Improved Algorithms for Elliptic Curve Arithmetic in $GF(2^n)$

Julio López and Ricardo Dahab

Relatório Técnico IC–98-39

Outubro de 1998

Improved Algorithms for Elliptic Curve Arithmetic in $GF(2^n)$

Julio López*and Ricardo Dahab†

October 30, 1998

Abstract

This paper describes three contributions for efficient implementation of elliptic curve cryptosystems in $GF(2^n)$. The first is a new method for doubling an elliptic curve point, which is simpler to implement than the fastest known method, due to Schroeppel, and which favors sparse elliptic curve coefficients. The second is a generalized and improved version of the Guajardo and Paar's formulas for computing repeated doubling points. The third contribution consists of a new kind of projective coordinates that provides the fastest known arithmetic on elliptic curves. The algorithms resulting from this new formulation lead to a running time improvement for computing a scalar multiplication of about 17% over previous projective coordinate methods.

1 Introduction

Elliptic curves defined over finite fields of characteristic two have been proposed for Diffie-Hellman type cryptosystems [1]. The calculation of $Q = mP$, for P a point on the elliptic curve and m an integer, is the core operation of elliptic curve public-key cryptosystems. Therefore, reducing the number of field operations required to perform the scalar multiplication mP is crucial for efficient implementation of these cryptosystems.

In this paper we discuss efficient methods for implementing elliptic curve arithmetic. We present better results than those reported in [8, 5, 2]; our basic technique is to rewrite the elliptic operations (doubling and addition) with less costly field operations (inversions and multiplications), and replace general field multiplications by multiplications by fixed elliptic coefficients.

The first method is a new formula for doubling a point, i.e., for calculating the sum of equal points. This method is simpler to implement than Schroeppel's method [8] since it does not require a quadratic solver. If the elliptic curve coefficient b is sparse, i.e., with few 1's in its representation, thus making the multiplication by the constant b more efficient than a general field multiplication, then our new formula should lead to an improvement of up to 12% compared to Schroeppel's method [8]. We also note that our formula can be applied to composite finite fields as well.

*Research supported by a CAPES-Brazil scholarship

†Partially supported by a PRONEX-FINEP research 107/97

In [2], a new approach is introduced for accelerating the computation of repeated doubling points. This method can be viewed as computing consecutive doublings using fractional field arithmetic. We have generalized and improved the formulas presented in that paper. The new formulas can be used to speed-up variants of the sliding-window method. For field implementations where the cost-ratio of inversion to multiplication varies form 2.5 to 4 (typical values of practical software field implementations), we expect a speed-up of 7% to 22% in performing a scalar multiplication.

In [7], Schroeppel proposes an algorithm for computing repeated doubling points removing most of the general field multiplications, and favoring elliptic curves with sparse coefficients. Using his method, the computation of $2^i P, i \geq 2$ requires i field inversions, i multiplications by a fixed constant, one general field multiplication, and a quadratic solver. Since inversion is the most expensive field operation, this method is suitable for finite fields where field inversion is relatively fast. If the cost-ratio of inversion to multiplication is less than 3, this algorithm may be faster than our repeated doubling algorithm.

When field inversion is costly (e.g., for normal basis representation, the cost-ratio of inversion to multiplication is at least 7 [2, 8]), projective coordinates offer an alternative method for efficiently implementing the elliptic curve arithmetic. Based on our doubling formula, we have developed a new kind of projective coordinates which should lead to an improvement of 38% over the traditional projective arithmetic coordinates [4] and 17% on the recent projective coordinates presented in [5], for calculating a multiple of a point.

The remainder of the paper is organized as follows. Section 2 presents a brief summary of elliptic curves defined over finite fields of characteristic two. In Section 3, we present our doubling point algorithm. Based on this method, we describe an algorithm for repeated doubling points in Section 4. In Section 5, we describe the new projective coordinates.An implementation of the doubling and adding projective algorithms is given in the appendix.

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

A non-supersingular elliptic curve E over $GF(2^n)$ is defined to be the set of solutions $(x, y) \in GF(2^n) \times GF(2^n)$ to the equation,

$$y^2 + xy = x^3 + ax^2 + b \ ,$$

where a and $b \in GF(2^n), b \neq 0$, together with the point at infinity denoted by $\mathcal{O}$.

It is well known that E forms a commutative finite group, with $\mathcal{O}$ as the group identity, under the addition operation known as the "tangent and chord method". Explicit rational formulas for the addition rule involve several arithmetic operations (adding, squaring, multiplication and inversion) in the underlying finite field. In what follows, we will only be concerned with formulas for doubling a point P in affine coordinates; formulas for adding two different points in affine or projective coordinates can be found in [4, 5].

Let $P = (x_1, y_1)$ be a point of E. The doubling point formula [4] to compute $2P = (x_2, y_2)$ is given by

$$\begin{cases} x_2 &= x_1^2 + \dfrac{b}{x_1^2} \ , \\ y_2 &= x_1^2 + (x_1 + \dfrac{y_1}{x_1}) \cdot x_2 + x_2 \ . \end{cases} \tag{1}$$

Note that the x-coordinate of doubling point formula $2P$ depends only on the x-coordinate of P and the coefficient b, but doubling a point requires two general field multiplications, one multiplication by the constant b and one field inversion.

Schroeppel [6] improved the doubling point formula saving the multiplication by the constant b. His improved doubling point formula is :

$$\begin{cases} x_2 &= M^2 + M + a \ , \\ y_2 &= x_1^2 + M \cdot x_2 + x_2 \ , \\ M &= x_1 + \frac{x_1}{y_1} \ . \end{cases} \tag{2}$$

Observe that the x-coordinates of the previous doubling point formula lead to the quadratic equation for M:

$$M^2 + M + a \ = \ x_1^2 + \frac{b}{x_1^2} \ . \tag{3}$$

If we assume that the cost of multiplying by a sparse fixed constant is comparable in speed to field addition, and that solving the previous quadratic equation is faster, then we obtain another method for doubling a point with an effective cost of one general multiplication and one field inversion. A description of this method, developed by Schroeppel, can be found in [8, pp. 370-371] and [5].

In the next section, we introduce a new doubling point formula which requires also a general field multiplication, one field inversion, but does not depend on a quadratic solver.

3 A New Doubling Point Formula

Given an elliptic curve point $P = (x_1, y_1)$, the coordinates of the doubling point $2P = (x_2, y_2)$ can be calculated by the following new doubling point formula:

$$\begin{cases} x_2 &= x_1^2 + \frac{b}{x_1^2} \ , \\ y_2 &= \frac{b}{x_1^2} + a x_2 + (y_1^2 + b) \cdot (1 + \frac{b}{x_1^4}) \ . \end{cases} \tag{4}$$

To derive the above formula we transform the y-coordinate of the doubling point formula (2):

$$\begin{aligned} y_2 \ &= \ x_1^2 + (x_1 + \frac{y_1}{x_1}) \cdot x_2 + x_2 = \frac{b}{x_1^2} + (\frac{y_1^2 + b + a x_1^2}{x_1^2}) \cdot x_2 \\ &= \ \frac{b}{x_1^2} + a x_2 + \frac{y_1^2 + b}{x_1^2} \cdot (\frac{x_1^4 + b}{x_1^2}) = \frac{b}{x_1^2} + a x_2 + (y_1^2 + b) \cdot (1 + \frac{b}{x_1^4}) \ . \end{aligned}$$

3.1 Performance Analysis

We begin with the observation that our doubling formula eliminates the need for computing the field element M from formula (2), which requires either one general multiplication or a

Table 1: The number of field multiplications for computing $2^5P + Q$.

Cost-Ratio	New Formula #Mult.	Schroeppel [8] #Mult.	Improv. %
$r = 2$	19	21.5	12
$r = 2.5$	22	24.5	10
$r = 3$	25	27.5	9
$r = 4$	31	33.5	7

quadratic solver. The calculation of $2P$ requires one general field multiplication, two field multiplications by the fixed constant b, and one field multiplication by the constant a. This last multiplication can be avoided by choosing the coefficient a to be 0 or 1.[1] Thus, our formula favors elliptic curves with sparse coefficients, i.e., those having relatively few 1's in their representation.

In order to compare the running time of our formula with Schroeppel's method [8] for computing a scalar multiplication, we made the following assumptions:

- Adding and squaring field elements is fast.
- Multiplying a field element by a sparse constant is comparable to adding.
- The cost of solving the quadratic equation (3) and determining the right solution is about half of that of a field multiplication (this is true for the finite field implementation given in [6], but no efficient method is known for tower fields [7]).

The fastest methods for computing a scalar multiplication [6, 3] perform five point doublings for every point-addition, on average. Table 1 compares our formula, in performing a scalar multiplication, for different values of the cost-ratio r of inversion to multiplication.

Therefore, for practical field implementations as those given in [6, 9, 2], our formula should lead to a running time improvement of up to 12% in computing a scalar multiplication. However, for elliptic curves selected at random (where the coefficient b is not necessarily sparse), both our and Schroeppel's method may not give a computational advantage. A better algorithm for computing 2^5P is presented in the next section.

4 Repeated Doubling Algorithm

We present a method for computing repeated doublings, $2^iP, i \geq 2$, which is based on fractional field arithmetic and the doubling formula. The idea is to successively compute the elliptic points $2^jP = (x_j, y_j)$, $j = 1, 2, \ldots, i$, as triples $(\nu_j, \omega_j, \delta_j)$ of field elements, where $x_i = \frac{\nu_i}{\delta_i}$ and $y_i = \frac{\omega_i}{\delta_i^2}$. The exact formulation is given in

[1] E is isomorphic to E_1: $y^2 + xy = x^3 + \alpha x^2 + b$, where $Tr(\alpha) = Tr(a)$, $\alpha = 0$ or γ and $Tr(\gamma) = 1$ (if n is odd, we can take $\gamma = 1$), see [4, p. 39].

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

$$x_i = \frac{\nu_i}{\delta_i} \ , \tag{5}$$

$$y_i = \frac{\omega_i}{\delta_i^2} \ . \tag{6}$$

where

$$\begin{aligned}
\nu_{k+1} &= \nu_k^4 + b\delta_k^4 \ , \quad \nu_0 = x \\
\delta_{k+1} &= (\delta_k \cdot \nu_k)^2 \ , \quad \delta_0 = 1 \\
\omega_{k+1} &= b\delta_k^4 \cdot \delta_{k+1} + \nu_{k+1} \cdot (a\delta_{k+1} + \omega_k^2 + b\delta_k^4) \ , \quad \omega_0 = y, \ 0 \leq k < i.
\end{aligned}$$

Proof. We will prove by induction on i that $x_i = \frac{\nu_i}{\delta_i}$ and $y_i = \frac{\omega_i}{\delta_i^2}$. This is easily true for $i = 1$. Now assume that the statement is true for $i = n$; we prove it for $i = n + 1$:

$$\begin{aligned}
x_{n+1} &= \frac{b}{x_n^2} + x_n^2 = \frac{b\delta_n^2}{\nu_n^2} + \frac{\nu_n^2}{\delta_n^2} \\
&= \frac{b\delta_n^4 + \nu_n^4}{\nu_n^2 \cdot \delta_n^2} = \frac{\nu_{n+1}}{\delta_{n+1}} \ ;
\end{aligned}$$

similarly, for y_{n+1} we obtain:

$$\begin{aligned}
y_{n+1} &= \frac{b}{x_n^2} + a x_{n+1} + (y_n^2 + b) \cdot (1 + \frac{b}{x_n^4}) \\
&= \frac{b\delta_n^2}{\nu_n^2} + a\frac{\nu_{n+1}}{\delta_{n+1}} + (\frac{\omega_n^2}{\delta_n^4} + b) \cdot (1 + \frac{b\delta_n^4}{\nu_n^4}) \\
&= \frac{b\delta_n^4}{\delta_{n+1}} + a\frac{\nu_{n+1}}{\delta_{n+1}} + \frac{(\omega_n^2 + b\delta_n^4) \cdot \nu_{n+1}}{\delta_{n+1}^2} \\
&= \frac{\omega_{n+1}}{\delta_{n+1}^2} \ .
\end{aligned}$$

The following algorithm, based on Theorem 1, implements repeated doublings in terms of the affine coordinates of $P = (x, y)$.

Algorithm 1: Repeated doubling points

INPUT: $P = (x, y) \in E \quad i \geq 2$.
OUTPUT: $Q = 2^i P$.

Set $V \leftarrow x^2$, $D \leftarrow V$, $W \leftarrow y$, $T \leftarrow b$.
for $k = 1$ **to** $i - 1$ **do**
  Set $V \leftarrow V^2 + T$.
  Set $W \leftarrow D \cdot T + V \cdot (aD + W^2 + T)$.
  if $k \neq i - 1$ **then**
    $V \leftarrow V^2$, $D \leftarrow D^2$, $T \leftarrow bD^2$, $D \leftarrow D \cdot V$.
  fi
od
Set $D \leftarrow D \cdot V$.
Set $M \leftarrow D^{-1} \cdot (V^2 + W)$.
Set $x \leftarrow D^{-1} \cdot V^2$.
Set $x_i \leftarrow M^2 + M + a, \quad y_i \leftarrow x^2 + M \cdot x_i + x_i$.
return$(Q = (x_i, y_i))$.

Note that the correctness of this algorithm follows directly from the proof of Theorem 1 and formula (2).

Corollary 1 *Assume that P is an elliptic point of order larger than 2^i. Then Algorithm 1 performs $3i - 1$ general field multiplications, $i - 1$ multiplications by the fixed constant b, and $5i - 4$ field squarings.*

4.1 Complexity Comparison

Since Algorithm 1 cuts down the number of field inversions at the expense of more field multiplications, the computational advantage of Algorithm 1 over repeated doubling (using the standard point doubling formula (2)) depends on r, the cost-ratio of inversion to multiplication. Assuming that adding and squaring is fast, we conclude, from Corollary 1, that Algorithm 1 outperforms the computation of five consecutive doublings when $r > 2$. Table 2 shows the number of field multiplications needed for computing $2^5 P + Q$ for several methods and for different values of r. Note that the standard algorithm and Guajardo and Paar's formulas do not use the elliptic coefficient b, whereas Algorithm 1 does.

Algorithm 1 obtains its best performance for field implementations when r is at least three. If the elliptic curve is selected at random, then we expect Algorithm 1 to be up to 22% faster than the standard algorithm. For field implementations where $r < 3$, (for example [6, 9]), Schroeppel's method [7] outperforms Algorithm 1.

Table 2: Comparison of Algorithm 1 with other algorithms.

Ratio	Algorithm1		Schroeppel [7]		G.P. [2]	Standard (2)
r	b sparse	b random	b sparse	b random	b random	b random
2.5	21	25	18.5	22.5	27	27
3	22	26	21.5	25.5	28	30
3.5	23	27	24.5	28.5	29	33
4	24	28	27.5	31.5	30	36

5 A New Kind of Projective Coordinates

When field inversion in $GF(2^n)$ is relatively expensive, then it may be of computational advantage to use fractional field arithmetic to perform elliptic curve additions, as well as, doublings. This is done with the use of projective coordinates.

5.1 Basic Facts

A *projective plane* P^2 is defined to be the set of equivalence classes of triples (X, Y, Z), not all zero, where (X_1, Y_1, Z_1) and (X_2, Y_2, Z_2) are said to be equivalent if there exists $\lambda \in GF(2^n)$, $\lambda \neq 0$ such that $X_1 = \lambda X_2, Y_1 = \lambda^2 Y_2$ and $Z_1 = \lambda Z_2$. Each equivalence class is called a *projective point*. Note that if a projective point $P = (X, Y, Z)$ has nonzero Z, then P can be represented by the projective point $(x, y, 1)$, where $x = X/Z$ and $y = Y/Z^2$. Therefore, the projective plane can be identified with all points (x, y) of the ordinary (affine) plane plus the points for which $Z = 0$.

Any equation $f(x, y) = 0$ of a curve in the affine plane corresponds to an equation $F(X, Y, Z) = 0$, where F is obtained by replacing $x = X/Z, y = Y/Z^2$, and multiplying by a power of Z to clear the denominators. In particular, the *projective equation* of the affine equation $y^2 + xy = x^3 + ax^2 + b$ is given by

$$Y^2 + XYZ = X^3Z + aX^2Z^2 + bZ^4 \ .$$

If $Z = 0$ in this equation, then $Y^2 = 0$, i.e., $Y = 0$. Therefore, $(1, 0, 0)$ is the only projective point that satisfies the equation for which $Z = 0$. This point is called *the point at infinity* (denoted $\mathcal{O}$).

The resulting projective elliptic equation is

$$E = \{(x, y, z) \in P^2, y^2 + xyz = x^3z + ax^2z^2 + bz^4\} \ .$$

To convert an affine point (x, y) to a projective point, one sets $X = x$, $Y = y$, $Z = 1$. Similarly, to convert a projective point (X, Y, Z) to an affine point, we compute $x = X/Z$, $y = Y/Z^2$. The projective coordinates of the point $-P(X, Y, Z)$ are given by $-P(X, Y, Z) = (X, XZ + Y, Z)$. The algorithms for adding two projective points are given below.

5.2 Projective Elliptic Arithmetic

In this section we present new formulas for adding elliptic curve points in projective coordinates.

Projective Elliptic Doubling

The projective form of the doubling formula is

$$2(X_1, Y_1, Z_1) = (X_2, Y_2, Z_2) \ ,$$

where

$$\begin{aligned} Z_2 &= Z_1^2 \cdot X_1^2 \ , \\ X_2 &= X_1^4 + b \cdot Z_1^4 \ , \\ Y_2 &= bZ_1^4 \cdot Z_2 + X_2 \cdot (aZ_2 + Y_1^2 + bZ_1^4) \ . \end{aligned}$$

Projective Elliptic Addition

The projective form of the adding formula is

$$(X_0, Y_0, Z_0) + (X_1, Y_1, Z_1) = (X_2, Y_2, Z_2) \ ,$$

where

$$\begin{array}{lll} A_0 = Y_1 \cdot Z_0^2 \ , & D = B_0 + B_1 \ , & H = C \cdot F \ , \\ A_1 = Y_0 \cdot Z_1^2 \ , & E = Z_0 \cdot Z_1 \ , & X_2 = C^2 + H + G \ , \\ B_0 = X_1 \cdot Z_0 \ , & F = D \cdot E \ , & I = D^2 \cdot B_0 \cdot E + X_2 \ , \\ B_1 = X_0 \cdot Z_1 \ , & Z_2 = F^2 \ , & J = D^2 \cdot A_0 + X_2 \ , \\ C = A_0 + A_1 \ , & G = D^2 \cdot (F + aE^2) \ , & Y_2 = H \cdot I + Z_2 \cdot J \ . \end{array}$$

These formulas can be improved for the special case $Z_1 = 1$:

$$(X_0, Y_0, Z_0) + (X_1, Y_1, 1) = (X_2, Y_2, Z_2),$$

where

$$\begin{array}{ll} A = Y_1 \cdot Z_0^2 + Y_0 \ , & E = A \cdot C \ , \\ B = X_1 \cdot Z_0 + X_0 \ , & X_2 = A^2 + D + E \ , \\ C = Z_0 \cdot B \ , & F = X_2 + X_1 \cdot Z_2 \ , \\ D = B^2 \cdot (C + aZ_0^2) \ , & G = X_2 + Y_1 \cdot Z_2 \ , \\ Z_2 = C^2 \ , & Y_2 = E \cdot F + Z_2 \cdot G \ . \end{array}$$

Table 3: The number of field operations for $2^5P + Q$ ($a = 0$ or 1, $Z_1 = 1$)

Projective coordinates	Doubling		Adding		Cost of $2^5P + Q$	
	#Mult.	#Sqr.	#Mult.	#Sqr.	#Mult.	#Sqr.
$(x/z, y/z^2)$	4	5	9	4	29	29
$(x/z^2, x/z^3)$	5	5	10	4	35	29
$(x/z, y/z)$	7	5	12	1	47	26

5.3 Performance Analysis

The new projective doubling algorithm requires three general field multiplications, two multiplications by a fixed constant, and five squarings. Since doubling a point takes one general field multiplication less than the previous projective doubling algorithm given in [5], we obtain an improvement of about 20% for doubling a point, in general. For sparse coefficients b, we may obtain an improvement of up to a 25%.

The new projective adding algorithm requires 13 general multiplications, one multiplication by a fixed constant and six squarings. If $a = 0$ (or $a = 1$) and $Z_1 = 1$, then only nine general field multiplications and four squarings are required. Thus, we obtain one field multiplication less than the previous projective addition algorithm presented in [5]. The number of field operations required to perform an elliptic addition for various kinds of projective coordinates is listed in Table 3.

Now we can estimate the improvement of a scalar multiplication using the new projective coordinates. We will consider only the case $a = 0$ (or $a = 1$) and $Z_1 = 1$, since for this situation we obtain the best improvement. The number of field operations for computing $2^5P + Q$ is given in Table 3. Using these values we can conclude that the computation of a scalar multiplication, based on the new projective coordinates, is on average 17% and 38% faster than the previous projective coordinates [4, 5].

6 Conclusions

We have presented improved methods for faster implementation of the arithmetic of an elliptic curve defined over $GF(2^n)$. Our methods are easy to implement and can be applied to all elliptic curves defined over fields of characteristic two, independently of the specific field representation. They favor sparse elliptic coefficients but also perform well for elliptic curves selected at random. In general, they should lead to an improvement of up to 20% in the computation of scalar multiplication.

7 Acknowledgments

We thank the referees for their helpful comments.

References

[1] W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE Transactions in Informations Theory,* IT-22:644-654, November 1976.

[2] J. Guajardo and C. Paar, "Efficient Algorithms for Elliptic Curve Cryptosystems", *Advances in Cryptology, Proc. Crypto'97, LNCS 1294,* B. Kaliski, Ed., Springer-Verlag,1997,pp. 342-356.

[3] K Koyama and Y. Tsuruoka, "Speeding up elliptic cryptosystems by using a signed binary window method," *Advances in Cryptology, Proc. Crypto'92, LNCS 740,* E. Brickell, Ed., Springer-Verlag, 1993, pp. 345-357.

[4] A. Menezes, *Elliptic curve public key cryptosystems,* Kluwer Academic Publishers, 1993.

[5] IEEE P1363: Editorial Contribution to Standard for Public Key Cryptography, February 9, 1998.

[6] R. Schroeppel, H. Orman, S. O'Malley and O. Spatscheck, "Fast key exchange with elliptic curve systems," *Advances in Cryptology, Proc. Crypto'95, LNCS 963,* D. Coppersmith, Ed., Springer-Verlag, 1995, pp. 43-56.

[7] R. Schroeppel, "Faster Elliptic Calculations in $GF(2^n)$," preprint, March 6, 1998.

[8] J. Solinas, "An improved algorithm for arithmetic on a family of elliptic curves," *Advances in Cryptology, Proc. Crypto'97, LNCS 1294,* B. Kaliski, Ed., Spring-Verlag, 1997, pp. 357-371.

[9] E. De Win, A. Bosselaers, S. Vanderberghe, P. De Gersem and J. Vandewalle, "A fast software implementation for arithmetic operations in $GF(2^n)$," *Advances in Cryptology, Proc. Asiacrypt'96, LNCS 1163,* K. Kim and T. Matsumoto, Eds., Springer-Verlag, 1996, pp. 65-76.

8 Appendix

Algorithm 2: Projective Elliptic Doubling Algorithm

Input: the finite field $GF(2^m)$; the field elements a and $c = b^{2^{m-1}} (c^2 = b)$ defining a curve E over $GF(2^m)$; projective coordinates (X_1, Y_1, Z_1) for a point P_1 on E.
Output: projective coordinates (X_2, Y_2, Z_2) for the point $P_2 = 2P_1$.

1. $T_1 \leftarrow X_1$
2. $T_2 \leftarrow Y_1$
3. $T_3 \leftarrow Z_1$
4. $T_4 \leftarrow c$
5. if $T_1 = 0$ or $T_3 = 0$ then
   output $(1, 0, 0)$ and stop.
6. $T_3 \leftarrow T_3^2$
7. $T_4 \leftarrow T_3 \times T_4$
8. $T_4 \leftarrow T_4^2$
9. $T_1 \leftarrow T_1^2$
10. $T_3 \leftarrow T_1 \times T_3$ $= Z_2$
11. $T_1 \leftarrow T_1^2$
12. $T_1 \leftarrow T_4 + T_1$ $= X_2$
13. $T_2 \leftarrow T_2^2$
14. if $a \neq 0$ then
    $T_5 \leftarrow a$
    $T_5 \leftarrow T_3 \times T_5$
    $T_2 \leftarrow T_5 + T_2$
15. $T_2 \leftarrow T_4 + T_2$
16. $T_2 \leftarrow T_1 \times T_2$
17. $T_4 \leftarrow T_3 \times T_4$
18. $T_2 \leftarrow T_4 + T_2$ $= Y_2$
19. $X_2 \leftarrow T_1$
20. $Y_2 \leftarrow T_2$
21. $Z_2 \leftarrow T_3$

This algorithm requires 3 general field multiplications, 5 field squarings and 5 temporary variables. If also $a = 0$, then only 4 temporary variables are required.

Algorithm 3: Projective Elliptic Adding Algorithm

Input: the finite field $GF(2^m)$; the field elements a and b defining a curve E over $GF(2^m)$; projective coordinates (X_0, Y_0, Z_0) and $(X_1, Y_1, 1)$ for points P_0 and P_1 on E.
Output: projective coordinates (X_2, Y_2, Z_2) for the point $P_2 = P_0 + P_1$, unless $P_0 = P_1$. In this case, the triple $(0,0,0)$ is returned. (The triple (0,0,0) is not a valid projective point on the curve, but rather a marker indicating that the Doubling Algorithm should be used, see [5].)

1. $T_1 \leftarrow X_0$
2. $T_2 \leftarrow Y_0$
3. $T_3 \leftarrow Z_0$
4. $T_4 \leftarrow X_1$
5. $T_5 \leftarrow Y_1$
6. $T_6 \leftarrow T_4 \times T_3$
7. $T_1 \leftarrow T_6 + T_1$ $= B$
8. $T_6 \leftarrow T_3^2$
9. if $a \neq 0$ the
   $T_7 \leftarrow a$
   $T_7 \leftarrow T_6 \times T_7$
10. $T_6 \leftarrow T_5 \times T_6$
11. $T_2 \leftarrow T_6 + T_2$ $= A$
12. if $T_1 = 0$ then
    if $T_2 = 0$ then output $(0,0,0)$ and stop.
    else output (1,0,0) and stop.
13. $T_6 \leftarrow T_1 \times T_3$ $= C$
14. $T_1 \leftarrow T_1^2$
15. if $a \neq 0$ then
    $T_7 \leftarrow T_6 + T_7$
    $T_1 \leftarrow T_7 \times T_1$ $= D$
    else $T_1 \leftarrow T_6 \times T_1$ $= D$
16. $T_3 \leftarrow T_6^2$ $= Z_2$
17. $T_6 \leftarrow T_2 \times T_6$ $= E$
18. $T_1 \leftarrow T_6 + T_1$
19. $T_2 \leftarrow T_2^2$
20. $T_1 \leftarrow T_2 + T_1$ $= X_2$
21. $T_4 \leftarrow T_3 \times T_4$
22. $T_5 \leftarrow T_3 \times T_5$
23. $T_4 \leftarrow T_1 + T_4$ $= F$
24. $T_5 \leftarrow T_1 + T_5$ $= G$
25. $T_4 \leftarrow T_6 \times T_4$
26. $T_5 \leftarrow T_3 \times T_5$
27. $T_2 \leftarrow T_4 + T_5$ $= Y_2$
28. $X_2 \leftarrow T_1$
29. $Y_2 \leftarrow T_2$
30. $Z_2 \leftarrow T_3$

This algorithm requires 9 general field multiplications, 4 field squarings and 7 temporary variables. If also $a = 0$, then only 6 temporary variables are required.