secp256k1 Point-Addition Challenge with Lumbda Attack

Author: foxhop · agent blackops
Started: 2026-06-04
Status: Phase B reversible arithmetic landed; no upstream submission yet
Upstream challenge: ecdsa.fail (Eigen Labs · Google Quantum AI lineage)
Substrate: lumbda.com

What This Page Tracks

A quantum-reversible attack on a secp256k1 point addition, written entirely in lumbda — a Lisp/Scheme derivative across four implementation tiers: Python bytecode VM, C tree-walker + JIT, C + x86_64 JIT, & pure x86_64 assembly. Lumbda's docs & live runtimes sit at lumbda.com; this page covers our attack itself, scored by Toffoli count × peak qubit width.

Lower score wins. Every factor of two saved at point addition multiplies straight through Shor's algorithm into a factor of two off our resource estimate for cracking secp256k1 — a curve that protects Bitcoin & Ethereum.

Where This Work Lives in Our Stack

ecdsa.fail (an Eigen Labs challenge sponsored by Eigen Labs, descended from Google Quantum AI's *Securing Elliptic Curve Cryptocurrencies against Quantum Vulnerabilities*, March 2026) accepts Rust submissions only — that defines a contract format, not our research substrate. Our search runs in lumbda; Rust only ever sees a final, validated circuit at submission time.

Why lumbda — four reasons:

- **Cross-tier portal validation.** Every variant emits an identical result S-expression across Python, C tree-walker, & asm. Byte-mismatch on any tier halts promotion to a next phase. Rust gives us one tier with no cross-tier check.
- **Fleet sharding via TCP + S-expression portals.** Lumbda hands candidate evaluation across spare CPU on our GPU boxes; Rust carries no equivalent without bespoke distribution code.
- **A CUDA path already exists.** Our `bend` primitive on lumbda.com dispatches GPU work over a binary wire (magic `BSHK`). Bend hands lumbda's upstream-format `ops.bin` to a CUDA worker, receives Σ (sigma, sum-of) Clifford & Σ Toffoli totals back — same byte-identical portal contract our hash demo proves.
- **Future CUDA growth feeds back.** Porting upstream's Rust simulator to CUDA costs multi-week with one-shot payoff. Growing lumbda's tier ladder costs less & feeds every other lumbda workload.

Phase B — Reversible Arithmetic Landed

Twelve steps after Roetteler et al. (2017), modular arithmetic over secp256k1's prime field, lumbda-native:

- **Steps 1–8** — modular add, subtract, multiply, square, inverse (Bernstein-Yang safegcd flavor), exponentiation
- **Steps 9–10** — refined Bernstein-Yang modular inverse wired into a real point-addition circuit
- **emit-ops.bin walker** — lumbda → (to) upstream QECCOPS1 binary format

Status: cross-tier validated on Python tier. C-tier & asm-tier escalation routes through a QEMU guest (`ecdsa/vm-runner.sh`) — host policy locked after a 2026-06-03 C-tier OOM crash. Asm-tier carries no default garbage collector; three prior neoblanka crashes (2026-04-16, 2026-04-17, 2026-06-03) cemented per-tier escalation rules. See lumbda's CLAUDE shard on asm memory discipline.

Form D Structural Finding — 2026-06-05

`cuda-clifford-stabilizer` as originally scoped on our bend catalog (lumbda.com/bend) does not apply to a point-addition circuit. Build agent measured Toffoli fraction at 13.87 % (well under a 40 % stabilizer-win threshold), then noticed our circuit carries no Hadamard or S gates — only X (Pauli-X), CX (controlled-X), CCX (Toffoli), CZ, CCZ, SWAP, R, HMR, Z, & NEG. State never leaves a computational basis. Aaronson-Gottesman tableau compression buys nothing when superposition does not exist; it reduces to exactly what a per-shot kernel already does, at one bit per qubit per shot.

Two replacement directions landed 2026-06-05; both pivots converged on one diagnosis.

Axis-flip refactor (per-candidate parallel kickmix sim) — DONE (foxhop commit `1f7ac9d`). 217 Mops/s (mega-ops per second) at K=32 M=4 on a RTX 3090; 23.7× (times) over per-shot N=4 at same M. Both kernels saturate at ~220–250 Mops/s. Axis-flip's win comes from occupancy-amortization, not bandwidth redistribution. Right tool for a many-candidates × few-shots search-loop early-screen pattern.

QECCOPS2 packed ops.bin — DONE (foxhop commit `90484ca`). 1.07× kernel speedup, 2.33× on-disk shrink (716 MB → 307 MB). Our original 3.5× projection assumed 56 B/op stayed VRAM-resident; `ops_loader.c` already narrowed to 28 B on load, so realistic ceiling sat at 1.17×. Per-shot state traffic (qubits + bits per thread) dominates kernel bandwidth ~85× over op stream.

Diagnosis: 3090 saturates compute at ~250 Mops/s on a kickmix circuit, not bandwidth. Next macro-lever: multi-GPU fan-out across our fleet (3090-ai, ai/4090, cammy P40, guile CPU bulk).

Measured Numbers — RTX 3090

Against HEAD's 12.8 M-op kickmix `ops.bin` (716 MB) via bend:

n_batches	shots	wire-s	cpu-ms	gpu-ms	gpu/cpu
1	64	5.5	42	5 107	0.008
16	1 024	7.3	850	5 800	0.146
64	4 096	11.4	3 444	6 216	0.553
128	8 192	17.1	7 060	6 604	**1.07**

Crossover at ~115 batches. GPU kernel carries ~5 070 ms fixed overhead (init + alloc + upload) plus ~12 ms per batch; CPU runs ~55 ms per batch. Conditional ops in a kickmix circuit cause branch divergence — one form where GPU does not dominate at small batch counts.

Honest signal, not hype.

What Lives Where

- **Lab repo (private)** — lumbda source for our attack, circuit build / sim / score / candidate sweep, QEMU envelope around C-tier & asm-tier runs, run artifacts under `runs/lumbda-*/`
- **Bend catalog & wire protocol** — lumbda.com/bend
- **Upstream challenge** — ecdsafail/ecdsafail-challenge
- **Lineage** — Google Quantum AI · ECC quantum vulnerabilities (Zenodo dataset 19597130, March 2026)

Pareto Frontier We Aim To Beat

Upstream README reports two Google private Pareto points sitting below a textbook 1.07×10^{10} score; upstream claims both points sit strictly beatable.

Variant	Toffoli (avg/shot)	Peak qubits	Score
	3 942 753	2 715	1.07×10^{10}

Challenge initial circuit (textbook)			
Google private, low-qubit Pareto point	2 700 000	1 175	3.2×10^9
Google private, low-gate Pareto point	2 100 000	1 425	3.0×10^9

Our lumbda-tier validation gates each variant against byte-identical cross-tier portals before we ever spend Rust submission budget against 9024 Fiat-Shamir-derived test points.

Why Track This Publicly

A challenge of this shape rewards aggressive, asymmetric search. Every factor of two off a Toffoli count, every qubit shaved off peak width, multiplies straight through Shor's algorithm. Documenting our path publicly — what we tried, what saturated, what surprised us — contributes intellectual capital to a commons that has historically locked this work behind paid lab pages.

This page tracks public-facing progress only. Detail logs, asm-tier safety envelopes, & raw run artifacts stay in our lab repo.

Source: https://foxhop.net/5243e3fe-6146-11f1-8ce9-040140774501/secp256k1-point-addition-challenge-with-lumbda-attack
Snapshot: 2026-06-06T06:23:54Z
Generator: Remarkbox `50b9d1e`