On the Security of Constraint-Friendly Map-to-Curve Relations

Youssef El Housni
youssef.elhousni@consensys.net
Consensys, Linea

Benedikt Bünz
bb@nyu.edu
New York University

Abstract. Standard hash-to-curve constructions first hash the message to a field element through a cryptographic hash-to-field step, then map this field element to an elliptic-curve point. Inside constraint systems, this inner cryptographic hash is often the dominant cost. Groth, Malvai, Miller and Zhang (Asiacrypt 2025) introduced *constraint-friendly map-to-elliptic-curve-group relations* that bypass this step, achieving substantial reductions in circuit size. Their security proof works in the Elliptic Curve Generic Group Model (EC-GGM).
We identify three gaps. First, the security bound in terms of the message space size M and tweak bound T is not explicitly analyzed, and the bounds stated for the concrete instantiations are loose. Second, the EC-GGM does not capture the algebraic structure of most deployed curves; we exhibit a concrete signature forgery using the parameters claimed secure. Third, the construction requires a congruence condition on the field that is not satisfied by all deployed curves; we extend it to any field, although witness generation remains practical only for fields of low 2-adicity.
As a countermeasure we propose a *y-increment* variant that, on the $j = 0$ curves used in practice, neutralizes the algebraic attack, removes the field restriction in practice, and preserves a comparable constraint count. We implement and benchmark both constructions in the open-source `gnark` (Go) library; the attack is additionally demonstrated via a self-contained SageMath simulation and confirmed at the circuit level against the authors' own `Noir` (Rust) implementation.

Keywords: map-to-curve relations · elliptic curves · generic group model · BLS signatures · zk-SNARKs

1 Introduction

Hashing to elliptic curve groups underlies two cryptographic primitives: *multi-set hashing* [14], used for memory checking, and *BLS signatures* [10], used for transactions and consensus in blockchain protocols. When proving these operations using zk-SNARKs, e.g. in zero-knowledge Virtual Machines (zkVM) or for signature aggregation, we must express them as low-constraint circuits to be practical. Standard hash-to-curve constructions combine an inner cryptographic

hash (e.g. SHA-256, Poseidon [19]) with an outer map-to-curve, making them a bottleneck at scale, because of the inner hash cost. In the original BLS setting, this inner hash first maps the message to a field element, $H' : \{0,1\}^* \to \mathbb{F}_q$, which is then used to derive a candidate abscissa x; if this candidate does not define a curve point, the procedure tries another candidate.

Groth, Malvai, Miller and Zhang [20] (GMMZ) proposed bypassing the inner hash entirely via a *map-to-curve relation*. Their construction sets $x = m \cdot T + k$ for a message m from a message space of size M, a small tweak $k \in [0, T)$, and a fixed tweak bound T determined by the target failure probability. It then checks whether x lands on the curve $E/\mathbb{F}_q$, and encodes the non-deterministic search for a valid k as a witness. The relevant encoding range has size $M \cdot T$, which is the parameter product underlying the security bound. The resulting verifier circuit costs a few constraints, with security analyzed in the Elliptic Curve Generic Group Model (EC-GGM) [21].

Constraint-friendliness. Throughout this paper, design choices are driven by the in-circuit cost model, which inverts the familiar CPU cost ordering: field multiplications cost one constraint each, additions and multiplications by public constants are free, while any operation that inspects the *bits* of a field element — range checks, integer comparisons, and a fortiori cryptographic hashing — requires a bit decomposition costing hundreds of constraints natively and thousands under non-native (emulated) field arithmetic. This is why the constructions studied here replace hashing with coordinate encodings, and why canonical choices are enforced through residuosity conditions (a few multiplications) rather than through seemingly simpler alternatives such as lexicographic comparison of y and $q - y$ (a full bit decomposition; see Section 4).

Our contributions. We identify three gaps in the GMMZ construction, leading to an existential forgery in the signature application, and address each.

1. **The security bound is loose.** GMMZ does not state a clear security threshold, but their concrete parameter choices in [20, Sections 5.4 and 6.4] implicitly treat $M \cdot T \ll q$ as the governing security condition, where M is the size of the message space $\mathcal{M}$. We prove that $M \cdot T < \sqrt{q}$ is the threshold below which the construction achieves discrete-logarithm-problem (DLP) level security.
2. **Field restriction.** The construction requires $q \equiv 3 \pmod 4$ for canonical square-root extraction. GMMZ requires this for point inverse exclusion. We replace the quadratic residuosity with *higher power residuosity*, extending the construction to any field, at the cost of a few additional constraints when $q \equiv 1 \pmod 4$. However, witness generation becomes impractical for highly 2-adic fields.
3. **The EC-GGM misses algebraic structure.** Most deployed curves have a j-invariant zero and thus carry a non-trivial automorphism that the EC-GGM does not capture. Specifically, for any such curve there are constants ω and λ such that, on the cryptographic subgroup, $P_1 = (x, y)$ and $P_2 = (\omega \cdot$

x, y) satisfy the known scalar relation $P_2 = [\lambda]P_1$. We exploit this via a short-vector lattice computation, yielding a BLS signature forgery in $O(\log^2 q)$ time (or $O(1)$ for parameterized families such as Barreto–Naehrig (BN) [7] and Barreto–Lynn–Scott (BLS) [6] curves) when $M \cdot T \gtrsim \sqrt{q}$, where $\gtrsim$ means "greater than or approximately equal to". We exhibit a concrete forgery with the GMMZ proposed parameters.

As a countermeasure we propose the *y-increment* construction (Section 5): encoding the message as a y-coordinate and recovering x. For $j = 0$ curves, the automorphism that breaks the GMMZ x-increment acts as the identity on y-coordinates, neutralizing the attack. Inverse exclusion becomes free, lifting the field restriction. Both constructions are implemented and benchmarked in `gnark` [11]; the attack is additionally demonstrated via a SageMath simulation and at the circuit level against GMMZ's own `Noir` circuits (Section 6). All source code is available at [16]:
`https://github.com/yelhousni/map-to-curve`.

Organization. Section 2 recalls background on BLS signatures, elliptic curves, root extraction, and the GMMZ construction. Section 3 analyzes the algebraic attack on $j = 0$ curves and tightens the security bound. Section 4 addresses the field restriction. Section 5 introduces the y-increment construction. Section 6 describes the implementation. Section 7 concludes.

2 Preliminaries

2.1 BLS signatures

Let $E/\mathbb{F}_q$ be a pairing-friendly curve with two prime-order subgroups $\mathbb{G}_1 \subset E(\mathbb{F}_q)$ and $\mathbb{G}_2 \subset E(\mathbb{F}_{q^k})$ of order p, generators $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, and a type-3 bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. A BLS signature [10] with hash $H : \{0,1\}^* \to \mathbb{G}_1$ works as follows. In classical instantiations, H denotes the full hash-to-curve map, including the inner hash-to-field step discussed in the introduction.

- KeyGen: sample $\mathsf{sk} \leftarrow \mathbb{Z}/p\mathbb{Z}$, set $\mathsf{vk} = [\mathsf{sk}]\, g_2 \in \mathbb{G}_2$.
- $\mathsf{Sign}(m)$: output $\sigma = [\mathsf{sk}]\, H(m) \in \mathbb{G}_1$.
- $\mathsf{Verify}(m, \sigma)$: accept iff $e(\sigma,\, g_2) = e(H(m),\, \mathsf{vk})$.

Security (existential unforgeability under adaptive chosen-message attack) reduces to the co-computational Diffie-Hellman problem relative to the pairing e, assuming H is modeled as a random oracle. In the GMMZ setting, H is replaced by the x-increment map-to-curve relation.

2.2 Elliptic curves

Let $E : y^2 = x^3 + ax + b$ be an elliptic curve over $\mathbb{F}_q$ with $q > 3$. Let $\mathbb{G}$ be a subgroup of $E(\mathbb{F}_q)$ of prime order p. We write $[k]P$ for scalar multiplication and $\mathcal{O}$ for the point at infinity. The j-invariant is $j(E) = 1728 \cdot 4a^3/(4a^3 + 27b^2)$.

The endomorphism ring $\mathrm{End}(E)$ consists of all rational maps $E \to E$ preserving the group law. The automorphism group $\mathrm{Aut}(E) \subset \mathrm{End}(E)$ is the invertible part:

- $j \neq 0, 1728$, curves $y^2 = x^3 + ax + b$ with $a \cdot b \neq 0$: $\mathrm{Aut}(E) = \{\pm 1\}$.
- $j = 0$, curves $y^2 = x^3 + b$ with $a = 0$: $\mathrm{Aut}(E) \cong \mathbb{Z}/6\mathbb{Z}$. When $q \equiv 1 \pmod 3$ the field $\mathbb{F}_q$ contains a primitive cube root of unity ω, and the *order-3 automorphism*
$$\phi : (x, y) \longmapsto (\omega x,\ y)$$
acts as scalar multiplication on every ϕ-stable prime-order subgroup: for the cryptographic subgroup considered below, $\phi(P) = [\lambda]P$ for a known scalar λ satisfying $\lambda^2 + \lambda + 1 \equiv 0 \pmod p$, where p is the subgroup order. The endomorphism ring is $\mathrm{End}(E) \cong \mathbb{Z}[\omega]$.
- $j = 1728$, curves $y^2 = x^3 + ax$ with $b = 0$: $\mathrm{Aut}(E) \cong \mathbb{Z}/4\mathbb{Z}$ when $q \equiv 1 \pmod 4$. The order-4 automorphism is $\psi : (x, y) \mapsto (-x,\ iy)$ where $i^2 = -1$ in $\mathbb{F}_q$.

Endomorphism action on coordinates. The *degree* of an endomorphism $\alpha \in \mathrm{End}(E)$ is $\deg(\alpha) = |\ker \alpha|$ (counted over $\bar{\mathbb{F}}_q$). Scalar multiplication satisfies $\deg([k]) = k^2$; automorphisms are precisely the degree-1 endomorphisms. An endomorphism of degree d acts on the x-coordinate via a rational function of degree d: for a generic point P, the value $x(\alpha(P))$ is a rational function of $x(P)$ of degree d. On short Weierstrass curves, degree-1 maps further reduce to scalings: every automorphism has x-map of the form $x \mapsto cx$ for some $c \in \mathbb{F}_q^*$ (for $j = 0$, $c = \omega$; for $j = 1728$, $c = -1$). Endomorphisms of degree $d \geq 2$ have x-maps of degree ≥ 2 and are not linear.

2.3 Root extraction in $\mathbb{F}_q$ and $\mathbb{F}_{q^2}$

Root extraction is the computational core of both constructions studied in this paper: the x-increment prover recovers y from a candidate x via a square root (and certifies its canonicality via a residuosity witness, Sections 2.5 and 4), while our y-increment prover recovers x from a candidate y via a cube root or a cubic solve (Sections 2.4 and 5), the latter requiring square and cube roots in $\mathbb{F}_{q^2}$. This subsection collects the algorithms and closed-form exponentiations used throughout.

Square roots in $\mathbb{F}_q$. A uniformly random element $a \in \mathbb{F}_q^*$ is a quadratic residue (QR) with probability exactly $1/2$. Write $q - 1 = 2^s \cdot d$ with d odd; the 2-adic valuation $s = v_2(q-1)$ — throughout, $v_\ell(n)$ denotes the largest integer e such that $\ell^e \mid n$ — determines the algorithm. When $q \equiv 3 \pmod 4$ ($s = 1$), exactly one of $\{a^{(q+1)/4},\ -a^{(q+1)/4}\}$ is a QR (since -1 is a quadratic non-residue QNR), giving a canonical square root with a single exponentiation. When $q \equiv 1 \pmod 4$ ($s \geq 2$), -1 is a QR and $y, -y$ are simultaneously QR or QNR: no canonical choice exists from residuosity alone. For $s \geq 3$, Tonelli–Shanks iteratively corrects a near-root via a discrete-log computation in the 2-Sylow subgroup.

Cube roots in $\mathbb{F}_q$. Write $q-1 = 3^v \cdot d$ with $\gcd(d,3)=1$. When $q \equiv 2 \pmod 3$ ($v=0$), cube roots are unique. When $q \equiv 1 \pmod 3$ ($v \geq 1$), each cube has exactly three roots $\{x, \omega x, \omega^2 x\}$, the orbit of the order-3 automorphism ϕ. For $v=1$, the cube root reduces to a single exponentiation; for $v=2$, one exponentiation followed by a conditional correction by a ninth root of unity (a single AMM step, see Table 1); for $v \geq 3$, the Adleman–Manders–Miller (AMM) [3] algorithm performs an iterative correction analogous to Tonelli–Shanks.

Table 1 summarises the closed-form exponentiation formulas. Most deployed curves fields have $v_3(q-1) \leq 2$, so AMM is never needed in practice.

Square roots in $\mathbb{F}_{q^2}$. Let $\mathbb{F}_{q^2} = \mathbb{F}_q[u]/(u^2+\beta)$ for a non-square $\beta \in \mathbb{F}_q^*$. Adj and Rodríguez-Henríquez [2], Scott [27, Sec. 6.3] and Aardal et al. [1, Alg. 3, $q \equiv 3 \pmod 4$ only] all reduce $\mathbb{F}_{q^2}$ *square roots* to computations in $\mathbb{F}_q$. The key identity is $x^{q+1} = \mathrm{Nm}(x) \in \mathbb{F}_q$. Since $q+1$ is even, $b = x^{(q+1)/2}$ satisfies $b^2 = \mathrm{Nm}(x) \in \mathbb{F}_q$, which forces $b \in \mathbb{F}_q$ (or $b \in \mathbb{F}_q \cdot u$). The $\mathbb{F}_{q^2}$ square root thus reduces to: compute b via an $\mathbb{F}_q$ exponentiation (using the Frobenius $x^q = \overline{x}$), take an $\mathbb{F}_q$ square root of $\mathrm{Nm}(x)$, and combine.

Cube roots in $\mathbb{F}_{q^2}$. El Housni [22] shows that when $q \equiv 1 \pmod 3$, a cube root of $x \in \mathbb{F}_{q^2}^*$ can be computed entirely in $\mathbb{F}_q$. The key insight is that the algebraic torus $T_2(\mathbb{F}_q) \cong \mathbb{F}_{q^2}^*/\mathbb{F}_q^*$ has order $q+1$ with $\gcd(3, q+1) = 1$, so every torus element has a unique cube root given by raising to the $3^{-1} \bmod (q+1)$ power via the Lucas V-sequence — an operation that stays entirely in $\mathbb{F}_q$. This covers all pairing-friendly primes of practical interest (BN, BLS12, BLS24), which satisfy $q \equiv 1 \pmod 3$ by construction.

2.4 Depressed cubic over $\mathbb{F}_q$

This subsection is used by our y-increment construction (Section 5): after encoding a message as a candidate y-coordinate, the verifier recovers an x-coordinate by solving $x^3 + ax + (b - y^2) = 0$.

For $a \neq 0$, the depressed cubic $x^3 + ax + c = 0$ over $\mathbb{F}_q$ has either zero, one, or three distinct roots in $\mathbb{F}_q$ when $\Delta \neq 0$, or exactly two distinct roots (a double root and a simple root) when $\Delta = 0$ — a triple root would force $a = 0$. Its behavior is controlled by the discriminant $\Delta = -4a^3 - 27c^2$. Compute the Legendre symbol $\left(\frac{\Delta}{q}\right)$ via the binary GCD in $O(\log^2 q)$ bit operations: -1 means one root, 0 means a repeated root, 1 means zero or three roots. In the $\left(\frac{\Delta}{q}\right) = 1$ case, three roots additionally require $-c/2 + D$ to be a cubic residue in $\mathbb{F}_q$ (so that Cardano's formula yields $u \in \mathbb{F}_q$); otherwise the cubic has no roots in $\mathbb{F}_q$.

All three cases fit the same Cardano framework. Let

$$D = \sqrt{\tfrac{c^2}{4} + \tfrac{a^3}{27}}, \qquad u = \sqrt[3]{-\tfrac{c}{2} + D}, \qquad v = -\tfrac{a}{3u},$$

and set $r_0 = u + v$, $r_1 = \omega u + \omega^2 v$, $r_2 = \omega^2 u + \omega v$, where ω is the primitive cube root of unity in $\mathbb{F}_q$.

	Condition	Formula	Examples
$\sqrt{a}$	$q \equiv 3 \pmod 4$	$a^{(q+1)/4}$	BN254 [8], BLS12-381 [12], P-256 [29]
	$q \equiv 5 \pmod 8$	Atkin†	
	general ($v_2 \geq 3$)	Tonelli–Shanks	BLS12-377 [13] (v_2=46)
$\sqrt[3]{a}$	$q \equiv 2 \pmod 3$	$a^{(2q-1)/3}$	(unique root)
	$q \equiv 4 \pmod 9$ (v_3=1)	$a^{(8q-5)/9}$	P-256 [29]
	$q \equiv 7 \pmod 9$ (v_3=1)	$a^{(q+2)/9}$	BLS12-377 [13]
	$q \equiv 10 \pmod{27}$ (v_3=2)	$\eta^j \cdot a^{(2q+7)/27}$	BLS12-381 [12]
	$q \equiv 19 \pmod{27}$ (v_3=2)	$\eta^j \cdot a^{(q+8)/27}$	BN254 [8], Grumpkin [5]
	general ($v_3 \geq 3$)	AMM	(rare)

† $t = (2a)^{(q-5)/8}$, $s = 2at^2$, $\sqrt{a} = at(s-1)$.

Table 1. Exponentiation formulas for square and cube roots in $\mathbb{F}_q$. Cube-root formulas assume a is a cubic residue ($a^{(q-1)/3} = 1$). In the $v_3 = 2$ rows the candidate $z = a^e$ satisfies $z^3 = a \cdot (a^{(q-1)/9})^2$ (resp. $a \cdot a^{(q-1)/9}$), where $a^{(q-1)/9} \in \mu_3$ is *not* necessarily 1 even for a cubic residue a; η is a precomputed primitive *ninth* root of unity and $j \in \{0, 1, 2\}$ is selected from the class $a^{(q-1)/9}$ so that $(\eta^j z)^3 = a$ — one conditional multiplication on top of the exponentiation.

One root (Δ non-square). Whether $D = \sqrt{-\Delta/108}$ lies in $\mathbb{F}_q$ or $\mathbb{F}_{q^2}$ depends on the Legendre symbol $\left(\frac{-\Delta/108}{q}\right) = \left(\frac{-1}{q}\right)\left(\frac{\Delta}{q}\right)\left(\frac{3}{q}\right)$, which is not determined by $q \bmod 4$ alone (e.g. for P-256 [29], $q \equiv 3 \pmod 4$ yet 3 is a non-residue mod q, so $D \notin \mathbb{F}_q$). Rather than case-splitting, compute D uniformly via the $\mathbb{F}_{q^2}$ square-root algorithm of Section 2.3 — if $-\Delta/108$ is a square in $\mathbb{F}_q$, the result lies there; otherwise it is a proper $\mathbb{F}_{q^2}$ element — then compute $u = \sqrt[3]{-c/2 + D}$ in $\mathbb{F}_{q^2}$ via the torus method of Section 2.3. The root $r_0 = u - a/(3u)$ is recovered as an element of $\mathbb{F}_q$. Both algorithms perform all arithmetic in $\mathbb{F}_q$.

Three or zero roots (Δ a nonzero square). Assume $q \equiv 1 \pmod 3$ (as for BN254, BLS12-381, and P-256), so $D \in \mathbb{F}_q$. Test whether $-c/2 + D$ is a cubic residue via $(-c/2 + D)^{(q-1)/3} \stackrel{?}{=} 1$. If not, the cubic has no roots in $\mathbb{F}_q$. Otherwise, compute $u = \sqrt[3]{-c/2 + D} \in \mathbb{F}_q$ via Table 1 and set

$$r_0 = u + v, \qquad r_{1,2} = \frac{-r_0 \pm \sqrt{-3r_0^2 - 4a}}{2};$$

the radicand $-3r_0^2 - 4a$ is then guaranteed to be a square in $\mathbb{F}_q$. All arithmetic stays in $\mathbb{F}_q$.

Repeated root ($\Delta = 0$). $r_0 = 3c/a$ and $r_1 = r_2 = -3c/(2a)$.

x-Increment Constructor

Inputs:

- An elliptic curve $E\colon y^2 = x^3 + ax + b$ over $\mathbb{F}_q$, with $q \equiv 3 \pmod 4$.
- An elliptic curve group $\mathbb{G} \subseteq E(\mathbb{F}_q)$.
- A message $m \in \mathcal{M}$.
- A tweak bound $T \in \mathbb{N}$.

Procedure:

1. Initialize tweak counter $k \leftarrow 0$.
2. While $k < T$:
   (a) Let $x \leftarrow m \cdot T + k$ and $f \leftarrow x^3 + ax + b$.
   (b) If f is a quadratic residue in $\mathbb{F}_q$:
      - Compute $y \leftarrow f^{(q+1)/4}$ and $z \leftarrow y^{(q+1)/4}$.
      - If $(x, y) \in \mathbb{G}$, output (x, y) with witness $(k, z) \in \mathcal{W}$ (witness space).
   (c) Otherwise, increment $k \leftarrow k + 1$.
3. If no valid point is found after T iterations, declare failure.

x-Increment Relation

Parameters:

- A prime-order subgroup $\mathbb{G} \subseteq E(\mathbb{F}_q)$ of an elliptic curve $E(\mathbb{F}_q)\colon y^2 = x^3 + ax + b$, with $q \equiv 3 \pmod 4$.
- A message space $\mathcal{M} \subseteq \mathbb{F}_q$ and a tweak space $[0, T)$ such that $M \cdot T < q$.

Relation $\mathcal{R}_x \subseteq \mathcal{M} \times \mathbb{G} \times \mathcal{W}$:
A triple $(m, (x, y), w)$ where $w = (k, z)$ belongs to $\mathcal{R}_x$ if:

1. $m \in \mathcal{M}, \quad k \in [0, T)$,
2. $x = m \cdot T + k$,
3. $z^2 = y$ (inverse exclusion),
4. $y^2 = x^3 + ax + b$,
5. $(x, y) \in \mathbb{G}$.

Fig. 1. The GMMZ [20] x-increment map-to-curve relation: top, the constructor; bottom, the efficiently constructible relation.

2.5 The x-increment construction

We describe the GMMZ map-to-curve relation [20]. Figure 1 gives the constructor and the corresponding relation.

Key properties.

- **Disjointness.** Distinct messages m_i map to disjoint x-ranges $[m_iT, (m_i{+}1)T)$; the pair (m, k) is recovered uniquely by Euclidean division of the abscissa x by T, namely $x = mT + k$ with $0 \le k < T$.

– **Inverse exclusion.** When $q \equiv 3 \pmod 4$, exactly one of $\{y, -y\}$ is a QR. The witness $z^2 = y$ forces y to be the canonical QR root, preventing substitution of the inverse point $(x, -y)$. This imposes the field restriction $q \equiv 3 \pmod 4$ (see Section 4).
– **Failure probability.** Each candidate x fails the quadratic residuosity test on $f(x) = x^3 + ax + b$ with probability $\approx 1/2$; the probability that all T trials fail is $\approx 2^{-T}$.

Security model. The security proof works in the EC-GGM [21], which models the encoding $\overline{\mathsf{enc}} : \mathbb{F}_q^{\pm} \to \mathbb{G}^{\pm}$ as a random bijection over inversion classes, where the superscript $\pm$ denotes quotienting by sign on field representatives and by point inversion on group elements. Morally, this accounts for the full automorphism group when $j \neq 0, 1728$, but curves with $j = 0$ or $j = 1728$ may have additional automorphisms not represented in the model (see Section 3.3).

3 The Attack

To forge a BLS signature (Section 2.1) on a fresh message m_2 after querying the signing oracle on m_1, an attacker needs two valid encodings $P_1 = H(m_1)$ and $P_2 = H(m_2)$ with a *known scalar relation* $P_2 = [\lambda]P_1$: then $[\lambda]\sigma_1$ is a valid signature on m_2 without knowledge of the secret key. The inner hash in classical BLS prevents an adversary from targeting such a relation through small, structured coordinates. In the GMMZ setting, however, the message itself determines the candidate coordinate range. Since the x-increment construction (Section 2.5) places every valid encoding's x-coordinate in $[0, M \cdot T)$, this reduces to finding two curve points with small x-coordinates related by a known scalar.

Automorphisms (the degree-1 endomorphisms, Section 2.2) are the only endomorphisms with a linear x-map $x \mapsto cx$. This linearity is key: a small $x(P_1)$ maps to $c \cdot x(P_1) \bmod q$, which can also be small. Higher-degree endomorphisms act as rational functions of degree ≥ 2, scattering x-coordinates unpredictably.

3.1 The order-3 automorphism attack on $j = 0$ curves

The automorphism ϕ from Section 2.2 maps $x \mapsto \omega x$ with $\omega \not\equiv \pm 1 \pmod q$, inducing the linear relation $x_2 \equiv \omega x_1 \pmod q$. This congruence constrains only the ratio of the two coordinates modulo q; to search for *small integer* solutions (x_1, x_2) we lift it to the rank-two integer lattice

$$L = \{(a, b) \in \mathbb{Z}^2 : b \equiv \omega a \pmod q\}, \tag{1}$$

which has determinant q. On the relevant prime-order subgroup it acts as $\phi(P) = [\lambda]P$, where $\lambda^2 + \lambda + 1 \equiv 0 \pmod p$ and p is the subgroup order.

Finding a short vector. The ℓ^∞ ball of radius r in $\mathbb{Z}^2$ has volume $4r^2$. By Minkowski's theorem a non-zero vector of L exists inside this ball whenever $4r^2 > 4\det(L) = 4q$, i.e., $r > \sqrt{q}$. In particular there exists $(a, b) \in L \setminus \{0\}$ with $\max(|a|, |b|) \leq \sqrt{q}$. Here and below $|a|$ and $|b|$ denote the absolute values of the centered integer representatives in $[-q/2, q/2]$.

The half-GCD algorithm (a variant of the extended Euclidean algorithm) finds this vector in $O(\log^2 q)$ time. It runs the extended Euclidean algorithm on the pair (q, ω), tracking the sequence of remainders $r_0 = q$, $r_1 = \omega$, $r_2, \ldots$ and cofactors t_i satisfying $r_i \equiv t_i\omega \pmod{q}$. The half-GCD stopping rule returns an index i with $|r_i|, |t_i| \leq \sqrt{q}$; the output (t_i, r_i) is a short vector of L since $r_i \equiv \omega t_i \pmod{q}$.

Alternatively, the short basis can be read off from the Complex Multiplication (CM) structure without lattice reduction, following Smith [28]: since $q \equiv 1 \pmod{3}$, the prime q splits in $\mathbb{Z}[\zeta_3]$ as $q = \pi\bar{\pi}$, and Cornacchia's algorithm computes integers satisfying the Eisenstein norm equation $a^2 + ab + b^2 = q$ directly. Choosing the root of $X^2 + X + 1$ that satisfies $b \equiv \omega a \pmod{q}$ gives (a, b) as the short vector. For curves in parameterized families the Cornacchia factorization is known symbolically: for BN curves, Proposition 1 gives it in $O(1)$ operations in the curve seed u.

When $M \cdot T > \sqrt{q}$, this short vector satisfies $|a|, |b| < M \cdot T$, so both $|a|$ and $|b|$ lie in the encoding range $[0, M \cdot T)$.

More generally, the adversary need not use the single shortest vector. The half-GCD computation, lattice reduction, and the Cornacchia/CM method all produce two independent short vectors $(a_1, b_1), (a_2, b_2) \in L$ (a short basis of the same two-dimensional lattice). Every integer linear combination

$$(a, b) = i(a_1, b_1) + j(a_2, b_2)$$

also lies in L. The adversary can therefore search over small coefficients i, j for a vector that remains in the encoding range and whose decomposition $|a| = m_1T + k_1$, $|b| = m_2T + k_2$ satisfies the source-side canonical-tweak condition needed for the signing query and the validity conditions needed for the forged verification instance below. This is the same linear-combination trick commonly used in discrete-logarithm cryptanalysis [23]. This linear-combination search is still over short lattice vectors and is finite once a coefficient bound is fixed.

Constructing the BLS signature forgery. The adversary proceeds as follows.

1. Compute a short basis $(a_1, b_1), (a_2, b_2)$ of L, using half-GCD, lattice reduction, or Smith's CM/Cornacchia approach. Search over small signed coefficients i, j and set $(a, b) = i(a_1, b_1) + j(a_2, b_2)$. For each such vector, choose an orientation (x_1, x_2, e) with $e \in \{1, 2\}$ such that $x_1, x_2 \in [0, M \cdot T)$ and $x_2 \equiv \omega^e x_1 \pmod{q}$. Decompose

$$x_1 = m_1T + k_1, \qquad x_2 = m_2T + k_2, \qquad 0 \leq k_1, k_2 < T.$$

2. For $i \in \{1, 2\}$, let k_i^* be the canonical tweak selected by the signing oracle on m_i. The signing query can be made on any side s for which $k_s^* = k_s$; if neither

equality holds, the adversary continues the linear-combination search. Query the signing oracle on m_s to obtain $\sigma_s = [\mathsf{sk}]P_s$ with $P_s = (x_s, y_s)$.

3. Let t be the other index. If $s = 1$ then set $\epsilon = e$, and if $s = 2$ then set $\epsilon = 3 - e$; in both cases $P_t = \phi^\epsilon(P_s) = (x_t, y_s) = [\lambda^\epsilon]P_s$. The forger outputs $\sigma_t = [\lambda^\epsilon]\sigma_s$ for the verification instance on m_t with tweak k_t. In the GMMZ relation setting the verifier checks that k_t gives a valid point, but does not require k_t to be the first valid tweak.

Under these conditions, the pair (m_t, σ_t) is a valid BLS forgery:

$$e(\sigma_t, g_2) = e([\lambda^\epsilon]\sigma_s,\, g_2) = e([\lambda^\epsilon][\mathsf{sk}]P_s,\, g_2) = e([\mathsf{sk}]\phi^\epsilon(P_s),\, g_2) = e(P_t,\, \mathsf{vk}).$$

Since ϕ is an automorphism, $P_t = \phi^\epsilon(P_s)$ lies on the curve automatically; no additional QR check is needed.

Scope. The attack requires a degree-1 automorphism acting as $x \mapsto cx$ with $c \not\equiv \pm 1 \pmod q$ (so that the lattice has short vectors below the encoding range).

- $j = 0$: the automorphism ϕ gives $c = \omega$ with $\omega \not\equiv \pm 1 \pmod q$, so the lattice L has non-trivial short vectors below the encoding range and **the attack applies**.
- $j = 1728$: the automorphism maps $x \mapsto -x$, i.e., $c = -1$. The induced relation $x_2 \equiv -x_1 \pmod q$ links a point to its inverse, which the construction already excludes (Section 2.5); the corresponding lattice contains only inverse-pair vectors, so no new cross-message relation arises.
- Generic j: $\mathrm{Aut}(E) = \{\pm 1\}$, so no non-trivial degree-1 x-map exists and there is no linear relation to lift to a lattice.
- Higher-degree endomorphisms ($d \geq 2$): the x-map is a rational function of degree ≥ 2, not a linear map, so it does not induce a lattice of small related coordinates; only the birthday-type argument of Section 3.2 applies.

Concrete instantiation

Polynomial short vector for BN curves. The family parameters are polynomials in an integer seed u: $q(u) = 36u^4 + 36u^3 + 24u^2 + 6u + 1$ and $p(u) = 36u^4 + 36u^3 + 18u^2 + 6u + 1$. The primitive cube root of unity and the lattice short vector admit closed-form polynomial expressions.

Proposition 1. *For any BN seed u such that $q(u)$ is prime and $q(u) \equiv 1 \pmod 3$, the element $\omega(u) = 18u^3 + 18u^2 + 9u + 1 \bmod q(u)$ is a primitive cube root of unity in $\mathbb{F}_q$, and the pair*

$$a(u) = 6u^2 + 2u + 1, \qquad b(u) = 2u$$

satisfies $\omega(u) \cdot a(u) \equiv b(u) \pmod{q(u)}$.

Proof. A direct expansion gives the polynomial identities

$$\omega(u)a(u) - b(u) = (3u+1)q(u), \qquad a(u)^2 + a(u)b(u) + b(u)^2 = q(u).$$

The first identity proves $b(u) \equiv \omega(u)a(u) \pmod{q(u)}$. Reducing the second identity modulo $q(u)$, substituting this congruence, and dividing by $a(u)^2$ gives

$$\omega(u)^2 + \omega(u) + 1 \equiv 0 \pmod{q(u)}.$$

Here $a(u) \not\equiv 0 \pmod{q(u)}$; otherwise the norm identity would force $q(u) \mid b(u)$ and hence $q(u)^2 \mid q(u)$, impossible for prime $q(u)$. Thus $\omega(u)$ is a primitive cube root of unity in $\mathbb{F}_q$. For the positive BN seeds of interest, the norm identity also gives $\max(|a(u)|, |b(u)|) \le \sqrt{q(u)}$.

The pair $(a(u), b(u))$ is therefore a polynomial short vector of L for the entire positive-seed BN curve family. In the concrete searches below we use the independent basis

$$(a_u, b_u), \qquad (a_u + b_u, -a_u),$$

where $a_u = a(u)$ and $b_u = b(u)$ for the chosen seed. If

$$(X, Y) = i(a_u, b_u) + j(a_u + b_u, -a_u),$$

then

$$i = \frac{a_u X + (a_u + b_u)Y}{q}, \qquad j = \frac{b_u X - a_u Y}{q}.$$

Thus, for a coordinate budget $N = MT$, every candidate with $|X|, |Y| < N$ must satisfy

$$|i| \le \left\lfloor \frac{(2a_u + b_u)(N-1)}{q} \right\rfloor, \qquad |j| \le \left\lfloor \frac{(a_u + b_u)(N-1)}{q} \right\rfloor.$$

The coefficient searches below are therefore exhaustive for the stated values of M and T; increasing M expands this finite range linearly.

Concrete end-to-end forgery on Beuchat et al. BN254B. Here and below, an *end-to-end* forgery is an existential forgery carried out in full: the signing-oracle side has a canonical tweak (so the oracle query is possible) and the forged side passes relation verification. Instantiate the GMMZ parameters $M = 2^{120}$ and $T = 128$ on the BN254B curve of Beuchat et al. [9],

$$E/\mathbb{F}_q : y^2 = x^3 + 5, \qquad u_B = 2^{62} - 2^{54} + 2^{44}.$$

This curve is listed in the CFRG Pairing-Friendly Curves Internet-Draft as TEPLA's BN254B, the BN254 curve proposed by Beuchat et al. [26, Section 4.1.2]. The base-field prime is

$q =$ `0x2370fb049d410fbe4e761a9886e502417d023f40180000017e80600000000001`,

so $q \approx 2^{254}$ and $M \cdot T = 2^{127}$. Let

$$a_B = 6u_B^2 + 2u_B + 1, \qquad b_B = 2u_B.$$

The coefficient bound gives $|i| \leq 2$ and $|j| \leq 1$. Exhaustive search in this range reports four *source-compatible* triples — triples for which at least one side s satisfies $k_s^* = k_s$, so that the signing query of step 2 can actually be made:

$$(i, j, e) \in \{(-2, 1, 1), (-2, 1, 2), (2, -1, 1), (2, -1, 2)\}.$$

These are the two coefficient pairs $(i, j) = (-2, 1)$ and $(i, j) = (2, -1)$, with the two possible orientations of the same unordered coordinate pair. Taking $(i, j) = (-2, 1)$ and $e = 1$ yields the oriented pair

$$x_1 = a_B - b_B = \texttt{0x5f408fd00600000000000000000000001},$$
$$x_2 = a_B + 2b_B = \texttt{0x5f408fd00060000017e80600000000001},$$

with $x_2 \equiv \omega x_1 \pmod{q}$. Both coordinates lie in $[0, M \cdot T)$. The corresponding messages, tweaks, and canonical tweaks are

$$m_1 = \texttt{0xbe811fa00c00000000000000000000},\qquad k_1 = 1,\qquad k_1^* = 0,$$
$$m_2 = \texttt{0xbe811fa00c000002fd00c000000000},\qquad k_2 = 1,\qquad k_2^* = 1.$$

Thus the signing oracle can be queried on m_2, since its canonical tweak is exactly $k_2 = 1$. Let

$$y = \texttt{0xcb335da15f334ac763fdfc861348606786810fbc4a303d48a2b58873fa33f6d},$$
$$P_2 = (x_2, y),$$
$$P_1 = \phi^2(P_2) = (x_1, y) = [\lambda^2]P_2.$$

The equality $y(P_1) = y(P_2)$ holds because $x_1^3 = x_2^3$ on this $j = 0$ curve. The forger obtains $\sigma_2 = [\mathsf{sk}]P_2$ from the signing oracle and outputs

$$\sigma_1 = [\lambda^2]\sigma_2 = [\mathsf{sk}]P_1$$

for the fresh verification instance (m_1, k_1). The forged-side tweak $k_1 = 1$ is valid, even though it is not the first valid tweak; this is enough for the GMMZ relation. The SageMath implementation (Section 6.1) confirms the pairing check

$$e(\sigma_1, g_2) = e([\lambda^2][\mathsf{sk}]P_2, g_2) = e([\mathsf{sk}]P_1, g_2) = e(P_1, \mathsf{vk}),$$

so (m_1, k_1, σ_1) is a concrete relation-based BLS forgery.

Ethereum BN254 with the GMMZ bound $M = 2^{120}$. For the Ethereum BN254 curve

$$E/\mathbb{F}_q : y^2 = x^3 + 3, \qquad u_0 = \texttt{0x44e992b44a6909f1},$$

the base-field prime is

$$q = \texttt{0x30644e72e131a029b85045b68181585d97816a916871ca8d3c208c16d87cfd47}.$$

With $M = 2^{120}$ and $T = 128$, the coordinate budget is $N = 2^{127}$ and the coefficient bound gives $|i| \leq 2$ and $|j| \leq 1$. This range contains valid relation instances. For example, the triple $(i, j, e) = (-1, 0, 1)$ gives

$$x_1 = \texttt{0x6f4d8248eeb859fc8211bbeb7d4f1129},$$
$$x_2 = \texttt{0x89d3256894d213e2},$$

with $x_2 \equiv \omega x_1 \pmod{q}$ and $x_1, x_2 \in [0, 2^{120} \cdot 128)$. The corresponding messages, tweaks, and canonical tweaks are

$$m_1 = \texttt{0xde9b0491dd70b3f9042377d6fa9e22}, \qquad k_1 = 41, \qquad k_1^* = 0,$$
$$m_2 = \texttt{0x113a64ad129a427}, \qquad k_2 = 98, \qquad k_2^* = 0.$$

Both chosen tweaks give valid curve points, but neither is the canonical tweak selected by the signing oracle. The exhaustive search over $|i| \leq 2$, $|j| \leq 1$ has the same issue for every valid oriented pair: it finds relation instances, but no side can be obtained by a message-only signing query.

Concrete end-to-end forgery on Ethereum BN254 when $M = 2^{124}$. If the message bound is increased to $M = 2^{124}$ while keeping $T = 128$, then $N = 2^{131}$ and the exhaustive coefficient range grows to $|i| \leq 36$ and $|j| \leq 18$. In this larger range the search finds end-to-end forgeries. For the coefficient triple $(i, j, e) = (-29, 15, 1)$, the search returns

$$x_1 = \texttt{0x6163d1ffd0e14ebc7099915c02203c600},$$
$$x_2 = \texttt{0x6858aa245fccd45db3bf53fa5336e4201},$$

with $x_2 \equiv \omega x_1 \pmod{q}$ and $x_1, x_2 \in [0, 2^{124} \cdot 128)$. The corresponding messages, tweaks, and canonical tweaks are

$$m_1 = \texttt{0xc2c7a3ffa1c29d78e13322b8044078c}, \qquad k_1 = 0, \qquad k_1^* = 0,$$
$$m_2 = \texttt{0xd0b15448bf99a8bb677ea7f4a66dc84}, \qquad k_2 = 1, \qquad k_2^* = 1.$$

Both sides are therefore canonical signing-oracle inputs. Let

$$y = \texttt{0x13d53ec9f7e4d92d65e3e66362c4f8c346548646b9f543fb4a2d9665fe334aed},$$
$$P_1 = (x_1, y),$$
$$P_2 = \phi(P_1) = (x_2, y) = [\lambda]P_1.$$

The forger queries the signing oracle on m_1 to obtain $\sigma_1 = [\mathsf{sk}]P_1$ and outputs $\sigma_2 = [\lambda]\sigma_1 = [\mathsf{sk}]P_2$ as a valid signature on the fresh message m_2.

3.2 Tightening the security bound

GMMZ's stated security of $p/(M \cdot T)$ queries overstates the actual level. Their own Theorem 3 [20] provides a bound

$$\Pr[\text{collision}] \;\leq\; c \cdot \frac{\left(\sum_m |S_m^{\pm}|\right) \cdot Q + Q^2}{p},$$

where c is a small constant, Q is the number of adversary queries, and $S_m^{\pm}$ is the set of curve points encoding message m (including both a point and its inverse), so $|S_m^{\pm}| \leq T$. Hence $\sum_m |S_m^{\pm}| \leq M \cdot T$, and the first term is $c \cdot M \cdot T \cdot Q/p$. Each term independently limits the number of useful queries: the first, $c \cdot M \cdot T \cdot Q/p$, reaches $\Omega(1)$ at $Q \approx p/(M \cdot T)$, while the second, Q^2/p reaches $\Omega(1)$ at $Q \approx \sqrt{p}$. Security is therefore capped at $\min\big(p/(M \cdot T),\ \sqrt{p}\big)$.

The same $\sqrt{q}$ threshold emerges from two independent mechanisms depending on the degree of the endomorphisms available.

Degree ≥ 2 — birthday argument. For α of degree $d \geq 2$, the map $x \mapsto x(\alpha(P))$ is a rational function of degree d, whose image behaves quasi-randomly in $\mathbb{F}_q$ (informally: it exhibits no structure an adversary can exploit; we do not claim pseudorandomness in the technical sense). Under this heuristic the events "$x(P_1) \in [0, M \cdot T)$" and "$x(\alpha(P_1)) \in [0, M \cdot T)$" are approximately independent, each with probability $\approx M \cdot T/q$. The expected number of valid pairs is $(M \cdot T)^2/q$, reaching 1 at $M \cdot T \approx \sqrt{q}$: a birthday threshold, captured by the Q^2 term in GMMZ's Theorem 3.

Degree 1 — lattice argument. For ϕ, the x-map $x \mapsto \omega x$ is a bijection, so the quasi-random heuristic fails entirely. The lattice L (Section 3.1) already gives the threshold *deterministically*: Minkowski guarantees a short vector whenever $M \cdot T > \sqrt{q}$, found in $O(\log^2 q)$ via Cornacchia's algorithm (or $O(1)$ for parameterized families) — no birthday luck required.

3.3 EC-GGM model gap

The gap exposed in Section 3 is structural, not merely a miscalculation of constants: the EC-GGM model itself is blind to the automorphism.

The Groth–Shoup EC-GGM. Following [21], the EC-GGM fixes a random *encoding function* $\pi : \mathbb{Z}_p \to \mathbb{G}$ that is injective, identity-preserving ($\pi(0) = \mathcal{O}$), and *inverse-preserving* ($\pi(-i) = -\pi(i)$ for all i). The last condition captures the only algebraic symmetry baked into the model: a point and its inverse share the same x-coordinate. Parties interact with a group oracle $\mathcal{O}_{\text{grp}}$:

- (`map`, i): returns $\pi(i)$;
- (`add`, P_1, P_2): returns $\pi\big(\pi^{-1}(P_1) + \pi^{-1}(P_2)\big)$.

No further curve structure is exposed; in particular, there is no query for ϕ.

Why ϕ is invisible to the model. The mismatch is between what the automorphism costs a real adversary and what it costs an adversary in the model. On a real $j = 0$ curve, an adversary holding a point $P = (x, y)$ computes $\phi(P) = (\omega x, y)$ with a single field multiplication — no group operation, hence, in model terms, no oracle query. Inside the EC-GGM, however, the same map is a scalar multiplication: since $\phi(P) = [\lambda]P$, computing it through the interface

requires `add` queries. The security proof charges the adversary only for oracle queries, so an attack that uses ϕ (like ours) is strictly cheaper in reality than the model assumes: the proof's query-counting argument never "sees" the automorphism, and its bound does not apply to the real adversary.

ϕ-augmented EC-GGM. A model that correctly reflects curves with automorphisms whose action is non-trivially linear on the encoding coordinate should add a third oracle query:

– (`aut`, P): returns $\pi\big(\lambda \cdot \pi^{-1}(P) \bmod p\big) = \phi(P)$.

In this *ϕ-augmented EC-GGM*, applying ϕ costs a query, so the model faithfully reflects the real adversary's capabilities. This captures both j-invariant 0 and 1728 automorphisms.

Relationship to the Algebraic Group Model. In the AGM [18], every group element the adversary outputs must be expressed as a linear combination of previously received elements. Since $\phi(P) = [\lambda]P$, it is such a linear combination, so the AGM *does* capture the automorphism — it is "free" in the AGM just as in reality. However, the AGM alone does not model coordinates (x, y), which are essential for map-to-curve security arguments that reason about encoding ranges. A hybrid "EC-AGM" combining EC-GGM coordinates with AGM representations would be the principled general model, but its formalization is beyond our scope. Our y-increment sidesteps this modelling question entirely: as shown in Theorem 1, it is provably secure in the *standard* EC-GGM because ϕ is harmless on y-encodings.

4 Removing the Field Restriction

The x-increment method requires $q \equiv 3 \pmod 4$ for inverse exclusion: as recalled in Section 2.3, only then is -1 a QNR, so the witness $z^2 = y$ forces y to be the canonical QR square root and excludes $-y$. When $q \equiv 1 \pmod 4$, -1 is a QR, y and $-y$ are simultaneously QR or QNR, and $z^2 = y$ no longer distinguishes them.

Why residuosity rather than a lexicographic choice. Outside a circuit, the natural canonical choice would be the lexicographically smaller of $\{y, q - y\}$. In-circuit, however, enforcing $y < q/2$ is an integer comparison of a full field element against $(q - 1)/2$, which requires a complete bit decomposition of y: hundreds of constraints natively and thousands under emulated arithmetic (Section 1), dwarfing the 3-multiplication curve check that constitutes the rest of the relation. A residuosity condition, by contrast, is certified by a witness and a few multiplications ($z^2 = y$, or the chain below). This cost inversion is why GMMZ — and this section — canonicalize via residuosity.

The fix is to replace quadratic residuosity with *higher power residuosity*. Let $s = v_2(q - 1)$.

Proposition 2 (Inverse exclusion via 2^s-th powers). *Let q be an odd prime and $s = v_2(q-1)$. For any $y \in \mathbb{F}_q^*$: if y is a 2^s-th power then $-y$ is not. Hence replacing the witness $z^2 = y$ with the chain $z^{2^s} = y$ restores inverse exclusion for any prime q.*

Proof. Let g generate $\mathbb{F}_q^*$ and write $q - 1 = 2^s r$ with r odd. The 2^s-th powers form the unique subgroup $H = \langle g^{2^s} \rangle$ of index 2^s.

Since $(-1) = g^{(q-1)/2} = g^{2^{s-1}r}$ and $2^{s-1}r$ is not divisible by 2^s (as r is odd), the element -1 lies in a coset of H distinct from H itself.

If $y = g^{2^s j} \in H$ then $-y = g^{2^{s-1}r + 2^s j}$; for $-y \in H$ one would need $2 \mid r$, which contradicts r odd. Hence $-y \notin H$.

Modified verifier. Replace $z^2 = y$ with $z^{2^s} = y$. This costs s multiplication constraints instead of 1, raising the total from 4 (for $s = 1$) to $3 + s$ in general. For $s = 1$, every quadratic residue is a 2^s-th power, so the original GMMZ success probability remains $\approx 1/2$ per trial. For general s, however, the prover additionally needs the square root y to be a 2^s-th power, so witness generation succeeds only when y lies in the subgroup of index 2^s of $\mathbb{F}_q^*$.

Instantiations.

- $s = 1$ ($q \equiv 3 \pmod 4$): original GMMZ.
- $s = 2$ ($q \equiv 5 \pmod 8$): witness $z^4 = y$, costing a single additional constraint.
- $s = 46$ (BLS12-377 $\mathbb{F}_q$): witness $z^{2^{46}} = y$. Still much cheaper than verifying the inner hash, but impractical for the prover.

Prover practicality. Proposition 2 shows that the x-increment construction *theoretically* extends to any prime field via the $z^{2^s} = y$ witness chain. However, the prover must find a y that is simultaneously a square root of $x^3 + ax + b$ *and* a 2^s-th power in $\mathbb{F}_q^*$. Since the 2^s-th powers form a subgroup of index 2^s, each candidate y passes this test with probability 2^{-s}, requiring $T \approx 2^s$ trials in expectation. For $s = 1$ this is negligible ($T = 128$ suffices), but for large s — e.g. $s = 46$ for BLS12-377 — the forward search becomes infeasible: the prover cannot map a *chosen* message m to a curve point within a practical tweak bound.

In summary, the higher-residuosity extension generalises the GMMZ construction to any field from the *verifier's* perspective (costing only s additional constraints), but renders *witness generation* impractical when s is large. The y-increment construction (Section 5) resolves this entirely.

5 The y-Increment Construction

The x-increment method (Section 2.5) encodes a message as a candidate x-coordinate and recovers y by taking a square root. We propose reversing the roles: encode the message as a candidate y-coordinate and recover x by solving the curve equation as a cubic. On $j = 0$ curves — the setting of all deployed pairing-friendly curves relevant to the applications considered here — this single swap

resolves all the limitations of GMMZ identified above. The two constructions are in fact dual: the x-increment suits $j = 1728$ curves and the y-increment suits $j = 0$ curves (Remark 2); neither is universal.

- **No field restriction.** The inverse-exclusion argument becomes trivial: inverse points (x, y) and $(x, -y)$ share x but have y-coordinates y and $q - y$ respectively. Since the message range $[0, M \cdot T)$ and its reflection $[q - M \cdot T, q)$ are disjoint (because $M \cdot T < q/2$), inverse points always encode different messages. No witness is needed and there is no constraint on $q \bmod 4$.
- **Order-3 automorphism is neutralized.** The automorphism ϕ (Section 2.2) acts as the *identity* on the y-coordinate. Therefore $\phi(P)$ and P always carry the same y-value and encode the same message. The automorphism cannot produce a cross-message collision.

On a $j = 0$ curve, i.e. $y^2 = x^3 + b$, recovering x from a given y amounts to computing a cube root of $y^2 - b$. As recalled in Section 2.3, when $q \equiv 1 \pmod 3$ each cube has three roots forming the orbit $\{x, \omega x, \omega^2 x\}$ of ϕ. Since ϕ fixes y, all three roots encode the *same message*. The prover witnesses any one of them; the verifier accepts it by checking the curve equation and the y-range.

On a $j \neq 0, 1728$ curve, i.e. $y^2 = x^3 + ax + b$ with $a \cdot b \neq 0$, recovering x from a candidate y requires solving the depressed cubic $x^3 + ax + c = 0$ with $c = b - y^2$. The cubic has at least one root in $\mathbb{F}_q$ with probability $\approx 2/3$ for a random y; none are related by a field automorphism. The root-finding algorithm is given in Section 2.4.

Witness multiplicity vs. inverse pairs. It may seem inconsistent that the x-increment needs machinery to exclude the inverse point while the y-increment tolerates up to three witnesses (x, y), $(\omega x, y)$, $(\omega^2 x, y)$ per message. The distinction is between same-message and cross-message structure. A forgery requires a known scalar relation between encodings of *distinct* messages (Section 3); relations among same-message witnesses only let an adversary produce another signature on a message it already queried, which is not a forgery. Quantitatively, this multiplicity enters the security bound of Theorem 1 merely as the factor 3 in $\sum_m |S_m^{\pm}| \leq 3MT$. Inverse pairs are different in kind: the Groth–Shoup/GMMZ theorem structurally requires each witness set S_m to be inverse-free, and the y-increment obtains this for free from the y-range (Remark 1), whereas the x-increment must buy it with residuosity (Section 4). Witness multiplicity does, however, affect signature *uniqueness* — in both constructions, including GMMZ's via the tweak k — a caveat for applications discussed in Section 7.

Remark 1 (Canonicality). In GMMZ's x-increment construction, the verifier enforces a canonical square-root choice for the y-coordinate. This is not a canonicality requirement per se but an *inverse-exclusion* measure: without it, a prover could substitute the point $(x, -y)$ for (x, y), and since both share the same x-coordinate the message encoding would be ambiguous.

The y-increment construction is free of this issue. The negation map sends $(x, y) \mapsto (x, -y)$, and $-y \bmod q \notin [0, M \cdot T)$ for any y in the encoding range, so inverse points automatically encode different messages. Moreover, since the prover witnesses the full point (x, y) in both the multiset-hashing and BLS signature settings, no *inverse-exclusion* constraint on x is ever needed: the verifier checks $(x, y) \in E(\mathbb{F}_q)$ and $\lfloor y/T \rfloor = m$, and in the BLS case the pairing equation $e(\sigma, g_2) = e((x, y), \mathsf{vk})$ uniquely determines the correct point.

Remark 2 ($j = 1728$ duality). On a $j = 1728$ curve the order-4 automorphism ψ acts as $x \mapsto -x$ on the x-coordinate and as $y \mapsto iy$ with $i \not\equiv \pm 1 \pmod q$ on the y-coordinate. Thus x-increment is safe on $j = 1728$, but y-increment is not: the lattice $L_{1728} = \{(a, b) \in \mathbb{Z}^2 : b \equiv i \cdot a \pmod q\}$ has determinant q and a short vector of norm $\leq \sqrt{q}$, giving an analogous attack when $M \cdot T \gtrsim \sqrt{q}$. Enforcing $M \cdot T < \sqrt{q}$ eliminates it. In practice, for the two applications discussed in this work, all deployed curves have $j \neq 1728$.

5.1 Construction

Figure 2 gives the constructor and the corresponding relation. The prover iterates candidate y-values of the form $m \cdot T + k$ until one yields a valid curve point. The procedure is uniform for $j = 0$ and $j \neq 0$ curves; the only difference is in step 2(b), where x-recovery reduces to a cube root for $j = 0$ or to solving the depressed cubic of Section 2.4 for $j \neq 0$. The tweak k is part of the witness, making the construction deterministic and constant-time from the verifier's perspective: the verifier checks four simple algebraic conditions without re-running the search.

Failure probability. On a $j = 0$ curve ($q \equiv 1 \pmod 3$), each candidate value $c = b - y^2$ is a cube in $\mathbb{F}_q$ independently with probability $\approx 1/3$, since cubes form the unique subgroup of index 3 in $\mathbb{F}_q^*$. The failure probability after T tweaks is $(2/3)^T$; $T = 256$ gives $2^{-150} < 2^{-128}$. (When $q \equiv 2 \pmod 3$ every element is a cube, so any T suffices.)

On a $j \neq 0$ curve, the depressed cubic $x^3 + ax + c = 0$ has at least one root in $\mathbb{F}_q$ with probability $\approx 2/3$, since irreducible cubics account for $\approx 1/3$ of monic cubics over $\mathbb{F}_q$. The failure probability after T tweaks is $(1/3)^T$; $T = 81$ is the minimum for 2^{-128} security ($\lceil 128/\log_2 3 \rceil = 81$), and $T = 128$ is a conservative choice.

Verifier cost. The verifier checks conditions 1–4 of Figure 2. Condition 1 is a range check on k; condition 2 is linear (free); condition 3 verifies the curve equation $x^3 + ax + b = y^2$ using 3 multiplications (x^2, x^3, y^2; the term ax is free since a is a public constant); condition 4 is free once the curve equation holds and $\mathbb{G}$ has prime order. The total is **3 multiplications + 1 range check** for both curve families.

Compared to GMMZ x-increment, whose verifier requires $3 + s$ multiplications (Section 4), where $s = v_2(q-1) \geq 1$: the extra s multiplications come from

y-Increment Constructor

Inputs:

- An elliptic curve $E\colon y^2 = x^3 + ax + b$ over $\mathbb{F}_q$.
- An elliptic curve group $\mathbb{G} \subseteq E(\mathbb{F}_q)$.
- A message $m \in \mathcal{M}$.
- A tweak bound $T \in \mathbb{N}$.

Procedure:

1. Initialize tweak counter $k \leftarrow 0$.
2. While $k < T$:
   (a) Let $y \leftarrow m \cdot T + k$ and $c \leftarrow b - y^2$.
   (b) Find $x \in \mathbb{F}_q$ such that $x^3 + ax + c = 0$.
   $j = 0$: compute a cube root of $-c$.
   $j \neq 0$: solve the depressed cubic
   - If a solution x exists and $(x, y) \in \mathbb{G}$, output (x, y).

   (c) Otherwise, increment $k \leftarrow k + 1$.
3. If no valid point is found after T iterations, declare failure.

y-Increment Relation

Parameters:

- A prime-order subgroup $\mathbb{G} \subseteq E(\mathbb{F}_q)$ of an elliptic curve $E(\mathbb{F}_q) : y^2 = x^3 + ax + b$.
- A message space $\mathcal{M} \subseteq \mathbb{F}_q$ and tweak space $[0, T)$ such that $M \cdot T < \sqrt{q}$.

Relation $\mathcal{R}_y \subseteq \mathcal{M} \times \mathbb{G} \times \mathcal{W}$:
A triple $(m, (x, y), w)$ where $w = k$ belongs to $\mathcal{R}_y$ if:

1. $m \in \mathcal{M}, \quad k \in [0, T)$,
2. $y = m \cdot T + k$,
3. $y^2 = x^3 + ax + b$,
4. $(x, y) \in \mathbb{G}$.

Fig. 2. The y-increment map-to-curve relation: top, the constructor; bottom, the efficiently constructible relation.

the square-root witness chain $z^{2^s} = y$ that enforces inverse exclusion. The y-increment eliminates this chain entirely, saving s multiplications — 1 for BN254 and BLS12-381 ($s = 1$, $q \equiv 3 \pmod 4$) and 46 for BLS12-377 ($s = 46$). The tradeoff is a slightly wider range check for $j = 0$: since each candidate succeeds with probability $\approx 1/3$ rather than $\approx 1/2$, one needs $T = 256$ instead of $T = 128$ for the $j = 0$ case, adding one bit to the range check. In our `gnark` implementation we use $T = 256$ uniformly across all curves. `gnark`'s commitment-based range checker decomposes a value into limbs of an auto-selected optimal width of size 4 bits. An 8-bit check ($T = 256$) decomposes evenly into two 4-bit limbs with no remainder, whereas a 7-bit check ($T = 128$) does not divide evenly by 4, pro-

ducing an overflowing most-significant limb that requires one extra constraint. Consequently, the more conservative $T = 256$ yields equal or better constraint counts than $T = 128$ in practice for all constructions in all settings.

5.2 Security analysis

We prove that the y-increment construction is secure in the *standard* Groth–Shoup EC-GGM [21] — no model augmentation is needed. The key insight is that on $j = 0$ curves the automorphism ϕ preserves y-coordinates, so it cannot produce cross-message collisions in the y-encoding.

Theorem 1 (y-increment security in the standard EC-GGM). *Let $E : y^2 = x^3+b$ be a $j = 0$ curve over $\mathbb{F}_q$ with a prime-order subgroup $\mathbb{G}$ of order p. Let $\mathcal{M}$ be a message space with $|\mathcal{M}| = M$ and tweak bound T satisfying $M \cdot T < \sqrt{q}$. In the standard EC-GGM, any adversary making at most Q oracle queries (*`map` *and* `add`*) has advantage at most*

$$\mathsf{Adv}^{\mathrm{forge}} \;\le\; \frac{c \cdot MT \cdot Q + Q^2}{p}$$

for an absolute constant $c > 0$. In particular, the construction achieves DLP-level security $\min\bigl(p/(MT),\ \sqrt{p}\bigr)$.

Proof. We work in the Groth–Shoup EC-GGM [21]. The encoding function $\pi : \mathbb{Z}_p \to \mathbb{G}$ is a random bijection satisfying $\pi(0) = \mathcal{O}$ and $\pi(-i) = -\pi(i)$. The adversary interacts with oracles `map` and `add` as defined in Section 3.3.

Step 1: Encoding sets $S_m^{\pm}$. For each message $m \in \mathcal{M}$, the set of valid encodings under the y-increment is $S_m = \{P \in \mathbb{G} : y(P) \in [mT, (m+1)T)\}$. On a $j = 0$ curve each y-value yields up to 3 points (the cube-root orbit $\{(x, y), (\omega x, y), (\omega^2 x, y)\}$), so $|S_m| \le 3T$. Inverse points are automatically excluded from S_m: if $y \in [0, MT)$ then $-y = q - y \notin [0, MT)$ since $MT < q/2$, so no $P \in S_m$ has $-P \in S_m$. Following the GMMZ convention, $S_m^{\pm}$ denotes the set of inversion classes $\{P, -P\}$ encoding m; since each point of S_m contributes a distinct class, $|S_m^{\pm}| \le 3T$ and $\sum_m |S_m^{\pm}| \le 3MT$.

Step 2: ϕ is harmless (no model augmentation needed). The adversary can compute $\phi(P) = (\omega x, y)$ from coordinates without any oracle query. But ϕ preserves y-coordinates: $y(\phi(P)) = y(P)$. Therefore $\phi(P)$ encodes the *same message* as P. The "free" points obtained via ϕ are same-message duplicates that do not produce cross-message collisions. This is the crucial difference from the x-increment, where ϕ maps $x \mapsto \omega x$ and *does* create cross-message collisions (Section 3). In particular, the standard EC-GGM already accounts for all points in $S_m^{\pm}$, including those related by ϕ, so no `aut` oracle is needed.

Step 3: Applying GMMZ Theorem 3. The y-increment relation (Figure 2) satisfies the preconditions of [20, Theorem 3]: (i) *completeness*—the prover finds a valid encoding with probability $1 - (2/3)^T$; (ii) *efficient constructibility*—the verifier checks 4 algebraic conditions in $O(1)$ field operations; (iii) *disjointness*—distinct messages m map to disjoint y-ranges $[mT, (m+1)T)$. By [20, Theorem 3],

the advantage is bounded by $c\big(\sum_m |S_m^{\pm}|\big)Q/p + Q^2/p$ for an absolute constant c. Substituting $\sum_m |S_m^{\pm}| \le 3MT$:

$$\mathsf{Adv}^{\mathrm{forge}} \;\le\; \frac{3c \cdot MT \cdot Q}{p} \;+\; \frac{Q^2}{p}\,.$$

The factor 3 is absorbed into the constant. The bound is negligible when $Q \ll p/(MT)$ and $Q \ll \sqrt{p}$.

6 Implementation and Benchmarks

We provide three complementary implementations. First, a SageMath simulation that demonstrates the algebraic attack end-to-end and checks the correctness of both x-increment and y-increment constructions. Second, a fork of the GMMZ `Noir` circuits that confirms the algebraic attack at the circuit level. Third, `gnark` [11] circuits for both relations, from which we derive concrete R1CS and PLONK (SCS) constraint counts. We cover native and emulated arithmetic, $j = 0$ and $j \neq 0$ curves, and both low ($s = 1$) and high ($s \gg 1$) 2-adicity regimes. All source code is available at [16]:

`https://github.com/yelhousni/map-to-curve`.

6.1 Attack — SageMath simulation

We provide a self-contained SageMath simulation of the full attack and forgery pipeline.

bn_poly_attack.sage. Derives the CM short vector $(a(u), b(u)) = (6u^2{+}2u{+}1,\ 2u)$ from first principles (Proposition 1). Verifies the polynomial identity $b(u) \equiv \omega(u) \cdot a(u) \pmod{q(u)}$ symbolically in $\mathbb{Q}[u]$, then evaluates at the BN254 seed $u_0 =$ `0x44e992b44a6909f1` and checks $|a(u_0)|, |b(u_0)| \le \sqrt{q}$.

bn254_forgery.sage. Instantiates the $O(1)$ BLS signature forgery of Section 3 for BN254. The pairing-level correctness of the forgery is confirmed by an explicit optimal ate pairing computation implemented in pairing_bn.sage.

Additionally, for correctness, we simulate both x-increment and y-increment provers and verifiers.

x_increment.sage, y_increment.sage. Implements curve-agnostic GMMZ x-increment and our y-increment encoders and verifiers. Both are tested on different elliptic curves.

cardano.sage. Implements the Cardano depressed cubic solver of Section 2.4. The implementation is validated against SageMath's `.roots()` on 200 random cubics over each of the P-256 and BN254 base fields, with 0-root / 1-root / 3-root breakdowns matching expectations ($\approx 1/3$, $1/2$, $1/6$ respectively).

6.2 Attack — GMMZ `Noir` circuits

The GMMZ paper [20, Section 7] describes `Noir` [15] circuit implementations of the x-increment relation but provides no link to the source code. Upon independent investigation, we located an implementation on GitHub under the account of co-author Harjasleen Malvai [24] (`https://github.com/Jasleen1/map-to-curve`) and included a fork of it (`https://github.com/yelhousni/map-to-curve-gmmz`) as a submodule in our artifact [16] at commit `aaaeea6`. On this fork we demonstrate the algebraic attack of Section 3 directly at the circuit level.

native_noir_map_to_curve/src/attacks.nr. Confirms that the circuit relation is non-injective under the order-3 automorphism. Given a valid witness, the automorphism ϕ yields a second valid witness for a different message.

noir_map_to_curve/src/attacks.nr. Repeats the same automorphism attack for the non-native circuit variant, where x-coordinates are represented as 256-bit limb arrays over the BN254 scalar field.

The circuit additionally omits the tweak range check $t \in [0, T)$ required for injectivity. This is an implementation gap that does not reflect the GMMZ construction as described in the paper and should not be treated as a vulnerability of the GMMZ scheme; the resulting exploits are deferred to Appendix A.

6.3 Constructions — `gnark` circuits

We implement both the x-increment and y-increment relations as `gnark` [11] circuits to obtain concrete constraint counts and to verify end-to-end satisfiability. Each circuit is compiled against the R1CS and PLONK (SCS) arithmetisations.

Circuit variants. Each construction is provided in two flavors.

- **Native.** The circuit arithmetic is performed directly over the target curve's base field, which must equal the scalar field of the proving system's curve. This is the case when the two curves form a *2-cycle* [4, Section 5.4] (each curve's base field is the other's scalar field) or sit in a *2-chain* [4, Section 5.2] (the inner curve's scalar field equals the outer curve's base field). We instantiate this for three $j = 0$ curves: BN254 [7] and Grumpkin [5], which form a 2-cycle, and BLS12-377 [6], which is the inner curve in a 2-chain with BW6-761 [17]. BN254 has $s = v_2(q-1) = 1$; Grumpkin has $s = 28$; BLS12-377 has $s = 46$.
- **Emulated.** The circuit arithmetic uses non-native field emulation over BN254, supporting any curve. We instantiate this for BN254, BLS12-377, and P-256, the latter being a $j \neq 0$ curve ($a = -3$).

Witness generation at high 2-adicity. For the x-increment circuit, the prover must produce a witness (k, z) satisfying $z^{2^s} = y$. When s is large (e.g. $s = 28$ for Grumpkin, $s = 46$ for BLS12-377), mapping a chosen message m to a valid witness is infeasible (Section 4). For benchmarking purposes we use a *reverse construction*: pick a random seed r, compute $y = r^{2^s}$ (guaranteeing that y is a 2^s-th power with $z = r$), solve for x via cube root ($x^3 = y^2 - b$), and decompose $x = m \cdot T + k$. This yields a valid witness for a *random* message rather than a chosen one, which suffices for circuit compilation and constraint counting but illustrates the prover limitation discussed in Section 4.

Shared helper packages. Witness generation requires field-level cube roots that are not provided by `gnark`. For the BN254 and BLS12-377 base fields we use the routines accompanying [22], which implement the $q \equiv 19 \pmod{27}$ case of Table 1 (with the conditional ninth-root-of-unity correction; $v_3(q-1) = 2$ for BN254) and the $q \equiv 7 \pmod 9$ case ($v_3(q-1) = 1$ for BLS12-377) respectively. The remaining helpers are factored into two reusable `internal/` packages.

- `internal/grumpkin` — cube root in Grumpkin $\mathbb{F}_q$ (= BN254 $\mathbb{F}_r$). Since $q \equiv 19 \pmod{27}$ ($v_3(q-1) = 2$), we compute $x^{(q+8)/27}$ and apply the conditional ninth-root-of-unity correction of Table 1. The exponentiation uses a short addition chain (301 operations, generated by `mmcloughlin/addchain` [25]).
- `internal/p256` — Cardano depressed-cubic solver and the necessary $\mathbb{F}_{q^2}$ arithmetic over the P-256 base field, used by the y-increment witness generator for P-256.
  - $\mathbb{F}_{q^2}$ *arithmetic.* $\mathbb{F}_{q^2} = \mathbb{F}_q[u]/(u^2 + 1)$ (valid since $q \equiv 3 \pmod 4$), with Karatsuba multiplication and complex squaring.
  - $\mathbb{F}_q$ *cube root.* Since $q \equiv 4 \pmod 9$ ($v_3(q-1) = 1$), a cube root of a cubic residue is given by $x^{(8q-5)/9}$, computed via a short addition chain (278 operations).
  - $\mathbb{F}_{q^2}$ *cube root.* Implemented via the torus method of Section 2.3 following [22], using an addition chain for $(q-4)/9$ (278 operations). All arithmetic stays in $\mathbb{F}_q$.
  - $\mathbb{F}_{q^2}$ *square root.* Used in Cardano's Case 2 (Δ non-square). Implemented via the algorithm of Section 2.3, using an addition chain for $(q-3)/4$ (264 operations). All arithmetic stays in $\mathbb{F}_q$.

Constraint counts. Table 2 reports the number of R1CS constraints and PLONK gates for each circuit, compiled with `gnark` v0.14 on an AMD EPYC 9R14.

The y-increment native circuit costs **16 R1CS constraints** across all three curves, regardless of s, saving 2 constraints over x-increment for $s = 1$ and 47 for $s = 46$ (BLS12-377). In the emulated setting the savings are more striking: BLS12-377 drops from 7 741 to 1 065 R1CS constraints (an 8.6× reduction) and from 31 175 to 4 065 SCS gates. For BN254 and P-256 the savings are modest (around 13–16% in R1CS) since $s = 1$; in all cases the y-increment emulated circuit is smaller than or equal to its x-increment counterpart.

Construction	Mode	Curve	s	R1CS	SCS
x-increment	Native	BN254	1	18	40
	Native	BLS12-377	46	63	85
	Native	Grumpkin	28	45	67
	Emulated	BN254	1	977	3 705
	Emulated	BLS12-377	46	7 741	31 175
	Emulated	P-256	1	1 065	4 082
y-increment	Native	BN254	—	16	38
	Native	BLS12-377	—	16	38
	Native	Grumpkin	—	16	38
	Emulated	BN254	—	821	3 065
	Emulated	BLS12-377	—	1 065	4 065
	Emulated	P-256	—	911	3 448

Table 2. Constraint counts for x-increment and y-increment circuits. R1CS = rank-1 constraint system; SCS = sparse constraint system (PLONK). $s = v_2(q-1)$.

6.4 Full pipeline benchmarks

We benchmark the full prove/verify pipeline — witness generation, proving, and verification — using Groth16 and PLONK (KZG) backends. All timings are collected on an AMD EPYC 9R14 (32 cores, 64 GB RAM) with `go test -bench`, running both constructions back-to-back in a single session; each reported value is the median of 3 repetitions of 10 iterations each. Native circuits compile over the cycle partner (BN254→Grumpkin, BLS12-377→BW6-761); emulated circuits compile over BN254. The native BN254 case (compiled over Grumpkin) is excluded from the pipeline benchmarks because Grumpkin is not pairing-friendly and thus lacks a Groth16/PLONK proving backend in `gnark`.

Figure 3 compares witness generation times. For $j = 0$ curves the y-increment computes a single cube root, while the x-increment must find a k such that $x = m \cdot T + k$ yields a quadratic residue *and* a 2^s-th root witness z. When s is large the forward search is infeasible. The reverse construction (Section 6.3) produces valid witnesses for circuit testing but does not represent a realistic prover workflow. For P-256 ($j \neq 0$), the y-increment pays the Cardano solver cost but avoids the 2-adicity witness chain entirely.

Figures 4 and 5 report proving and verification times respectively. The gap is most pronounced on emulated BLS12-377: the x-increment's $s = 46$ squaring chain inflates Groth16 proving from 8.9 ms to 23.9 ms (2.7×) and PLONK from 30.6 ms to 147.7 ms (4.8×). For native Grumpkin and BLS12-377 the y-increment is 1.2× faster.

For the $s = 1$ emulated circuits (BN254, P-256), PLONK proving times are statistically indistinguishable between the two constructions despite the y-increment's smaller constraint counts. This is structural rather than noise:

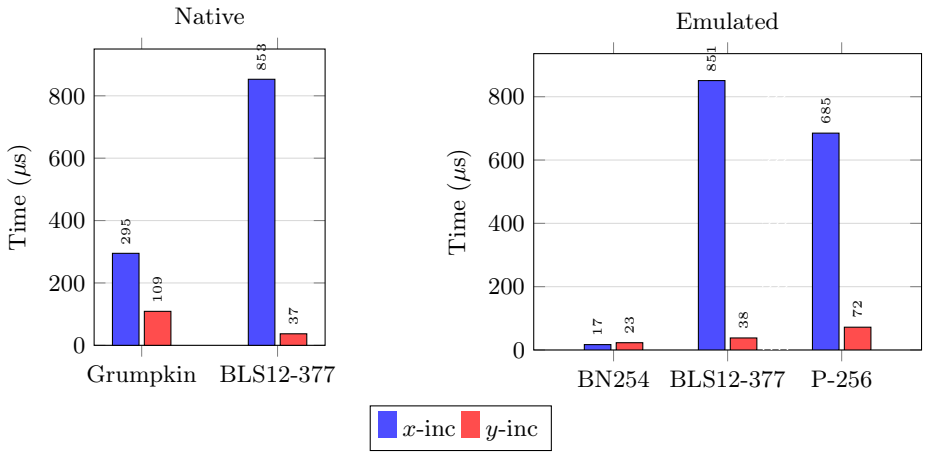


Fig. 3. Witness generation time (μs). Hatched bars denote the x-increment on curves with high 2-adicity ($s = 28$ for Grumpkin, $s = 46$ for BLS12-377): the forward message-to-x map is infeasible so these bars are capped at chart height; the reverse construction used for correctness testing is not a meaningful benchmark.

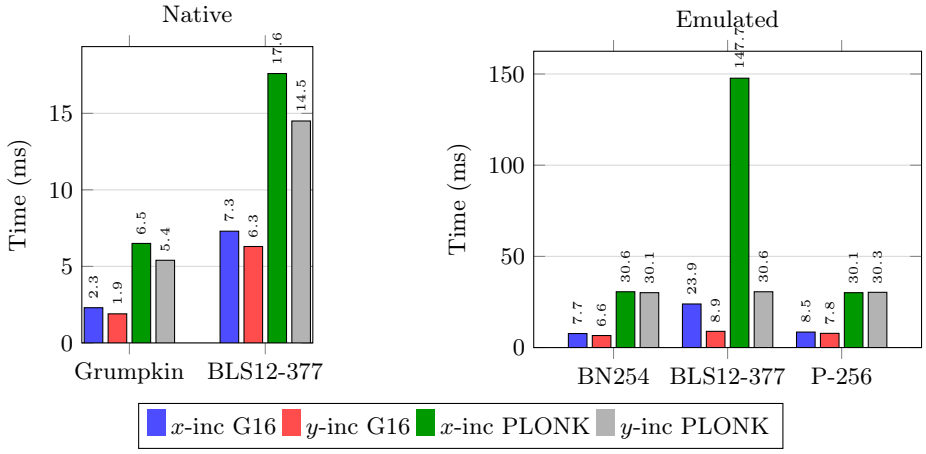


Fig. 4. Proving time (ms). The y-increment's constraint savings yield 2.7× Groth16 and 4.8× PLONK speedups on emulated BLS12-377.

PLONK proving cost is governed by the size of the FFT/KZG evaluation domain, which is the constraint count rounded up to the next power of two, and both constructions' $s = 1$ circuits fall in the same 4096-row domain (Table 2). Constraint savings therefore translate into proving time only when they cross a power-of-two boundary — which is precisely the high-2-adicity case, where the x-increment BLS12-377 circuit requires a 32768-row domain against the y-increment's 4096. Groth16, whose multi-scalar multiplications scale more smoothly with the num-

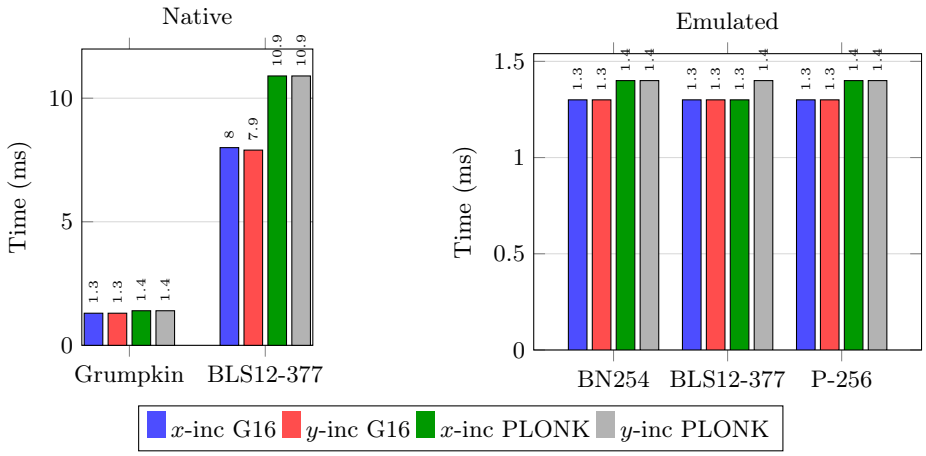


Fig. 5. Verification time (ms). Verification is fast and nearly identical for both constructions across all configurations.

ber of constraints and variables, shows the y-increment consistently faster even at $s = 1$ (6.6 vs 7.7 ms on emulated BN254). Verification times are comparable across all configurations, as expected for succinct proofs.

7 Conclusion

The GMMZ x-increment construction achieves remarkable constraint savings, but its EC-GGM security analysis has three gaps. First, it misses the order-3 automorphism on usually deployed curves, enabling a concrete BLS signature forgery in $O(\log^2 q)$ time (or $O(1)$ for parameterized families) when $M \cdot T \gtrsim \sqrt{q}$. Second, it is implicitly restricted to fields with $q \equiv 3 \pmod 4$; higher power residuosity theoretically restores correctness for any field, but witness generation is only practical when the 2-adicity $s = v_2(q-1)$ is small. Third, the stated security level $p/(M \cdot T)$ ignores the Q^2/p term in GMMZ's own theorem, which caps security at the DLP-level $\sqrt{p}$.

Our y-increment construction resolves all three issues on $j = 0$ curves. By encoding messages as y-coordinates, the order-3 automorphism (which fixes y) is neutralized by construction; inverse exclusion is free; and the constraint count is comparable or lower.

Relations are not hashes. A caveat for practitioners: map-to-curve *relations* — GMMZ's and ours alike — are nondeterministic and non-unique, and therefore behave differently from the hashes-to-curve they replace in two ways that surrounding systems may silently rely on. First, multiset hashing built on hash-to-curve supports proofs of both equality and *non-equality* of multisets; built on a relation, it supports only equality. For a given input m the prover may choose

any witness satisfying the relation — and m typically has several (the tweak k in both constructions, the cube-root orbit in ours) — so a prover can exhibit $\mathsf{Map}(m_1) \neq \mathsf{Map}(m_2)$ even when $m_1 = m_2$. Second, BLS signatures built on hash-to-curve are unique; built on a map-to-curve relation they are not, for the same reason. Deployments that rely on signature uniqueness (e.g. for randomness beacons) must not treat a relation-based BLS as a drop-in replacement.

Coverage and an open question. With parameters $M \cdot T < \sqrt{q}$, the y-increment *alone* gives a prover-practical, secure relation for *every* curve — $j = 0$, $j = 1728$, and generic. Its witness generation is a cube root or a cubic solve (Section 2.4); it never invokes the 2^s witness chain and is therefore insensitive to the base-field 2-adicity that makes the x-increment prover impractical. Security is moreover *provable* in the standard EC-GGM whenever the curve has no degree-1 automorphism acting on the y-coordinate: this covers $j = 0$ (Theorem 1) and, by the same argument, generic curves, whose only automorphism — negation — is excluded by the y-range. The lone class where the proof is missing is $j = 1728$: there the order-4 automorphism ψ acts non-trivially on y and is invisible to the standard EC-GGM, exactly as ϕ is for the x-increment. Enforcing $M \cdot T < \sqrt{q}$ rules out the resulting lattice equivocation (Remark 2), so the construction is secure, but its security rests on the parameter bound rather than a model proof. The x-increment supplies such a proof for $j = 1728$ where its prover is practical (low 2-adicity); at high 2-adicity it is not. A security proof for the $j = 1728$ y-increment in a ψ-augmented EC-GGM — the case that matters at high 2-adicity — is the natural open question.

We provide a full open-source implementation comprising a SageMath end-to-end attack simulation, correctness tests for both constructions on BN254, Grumpkin, and P-256, and `gnark` circuits benchmarked against R1CS and PLONK. The y-increment native circuit requires only 16 R1CS constraints across all tested curves, independent of the 2-adicity s.

Responsible Disclosure. We communicated these findings to the GMMZ authors prior to public release. They acknowledged the gap and confirmed they will address it by tightening the parameter constraints in a revised version. To the best of our and their knowledge, no production system currently deploys the x-increment construction.

Acknowledgments. We thank Aurore Guillevic for observing that the direct attack on Ethereum BN254 does not yield a canonical tweak for the signing oracle, and for suggesting the refined linear-combination attack, along with a SageMath script.

References

1. Aardal, M.A., Adj, G., Alblooshi, A., Aranha, D.F., Martinez, I.A.C., Chávez-Saab, J., Filho, D.L.G., Reijnders, K., Rodríguez-Henríquez, F.: Optimized one-dimensional SQIsign verification on intel and cortex-M4. IACR TCHES **2025**(1), 497–522 (2025).

2. Adj, G., Rodríguez-Henríquez, F.: Square root computation over even extension fields. IEEE Transactions on Computers **63**(11), 2829–2841 (2014).
3. Adleman, L.M., Manders, K., Miller, G.: On taking roots in finite fields. In: 18th Annual Symposium on Foundations of Computer Science (SFCS 1977). pp. 175–178. IEEE (1977).
4. Aranha, D.F., El Housni, Y., Guillevic, A.: A survey of elliptic curves for proof systems. DCC **91**(11), 3333–3378 (2023).
5. Aztec Protocol: Grumpkin: a cycle companion for BN254. `https://aztecprotocol.github.io/aztec-connect/primitives.html` (2019)
6. Barreto, P.S.L.M., Lynn, B., Scott, M.: Constructing elliptic curves with prescribed embedding degrees. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 02. LNCS, vol. 2576, pp. 257–267. Springer, Berlin, Heidelberg (Sep 2003).
7. Barreto, P.S.L.M., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 319–331. Springer, Berlin, Heidelberg (Aug 2006).
8. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Succinct non-interactive zero knowledge for a von neumann architecture. In: Fu, K., Jung, J. (eds.) USENIX Security 2014. pp. 781–796. USENIX Association (Aug 2014), `https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/ben-sasson`
9. Beuchat, J.L., González-Díaz, J.E., Mitsunari, S., Okamoto, E., Rodríguez-Henríquez, F., Teruya, T.: High-speed software implementation of the optimal Ate pairing over Barreto-Naehrig curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) PAIRING 2010. LNCS, vol. 6487, pp. 21–39. Springer, Berlin, Heidelberg (Dec 2010).
10. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. Journal of Cryptology **17**(4), 297–319 (Sep 2004).
11. Botrel, G., Piellard, T., Housni, Y.E., Kubjas, I., Tabaie, A., Galteland, Y.J.: gnark a fast zk-snark library by consensys (2026), `https://github.com/consensys/gnark`
12. Bowe, S.: BLS12-381: New zk-SNARK elliptic curve construction. Zcash blog (March 11 2017), `https://blog.z.cash/new-snark-curve/`
13. Bowe, S., Chiesa, A., Green, M., Miers, I., Mishra, P., Wu, H.: ZEXE: Enabling decentralized private computation. In: 2020 IEEE Symposium on Security and Privacy. pp. 947–964. IEEE Computer Society Press (May 2020).
14. Clarke, D.E., Devadas, S., van Dijk, M., Gassend, B., Suh, G.E.: Incremental multiset hash functions and their application to memory integrity checking. In: Laih, C.S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 188–207. Springer, Berlin, Heidelberg (Nov / Dec 2003).
15. contributors, N.: Noir a domain specific language for zero knowledge proofs (2026), `https://github.com/noir-lang/noir`
16. El Housni, Y., Bunz, B.: On the security of constraint-friendly map-to-curve relations (2026), `https://github.com/yelhousni/map-to-curve`
17. El Housni, Y., Guillevic, A.: Optimized and secure pairing-friendly elliptic curves suitable for one layer proof composition. In: Krenn, S., Shulman, H., Vaudenay, S. (eds.) CANS 20. LNCS, vol. 12579, pp. 259–279. Springer, Cham (Dec 2020).
18. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 33–62. Springer, Cham (Aug 2018).

19. Grassi, L., Khovratovich, D., Rechberger, C., Roy, A., Schofnegger, M.: Poseidon: A new hash function for zero-knowledge proof systems. In: Bailey, M., Greenstadt, R. (eds.) USENIX Security 2021. pp. 519–535. USENIX Association (Aug 2021), https://www.usenix.org/conference/usenixsecurity21/presentation/grassi
20. Groth, J., Malvai, H., Miller, A., Zhang, Y.N.: Constraint-friendly map-to-elliptic-curve-group relations and their applications. In: Hanaoka, G., Yang, B.Y. (eds.) ASIACRYPT 2025, Part II. LNCS, vol. 16246, pp. 511–543. Springer, Singapore (Dec 2025).
21. Groth, J., Shoup, V.: On the security of ECDSA with additive key derivation and presignatures. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part I. LNCS, vol. 13275, pp. 365–396. Springer, Cham (May / Jun 2022).
22. Housni, Y.E.: Fast cube roots in fp2 via the algebraic torus. Cryptology ePrint Archive, Paper 2026/392 (2026), https://eprint.iacr.org/2026/392
23. Joux, A., Lercier, R.: Improvements to the general number field sieve for discrete logarithms in prime fields. A comparison with the Gaussian integer method. Math. Comp. **72**(242), 953–967 (2003).
24. Malvai, H.: Noir implementation of the GMMZ map-to-curve construction (2025), https://github.com/Jasleen1/map-to-curve, no link provided in the GMMZ paper; located independently on GitHub under the account of co-author Harjasleen Malvai
25. McLoughlin, M.: addchain: Cryptographic addition chain generation in go (2024), https://github.com/mmcloughlin/addchain
26. Sakemi, Y., Kobayashi, T., Saito, T.: Pairing-friendly curves. Internet-Draft draft-irtf-cfrg-pairing-friendly-curves-02, Internet Research Task Force (Mar 2020), https://www.ietf.org/archive/id/draft-irtf-cfrg-pairing-friendly-curves-02.html, work in progress
27. Scott, M.: A note on the calculation of some functions in finite fields: Tricks of the trade. Cryptology ePrint Archive, Report 2020/1497 (2020), https://eprint.iacr.org/2020/1497
28. Smith, B.: Easy scalar decompositions for efficient scalar multiplication on elliptic curves and genus 2 jacobians. In: Algorithmic Arithmetic, Geometry, and Coding Theory (AGCT 2013), Contemporary Mathematics, vol. 637, pp. 151–164. American Mathematical Society (2015).
29. Standards for Efficient Cryptography Group (SECG): SEC 2: Recommended Elliptic Curve Domain Parameters, Version 2.0. Tech. rep., Certicom Research (2010), https://www.secg.org/sec2-v2.pdf

A Implementation-Gap Exploits on GMMZ's `Noir` Circuits

Here we exploit the absence of the tweak range check $t \in [0, T)$ which are unrelated to the algebraic vulnerability of the GMMZ construction analyzed in Section 3.

native_noir_map_to_curve/src/attacks.nr. Three test cases against the native Grumpkin circuit.

- `exploit_missing_range_check`: the circuit accepts the same curve point for two messages $m_1 = 7$ and $m_2 = 6$ by choosing tweaks $(t_1, t_2) = (0, 256)$ satisfying $m_1 T + t_1 = m_2 T + t_2 = 1792$; $t_2 = 256 \geq T$ goes unchecked.
- `exploit_zero_message`: setting $m = 0$ and $t = x$ satisfies the embedding $x = 0 \cdot T + x$ for any valid point (x, y), claiming it as the encoding of the zero message without performing a hash.
- `exploit_multiset_hash_collision`: the injectivity break propagates to elliptic-curve accumulation-based multiset hashing; two multisets differing by the swap $m_1 \leftrightarrow m_2$ produce the same Grumpkin-addition digest, forging a zkVM memory-consistency check.

noir_map_to_curve/src/attacks.nr. Four test cases against the non-native circuit variant. The first three mirror those above.

- `exploit_missing_range_check_nonnative`: same injectivity break as above, adapted to the 256-bit limb representation of x.
- `exploit_zero_message_nonnative`: same zero-message forgery adapted to the non-native field representation.
- `exploit_multiset_hash_collision_nonnative`: same multiset hash collision adapted to the non-native circuit.
- `exploit_mul_mod_p_noncanonical`: the `mul_mod_p` helper does not enforce a canonical representative after reduction modulo p; a prover can supply a non-canonical limb encoding of y that passes the `is_equal` check against z^2 while encoding a different field value, bypassing the quadratic-residue condition.