Point-Halving and Subgroup Membership in Twisted Edwards Curves

Thomas Pornin

NCC Group, thomas.pornin@nccgroup.com

6 September, 2022

Abstract. In this short note, we describe a process for halving a point on a twisted Edwards curve. This can be used to test whether a given point is in the subgroup of prime order ℓ, which is used by some cryptographic protocols. On Curve25519, this new test is about twice faster than the classic method consisting of multiplying the source point by ℓ.

1 Introduction

Twisted Edwards curves are elliptic curves that offer good performance with complete addition formulas that are relatively simple to implement in an efficient and safe way. However, they suffer from the *cofactor issue*: the order of a twisted Edwards curve is always a multiple of 4, and thus the curve cannot be a prime-order group. The curve will typically be chosen to have order $h\ell$ for some (big) prime ℓ, and (small) cofactor h, with h being a multiple of 4. The two most used twisted Edwards curves are Curve25519 (with $h = 8$ and $\ell \approx 2^{252}$)[3] and Curve448 (with $h = 4$ and $\ell \approx 2^{446}$)[7].

The existence of points on the curve that are not in the subgroup of prime order ℓ can lead to some serious issues, depending on the protocol in which the curve is used[5]. To obtain a prime order group out of a twisted Edwards curve, some "cofactor elimination" techniques can be used, leading to the Decaf/Ristretto groups[6,1]. However, for interoperability reasons, it may be required, in some situations, to use the plain twisted Edwards curve. We use here as an example the FROST scheme[8]; this is a threshold signature scheme that allows a quorum of private key share holders to conjointly compute a Schnorr signature on a message. The current draft for the FROST standard[4] defines several ciphersuites, over some existing elliptic curves or other groups. In the case of Curve25519 and Curve448, an explicit goal of the FROST draft is to achieve signature interoperability, i.e. that signatures generated with FROST can be verified with an unmodified Ed25519 or Ed448 implementation. This requires working with the plain curve. In order to avoid cofactor issues, an explicit validation that incoming curve points (e.g. public keys) are in the prime order subgroup is mandated (section 6.1):

> `DeserializeElement(buf): Implemented as specified in [RFC8032], Section 5.1.3. Additionally, this function validates that the resulting element is not the group identity element and is in the prime-order subgroup. The latter check can be implemented by multiplying the resulting point by the order of the group and checking that the result is the identity element.`

The classic method for this check is, as the FROST draft explains, to multiply the point to test by the prime ℓ, in order to check whether the result is the curve neutral point $\mathbb{O} = (0, 1)$. This operation is rather expensive; it is typically a bit faster than multiplying a point by any scalar (because the multiplier is the known integer ℓ, for which an efficient addition chain can be precomputed), but still has a cost in the same order of magnitude. In this note, we describe an alternate method for this subgroup membership test, which offers better performance: for Curve25519, it is about twice faster than multiplying by ℓ.

2 Isogenies and Point Halving

We consider here a twisted Edwards curve defined in the finite field $\mathbb{F}_q$ (of characteristic 5 or more). We designate by QR the set of quadratic residues in $\mathbb{F}_q$. The curve is the set of points $(x, y) \in \mathbb{F}_q \times \mathbb{F}_q$ that fulfill the curve equation $ax^2 + y^2 = 1 + dx^2y^2$ for two constants a and d. We suppose that $a \in QR$ and $d \notin QR$ (this is the case for both Curve25519, with $a = -1$, and Curve448, with $a = 1$). We denote $\mathcal{E}(a, d)$ this curve. On such curves:

- The neutral point is $\mathbb{O} = (0, 1)$.
- There is a single point of order 2: $N = (0, -1)$.
- There are two points of order 4.

The curve is birationally equivalent to a Weierstraß curve of equation $v^2 = u(u^2+Au+B)$ (denoted $\mathcal{C}(A, B)$) with $A = 2(a + d)$ and $B = (a - d)^2$, using the following map[9]:

$$\begin{aligned} f : \mathcal{E}(a, d) &\longrightarrow \mathcal{C}(2(a + d), (a - d)^2) \\ (x, y) &\longmapsto (u, v) = \left(\frac{(a - d)(1 + y)}{1 - y}, \frac{2(a - d)(1 + y)}{x(1 - y)}\right) \end{aligned}$$

This map is well-defined for all points other than $\mathbb{O}$ and N; for now, we suppose that the source point to halve is not one of these points.

On the Weierstraß curve, it is convenient to introduce a third coordinate $w = v/u = 2/x$. In (u, w) coordinates, the curve equation is $uw^2 = u^2 + Au + B$. This equation format is reminiscent of double-odd curves[10], though the curves considered here are not double-odd (their order is a multiple of 4, not twice an odd integer); indeed, for the curves we are interested in (with $a \in QR$ and $d \notin QR$), we have $B \in QR$ and $A^2 - 4B \notin QR$. We can nonetheless reuse part of the analysis in sections 2 and 3.7 of [10]; we thus define two additional curves $\mathcal{C}(A', B') = \mathcal{C}(-2A, A^2 - 4B)$ and $\mathcal{C}(A'', B'') = \mathcal{C}(4A, 16B)$, and the following functions:

$$\begin{aligned} \psi_1 : \quad \mathcal{C}(A, B) &\longrightarrow \mathcal{C}(A', B') \\ (u, w) &\longmapsto (u', w') = \left(w^2, -\frac{u - B/u}{w}\right) \\ \psi_2 : \quad \mathcal{C}(A', B') &\longrightarrow \mathcal{C}(A'', B'') \\ (u', w') &\longmapsto (u'', w'') = \left(w'^2, -\frac{u' - B'/u'}{w'}\right) \\ \theta : \mathcal{C}(A'', B'') &\longrightarrow \mathcal{C}(A, B) \\ (u'', w'') &\longmapsto (u, w) = \left(\frac{u''}{4}, \frac{w''}{2}\right) \end{aligned}$$

Function θ is an isomorphism (curves $\mathcal{C}(A, B)$ and $\mathcal{C}(A'', B'')$ are isomorphic to each other). Functions ψ_1 and ψ_2 are isogenies which are obtained by applying Vélu's formulas[11] on the 2-torsion subgroup $\{\mathbb{O}, N\}$ in $\mathcal{C}(A, B)$ and $\mathcal{C}(A', B')$, respectively. Therefore, for all points $P \in \mathcal{C}(A, B)$, we have:

$$2P = \theta(\psi_2(\psi_1(P)))$$

This implies that we can halve a given point $Q = (u''', w''')$ by computing inverses of θ, ψ_2 and ψ_1, successively.

It is trivial to invert θ:

$$u'' = 4u'''$$
$$w'' = 2w'''$$

To invert ψ_2, i.e. find (u', w') from (u'', w'') such that $(u'', w'') = \psi_2(u', w')$, we can apply the following formulas:

$$w' = \sqrt{u''}$$
$$u' = (w'^2 - A' - w'w'')/2$$

The second formula leverages the curve equation: $u' + B'/u' = w'^2 - A'$, therefore $u' - B'/u' = 2u' - (u' + B'/u') = 2u' - w'^2 + A'$. These formulas work as long as u'' is a square; if $u'' \notin QR$, then there is no solution, which means that the source point is not the double of any other point on the curve.

The inversion of ψ_2 yields two solutions, depending on which square root of u'' we use. The two solutions are (u', w') and $(B'/u', -w')$. Note that $B' = A^2 - 4B = 16ad \notin QR$; thus, exactly one of u' and B'/u' is a square.

Inversion of ψ_1 is similar to the case of ψ_2:

$$w = \sqrt{u'}$$
$$u = (w^2 - A - ww')/2$$

As noted above, exactly one of the two possible antecedents of (u'', w'') by ψ_2 has a square u' coordinate; using that point, ψ_1 can always be inverted, and this finally yields a point $P = (u, w)$ such that $2P = Q$. There again, there are two solutions, depending on the choice of square root; the other solution is $(B/u, -w)$.

The following salient points must be noted:

- A point (u, w) can be halved if and only if $u \in QR$.
- Halving implies computing two square roots in $\mathbb{F}_q$.
- Formally, the choice of the right u' (the one which is a square) seems to require an extra square root attempt or Legendre symbol computation; however, it is often possible to modify the square root extraction process so that on failure it returns a predictable value that allows computing the square root of B'/u' instead, in case $u' \notin QR$, with negligible extra work. This will be explained in the next section.

The output point (u, w) can be converted back to twisted Edwards curve coordinates (x, y) using the inverse birational map f^{-1}; see [9] for details.

3 Subgroup Membership Test

We now consider the functionality we are interested in: given a point P on a twisted Edwards curve $\mathcal{E}(a, d)$ of order $h\ell$ (with ℓ an odd prime), find out whether P is in the subgroup of prime order ℓ or not. We limit ourselves to the curves such that:

- $h = 2^t$ for some integer $t \geq 2$.
- There is a point of order exactly h on the curve.

The second property means that the subgroup of h-torsion points is homomorphic to $\mathbb{Z}_h$, and not $\mathbb{Z}_i \times \mathbb{Z}_j$ for some integers i and j. We note that both Curve25519 and Curve448 match these properties. These properties imply the following:

- $a \in QR$ and $d \notin QR$ (i.e. the curve is in the case covered in section 2).
- A point Q on the curve is in the subgroup of order ℓ if and only if $Q = hP$ for some point P.
- For $1 \leq i \leq t$, if point $Q_i = 2^i P$ for some point P, then there are exactly 2^i distinct points P such that $Q_i = 2^i P$. Moreover, there are two points R such that $Q_i = 2R$, and each of these two points is such that $R = 2^{i-1}P$ for some point P.

In other words, we can check whether a point Q is in the subgroup of order ℓ by halving it t times. At each halving, we are free to use either of the two solutions; both work equally well. If we can compute t successive halvings, then Q is in the subgroup; otherwise, it is not. Each halving uses the process described in section 2. We can also replace the last halving by a cheaper single Legendre computation on u, because we are only interested in knowing whether halving is possible, not in learning the final half point; similarly, we do not have to convert back to twisted Edwards coordinates. In total, this means that we can check membership of the prime order subgroup with the following cost (ignoring a few cheap multiplications and additions in the field):

- Curve25519: four square roots and one Legendre symbol.
- Curve448: two square roots and one Legendre symbol.

In practice, the source point Q is provided with projective coordinates $(X:Y:Z)$ such that $x = X/Z$ and $y = Y/Z$. We want to avoid inversions; to do so, we jump to isomorphic curves as needed (membership in the prime order subgroup is conserved by isomorphism). We thus maintain in the implementation an isomorphism factor $e \neq 0$, such that the running point (u, w) is in curve $\mathcal{C}(Ae^2, Be^4)$. Indeed, the isopmorphism is expressed as:

$$\begin{aligned} \phi_e : \mathcal{C}(A, B) &\longrightarrow \mathcal{C}(Ae^2, Be^4) \\ (u, w) &\longmapsto (ue^2, we) \end{aligned}$$

The isomorphism θ encounted in section 2 was really $\phi_{1/2}$, and its inverse is $\theta^{-1} = \phi_2$.

Algorithm 1 applies these formulas to test whether a point is in the subgroup of order ℓ.

Algorithm 1 Test membership in the prime-order subgroup

Require: $P = (X{:}Y{:}Z)$ in curve $\mathcal{E}(a, d)$ of order $h\ell$ (projective coordinates)
Ensure: TRUE if $\ell P = \mathbb{O}$, FALSE otherwise

1: **if** $hP = \mathbb{O}$ **then**
2: **return** TRUE if $P = \mathbb{O}$, FALSE otherwise
3: $(A, B) \leftarrow (2(a + d), (a - d)^2)$
4: $(A', B') \leftarrow (-2A, 16ad)$
5: $e \leftarrow X(Z - Y)$
6: $u \leftarrow (a - d)(Z + Y)(Z - Y)X^2$
7: $w \leftarrow 2Z(Z - Y)$
8: **for** $i = 1$ to $\log_2(h) - 1$ **do**
9: $(u'', w'') \leftarrow (4u, 2w)$
10: **if** $u'' \notin QR$ **then**
11: **return** FALSE
12: $w' = \sqrt{u''}$
13: $u' = (w'^2 - A'e^2 - w'w'')/2$
14: **if** $u' \notin QR$ **then**
15: $(u', w') \leftarrow (B'u', -w'u')$
16: $e \leftarrow eu'$
17: $w \leftarrow \sqrt{u'}$
18: $u \leftarrow (w^2 - Ae^2 - ww')/2$
19: **return** TRUE if $u \in QR$, FALSE otherwise

Low-order points. The initial comparison of hP with $\mathbb{O}$ is really a test on whether the source point P is a low-order point. We handle these points separately because $\mathbb{O}$ and N do not have well-defined (u, w) coordinates, and the formulas do not apply to them; conversely, if P is not a low-order point, then the successive halvings cannot yield any exceptional case for our formulas. Among low-order points, only $\mathbb{O}$ is in the subgroup of order ℓ. This step can be performed quite efficiently on Curve25519 and Curve448:

- On Curve25519, P is a low-order point if and only if $X = 0$, or $Y = 0$, or $X = \pm iY$ (for a given $i = \sqrt{-1}$ in the field).
- On Curve448, P is a low-order point if and only if $X = 0$ or $Y = 0$.

Failed square roots. It is possible to optimize away the Legendre symbol test in steps 10 and 14 by using a specific square root implementation that returns a predictable value on non-squares.

Suppose that the field modulus is $q = 5 \bmod 8$; this is the case for Curve25519. A candidate square root y of x can be computed using Atkin's formulas[2]:

$$b \leftarrow (2x)^{(q-5)/8}$$
$$c \leftarrow 2xb^2$$
$$y \leftarrow xb(c - 1)$$

The computed value c is equal to $(2x)^{(q-1)/4}$, i.e. a square root of the Legendre symbol of $2x$. If x is a non-zero square, then $2x \notin QR$ (since 2 is not a square modulo such a q) and we can

verify that $y^2 = x$. Otherwise, if $x \notin QR$, then $c = \pm 1$. We can detect that case by comparing c with 1 and -1 (or, equivalently, comparing c^2 with 1) and computing $y = 2xb$ instead. This leads to $y^2 = \pm 2x$. A further comparison of y^2 with $2x$, and conditional multiplication of y by a precomputed square root of -1 if $y^2 = -2x$, allows to reliably return $\sqrt{x}$ if $x \in QR$ or $\sqrt{2x}$ if $x \notin QR$. The process also returns whether x was a square, so that the caller knows exactly what value was obtained.

Using this extended square root, we can modify steps 14 to 17 into the following:

1. Compute w' as an extended square root of u'.
2. If u' was not a square, and we really computed $z = \sqrt{2u'}$, then continue the process as:

$$\begin{aligned}(u', w') &\leftarrow (2u'^2, w'z)\\ w &\leftarrow -e^2\sqrt{2B'}\\ e &\leftarrow ez\end{aligned}$$

3. In both cases, we compute $u = (w^2 - Ae^2 - ww')/2$.

The square root of $2B'$ is a precomputed constant.

If the field modulus is $q = 3 \bmod 4$ (as in the case of Curve448), a similar process can be applied. The candidate square root of a value x is $y = x^{(q+1)/4}$; if $x \notin QR$ then this returns a square root of $-x$ instead. Comparing y^2 with x reveals whether the source was a square or not. Steps 14 to 17 can then be replaced with:

1. Compute w' as an extended square root of u'.
2. If u' was not a square, and we really computed $z = \sqrt{-u'}$, then continue the process as:

$$\begin{aligned}(u', w') &\leftarrow (B'u', -w'u')\\ w &\leftarrow z\sqrt{-B'}\\ e &\leftarrow eu'\end{aligned}$$

3. In both cases, we compute $u = (w^2 - Ae^2 - ww')/2$.

The square root of $-B'$ is a precomputed constant.

Constant-time implementation. Algorithm 1 can be implemeted in a constant-time way (in case the source point to test is a secret value) by using constant-time tests and replacement operations where appropriate. In particular, when the source point is a low-order point, all halvings are still performed (possibly incorrectly), and the computed output must be discarded at the end, and replaced with a simple comparison with the neutral $\mathbb{O}$.

4 Implementation

We implemented the subgroup membership test for Curve25519 as part of the `crrl` library (written in Rust) available at:

`https://github.com/pornin/crrl`

Our implementation is fully constant-time.

On a 64-bit x86 system (Intel i5-8259U "Coffee Lake", clocked at 2.3 GHz, TurboBoost is disabled), we get the following performance:

- Generic multiplication of a point by a (secret) scalar: 107638 cycles.
- Multiplication by ℓ of a point: 84686 cycles.
- Subgroup membership test (algorithm 1): 44835 cycles.

Our new method is thus roughly twice faster than the classic method of multiplication by ℓ, and down to about 41% of the cost of the generic point multiplication. On Curve448, the cofactor is only 4 instead of 8, and algorithm 1 is expected to be faster (relatively) since it will involve only two square root operations instead of four. The subgroup membership test is still relatively expensive; if possible, use of a true prime-order group such as Ristretto[1] is highly recommended instead. The plain twisted Edwards curve should be used only when interoperability with some existing systems (e.g. EdDSA signature verifiers) is required.

Acknowledgements

We thank Giacomo Pope, who reviewed this paper.

References

1. T. Arcieri, I. Lovecruft and H. de Valence, *The Ristretto Group*, https://ristretto.group/
2. A. Atkin, *Probabilistic primality testing (summary by F. Morain)*, Technical Report 1779, INRIA, 1992, http://algo.inria.fr/seminars/sem91-92/atkin.pdf
3. D. Bernstein, N. Duif, T. Lange, P. Schwabe and B.-Y. Yang, *High-speed high-security signatures*, Journal of Cryptographic Engineering, vol. 2, issue 2, pp. 77-89, 2012.
4. D. Connolly, C. Komlo, I. Goldberg and C. Wood, *Two-Round Threshold Schnorr Signatures with FROST*, https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-frost-08
5. C. Cremers and D. Jackson, *Prime, Order Please! Revisiting Small Subgroup and Invalid Curve Attacks on Protocols using Diffie-Hellman*, IEEE 32nd Computer Security Foundations Symposium (CSF), 2019.
6. M. Hamburg, *Decaf: Eliminating cofactors through point compression*, Advances in Cryptology - CRYPTO 2015, Lecture Notes in Computer Science, vol. 9215, pp. 705-723, 2015.
7. M. Hamburg, *Ed448-Goldilocks, a new elliptic curve*, https://eprint.iacr.org/2015/625
8. C. Komlo and I. Goldberg, *FROST: Flexible Round-Optimized Schnorr Threshold Signatures*, https://eprint.iacr.org/2020/852
9. D. Moody and D. Shumow, *Analogues of Vélu's Formulas for Isogenies on Alternate Models of Elliptic Curves*, https://eprint.iacr.org/2011/430
10. T. Pornin, *Double-Odd Elliptic Curves*, https://eprint.iacr.org/2020/1558
11. J. Vélu, *Isogénies entre courbes elliptiques*, C.R. Acad. Sc. Paris, Série A, vol. 273, pp. 238-241, 1971.