Performance evaluation of a new coordinate system for elliptic curves

Daniel J. Bernstein[1] and Tanja Lange[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
tanja@hyperelliptic.org

Abstract. Edwards recently introduced a new coordinate system for non-binary elliptic curves. This paper evaluates the cost of the new coordinate system for cryptographic applications. In particular, this paper shows how to compute a doubling using only 3 field multiplications and 4 field squarings; a mixed addition using only 9 field multiplications and 1 field squaring, when curve parameters are chosen to be small; and a non-mixed addition using only 10 field multiplications and 1 field squaring, when curve parameters are chosen to be small. Furthermore, the non-mixed addition formulas can be used for doublings at no extra cost, simplifying side-channel protection. Even better, for some curves, the non-mixed addition formulas work for *all* pairs of inputs.

Keywords: elliptic curves, efficient implementation, multi-scalar multiplication, side-channel countermeasures, unified addition formulas

1 Introduction

The core operations in elliptic-curve cryptography are single-scalar multiplication $(m, P \mapsto mP)$, double-scalar multiplication $(m, n, P, Q \mapsto mP + nQ)$, etc.

In the Crypto '85 paper that introduced elliptic-curve cryptography, Miller proposed carrying out these operations in Jacobian coordinates: "Each point is represented by the triple (x, y, z) which corresponds to the point $(x/z^2, y/z^3)$" on a curve $y^2 = x^3 + a_4x + a_6$. See [21, page 424]. One can add two points in Jacobian coordinates using 16 field multiplications, specifically $11\mathbf{M} + 5\mathbf{S}$; here we keep separate tallies of squarings $\mathbf{S}$ and general multiplications $\mathbf{M}$. A precomputation suggested by Chudnovsky and Chudnovsky reduces $11\mathbf{M} + 5\mathbf{S}$

[*] Date of this document: 2007.05.22.

[*] Permanent ID of this document: 95616567a6ba20f575c5f25e7cebaf83.

[*] This work has been supported in part by the European Commission through the IST Programme under Contract IST–2002–507932 ECRYPT.

to $10\mathbf{M}+4\mathbf{S}$. A mixed addition — this means that one input has $z=1$ — takes only $8\mathbf{M}+3\mathbf{S}$. A doubling takes only $3\mathbf{M}+5\mathbf{S}$ if $a_4=-3$.

Several subsequent papers have analyzed the performance of other coordinate systems proposed in the mathematical literature. See, e.g., [20] for the speed of Jacobi intersections, [15] for the speed of Hessians, and [6] for the speed of Jacobi quartics; see also [22], which introduced Montgomery coordinates and analyzed their speed. These coordinate systems have attracted some interest — in particular, many of these coordinate systems simplify protection against side-channel attacks, and the speed records in [5] for single-scalar multiplication were set with Montgomery coordinates — but Jacobian coordinates are still the overall speed leaders for multi-scalar multiplication.

A few months ago a new coordinate system was added to the mathematical literature: Edwards showed in [14] that all elliptic curves over a number field could be transformed to the shape $x^2+y^2=c^2(1+x^2y^2)$, with $(0,c)$ as neutral element and with the surprisingly simple and symmetric addition law

$$(x_1,y_1),(x_2,y_2)\mapsto\left(\frac{x_1y_2+y_1x_2}{c(1+x_1x_2y_1y_2)},\frac{y_1y_2-x_1x_2}{c(1-x_1x_2y_1y_2)}\right).$$

To keep this paper self-contained, we include a section explaining how to transform an elliptic curve over a finite field from short Weierstrass form $Y^2=X^3+aX+b$ to Edwards coordinates. Some elliptic curves — for example, the NIST curves — require a field extension for the transformation; but many other elliptic curves — for example, the curve "Curve25519" previously used to set speed records for single-scalar multiplication — have transformations defined over the original field. See Section 3.

Our main goal in this paper is to analyze the impact of Edwards coordinates on cryptographic applications. Our main conclusions are that Edwards coordinates (1) break solidly through the Jacobian-coordinate speed barrier, (2) are competitive with Montgomery coordinates for single-scalar multiplication, and (3) are the new speed leaders for multi-scalar multiplication. Specifically, by tweaking the Edwards coordinate system and streamlining the addition formulas, we show that

- one can perform an inversion-free doubling using $3\mathbf{M}+4\mathbf{S}$;
- one can perform an inversion-free addition using $10\mathbf{M}+1\mathbf{S}+1\mathbf{d}$ — here $\mathbf{d}$ is the cost of multiplying by a selectable curve parameter; and
- one can perform a mixed addition (inversion-free with one input affine) using $9\mathbf{M}+1\mathbf{S}+1\mathbf{d}$.

See Section 2 for details of these computations, and Section 4 for a thorough comparison of the resulting costs for single-scalar multiplication, double-scalar multiplication, etc.

Half of the curves that we consider have the extra feature that our inversion-free addition formulas are **complete**. This means that the formulas work for *all* pairs of input points on the curve, with no exceptions for doubling, no exceptions for the neutral element, no exceptions for negatives, etc. Some previous addition formulas have been advertised as **unified** formulas that can handle generic doublings, simplifying protection against side-channel attacks; our addition formulas have the stronger property of completeness, and are less expensive than previous unified formulas. See Section 4 for further discussion.

2 Computations in the new coordinate system

Let k be a field of characteristic different from 2. Fix $c, d, e \in k$ such that $c \neq 0$, $d \neq 0$, $e \neq 0$, and $e = 1 - dc^4$. The curve $x^2 + y^2 = c^2(1 + dx^2y^2)$ over k is then birationally equivalent to the elliptic curve $V^2 = (U + e)(U^2 + 4U + 4e)$ by the map $(x, y) \mapsto (U, V) = (-2ey/(y - c), -2ec(c^2dy^2 - 1)x/(y - c)^2)$; furthermore, the addition law

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

on the curve $x^2 + y^2 = c^2(1 + dx^2y^2)$ corresponds to the standard addition law on the elliptic curve. See Section 3.

We refer to the curve $x^2 + y^2 = c^2(1 + dx^2y^2)$ as an **Edwards curve**, and the coordinates (x, y) as **affine Edwards coordinates**. This is only a slight generalization from the case $d = 1$ originally considered by Edwards; all Edwards curves can easily be transformed to the case $d = 1$ over an algebraically closed field. However, the extra parameter d allows extra curves over a finite field, and in particular it allows curves where the addition law is complete, namely curves where d is not a square.

This section presents streamlined formulas for doubling, mixed addition, etc. in Edwards coordinates with arbitrary parameters c, d. As usual we count the number of operations in the underlying field. We keep separate tallies of the number of

- general multiplications (each costing $\mathbf{M}$),
- squarings (each costing $\mathbf{S}$),
- multiplications by c (each costing $\mathbf{c}$),
- multiplications by d (each costing $\mathbf{d}$), and
- additions/subtractions (each costing $\mathbf{a}$).

The costs $\mathbf{M}, \mathbf{S}, \mathbf{c}, \mathbf{d}, \mathbf{a}$ depend on the choice of platform, on the choice of finite field, and on the choice of c and d.

The fastest case for our formulas is the case $c = 1$. Every Edwards curve can easily be transformed to an Edwards curve over the same field having $c = 1$. In Section 4 we assume that $c = 1$.

Addition. To avoid the inversions in the original addition formulas

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

we homogenize the curve equation to $(X^2 + Y^2)Z^2 = c^2(Z^4 + dX^2Y^2)$, where the projective point $(X_1 : Y_1 : Z_1)$ corresponds to the affine point $(X_1/Z_1, Y_1/Z_1)$.

The neutral element is $(0 : c : 1)$ in projective coordinates, and the inverse of $(X_1 : Y_1 : Z_1)$ is $(-X_1 : Y_1 : Z_1)$. We refer to these coordinates as **projective Edwards coordinates**.

The following formulas, given $(X_1 : Y_1 : Z_1)$ and $(X_2 : Y_2 : Z_2)$, compute the sum $(X_3 : Y_3 : Z_3) = (X_1 : Y_1 : Z_1) + (X_2 : Y_2 : Z_2)$:

$$\begin{aligned} A &= Z_1 \cdot Z_2;\ B = A^2;\ C = X_1 \cdot X_2;\ D = Y_1 \cdot Y_2; \\ E &= (X_1 + Y_1) \cdot (X_2 + Y_2) - C - D;\ F = d \cdot C \cdot D; \\ X_3 &= A \cdot E \cdot (B - F);\ Y_3 = A \cdot (D - C) \cdot (B + F); \\ Z_3 &= c \cdot (B - F) \cdot (B + F). \end{aligned}$$

One readily counts $10\mathbf{M} + 1\mathbf{S} + 1\mathbf{c} + 1\mathbf{d} + 7\mathbf{a}$, as stated in Section 1.

The following specific sequence of operations starts with registers R_1, R_2, R_3 containing X_1, Y_1, Z_1 and registers R_4, R_5, R_6 containing X_2, Y_2, Z_2, uses just two temporary registers R_7, R_8 and constants c, d, ends with registers R_1, R_2, R_3 containing X_3, Y_3, Z_3 and untouched registers R_4, R_5, R_6 containing X_2, Y_2, Z_2, and uses $10\mathbf{M} + 1\mathbf{S} + 1\mathbf{c} + 1\mathbf{d} + 7\mathbf{a}$:

$$\begin{aligned} &R_3 \leftarrow R_3 \cdot R_6;\ R_7 \leftarrow R_1 + R_2;\ R_8 \leftarrow R_4 + R_5;\ R_1 \leftarrow R_1 \cdot R_4; \\ &R_2 \leftarrow R_2 \cdot R_5;\ R_7 \leftarrow R_7 \cdot R_8;\ R_7 \leftarrow R_7 - R_1;\ R_7 \leftarrow R_7 - R_2; \\ &R_7 \leftarrow R_7 \cdot R_3;\ R_8 \leftarrow R_1 \cdot R_2;\ R_8 \leftarrow d \cdot R_8;\ R_2 \leftarrow R_2 - R_1; \\ &R_2 \leftarrow R_2 \cdot R_3;\ R_3 \leftarrow R_3^2;\ R_1 \leftarrow R_3 - R_8;\ R_3 \leftarrow R_3 + R_8; \\ &R_2 \leftarrow R_2 \cdot R_3;\ R_3 \leftarrow R_3 \cdot R_1;\ R_1 \leftarrow R_1 \cdot R_7;\ R_3 \leftarrow c \cdot R_3. \end{aligned}$$

We emphasize that these formulas work whether or not $(X_1 : Y_1 : Z_1) = (X_2 : Y_2 : Z_2)$. There is no need to go to extra effort to unify the addition formulas with separate doubling formulas; the addition formulas are already unified. See Section 3 for further discussion of the scope of validity of the addition formulas; in particular, we repeat the fact that the addition law works for *all* pairs of input points if d is not a square.

Mixed addition. "Mixed addition" refers to the case that Z_2 is known to be 1. In this case the multiplication $A = Z_1 \cdot Z_2$ can be eliminated, reducing the total costs to $9\mathbf{M} + 1\mathbf{S} + 1\mathbf{c} + 1\mathbf{d} + 7\mathbf{a}$:

$$\begin{aligned} B &= Z_1^2;\ C = X_1 \cdot X_2;\ D = Y_1 \cdot Y_2; \\ E &= (X_1 + Y_1) \cdot (X_2 + Y_2) - C - D;\ F = d \cdot C \cdot D; \\ X_3 &= Z_1 \cdot E \cdot (B - F);\ Y_3 = Z_1 \cdot (D - C) \cdot (B + F); \\ Z_3 &= c \cdot (B - F) \cdot (B + F). \end{aligned}$$

Doubling. "Doubling" refers to the case that $(X_1 : Y_1 : Z_1)$ and $(X_2 : Y_2 : Z_2)$ are known to be equal. In this case one can use the following formulas to speed up the computation to $3\mathbf{M} + 4\mathbf{S} + 3\mathbf{c} + 6\mathbf{a}$:

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

The following specific sequence of operations, starting with registers R_1, R_2, R_3 containing X_1, Y_1, Z_1, changes registers R_1, R_2, R_3 to contain X_3, Y_3, Z_3, using $3\mathbf{M} + 4\mathbf{S} + 3\mathbf{c} + 6\mathbf{a}$ and using just two temporary registers R_4, R_5:

$$\begin{aligned}
&R_4 \leftarrow R_1 + R_2;\ R_3 \leftarrow c \cdot R_3;\ R_1 \leftarrow R_1^2;\ R_2 \leftarrow R_2^2;\\
&R_3 \leftarrow R_3^2;\ R_4 \leftarrow R_4^2;\ R_3 \leftarrow R_3 + R_3;\ R_5 \leftarrow R_1 + R_2;\\
&R_2 \leftarrow R_1 - R_2;\ R_4 \leftarrow R_4 - R_5;\ R_3 \leftarrow R_5 - R_3;\ R_1 \leftarrow R_3 \cdot R_4;\\
&R_3 \leftarrow R_3 \cdot R_5;\ R_2 \leftarrow R_2 \cdot R_5;\ R_1 \leftarrow c \cdot R_1;\ R_2 \leftarrow c \cdot R_2.
\end{aligned}$$

The following alternate sequence of operations uses one more addition, totalling $3\mathbf{M} + 4\mathbf{S} + 3\mathbf{c} + 7\mathbf{a}$, but uses just one additional register R_4:

$$\begin{aligned}
&R_3 \leftarrow c \cdot R_3;\ R_4 \leftarrow R_1^2;\ R_1 \leftarrow R_1 + R_2;\ R_1 \leftarrow R_1^2;\ R_2 \leftarrow R_2^2;\\
&R_3 \leftarrow R_3^2;\ R_3 \leftarrow R_3 + R_3;\ R_4 \leftarrow R_2 + R_4;\ R_2 \leftarrow R_2 + R_2;\\
&R_2 \leftarrow R_4 - R_2;\ R_1 \leftarrow R_1 - R_4;\ R_2 \leftarrow R_2 \cdot R_4;\ R_3 \leftarrow R_4 - R_3;\\
&R_1 \leftarrow R_1 \cdot R_3;\ R_3 \leftarrow R_3 \cdot R_4;\ R_1 \leftarrow c \cdot R_1;\ R_2 \leftarrow c \cdot R_2.
\end{aligned}$$

Another option is to scale $(X_3 : Y_3 : Z_3)$ to $(X_3/c : Y_3/c : Z_3/c)$, replacing two multiplications by c with one multiplication by $1/c$. Of course, all three multiplications by c can be skipped if $c = 1$.

3 From Weierstrass coordinates to the new coordinate system

Given a large-characteristic elliptic curve E in Weierstrass form, one can transform group operations on E into group operations on the curve $x^2 + y^2 = c^2(1 + dx^2y^2)$, for an appropriate choice of (c, d). This section presents the details of the transformation. Proofs appear in Appendix A.

To illustrate the transformation, we use an elliptic curve previously published for fast scalar multiplication using Montgomery coordinates, namely the elliptic curve $Y^2 = (X')^3 + 486662(X')^2 + (X')$ modulo $p = 2^{255} - 19$; see [5]. Group operations on this curve “Curve25519” can be transformed into group operations on the curve $x^2 + y^2 = c^2(1 - 2x^2y^2)$ where

$$c = 26923790352479969033134494549992980665882225819690684221603341630881131953763;$$

this section explains how this c was calculated and how the transformation works. Applications that are free to choose curves can save time by choosing both c and d to be small.

Overview of the transformation. The transformation has three parts:

- Transform to short Weierstrass coordinates, i.e., to (X, Y) satisfying $Y^2 = X^3 + aX + b$.
- Transform to (U, V) satisfying $V^2 = (U + e)(U^2 + 4U + 4e)$, where e is a root of the polynomial $4a^3(e-2)^2(e^2 + 32e - 32)^2 + 27b^2(e^2 - 16e + 16)^3$.

- Transform to (x, y) satisfying $x^2 + y^2 = c^2(1 + dx^2y^2)$, where c, d satisfy $e = 1 - dc^4$.

For computational purposes it is of course most convenient if all of the objects here — the parameter e for the intermediate curve $V^2 = (U+e)(U^2+4U+4e)$, the parameters c, d for the final curve $x^2+y^2 = c^2(1+dx^2y^2)$, and the coefficients of the transformations between curves — are as small as possible. A multiplication by c in a field of size p^2 can be made faster if c is actually in a subfield of size p, for example, and can be made even faster if c is actually an integer much smaller than p. We have made no attempt to optimize our transformation from this perspective. We comment that the final curve $x^2 + y^2 = c^2(1 + dx^2y^2)$ has a point $(0, -c)$ of order 2, so it cannot be defined over a prime field if the original elliptic curve has odd order over that field; in particular, we would not expect this coordinate system to provide good performance for the NIST curves. But many other curves allow c, d, and the transformations to be defined over prime fields, as illustrated by the Curve25519 example.

Step 1: transform to (X, Y) on $Y^2 = X^3 + aX + b$. We assume that the original elliptic curve E is defined over a field k of characteristic different from 2 or 3; for example, a large prime field. Then E can be transformed to short Weierstrass form $Y^2 = X^3 + aX + b$ for some $a, b \in k$ with $4a^3 + 27b^2 \neq 0$.

Curve25519 example: The substitution $X = X' + 486662/3$ transforms $Y^2 = (X')^3 + 486662(X')^2 + (X')$ into the elliptic curve $Y^2 = X^3 + aX + b$ where $a = -236839902241/3$ and $b = 230521961007359098/27$.

Step 2: choose root e of $4a^3(e-2)^2(e^2+32e-32)^2+27b^2(e^2-16e+16)^3$. Extend k if necessary so that $4a^3(e-2)^2(e^2+32e-32)^2+27b^2(e^2-16e+16)^3$ has roots in k. Choose a root e of this polynomial. Appendix A shows that $e \neq 1$.

Curve25519 example, continued: The integer

$$e = 37095705934669439343138083508754565189542113879843219016388785533085940283556$$

is a root of the polynomial $4a^3(e-2)^2(e^2+32e-32)^2+27b^2(e^2-16e+16)^3$ modulo $2^{255} - 19$ when $a = -236839902241/3$ and $b = 230521961007359098/27$.

Step 3: choose t with $at^2 = -3(e^2 - 16e + 16)$ and $bt^3 = 2(e - 2)(e^2 + 32e - 32)$. There are three cases here:

- For $b = 0$: Extend k if necessary so that $-3(e^2 - 16e + 16)/a$ has a square root t in k.
- For $a = 0$: Extend k if necessary so that $2(e-2)(e^2+32e-32)/b$ has a cube root t in k.
- For $ab \neq 0$: Define $t = -2a(e - 2)(e^2 + 32e - 32)/3b(e^2 - 16e + 16)$.

In all three cases, $at^2 = -3(e^2 - 16e + 16)$ and $bt^3 = 2(e - 2)(e^2 + 32e - 32)$. Appendix A shows that $t \neq 0$.

Curve25519 example, continued: The integer

$$t = 53391073185350220317628758021919741641991349306709375029437564595301256030719$$

satisfies $at^2 = -3(e^2 - 16e + 16)$ and $bt^3 = 2(e-2)(e^2 + 32e - 32)$ for the a, b, e shown above.

Step 4: choose s with $s^2 = (t/3)^3$. Extend k if necessary so that $(t/3)^3$ has a square root s in k. Note that $s \neq 0$.

Curve25519 example, continued: The integer

$$s = 18649689278727228840354232979487548124034359932680984740981561174796302117467$$

is a square root of $(t/3)^3$ modulo $2^{255} - 19$ for the t shown above.

Step 5: transform to (U, V) on $V^2 = (U + e)(U^2 + 4U + 4e)$. If (X, Y) satisfies $Y^2 = X^3 + aX + b$ then $(U, V) = ((tX - e - 4)/3, sY)$ satisfies $V^2 = (U + e)(U^2 + 4U + 4e)$. See Appendix A.

Curve25519 example, continued: If $X_1 = 7$ and

$$Y_1 = 19172526001133118116405784977723800924384594365247398516339237030405822788330$$

then $Y_1^2 = X_1^3 + aX_1 + b$ modulo $2^{255} - 19$. The integers

$$U_1 = 54317890835602603248302248377217254841497451422887186710162596874051052490542,$$
$$V_1 = 27265453978268041840748490304217756379746991228816427167751352985036992860401$$

are, respectively, $(tX_1 - e - 4)/3$ and sY_1 modulo $2^{255} - 19$, for the t, s shown above. As a double-check, V_1^2 is the same as $(U_1 + e)(U_1^2 + 4U_1 + 4e)$ modulo $2^{255} - 19$, for the e shown above.

Step 6: choose c and d with $e = 1 - dc^4$. Write e in the form $1 - dc^4$ by selecting $c \in k - \{0\}$ and defining $d = (1 - e)/c^4$. Alternatively, first choose d such that $(1 - e)/d$ is a nonzero 4th power in k, and then choose c as a 4th root. If k is a finite field then at least $1/4$ of the nonzero elements of k are 4th powers; this is why d can always be chosen to be extremely small.

Curve25519 example, continued: If $d = -2$ and

$$c = 26923790352479969033134494549992980665882225819690684221603341630881131953763$$

then $1 - dc^4$ modulo $2^{255} - 19$ is the same as the integer e shown above. Note that this is not our recommended choice of (c, d), namely $c = 1$ and $d = 1 - e$.

Applications that are free to choose curves can first choose small c, d, then compute $e = 1 - dc^4$, then write down the curve $V^2 = (U + e)(U^2 + 4U + 4e)$.

Step 7: transform to (x, y) on $x^2 + y^2 = c^2(1 + dx^2y^2)$. If (U, V) satisfies $V^2 = (U + e)(U^2 + 4U + 4e)$, and if $U + 2e$ and $U^2 + 4U + 4e$ are nonzero, then $(x, y) = (2cV/(U^2 + 4U + 4e), cU/(U + 2e))$ satisfies $x^2 + y^2 = c^2(1 + dx^2y^2)$. See Appendix A.

To recap: We have transformed points (X, Y) on the curve $Y^2 = X^3 + aX + b$ into points (U, V) on the curve $V^2 = (U + e)(U^2 + 4U + 4e)$, and then—except for a few points where $U + 2e = 0$ or $U^2 + 4U + 4e = 0$—into points $(x, y) = (2cV/(U^2 + 4U + 4e), cU/(U + 2e))$ on the curve $x^2 + y^2 = c^2(1 + dx^2y^2)$.

Curve25519 example, continued: The integers

$$x_1 = 38449057356254576649503480194105044252465313685119321805170194941589028170479,$$
$$y_1 = 15703921733006088416774302803525630526894002882819000026831678189383841685889$$

are, respectively, $2cV_1/(U_1^2 + 4U_1 + 4e)$ and $cU_1/(U_1 + 2e)$ modulo $2^{255} - 19$ for the integers c, e, U_1, V_1 shown above. As a double-check, $x_1^2 + y_1^2$ is the same as $c^2(1 + dx_1^2 y_1^2)$ modulo $2^{255} - 19$, where $d = -2$ as above.

Step 8: perform group operations on $x^2 + y^2 = c^2(1 + dx^2y^2)$. If (x_1, y_1) and (x_2, y_2) are points on $x^2 + y^2 = c^2(1 + dx^2y^2)$, and if $dx_1x_2y_1y_2 \notin \{-1, 1\}$, then

$$(x_3, y_3) = \left(\frac{x_1y_2 + y_1x_2}{c(1 + dx_1x_2y_1y_2)}, \frac{y_1y_2 - x_1x_2}{c(1 - dx_1x_2y_1y_2)} \right)$$

is also a point on $x^2 + y^2 = c^2(1 + dx^2y^2)$. This operation $(x_1, y_1), (x_2, y_2) \mapsto (x_3, y_3)$ is consistent with the standard addition law on the elliptic curve $V^2 = (U+e)(U^2+4U+4e)$, and therefore with the addition law on the original elliptic curve E. See Appendix A.

We emphasize that this addition law for the curve $x^2 + y^2 = c^2(1 + dx^2y^2)$ does not need any modification for the doubling case $(x_1, y_1) = (x_2, y_2)$. The only failure cases of the addition law are the cases $dx_1x_2y_1y_2 = \pm 1$, which for doubling means $dx_1^2y_1^2 = \pm 1$, occurring for only a few points on the curve. If d is not a square in k then there are *no* failure cases for addition; see Appendix A.

Curve25519 example, continued: The integers

$$x_3 = 44070118552736708090129770544946991295910146229981336346215991426600634526292,$$
$$y_3 = 49211788949231821853202700980531119200569440922762704174632595798780438797193$$

are, respectively, $2x_1y_1/c(1+dx_1^2y_1^2)$ and $(y_1^2 - x_1^2)/c(1 - dx_1^2y_1^2)$ modulo $2^{255} - 19$, for the integers c, d, x_1, y_1 shown above. As a double-check, $x_3^2 + y_3^2$ is the same as $c^2(1 + dx_3^2y_3^2)$ modulo $2^{255} - 19$.

Step 9: transform back to (U, V) on $V^2 = (U+e)(U^2+4U+4e)$. After a sequence of group operations produces a point (x, y) on $x^2 + y^2 = c^2(1+dx^2y^2)$, solve the equations $(x, y) = (2cV/(U^2 + 4U + 4e), cU/(U + 2e))$ to find the corresponding point (U, V) on $V^2 = (U + e)(U^2 + 4U + 4e)$. In other words, define $U = -2ey/(y - c)$ and $V = -2ec(c^2dy^2 - 1)x/(y - c)^2$.

Curve25519 example, continued: The integers

$$U_3 = 19353927226809319800023215682473111991996239556853927784590583590843827970744,$$
$$V_3 = 15914012472049123343512503873797345822611472250094411327637863525569550289727$$

are, respectively, $-2ey_3/(y_3 - c)$ and $-2ec(c^2dy_3^2 - 1)x_3/(y_3 - c)^2$ modulo $2^{255} - 19$, for the integers c, d, e, x_3, y_3 shown above. As a double-check, V_3^2 is the same as $(U_3 + e)(U_3^2 + 4U_3 + 4e)$ modulo $2^{255} - 19$.

Step 10: transform back to (X, Y) on $Y^2 = X^3 + aX + b$. Finally, solve the equations $(U, V) = ((tX - e - 4)/3, sY)$ to find the corresponding point

(X, Y) on $Y^2 = X^3 + aX + b$. In other words, define $X = (3U + e + 4)/t$ and $Y = V/s$.

Curve25519 example, continued: The integers

$$X_3 = 44177973078763452834055410435474786018253753500561242948153168022378284829263,$$
$$Y_3 = 34723047021396797201571638306466206286938174048238133551788788106595160363933$$

are, respectively, $(3U_3 + e + 4)/t$ and V_3/s modulo $2^{255} - 19$, for the integers e, s, t, U_3, V_3 shown above. This point (X_3, Y_3) is the same as the output of the standard doubling law on $Y^2 = X^3 + aX + b$ when the input is the point (X_1, Y_1) shown above: (X_3, Y_3) is the same as $(\lambda^2 - 2X_1, \lambda(X_1 - X_3) - Y_1)$ modulo $2^{255} - 19$ where $\lambda = (3X_1^2 + a)/2Y_1$.

Of course, one does not need to transform results back and forth after each group operation. One can transform points to the curve $x^2 + y^2 = c^2(1 + dx^2y^2)$ and carry out many group operations on that curve before transforming the results back.

Notes regarding exceptional points. The point at infinity on the elliptic curve $V^2 = (U + e)(U^2 + 4U + 4e)$ corresponds to the point $(0, c)$ on the curve $x^2 + y^2 = c^2(1 + dx^2y^2)$. The addition law does not need any modification to handle $(0, c)$ as an input or output: for each point $P = (x_1, y_1)$ on the curve, $(0, c)$ is the sum of P and $-P = (-x_1, y_1)$, and P is the sum of $(0, c)$ and P.

If d is a square in k then there are additional points to worry about. The points with $U + 2e = 0$ are $(-2e, \pm 4c^2\sqrt{d}e)$; both of these points correspond to $(0 : 1 : 0)$, a singular point at infinity on the curve $x^2 + y^2 = c^2(1 + dx^2y^2)$. The points with $U^2 + 4U + 4e = 0$ are $(-2 \pm 2c^2\sqrt{d}, 0)$; both of these points correspond to $(1 : 0 : 0)$, another singular point at infinity on the curve $x^2 + y^2 = c^2(1 + dx^2y^2)$. The addition law for $x^2 + y^2 = c^2(1 + dx^2y^2)$ is not defined for the singular points. One can, in many applications, rely on randomization to avoid the singular points, or one can fall back on group operations in $V^2 = (U + e)(U^2 + 4U + 4e)$ if singular points occur.

If d is not a square in k then these points do not occur and there is nothing to worry about. The affine points on $x^2 + y^2 = c^2(1 + dx^2y^2)$ form a group under the addition law; there are no failure cases. See Appendix A. The Curve25519 example is in this situation: $d = -2$ is not a square modulo $2^{255} - 19$.

4 Comparison to previous addition speeds

This section compares the speeds announced in this paper to speeds announced in the literature for previous coordinate systems. The first four tables in this section are for low-level operations such as doubling; the remaining five tables in this section are for higher-level operations relevant to various applications.

Level of detail of the comparison. We follow most of the literature in ignoring the costs of additions, subtractions, and multiplications by small constants. We recognize that these costs (and the costs of non-arithmetic operations) can be quite noticeable in practice, and we plan a more detailed cost evaluation of Edwards coordinates on various platforms along the lines of [5], but for this paper we ignore the costs.

Consider, for example, the usual Jacobian-coordinate doubling algorithm in the case $a_4 = -3$: there are 4 squarings, 4 general multiplications, 5 additions and subtractions, and 5 multiplications by the constants $2, 3, 4, 8, 8$. We summarize this cost as $4\mathbf{M} + 4\mathbf{S}$. The improved algorithm in [3, page 16] involves 5 squarings, 3 general multiplications, 8 additions and subtractions, and 4 multiplications by small constants; we summarize this cost as $3\mathbf{M} + 5\mathbf{S}$. A similar multiplication-for-squaring tradeoff is possible for addition in Jacobian and Chudnovsky coordinates, according to [3, page 17], and is used in our operation counts without further comment.

Some algorithms involve multiplications by curve parameters, such as the parameter d in Edwards coordinates. Some applications can take advantage of multiplying by a constant d, and some applications can choose curves where d is small, but other applications cannot. To cover both situations we separately tally the cost $\mathbf{d}$ of multiplying by a curve parameter; the reader can substitute $\mathbf{d} = 0$, $\mathbf{d} = \mathbf{M}$, or anything in between.

We sort our tables using the standard, but debatable, approximations $\mathbf{S} \approx 0.8\mathbf{M}$ and $\mathbf{d} \approx 0\mathbf{M}$. We do not claim that these approximations are valid for most applications. The order of entries in our tables can easily be affected by small changes in the $\mathbf{S}/\mathbf{M}$ ratio, the $\mathbf{d}/\mathbf{M}$ ratio, etc.

Low-level operations. Inversion-free doubling in the new coordinate system is faster than inversion-free doubling in previous systems (disregarding the extreme case $\mathbf{S}/\mathbf{M} = 1$):

System	Cost of doubling
Projective	$6\mathbf{M} + 5\mathbf{S} + 1\mathbf{d}$; see [13, page 281]
Hessian	$6\mathbf{M} + 3\mathbf{S}$; see [15, page 405, display 9]
Jacobi quartic	$1\mathbf{M} + 9\mathbf{S} + 3\mathbf{d}$; see [6, page 5, display 11]
Jacobian	$2\mathbf{M} + 7\mathbf{S} + 1\mathbf{d}$
Jacobian/Chudnovsky if $a_4 = -3$	$3\mathbf{M} + 5\mathbf{S}$
Jacobi intersection	$4\mathbf{M} + 3\mathbf{S}$; see [20, page 397, long display]
This paper	$3\mathbf{M} + 4\mathbf{S}$

Mixed addition (affine plus inversion-free producing inversion-free) in the new coordinate system is faster than mixed addition in previous systems if $\mathbf{d}/\mathbf{M} < 1 - \mathbf{S}/\mathbf{M}$:

System	Cost of mixed addition
Jacobi intersection	$11\mathbf{M} + 2\mathbf{S} + 1\mathbf{d}$; see [20, page 396, long display]
Projective	$9\mathbf{M} + 2\mathbf{S}$; see [13, page 281]
Jacobi quartic	$8\mathbf{M} + 3\mathbf{S} + 3\mathbf{d}$; see [6, page 6, Figure 1]
Jacobian/Chudnovsky	$8\mathbf{M} + 3\mathbf{S}$; see [13, page 284]
Hessian	$10\mathbf{M}$; see [15, Section 5, first paragraph]
This paper	$9\mathbf{M} + 1\mathbf{S} + 1\mathbf{d}$

Non-mixed addition in the new coordinate system is faster than non-mixed addition in previous systems (disregarding the extreme case $\mathbf{d}/\mathbf{M} = \mathbf{S}/\mathbf{M} = 1$):

System	Cost of non-mixed addition
Jacobian	$11\mathbf{M} + 5\mathbf{S}$
Jacobi intersection	$13\mathbf{M} + 2\mathbf{S} + 1\mathbf{d}$; see [20, page 396, long display]
Projective	$12\mathbf{M} + 2\mathbf{S}$; see [13, page 281]
Chudnovsky	$10\mathbf{M} + 4\mathbf{S}$
Jacobi quartic	$10\mathbf{M} + 3\mathbf{S} + 3\mathbf{d}$; see [6, page 6, Figure 1]
Hessian	$12\mathbf{M}$; see [15, page 407, Figure 1]
This paper	$10\mathbf{M} + 1\mathbf{S} + 1\mathbf{d}$

Similar comments apply to unified addition-or-doubling:

System	Cost of unified addition-or-doubling
Jacobian	$11\mathbf{M} + 6\mathbf{S} + 1\mathbf{d}$; see [8, page 339, bottom]
Jacobian if $a_4 = -1$	$13\mathbf{M} + 3\mathbf{S}$; see [8, page 340, top]
Jacobi intersection	$13\mathbf{M} + 2\mathbf{S} + 1\mathbf{d}$; see [20, page 396, long display]
Jacobi quartic	$10\mathbf{M} + 3\mathbf{S} + 3\mathbf{d}$; see [6, page 6, Figure 1]
Hessian	$12\mathbf{M}$; see [15, page 408]
This paper	$10\mathbf{M} + 1\mathbf{S} + 1\mathbf{d}$

Multi-scalar multiplication. The general multi-scalar multiplication problem is to compute $\sum n_i P_i$ given integers n_i and curve points P_i. Specific problems are obtained by specifying the number of points, by specifying which points are known in advance, by specifying which integers are known in advance, etc.

The classic solutions to multi-scalar multiplication problems — see [2] and [12] for surveys — are obtained by combining doublings, additions, and mixed additions in a pattern that depends on the integers n_i.

For example, the popular JSF approach to computing $n_1 P_1 + n_2 P_2$, given b-bit integers n_1, n_2 and curve points $P_1, P_2, P_1 \pm P_2$ in affine coordinates, uses about b doublings and (for average n_1, n_2) about $0.5b$ mixed additions. So we tally the cost of 1 doubling and 0.5 mixed additions:

System	Cost of 1 doubling and 0.5 mixed additions
Projective	$10.5\mathbf{M} + 6\mathbf{S} + 1\mathbf{d} \approx 15.3\mathbf{M}$
Jacobi quartic	$5\mathbf{M} + 10.5\mathbf{S} + 4.5\mathbf{d} \approx 13.4\mathbf{M}$
Hessian	$11\mathbf{M} + 3\mathbf{S} \approx 13.4\mathbf{M}$
Jacobian	$6\mathbf{M} + 8.5\mathbf{S} + 1\mathbf{d} \approx 12.8\mathbf{M}$
Jacobi intersection	$9.5\mathbf{M} + 4\mathbf{S} + 0.5\mathbf{d} \approx 12.7\mathbf{M}$
Jacobian/Chudnovsky if $a_4 = -3$	$7\mathbf{M} + 6.5\mathbf{S} \approx 12.2\mathbf{M}$
This paper	$7.5\mathbf{M} + 4.5\mathbf{S} + 0.5\mathbf{d} \approx 11.1\mathbf{M}$

As a general rule, more points — and more fixed points — produce larger ratios between the number of additions and the number of doublings. For example, the accelerated ECDSA verification method in [1, page 9] uses about $b/3$ doublings and about $b/2 = 1.5(b/3)$ mixed additions. So we tally the cost of 1 doubling and 1.5 mixed additions:

System	Cost of 1 doubling and 1.5 mixed additions
Projective	$19.5\mathbf{M} + 8\mathbf{S} + 1\mathbf{d} \approx 25.9\mathbf{M}$
Jacobi intersection	$20.5\mathbf{M} + 6\mathbf{S} + 1.5\mathbf{d} \approx 25.3\mathbf{M}$
Jacobi quartic	$13\mathbf{M} + 13.5\mathbf{S} + 7.5\mathbf{d} \approx 23.8\mathbf{M}$
Hessian	$21\mathbf{M} + 3\mathbf{S} \approx 23.4\mathbf{M}$
Jacobian	$14\mathbf{M} + 11.5\mathbf{S} + 1\mathbf{d} \approx 23.2\mathbf{M}$
Jacobian/Chudnovsky if $a_4 = -3$	$15\mathbf{M} + 9.5\mathbf{S} \approx 22.6\mathbf{M}$
This paper	$16.5\mathbf{M} + 5.5\mathbf{S} + 1.5\mathbf{d} \approx 20.9\mathbf{M}$

One can save an initial inversion at the cost of changing about half of the mixed additions to non-mixed additions; this is worthwhile on platforms where inversion is extremely expensive, such as embedded systems equipped with typical multiplication coprocessors. So we tally the cost of 1 doubling, 0.75 non-mixed additions, and 0.75 mixed additions:

System	Cost of 1 doubling, 0.75 adds, 0.75 mixed adds
Projective	$21.75\mathbf{M} + 8\mathbf{S} + 1\mathbf{d} \approx 28.15\mathbf{M}$
Jacobi intersection	$22\mathbf{M} + 6\mathbf{S} + 1.5\mathbf{d} \approx 26.8\mathbf{M}$
Jacobian	$16.25\mathbf{M} + 13\mathbf{S} + 1\mathbf{d} \approx 26.65\mathbf{M}$
Jacobian if $a_4 = -3$	$17.25\mathbf{M} + 11\mathbf{S} \approx 26.05\mathbf{M}$
Jacobi quartic	$14.5\mathbf{M} + 13.5\mathbf{S} + 7.5\mathbf{d} \approx 25.3\mathbf{M}$
Hessian	$22.5\mathbf{M} + 3\mathbf{S} \approx 24.9\mathbf{M}$
Chudnovsky if $a_4 = -3$	$16.5\mathbf{M} + 10.25\mathbf{S} \approx 24.7\mathbf{M}$
This paper	$17.25\mathbf{M} + 5.5\mathbf{S} + 1.5\mathbf{d} \approx 21.65\mathbf{M}$

Doublings become even less important as the number of scalars increases. In particular, almost all doublings disappear in the context of fixed-point scalar multiplication. For example, a b-bit fixed-point single-scalar multiplication can be computed as a 24-scalar multiplication with about $b/24$ doublings and about $15b/64 = 5.625(b/24)$ mixed additions, using a fixed “comb” table containing 90 points:

System	Cost of 1 doubling, 5.625 mixed additions
Jacobi intersection	$65.875\mathbf{M} + 14.25\mathbf{S} + 5.625\mathbf{d} \approx 77.275\mathbf{M}$
Projective	$56.625\mathbf{M} + 16.25\mathbf{S} + 1\mathbf{d} \approx 69.625\mathbf{M}$
Jacobi quartic	$46\mathbf{M} + 25.875\mathbf{S} + 19.875\mathbf{d} \approx 66.7\mathbf{M}$
Jacobian	$47\mathbf{M} + 23.875\mathbf{S} + 1\mathbf{d} \approx 66.1\mathbf{M}$
Jacobian/Chudnovsky if $a_4 = -3$	$48\mathbf{M} + 21.875\mathbf{S} \approx 65.5\mathbf{M}$
Hessian	$62.25\mathbf{M} + 3\mathbf{S} \approx 64.65\mathbf{M}$
This paper	$53.625\mathbf{M} + 9.625\mathbf{S} + 5.625\mathbf{d} \approx 61.325\mathbf{M}$

Side-channel countermeasures. The above approaches are often unacceptable for cryptographic hardware and embedded systems. Many secret bits of the integers n_i are leaked, through the pattern of doublings and mixed additions and non-mixed additions, to side-channel attacks such as simple power analysis.

One response — proposed by Liardet and Smart ([20], Jacobi intersection), Joye and Quisquater ([15], Hessian), Billet and Joye ([6], Jacobi quartic), and Brier and Joye ([8], Jacobian) — is to hide the pattern of doublings, mixed additions, and non-mixed additions by carrying out all of the operations with unified addition-or-doubling formulas. For example, b doublings and $b/2$ mixed additions would be converted into $3b/2$ unified addition-or-doubling operations:

System	Cost of 1.5 unified addition-or-doubling operations
Jacobian	$16.5\mathbf{M} + 9\mathbf{S} + 1.5\mathbf{d} \approx 23.7\mathbf{M}$
Jacobian if $a_4 = -1$	$19.5\mathbf{M} + 4.5\mathbf{S} \approx 23.1\mathbf{M}$
Jacobi intersection	$19.5\mathbf{M} + 3\mathbf{S} + 1.5\mathbf{d} \approx 21.9\mathbf{M}$
Jacobi quartic	$15\mathbf{M} + 4.5\mathbf{S} + 4.5\mathbf{d} \approx 18.6\mathbf{M}$
Hessian	$18\mathbf{M}$
This paper	$15\mathbf{M} + 1.5\mathbf{S} + 1.5\mathbf{d} \approx 16.2\mathbf{M}$

The Hamming weight in the single-scalar case, and the total number of operations in the general case, is still leaked but can be shielded at low cost in other ways. Of course, at a lower level, field operations must be individually shielded. In particular, an operation counted as **M** must be carried out by a multiplication unit whose time, power consumption, etc. do not depend on the inputs. Even if the inputs happen to be the same, and even if a faster squaring unit is available, the multiplication must not be carried out by the squaring unit. An operation counted as **S** can be carried out by a faster squaring unit whose time, power consumption, etc. do not depend on the input.

The new coordinate system can be combined with various countermeasures against differential power attacks:

- Randomized representations of scalars as addition-subtraction chains; see, e.g., [24] and [20, Section 4]. The coordinate system supports arbitrary additions and subtractions.
- Randomized scalars; see, e.g., [11, Section 5.1].
- Randomized representations of points; see, e.g., [11, Section 5.3]. Our point representation is inversion-free and can be scaled freely.
- Randomized points, for example computing nP as $n(P+Q) - nQ$; see, e.g., [11, Section 5.2].

We suggest using a combination of these countermeasures. In particular, point randomization appears to be vital to counteract Goubin-type attacks. We also comment that our formulas are unified without any dummy operations; dummy operations would allow easy fault attacks.

Differential addition chains. To finish our comparison we mention another approach to high-speed multi-scalar multiplications, namely Montgomery coordinates. These coordinates were introduced by Montgomery in [22] and have attracted interest both for their side-channel resistance (see [8, Section 4] and [16]) and for their high speed. Montgomery coordinates do not support fast addition $P, Q \mapsto P + Q$, and therefore do not fit into our previous tables, but they nevertheless support fast computation of $P - Q, P, Q \mapsto P + Q$, and therefore fast computation of arbitrary "differential addition chains." In particular, Montgomery coordinates use $5\mathbf{M} + 4\mathbf{S} + 1\mathbf{d}$ per bit of n to compute $P \mapsto nP$.

Are Edwards coordinates faster than Montgomery coordinates for single-scalar variable-point multiplication? Perhaps. Standard addition chains use a doubling for each bit of n, plus an occasional addition; Edwards coordinates thus use $3\mathbf{M} + 4\mathbf{S}$ for each bit of n, plus an occasional extra $10\mathbf{M} + 1\mathbf{S} + 1\mathbf{d}$. For typical curves, $\mathbf{d}$ is as large as $\mathbf{M}$, so Edwards coordinates are faster than Montgomery coordinates when there are fewer than about 0.25 additions per bit. For selected curves, $\mathbf{d}$ is negligible, so Edwards coordinates are faster than Montgomery coordinates when there are fewer than about 0.18 additions per bit.

What about double-scalar multiplication and beyond? The answer here is that Montgomery coordinates do not seem to provide competitive performance as the number of scalars increases, despite recent differential-addition-chain improvements in [4] and [9].

References

1. Adrian Antipa, Daniel Brown, Robert Gallant, Rob Lambert, René Struik, Scott Vanstone, *Accelerated verification of ECDSA signatures* (2005). URL: `http://www.cacr.math.uwaterloo.ca/techreports/2005/tech_reports2005.html`.
2. Roberto M. Avanzi, *The complexity of certain multi-exponentiation techniques in cryptography*, Journal of Cryptology **18** (2005), 357–373.
3. Daniel J. Bernstein, *A software implementation of NIST P-224* (2001). URL: `http://cr.yp.to/talks.html#2001.10.29`.
4. Daniel J. Bernstein, *Differential addition chains* (2006). URL: `http://cr.yp.to/papers.html#diffchain`.
5. Daniel J. Bernstein, *Curve25519: new Diffie-Hellman speed records*, in [26] (2006), 207–228. URL: `http://cr.yp.to/papers.html#curve25519`.
6. Olivier Billet, Marc Joye, *The Jacobi model of an elliptic curve and side-channel analysis* (2002). URL: `http://eprint.iacr.org/2002/125`.
7. Wieb Bosma, Hendrik W. Lenstra, Jr., *Complete systems of two addition laws for elliptic curves*, Journal of Number Theory **53** (1995), 229–240.
8. Éric Brier, Marc Joye, *Weierstrass elliptic curves and side-channel attacks*, in [23] (2002), 335–345.
9. Daniel R. L. Brown, *Multi-dimensional Montgomery ladders for elliptic curves* (2006). URL: `http://eprint.iacr.org/2006/220`.
10. Henri Cohen, Gerhard Frey (editors), *Handbook of elliptic and hyperelliptic curve cryptography*, CRC Press, 2005. ISBN 1–58488–518–1.
11. Jean-Sébastien Coron, *Resistance against differential power analysis for elliptic curve cryptosystems*, in [18] (1999), 292–302.
12. Christophe Doche, *Exponentiation*, in [10] (2005), 145–168.
13. Christophe Doche, Tanja Lange, *Arithmetic of elliptic curves*, in [10] (2005), 267–302. MR 2162729.
14. Harold M. Edwards, Jr., *A normal form for elliptic curves*, to appear, Bulletin of the American Mathematical Society (2007).

15. Marc Joye, Jean-Jacques Quisquater, *Hessian elliptic curves and side-channel attacks*, in [19] (2001), 402–410.

16. Marc Joye, Sung-Ming Yen, *The Montgomery powering ladder*, in [17] (2003), 291–302. URL: `http://www.gemplus.com/smart/rd/publications/pdf/JY03mont.pdf`.

17. Burton S. Kaliski Jr., Çetin Kaya Koç, Christof Paar (editors), *Cryptographic hardware and embedded systems — CHES 2002, 4th international workshop, Redwood Shores, CA, USA, August 13–15, 2002, revised papers*, Lecture Notes in Computer Science, 2523, Springer-Verlag, 2003. ISBN 3–540–00409–2.

18. Çetin Kaya Koç, Christof Paar (editors), *Cryptographic hardware and embedded systems, first international workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, proceedings*, Lecture Notes in Computer Science, 1717, Springer, 1999. ISBN 3–540–66646–X.

19. Çetin Kaya Koç, David Naccache, Christof Paar (editors), *Cryptographic hardware and embedded systems — CHES 2001, third international workshop, Paris, France, May 14–16, 2001, proceedings*, Lecture Notes in Computer Science, 2162, Springer, 2001. ISBN 3–540–42521–7.

20. Pierre-Yvan Liardet, Nigel P. Smart, *Preventing SPA/DPA in ECC systems using the Jacobi form*, in [19] (2001), 391–401.

21. Victor S. Miller, *Use of elliptic curves in cryptography*, in [25] (1986), 417–426. MR 88b:68040.

22. Peter L. Montgomery, *Speeding the Pollard and elliptic curve methods of factorization*, Mathematics of Computation **48** (1987), 243–264. ISSN 0025–5718. MR 88e:11130. URL: `http://links.jstor.org/sici?sici=0025-5718(198701)48:177<243:STPAEC>2.0.CO;2-3`.

23. David Naccache, 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*, Lecture Notes in Computer Science, 2274, Springer, 2002. ISBN 3–540–43168–3.

24. Elisabeth Oswald, Manfred Aigner, *Randomized addition-subtraction chains as a countermeasure against power attacks*, in [19] (2001), 39–50.

25. Hugh C. Williams (editor), *Advances in cryptology: CRYPTO '85*, Lecture Notes in Computer Science, 218, Springer, Berlin, 1986. ISBN 3–540–16463–4.

26. Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, Tal Malkin (editors), *9th international conference on theory and practice in public-key cryptography, New York, NY, USA, April 24–26, 2006, proceedings*, Lecture Notes in Computer Science, 3958, Springer, Berlin, 2006. ISBN 978–3–540–33851–2.

A Appendix: Proofs

Proof that $e \neq 1$. If $e = 1$ then $0 = 4a^3(-1)^2(1^2)^2 + 27b^2(1^2)^3 = 4a^3 + 27b^2 \neq 0$, contradiction.

Proof that $t \neq 0$. Observe that $(39e - 582)(2(e-2)(e^2+32e-32)) - (26e^2 + 808e - 1624)(3(e^2 - 16e + 16)) = 2^7 3^3$. If $t = 0$ then $-3(e^2 - 16e + 16) = at^2 = 0$ and $2(e-2)(e^2+32e-32) = bt^3 = 0$ so $2^7 3^3 = 0$ so k has characteristic 2 or 3, contradiction.

Proof that $(U,V)=((tX-e-4)/3,sY)$ satisfies $V^2=(U+e)(U^2+4U+4e)$. $V^2=(sY)^2=(t/3)^3Y^2=(tX)^3/27+at^2(tX)/27+bt^3/27=(3U+e+4)^3/27+at^2(3U+e+4)/27+bt^3/27=(3U+e+4)^3/27-3(e^2-16e+16)(3U+e+4)/27+2(e-2)(e^2+32e-32)/27=U^3+(e+4)U^2+8eU+4e^2=(U+e)(U^2+4U+4e)$.

Proof that $(x,y)=(2cV/(U^2+4U+4e),cU/(U+2e))$ satisfies $x^2+y^2=c^2(1+dx^2y^2)$. The difference $1+dx^2y^2-(x^2+y^2)/c^2$, times the nonzero quantity $(U^2+4U+4e)^2(U+2e)^2$, is $(U^2+4U+4e)^2(U+2e)^2+4dc^4V^2U^2-4V^2(U+2e)^2-U^2(U^2+4U+4e)^2=(4dc^4U^2-4(U+2e)^2)V^2+(U^2+4U+4e)^2((U+2e)^2-U^2)=(4(1-e)U^2-4(U+2e)^2)(U+e)(U^2+4U+4e)+(U^2+4U+4e)^2((U+2e)^2-U^2)=0$.

Proof that (x_3,y_3) is a point on the curve. Subtract the equation $(x_2^2+y_2^2)dx_1^2y_1^2=c^2(1+dx_2^2y_2^2)dx_1^2y_1^2$ from the equation $x_1^2+y_1^2=c^2(1+dx_1^2y_1^2)$ to see that $x_1^2+y_1^2-(x_2^2+y_2^2)dx_1^2y_1^2=c^2(1-d^2x_1^2x_2^2y_1^2y_2^2)$. Similarly $x_2^2+y_2^2-(x_1^2+y_1^2)dx_2^2y_2^2=c^2(1-d^2x_1^2x_2^2y_1^2y_2^2)$. The difference $c^2(1+dx_3^2y_3^2)-x_3^2-y_3^2$, times the nonzero quantity $c^2(1+dx_1x_2y_1y_2)^2(1-dx_1x_2y_1y_2)^2$, is $c^4(1+dx_1x_2y_1y_2)^2(1-dx_1x_2y_1y_2)^2+d(x_1y_2+y_1x_2)^2(y_1y_2-x_1x_2)^2-(x_1y_2+y_1x_2)^2(1-dx_1x_2y_1y_2)^2-(y_1y_2-x_1x_2)^2(1+dx_1x_2y_1y_2)^2=c^4(1-d^2x_1^2x_2^2y_1^2y_2^2)^2-(x_1^2+y_1^2-(x_2^2+y_2^2)dx_1^2y_1^2)(x_2^2+y_2^2-(x_1^2+y_1^2)dx_2^2y_2^2)=c^4(1-d^2x_1^2x_2^2y_1^2y_2^2)^2-c^2(1-d^2x_1^2x_2^2y_1^2y_2^2)c^2(1-d^2x_1^2x_2^2y_1^2y_2^2)=0$.

Proof that $(x_1,y_1),(x_2,y_2)\mapsto(x_3,y_3)$ corresponds to elliptic-curve addition. The hypotheses here are that

- $V_1^2=U_1^3+(e+4)U_1^2+8eU_1+4e^2$, $V_2^2=U_2^3+(e+4)U_2^2+8eU_2+4e^2$, and $V_3^2=U_3^3+(e+4)U_3^2+8eU_3+4e^2$;
- $(U_3,V_3)=(U_1,V_1)+(U_2,V_2)$ under the standard addition law;
- $(x_1,y_1)=(2cV_1/(U_1^2+4U_1+4e),cU_1/(U_1+2e))$;
- $(x_2,y_2)=(2cV_2/(U_2^2+4U_2+4e),cU_2/(U_2+2e))$; and
- $dx_1x_2y_1y_2\notin\{1,-1\}$.

The goal is to prove that $(x_3,y_3)=(2cV_3/(U_3^2+4U_3+4e),cU_3/(U_3+2e))$.

The standard addition law says that $U_3=\lambda^2-(e+4)-U_1-U_2$ and $V_3=\lambda(U_1-U_3)-V_1$. Here $\lambda=(3U_1^2+2(e+4)U_1+8e)/2V_1$ in the doubling case $(U_1,V_1)=(U_2,V_2)$, and $\lambda=(V_2-V_1)/(U_2-U_1)$ otherwise. Note that other cases in the standard addition law, such as $(U_2,V_2)=-(U_1,V_1)$, are ruled out by the hypothesis that $(U_1,V_1)+(U_2,V_2)=(U_3,V_3)$.

We fed the following commands to the Magma computer-algebra system to check the equation $(x_3,y_3)=(2cV_3/(U_3^2+4U_3+4e),cU_3/(U_3+2e))$ in the doubling case:

```
K<c,d,u1>:=FieldOfFractions(PolynomialRing(Rationals(),3));
e:=1-d*c^4; R<v1>:=PolynomialRing(K,1);
S:=quo<R|u1^3+(e+4)*u1^2+8*e*u1+4*e^2-v1^2>;
lam:=(3*u1^2+2*(e+4)*u1+8*e)/(2*v1); u2:=u1; v2:=v1;
u3:=lam^2-(e+4)-u1-u2; v3:=lam*(u1-u3)-v1;
```

```
x1:=2*c*v1/(u1^2+4*u1+4*e); y1:=c*u1/(u1+2*e);
x2:=2*c*v2/(u2^2+4*u2+4*e); y2:=c*u2/(u2+2*e);
x3:=(x1*y2+y1*x2)/(c*(1+d*x1*x2*y1*y2));
y3:=(y1*y2-x1*x2)/(c*(1-d*x1*x2*y1*y2));
S!(x3-2*c*v3/(u3^2+4*u3+4*e)); S!(y3-c*u3/(u3+2*e));
```

We then used similar commands to check the non-doubling case:

```
K<c,d,u1,u2>:=FieldOfFractions(PolynomialRing(Rationals(),4));
e:=1-d*c^4; R<v1,v2>:=PolynomialRing(K,2);
S:=quo<R|u1^3+(e+4)*u1^2+8*e*u1+4*e^2-v1^2,
         u2^3+(e+4)*u2^2+8*e*u2+4*e^2-v2^2>;
lam:=(v2-v1)/(u2-u1); u3:=lam^2-(e+4)-u1-u2; v3:=lam*(u1-u3)-v1;
x1:=2*c*v1/(u1^2+4*u1+4*e); y1:=c*u1/(u1+2*e);
x2:=2*c*v2/(u2^2+4*u2+4*e); y2:=c*u2/(u2+2*e);
x3:=(x1*y2+y1*x2)/(c*(1+d*x1*x2*y1*y2));
y3:=(y1*y2-x1*x2)/(c*(1-d*x1*x2*y1*y2));
S!(x3-2*c*v3/(u3^2+4*u3+4*e)); S!(y3-c*u3/(u3+2*e));
```

Proof that the addition law works for all pairs of points if *d* is not a square. Assume that (x_1, y_1) and (x_2, y_2) are points on the curve; i.e., that $x_1^2 + y_1^2 = c^2(1 + dx_1^2y_1^2)$ and $x_2^2 + y_2^2 = c^2(1 + dx_2^2y_2^2)$. Write $\epsilon = dx_1x_2y_1y_2$.

Suppose that $\epsilon \in \{-1, 1\}$. Then $dx_1^2y_1^2(x_2^2 + y_2^2) = c^2(dx_1^2y_1^2 + d^2x_1^2y_1^2x_2^2y_2^2) = c^2(dx_1^2y_1^2 + \epsilon^2) = c^2(1 + dx_1^2y_1^2) = x_1^2 + y_1^2$, so $(x_1 + \epsilon y_1)^2 = x_1^2 + y_1^2 + 2\epsilon x_1y_1 = dx_1^2y_1^2(x_2^2 + y_2^2) + 2x_1y_1dx_1x_2y_1y_2 = dx_1^2y_1^2(x_2^2 + 2x_2y_2 + y_2^2) = dx_1^2y_1^2(x_2 + y_2)^2$. If $x_2 + y_2 \neq 0$ then $d = ((x_1 + \epsilon y_1)/x_1y_1(x_2 + y_2))^2$, contradiction. Similarly, if $x_2 - y_2 \neq 0$ then $d = ((x_1 - \epsilon y_1)/x_1y_1(x_2 - y_2))^2$, contradiction. If both $x_2 + y_2$ and $x_2 - y_2$ are 0 then $x_2 = 0$ and $y_2 = 0$, so $c = 0$, contradiction.

Thus $\epsilon \neq 1$ and $\epsilon \neq -1$; i.e., $1 - dx_1x_2y_1y_2 \neq 0$ and $1 + dx_1x_2y_1y_2 \neq 0$.

The reader might wonder why [7, Theorem 1] ("The smallest cardinality of a complete system of addition laws on E equals two") does not force exceptional cases in the addition law for the curve $x^2 + y^2 = c^2(1 + dx^2y^2)$. The simplest answer is that [7, Theorem 1] is concerned with exceptional cases in the algebraic closure of k, whereas we are concerned with exceptional cases in k itself.