An Improvement of the Guajardo-Paar Method for Multiplication on Non-supersingular Elliptic Curves

Julio López *Ricardo Dahab*

Relatório Técnico IC–98-12

Abril de 1998

An Improvement of the Guajardo-Paar Method for Multiplication on Non-supersingular Elliptic Curves*

Julio López † Ricardo Dahab ‡

Email: {julioher, rdahab}@dcc.unicamp.br

Institute of Computing
State University of Campinas
Campinas, 13081-970 São Paulo, Brazil
January 28, 1998

Abstract

This paper presents improved formulae for computing repeated doubling points on non-supersingular elliptic curves defined over finite fields of characteristic two. These formulae, in combination with variants of the sliding-window method, lead to efficient algorithms for computing a multiple of a point on such elliptic curves. For many practical implementations of the finite field $GF(2^n)$, our formulae offer a computational advantage over Guajardo and Paar's results in [2].

Keywords: Elliptic Curve Cryptosystem, Multiplication.

1 Introduction

Computing a *multiple of an elliptic point* is one of the most fundamental operations in elliptic curve public-key cryptosystems [4]. If m is a positive integer and P is an elliptic point, a multiple of an elliptic point mP is the result of adding m copies of P. The number of field additions and doublings necessary for computing this high-level operation is an important factor in the speed up the implementations of cryptographic applications based on elliptic curves. This problem, similar to the exponentiation in Z_n, is studied in the literature in the context of exponentiation of an element from a finite cyclic group. The most efficient algorithms use strategies for decreasing the time required to multiply two elements in the group and/or reducing the number of multiplications used for the exponentiation.

*Partially supported by CNPq (grant 301172/95-1) and CAPES.

†Departamento de Ciencias de la Computación, Universidad del Valle, Cali, Colombia.

‡Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP, Brasil.

For our particular case of the group formed by the points on a non-supersingular elliptic curve defined over $GF(2^n)$, the usual method for computing mP is the binary (double-and-add) method. Several proposed generalizations of the binary method as the k-ary method, sliding-window method, signed binary window method [4], are based on computing 2^iP, which requires i inversions in the underlying field.

As the inversion is the most expensive operation in the underlying finite field, the inverse operation required for doubling a point can be eliminated by resorting to projective coordinates; however extra registers and more field multiplications are required (see [4]).

Guajardo and Paar, in a recent paper [2], introduced a new approach to point doubling on elliptic curves. The general idea is to develop efficient formulae for computing 2^iP directly from $P = (x, y)$ with only one field inversion; they presented formulae for computing $2^iP, i = 1, 2, 3, 4, 5$ on elliptic curves over the finite field $GF(2^n)$.

In this paper we have generalized and improved their formulae for computing $2^iP, i \geq 2$ (with only one field inverse) on elliptic curves defined over $GF(2^n)$. Moreover, the new formulae can be used in a larger number of field implementations of $GF(2^n)$ no covered by Guajardo's formulae.

This paper is organized as follows. In Section 2, we present a brief review of non-supersingular elliptic curves defined over the finite field $GF(2^n)$. The generalization of Guajardo's formulae is presented in Section 3. In Section 4, we analyze the improved formulae and compare them to Guajardo and Paar's results. In Section 5, we describe an efficient algorithm for multiplying elliptic points.

2 Elliptic Curves over $GF(2^n)$

A non-supersingular elliptic curve E defined over $GF(2^n)$ is a set of points $P = (x, y)$ where x and y are elements of $GF(2^n)$ that satisfy the following equation $y^2 + xy = x^3 + ax^2 + b$, where $a, b \in GF(2^n), b \neq 0$, together with an extra point $\mathcal{O}$, the point at infinity.

It is well known that the set of points E forms a commutative finite group, with $\mathcal{O}$ as the group identity, under "addition". Explicit rational formulae for the addition rule involve several arithmetic operations (adding, squaring, multiplication and inversion) in the underlying finite field. The formula for adding points, as presented by Menezes in [4], is given by: Let $P = (x_1, y_1) \in E$; then $-P = (x_1, y_1 + x_1)$. If $Q = (x_2, y_2) \in E$ and $Q \neq -P$, then $P + Q = (x_3, y_3)$, where

$$x_3 = \begin{cases} (\frac{y_1 + y_2}{x_1 + x_2})^2 + \frac{y_1 + y_2}{x_1 + x_2} + x_1 + x_2 + a, & P \neq Q, \\ x_1^2 + \frac{b}{x_1^2}, & P = Q, \end{cases} \quad (1)$$

and

$$y_3 = \begin{cases} (\frac{y_1 + y_2}{x_1 + x_2})(x_1 + x_3) + x_3 + y_1, & P \neq Q, \\ x_1^2 + (x_1 + \frac{y_1}{x_1}) \cdot x_3 + x_3, & P = Q. \end{cases} \quad (2)$$

Note that x-coordinate of doubling formula $2P$ depends only on the x-coordinate of P and the parameter b, but doubling a point requires two general field multiplications and one multiplication by a fixed constant. Schroeppel [6] improved the doubling point formula saving the extra multiplication by a fixed constant, and an extra addition, in case $a = 0$. Schroeppel's formulae are :

$$x_3 = (x_1 + \frac{y_1}{x_1})^2 + (x_1 + \frac{y_1}{x_1}) + a \tag{3}$$

$$y_3 = x_1^2 + (x_1 + \frac{y_1}{x_1}) \cdot x_3 + x_3 \tag{4}$$

We use the doubling point formula (1)-(2) for deriving direct formulae for the x-coordinate of $4P, 8P, 16P, \ldots$, and the "structure" of formulae (3) and(4) to express the new formulae. Theorem 1, in the next section, leads to an algorithm for computing $2^i P$ with only one inversion.

3 Improved Formulae

In this section we present explicit rational formulae for computing $2^i P, i \geq 2$ where $P = (x, y)$ is a point of E. We also consider the particular case when the parameter a of the elliptic curve E is zero. We obtain these formulae by repeatedly doubling P, and writing the result using only one inversion. The following theorem shows how to compute $2^i P$.

Theorem 1 *Let $P = (x, y)$ be a point of the non-supersingular elliptic curve E. Then the coordinates (x_i, y_i) of the point $2^i P, i \geq 2$, are given by*

$$x_i = (\frac{\omega_{i-1}}{\rho_{i-1}})^2 + \frac{\omega_{i-1}}{\rho_{i-1}} + a \tag{5}$$

$$y_i = (\frac{(\nu_{i-1}^2)}{\rho_{i-1}})^2 + \frac{\omega_{i-1}}{\rho_{i-1}} \cdot x_i + x_i \tag{6}$$

where

$$\tau_j = (\delta_j^2)^2 \cdot b, \tau_0 = b \tag{7}$$

$$\nu_j = (\nu_{j-1}^2)^2 + \tau_{j-1}, \quad \nu_0 = x \tag{8}$$

$$\delta_j = \rho_{j-1}^2, \quad \delta_0 = 1 \tag{9}$$

$$\omega_j = \begin{cases} \nu_j \cdot (\nu_j + \rho_{j-1} \cdot \omega_{j-1}) + \delta_j \cdot \tau_{j-1}, & a \neq 0, \\ \nu_j \cdot \omega_{j-1}^2 + \delta_j \cdot \tau_{j-1}, & a = 0, \\ \rho_0^2 + y, & j = 0. \end{cases} \tag{10}$$

$$\rho_j = \delta_j \cdot \nu_j, \quad \rho_0 = x, \quad j = 0 \ldots i-1. \tag{11}$$

Proof. By repeatedly applying formula (1), one can prove easily by induction that the x-coordinate of 2^iP is ν_i/δ_i. We prove by induction that $\omega_i/\rho_i = (x_i^2 + y_i)/x_i$. For the initial condition $i = 0$, we have $(x_0^2 + y_0)/x_0 = (x^2 + y)/x = \omega_0/\rho_0$.
Assume it is true for $i = n$. We prove it for $i = n + 1$:

$$\begin{aligned}
\frac{x_{n+1}^2 + y_{n+1}}{x_{n+1}} &= \frac{1}{x_{n+1}}\left(x_{n+1}^2 + x_n^2 + \frac{\omega_n}{\rho_n}x_{n+1} + x_{n+1} \right) \\
&= x_{n+1} + \frac{x_n^2 + x_{n+1}}{x_{n+1}} + \frac{\omega_n}{\rho_n} \\
&= \frac{\nu_{n+1}}{\delta_{n+1}} + \frac{\delta_{n+1}\nu_n^2 + \nu_{n+1}\delta_n^2}{\nu_{n+1}\delta_{n+1}} + \frac{\omega_n}{\rho_n} \\
&= \frac{1}{\rho_{n+1}}\left(\nu_{n+1}^2 + \delta_{n+1}(\nu_n^4 + \nu_{n+1}) + \nu_{n+1}\rho_n\omega_n \right) \\
&= \frac{\omega_{n+1}}{\rho_{n+1}}
\end{aligned}$$

Now we prove by induction that the right hand sides of (5) and (6) compute $(x_i, y_i) = 2^iP$. For $i = 1$, we have

$$\begin{aligned}
(\frac{\omega_0}{\rho_0})^2 + \frac{\omega_0}{\rho_0} + a &= (x + \frac{x}{y})^2 + (x + \frac{y}{x}) + a = x_1 \\
(\frac{\nu_0^2}{\rho_0})^2 + \frac{\omega_0}{\rho_0}x_1 + x_1 &= x^2 + (x + \frac{y}{x})x_1 + x_1 = y_1
\end{aligned}$$

which indeed is $2P = (x_1, y_1)$. Assuming that the statement is true for $i = n$, we have

$$\begin{aligned}
(\frac{\omega_n}{\rho_n})^2 + \frac{\omega_n}{\rho_n} + a &= (\frac{x_n^2 + y_n}{x_n})^2 + (\frac{x_n^2 + y_n}{x_n}) + a \\
&= (x_n + \frac{y_n}{x_n})^2 + (x_n + \frac{y_n}{x_n}) + a = x_{n+1}.
\end{aligned}$$

By observing that $\frac{\nu_n^2}{\rho_n} = \frac{\nu_n}{\delta_n} = x_n$, we obtain that y_{n+1} can be computed by the right hand side of formula (6). Finally, the parameter a in the non-supersingular elliptic curve E can be chosen to reduce the number of operations needed to multiply points on an elliptic curve. For the particular case $a = 0$, we obtain better formulae: From (5) we can write

$$x_i = \frac{w_{i-1}^2 + w_{i-1}\rho_{i-1} + a\rho_{i-1}^2}{\rho_{i-1}^2} = \frac{\nu_i}{\delta_i}$$

then $\nu_i = \omega_{i-1}^2 + \omega_{i-1}\rho_{i-1} + a\rho_{i-1}^2$. Therefore, we can reduce one multiplication for the expression ω_i :

$$\begin{aligned}
\omega_i &= \nu_i \cdot (\nu_i + \rho_{i-1} \cdot \omega_{i-1}) + \delta_i \cdot \tau_{i-1} \\
&= \nu_i \cdot \omega_{i-1}^2 + \delta_i \cdot \tau_{i-1}.
\end{aligned}$$

This completes the proof.

Notice however that the order of the elliptic curve group with $a = 0$ is a bit less (of security) than the order of the group for $a \neq 0$ [6].

Using Theorem 1, we immediately describe the following algorithm.

Algorithm 1 (Repeated Doubling Points)

Input : $P = (x, y) \in E, i \geq 1$
Output : $Q = 2^i P$

1. $v \leftarrow x$, $v_2 \leftarrow x^2$, $p \leftarrow x$, $w \leftarrow v_2 + y$, $d \leftarrow v_2$, $t \leftarrow b$.
2. For j from 1 to $i - 1$ do :
   2.1 $v \leftarrow v_2^2 + t$
   2.2 $v_2 \leftarrow v^2$.
   2.3 If $a \neq 0$, then $w \leftarrow v(v + pw) + dt$; otherwise, $w \leftarrow vw^2 + dt$.
   2.4 $p \leftarrow dv$.
   2.5 If $j \neq (i - 1)$, then $t \leftarrow (d^2)^2 b$, $d \leftarrow p^2$.
3. $l_1 \leftarrow p^{-1} w$, $l_2 \leftarrow p^{-1} v_2$.
4. $x_i \leftarrow l_1^2 + l_1 + a$, $y_i \leftarrow l_2^2 + l_1 x_i + x_i$.
5. Return ($Q = (x_i, y_i)$).

Note that the operation of multiplying by a sparse (or small) constant (b), in steps 2.3 and 2.5, is comparable in speed to field addition. The following corollary counts the number of (quadratic) field operations required by Algorithm 1 to compute $2^i P$.

Corollary 1 *Consider an arbitrary point $P \in E$, with an order larger than 2^i. Then the number of field operations to compute $2^i P$, by using Algorithm 1, are summarized in the following table:*

Table 1: Number of field operations for computing $2^i P$

Curve	#MULT.	#SQ.	#ADD.	#INV.
$a \neq 0$	$4i - 2$	$5i - 5$	$3i + 2$	1
$a = 0$	$3i - 1$	$6i - 6$	$2i + 2$	1

4 Complexity Comparison

Our formulae reduce the number of inversions in the finite field at the cost of multiplications. Table 2, from Corollary 1, gives the number of each basic operation in the finite field for computing the high-level operation $4P, 8P$ or $16P$. In all cases our formulae are superior to those presented by Guajardo [2], saving, for example, 5 field multiplications in the computation of $16P$.

Table 2: Comparing the new, Guajardo's and the standard doubling formulae. The standard method computes $2^i P$ by doubling P, i times.

Calculation	Method	#MUL	#SQ	#ADD	#INV
2P	Formulae (1)-(2)	2	2	4	1
	Formulae (3)-(4)	2	2	5	1
4P	New Formulae, $a = 0$	6, 5	5, 6	8, 6	1
	G. Formulae[2], $a = 0$	9,8	6	10	1
	Standard	4	4	10	2
8P	New Formulae, $a = 0$	10, 8	10, 12	11, 8	1
	G. Formulae[2], $a = 0$	14,12	11	17	1
	Standard	6	6	15	3
16P	New Formulae, $a = 0$	14, 11	15, 18	14, 10	1
	G. Formulae[2], $a = 0$	19,16	15	20	1
	Standard	8	8	20	4

Now we give conditions, on the underlying finite field, for which the new formulae outperform the standard method. Given an implementation of an arithmetic over $GF(2^n)$, consider the followings cost ratios:

$$r_1 = \frac{\text{time required for one inversion}}{\text{time required for one multiplication}}$$

$$r_2 = \frac{\text{time required for one multiplication}}{\text{time required for one squaring}}$$

These costs depend on several factors, such as the specific field representation, the field size, the algorithms (hardware or software implementation) for multiplication and inversion, and the computer architecture used. Some experimental values for r_1 and r_2 are presented in [2, 1, 7]. Table 3 shows these values:

Table 3: Experimental values for r_1 and r_2

Author	Finite Field	r_1	r_2
Harper[1]	$GF(2^{104})$	4.85	13
Schroeppel[6]	$GF(2^{155})$	2.48, 3.55	13.82-14.48
De Win [7]	$GF(2^{176})$	2.55	10.62
De Win [7]	$GF(2^{177})$	3.13	26.59
Guajardo [2]	$GF(2^{176})$	4.11	9.11

From Corollary 1 we obtain bounds between ratios r_1 and r_2 for which our algorithm for computing $2^i P$ offers a computational advantage with respect to the standard method. The following theorem gives these bounds.

Theorem 2 *Consider an implementation of the finite field $GF(2^n)$. Then our algorithm for computing 2^iP outperforms the standard method, under the following conditions:*

(i) $r_1 \geq 2 + \frac{3}{r_2}$, *in the case* $a \neq 0$.

(ii) $r_1 \geq 1 + \frac{4}{r_2}$, *in the case* $a = 0$.

For many practical implementations of $GF(2^n)$, such as those reported in [2, 1, 7] Theorem 2 applies. In De Win's case [7], for example, since $r_1 = 2.55$, we obtain an improvement of about 18% in computing $16P$, when $a = 0$ and b is sparse (or small).

5 Algorithms for Computing mP

The naive algorithm for computing a multiple of an elliptic curve point is the binary method (or double-and-add algorithm). The fastest generalizations of this method are based on recoding the exponent [4] and the sliding-window method. Koyama [3] and Schroeppel [6] present algorithms based on the signed-digit representation. Basically, all methods based on recoding the exponent present a compromise between a representation of approximately minimum *weight* (number of non-zeros entries) and the number of precomputed points generated by the size of the *window* (block of digits). Since in each iteration these algorithms process several exponent bits, we can use our formulae for speeding up the computation of mP.

Next we illustrate the application of the new formulae for the basic version of the k-ary method [2, 4]. For multipliers of length about 150-200 bits, $k = 4$ is the optimal value for this method. In this case we make use of the fact that computing $16P$ (with one inversion and extra multiplications) is more efficient than computing $16P$ by the standard method. The basic version of the k-ary algorithm consists of three phases:

(1) 2^k-adic representation of m:

$$m = \sum_{j=0}^{t} m_j b^j, \quad b = 2^k.$$

(2) Precomputation: the algorithm requires a precomputed table of multiples $\mathcal{O}, 2P, 3P, \ldots (2^k - 1)P$, obtained simply by

$$P_0 = \mathcal{O}, \; P_i = P_{i-1} + P; \; i = 1, \ldots, 2^k - 1.$$

(3) Processing blocks of k bits in one iteration : we use the following equation

$$m = b \cdot (\ldots b \cdot (b \cdot P_{m_t} + P_{m_{t-1}}) \ldots + P_{m_1}) + P_{m_0}, \quad b = 2^k.$$

Further improvements of the k-ary method are treated in [3, 5]. The description of the basic version of the k-ary method is given by the following algorithm.

Algorithm 2 (k-ary method)

Input	$P = (x, y) \in E,\ m = (m_t m_{t-1} \ldots m_1 m_0)_b,\ b = 2^k,\ k \geq 1.$
Output	$Q = mP$

1. Precomputation
   1.1 $P_0 \leftarrow \mathcal{O}$.
   1.2 For i from 1 to $2^k - 1$ do :
   $P_i \leftarrow P_{i-1} + P$.
2. $Q \leftarrow P_{m_t}$.
3. For i from $t - 1$ down to 0 do:
   3.1 $Q \leftarrow 2^k Q$.
   3.2 If $m_i \neq 0$ then $Q \leftarrow Q + P_{m_i}$.
4. Return(Q)

5.1 Theoretical Complexity Analysis

In this section we provide a theoretical complexity analysis for the k-ary method combined with Algorithm 1. For a "random multiplier" m, the expected number of non-zero digits in the 2^k-adic representation of m is about $\frac{2^k - 1}{k2^k}\lfloor \log_2 m \rfloor$. Precomputing points in the k-ary method require $2^k - 3$ additions and one doubling. Notice that step 3.1 computes $\lfloor \log_2 m \rfloor$ times the point $2^k Q$, and the expected number of additions in step 3.2 is approximately $\frac{2^k - 1}{k2^k}\lfloor \log_2 m \rfloor$. The expected number of elementary field operations for different variants of the k-ary method are summarized in Table 4. From Table 4, the 4-ary method, combined with our formulae, computes mP using $1.25 \log_2 m$ multiplications less than Guajardo and Paar's method.

Table 4 : Computational Efficiency

Method	Multiplication	Squaring	Inversion
Binary	$3 \log_2 m$	$\frac{5}{2} \log_2 m$	$\frac{3}{2} \log_2 m$
4-ary + G.-Paar	$\frac{334}{64} \log_2 m + 28$	$\frac{255}{64} \log_2 m + 15$	$\frac{31}{64} \log_2 m + 14$
4-ary + Alg.1	$\frac{254}{64} \log_2 m + 28$	$\frac{255}{64} \log_2 m + 15$	$\frac{31}{64} \log_2 m + 14$
4-ary+ Alg. 1.(a=0)	$\frac{206}{64} \log_2 m + 28$	$\frac{303}{64} \log_2 m + 15$	$\frac{31}{64} \log_2 m + 14$
SBWM	$\frac{7}{3} \log_2 m + 14$	$\frac{13}{6} \log_2 m + 8$	$\frac{7}{6} \log_2 m + 7$
SBWM + Alg. 1.	$4 \log_2 m + 14$	$\frac{13}{3} \log_2 m + 8$	$\frac{1}{3} \log_2 m + 7$

We point out that the signed binary window method (SBWM) given by Koyama [3], combined with Algorithm 1, gives the most efficient generalization of the binary method for computing mP for field implementations where Theorem 2 applies.

6 Conclusion

We have presented improved formulae for computing 2^iP. These formulae, in conjunction with variants of the sliding-window method yield efficient algorithms for computing a multiple of a point on non-supersingular elliptic curves defined over finite fields of characteristic 2. For practical implementations of the finite field $GF(2^n)$ with rate r_1 bigger than 1.5 (when $a = 0$), our formulae outperform the standard method. The new formulae can be used for accelerating several algorithms such as the signed window method, and the k-ary string-replacement representation method.

References

[1] S. Vanstone G. Harper, A. Menezes. "Public-key Cryptosystems with Very Small Key Lengths". In *Advances in Cryptology- EUROCRYPT '92*, LNCS 658, pages 163–173. Springer-Verlag, 1992.

[2] J. Guajardo and C. Paar. "Efficient Algorithms for Elliptic Curve Cryptosystems". In *CRYPTO '97, LNCS*. Springer-Verlag, 1997.

[3] K. Koyama and Y. Tsuruoka. "Speeding up Elliptic Cryptosystems by Using a Signed Binary Method ". In *Advances in Cryptology-CRYPTO'92*, LNCS 740, pages 345–357. Springer-Verlag, 1993.

[4] A. J. Menezes. *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, 1993.

[5] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[6] R. Schroeppel, H. Orman, S. O'Malley, and O. Spatscheck. "Fast Key Exchange with Elliptic Curve Systems". In *Advances in Cryptography, Crypto '95*, LNCS 963, pages 43–56. Springer-Verlag, 1995.

[7] E. De Win, A. Bosselaers, S. Vandenberghe, P. De Gersem, and J. Vandewalle. "A Fast Software Implementation for Arithmetic Operation in $GF(2^n)$.". In *Asiacrypt '96*, LNCS. Springer-Verlag, 1996.