

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

Julio López  
DCS - U of Valle A.A., Colombia  
and IC-UNICAMP, Brazil

Ricardo Dahab  
IC-UNICAMP  
Campinas, Brazil \*

## Abstract

*Calculation of multiples of elliptic points plays a central role in elliptic curve public-key cryptosystems. In this paper, we present improved formulae for computing repeated doubling points on non-supersingular elliptic curves 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 in such elliptic curves. For many practical implementations of the finite field  $GF(2^n)$ , our formulae can achieve a running time improvement of up to 25% when compared to Guajardo and Paar's formulae.*

**Keywords:** *Elliptic Curve Cryptosystem, Multiplication.*

## 1. Introduction

Elliptic Curve Cryptosystems (ECC) were independently proposed by Miller [7] and Koblitz [3], in 1985. The calculation of  $Q = mP$ , for  $P$  a point on the elliptic curve and  $m$  an integer, is the preponderant operation for the efficiency of ECCs, similarly to what modular exponentiation is to the efficiency of the RSA scheme. Therefore, reducing the number of field operations required to perform the scalar multiplication  $mP$  is crucial for efficient implementations of ECCs.

The usual method for computing  $mP$  is the binary (or double-and-add) method. This method requires  $\lfloor \frac{1}{2} \log_2 m \rfloor$  elliptic additions and  $\lfloor \log_2 m \rfloor$  elliptic doublings on average. Most of the generalizations of the binary method, based on recoding the exponent and/or precomputations, reduce the number of elliptic additions. Since in each iteration these methods compute consecutive doublings, then, for some finite fields im-

plementations, it is also possible to reduce the computation amount of doublings.

Guajardo and Paar, in a recent paper [1], developed formulas for computing consecutive doublings  $2^i P$  directly from  $P = (x, y)$  with only one field inversion; they presented formulas for computing  $2^i P, i = 2, 3, 4, 5$  on elliptic curves over the finite field  $GF(2^n)$ . In this paper we have generalized and improved their formulas for computing  $2^i P, i \geq 2$  (with only one field inverse) on elliptic curves defined over  $GF(2^n)$ . Moreover, the new formulas can be used in a larger number of field implementations of  $GF(2^n)$  not covered by Guajardo and Paar's formulas.

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 and Paar's formulas is presented in Section 3. In Section 4, we analyze the improved formulas and compare them to Guajardo and Paar's results. In Section 5, we describe an efficient algorithm for elliptic scalar multiplication.

## 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 [5], 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  $P + Q = (x_3, y_3)$ ,

---

\*Research partially supported by a PRONEX-Finep grant 107/97

where

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

and

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

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. Schroepel [8] improved the doubling point formula saving the extra multiplication by a fixed constant, and an extra addition, in case  $a = 0$ . Schroepel's formula is :

$$\begin{cases} x_3 &= (x_1 + \frac{y_1}{x_1})^2 + (x_1 + \frac{y_1}{x_1}) + a \\ y_3 &= x_1^2 + (x_1 + \frac{y_1}{x_1}) \cdot x_3 + x_3. \end{cases} \quad (3)$$

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

### 3. Improved Formulas

In this section we present explicit rational formulas for computing  $2^i P, i \geq 2$  where  $P = (x, y)$  is a point of  $E$ . We obtain these formulas 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 an elliptic point of order larger than  $2^i$ .  $E$ . Then the coordinates  $(x_i, y_i)$  of the point  $2^i P, i \geq 2$ , are given by*

$$x_i = M_{i-1}^2 + M_{i-1} + a \quad (4)$$

$$y_i = A_{i-1}^2 + M_{i-1} \cdot x_i + x_i \quad (5)$$

where  $M_{i-1} = \rho_{i-1}^{-1} \cdot \omega_{i-1}$ ,  $A_{i-1} = \rho_{i-1}^{-1} \cdot \nu_{i-1}^2$  and for  $j = 1 \dots i - 1$

$$\begin{aligned} \tau_j &= (\delta_j^2)^2 \cdot b, \quad \tau_0 = b \\ \nu_j &= (\nu_{j-1}^2)^2 + \tau_{j-1}, \quad \nu_0 = x \\ \delta_j &= \rho_{j-1}^2, \quad \delta_0 = 1 \\ \omega_j &= \nu_j \cdot (\omega_{j-1}^2 + a\rho_{j-1}^2) + \delta_j \cdot \tau_{j-1}, \quad \omega_0 = \rho_0^2 + y \\ \rho_j &= \delta_j \cdot \nu_j, \quad \rho_0 = x. \end{aligned}$$

**Proof.** By repeatedly applying formula (1), one can prove easily by induction that the  $x$ -coordinate of  $2^i P$  is  $\nu_i/\delta_i$ . We prove by induction that  $M_i = \omega_i/\rho_i$ , where  $M_i = (x_i^2 + y_i)/x_i$ . For the initial condition  $i = 0$ , we have  $M_0 = \frac{x_0^2 + y_0}{x_0} = \frac{\omega_0}{\rho_0}$ . Assume it is true for  $i = n$ . We prove it for  $i = n + 1$ :

$$\begin{aligned} M_{n+1} &= \frac{x_{n+1}^2 + y_{n+1}}{x_{n+1}} \\ &= x_{n+1} + \frac{x_n^2 + M_n x_{n+1} + x_{n+1}}{x_{n+1}} \\ &= M_{n+1}^2 + M_n + a + \frac{x_n^2}{x_{n+1}} + M_n + 1 \\ &= M_n^2 + \frac{x_n^2 + x_{n+1}}{x_{n+1}} + a. \end{aligned}$$

Since  $x_n = \frac{\nu_n}{\delta_n}$  and  $x_{n+1} = x_n^2 + \frac{b}{x_n^2}$  we obtain

$$\begin{aligned} M_{n+1} &= M_n^2 + \frac{b\delta_n^4}{\nu_{n+1}} + a \\ &= \frac{\omega_n^2}{\rho_n^2} + \frac{b\delta_n^4}{\nu_{n+1}} + a \\ &= \frac{\nu_{n+1} \cdot (\omega_n^2 + a\rho_n^2) + b\delta_n^4 \cdot \rho_n^2}{\rho_{n+1}} \\ &= \frac{\omega_{n+1}}{\rho_{n+1}}. \end{aligned}$$

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  $a \neq 0$  [8]. Using Theorem 1, we immediately describe the following algorithm.

#### Algorithm 1: Repeated Doubling Points

**Input:**  $P = (x, y) \in E, i \geq 2$

**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  $v_2 \leftarrow v^2.$

Table 1. Comparing the new, Guajardo and Paar's and the standard doubling formulas. The standard method computes  $2^i P$  by doubling  $P$ ,  $i$  times.

Calculation	Method	#MUL	#SQ	#INV
4P	New	6	6	1
	G & P	8	6	1
	Standard	4	4	2
8P	New	10	12	1
	G & P	12	11	1
	Standard	6	6	3
16P	New	14	18	1
	G & P	16	15	1
	Standard	8	8	4

$$2.3 \quad w \leftarrow v(w^2 + ap^2) + dt.$$

$$2.4 \quad p \leftarrow dv.$$

$$2.5 \quad \text{If } j \neq (i-1), \text{ then } t \leftarrow (d^2)^2 b, \quad d \leftarrow p^2.$$

$$3. \quad M \leftarrow p^{-1} \cdot w, \quad A \leftarrow p^{-1} \cdot v2.$$

$$4. \quad x_i \leftarrow M^2 + M + a, \quad y_i \leftarrow A^2 + M \cdot x_i + x_i.$$

$$5. \quad \text{Return } (Q = (x_i, y_i)).$$

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 Algorithm 1 performs  $3i - 1$  general field multiplications,  $i - 1$  multiplications by the constant  $b$ ,  $i - 1$  multiplications by the coefficient  $a$ ,  $7i - 1$  squarings and one field inversion.

## 4. Complexity Comparison

Our formulae reduce the number of inversions in the finite field at the cost of multiplications. Table 1, from Corollary 1, gives the number of each basic operation in the finite field for computing the high-level operation  $4P$ ,  $8P$  or  $16P$  (with  $a = 0$ ). In all cases our formulae are superior to those presented by Guajardo [1], saving, for example, two general field multiplications in the computation of  $16P$ . Moreover, our formulas favor sparse (or small) elliptic coefficients.

Now we give conditions, on the underlying finite field, for which the new formulas outperform the standard method. Given an implementation of an arith-

Table 2. Experimental values for  $r_1$  and  $r_2$

Author	Field	$r_1$	$r_2$
Harper[2]	$GF(2^{104})$	4.85	13
Schroep.[8]	$GF(2^{155})$	2.48-3.55	13.82-14.48
De Win [9]	$GF(2^{176})$	2.55	10.62
De Win [9]	$GF(2^{176})$	3.13	26.59
G & P [1]	$GF(2^{176})$	4.11	9.11

metic 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 [1, 2, 9]. Table 2 shows these values:

From Corollary 1 we obtain a bound 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 this bound.

**Theorem 2** Consider an implementation of the finite field  $GF(2^n)$ . Then our algorithm for computing  $2^i P$  outperforms the standard method, under the following condition:  $r_1 \geq 2 + \frac{4}{r_2}$ .

For many practical implementations of  $GF(2^n)$ , such as those reported in [1, 2, 9] Theorem 2 applies. In De Win's case [9], for example, since  $r_1 = 2.55$ , we expect an improvement of about 23% for 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 [5] and the sliding-window method. Koyama [4] and Schroepel [8] 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 formulas 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 [1, 5]. 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

$\mathcal{O}, 2P, 3P, \dots, (2^k - 1)P$ , obtained simply by

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

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

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

Further improvements of the  $k$ -ary method are treated in [4, 6]. The description of the basic version of the  $k$ -ary method is given by the following algorithm.

**Algorithm 2:  $k$ -ary method**

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

**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

Table 3. Computational Efficiency

Method	Multiplications	Inversions
Binary	$3 \log_2 m$	$\frac{3}{2} \log_2 m$
4-ary + G & P	$\frac{286}{64} \log_2 m + 28$	$\frac{31}{64} \log_2 m + 14$
4-ary+Alg. 1.	$\frac{254}{64} \log_2 m + 28$	$\frac{31}{64} \log_2 m + 14$
SBWM	$\frac{7}{3} \log_2 m + 14$	$\frac{7}{6} \log_2 m + 7$
SBWM+Alg. 1	$4 \log_2 m + 14$	$\frac{1}{3} \log_2 m + 7$

number of non-zero digits in the  $2^k$ -adic representation of  $m$  is about  $\frac{2^k - 1}{k 2^k} [\log_2 m]$ . Precomputing points in the  $k$ -ary method require  $2^k - 3$  additions and one doubling. Notice that step 3.1 computes  $[\log_2 m]$  times the point  $2^k Q$ , and the expected number of additions in step 3.2 is approximately  $\frac{2^k - 1}{k 2^k} [\log_2 m]$ . The expected number of elementary field operations for different variants of the  $k$ -ary method are summarized in Table 3. From Table 3, the 4-ary method, combined with our formulas ( $a = 0$  and  $b$  random), computes  $mP$  using  $0.5 [\log_2 m]$  (or  $1.25 [\log_2 m]$  when  $b$  is sparse) multiplications less than Guajardo and Paar’s method.

We point out that the signed binary window method (SBWM) given by Koyama [4], combined with Algorithm 1, presents the best trade-off between speedup and precomputed points among known generalizations of the binary method for field implementations where Theorem 2 applies.

**6. Conclusion**

We have presented improved formula for computing consecutive doublings ( $2^i P, i \geq 2$ ). 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 many practical implementations of the finite field  $GF(2^n)$ , our formulas outperform the standard method. The new formulas can be used for accelerating several algorithms such as the signed window method, and the  $k$ -ary string-replacement representation method.

**References**

[1] J. Guajardo and C. Paar. “Efficient Algorithms for Elliptic Curve Cryptosystems”. In *CRYPTO ’97, LNCS*, pages 343–356. Springer-Verlag, 1997.

- [2] G. Harper, A. Menezes, and S. Vanstone. "Public-key Cryptosystems with Very Small Key Lengths". In *Advances in Cryptology- EUROCRYPT '92*, LNCS 658, pages 163–173. Springer-Verlag, 1992.
- [3] N. Koblitz. "Elliptic curve cryptosystem". *Mathematics of Computation*, (48):203–209, 1987.
- [4] 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.
- [5] A. J. Menezes. *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, 1993.
- [6] A. J. Menezes, P. C. v. Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
- [7] V. Miller. "Use of elliptic curves in cryptography". In *Advances in Cryptology-Proceedings of CRYPTO'85*, Lecture Notes in Computer Science 218, pages 417–426. Springer Verlag, 1986.
- [8] R. Schroepfel, 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.
- [9] E. D. Win, A. Bosselaers, S. Vandenberghe, P. D. Gersem, and J. Vandewalle. "A Fast Software Implementation for Arithmetic Operations in  $GF(2^n)$ ". In *Asiacrypt '96*, LNCS. Springer-Verlag, 1996.