform grid.* As in Theorem 8.4, $t_P = (2^{257}a \bmod r)/r$ and $t_Q = (2^{257}ad \bmod r)/r$. Both maps $a \mapsto t$ are bijections of $\{0, 1/r, \dots, (r-1)/r\}$, because r is an odd prime and $d \neq 0$. With a uniform, each of the events G_P and G_Q (the first, respectively second, outcome is k -good) therefore has probability at least $\bar{g}_{k,r} = r^{-1} \sum_{i < r} g_k(i/r)$. No independence is used.
- (3) *Union bound.* $\Pr[G_P \wedge G_Q \wedge a \neq 0] \geq 2\bar{g}_{k,r} - 1 - 1/r$. By Lemma 9.1(b) this event is contained in the success event of $\mathcal{R}_k$. The success event depends only on (j, j') , so it is a projector on the label registers, as Theorem 8.3 requires.
- (4) *Finite grid.* From $f(t) = 2 \int_0^1 (1-u) \cos(2\pi ut) du$ we get $|f'| \leq 2\pi/3$, so g_k is Lipschitz on $[0, 1]$ with constant $\text{Lip}_k \leq 2k \cdot 2\pi/3 < 5k$; it is continuous at both ends, with $g_k(0) = g_k(1) = 1$. A left Riemann sum with r points then gives $\bar{g}_{k,r} \geq \bar{g}_k - \text{Lip}_k/(2r) \geq \bar{g}_k - 5k/(2r)$. Combined with (3), this gives the first inequality.
- (5) *The constant.* Write $\sin^2 \pi u = (1 - \cos 2\pi u)/2$ and $u^{-2} = \int_0^\infty u' e^{-uu'} du'$. The integrand is non-negative, so Tonelli’s theorem allows the integrals to be exchanged, and since $\cos 2\pi k = 1$,

$$\int_k^\infty f(u) du = \frac{1}{\pi} \Psi(2\pi k), \quad \Psi(\varpi) = \int_0^\infty \frac{e^{-\varpi u'}}{1 + u'^2} du',$$

the auxiliary function of the sine integral [9, §6.2, §6.7]. Hence $\bar{g}_k = 1 - \frac{2}{\pi} \Psi(2\pi k)$. Expanding $1/(1+u'^2) = \sum_{i < n_\Psi} (-1)^i u'^{2i} + (-1)^{n_\Psi} u'^{2n_\Psi}/(1+u'^2)$ shows that $\Psi(\varpi)$ lies between any two consecutive partial sums of $\sum_i (-1)^i (2i)!/\varpi^{2i+1}$. In particular $1/\varpi - 2/\varpi^3 \leq \Psi(\varpi) \leq 1/\varpi$, which gives $1 - 1/(\pi^2 k) \leq \bar{g}_k \leq 1 - 1/(\pi^2 k) + 1/(2\pi^4 k^3)$. $\square$

For $k = 1$ the correction $(5k+1)/r$ is the $6/r$ of Theorem 8.4, and the theorem reduces to Theorem 8.4. For $k \leq 32$ the correction is below 2^{-248} . As in Section 8.7, the average is over the internal label a , not over keys: the bound holds for every fixed nonzero d .

Rigorous values. We evaluate $\bar{g}_k = \sum_{i \geq 0} (-1)^i 4k(2\pi k)^{2i}/((2i+1)(2i+1)!)$ in fixed-point integer arithmetic with outward rounding, with π enclosed by two Machin-type identities, each summed as an alternating arctangent series. Partial sums are taken only beyond the index at which the term ratio $\varpi^2(2i+1)/((2i+3)^2(2i+2))$, evaluated at the upper end of $\varpi = 2\pi k$, drops below one; past that index the terms decrease in magnitude, so a partial sum ending at an odd index is a lower bound and one ending at an even index an upper bound. The enclosures have width below 10^{-120} . They agree with the bounds from the Ψ -series of step (5), and for $k = 1$ they reproduce the enclosure of $\bar{g}$ in Section 8.7.

Table 16. The constant $\bar{g}_k$ and the resulting lower bound on the ideal success probability $P_0^{(k)}$ of the rule $\mathcal{R}_k$ (Theorem 9.2). The row $k = 1$ is the averaged anchor of Theorem 8.4.

k	$\bar{g}_k$ (first 24 digits, common to both ends of the enclosure)	$P_0^{(k)} \geq$
1	0.902823333580280626795700	0.805646667160
4	0.974748450115639943447469	0.949496900231
8	0.987344830356441914930356	0.974689660712

	$\bar{g}_k$ (first 24 digits, common to k both ends of the enclosure)	$P_0^{(k)} \geq$
16	0.993668677710445516664831	0.987337355420
32	0.996833869611170305921802	0.993667739222

The simple closed form $1 - 2/(\pi^2 k)$ is lower than these values by 1.6×10^{-4} at $k = 4$ and by 2.5×10^{-6} at $k = 16$, so a reader who checks only the closed form loses almost nothing.

9.4 Certified success of the designs

Theorem 8.3 is unchanged: it needs only a lower bound on P_0 for a projector on the label registers. We substitute $P_0^{(k)}$ into $[\max\{0, (1 - \epsilon)\sqrt{P_0} - \epsilon\}]^2$ with the exact ϵ of each design, taking square roots rounded down and rounding the result down to six decimals. Toffolis per certified success use the unrounded bound and are rounded up. The column $k = 1$ reproduces the figures of Sections 7 and 8 exactly. The statistical confidence, $1 - 2^{-129}$, is unchanged, because the circuits and their ϵ are unchanged.

Table 17. Certified success probability / Toffolis per certified success under the standard rule and the recommended rule $\mathcal{R}_{16}$, for every design of Table 1. The quantum circuits are identical in every column; only the classical recovery changes. The last row is the worst-case classical cost in group operations (Table 15).

Peak qubits; walk	$k = 1$ (standard)	$k = 16$
1,380; 710	0.744075 / 37,964,775	0.915652 / 30,850,845
1,388; 720	0.786674 / 36,513,027	0.965264 / 29,757,508
1,457; 710	0.746924 / 36,843,419	0.918972 / 29,945,660
1,457; 720	0.789234 / 35,537,675	0.968243 / 28,967,458
1,502; 710	0.749339 / 36,101,457	0.921786 / 29,347,605
1,510; 720	0.792034 / 34,675,864	0.971502 / 28,270,127
1,713; 710	0.746924 / 34,897,574	0.918972 / 28,364,112
1,713; 720	0.789234 / 33,694,121	0.968243 / 27,464,741
1,721; 720	0.789562 / 33,609,086	0.968624 / 27,396,013
1,758; 710	0.749338 / 34,253,359	0.921786 / 27,845,249
1,766; 720	0.792034 / 32,983,150	0.971501 / 26,890,110
2,014; 710	0.749338 / 33,185,462	0.921786 / 26,977,133
2,022; 720	0.792034 / 32,001,879	0.971501 / 26,090,112
Classical worst case ($\mathcal{T} = 2^{26}$; 2^{24} at $k = 1$)	3.4×10^7	1.1×10^9

Recommended convention: $k = 16$. In the worst case the classical search costs about 1.1×10^9 group operations and a 2.4 GB table. Assuming about one microsecond per elliptic-curve group operation on one ordinary CPU core (a conservative figure), that is about 1,100 seconds, roughly 20 minutes, and less with optimized arithmetic or several cores. Going to $k = 32$ (with $P_0^{(32)}$ from Table 16) lowers Toffolis per certified success by only 0.6 to 0.7% and multiplies the worst-case classical cost by nearly four. Once $P_0^{(k)}$ is close to one, the certified success is limited by ϵ .

10 Verification

Every circuit of this paper was executed at full 256-bit width on basis inputs, following the value of every wire and the sign of every input through each gate and measurement. For every design,

its 17-bit and 18-bit ordinary point additions and its 18-bit terminal point addition were executed at the design’s qubit budget on 32 to 64 random inputs each, with at least two independent choices of inputs and measurement outcomes, and compared with direct elliptic-curve arithmetic. Every output was correct with no residual phase, every temporary register returned to zero and the address was restored. The initial lookup was executed in the same way at every budget. In every execution the Toffoli count, the peak number of live qubits and the final number of live qubits equal the formulas of Sections 3 to 7, so the window counts add up exactly to the totals of Table 1.

The parts were also checked on their own: replay cells in plain and split form at both split boundaries, the kernels, the recursive square and the clean-target correction on all values of the top ten source bits, the resident columns, and the measured division history, whose measured bits equalled the classical walk on $x_1 - u_m$ on every input. Exact simulations on small curves confirmed the channel of the terminal window (Lemma 5.1) and every step of Lemma 9.1 and Theorem 9.2.

These executions check the counts and the gate identities, with their phases, on the inputs tried; they are not an exhaustive simulation of the algorithm’s superposition. General correctness and the error bounds rest on the proofs of Sections 8 and 9 and Appendix D, together with the statistical walk certificate.

11 Limitations and scope

The claim is logical. We give complete logical circuits with the stated Toffoli counts, qubit counts and certified success. We have not compiled them onto a physical architecture, and no runtime or physical-qubit estimate follows from the Toffoli count alone; [1] obtains its physical estimate from a specific trapped-ion compilation that we have not repeated.

Classical feed-forward is not counted. The terminal window requires a classical computation in the middle of the run: one square root, one batch inversion over 2^{17} entries, 2^{17} selector words, and a phase table of 2^{18} bits that includes the integer squares of candidate slopes and, in the designs from 1,713 qubits upward, 2^{18} walks of the measured-history term. Its cost, its latency (during which the quantum state must be stored), and the Clifford gates it drives are not in the Toffoli count. The ordinary lookups also rely on classically compiled phase tables, as in [1].

Other operations are not counted. As in [1], single-qubit rotations of the semiclassical Fourier transform are treated as ideal, and Clifford gates, measurements and resets are not counted. A realization that synthesizes the rotations must add their cost and their approximation error to the bound.

The certificate has a statistical part. The walk’s failure rate is bounded from 10^8 pseudo-random samples per walk schedule. The resulting confidence, $1 - 2^{-129}$, holds under the model that the samples are independent and uniform, which rests on the pseudorandom generator being indistinguishable from random for this purpose.

The split-carry charge depends on the schedule. The split-carry repair is charged $2^{1-\delta'}$ per split cell on the basis of a uniform target register, which requires the target’s coefficient to be nonzero (Appendix D.1). This holds at every split cell of every design, at both split boundaries and every kernel budget used, but it must be rechecked for any other walk, split policy or qubit budget.

The comparison with [1] uses its published logical count. Its arithmetic figure is the largest count observed in 100 sampled runs, an approximate maximum, while our counts are

worst-case over all inputs and charge every conditional gate. We do not compare with its compiled Toffoli-injection total.

The main comparison keeps the recovery rule of [1]. Every success bound in this paper outside Section 9, and in particular every entry of the comparison in Table 1, uses the standard recovery rule. The wider rule of Section 9 raises the certified success of the same circuits, but it changes the post-processing convention and adds a classical search of up to about 2^{30} group operations per run, with a precomputed table of 2^{26} points; its bounds rest on the same assumptions as Section 8.

12 Conclusion

We have given complete logical circuits for Shor’s algorithm on secp256k1 at thirteen points between 1,380 and 2,022 logical qubits, from 28,248,654 Toffoli gates per run at 1,380 qubits to 24,867,156 at 2,014, each with a certified single-run success probability of 0.744 to 0.792. At the 1,457 qubits of the published logical estimate of [1], they use 27,519,242 Toffoli gates with certified success at least 0.746924, or 28,047,551 with at least 0.789234: 29.9% and 28.6% fewer Toffoli gates, and about half the Toffoli gates per certified success. The lowest cost per certified success, 32,001,879 Toffoli gates, is reached at 2,022 qubits.

Most of the gate saving comes from field kernels whose control walk follows a fixed public schedule, and from a last point addition that measures its output early and skips its multiplication; recursive squaring, the clean-target correction and mask folding add the rest at 1,457 qubits. The qubit budget is then a dial. Below 1,457 qubits a balanced carry split keeps the kernels within 1,380 qubits for about 9,500 Toffolis per qubit saved. Above it, removing every carry split, planning the lookups for the larger budget, keeping one or two table columns resident and measuring the last division’s history save 2.65 million Toffolis per run by 2,014 qubits. Beyond that, the 51 kernel calls are more than 90% of the cost, so further gains must come from cheaper division and multiplication rather than more space. Compiling the circuits for a physical architecture, and pricing the terminal window’s classical feed-forward, are the natural next steps.

A Measured arithmetic and explicit circuits

Wires within a register are ordered from least to most significant, and a slice $x_{[i,j]}$ holds bits i to $j - 1$; in the pseudocode listings, $A[i:j]$ has the same meaning. Wire names local to a circuit description are set in typewriter type $(u, v, \dots)$. $CX(u \rightarrow v)$ maps $v \mapsto v \oplus u$, $CCX(u, v \rightarrow z)$ maps $z \mapsto z \oplus uv$, and $CZ(u, v)$ multiplies the amplitude by $(-1)^{uv}$.

A.1 Measurement conventions

MX is a destructive X -basis measurement followed by reset. With outcome ϕ it acts on a stored Boolean value ϑ as $|\vartheta\rangle \mapsto 2^{-1/2}(-1)^{\phi\vartheta}|0\rangle$. If $\vartheta = uv$ was computed into a clean wire, it is erased by MX followed by $CZ(u, v)$ when $\phi = 1$ (EraseAND⁺); if $\vartheta = u(1 \oplus v)$, by $CZ(u, v)$ and $Z(u)$ when $\phi = 1$ (EraseAND[−]). Neither uses a Toffoli.

For a clean target t put $\text{maj}(a, b, d) = ab \oplus ad \oplus bd$. The macro $\text{MAJ}(a, b, d \rightarrow t)$ applies CX from d into a and b , $CCX(a, b \rightarrow t)$, CX from d into t , and undoes the first two CNOTs; it uses one Toffoli, preserves its inputs and, when d is a known zero, omits every gate involving d . It is erased by $MX(t)$ followed, when $\phi = 1$, by $CZ(a, b) CZ(a, d) CZ(b, d)$ (only $CZ(a, b)$ if $d = 0$). Every branch is then the input-preserving map times $2^{-1/2}$, also on superpositions entangled with other registers.

$\text{ADD}_{n_+}(\mathbf{X}; \mathbf{Y})$ performs $\mathbf{Y} \leftarrow \mathbf{X} + \mathbf{Y} \bmod 2^{n_+}$: compute the carries $\mathbf{d}_i = \text{maj}(\mathbf{X}[i], \mathbf{Y}[i], \mathbf{d}_{i-1})$ for $0 \leq i < n_+ - 1$ ($\mathbf{d}_{-1} = 0$), XOR $\mathbf{X}[i]$ and, for $i > 0$, $\mathbf{d}_{i-1}$ into each $\mathbf{Y}[i]$, then erase the carries in descending order (before erasing $\mathbf{d}_i$, undo those two XORs on $\mathbf{Y}[i]$, apply the erasure, and restore the sum bit). It costs $n_+ - 1$ Toffolis. Subtraction complements the target before and after. IADD_{n_+} also computes $\mathbf{d}_{n_+-1}$ and returns it as a live carry-out (n_+ Toffolis). A strict comparison computes the carries of $\mathbf{Y} - \mathbf{X}$ with complemented inputs, so that the last carry is $[\mathbf{X} < \mathbf{Y}]$ (n_+ Toffolis), and is erased with complemented-input corrections. A phase-only comparison applies the last majority as its three CZ terms without building it ($n_+ - 1$ Toffolis).

A.2 The integer stages of the replay cell

The source word is the conditionally complemented R_β . Arguments named 0 are known constants, not wires.

Forward stage, $2S + R_\beta$. Allocate clean wires lo, hi and carries $\mathbf{d}_1, \dots, \mathbf{d}_{n-1}$; let $\mathbf{A} = (\text{lo}, S[0], \dots, S[n-1])$.

```

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

```

The first $n - 1$ Toffolis build the carries; the last computes q_1 from the top target bit and the final carry, and the former top wire becomes q_0 .

Inverse stage. After q_0, q_1 are prepared and the short correction is undone, the low word holds ℓ' . Set $\mathbf{A} = (\ell'[0], \dots, \ell'[n-1], q_0)$. By the parity identity for q_0 , the low bits of ℓ' and R_β agree, so the subtraction has no initial borrow.

```

CX(Rbeta[0] -> A[0])
for i = 1, ..., n-1: X(A[i]); MAJ(Rbeta[i], A[i], d[i-1] -> d[i]); X(A[i])
for i = 1, ..., n-1: CX(Rbeta[i] -> A[i]); if i > 1: CX(d[i-1] -> A[i])
CX(d[n-1] -> A[n])
phi1 = MX(q1); if phi1 == 1: CZ(d[n-1], A[n])
for i = n-1, ..., 1:
    CX(Rbeta[i] -> A[i]); if i > 1: CX(d[i-1] -> A[i])
    X(A[i]); eraseMAJ(Rbeta[i], A[i], d[i-1] -> d[i]); X(A[i])
    if i > 1: CX(d[i-1] -> A[i]); CX(Rbeta[i] -> A[i])
release A[0], q1; return S = A[1:n+1]

```

With $I' = q_0 M + \ell' - R_\beta$ the stage outputs $(I' \bmod 2M)/2$, and the measurement leaves the phase $(-1)^{\phi_1(q_1 \oplus [I' < 0])}$, which vanishes on a forward-good image.

Complete cell. Put $\mathbf{t}_0 = \beta q_0$, $\mathbf{t}_1 = \beta q_1$ and $\Upsilon_{\beta q} = \Upsilon_\beta(q) \bmod 2^\kappa$. The correction word is loaded into a clean κ -bit register by XORing, under the controls $1, q_0, q_1, \beta, \mathbf{t}_0, \mathbf{t}_1$ respectively, the masks

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

exact because $q_0q_1 = 0$. The forward cell complements R by CNOTs from β , runs the forward stage, computes $\mathbf{t}_0, \mathbf{t}_1$ (two Toffolis), loads the word, adds it into $Z_{[0,\kappa]}$ with ADD_κ , unloads it and erases $\mathbf{t}_0, \mathbf{t}_1$. It then measures q_1 with outcome ϕ_1 ; if $\phi_1 = 1$ it computes the flag $\mathbf{f} = [\lfloor Z/2^{n-\delta} \rfloor < \lfloor R_\beta/2^{n-\delta} \rfloor]$ with a δ -bit comparator, applies $Z(\mathbf{f}) \text{CZ}(q_0, \mathbf{f})$, which implements $(-1)^{(1-q_0)\mathbf{f}}$, and erases the comparator. Finally it clears q_0 from $q_0 = R_\beta[0] \oplus Z[0] \oplus (1 - \beta)$ by three CNOTs and an X, and undoes the complement of R . The inverse cell complements R , prepares q_0 by the same parity, computes the comparison, obtains q_1 by one Toffoli controlled by $1 - q_0$ and the comparison flag, erases the comparison, loads and subtracts the same word, erases $\mathbf{t}_0, \mathbf{t}_1$, runs the inverse stage and undoes the complement.

Split form. With split boundary $b_{\text{sp}} \in \{128, 192\}$ and $b' = b_{\text{sp}} - \delta'$, the carry loop runs over bits 1 to $b_{\text{sp}} - 1$ with carry wires $\mathbf{d}_1, \dots, \mathbf{d}_{b_{\text{sp}}-1}$; the carry γ_{sp} into bit b_{sp} is copied by a CNOT into a fresh wire $\mathbf{cy}$; the low carries are erased; the loop runs over bits b_{sp} to 255 with carry-in $\mathbf{cy}$ and is erased; then $\mathbf{cy}$ is measured with outcome ϕ_{sp} , and if $\phi_{\text{sp}} = 1$ the strict comparison of $Z_{[b', b_{\text{sp}}]}$ against $R_{[b', b_{\text{sp}}]}$ (the inverse cell uses $\bar{R}_{[b', b_{\text{sp}}]}$ against $D_{[b', b_{\text{sp}}]}$, where $D = I' \bmod 2M$ is the difference word written by the inverse stage, held in $\mathbf{A}$ before its final halving) is computed, its result receives a Z, and it is erased. For $b_{\text{sp}} = 192$ the compared bits are 165 to 191, and for $b_{\text{sp}} = 128$ they are 101 to 127.

A.3 Pruned unary row loading

Let L be the number of rows, $n_L = \lceil \log_2 L \rceil$, row-address wires $\mathbf{x}_0, \dots, \mathbf{x}_{n_L-1}$ and public row words $\Omega_0, \dots, \Omega_{L-1}$ of the table Ω being read. The output wires $\mathbf{0}$ start at zero; $n_L - 1$ prefix wires $\mathbf{z}_0, \dots, \mathbf{z}_{n_L-2}$ are allocated. $\text{Emit}(\mathbf{v}, \Omega_u)$ applies $\text{CX}(\mathbf{v} \rightarrow \mathbf{0}[i])$ for every i with $\Omega_u[i] = 1$.

`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])

```

`top level (nL >= 2):`

```

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

```

The Toffoli count is $C_{\text{load}}(L) = C_{\text{visit}}(0, n_L - 2) + C_{\text{visit}}(2^{n_L-1}, n_L - 2)$ with $C_{\text{visit}}(\mathbf{u}, \mathbf{b}) = 0$ if $\mathbf{u} \geq L$ or $\mathbf{b} < 0$, and $1 + C_{\text{visit}}(\mathbf{u}, \mathbf{b} - 1) + C_{\text{visit}}(\mathbf{u} + 2^{\mathbf{b}}, \mathbf{b} - 1)$ otherwise. For $L = 2^{n_L}$ this is $L - 2$; the plans use $C_{\text{load}}(16,384) = 16,382$, $C_{\text{load}}(2,731) = 2,735$, $C_{\text{load}}(32,768) = 32,766$, $C_{\text{load}}(5,462) = 5,466$, $C_{\text{load}}(87,382) = 87,388$, $C_{\text{load}}(10,923) = 10,928$ and $C_{\text{load}}(1,024) = 1,022$.

A.4 The divide-by-three cell

For bank counts $3 \cdot 2^i$ the magnitude address is divided by three. The cell acts on wires $(\mathbf{r}_0, \mathbf{r}_1, \mathbf{x})$, where $\mathbf{r}_0 + 2\mathbf{r}_1$ is the running remainder $\mathbf{r}$ and $\mathbf{x}$ an address bit:

```

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

```

On a valid remainder it maps $(\mathbf{r}, \mathbf{x}) \mapsto ((2\mathbf{r} + \mathbf{x}) \bmod 3, \lfloor (2\mathbf{r} + \mathbf{x})/3 \rfloor)$ and it permutes all eight basis states. Applied to the n_{ad} address bits from most to least significant, starting from a zero remainder, it leaves $\lfloor m/3 \rfloor$ in the address wires and $m \bmod 3$ in the remainder ($2n_{\text{ad}}$ Toffolis); reversing restores m ($2n_{\text{ad}}$ more).

A.5 The Boolean phase circuit

To apply $(-1)^{F(m)}$ for an arbitrary table F on n_{ad} bits with split n_{lo} : decode the low n_{lo} bits into $2^{n_{\text{lo}}}$ one-hot wires $\text{oh}_{i'}$ (start with $\text{oh}_0 = 1$; for bit i and each $i' < 2^i$ apply $\text{oh}_{i'+2^i} \oplus = \text{oh}_{i'} m[i]$ and $\text{oh}_{i'} \oplus = \text{oh}_{i'+2^i}$; $2^{n_{\text{lo}}} - 2$ Toffolis). Traverse the $n_{\text{ad}} - n_{\text{lo}}$ high bits with the Visit recursion, replacing Emit at high leaf i_{hi} by $\prod_{i_{\text{lo}}: F(i_{\text{lo}} + 2^{n_{\text{lo}}} i_{\text{hi}}) = 1} \text{CZ}(v, \text{oh}_{i_{\text{lo}}})$ ($2^{n_{\text{ad}} - n_{\text{lo}}} - 2$ Toffolis), erase the traversal with its local corrections and erase the decoder in reverse with EraseAND^+ . For each basis address exactly one low wire and one leaf are active, so the product is $(-1)^{F(m)}$. The total is $C_{\text{phase}}(n_{\text{ad}}, n_{\text{lo}}) = 2^{n_{\text{lo}}} + 2^{n_{\text{ad}} - n_{\text{lo}}} - 4$.

A.6 The triangular integer square

For an n_{sq} -bit x and a clean $2n_{\text{sq}}$ -bit z ($n_{\text{sq}} \geq 2$):

1. Subtract the zero-extended x from $z_{[0, n_{\text{sq}}+1)}$.
2. For $i = 0, \dots, n_{\text{sq}} - 2$: zero-extend $x_{[i+1, n_{\text{sq}})}$ to a word of length $n_{\text{sq}} + 1 - i$; apply $\text{CX}(z[n_{\text{sq}} + i] \rightarrow z[n_{\text{sq}} + i + 1])$ (sign extension); complement the word under control of $x[i]$; subtract it from $z_{[2i+1, n_{\text{sq}}+i+2)}$; undo the complement.
3. Apply $\text{CX}(x[n_{\text{sq}} - 1] \rightarrow z[2n_{\text{sq}} - 1])$.

Step 1 costs n_{sq} Toffolis and iteration i costs $n_{\text{sq}} - i$, in total $\frac{1}{2}n_{\text{sq}}(n_{\text{sq}} + 3) - 1$. Writing $x_{<i} = x \bmod 2^i$ and $x_{\geq i} = \lfloor x/2^i \rfloor$, the partial value before iteration i is $z^{(i)} = x_{<i}^2 + 2^i x_{\geq i} (2x_{<i} - 2^i)$ with $x^2 - z^{(i)} = 2^{2i} x_{\geq i} (x_{\geq i} + 1)$, which proves the identity and the validity of the sign extension ($|z^{(i)}| < 2^{n_{\text{sq}}+i}$). The inverse reverses the order and exchanges additions and subtractions, each with its own measured cleanup.

B Table lookups and phase cleanup

B.1 Segmented banked lookups

Let the magnitude address have $n_{\text{ad}} = w - 1$ bits. Each 256-bit column is read in two segments: a wide first segment with few banks, whose selected bits stay live, and a narrow second segment with many banks. For a segment of n_{seg} payload bits with B banks, the address is split as $m = Bi_{\text{row}} + i_{\text{bank}}$; for $B = 3 \cdot 2^i$ it is first divided by three with the cell of Appendix A ($\lfloor m/3 \rfloor = 2^i i_{\text{row}} + i_{\text{bank}}$, and the row holds values with offsets $3i_{\text{bank}} + (m \bmod 3)$). The pruned unary loader of Appendix A XORs the addressed row’s public word, the concatenation of its B bank words, into the banks. A binary tree of controlled word swaps brings bank i_{bank} (and, for ternary counts, the remainder position) to bank zero; the static swap permutation is kept as classical metadata. Every bit of banks $1, \dots, B - 1$ is then measured in the X basis and released, its outcome recorded. For $L = \lceil 2^{n_{\text{ad}}} / B \rceil$ rows a segment costs

$$C_{\text{seg}} = C_{\text{load}}(L) + (B - 1) n_{\text{seg}} + 4n_{\text{ad}} \mathbf{1}_{B=3 \cdot 2^i}$$

Toffolis and peaks at $n_{\text{live}} + n_{\text{ad}} + n_{\text{sign}} + n_{\text{ret}} + Bn_{\text{seg}} + 2 \cdot \mathbf{1}_{B=3 \cdot 2^i} + \max(0, \lceil \log_2 L \rceil - 1)$ live qubits, with n_{live} other occupied wires, n_{sign} sign wires and n_{ret} retained bits of earlier segments. A dynamic program over the split point and the bank counts $B \in \{2^i : i \leq 8\} \cup \{3 \cdot 2^i : i \leq 7\}$, with every segment constrained to fit the design’s qubit budget, selects the plans; it is optimal within this family, not a lower bound for all lookups. The table below gives the plans at 1,457 qubits. Every other design is planned by the same program with its own budget and with any resident columns counted as occupied; Tables 10 and 11 give the resulting lookup costs per window.

Table 18. Lookup plans at 1,457 qubits. One column read costs 20,509 Toffolis for a 17-bit window and 39,628 for an 18-bit window. Peaks include the accumulator and all 18 address bits; the terminal selector’s peaks are those of the whole terminal gadget.

Width	Segment	Bits	Banks	Rows L	$C_{\text{load}}(L)$	Selector	Divider	Toffolis	Peak
17	low	228	4	16,384	16,382	684	0	17,066	1,455
17	high	28	24	2,731	2,735	644	64	3,443	1,443
18	low	228	4	32,768	32,766	684	0	33,450	1,456
18	high	28	24	5,462	5,466	644	68	6,178	1,444
initial	low	473	3	87,382	87,388	946	72	88,406	1,455
initial	high	39	24	10,923	10,928	897	72	11,897	1,442
terminal selec- tor, 18	one	9	128	1,024	1,022	1,143	0	2,165	1,436

After use, the selected 256-bit word of an ordinary read is itself released by X measurement and its outcome mask recorded. The initial lookup has an 18-bit address (the top 17 P bits and the top Q bit) and a 512-bit payload that becomes the accumulator; its segments cost 100,303 Toffolis and its phase correction $C_{\text{phase}}(18, 9) = 1,020$, for 101,323 in all.

Plans at other budgets. Tables 19 and 20 give the plans of every design, written as segments $n_{\text{seg}} \times B$ (n_{seg} payload bits read with B banks) from the lowest payload bits up. A read is planned with the other occupied wires counted, so a read made while a resident column is live has a different plan from one made without. In an ordinary window of a one-column design, column 0 is read without a resident column and columns 1 and 3 with one; in a two-column design, column 0 is read without and column 1 with one. In the terminal window the one-column designs make all three reads without a resident column live, and the two-column designs read column 0 without and column 1 with one. The phase corrections cost 508 (17-bit window), 764 (18-bit window) and 1,020 (terminal and initial lookup) at every budget, and the terminal selector uses 9-bit segments with 128 banks (2,165 Toffolis), except at 1,380 and 1,388 qubits, where it uses 96 banks (2,292).

Table 19. Plans of one 256-bit column read for every design. “All” covers every read of a design without resident columns; “without” and “with” mean without or with one resident column live during the read. For example, an ordinary 17-bit window at 1,758 qubits costs $16,906 + 2 \cdot 19,672 + 508 = 56,758$, and its 18-bit terminal window reads $3 \cdot 31,929 = 95,787$.

Peak	Read	17-bit window: plan	Toffolis	18-bit window: plan	Toffolis
1,380	all	256×3	22,427	256×3	44,277
1,388	all	256×3	22,427	$211 \times 4 + 6 \times 96 + 39 \times 16$	44,181
1,457	all	$228 \times 4 + 28 \times 24$	20,509	$228 \times 4 + 28 \times 24$	39,628
1,502	all	$239 \times 4 + 17 \times 32$	19,672	$239 \times 4 + 15 \times 48 + 2 \times 256$	38,011
1,510	all	$241 \times 4 + 15 \times 48$	19,243	$241 \times 4 + 15 \times 48$	36,997
1,713	without	256×4	17,150	$1 \times 256 + 194 \times 6 + 61 \times 16$	32,759
1,713	with	$228 \times 4 + 28 \times 24$	20,509	$228 \times 4 + 28 \times 24$	39,628
1,721	without	$196 \times 6 + 60 \times 16$	16,966	$195 \times 6 + 61 \times 16$	31,999
1,721	with	$230 \times 4 + 26 \times 24$	20,469	$230 \times 4 + 26 \times 24$	39,588
1,758	without	$202 \times 6 + 54 \times 16$	16,906	$202 \times 6 + 54 \times 16$	31,929
1,758	with	$239 \times 4 + 17 \times 32$	19,672	$239 \times 4 + 15 \times 48 + 2 \times 256$	38,011
1,766	without	$203 \times 6 + 53 \times 16$	16,896	$203 \times 6 + 53 \times 16$	31,919
1,766	with	$241 \times 4 + 15 \times 48$	19,243	$241 \times 4 + 15 \times 48$	36,997

Peak	Read	17-bit window: plan	Toffolis	18-bit window: plan	Toffolis
2,014	without	$244 \times 6 + 12 \times 64$	13,990	$244 \times 6 + 12 \times 96$	25,716
2,014	with	$202 \times 6 + 54 \times 16$	16,906	$202 \times 6 + 54 \times 16$	31,929
2,022	without	$246 \times 6 + 10 \times 64$	13,874	$246 \times 6 + 10 \times 96$	25,536
2,022	with	$203 \times 6 + 53 \times 16$	16,896	$203 \times 6 + 53 \times 16$	31,919

Table 20. Plans of the initial lookup (18-bit address, 512-bit payload) for every design.

Peak	Plan of the 512-bit read	Read	With phase correction
1,380	$448 \times 3 + 56 \times 16 + 8 \times 96$	109,145	110,165
1,388	$450 \times 3 + 6 \times 128 + 56 \times 16$	108,390	109,410
1,457	$473 \times 3 + 39 \times 24$	100,303	101,323
1,502	$488 \times 3 + 4 \times 192 + 20 \times 48$	97,119	98,139
1,510	$1 \times 384 + 491 \times 3 + 20 \times 48$	96,061	97,081
1,713	512×3	88,484	89,504
1,721	$422 \times 4 + 79 \times 16 + 11 \times 96$	88,219	89,239
1,758	$431 \times 4 + 81 \times 16$	84,424	85,444
1,766	$433 \times 4 + 79 \times 16$	84,400	85,420
2,014	$495 \times 4 + 17 \times 64$	72,184	73,204
2,022	$497 \times 4 + 15 \times 96$	71,257	72,277

B.2 Phase cleanup

An X measurement of a bit holding the basis value ϑ with outcome ϕ contributes the factor $2^{-1/2}(-1)^{\phi\vartheta}$. Because all bank contents are fixed functions of the address, the measured banks leave only an address-dependent sign.

Lemma B.1. For a read of table Ω with bank-selection permutation $\text{sel}_{i_{\text{bank}}}$ and recorded masks ϕ_i , the phase on magnitude address $m = Bi_{\text{row}} + i_{\text{bank}}$ is $(-1)^{F_{\Omega}(m)}$ with $F_{\Omega}(m) = \bigoplus_i \langle \phi_i, \Omega_{Bi_{\text{row}} + \text{sel}_{i_{\text{bank}}}(i)}^{\text{seg}} \rangle$ (padded entries are zero, unmeasured banks have mask zero); segments and reads combine by XOR. The phase can be corrected once, after all reads of a point addition, provided the address has not yet been unfolded, 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. The loader’s ANDs carry their own exact local corrections and the divider is uncomputed, so these are all the phase sources. Every intervening operation, including the address-controlled arithmetic, has block form $\sum_m |m\rangle\langle m| \otimes V_m$, so a phase diagonal in m commutes with it, and every transcript has the same normalization. $\square$

The correction is the Boolean phase circuit of Appendix A with cost $C_{\text{phase}}(n_{\text{ad}}, n_{\text{lo}}) = 2^{n_{\text{lo}}} + 2^{n_{\text{ad}} - n_{\text{lo}}} - 4$. Ordinary windows use $n_{\text{lo}} = 8$ for $n_{\text{ad}} = 16$ (508 Toffolis) and $n_{\text{ad}} = 17$ (764). Five reads with their cleanup cost $5 \cdot 20,509 + 508 = 103,053$ for a 17-bit window and $5 \cdot 39,628 + 764 = 198,904$ for an 18-bit window. The terminal window applies the same circuit to the 18 bits (m, η) : $C_{\text{phase}}(18, 9) = 1,020$. Omitting the unselected banks from F_{Ω} would leave every coordinate correct but the interference wrong.

B.3 Window widths

Let C_w be the cost of a width- w point addition (arithmetic plus the five reads with cleanup). For a remaining scalar length n_{rem} the least cost $C^*(n_{\text{rem}}) = \min_{3 \leq w \leq 18, w \leq n_{\text{rem}}} \{C^*(n_{\text{rem}} - w) + C_w\}$,

$C^*(0) = 0$, solved separately for the 240 remaining P bits and the 208 remaining Q bits, selects twelve 17-bit and two 18-bit windows on the P side and eight 17-bit and four 18-bit windows on the Q side: the fewest windows that cover the bits with widths of at most 18, with as many 17-bit windows as possible. Induction on the last window proves optimality within the stated width range and bank family. The terminal window then moves one 18-bit Q window to the end (Section 5.6).

C Width schedules

The two walk schedules are fixed public constants of the circuit specification. Each is a non-increasing sequence $W_0, \dots, W_T$ that decreases by at most one per step, so it is fixed by the first step i_W at which each width W occurs: $W_i = W$ for $i_W \leq i < i_{W-1}$, with $i_{W_T-1} = T + 1$. In the tables below, the row headed W and the column headed $-j$ give i_{W-j} . The 710-step schedule is used with $\kappa = 60$, $\delta = 27$, and the 720-step schedule with $\kappa = 62$, $\delta = 29$; both use $\delta' = 27$ for split cells.

Schedule $T = 710$. $W_T = 9$ and $\sum_{i < T} W_i = 100,143$.

W	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9
258	0	20	22	27	32	35	39	42	46	49
248	52	56	59	62	66	69	72	75	78	81
238	85	88	91	94	97	100	103	106	109	112
228	115	118	121	124	127	130	133	136	139	141
218	144	147	150	153	156	159	162	165	167	170
208	173	176	179	182	184	187	190	193	196	199
198	201	204	207	210	213	216	219	221	224	227
188	230	233	235	238	241	244	246	249	252	255
178	257	260	263	266	268	271	274	277	279	282
168	285	288	290	293	296	299	302	304	307	310
158	312	315	318	321	323	326	329	331	334	337
148	340	342	345	348	351	353	356	359	361	364
138	367	369	372	375	378	380	383	386	388	391
128	393	396	399	402	405	407	410	413	415	418
118	421	424	426	429	432	434	437	440	442	445
108	448	451	453	456	459	461	464	466	469	472
98	474	477	480	482	485	488	490	493	496	499
88	501	504	506	509	512	514	517	520	522	525
78	527	530	533	536	538	541	543	546	549	551
68	554	557	559	562	564	567	570	572	575	578
58	580	583	586	588	591	593	596	599	602	604
48	607	610	612	615	617	620	623	625	628	630
38	633	636	638	641	644	647	649	652	654	657
28	660	663	665	667	670	673	675	678	680	683
18	686	688	691	694	696	698	701	704	706	709

Schedule $T = 720$. $W_T = 8$ and $\sum_{i < T} W_i = 101,750$.

W	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9
258	0	22	24	30	34	37	41	45	48	52
248	55	59	62	65	69	72	75	78	81	85

W	−0	−1	−2	−3	−4	−5	−6	−7	−8	−9
238	87	91	94	97	101	104	107	110	113	116
228	119	122	125	128	131	134	137	140	143	146
218	149	151	154	157	160	163	166	169	172	175
208	178	181	184	186	189	192	195	198	201	203
198	206	209	212	215	218	220	223	226	229	232
188	234	237	240	243	246	249	251	255	257	260
178	262	265	268	271	274	276	279	282	285	288
168	291	293	296	299	302	304	308	311	313	315
158	318	321	324	327	329	332	335	338	340	343
148	345	348	351	354	356	359	362	364	368	370
138	373	375	379	382	384	388	390	392	395	398
128	401	403	406	409	411	414	416	419	422	424
118	427	430	433	435	437	440	443	446	450	452
108	455	457	460	463	465	468	471	473	476	479
98	481	484	487	489	492	495	497	500	502	505
88	508	511	514	516	519	522	524	527	529	532
78	534	538	540	543	546	548	550	553	556	560
68	562	565	567	569	573	575	578	580	583	586
58	588	591	595	596	600	602	604	606	610	613
48	615	617	620	623	625	628	631	633	636	639
38	641	644	647	650	652	654	657	660	662	665
28	667	670	673	676	678	681	683	687	690	692
18	694	696	699	702	704	708	710	712	715	718
8	720									

D Proofs of the error bounds

D.1 The replay bound

We prove Theorem 8.1. Put $M = 2^n$, $\Lambda = 2^\kappa$, $\Lambda_\delta = 2^{n-\delta}$ and $\mu = M/2 - c$. Fix a valid history and draw one uniform $y \in \mathbb{F}_p$. At every ideal step there are fixed field coefficients $\alpha, \alpha_{\text{op}}$ with $y_S = \alpha y$ (target) and $y_R = \alpha_{\text{op}} y$ (operand). A register with a nonzero coefficient is uniform on $\mathbb{F}_p$; a zero coefficient is treated directly. The two registers need not be independent.

Value errors of a doubling cell. Let $I = \ell + \Upsilon_\beta(q)$ be the integer before the low-word carry is discarded and $Z_* = (2S + R_\beta - \Theta_\beta) \bmod p$ the ideal encoded result; $Z_* = I \bmod p$ because $M \equiv c$. The only negative correction is $1 - c$ (for $\beta = 1, q = 0$, where $\ell = 2S + R_\beta \geq c$), so $I \geq 0$; and checking the six cases gives $I \leq p + 3c - 1 < 2p$ since $p > 8c$. If $I \geq p$ then $Z_* < 3c$. If $I < p$, a value error needs a positive correction of at most $3c$ to cross a multiple of Λ , or the correction $1 - c$ to borrow across one. All value errors therefore lie in $Z_* \bmod \Lambda \in \mathcal{B} = [0, 3c) \cup [\Lambda - c + 1, \Lambda)$, a set of $4c - 1$ residues inside one Λ -word because $4c \leq \Lambda$. With the updated coefficient $\alpha_{\text{new}} = 2\alpha + (-1)^\beta \alpha_{\text{op}} \neq 0$, $E(\alpha_{\text{new}} y)$ is uniform and at most $(4c - 1)\lceil p/\Lambda \rceil$ values of y fall in this set. If $\alpha_{\text{new}} = 0$ the ideal result is μ , and $\mu \bmod \Lambda = \Lambda - c \notin \mathcal{B}$ since $\Lambda \mid M/2$: no value error.

Phase-only errors. On a value-correct input no low-word carry was discarded: the output is $I - i\Lambda$ with $i \in \{-1, 0, 1\}$ and the ideal is $I - i'p$ with $i' \in \{0, 1\}$, and $i\Lambda = i'p$ forces $i = i' = 0$ because p is odd. Then $Z_* + \Theta_\beta - R_\beta = 2S - qp$ and $q_1 = (1 - q_0)[Z_* + \Theta_\beta < R_\beta]$. The circuit compares the top δ bits of Z and R_β , and $|\Theta_\beta| \leq c$, so a disagreement requires $|Z_* - R_\beta| \leq \Lambda_\delta + c$.

Modulo p , $Z_* - R_\beta \equiv 2ys$ for $\beta = 0$ and $2ys - (2c - 1)$ for $\beta = 1$; if $\alpha \neq 0$ this is uniform and at most $2\Lambda_\delta + 2c + 1$ values of y qualify. If $\alpha = 0$, $S = \mu$ identically: for $\beta = 0$ every value-correct case has $q = 1$, so both predicates vanish; for $\beta = 1$ the quotient is 0 or 1, and for $q = 0$, $Z_* - R_\beta = M - 3c + 1 > 0$, so both comparisons are false. Hence one cell fails on at most a fraction $[(4c - 1)\lceil p/2^\kappa \rceil + 2^{n-\delta+1} + 2c + 1]/p$ of y , counting a wrong phase predicate even when the measurement outcome would hide it.

Halving cells. Call a forward input good if it is value-correct and $q_1 = \widehat{q}_1$. The halving cell receives (R, Z_*, β) , which has a unique ideal preimage (R, S, β) under the doubling map. If that preimage is good, the parity gives the true q_0 , the same truncated comparison on the same pair gives $\widehat{q}_1 = q_1$, subtracting $\Upsilon_\beta(q)$ restores ℓ without a borrow, and the inverse integer stage returns S with $[I' < 0] = q_1$, so its measured phase is trivial. The halving cell’s bad inputs are therefore images of the doubling cell’s bad inputs on the family $(\alpha_{\text{op}}, \alpha)$ with $\alpha = (\alpha_{\text{new}} - (-1)^\beta \alpha_{\text{op}})/2$, counted by the same bound.

Split repairs. In a split cell the repaired phase is $(-1)^{\phi_{\text{sp}}[\text{top-27 comparison}]}$ instead of $(-1)^{\phi_{\text{sp}}\gamma_{\text{sp}}}$. Put $\Lambda' = 2^{b_{\text{sp}}}$ and $\Lambda_{\delta'} = 2^{b_{\text{sp}}-\delta'}$, which is 2^{165} for $b_{\text{sp}} = 192$ and 2^{101} for $b_{\text{sp}} = 128$. In a split doubling cell write $(2S)_{\text{lo}} = 2S \bmod \Lambda'$ and $R_{\text{lo}} = R_\beta \bmod \Lambda'$; a missed carry requires the written low word Z_{lo} to satisfy $Z_{\text{lo}} < R_{\text{lo}}$ with $\lfloor Z_{\text{lo}}/\Lambda_{\delta'} \rfloor = \lfloor R_{\text{lo}}/\Lambda_{\delta'} \rfloor$, hence $0 < \Lambda' - (2S)_{\text{lo}} = R_{\text{lo}} - Z_{\text{lo}} < \Lambda_{\delta'}$, so $(2S)_{\text{lo}} \in (\Lambda' - \Lambda_{\delta'}, \Lambda')$. If the target coefficient α is nonzero, S is uniform and at most $(\Lambda_{\delta'}/2)\lceil p/(\Lambda'/2) \rceil$ values of y qualify, a fraction below $2^{1-\delta'}$. In a split halving cell a missed borrow forces the raw forward low word into $[0, \Lambda_{\delta'})$ modulo Λ' ; since the correction $\Upsilon_\beta(q)$ lies in $[1 - c, 3c]$, the ideal doubling output Z_* then lies, modulo Λ' , in a circular interval of length at most $\Lambda_{\delta'} + 4c$, and if its coefficient is nonzero at most $(\Lambda_{\delta'} + 4c)\lceil p/\Lambda' \rceil$ values of y qualify, again below $2^{1-\delta'}$. The argument uses only that Λ' divides 2^{256} and that $b_{\text{sp}} - \delta' \geq \kappa$ (101 and 165 both exceed 62), so that the short correction leaves the compared bits alone; both fractions are in fact about $2^{-\delta'}$ at either boundary, half the charge. Neither bound assumes independence of the two registers, and a uniform *operand* alone would not suffice: a target fixed at $E(0)$ has $2S \bmod \Lambda' = \Lambda' - 2c$, for which a uniform operand usually produces a carry that the shortened comparison misses. The nonzero-coefficient hypothesis holds at every split cell of both schedules. A walk register is congruent to zero modulo p only for the initial $\mathbf{v}_0 = p$ before its first update: every updated value is odd, and either has magnitude below p or has the form $p + x/2^i$ with $0 < x/2^i < p$. In multiplication the target of every doubling cell is an already updated walk value, so $\alpha \neq 0$. In division the hypothesis concerns the halving cell’s input target, which is $E(0)$ only at step 1 (counting from 0); that step is never split, since 1,049 live wires plus 257 scratch wires is 1,306, below every kernel budget used (at least 1,380). The first split cells are steps 467 ($T = 710$) and 447 ($T = 720$) at a kernel budget of 1,457, later at 1,465, and steps 180 ($T = 710$) and 196 ($T = 720$) with the balanced split at 1,380 and 1,388. The charge is $2^{1-\delta'}$ once per split cell.

Interfaces and negations. For canonical $\zeta < M/2$ the encoder is exact exactly when $\zeta \bmod \Lambda \geq c$, and for $\zeta \geq M/2$ always; its bad set has $c2^{n-\kappa-1}$ elements, and the decoder’s has the same size. The live boundary coefficients y and xy (or y/x) are uniform, so the two interfaces contribute $c2^{n-\kappa}/p$. A known encoded zero is cleared as the constant μ , never decoded. Each conditional negation subtracts $2c - 1$ in the low κ bits and is charged $2c/2^\kappa$ for a lost borrow and $2c/p$ for a non-canonical result.

Composition. Union-bound the first possible deviation over the T cell events, the split repairs, the two interfaces and the two negations on the unconditioned ideal trajectory. No independence is used, and conditioning on earlier success, which would destroy uniformity, is unnecessary. This proves Theorem 8.1. $\square$

D.2 The outer arithmetic

Addition. Over ordered canonical pairs (x, y) , the adder of Section 4.1 gives a wrong value, a non-canonical result or a wrong phase on at most

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

pairs: sums in $[p, 2^n)$, which have no carry and stay unreduced; overflows of the low-word correction; and missed carry phases, where the top 32 bits of the result and of x agree.

Direct subtraction ($y \leftarrow y - x$). If $\gamma = 0$ the result is $y - x \leq \bar{x}$ and the truncated test $[\bar{x}_{\text{top}} < y_{\text{top}}]$ is correctly false. If $\gamma = 1$ the result is $y - x + p$ and $\bar{x} - (y - x + p) = c - 1 - y$, so the test can fail only if $y < c + 2^{n-\delta_o}$. A lost borrow of the low-65-bit subtraction of c needs the low word to be below c . Hence

$$N_{\text{sub}} \leq p(c + 2^{n-\delta_o}) + p c (2^{n-\kappa_o} - 1).$$

Square subtraction. Fix the source x ; the target is uniform through the mask. A local slice has fewer than $2^{n-n_{\text{pad}}}$ bad full-word targets (a carry escaping the padding). A wrapping slice has at most $c2^{n-\kappa_o}$ bad targets from its correction and at most $2^{n-\delta_o} + c$ from its phase comparison. The schedule has 16 local slices (five in each of the three stages plus the 97-bit slice of cH) and 9 wrapping slices (two in the low stage, six in the high stage, one in the sum stage), and no complement sites. The exact integer squares and the exact preparation of cH add no exceptions. Expressing the counts as fractions of 2^n and dividing by p ,

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

This needs no uniformity after earlier slices: for fixed x , every surviving prefix applies a fixed residue translation to the masked target, so distinct masks remain distinct and a bad set $\mathcal{S}$ at any stage has at most $|\mathcal{S}|$ preimages among the p masks.

D.3 Transfer from field inputs to point pairs

Put $D_0 = (r - 1)(r - K)$; each pair (accumulator, table point) has probability at most $1/D_0$. *Denominators:* for a nonzero $\hat{u} = x_1 - x_2$ and a fixed second point, the first point has x-coordinate $x_2 + \hat{u}$, shared by at most two points, so a bad denominator set of fraction q_{walk} has at most $2(r - 1)(p - 1)q_{\text{walk}}$ preimages and probability at most $D_{\text{walk}}q_{\text{walk}}$. *Field pairs:* with $A = (u + \hat{u}, v + \hat{v})$, $U = (u, v)$, $\hat{u} \neq 0$, subtracting the curve equations gives $2\hat{v}v + \hat{v}^2 = 3\hat{u}u^2 + 3\hat{u}^2u + \hat{u}^3$; for $\hat{v} \neq 0$ this fixes v as a quadratic in u and substitution gives a quartic with leading coefficient $(3\hat{u}/2\hat{v})^2 \neq 0$; for $\hat{v} = 0$ it is a quadratic in u with two v values each. Either way at most four preimages, so a bad pair set of fraction q_{pair} has probability at most $D_{\text{pair}}q_{\text{pair}}$. The division receives $(x_1 - x_2, y_1 - y_2)$ directly; for the multiplication apply the invertible change $(A, U) \mapsto (U, -(A + U))$ and the bijection $(\hat{u}, \lambda) \mapsto (\hat{u}, \hat{u}\lambda)$ for $\hat{u} \neq 0$ first.

Outer operations. A fixed x-coordinate belongs to at most two points and a fixed y-coordinate to at most three, so the two x-column subtractions have multiplicity 4 each and the two y-column subtractions 9 each, 26 in all: $p_{\text{sub}} = 26N_{\text{sub}}/D_0$. For the additions, fixing both input words of the first or third addition fixes the mask and an x-difference, leaving at most $4p$ point and mask preimages; the middle addition has at most $2(r - 1)$. With each point and mask triple of probability at most $1/(pD_0)$, $p_{\text{add}} \leq (8p + 2(r - 1))N_{\text{add}}/(pD_0)$. After the mask is folded into the table, the fused addition’s operand $(3u + \rho) \bmod p$ is uniform for every (A', U) , so it has the first addition’s preimage bound; we keep the whole three-addition charge.

D.4 Why the success theorem applies

Theorem 8.3 needs, for every scalar label, a normalized output block that equals the ideal block outside a bad event. We defer the scalar Fourier transforms in the analysis and keep the labels in the computational basis; the address folds are undone on every path and work bits never control a change of a label. Between measurements every operation is a computational-basis permutation or a diagonal sign, with classical feed-forward, so for a fixed label, randomization and earlier outcomes the work state is one basis vector times a phase. An X measurement of a work bit with value ϑ then gives each outcome ϕ with probability $1/2$ and the phase $(-1)^{\phi\vartheta}$, on good and bad paths alike. Skipped conditional measurements are padded with unused fair bits to a fixed public length J , so every padded transcript has probability 2^{-J} for every label. Every release by Z measurement and reset is purified as $|\vartheta\rangle_{\text{wk}}|0\rangle_{\text{env}} \mapsto |0\rangle_{\text{wk}}|\vartheta\rangle_{\text{env}}$ (work and environment registers) with a fresh environment bit, and nothing is conditioned on that record, so each block stays normalized even when a released value is nonzero. On the good domain all environment bits are zero and exact phase repairs give $|\Phi'_{\omega,\mathbf{k}}\rangle = |\Phi_{\mathbf{k}}\rangle \otimes |0\rangle_{\text{env}}$ for every realization ω (randomization and padded transcript) outside the bad event, up to an output-only relabeling.

The terminal window fits this form. Its Z measurement of the output x-coordinate is purified like any reset; the feed-forward of x_3 is a control from the environment record; its fallback branches execute the same public sequence of gates and measurements; and by Lemma 5.1, on the good domain, the record of x_3 and the bit η_{out} are an output-only isometry of the ideal final point, which commutes with the success projector on the scalar registers.

A bound linear in the error probability cannot come from one fixed circuit, in which a bad subspace of weight ϵ can carry amplitude $\sqrt{\epsilon}$. Theorem 8.3 obtains it by averaging the normalized blocks over the randomization and the transcript before applying Π .

E Gate accounting and certificate arithmetic

E.1 Gate accounting

Primitives. Modular addition or direct subtraction, including a constant addition: $256+64+31 = 351$. Replay cell: $n + \kappa + \delta + 1$, that is 344 ($\kappa = 60, \delta = 27$) or 348 ($\kappa = 62, \delta = 29$); split repair $+27$. Interface or conditional negation: $\kappa - 1$.

Walk passes. $C_{\text{rec}} = 257 + \sum_{i < T} (W_i - 1)$: $257 + 100,143 - 710 = 99,690$ ($T = 710$) and $257 + 101,750 - 720 = 101,287$ ($T = 720$). The eraser costs the same.

Kernels. $2 \cdot 99,690 + 710 \cdot 344 + 4 \cdot 59 = 443,856$ and $2 \cdot 101,287 + 720 \cdot 348 + 4 \cdot 61 = 453,378$ without splits; add $27 n_{\text{split}}$: $443,856 + 242 \cdot 27 = 450,390$, $443,856 + 528 \cdot 27 = 458,112$; $453,378 + 241 \cdot 27 = 459,885$, $453,378 + 273 \cdot 27 = 460,749$, $453,378 + 523 \cdot 27 = 467,499$. A terminal division with measured history omits one walk pass: 344,166, 350,700; 352,091, 358,598, 359,462.

Square subtraction. Integer squares: triangular $\frac{1}{2}n_{\text{sq}}(n_{\text{sq}} + 3) - 1$ ($n_{\text{sq}} \leq 40$, and 8,383 and 8,513 for 128 and 129 bits); recursive $C_{\text{sq}}(128) = 6,654$, $C_{\text{sq}}(129) = 6,774$. Slices: $2^{128}z$ adds 1,018 (256-bit z) or 1,028 (258-bit z); $-z$ 351; $-cz$ 1,703 plus 96 for the cH slice; cH word round trip 2,144 (twenty additions) or $2 \cdot 185 = 370$ (clean-target). Stages $14,677 + 16,495 + 14,832 = 46,004$. With measured scratch in the terminal window: $46,004 - 20,082 - 185 = 25,737$.

Outer arithmetic. Ordinary window: $6 \cdot 351 + 46,004 = 48,110$; with one resident column $10 \cdot 351 + 46,004 = 49,514$; with two $11 \cdot 351 + 46,004 = 49,865$. Terminal window: $5 \cdot 351 + 25,737 = 27,492$, or $9 \cdot 351 + 25,737 = 28,896$ with a resident column.

Ordinary point addition at 1,457 qubits. $2 \cdot 450,390 + 48,110 = 948,890$; with lookups $948,890 + 103,053 = 1,051,943$ and $948,890 + 198,904 = 1,147,794$. With the 720-step kernels $2 \cdot 460,749 + 48,110 = 969,608$, giving 1,072,661 and 1,168,512.

Terminal point addition at 1,457 qubits (18-bit, 710-step). $4 \cdot 39,628 + 450,390 + 27,492 + 2,165 + 510 + 1,020 = 640,089$. With the 720-step division, 650,448.

Other designs. Every ordinary window is two kernel calls plus the outer arithmetic plus the lookups, and every terminal window is the terminal division plus the terminal outer arithmetic, the lookups and the gadget, with the values of Tables 10 and 11.

Runs. $C_{\text{init}} + 20C_{17} + 5C_{18} + C_{\text{term}}$, for example $101,323 + 20 \cdot 1,051,943 + 5 \cdot 1,147,794 + 640,089 = 27,519,242$ and $73,204 + 20 \cdot 968,981 + 5 \cdot 995,986 + 434,402 = 24,867,156$.

E.2 Certificate arithmetic

All quantities below were computed in exact rational arithmetic; decimals are shown rounded to nearest. Section 8 prints some of the same quantities rounded up, as upper bounds.

Table 21. Certificate terms for one kernel call, for the two walk schedules; Table 13 lists ϵ_{call} for every kernel used.

Quantity	710-step kernel	720-step kernel
Cell term per cell	2.980233×10^{-8}	7.450581×10^{-9}
Split term per split cell	1.490116×10^{-8}	1.490116×10^{-8}
N_s, n_{fail}	$10^8, 10,769$	$10^8, 2,138$
q_{walk}^+	1.2263×10^{-4}	2.860×10^{-5}
ϵ_{call} , no split cells	3.299731×10^{-4}	7.867630×10^{-5}
ϵ_{call} at 1,457 qubits (242 / 273 split cells)	3.443974×10^{-4}	9.494837×10^{-5}

Outer terms per window: $p_{\text{add}} = 1.746230 \times 10^{-9}$, $p_{\text{sub}} = 9.080396 \times 10^{-9}$, $p_{\text{sq}} = 6.868504 \times 10^{-9}$, $[4 + 32(\tau + 1)]/(r - K) = 5.93 \times 10^{-67}$; ordinary window $\epsilon_{\text{out}} = 1.769513 \times 10^{-8}$, terminal window $\epsilon_{\text{out}} - \frac{9}{26}p_{\text{sub}} = 1.455192 \times 10^{-8}$. Resident columns: $\epsilon_{u\tau} = 2.095476 \times 10^{-9}$ and $\epsilon_{v\tau} = 1.11 \times 10^{-67}$. The redundant term $K/(r - K)$ is 2.26×10^{-72} .

With $P_0 \geq 2\bar{g} - 1 - 6/r = 0.80564666716\dots$, the 710-step design at 1,457 qubits has $\epsilon = 0.01756472612$ and $[(1 - \epsilon)\sqrt{P_0} - \epsilon]^2 = 0.7469242\dots$, reported as 0.746924; the 720-step design at 1,457 qubits has $\epsilon = 0.00484282375$ and $0.78923428\dots$, reported as 0.789234. Toffolis per certified success, using the unrounded bounds and rounding up: 36,843,419 and 35,537,675. Table 14 gives every design. With the classical anchor $64(r - 1)/(r\pi^4)$ instead, the 710-step design at 1,457 qubits would have only 0.606478.

F Notation

Each symbol below has one meaning throughout the paper. Indices i, i', j used only inside one formula, sum, loop or proof step, the integration and summation variables u, u' , and wire names set in typewriter type (**a, u, cy, . . .**, Appendix A) are local to the place where they appear, as are the variables of the pseudocode listings. For a register or integer x , $x[i]$ is bit i and $x_{[i,j]}$ the slice of bits i to $j - 1$.

Symbol	Meaning
n	Field bit length, $n = 256$

Symbol	Meaning
c	Field constant, $c = 2^{32} + 977$; $\sigma_i \in \{\pm 1\}$ its signed digits at offsets i
p	Field prime, $p = 2^n - c$
r	Prime order of the curve group
P, Q, d	Generator, public point and discrete logarithm, $Q = [d]P$
k_P, k_Q	Scalar register values on the P side (257 bits) and Q side (209 bits)
$\mathbf{k}$	Scalar label (k_P, k_Q) in Theorem 8.3
n_{ctl}	Bit length of a scalar register in the semiclassical Fourier transform
$\mathcal{O}$	Point at infinity (group identity)
w	Window width; $N = 2^w$
$e; e_{\min}, e_{\max}$	Least scalar bit of a window; ends of a bit interval tiled by windows
b	Random scale, uniform in $\mathbb{Z}_r^*$
$\mathcal{Z}$	Random translation point
K	Number of excluded translations, $K = 2^{18}$
Γ	Weighted base of a window, $[b2^e]P$ or $[b2^e]Q$
$\mathbf{a}$	Window address (w bits); $\tilde{U}_{\mathbf{a}}$ the signed table entry
m	Folded magnitude address ($w - 1$ bits); $U_m = (u_m, v_m)$ the unsigned table point, also written $U = (u, v)$
s	Address sign bit
A, A'	Accumulator point and its sign-conjugated form $(-1)^s A = (x_1, y_1)$
$A_{\text{new}}, A'_{\text{new}}$	Result $A + (-1)^s U_m$ and unsigned sum $A' + U_m = (x_3, y_3)$
$\hat{u}, \hat{v}$	Coordinate differences of two points (Appendix D.3)
ρ, ρ'	Random mask in $\mathbb{F}_p$; masked column value $(3u + \rho) \bmod p$
τ	Coordinate encoding offset $(c - 1)/2$; $\mathcal{E}(\zeta) = \zeta + \tau$
X, Y	Encoded x-coordinate and y-coordinate (slope) registers
x, y	Field elements: denominator and numerator of a kernel, or coordinates
ζ	Generic field element argument of the encodings $\mathcal{E}$ and E
λ	Slope; $\lambda_0(m), \lambda_1(m)$ its two candidates in the terminal window, $\lambda_\eta(m)$ the present one
D_0	Pair-density denominator $(r - 1)(r - K)$
T	Walk length (710 or 720 steps)
$W_i; W$	Walk register width at step i ; a width value, first used at step i_W
$\mathfrak{u}, \mathfrak{v}$	Walk values (registers $\mathfrak{U}, \mathfrak{V}$)
$\mathfrak{t}, \mathfrak{w}, \mathfrak{n}$	Walk target, operand and sum $\mathfrak{t} \pm \mathfrak{w}$ (registers $\mathfrak{T}_i, \mathfrak{D}_i$)
h_i	History bit of walk step i
o	Orientation bit $1 - (x \bmod 2)$
$s_{\mathfrak{u}}, s_{\mathfrak{v}}$	Sign bits of the final walk values
$\mathcal{L}_i, \mathcal{L}$	Linear maps of one walk step and of the whole walk
M	2^n
E, μ	Replay encoding $E(\zeta) = (\zeta + \mu) \bmod p$ and its offset $\mu = M/2 - c$
R, S	Replay operand and target registers, holding $E(y_R)$ and $E(y_S)$
y_R, y_S	Field values of the replay operand and target
β	Replay sign control (0 or 1); R_β, Θ_β the source word and constant it selects
$q; q_0, q_1$	Quotient of a replay cell and its two bits; $\hat{q}_1$ the truncated estimate of q_1
ℓ, ℓ'	Low word of the forward and inverse integer stages of a replay cell
$\Upsilon_\beta(q)$	Short correction $qc - \Theta_\beta$; $\Upsilon_{\beta q}$ its low κ bits
$Z; Z_*$	Word written by a replay cell; its ideal value
I, I'	Integers of the forward and inverse cell stages (Appendices A and D.1)
$Z_{\text{lo}}, R_{\text{lo}}, (2S)_{\text{lo}}, D_{\text{lo}}$	Low b_{sp} bits of the written sum, the operand R_β , the doubled target, and the difference word $I' \bmod 2M$ of a split cell
κ, δ, δ'	Replay correction width, comparison width, split-repair width

Symbol	Meaning
κ_o, δ_o	Correction and comparison widths of the outer arithmetic
$\gamma; \gamma_{\text{sp}}$	Carry-out (borrow, for subtraction) of a modular adder; carry into bit b_{sp} of a split cell
b_{sp}	Split boundary of a split cell (128 or 192)
ϕ	X-measurement outcome ($\phi_1, \phi_{\text{sp}}, \phi_\gamma$ for named wires, ϕ_λ the slope mask, ϕ_h the history mask, ϕ_i lookup masks)
ϑ	Basis value of a measured bit
n_{split}	Split cells per kernel call
n_{ker}	Kernel budget: the qubit budget that sets the split cells of a kernel call
$n_+, n_{\text{sl}}, n_{\text{pad}}, n_z$	Adder width; slice width; slice padding (32); square word length
$i_{\text{src}}, i_{\text{dst}}$	Source and destination offsets of a slice
$n_{\text{sq}}; n_0, n_1$	Input width of the integer squarer; its two recursion halves $x = x' + 2^{n_0}x''$
z	Integer square word
C	Cross-term register of the recursive squarer
$H; H', z_{\text{top}}$	High overflow word of the cz slice; its part from the top ten bits of z , and those bits (Section 4.4)
$x_{>i}, x_{<i}, x_{\geq i}$	$\lfloor x/2^{i+1} \rfloor, x \bmod 2^i, \lfloor x/2^i \rfloor; z^{(i)}$ the partial square
$C_{\text{rec}}, C_{\text{cell}}, C_{\text{kernel}}$	Toffoli cost of a walk pass, a replay cell, a kernel call
$C_{\text{sq}}, C_{\text{phase}}, C_{\text{seg}}, C_{\text{load}}, C_{\text{visit}}$	Toffoli cost of an integer square, phase circuit, lookup segment, row loader, loader recursion
C_w, C^*	Cost of a width- w point addition; least cost for a remaining length n_{rem}
$C_{\text{init}}, C_{17}, C_{18}, C_{\text{term}}$	Toffoli cost of the initial lookup, of ordinary 17- and 18-bit point additions and of the terminal point addition
C_k	Worst-case classical cost of the rule $\mathcal{R}_k$
$\eta, \eta_0(m), \eta_{\text{out}}$	Terminal selection bit; selector bit; output sign bit
$\iota(m)$	Selector bit position
$x_3, y_\star$	Measured terminal x-coordinate; smaller square root of $x_3^3 + 7$
$F, F_{\text{prefix}}, F_{\text{sel}}, F_\Omega, G_{\text{sq}}$	Phase tables
Ω	Public table read by a lookup, with row words Ω_u
J	Number of X measurements
B, L	Banks of a lookup segment; rows ($n_L = \lceil \log_2 L \rceil$)
$n_{\text{ad}}, n_{\text{lo}}, n_{\text{seg}}$	Magnitude address bits; low bits of a phase circuit; payload bits of a segment
$n_{\text{live}}, n_{\text{sign}}, n_{\text{ret}}$	Other live wires, sign wires, retained bits (lookup peak)
$i_{\text{row}}, i_{\text{bank}}$	Row and bank parts of a magnitude address; sel the bank permutation
$\text{Col}_0, \text{Col}_1$	Resident column registers holding $u_m + \tau$ and $v_m + \tau$ (Section 6.4)
$\mathbf{h}(x)$	History and endpoint bits of the walk on x ; F_{hist} its phase term (Section 6.5)
$q_{\text{rep}}, q_{\text{walk}}, q_{\text{pair}}$	Replay failure probability; walk failure fraction (q_{walk}^+ its bound); bad pair fraction
N_s, n_{fail}	Samples and observed failures of the walk certificate; $\bar{n}_{\text{fail}}$ the mean, n_{thr} a threshold (Lemma 8.2)
ξ, Ξ	Confidence exponent and $\Xi = \lceil 7\xi/10 \rceil$
$D_{\text{walk}}, D_{\text{pair}}$	Transfer factors to point pairs
ϵ	Total error budget of a design
$\epsilon_{u\tau}, \epsilon_{v\tau}; \epsilon_{\text{res}}, \epsilon'_{\text{res}}$	Extra error terms of the resident columns; their sums per ordinary and terminal window
$\epsilon_{\text{call}}; \epsilon_{710}, \epsilon_{720}$	Per-kernel-call error term; its value for each schedule
ϵ_{out}	Outer-arithmetic error term per window
$\epsilon_{\mathbf{k}}$	Bad-event probability for label $\mathbf{k}$ (Theorem 8.3)

Symbol	Meaning
$p_{\text{add}}, p_{\text{sub}}, p_{\text{sq}}$	Outer error bounds per window
$\psi, \varrho_{\mathbf{k}}, \Phi_{\mathbf{k}}, \Phi'_{\omega, \mathbf{k}}, \Delta_{\mathbf{k}}$	Ideal state, its amplitudes, ideal and realized blocks, block deviation (Theorem 8.3)
ω	Random realization (randomization and measurement record)
Π, P_0	Success projector; ideal success probability ($P_0^{(k)}$ under $\mathcal{R}_k$)
M_P, M_Q, M_{reg}	$2^{257}, 2^{209}$, and the size of either register
$a; a', a_0$	Fourier label in $\mathbb{Z}_r$; candidate labels
j, j'	Fourier measurement outcomes of the two registers
ν, ν_0, n_ν	Phase numerator $2^{48}ad \bmod r$; start and length of its candidate interval
χ_a	Group Fourier state
θ, t, n_θ	True phase; fractional part of $M_{\text{reg}}\theta$; integer part
$f, g, \bar{g}, g_k, \bar{g}_k$	Phase-estimation functions of Theorems 8.4 and 9.2; $\bar{g}_r, \bar{g}_{k,r}$ their averages over the r -point grid
scr	Function of the slope held in the measured square scratch (Section 5.4)
$x_{\text{lo}}, x_{\text{hi}}$	Low and high 128-bit halves of the square’s source x (Section 4.2)
t_P, t_Q, G_P, G_Q	Phase offsets and good events of the two registers
k	Width of the recovery rule $\mathcal{R}_k$
$\mathcal{C}(j), n_a, N_k$	Candidate labels, their number, and the ν -interval bound
$\mathcal{T}; i_0, i_1$	Baby-step table size; baby-step and giant-step indices
$\mathcal{Y}_a$	Giant-step starting point for label a
Lip_k	Lipschitz constant of g_k
Ψ, ϖ	Auxiliary function of the sine integral and its argument
$\Lambda, \Lambda', \Lambda_\delta, \Lambda_{\delta'}$	$2^\kappa, 2^{b_{\text{sp}}}, 2^{n-\delta}, 2^{b_{\text{sp}}-\delta'}$ (Appendix D.1)
$\alpha; \alpha_{\text{op}}, \alpha_{\text{new}}$	Coefficient of the replay target, of the operand, and the updated target coefficient
$\mathcal{B}, \mathcal{S}$	Bad residue band (Appendix D.1); bad target set (Appendix D.2)
$N_{\text{add}}, N_{\text{sub}}$	Bad-pair counts of the outer adder and subtractor
$\mathcal{R}_k$	Recovery rule of width k

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.
- [2] A. Schrottenloher. *Optimized Point Addition Circuits for Elliptic Curve Discrete Logarithms*. arXiv:2606.02235v1, June 2026.
- [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).
- [5] A. Karatsuba and Yu. Ofman. *Multiplication of multidigit numbers on automata*. Soviet Physics Doklady 7, 595 (1963).
- [6] P. L. Montgomery. *Speeding the Pollard and elliptic curve methods of factorization*. Mathematics of Computation 48, 243 (1987).
- [7] D. Shanks. *Class number, a theory of factorization, and genera*. In Proc. Sympos. Pure Math. 20, 415–440. American Mathematical Society (1971).
- [8] R. Cleve, A. Ekert, C. Macchiavello and M. Mosca. *Quantum algorithms revisited*. Proc. R. Soc. Lond. A 454, 339–354 (1998).

- [9] F. W. J. Olver et al. (eds.). *NIST Digital Library of Mathematical Functions*, Chapter 6: Exponential, Logarithmic, Sine, and Cosine Integrals. <https://dlmf.nist.gov/6>.
- [10] M. Roetteler, M. Naehrig, K. M. Svore and K. Lauter. *Quantum resource estimates for computing elliptic curve discrete logarithms*. In *Advances in Cryptology – ASIACRYPT 2017*, Lecture Notes in Computer Science 10625, 241–270. Springer (2017). arXiv:1706.06752.
- [11] T. Häner, S. Jaques, M. Naehrig, M. Roetteler and M. Soeken. *Improved quantum circuits for elliptic curve discrete logarithms*. In *Post-Quantum Cryptography – PQCrypto 2020*, Lecture Notes in Computer Science 12100, 425–444. Springer (2020). arXiv:2001.09580.
- [12] D. Litinski. *How to compute a 256-bit elliptic curve private key with only 50 million Toffoli gates*. arXiv:2306.08585, June 2023.