A note on López-Dahab coordinates

Tanja Lange

Information-Security and Cryptography,
Ruhr-University of Bochum,
Universitätsstr. 150,
44780 Bochum, Germany,
lange@itsc.ruhr-uni-bochum.de,
http://www.itsc.ruhr-uni-bochum.de/tanja

Abstract

López-Dahab coordinates are usually the system of choice for implementations of elliptic curves over binary fields. We give new formulas for doubling which need one squaring less and one more addition. This leads to a speed-up for binary fields in polynomial basis representation.

1 Introduction

Elliptic curves are studied for cryptographic applications as a group in which the *discrete logarithm problem* is believed to be hard. In a general cyclic group $G = \langle P \rangle$, for an element $Q \in G$ this is the problem of finding an integer n such that $Q = [n]P$, where $[n]P$ denotes the result of P added to itself n times. In protocols based on the discrete logarithm problem the most time consuming operation is the computation of scalar multiples. This is done by a double-and-add algorithm which may use some precomputations. Therefore, the most important parameter for the speed of the system is the time needed for doublings and additions. If windowing methods are applied the doubling is the much more frequent operation. We like to point out already here that on elliptic curves the negative of a point is easily determined, therefore, signed digit representations can be used reducing the storage requirements.
The main reason for using elliptic curves over finite fields is that they have shorter key sizes compared to RSA and discrete logarithms in finite fields. This is mainly relevant for small embedded devices. Such systems often profit from arithmetic in binary fields and furthermore, inversions are usually prohibitively slow. Therefore, inversion-free coordinate systems have been introduced for both even and odd characteristic. We give a new formula for doubling on elliptic curves in López-Dahab coordinates which needs 4 instead of 5 squarings. This is of interest for practical applications as already before this system offered the best performance.

We briefly recall the definitions of elliptic curves and López-Dahab coordinates. For comparison we give the published algorithms for addition and doubling. Then we present our improved doubling formulas and show their correctness.

2 Elliptic curves over finite fields

For much more material on elliptic curves we refer to [2, 5]. For background on finite fields consider [7].

An *elliptic curve E over a field* $\mathbb{K}$ denoted by $E/\mathbb{K}$ is given by the *Weierstraß equation*

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \tag{1}$$

where the coefficients $a_1, a_2, a_3, a_4, a_6 \in \mathbb{K}$ are such that for each point (x_1, y_1) with coordinates in $\overline{\mathbb{K}}$ satisfying (1), the partial derivatives $2y_1 + a_1x_1 + a_3$ and $3x_1^2 + 2a_2x_1 + a_4 - a_1y_1$ do not vanish simultaneously. The last condition says that an elliptic curve is *nonsingular*. The negative of the point $P = (x_1, y_1)$ is given by $-P = (x_1, -y_1 - a_1x_1 - a_3x_1)$. Using the projective closure, which in the case of elliptic curves is simply given by

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3,$$

one sees that there is exactly one point P_∞ on the line at infinity ($Z = 0$). It has projective coordinates $(0 : 1 : 0)$ and serves as the neutral element of the group law which state now.

Addition and doubling can be defined geometrically by the chord and tangent method which gives rise to the following formulas:

$$\begin{aligned} P \oplus Q &= (\lambda^2 + a_1\lambda - a_2 - x_1 - x_2, \lambda(x_1 - x_3) - y_1 - a_1x_3 - a_3), \text{ where} \\ \lambda &= \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq \pm Q, \\[2ex] \dfrac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3} & \text{if } P = Q. \end{cases} \end{aligned}$$

These formulas depend on the curve equation and it is obvious that to reach fast formulas many zero coefficients a_i are desirable.

By a change of variables $x \mapsto u^2x + r = x'$ and $y \mapsto u^3y + u^2sx + t = y'$ with $(u, r, s, t) \in \mathbb{K}^* \times \mathbb{K}^3$, the curve is transformed to an *isomorphic* one.

In this paper we are concerned with elliptic curves over binary fields $\mathbb{F}_{2^d}$. Supersingular curves over $\mathbb{F}_{2^d}$ can be characterized by the fact that they have no point of order 2 over the algebraic closure $\bar{\mathbb{F}}_2$ and these are exactly those curves for which $a_1 = 0$. Supersingular curves have been shown to lead to significantly weaker DL systems [4, 9].

Hence, we concentrate on $a_1 \neq 0$. One can transform the curve using

$$y \mapsto a_1^3y + \frac{a_3^2 + a_1^2a_4}{a_1^3}, \quad x \mapsto a_1^2x + \frac{a_3}{a_1}$$

followed by a division by a_1^6 to an isomorphic curve given by

$$y^2 + xy = x^3 + a_2' x^2 + a_6',$$

which is nonsingular whenever $a_6' \neq 0$.
This equation can be simplified even further if d is odd, which is the case in most applications. Then we have $\mathrm{Tr}(1) = 1$. We now use the additivity of the trace $\mathrm{Tr}(a + b) = \mathrm{Tr}(a) + \mathrm{Tr}(b)$ and consider two cases:
If $\mathrm{Tr}(a_2') = 0$ there exists a solution a to a^2+a+a_2' and thus the transformation $y \mapsto y + ax$ leads to $y^2 + xy = x^3 + a_6'$.
Otherwise, let a be a solution to $a^2 + a + a_2' + 1$ which exists as $\mathrm{Tr}(a_2' + 1) = \mathrm{Tr}(a_2') + \mathrm{Tr}(1) = 0$. In this case the transformation $y \mapsto y + ax$ leads to $y^2 + xy = x^3 + x^2 + a_6'$.
For the operation count we will always assume that $a_2 \in \{0, 1\}$ but to allow applicability of the formulas also for even d we use the general equation

$$y^2 + xy = x^3 + a_2 x^2 + a_6, \text{ for } a_2 \in \mathbb{F}_{2^d}, a_6 \in \mathbb{F}_{2^d}^*.$$

3 Coordinate systems

So far we have stated the curve in affine coordinates and briefly mentioned projective coordinates. As projective coordinates are unique up to multiplication by scalars one easily sees that addition and doubling can be described without using inversions. As a drawback much more multiplications are introduced. This effect punishes additions harder as there the coordinates need to be multiplied with the respective other Z-coordinates to reach common denominators.
If the operations steam from a scalar multiplication and the input was in affine coordinates, one can use the correspondence $(x_1, y_1) \sim (x_1 : y_1 : 1)$ assigning projective coordinates to a point in affine ones. An operation involving two types of input coordinates and also allowing a different type of output coordinates is called a *mixed operation.* If one input is in affine coordinates, mixed additions are usually much faster. This can be used in a left-to-right scalar multiplication algorithm if also the precomputed points are given in affine coordinates.
Starting from the idea of projective coordinates other inversion-free systems were introduced in which the correspondence to affine coordinates assigns weights to the coordinates. We concentrate on *López-Dahab* coordinates. Here for $Z \neq 0$ the point $(X : Y : Z)$ corresponds to the affine point $(X/Z, Y/Z^2)$ and the neutral element is given by $(1 : 0 : 0)$. The points satisfy the equation

$$Y^2 + XYZ = X^3 Z + a_2 X^2 Z^2 + a_6 Z^4. \tag{2}$$

Starting from the formulas given in [8] both addition and mixed addition were improved.
Addition
We give the addition formulas as stated in [6]. Let $P = (X_1 : Y_1 : Z_1)$, $Q =$

$(X_2 : Y_2 : Z_2)$ such that $P \neq \pm Q$ then $P \oplus Q = (X_3 : Y_3 : Z_3)$ is given by

$$\begin{aligned}
A_1 &= X_1Z_2,\ A_2 = X_2Z_1,\ C = A_1 + A_2,\\
B_1 &= A_1^2,\ B_2 = A_2^2,\ D = B_1 + B_2,\\
E_1 &= Y_1Z_2^2,\ E_2 = Y_2Z_1^2,\ F = E_1 + E_2\\
G &= CF,\ Z_3 = Z_1Z_2D,\ X_3 = A_1(E_2 + B_2) + A_2(E_1 + B_1)\\
Y_3 &= (A_1G + E_1D)D + (G + Z_3)X_3.
\end{aligned}$$

A general addition in this coordinate system takes $13M + 4S + 9A$, where M denotes a multiplication, S means a squaring and A an addition. In the original formulas one more multiplication by a_2 was used which could be neglected in our case. The savings of [6] over [8] consist in 2 squarings less on the cost of one addition, which is worthwile in the setting we consider here.

Mixed addition

Mixed addition is much faster needing only $8M+5S+8A$ and one multiplication by a_2 as shown in [1].

Let $P = (X_1 : Y_1 : 1)$, $Q = (X_2 : Y_2 : Z_2)$ such that $P \neq \pm Q$ then $P \oplus Q = (X_3 : Y_3 : Z_3)$ is given by

$$\begin{aligned}
U &= Z_2^2Y_1 + Y_2,\ S = Z_2X_1 + X_2,\ T = Z_2S,\ Z_3 = T^2,\\
V &= Z_3X_1,\ C = X_1 + Y_1,\ X_3 = U^3 + T(U + S^2 + a_2T),\\
Y_3 &= (V + X_3)(TU + Z_3) + Z_3^2C.
\end{aligned}$$

Doubling

If $P = (X_1 : Y_1 : Z_1)$ then $[2]P = (X_3 : Y_3 : Z_3)$ is given by

$$\begin{aligned}
S &= X_1^2,\ T = Z_1^2,\ Z_3 = ST,\ T = a_6T^2\\
X_3 &= S^2 + T,\ Y_3 = (Y_1^2 + a_2Z_3 + T)X_3 + TZ_3
\end{aligned} \tag{3}$$

requiring $4M + 5S + 4A$ and one multiplication by a_2.

4 Improved doubling formulas

Now we show how to save one squaring in the doubling step. We first state the doubling formulas and then show their correctness by giving the relation to (3).

Doubling

If $P = (X_1 : Y_1 : Z_1)$ then $[2]P = (X_3 : Y_3 : Z_3)$ is given by

$$\begin{aligned}
S &= X_1^2,\ U = S + Y_1,\ T = X_1Z_1,\ Z_3 = T^2, T = UT\\
X_3 &= U^2 + T + a_2Z_3,\ Y_3 = (Z_3 + T)X_3 + S^2Z_3
\end{aligned} \tag{4}$$

requiring $4M + 4S + 5A$ and one multiplication by a_2.

Proof.

We first expand the expressions in (3) and (4) and then show their equality.

For X_3 we have by (3): $X_3 = X_1^4 + a_6 Z_1^4$ which should be equal to $X_3 = (X_1^2+Y_1)^2 + (X_1^2+Y_1)X_1Z_1 + a_2X_1^2Z_1^2$ obtained by (4). In fact one has that

$$\begin{aligned} X_3 &= X_1^4 + a_6 Z^4 = X_1^4 + Y_1^2 + X_1^3 Z_1 + X_1 Y_1 Z_1 + a_2 X_1^2 Z_1^2 \\ &= (X_1^2+Y_1)^2 + (X_1^2+Y_1)X_1Z_1 + a_2 X_1^2 Z_1^2, \end{aligned}$$

where the second equality follows from the curve equation (2).
The expressions for $Z_3 = X_1^2 Z_1^2$ coincide obviously.
For Y_3 we have from (3)

$$\begin{aligned} Y_3 &= a_6 Z_1^4 Z_3 + (Y_1^2 + a_2 Z_3 + a_6 Z_1^4) X_3 \\ &= (X_3 + X_1^4) Z_3 + (Y_1^2 + a_2 Z_3 + a_6 Z_1^4) X_3 \\ &= (Z_3 + X_1^3 Z_1 + X_1 Y_1 Z_1) X_3 + X_1^4 Z_3 \\ &= (Z_3 + (X_1^2 + Y_1) X_1 Z_1) X_3 + (X_1^2)^2 Z_3 \end{aligned}$$

and the final equation equals the formula in (4).

5 Comparison

From the operation count we see that the number of multiplications remains unchanged while the number of squarings is decreased by 1 and one more addition is needed if one uses the new doubling formula.

If the field $\mathbb{F}_{2^d}$ is represented with respect to a normal basis then a squaring is *less* expensive than an addition, hence, the known formulas are more efficient in this case. The same conclusion is to be drawn for hardware implementations for which the curve is fixed and thus multiplications by the constant coefficient a_6 can be hard-coded in the design of the chip, as the new system has 4 general multiplications whereas the old system had one multiplication by a_6.

In all other situations, i. e. in the usual polynomial basis representation and for varying curves the new doubling formulas are advantageous as one squaring is more time consuming than an addition. Namely, the squaring consists of the cheap step of squaring the representing polynomial by inserting zeros in the representation and of the reduction modulo the irreducible polynomial. As this polynomial is at least a trinomial one needs at least 2 addition per squaring and more for less sparse irreducible polynomials like pentanomials. In this context we like to mention the recent preprint [3] showing that one can use a redundant polynomial representation by a trinomial of slightly larger degree if no irreducible trinomial is available.

To sum up, in the cases of practical importance our method speeds up the doubling which is the most important operation in scalar multiplication and, hence, in the cryptographic protocols.

References

[1] E. Al-Daoud, R. Mahmod, M. Rushdan, and A. Kilicman. A new addition formula for elliptic curves over $gf(2^n)$. *IEEE Trans. on Comput.*, 51(8):972–975, 2002.

[2] I. F. Blake, G. Seroussi, and N. P. Smart. *Elliptic curves in cryptography.* London Mathematical Society Lecture Note Series. 265. Cambridge University Press, 1999.

[3] C. Doche. Redundant Trinomials for Finite Fields of Characteristic 2. Cryptology ePrint Archive, Report 2004/055, 2004. `http://eprint.iacr.org/`.

[4] G. Frey and H. G. Rück. A remark concerning m-divisibility and the discrete logarithm problem in the divisor class group of curves. *Math. Comp.*, 62:865–874, 1994.

[5] D. Hankerson, A. J. Menezes, and S. Vanstone. *Guide to elliptic curve cryptography.* Springer-Verlag, Berlin, 2003.

[6] A. Higuchi and N. Takagi. A fast addition algorithm for elliptic curve arithmetic in $GF(2^n)$ using projective coordinates. *Information Processing Letters*, 76:101–103, 2000.

[7] R. Lidl and H. Niederreiter. *Finite Fields*, volume 20 of *Encyclopedia of Mathematics and its Applications.* Addison-Wesley, Reading MA, 1983.

[8] J. López and R. Dahab. Improved algorithms for elliptic curve arithmetic in $GF(2^n)$. Technical Report IC-98-39, Relatório Técnico, October 1998.

[9] A. Menezes, T. Okamoto, and S. Vanstone. Reducing ellipitc curve logarithms to a finite field. *IEEE Trans. on Inform. Theory*, 39:1639–1646, 1993.