Divide-and-Pair: Faster subgroup membership testing for elliptic curves

Yu Dai[1], Youssef El Housni[2], Dimitri Koshelev[3] and Krijn Reijnders[4]

[1] Wuhan University of Technology, China
eccdaiy39@gmail.com
[2] Consensys, Linea, USA
youssef.elhousni@consensys.net
[3] National Academy of Sciences of Belarus, Institute of Mathematics, Department of Algebra, Minsk, Belarus
dimitri.koshelev@gmail.com
[4] COSIC, KU Leuven, Belgium
crypto.krijn@gmail.com

Abstract. Subgroup membership testing (SMT) on an elliptic curve with non-trivial cofactor is essential to prevent small-subgroup attacks in cryptographic protocols. In the existing literature, there exist two non-trivial methods for SMT on elliptic curves with modest cofactor (typically a power of 2): Pornin's approach tests membership by repeatedly dividing by prime divisors of the cofactor (mostly halving), finishing with a Legendre symbol; Koshelev's approach replaces all divisions with Tate pairings (possibly in the quadratic extension of the base field), but requires non-degeneracy conditions that are not always met. In this paper, we observe that both approaches sit at the extremes of a single division-pairing trade-off. The resulting method, Divide-and-Pair, is always at least as fast as either Pornin's or Koshelev's method and strictly faster in many cases. We instantiate Divide-and-Pair on five curves in widespread use, including Curve25519, Curve448, GC256A, Four$\mathbb{Q}$ and Jubjub. Our Go implementation, built on the open-source `gnark-crypto` library, achieves significant speedups over state-of-the-art tests, namely of 1.6× on Curve25519, 1.3× on Curve448, 1.4× on GC256A, 10.8× on Four$\mathbb{Q}$ and 7× on Jubjub.

Keywords: algebraic torus arithmetic · elliptic curve cryptography · point halving · quartic residue symbol · subgroup membership testing · Tate pairing

1 Introduction

Despite the widely discussed threat to classical public-key systems by Shor's quantum algorithm [Sho99], elliptic curve cryptography (ECC) remains highly relevant in modern cryptographic practice as an indispensable building block in contemporary protocols. Although the transition to post-quantum cryptography is underway, such transitions are generally slow and the maturity of such schemes is generally lower than that of ECC. Therefore, ECC continues to be a cornerstone of secure digital infrastructure for the foreseeable future. Hybrid approaches, combining ECC with quantum-resistant schemes, are a transitional strategy reflecting a pragmatic balance: maintaining the proven efficiency and security of ECC today, while preparing for a post-quantum future.

In real-world cryptosystems, a small number of elliptic curves are widely used. Curve25519 [Ber06] and Curve448 [Ham15b] form the basis of the widely adopted schemes X25519/Ed25519 and X448/Ed448, respectively. These two curves have been included in the NIST standardization efforts [CMR+23]. Nowadays, the former is actively used in modern security

protocols, such as TLS 1.3, SSH, WireGuard, Signal, and Tor (see, e.g., [26a; 26b]). Four$\mathbb{Q}$ [CL15] is a curve that targets almost the same security level as Curve25519, although the two curves are quite different objects with different attack surfaces. Four$\mathbb{Q}$ is defined over the quadratic extension of the Mersenne prime field with $p = 2^{127} - 1$ and is equipped with two cheap linearly independent non-scalar endomorphisms (a low-degree CM endomorphism and a $\mathbb{Q}$-curve endomorphism), enabling a four-dimensional scalar decomposition and hence a faster scalar multiplication than on Curve25519. However, this additional structure also makes it the less conservative choice: the extension base field and the small-discriminant complex multiplication constitute potential attack vectors that are absent for Curve25519, and exploiting the endomorphisms adds implementation complexity. GC256A is the unique 256-bit twisted Edwards curve specified in the Russian GOST standard [ANS18; SBA22]. Finally, Jubjub [Ele] serves as a companion to BLS12-381 [Bow17] and is used for elliptic-curve operations within zero-knowledge proof systems and verifiable computation frameworks.

Subgroup membership testing. A critical aspect of practical importance in ECC is *subgroup membership testing* (SMT). In most cryptographic constructions, one works not with the full group $E(\mathbb{F}_q)$ but with a subgroup of large prime order r. All the above five curves have a non-trivial cofactor h. In this situation, the absence of SMT in an elliptic curve cryptosystem can lead to serious vulnerabilities, e.g., small-subgroup attacks [LL97]. In this scenario, an adversary deliberately supplies points from carefully chosen low-order subgroups to extract information about secret scalars. Therefore, it is crucial to test that a given point belongs to the intended subgroup. Cofactor multiplication is known to be another effective measure to prevent such attacks. It must, however, be applied consistently across a protocol, as illustrated by the double-spending vulnerability in the Monero cryptocurrency [Spa17]. In this case, the ring-signature verification equations constrain points only up to their small-order components (i.e., they hold in the subgroup of order r), while double spends are detected by comparing *key images* for equality as elements of the full group. Since key images were neither tested for membership in the order-r subgroup nor cleared of torsion, an adversary could add a small-order point to a valid key image and obtain up to h distinct key images for the same output, each passing verification. The deployed fix is precisely to perform a SMT on the key image I, namely $[r]I \stackrel{?}{=} \mathcal{O}$. The alternative approaches Decaf [Ham15a] and Ristretto [dLA] (see also RFC 9496 [dGH+23]) avoid such pitfalls by quotienting out the cofactor entirely but require protocol-level changes.

SMT recently got new impetus because of an incorrect implementation [Cha26; Den25] of the given test in the popular Libsodium library [di26]. Naively, one can always verify that $P \in E[r](\mathbb{F}_q)$ given $P \in E(\mathbb{F}_q)$ by means of the simple scalar multiplication $[r]P \stackrel{?}{=} \mathcal{O}$, where $\mathcal{O}$ is the zero point on E. However, this approach is very slow due to the large size of the prime r. To resolve this issue, more efficient methods for SMT have emerged. For curves with cyclic group $E(\mathbb{F}_q)$ and 2-adic cofactor h, King [Kin04] describes a method based on successive point halvings when q is even. This method is optimized by Pornin [Por22] for twisted Edwards curves and odd q. Koshelev [Kos23; Kos24], on the other hand, proposes an approach using the Tate pairing of some degree dividing h. This technique is relevant for more general group structures on $E/\mathbb{F}_q$ but requires that the pairing is non-degenerate.

Key insight. For simplicity, we consider an elliptic $\mathbb{F}_q$-curve E with cyclic group $E(\mathbb{F}_q)$ and 2-adic cofactor h. The new insight into testing $P \in E[r](\mathbb{F}_q)$ on such a curve corresponds to the subgroup sequence

$$E(\mathbb{F}_q) \supset [d]E(\mathbb{F}_q) = E[rh/d](\mathbb{F}_q) \supset \cdots \supset E[2r](\mathbb{F}_q) \supset E[r](\mathbb{F}_q),$$

where $d = \gcd(h, q-1)$. Each step is either *halving* (quite expensive but always works) or *Tate pairing* (much cheaper but available only for the non-degenerate part of degree d). Pornin's and Koshelev's methods sit at the opposite ends of this trade-off (cf. Table 1).

Table 1: Comparison between three SMT methods for curves with cyclic group $E(\mathbb{F}_q)$ and 2-adic cofactor h.

Method	Halvings	Pairing degree	Valid when
Pornin [Por22]	$\log_2(h) - 1$	2 (Legendre)	always
Koshelev [Kos23]	0	h	$h \mid q-1$
This work	$\log_2(h/d)$	$d = \gcd(h, q-1)$	**always**

Pornin always halves down to the last step and uses only the degree-2 Tate pairing (i.e., the Legendre symbol), which is universally available since $2 \mid q-1$ for any odd q but leaves performance on the table. In turn, Koshelev applies the Tate pairing (essentially the residue symbol) of full degree h, avoiding all halvings, but this requires $h \mid q-1$ (i.e., non-degeneracy), which fails for many curves in practice. Our Divide-and-Pair algorithm consists in the optimal split: halve only $\log_2(h/d)$ times where the Tate pairing is still degenerate, then use the non-degenerate pairing of degree d.

Our Contribution. The main contributions of the paper are the following.

1. Theorem 1 is the core theorem for combining divisions and pairings in the context of SMT, resulting in Algorithm 1: the general algorithm Divide-and-Pair.

2. We instantiate Divide-and-Pair on five practical curves in large characteristic p (see Table 2)[1]. For each curve, we highly optimize the algorithm, using division-free halvings [Por22], Weilert's Euclidean-type algorithm [Wei02] for the quartic symbol over $\mathbb{F}_p$, torus $\mathcal{T}_2(\mathbb{F}_p)$ arithmetic [Gal12, Section 6.3.2] for the quartic and octic symbols over $\mathbb{F}_{p^2}$, and the norm map $N_{\mathbb{F}_{p^2}/\mathbb{F}_p}$ for the septic symbol over $\mathbb{F}_{p^2}$.

Table 2: The essential properties with respect to the proposed SMT algorithm for five popular elliptic curves. In column five, t_n denotes the Tate pairing of degree n.

Curve	h	d	Divisions	Pairings
Curve25519	8	4	1 halving	t_4
Curve448/GC256A (over $\mathbb{F}_p$)	4	2	1 halving	t_2
Curve448/GC256A (over $\mathbb{F}_{p^2}$)	$*$	$*$	0	t_4
Four$\mathbb{Q}$	392	$*$	0	$t_8 + 2\times t_7$
Jubjub	8	8	0	t_8

3. Using the gnark-crypto library [BPE+26], we provide a high-performance Go implementation. On Curve25519 and Jubjub, Divide-and-Pair achieves a 1.6× and 7× speedup over Pornin's method (2.8× and 9× over naive), respectively; on Four$\mathbb{Q}$, 13.2× over naive and 10.8× over the endomorphism-based SMT adapted from [DLZZ23]; finally, on Curve448 and GC256A, by extending the residuosity check to $\mathbb{F}_{p^2}$ via arithmetic in the torus $\mathcal{T}_2(\mathbb{F}_p)$, we improve on Pornin's method by 1.3× and 1.4× (on naive by 3.5× and 3.6×), respectively.

[1] We have omitted the parameters h and d for Curve448/GC256A over $\mathbb{F}_{p^2}$, as well as d for Four$\mathbb{Q}$, as $E(\mathbb{F}_{p^2})$ is non-cyclic for these curves. Furthermore, for Curve448/GC256A, the candidate point is defined over $\mathbb{F}_p$ and we do not require the values h and d over $\mathbb{F}_{p^2}$.

Code. Our source code is open-source and available at [DEKR26]:

https://github.com/yelhousni/divide-and-pair.

2 Preliminaries

Notation. We use $\mathbb{Z}_N$ to denote the ring of integers modulo N. We use $\mathbb{F}_p$ to denote the finite field with p elements, and $\mathbb{F}_q$ to denote a general finite field with p^k elements for some $k \geq 1$. We use $\mathrm{QR}(\mathbb{F}_p)$ to denote all quadratic residues $\alpha \in \mathbb{F}_p$, i.e. squares, and μ_n to refer to the n-th roots of unity $\mu_n := \{\alpha \in \overline{\mathbb{F}_q}^* \mid \alpha^n = 1\}$. We refer to the kernel of the norm map $\mathbb{F}_{p^2} \to \mathbb{F}_p$ as the *torus* $\mathcal{T}_2(\mathbb{F}_p)$, which is therefore isomorphic to μ_{p+1}. We say something is *rational* when it is defined over the base field $\mathbb{F}_q$. We write $\chi_n(\zeta)$ for the n-ary residuosity $\zeta^{\frac{q-1}{n}}$, where the field is clear from context.

2.1 Elliptic Curves

Let $\mathbb{F}_q$ be a finite field with a large characteristic p. An elliptic curve E over $\mathbb{F}_q$ is a smooth projective, algebraic curve of genus one. The set of points on E forms an abelian group with an identity element $\mathcal{O}$. We use $E(\mathbb{F}_q)$ to denote the (finite) subgroup that consists of $\mathbb{F}_q$-rational points. It has the structure

$$E(\mathbb{F}_q) \cong \mathbb{Z}_N \times \mathbb{Z}_M$$

as an abelian group, where $M \mid N$ and $M \mid (q-1)$. Repeatedly adding a point $P \in E(\mathbb{F}_q)$ to itself n times gives a map

$$[n] : E \to E, \quad P \mapsto P + \ldots + P,$$

called *scalar multiplication*. The order of P is the smallest positive integer such that $[n]P = \mathcal{O}$, and we denote the kernel of $[n]$ by $E[n]$. We denote the intersection $E[n] \cap E(\mathbb{F}_q)$ by $E[n](\mathbb{F}_q)$. When n is coprime to p, as an abelian group $E[n] \cong \mathbb{Z}_n \times \mathbb{Z}_n$.

Division on Elliptic Curves. Given a point $R \in E(\mathbb{F}_q)$ and a positive integer n coprime to p, there are n^2 pre-images $R' \in E$ such that $[n]R' = R$ as $E[n] \cong \mathbb{Z}_n \times \mathbb{Z}_n$. If there is at least one $\mathbb{F}_q$-rational pre-image R', then we say that R is *divisible* by n and R' is a *rational n-division point*. The rational points divisible by n are, by definition, the elements of $[n]E(\mathbb{F}_q)$, and each $P \in [n]E(\mathbb{F}_q)$ has $|E[n](\mathbb{F}_q)|$ rational n-division points.

A *halving* refers specifically to division by $n = 2$. Halvings are the simplest case of division, and also the most efficient. For the rest of the practical applications in this work, we almost exclusively use halvings whenever we need division of elliptic curve points.

2.2 The Tate Pairing

Let n be a positive integer such that $n \mid q-1$. The (reduced) Tate pairing of degree n on $E/\mathbb{F}_q$, denoted t_n, is a bilinear pairing

$$\begin{aligned} t_n : & E[n](\mathbb{F}_q) \times E(\mathbb{F}_q)/[n]E(\mathbb{F}_q) \to \mu_n, \\ & (P, R) \to \big(f_{n,P}(R)\big)^{(q-1)/n}, \end{aligned}$$

where $f_{n,P}$, called the *Miller function*, is a normalized rational function with divisor $\mathrm{div}(f_{n,P}) = n(P) - ([n]P) - (n-1)(\mathcal{O})$. The Tate pairing t_n satisfies the following properties:

1. **Bilinearity**: For any $P_1, P_2 \in E[n](\mathbb{F}_q)$ and $R_1, R_2 \in E(\mathbb{F}_q)$,

$$t_n(P_1, R_1 + R_2) = t_n(P_1, R_1) \cdot t_n(P_1, R_2),$$
$$t_n(P_1 + P_2, R_1) = t_n(P_1, R_1) \cdot t_n(P_2, R_1).$$

2. **Non-degeneracy**:

   a. Let $P \in E[n](\mathbb{F}_q)$. Then, $t_n(P, R) = 1$ for all $R \in E(\mathbb{F}_q)$ if and only if $P = \mathcal{O}$.

   b. Let $R \in E(\mathbb{F}_q)$. Then, $t_n(P, R) = 1$ for all $P \in E[n](\mathbb{F}_q)$ if and only if $R \in [n]E(\mathbb{F}_q)$.

For this work, this last property is essential, as it allows us to efficiently test rational divisibility of a point $R \in E(\mathbb{F}_q)$. That is, if we are given a basis $\langle P, Q\rangle$ of $E[n](\mathbb{F}_q)$, we know that there exists a point $R' \in E(\mathbb{F}_q)$ such that $R = [n]R'$ if and only if the degree-n Tate pairings $t_n(P, R) = t_n(Q, R) = 1$.

Degeneracy of the Tate Pairing. When $n \nmid q - 1$, the Tate pairing may be degenerate and only the unreduced Tate pairing is well-defined. This case is often avoided, as we usually construct our curves to work with pairings that are non-degenerate. However, for ECC, curves are constructed for efficient arithmetic in the large prime-order subgroup where the cofactor h is mainly required to be small and 2-adic, and not necessarily so that the Tate pairing t_h is non-degenerate.

Curve25519 is such a curve, with cofactor $h = 8$ and $8 \nmid p - 1$. In this case, we find degeneracy for $P = (0, 0)$, which has the property that $t_8(P, R) = 1$ for all $R \in E(\mathbb{F}_p)$, even though $P \neq \mathcal{O}$. Such degeneracies may be resolved by moving to field extensions: for the embedding degree k of h, we get $h \mid p^k - 1$ by definition, and so the Tate pairing over $\mathbb{F}_{p^k}$ is non-degenerate. Following the above example we have $k = 2$, hence the degeneracy is resolved over $\mathbb{F}_{p^2}$ where $P = (0, 0)$ now pairs non-trivially with half of the points in $E(\mathbb{F}_{p^2})$. We will say that n is *degenerate* over a given field whenever t_n is degenerate over that field, where we will often assume that the field is obvious from context.

Generalized Tate Pairing. Bruin [Bru11] introduced a generalization of the Tate pairing to any isogeny $\varphi : E \to E'$ as a pairing

$$t_\varphi : \ker \varphi(\mathbb{F}_q) \times E(\mathbb{F}_q)/\hat{\varphi}(E'(\mathbb{F}_q)) \to \mu_n,$$

where $n = \deg \varphi$. This pairing is non-degenerate when $\ker \varphi \subseteq E[q - 1]$, and the computation is the same as the degree-n Tate pairing. However, the generalized Tate pairing t_φ does something slightly more precise than the regular Tate pairing: whereas t_n allows us to test divisibility of $R \in E(\mathbb{F}_q)$, with t_φ we can test if R is in the image of the dual $\hat{\varphi}$.

2.3 Subgroup Membership Testing

Subgroup membership testing (SMT) asks us to verify that for a given point $R \in E(\mathbb{F}_q)$, we have $R \in \mathbb{G}$ for a given subgroup $\mathbb{G} \subseteq E(\mathbb{F}_q)$. Most commonly, this refers to verifying that R is a member of the large prime-order subgroup $\mathbb{G} = E[r](\mathbb{F}_q)$ in which we perform standard elliptic curve cryptography. The naive technique simply verifies $[r]R \stackrel{?}{=} \mathcal{O}$, but this may be considered excessively costly in certain applications due to the large size of the group order r.

Pornin's method using point halvings. King [Kin04] introduced a non-trivial technique for SMT on plain elliptic curves under the condition that the cofactor h is a power of 2. Let the curve equation of E be given as $y^2 = g(x)$ and assume $E(\mathbb{F}_q) \cong \mathbb{Z}_{2^n \cdot r} \times \mathbb{Z}_{2^m}$. The key observation is that a point $R \in E(\mathbb{F}_q)$ has order r if and only if $R_i \in E(\mathbb{F}_q)$ for $i = 0, 1, \ldots, n$, where R_i is the halving of R_{i-1} and $R_0 = R$. Pornin [Por22] specializes and optimizes this technique to twisted Edwards curves. We refer to this specialized approach as *Pornin's method*, which proceeds as follows:

1. Compute $n-1$ successive point halvings $R_0 \to R_1 \to \cdots \to R_{n-1}$. The procedure can be aborted early if $R_i \notin E(\mathbb{F}_q)$. Otherwise, perform step 2.
2. Compute the quadratic residue symbol $t = \chi_2\big(x(R_{n-1})\big)$, where $x(R_{n-1})$ denotes the x-coordinate of R_{n-1}. If $t = 1$, then return True; otherwise, return False.

The main optimization by Pornin [Por22] is an efficient algorithm for computing the point halving at a cost of two square roots in $\mathbb{F}_q$. Hence, Pornin's method comes at a total cost of $2(n-1)$ square roots and one quadratic residue symbol in $\mathbb{F}_q$. This method is tailored to curves with 2-adic cofactors such as Curve25519, Curve448, GC256A, and Jubjub, but generalizes easily to more general cofactors by dividing by other factors $\ell \neq 2$.

Koshelev's method using Tate pairings. Koshelev [Kos23] proposes a different technique for SMT on plain elliptic curves. Based on the observation that $[h]E(\mathbb{F}_q) = E[r](\mathbb{F}_q)$ for the cofactor h, instead of verifying $R \in E[r](\mathbb{F}_q)$, we may also check whether $R \in [h]E(\mathbb{F}_q)$. If t_h is non-degenerate, the latter can be tested efficiently using one or two Tate pairings. In more detail,

$$R \in E[r](\mathbb{F}_q) \quad \Leftrightarrow \quad t_h(P, R) = 1 \text{ for all } P \in E[h](\mathbb{F}_q).$$

Since $E[h](\mathbb{F}_q) \cong \mathbb{Z}_h \times \mathbb{Z}_{h'}$ for some integer $h' \mid h$, Koshelev's method requires computing two Tate pairings with degrees h' and h, respectively. When $E[h](\mathbb{F}_q)$ is cyclic, we get $h' = 1$, so that Koshelev's method requires only one Tate pairing.

By combining with efficiently computable endomorphisms, this technique has been extended to pairing-friendly curves [DHK+24], which is well-suitable to the BW13 [FST10, Construction 6.6] and BLS48 [BLS04] families, and to higher-dimensional abelian varieties as well as to Kummer surfaces [Reij25].

Comparison. The above two methods have their own strengths and weaknesses. More specifically,

- Pornin's method applies only to curves whose cofactor is a power of 2, while Koshelev's method requires $h \mid q - 1$.
- The computational cost of Pornin's method comes largely from the $n-1$ point halvings, while the cost of Koshelev's method is dominated by the final exponentiations in the two Tate pairing computations.
- Pornin's method does not need additional storage requirement, while Koshelev's method requires storing a (pre-computable) basis of $E[h]$.

3 Generalizing SMT with Division

In its most general form, we present an algorithm that performs subgroup membership testing on P by *dividing up* to the non-degenerate degree, where we can use Tate pairings to establish membership. This is captured in Theorem 1, which generalizes Koshelev's technique for SMT by using divisions.

3.1 Divide-and-Pair Algorithm

Theorem 1. *Let $E/\mathbb{F}_q$ be an elliptic curve such that $E(\mathbb{F}_q) \cong \mathbb{Z}_{r\cdot n_1} \times \mathbb{Z}_{n_2}$, where r is a large prime and $n_2 \mid n_1$ and $n_2 \mid q-1$. Let $d = \gcd(n_1, q-1)$ and write $n_1 = d \cdot n'$. Let $P \in E[d](\mathbb{F}_q)$ and $Q \in E[n_2]$ such that $E[d](\mathbb{F}_q) = \langle P, Q\rangle$ Then, $R \in E[r](\mathbb{F}_q)$ if and only if*

$$t_d(P, R') = t_{n_2}(Q, R') = 1$$

for any $R' \in E(\mathbb{F}_q)$ such that $[n']R' = R$.

Proof. As $d \mid (q-1)$, the pairing t_d is non-degenerate (2.b). Hence, for $\langle P, Q\rangle = E[d](\mathbb{F}_q)$, we get $t_d(P, R') = 1$ and $t_{n_2}(Q, R') = 1$ if and only if $t_d(T, R') = 1$ for any $T \in E[d](\mathbb{F}_q)$, which happens if and only if $R' \in [d]E(\mathbb{F}_q)$. As $[n']R' = R$, we get that $R \in [n'd]E(\mathbb{F}_q) = [n_1]E(\mathbb{F}_q) = E[r](\mathbb{F}_q)$. □

In practice, we may have $n_2 = 1$ and therefore $Q = \mathcal{O}$, making the second pairing trivial. For example, many of the standardized curves in use have cyclic groups $E(\mathbb{F}_q) \cong \mathbb{Z}_{n_1\cdot r}$, hence $n_2 = 1$, and often have $n_1 = 2^k$. In these cases, we may simplify Theorem 1.

Corollary 1. *Let $E(\mathbb{F}_q)$ be cyclic of order $2^k \cdot r$, where r is a large prime, and let $t = \log_2(\gcd(2^k, q-1))$. Then we can verify $R \in E[r](\mathbb{F}_q)$ using $(k-t)$ halvings of R, and a single Tate pairing of degree 2^t.*

Remark 1 (Connection to isogenies). For the cyclic case with $n_1 = 2^k$, the degree-2^t Tate pairing $t_{2^t}(T, R)$ can be evaluated as $\chi_{2^t}(f_{2^t,T}(R))$, where $f_{2^t,T}$ is the Miller function with $\mathrm{div}(f) = 2^t(T) - 2^t(\mathcal{O})$. This is equivalently the descent criterion for the 2^t-isogeny φ with kernel $\langle T\rangle$: the point R lies in the image of the dual $\hat{\varphi}$ if and only if the generalized Tate pairing $t_\varphi(T, R) = 1$.

When $t = 2$, we can write the Miller function f_{4,T_4} evaluated at a point $R = (u_R, w_R)$ on the Pornin Montgomery model $uw^2 = u^2 + Au + B$ (see Section 4.1) as

$$f_{4,T_4}(R) = u_R \cdot (w_R - 2i)^2,$$

where $T_4 = (\sqrt{B}, 2i)$ generates $E[4](\mathbb{F}_q)$ and $i = \sqrt{-1}$. This is exactly the case of Curve25519.

Algorithm. Based on Theorem 1, we propose a new algorithm to execute SMT on elliptic curves, which is presented in Algorithm 1. In Section 5, we specialize this algorithm to specific curves and highly optimize the division and pairing computations. We assume the existence of an algorithm `RationalDivision`(R, ℓ) that takes as input $R \in E(\mathbb{F}_q)$ and a prime ℓ, and returns a point $R' \in E(\mathbb{F}_q)$ such that $[\ell]R' = R$, if such a point exists. If such a point does not exist, our algorithm returns false as this implies that the original point was not in $E[r](\mathbb{F}_q)$. The function Factors(n) returns a list of prime factors ℓ_i of n each repeated e_i times for $n = \prod \ell_i^{e_i}$.

3.2 Examples

We showcase some applications of Theorem 1 for SMT on a list of well-known plain elliptic curves.

Curve25519. This curve is a widespread curve for the ≈ 128-bit security level generated by Bernstein [Ber06] and subsequently standardized by the IETF [LHT16] and NIST [CMR+23, Sec. 3.2]. For Theorem 1, we need the following properties.

- The base field is $\mathbb{F}_p$ with $p = 2^{255} - 19$.

Algorithm 1 Divide-and-Pair

Input: A point $R \in E(\mathbb{F}_q)$, and P, Q, d, n' and n_2 as in Theorem 1
Output: A boolean indicating if $R \in E[r](\mathbb{F}_q)$.

$R' \leftarrow R$
for $\ell \in \text{Factors}(n')$ **do**
  $R' \leftarrow \texttt{RationalDivision}(R', \ell)$
if $t_d(P, R') = 1$ and $t_{n_2}(Q, R') = 1$ **then**
  return `true`
else
  return `false`

- The group structure is $E(\mathbb{F}_p) \cong \mathbb{Z}_{8 \cdot r}$, where r is a large prime.
- We get cofactor $h = 8$ and $d = \gcd(h, p-1) = 4$.
- Thus, Divide-and-Pair requires 1 point halving and 1 Tate pairing of degree 4.

Curve448 and GC256A. Curve448 is the so-called *Goldilocks curve* generated by Hamburg [Ham15b] and also standardized by the IETF [LHT16] and NIST [CMR+23, Sec. 3.2], while GC256A is the unique 256-bit[2] twisted Edwards/Montgomery curve accepted in the Russian ECC standard `GC256A` [SBA22, Table 2]. For Theorem 1, we need the following properties.

- Curve448 and GC256A are both defined over a prime field $\mathbb{F}_p$ with $p = 2^{448} - 2^{224} - 1$ and $p = 2^{256} - 617$, respectively.
- Both curves have group structure $E(\mathbb{F}_p) \cong \mathbb{Z}_{4 \cdot r}$, where r is a large prime.
- Both have cofactor $h = 4$ and $d = \gcd(h, p-1) = 2$.
- *Option 1 (over $\mathbb{F}_p$).* Divide-and-Pair requires 1 point halving and 1 Tate pairing of degree 2.
- *Option 2 (over $\mathbb{F}_{p^2}$).* Since $p \equiv 3 \pmod 4$, both curves have embedding degree 2 with respect to 4. In other words, the extension field $\mathbb{F}_{p^2}$ is the smallest field such that $E[4] \subset E(\mathbb{F}_{p^2})$. Hence, a point $R \in E[r](\mathbb{F}_p)$ if and only if $t_4(P, R) = t_4(Q, R) = 1$, where $\langle P, Q \rangle$ is a basis of $E[4]$. Fortunately, we can select the point P as a generator of $E[4](\mathbb{F}_p)$. Then, for any $R \in E(\mathbb{F}_p)$ we get

$$t_4(P, R) = f_{4,P}(R)^{(p^2-1)/4} = \left(f_{4,P}(R)^{p-1}\right)^{(p+1)/4} = 1,$$

  as $f_{4,P}(R) \in \mathbb{F}_p$. Hence, we only need to verify the $\mathbb{F}_{p^2}$-pairing $t_4(Q, R) = 1$. In conclusion, Divide-and-Pair on both curves requires **0 point halvings** and 1 Tate pairing of degree 4 over $\mathbb{F}_{p^2}$.

Four$\mathbb{Q}$. This curve is an exceptionally fast 256-bit curve, due to the existence of efficiently computable endomorphisms, proposed in [CL15]. Up to now, Four$\mathbb{Q}$ has not found a place in official documents, but it is possibly the fastest among ones in large characteristic at the 128-bit security level. For Theorem 1, we need the following properties.

[2] In the Russian ECC standard there is one more twisted Edwards/Montgomery curve under the name GC512C or id-tc26-gost-3410-2012-512-paramSetC. Since the given curve is 512-bit and is used only in a rather small and local market (unlike Curve448), its applicability is very restricted. Even though GC512C enjoys the identical SMT as GC256A (cf. [Kos24, Table 1]), we omit the corresponding fast implementation and benchmarks.

- This curve is defined over $\mathbb{F}_{p^2}$, where $p = 2^{127} - 1$.
- The group structure is $E(\mathbb{F}_{p^2}) \cong \mathbb{Z}_{2^3 \cdot 7 \cdot r} \times \mathbb{Z}_7$, where r is a large prime.
- We get $n_1 = 2^3 \cdot 7$, $n_2 = 7$ and $n_1 \mid p^2 - 1$.
- Thus, Divide-and-Pair requires **0 point halvings** and two Tate pairings, one of degree $2^3 \cdot 7$, and one of degree 7. For efficiency, we compute these as three Tate pairings: one of degree 8 and two of degree 7.

An alternative SMT for Four$\mathbb{Q}$ can be obtained from the efficiently computable endomorphisms ψ and φ [CL15], adapting the techniques of [DLZZ23] to reduce the order check to a 4-dimensional multi-scalar relation with short ($\sim$61-bit) scalars. We implement this approach as a baseline.

Jubjub. This curve is a twisted Edwards "companion curve"[3] of BLS12-381, used in Zcash's Sapling protocol for efficient in-circuit elliptic curve operations [Ele]. For Theorem 1, we need the following properties.

- The base field is $\mathbb{F}_p$ with p the BLS12-381 scalar field prime ($\approx 2^{255}$), with $2^{32} \mid p-1$.
- The group structure is $E(\mathbb{F}_p) \cong \mathbb{Z}_{8 \cdot r}$, where r is a large prime.
- We get cofactor $h = 8$ and $d = \gcd(h, p-1) = 8$.
- Thus, Divide-and-Pair requires **0 point halvings** and 1 Tate pairing of degree 8.

This is the most favorable case among the cofactor-8 curves: since $8 \mid p-1$, the degree-8 Tate pairing is non-degenerate over $\mathbb{F}_p$, and the entire subgroup membership test reduces to evaluating a single octic residuosity symbol $\chi_8(\zeta) = \zeta^{(p-1)/8}$ on the Miller function value $\zeta = f_{8,T_8}(R)$, where T_8 is a precomputed 8-torsion point. No point halvings are required.

4 Optimizing SMT on specific curves

In this section, we discuss low-level optimizations to divisions and pairings on the curves Curve25519, Jubjub, Four$\mathbb{Q}$, Curve448, and GC256A. All the five target curves are given in twisted Edwards model, which provides the fastest group law for general scalar multiplication. The Divide-and-Pair itself is model-agnostic—it only requires divisions and residuosity checks, both of which are algebraic invariants. In our implementation, however, we map to other models at specific stages: the Pornin Montgomery (u, w) model for the division-free halving procedure (Section 4.1), and the short Weierstrass model for the degree-≥ 7 Miller function evaluations, where polynomial line functions enable clean denominator absorption (Section 4.2.3).

Pairings of small degree. For the pairings, we are in a highly specific situation compared to ordinary pairings on elliptic curves: the pairings we compute are of the form $t_n(P, -)$ for small $n \in \{2, 4, 7, 8\}$ on a fixed curve with a fixed torsion point P. In general, a (reduced) Tate pairing can be computed as

$$t_n(P, R) = f_{n,P}(R)^{\frac{q-1}{n}},$$

[3] *Bandersnatch* [MSZ24] is yet another companion curve for BLS12-381 in the twisted Edwards form. Unlike Jubjub, Bandersnatch possesses a small CM discriminant (namely -8), admitting the standard GLV scalar multiplication technique. Moreover, cheap SMT is already available for the given curve in gnark-crypto thanks to [El 25], since just two Legendre symbols are enough for it.

where $q = p$ or $q = p^2$. Hence, for our application, we can precompute these specific Miller functions, and so the cost to evaluate $f_{n,P}(R)$ on any $R \in E(\mathbb{F}_q)$ for small n is only a few finite field multiplications. The cost for computing $t_n(P, R)$ is therefore dominated by the *final exponentiation* part $\zeta \mapsto \zeta^{\frac{q-1}{n}}$, where $\zeta = f_{n,P}(R)$. In Divide-and-Pair, however, we only need to know if $t_n(P, R)$ is *trivial*, which translates to the question of whether ζ is an n-th power. Hence, instead of the full exponentiation, we perform a residuosity check on ζ. We analyse optimizations of these residuosity computations in Section 4.2.

4.1 Optimizing Point Halvings

We describe the point halving procedure on twisted Edwards curves, following Pornin [Por22]. Let $\mathcal{E}(\tilde{a}, \tilde{d}) : \tilde{a}x^2 + y^2 = 1 + \tilde{d}x^2y^2$ be a twisted Edwards curve over $\mathbb{F}_p$. The curve is birationally equivalent to the Pornin Montgomery curve

$$\mathcal{C}(A, B) : uw^2 = u^2 + Au + B,$$

where $A = 2(\tilde{a}+\tilde{d})$, $B = (\tilde{a}-\tilde{d})^2$, with the birational maps $u = (\tilde{a}-\tilde{d})(1+y)/(1-y)$ and $w = 2/x$. We work on this *un-normalized* model (with $B \neq 1$) rather than the standard Montgomery form $v^2 = u^3 + A'u^2 + u$ (which requires $A' = 2(\tilde{a}+\tilde{d})/(\tilde{a}-\tilde{d})$) in order to avoid a division by $(\tilde{a}-\tilde{d})$ in the birational map. Since the point halving procedure operates entirely on isomorphic curves $\mathcal{C}(Ae^2, Be^4)$ tracked via a scaling factor e, the value of B does not affect the halving formulas—it only appears in the precomputed constant $B' = A^2 - 4B$.

We now describe the halving procedure. For curves $\mathcal{C}(A, B)$, $\mathcal{C}(A', B')$, and $\mathcal{C}(A'', B'')$ with $A' = -2A$, $B' = A^2-4B$, $A'' = 4A$, $B'' = 16B$, the isogenies $\psi_1 : \mathcal{C}(A, B) \to \mathcal{C}(A', B')$ and $\psi_2 : \mathcal{C}(A', B') \to \mathcal{C}(A'', B'')$ together with the isomorphism $\text{iso} : \mathcal{C}(A'', B'') \to \mathcal{C}(A, B)$, $(u, w) \mapsto (u/4, w/2)$, satisfy $[2] = \text{iso} \circ \psi_2 \circ \psi_1$. To halve, we invert iso, ψ_2, and ψ_1, in that order.

Given a point (u, w) on $\mathcal{C}(Ae^2, Be^4)$, one halving step proceeds as follows:

1. **Inverse** iso**:** $(u_s, w_s) = (4u, 2w)$.
2. **Inverse** ψ_2**:** Set $w' = \sqrt{u_s}$; if $u_s \notin \text{QR}(\mathbb{F}_p)$, the point is not halvable and we return **false**. Then $u' = (u_s - A'e^2 - w'w_s)/2$.
3. **Inverse** ψ_1**:** If $u' \in \text{QR}(\mathbb{F}_p)$, set $w_R = \sqrt{u'}$ and $u_R = (w_R^2 - Ae^2 - w_Rw')/2$. If $u' \notin \text{QR}(\mathbb{F}_p)$, we avoid division by switching to an isomorphic curve: for Curve25519, where $p \equiv 5 \pmod 8$ and 2 is a quadratic non-residue, the product $2u'$ is a QR. We compute $t = \sqrt{2u'}$ and update: $w' \leftarrow w' \cdot t$, $w_R \leftarrow \sqrt{2B'} \cdot e^2$, $e \leftarrow e \cdot t$, then $u_R = (w_R^2 - Ae^2 - w_Rw')/2$ and $w_R \leftarrow -w_R$. For Curve448 and GC256A, where $p \equiv 3 \pmod 4$, we instead use $-u'$ which is a QR, compute $t = \sqrt{-u'}$, and update: $w' \leftarrow -w' \cdot u'$, $w_R \leftarrow t \cdot \sqrt{-B'} \cdot e^2$, $e \leftarrow e \cdot u'$.

For Curve25519 (cofactor 8), we apply this point halving procedure twice in a loop, then check whether a third halving is possible by testing $u_R \in \text{QR}(\mathbb{F}_p)$ via the Legendre symbol—without actually computing the third halving. For Curve448 and GC256A (cofactor 4), we only perform a single point halving, followed by a Legendre symbol.

Scaled Pornin–Montgomery coordinates. In the implementation of the halving-based SMTs (Curve25519, Curve448, GC256A, and Jubjub), we avoid the initial Edwards-to-Montgomery inversion entirely. Instead of computing affine Pornin–Montgomery coordinates, we map directly to the scaled model $\mathcal{C}(Ae^2, Be^4)$ with

$$e = x(1 - y), \qquad u = (\tilde{a} - \tilde{d})(1 + y)x^2(1 - y), \qquad w = 2(1 - y),$$

and carry the scale factor e through the whole halving chain. Division by 2 in the halving uses the field's Halve operation (conditional add of p followed by a right shift) rather than a full inversion. Thus the entire halving procedure, including the QR/NQR twist, is *division-free*. This is the same as [Por22].

4.2 Optimizing Residuosity

The Divide-and-Pair algorithm reduces the final pairing check to a residuosity test: is a given field element ζ an n-th power? We describe optimized methods for each degree n arising in our five curves, in order of increasing complexity.

4.2.1 Quadratic residuosity ($n = 2$): Legendre symbol.

For all Pornin methods, we check $\chi_2(\zeta) = \zeta^{(q-1)/2} \stackrel{?}{=} 1$. This is the classical Legendre symbol. In our `gnark-crypto`-based implementation, it is computed via Pornin's binary-GCD algorithm [Por22] when the field modulus is not full-width; for full-width fields such as Curve448 and GC256A, we fall back to Euler criterion using a fixed addition chain. It serves as the cost baseline: all higher-degree checks are measured relative to one Legendre evaluation.

4.2.2 Quartic residuosity ($n = 4$).

For Curve25519, we check $\chi_4(\zeta) = \zeta^{(q-1)/4} \stackrel{?}{=} 1$. We describe two methods with complementary trade-offs.

Since the halving procedure leaves us on an isomorphic curve scaled by e, the Miller function argument becomes $\zeta = u_R \cdot (w_R + 2se)^2$ where $s = \sqrt{A+2\sqrt{B}} \in \mathbb{F}_p$ (for Curve25519, $s = 2i$ with $i = \sqrt{-1}$). The factor e enters because the 4-torsion point on the isomorphic curve $\mathcal{C}(Ae^2, Be^4)$ has w-coordinate se rather than s. Since $\chi_4(e^4) = 1$, any power of e that appears as a fourth power in ζ does not affect the quartic residuosity test.

Using addition chains. For $q \equiv 5 \pmod 8$ (the case of Curve25519), the quartic residue symbol can be computed via the identity

$$\chi_4(\zeta) = \zeta \cdot \left(\zeta^{(q-5)/8}\right)^2.$$

This reduces the quartic symbol to the *same* exponentiation used for Atkin's square-root algorithm needed in the halving step. More generally, for any $q \equiv 1 \pmod 4$, one can express $(q-1)/4$ in terms of a fixed addition chain. The cost is one addition-chain exponentiation + 1 squaring + 1 multiplication—essentially one square root.

This replaces two halvings (2 exponentiations) + Legendre with one halving + one quartic residuosity check (1 exponentiation), saving one exponentiation overall.

Using a Euclidean algorithm over $\mathbb{Z}[i]$. A faster approach avoids exponentiation entirely. The quartic residue symbol $\chi_4(\zeta) = [\zeta/\pi]$ is the biquadratic residue symbol in $\mathbb{Z}[i]$, where π is a primary Gaussian prime with norm q. Following Weilert [Wei02], we compute $[\zeta/\pi]$ via a Euclidean algorithm over $\mathbb{Z}[i]$:

1. **Phase 1:** One Gaussian Euclidean step reduces the m-bit input $\zeta + 0i$ modulo π to a Gaussian integer γ whose real and imaginary parts each have $\sim m/2$-bit.

2. **Phase 2:** Iterated Euclidean division in $\mathbb{Z}[i]$ on two operands with $\sim m/2$-bit parts, tracking the quartic symbol via supplementary laws and quartic reciprocity.

At each step, the remainder $\gamma = \alpha - s\beta$ is decomposed as $\gamma = (1+i)^j \cdot i^k \cdot \gamma''$ where γ'' is *primary* ($\gamma'' \equiv 1 \bmod (1+i)^3$). Division by $(1+i)$ costs two additions and one right-shift:

$$\frac{a+bi}{1+i} = \frac{a+b}{2} + \frac{b-a}{2}\, i.$$

The symbol correction uses the supplementary laws for primary $\beta = b_0 + b_1 i$ [Wei02], and with $\alpha = a_0 + a_1 i$:

$$\left[\frac{i}{\beta}\right] = i^{(1-b_0)/2}, \tag{1}$$

$$\left[\frac{1+i}{\beta}\right] = i^{(b_0-b_1-b_1^2-1)/4}, \tag{2}$$

$$\left[\frac{\alpha}{\beta}\right]\left[\frac{\beta}{\alpha}\right]^{-1} = (-1)^{(a_0-1)(b_0-1)/4}, \tag{3}$$

depending only on the low bits (mod 8) of the operands. Phase 2 terminates in $O(\log q)$ steps. The total cost is approximately 3× a Legendre symbol—far cheaper than an exponentiation.

This replaces one halving + Legendre (2 exponentiations total) with one halving + a quartic residuosity check costing ≈3 Legendre symbols, saving nearly one full exponentiation.

Gaussian prime. We precompute the Gaussian prime $\pi = \pi_{\mathrm{re}} + \pi_{\mathrm{im}}\, i$ with $\pi_{\mathrm{re}}^2 + \pi_{\mathrm{im}}^2 = q$ using Cornacchia's algorithm [Cor08] and normalize π to be primary.

4.2.3 Octic residuosity over $\mathbb{F}_p$ ($n = 8$).

For Jubjub ($d = 8$, $p \equiv 1 \bmod 8$), we check $\chi_8(\zeta) = \zeta^{(p-1)/8} \stackrel{?}{=} 1$.

The Miller function $f_{8,T_8}(R)$ is evaluated on the short Weierstrass model $Y^2 = X^3 + aX + b$, obtained from the Pornin Montgomery (u, w) model via $X = u + A/3$ and $Y = uw$, using a precomputed 8-torsion point T_8 with $[2]T_8 = T_4$, $[4]T_8 = T_2$ (a 2-torsion point with $Y = 0$). The Miller algorithm for $n = 8 = (1000)_2$ performs three doubling steps:

1. $T_8 \to T_4$: tangent line ℓ_1 at T_8, vertical $v_1 = X_R - X_{T_4}$,
2. $T_4 \to T_2$: tangent line ℓ_2 at T_4, vertical $v_2 = X_R - X_{T_2}$,
3. $T_2 \to \mathcal{O}$: tangent at 2-torsion is vertical, giving $g_3 = X_R - X_{T_2} = v_2$.

The Miller recurrence gives $f_8 = g_1^4 \cdot g_2^2 \cdot g_3$ where $g_j = \ell_j / v_j$ for $j = 1, 2$. Multiplying by $v_1^8 \cdot v_2^8$ (both 8-th powers, invisible to χ_8) yields the *inversion-free* expression

$$f_8^{\mathrm{equiv}} = \ell_1^4 \cdot v_1^4 \cdot \ell_2^2 \cdot v_2^7,$$

requiring only field multiplications and squarings—no inversions.

We use the Weierstrass model, rather than the Pornin Montgomery (u, w) model, because the line functions $\ell_j(R) = Y_R - Y_{[j]T_8} - \lambda(X_R - X_{[j]T_8})$ and $v_j(R) = X_R - X_{[2^j]T_8}$ are *polynomial* in (X_R, Y_R). On the Pornin–Montgomery model, Vélu's isogeny formula produces *rational* expressions of the form $(u_R u_{[j]T_8} - B)/(u_R - u_{[j]T_8})$ at each step. Divisions are avoidable in any model by the usual denominator-elimination trick from pairings—a/b and $a \cdot b^{n-1}$ lie in the same class modulo n-th powers—but eliminating the extra denominators introduced at every step of the chain costs additional multiplications, whereas the Weierstrass line functions are division-free from the start.

The remaining three subsections concern residuosity checks over the quadratic extension $\mathbb{F}_{p^2}$, arising for Curve448, GC256A (quartic) and Four$\mathbb{Q}$ (octic and septic).

4.2.4 Quartic residuosity over $\mathbb{F}_{p^2}$ ($n = 4$).

Since the characteristic $p \equiv 3 \pmod 4$ for Curve448 and GC256A, there exists no quartic character over $\mathbb{F}_p$. However, the degree-4 Tate pairing t_4 is non-degenerate over $\mathbb{F}_{p^2} = \mathbb{F}_p(i) = \mathbb{F}_p[x]/(x^2+1)$. This enables a quartic residuosity check in $\mathbb{F}_{p^2}$, eliminating all point halvings.

Miller function in $\mathbb{F}_{p^2}$. On the Pornin Montgomery model, let T_2 be the 2-torsion point with u-coordinate $u_{T_2} = c_0 + c_1 i \in \mathbb{F}_{p^2}$ (where c_0, c_1 depend on the curve parameters). We precompute the 4-torsion point $T_4 = (u_{T_4}, w_{T_4}) \in E(\mathbb{F}_{p^2}) \setminus E(\mathbb{F}_p)$ by halving T_2. The tangent line at T_4 defines precomputed constants $\lambda, C \in \mathbb{F}_{p^2}$ such that the line evaluation at a rational point $R = (u_R, w_R)$ is $\ell_1(R) = w_R - \lambda\, u_R - C$. The degree-4 Miller function value $\alpha = \ell_1(R)^2 \cdot (u_R - c_0 + c_1 i) \in \mathbb{F}_{p^2}$ is computed with a few $\mathbb{F}_p$-multiplications and one $\mathbb{F}_{p^2}$-squaring.

Torus compression. The residuosity check is the exponentiation

$$\chi_4(\alpha) = \alpha^{(p^2-1)/4} = \left(\alpha^{p-1}\right)^{(p+1)/4}.$$

Since $p \equiv 3 \pmod 4$, the Frobenius endomorphism acts as complex conjugation, $\alpha^p = \bar{\alpha}$, so the inner exponentiation is a single Frobenius quotient $g = \alpha^{p-1} = \bar{\alpha}/\alpha$, and it remains to decide whether $g^{(p+1)/4} = 1$. The quotient g has norm $g\bar{g} = 1$, i.e., it lies in the algebraic torus $\mathcal{T}_2(\mathbb{F}_p) \simeq \mu_{p+1} \subset \mathbb{F}_{p^2}^*$, which is stable under exponentiation. A norm-1 element equals 1 if and only if its trace $t = g + g^{-1} = g + \bar{g} \in \mathbb{F}_p$ equals 2 (indeed, $g + 1/g = 2$ forces $(g-1)^2 = 0$), so we can follow traces in $\mathbb{F}_p$ instead of torus elements in $\mathbb{F}_{p^2}$. For $\alpha = a_0 + a_1 i$, the trace of g is computed projectively as $N = 2(a_0^2 - a_1^2)$, $D = a_0^2 + a_1^2$, followed by a single $\mathbb{F}_p$-inversion to obtain $t = N/D$. Traces of powers of g are governed by the Lucas V-sequence [Gal12, Section 6.3.2]: writing $t = \mathrm{Tr}(g)$, one has $\mathrm{Tr}(g^k) = V_k(t)$, where $V_0(t) = 2$, $V_1(t) = t$, and $V_k(t) = t \cdot V_{k-1}(t) - V_{k-2}(t)$. The check $\chi_4(\alpha) = 1$ is therefore equivalent to $V_e(t) = 2$ with $e = (p+1)/4$. Trading full extension-field elements for their traces in a subfield is a classical idea, underlying the LUC [SL93], XTR [LV00], and CEILIDH [RS03] cryptosystems.

PRAC chain. The Lucas V-sequence satisfies the addition and doubling rules $V_{j+k} = V_j V_k - V_{j-k}$ and $V_{2j} = V_j^2 - 2$, so $V_e(t)$ is evaluated via a PRAC differential addition chain [Mon92]. PRAC maintains the triple $(V_d, V_e, V_{|d-e|})$ and at each step applies one of a small set of update rules, each costing 1–4 $\mathbb{F}_p$-multiplications or squarings. The key advantage is that after the initial trace computation, the entire evaluation uses only $\mathbb{F}_p$-arithmetic—no $\mathbb{F}_{p^2}$-operations are needed. For Curve448, the PRAC chain for $(p+1)/4$ costs 632 $\mathbb{F}_p$-ops (vs. 890 for a standard binary ladder, a 29% saving). For GC256A, it costs 470 $\mathbb{F}_p$-ops (7% saving).

The overall method requires *no halvings* and no $\mathbb{F}_{p^2}$-inversions: just a few $\mathbb{F}_p$-multiplications for the Miller value, one $\mathbb{F}_p$-inversion for the affine trace, and the PRAC chain.

4.2.5 Octic residuosity over $\mathbb{F}_{p^2}$ ($n = 8$).

For the 2^3-part of Four$\mathbb{Q}$ over $\mathbb{F}_{p^2}$, we check $\chi_8(\zeta) = \zeta^{(p^2-1)/8} \stackrel{?}{=} 1$.

The degree-8 Miller function $f_{8,T_8}(R)$ is evaluated on the short Weierstrass model over $\mathbb{F}_{p^2}$ using the same three-step doubling chain as in Section 4.2.3 (precomputed 8-torsion point T_8 with $[2]T_8 = T_4$, $[4]T_8 = T_2$), yielding the division-free expression $f_8^{\text{equiv}} = \ell_1^4 \cdot v_1^4 \cdot \ell_2^2 \cdot v_2^7$ via the same denominator-absorption trick.

The Frobenius endomorphism $\zeta \mapsto \bar{\zeta}$, which equals complex conjugation in $\mathbb{F}_{p^2} = \mathbb{F}_p(i)$, gives $\zeta^p = \bar{\zeta}$, so $\zeta^{p-1} = \bar{\zeta}/\zeta$ and

$$\chi_8(\zeta) = (\bar{\zeta}/\zeta)^{(p+1)/8}.$$

Since $p = 2^{127} - 1$ yields $p + 1 = 2^{127}$ and $(p+1)/8 = 2^{124}$, the final exponentiation is just 124 $\mathbb{F}_{p^2}$-squarings after one $\mathbb{F}_{p^2}$-inversion—no multiplications needed.

To avoid the $\mathbb{F}_{p^2}$ arithmetic, we use the torus/trace approach from Section 4.2.4: the Frobenius quotient $g = \bar{\zeta}/\zeta$ lies in $T_2(\mathbb{F}_p) \simeq \mu_{p+1}$, and we compute its trace $t = g + g^{-1} = 2(a^2 - b^2)/(a^2 + b^2) \in \mathbb{F}_p$ projectively. The check $\chi_8(\zeta) = 1$ then becomes $V_e(t) = 2$ with $e = (p+1)/8 = 2^{124}$. Since e is a power of two, this is simply 124 iterations of $t \mapsto t^2 - 2$, which cost one $\mathbb{F}_p$-squaring and one $\mathbb{F}_p$-subtraction per step.

4.2.6 Septic residuosity over $\mathbb{F}_{p^2}$ ($n = 7$).

For the 7^2-part of the Four$\mathbb{Q}$ cofactor, we check $\chi_7(\zeta) = \zeta^{(p^2-1)/7} \stackrel{?}{=} 1$ on the Weierstrass model over $\mathbb{F}_{p^2}$. While one could take $\zeta = f_{7,T_7}(R)$, we instead use the divisor-equivalent degree-8 Miller value $\zeta = f_{8,T_7}(R)$. Indeed,

$$\begin{aligned} \operatorname{div}(f_{8,T_7}) &= 8(T_7) - ([8]T_7) - 7(\mathcal{O}) \\ &= 8(T_7) - (T_7) - 7(\mathcal{O}) \\ &= 7(T_7) - 7(\mathcal{O}) \\ &= \operatorname{div}(f_{7,T_7}), \end{aligned} \tag{4}$$

Since $[8]T_7 = T_7$, the Miller loop for $f_{8,T_7}(R)$ consists of three doubling steps

$$T_7 \to [2]T_7 \to [4]T_7 \to [8]T_7 = T_7,$$

with the intermediate points $[2]T_7$ and $[4]T_7$ and the tangent slopes at T_7, $[2]T_7$, and $[4]T_7$ precomputed.

Norm-accumulated Miller function. The Frobenius identity $\zeta^{(p^2-1)/7} = \mathrm{N}(\zeta)^{(p-1)/7} \in \mathbb{F}_p$ holds since $7 \mid p-1$ (for $p = 2^{127} - 1$), where N denotes the norm $N_{\mathbb{F}_{p^2}/\mathbb{F}_p}$. Thus, rather than computing the full Miller value $\zeta \in \mathbb{F}_{p^2}$ and then taking its norm, we exploit the multiplicativity of N to accumulate norms *inside* the Miller loop. In more detail, we use ℓ_j and v_j to denote the tangent line and vertical line passing through $[j]T_7$, respectively. Each line evaluation $\ell_j(R)$ and vertical $v_j(R)$ is an element of $\mathbb{F}_{p^2}$ (since $T_7 \in E(\mathbb{F}_{p^2}) \setminus E(\mathbb{F}_p)$), but its norm $\mathrm{N}(\ell_j) = \ell_{j,0}^2 + \ell_{j,1}^2$ is computed immediately in $\mathbb{F}_p$ at cost $2\mathbf{S}_p + 1\mathbf{A}$. Based on (4), the norm of the Miller value can be written as

$$\mathrm{N}(f_{8,T_7}) = \mathrm{N}(f_{7,T_7}) = \frac{\mathrm{N}(\ell_1)^4 \cdot \mathrm{N}(\ell_2)^2 \cdot \mathrm{N}(\ell_4)}{\mathrm{N}(v_2)^4 \cdot \mathrm{N}(v_4)^2 \cdot \mathrm{N}(v_1)}. \tag{5}$$

Furthermore, as in the case of Jubjub, multiplying the right-hand side of Eq. (5) by $\mathrm{N}(v_2)^7 \cdot \mathrm{N}(v_4)^7 \cdot \mathrm{N}(v_1)^7$ gives the following inversion-free expression:

$$\mathrm{N}(f_{8,T_7})^{\mathrm{equiv}} = \mathrm{N}(\ell_1)^4 \cdot \mathrm{N}(\ell_2)^2 \cdot \mathrm{N}(\ell_4) \cdot \mathrm{N}(v_2)^3 \cdot \mathrm{N}(v_4)^5 \cdot \mathrm{N}(v_1)^6.$$

Final exponentiation. The septic check reduces to:

1. Compute $\mathrm{N}(f_{8,T_7})^{\mathrm{equiv}} \in \mathbb{F}_p$ via the norm-accumulated Miller loop above.

2. Compute $\left(\mathrm{N}(f_{8,T_7})^{\mathrm{equiv}}\right)^{(p-1)/7}$ in $\mathbb{F}_p$: the exponent

$$(p-1)/7 = \texttt{0x1 249 249 249 249 249 249 249 249 249 249 249 2}$$

has a regular repeating pattern (period 3), which a 131-step addition chain (generated by `mmcloughlin/addchain`) exploits.

3. Check if the result equals 1.

4.3 Constant-time considerations

Side-channel resistance is important for deployed implementations. We briefly discuss the timing profile of each building block.

Constant-time operations. All building blocks except one (see below) execute a fixed sequence of field operations independent of the input point R: halvings use a constant-time conditional move for the QR/NQR branch [Por22]; all exponentiations (square roots, Legendre, quartic/octic/septic symbols) use fixed addition chains; Miller function evaluations are fixed doubling or double-and-add chains on precomputed torsion points (including the norm-accumulated septic variant); torus compressions, PRAC chains, and the 124 iterations of $t \mapsto t^2 - 2$ are fixed loops; and coordinate conversions use inversions computed by Pornin's constant-time binary GCD [Por20] or a fixed addition chain of $p - 2$.

Variable-time exception. The Weilert GCD for the quartic symbol (Section 4.2.2) is variable-time: the number of Euclidean iterations depends on the input. This is acceptable when SMT operates on public data (e.g., verifying a received point). When constant-time is required, the addition-chain variant (Section 4.2.2) should be used instead.

Control flow. The SMT functions branch on low-order point detection (identity, 2- or 4-torsion). Since the input to SMT is typically public, this does not leak secret information.

5 Performance

All benchmarks are run on an AWS `c7a.8xlarge` instance (AMD EPYC 9R14) using Go 1.26 with the `purego` build tag. Field arithmetic uses `gnark-crypto` [BPE+26] generated pure-Go code. Scalar multiplications use projective coordinates. The naive baseline computes $[r]R$ with a fixed short addition chain for r, generated with the `addchain` tool[4], rather than a generic double-and-add loop. Each benchmark cycles over 256 uniformly random input points: the running time of the halving-based methods depends on the input (halving computes one extra square root when an intermediate value is a non-residue), and repeatedly timing a single point also lets the branch predictor learn the data-dependent branch pattern of the field arithmetic, understating the cost of every method on random inputs. Our source code is open-source and available at [DEKR26]:

`https://github.com/yelhousni/divide-and-pair`.

5.1 Curve25519

Divide-and-Pair on Curve25519 requires one point halving and a quartic symbol (Section 4.2.2). We state the benchmarking results in Table 3.

The Weilert GCD variant (23.8 µs) is approximately 1.6× faster than Pornin's method and avoids multi-precision exponentiation entirely.

[4] `https://github.com/mmcloughlin/addchain`

Table 3: SMT on Curve25519 ($h = 8$, $d = 4$).

Method	Time (µs)	Speedup
Naive ($[r]R \stackrel{?}{=} \mathcal{O}$)	66	1
Pornin (2 halvings + Legendre)	37.5	1.8
Ours: 1 halving + χ_4 (exp)	24.2	2.7
Ours: 1 halving + χ_4 (GCD)	23.8	2.8

Comparison with `crrl`. Pornin [Por22] reports 44 835 cycles for his original Curve25519 subgroup test on a 64-bit x86 system (Intel i5-8259U "Coffee Lake", clocked at 2.3 GHz). The corresponding code is available in the `crrl` library.[5] Benchmarking that implementation on our AWS machine gives 35 900 cycles, or about 13.7 µs, which is faster than both our pure-Go Pornin implementation and our two divide-and-pair implementations. The main reason is the field backend: `crrl` uses hand-optimized assembly arithmetic for the Curve25519 prime with five 51-bit limbs and tailored reduction modulo 19, whereas our code uses generic pure-Go arithmetic on four 64-bit limbs. Our design choice, for this article, is to keep the implementation simple and uniform across all curves. To confirm that the divide-and-pair advantage is algorithmic rather than an artifact of the Go backend, we also implemented the χ_4 (exp) variant directly in `crrl`; the code is included in the supplementary material. On the same AWS machine, this implementation runs in 23 200 cycles, or about 8.8 µs, validating a 1.5× speedup over Pornin's original method. We expect that a Rust implementation of Weilert's quartic residue symbol would increase this speedup.

5.2 Curve448

Divide-and-Pair on Curve448 requires a quartic torus/PRAC over $\mathbb{F}_{p^2}$ with 0 halvings (Section 4.2.4). We state the benchmarking results in Table 4.

Table 4: SMT on Curve448 ($h = 4$, $d = 4$ over $\mathbb{F}_{p^2}$).

Method	Time (µs)	Speedup
Naive ($[r]R \stackrel{?}{=} \mathcal{O}$)	263	1
Pornin (1 halving + Legendre)	100	2.6
Ours: quartic torus/PRAC (0 halvings)	74.5	3.5

The PRAC chain (632 $\mathbb{F}_p$-ops) gives approximately 1.3× over Pornin's method. The performance gap is more pronounced because Curve448 field is full-width 448 bits over 7 limbs. There is no spare bit to implement efficiently the binary-GCD-based Legendre symbol needed for Pornin's method. Hence, we fall back to an exponentiation by $(p-1)/2$ using an addition chain.

5.3 GC256A

Divide-and-Pair on GC256A requires the same torus/PRAC approach as Curve448 (Section 4.2.4). We state the benchmarking results in Table 5.

Our approach is 1.4× faster than Pornin's method; the PRAC chain costs 470 $\mathbb{F}_p$-ops. As with Curve448, the field backend matters here: GC256A is also full-width on four 64-bit limbs, so there is no spare bit for the binary-GCD Legendre and inverse routines

[5] https://github.com/pornin/crrl

Table 5: SMT on GC256A ($h = 4$, $d = 4$ over $\mathbb{F}_{p^2}$).

Method	Time (µs)	Speedup
Naive ($[r]R \stackrel{?}{=} \mathcal{O}$)	66	1
Pornin (1 halving + Legendre)	24.8	2.7
Ours: quartic torus/PRAC (0 halvings)	18.2	3.6

generated by `gnark-crypto`. Consequently, Pornin's path uses Euler criterion for the Legendre symbol and a generic inversion routine, which widens the gap in favor of the quartic torus/PRAC method.

5.4 Four$\mathbb{Q}$

Divide-and-Pair on Four$\mathbb{Q}$ requires an octic torus (Section 4.2.5) and two septic norm checks (Section 4.2.6), with 0 point divisions. Since $E[7](\mathbb{F}_{p^2}) \cong \mathbb{Z}_7 \times \mathbb{Z}_7$, two independent 7-torsion points are needed. We state the benchmarking results in Table 6.

Table 6: SMT on Four$\mathbb{Q}$ ($h = 2^3 \cdot 7^2$, $d = 2^3 \cdot 7$).

Method	Time (µs)	Speedup
Naive ($[r]R \stackrel{?}{=} \mathcal{O}$)	102	1
Endomorphism [DLZZ23] (ψ/φ decomposition)	83	1.2
Ours: octic (torus) + 2×septic (Norm)	7.7	13.2

We get a speed-up of 13.2× over naive and 10.8× over the endomorphism-based method adapted from [DLZZ23].

Four$\mathbb{Q}$ is also the one curve among our five where Koshelev's original criterion [Kos23; Kos24] applies as is: since $h = 392 \mid p^2 - 1$, membership is decided by pairings alone, with no point divisions. Our row "octic + 2×septic" is therefore not a new criterion but, to the best of our knowledge, the first efficient implementation of Koshelev's method—the novelty lies in the optimization (torus/trace octic check and norm-accumulated septic checks, Sections 4.2.5 and 4.2.6). We do not list Pornin's method here: the 7^2-part of the cofactor would require point divisions by 7, which, while theoretically possible, is impractical.

5.5 Jubjub

Divide-and-Pair on Jubjub requires a single octic symbol χ_8 with 0 point halvings (Section 4.2.3). We state the benchmarking results in Table 7.

Table 7: SMT on Jubjub ($h = 8$, $d = 8$).

Method	Time (µs)	Speedup
Naive ($[r]R \stackrel{?}{=} \mathcal{O}$)	63	1
Pornin (2 halvings + Legendre)	49	1.3
Ours: χ_8 (0 halvings)	7.0	9.0

This is the most favorable case: $8 \mid p-1$ gives non-degeneracy over $\mathbb{F}_p$, yielding 7.0× over Pornin's method.

5.6 Summary

Figure 1 compares all methods across the five curves on a logarithmic scale. Across all curves, Divide-and-Pair delivers 2.8–13.2× speedups over naive scalar multiplication (a fixed addition chain for r in projective coordinates) and 1.3–10.8× over the previous state-of-the-art.

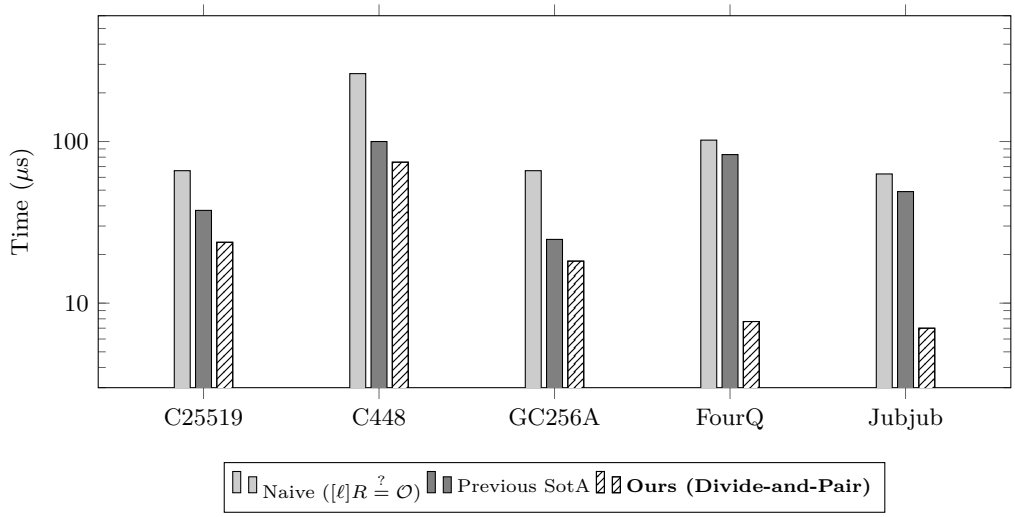

Figure 1: Subgroup membership testing times across all five curves (log scale).

Cofactor clearing. The companion code also exposes, for each curve, a `ClearCofactor` routine that computes $[h]P$ via a fixed chain of doublings (plus two additions for Four$\mathbb{Q}$, where $h = 392$), followed by a single projective-to-affine conversion. Under the same benchmark methodology, clearing costs 3.0 µs on Curve25519, 29.5 µs on Curve448 (where the field inversion dominates), 2.2 µs on GC256A, 4.6 µs on Four$\mathbb{Q}$ and 2.8 µs on Jubjub; the work grows with $\log h$ rather than $\log r$. Cofactor clearing is nevertheless not a substitute for membership testing: it maps a point *into* the prime-order subgroup instead of deciding membership and, as the Monero example from the introduction illustrates, it must be composed consistently with the surrounding protocol.

6 Conclusion

We have presented Divide-and-Pair, a general framework for SMT on elliptic curves with small cofactor. The key insight is that the cofactor h splits into a *degenerate* part, handled by point divisions, and a *non-degenerate* part $d = \gcd(n_1, q-1)$, handled by a Tate pairing, interpolating between the extremes of Pornin's all-division approach and Koshelev's all-pairing approach.

We instantiated Divide-and-Pair on five popular curves and developed curve-specific optimizations for each: Weilert's Euclidean algorithm for quartic symbols over $\mathbb{F}_p$, torus compression and PRAC chains for quartic symbols over $\mathbb{F}_{p^2}$, division-free Miller functions with torus and norm tricks for octic and septic symbols. Our pure-Go implementation achieves 2.8–13.2× speedups over naive scalar multiplication and 1.3–10.8g× over the previous state-of-the-art across all curves.

Future work. Several directions remain open. First, the division step in Divide-and-Pair is not restricted to halving: the cofactor h need not be a power of 2. For an arbitrary

prime factor ℓ of h/d, division by ℓ can be expensive as $[\ell]$ has degree ℓ^2. However, performance may be improved by factoring $[\ell]$ into $\hat{\varphi} \circ \varphi$, where φ is an isogeny of degree ℓ and can be computed using Vélu's formulas in $O(\ell)$ field operations. Extending the implementation to curves with odd cofactor factors (beyond Four$\mathbb{Q}$, where $7 \mid h$ but the septic part is handled by pairings) is a natural next step. In particular, extending the framework to pairing-friendly curves with larger cofactors would broaden the applicability of the approach. Second, a Euclidean algorithm for octic residuosity, analogous to Weilert's quartic GCD, could further reduce the cost of SMT on cofactor-8 curves like Jubjub, avoiding the $(p-1)/8$ exponentiation entirely.

Acknowledgments. Yu Dai was supported by the National Natural Science Foundation of China (No. 62402347). Dimitri Koshelev was supported by the Belarusian State Research Program "Convergence–2030", 1.02.1. Krijn Reijnders was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement ISOCRYPT – No. 101020788), by the Research Council KU Leuven grant C14/24/099 and by CyberSecurity Research Flanders with reference number VOEWICS02.

The authors express their gratitude to Kaizhan Lin and Oleg Taraskin for fruitful discussions related to this article. Furthermore, Dimitri Koshelev acknowledges the Cryptography and Graphs research group of the University of Lleida and in particular its head Josep Maria Miret Biosca for its hospitality even after the end of his post-doctoral contract.

References

[26a] *Things that use Ed25519*. 2026. URL: https://ianix.com/pub/ed25519-deployment.html (cit. on p. 2).

[26b] *Things that use X25519*. 2026. URL: https://ianix.com/pub/x25519-deployment.html (cit. on p. 2).

[ANS18] E. K. Alekseev, V. D. Nikolaev, and S. V. Smyshlyaev. "On the security properties of Russian standardized elliptic curves". In: *Mathematical Aspects of Cryptography* 9.3 (2018), pp. 5–32. DOI: 10.4213/mvk260 (cit. on p. 2).

[Ber06] D. J. Bernstein. "Curve25519: New Diffie–Hellman speed records". In: *Public Key Cryptography – PKC 2006*. Ed. by M. Yung, Y. Dodis, A. Kiayias, and T. Malkin. Vol. 3958. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2006, pp. 207–228. DOI: 10.1007/11745853_14 (cit. on pp. 1, 7).

[BLS04] P. S. L. M. Barreto, B. Lynn, and M. Scott. "On the selection of pairing-friendly groups". In: *Selected Areas in Cryptography – SAC 2003*. Ed. by M. Matsui and R. J. Zuccherato. Vol. 3006. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2004, pp. 17–25. DOI: 10.1007/978-3-540-24654-1_2 (cit. on p. 6).

[Bow17] S. Bowe. *BLS12-381: New zk-SNARK elliptic curve construction*. 2017. URL: https://electriccoin.co/blog/new-snark-curve (cit. on p. 2).

[BPE+26] G. Botrel, T. Piellard, Y. El Housni, A. Tabaie, G. Gutoski, I. Kubjas, and Y. J. Galteland. *Consensys/gnark-crypto*. Version v0.20.1. 2026. DOI: 10.5281/zenodo.5815453 (cit. on pp. 3, 15).

[Bru11] P. Bruin. "The Tate pairing for abelian varieties over finite fields". In: *Journal de Théorie des Nombres de Bordeaux* 23.2 (2011), pp. 323–328. DOI: 10.5802/jtnb.764 (cit. on p. 5).

[Cha26] ChatGPT. *CVE-2025-69277: Libsodium Ed25519 point validation bug fixed*. 2026. URL: https://windowsforum.com/threads/cve-2025-69277-libsodium-ed25519-point-validation-bug-fixed.395569 (cit. on p. 2).

[CL15] C. Costello and P. Longa. "Four$\mathbb{Q}$: Four-dimensional decompositions on a $\mathbb{Q}$-curve over the Mersenne prime". In: *Advances in Cryptology – ASIACRYPT 2015*. Ed. by T. Iwata and J. Cheon. Vol. 9452. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2015, pp. 214–235. DOI: 10.1007/978-3-662-48797-6_10 (cit. on pp. 2, 8, 9).

[CMR+23] L. Chen, D. Moody, A. Regenscheid, A. Robinson, and K. Randall. *Recommendations for discrete logarithm-based cryptography: Elliptic curve domain parameters (NIST Special Publication 800-186)*. 2023. URL: https://csrc.nist.gov/publications/detail/sp/800-186/final (cit. on pp. 1, 7, 8).

[Cor08] G. Cornacchia. "Su di un metodo per la risoluzione in numeri interi dell'equazione $\sum_{h=0}^{n} C_h x^{n-h} y^h = P$". In: *Giornale di Matematiche di Battaglini* 46 (1908), pp. 33–90 (cit. on p. 12).

[DEKR26] Y. Dai, Y. El Housni, D. Koshelev, and K. Reijnders. *Divide-and-Pair: Faster subgroup membership testing on elliptic curves*. 2026. URL: https://github.com/yelhousni/divide-and-pair (cit. on pp. 4, 15).

[Den25] F. Denis. *A vulnerability in Libsodium*. 2025. URL: https://00f.net/2025/12/30/libsodium-vulnerability (cit. on p. 2).

[dGH+23] H. de Valence, J. Grigg, M. Hamburg, I. Lovecruft, G. Tankersley, and F. Valsorda. *The ristretto255 and decaf448 groups (RFC 9496)*. 2023. URL: https://datatracker.ietf.org/doc/rfc9496 (cit. on p. 2).

[DHK+24] Y. Dai, D. He, D. Koshelev, C. Peng, and Z. Yang. *Revisiting subgroup membership testing on pairing-friendly curves via the Tate pairing*. 2024. URL: https://eprint.iacr.org/2024/1790 (cit. on p. 6).

[di26] Libsodium designers and implementors. *Libsodium*. Version 1.0.22. 2026. URL: https://doc.libsodium.org (cit. on p. 2).

[dLA] H. de Valence, I. Lovecruft, and T. Arcieri. *The Ristretto group*. URL: https://ristretto.group/ristretto.html (cit. on p. 2).

[DLZZ23] Y. Dai, K. Lin, C.-A. Zhao, and Z. Zhou. "Fast subgroup membership testings for $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ on pairing-friendly curves". In: *Designs, Codes and Cryptography* 91.10 (2023), pp. 3141–3166. DOI: 10.1007/s10623-023-01223-7 (cit. on pp. 3, 9, 17).

[El 25] Y. El Housni. *A note on implementing subgroup membership using the Tate pairing*. 2025. URL: https://hackmd.io/@yelhousni/tate-subgroup (cit. on p. 9).

[Ele] Electric Coin Company. *What is Jubjub?* URL: https://bitzecbzc.github.io/technology/jubjub/index.html (cit. on pp. 2, 9).

[FST10] D. Freeman, M. Scott, and E. Teske. "A taxonomy of pairing-friendly elliptic curves". In: *Journal of Cryptology* 23.2 (2010), pp. 224–280. DOI: 10.1007/s00145-009-9048-z (cit. on p. 6).

[Gal12] S. D. Galbraith. *Mathematics of public key cryptography*. New York: Cambridge University Press, 2012. URL: https://doi.org/10.1017/CBO9781139012843 (cit. on pp. 3, 13).

[Ham15a] M. Hamburg. "Decaf: Eliminating cofactors through point compression". In: *Advances in Cryptology – CRYPTO 2015*. Ed. by R. Gennaro and M. Robshaw. Vol. 9215. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2015, pp. 705–723. DOI: 10.1007/978-3-662-47989-6_34 (cit. on p. 2).

[Ham15b] M. Hamburg. *Ed448-Goldilocks, a new elliptic curve*. 2015. URL: https://eprint.iacr.org/2015/625 (cit. on pp. 1, 8).

[Kin04] B. King. "A point compression method for elliptic curves defined over GF(2^n)". In: *Public Key Cryptography – PKC 2004*. Ed. by F. Bao, R. Deng, and J. Zhou. Vol. 2947. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2004, pp. 333–345. DOI: 10.1007/978-3-540-24632-9_24 (cit. on pp. 2, 6).

[Kos23] D. Koshelev. "Subgroup membership testing on elliptic curves via the Tate pairing". In: *Journal of Cryptographic Engineering* 13.1 (2023), pp. 125–128. DOI: 10.1007/s13389-022-00296-9 (cit. on pp. 2, 3, 6, 17).

[Kos24] D. Koshelev. "Correction to: Subgroup membership testing on elliptic curves via the Tate pairing". In: *Journal of Cryptographic Engineering* 14.1 (2024), pp. 127–128. DOI: 10.1007/s13389-023-00331-3 (cit. on pp. 2, 8, 17).

[LHT16] A. Langley, M. Hamburg, and S. Turner. *Elliptic Curves for Security*. RFC 7748. Jan. 2016. DOI: 10.17487/RFC7748. URL: https://www.rfc-editor.org/info/rfc7748 (cit. on pp. 7, 8).

[LL97] C. H. Lim and P. J. Lee. "A key recovery attack on discrete log-based schemes using a prime order subgroup". In: *Advances in Cryptology – CRYPTO 1997*. Ed. by B. S. Kaliski. Vol. 1294. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1997, pp. 249–263. DOI: 10.1007/BFb0052240 (cit. on p. 2).

[LV00] Arjen K. Lenstra and Eric R. Verheul. "The XTR Public Key System". In: *Advances in Cryptology — CRYPTO 2000*. Ed. by Mihir Bellare. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 1–19. ISBN: 978-3-540-44598-2 (cit. on p. 13).

[Mon92] P. L. Montgomery. *Evaluating recurrences of form $X_{m+n} = f(X_m, X_n, X_{m-n})$ via Lucas chains*. 1992. URL: https://cr.yp.to/bib/1992/montgomery-lucas.pdf (cit. on p. 13).

[MSZ24] S. Masson, A. Sanso, and Z. Zhang. "Bandersnatch: A fast elliptic curve built over the BLS12-381 scalar field". In: *Designs, Codes and Cryptography* 92.12 (2024), pp. 4131–4143. URL: https://doi.org/10.1007/s10623-024-01472-0 (cit. on p. 9).

[Por20] T. Pornin. *Optimized binary GCD for modular inversion*. 2020. URL: https://eprint.iacr.org/2020/972 (cit. on p. 15).

[Por22] T. Pornin. *Point-halving and subgroup membership in twisted Edwards curves*. 2022. URL: https://eprint.iacr.org/2022/1164 (cit. on pp. 2, 3, 6, 10, 11, 15, 16).

[Reij25] K. Reijnders. "The Tate profile". In: *Progress in Cryptology – LATINCRYPT 2025*. Ed. by D. Escudero and I. Damgård. Vol. 16129. Lecture Notes in Computer Science. Cham: Springer, 2025, pp. 209–233. DOI: 10.1007/978-3-032-06754-8_8 (cit. on p. 6).

[RS03] Karl Rubin and Alice Silverberg. "Torus-Based Cryptography". In: *Advances in Cryptology - CRYPTO 2003*. Ed. by Dan Boneh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 349–365. ISBN: 978-3-540-45146-4 (cit. on p. 13).

[SBA22] S. V. Smyshlyaev, D. M. Belyavskiy, and E. K. Alekseev. *GOST cipher suites for transport layer security (TLS) protocol version 1.2*. 2022. URL: https://datatracker.ietf.org/doc/rfc9189 (cit. on pp. 2, 8).

[Sho99] P. W. Shor. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer". In: *SIAM Review* 41.2 (1999), pp. 303–332. DOI: 10.1137/S0036144598347011 (cit. on p. 1).

[SL93] Peter J. Smith and Michael J. J. Lennon. "LUC: A New Public Key System". In: *Proceedings of the IFIP TC11, Ninth International Conference on Information Security: Computer Security*. IFIP/Sec '93. NLD: North-Holland Publishing Co., 1993, pp. 103–117. ISBN: 0444817484 (cit. on p. 13).

[Spa17] R. Spagni. *Disclosure of a major bug in CryptoNote based currencies*. 2017. URL: https://www.getmonero.org/2017/05/17/disclosure-of-a-major-bug-in-cryptonote-based-currencies.html (cit. on p. 2).

[Wei02] A. Weilert. "Fast computation of the biquadratic residue symbol". In: *Journal of Number Theory* 96.1 (2002), pp. 133–151. DOI: 10.1006/jnth.2002.2783 (cit. on pp. 3, 11, 12).