Faster Montgomery and double-add ladders for short Weierstrass curves

Mike Hamburg

Rambus mhamburg@rambus.com

Abstract. The Montgomery ladder and Joye double-add ladder are well-known algorithms for elliptic curve scalar multiplication with a regular structure. The Montgomery ladder is best known for its implementation on Montgomery curves, which requires $5\mathbf{M}+4\mathbf{S}+1\mathbf{m}+8\mathbf{A}$, and 6 field registers. Here $(\mathbf{M}, \mathbf{S}, \mathbf{m}, \mathbf{A})$ represent respectively field multiplications, squarings, multiplications by a curve constant, and additions or subtractions. This ladder is also *complete*, meaning that it works on all input points and all scalars.

Many protocols do not use Montgomery curves, but instead use prime-order curves in short Weierstrass form. As of 2011, the fastest formulas for the Montgomery and Joye ladders on these curves each required $9\mathbf{M}+5\mathbf{S}+18\mathbf{A}$ per bit. In 2017, Kim et al. improved this for the Montgomery ladder only to $8\mathbf{M}+4\mathbf{S}+12\mathbf{A}+1\mathbf{H}$ per bit using 9 registers, where the $\mathbf{H}$ represents a halving. Hamburg simplified Kim et al.'s formulas to $8\mathbf{M}+4\mathbf{S}+8\mathbf{A}+1\mathbf{H}$ using 6 registers.

Here we present improved formulas which compute the Montgomery ladder on short Weierstrass curves using $8\mathbf{M}+3\mathbf{S}+7\mathbf{A}$ per bit, and requiring 6 registers. We also give formulas for the Joye ladder that use $9\mathbf{M}+3\mathbf{S}+7\mathbf{A}$ per bit, requiring 5 registers. We also show a novel technique to make these ladders complete when the curve order is not divisible by 2 or 3, at a modest increase in cost.

Finally, we discuss curve invariants, exceptional points, side-channel protection and how to set up and finish these ladder operations.

Keywords: Elliptic Curve Cryptography, Montgomery Ladder, Joye Ladder, Short Weierstrass Curve, Scalar Multiplication

1 Introduction and related work

The core operation of most elliptic curve cryptography algorithms is *scalar multiplication*, in which an element P of the elliptic curve group is multiplied by an integer ("scalar") x. Therefore, considerable study has been devoted to optimizing scalar multiplication algorithms.

The present paper is concerned with *variable-base* scalar multiplication algorithms, meaning that P is not known ahead of time, so no precomputation has been done on it. Typically x is secret, so to avoid side-channel attacks, the algorithm's timing and control flow should not depend on x.

1.1 The Montgomery and double-add ladders

The Montgomery ladder [Mon87] is a general algorithm for computing a power or scalar multiple of a group element. This algorithm's regular structure is conducive to implementations that resist side-channel attack. This technique is fastest on elliptic curves in the *Montgomery form*

$$y^2 = x^3 + Ax^2 + x.$$

Montgomery ladder	Double-add ladder
$(Q_0, Q_1) \leftarrow (0, P)$	$(Q_0, Q_1) \leftarrow (0, P)$
For $i = n - 1$ down to 0:	For $i = 0$ to $n - 1$:
$\quad b \leftarrow k_i$	$\quad b \leftarrow k_i$
$\quad Q_{\neg b} \leftarrow Q_{\neg b} + Q_b$	$\quad Q_{\neg b} \leftarrow 2 \cdot Q_{\neg b} + Q_b$
$\quad Q_b \leftarrow 2Q_b$	Output $Q_0 = x \cdot P$
Output $Q_0 = x \cdot P$	

Figure 1: The Montgomery and double-add ladders

But it can be efficiently applied on any elliptic curve, in particular those in the *short Weierstrass form*

$$y^2 = x^3 + ax + b.$$

Joye's double-add ladder [Joy07] is a similar algorithm, which is also used in efficient implementations of elliptic curve scalar multiplication. In some sense, the Montgomery ladder and the Joye algorithm are dual [Wal17].

Each algorithm takes as input a group element P and a scalar k, and computes $k \cdot P$. Let k have a binary representation

$$k := \sum_{i=0}^{n-1} 2^i k_i \text{ where each } k_i \in \{0, 1\}.$$

The two algorithms are shown in Figure 1.

While the Montgomery ladder can be implemented using only Q_0 and Q_1, many implementations also use the base point $P = Q_1 - Q_0$. Likewise, implementations of the Joye ladder often track a third point

$$R := Q_1 + Q_0 = 2^i \cdot P$$

in the ith iteration. In these cases, the ladder update steps are rearrangements of each other: both map triples of points (P, Q, R) to $(P, P + Q, 2 \cdot R)$ in some order. This often makes it possible to convert between formulas for the Montgomery ladder and those for the double-add ladder.

However, the converted formulas aren't always equally performant or side-channel resistant. They also may need adjustment if the three points don't use the same representation in the ladder state. For example, the base point never changes in the Montgomery ladder, so it might be stored in affine coordinates; or the ladder might track the y-coordinates of some points but not others.

1.2 Co-Z coordinates

Elliptic curve implementations typically calculate using a projectivized version of the elliptic curve for efficiency. Instead of storing (x, y), points are represented in either *projective coordinates* as $(xZ : yZ : Z)$ or in *Jacobian coordinates* as $(xZ^2 : yZ^3 : Z)$. This avoids costly finite-field divisions except at the end of the computation. When storing multiple points in a ladder, the straightforward way is to use a separate Z-coordinate for each point.

Co-Z formulas [Mel07, GJM10, Riv11] instead use the same Z-coordinate for all points in the ladder state. This reduces memory usage, and it can also improve performance because often the first step in a point addition is to rescale both points to have the same Z-coordinate, and that step is not needed for co-Z representations. In some cases, that

Z-coordinate need not even be stored, as having the X and/or Y coordinates for multiple points will provide enough information to recover Z. This provides a further savings.

Most formulas on short Weierstrass curves need special cases around the identity point (also known as the "neutral point" or the "point at infinity"), which is written with $Z = 0$. This is especially true for co-Z formulas, because if one point has $Z = 0$ then they all do; the finite points would then be represented as $(0 : 0 : 0)$, which is indeterminate. In particular, this means that a co-Z Montgomery ladder cannot begin with the points $(0, P)$: it must begin instead after the first nonzero key bit and the state $(P, 2P)$. Likewise, the Joye ladder must begin with (P, P). To avoid side-channel attacks, the secret scalar is typically rewritten by adding multiples of the curve order q, so that its first[1] bit is always 1, and the ladder is begun after the first step.

1.3 The Kim et al. formulas

In 2017, Kim et al. published a variant of the Montgomery ladder with "on-the-fly adaptive coordinates" of the form $(X_1, Y_1, X_2, Y_2, S, T, R)$ [KCK+17]. These formulas improved the previous state of the art of $9\mathbf{M} + 5\mathbf{S} + 18\mathbf{A}$ per bit [Riv11] to $8\mathbf{M} + 4\mathbf{S} + 12\mathbf{A} + 1\mathbf{H}$ per bit, using 9 registers which each hold one field element. The formulas are highly complex, with each operation depending on two bits of the key instead of one. Hamburg presented a simplified and optimized version of Kim et al.'s main ladder formula at the CHES 2017 rump session [Ham17], using modified Jacobian co-Z coordinates of the form $(3X_0, 2Y_0, X_1 - X_0, X_2 - X_0, 2M)$ where the three points lie on the line $Y = M(X - X_0) + Y_0$. Hamburg's formulas require $8\mathbf{M} + 4\mathbf{S} + 8\mathbf{A} + 1\mathbf{H}$ per bit, and 6 field registers.

1.4 Our contribution

In Section 2.1 and Appendix B we give two new formulas for the Montgomery ladder. Both use $8\mathbf{M} + 3\mathbf{S} + 7\mathbf{A}$ and 6 field registers. The formula in Section 2.1 is closely related to Hamburg's rump session formula. It keeps a ladder state of (X_{QP}, X_{RP}, Y_P, M), dropping the X_P from [Ham17] (called X_0 in that paper). In addition to improving performance, this change gives an improvement in resistant to horizontal side-channel attacks: it no longer reuses the round outputs as intermediates within the round.

An alternative formula in Appendix B instead tracks $(X_{QP}, X_{RP}, {X_{RQ}}^2, Y_Q, Y_R)$. This formula has the same performance as the one in Section 2.1 except that it requires an extra conditional swap for Y_Q and Y_R. It can be parallelized efficiently over four multiplication units instead of three, but does not share the additional resistance to horizontal side-channel attacks. [2]

In Section 2.2 we show a Joye ladder formula, based on our first Montgomery ladder formula, using $9\mathbf{M} + 3\mathbf{S} + 7\mathbf{A}$ and 5 registers. We are not aware of any previous Montgomery or Joye ladder formula that requires only 5 field registers.

Counting registers is somewhat tricky. Our register counts do not count the scalar itself, the curve constants, or small constants such as 2 and 3. They also assume that the processor supports multiplication in place ($X \leftarrow X \cdot Y$) and for the complete Joye formulas, reverse subtraction in place $X \leftarrow Y - X$). If it does not support these operations, an additional register is required.

We show how to set up and finalize the ladder state for either x-only or (x, y) calculations. For the Joye ladder, (x, y) calculations generally require an extra $\mathbf{M}$ per bit to track the

[1] "First" meaning least-significant for the Joye ladder, or most-significant for the Montgomery ladder. For example, on the Joye ladder we would add q to the scalar if it is even.

[2] A preprint of this paper presents the formulas in Appendix B as tracking $(X_{QP}, X_{RP}, M, \bar{M})$ where $\bar{M}$ is an additional slope variable. The present version performs the same calculations, but has a different boundary between the end of one iteration and the beginning of the next. We have chosen to use the present version because it is more similar to the calculations in the rest of the paper.

Curve	Ladder	Ref	Cost per scalar bit	Swap	Regs
Mont	Mont†	[Mon87]	$5\mathbf{M} + 4\mathbf{S} + \mathbf{m} + 8\mathbf{A}$	2	6
Weier	Either	[Riv11]	$9\mathbf{M} + 5\mathbf{S} + 18\mathbf{A}$	2	9
Weier	Mont†	[SM16]	$10\mathbf{M} + 5\mathbf{S} + 2\mathbf{m} + 17\mathbf{A}$	2	9
Weier	Mont	[KCK+17]	$8\mathbf{M} + 4\mathbf{S} + 12\mathbf{A} + 1\mathbf{H}$	$*$	9
Weier	Mont	[Ham17]	$8\mathbf{M} + 4\mathbf{S} + 8\mathbf{A} + 1\mathbf{H}$	1	6
Weier	Mont	Figure 3	$8\mathbf{M} + 3\mathbf{S} + 7\mathbf{A}$	1	6
Weier	Mont	Figure 6	$8\mathbf{M} + 3\mathbf{S} + 7\mathbf{A}$	2	6
Weier	Mont†	Suppl.	$9\mathbf{M} + 3\mathbf{S} + 8\mathbf{A} + 1\mathbf{E}$	6	6
Weier	Joye	Figure 4	$9\mathbf{M} + 3\mathbf{S} + 7\mathbf{A}$	2	5
Weier	Joye†	Suppl.	$9\mathbf{M} + 3\mathbf{S} + 9\mathbf{A} + 1\mathbf{E}$	6	5

Notes:

- † These ladders are complete, at least for a subset of curves. Our new complete ladder formulas require comparisons, written as $\mathbf{E}$.
- * The Kim et al. ladder is not written using conditional swaps, and we did not attempt to convert it.
- The costs do not include setup or finalization. Setup may include an on-curve check and finalization always includes a division. Division typically costs between $1\mathbf{S}$/bit and $(1\mathbf{S} + 1\mathbf{M})$/bit, depending on the modulus and on memory constraints.
- Some of these formulas have ways to turn multiplications into squarings at the cost of several additions, possibly requiring more registers.
- The register counts for our new formulas count setup and finalization, when using the technique fromSection 2.4.2.
- These costs are for x-only ladders; recovering y requires extra storage. It also costs an extra $1\mathbf{M}$/bit for our Joye ladders but not our Montgomery ladders.

Figure 2: Comparison to previous work.

Z coordinate. We also give invariants on the ladder state and a discussion of side-channel protection.

See Figure 2 for a comparison of our new formulas to past work. They are still not as fast as the Montgomery ladder on Montgomery curves [Mon87]: approximating $1\mathbf{S} \approx 0.75\mathbf{M}, \mathbf{m} \approx 0.25\mathbf{M}, 1\mathbf{A} \approx \mathbf{H} \approx 0.1\mathbf{M}$ gives an estimate of 21% more compute time per scalar bit, excluding the final division. This is an improvement from [Ham17] and [Riv11], which use respectively 31% and 61% more compute time per bit than [Mon87]. Our new formulas also support the usual $\mathbf{S} - \mathbf{M}$ tradeoffs at the cost of extra additions and registers, but we present them in their simplest and most compact form.

The formulas in Section 2.1, Section 2.2 and Appendix B are not complete: they break down if the neutral point appears in the ladder state but not in any other case. In Section 3 we give an analysis of this problem and a novel solution, which is not specific to our formulas: it potentially allows other ladders to implement complete scalar multiplication at a modest performance cost. These formulas are given in the supplemental material.

2 Ladder Formulas

Our ladder works on short Weierstrass curves over large-characteristic fields. They are derived from the following theorem:

Theorem 1 (Ladder formulas)**.** *Let*

$$P = (x_P, y_P), \qquad Q := (x_Q, y_Q), \qquad R := P + Q := (x_R, y_R)$$

be the state of the Montgomery ladder on an elliptic curve $y^2 = x^3 + ax + b$ defined over a field of characteristic other than 2. The three points $(P, Q, -R)$ lie on a line with slope

$m := (y_Q + y_R)/(x_Q - x_R)$. *Let*

$$P = (x_P, y_P), \qquad S := Q + R = (x_S, y_S), \qquad T := 2R = (x_T, y_T)$$

be the state after a ladder operation, where $(P, S, -T)$ *lie on a line of slope* m'. *Let*

$$r := \frac{x_Q - x_R}{2y_R}, \qquad s := (x_R - x_P) \cdot r, \qquad t := \frac{2y_R}{x_Q - x_R}, \qquad u := \frac{2y_Q}{x_Q - x_R}.$$

Then

$$\begin{aligned} x_S - x_P &= -t \cdot u = t \cdot (t - 2m) \\ x_T - x_P &= s^2 - 2y_P \cdot r \\ m' &= t - m - s. \end{aligned}$$

Proof. Deferred until Appendix A. □

These formulas are compatible with Jacobian coordinates: if (m, x, y) are replaced by (mZ, xZ^2, yZ^3) then the outputs will also be in that form, with the same Z.

2.1 Formulas for the Montgomery ladder

Theorem 1 gives a straightforward strategy to implement the Montgomery ladder. The state variables will be

$$\begin{aligned} X_{QP} &:= (x_Q - x_P) \cdot Z^2 \\ X_{RP} &:= (x_R - x_P) \cdot Z^2, \\ M &:= m \cdot Z \\ Y_P &:= 2y_P \cdot Z^3 \end{aligned}$$

For most curves, the value of Z need not be represented; see Section 2.4. On each step, we calculate $Y_R = Y_P + 2MX_{RP}$. Then Z will be multiplied by a local denominator $z := Y_R \cdot (X_{QP} - X_{RP})$. We can then easily compute rz, sz, tz and mz, from which the rest of the terms follow homogeneously.

An optimized Montgomery ladder implementation is shown in Figure 3. It is also possible to use the slope $\bar{m}$ of the line connecting Q and R in the ladder state with comparable performance, but this doesn't generalize as well to the Joye ladder. That result is shown in Appendix B.

2.2 Formulas for the Joye ladder

For the Joye double-add ladder, the same outline works, but we are conditionally swapping $(x_P, y_P) \leftrightarrow (x_Q, y_Q)$ instead of $(x_Q, y_Q) \leftrightarrow (x_R, y_R)$. The x-coordinates are easily rearranged to support this by tracking $X_{RP} := X_R - X_P$ and $X_{RQ} := X_R - X_Q$. For y-coordinates, we now need to track both y_P and y_Q. Conveniently, we can use both coordinates to compute $x_S - x_P = -tu$ from Theorem 1. The Joye ladder state is:

$$\begin{aligned} X_{RP} &:= (x_R - x_P) \cdot Z^2 \\ X_{RQ} &:= (x_R - x_Q) \cdot Z^2, \\ M &:= -2m \cdot Z \\ Y_P &:= 2y_P \cdot Z^3 \\ Y_Q &:= 2y_Q \cdot Z^3 \end{aligned}$$

An optimized Joye ladder is shown in Figure 4.

Montgomery ladder. Input: ladder state (X_{QP}, X_{RP}, M, Y_P)

1	$Y_R = Y_P + 2 \cdot M \cdot X_{RP}$	8	$Y'_P = Y_P \cdot F \cdot G$
2	$E = X_{QP} - X_{RP}$	9	$K = H + M'$
3	$F = Y_R \cdot E$	10	$L = K + M'$
4	$G = E^2$	11	$M'' = X'_{RP} - K$
5	$X'_{RP} = X_{RP} \cdot G$	12	$X_{SP} = H \cdot L$
6	$H = Y_R^2$	13	$X_{TP} = {X'_{RP}}^2 + Y'_P$
7	$M' = M \cdot F$	14	$Y''_P = Y'_P \cdot H$

Output: ladder state $(X_{SP}, X_{TP}, M'', Y''_P)$

Notes:
- These formulas use $8\mathbf{M} + 3\mathbf{S} + 7\mathbf{A}$ and two temporary registers for a total of 6.
- An $\mathbf{S} - \mathbf{M}$ tradeoff is available by computing $F = \frac{1}{2} \cdot ((Y_R + E)^2 - G - H)$, yielding $7\mathbf{M} + 4\mathbf{S} + 10.5\mathbf{A}$.
- The multiplications can easily be parallelized over 2 or 3 units. When parallelizing over 2 multiplication units, 4 of the 5 pairs can be made to share an operand.

Figure 3: Improved Montgomery ladder.

2.3 Ladder setup

The initial state of the ladder encodes the points $P = (x_P, y_P)$ and $R = 2P = (x_R, y_R)$. Let (x, y) lie on the elliptic curve $y^2 = x^3 + ax + b$. We need to compute $(x_R - x_P)Z^2, 2y_P Z^3$ and mZ, where $m = (3x_P^2 + a)/(2y_P)$ is the slope of the tangent at P.

Since $x_R + 2x_P = m^2$, we have $(x_R - x_P)Z^2 = (mZ)^2 - 3x_P Z^2$. Setting $Z = 2y_P$, we get

$$\begin{aligned} Z^2 &= 4y_P^2 = 4(x_P^3 + ax_P + b) \\ mZ &= 3x_P^2 + a \\ X_{RP} &= (mZ)^2 - 3x_P Z^2 \\ 2y_P Z^3 &= (Z^2)^2. \end{aligned}$$

Since these formulas depend on y_P^2 rather than y_P, they still work if y_P is not given, which is common for elliptic curve Diffie-Hellman protocols. However, if the elliptic curve is not twist-secure, then the implementation must check that the putative Z^2 is actually square. Otherwise the ladder will still work, but with arithmetic on the curve's quadratic twist. If power analysis is not a concern, or if the twist has no small subgroups, then the check can be implemented in a batch with the final division [Ham12].

The ladder setup can easily be accomplished in 5 registers or fewer, so this routine doesn't increase the memory footprint of either implementation. This includes the check that Z^2 is actually square.

2.4 Finalization

2.4.1 Simple technique

To complete the ladder, we must recover the final x_Q, and optionally y_Q, from the ladder state. In the Montgomery ladder, if the original coordinates (x_P, y_P) are retained, this is easy. We have both $y_P Z^3 = Y_P/2$ and $x_P Z^2 = (M^2 - X_{QP} - X_{RP})/3$, so we can calculate[3]

$$\frac{1}{z} = \frac{y_P \cdot x_P Z^2}{x_P \cdot y_P Z^3}$$

[3]These calculations require trivial modifications if the constants 1/2 and 1/3 are not available.

Joye ladder. Input: ladder state $(X_{RP}, X_{RQ}, Y_P, Y_Q, M)$

1 $Y_R = M \cdot X_{RP} - Y_P$	7 $M' = H - Y'_Q - 2 \cdot X'_{RP}$
2 $G = X_{RQ}^2$	8 $X_{TP} = {X'_{RP}}^2 + Y'_P$
3 $X'_{RP} = X_{RP} \cdot G$	9 $Y''_Q = Y'_Q \cdot H$
4 $Y'_P = Y_P \cdot X_{RQ} \cdot Y_R \cdot G$	10 $Y''_P = Y'_P \cdot H$
5 $Y'_Q = Y_Q \cdot Y_R$	11 $X_{TS} = X_{TP} + Y''_Q$
6 $H = Y_R^2$	12 $Y_S = M' \cdot Y''_Q + Y''_P$

Output: ladder state $(X_{TP}, X_{TS}, Y''_P, Y_S, M')$

Notes:
- These formulas use $9\mathbf{M} + 3\mathbf{S} + 7\mathbf{A}$, and no temporary registers for a total of 5.
- An $\mathbf{S} - \mathbf{M}$ tradeoff is available by computing $X_{RQ} \cdot Y_R = \frac{1}{2} \cdot ((Y_R + X_{RQ})^2 - G - H)$, yielding $8\mathbf{M} + 4\mathbf{S} + 10.5\mathbf{A}$.
- If Z is to be tracked, it should be multiplied by $X_{RQ} \cdot Y_R$.
- The calculation of Y_R can be moved to the end of the round, so that Y_R is a state variable instead of M.
- The multiplications can easily be parallelized over 2 or 3 units. When parallelized over 2 units, if Y_R is moved to the end of the round then 5 of the 6 pairs can be made to share an operand.

Figure 4: Improved Joye ladder.

and recover the final point (x_Q, y_Q). Likewise, if only the original x_P is retained, we can calculate $1/Z^2 = x_P/(x_P Z^2)$. However, this technique does not work if $x_P = 0$, which can happen on certain curves [AT03], so we will propose an improved technique.

The improved technique doesn't work with curves of j-invariant 0 or 1728. The most popular curve with j-invariant 0, NIST's secp256k1, has no points with $xy = 0$, so the simple technique can be used for the Montgomery ladder on that curve.

2.4.2 Improved technique

We want to avoid incompleteness when the starting point has $xy = 0$. We also need an alternative technique for the Joye ladder: the point P in the ladder isn't the base point P_0, but instead on the nth step the ladder has $R = 2^n P_0$. Unless that point has been precomputed, we cannot take advantage of a known point to determine Z. Likewise, low-memory implementations of the Montgomery ladder may wish to discard the base point.

If the curve doesn't have j-invariant 0 (meaning that $a = 0$) or 1728 (meaning that $b = 0$), then we have an alternative technique to recover Z^2, which allows us to recover x_Q but not y_Q. Let $c := y_P - m x_P$ be the y-intercept of the line connecting the $(P, Q, -R)$. Then (x_P, x_Q, x_R) are the roots of

$$y^2 = (mx + c)^2 = x^3 + ax + b,$$

meaning that

$$x^3 - m^2 x^2 + (a - 2mc)x + (b - c^2) = (x - x_P) \cdot (x - x_Q) \cdot (x - x_R).$$

Rearranging, we get

$$\begin{aligned} a &= 2mc + x_P x_Q + x_Q x_R + x_R x_P \\ b &= c^2 - x_P x_Q x_R. \end{aligned}$$

Note that in the ladder state, the values of m, x_i and c are scaled by Z, Z^2 and Z^3 respectively, so these formulas compute $A := aZ^4$ and $B := bZ^6$ instead. This allows us to

calculate $1/Z^2 = Ab/(aB)$, which is enough to calculate x_Q but not y_Q. The division will be $0/0$ if and only if $a = 0, b = 0$ or $Z = 0$.

Alternatively, if y_P is available we can compute $1/Z = aBy_P/(Aby_PZ^3)$. This would fail if $y_P = 0$, but in that case even the starting state of the ladder contains the point at infinity.

Because $Y = 2 \cdot y \cdot Z^3$, the true value of y is in the field if and only if this putative Z^2 is actually square. Its Jacobi symbol can be checked during the inversion at little extra cost using the batching technique in [Ham12]. This means that if power analysis is not a concern, or if the curve is twist-secure, then the initial on-curve check can be deferred until finalization.

With careful use of the identity $x_P + x_Q + x_R = m^2$, the improved finalization step can also be calculated in 5 registers, including the invariant and Jacobi symbol checks. As a result, using our Joye ladder formulas it is possible to calculate an entire x-only variable-base scalar multiplication using only 5 mutable field registers, plus the scalar and the curve constants a and b.

2.4.3 Tracking Z

If y_P is given and we want to recovery y_Q on the Joye ladder, or if $j \in \{0, 1728\}$, then we can instead track Z at a cost of $1\mathbf{M}$ per bit. If y_P is not given and $j \in \{0, 1728\}$, then we can instead track Z^2 at a cost of $1\mathbf{M} + 1\mathbf{S}$ per bit and one extra register[4], or $(Z/y_P, y_P^2)$ at a cost of $1\mathbf{M}$ per bit and 2 extra registers.

2.5 Ladder state invariants

The improved technique's formulas for $A = aZ^4$ and $B = bZ^6$ also serve as a ladder state invariant: we must have

$$A^3b^2 = a^3B^2.$$

This equation can be checked during finalization, or periodically between ladder steps, as a fault attack countermeasure. If Z or Z^2 is tracked, then the stronger conditions $A = aZ^4$ and $B = bZ^6$ can be checked instead.

3 Completeness

3.1 Completeness of the Montgomery and Joye formulas

This section applies principally to curves of prime order $q \geq 5$, since those are the most common use case for short Weierstrass curves. However, it can be adapted to curves of other orders. Recall that the ladder operation computes

$$(P, Q, R) \to (P, Q + R, 2R).$$

The Montgomery and Joye ladders will resolve to $0/0$ if the (possibly untracked) Z-coordinate ever becomes 0. In each iteration, Z is scaled by $2Y_R(X_R - X_Q)$. This is zero if and only if either $Y_R = 0$ (meaning that $2R = 0$), or $X_R = X_Q$ (meaning that $Q = \pm R$). This condition cannot begin with $Q = R$, because then $P = R - Q$ would already be the neutral point and we would already have $Z = 0$. If the curve has odd prime order q, then we also cannot have $2R = 0$.

This leaves the case $R = -Q = P/2$. How soon after the initial state $(P_0, P_0, 2P_0)$ can this occur? In the Montgomery ladder, we would have

$$Q = (\ell q - 1)/2 \cdot P = (\ell q - 1)/2 \cdot P_0 \quad \text{for some odd } \ell,$$

[4] $2\mathbf{M}$ or two extra registers for the Joye ladder, because the 5-register version doesn't calculate the value $X_Q \cdot Y_R$ to multiply into Z.

which cannot occur until $\lceil \log_2 q \rceil - 2$ ladder steps after setup. Likewise, in the Joye ladder, if the condition $R = -Q$ occurs i steps after setup, then

$$2^{i+1} P_0 = R = -Q = -k_{0..i} P_0,$$

where the initial segment $k_{0..i}$ of the scalar satisfies $0 < k_{0..i} < 2^{i+1}$. This implies that $2^{i+2} \geq q$, so again $i \geq \lceil \log_2 q \rceil - 2$. If the scalar is not extended beyond $\lceil \log_2 q \rceil$ bits, then incompleteness is a risk only for the last two ladder steps.

Overall, the ladder and finalization formulas are correct when the ladder never reaches the neutral point, and one of the following finalization techniques is used:

- The simple technique in Section 2.4.1 for the Montgomery ladder on curves with no point of the form $(0, y)$.
- The improved technique in Section 2.4.2 on curves with j-invariant neither 0 nor 1728.
- The Z-tracking technique in Section 2.4.3.

3.2 Avoiding the neutral point

The ladder formulas are still incomplete when they compute a state containing the neutral point. The probability that this happens is negligible if all the following conditions are all met:

- The curve's order q is a large prime.
- The scalar is uniformly random mod q, or at least has sufficiently high min-entropy.
- If the scalar is represented using more bits than the curve order, then the representation's initial segments of length at least $\geq \lceil \log_2 q \rceil - 2$ must also have high min-entropy.
- Either the twist also has prime order, or the calculation begins with an on-curve check.

If power analysis isn't a concern, then reaching the neutral point on the twist may not matter, so long as the result is rejected by a timing-invariant check at the end of the calculation.

In this section, we will show a technique which avoids the neutral point when the curve's order is not divisible by 2 or 3. This technique is generally applicable, and for some ladders (e.g. our Joye ladder) it adds no extra multiplications. However, it does require extra adds and conditional swaps, and it doesn't prevent an attacker from using power analysis to determine whether the neutral point was reached. If q is prime, the avoidance technique doesn't need to be used for the first $\lceil \log_2 q \rceil - 2$ steps of the ladder.

3.3 Notation change: ladder state sums to 0

For both the Joye and Montgomery ladder, the state (P, Q, R) normally satisfies $R = P + Q$. However, the state is typically permuted between ladder steps, and we will further permute it here. To ensure that the ladder state satisfies a consistent invariant without introducing additional cases, we will write it as $(P, Q, \bar{R})$ where $\bar{R} = -R$, so that no matter how the state is permuted, $P + Q + \bar{R} = 0$. This changes the ladder operation to $(P, Q, \bar{R}) \to (P, Q - \bar{R}, 2\bar{R})$.

3.4 Entering and leaving the neutral zone

We call the set of states where $0 \in \{P, Q, \bar{R}\}$ the "neutral zone". If the curve's order is odd[5], then $2\bar{R} \neq 0$. So on such curves, the only way to enter the neutral zone[6] is by the ladder step

$$(-2Q, Q, Q) \to (-2Q, 0, 2Q).$$

On the next step, the Montgomery ladder either stays in this state or exits the neutral zone to $(-2Q, -2Q, 4Q)$. Likewise, the Joye ladder either doubles to $(-4Q, 0, 4Q)$ or exits to $(-2Q, -2Q, 4Q)$.

3.5 The shadow state

We propose to avoid this set of transitions by recognizing that $Q = \bar{R}$ and instead using an equivalent, equally long sequence of transitions that avoids the neutral zone. Specifically, we permute the state $(-2Q, Q, Q)$ to $(Q, -2Q, Q)$ so that the ladder operation moves it to the "shadow" state $(Q, -3Q, 2Q)$. This state is outside the neutral zone if the curve's order is not divisible by 3. It can transition to $(2Q, -4Q, 2Q)$, which can be negated and permuted into the exit state $(-2Q, -2Q, 4Q)$. Or, again through the usual ladder operation, it can remain the same (like the Montgomery ladder) or double (like the Joye ladder).[7] If the ladder should end in the shadow state, then the answer is either the neutral point or $\pm 2Q$, which can also be extracted from that state.

This technique is simplest to implement using our Joye ladder formulas with a state of

$$\begin{aligned} X_{PR} &= (x_P - x_R) \cdot Z^2 \\ X_{QR} &= (x_Q - x_R) \cdot Z^2 \\ Y_P &= 2y_P \cdot Z^3 \\ Y_Q &= 2y_Q \cdot Z^3 \\ Y_R &= 2y_R \cdot Z^3 \end{aligned}$$

and optionally Z. This allows permutations of the Y-coordinates to be done directly. Negating the state requires only negating Z, if it is present, and is free if Z is not present. The only tricky part is how to permute the X-coordinates. The simple case is condswap(P, Q), which is equivalent to

$$\text{condswap}(X_{PR}, X_{QR}); \quad \text{condswap}(Y_P, Y_Q).$$

The other swaps require arithmetic on the X-coordinates, and it is preferable not to perform that conditionally. But we can build these swaps using only unconditional arithmetic and conditional swaps. For example, we can implement condswap(P, R) as

$$\text{swap}(Q, R); \quad \text{condswap}(P, Q); \quad \text{swap}(Q, R).$$

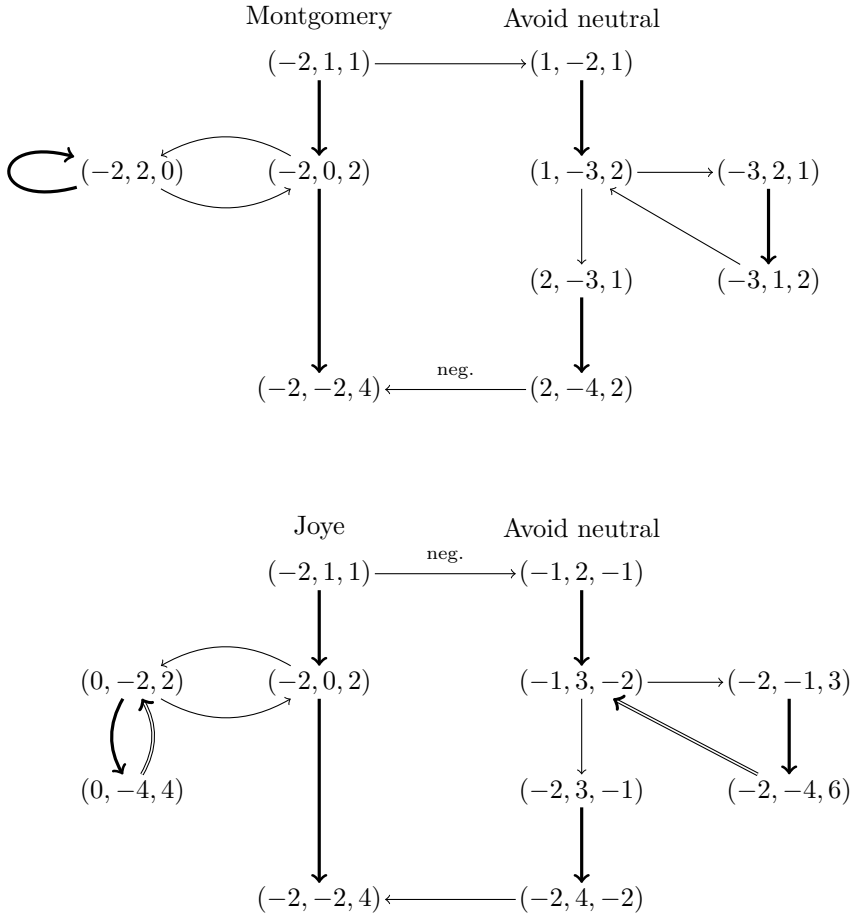
Though they are described as Joye ladder formulas, they could also be used for the Montgomery ladder, except that we would track the x-coordinates X_{QP} and X_{RP}.

A state diagram of the neutral point avoidance technique is shown in Figure 5, and full formulas for the x-only case are given in the supplemental material. These complete formulas do not require any extra registers, at least if a "reverse subtraction" operation $X \leftarrow Y - X$ is available.

[5] A similar avoidance scheme should work to avoid $2\bar{R} = 0$, but it requires a more detailed case analysis. In practice, usually curves with even order are Montgomery curves, and the Montgomery ladder on those curves is complete [BL17].

[6] If the protocol allows the input to be the neutral point, then this must be handled separately.

[7] States proportional to permutations of $(1, 2, -3)$ are the only states outside the neutral zone with either of these properties.



Notes:

- The left sides show the transitions into, within and out of the neutral zone. The right sides show the corresponding transitions using our neutral-point avoidance technique.
- All ladder states are given multiples of Q, with the last point R negated to $\bar{R}$.
- Thick arrows denote the ladder operation $(P, Q, \bar{R}) \longrightarrow (P, Q - R, 2\bar{R})$.
- Thin arrows denote permutations, and with the marking "neg", negations.
- Double arrows mean that one state is a permutation of another, except with all points doubled. So the state transitions are the same from there, but with Q replaced by $2Q$.

Figure 5: Avoiding the neutral point.

4 Side-Channel Protections

We offer some insight into side-channel attacks and mitigations for these new ladder formulas.

Horizontal attacks: Horizontal attacks are power analysis attacks which look for correlations between values calculated or stored between one ladder step and the next. This enables the attacker to determine whether the points have been swapped between rounds or not, which determines the key bit. With our formulas, the attack should be harder on the Montgomery ladder than on the Joye ladder, because the former does not reuse any of its outputs within the ladder step. However, this is likely only a partial mitigation of the horizontal attack. Suitable additional defenses against this attack include scalar blinding or inter-step projective reblinding.

Fault attacks: All values within the ladder are used, and none of the intermediates are expected to be the same once RPA has been mitigated. This should make ineffective-fault attacks more difficult. To defend against other fault attacks, implementations can check the ladder invariants from Section 2.5, either periodically or at the end of the ladder computation. For full defense against fault attacks other countermeasures are required, and are well beyond the scope of this work.

5 Conclusions and Future Work

We have presented new, optimized formulas for the Montgomery and Joye ladders on short Weierstrass curves. Our Montgomery ladder formulas are faster and slightly more side-channel-resistant than our Joye ladder formula, and we hope that future work can improve the Joye formula to match. We have not attempted a computer search for more optimal formulas with this representation or similar representations, which might well be fruitful.

For typical applications such as the NIST and Brainpool curves, our ladders are complete except with negligible probability, and even that negligible probability can be avoided at a reasonable additional cost. However, the avoidance technique would be more useful if it could be simplified.

5.1 Acknowledgments

Thanks to Mark Marson for feedback on drafts of this work, and to everyone who pointed out typos in the preprint.

5.2 Intellectual property disclosure

Some of these techniques may be covered by US and/or international patents.

References

[AT03] Toru Akishita and Tsuyoshi Takagi. Zero-value point attacks on elliptic curve cryptosystem. In Colin Boyd and Wenbo Mao, editors, *ISC 2003*, volume 2851 of *LNCS*, pages 218–233. Springer, Heidelberg, October 2003.

[BL17] Daniel J. Bernstein and Tanja Lange. Montgomery curves and the Montgomery ladder. Cryptology ePrint Archive, Report 2017/293, 2017. http://eprint.iacr.org/2017/293.

[GJM10] Raveen R. Goundar, Marc Joye, and Atsuko Miyaji. Co-z addition formulae and binary ladders on elliptic curves. Cryptology ePrint Archive, Report 2010/309, 2010. http://eprint.iacr.org/2010/309.

[Ham12] Mike Hamburg. Fast and compact elliptic-curve cryptography. Cryptology ePrint Archive, Report 2012/309, 2012. http://eprint.iacr.org/2012/309.

[Ham17] Mike Hamburg. Speeding up elliptic curve scalar multiplication without either precomputation or adaptive coordinates, 2017. https://ches.2017.rump.cr.yp.to/a1933e522beb16591d9dc8e373ad7079.pdf.

[Joy07] Marc Joye. Highly regular right-to-left algorithms for scalar multiplication. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES 2007*, volume 4727 of *LNCS*, pages 135–147. Springer, Heidelberg, September 2007.

[KCK+17] Kwang Ho Kim, Junyop Choe, Song Yun Kim, Namsu Kim, and Sekung Hong. Speeding up elliptic curve scalar multiplication without precomputation. Cryptology ePrint Archive, Report 2017/669, 2017. http://eprint.iacr.org/2017/669.

[Mel07] Nicolas Meloni. New point addition formulae for ECC applications. In *International Workshop on the Arithmetic of Finite Fields*, pages 189–201. Springer, 2007.

[Mon87] Peter L Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of computation*, 48(177):243–264, 1987.

[Riv11] Matthieu Rivain. Fast and regular algorithms for scalar multiplication over elliptic curves. Cryptology ePrint Archive, Report 2011/338, 2011. http://eprint.iacr.org/2011/338.

[SM16] Ruggero Susella and Sofia Montrasio. A compact and exception-free ladder for all short Weierstrass elliptic curves. In *International Conference on Smart Card Research and Advanced Applications*, pages 156–173. Springer, 2016.

[Wal17] Colin D. Walter. The Montgomery and Joye powering ladders are dual. Cryptology ePrint Archive, Report 2017/1081, 2017. https://eprint.iacr.org/2017/1081.

A Proof of ladder formulas

Theorem 1 (Ladder formulas)**.** *Let*

$$P = (x_P, y_P), \qquad Q := (x_Q, y_Q), \qquad R := P + Q := (x_R, y_R)$$

be the state of the Montgomery ladder on an elliptic curve $y^2 = x^3 + ax + b$ defined over a field of characteristic other than 2. The three points $(P, Q, -R)$ lie on a line with slope $m := (y_Q + y_R)/(x_Q - x_R)$. Let

$$P = (x_P, y_P), \qquad S := Q + R = (x_S, y_S), \qquad T := 2R = (x_T, y_T)$$

be the state after a ladder operation, where $(P, S, -T)$ lie on a line of slope m'. Let

$$r := \frac{x_Q - x_R}{2y_R}, \qquad s := (x_R - x_P) \cdot r, \qquad t := \frac{2y_R}{x_Q - x_R}, \qquad u := \frac{2y_Q}{x_Q - x_R}.$$

Then

$$\begin{aligned} x_S - x_P &= -t \cdot u = t \cdot (t - 2m) \\ x_T - x_P &= s^2 - 2y_P \cdot r \\ m' &= t - m - s. \end{aligned}$$

Proof. The three points $P, Q, -P - Q$ lie on the intersection

$$y^2 = (mx + c)^2 = x^3 + ax + b.$$

for some c. The solutions to this equation are roots of a polynomial of the form $x^3 - m^2x^2 + O(x)$, so that

$$x_P + x_Q + x_R = m^2.$$

Let $\bar{m} := (y_Q - y_R)/(x_Q - x_R)$ be the slope of the line connecting Q, R and $-(Q + R) = (x_S, -y_S)$. We have

$$\bar{m}^2 = x_Q + x_R + x_S; \qquad \bar{m} - m = -t; \qquad \bar{m} + m = u.$$

Therefore

$$\begin{aligned} x_S - x_P &= \bar{m}^2 - x_Q - x_R - x_P \\ &= \bar{m}^2 - m^2 \\ &= (\bar{m} + m) \cdot (\bar{m} - m) \\ &= -tu. \end{aligned}$$

We also have $-u = -(\bar{m} + m) = (m - \bar{m}) - 2m = t - 2m$. Next, we have

$$\begin{aligned} y_S - y_P &= -(-y_S - y_R) + (-y_R - y_P) \\ &= -\bar{m} \cdot (x_S - x_R) + m \cdot (x_R - x_P) \\ &= -\bar{m} \cdot (x_S - x_P) + (\bar{m} + m) \cdot (x_R - x_P) \\ &= -\bar{m} \cdot (x_S - x_P) + u \cdot (x_R - x_P) \end{aligned}$$

whence the output slope from P to $Q + R$ is

$$\begin{aligned} m_{\text{out}} := \frac{y_S - y_P}{x_S - x_P} &= -\bar{m} + \frac{u(x_R - x_P)}{-tu} \\ &= -\bar{m} - s \\ &= t - m - s \end{aligned}$$

Finally, let's calculate $x_T - x_P$. We have $m_{\text{out}} = -\bar{m} - s$. In calculating $x_T - x_P = m_{\text{out}}^2 - 2x_P - x_S$, we may expand

$$m_{\text{out}}^2 = \bar{m}^2 + 2\bar{m}s + s^2 = x_Q + x_R + x_S + 2\bar{m}s + s^2$$

wherein

$$2\bar{m}s = 2\frac{y_Q - y_R}{x_Q - x_R}\frac{(x_R - x_P)(x_Q - x_R)}{2y_R} = \frac{(y_Q - y_R)(x_R - x_P)}{y_R}.$$

We also have the cross product identity

$$\begin{aligned} (x_Q - x_P)y_R + (x_R - x_P)y_Q &= (x_Q - x_P)(-y_P - m(x_R - x_P)) \\ &\quad +(x_R - x_P)(y_P + m(x_Q - x_P)) \\ &= (x_R - x_Q)y_P. \end{aligned}$$

Combining these,

$$\begin{aligned}
x_T - x_P &= m_{\text{out}}^2 - 2x_P - x_S \\
&= x_Q + x_R + x_S + 2\bar{m}r + s^2 - 2x_P - x_S \\
&= (x_Q - x_P) + (x_R - x_P) + \frac{(y_Q - y_R)(x_R - x_P)}{y_R} + s^2 \\
&= \frac{(x_Q - x_P)y_R + (x_R - x_P)y_R + (y_Q - y_R)(x_R - x_P)}{y_R} + s^2 \\
&= \frac{(x_Q - x_P)y_R + (x_R - x_P)y_Q}{y_R} + s^2 \\
&= \frac{(x_R - x_Q)y_P}{y_R} + s^2 \\
&= s^2 - 2 \cdot y_P \cdot r
\end{aligned}$$

This completes the proof. □

B The (Y_Q, Y_R) ladder

It is also possible to perform a Montgomery ladder whose state incorporates Y_Q and Y_R, instead of Y_P. The ladder state comprises

$$\begin{aligned}
X_{QP} &:= (x_Q - x_P) \cdot Z^2 \\
X_{RP} &:= (x_R - x_P) \cdot Z^2 \\
G &:= (x_R - x_Q)^2 \cdot Z^4 \\
Y_Q &:= 2y_Q \cdot Z^3 \\
Y_R &:= 2y_R \cdot Z^3
\end{aligned}$$

The ladder requires $8\mathbf{M} + 3\mathbf{S} + 7\mathbf{A}$ per bit, and is shown in Figure 6.

Montgomery ladder. Input: ladder state $(X_{QP}, X_{RP}, G, Y_Q, Y_R)$

1. $X'_{QP} = X_{QP} \cdot G$
2. $X'_{RP} = X_{RP} \cdot G$
3. $L = Y_Q \cdot Y_R$
4. $H = Y_R^2$
5. $J = X_{RP} - L$
6. $M = J + X_{RP} - H$
7. $X_{SP} = H \cdot L$
8. $K = J^2$
9. $X_{TP} = X'_{RP} \cdot J + X'_{QP} \cdot H$
10. $X_{TS} = X_{TP} - X_{SP}$
11. $Y_S = (X_{TS} - K) \cdot H$
12. $Y_T = M \cdot X_{TS} + Y_S$
13. $G' = X_{TS}^2$

Output: ladder state $(X_{SP}, X_{TP}, G', Y_S, Y_T)$

Notes:

- These formulas use $8\mathbf{M} + 3\mathbf{S} + 7\mathbf{A}$ and 6 field registers.
- The Z value is multiplied by $X_{TS} \cdot Y_R$ in each iteration.
- The formulas can be parallelized over two, three or four multiplication units.
- When parallelized over 2 multiplication units, they can be arranged so that all pairs of products share an operand. These multiplications can in turn be paired up to run across 4 multiplication units.

Figure 6: (Y_Q, Y_R) Montgomery ladder.

The main value of this ladder seems to be parallelization. For serial use cases, while the performance of the Montgomery ladder is the same, the outputs are reused within the round, which increases the risk of a horizontal attack.

The finalization equations for this version of the ladder state are similar to those in Section 2.4, since the value of M can be remembered as a round output.