Montgomery curves and their arithmetic

Craig Costello, Benjamin Smith

Montgomery curves and their arithmetic

The case of large characteristic fields

Craig Costello · Benjamin Smith

A survey in tribute to Peter L. Montgomery

Abstract Three decades ago, Montgomery introduced a new elliptic curve model for use in Lenstra's ECM factorization algorithm. Since then, his curves and the algorithms associated with them have become foundational in the implementation of elliptic curve cryptosystems. This article surveys the theory and cryptographic applications of Montgomery curves over non-binary finite fields, including Montgomery's x-only arithmetic and Ladder algorithm, x-only Diffie–Hellman, y-coordinate recovery, and 2-dimensional and Euclidean differential addition chains such as Montgomery's PRAC algorithm.

Keywords Montgomery curve · Montgomery ladder · elliptic curve cryptography · scalar multiplication

1 Introduction

Peter L. Montgomery's landmark 1987 paper *Speeding the Pollard and elliptic curve methods of factorization* [38] introduced what became known as *Montgomery curves* and the *Montgomery ladder* as a way of accelerating Lenstra's ECM factorization method [33]. However, they have gone on to have a far broader impact: while remaining a crucial component of modern factoring software, they have also become central to elliptic curve cryptography.

Consider the following situation: let q be a prime power and let $\mathcal{E}$ be an elliptic curve over $\mathbb{F}_q$, with group law $\oplus$, identity point O, and negation map $\ominus$. We have *scalar multiplication* endomorphisms on $\mathcal{E}$ defined by

$$[k] : P \longmapsto \underbrace{P \oplus \cdots \oplus P}_{k \text{ times}}$$

Craig Costello
Microsoft Research, USA
E-mail: craigco@microsoft.com

Benjamin Smith
INRIA *and* Laboratoire d'Informatique de l'École polytechnique (LIX), France
E-mail: smith@lix.polytechnique.fr

for every k in $\mathbb{Z}$ (with $[-k]P := [k](\ominus P)$). The negation map $\ominus$ is an automorphism of $\mathcal{E}$, and the quotient of $\mathcal{E}$ by $\langle\ominus\rangle$ is the 2-to-1 mapping

$$\mathbf{x} : \mathcal{E} \longrightarrow \mathbb{P}^1 \cong \mathcal{E}/\langle\ominus\rangle$$

such that $\mathbf{x}(P) = \mathbf{x}(Q)$ if and only if $P = Q$ or $P = \ominus Q$. If $\mathcal{E}$ is defined by a Weierstrass equation $y^2 = f(x)$, then $\mathbf{x}$ is just $P \mapsto x(P)$; in other models or coordinate systems, $\mathbf{x}$ may be more complicated. We call $\mathbb{P}^1$ the *x-line* of $\mathcal{E}$.

Now, $\mathbb{P}^1$ does not inherit any group structure from $\mathcal{E}$. But since $\ominus$ commutes with $[k]$ (i.e., $[k](\ominus P) = \ominus[k]P$), we still get an induced *pseudomultiplication*

$$\mathbf{x}(P) \longmapsto \mathbf{x}([k]P)$$

on $\mathbb{P}^1$ for every k in $\mathbb{Z}$.

In his seminal article proposing elliptic curves for use in cryptography, Miller [37] pointed out that elliptic curve Diffie–Hellman key exchange can be expressed entirely in terms of the maps $\mathbf{x}(P) \mapsto \mathbf{x}([k]P)$: given a public base point $\mathbf{x}(P)$ on $\mathbb{P}^1$, Alice (resp. Bob) can compute and publish $\mathbf{x}([a]P)$ (resp. $\mathbf{x}([b]P)$) for some secret a and b, and then derive the shared secret $\mathbf{x}([b][a]P) = \mathbf{x}([a][b]P)$ from $\mathbf{x}([b]P)$ (resp. $\mathbf{x}([a]P)$). We can always lift the resulting Diffie–Hellman problem to $\mathcal{E}$, so $\mathcal{E}$ provides not only the map $\mathbf{x}$, but also the security of the whole protocol; but by working in $\mathbb{P}^1(\mathbb{F}_q)$ instead of $\mathcal{E}(\mathbb{F}_q)$ we can save some space, and (hopefully) some time.

As a second application, consider ECM. We are given an integer N whose prime factorization is unknown. Choosing a random elliptic curve $\mathcal{E}$ over $\mathbb{Z}/N\mathbb{Z}$ and constructing a point P in $\mathcal{E}(\mathbb{Z}/N\mathbb{Z})$, we can compute $[B!]P$ for some moderate bound B; if there exists a prime factor p of N such that $\#\mathcal{E}(\mathbb{F}_p)$ is B-smooth, then we will have $[B!]P \equiv O \pmod{p}$; and we can detect this situation by taking the GCD of the projective Z-coordinate of $[B!]P$ (for a Weierstrass model of $\mathcal{E}$) with N. If the GCD is neither 1 nor N, then we have found a factor of N; otherwise, we can try again with another random $\mathcal{E}$. But all this still holds if we replace P with $\mathbf{x}(P)$ and $[B!]P$ with $\mathbf{x}([B!]P)$; and the advantages of compactness and simplicity that we can see in Diffie–Hellman on $\mathbb{P}^1$ are magnified in the ECM context, where the integer N can be much larger than any reasonable cryptographic q.

In both scenarios, our task is simple: we need to define a class of curves $\mathcal{E}$ equipped with efficient algorithms for computing $\mathbf{x}(P) \mapsto \mathbf{x}([k]P)$ on $\mathbb{P}^1$.

Montgomery's work provides a brilliant answer to this problem. The *Montgomery ladder* is a simple yet efficient algorithm for computing $\mathbf{x}(P) \mapsto \mathbf{x}([k]P)$. In the abstract, the ladder is a sequence of steps of pseudo-operations derived from the curve $\mathcal{E}$ (see §3.1); choosing a *Montgomery curve* for $\mathcal{E}$ ensures that each of those steps is optimized. The result, for Montgomery, was an extremely efficient implementation of ECM; and even today, three decades later, Montgomery's methods remain at the heart of state-of-the-art factoring software such as the widely-distributed `GMP-ECM` package [24, 49]. Later, these same qualities also led to many efficient implementations of elliptic curve cryptosystems, most notably Bernstein's Curve25519 software [3].

Notation. Throughout, we work over the finite field $\mathbb{F}_q$, where q is a power of an odd prime p (for most contemporary applications, $q = p$ or p^2). We write `M`, `S`, `a`, and `s` for the cost of a single multiplication, squaring, addition, and subtraction in $\mathbb{F}_q$, respectively. We will occasionally need to multiply by some constant elements of $\mathbb{F}_q$, which we hope to make as cheap as possible: we write `c` for the cost of a single such multiplication, to help keep track of this cost separately from the other multiplications with two variable inputs.

2 Montgomery curves

A *Montgomery curve* over $\mathbb{F}_q$ is an elliptic curve defined by an affine equation

$$\mathcal{E}_{(A,B)} : By^2 = x(x^2 + Ax + 1) \ ,$$

where A and B are parameters in $\mathbb{F}_q$ satisfying[1] $B \neq 0$ and $A^2 \neq 4$. Moving to projective plane coordinates $(X : Y : Z)$, with

$$x = X/Z \quad \text{and} \quad y = Y/Z \ ,$$

we have the projective model

$$\mathcal{E}_{(A,B)} : BY^2Z = X(X^2 + AXZ + Z^2) \subseteq \mathbb{P}^2 \ . \tag{1}$$

There is a unique point $O = (0 : 1 : 0)$ at infinity on $\mathcal{E}_{(A,B)}$: it is the only point on $\mathcal{E}_{(A,B)}$ where $Z = 0$.

2.1 The parameters A and B

A Montgomery curve $\mathcal{E}_{(A,B)}$ is specified by two parameters, A and B. The most important of the two is A, which controls the geometry of $\mathcal{E}_{(A,B)}$. Indeed, the j-invariant of $\mathcal{E}_{(A,B)}$ is

$$j(\mathcal{E}_{(A,B)}) = \frac{256(A^2 - 3)^3}{A^2 - 4} \ , \tag{2}$$

so the $\overline{\mathbb{F}}_q$-isomorphism class of $\mathcal{E}_{(A,B)}$ is completely determined by A^2, and independent of B. We see immediately that not every elliptic curve over $\mathbb{F}_q$ has a Montgomery model over $\mathbb{F}_q$, since while every element of $\mathbb{F}_q$ is the j-invariant of some curve over $\mathbb{F}_q$, not every element of $\mathbb{F}_q$ is in the form of the right-hand-side of (2) for some A in $\mathbb{F}_q$ (we return to the question of which curves have Montgomery models in §2.4 below).

The parameter B should be thought of as a "twisting factor": if B' is another nonzero element of $\mathbb{F}_q$, then $\mathcal{E}_{(A,B)} \cong \mathcal{E}_{(A,B')}$ via $(x, y) \mapsto (x, \sqrt{B/B'}y)$. This isomorphism is defined over $\mathbb{F}_q$ precisely when B/B' is a square in $\mathbb{F}_q$.

[1] These conditions imply nonsingularity: if $B = 0$ then $\mathcal{E}_{(A,B)}$ is a union of three lines, while if $A^2 = 4$ then $\mathcal{E}_{(A,B)}$ is a nodal cubic.

Otherwise, $\mathcal{E}_{(A,B')}$ is a *quadratic twist* of $\mathcal{E}_{(A,B)}$: isomorphic to $\mathcal{E}_{(A,B)}$ over $\mathbb{F}_{q^2}$, but not over $\mathbb{F}_q$.[2] All such quadratic twists $\mathcal{E}_{(A,B')}$ of $\mathcal{E}_{(A,B)}$ are isomorphic to each other over $\mathbb{F}_q$: if B'' is another element of $\mathbb{F}_q$ such that B/B'' is not a square, then B'/B'' must be a square, and then $\mathcal{E}_{(A,B')}$ and $\mathcal{E}_{(A,B'')}$ are isomorphic over $\mathbb{F}_q$ via $(x, y) \mapsto (x, \sqrt{B'/B''}y)$.

The value of B (modulo squares) is more or less incidental for cryptographic implementations. In the context of ECM, the ability to choose B gives us an important degree of freedom for constructing interesting points in $\mathcal{E}(\mathbb{Z}/N\mathbb{Z})$. But when the values of B and B' are mathematically and practically irrelevant (i.e., most of the time), we will simply call $\mathcal{E}_{(A,B')}$ *the* quadratic twist of $\mathcal{E}_{(A,B)}$. In general, we use $\mathcal{E}'$ to denote the quadratic twist of $\mathcal{E}$.

2.2 The group law

Since $\mathcal{E}_{(A,B)}$ is an elliptic curve, there is a group law $\oplus$ on its points; our first task is to describe it. For the moment, it suffices to work in affine coordinates; optimized projective formulæ will follow later in §3.

The point O at infinity acts as the zero element of the group structure; the negation map is $\ominus : (x : y : 1) \mapsto (x : -y : 1)$. For addition, if $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ are points on $\mathcal{E}_{(A,B)}$, then $P \oplus Q = (x_\oplus, y_\oplus)$ where

$$x_\oplus = B\lambda^2 - (x_P + x_Q) - A \quad \text{and} \quad \begin{aligned} y_\oplus &= (2x_P + x_Q + A)\lambda - B\lambda^3 - y_P \\ &= \lambda(x_P - x_Q) - y_P \ , \end{aligned}$$

with

$$\lambda = \begin{cases} (y_Q - y_P)/(x_Q - x_P) & \text{if } P \neq Q \text{ or } \ominus Q \ , \\ (3x_P^2 + 2Ax_P + 1)/(2By_P) & \text{if } P = Q \ ; \end{cases}$$

if $P = \ominus Q$, then $P \oplus Q = O$. We note that λ is the slope of the secant through P and Q (or the tangent to $\mathcal{E}_{(A,B)}$ at P, in the case $P = Q$).

As usual, we write $\mathcal{E}_{(A,B)}(\mathbb{F}_q)$ for the group of rational points of $\mathcal{E}_{(A,B)}$ (that is, projective solutions of (1)) with coordinates in $\mathbb{F}_q$. The m-torsion $\mathcal{E}[m]$ is the kernel of the scalar multiplication $[m]$. In general, its elements are defined over some extension of $\mathbb{F}_q$; we write $\mathcal{E}[m](\mathbb{F}_q)$ for the group of m-torsion elements whose coordinates are in $\mathbb{F}_q$.

2.3 Special torsion and group structures

What can we say about the group structure of $\mathcal{E}_{(A,B)}(\mathbb{F}_q)$?

[2] We saw above that since $j(\mathcal{E}_{(A,B)})$ is a function of A^2, the $\overline{\mathbb{F}}_q$-isomorphism class of $\mathcal{E}_{(A,B)}$ depends only on A^2. Indeed, $\mathcal{E}_{(-A,B)}$ is $\mathbb{F}_q$-isomorphic (via $(x, y) \mapsto (-x, y)$) to $\mathcal{E}_{(A,-B)}$, which is $\mathbb{F}_q$-isomorphic to $\mathcal{E}_{(A,B)}$ if -1 is a square in $\mathbb{F}_q$ (otherwise it is a quadratic twist).

Table 1 4-torsion structures on a Montgomery curve $\mathcal{E}_{(A,B)}$ and its quadratic twist $\mathcal{E}_{(A,B')}$.

B	$A-2$ and $A+2$	$\mathcal{E}_{(A,B)}[4](\mathbb{F}_q)$ contains	$\mathcal{E}_{(A,B')}[4](\mathbb{F}_q)$ contains
square	both square	$\mathbb{Z}/4\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$	$\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$
square	one square	$\mathbb{Z}/4\mathbb{Z}$	$\mathbb{Z}/4\mathbb{Z}$
square	neither square	$\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$	$\mathbb{Z}/4\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$
nonsquare	both square	$\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$	$\mathbb{Z}/4\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$
nonsquare	one square	$\mathbb{Z}/4\mathbb{Z}$	$\mathbb{Z}/4\mathbb{Z}$
nonsquare	neither square	$\mathbb{Z}/4\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$	$\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$

We note immediately that $\mathcal{E}_{(A,B)}$ always has a rational point of order two,

$$T := (0:0:1) \in \mathcal{E}_{(A,B)}[2](\mathbb{F}_q) \ ;$$

the translation map on $\mathcal{E}_{(A,B)}$ taking P to $P \oplus T$ is

$$\tau_T : (x, y) \longmapsto (1/x, -y/x^2) \ . \tag{3}$$

In projective coordinates, we can see one of the characteristic features of Montgomery curves more clearly. On any elliptic curve $\mathcal{E}$ with a point T of order 2, the translation-by-T map on $\mathcal{E}$ commutes with $\ominus$, so it induces an involution on the x-line $\mathbb{P}^1 = \mathcal{E}/\langle\ominus\rangle$, which has the form $(X : Z) \mapsto (aX + bZ : cX + dZ)$. But for $T = (0, 0)$ on a Montgomery curve $\mathcal{E}_{(A,B)}$, this involution is as simple as possible: here, τ_T induces $(X : Z) \mapsto (Z : X)$ on $\mathbb{P}^1$.

Montgomery notes in [38], following Suyama, that the order of $\mathcal{E}_{(A,B)}(\mathbb{F}_q)$ is always divisible by 4. The nonsingularity condition requires B, $A + 2$, and $A - 2$ to be nonzero. If we let $\mathcal{E}_{(A,B')}$ be the quadratic twist of $\mathcal{E}_{(A,B)}$, then the following facts are easy to check:

1. If $B(A+2)$ is a square in $\mathbb{F}_q$, then $(1, \pm\sqrt{(A+2)/B})$ are points of order 4 in $\mathcal{E}_{(A,B)}(\mathbb{F}_q)$ (and if $B(A+2)$ is not a square, then $(1, \pm\sqrt{(A+2)/B'})$ are points of order 4 in $\mathcal{E}_{(A,B')}(\mathbb{F}_q)$).[3]
2. Similarly, if $B(A-2)$ is a square in $\mathbb{F}_q$, then $(-1, \pm\sqrt{(A-2)/B})$ are points of order 4 in $\mathcal{E}_{(A,B)}(\mathbb{F}_q)$; otherwise, $(-1, \pm\sqrt{(A-2)/B'})$ are points of order 4 in $\mathcal{E}_{(A,B')}(\mathbb{F}_q)$.
3. If neither $B(A+2)$ nor $B(A-2)$ is a square, then A^2-4 must be a square; then $x^2 + Ax + 1$ splits completely over $\mathbb{F}_p$, so $\mathcal{E}_{(A,B)}$ has full rational 2-torsion. If α is one root of x^2+Ax+1, then the other root is $1/\alpha$; the points $(\alpha : 0 : 1)$ and $(1 : 0 : \alpha)$ have order 2, and are exchanged by τ_T. There are also points of order two on $\mathcal{E}_{(A,B')}$ with exactly the same coordinates.

The key thing is that in any finite field, $B(A+2)$, $B(A-2)$, and A^2-4 cannot all be nonsquares simultaneously: *at least one* of the above situations is forced to occur, and so $\#\mathcal{E}_{(A,B)}(\mathbb{F}_q)$ is always divisible by 4. Table 1 summarizes the resulting group structures for various combinations of the three conditions.

Going further, Suyama shows that if we take $A = -(3a^4+6a^2-1)/4a^3$ and $B = (a^2-1)^2/4ab^2$ for some a and b in $\mathbb{F}_q$ with $ab(a^2-1)(9a^2-1) \neq 0$, then

[3] Montgomery notes that in an ECM context we can take $B = A + 2$ in order to force $(1, 1)$ to be a rational point of order 4.

(a, b) is a point of order 3 in $\mathcal{E}_{(A,B)}(\mathbb{F}_q)$. This is useful in ECM, where we want to produce curves whose reduction modulo p have smooth order; reducing a curve from Suyama's parametrization yields a curve whose order is divisible by 12, and hence more probably smooth than a the order of a random curve.

For cryptographic constructions, we generally want $\#\mathcal{E}_{(A,B)}(\mathbb{F}_q)$ to be as near prime as possible. A Montgomery curve over a finite field can never have prime order: we always have $4 \mid \#\mathcal{E}_{(A,B)}(\mathbb{F}_q)$ (and $4 \mid \#\mathcal{E}_{(A,B')}(\mathbb{F}_q)$), so the best we can hope for is $\#\mathcal{E}_{(A,B)}(\mathbb{F}_q) = 4r$ with r prime. Reassuringly, there is no theoretical obstruction to the existence of $\mathcal{E}_{(A,B)}/\mathbb{F}_q$ such that $\#\mathcal{E}_{(A,B)}(\mathbb{F}_q) = 4r$ with r prime; and indeed, in practice we have no trouble finding A and B in $\mathbb{F}_q$ such that $\#\mathcal{E}_{(A,B)}(\mathbb{F}_q)/4$ is prime.

2.4 Correspondence with short Weierstrass models

Any Montgomery curve over $\mathbb{F}_q$ can be transformed into a short Weierstrass model over $\mathbb{F}_q$ (assuming q is not a power of 3): for example, the rational maps $(x, y) \mapsto (u, v) = (B(x + A/3), B^2 y)$ and $(u, v) \mapsto (x, y) = (u/B - A/3, v/B^2)$ define an isomorphism over $\mathbb{F}_q$ between $\mathcal{E}_{(A,B)}$ and the short Weierstrass model

$$\mathcal{E}^W : v^2 = u^3 + (B^2(1 - A^2/3))u + B^3 A/3(2A^2/9 - 1) \ .$$

The converse does not hold: not every short Weierstrass model can be transformed into a Montgomery model over $\mathbb{F}_q$. This is obvious enough: not every short Weierstrass model has a rational point of order 2 to map to T (and not every j-invariant in $\mathbb{F}_q$ can be expressed in the form of (2) with A in $\mathbb{F}_q$).

Still, it is useful to have a simple algebraic condition on the coefficients of a short Weierstrass model that encapsulate its transformability to Montgomery form. With this in mind, Okeya, Kurumatani, and Sakurai [41] observe that $\mathcal{E}^W : v^2 = u^3 + au + b$ has a Montgomery model if and only if there exist α and β in $\mathbb{F}_q$ such that $\alpha^3 + a\alpha + b = 0$ and $3\alpha^2 + a = \beta^2$: we then have an isomorphism $(u, v) \mapsto (x, y) = ((u - \alpha)/\beta, v/\beta)$ from $\mathcal{E}^W$ to the Montgomery curve $\mathcal{E}_{(3\alpha/\beta, 1/\beta)}$. The first relation ensures that there is a rational 2-torsion point $(\alpha, 0)$ in $\mathcal{E}^W(\mathbb{F}_q)$ (which is mapped to $T = (0, 0)$ by the isomorphism); the second ensures that translation by the image of that point acts as in (3).

2.5 Correspondence with twisted Edwards models

The last decade has seen the great success of Edwards models for elliptic curves in cryptographic implementations (see eg. [23], [7], and [5]). It turns out that every Montgomery curve over $\mathbb{F}_q$ is $\mathbb{F}_q$-isomorphic to a twisted Edwards model, and vice versa [5, §3]. The rational maps

$$(x, y) \longmapsto (u, v) = (x/y, (x - 1)/(x + 1)) \tag{4}$$
$$(u, v) \longmapsto (x, y) = ((1 + v)/(1 - v), (1 + v)/((1 - v)u) \tag{5}$$

define an isomorphism between $\mathcal{E}_{(A,B)}$ and the twisted Edwards model

$$\mathcal{E}^{\mathrm{Ed}}_{(a,d)} : au^2 + v^2 = 1 + du^2v^2 \qquad \text{where} \quad \begin{cases} a = (A+2)/B \ , \\ d = (A-2)/B \ . \end{cases} \tag{6}$$

The most natural projective closure for $\mathcal{E}^{\mathrm{Ed}}_{(a,d)}$ is not in $\mathbb{P}^2$, but in $\mathbb{P}^1 \times \mathbb{P}^1$, which embeds into $\mathbb{P}^3$ via the Segre morphism. Taking coordinates $(U_0 : U_1 : U_2 : U_3)$ on $\mathbb{P}^3$, with $u = U_1/U_0$, $v = U_2/U_0$, and $uv = U_3/U_0$, we obtain the projective model $\mathcal{E}^{\mathrm{Ed}}_{(a,d)} : U_0^2 + dU_3^2 = aU_1^2 + U_2^2, U_0U_3 = U_1U_2$ (the second equation defines the image of $\mathbb{P}^1 \times \mathbb{P}^1$ in $\mathbb{P}^3$); these are the *extended Edwards coordinates* of [29].

The point O on $\mathcal{E}_{(A,B)}$ maps to $(0,1) = (1:0:1:0)$ on $\mathcal{E}^{\mathrm{Ed}}_{(a,d)}$. The negation is $\ominus(u,v) = (-u,v)$, and the map $\mathbf{x} : \mathcal{E}^{\mathrm{Ed}}_{(a,d)} \to \mathbb{P}^1$ sends P to $v(P)$; if $\mathcal{E}^{\mathrm{Ed}}_{(a,d)}$ is viewed as a curve in $\mathbb{P}^1 \times \mathbb{P}^1$, then $\mathbf{x}$ is just projection onto the second factor. The distinguished 2-torsion point T maps to $(0,-1) = (1:0:-1:0)$, and the translation τ_T becomes $(U_0 : U_1 : U_2 : U_3) \mapsto (U_0 : U_1 : -U_2 : -U_3)$ on $\mathcal{E}^{\mathrm{Ed}}_{(a,d)}$.

We can also consider the images of the other torsion points described in §2.3. The 2-torsion points $(\frac{-1}{2}(A \pm \sqrt{A^2-4}), 0)$ on $\mathcal{E}_{(A,B)}$ map to $(0 : (A-2) : 0 : \pm\sqrt{A^2-4})$ on $\mathcal{E}^{\mathrm{Ed}}_{(a,d)}$, while the 4-torsion points $(1, \pm\sqrt{(A+\epsilon 2)/B})$ map to $(\pm\sqrt{(A+\epsilon 2)/B} : 1 : 0 : 0)$ for $\epsilon = \pm 1$.

3 Fast differential arithmetic in $\mathbb{P}^1$

In projective coordinates, the quotient map $\mathbf{x} : \mathcal{E}_{(A,B)} \to \mathcal{E}/\langle\ominus\rangle = \mathbb{P}^1$ is

$$\mathbf{x} : P \longmapsto \begin{cases} (x_P : 1) & \text{if } P = (x_P : y_P : 1) \ , \\ (1:0) & \text{if } P = O = (0:1:0) \ . \end{cases}$$

We emphasize that the formula $\mathbf{x}((X:Y:Z)) = (X:Z)$ only holds on the open subset of $\mathcal{E}_{(A,B)}$ where $Z \neq 0$; it does not extend to the point $O = (0 : 1 : 0)$ at infinity, because $(0:0)$ is not a projective point.

3.1 Pseudo-operations

Our first step towards computing $\mathbf{x}(P) \mapsto \mathbf{x}([k]P)$ is to define efficient pseudo-group operations on $\mathbb{P}^1$ derived from the group operation on $\mathcal{E}_{(A,B)}$. As we noted earlier, $\mathbb{P}^1$ inherits no group structure from $\mathcal{E}_{(A,B)}$: in particular, there is no map $(\mathbf{x}(P), \mathbf{x}(Q)) \mapsto \mathbf{x}(P \oplus Q)$. This is because $\mathbf{x}(P)$ determines P only up to sign, so while $(\mathbf{x}(P), \mathbf{x}(Q))$ mathematically determines the pair $\{\mathbf{x}(P \oplus Q), \mathbf{x}(P \ominus Q)\}$, we cannot tell which of the values is the correct "sum". However, any three of the values $\{\mathbf{x}(P), \mathbf{x}(Q), \mathbf{x}(P \oplus Q), \mathbf{x}(P \ominus Q))\}$ determines the fourth, so we *can* define a *pseudo-addition* on $\mathbb{P}^1$ by

$$\mathtt{xADD} : (\mathbf{x}(P), \mathbf{x}(Q), \mathbf{x}(P \ominus Q)) \longmapsto \mathbf{x}(P \oplus Q) \ .$$

The degenerate case where $P = Q$ becomes a *pseudo-doubling*

$$\texttt{xDBL} : \mathbf{x}(P) \longmapsto \mathbf{x}([2]P) \ .$$

These two operations will be the basis of our efficient pseudomultiplications.

Our first task is to compute `xADD` and `xDBL` efficiently. Let $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ be points on $\mathcal{E}_{(A,B)}$, with neither equal to T or O (so in particular, x_P and x_Q are both nonzero). Montgomery observed that if $P \neq Q$, then the x-coordinates of P, Q, $P \oplus Q$, and $P \ominus Q$ are related by

$$x_{P\oplus Q} x_{P\ominus Q} (x_P - x_Q)^2 = (x_P x_Q - 1)^2 \ , \tag{7}$$

while in the case $P = Q$, writing $[2]P = (x_{[2]P}, y_{[2]P})$, we have

$$4x_{[2]P} x_P (x_P^2 + Ax_P + 1) = (x_P^2 - 1)^2 \ . \tag{8}$$

Analogous identities exist for general Weierstrass models, but they are much simpler for Montgomery curves—as we will see in §3.4. It is this simplicity, due to the special form of τ_T discussed in §2.3, that leads to particularly efficient pseudo-operations for Montgomery curves.

3.2 Pseudo-addition

Our aim is to compute $\mathbf{x}(P \oplus Q)$ in terms of $\mathbf{x}(P)$, $\mathbf{x}(Q)$, and $\mathbf{x}(P \ominus Q)$; we suppose $P \neq Q$ and $P \ominus Q \neq T$. Following Montgomery [38, §10.3.1], we move to projective coordinates and write

$$\begin{aligned} (X_P : Z_P) &:= \mathbf{x}(P) \ , & (X_Q : Z_Q) &:= \mathbf{x}(Q) \ , \\ (X_\oplus : Z_\oplus) &:= \mathbf{x}(P \oplus Q) \ , & (X_\ominus : Z_\ominus) &:= \mathbf{x}(P \ominus Q) \ . \end{aligned}$$

Since $P \ominus Q \notin \{O, T\}$, we know that $X_\ominus \neq 0$ and $Z_\ominus \neq 0$. Equation (7) therefore becomes the pair of simultaneous relations

$$\begin{cases} X_\oplus = Z_\ominus \left[(X_P - Z_P)(X_Q + Z_Q) + (X_P + Z_P)(X_Q - Z_Q)\right]^2 \ , \\ Z_\oplus = X_\ominus \left[(X_P - Z_P)(X_Q + Z_Q) - (X_P + Z_P)(X_Q - Z_Q)\right]^2 \ , \end{cases} \tag{9}$$

which Algorithm 1 applies to efficiently compute $\mathbf{x}(P \oplus Q)$. If the "difference" $\mathbf{x}(P \ominus Q)$ is fixed then we can normalize it to $(X_P : 1)$, thus saving one multiplication in Step 11. Note that `xADD` involves neither of the curve parameters A or B, so it is identical for all Montgomery curves.

Algorithm 1: `xADD`: differential addition on $\mathbb{P}^1$

Input: (X_P, Z_P), (X_Q, Z_Q), and $(X_\ominus, Z_\ominus)$ in $\mathbb{F}_q^2$ such that $(X_P : Z_P) = \mathbf{x}(P)$, $(X_Q : Z_Q) = \mathbf{x}(Q)$, and $(X_\ominus : Z_\ominus) = \mathbf{x}(P \ominus Q)$ for P and Q in $\mathcal{E}(\mathbb{F}_q)$
Output: $(X_\oplus, Z_\oplus)$ in $\mathbb{F}_q^2$ such that $(X_\oplus : Z_\oplus) = \mathbf{x}(P \oplus Q)$ if $P \ominus Q \notin \{O, T\}$, otherwise $X_\oplus = Z_\oplus = 0$
Cost: $4\mathtt{M} + 2\mathtt{S} + 3\mathtt{a} + 3\mathtt{s}$, or $3\mathtt{M} + 2\mathtt{S} + 3\mathtt{a} + 3\mathtt{s}$ if $Z_\ominus$ is normalized to 1

1 $\mathsf{V}_0 \leftarrow X_P + Z_P$ // 1a
2 $\mathsf{V}_1 \leftarrow X_Q - Z_Q$ // 1s
3 $\mathsf{V}_1 \leftarrow \mathsf{V}_1 \cdot \mathsf{V}_0$ // 1M
4 $\mathsf{V}_0 \leftarrow X_P - Z_P$ // 1s
5 $\mathsf{V}_2 \leftarrow X_Q + Z_Q$ // 1a
6 $\mathsf{V}_2 \leftarrow \mathsf{V}_2 \cdot \mathsf{V}_0$ // 1M
7 $\mathsf{V}_3 \leftarrow \mathsf{V}_1 + \mathsf{V}_2$ // 1a
8 $\mathsf{V}_3 \leftarrow \mathsf{V}_3^2$ // 1S
9 $\mathsf{V}_4 \leftarrow \mathsf{V}_1 - \mathsf{V}_2$ // 1s
10 $\mathsf{V}_4 \leftarrow \mathsf{V}_4^2$ // 1S
11 $X_\oplus \leftarrow Z_\ominus \cdot \mathsf{V}_3$ // 1M / 0M if $Z_\ominus = 1$
12 $Z_\oplus \leftarrow X_\ominus \cdot \mathsf{V}_4$ // 1M
13 **return** $(X_\oplus : Z_\oplus)$

3.3 Pseudo-doubling

It remains to handle the doubling case, where $Q = P$. We want to compute $(X_{[2]P} : Z_{[2]P}) := \mathbf{x}([2]P)$ in terms of $\mathbf{x}(P)$. Again, we follow [38, §10.3.1]: in projective coordinates, Equation (8) becomes the pair of simultaneous relations

$$\begin{cases} X_{[2]P} = (X_P + Z_P)^2 (X_P - Z_P)^2 \ , \\ Z_{[2]P} = (4X_P Z_P)((X_P - Z_P)^2 + ((A+2)/4)(4X_P Z_P)) \ . \end{cases} \tag{10}$$

Algorithm 2 uses these, and the identity $4X_P Z_P = (X_P + Z_P)^2 - (X_P - Z_P)^2$, to efficiently compute $(X_{[2]P} : Z_{[2]P})$. The fact that the right-hand-sides of (10) are symmetric under $(X_P : Z_P) \leftrightarrow (Z_P : X_P)$ reflects the fact that the doubling map [2] factors through the 2-isogeny $\mathcal{E}_{(A,B)} \to \mathcal{E}_{(A,B)}/\langle T \rangle$; this aspect of Montgomery's x-line arithmetic later found an echo in Doche, Icart, and Kohel's work on efficient doubling and tripling [19].

Algorithm 2: `xDBL`: pseudo-doubling on $\mathbb{P}^1$ from $\mathcal{E}_{(A,B)}$

Input: (X_P, Z_P) in $\mathbb{F}_q^2$ such that $(X_P : Z_P) = \mathbf{x}(P)$ for P in $\mathcal{E}(\mathbb{F}_q)$
Output: $(X_{[2]P}, Z_{[2]P})$ in $\mathbb{F}_q^2$ such that $(X_{[2]P} : Z_{[2]P}) = \mathbf{x}([2]P)$ if $P \notin \{O, T\}$, otherwise $Z_{[2]P} = 0$
Cost: $2\mathtt{M} + 2\mathtt{S} + 1\mathtt{c} + 3\mathtt{a} + 1\mathtt{s}$

1 $\mathsf{V}_1 \leftarrow X_P + Z_P$ // 1a
2 $\mathsf{V}_1 \leftarrow \mathsf{V}_1^2$ // 1S
3 $\mathsf{V}_2 \leftarrow X_P - Z_P$ // 1s
4 $\mathsf{V}_2 \leftarrow \mathsf{V}_2^2$ // 1S
5 $X_{[2]P} \leftarrow \mathsf{V}_1 \cdot \mathsf{V}_2$ // 1M
6 $\mathsf{V}_1 \leftarrow \mathsf{V}_1 - \mathsf{V}_2$ // 1s
7 $\mathsf{V}_3 \leftarrow ((A+2)/4) \cdot \mathsf{V}_1$ // 1c
8 $\mathsf{V}_3 \leftarrow \mathsf{V}_3 + \mathsf{V}_2$ // 1a
9 $Z_{[2]P} \leftarrow \mathsf{V}_1 \cdot \mathsf{V}_3$ // 1M
10 **return** $(X_{[2]P} : Z_{[2]P})$

Step 7 of Algorithm 2 is a multiplication by the constant $(A+2)/4$. We emphasize that this is the *only* place in the x-line arithmetic where the parameter A appears. The parameter B never appears at all: x-line arithmetic is twist-agnostic. For implementations, therefore, we try to choose A such that the cost of multiplying by $(A+2)/4$ is minimised (for example, taking A such that $(A+2)/4$ is particularly small).

3.4 Comparison with general x-line arithmetic

To see how Montgomery's choice of curve model contributes to efficient x-line arithmetic, it is instructive to compare the pseudo-addition with the equivalent formulæ for a general Weierstrass model

$$\mathcal{E} : y^2 = x^3 + f_2x^2 + f_1x + f_0 \ .$$

The analogues of (7) and (8) for $\mathcal{E}$ are $x_\oplus x_\ominus (x_P - x_Q)^2 = (x_Px_Q - f_1)^2 - 4f_0(x_P{+}x_Q{+}f_2)$ and $4x_{[2]P}(x_P^3{+}f_2x_P^2{+}f_1x_P{+}f_0) = (x_P^2{-}f_1)^2{-}4f_0(2x_P{+}f_2)$, respectively. In projective coordinates, these become the relatively complicated pseudo-addition formulæ

$$\begin{cases} X_\oplus = Z_\ominus\big[(X_PX_Q - f_1Z_PZ_Q)^2 - 4f_0(Z_PX_Q + X_PZ_Q + f_2Z_PZ_Q)Z_PZ_Q\big] \\ Z_\oplus = X_\ominus(Z_PX_Q - X_PZ_Q)^2 \end{cases}$$

and pseudo-doubling formulæ

$$\begin{cases} X_{[2]P} = (X_P^2 - f_1Z_P^2)^2 - 4f_0(2X_PZ_P + f_2Z_P^2)Z_P^2 \ , \\ Z_{[2]P} = 4(X_P^4 + f_2X_P^2Z_P^2 + f_1X_PZ_P^3 + f_0Z_P^4) \ . \end{cases}$$

If we specialize and take $f_2 = 0$, leaving f_1 and f_0 free, then we are in the case of short Weierstrass models (for which the affine formulæ are classical: see eg. [12, Formulary]). This imposes no special structure on $\mathcal{E}$, since every elliptic curve is isomorphic to a short Weierstrass model over $\mathbb{F}_q$. The resulting projective formulæ were proposed for side-channel-aware implementations by Brier and Joye [10, §4], with pseudo-addition requiring $7\mathtt{M} + 2\mathtt{S} + 2\mathtt{c} + 3\mathtt{a} + 2\mathtt{s}$ and pseudo-doubling $3\mathtt{M} + 4\mathtt{S} + 2\mathtt{c} + 6\mathtt{a} + 2\mathtt{s}$.

But we can simplify things more dramatically by taking $f_0 = 0$ instead, leaving f_1 and f_2 free. This is equivalent to requiring a rational 2-torsion point on $\mathcal{E}$, which we move to $(0, 0)$. The pseudo-addition formulæ become

$$(X_\oplus : Z_\oplus) = \big(Z_\ominus(X_PX_Q - f_1Z_PZ_Q)^2 : X_\ominus(Z_PX_Q - X_PZ_Q)^2\big) \ ,$$

which can be evaluated in $6\mathtt{M} + 2\mathtt{S} + 1\mathtt{c} + 2\mathtt{s}$, while pseudo-doubling becomes

$$(X_{[2]P} : Z_{[2]P}) = \big((X_P^2 - f_1Z_P^2)^2 : 4(X_P^4 + f_2X_P^2Z_P^2 + f_1X_PZ_P^3)\big) \ ,$$

which can be evaluated in $2\mathtt{M} + 4\mathtt{S} + 2\mathtt{c} + 4\mathtt{a} + 1\mathtt{s}$.

Now taking $f_1 = 1$ not only eliminates $1\mathtt{c}$ in the pseudo-addition and pseudo-doubling, it also allows us to save $2\mathtt{M}$ in the pseudo-addition and $2\mathtt{S}$ in the pseudo-doubling (at the cost of a few more additions and subtractions) by exploiting the symmetry of the resulting forms[4]. Indeed, if we write A for f_2 and allow a possible quadratic twist by B, then we have arrived at Montgomery's model $\mathcal{E} : By^2 = x(x^2 + Ax + 1)$, and we recover the efficient pseudo-addition and pseudo-doubling in (9) and (10).

[4] Translation by the 2-torsion point $(0, 0)$ is defined by $(x, y) \mapsto (f_1/x, -f_1y/x^2)$; taking $f_1 = 1$ is therefore equivalent to putting this translation map in the special form of (3).

4 The Montgomery ladder

We now resume our task of computing $\mathbf{x}(P) \mapsto \mathbf{x}([k]P)$. We begin by explaining a version of the Montgomery ladder that uses full group operations to compute $P \mapsto [k]P$ on $\mathcal{E}_{(A,B)}$, before deriving the classic x-line Montgomery ladder which computes $\mathbf{x}(P) \mapsto \mathbf{x}([k]P)$ using only `xADD` and `xDBL`. We then present Okeya and Sakurai's version of the López–Dahab trick, an appendix to the x-only ladder which recovers the full image point $[k]P$ on $\mathcal{E}_{(A,B)}$.

4.1 The ladder in a group

Algorithm 3 presents the Montgomery ladder algorithm in the context of a group, for ease of analysis. (While the algorithm is presented using a Montgomery curve $\mathcal{E}_{(A,B)}$, it is clear that it works in any abelian group).

Algorithm 3: Montgomery's binary algorithm in the group $\mathcal{E}_{(A,B)}(\mathbb{F}_q)$

Input: $k = \sum_{i=0}^{\ell-1} k_i 2^i$ with $k_{\ell-1} = 1$, and $P \in \mathcal{E}(\mathbb{F}_q)$
Output: $[k]P$
Cost: $\ell - 1$ calls to $\oplus$ and ℓ calls to [2]

```
1 (R0, R1) ← (P, [2]P)
2 for i = ℓ − 2 down to 0 do
3 |   if k_i = 0 then
4 |   |   (R0, R1) ← ([2]R0, R0 ⊕ R1)
5 |   else
6 |   |   (R0, R1) ← (R0 ⊕ R1, [2]R1)
7 return R0
```

Algorithm 3 maintains two important invariants. First, looking at Lines 4 and 6, we see that the difference $R_1 \ominus R_0$ never changes; then, looking at Line 1, we see that that difference must always be P; hence, at all times,

$$R_1 = R_0 \oplus P \ . \tag{11}$$

Second, after iteration i of the loop (counting downwards from $\ell - 2$), we have

$$R_0 = [(k)_i]P \quad \text{and} \quad R_1 = [(k)_i + 1]P \ , \quad \text{where} \quad (k)_i := \lfloor k/2^i \rfloor \ . \tag{12}$$

This proves the correctness of Algorithm 3: at Line 7 we have just finished iteration $i = 0$, so we return $R_0 = [(k)_0]P = [k]P$. Equation (12) is easy to see once we know (11): looking at R_0 in each iteration, we recover the classic double-and-add method for computing $[k]P$. First, R_0 is initialized to P; then, if $k_i = 0$ we double R_0, while if $k_i = 1$ we replace R_0 with $R_0 \oplus R_1$, which is $[2]R_0 \oplus P$ by (11)—that is, we double R_0 and add P.

4.2 The Montgomery ladder

Equations (11) and (12) allow us to transform Algorithm 3 into Algorithm 4, known as the Montgomery ladder. We want to compute $\mathbf{x}(P) \mapsto \mathbf{x}([k]P)$. We may suppose $P \neq O$, since $\mathbf{x}([k]O) = \mathbf{x}(O)$ for all k, and we may also suppose $P \neq T$, since $\mathbf{x}([k]T) = \mathbf{x}(T)$ for odd k and $\mathbf{x}(O)$ for even k.

Maintaining the notation of (12), let $(\mathsf{x}_0, \mathsf{x}_1) = (\mathbf{x}([(k)_i]P), \mathbf{x}([(k)_i + 1]P))$. Then (11) shows that we can compute $(\mathbf{x}([(k)_{i-1}]P), \mathbf{x}([(k)_{i-1} + 1]P))$ as $(\mathtt{xDBL}(\mathsf{x}_0), \mathtt{xADD}(\mathsf{x}_0, \mathsf{x}_1, \mathbf{x}(P)))$ or $(\mathtt{xADD}(\mathsf{x}_0, \mathsf{x}_1, \mathbf{x}(P)), \mathtt{xDBL}(\mathsf{x}_1))$, depending on the value of the bit k_i.[5] We can therefore initialize $(\mathsf{x}_0, \mathsf{x}_1)$ to $(\mathbf{x}(P), \mathbf{x}([2]P)) = (\mathbf{x}(P), \mathtt{xDBL}(\mathbf{x}(P)))$, and then applying the transitions above for each bit of k will yield $(\mathsf{x}_0, \mathsf{x}_1) = (\mathbf{x}([k]P), \mathbf{x}([k+1]P))$.

While the final value of x_0 is $\mathbf{x}([k]P)$, the target of our calculation, we will see in §4.3 that the final value $\mathbf{x}([k+1]P)$ of x_1 can be used to help recover the full group element $[k]P$, if desired. We therefore include x_1 as an optional second return value in Algorithm 4.

Algorithm 4: `LADDER`: The Montgomery ladder

Input: $k = \sum_{i=0}^{\ell-1} k_i 2^i$ with $k_{\ell-1} = 1$, and (X_P, Z_P) in $\mathbb{F}_q^2$ s.t. $(X_P : Z_P) = \mathbf{x}(P)$
Output: $(X_k, Z_k) \in \mathbb{F}_q^2$ s.t. $(X_k : Z_k) = \mathbf{x}([k]P)$ if $P \notin \{O, T\}$, otherwise $Z_k = 0$.
Also optionally returns $(X_{k+1}, Z_{k+1}) \in \mathbb{F}_q^2$ s.t. $(X_{k+1} : Z_{k+1}) = \mathbf{x}([k]P)$ if $P \notin \{O, T\}$, otherwise $Z_{k+1} = 0$.
Cost: $\ell - 1$ calls to `xADD` and ℓ calls to `xDBL`

```
(x0, x1) ← ((X_P, Z_P), xDBL((X_P, Z_P)))
for i = ℓ − 2 down to 0 do
    if k_i = 0 then
        (x0, x1) ← (xDBL(x0), xADD(x0, x1, (X_P, Z_P)))
    else
        (x0, x1) ← (xADD(x0, x1, (X_P, Z_P)), xDBL(x1))
return x0 (and optionally x1)
```

4.3 Recovery of y-coordinates

On the surface, the Montgomery ladder appears to be an algorithm for computing $\mathbf{x}([k]P)$ from $\mathbf{x}(P)$. However, López and Dahab [35] observed that since it actually computes $\mathbf{x}([k]P)$ and $\mathbf{x}([k+1]P)$, the ladder can easily be extended to compute the full scalar multiplication $P \mapsto [k]P$ by first computing $\mathbf{x}(P) \mapsto (\mathbf{x}([k]P), \mathbf{x}([k+1]P))$ and then recovering $[k]P$ from the data $(P, \mathbf{x}([k]P), \mathbf{x}([k+1]P))$.

[5] Since the `xADD` and `xDBL` calls always share an argument, it is common for high-performance implementations to exploit any overlap between intermediate calculations in the `xADD` and `xDBL` by merging them in one combined function.

López and Dahab originally gave formulæ for this recovery step specific to Montgomery curves over binary fields. Their results were extended to prime-field Montgomery curves by Okeya and Sakurai [42], and later to short Weierstrass models by Brier and Joye [10]. Kohel provides a more general and powerful point of view in [31], treating the image of the curve under the map $Q \mapsto (\mathbf{x}(Q), \mathbf{x}(Q \oplus P))$ as a new model of the elliptic curve itself.

Okeya and Sakurai proceed as follows. Suppose P is not in $\mathcal{E}_{(A,B)}[2]$, and Q is not in $\{P, \ominus P, O\}$.[6] In affine coordinates, writing $(x_P, y_P) = P$, $(x_Q, y_Q) = Q$, and $(x_\oplus, y_\oplus) = P \oplus Q$ as usual, the group law formulæ in §2.2 show that y_Q can be deduced from x_P, y_P, x_Q, and $x_\oplus$ using the relation

$$y_Q = \frac{(x_P x_Q + 1)(x_P + x_Q + 2A) - 2A - (x_P - x_Q)^2 x_\oplus}{2B y_P}$$

(note the re-appearance of the twisting parameter B).

Algorithm 5, taken from [42, Algorithm 1], applies this to compute Q from P, $\mathbf{x}(Q)$, and $\mathbf{x}(P \oplus Q)$. Lines 6 and 15 involve multiplications by the constants $2A$ and $2B$. Referring back to §3.3, if $(A+2)/4$ is chosen to be advantageously small, then $2A = 8((A+2)/4) - 4$ is also small. Similarly, referring back to §2.1, we can choose B such that multiplication by $2B$ is essentially free.

Algorithm 5: `Recover`: Okeya–Sakurai y-coordinate recovery

Input: $(x_P : y_P : 1) = P$, $(X_Q : Z_Q) = \mathbf{x}(Q)$, and $(X_\oplus : Z_\oplus) = \mathbf{x}(P \oplus Q)$ for P and Q in $\mathcal{E}_{(A,B)}(\mathbb{F}_q)$ with $P \notin \mathcal{E}_{(A,B)}[2]$ and $Q \notin \{P, \ominus P, O\}$.
Output: $(X' : Y' : Z') = Q$
Cost: $10\mathtt{M} + 1\mathtt{S} + 2\mathtt{c} + 3\mathtt{a} + 3\mathtt{s}$

1 $\mathsf{v}_1 \leftarrow x_P \cdot Z_Q$ // 1M
2 $\mathsf{v}_2 \leftarrow X_Q + \mathsf{v}_1$ // 1a
3 $\mathsf{v}_3 \leftarrow X_Q - \mathsf{v}_1$ // 1s
4 $\mathsf{v}_3 \leftarrow \mathsf{v}_3^2$ // 1S
5 $\mathsf{v}_3 \leftarrow \mathsf{v}_3 \cdot X_\oplus$ // 1M
6 $\mathsf{v}_1 \leftarrow 2A \cdot Z_Q$ // 1c
7 $\mathsf{v}_2 \leftarrow \mathsf{v}_2 + \mathsf{v}_1$ // 1a
8 $\mathsf{v}_4 \leftarrow x_P \cdot X_Q$ // 1M
9 $\mathsf{v}_4 \leftarrow \mathsf{v}_4 + Z_Q$ // 1a
10 $\mathsf{v}_2 \leftarrow \mathsf{v}_2 \cdot \mathsf{v}_4$ // 1M
11 $\mathsf{v}_1 \leftarrow \mathsf{v}_1 \cdot Z_Q$ // 1M
12 $\mathsf{v}_2 \leftarrow \mathsf{v}_2 - \mathsf{v}_1$ // 1s
13 $\mathsf{v}_2 \leftarrow \mathsf{v}_2 \cdot Z_\oplus$ // 1M
14 $Y' \leftarrow \mathsf{v}_2 - \mathsf{v}_3$ // 1s
15 $\mathsf{v}_1 \leftarrow 2B \cdot y_P$ // 1c
16 $\mathsf{v}_1 \leftarrow \mathsf{v}_1 \cdot Z_Q$ // 1M
17 $\mathsf{v}_1 \leftarrow \mathsf{v}_1 \cdot Z_\oplus$ // 1M
18 $X' \leftarrow \mathsf{v}_1 \cdot X_Q$ // 1M
19 $Z' \leftarrow \mathsf{v}_1 \cdot Z_Q$ // 1M
20 **return** $(X' : Y' : Z')$

Combining Algorithms 4 and 5 yields Algorithm 6, an efficient scalar multiplication routine for the full group $\mathcal{E}_{(A,B)}(\mathbb{F}_q)$ using x-only arithmetic. This is generally much more efficient than Algorithm 3.

5 Montgomery curves and ladders in elliptic curve cryptography

We saw in §2.3 that no Montgomery curve can have prime order, since their order is always divisible by 4. The presence of this small cofactor 4 has no serious impact on the security level of a well-chosen Montgomery curve, since the state of the art for solving discrete logarithms in large prime-order subgroups of Montgomery curves is still Pollard's rho method [43].

[6] This is not a serious restriction in the context of scalar multiplication, where $Q = [k]P$: if P is a point of order 2, then either $[k]P = O$ or $y([k]P) = 0$.

Algorithm 6: Scalar multiplication on $\mathcal{E}_{(A,B)}(\mathbb{F}_q)$, combining the Montgomery ladder with y-coordinate recovery

Input: $k \in \mathbb{Z}_{>0}$, and P in $\mathcal{E}_{(A,B)}(\mathbb{F}_q) \setminus \mathcal{E}_{(A,B)}[2](\mathbb{F}_q)$
Output: $[k]P$
1 $(\mathsf{x}_0, \mathsf{x}_1) \leftarrow \texttt{LADDER}(k, (X_P, Z_P))$ where $(X_P : Z_P) = \mathbf{x}(P)$
2 $Q \leftarrow \texttt{Recover}(P, \mathsf{x}_0, \mathsf{x}_1)$
3 **return** Q

5.1 Montgomery curves in cryptographic standards

None of the elliptic curves so far standardardized by NIST [40], Brainpool [22], or ANSSI [1] can be transformed into Montgomery form over the base field, because they all have prime order, so they are unfortunately incompatible with Montgomery arithmetic. However, the two curves recently proposed by the Internet Research Task Force (IRTF) for standardization in the Transport Layer Security (TLS) protocol are specified in Montgomery form [32].

Example 1 (Curve25519 [3]) The most widely-known Montgomery curve in contemporary cryptography is the curve used in Bernstein's Curve25519 software for Diffie–Hellman key exchange. This curve is defined by

$$\mathcal{E}/\mathbb{F}_p : y^2 = x(x^2 + 486662x + 1) \quad \text{where} \quad p = 2^{255} - 19 \ .$$

We find $\#\mathcal{E}(\mathbb{F}_p) = 8r$ and $\#\mathcal{E}'(\mathbb{F}_p) = 4r'$, where r and r' are 253-bit primes.

Example 2 (Curve448 [28]) Hamburg's Curve448 offers a conservative, high-strength alternative to Curve25519 for TLS. This curve is defined by

$$\mathcal{E}/\mathbb{F}_p : y^2 = x(x^2 + 156326x + 1) \quad \text{where} \quad p = 2^{448} - 2^{224} - 1 \ .$$

We find $\#\mathcal{E}(\mathbb{F}_p) = 4r$ and $\#\mathcal{E}'(\mathbb{F}_p) = 4r'$, where r and r' are 446- and 447-bit primes, respectively.

5.2 Diffie–Hellman with x-coordinates

Recall Miller's x-only Diffie–Hellman protocol, described in §1: Alice computes $(a, \mathbf{x}(P)) \mapsto \mathbf{x}([a]P)$ and transmits her public key $\mathbf{x}([a]P)$ to Bob. Upon receiving Bob's public key $\mathbf{x}([b]P)$, she computes $(a, \mathbf{x}([b]P)) \mapsto \mathbf{x}([ab]P)$ to arrive at the same shared secret as Bob, who computes $(b, \mathbf{x}([a]P)) \mapsto \mathbf{x}([ba]P)$. This x-only protocol is the basis of Bernstein's Curve25519 key exchange software. But in addition to working entirely with x-coordinates, Bernstein observed that the Montgomery form allows for another simplification in real-world implementations. Recall from §2.1 that if $\mathcal{E}_{(A,B')}(\mathbb{F}_q)$ is the quadratic twist of $\mathcal{E}_{(A,B)}(\mathbb{F}_q)$, then precisely one of B and B' is a square in $\mathbb{F}_q$, and the other is a non-square. It follows that every element x_P in $\mathbb{F}_q$ corresponds to a point $P = (x_P, y_P)$ which is either in $\mathcal{E}_{(A,B)}(\mathbb{F}_q)$, in $\mathcal{E}_{(A,B')}(\mathbb{F}_q)$, or (if $y_P = 0$)

in both $\mathcal{E}_{(A,B)}[2](\mathbb{F}_q)$ and $\mathcal{E}_{(A,B')}[2](\mathbb{F}_q)$. Bernstein chose the curve in Example 1 by insisting that both $\mathcal{E}_{(A,B)}(\mathbb{F}_q)$ and $\mathcal{E}_{(A,B')}(\mathbb{F}_q)$ have cryptographically strong (i.e., almost prime) group orders. This means that every element of $\mathbb{F}_q$ corresponds to a point on a cryptographically strong Montgomery curve, and implementers need not perform any *point validation* checks [8] in this protocol. Moreover, since the Montgomery ladder does not use the constant B, it correctly computes $(\mathbf{x}(P), k) \mapsto \mathbf{x}([k]P)$ irrespective of the twist that P lies on. If both $\mathcal{E}_{(A,B)}$ and $\mathcal{E}_{(A,B')}$ are secure, then $\mathcal{E}_{(A,B)}$ is said to be *twist-secure*.

As we noted above, when implementing cryptosystems based on the discrete logarithm or Diffie–Hellman problems, we generally want $\#\mathcal{E}_{(A,B)}(\mathbb{F}_q)$ as close to prime as possible. We have $\#\mathcal{E}_{(A,B)}(\mathbb{F}_q) + \#\mathcal{E}_{(A,B')}(\mathbb{F}_q) = 2q + 2$, as with any elliptic curve-twist pair over $\mathbb{F}_q$; so if $\mathcal{E}_{(A,B)}(\mathbb{F}_q) = 4r$ with r prime, then the closest we can come to prime order for the twist is $\#\mathcal{E}_{(A,B')}(\mathbb{F}_q) = 4r'$ with r' prime if $q \equiv 3 \pmod 4$, and $\#\mathcal{E}_{(A,B')}(\mathbb{F}_q) = 8r'$ with r' prime if $q \equiv 1 \pmod 4$. We do not know of any theoretical asymptotic results guaranteeing a density or distribution of twist-secure Montgomery curves over any finite field, but there does not seem to be any problem in finding such curves in practice.[7]

The presence of non-trivial cofactors means that care must be taken in certain scenarios to thwart the threat of small subgroup attacks [34]. For x-line Diffie–Hellman, the easiest way to do this is to define all secret scalars to be a multiple of the lowest common multiple of the curve and twist cofactors.

5.3 Constant-time ladders

For secure software implementations of ECC, it is important that scalar multiplication routines exhibit uniform execution patterns with no correlation between timing and secret data; such *constant-time* behaviour is an essential first step towards preventing timing attacks [30]. Unlike many other addition chains and scalar multiplication algorithms, the Montgomery ladder has an inherently uniform execution pattern. Nevertheless, a number of issues must still be addressed in order to achieve constant-time implementations.

As it stands, the length of the main loop in Algorithm 4 is determined by the bitlength k of the input scalar $m \in [0, r)$. There are two common strategies for making the loop length independent of k. One option is to require all scalars to have their top bit set, either by defining them that way (as was done in [3]) or by adding a small, fixed multiple of the (sub)group order to each scalar. A second option is to modify Step 1 of Algorithm 4, setting $(\mathsf{x}_0, \mathsf{x}_1) \leftarrow (\mathbf{x}(O), \mathbf{x}(P))$ instead of $(\mathbf{x}(P), \mathbf{x}([2]P))$. Since the formulæ for `xDBL` and `xADD` behave correctly under these inputs, x_0 and x_1 will remain unchanged until the first non-zero bit of the scalar k is encountered, so a constant-length loop can be achieved by accepting scalars as all bitstrings of a fixed length.

Algorithm 4 presents the ladder using an **if** statement. Since each **if** represents branching on potentially secret data, and these branches may be mea-

[7] An analysis of the frequency of prime-order curves with prime-order twists appears in [45]; but this does not apply to Montgomery curves, since they cannot have prime order.

sured in timing variations, it is standard practice to replace the branches with conditional swaps (such as the `SWAP` defined in Algorithm 7) to avoid leaking information on secret scalars. The result is Algorithm 8, which consists of a uniform sequence of `xDBL`s and `xADD`s. Provided the field arithmetic used by the `xADD` and `xDBL` specified in Algorithms 1 and 2 is implemented in a completely uniform way, this yields a completely uniform algorithm.

Algorithm 7: `SWAP`: Constant-time conditional swap.

Input: $b \in \{0,1\}$ and a pair $(\mathsf{x}_0, \mathsf{x}_1)$ of objects encoded as n-bit strings
Output: $(\mathsf{x}_b, \mathsf{x}_{1-b})$

```
b ← (b, ..., b)_n
v ← b and (x_0 xor x_1)          // bitwise and, xor; do not short-circuit and
return (x_0 xor v, x_1 xor v)
```

Algorithm 8: A uniform Montgomery ladder

Input: $k = \sum_{i=0}^{\ell-1} k_i 2^i$ with $k_{\ell-1} = 1$, and $\mathbf{x}(P)$ for P in $\mathcal{E}_{(A,B)}(\mathbb{F}_q)$
Output: $(X_k, Z_k) \in \mathbb{F}_q^2$ s.t. $(X_k : Z_k) = \mathbf{x}([k]P)$ if $P \notin \{O, T\}$, otherwise $Z_k = 0$.
Cost: $\ell - 1$ calls to `xADD`, ℓ calls to `xDBL`, and $\ell - 1$ calls to `SWAP`

```
(x_0, x_1) ← (xDBL((X_P, Z_P)), (X_P, Z_P))
for i = ℓ − 2 down to 0 do
    (x_0, x_1) ← SWAP((k_{i+1} xor k_i), (x_0, x_1))
    (x_0, x_1) ← (xDBL(x_0), xADD(x_0, x_1, (X_P, Z_P)))
(x_0, x_1) ← SWAP(k_0, (x_0, x_1))
return x_0
```

5.4 Completeness and Bernstein's modified **x**-map

Algorithms 4 and 8 do not compute the pseudomultiplication $\mathbf{x}([k]P)$ correctly if $P = O$ or T: instead, they return $(X_k, Z_k) = (e, 0)$ for some e in $\mathbb{F}_q$ (see [2, Theorem 4.3] for a more precise statement). In some cases e may be 0, in which case $(X_k : Z_k) = (0 : 0)$ is not even a projective point; but even if $e \neq 0$, the resulting $(X_k : Z_k) = (1 : 0)$ may not be equal to $\mathbf{x}([k]P)$.

In Montgomery's original context of ECM, this is a feature, not a bug: the ultimate goal there is to produce (X_k, Z_k) such that $\gcd(Z_k, N) > 1$, regardless of whether or not $(X_k : Z_k)$ is a correct pseudomultiplication result, or even a legal projective point. But it presents a complication for x-line Diffie–Hellman, because it appears that the parties in a key exchange are obliged to carry out some zero checks to ensure that the inputs are not $\mathbf{x}(O)$ or $\mathbf{x}(T)$.

Bernstein shows that a modest modification to the map $\mathbf{x}$ allows such checks to be omitted entirely [2, Theorem 5.1]. The result is a Diffie–Hellman

key exchange where the inputs are not points on the x-line, but simple finite field elements, by using the mapping $\mathbf{x}_0 : \mathcal{E}_{(A,B)}(\mathbb{F}_q) \to \mathbb{F}_q$ defined by

$$\mathbf{x}_0 : P \longmapsto \begin{cases} 0 & \text{if } P = O \\ x & \text{if } P = (x, y) \end{cases}.$$

in place of $\mathbf{x}$. This means using the ladder with input $(X_P, Z_P) = (x, 1)$, where $x = \mathbf{x}_0(P)$ for any P on $\mathcal{E}_{(A,B)}$ or its twist—so x can be any element of $\mathbb{F}_q$—then returning $x_k = X_k Z_k^{q-2}$ in $\mathbb{F}_q$ instead of (X_k, Z_k). Note that $x_k = X_k/Z_k$ if $Z_k \neq 0$, and 0 otherwise; in either case, $x_k = \mathbf{x}_0([k]P)$. Bernstein's $\mathbf{x}_0$ therefore provides pseudo-completeness for the ladder on Montgomery curves, and an extremely simple and efficient key exchange based on the maps $x \mapsto x_k$.

6 Differential addition chains and higher-dimensional algorithms

An *addition chain* of length ℓ for a nonnegative integer k is an increasing sequence of nonnegative integers, $(c_0, \ldots, c_\ell)$, with $c_0 = 1$ and $c_\ell = k$, satisfying the following property: for all $0 < i \leq m$, there exist j and j' such that $c_i = c_j + c_{j'}$ with $0 \leq j \leq j' \leq i$. Addition chains have wide application in public key cryptography due to their correspondence with group exponentiations. In the context of elliptic curve cryptography, the existence of a length ℓ addition chain $(c_0, \ldots, c_\ell)$ for k implies that the scalar multiplication $(k, P) \mapsto [k]P$ can be computed using ℓ group operations. Thus, shorter addition chains require fewer operations, and ultimately yield faster scalar multiplication routines.

To use the fast x-line arithmetic of Montgomery curves, we need a special type of addition chain. A *differential addition chain* of length ℓ for an integer k is a sequence $(c_0, \ldots, c_{\ell+1})$ with $c_0 = 0$, $c_1 = 1$, and (for our purposes) $k \in \{c_\ell, c_{\ell+1}\}$, together with the property that for all $1 < i \leq \ell + 1$, there exist i', j and j' such that $c_i = c_j + c_{j'}$ and $c_{i'} = c_{j'} - c_j$ with $0 \leq i' \leq j \leq j' < i$. The existence of a length-ℓ differential addition chain for k implies that the pseudomultiplication $(k, \mathbf{x}(P)) \mapsto \mathbf{x}([k]P)$ can be computed using a total of ℓ differential operations (i.e., `xADD`s and `xDBL`s).

For an ℓ-bit scalar, the Montgomery ladder corresponds to a length $2\ell - 1$ differential addition chain $(c_0, \ldots, c_{2\ell})$; it requires two additions (one of which is a doubling) for each bit of the scalar except the top bit, where only one operation is required. In his search for shorter differential addition chains [39], Montgomery proved that any ℓ-bit prime scalar requires at least 1.440ℓ x-line operations, providing a lower bound on the number of operations required in a differential addition chain in the worst case [39, §3]. (The best-case exponents, powers of 2, require 1 operation per bit.)

6.1 Montgomery's Euclidean algorithms

The Montgomery ladder is a differential addition chain where the difference index $c_{i'}$ is in $\{1, 0\}$ throughout; this corresponds to every `xADD` taking the same difference $\mathbf{x}(P)$. While this yields a simple and uniform algorithm, allowing $c_{i'}$ to vary further can yield shorter addition chains and faster scalar multiplication.

Starting from the (subtractive) Euclidean algorithm for finding the greatest common divisor of two integers, Montgomery derived several algorithms for producing differential addition chains that were significantly shorter than his ladder. The idea is to have a coprime auxiliary exponent alongside the input k, and use the intermediate steps in the Euclidean algorithm to write down a differential addition chain for k. This process computes a differential addition chain for the auxiliary exponent as well, so these algorithms are inherently 2-dimensional. Here we discuss one of these algorithms, PRAC, beginning with its 2-dimensional core before returning to the 1-dimensional wrapper.

Algorithm 9 (`EUCLID2D`) is a version of Montgomery's 2-dimensional PRAC subroutine using only the "binary" transformations proposed by Montgomery in [39, Table 4]. Given a multiscalar (m, n) and $\mathbf{x}(P)$, $\mathbf{x}(Q)$, and $\mathbf{x}(P \ominus Q)$, it computes $\mathbf{x}([m]P \oplus [n]Q)$ using only `xADD` and `xDBL` operations. This variant chain was used (for non-uniform scalar multiplications) by the implementation of endomorphism-accelerated scalar multiplications on the curve in Example 3.

Since the difference arguments to the `xADD`s in `EUCLID2D` vary, we must be careful about handling differences like $\mathbf{x}(O)$ and $\mathbf{x}(T)$, which cause the `xADD` and `xDBL` we defined in Algorithms 1 and 2 to degenerate. For simplicity, in this section we suppose that `xADD` and `xDBL` have been extended to cover all inputs (perhaps using conditional code, which is acceptable in these non-uniform algorithms).

At first glance, `EUCLID2D` is much more complicated than `LADDER`, but it is in fact remarkably simple and elegant. Suppose we want to compute $[m]P \oplus [n]Q$. Lines 1 through 12 maintain the following invariants: $\gcd(\mathtt{s}_0, \mathtt{s}_1) = \gcd(m, n)$ (the reader may recognise a subtractive Euclidean algorithm here), and $(\mathtt{x}_0, \mathtt{x}_1, \mathtt{x}_\ominus) = (\mathbf{x}(R_0), \mathbf{x}(R_1), \mathbf{x}(R_1 \ominus R_0))$ for some R_0 and R_1 in $\mathcal{E}_{(A,B)}$ such that $[\mathtt{s}_0]R_0 \oplus [\mathtt{s}_1]R_1 = [m]P \oplus [n]Q$. Hence after the first `while` loop, having arrived at $\mathtt{s}_0 = 0$, we must have $\mathtt{s}_1 = \gcd(m, n)$ and $\mathtt{x}_1 = \mathbf{x}([m/\mathtt{s}_1]P \oplus [n/\mathtt{s}_1]Q)$. To complete the task, it suffices to carry out a 1-dimensional pseudomultiplication of $\mathtt{x}_1$ by $\mathtt{s}_1 = \gcd(m, n)$ (in Line 16 we use `LADDER`); of course, if m and n are random, then we expect $\mathtt{s}_1$ to be quite small at this point. The second `while` loop (Lines 13-14) slightly optimizes this final 1-dimensional pseudomultiplication by using pure pseudo-doubling to exhaust any power of 2 in $\mathtt{s}_1$, saving a few superfluous `xADD`s in the `LADDER` call.

If we had an efficient pseudo-tripling operation $\mathbf{x}(P) \mapsto \mathbf{x}([3]P)$ are relatively efficient, it would be advantageous to include Montgomery's "ternary" transformations in the loop of Algorithm 9. Independent analyses by Stam [47, Conjecture 3.29] and Bernstein [4, §3] conclude that this full version of Mont-

Algorithm 9: EUCLID2D: 2-dimensional scalar pseudomultiplication

Input: $m, n \in \mathbb{Z}_{>0}$, and $\mathbf{x}(P)$, $\mathbf{x}(Q)$, and $\mathbf{x}(Q \ominus P)$ for some P and Q in $\mathcal{E}_{(A,B)}(\mathbb{F}_q)$
Output: $\mathbf{x}([m]P \oplus [n]Q)$

1 $\big((\mathsf{s}_0, \mathsf{s}_1), (\mathsf{x}_0, \mathsf{x}_1, \mathsf{x}_\ominus)\big) \leftarrow \big((m, n), (\mathbf{x}(P), \mathbf{x}(Q), \mathbf{x}(Q \ominus P))\big)$
2 **while** $\mathsf{s}_0 \neq 0$ **do**
3 **if** $\mathsf{s}_1 < \mathsf{s}_0$ **then**
4 $\big((\mathsf{s}_0, \mathsf{s}_1), (\mathsf{x}_0, \mathsf{x}_1, \mathsf{x}_\ominus)\big) \leftarrow \big((\mathsf{s}_1, \mathsf{s}_0), (\mathsf{x}_1, \mathsf{x}_0, \mathsf{x}_\ominus)\big)$
5 **if** $\mathsf{s}_1 \leq 4\mathsf{s}_0$ **then** // "Fibonacci" step
6 $\big((\mathsf{s}_0, \mathsf{s}_1), (\mathsf{x}_0, \mathsf{x}_1, \mathsf{x}_\ominus)\big) \leftarrow \big((\mathsf{s}_0, \mathsf{s}_1 - \mathsf{s}_0), (\mathtt{xADD}(\mathsf{x}_1, \mathsf{x}_0, \mathsf{x}_\ominus), \mathsf{x}_1, \mathsf{x}_0)\big)$
7 **else if** $\mathsf{s}_0 \equiv \mathsf{s}_1 \pmod 2$ **then**
8 $\big((\mathsf{s}_0, \mathsf{s}_1), (\mathsf{x}_0, \mathsf{x}_1, \mathsf{x}_\ominus)\big) \leftarrow \big((\mathsf{s}_0, \frac{\mathsf{s}_1 - \mathsf{s}_0}{2}), (\mathtt{xADD}(\mathsf{x}_1, \mathsf{x}_0, \mathsf{x}_\ominus), \mathtt{xDBL}(\mathsf{x}_1), \mathsf{x}_\ominus)\big)$
9 **else if** $\mathsf{s}_1 \equiv 0 \pmod 2$ **then**
10 $\big((\mathsf{s}_0, \mathsf{s}_1), (\mathsf{x}_0, \mathsf{x}_1, \mathsf{x}_\ominus)\big) \leftarrow \big((\mathsf{s}_0, \mathsf{s}_1/2), (\mathsf{x}_0, \mathtt{xDBL}(\mathsf{x}_1), \mathtt{xADD}(\mathsf{x}_1, \mathsf{x}_\ominus, \mathsf{x}_0))\big)$
11 **else**
12 $\big((\mathsf{s}_0, \mathsf{s}_1), (\mathsf{x}_0, \mathsf{x}_1, \mathsf{x}_\ominus)\big) \leftarrow \big((\mathsf{s}_0/2, \mathsf{s}_1), (\mathtt{xDBL}(\mathsf{x}_0), \mathsf{x}_1, \mathtt{xADD}(\mathsf{x}_0, \mathsf{x}_\ominus, \mathsf{x}_1))\big)$
13 **while** $\mathsf{s}_1 \equiv 0 \pmod 2$ **do**
14 $(\mathsf{s}_1, \mathsf{x}_1) \leftarrow (\mathsf{s}_1/2, \mathtt{xDBL}(\mathsf{x}_1))$
15 **if** $\mathsf{s}_1 > 1$ **then**
16 $\mathsf{x}_1 \leftarrow \mathtt{LADDER}(\mathsf{s}_1, \mathsf{x}_1)$
17 **return** x_1 // $\mathsf{x}_1 = \mathbf{x}([m]P \oplus [n]Q)$

gomery's 2-dimensional PRAC subroutine computes $\mathbf{x}([m]P \oplus [n]Q)$ for ℓ-bit multiscalars (m, n) using an average total of 1.82ℓ differential operations.

Returning to the 1-dimensional problem, Montgomery suggests computing $\mathbf{x}([k]P)$ as $\mathbf{x}([m]P \oplus [n]Q)$, where (m, n) and $Q = [\lambda]P$ satisfy $[m + n\lambda]P = [k]P$ for some λ chosen to minimize the complexity of the pseudomultiplication. If we can precompute $\mathbf{x}([\lambda]P)$ from $\mathbf{x}(P)$, then any ℓ-bit pseudomultiplication can be performed as an $\ell/2$-bit double-pseudomultiplication, and the resulting differential addition chains have average length 0.92ℓ [47, Corollary 3.32]. But without precomputation, and following his heuristics for choosing λ, Montgomery's PRAC algorithm produces differential addition chains of average length 1.64ℓ for ℓ-bit scalars [47, Corollary 3.35].

Algorithm 10 is a simplified version of PRAC: we omit the repeated-tripling step that, like the ternary steps omitted in its subroutine Algorithm 9, presupposes the existence of rapid pseudo-tripling on $\mathcal{E}_{(A,B)}$. The idea of taking $(m, n) = (\lfloor k/\varphi \rfloor, k - \lfloor k/\varphi \rfloor)$, where $\varphi = (1 + \sqrt{5})/2$ is the famous golden ratio, is that this induces a sequence of roughly $(\frac{1}{2} \log_\varphi 2)\ell$ of the relatively cheap so-called Fibonacci branch (Lines 5-6) of Algorithm 9, followed by roughly $\frac{1}{2}\ell$ branches distributed as for random multiscalars. We refer the reader to Stam's thesis [47, §3.3.4] for further details and analysis.

6.2 Higher-dimensional ladder analogues

Although the short addition chains produced by Montgomery's Euclidean algorithms are suitable for his original application to ECM, their non-uniform

Algorithm 10: PRAC: (simplified) 1-D Euclidean pseudomultiplication

Input: $k \in \mathbb{Z}_{>0}$, and $\mathbf{x}(P)$ for some P in $\mathcal{E}_{(A,B)}(\mathbb{F}_q)$
Output: $\mathbf{x}([k]P)$

$(\mathsf{s}, \mathsf{x}) \leftarrow (k, \mathbf{x}(P))$
while $\mathsf{s} \equiv 0 \pmod 2$ **do**
  $(\mathsf{s}, \mathsf{x}) \leftarrow (\mathsf{s}/2, \mathtt{xDBL}(\mathsf{x}))$
$\mathsf{r} \leftarrow \lfloor \mathsf{s}/\varphi \rfloor$ **where** $\varphi = (1+\sqrt{5})/2$
$\mathsf{x} \leftarrow \mathtt{EUCLID2D}((\mathsf{r}, \mathsf{s}-\mathsf{r}), (\mathsf{x}, \mathsf{x}, \mathbf{x}(0)))$
return x

and variable-time behavior makes them less suitable for application to ECC, where uniformity is a mandatory first step towards hiding secret exponents from adversaries exploiting side-channels. For cryptographic multiscalar multiplications, we may need higher-dimensional differential addition chains that share the uniform behavior of the 1-dimensional Montgomery ladder.

Bernstein defines a 2-dimensional analogue of the Montgomery ladder in [4] (the "binary chain"). Given an ℓ-bit multiscalar (m, n) and the values of $\mathbf{x}(P)$, $\mathbf{x}(Q)$, $\mathbf{x}(P \ominus Q)$ and $\mathbf{x}(P \oplus Q)$, this algorithm computes $\mathbf{x}([m]P \oplus [n]Q)$ using exactly 3ℓ differential operations. If `xDBL` and `xADD` are implemented in a uniform way, then Bernstein's algorithm is also uniform: it is a sequence of ℓ steps, each consisting of one `xDBL` and two `xADD`s. Further, in each of those steps, the argument of `xDBL` is shared by one of the `xADD`s, and in some contexts these calls may be merged to use some shared computations.

Every call to `xADD` in Bernstein's algorithm takes one of the four input values $\mathbf{x}(P)$, $\mathbf{x}(Q)$, $\mathbf{x}(P \ominus Q)$ and $\mathbf{x}(P \oplus Q)$ as its difference argument; this makes it possible to recover the correct y-coordinate for the output $\mathbf{x}([m]P \oplus [n]Q)$ using exactly the same technique as in §4.3. This makes Bernstein's chain a viable option for computing the multiscalar multiplication during the verification phase of signature schemes that take advantage of differential arithmetic.

Brown defined an n-dimensional analogue of the Montgomery ladder in [11]. Given an n-dimensional multiscalar $(m_1, \ldots, m_n)$ in $\mathbb{Z}^n$, Brown's algorithm uses x-line arithmetic to compute $\mathbf{x}([m_1]P_1 \oplus \cdots \oplus [m_n]P_n)$ from the $(3^n - 1)/2$ input values $\mathbf{x}([e_1]P_1 \oplus \cdots \oplus [e_n]P_n)$ with the $e_i \in \{-1, 0, 1\}$ and not all zero.

Example 3 (A Montgomery $\mathbb{Q}$-curve reduction [18]) The Montgomery curve

$$\mathcal{E} : y^2 = x(x^2 + Ax + 1) \quad \text{over} \quad \mathbb{F}_{p^2} = \mathbb{F}_{2^{127}-1}(\sqrt{-1})$$

with $A = 451165543445558750850176275933214854 21 + 2415910908\sqrt{-1}$ has $\#\mathcal{E}(\mathbb{F}_p) = 4r$ and $\#\mathcal{E}'(\mathbb{F}_p) = 8r'$, where r and r' are 252- and 251-bit primes, respectively. In [18] we see that $\mathcal{E}$ is equipped with an efficiently computable endomorphism ψ of degree $2p$, which acts on $\mathcal{E}(\mathbb{F}_{p^2})[r]$ as $[\lambda]$ where $\lambda^2 \equiv -2 \pmod r$. The classic Gallant–Lambert–Vanstone (GLV) technique [25] could be used to compute 1-dimensional scalar multipications $[k]P$ as 2-dimensional multiplications $[m]P \oplus [n](\psi(P))$, with m and n of roughly 128 bits. The scalar multiplication implementation in [18] transports the GLV approach to

the x-only setting, simultaneously exploiting Montgomery arithmetic. First, we compute $\mathbf{x}(P) \mapsto \mathbf{x}(\psi(P))$ and $\mathbf{x}(P) \mapsto \mathbf{x}((\psi - 1)P) = \mathbf{x}(\psi(P) \ominus P)$. Then given any k in $[0, r)$ we can compute $\mathbf{x}([k]P) = \mathbf{x}([m]P \oplus [n]\psi(P))$, where m and n are 128-bit scalars, using Algorithm 9 for public scalars and Bernstein's uniform 2-dimensional binary chain for private scalars.

7 Generalizations and other applications

The Montgomery ladder is a general algorithm that works in any abelian group, and in group quotients where analogues of `xADD` are available. The most interesting groups of this kind—at least from a cryptographic point of view—are other models of elliptic curves, and Jacobians of hyperelliptic curves.

7.1 The Montgomery ladder on other models of elliptic curves

The Montgomery ladder can be applied to any model of an elliptic curve, using either the usual addition and doubling operations, or the analogue of x-only arithmetic. For Weierstrass models, the chief interest in the Montgomery ladder is its side-channel resistance when computing scalar multiples with secret scalars, as explored by Brier and Joye in [10]. Gaudry and Lubicz [27] used the classical complex-analytic theory of theta functions to derive pseudo-group law formulæ for $\mathbb{P}^1 \cong \mathcal{E}_\lambda/\langle\ominus\rangle$, where $\mathcal{E}_\lambda\colon y^2 = x(x-1)(x-\lambda)$ is a Legendre model. Their formulæ offer trade-offs with the `xADD` and `xDBL` operations on Montgomery curves, which could make them favorable in certain scenarios.

Castryck, Galbraith, and Farashahi [14] made the striking suggestion of using Montgomery curves *in conjunction* with their corresponding twisted Edwards models. The transformations of (4) and (5) are extremely easy to compute; this allows a mixed arithmetic, passing back-and-forth between x-only pseudo-additions on the Montgomery curve and v-only pseudo-doublings on the corresponding Edwards curve.

7.2 The Montgomery ladder on hyperelliptic curves

The Montgomery ladder has been applied with great success in hyperelliptic cryptography based on genus-2 curves. The story begins with Smart and Siksek [46], who observed that we can use pseudo-additions to instantiate Diffie–Hellman key exchange on Kummer surfaces (quotients of Jacobians of genus-2 curves by ± 1). Duquesne [20] made this concrete by combining the ladder with explicit formulæ for arithmetic on genus-2 curves of the form $\mathcal{H} : y^2 = xf(x)$, where f is a squarefree degree-4 polynomial. While the factor of x on the right-hand-side also appears in the defining equations of Montgomery curves, Duquesne's curves are not a true genus-2 Montgomery analogue: we would expect a particularly simple action on the Jacobian of the elements of the kernel of a $(2, 2)$-isogeny factoring [2], for example, mirroring the special form

of the translation-by-T map on Montgomery curves, but no such structure is imposed by Duquesne's form. Nevertheless, this special curve form allows a small speedup over general genus-2 arithmetic. Duquesne later described the Montgomery ladder on Kummer surfaces of arbitrary genus-2 curves [21], building on Flynn's arithmetic of general Kummer surfaces [13, Chapter 3].

Gaudry used the Montgomery ladder for efficient pseudomultiplication on a model of the Kummer with especially fast pseudo-addition and pseudo-doubling operations [26], building on observations of D. V. and G. V. Chudnovsky [15]. Here, the efficiency really is a consequence of a special 2-torsion structure: all of the 2-torsion points are defined over $\mathbb{F}_q$, and their translations act linearly on the Kummer by diagonal and permutation matrices. This approach has successfully used in high-speed Diffie–Hellman implementations [9, 6, 44], and a hyperelliptic generalization of ECM factorization [17].

An analogue of y-coordinate recovery (see §4.3) exists in genus 2: Chung and the authors give an explicit algorithm in [16], recovering Jacobian elements from the output of the Montgomery ladder on the Kummer. This enables the implementation of full signature schemes using Kummer surfaces [44].

All of these ideas and techniques are carried much further in the setting of higher-dimensional abelian and Kummer varieties by Lubicz and Robert [36].

References

1. Agence nationale de la sécurité des sysèmes d'information (ANSSI). Mécanismes cryptographiques: Règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques. `http://www.ssi.gouv.fr/uploads/2015/01/RGS_v-2-0_B1.pdf`, 2014.
2. Daniel J. Bernstein. Can we avoid zero tests for fast elliptic-curve arithmetic? `https://cr.yp.to/ecdh/curvezero-20060726.pdf`, 2006.
3. Daniel J. Bernstein. Curve25519: New Diffie-Hellman speed records. In Yung et al. [48], pages 207–228.
4. Daniel J. Bernstein. Differential addition chains. `http://cr.yp.to/ecdh/diffchain-20060219.pdf`, 2006.
5. Daniel J. Bernstein, Peter Birkner, Marc Joye, Tanja Lange, and Christiane Peters. Twisted Edwards curves. In Serge Vaudenay, editor, *Progress in Cryptology - AFRICACRYPT 2008, First International Conference on Cryptology in Africa, Casablanca, Morocco, June 11-14, 2008. Proceedings*, volume 5023 of *Lecture Notes in Computer Science*, pages 389–405. Springer, 2008.
6. Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Peter Schwabe. Kummer strikes back: New DH speed records. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 317–337. Springer, 2014.
7. Daniel J. Bernstein and Tanja Lange. Faster addition and doubling on elliptic curves. In Kaoru Kurosawa, editor, *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, volume 4833 of *Lecture Notes in Computer Science*, pages 29–50. Springer, 2007.
8. Ingrid Biehl, Bernd Meyer, and Volker Müller. Differential fault attacks on elliptic curve cryptosystems. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August*

20-24, 2000, Proceedings, volume 1880 of *Lecture Notes in Computer Science*, pages 131–146. Springer, 2000.
9. Joppe W. Bos, Craig Costello, Hüseyin Hisil, and Kristin E. Lauter. Fast cryptography in genus 2. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 194–210. Springer, 2013.
10. Eric Brier and Marc Joye. Weierstraß elliptic curves and side-channel attacks. In David Naccache and Pascal Paillier, editors, *Public Key Cryptography, 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002, Paris, France, February 12-14, 2002, Proceedings*, volume 2274 of *Lecture Notes in Computer Science*, pages 335–345. Springer, 2002.
11. Daniel R. L. Brown. Multi-dimensional Montgomery ladders for elliptic curves. `http://eprint.iacr.org/2006/220`, 2006.
12. J. W. S. Cassels. *Lectures on elliptic curves*, volume 240 of *London Mathematical Society Student Texts*. Cambridge University Press, 1991.
13. J. W. S. Cassels and E. V. Flynn. *Prolegomena to a middlebrow arithmetic of curves of genus 2*, volume 230 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 1996.
14. Wouter Castryck, Steven D. Galbraith, and Reza Rezaeian Farashahi. Efficient arithmetic on elliptic curves using a mixed Edwards-Montgomery representation. *IACR Cryptology ePrint Archive*, 2008:218, 2008.
15. David V Chudnovsky and Gregory V Chudnovsky. Sequences of numbers generated by addition in formal groups and new primality and factorization tests. *Advances in Applied Mathematics*, 7(4):385–434, 1986.
16. Ping Ngai Chung, Craig Costello, and Benjamin Smith. Fast, uniform scalar multiplication for genus 2 Jacobians with fast Kummers. In Roberto Avanzi and Howard Heys, editors, *Selected Areas in Cryptography - SAC 2016 - 23rd International Conference, Newfoundland, NL, Canada, August 10-12, 2016, Revised Selected Papers*, Lecture Notes in Computer Science. Springer, 2016.
17. Romain Cosset. Factorization with genus 2 curves. *Math. Comput.*, 79(270):1191–1208, 2010.
18. Craig Costello, Hüseyin Hisil, and Benjamin Smith. Faster compact Diffie–Hellman: Endomorphisms on the x-line. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 183–200. Springer, 2014.
19. Christophe Doche, Thomas Icart, and David R. Kohel. Efficient scalar multiplication by isogeny decompositions. In Yung et al. [48], pages 191–206.
20. Sylvain Duquesne. Montgomery scalar multiplication for genus 2 curves. In Duncan A. Buell, editor, *Algorithmic Number Theory, 6th International Symposium, ANTS-VI, Burlington, VT, USA, June 13-18, 2004, Proceedings*, volume 3076 of *Lecture Notes in Computer Science*, pages 153–168. Springer, 2004.
21. Sylvain Duquesne. Traces of the group law on the Kummer surface of a curve of genus 2 in characteristic 2. *Mathematics in Computer Science*, 3(2):173–183, 2010.
22. ECC Brainpool. ECC Brainpool Standard Curves and Curve Generation. `http://www.ecc-brainpool.org/download/Domain-parameters.pdf`, 2005.
23. Harold Edwards. A normal form for elliptic curves. *Bulletin of the American Mathematical Society*, 44(3):393–422, 2007.
24. Paul Zimmermann et al. GMP-ECM software, 2016. `http://ecm.gforge.inria.fr/`.
25. Robert P. Gallant, Robert J. Lambert, and Scott A. Vanstone. Faster point multiplication on elliptic curves with efficient endomorphisms. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 190–200. Springer, 2001.
26. Pierrick Gaudry. Fast genus 2 arithmetic based on Theta functions. *J. Mathematical Cryptology*, 1(3):243–265, 2007.

27. Pierrick Gaudry and David Lubicz. The arithmetic of characteristic 2 Kummer surfaces and of elliptic Kummer lines. *Finite Fields and Their Applications*, 15(2):246–260, 2009.
28. Mike Hamburg. Ed448-Goldilocks, a new elliptic curve. Cryptology ePrint Archive, Report 2015/625, 2015. `http://eprint.iacr.org/2015/625`.
29. Hüseyin Hisil, Kenneth Koon-Ho Wong, Gary Carter, and Ed Dawson. Twisted Edwards curves revisited. In Josef Pieprzyk, editor, *Advances in Cryptology - ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7-11, 2008. Proceedings*, volume 5350 of *Lecture Notes in Computer Science*, pages 326–343. Springer, 2008.
30. Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.
31. David Kohel. Arithmetic of split kummer surfaces: Montgomery endomorphism of edwards products. In Yeow Meng Chee, Zhenbo Guo, San Ling, Fengjing Shao, Yuansheng Tang, Huaxiong Wang, and Chaoping Xing, editors, *Coding and Cryptology - Third International Workshop, IWCC 2011, Qingdao, China, May 30-June 3, 2011. Proceedings*, volume 6639 of *Lecture Notes in Computer Science*, pages 238–245. Springer, 2011.
32. Adam Langley, Mike Hamburg, and Sean Turner. Elliptic curves for security. Internet Research Task Force RFC 7748, 2016. `https://tools.ietf.org/html/rfc7748`.
33. Hendrik W Lenstra Jr. Factoring integers with elliptic curves. *Annals of mathematics*, pages 649–673, 1987.
34. Chae Hoon Lim and Pil Joong Lee. A key recovery attack on discrete log-based schemes using a prime order subgroupp. In Burton S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 249–263. Springer, 1997.
35. Julio López and Ricardo Dahab. Fast multiplication on elliptic curves over GF(2^{m}) without precomputation. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 316–327. Springer, 1999.
36. David Lubicz and Damien Robert. Arithmetic on abelian and Kummer varieties. *Finite Fields and Their Applications*, 39:130–158, 2016.
37. Victor S Miller. Use of elliptic curves in cryptography. In *Advances of Cryptology - CRYPTO*, pages 417–426. Springer, 1985.
38. Peter L Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of computation*, 48(177):243–264, 1987.
39. Peter L. Montgomery. Evaluating recurrences of form $x_{m+n} = f(x_m, x_n, x_{m-n})$ via Lucas chains. `https://cr.yp.to/bib/1992/montgomery-lucas.ps`, 1992.
40. National Institute for Standards and Technology (NIST). Digital Signature Standard. Federal Information Processing Standards Publication 186-4. `http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf`, 2013.
41. Katsuyuki Okeya, Hiroyuki Kurumatani, and Kouichi Sakurai. Elliptic curves with the Montgomery-form and their cryptographic applications. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography, Third International Workshop on Practice and Theory in Public Key Cryptography, PKC 2000, Melbourne, Victoria, Australia, January 18-20, 2000, Proceedings*, volume 1751 of *Lecture Notes in Computer Science*, pages 238–257. Springer, 2000.
42. Katsuyuki Okeya and Kouichi Sakurai. Efficient elliptic curve cryptosystems from a scalar multiplication algorithm with recovery of the y-coordinate on a Montgomery-form elliptic curve. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 126–141. Springer, 2001.

43. John M Pollard. Monte Carlo methods for index computation (mod p). *Mathematics of computation*, 32(143):918–924, 1978.
44. Joost Renes, Peter Schwabe, Benjamin Smith, and Lejla Batina. μKummer: Efficient hyperelliptic signatures and key exchange on microcontrollers. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems – CHES 2016: 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, pages 301–320, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.
45. Igor E. Shparlinski and Daniel Sutantyo. Distribution of elliptic twin primes in isogeny and isomorphism classes. *Journal of Number Theory*, 137:1 – 15, 2014.
46. Nigel P. Smart and Samir Siksek. A fast Diffie–Hellman protocol in genus 2. *Journal of cryptology*, 12(1):67–73, 1999.
47. Martijn Stam. *Speeding up subgroup cryptosystems*. PhD thesis, Technische Universiteit Eindhoven, 2003.
48. Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors. *Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography, New York, NY, USA, April 24-26, 2006, Proceedings*, volume 3958 of *Lecture Notes in Computer Science*. Springer, 2006.
49. Paul Zimmermann and Bruce Dodson. 20 years of ECM. In Florian Hess, Sebastian Pauli, and Michael E. Pohst, editors, *Algorithmic Number Theory, 7th International Symposium, ANTS-VII, Berlin, Germany, July 23-28, 2006, Proceedings*, volume 4076 of *Lecture Notes in Computer Science*, pages 525–542. Springer, 2006.