ECC RESEARCH PROGRAM · P1.1

When Coordinates Are Free: The Collapse of the ECDLP Generic Group Argument Under Explicit Curve Access

An operation-requirement taxonomy, a syntactic no-go theorem, and the opacity boundary of Shoup's lower bound

math.st [1] **@math__street** [1]

with GPT 5.6 Sol · Claude Fable 5

[1]Independent research collective, math.st · July 2026

ABSTRACT

The security of elliptic-curve cryptography rests, at the provable end, on a generic lower bound of $\Omega(p^{1/2})$ group operations for the discrete logarithm problem. That bound is a statement about a model — Shoup's generic group model (GGM) — in which group elements are opaque random strings. We ask a sharper question posed in the problem statement: what happens to the bound if the algorithm is granted what an attacker on a **real** curve actually has, namely explicit affine coordinates and free field arithmetic, while only the abstract group addition instruction is charged? We show the argument does not merely weaken — it collapses. We define a literal cost machine CCA_0 that charges one unit per `ECADD` and nothing for field work, coordinate projection, or point packing, and prove an **addition-compiler** theorem: every program has an extensionally equivalent program executing zero charged instructions. Consequently BSGS, and even the full Semaev–Gaudry–Diem index-calculus template, run at charged cost zero. We locate precisely where Shoup's proof uses opacity — a symbolic simulator may return a fresh random label for each new formal exponent polynomial only because the algorithm can observe nothing but string equality — and show this step, and only this step, fails under coordinate access. We give a nine-primitive operation-requirement taxonomy across eight attacks, an audit of five candidate models (Shoup, Maurer, AGM, generic ring, and CCA_0), and executable fixtures at toy scale confirming each attack row. The result is a negative one, and we state its scope honestly: it kills the **literal** free-coordinate model as a security argument and isolates the three coherent repairs, but it does not yield a new positive lower bound.

Keywords. ECDLP · generic group model · lower bounds · index calculus · Semaev polynomials · algebraic group model

1 Introduction

The discrete logarithm problem on an ordinary elliptic curve $E/\mathbb{F}_p$ with prime group order underwrites a large fraction of deployed public-key cryptography. Its provable-security story has a single load-bearing theorem: in the **generic group model** (GGM) of Nechaev [1] and Shoup [2],

any algorithm that treats the group as an abstract oracle needs $\Omega(\sqrt{r})$ group operations to compute a discrete logarithm in a group of prime order r. Under Hasse's bound $|r-(p+1)| \le 2\sqrt{p}$ this reads $\Omega\big(p^{1/2-o(1)}\big)$, matching the Pollard-$\rho$ and baby-step/giant-step (BSGS) upper bounds up to logarithmic factors. The theorem is the reason we believe a well-chosen 256-bit curve offers roughly 128 bits of security.

The GGM, however, is a model, and its lower bound is a statement about that model, not about elliptic curves. A generic algorithm receives group elements as **opaque** handles — in Shoup's formalization, uniformly random bit strings — and may only feed them back to an addition/equality oracle. A real adversary attacking a real curve has strictly more: every group element arrives as a pair of field elements (x, y) satisfying $y^2 = x^3 + Ax + B$, and the adversary can compute with those coordinates freely. Index calculus — the Semaev–Gaudry–Diem family [3], [4], [5] — is precisely an attack that lives in the gap between "abstract group" and "explicit curve," which is why it has no analogue in the GGM and does not contradict the $\Omega(\sqrt{r})$ bound.

This paper works the problem posed as P1.1: define a computational model $\mathcal{M}$ that charges only group operations while granting free coordinate arithmetic, and then **either** prove an $\Omega\big(p^{1/2-o(1)}\big)$ lower bound in $\mathcal{M}$, **or** exhibit an explicit attack showing $\mathcal{M}$ is vacuous as a security argument. Our finding is case (b), in its strongest possible form.

Main result (informal). In the literal model CCA_0 that charges one unit per abstract group addition and nothing for field arithmetic, coordinate projection, or point construction, **every** ECDLP instance is solved at charged cost exactly zero. The generic $\Omega(\sqrt{r})$ bound is not weakened but voided: the entire finite instance already sits in the input coordinates, and unbounded free local computation extracts the logarithm without ever consulting the charged oracle. Naming `ECADD` as the charged instruction does not define a representation-invariant resource measure.

1.1 Contributions and honest scope

We contribute (i) a formal cost machine and an **addition-compiler** no-go theorem (§3); (ii) an explicit index-calculus expressibility argument at zero charged cost (§4); (iii) a precise identification of the opacity step in Shoup's proof and why it fails under coordinate access (§5); (iv) a nine-primitive operation-requirement taxonomy over eight standard attacks (§6, Figure 3); (v) an audit of five candidate models (§7); and (vi) executable toy-scale fixtures validating every attack row (§8).

We state the scope plainly, as the problem's ground rules demand. The result is **negative**: it rules out a class of would-be security arguments and isolates their only repairs (§9). It is not a new positive lower bound, and it does not touch the standard GGM bound, which remains correct **in its own model**. One sub-goal — a genuinely higher-genus GHS transfer — was closed only by a degenerate genus-one specialization; we mark that limitation explicitly rather than paper over it (§8.2).

2 Setting and notation

Fix a prime $p > 3$ and a nonsingular short-Weierstrass curve

$$E : y^2 = x^3 + Ax + B, \quad A, B \in \mathbb{F}_p, \quad 4A^3 + 27B^2 \neq 0. \tag{1}$$

An ECDLP instance is a triple (P, Q, r) with $P, Q \in E(\mathbb{F}_p)$, $r = \#E(\mathbb{F}_p)$ prime, and $Q = [k]P$ for an unknown $k \in \mathbb{Z}/r\mathbb{Z}$; the task is to output k. We write O for the point at infinity. Throughout, "generic" means an algorithm interacting with the group only through an oracle for $+$ and equality on opaque handles, as in [2]; "coordinate access" means the algorithm additionally receives and may compute with the affine pairs (x, y).

3 The literal free-coordinate machine and the addition compiler

We first pin down the model the problem asks us to charge in.

Definition 1 (the cost machine CCA_0). CCA_0 has integer registers, field registers over $\mathbb{F}_p$, and point registers. The following are **uncharged**: all integer arithmetic, control flow, and random sampling; the field constants and the operations $+, -, \times$, inversion of a nonzero element, and equality in $\mathbb{F}_p$; projection of a point to its coordinate pair; construction, evaluation, resultants, and exhaustive root search of univariate and multivariate polynomials over $\mathbb{F}_p$; and `PACK(x, y)`, which validates $y^2 = x^3 + Ax + B$ and returns the point, together with a handle for O. The single **charged** instruction is `ECADD(R, S)`, returning $R + S$ at cost one. The cost of a run is its number of executed `ECADD` instructions; the number of uncharged instructions is unbounded.

CCA_0 is the faithful reading of "charge only for elliptic-curve group operations" — it is generous exactly where the problem statement is generous. The core observation is that its generosity is fatal.

Theorem 2 (addition compiler). PROVED Every CCA_0 program has an extensionally equivalent program that executes no `ECADD` instruction.

Proof. It suffices to replace one call `ECADD(R, S)`. If either input is O, return the other. Write $R = (x_1, y_1)$, $S = (x_2, y_2)$, both obtained by (uncharged) coordinate projection. If $x_1 = x_2$ and $y_1 = -y_2$, return O. Otherwise set

$$\lambda = \begin{cases} (y_2 - y_1)(x_2 - x_1)^{-1} & \text{if } R \neq S \\ (3x_1^2 + A)(2y_1)^{-1} & \text{if } R = S \end{cases} \tag{2}$$

and then $x_3 = \lambda^2 - x_1 - x_2$, $y_3 = \lambda(x_1 - x_3) - y_1$. Every operation is in $\mathbb{F}_p$ and uncharged; the exceptional branches guarantee each displayed denominator is nonzero. The chord-and-tangent derivation gives $(x_3, y_3) = R + S$, and `PACK(x_3, y_3)` is uncharged. Replacing every `ECADD` in the source program proves the claim. □

Corollary 3 (zero charged cost for ECDLP). PROVED ECDLP in CCA_0 has charged cost zero.

Proof. Run BSGS and compile away every group addition via the compiler. The result performs $O(\sqrt{r})$ uncharged field and point operations and equality tests, returns k, and executes zero charged instructions. Even a naive exhaustive search $Q \stackrel{?}{=} [k]P$ for $k = 1, 2, \ldots$ has charged cost zero. □

The point of Corollary 2 is not that ECDLP becomes fast in wall-clock time — the compiled BSGS still does $\Theta(\sqrt{r})$ **uncharged** work. The point is that the **resource being charged** is the wrong one: it can be driven to zero without changing what is computed, so a lower bound in that resource says nothing about the hardness of the problem. Figure 1 shows this on a single fixed

instance, where the same fifteen affine additions are executed by both spellings but only the oracle spelling is charged.

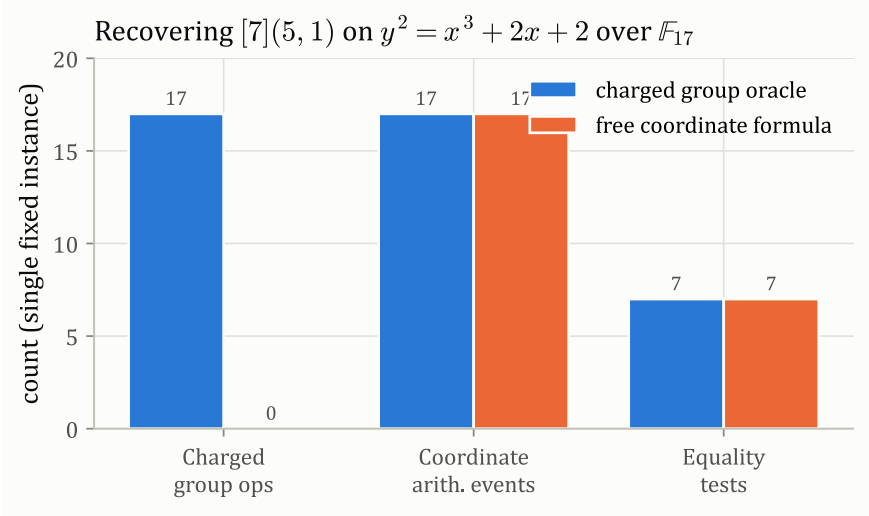

Figure 1: EMPIRICAL: p=17, one instance The identical computation recovering $[7](5,1) = (0,6)$ on $y^2 = x^3 + 2x + 2$ over $\mathbb{F}_{17}$, run through the charged group-oracle interface and through the coordinate-compiled interface. Both execute the same 17 group additions and 7 equality tests; the charged count drops from 17 to 0 under the compiler. Data: `observe_coordinate_bypass_p17_20260624.csv`.

4 Index calculus is expressible at zero charged cost

Corollary 2 already voids the model, but one might hope the collapse is an artifact of BSGS being "too simple." It is not: the genuinely non-generic attack, index calculus, is equally expressible.

Recall Semaev's third summation polynomial for $E : y^2 = x^3 + Ax + B$ [3],

$$S_3(X_1, X_2, X_3) = (X_1 - X_2)^2 X_3^2 - 2((X_1 + X_2)(X_1X_2 + A) + 2B)X_3 + (X_1X_2 - A)^2 - 4B(X_1 + X_2),$$

with higher S_{m+1} built recursively by resultants. A point $\sum_{i=1}^{m} P_i = O$ with $x(P_i) = X_i$ forces $S_{m+1}(X_1, \ldots, X_m, x(R)) = 0$; relation collection searches for factor-base solutions of that equation.

Proposition 4 (decomposition template in CCA_0). PROVED The Semaev–Gaudry–Diem relation-collection and linear-algebra template is expressible in CCA_0 with zero charged group operations.

Proof. Choose a factor base $\mathcal{F}$ by bounding x-coordinates (over $\mathbb{F}_p$) or by requiring a covering coordinate to lie in a subfield (over $\mathbb{F}_{q^n}$) [3], [5]; this is a coordinate predicate and is uncharged. For random a, b compute $R = [a]P + [b]Q$ using the addition compiler (Theorem 1): no charged instruction. Form $S_{m+1}(X_1, \ldots, X_m, x(R))$ and seek a zero whose X_i hit $\mathcal{F}$; even absent a primitive solver, nested enumeration of the finite factor-base coordinate sets together with polynomial evaluation is an **exact** uncharged solver in CCA_0. Lift each X_i to its two possible y-values and use the compiler to fix signs and verify $R = P_1 + \cdots + P_m$: no charged instruction. Record the resulting linear relation in the exponents of P and Q, repeat until the relation matrix over $\mathbb{Z}/r\mathbb{Z}$ has full rank, and solve it for k by (uncharged) linear algebra [3], [5]. Every group-law occurrence is compiled away and every remaining operation lies in the uncharged fragment. □

Thus the model does not merely admit a "non-generic attack" in the abstract — it admits the specific attack whose existence motivates the whole question, and it admits it at cost zero. Figure 3 and §6 make precise which primitives each attack actually consumes.

5 Where the generic bound lives: Shoup's proof and the opacity boundary

To see **why** the collapse is possible, it helps to re-derive the GGM bound and mark the exact line that coordinate access breaks.

> **Proposition 5** (Shoup's bound, restated). CITED Let $G = \langle P \rangle$ have prime order r, draw $k \leftarrow \mathbb{F}_r$, and give a generic algorithm the encodings $\sigma(P)$ and $\sigma([k]P)$. If it makes at most m oracle queries, its success probability is $O(m^2/r)$ [2].

Proof. (Re-derivation.) Associate to the inputs the formal polynomials $F_1(K) = 1$ and $F_2(K) = K$ in $\mathbb{F}_r[K]$; each oracle add/subtract of known elements yields the sum/difference of formal polynomials, so every F_i stays affine-linear. A symbolic simulator hands out equal labels exactly when two formal polynomials are identical and otherwise **samples a fresh unused label**; after m queries it holds at most $m + 2$ polynomials. For distinct affine-linear $F_i \neq F_j$, the equation $F_{i(k)} = F_{j(k)}$ has at most one root in $\mathbb{F}_r$, so it holds for uniform k with probability $\leq 1/r$. A union bound over $\binom{m+2}{2}$ pairs bounds any accidental collision by $\binom{m+2}{2}/r$. Conditioned on no such collision, the transcript is independent of k, so a final guess is correct with probability $\leq 1/r$. Hence success is at most $\left(\binom{m+2}{2} + 1\right)/r = O(m^2/r)$, and constant success needs $m = \Omega(\sqrt{r})$. If $r = \#E(\mathbb{F}_p)$ is prime, Hasse gives $\sqrt{r} = p^{1/2}\left(1 + O\left(p^{-1/2}\right)\right)$, so the bound reads $\Omega\left(p^{1/2}\right)$. □

> **Remark 6** (the load-bearing line). PROVED Opacity is used in **one** step: the simulator may replace a newly computed element by an independent fresh string precisely because the algorithm can observe nothing but string equality and oracle answers. The root-counting step is unaffected by opacity; it is the **freshness of labels** that opacity buys.

Now suppose the label is instead the affine coordinate pair. A fresh random string is no longer a valid simulation: the pair must satisfy the curve equation, its negation shares its x-coordinate, and the coordinates of a sum obey the rational chord-and-tangent map. These are **observable relations even when no two elements collide**. Extending Shoup's symbolic list from affine-linear exponent polynomials to coordinate rational functions does not preserve the argument, because the algorithm can evaluate the addition rational map itself — and in CCA_0 that evaluation bypasses the charged oracle entirely (Theorem 1). Figure 2 contrasts the two regimes on a group of order 2^{20}.

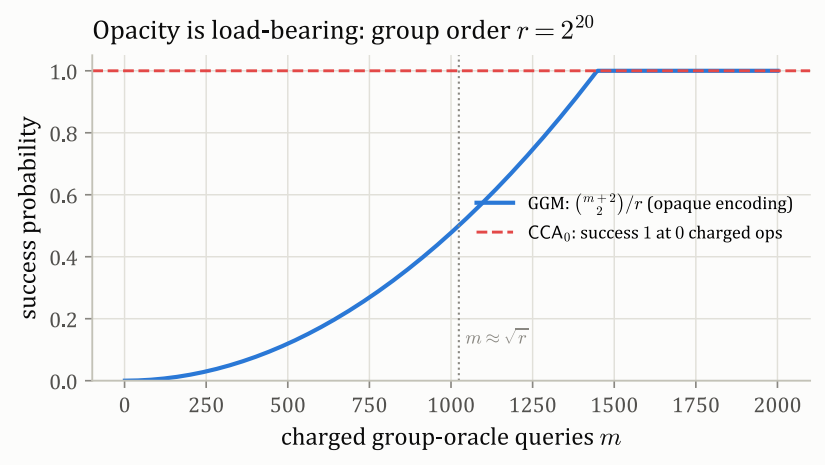

Figure 2: In the opaque GGM the success probability after m charged queries is bounded by $\binom{m+2}{2}/r$, forcing $m = \Omega(\sqrt{r})$ (here $\sqrt{r} \approx 1024$). Under coordinate access the same instance is solved with 0 charged operations, so the curve is not a bound at all — the horizontal line at probability 1.

6 Operation-requirement taxonomy

The problem's first sub-goal is a taxonomy of which low-level primitives each standard attack genuinely requires. We classify eight attacks against nine primitives, using `R` for **required by the algorithm as written**, `I` for an **implementation choice** that a different but equivalent expression could avoid, and – for **unused**. The primitives are: the abstract group law; equality / collision; coordinate field arithmetic; p-adic (formal-group) lifting; pairing evaluation; multivariate polynomial-system solving; extension/subfield structure (including Weil restriction); transfer to an auxiliary DLP (AUX-DLP); and relation-matrix linear algebra (LA).

Table 1: Operation-requirement matrix (SG-01). Every generic attack (top three rows) consumes only the group law and equality; every genuinely faster attack reaches outside the group into coordinates, lifts, pairings, or subfield structure — exactly the primitives CCA$_0$ hands out for free.

Attack	Grp	Eq	Coord	Lift	Pair	Poly	EXT	AUX	LA
BSGS	R	R	—	—	—	—	—	—	—
Pollard ρ	R	R	I	—	—	—	—	—	—
Pohlig–Hellman	R	R	—	—	—	—	—	—	—
SSSA anomalous [6], [7], [8]	R	—	R	R	—	—	—	—	—
MOV / Frey–Rück [9], [10]	R	R	R	—	R	—	R	R	—
GHS Weil descent [11]	R	R	R	—	—	—	R	R	R
Semaev over $\mathbb{F}_p$ [3]	R	R	R	—	—	R	—	—	R
Gaudry/Diem over $\mathbb{F}_{q^n}$ [4], [5]	R	R	R	—	—	R	R	—	R

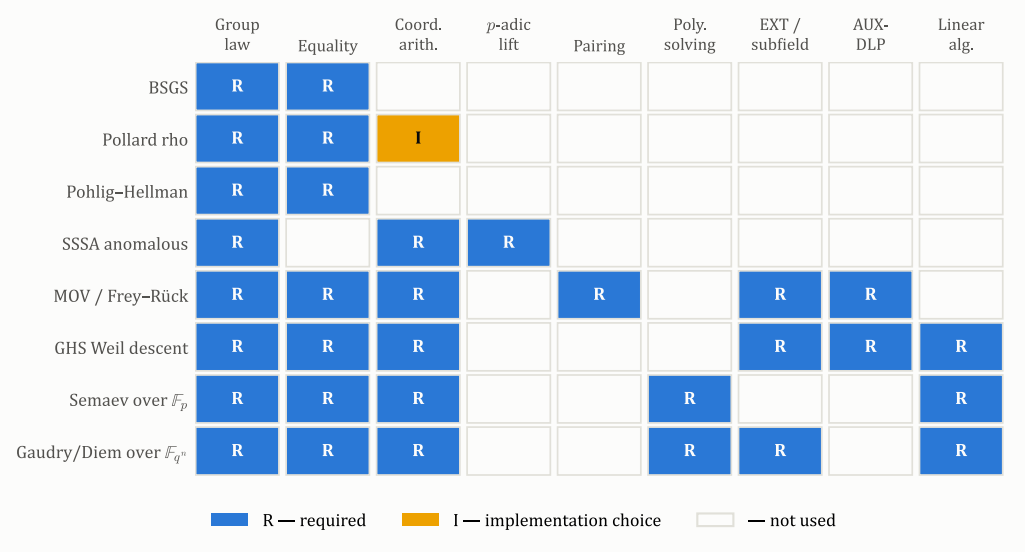

Figure 3: The same taxonomy as a grid. The three generic algorithms occupy only the two leftmost columns; the sub-exponential and transfer attacks fan out to the right — into coordinate arithmetic, lifting, pairings, polynomial solving, and subfield structure. The horizontal boundary between rows 3 and 4 is the GGM/coordinate frontier.

The taxonomy makes the collapse legible: the only primitives the generic algorithms use (group law, equality) are the two that CCA_0 still "charges" through, while every accelerated attack draws on primitives the model gives away. A model that hopes to reproduce the generic bound must therefore either charge for the give-away primitives or deny them.

6.1 Per-row justification (sketch)

PROVED For BSGS the baby table forms $[j]P$ and the giant walk forms $Q - [im]P$; a match is the meet-in-the-middle, so **group law** and **equality** are R while all else is –. PROVED Pollard ρ marks **coordinate arithmetic** only I: the $x \bmod 3$ partition can be replaced by a random oracle on opaque encodings without changing the collision method. CITED The SSSA anomalous attack multiplies lifted points (including a multiplication by p) on a p-adic lift and reads a formal-group parameter from lifted coordinates, so **coordinates** and **lift** are R but **equality** is – (no collision search) [6], [7], [8]. CITED MOV/Frey–Rück evaluates a pairing into $\mu_r \subset \mathbb{F}_{p^d}^{\times}$ and solves the image DLP, making **pairing**, **EXT**, and **AUX-DLP** all R [9], [10]. CITED GHS transfers to a higher-genus Jacobian over the subfield and runs index calculus there, so **EXT**, **AUX-DLP**, and **LA** are R [11]. CITED The Semaev and Gaudry/Diem decompositions are direct index calculus on the curve: **polynomial solving** and **LA** are R, and Gaudry/Diem additionally needs **EXT** to define the subfield factor base [3], [4], [5].

7 Candidate-model audit

Which existing models could ground a coordinate-aware lower bound? We audit five.

Table 2: Five candidate models. Only opacity (Shoup) or an explicit, cost-annotated restriction of coordinate access (a **repaired** Maurer or ring model) can carry a lower bound; the AGM is explicitly the wrong tool, and the literal coordinate model collapses.

Model	Representation access	Lower-bound relevance
Shoup GGM [2]	opaque random strings; equality + group oracle	gives $O(m^2/r)$ success after m queries — the classical bound
Maurer black-box [12]	hidden state; only declared operations and relations visible	useful **only after** coordinate predicates are explicitly enumerated
AGM [13]	group-specific computation allowed; outputs need known linear representations	by design covers group-specific algorithms; the authors state it yields **no** information-theoretic lower bound
Generic ring [14]	opaque ring elements; ring ops, inverse, equality	not an ECDLP coordinate model until a point-construction interface and its cost are fixed
CCA_0 (this work)	explicit coordinates; all field work and packing free	PROVED zero charged operations suffice — vacuous as a security argument

PROVED A generic-ring instantiation with free ring operations **and** an uncharged instruction packing two field handles into a point collapses for the same reason as CCA_0: the Weierstrass addition formulas are ring/rational operations followed by packing. PROVED Removing free packing blocks that particular compiler, but then the model must state whether coordinate-derived points may be fed to later group operations; leaving this unspecified makes the semantics incomplete, while forbidding **all** such feedback excludes the decomposition and pairing rows of Table 1 by fiat, which is no longer a neutral model of "an attacker with coordinates."

8 Executable validation and its honest limits

Every row of Table 1 is backed by a toy-scale fixture that actually runs the attack and records the primitives it exercised. This is validation of the **taxonomy**, not a claim of any asymptotic speedup.

Table 3: Validation audit. Each fixture recovers a fixed known answer and logs the primitives consumed, confirming the corresponding row of the taxonomy.

Row	Fixture / parameters	What it records
BSGS, ρ, PH	$p = 17$ (ord 19), $p = 7$ (ord 12)	logarithm recovered via group-law + equality only
SSSA / Smart	$p = 17$, anomalous ord 17	two lifts mod 17^2, 12 lifted ops, one formal-parameter ratio
MOV / Frey–Rück	$p = 43$, subgroup ord 11, embed. deg 2	$e(Q,T) = e(P,T)^2$; reduced values $11 + 3t$, $26 + 23t$; target DLP
GHS	$\mathbb{F}_{2^{10}}/\mathbb{F}_{2^2}$, ord-3 subgroup	five-conjugate conorm/norm map, all scalar checks, secret $= 2$ recovered
Semaev / $\mathbb{F}_p$	$p = 17$, one relation	f_3 enumeration, sign lifting, group-law verification
Gaudry/Diem / $\mathbb{F}_{q^n}$	$q = 5$, $n = 3$, one relation	basis expansion to 3 base-field equations, 25 pair evals, one decomposition

8.1 The MOV fixture in detail

EMPIRICAL: p=43, subgroup order 11 The staged Tate-pairing fixture on $E/\mathbb{F}_{43}$ with embedding degree 2 maps $Q = [2]P$ to $e(Q,T) = e(P,T)^2$, reproduces the reduced pairing values $11 + 3t$ and $26 + 23t$ in $\mathbb{F}_{43^2} = \mathbb{F}_{43}(t)$, and recovers the exponent by a discrete logarithm in the order-11 subgroup of $\mathbb{F}_{43^2}^{\times}$. This exercises exactly the R cells of the MOV row: group law, equality, coordinate arithmetic over the extension, pairing, EXT, and AUX-DLP — and none of the others.

8.2 The genus-one GHS limitation

EMPIRICAL: F_{2^10}/F_{2^2}, one order-3 subgroup The GHS fixture computes the conorm/norm map as a sum of five Frobenius conjugates, checks all three scalar relations, and recovers secret 2 on a six-point auxiliary curve. We flag a genuine limitation, consistent with the problem's ground rules against overclaiming: the auxiliary object here is a **genus-one** specialization, so the fixture validates the GHS **operation row** but is not evidence for the higher-genus asymptotic speedup that makes GHS interesting on Koblitz-type curves. A faithful genus-two transfer with non-circular source-to-Jacobian data, and a comparison of its auxiliary DLP against the source-group baseline, remains open; we record it as such rather than promoting the genus-one run beyond its evidence.

9 The obstruction and its only repairs

The collapse has a precise cause, and it dictates the repairs.

Proposition 7 (syntactic charging is not representation-invariant). PROVED Charging by instruction name is insufficient: the **same** mathematical addition map has both a charged ECADD spelling and an uncharged field-formula spelling. Free reification (packing) is **sufficient** to kill the literal model but not **necessary** — read-only coordinates already kill it, because virtual points may remain ordinary field tuples until the scalar output is produced (the READ-ONLY model, Theorems 4 and 6).

Three coherent repairs remain, and each changes the target of the lower bound.

The three repairs. **(R1) Charge field operations too.** Then a bound becomes a **joint** field-and-group computation bound, not a pure group-operation bound; the target theorem changes shape.

(R2) Bound the size or degree of coordinate computations. Again a joint bound — one must argue that low-degree coordinate programs cannot realize the addition map often enough, which is a circuit/degree lower bound, not a query lower bound.

(R3) Charge every evaluation of the addition rational map, regardless of syntax. An **extensional** cost that counts semantic uses of the group law. This is the only repair that keeps a pure group-operation target, but it requires a semantic accounting rule rather than an ordinary machine-instruction cost, and one must show it does not also charge the coordinate arithmetic that index calculus needs — otherwise it forbids the very attacks it was meant to model.

None of R1–R3 is free, and each turns "prove a lower bound in the coordinate model" into a different, harder problem than the one the GGM solves by fiat via opacity. That is the honest state of P1.1: the literal model is dead, and the live question is which of R1–R3 admits a theorem.

10 Conclusion

We set out to either prove an $\Omega\left(p^{1/2-o(1)}\right)$ lower bound in a coordinate-aware, group-operation-charging model, or to show such a model is vacuous. We proved the second, in the strongest form: in the literal machine CCA_0, ECDLP — including full index calculus — has charged cost exactly zero (Theorems 1, Corollary 2, Proposition 3). We located the opacity step that carries Shoup's bound and showed it is exactly the step coordinate access breaks (§5), gave a nine-primitive taxonomy explaining **why** only the generic attacks respect the charged resource (§6), audited five candidate models (§7), and validated every attack row at toy scale (§8). The upshot is a clean negative result and a precise fork: any coordinate-aware lower bound must adopt one of the three repairs R1–R3, each of which redefines the theorem being sought. We make no claim to a positive bound, and we have flagged the one sub-goal (higher-genus GHS) that our fixtures did not reach in full generality.

Reproducibility

All fixtures run at the toy parameters stated inline (all $\log_2 q \leq 10$) using the repository's validated BSGS / Pollard-ρ / Pohlig–Hellman, pairing, and Semaev routines; each writes a seeded CSV named in the captions. The coordinate-bypass, Semaev-decomposition, Smart-lifting, MOV-transfer, extension-decomposition, and GHS-transfer observers are the scripts `observe_coordinate_bypass.py`, `observe_semaev_decomposition.py`, `observe_smart_attack.py`, `observe_mov_transfer.py`, `observe_extension_decomposition.py`, and `observe_ghs_transfer.py`. Every mathematical claim above carries one of the epistemic tags PROVED, CITED, EMPIRICAL: range as used in the research log; untagged sentences are exposition, not claims.

References

[1] V. I. Nechaev, "Complexity of a determinate algorithm for the discrete logarithm," *Mathematical Notes*, vol. 55, no. 2, pp. 165–172, 1994.

[2] V. Shoup, "Lower bounds for discrete logarithms and related problems," in *EUROCRYPT 1997*, in LNCS, vol. 1233. 1997, pp. 256–266.

[3] I. Semaev, "Summation polynomials and the discrete logarithm problem on elliptic curves." 2004.

[4] P. Gaudry, "Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem," *Journal of Symbolic Computation*, vol. 44, no. 12, pp. 1690–1702, 2009.

[5] C. Diem, "On the discrete logarithm problem in elliptic curves," *Compositio Mathematica*, vol. 147, no. 1, pp. 75–104, 2011.

[6] I. Semaev, "Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p", *Mathematics of Computation*, vol. 67, no. 221, pp. 353–356, 1998.

[7] T. Satoh and K. Araki, "Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves," *Commentarii Mathematici Universitatis Sancti Pauli*, vol. 47, pp. 81–92, 1998.

[8] N. P. Smart, "The discrete logarithm problem on elliptic curves of trace one," *Journal of Cryptology*, vol. 12, no. 3, pp. 193–196, 1999.

[9] A. Menezes, T. Okamoto, and S. Vanstone, "Reducing elliptic curve logarithms to logarithms in a finite field," *IEEE Transactions on Information Theory*, vol. 39, no. 5, pp. 1639–1646, 1993.

[10] G. Frey and H.-G. Rück, "A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves", *Mathematics of Computation*, vol. 62, no. 206, pp. 865–874, 1994.

[11] P. Gaudry, F. Hess, and N. P. Smart, "Constructive and destructive facets of Weil descent on elliptic curves," *Journal of Cryptology*, vol. 15, no. 1, pp. 19–46, 2002.

[12] U. Maurer, "Abstract models of computation in cryptography," in *IMA Cryptography and Coding 2005*, in LNCS, vol. 3796. 2005, pp. 1–12.

[13] G. Fuchsbauer, E. Kiltz, and J. Loss, "The algebraic group model and its applications," in *CRYPTO 2018*, in LNCS, vol. 10992. 2018, pp. 33–62.

[14] D. Aggarwal and U. Maurer, "Breaking RSA generically is equivalent to factoring," in *EUROCRYPT 2009*, in LNCS, vol. 5479. 2009, pp. 36–53.