qBitTensor Labs

Breaking BTC

A Public Stress-Test of the Elliptic-Curve Cryptography
Securing Bitcoin

qBitTensor Labs • Enigma / Subnet 63

Abstract

Every Bitcoin is protected by a single mathematical assumption: that recovering a private key from its public key, the *elliptic-curve discrete-logarithm problem* (ECDLP) on the curve **secp256k1**, is computationally infeasible. That assumption guards on the order of a trillion dollars of assets, and, like RSA, it is broken outright by a large enough quantum computer running Shor's algorithm, at a lower qubit cost than RSA. **Breaking BTC** is a continuously running, incentivized, reproducible challenge that measures the real-world hardness of that problem. It hands participants *scaled-down* instances on Bitcoin's actual curve (recover a private key known to lie in a bounded interval) on identical hardware and a fixed four-hour budget, and pays them when they succeed. It formalizes the frontier already visible in the real "Bitcoin puzzle" transactions, turns it into a public scoreboard, and places it alongside its sibling challenge Breaking RSA so that two pillars of modern cryptography, integer factorization and elliptic curves, can be tracked under one operational yardstick. This paper describes the challenge, its mathematics and engineering, the quantum context, and how to participate through Bittensor or at qbittensorlabs.com. It is the initial design, and we expect to refine it as participants and validators put it to use.

Part I

Overview

1 Why this matters

A large share of the digital signatures in use today (every Bitcoin and Ethereum transaction, much of the TLS that secures the web) relies on **elliptic-curve cryptography (ECC)**. Its security rests on one problem: given a public key $\mathbf{Q} = k \cdot \mathbf{G}$ formed by adding a base point $\mathbf{G}$ to itself k times, recovering the secret multiplier k is believed infeasible. For Bitcoin the curve is **secp256k1**, and k is your private key.

Two facts make this uniquely worth measuring in public. First, the **stakes**: secp256k1 secures on the order of a trillion dollars of Bitcoin alone. Second, and unlike the lattice-based post-quantum standards, **ECC is not quantum-resistant**: a large enough quantum computer running Shor's algorithm recovers k efficiently. ECC is the cryptography quantum computers break *first*.

As with RSA, the way to earn confidence is to watch the world's best try to break it, in the open, at increasing scale, and to know exactly how far they get. The elliptic-curve world already has a natural public frontier (the "Bitcoin puzzle" transactions, below), but no fixed, incentivized, apples-to-apples measurement of it. Breaking BTC provides one.

2 The central idea

> **Breaking RSA** asks how large a number can be factored in four hours on one GPU. **Breaking BTC** asks: *how large a Bitcoin private-key range can be cracked in four hours on one GPU?*

It is the same fixed hardware and four-hour clock, now pointed at the elliptic-curve mathematics on every Bitcoin, and at the cryptographic pillar with the nearest quantum expiry date.

3 What "breaking it at small scale" means

We do **not** attempt to break real 256-bit Bitcoin keys; that should be infeasible, which is the point. Instead we take the *same* problem on Bitcoin's *real* curve and shrink its one security-critical parameter: the size of the private key. In each challenge the private key k is known to lie in an interval $[1, 2^n)$, and the difficulty is the interval bit-width n. We start at small n (crackable in seconds) and climb toward the full 256 bits.

The construction is not artificial: it is exactly how the famous **Bitcoin puzzle** transactions are built (private keys drawn from bounded intervals, with the public keys of many deliberately published), and it is how those puzzles are being solved today. Somewhere up the ladder, the best hardware and algorithms hit a wall. That wall is the measurement we publish: how much elliptic-curve security a single machine can peel away in four hours. Read against the 256-bit keys Bitcoin actually uses, it is also a concrete marker of how far off the quantum cliff still is.

Why four hours on one GPU: a test bed that scales. The small, fixed envelope is deliberate. Four hours on one identical card is a controlled measurement: every entrant runs on the same clock and the same silicon, so a result reflects method rather than hardware budget. The metric also generalizes upward. An algorithm that extracts more discrete-logarithm progress per GPU-hour than the best public solver is a better algorithm at any scale, and that same per-card efficiency, multiplied across an HPC cluster, is what moves the real puzzle frontier. Shrinking the interval changes only the budget, not the curve or the mathematics, so the contest isolates the one variable that matters: capability per unit of compute.

4 Why a Bittensor subnet is the right instrument

Rather than one lab running one experiment, Breaking BTC draws on a global pool of economically motivated participants who attack the same reproducible instances on identical hardware, continuously, while the subnet's validators independently verify every claimed break.

What is new, relative to the Bitcoin puzzle? The puzzle already invites the world to break interval ECDLP. What Breaking BTC adds is discipline of measurement and, above all, capture of *method*:

- **The method is captured, not only the outcome.** A honeypot address teaches nothing: whoever cracks it sweeps the funds, and the technique stays private unless the winner chooses to boast. In Breaking BTC the winning *solver* is the deliverable (submitted, independently verified, and published under AGPL-3.0), so we learn how each break was achieved and when the frontier moved. Every solve becomes dated, reproducible public knowledge rather than an anonymous disappearance of coins.
- **Fixed, identical hardware and a fixed four-hour envelope**, so results are comparable across time, across entrants, and against the sibling Breaking RSA challenge, which the open-ended puzzle is not.
- **A fresh random instance drawn per submission**, so the challenge measures *capability*, not luck or a land-grab. (The puzzle's #66 solve was infamously front-run in the mempool by a bot; a per-submission instance removes that failure mode entirely.)

- **Continuous, sustained incentives** in place of a static one-time pot.

Together these turn scattered, private effort into a comparable and well-documented scoreboard.

Part II

Technical Specification

5 Background: ECDLP, secp256k1, and Bitcoin

An elliptic curve over a finite field $\mathbb{F}_p$ is the set of points (x, y) satisfying a cubic equation, plus a point at infinity; points can be "added" by a geometric rule, making the set a finite abelian group. **secp256k1**, Bitcoin's curve, is $y^2 = x^3 + 7$ over the prime field $\mathbb{F}_p$ with $p = 2^{256} - 2^{32} - 977$. Its base point $\mathbf{G}$ generates a group of prime order $N \approx 2^{256}$ (cofactor 1), so every nonzero multiple of $\mathbf{G}$ is a valid public key and each corresponds to a unique private key.

A Bitcoin private key is a scalar k; the public key is $\mathbf{Q} = k \cdot \mathbf{G}$ (repeated point addition); the address is a hash of $\mathbf{Q}$. Signing with ECDSA proves knowledge of k without revealing it. The **security of the whole edifice reduces to the ECDLP**: given $\mathbf{G}$ and $\mathbf{Q}$, find k. The best generic classical attack costs about 2^{128} operations for a 256-bit order, which is infeasible. (Real-world ECDSA failures are almost always implementation bugs, e.g. repeated signing nonces, not ECDLP breaks; this challenge targets the ECDLP itself.)

A note on exposure that motivates the design: an address is a *hash* of the public key, so an unspent, never-reused address reveals only that hash. But the public key itself is published the moment the address is spent from, and for P2PK outputs and any reused address it is already public. Exposed public keys are the direct target of an ECDLP attack (and of a future quantum attack); the Bitcoin puzzle exposes them deliberately.

6 The challenge instance

An instance at difficulty n is generated as follows.

1. Sample a 256-bit master seed $\mu \leftarrow$ `secrets.randbits(256)` (CSPRNG).
2. Derive the private key uniformly in the interval. With $B = \min(2^n, N)$ and a 512-bit SHAKE-256 output (so the modular bias is below 2^{-250}), $k = 1 + \big(\text{SHAKE-256}_{512}(\mu) \bmod (B - 1)\big) \in [1, B)$.
3. Compute the public key $\mathbf{Q} = k \cdot \mathbf{G}$ on secp256k1.

The solver receives $(n, \mathbf{Q})$ on the fixed curve secp256k1 and must output k. The curve is never scaled or altered, and the base point is always Bitcoin's standard $\mathbf{G}$; only the interval shrinks. (The code treats the base point as a parameter, but it is always set to $\mathbf{G}$: the target is Bitcoin's keys, not elliptic curves in general.)

7 Verification: an exact predicate

Verification is a single elliptic-curve scalar multiplication: no simulator, no floating point, no tolerance, and *no probabilistic argument*. A submission $k^\star$ is accepted iff:

$$\textbf{(i)}\ \ 1 \le k^\star < \min(2^n, N), \qquad \textbf{(ii)}\ \ k^\star \cdot \mathbf{G} = \mathbf{Q}, \qquad \textbf{(iii)}\ \ k^\star = k.$$

Because $\mathbf{G}$ generates a group of **prime order** N, the discrete logarithm of $\mathbf{Q}$ is *unique* in $\{0, \ldots, N-1\}$. Keys are drawn from $[1, \min(2^n, N)) \subseteq [1, N)$. Capping the interval at the group order means that even the full-width Satoshi Line (n=256) is Bitcoin's real keyspace $[1, N)$ rather than $[1, 2^{256})$, which would overshoot N and admit two representatives (k and k+N) of the same key. Within $[1, N)$ the restriction leaves *exactly* one solution for a well-formed instance, so (i)+(ii) already determine k with certainty. There is no

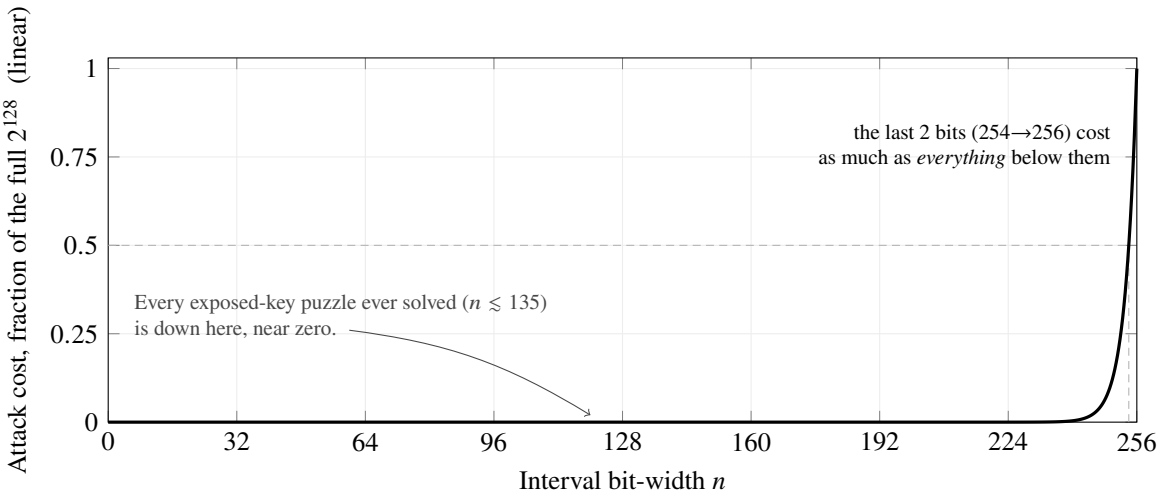


Figure 1: **Why the ladder is a wall** (illustrative). Kangaroo cost $\sim 2^{n/2}$ plotted on a *linear* axis, normalized so the full 256-bit key (the Satoshi Line) is 1. Because cost doubles every two bits of width, the curve is flat-then-vertical: widening the interval by its final two bits (254 → 256) adds as much work as the entire remainder of the keyspace below it (a geometric series with ratio 2). Every interval ever broken in public sits in the sliver near zero.

"unique-decoding regime" to assume; the uniqueness here is a theorem rather than a heuristic. Condition (iii) is belt-and-suspenders that makes a false accept impossible even against a malformed instance. The solver cannot exploit (iii): the key is never transmitted (only the *public* key $\mathbf{Q}$ is, which reveals nothing without solving the ECDLP), and a random guess lands in the interval-restricted solution set with probability 2^{-n}. Hostile inputs (non-integers, enormous values, out-of-interval keys, or $k+N$, which generates the same point but fails the range test) are rejected before any curve arithmetic.

8 The attack surface

For a private key confined to a known interval of width $W = 2^n$, the method of choice is **Pollard's kangaroo** (the "lambda" method). Two herds of pseudo-random walks (a *tame* herd started from a known position and a *wild* herd started from $\mathbf{Q}$) take jumps determined by the current point, recording "distinguished" points until a tame and a wild walk collide; the collision reveals k. The expected work is

$$\mathbb{E}[\,\#\text{group ops}\,] \;\approx\; \underbrace{c}_{\substack{\text{kangaroo}\\ \text{constant}\\ 1.15\text{–}2}} \cdot \underbrace{\sqrt{W}}_{=2^{n/2}}, \qquad W = 2^n,$$

with *negligible* memory, which is why it (rather than baby-step/giant-step, which needs $2^{n/2}$ memory) is what actually solves the exposed-key Bitcoin puzzles. Pollard's rho attacks the full group at the same $\sim 2^{n/2}$ cost when no interval is known. Kangaroo is the best *known* method, and so it is the yardstick: it sets the difficulty scale and the baseline every milestone is measured against. It is not the expected path: how a solver gets there is up to the participant, and any approach is welcome (§9). Figure 1 first shows, on a linear scale, why a mere ladder of bit-widths behaves like a wall; Figure 2 then plots the same cost on a log axis against difficulty.

The empirical frontier. The Bitcoin puzzle transactions (created 2015, keys drawn from $[2^{k-1}, 2^k)$) are the live benchmark. For the subset whose *public keys were published*, kangaroo on GPU pools has solved

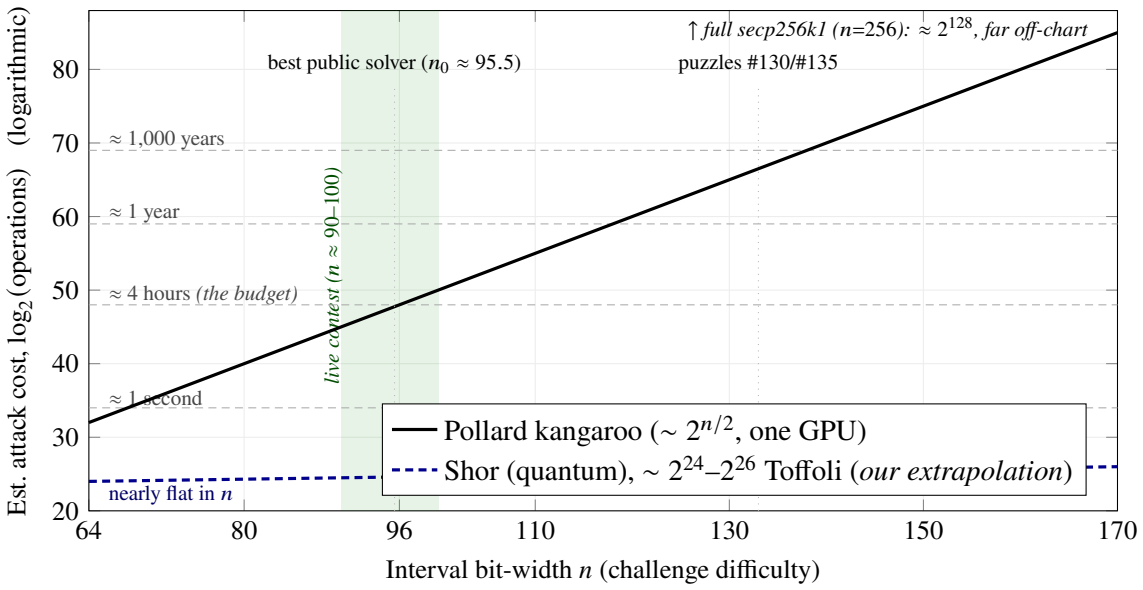


Figure 2: Estimated single-GPU attack cost versus difficulty. **The vertical axis is logarithmic**; because the kangaroo cost is $\sim 2^{n/2}$, the plot is a straight line of slope $\frac{1}{2}$: cost grows *exponentially* in the interval bit-width. Horizontal references translate operation counts into wall-clock time at a GPU kangaroo rate of $\sim 2^{34}$ jumps per second, close to the 19.4 GKeys/s we measured on the validator card. There the best public solver's average run fills the four-hour budget at $n_0 \approx 95.5$ (Table 1), and the shaded **live-contest band** ($n \approx 90$–100) is where beating the best public solver on the fixed card is in reach; the real puzzle record has single/few GPUs reaching $n \approx 90$–96 while $n \gtrsim 110$ (puzzles #130/#135) needs a pool. Bitcoin's real keys sit at $n = 256$ ($\approx 2^{128}$ work), far above the plotted range. The near-flat **Shor line** shows a quantum attack's cost barely growing with key size; its vertical placement ($\sim 2^{24}$–2^{26} Toffoli gates) is *our own extrapolation*, scaled down linearly in n from Google Quantum AI's 2026 full-key gate estimate (§11), and is illustrative only; a large quantum computer collapses the classical curve to a low-order polynomial.

up to puzzle **#130** ($\approx 2^{65}$ work) and **#135** ($\approx 2^{67}$ work; solved July 2026, reportedly ~5 months on ~200 GPUs); a single high-end GPU (e.g. an RTX 4090 at ~ 14.5 GH/s, RCKangaroo's published figure) clears an $n \approx 90$ interval in about an hour and $n \approx 94$–96 in about four. Separately, the largest *full-group* ECDLP solved on a general prime-field curve remains **ECCp-109** (2002) and the **112-bit** secp112r1 (2009, $\sim 2^{56}$ effort, i.e. half the order); 131-bit and beyond are unsolved. These full-group records (Pollard rho on the whole group) calibrate the top of the ladder (the Satoshi Line), while the interval-puzzle numbers calibrate the interval rungs.

9 What we hope to see

Breaking BTC sets no expectation about method. Kangaroo is the best known approach today, but it is the starting line, not the finish. At a minimum we hope to be impressed. The real point of putting an open problem in front of innovators around the world is to be blown away: by radical breakthroughs that redefine the state of the art.

Progress could come from many directions, e.g.:

- **engineering**, e.g. faster kernels, better use of the card's memory and parallelism, smarter distinguished-point strategy, careful tuning;

- **algorithms**, e.g. new walk designs, new collision strategies, or ways to exploit the structure of secp256k1 itself, such as its efficient endomorphism;
- **ideas nobody has had yet**, which is the reason to run an open contest at all.

The fixed card and fixed budget make the size of a result legible. Engineering gains tend to arrive as a steady climb, roughly two bits of interval width for each doubling of effective speed. Beyond that, for methods that treat the group as a black box, square-root cost is a proven floor (Shoup, 1997), so a result whose advantage keeps growing with n would mean something specific to secp256k1 has been found. That would be a genuine advance in cryptanalysis, exactly the kind of result this challenge exists to surface, and our disclosure commitment would apply. Like the quantum red flag (§11), it is a signal the board is built to make visible.

10 Difficulty and progression

Difficulty is the single knob n, the interval bit-width, and kangaroo work is $\sim 2^{n/2}$ (Figure 2). The same exponential curve that makes a small interval trivial makes each additional handful of bits exponentially harder, which turns one problem into an open-ended ladder rather than a fixed puzzle. A few natural regimes fall out of the four-hour, single-GPU budget:

- **On-ramps** (roughly $n \lesssim 80$): seconds to minutes with off-the-shelf tools. These reproduce trivially solvable intervals to validate solvers and seed the board; they are not where the contest lives.
- **The single-GPU frontier** (roughly $n \approx 90$–100): on the validator card the best public solver averages under an hour near $n \approx 90$ and fills the budget at $n_0 \approx 95.5$ (measured). This band is the live contest: progress here means getting more out of the fixed hardware than the best public solver does, by whatever means.
- **Beyond single-GPU reach** ($n \gtrsim 110$): where the exposed-key Bitcoin puzzles sit (#130 at $\approx 2^{65}$; #135 at $\approx 2^{67}$, reportedly ∼5 months on ∼200 GPUs), out of reach of one card in the budget, and a marker for pooled effort or algorithmic advance.
- **Full secp256k1** ($n = 256$, the full keyspace $[1, N)$), the **Satoshi Line**: a full Bitcoin key, 256 bits, about 2^{128} operations with the best classical method. It is Bitcoin's real lock, a standing line that classical hardware does not cross, and that a large quantum computer would (§11).

Passing means five for five. To win, a solver must pass five complete validations. Each validation uses its own freshly generated key and gets its own four-hour window, and all five must recover the key, with no misses. Kangaroo run time is random: a collision can arrive early or late, and the five-for-five rule is what keeps luck out of the result. A solver whose average run is exactly four hours passes all five only about one time in twenty; one whose average run is half the budget passes about 80% of the time, and one whose average run is a quarter of the budget virtually always passes. A solver two bits short of a milestone would need thousands of resubmissions to win by chance. The practical frontier therefore sits about three bits below the width at which the average run fills the budget, and milestones are set against that practical frontier. These figures come from the standard collision-time model; our 123 measured solves follow it closely.

Measured baseline. We benchmarked the reference solver on an NVIDIA RTX PRO 6000 Blackwell Workstation Edition (600 W, sm_120), the validator card, in November 2026: 123 timed solves on random keys at

Speed	K	n_0	Passes all five validations at		
(GKeys/s)			$n_0 - 2.5$	$n_0 - 1.5$	$n_0 + 0.5$
19.4	1.18 ± 0.07	95.5 ± 0.2	94% (93)	56% (94)	1.4% (96)

Table 1: Measured reference-solver baseline on the validator card (RCKangaroo v4.0). K is the effective kangaroo constant including distinguished-point and GPU overheads; widths in parentheses are the interval bit-widths for each pass rate.

$n = 76$–88, all recovered correctly (Table 1). Here n_0 is the width at which the average run fills the four-hour budget.

Funded challenges use a **minimum of** $n \geq 64$, since a smaller interval is a matter of seconds. Specific funded targets are set by the subnet operator and published on the Enigma page; they advance over time as the frontier moves, rather than being fixed in advance here.

11 The quantum dimension

The post-quantum standards and ECC part ways on exactly this point. Lattice schemes such as ML-KEM are believed to resist quantum attack; **ECC does not**. Shor's algorithm solves the ECDLP in polynomial time, so a sufficiently large fault-tolerant quantum computer recovers any Bitcoin private key from its public key. For a key in an interval of width 2^n on secp256k1, whose field size is fixed at $b = 256$ bits, the whole security margin is the gap between these two scalings:

$$\underbrace{T_{\text{classical}}(n) \;\approx\; c\,2^{n/2}}_{\text{exponential in } n} \quad \xrightarrow{\text{Shor}} \quad \underbrace{T_{\text{quantum}}(n) \;=\; O\big(n\,b^2\big)}_{\text{linear in } n\text{; qubits set by } b}.$$

Shor needs about n controlled point additions, each carried out in full 256-bit field arithmetic. Its gate count therefore grows only linearly with the interval width, and its qubit count is set by the field, not by the key: even a small-interval secp256k1 key needs most of a full-scale machine. (For a full key on a curve of size b, $n = b$ and the familiar $O(b^3)$ follows.)

The estimates keep shrinking. Resource estimates for a full 256-bit secp256k1 break have fallen steadily: logical-qubit counts have roughly halved since 2017, and physical-qubit estimates have fallen by well over an order of magnitude. Early analysis put it at roughly 2,330 logical qubits (Roetteler, Naehrig, Svore, Lauter 2017) and, translated to hardware, 10^7–10^8 physical qubits (Webber et al. 2022: ~13M physical qubits for a one-day break, ~317M for one hour, under surface-code assumptions). The leanest to date is the **March 2026 Google Quantum AI** estimate (Babbush, Zalcman, Gidney et al., with Ethereum Foundation and Stanford coauthors): **under** 1,200**–**1,450 **logical qubits** and ≤ 70**–**90**M Toffoli gates**, executable in *minutes* on **fewer than half a million physical qubits** at a 10^{-3} error rate. (It reports two operating points: a low-qubit variant, <1,200 logical / <90M Toffoli, and a low-gate variant, <1,450 logical / <70M Toffoli.)

The hardware is far away, but moving. On real hardware, the largest public quantum attack on an elliptic-curve key is **15 bits** (Project Eleven's Q-Day Prize, awarded to Giancarlo Lelli on 24 April 2026; a 32,767-point search space), up from **6 bits** in September 2025. That is a vast distance from the 2^{256} keyspace of a real key, but what matters is the *rate* of change rather than the current record. How lean the attack *circuit* can be is itself now a public contest: **ECDSA.fail** (Eigen Labs) scores candidate circuits for one secp256k1 point addition on the product of logical qubits and Toffoli gates; since its launch on 30 May 2026, submitters

have reported low-qubit circuits as small as **676 logical qubits** (as of November 2026). These are resource estimates rather than hardware runs.

Three moving lines. Three quantities are moving at once:

1. **how far classical computers can go** (the frontier Breaking BTC measures directly);
2. **how lean a quantum attack can get** (the Google 2026 estimate and the ECDSA.fail benchmark);
3. **how fast quantum hardware grows** (the vendor roadmaps, Figure 3).

When quantum breaks Bitcoin is set by two of them. Q-Day is where rising hardware (3) meets the falling cost of the leanest attack (2). That crossing is the picture in Figure 3: roadmaps climbing toward a threshold band that itself keeps moving down as estimates improve. Classical progress does not move it, since no classical solver threatens a full key.

When to start worrying depends on all three. The earliest real signal is the *red flag*: the moment quantum hardware, running the leanest available attack, recovers a larger interval key than the best classical solver can. That is a crossing between the quantum line (2 and 3 together) and the classical line (1). qBitTensor Labs moves the classical line, in public and continuously, and in doing so defines the line the red flag has to cross.

Expert surveys vs. vendor targets. Two timelines are often quoted, and they are easy to mistake for a contradiction. Independent expert surveys of when a *cryptographically relevant* machine might exist range from roughly **a decade to several decades**, and are openly contested. **Vendor roadmaps**, by contrast, publish targets that reach the 1,200–1,450-logical-qubit range between about **2028 and 2033** (Figure 3). Both can hold at once: the first is a survey of *independent expectation*; the second is a set of *company targets*, dates on which vendors have historically slipped and lately sometimes accelerated. We quote both and label which is which. The reasonable reading is a wide and contested window rather than a date.

The exposure is already real. Over **1.7 million BTC** sit in pay-to-public-key (P2PK) outputs whose keys are permanently on chain, and by one widely cited accounting about **2.3 million BTC** are both quantum-vulnerable and dormant (unmoved for five years or more); reused addresses expose more. All of it is attackable under "harvest-now, decrypt-later" the moment the hardware exists. Bitcoin has no easy upgrade path: post-quantum signatures are an order of magnitude or more larger, and migration requires network-wide coordination, which is why the community is already debating proposals such as BIP-360.

11.1 The red flag and the danger zone

Three terms make it clearer what to watch for.

- **The red flag:** the moment a quantum computer recovers a *larger* secp256k1 interval key than the best classical solver can. It is a dated event anyone can check, and Breaking BTC is what keeps "the best classical solver" current.
- **Q-Day:** the later point when quantum hardware can break a *full* 256-bit key, the Satoshi Line.
- **The danger zone:** the stretch from the earliest plausible red flag to the latest plausible Q-Day. On today's vendor roadmaps that is roughly 2028 to 2033 (Figure 4).

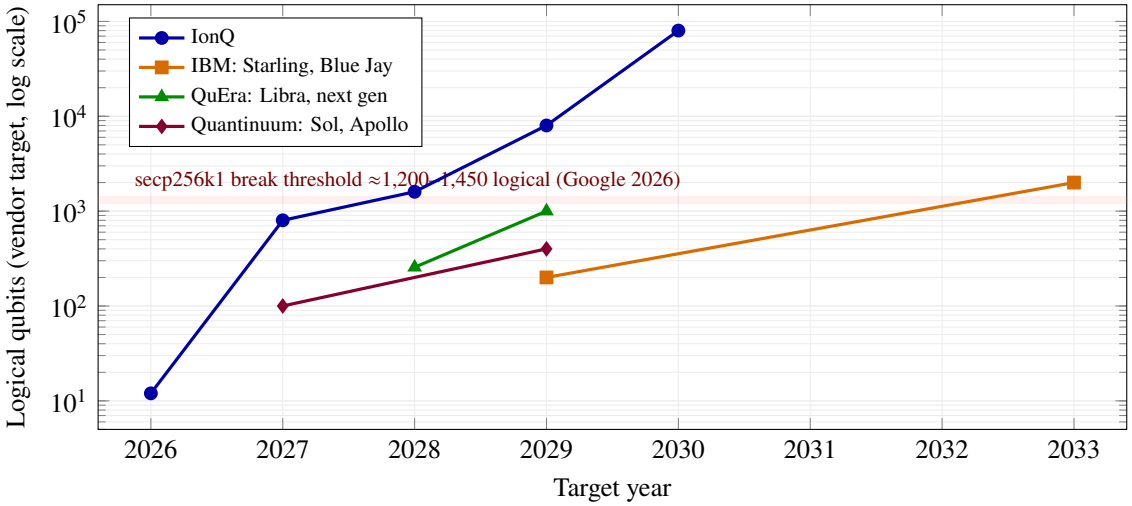


Figure 3: **Vendor logical-qubit roadmaps against the quantum-break threshold** (as of November 2026). Points are *published targets, not deliveries*, and vendors define "logical qubit" differently; error rate and circuit depth matter as much as the count. The shaded band is Google Quantum AI's 2026 resource estimate for a full secp256k1 break (≈1,200–1,450 logical qubits); the years at which roadmaps cross it (roughly 2028–2033) set the width of the danger zone in Figure 4. Quantinuum's "Apollo" is stated as "hundreds" of logical qubits (plotted ≈400, order-of-magnitude); IBM's Blue Jay is a 2033+ target.

The red flag comes first and is the easier signal to read. It is the one Breaking BTC is built to catch, in public, on Bitcoin's own curve.

A measured posture. None of this means the sky is falling. It means there is a specific, observable line worth watching, and a public instrument pointed at it. **Be there when it crosses.**

12 Implementation

Breaking BTC is a standard Enigma challenge and reuses the existing validation, consensus, and payout infrastructure.

- **Challenge contract.** A `Challenge[Problem, Solution, Verif]` with `generate(seed)` and a shared `validate_btc_solution()` used by both the offline workbench and the live validator. secp256k1 arithmetic is implemented in pure Python, with *no third-party dependency* on the validator side.
- **Secret isolation.** The generator writes the solver's `challenge_input.json` (only n and the public key $\mathbf{Q}$) into the read-only mount, and the ground-truth private key into `verif.json` *outside* that mount. The master seed is never persisted.
- **Runtime.** The validator's hardened profile: four-hour wall clock, `--network none`, read-only root filesystem, non-root user, NVIDIA RTX PRO 6000 (96 GB), 24 vCPU, 85 GB RAM.
- **Anti-replay and anti-gaming.** Each submission goes through five complete validations, each with its own fresh, independent random key and its own four-hour window, so a memorized answer cannot pass; the submission wins only if all five succeed. The validator set is permissioned, bounding rubber-stamp and Sybil risk.

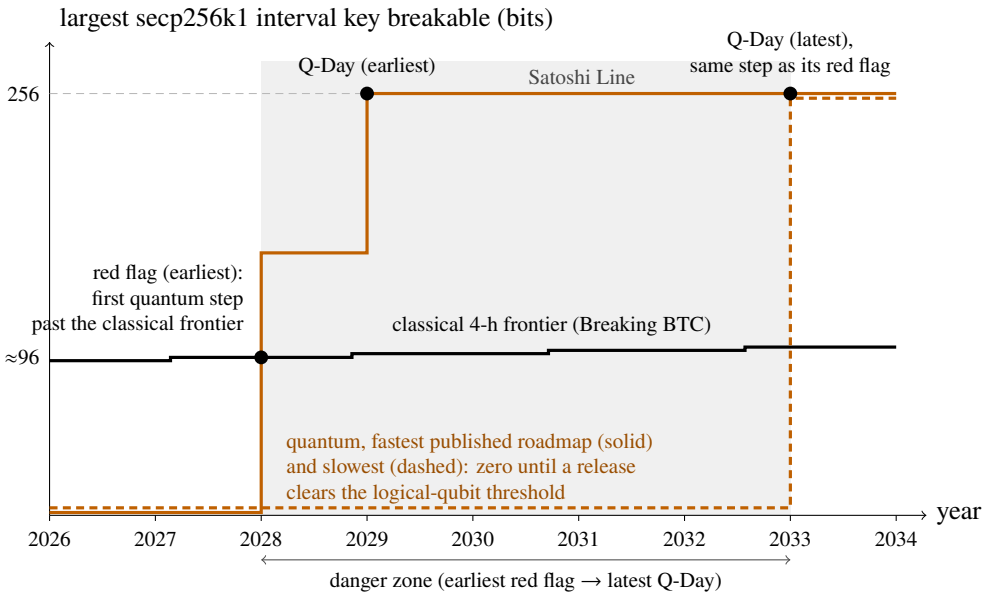


Figure 4: **Three lines, the red flag, and the danger zone** (illustrative; step heights are not to scale). Quantum capability on secp256k1 arrives in *steps* tied to hardware releases rather than as a smooth climb: below the logical-qubit threshold a machine cannot attack secp256k1 at any key size, because its qubit count is set by the 256-bit field, and above it the breakable interval jumps, limited at first by circuit depth and error rate and then reaching the full 256 bits. The orange steps combine two of the three lines, hardware and attack cost: they are placed at the years in which the fastest and slowest published roadmaps first meet the ≈1,200–1,450 logical-qubit estimate (Figure 3); these are vendor targets, not deliveries. The black line is the classical four-hour frontier, which Breaking BTC raises in small steps as milestones fall. The **red flag** is the first quantum step that clears the classical frontier; **Q-Day** is the step that reaches the Satoshi Line; the **danger zone** spans the earliest plausible red flag to the latest plausible Q-Day. On a slow roadmap the red flag and Q-Day can be the same step.

- **No precomputation.** Every solve starts from scratch. The interval $[1, 2^n)$ is the same for every instance, so a solver could in principle ship a precomputed table of tame-walk points and reuse it across keys. Our measurements show why this matters: at $n = 80$, tables of 2 and 6 GB made the reference solver 3.8 and 5.4 times faster. Such tables are not allowed. Solvers build on an approved base image, and what they add on top of it is held to a small size cap published with the challenge rules. Anything downloaded while the image is built lands in the image and counts toward the cap, and the run itself has no network. A cap of 50 MB, for example, would limit any table to roughly a 30% speedup at $n \approx 96$.
- **Reference solver.** The shipped reference is a serious baseline rather than a toy: it wraps **RCKangaroo** (RetiredCoder), the strongest public GPU kangaroo for secp256k1, and its measured results are in Table 1. Operational details (build, driver requirements, failure handling) are in the challenge README. Milestones ask entrants to do better than this solver on the one fixed card, by any means (§9) other than more hardware or precomputed data.

An automated test suite checks the curve constants against known vectors, generation determinism, secret non-leakage, positive, negative and hostile verification, the reference wrapper's handling of the GPU solver, and an end-to-end kangaroo solve. The developer workbench exercises the full pipeline: generate, solve,

verify.

Part III

Participation and Observation

13 How to participate

Breaking BTC operates on **Enigma, Subnet 63** of Bittensor, with a front door at qbittensorlabs.com/enigma. You build a **containerized solver**: a Docker image that reads a problem instance and emits a `result.json` with the recovered private key, following the Enigma output protocol. Any language, any algorithm (GPU kangaroo, rho, or a novel method) is admissible; the constraints are the sandbox (four hours, no network, one GPU), no precomputed data, and source transparency. Develop offline with the workbench:

```
python -m workbench test breaking-btc --difficulty 90 --solution ./my_solver/
```

Through Bittensor: register a hotkey on **netuid 63**, run your solver, and compete in the subnet's incentive mechanism; Subnet 63 validators are the decentralized verification layer. **Directly through qbittensorlabs.com:** pick a milestone (an interval width) on the Enigma page, upload your solver, submit an entry (anchored to your hotkey with a nominal 0.1 TAO anti-spam fee); it is run in five complete validations, each with its own freshly generated key and its own four-hour window, all five must succeed, and the prize is paid after review.

Prizes are published per milestone in USD (paid in Subnet 63 Alpha, optional USD election); **eligibility** follows the published rules (18+, no sanctioned parties, no insiders); **winning solutions are published under AGPL-3.0**, so each advance becomes shared knowledge. The authoritative terms live in the Enigma rules at qbittensorlabs.com; this is orientation only, and makes no representation as to token prices or timelines.

An initial design. This paper describes Breaking BTC as it launches. Several settings (the milestone widths, the five-for-five pass rule, the image size cap, the validator driver) are first choices based on our own measurements rather than on contest experience. We expect to learn a good deal once participants and validators start using the challenge, and we will likely adjust these settings as we do. Changes will be published on the Enigma page and in the rules.

14 How progress is observed

The **Enigma milestone board** shows each milestone's status (*Incomplete*, *In Review*, *Complete*) and, once solved, the winning address and on-chain payout; the highest solved interval is the current empirical frontier.

Part IV

Context

15 Relation to Breaking RSA

Breaking RSA and Breaking BTC run on identical hardware within an identical four-hour budget, so two pillars of modern cryptography, integer factorization and elliptic curves, can be tracked against one operational yardstick. They are *not* commensurable: a factored modulus and a recovered elliptic-curve key share no common security-bit metric, so this is a narrative device rather than a unified measure. What it does state is what each yields under one fixed compute envelope. And because elliptic curves are the pillar quantum computers break first, Breaking BTC is the one whose frontier doubles as a countdown.

16 Responsible disclosure

Breaking BTC operates on deliberately scaled-down intervals and *never* targets real, funded keys; instances are freshly generated random keys rather than addresses holding value. It does not break, and is not intended to break, live Bitcoin. Should any participant's technique, classical or quantum-assisted, demonstrably threaten deployed key sizes, qBitTensor Labs is committed to coordinated disclosure with the relevant communities before publication.

A secp256k1 parameters

Field prime	$p = 2^{256} - 2^{32} - 977$
Curve	$y^2 = x^3 + 7$ (cofactor 1)
Order (prime)	$N =$ `FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141`
Difficulty	interval bit-width n (private key in $[1, 2^n)$); $n \geq 64$ for prizes
Wall time	14,400 s (4 hours)
Compute	RTX PRO 6000 (96 GB), 24 vCPU, 85 GB RAM

B Glossary

secp256k1 Bitcoin's elliptic curve, $y^2 = x^3 + 7$ over $\mathbb{F}_p$ with $p = 2^{256} - 2^{32} - 977$ and prime group order $N \approx 2^{256}$. Every Bitcoin key lives on it.

Satoshi Line A full Bitcoin key: 256 bits, about 2^{128} operations with the best classical method. The top rung of the ladder and Bitcoin's real lock.

Bitcoin puzzles The 2015 "puzzle" transactions, with private keys drawn from bounded intervals $[2^{k-1}, 2^k)$ and many public keys deliberately published; the live public ECDLP frontier. Referred to as "puzzle #N" and kept strictly separate from our funded milestones.

Funded milestones Fresh random keys at interval widths the subnet operator sets, each placed just past the best known solution and published on the Enigma page. Distinct from the 2015 puzzles, and set as the frontier advances.

Reference solver The shipped baseline: a wrapper around RCKangaroo, the best public GPU kangaroo. On the validator card its average run fills the four-hour budget at $n \approx 95.5$, and it passes all five validations reliably up to $n \approx 93$. The bar entrants must beat.

Shor's algorithm The quantum algorithm that solves the ECDLP (and integer factoring) in polynomial time. On a fixed curve such as secp256k1 its qubit count is set by the field size, not the key's interval, so even a small-interval key needs a near-full-scale machine.

Logical qubit An error-corrected qubit assembled from many physical qubits. Vendors define it differently, so raw counts are not directly comparable.

ECDSA.fail Eigen Labs' open benchmark scoring the leanest quantum circuit for one secp256k1 point addition, on logical qubits × Toffoli gates. Resource estimates, not hardware runs.

Q-Day The day quantum hardware can break a full 256-bit key.

Red flag The earlier signal: the moment a quantum computer recovers a larger secp256k1 interval key than the best classical solver. The event Breaking BTC is built to catch.

C Selected references

1. Certicom, *SEC 2: Recommended Elliptic Curve Domain Parameters* (secp256k1).
2. Pollard, *Monte Carlo methods for index computation mod p*, 1978 (rho / kangaroo).
3. van Oorschot, Wiener, *Parallel collision search with cryptanalytic applications*, 1999.
4. Galbraith, Pollard, Ruprai, *Computing discrete logarithms in an interval*, 2010.
5. Shoup, *Lower bounds for discrete logarithms and related problems*, EUROCRYPT 1997.
6. Bos, Kaihara, Kleinjung, Lenstra, Montgomery, *Solving a 112-bit prime ECDLP*, 2009.
7. Monico et al., *Certicom ECCp-109 solution*, 2002.
8. Bitcoin puzzle transaction, privatekeys.pw / btcpuzzlesearch.com (#130 solved by GPU-pool kangaroo; #135 solved July 2026, ~5 months on ~200 GPUs).
9. Roetteler, Naehrig, Svore, Lauter, *Quantum resource estimates for computing elliptic curve discrete logarithms*, 2017.
10. Häner, Jaques, Naehrig, Roetteler, Soeken, *Improved quantum circuits for elliptic curve discrete logarithms*, 2020.
11. Webber et al., *The impact of hardware specifications on reaching quantum advantage in the fault-tolerant regime*, AVS Quantum Sci. 2022.
12. Babbush, Zalcman, Gidney et al., *Securing Elliptic Curve Cryptocurrencies against Quantum Vulnerabilities: Resource Estimates and Mitigations*, Google Quantum AI (with Ethereum Foundation and Stanford), March 2026. quantumai.google/…/cryptocurrency-whitepaper.pdf.
13. Project Eleven, *Q-Day Prize: 15-bit ECC key broken on quantum hardware* (Giancarlo Lelli), 24 April 2026; prior public record 6 bits, September 2025.
14. Eigen Labs, *ECDSA.fail*: open benchmark for minimal quantum circuits attacking secp256k1 (logical qubits × Toffoli gates), launched 30 May 2026. ecdsa.fail.
15. Quantum hardware roadmaps (logical-qubit targets): IonQ, IBM, QuEra, and Quantinuum roadmap pages, accessed November 2026.
16. NIST, *FIPS 203/204/205 post-quantum standards*, 2024 (the quantum-resistant contrast).

qBitTensor Labs: advancing quantum computing and cryptographic science on decentralized infrastructure. Enigma / Subnet 63. This document is a technical proposal and is subject to revision; economic terms are governed by the official rules published at qbittensorlabs.com.