Optimizing double-base elliptic-curve single-scalar multiplication

Daniel J. Bernstein[1], Peter Birkner[2], Tanja Lange[2], and Christiane Peters[2]

[1] Department of Mathematics, Statistics, and Computer Science (M/C 249)
University of Illinois at Chicago, Chicago, IL 60607–7045, USA
`djb@cr.yp.to`
[2] Department of Mathematics and Computer Science
Technische Universiteit Eindhoven, P.O. Box 513, 5600 MB Eindhoven, Netherlands
`p.birkner@tue.nl, tanja@hyperelliptic.org, c.p.peters@tue.nl`

Abstract. This paper analyzes the best speeds that can be obtained for single-scalar multiplication with variable base point by combining a huge range of options:

- many choices of coordinate systems and formulas for individual group operations, including new formulas for tripling on Edwards curves;
- double-base chains with many different doubling/tripling ratios, including standard base-2 chains as an extreme case;
- many precomputation strategies, going beyond Dimitrov, Imbert, Mishra (Asiacrypt 2005) and Doche and Imbert (Indocrypt 2006).

The analysis takes account of speedups such as $\mathbf{S}-\mathbf{M}$ tradeoffs and includes recent advances such as inverted Edwards coordinates.

The main conclusions are as follows. Optimized precomputations and triplings save time for single-scalar multiplication in Jacobian coordinates, Hessian curves, and tripling-oriented Doche/Icart/Kohel curves. However, even faster single-scalar multiplication is possible in Jacobi intersections, Edwards curves, extended Jacobi-quartic coordinates, and inverted Edwards coordinates, thanks to extremely fast doublings and additions; there is no evidence that double-base chains are worthwhile for the fastest curves. Inverted Edwards coordinates are the speed leader.

Key words: Edwards curves, double-base number systems, double-base chains, addition chains, scalar multiplication, tripling, quintupling

1 Introduction

Double-base number systems have been suggested as a way to speed up scalar multiplication on elliptic curves. The idea is to expand a positive integer n as

---

[*] Permanent ID of this document: `d721c86c47e3b56834ded945c814b5e0`. Date of this document: 2007.10.28. This work has been supported in part by the National Science Foundation under grant ITR–0716498. This work has been supported in part by the European Commission through the IST Programme under Contract IST–2002–507932 ECRYPT.

a sum of very few terms $c_i 2^{a_i} 3^{b_i}$ with $c_i = 1$ or $c_i = -1$, and thus to express a scalar multiple nP as a sum of very few points $c_i 2^{a_i} 3^{b_i} P$. Unfortunately, the time to add these points is only one facet of the time to compute nP; computing the points in the first place requires many doublings and triplings. Minimizing the number of additions is minimizing the wrong cost measure.

At Asiacrypt 2005, Dimitrov, Imbert, and Mishra [11] introduced double-base chains $\sum c_i 2^{a_i} 3^{b_i}$, where again $c_i = 1$ or $-c_i = 1$, with the new restrictions $a_1 \geq a_2 \geq a_3 \geq \cdots$ and $b_1 \geq b_2 \geq b_3 \geq \cdots$ allowing a Horner-like evaluation of nP with only a_1 doublings and only b_1 triplings. But the new restrictions introduced by double-base chains substantially increase the number of additions.

At Indocrypt 2006, Doche and Imbert [13] improved double-base chains by introducing an analogue of signed-sliding-window methods, keeping the restrictions $a_1 \geq a_2 \geq a_3 \geq \cdots$ and $b_1 \geq b_2 \geq b_3 \geq \cdots$ but allowing c_i and $-c_i$ to be chosen from a coefficient set S larger than $\{1\}$, leading to shorter chains and thus fewer additions. Doche and Imbert studied in detail the sets $\{1\}$, $\{1, 2, 3, 4, 9\}$, $\{1, 2, \ldots, 2^4, 3, \ldots, 3^4\}$, $\{1, 5, 7\}$, $\{1, 5, 7, 11, 13, 17, 19, 23, 25\}$. For each set they counted (experimentally) the number of additions, doublings, and triplings in their chains, compared these to the number of additions and doublings in the standard single-base sliding-window methods, and concluded that these new double-base chains save time in scalar multiplication. However, there are several reasons to question this conclusion:

- The comparison ignores the cost of precomputing all the cP for $c \in S$. These costs are generally lower for single-base chains, and are incurred for every scalar multiplication (unless P is reused, in which case there are much faster scalar-multiplication methods).
- The comparison relies on obsolete addition formulas. For example, [13] uses mixed-addition formulas that take $8\mathbf{M} + 3\mathbf{S}$: i.e., 8 field multiplications and 3 squarings. Faster formulas are known, taking only $7\mathbf{M} + 4\mathbf{S}$; this speedup has a larger benefit for single-base chains than for double-base chains.
- The comparison relies on obsolete curve shapes. For example, [13] uses doubling formulas that take $4\mathbf{M} + 6\mathbf{S}$, but the standard choice $a_4 = -3$ improves Jacobian-coordinate doubling to $3\mathbf{M} + 5\mathbf{S}$, again making single-base chains more attractive. Recent work has produced extremely fast doubling and addition formulas for several non-Jacobian curve shapes.

In this paper we carry out a much more comprehensive comparison of elliptic-curve scalar-multiplication methods. We analyze a much wider variety of coordinate systems, including the most recent innovations in curve shapes and the most recent speedups in addition formulas; see Section 3. In particular, we include Edwards curves in our comparison; in Section 2 we introduce new fast tripling formulas for Edwards curves, and in the appendix we introduce quintupling formulas. Our graphs include the obsolete addition formulas for Jacobian coordinates ("Std-Jac" and "Std-Jac-3") to show how striking the advantage of better group operations is. We account for the cost of precomputations, and we account for the difference in speeds between addition, readdition, and mixed addition. We include more choices of chain parameters, and in particular identify better choices of S for double-base chains. We cover additional exponent lengths of interest in cryptographic applications.

We find, as in [13], that double-base chains achieve significant improvements for curves in Jacobian coordinates and for tripling-oriented Doche/Icart/Kohel curves; computing scalar multiples with the $\{2,3\}$-double-base chains is faster than with the best known single-base chains. For integers of bit-length ℓ about 0.22ℓ triplings and 0.65ℓ doublings are optimal for curves in Jacobian coordinates; for the Doche/Icart/Kohel curves the optimum is about 0.29ℓ triplings and 0.54ℓ doublings. For Hessian curves we find similar results; the optimum is about 0.25ℓ triplings and 0.6ℓ doublings.

On the other hand, for Edwards curves it turns out that the optimum for base-$\{2,3\}$ chains uses very few triplings. This makes the usefulness of double-base chains for Edwards curves questionable. The same result holds for $\{2,5\}$-double-base chains. Based on our results we recommend traditional single-base chains for implementors using Edwards curves. Similar conclusions apply to Jacobi intersections, extended Jacobi-quartic coordinates, and inverted Edwards coordinates.

In the competition between coordinate systems, inverted Edwards coordinates are the current leader, followed closely by extended Jacobi-quartic coordinates and standard Edwards coordinates, and then by Jacobi intersections. Jacobian coordinates with $a_4 = -3$, despite double-base chains and all the other speedups we consider, are slower than Jacobi intersections. Tripling-oriented Doche/Icart/Kohel curves are competitive with Jacobian coordinates — but not nearly as impressive as they seemed in [13]. For the full comparison see Section 5.

2 Edwards curves

Edwards [16] introduced a new form for elliptic curves over fields of characteristic different from 2 and showed that — after an appropriate field extension — every elliptic curve can be transformed to this normal form. We now briefly review arithmetic on Edwards curves and then develop new tripling formulas. Hisil, Carter, and Dawson independently developed essentially the same tripling formulas; see [18].

Background on Edwards curves. We present Edwards curves in the slightly generalized version due to Bernstein and Lange [5]. An *elliptic curve in Edwards form*, or simply *Edwards curve*, over a field k is given by an equation

$$x^2 + y^2 = 1 + dx^2y^2, \qquad \text{where } d \in k \setminus \{0,1\}.$$

Two points (x_1, y_1) and (x_2, y_2) are added according to the *Edwards addition law*

$$(x_1, y_1), (x_2, y_2) \mapsto \left(\frac{x_1y_2 + y_1x_2}{1 + dx_1x_2y_1y_2}, \frac{y_1y_2 - x_1x_2}{1 - dx_1x_2y_1y_2} \right). \tag{2.1}$$

The neutral element of this addition is $(0,1)$. The inverse of any point (x_1, y_1) on E is $(-x_1, y_1)$. Doubling can be performed with exactly the same formula as addition. If d is not a square in k the addition law is complete, i.e., it is defined for all pairs of input points on the Edwards curve over k and the result matches the sum of the input points on any birationally equivalent elliptic curve.

Bernstein and Lange [5] study Edwards curves for cryptographic applications and give efficient explicit formulas for the group operations. To avoid inversions they work with the homogenized equation in which a point $(X_1 : Y_1 : Z_1)$ corresponds to the affine point $(X_1/Z_1, Y_1/Z_1)$ on the Edwards curve. Their newer paper [6] proposes using $(X_1 : Y_1 : Z_1)$ to represent $(Z_1/X_1, Z_1/Y_1)$. These *inverted Edwards coordinates* save $1\mathbf{M}$ in addition. For contrast we refer to the former as *standard Edwards coordinates.*

Doubling on Edwards curves. If both inputs are known to be equal the result of the addition can be obtained using fewer field operations. We briefly describe how [5] derived the special formulas for doubling from the general addition law (2.1). The same approach will help in tripling.

Since (x_1, y_1) is on the Edwards curve one can substitute the coefficient d by $(x_1^2 + y_1^2 - 1)/(x_1^2 y_1^2)$ in the following sum:

$$(x_1, y_1), (x_1, y_1) \mapsto \left(\frac{2x_1y_1}{1 + dx_1^2y_1^2}, \frac{y_1^2 - x_1^2}{1 - dx_1^2y_1^2} \right) = \left(\frac{2x_1y_1}{x_1^2 + y_1^2}, \frac{y_1^2 - x_1^2}{2 - (x_1^2 + y_1^2)} \right).$$

This substitution reduces the degree of the denominator from 4 to 2 which is reflected in faster doublings.

Tripling on Edwards curves. One can triple a point by first doubling it and then adding the result to itself. By applying the curve equation as in doubling we obtain

$$3(x_1, y_1) = \left(\frac{((x_1^2 + y_1^2)^2 - (2y_1)^2)}{4(x_1^2 - 1)x_1^2 - (x_1^2 - y_1^2)^2} x_1, \frac{((x_1^2 + y_1^2)^2 - (2x_1)^2)}{-4(y_1^2 - 1)y_1^2 + (x_1^2 - y_1^2)^2} y_1 \right).$$

We present two sets of formulas to do this operation in standard Edwards coordinates. The first one costs $9\mathbf{M} + 4\mathbf{S}$ while the second needs $7\mathbf{M} + 7\mathbf{S}$. If the $\mathbf{S}/\mathbf{M}$ ratio is very small, specifically below $2/3$, then the second set is better while for larger ratios the first one is to be preferred.

The explicit formulas were verified to produce the 3-fold of the input point $(X_1 : Y_1 : Z_1)$ by symbolically computing $3(X_1 : Y_1 : Z_1)$ using the addition and doubling formulas from [5] and comparing it with $(X_3 : Y_3 : Z_3)$.

Here are our $9\mathbf{M} + 4\mathbf{S}$ formulas for tripling:

$$\begin{gathered} A = X_1^2;\ B = Y_1^2;\ C = (2Z_1)^2;\ D = A + B;\ E = D^2;\ F = 2D \cdot (A - B); \\ G = E - B \cdot C;\ H = E - A \cdot C;\ I = F + H;\ J = F - G; \\ X_3 = G \cdot J \cdot X_1;\ Y_3 = H \cdot I \cdot Y_1;\ Z_3 = I \cdot J \cdot Z_1. \end{gathered}$$

Here are our $7\mathbf{M} + 7\mathbf{S}$ formulas for tripling:

$$\begin{gathered} A = X_1^2;\ B = Y_1^2;\ C = Z_1^2;\ D = A + B;\ E = D^2;\ F = 2D \cdot (A - B); \\ K = 4C;\ L = E - B \cdot K;\ M = E - A \cdot K;\ N = F + M;\ O = N^2;\ P = F - L; \\ X_3 = 2L \cdot P \cdot X_1;\ Y_3 = M \cdot ((N + Y_1)^2 - O - B);\ Z_3 = P \cdot ((N + Z_1)^2 - O - C). \end{gathered}$$

Appendix A contains formulas for quintupling on Edwards curves.

3 Fast Addition on Elliptic Curves

There is a vast literature on elliptic curves. See [14, 7, 17] for overviews of efficient group operations on elliptic curves, and [5, Section 6] for an analysis of scalar-multiplication performance without triplings.

Those overviews are not a satisfactory starting point for our analysis, because they do not include the most recent improvements in curve shapes and in addition formulas. Fortunately, all of the latest improvements have been collected into the Bernstein/Lange "Explicit-Formulas Database" (EFD) [4], with Magma scripts verifying the correctness of the formulas. For example, this database now includes our tripling formulas, the tripling formulas from [6] (modeled after ours) for inverted Edwards coordinates, and the formulas from [18] for other systems.

Counting operations. In Section 5 we assume $\mathbf{S} = 0.8\mathbf{M}$, but in this section we record the costs separately. We ignore costs of the cheaper field operations such as field additions, field subtractions, and field doublings.

We also ignore the costs of multiplications by curve parameters (for example, d in Edwards form). We assume that curves are sensibly selected with small parameters so that these multiplications are easy.

Jacobian coordinates. Let k be a field of characteristic at least 5. Every elliptic curve over k can then be written in short Weierstrass form $E : y^2 = x^3 + a_4x + a_6,\ a_4, a_6 \in k$, where $f(x) = x^3 + a_4x + a_6$ is squarefree. The set $E(k)$ of k-rational points of E is the set of tuples (x_1, y_1) satisfying the equation together with a point P_∞ at infinity.

The most popular representation of an affine point $(x_1, y_1) \in E(k)$ is as *Jacobian coordinates* $(X_1 : Y_1 : Z_1)$ satisfying $Y_1^2 = X_1^3 + a_4X_1Z_1^2 + a_6Z_1^6$ and $(x_1, y_1) = (X_1/Z_1^2, Y_1/Z_1^3)$. An *addition* of generic points $(X_1 : Y_1 : Z_1)$ and $(X_2 : Y_2 : Z_2)$ in Jacobian coordinates costs $11\mathbf{M} + 5\mathbf{S}$. A *readdition*—i.e., an addition where $(X_2 : Y_2 : Z_2)$ has been added before—costs $10\mathbf{M} + 4\mathbf{S}$, because Z_2^2 and Z_2^3 can be cached and reused. A *mixed addition*—i.e., an addition where Z_2 is known to be 1—costs $7\mathbf{M} + 4\mathbf{S}$. A *doubling*—i.e., an addition where $(X_1 : Y_1 : Z_1)$ and $(X_2 : Y_2 : Z_2)$ are known to be equal—costs $1\mathbf{M} + 8\mathbf{S}$. A *tripling* costs $5\mathbf{M} + 10\mathbf{S}$.

If $a_4 = -3$ then the cost for doubling changes to $3\mathbf{M} + 5\mathbf{S}$ and that for tripling to $7\mathbf{M} + 7\mathbf{S}$. Not every curve can be transformed to allow $a_4 = -3$ but important examples such as the NIST curves [1] make this choice. We refer to this case as Jacobian-3.

Most of the literature presents slower formulas producing the same output, and correspondingly reports higher costs for arithmetic in Jacobian coordinates. See, for example, [1, Section A.10.4] and the aforementioned overviews. We include the slower formulas in our experiments to simplify the comparison of our results to previous results in [13] and [11] and to emphasize the importance of using faster formulas. We refer to the slower formulas as Std-Jac and Std-Jac-3.

More coordinate systems. Several other representations of elliptic curves have attracted attention because they offer faster group operations or extra features such as unified addition formulas that also work for doublings. Some of these representations can be reached through isomorphic transformation for any curve in Weierstrass form while others require, for example, a point of order 4. Our analysis includes all of the curve shapes listed in the following table:

Curve shape	ADD	reADD	mADD	DBL	TRI
3DIK	11**M** + 6**S**	10**M** + 6**S**	7**M** + 4**S**	2**M** + 7**S**	6**M** + 6**S**
Edwards	10**M** + 1**S**	10**M** + 1**S**	9**M** + 1**S**	3**M** + 4**S**	9**M** + 4**S**
ExtJQuartic	8**M** + 3**S**	8**M** + 3**S**	7**M** + 3**S**	3**M** + 4**S**	4**M** + 11**S**
Hessian	12**M** + 0**S**	12**M** + 0**S**	10**M** + 0**S**	7**M** + 1**S**	8**M** + 6**S**
InvEdwards	9**M** + 1**S**	9**M** + 1**S**	8**M** + 1**S**	3**M** + 4**S**	9**M** + 4**S**
JacIntersect	13**M** + 2**S**	13**M** + 2**S**	11**M** + 2**S**	3**M** + 4**S**	4**M** + 10**S**
Jacobian	11**M** + 5**S**	10**M** + 4**S**	7**M** + 4**S**	1**M** + 8**S**	5**M** + 10**S**
Jacobian-3	11**M** + 5**S**	10**M** + 4**S**	7**M** + 4**S**	3**M** + 5**S**	7**M** + 7**S**
Std-Jac	12**M** + 4**S**	11**M** + 3**S**	8**M** + 3**S**	3**M** + 6**S**	9**M** + 6**S**
Std-Jac-3	12**M** + 4**S**	11**M** + 3**S**	8**M** + 3**S**	4**M** + 4**S**	9**M** + 6**S**

The speeds listed here, and the speeds used in our analysis, are the current speeds in EFD.

"ExtJQuartic" and "Hessian" and "JacIntersect" refer to the latest addition formulas for Jacobi quartics $Y^2 = X^4 + 2aX^2Z^2 + Z^4$, Hessian curves $X^3 + Y^3 + Z^3 = 3dXYZ$, and Jacobi intersections $S^2 + C^2 = T^2, aS^2 + D^2 = T^2$. EFD takes account of the improvements in [15] and [18].

"3DIK" is an abbreviation for "tripling-oriented Doche/Icart/Kohel curves," the curves $Y^2 = X^3 + a(X + Z^2)^2 Z^2$ introduced last year in [12]. (The same paper also introduces doubling-oriented curves that do not have fast additions or triplings and that are omitted from our comparison.) We note that [12] states incorrect formulas for doubling. The corrected and faster formulas are:

$$\begin{aligned}
&B = X_1^2;\ C = 2a \cdot Z_1^2 \cdot (X_1 + Z_1^2);\ D = 3(B + C);\ E = Y_1^2;\ F = E^2;\\
&Z_3 = (Y_1 + Z_1)^2 - E - Z_1^2;\ G = 2((X_1 + E)^2 - B - F);\\
&X_3 = D^2 - 3a \cdot Z_3^2 - 2G;\ Y_3 = D \cdot (G - X_3) - 8F;
\end{aligned}$$

which are now also included in the EFD.

4 Background: Double-Base Chains for Single-Scalar Multiplication

This section reviews the previous state of the art in double-base chains for computing nP given P.

The non-windowing case. The "base-2" equation

$$\begin{aligned}
&314159P\\
&= 2(2(2(2(2(2(2(2(2(2(2(2(2(2(2(2(2(P))+P))-P)))+P)+P))-P)))+P)+P))))-P
\end{aligned}$$

can be viewed as an algorithm to compute $314159P$, starting from P, with a chain of 18 doublings and 8 additions of P; here we count subtractions as additions. One can express this chain more concisely — with an implicit application of Horner's rule — as

$$314159P = 2^{18}P + 2^{16}P - 2^{14}P + 2^{11}P + 2^{10}P - 2^{8}P + 2^{5}P + 2^{4}P - 2^{0}P.$$

The slightly more complicated “double-base-2-and-3” equation

$$\begin{aligned}314159P &= 2^{15}3^{2}P + 2^{11}3^{2}P + 2^{8}3^{1}P + 2^{4}3^{1}P - 2^{0}3^{0}P\\ &= 3(2(2(2(2(2(2(2(2(3(2(2(2(2(2(2(P)))) + P)))) + P)))) + P))))) - P\end{aligned}$$

can be viewed as a better algorithm to compute $314159P$, starting from P, with a chain of 2 triplings, 15 doublings, and 4 additions of P. If 1 tripling has the same cost as 1 doubling and 1 addition then this chain has the same cost as 17 doublings and 6 additions which is fewer operations than the 18 doublings and 8 additions of P needed in the base-2 expansion.

One can object to this comparison by pointing out that adding mP for $m > 1$ is more expensive than adding P — typically P is provided in affine form, allowing a mixed addition of P, while mP requires a more expensive non-mixed addition — so a tripling is more expensive than a doubling and an addition of P. But this objection is amply answered by dedicated tripling formulas that are *less* expensive than a doubling and an addition. See Sections 2 and 3.

Double-base chains were introduced by Dimitrov, Imbert, and Mishra in a paper [11] at Asiacrypt 2005. There were several previous “double-base number system” papers expanding nP in various ways as $\sum c_i 2^{a_i} 3^{b_i} P$ with $c_i \in \{-1, 1\}$; the critical advance in [11] was to require $a_1 \geq a_2 \geq a_3 \geq \cdots$ and $b_1 \geq b_2 \geq b_3 \geq \cdots$, allowing a straightforward chain of doublings and triplings without the expensive backtracking that plagued previous papers.

Issues in comparing single bases to double bases. One can object that the benefit of fast double-base chains is outweighed by the cost of finding those chains. Perhaps this objection will be answered someday by an optimized algorithm that finds a double-base chain in less time than is saved by applying that chain. We rely on a simpler answer: we focus on cryptographic applications in which the same n is used many times (as in [10, Section 3]), allowing the chain for n to be constructed just once and then reused. Our current software has not been heavily optimized but takes under a millisecond to compute an expansion of a cryptographic-size integer n.

A more troubling objection is that the simple base-2 chains described above were obsolete before the advent of double-base chains. Typical speed-oriented elliptic-curve software instead uses “sliding window” base-2 chains that use marginally more temporary storage but considerably fewer additions — see below. Even if double-base chains are faster than obsolete base-2 chains, there is no reason to believe that they are faster than state-of-the-art sliding-window base-2 chains. This objection is partly answered by an analogous improvement to double-base chains — see below — but the literature does not contain a careful comparison of optimized double-base chains to optimized single-base chains.

The sliding-windows case. The "sliding-windows base-2" equation

$$\begin{aligned}314159P &= 2^{16}5P - 2^{11}7P + 2^{8}3P + 2^{4}3P - 2^{0}P\\ &= 2(2(2(2(2(2(2(2(2(2(2(2(2(2(2(2(5P))))) - 7P))) + 3P)))) + 3P)))) - P\end{aligned}$$

can be viewed as an algorithm to compute $314159P$ starting from $\{P, 3P, 5P, 7P\}$ with a chain of 16 doublings and 4 additions. It can therefore be viewed as an algorithm to compute $314159P$, starting from P, with 17 doublings and 7 additions; this operation count includes the obvious chain of 1 doubling and 3 additions to produce $2P, 3P, 5P, 7P$ from P.

The idea of starting with $\{P, 2P, 3P, 4P, \ldots, (2^w - 1)P\}$ ("fixed length-w windows") was introduced by Brauer long ago in [9]. By optimizing the choice of w as a function of the bitlength ℓ, Brauer showed that one can compute nP for an ℓ-bit integer n using $\approx \ell$ doublings and at most $\approx \ell/\lg \ell$ additions (even without subtractions). The idea to start with $\{P, 2P, 3P, 5P, 7P, \ldots, (2^w - 1)P\}$ ("sliding length-w windows") was introduced by Thurber in [22], saving some additions. For comparison, the simple base-2 chains considered earlier use $\approx \ell$ doublings and $\approx \ell/3$ additions (on average; as many as $\ell/2$ in the worst case). The benefit of windows increases slowly with ℓ.

Doche and Imbert, in their paper [13] at Indocrypt 2006, introduced an analogous improvement to double-base chains. Example: The "sliding-windows double-base-2-and-3" equation

$$\begin{aligned}314159P &= 2^{12}3^{3}3P - 2^{7}3^{3}5P - 2^{4}3^{1}7P - 2^{0}3^{0}P\\ &= 3(2(2(2(2(3(3(2(2(2(2(2(2(2(2(3P))))) - 5P))))) - 7P))))) - P\end{aligned}$$

can be viewed as an algorithm to compute $314159P$ starting from $\{P, 3P, 5P, 7P\}$ with a chain of 3 triplings, 12 doublings, and 3 additions. It can therefore be viewed as an algorithm to compute $314159P$, starting from P, with 3 triplings, 13 doublings, and 6 additions. (The set $\{P, 3P, 5P, 7P\}$ was not considered in [13]; we use this example to emphasize the analogy between single-base chains and double-base chains.)

Doche and Imbert state an algorithm to compute double-base chains for arbitrary coefficient sets S containing 1. In their experiments they focus on sets of the form $\{1, 2, 3, 2^2, 3^2, \ldots, 2^k, 3^k\}$ or sets of odd integers co-prime to 3. In this paper we study several coefficient sets including all sets considered in [13] and additional sets such as $\{P, 2P, 3P, 5P, 7P\}$.

Computing a chain. Finding the chain $314159 = 2^{18} + 2^{16} - 2^{14} + 2^{11} + 2^{10} - 2^{8} + 2^{5} + 2^{4} - 2^{0}$ is a simple matter of finding the closest power of 2 to 314159, namely $2^{18} = 262144$; then finding the closest power of 2 to the difference $|314159 - 262144| = 52015$, namely $2^{16} = 65536$; and so on.

Similarly, by inspecting the first few bits of a nonzero integer n one can easily see which of the integers

$$\begin{array}{llllll} \pm 1, & \pm 2, & \pm 2^2, & \pm 2^3, & \pm 2^4, & \dots \\ \pm 3, & \pm 2\cdot 3, & \pm 2^2 3, & \pm 2^3 3, & \pm 2^4 3, & \dots \\ \pm 5, & \pm 2\cdot 5, & \pm 2^2 5, & \pm 2^3 5, & \pm 2^4 5, & \dots \\ \pm 7, & \pm 2\cdot 7, & \pm 2^2 7, & \pm 2^3 7, & \pm 2^4 7, & \dots \end{array}$$

is closest to n. By subtracting that integer from n and repeating the same process one expands n into Thurber's base-2 sliding-window chain $\sum_i c_i 2^{a_i}$ with $\pm c_i \in \{1,3,5,7\}$ and $a_1 > a_2 > a_3 > \cdots$. For example, $2^{16}\cdot 5 = 327680$ is closest to 314159; $-2^{11}\cdot 7 = -14336$ is closest to $314159 - 327680 = -13521$; continuing in the same way one finds the chain $314159 = 2^{16}5P - 2^{11}7P + 2^8 3P + 2^4 3P - 2^0 P$ shown above. Similar comments apply to sets other than $\{1,3,5,7\}$.

Dimitrov, Imbert, and Mishra in [11, Section 3] proposed a similar, although slower, algorithm to find double-base chains with $c_i \in \{-1,1\}$; Doche and Imbert in [13, Section 3.2] generalized the algorithm to allow a wider range of c_i. For example, given n and the set $\{1,3,5,7\}$, the Doche-Imbert algorithm finds the product $c_1 2^{a_1} 3^{b_1}$ closest to n, with $\pm c_1 \in \{1,3,5,7\}$, subject to limits on a_1 and b_1; it then finds the product $c_2 2^{a_2} 3^{b_2}$ closest to $n - c_1 2^{a_1} 3^{b_1}$, with $\pm c_2 \in \{1,3,5,7\}$, subject to the chain conditions $a_1 \ge a_2$ and $b_1 \ge b_2$; continuing in this way it expands n as $\sum_i c_i 2^{a_i} 3^{b_i}$ with $\pm c_i \in \{1,3,5,7\}$, $a_1 \ge a_2 \ge \cdots$, and $b_1 \ge b_2 \ge \cdots$.

(The algorithm statements in [11] and [13] are ambiguous on the occasions that n is equally close to two or more products $c2^a3^b$. Which (c,a,b) is chosen? In our new experiments, when several $c2^a3^b$ are equally close to n, we choose the first (c,b,a) in lexicographic order: we prioritize a small c, then a small b, then a small a.)

The worst-case and average-case chain lengths produced by this double-base algorithm are difficult to analyze mathematically. However, the average chain length for all n's can be estimated with high confidence as the average chain length seen for a large number of n's. Dimitrov, Imbert, and Mishra used 10000 integers n for each of their data points; Doche and Imbert used 1000; our new experiments use 10000. We also plan to compute variances but have not yet done so.

5 New results

This section describes the experiments that we carried out and the multiplication counts that we achieved. The results of the experiments are presented as a table and a series of graphs.

Parameter space. Our experiments included several bit sizes ℓ, namely 160, 200, 256, 300, 400, and 500. The choices $200, 300, 400, 500$ were used in [13] and we include them to ease comparison. The choices 160 and 256 are common in cryptographic applications.

Our experiments included the eight curve shapes described in Section 3: 3DIK, Edwards, ExtJQuartic, Hessian, InvEdwards, JacIntersect, Jacobian, and Jacobian-3. For comparison with previous results, and to show the importance of optimized curve formulas, we also carried out experiments for Std-Jac and Std-Jac-3.

Our experiments included many choices of the parameter a_0 in [13, Algorithm 1]. The largest power of 2 allowed in the algorithm is 2^{a_0}, and the largest power of 3 allowed in the algorithm is 3^{b_0} where $b_0 = \lceil(\ell - a_0)/\lg 3\rceil$. Specifically, we tried each $a_0 \in \{0, 10, 20, \ldots, 10\lfloor\ell/10\rfloor\}$. This matches the experiments reported in [13] for $\ell = 200$. We also tried all integers a_0 between 0.95ℓ and 1.00ℓ.

Our experiments included several sets S, i.e., sets of coefficients c allowed in $c2^a3^b$: the set $\{1\}$ used in [11]; the sets $\{1, 2, 3\}$, $\{1, 2, 3, 4, 8, 9, 16, 27, 81\}$, $\{1, 5, 7\}$, $\{1, 5, 7, 11, 13, 17, 19, 23, 25\}$ appearing in the graphs in [13, Appendix B] with labels "(1, 1)" and "(4, 4)" and "S_2" and "S_8"; and the sets $\{1, 2, 3, 4, 9\}$, $\{1, 2, 3, 4, 8, 9, 27\}$, $\{1, 5\}$, $\{1, 5, 7, 11\}$, $\{1, 5, 7, 11, 13\}$, $\{1, 5, 7, 11, 13, 17, 19\}$ appearing in the tables in [13, Appendix B]. We also included the sets $\{1, 2, 3, 5\}$, $\{1, 2, 3, 5, 7\}$, and so on through $\{1, 2, 3, 5, 7, 9, 11, 13, 15, 17, 19, 23, 25\}$; these sets are standard in the base-2 context but do not seem to have been included in previous double-base experiments.

(We have considered additional sets such as $\{1, 2, 3, 4, 5, 7, 9\}$. Multiples of 6 are not worthwhile but we see some potential for coefficients $4, 8, 10, \ldots$ in the context of 3DIK coordinates. However, those sets are not yet included in our experiments.)

We used straightforward combinations of additions, doublings, and triplings for the initial computation of cP for each $c \in S$. We have considered, but not yet included in our experiments, the use of quintuplings, merged operations, etc. for this computation. Reader beware: as mentioned in Section 1, the costs of these computations are ignored in [13], allowing arbitrarily large sets S for free and allowing arbitrarily small costs of computing nP; the costs in [13] thus become increasingly inconsistent with the costs in this paper (and in reality) as S grows.

We follow the standard (although debatable) practice of counting $\mathbf{S} = 0.8\mathbf{M}$ and disregarding other field operations. We caution the reader that other weightings of field operations can easily change the order of two systems with similar levels of performance.

Experiments and results. There are 8236 combinations of ℓ, a_0, and S described above. For each combination, we

- generated 10000 uniform random integers $n \in \{0, 1, \ldots, 2^\ell - 1\}$,
- converted each integer into a chain as specified by a_0 and S,
- checked that the chain indeed computed n starting the chain from 1, and
- counted the number of triplings, doublings, additions, readditions, and mixed additions for those 10000 choices of n.

We converted the results into multiplication counts for the curve shapes 3DIK, Edwards, ExtJQuartic, Hessian, InvEdwards, JacIntersect, Jacobian, Jacobian-3, Std-Jac, and Std-Jac-3, obtaining a cost for each of the 82360 combinations of ℓ, curve shape, a_0, and S.

Figure 1 shows, for each ℓ (horizontal axis) and each curve shape, the minimum cost per bit obtained when a_0 and S are chosen optimally. The implementor can easily read off the ranking of coordinate systems from this graph. Table 1 displays the same information in tabular form, along with the choices of a_0 and S.

There is no unique optimal choice of a_0 and S for every curve shape which gives rise to the fastest computation of a given ℓ-bit integer. For example, using Jacobian coordinates the best result is achieved by precomputing odd coefficients up to 13 for an integer of bit length at most 300. For 400-bit integers the optimum uses $S = \{1, 2, 3, 5, \ldots, 17\}$ and in the 500-bit case also 19 is included.

None of the optimal results for $\ell \geq 200$ uses a set of precomputed points discussed in [11] or [13]. The optimal coefficient sets in every case were those used in (fractional) sliding-window methods, i.e. the sets $\{1, 2, 3, 5, \ldots\}$.

Figure 2 shows, for each a_0 (horizontal axis) and each curve shape, the cost for $\ell = 200$ when S is chosen optimally. This graph demonstrates the importance of choosing the right bounds for a_0 and b_0 depending on the ratio of the doubling/tripling costs. We refer to Table 1 for the best choices of a_0 and S for each curve shape.

The fastest systems are Edwards, ExtJQuartic, and InvEdwards. They need the lowest number of multiplications for values of a_0 very close to ℓ. These systems are using larger sets of precomputations than slower systems such as Jacobian-3 or Jacobian, and fewer triplings. The faster systems all come with particularly fast addition laws, making the precomputations less costly, and particularly fast doublings, making triplings less attractive. This means that currently double-base chains offer no or very little advantage for the fastest systems. See [5] for a detailed description of single-base scalar multiplication on Edwards curves.

Not every curve can be represented by one of these fast systems. For curves in Jacobian coordinates values of a_0 around 0.6ℓ seem optimal and produce significantly faster scalar multiplication than single-base representations.

Figure 3 shows, for a smaller range of a_0 (horizontal axis) and each choice of S, the cost for Jacobian-3 coordinates for $\ell = 200$. This graph demonstrates several interesting interactions between the doubling/tripling ratio, the choice of S, and the final results. Figure 4 is a similar graph for Edwards curves. The optimal scalar-multiplication method in that graph uses $a_0 \approx 195$ with coefficients in the set $\pm\{1, 2, 3, 5, 7, 11, 13, 15\}$. The penalty for using standard single-base sliding-window methods is negligible. On the other hand, triplings are clearly valuable if storage for precomputed points is extremely limited.

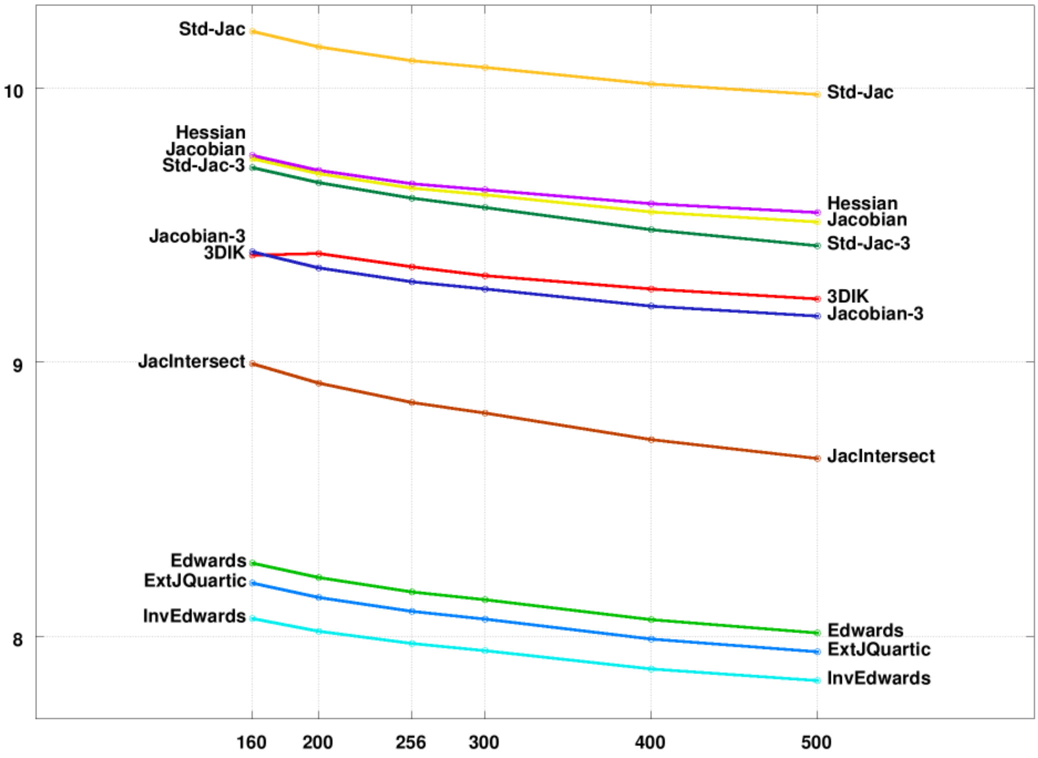
References

1. *P1363: Standard specifications for public key cryptography.* IEEE, 2000.
2. Roberto M. Avanzi, Henri Cohen, Christophe Doche, Gerhard Frey, Tanja Lange, Kim Nguyen, and Frederik Vercauteren. *The Handbook of Elliptic and Hyperelliptic Curve Cryptography.* CRC, 2005.

Table 1. Optimal parameters for each curve shape and each ℓ

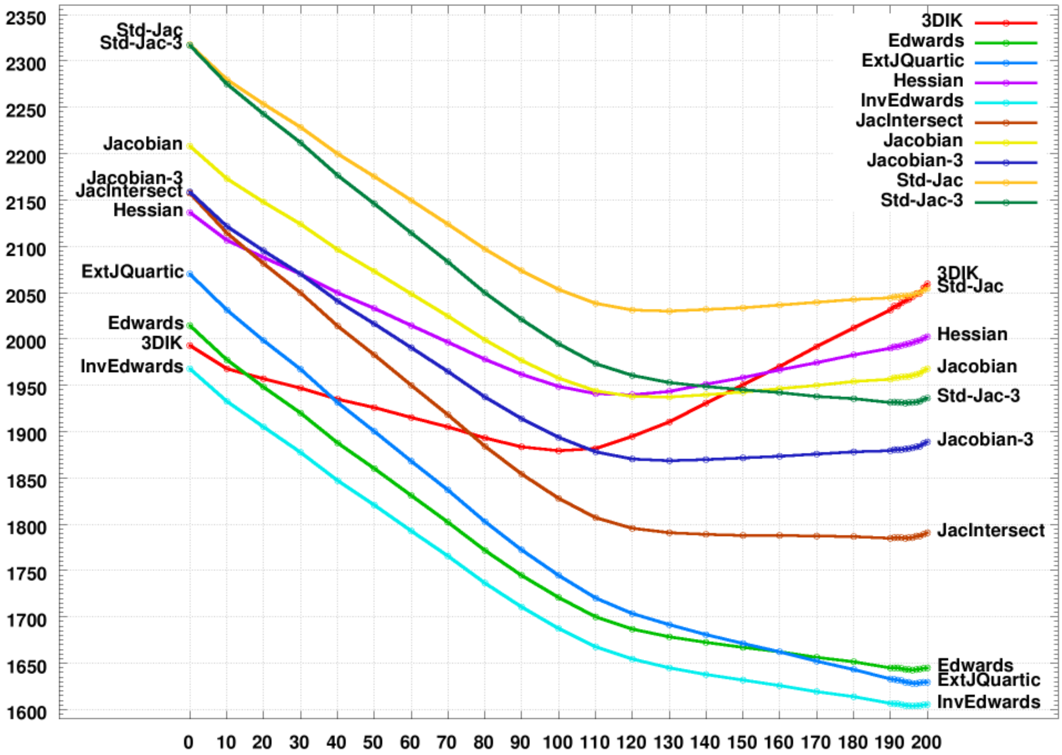
ℓ	Curve shape	Mults	Mults/ℓ	a_0	a_0/ℓ	S
160	3DIK	1502.393800	9.389961	80	0.5	$\{1\}$
200	3DIK	1879.200960	9.396005	100	0.5	$\{1,2,3,5,7\}$
256	3DIK	2393.193800	9.348413	130	0.51	$\{1,2,3,5,\dots,13\}$
300	3DIK	2794.431020	9.314770	160	0.53	$\{1,2,3,5,\dots,13\}$
400	3DIK	3706.581360	9.266453	210	0.53	$\{1,2,3,5,\dots,13\}$
500	3DIK	4615.646620	9.231293	270	0.54	$\{1,2,3,5,\dots,17\}$
160	Edwards	1322.911120	8.268194	156	0.97	$\{1,2,3,5,\dots,13\}$
200	Edwards	1642.867360	8.214337	196	0.98	$\{1,2,3,5,\dots,15\}$
256	Edwards	2089.695120	8.162872	252	0.98	$\{1,2,3,5,\dots,15\}$
300	Edwards	2440.611880	8.135373	296	0.99	$\{1,2,3,5,\dots,15\}$
400	Edwards	3224.251900	8.060630	394	0.98	$\{1,2,3,5,\dots,25\}$
500	Edwards	4005.977080	8.011954	496	0.99	$\{1,2,3,5,\dots,25\}$
160	ExtJQuartic	1310.995340	8.193721	156	0.97	$\{1,2,3,5,\dots,13\}$
200	ExtJQuartic	1628.386660	8.141933	196	0.98	$\{1,2,3,5,\dots,15\}$
256	ExtJQuartic	2071.217580	8.090694	253	0.99	$\{1,2,3,5,\dots,15\}$
300	ExtJQuartic	2419.026660	8.063422	299	1	$\{1,2,3,5,\dots,21\}$
400	ExtJQuartic	3196.304940	7.990762	399	1	$\{1,2,3,5,\dots,25\}$
500	ExtJQuartic	3972.191800	7.944384	499	1	$\{1,2,3,5,\dots,25\}$
160	Hessian	1560.487660	9.753048	100	0.62	$\{1,2,3,5,\dots,13\}$
200	Hessian	1939.682780	9.698414	120	0.6	$\{1,2,3,5,\dots,13\}$
256	Hessian	2470.643200	9.650950	150	0.59	$\{1,2,3,5,\dots,13\}$
300	Hessian	2888.322160	9.627741	170	0.57	$\{1,2,3,5,\dots,13\}$
400	Hessian	3831.321760	9.578304	240	0.6	$\{1,2,3,5,\dots,17\}$
500	Hessian	4772.497740	9.544995	300	0.6	$\{1,2,3,5,\dots,19\}$
160	InvEdwards	1290.333920	8.064587	156	0.97	$\{1,2,3,5,\dots,13\}$
200	InvEdwards	1603.737760	8.018689	196	0.98	$\{1,2,3,5,\dots,15\}$
256	InvEdwards	2041.223320	7.973529	252	0.98	$\{1,2,3,5,\dots,15\}$
300	InvEdwards	2384.817880	7.949393	296	0.99	$\{1,2,3,5,\dots,15\}$
400	InvEdwards	3152.991660	7.882479	399	1	$\{1,2,3,5,\dots,25\}$
500	InvEdwards	3919.645880	7.839292	496	0.99	$\{1,2,3,5,\dots,25\}$
160	JacIntersect	1438.808960	8.992556	150	0.94	$\{1,2,3,5,\dots,13\}$
200	JacIntersect	1784.742200	8.923711	190	0.95	$\{1,2,3,5,\dots,15\}$
256	JacIntersect	2266.135540	8.852092	246	0.96	$\{1,2,3,5,\dots,15\}$
300	JacIntersect	2644.233000	8.814110	290	0.97	$\{1,2,3,5,\dots,15\}$
400	JacIntersect	3486.773860	8.716935	394	0.98	$\{1,2,3,5,\dots,25\}$
500	JacIntersect	4324.718620	8.649437	492	0.98	$\{1,2,3,5,\dots,25\}$
160	Jacobian	1558.405080	9.740032	100	0.62	$\{1,2,3,5,\dots,13\}$
200	Jacobian	1937.129960	9.685650	130	0.65	$\{1,2,3,5,\dots,13\}$
256	Jacobian	2466.150480	9.633400	160	0.62	$\{1,2,3,5,\dots,13\}$
300	Jacobian	2882.657400	9.608858	180	0.6	$\{1,2,3,5,\dots,13\}$
400	Jacobian	3819.041260	9.547603	250	0.62	$\{1,2,3,5,\dots,17\}$
500	Jacobian	4755.197420	9.510395	310	0.62	$\{1,2,3,5,\dots,19\}$
160	Jacobian-3	1504.260200	9.401626	100	0.62	$\{1,2,3,5,\dots,13\}$
200	Jacobian-3	1868.530560	9.342653	130	0.65	$\{1,2,3,5,\dots,13\}$
256	Jacobian-3	2378.956000	9.292797	160	0.62	$\{1,2,3,5,\dots,13\}$
300	Jacobian-3	2779.917220	9.266391	200	0.67	$\{1,2,3,5,\dots,17\}$
400	Jacobian-3	3681.754460	9.204386	260	0.65	$\{1,2,3,5,\dots,17\}$
500	Jacobian-3	4583.527180	9.167054	330	0.66	$\{1,2,3,5,\dots,21\}$

Fig. 1. Multiplications per bit (all bits, all shapes)



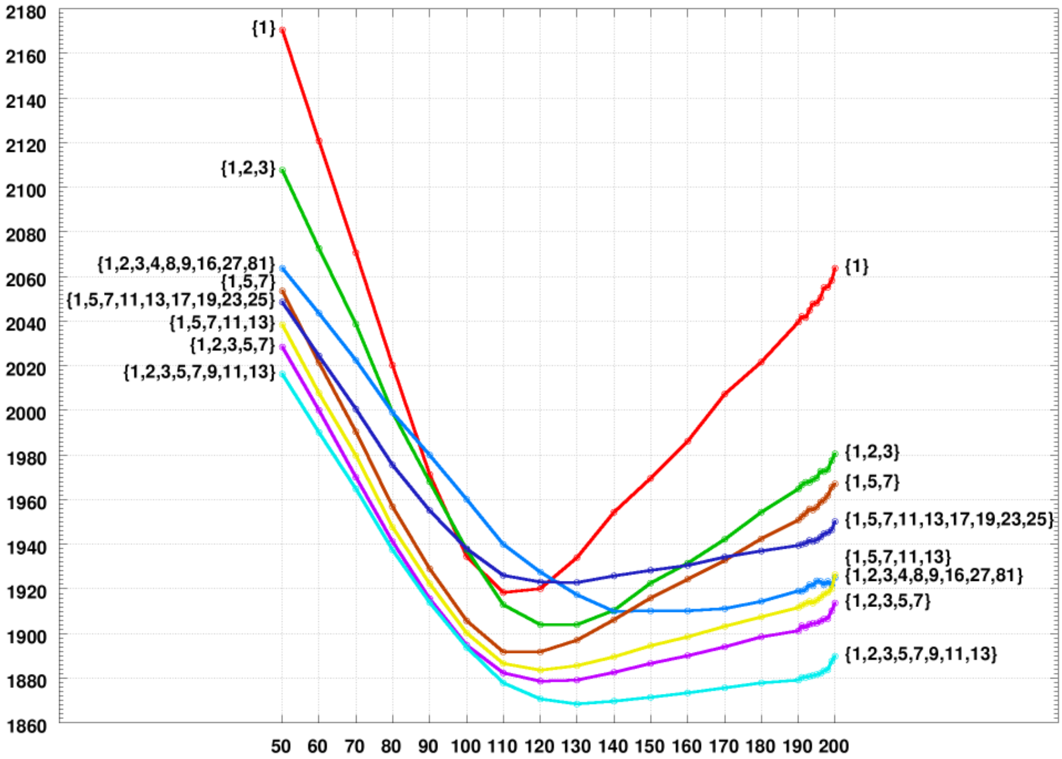
3. Rana Barua and Tanja Lange, editors. *Progress in Cryptology — INDOCRYPT 2006, 7th International Conference on Cryptology in India, Kolkata, India, December 11–13, 2006, Proceedings*, volume 4329 of *Lecture Notes in Computer Science*, Berlin, 2006. Springer.
4. Daniel J. Bernstein and Tanja Lange. Explicit-formulas database. `http://www.hyperelliptic.org/EFD`.
5. Daniel J. Bernstein and Tanja Lange. Faster addition and doubling on elliptic curves. In *ASIACRYPT 2007 [19]*, pages 29–50, 2007. `http://cr.yp.to/newelliptic/`.
6. Daniel J. Bernstein and Tanja Lange. Inverted Edwards coordinates. In *AAECC 2007 [8]*, pages 20–27, 2007. `http://cr.yp.to/newelliptic/`.
7. Ian F. Blake, Gadiel Seroussi, and Nigel P. Smart. *Elliptic curves in cryptography*, volume 265 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, Cambridge, 1999.
8. Serdar Boztas and Hsiao-Feng Lu, editors. *AAECC 2007*, volume 4851 of *Lecture Notes in Computer Science*, Berlin, 2007. Springer.
9. Alfred Brauer. On addition chains. *Bulletin of the American Mathematical Society*, 45:736–739, 1939.
10. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
11. Vassil Dimitrov, Laurent Imbert, and Pradeep K. Mishra. Efficient and secure elliptic curve point multiplication using double-base chains. In *ASIACRYPT 2005 [20]*, pages 59–78, 2005.

Fig. 2. Importance of doubling/tripling ratio (200 bits, all shapes)



12. Christophe Doche, Thomas Icart, and David R. Kohel. Efficient scalar multiplication by isogeny decompositions. In *PKC 2006 [23]*, pages 191–206, 2006.
13. Christophe Doche and Laurent Imbert. Extended double-base number system with applications to elliptic curve cryptography. In *INDOCRYPT 2006 [3]*, pages 335–348, 2006.
14. Christophe Doche and Tanja Lange. *Arithmetic of elliptic curves*, chapter 13 in [2], pages 267–302. 2005.
15. Sylvain Duquesne. Improving the arithmetic of elliptic curves in the Jacobi model. *Information Processing Letters*, 104:101–105, 2007.
16. Harold M. Edwards. A normal form for elliptic curves. *Bulletin of the American Mathematical Society*, 44:393–422, 2007. `http://www.ams.org/bull/2007-44-03/S0273-0979-07-01153-6/home.html`.
17. Darrel Hankerson, Alfred J. Menezes, and Scott A. Vanstone. *Guide to elliptic curve cryptography.* Springer, Berlin, 2003.
18. Huseyin Hisil, Gary Carter, and Ed Dawson. New formulae for efficient elliptic curve arithmetic. In *INDOCRYPT 2007 [21]*, 2007.
19. Kaoru Kurosawa, editor. *Advances in Cryptology — ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, Berlin Heidelberg, 2007. Springer.
20. Bimal Roy, editor. *Advances in Cryptology — ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4–8, 2005, Proceedings*, volume 3788, Berlin, 2005. Springer.

Fig. 3. Importance of parameter choices (200 bits, Jacobian-3)



21. Kannan Srinathan, Chandrasekaran Pandu Rangan, and Moti Yung, editors. *Progress in Cryptology — INDOCRYPT 2007*, volume 4859 of *Lecture Notes in Computer Science*, Berlin, 2007. Springer.
22. Edward G. Thurber. On addition chains $l(mn) \leq l(n) - b$ and lower bounds for $c(r)$. *Duke Mathematical Journal*, 40:907–913, 1973.
23. Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors. *9th international conference on theory and practice in public-key cryptography, New York, NY, USA, April 24–26, 2006, proceedings*, volume 3958 of *Lecture Notes in Computer Science*, Berlin, 2006. Springer.

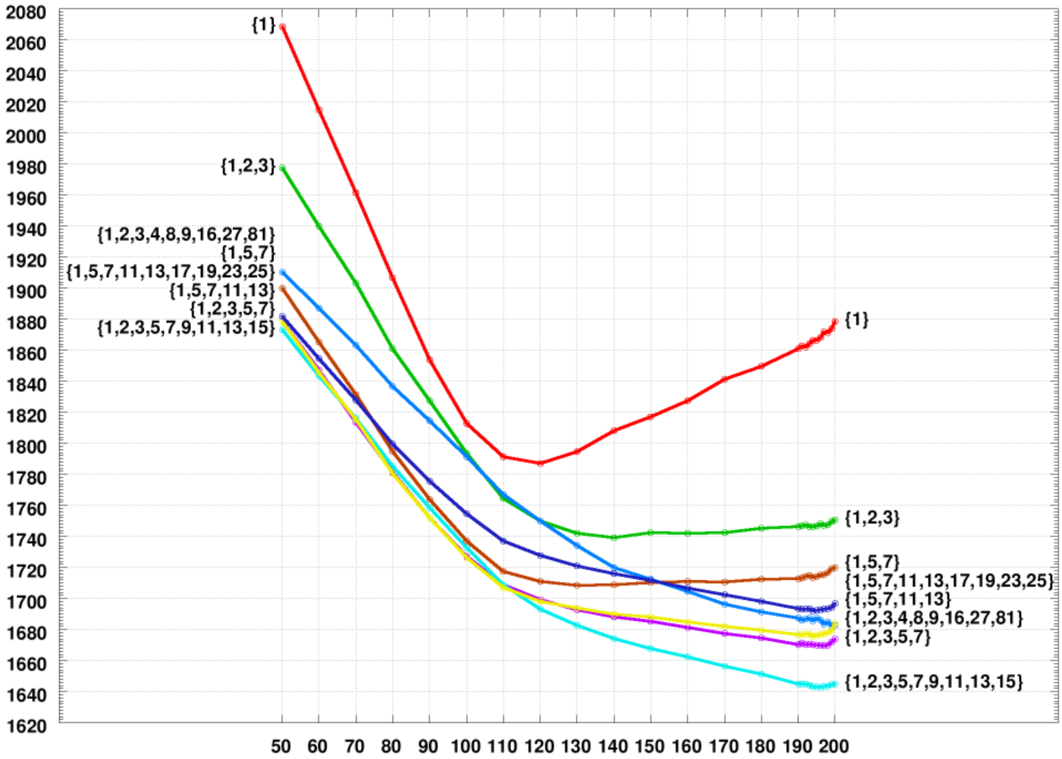
A Appendix: Quintupling on Edwards Curves

In this section we give formulas to compute the 5-fold of a point on an Edwards curve. We present two different versions which lead to the same result but have a different complexity. The first version needs $17\mathbf{M} + 7\mathbf{S}$ while version 2 needs $14\mathbf{M} + 11\mathbf{S}$. The second version is better if $\mathbf{S}/\mathbf{M}$ is small.

Both versions were verified to produce the 5-fold of the input point $(X_1 : Y_1 : Z_1)$ by symbolically computing $5(X_1 : Y_1 : Z_1)$ using the addition and doubling formulas from [5] and comparing it with $(X_5 : Y_5 : Z_5)$.

It is interesting to note that the formulas do not have minimal degree. The new variables X_5, Y_5, Z_5 have total degree 33 in the initial variables even though

Fig. 4. Importance of parameter choices (200 bits, Edwards)



one would expect degree 5^2. Indeed, X_5, Y_5, Z_5 are all divisible by the degree-8 polynomial $((X_1^2-Y_1^2)^2+4Y_1^2(Z_1^2-Y_1^2))((X_1^2-Y_1^2)^2+4X_1^2(Z_1^2-X_1^2))$. Minimizing the number of operations led to better results for our extended polynomials.

The $17\mathbf{M}+7\mathbf{S}$-formulas for quintupling:

$$A = X_1^2;\ B = Y_1^2;\ C = Z_1^2;\ D = A + B;$$

$$E = 2C - D;\ F = D \cdot (B - A);\ G = E \cdot ((X_1 + Y_1)^2 - D);$$

$$H = F^2;\ I = G^2;\ J = H + I;\ K = H - I;\ L = J \cdot K;$$

$$M = D \cdot E;\ N = E \cdot F;\ O = 2M^2 - J;\ P = 4N \cdot O;$$

$$Q = 4K \cdot N \cdot (D - C);\ R = O \cdot J;\ S = R + Q;\ T = R - Q;$$

$$X_5 = X_1 \cdot (L + B \cdot P) \cdot T;\ Y_5 = Y_1 \cdot (L - A \cdot P) \cdot S;\ Z_5 = Z_1 \cdot S \cdot T.$$

The $14\mathbf{M}+11\mathbf{S}$-formulas for quintupling:

$$A = X_1^2;\ B = Y_1^2;\ C = Z_1^2;\ D = A + B;\ E = 2C - D;$$

$$F = A^2;\ G = B^2;\ H = F + G;\ I = D^2 - H;\ J = E^2;$$

$$K = G - F;\ L = K^2;\ M = 2I \cdot J;\ N = L + M;\ O = L - M;$$

$$P = N \cdot O;\ Q = (E + K)^2 - J - L;\ R = ((D + E)^2 - J - H - I)^2 - 2N;$$

$$S = Q \cdot R;\ T = 4Q \cdot O \cdot (D - C);\ U = R \cdot N;\ V = U + T;\ W = U - T;$$

$$X_5 = 2X_1 \cdot (P + B \cdot S) \cdot W;\ Y_5 = 2Y_1 \cdot (P - A \cdot S) \cdot V;\ Z_5 = Z_1 \cdot V \cdot W.$$

Note that only variables $A \ldots E$ have the same values in the two versions.