# Progressive Synthesis of a Reversible 5-Bit Elliptic-Curve Group-Law Oracle

@jackylee0424, @gnuchev, @runeape-sats, @jieyilong, @edi3on

**Abstract.** Shor's discrete-logarithm algorithm calls elliptic-curve point addition inside its main oracle. Reversible arithmetic for that oracle dominates published fault-tolerant resource estimates. Toffoli count and Toffoli depth price its non-Clifford work and critical path. The cost proxy $q\sqrt{TD_T}$ combines those quantities with logical width. Here, $q$ is the logical-qubit count, $T$ is the Toffoli count, and $D_T$ is Toffoli depth. We synthesize the evolving design of a verified 5-bit oracle over $\mathbb{F}_{31}$. The design moved from table-free arithmetic to a three-point scalar pebble. Later work refined cleanup order, live ranges, controlled-add timing, and an inverse-case witness. The current circuit uses 315 logical qubits and 4,724,217 Toffoli gates. Its Toffoli depth is 3,523,665, giving a score of 1,285,206,102.9293. The small instance exposes interactions between width, arithmetic, and scheduling under complete reversible cleanup.

## 1 Introduction

The cost of elliptic-curve discrete logarithms depends on reversible group arithmetic. Shor's algorithm supplies the polynomial-time quantum framework [1]. Its elliptic-curve realization repeatedly evaluates controlled point additions. Those additions require finite-field arithmetic, exception handling, and clean uncomputation.

Existing estimates place this arithmetic at the center of fault-tolerant resource accounting. Roetteler et al. quantified complete circuits for elliptic-curve discrete logarithms [2]. Häner et al. later reduced resources through improved reversible circuits [3]. Both works motivate studying the oracle below the algorithm level.

This work isolates a variable-base group-law oracle at a fully testable field size. The curve is $y^2 = x^3 + 7$ over $\mathbb{F}_{31}$. Its group has order 21. The oracle maps two scalars and two points to their linear combination. It preserves every input and clears every ancillary qubit [5].

The cost model rewards joint control of space, work, and serialized non-Clifford work. Its score is

$$S = q\sqrt{TD_T}. \tag{1}$$

Width enters linearly. Toffoli count and depth enter through their geometric mean. This form permits a width reduction to offset more arithmetic.

The current design reaches $S = 1{,}285{,}206{,}102.9293$ under the fixed oracle contract. It uses 315 logical qubits, 4,724,217 Toffoli gates, and depth 3,523,665. The result emerged through a public, strict-improvement research process. This paper presents that process as progressive synthesis rather than a ranking report.

**Contributions.** The authors contributed distinct phases of the frontier. `@jackylee0424` maintained the effort and established the table-free baseline. `@runeape-sats` introduced the width cut and several cleanup schedules. `@edi3on` reduced cleanup work by changing the recycled power slot. `@jieyilong` shortened live ranges through delayed scratch allocation. `@gnuchev` combined early scheduling with a predicate-specific field witness.

## 2 Related Work

Shor established polynomial-time quantum algorithms for discrete logarithms [1]. The algorithm prepares a periodic quantum state and extracts its period information. For elliptic curves, the period function evaluates scalar combinations of points. That evaluation turns the group law into an inner reversible computation.

Roetteler et al. connected this algorithm to concrete elliptic-curve resource estimates [2]. Their analysis accounted for logical qubits and quantum gates at cryptographic sizes. It showed how field inversion and point addition shape the total circuit. The present work studies those mechanisms in a smaller verified setting.

Häner et al. developed lower-resource circuits for elliptic-curve discrete logarithms [3]. Their work demonstrates that representation and arithmetic choices change full-algorithm costs. Our metric adds explicit pressure on Toffoli depth. It also exposes scratch lifetime through the factor $q$.

The eventual real-world target includes curves such as secp256k1. SEC 2 specifies that curve and its domain parameters [4]. The 5-bit curve here is not a resource estimate for secp256k1. It is an exact test bed for reversible design choices.

The contest repository defines the executable contract and ongoing research record [5]. This paper uses that record for metrics and lineage. It does not reproduce implementation artifacts or validation logs.

## 3 The Oracle and Its Cost Model

The target operation preserves four inputs and writes one output point. It is

$$|a\rangle|b\rangle|P\rangle|Q\rangle|0\rangle \longmapsto |a\rangle|b\rangle|P\rangle|Q\rangle |aP + bQ\rangle. \tag{2}$$

Both 5-bit scalars are interpreted modulo 21. The points are variable inputs rather than compile-time constants.

| Register | Width | Meaning |
|---|---|---|
| $a, b$ | 5 each | scalars modulo 21 |
| $p_x, p_y, p_{\text{inf}}$ | $5, 5, 1$ | input point $P$ |
| $q_x, q_y, q_{\text{inf}}$ | $5, 5, 1$ | input point $Q$ |
| $r_x, r_y, r_{\text{inf}}$ | $5, 5, 1$ | output point $R$ |

Table 1: Public registers for the oracle in Eq. (2).

Each point uses affine coordinates plus an infinity flag. Table 1 gives the public register layout. Each coordinate occupies five qubits. The flag distinguishes the point at infinity from finite coordinate encodings. The output begins in the all-zero state.

The arithmetic contract is reversible and in place. Its field facade provides addition, subtraction, multiplication, inversion, comparison, zero testing, and multiplexing. The scalar facade permits point doubling and controlled affine addition. Scratch may be reused only after it has been returned to zero.

The score in Eq. (1) measures a compiled logical circuit. The logical width $q$ includes public registers and live workspace. The count $T$ measures all Toffoli gates. The depth $D_T$ measures Toffoli layers along the critical path. Clifford gates and total operations remain reported diagnostics.

The frontier gate requires a strictly lower score and complete verification. A candidate with unchanged width may win through count or depth. A narrower candidate may win even when both Toffoli measures rise. This rule made space-time tradeoffs observable throughout the lineage.

Verification uses 9,024 deterministic Fiat–Shamir shots. The first check establishes oracle correctness. The second checks in-place $\mathbb{F}_{31}$ arithmetic composition. The third enforces the restricted scalar strategy interface. The remaining checks cover input preservation, phase cleanliness, and ancilla cleanup. All six conditions must hold for the score to enter the frontier [5].

## 4 Progressive Synthesis of the Frontier

The frontier evolved through one width cut and then intertwined arithmetic and scheduling refinements. Methodologically, the work separates into baseline, width, field arithmetic, and scalar scheduling. Chronologically, scheduling refinements preceded the final field specialization. Table 2 records representative frontier states in their actual order.

### 4.1 Table-Free Baseline

The baseline established an arithmetic-first oracle without field or point lookup tables. The field layer used reversible Boolean kernels over $\mathbb{F}_{31}$. Multiplication used controlled cyclic shifts for the modulus $2^5 - 1$. Inversion used a Fermat exponentiation chain.

The baseline also specialized repeated field operations. Multiplication by 3 became addition to a one-bit cyclic rotation. The first product term was copied without an add from zero. Reduced bits and the all-ones reduction flag were materialized once.

The first scalar strategy stored a full dynamic power chain. It computed $2P$, $4P$, $8P$, and $16P$. It applied the same process to $Q$. Each chain was reversed after its controlled additions.

### 4.2 Width Reduction

The width phase replaced four live powers with a three-point pebble. One scratch point first held $2P$ and was later reused for $16P$. The other two scratch points retained $4P$ and $8P$. The schedule accepted two extra doublings to enable reuse.

This exchange reduced width from 326 to 315 logical qubits. Toffoli count rose from 4,624,825 to 4,791,417. Toffoli depth rose from 3,477,469 to 3,579,733. The linear width gain still lowered the score by 2,793,686.3103.

### 4.3 In-Place Field Arithmetic

Field refinement specialized a value that served only as a predicate. The inverse-point branch needs to know whether two finite $y$ coordinates sum to zero. It does not need the sum after that test. A reversible witness can therefore replace a full modular addition.

The witness uses complementation in five-bit encodings modulo 31. For a nonzero field encoding, negation is its bitwise complement. The kernel writes

$$w = y_{\text{left}} \oplus y_{\text{right}} \oplus 11111_2. \tag{3}$$

The zero test on $w$ preserves the inverse-case decision.

The first witness reduced both Toffoli count and depth at width 315. It lowered $T$ by 10,206 and $D_T$ by 5,502. Writing witness bits in order $3, 0, 1, 2, 4$ then removed 160 depth layers. That permutation left every recorded gate count unchanged.

### 4.4 Scalar-Strategy Scheduling

Scalar scheduling reduced work and depth within the three-point budget. `@edi3on` kept $2P$ live and recycled the $4P$ slot for $16P$. Cleanup then recreated $4P$ from $2P$ before clearing $8P$. This change cut both Toffoli count and depth.

Cleanup-pebble ordering exposed further depth-only improvements. `@runeape-sats` converted the $16P$ slot into a valid $2P$ cleanup value. The temporary $16P \oplus 2P$ pattern was never used as a point. Later controlled additions were placed near each power's last use.

Live-range control shortened the critical path without changing work. `@jieyilong` delayed allocation of the third scratch point. The schedule then added $16P$, $8P$, $4P$, $2P$, and $P$. It cleaned power slots near their last use.

Early-staircase retiming completed the accepted scalar schedule. `@runeape-sats` moved low controlled additions

| Frontier phase | Main method | Contributors | $q$ | $T$ | $D_T$ | Score |
|---|---|---|---|---|---|---|
| Table-free baseline | Dynamic powers and reversible $\mathbb{F}_{31}$ kernels | `@jackylee0424` | 326 | 4,624,825 | 3,477,469 | 1,307,365,094.9206 |
| Width reduction | Three-point scalar pebble | `@runeape-sats` | 315 | 4,791,417 | 3,579,733 | 1,304,571,408.6103 |
| Scalar refinement | Recycled slot, cleanup order, and retiming | `@edi3on`, `@runeape-sats`, `@jieyilong` | 315 | 4,734,423 | 3,529,327 | 1,287,626,872.9591 |
| Field–schedule co-design | Complement witness and bit-order retiming | `@gnuchev` | 315 | 4,724,217 | 3,523,665 | 1,285,206,102.9293 |

Table 2: Era-level progression. Each row is a representative accepted frontier state.

toward power creation. `@gnuchev` retained that staircase beside the field witness. The final schedule keeps three 11-qubit scratch points.

## 5 The Leading Method in Depth

The leading method combines predicate-specific field arithmetic with an early scalar staircase. The field change removes work that the point adder never observes. The scalar change controls the lifetime and dependency order of point powers. Both retain compute–copy–uncompute cleanup.

### 5.1 Arithmetic Scheme

The arithmetic scheme implements field operations as reversible transformations. Addition and subtraction reduce their results modulo 31. Multiplication accumulates controlled cyclic shifts. Inversion applies a Fermat exponentiation chain. Comparison, zero testing, and multiplexing support exceptional group-law branches.

The inverse-case witness is narrower in purpose than modular addition. Equation (3) is used only for a zero predicate. Its bits can be written in an order chosen for downstream depth. The selected order is $3, 0, 1, 2, 4$.

The witness remains reversible because it is later uncomputed. The point adder first computes $w$ into zeroed scratch. It copies the zero-test decision into the branch logic. It then reverses the witness construction. No residual witness remains in the final state.

### 5.2 Scalar Strategy

The scalar strategy builds each multiple only while it supports additions or cleanup. Successive doublings provide $2P$, $4P$, $8P$, and $16P$. Controlled additions consume scalar bits as their powers become available. The same construction forms the contribution from $Q$.

Three scratch points form the live pebble budget. The first slot is cleared and reused for $8P$. The final power slot later becomes a valid $2P$ cleanup pebble. This state supports reversing the earlier doublings.

The transient XOR pattern is an output state, not a point value. During cleanup, a slot may briefly contain $16P \oplus 2P$. No doubling or controlled addition consumes that pattern. The next inverse operation recovers a valid cleanup point.

The schedule contains ten doubling calls and five controlled additions. It performs controlled additions in an early staircase. Low powers enter the accumulator soon after they exist. Cleanup then reverses every remaining power dependency.

### 5.3 Correctness Invariants

Correctness follows from four invariants maintained across every reversible segment. First, the public inputs $a$, $b$, $P$, and $Q$ never change. Second, every consumed point source is a valid affine-plus-flag encoding. Third, the output accumulates exactly the selected powers. Fourth, every scratch qubit returns to zero.

The infinity flag makes the point representation total for the group law. Finite points use their two affine coordinates. The flag represents the identity without assigning it finite coordinates. Point operations preserve this encoding convention.

The scalar invariant ties the binary schedule to the group expression. Each controlled addition contributes one selected power of its input point. The dynamic chain supplies all five scalar positions. Combining the two held products yields $aP + bQ$.

The witness invariant concerns only the inverse-point predicate. For its stated nonzero field inputs, complementation realizes modular negation. Thus $w = 0$ exactly when the two encoded $y$ values are inverses. Uncomputing $w$ restores its scratch register.

The validation conditions test the observable consequences of these invariants. Oracle correctness checks the final group element. Input preservation checks the public registers. Phase and ancilla checks detect reversible garbage. Field composition and scalar-interface checks constrain the implementation boundary.

### 5.4 Where the Budget Goes

The remaining budget is bounded by repeated point operations and their field kernels. The source papers identify inversion and multiplication as the main arithmetic targets. Each dynamic point power invokes those kernels through doubling. Each controlled affine addition adds another dependency chain.

The final arithmetic optimization affects a small but repeated decision path. It removes 10,206 Toffoli gates

| Metric | Final value |
|---|---|
| Logical qubits | 315 |
| Toffoli count | 4,724,217 |
| Toffoli depth | 3,523,665 |
| Clifford count | 14,182,788 |
| Total operations | 26,076,249 |
| $q\sqrt{TD_T}$ | 1,285,206,102.9293 |

Table 3: Final compiled circuit metrics.

compared with the preceding scheduled design. Its bit permutation removes another 160 Toffoli layers. Width stays fixed because the witness reuses existing scratch.

The scalar schedule now offers less depth-only headroom than earlier phases. Several refinements changed only $D_T$ while holding all counts fixed. Their reductions were 1,216, 2,282, 116, and 34 layers. The sequence shows diminishing gains from dependency-tail retiming.

## 6 Results

The final circuit improves the score through width, arithmetic, and critical-path control. Table 3 reports the complete recorded metrics. The circuit preserves the 315-qubit width reached in the second frontier phase. It then reduces both Toffoli measures relative to that width milestone.

The last frontier step isolates a pure depth improvement. Logical width, Toffoli count, Clifford count, and total operations were unchanged. Toffoli depth fell by 160 layers. The score fell by 29,178.5176 through the $\sqrt{D_T}$ factor.

The score decomposition now favors structural changes over small retimings. For proportional changes, width has full weight in Eq. (1). Count and depth each have half weight. A tenfold score cut from width alone requires a tenfold width cut. A cut from one Toffoli factor alone requires a hundredfold reduction.

The 9,024-shot gate found no failures for the final circuit. All six required checks passed. This result supports correctness under the defined deterministic test suite. It is not a proof for larger fields or another oracle contract.

## 7 Discussion and Future Work

The remaining headroom lies in field kernels, a smaller live set, or both. Depth-only scheduling still helps, but its latest gains are narrow. Inversion and multiplication repeat across doublings and additions. Reducing them can affect both $T$ and $D_T$.

A lower-width scalar strategy must account for recomputation costs. The first width cut saved 11 qubits but added two doublings. Equation (1) accepted that trade at this size. A further cut must preserve clean reversal and the restricted interface.

The 5-bit result does not directly extrapolate to cryptographic curves. Field registers, multiplication circuits, and inversion chains grow with the modulus. Their relative costs can change with arithmetic architecture. The secp256k1 domain is therefore a separate compilation target [4].

Several design principles do transfer beyond this instance. Predicate-specific computation avoids producing unused values. Pebbling makes width and recomputation explicit. Live-range ordering exposes depth without changing gate totals. Compute–copy–uncompute gives local cleanup obligations.

The contest format contributes a reproducible method for circuit synthesis. A fixed interface prevents improvements from changing the mathematical task. A scalar score makes space-time exchanges visible. Deterministic verification keeps each accepted frontier state comparable. The ongoing record can therefore support further collective refinement [5].

## Acknowledgments

We thank the ecdsa.fail community [6] for pioneering this effort; resource estimates and mitigations for 256-bit curves are surveyed in [7].

The contributions were assisted by GPT-5, GPT-5.5, and Claude Sonnet model families. The draft was auto-assembled from the contest record and contestant-derived papers. It has not been peer reviewed.

## References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, 1997. arXiv:quant-ph/9508027.

[2] M. Roetteler, M. Naehrig, K. M. Svore, and K. Lauter, "Quantum resource estimates for computing elliptic curve discrete logarithms," ASIACRYPT, 2017. arXiv:1706.06752.

[3] T. Häner, S. Jaques, M. Naehrig, M. Roetteler, and M. Soeken, "Improved quantum circuits for elliptic curve discrete logarithms," PQCrypto, 2020. arXiv:2001.09580.

[4] Certicom Research, "SEC 2: Recommended Elliptic Curve Domain Parameters," version 2, 2010.

[5] 5-bit Shor ECDLP Contest, "In-place $\mathbb{F}_{31}$ field arithmetic and scalar strategy oracle." github.com/ecdlp-contest/ecdlp-5-bit.

[6] The ECDSA.fail community (an Eigen Labs project), quantum circuit construction challenge against ECDSA. ecdsa.fail.

[7] R. Babbush, A. Zalcman, C. Gidney, M. Broughton, T. Khattar, H. Neven, T. Bergamaschi, J. Drake, and D. Boneh, "Securing elliptic curve cryptocurrencies against quantum vulnerabilities: resource estimates and mitigations," 2026. arXiv:2603.28846.