Fast Quadrupling of a Point on Elliptic Curves Cryptography

Duc-Phong Le
Temasek Laboratories
National University of Singapore
tslld@nus.edu.sg

Abstract

Ciet et al. proposed a very elegant method for trading inversions for multiplications when computing $2P + Q$ from given points P and Q on elliptic curves of Weierstrass form. In this paper we extend their method and present a fast algorithm for computing $4P$ with only one inversion in affine coordinates. Our algorithm is faster than two repeated doublings whenever the cost of one field inversion is more expensive than the cost of four field multiplications plus three field squarings (i.e. $\mathsf{I} > 4\mathsf{M} + 4\mathsf{S}$). It saves one filed multiplication and one field squaring in comparison with Sakai-Sakurai's method. We also show that on particular curves (i.e. $a = 0$ or $b = 0$), our algorithm gains better results.

Keywords: Elliptic curve cryptography, fast arithmetic, affine coordinates.

1 Introduction

The use of elliptic curve in cryptography was suggested independently by Miler in [1] and Koblitz in [2] in 1985. Since then, Elliptic Curve Cryptography (ECC) have received a lot of attention due to the fast group law on elliptic curves and because there is no subexponential attack on the discrete logarithm problem defined over elliptic curves. Thus, it can provide the same security level as RSA (or Diffie-Hellman) but with much shorter keys. In particular, this is mainly relevant for small embedded devices.

Recently, Ciet et al. [3] introduced a fast algorithm trading one inversion for some multiplications, to compute $2P + Q$ from given points P and Q on an elliptic curve. Their algorithm is faster previous if the cost of one field inversion is more expensive than the cost of six field multiplications. This was achieved as follows: Eisentrager et al. in [4] first observed that by performing two additions $(P + (P + Q))$ instead of one doubling and then one addition when computing $2P + Q$, we can omit the y-coordinate of $(P + Q)$ and thus eliminate one field multiplication. From this observation,

Ciet et al. [3] went further and showed that the x-coordinate of the point $(P+Q)$ can also be omitted and two divisions from calculation of λ_1 and λ_2 can obtained by one inversion.

The idea of trading field inversions for field multiplications when computing $4P$ has been appeared in [5], where the authors presented a general formulas for computing 2^kP in both of affine and projective coordinates. In affine coordinates, their method requires one field inversion, $(4k+1)$ field multiplications and $(4k+1)$ field squarings. For $k=2$, the cost of their algorithm is one inversion, 9 multiplications and 9 squarings.

In this paper, due to the Ciet et al.'s method, we present new formulas for quadrupling of point on elliptic curves over finite fields of odd characteristic $p>3$ in affine coordinates. Our algorithm requires one field inversion, 8 field multiplications and 8 field squarings on general curves. Thus, the algorithm saving one field multiplication and one field squarings from the results of Sakai-Sakurai [5] is faster than two repeated doublings if the cost of one field inversion is more expensive than the cost of four field multiplications plus four field squarings.

We also present new quadrupling formulas for special curves with $a=0$ or $b=0$ which are even faster than that on general curves.

The rest of the paper is organized as follows. We briefly recall definitions elliptic curve cryptography in Section 2. Section 3 presents our algorithms. We also give some analysis in this section. The conclusion will be given in Section 4.

2 Preliminaries

In this section, we first recall some basic definitions in elliptic curve cryptography. For much more material on elliptic curve cryptography we refer to [6]. Then we review the algorithm of Ciet al et. [3] for tripling a point.

2.1 Elliptic curves over finite fields

For p prime and $p>3$, an elliptic curve defined over $\mathbb{F}_p$ in short Weierstrass form is the set of solutions (x,y) to the following equation:

$$E: y^2 = x^3 + ax + b, \tag{1}$$

together with an extra point $\mathcal{O}$ which is called the point at infinity. Where $a, b \in \mathbb{F}_p$ such that the discriminant $\Delta = -16(4a^3+27b^2)$ is nonzero.

We usually use the notation $E(\mathbb{F}_q)$ for the set of points (x,y) with coordinates in the field $\mathbb{F}_q$ together with the point $\mathcal{O}$, the identity element of the group. Points on an elliptic curve can be represented in several coordinate systems, such as affine coordinates ($\mathcal{A}$), and projective ($\mathcal{P}$) coordinates.

We usually use the notation $E(\mathbb{F}_p)$ for the set of points (x, y) with coordinates in the field $\mathbb{F}_p$. The set of points on an elliptic curve forms a group under a certain addition rule.

The set of points on an elliptic curve forms a group under a certain addition rule. The negative of the point $P = (x_1, y_1)$ is given $-P = (x_1, -y_1)$. Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be two points on elliptic curve E with $P_1 \neq -P_2$. Then the coordinates of $P_3 = P_1 + P_2 = (x_3, y_3)$ can be computed as follows:

$$x_3 = \lambda^2 - x_1 - x_2, \qquad y_3 = \lambda(x_1 - x_3) - y_1, \quad \text{where}$$

$$\lambda = \frac{3x_1^2 + a}{2y_1} \text{ if } P_1 = P_2, \quad \lambda = \frac{y_2 - y_1}{x_2 - x_1} \text{ if } P_1 \neq P_2.$$

The doubling and addition costs are $1\mathsf{I} + 2\mathsf{M} + 2\mathsf{S}$ and $1\mathsf{I} + 2\mathsf{M} + 1\mathsf{S}$ respectively in affine coordinates, where I, M and S denote field inversion, field multiplication and field squaring, respectively.

2.2 Ciet et al.'s Algorithm

In [3], the authors gave formulas to directly compute $2P + Q$ from given points P and Q on elliptic curves with one field inversion, 2 field squarings and 9 field multiplications. The algorithm is described as in Table 1.

Input: $P = (x_1, y_1) \neq \mathcal{O}, \mathcal{Q} = (\S_\in, \dagger_\in) \neq \mathcal{O}$	
Output: $T = 2P + Q = (x_4, y_4)$	
if $(x_1 = x_2)$ **then**	
if $(y_1 = y_2)$ **then return** $3P$ **else return** P	
$A \leftarrow (x_2 - x_1)^2; B = (y_2 - y_1)^2$	$2\mathsf{S}$
$d \leftarrow A(2x_1 + x_2) - B$	$1\mathsf{M}$
if $(d = 0)$ **then return** $\mathcal{O}$	
$D = (x_2 - x_1)d$; $I = D^{-1}$	$1\mathsf{I} + 1\mathsf{M}$
$\lambda_1 \leftarrow dI(y_2 - y_1)$	$2M$
$\lambda_2 \leftarrow 2y_1A(x_2 - x_1)I - \lambda_1$	$3\mathsf{M}$
$x_4 \leftarrow (\lambda_2 - \lambda_1)(\lambda_2 + \lambda_1) + x_2$; $y_4 \leftarrow \lambda_2(x_1 - x_4) - y_1$	$2\mathsf{M}$
return (x_4, y_4)	
	$1\mathsf{I} + 2\mathsf{S} + 9\mathsf{M}$

Table 1: $(2P + Q)$ algorithm, for Weierstrass elliptic curves over a prime field $GF(p)$

3 The Algorithm

Our algorithm performs a quadrupling $4P$ on an elliptic curve E in affine coordinates using only one field inversion, 8 field multiplications, and 8 field squarings.

3.1 Description of the Algorithm

Let $P = (x_1, y_1)$, we need to compute $4P = (x_4, y_4)$. Let

$$d = (3x_1^2 + a)(12x_1y_1^2 - (3x_1^2 + a)^2) - 8y_1^4. \tag{2}$$

we see that $d = 8y_1^3y_3$, where y_3 is y-coordinate of the point $2P$. The computation of the value d requires $1\mathsf{M} + 5\mathsf{S}$. If a and b are small, d can be computed by $4\mathsf{S}$ as follows:

$$d = y_1^4 + 3ax_1^4 - 6a^2x_1^2 + 18by_1^2 - 24abx_1 - a^3 - 27b^2. \tag{3}$$

Defining $D = (2y_1)d$ and $I = D^{-1}$, we have:

$$\frac{1}{2y_1} = dI \qquad \text{and} \qquad \frac{1}{2y_3} = 8y_1^4 I.$$

The algorithm works as in Table 2.

Input: $P = (x_1, y_1) \neq \mathcal{O}$	
Output: $T = 4P = (x_4, y_4)$	
if $(y_1 = 0)$ **then return** P	
$A \leftarrow x_1^2; B \leftarrow 3x_1^2 + a$	$1\mathsf{S}$
$C \leftarrow 2y_1^2; D \leftarrow C^2$	$2\mathsf{S}$
$E \leftarrow (x_1 + B)^2 - A - D$	$1\mathsf{S}$
$d \leftarrow B(3E - B^2) - 2D$	$1\mathsf{M} + 1\mathsf{S}$
if $(d = 0)$ **then return** $\mathcal{O}$	
$F \leftarrow (2y_1)d;\ I \leftarrow F^{-1}$	$1\mathsf{I} + 1\mathsf{M}$
$\lambda_1 \leftarrow dIB$	$2\mathsf{M}$
$x_3 \leftarrow \lambda_1^2 - 2x_1; y_3 \leftarrow \lambda_1(x_1 - x_3) - y_1$	$1\mathsf{M} + 1\mathsf{S}$
$H \leftarrow 3x_3^2 + a$	$1\mathsf{S}$
$\lambda_2 \leftarrow 2DIH$	$2\mathsf{M}$
$x_4 \leftarrow \lambda_2^2 - 2x_3; y_4 \leftarrow \lambda_2(x_3 - x_4) - y_3$	$1\mathsf{M} + 1\mathsf{S}$
return (x_4, y_4)	
	$1\mathsf{I} + 8\mathsf{M} + 8\mathsf{S}$

Table 2: Quadrupling algorithm of a point on elliptic curves with the short Weierstrass form

Quadrupling on curves with $b = 0$ In the case of $b = 0$, d should be set to

$$d = (x_1^2 - a)((x_1^2 + a)^2 + 4ax_1^2).$$

That is because, in this setting, $y_3 = (x_1^2 - a)((x_1^2 + a)^2 + 4ax_1^2)/(8y_1^3)$. Thus, we need $2\mathsf{M} + 2\mathsf{S}$ to compute the value of d. By performing similarly as in Table 2, the quadrupling computation of a point on an elliptic curve requires $1\mathsf{I} + 5\mathsf{S} + 9\mathsf{M}$.

Quadrupling on curves with $a = 0$ In the case of $a = 0$, the slopes $\lambda_1 = 3x_1^2/2y_1$ and $\lambda_2 = 3x_1^3/2y_3$ are particularly simple. The value d in the Eq. (2) can be replaced by

$$d = y_1^4 + 18by_1^2 - 27b^2,$$

which requires only two squarings. We save one multiplication and two squarings for computing d.

In this setting, $x_3 = \lambda_1^2 - 2x_1 = x_1(y_1^2 - 9b)/(4y_1^2)$, which will be used for computing λ_2. We have:

$$\begin{aligned}\lambda_2 =& \frac{3x_3^2}{2y_3} = \frac{3x_1^2(y_1^2 - 9b)^2}{32y_1^4y_3} \\ &= \frac{3x_1^2(y_1^4 - 18by_1^2 + 81b^2)}{2D} \\ &= \frac{3}{2}(x_1^2(y_1^4 - 18by_1^2 + 81b^2)I),\end{aligned}$$

where $D = (2y_1)d = 16y_1^4y_3$ and $I = D^{-1}$.

The following formulas compute a quadrupling in $1\mathsf{I} + 5\mathsf{S} + 6\mathsf{M}$.

3.2 Analyze

From the operation count we see that the algorithm is faster than two repeated doublings if one field inversion is more expensive than $4\mathsf{M} + 4\mathsf{S}$ on general curves. In comparison with the Sakai-Sakurai [5] algorithm, our method saves one field multiplication and two field squarings.

The advantage of a method depends on I/M ratios and S/M-ratios over prime fields. In this analysis, the ratio of a field squaring to a field multiplication is set to be $\mathsf{S} = 0.8\mathsf{M}$ as commonly used in the literature, see [7]. The I/M-ratios deeply depend on many factors such as the implementations, hardware architecture, the prime characteristic of finite fields, the size of finite fields, etc. For example, the inversion-to-multiplication (I/M) ratio is bigger than 100 on smart cards (see [8]). On workstations, for NIST-recommended elliptic curves over prime fields chosen to either be a Mersenne prime, or a Mersenne-like prime for fast modular reduction and

Input: $P = (x_1, y_1) \neq \mathcal{O}$	
Output: $T = 4P = (x_4, y_4)$	
if $(y_1 = 0)$ **then return** P	
$A \leftarrow y_1^2; B \leftarrow A^2$	$2\mathsf{S}$
$C \leftarrow 3x_1^2$	$1\mathsf{S}$
$d \leftarrow B + 18bA - 27b^2$	$-$
if $(d = 0)$ **then return** $\mathcal{O}$	
$D \leftarrow 2y_1 d;\ I \leftarrow D^{-1}$	$1\mathsf{I} + 1\mathsf{M}$
$I' = CI;\ \lambda_1 \leftarrow I'd$	$2\mathsf{M}$
$x_3 \leftarrow \lambda_1^2 - 2x_1; y_3 \leftarrow \lambda_1(x_1 - x_3) - y_1$	$1\mathsf{M} + 1\mathsf{S}$
$\lambda_2 \leftarrow \frac{I'(y_1^4 - 18by_1^2 + 81b^2)}{2}$	$1\mathsf{M}$
$x_4 \leftarrow \lambda_2^2 - 2x_3; y_4 \leftarrow \lambda_2(x_3 - x_4) - y_3$	$1\mathsf{M} + 1\mathsf{S}$
return (x_4, y_4)	
	$1\mathsf{I} + 5\mathsf{S} + 6\mathsf{M}$

Table 3: Quadrupling algorithm of a point on elliptic curves with the short Weierstrass form

multiplication, this ratio is roughly 80 (see benchmarks reported in [7]). In the cases when the Mersenne prime cannot used (e.g. pairing-based cryptography), the I/M-ratio is often reported to be 13 on 32-bits Intel processors (see benchmarks reported in [9]).

In this setting, our algorithm is better than two repeated doubling in the case $1\mathsf{I} > 7.2\mathsf{M}$.

On special elliptic curves, our algorithms are even better. For curves with $b = 0$, our algorithm requires only $1\mathsf{I} + 5\mathsf{S} + 9\mathsf{M}$, it is faster than two repeated doublings if $1\mathsf{I} > 5.8\mathsf{M}$. In particular, for curves with $a = 0$, our algorithm requiring only $1\mathsf{I} + 5\mathsf{S} + 6\mathsf{M}$ is faster than two repeated doublings if $1\mathsf{I} > 2.8\mathsf{M}$.

4 Conclusion

In this paper, we presented the fast algorithms for quadrupling of point on three forms of elliptic curves that offer a better performance than a repeated doubling if $1\mathsf{I} > 7.2\mathsf{M}$, $1\mathsf{I} > 5.8\mathsf{M}$, and $1\mathsf{I} > 2.8\mathsf{M}$, respectively. This can be helpful to speedup the scalar multiplication in elliptic curve cryptography as indicated in [5, 3].

References

[1] V. S. Miller, Use of Elliptic Curves in Cryptography, in: Lecture notes in computer sciences; 218 on Advances in cryptology—CRYPTO 85,

Springer-Verlag New York, Inc., New York, NY, USA, 1986, pp. 417–426.
URL http://portal.acm.org/citation.cfm?id=18262.25413

[2] N. Koblitz, Elliptic curve cryptosystems, Mathematics of Computation 48 (177) (1987) 203–209.

[3] M. Ciet, M. Joye, K. Lauter, P. Montgomery, Trading inversions for multiplications in elliptic curve cryptography, Designs, Codes and Cryptography 39 (2006) 189–206, 10.1007/s10623-005-3299-y.

[4] K. Eisenträger, K. Lauter, P. L. Montgomery, Fast Elliptic Curve Arithmetic and Improved Weil Pairing Evaluation, in: M. Joye (Ed.), CT-RSA, Vol. 2612 of Lecture Notes in Computer Science, Springer, 2003, pp. 343–354.

[5] Y. Sakai, K. Sakurai, Efficient scalar multiplications on elliptic curves with direct computations of several doublings, IEICE Transactions Fundamentals E84-A(1) (2001) 120–129.

[6] D. Hankerson, A. J. Menezes, S. Vanstone, Guide to Elliptic Curve Cryptography, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[7] M. Brown, D. Hankerson, J. López, A. Menezes, Software Implementation of the NIST Elliptic Curves Over Prime Fields, in: Proceedings of the 2001 Conference on Topics in Cryptology: The Cryptographer's Track at RSA, CT-RSA 2001, Springer-Verlag, 2001, pp. 250–265.
URL http://portal.acm.org/citation.cfm?id=646139.680803

[8] M. Seysen, Using an RSA Accelerator for Modular Inversion, in: J. R. Rao, B. Sunar (Eds.), Cryptographic Hardware and Embedded Systems – CHES 2005, Vol. 3659 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2005, pp. 226–236.
URL http://www.iacr.org/cryptodb/archive/2005/CHES/769/769.pdf

[9] T. Acar, K. Lauter, M. Naehrig, D. Shumow, Affine Pairings on ARM, Cryptology ePrint Archive, Report 2011/243, http://eprint.iacr.org/ (2011).