A New Addition Formula for Elliptic Curves over $\mathrm{GF}(2^n)$

Essam Al-Daoud, Ramlan Mahmod, Mohammad Rushdan, and Adem Kilicman

Abstract—In this paper, we propose a new addition formula in projective coordinates for elliptic curves over $GF(2^n)$. The new formula speeds up the elliptic curve scalar multiplication by reducing the number of field multiplications. This was achieved by rewriting the elliptic curve addition formula. The complexity analysis shows that the new addition formula speeds up the addition in projective coordinates by about 10-12 percent, which leads to enhanced scalar multiplication methods for random and Koblitz curves.

Index Terms—Public-key cryptography, elliptic curves, point addition, projective coordinates, scalar multiplication.

1 Introduction

THE main advantage of using the finite group of elliptic curve (EC) is that its discrete logarithm problem is believed to be harder than the discrete logarithm problem for the multiplication group of a finite field. There is no known subexponential algorithm that can be applied to the elliptic curve discrete logarithm problem. Another advantage that makes elliptic curves more attractive is the possibility of optimizing the arithmetic operations in the underlying field. This has led to the appearance of several elliptic curve cryptography products such as Security Builder, SSL Plus, WTLS Plus, TrustPoint, etc. In addition, many companies have purchased licenses to use EC codes and embed them in their products [1].

In this paper, we speed up the scalar multiplication $Q = kP$ by reducing the number of underlying field multiplications for each elliptic curve addition operation in projective coordinates.

The remainder of this paper is organized as follows: Section 2 reviews the previous elliptic curve formulas. Section 3 introduces the new addition formula where the number of field multiplications has been reduced from 10 to nine in the projective coordinates $(X/Z, Y/Z^2)$. Section 4 gives a brief review of scalar multiplication methods for elliptic curves and discusses the applicability of the new addition formula to these methods. Section 5 compares the scalar multiplication methods using different formulae for random and Koblitz curves.

2 Adding Two Points on Elliptic Curve over $\mathrm{GF}(2^n)$

A non-supersingular elliptic curve E defined over $GF(2^n)$ is an equation:

$$y^2 + xy = x^3 + a_2x^2 + a_6,$$

where $a_2, a_6 \in GF(2^n)$ and $a_6 \neq 0$. Assume $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$, and $P_1 \neq -P_2$. The sum $P_3 = (x_3, y_3) = P_1 + P_2$ is computed as follows [2]: If $P_1 \neq P_2$,

$$\begin{aligned} \lambda &= \frac{y_2 + y_1}{x_2 + x_1}, \\ x_3 &= \lambda^2 + \lambda + x_1 + x_2 + a_2, \\ y_3 &= \lambda(x_1 + x_3) + x_3 + y_1. \end{aligned} \tag{1}$$

If $P_1 = P_2$,

$$\begin{aligned} \lambda &= \frac{y_1}{x_1} + x_1, \\ x_3 &= \lambda^2 + \lambda + a_2, \\ y_3 &= \lambda(x_1 + x_3) + x_3 + y_1. \end{aligned} \tag{2}$$

In either case, the computation requires two general multiplications, a squaring, and a field inversion, denoted by $2M + 1S + 1I$ [2]. Actual implementation of the elliptic curves indicates that the squaring and field addition are much faster than the field multiplication, while field inversion is more expensive than the multiplication (inversion based on Fermat's theorem requires at least seven multiplications in $GF(2^n)$ if $n \geq 128$ [3]).Therefore, the projective coordinates have been suggested to replace the inverse operation by multiplications, where a projective point (X, Y, Z), $Z \neq 0$, maps to the affine point $(X/Z^2, Y/Z^3)$. The sum $P_3 = (X_3, Y_3, Z_3) = P_1 + P_2$ in projective coordinates is computed as follows [2], [4]: If $P_1 \neq P_2$,

$$\begin{aligned} &\mathrm{U}_1 = \mathrm{X}_1\mathrm{Z}_2^2, \mathrm{S}_1 = \mathrm{Y}_1\mathrm{Z}_2^3, \mathrm{U}_2 = \mathrm{X}_2\mathrm{Z}_1^2, \mathrm{S}_2 = \mathrm{Y}_2\mathrm{Z}_1^3, \mathrm{W} = \mathrm{U}_1 + \mathrm{U}_2, \\ &\mathrm{R} = \mathrm{S}_1 + \mathrm{S}_2 \\ &\mathrm{L} = \mathrm{Z}_1\mathrm{W}, \mathrm{V} = R\mathrm{X}_2 + \mathrm{L}\mathrm{Y}_2, \mathrm{Z}_3 = L\mathrm{Z}_2, \mathrm{T} = \mathrm{R} + \mathrm{Z}_3, \\ &\mathrm{X}_3 = a_2\mathrm{Z}_3^2 + \mathrm{RT} + \mathrm{W}^3 \\ &\mathrm{Y}_3 = T\mathrm{X}_3 + \mathrm{VL}^2. \end{aligned} \tag{3}$$

If $P_1 = P_2$,

$$\begin{aligned} &Z_3 = X_1Z_1^2, X_3 = (X_1 + \sqrt[4]{a_6}Z_1^2)^4, U = Z_3 + X_1^2 + Y_1Z_1, \\ &Y_3 = X_1^4Z_3 + UX_3. \end{aligned} \tag{4}$$

Observe that (3) (point addition) requires five field squarings, 15 general field multiplications, and nine temporary variables. In the case of $Z_1 = 1$ and $a_2 = 0$ or 1, only four field squarings, 10 general field multiplications, and eight temporary variables are required, while (4) (point doubling) requires five field squarings, five general field multiplications, and four temporary variables.

López and Dahab introduced new formulae in projective coordinates, where a projective point (X, Y, Z), $Z \neq 0$, maps to the affine point $(X/Z, Y/Z^2)$. The sum $P_3 = (X_3, Y_3, Z_3) = P_1 + P_2$ is computed as follows [5]: If $P_1 \neq P_2$ and $Z_1 = 1$,

$$\begin{aligned} &U = Z_2^2Y_1 + Y_2, S = Z_2X_1 + X_2, T = Z_2S, R = S^2(T + a_2Z_2^2), \\ &Z_3 = T^2, Q = UT, X_3 = U^2 + R + Q, V = X_3 + X_1Z_3, \\ &W = X_3 + Y_1Z_3, Y_3 = QV + Z_3W. \end{aligned} \tag{5}$$

If $P_1 = P_2$,

$$\begin{aligned} Z_3 &= Z_1^2X_1^2, X_3 = X_1^4 + a_6Z_1^4 \\ Y_3 &= a_6Z_1^4Z_3 + X_3(a_2Z_3 + Y_1^2 + a_6Z_1^4). \end{aligned} \tag{6}$$

Formula (5) (point addition) requires four field squarings and 10 general field multiplications. In the case of $a_2 = 0$ or 1, only nine general field multiplications are required. Formula (6) (point doubling) requires five field squarings and five general field multiplications. If $a_2 = 0$ or 1 in (6), it would then require four general field multiplications. Note that there is no known security threat if a_2 is restricted to 0 or 1.

• *The authors are with the University Putra Malaysia, 43400 Serdang, selangor, Malaysia. E-mail: essamdz@hotmail.com.*

Manuscript received 23 Jan. 2001; revised 24 Aug. 2001; accepted 28 Dec. 2001.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 113520.

0018-9340/02/$17.00 © 2002 IEEE

3 A New Addition Formula

Theorem 1. *Let* $P_1 = (X_1/Z_1, Y_1/Z_1^2)$ *and* $P_2 = (X_2/Z_2, Y_2/Z_2^2)$ *be two points on the elliptic curve* E. *If* $Z_1 = 1$, *then the addition formula is* $P_1 + P_2 = (X_3/Z_3, Y_3/Z_3^2)$, *where*

$$\begin{aligned} &U = Z_2^2 Y_1 + Y_2, S = Z_2 X_1 + X_2, T = Z_2 S, Z_3 = T^2, \\ &V = Z_3 X_1, X_3 = U^2 + T(U + S^2 + T a_2), \\ &Y_3 = (V + X_3)(TU + Z_3) + Z_3^2(Y_1 + X_1). \end{aligned} \tag{7}$$

Proof. We will prove that the previous formula will lead to the same elliptic curve point if the standard addition formula of affine coordinates is used.

Assume that $X_1/Z_1 = x_1$, $Y_1/Z_1^2 = y_1$, $X_2/Z_2 = x_2$, $Y_2/Z_2^2 = y_2$, and $Z_1 = 1$, we will show that $X_3/Z_3 = x_3$ and $Y_3/Z_3^2 = y_3$, where (x_3, y_3) is generated by (x_1, y_1) and (x_2, y_2) by using the standard addition formula of affine coordinates.

$$\begin{aligned} \frac{X_3}{Z_3} &= \frac{U^2 + T(U + S^2 + T a_2)}{T^2} \\ &= \frac{U^2}{Z_2^2 S^2} + \frac{U}{Z_2 S} + \frac{S}{Z_2} + a_2 \\ &= \frac{(Y_1 + Y_2/Z_2^2)^2}{(X_1 + X_2/Z_2)^2} + \frac{(Y_1 + Y_2/Z_2^2)}{(X_1 + X_2/Z_2)} + \frac{X_2}{Z_2} + X_1 + a_2 \\ &= \left(\frac{y_1 + y_2}{x_1 + x_2}\right)^2 + \left(\frac{y_1 + y_2}{x_1 + x_2}\right) + x_1 + x_2 + a_2 = x_3 \\ \frac{Y_3}{Z_3^2} &= \frac{(V + X_3)(TU + Z_3) + Z_3^2(Y_1 + X_1)}{T^4} \\ &= \frac{UT(Z_3 X_1 + X_3) + Z_3(Z_3 Y_1 + X_3)}{T^4} \\ &= \frac{(Z_2^2 Y_1 + Y_2)(Z_3 X_1 + X_3)}{T^3} + \frac{X_3}{T^2} + Y_1 \\ &= \frac{(Y_1 + Y_2/Z_2^2)}{(X_1 + X_2/Z_2)}(X_1 + X_3/Z_3) + X_3/Z_3 + Y_1 \\ &= \left(\frac{y_1 + y_2}{x_1 + x_2}\right)(x_1 + x_3) + x_3 + y_1 = y_3. \end{aligned}$$

In (7), the number of field multiplications is reduced from 10 to nine and the number of squarings is increased from four to five as compared to (5). However, the squaring is very cheap in $GF(2^n)$ and, therefore, negligible. Hence, the implementation of the elliptic curve addition operation will be improved by 10 percent. If $a_2 = 0$ or 1, then eight general field multiplications are required. It follows that the improvement is about 12 percent. □

4 Scalar Multiplication

This section gives a brief review of methods computing kP, where P is a point on the elliptic curve E over $GF(2^n)$ and k is an integer. This operation is called point multiplication or scalar multiplication. For cryptography purposes, we will assume that P has a prime order $r \approx 2^n$ and $1 \le k \le r - 1$.

Algorithm 1 introduces the binary method, which is considered the simplest (and oldest) algorithm for scalar multiplication [6].

Algorithm 1: (Left to right) The binary method for scalar multiplication.

Input: $k = (k_{t-1}, \ldots, k_1, k_0)_2$ and P on the elliptic curve E over $GF(2^n)$

Output: $Q = kP$

1. $Q \leftarrow O$.
2. For i from $t - 1$ down to 0 do
   2.1. $Q \leftarrow 2Q$.
   2.2. If $k_i = 1$, then $Q \leftarrow Q + P$
3. Return Q

The expected number of elliptic curve additions in Algorithm 1 is approximately $0.5n$ and the number of doublings is approximately n, denoted by $0.5nA + nD$. The number of elliptic curve additions can be reduced by replacing the binary representation of k with a representation that has fewer nonzero coefficients. Non-adjacent form (NAF) of an integer k can be used to reduce the non-zero coefficients to nearly $n/3$. It follows that the expected running time of scalar multiplication is approximately $n/3A + nD$ [6].

If extra memory is available, then a width-w NAF can be used to reduce the number of non-zero coefficients. A width-w NAF of an integer k is an expression of the form $k = \sum_{i=0}^{l-1} k_i 2^i$, where l is at most one bit longer than the binary representation of k, $|k_i| < 2^{w-1}$; it is denoted by $\mathrm{NAF}_w(k)$. The number of non -zero coefficients in $\mathrm{NAF}_w(k)$ is approximately $n/(w+1)$. $\mathrm{NAF}_w(k)$ can be efficiently computed using Algorithm 2 [6].

Algorithm 2: Computing the $\mathrm{NAF}_w(k)$

Input: A positive integer k

Output: $\mathrm{NAF}_w(k)$

1. $i \leftarrow 0$.
2. While $k \ge 1$ do
   2.1. If k is odd, then $k_i \leftarrow k \bmod 2^w$.
      2.1.1. If $k_i > 2^{w-1} - 1$, then $k_i \leftarrow k_i - 2^w$.
      2.1.2. $k \leftarrow k - k_i$.
   2.2. Else $k_i \leftarrow 0$.
   2.3. $k \leftarrow k/2$, $i \leftarrow i + 1$.
3. Return $(k_{i-1}, k_{i-2}, \ldots, k_1, k_0)$.

The expected running time for scalar multiplication using $\mathrm{NAF}_w(k)$ (Algorithm 3) is approximately $1D + (2^{w-2} - 1)A + n/(w+1)A + nD$ [6].

Algorithm 3: Window NAF method for scalar multiplication.

Input: Window width w, $\mathrm{NAF}_w(k) = \sum_{i=0}^{l-1} k_i 2^1$ and P on E over $GF(2^n)$.

Output: $Q = kP$.

1. Precomputation. Compute $P_i = iP$, for $i \in \{1, 3, \ldots, 2^{w-1} - 1\}$.
2. $Q \leftarrow O$.
3. For i from $l - 1$ down to 0 do
   3.1. $Q \leftarrow 2Q$.
   3.2. If $k_i \ne 0$, then:
      If $k_i > 1$, then $Q \le Q + P_{k_i}$;
      Else $Q \leftarrow Q - P_{k_i}$
4. Return Q.

If $w = 4$, then Step 1 in Algorithm 3 needs $1D + 3A$, where doublings and additions in this step should be computed using affine coordinates formulae. The new addition formula can be used in Step 3.2 because all the points P_i, $i \in \{1, 3, \ldots, 2^{w-1} - 1\}$, are stored in the form $(x, y, 1)$ and $-(x, y, 1) = (x, x + y, 1)$.

If P is on a Koblitz curve (there are two Koblitz curves: $E_0 : y^2 + xy = x^3 + 1$ and $E1 : y^2 + xy = x^3 + x^2 + 1$), then width-w TNAF is a suitable representation to compute kP [7]. Window TNAF for scalar multiplications on Koblitz curves is analogous to Algorithm 3 and has an expected running time of approximately $(2^{w-2} - 1 + n/(w+1))A$ (for more information see [7]).

Unfortunately, the addition formulae (3), (5), and (7) do not seem to be applicable to two recent methods, namely, Montgomery and point halving. The Montgomery method is based on the observation that the x-coordinate of the sum of two points whose

TABLE 1
Experimental Values for r_1

Author	Finite Field	r_1
Schroeppel [10]	GF(2^{155})	2.48, 3.55
De Win [11]	GF(2^{177})	3.13
Hankerson [6]	GF(2^{163}), GF(2^{233}), GF(2^{283}).	10
Weimerskirch [12]	GF(2^{163})	16.17

TABLE 2
Approximate Cost of Point Addition Formulae

Coordinates	*Formula*	Field operation		
		M	***I***	***Total (M+10I)***
affine	1	2	1	12
projective	3	10	0	10
projective	5	9	0	9
projective	7	8	0	8

TABLE 3
Rough Estimates of Scalar Multiplication Costs for n = 163

Method	Coordinates	*Formulas*	*Points Stored*	EC operation		Field operation		
				A	***D***	***M***	***I***	***Total M+10I***
Binary method	affine	*1, 2*	*0*	*82*	*162*	*488*	*244*	*2928*
	projective	*5, 6*	*0*	*82*	*162*	*1386*	*1*	*1396*
	projective	*6, 7*	*0*	*82*	*162*	*1304*	*1*	*1314*
Window NAF w = 2	affine	*1, 2*	*0*	*54*	*162*	*432*	*216*	*2592*
	projective	*5, 6*	*0*	*54*	*162*	*1136*	*1*	*1146*
	projective	*6, 7*	*0*	*54*	*162*	*1082*	*1*	*1092*
Window NAF w = 4	affine	*1, 2*	*3*	*36*	*163*	*398*	*199*	*2388*
	projective	*5, 6*	*3*	*3+33*	*1+162*	*955*	*5*	*1005*
	projective	*6, 7*	*3*	*3+33*	*1+162*	*922*	*5*	***972***
Montgomery	affine	See [3]	*0*	*163*	*163*	*329*	*327*	*3600*
	projective	See [3]	*0*	*163*	*163*	*988*	*1*	***998***
Window TNAF w = 2 (Koblitz curves)	affine	*1, 2*	*0*	*54*	*0*	*108*	*54*	*648*
	projective	*5, 6*	*0*	*54*	*0*	*488*	*1*	***498***
	projective	*6, 7*	*0*	*54*	*0*	*434*	*1*	***444***
Window TNAF w = 4 (Koblitz curves)	affine	*1, 2*	*3*	*36*	*0*	*72*	*36*	*432*
	projective	*5, 6*	*3*	*3+33*	*0*	*305*	*4*	*345*
	projective	*6, 7*	*3*	*3+33*	*0*	*272*	*4*	*312*

difference is known can be computed in terms of the x-coordinates of the involved points. The Montgomery method needs approximately $nA + nD$, where additions and doublings are used to compute the x-coordinate only [3]. The approximate number of field operations of the Montgomery affine version is $(2n + 2)I + (2n + 4)M$ and, for the Montgomery projective version it is $I + (6n + 10)M$ [3].

In [8], Knudsen introduces a new method for scalar multiplication on a non-supersingular elliptic curve over $GF(2^n)$. The idea is to replace all point doublings with a faster operation, called point halving. Moreover, Knudsen shows that the halving algorithm is superior to previous algorithms when it is implemented using affine coordinates and normal basis. However, the halving algorithm has a storage limitation if a polynomial basis is used, where the required storage is in the order of magnitude $O(n^2)$ bits. The halving algorithm and the Montgomery method cannot take advantage of Koblitz curves properties.

5 Complexity Comparison

This section compares the scalar multiplication methods using different formulae. The analysis will be restricted to polynomial bases with a small amount of storage (at most three precomputed points). Therefore, the halving algorithm will not be discussed here.

The following cost-ratios are useful to compare the complexity [9]:

$$r_1 = \frac{time\ required\ for\ inversion}{time\ required\ for\ multiplication}$$
$$r_2 = \frac{time\ required\ for\ multiplication}{time\ required\ for\ squaring}$$
$$r_3 = \frac{time\ required\ for\ multiplication}{time\ required\ for\ adding}.$$

These costs depend on several factors such as the field size, the field representation, the reduction polynomial, the algorithms for multiplication and inversion, and the platform (software, firmware, or hardware). The experiments indicate

that r_2 and r_3 are large; hence, squarings and adding can be neglected (r_3 is much larger since adding in $GF(2)$ is directly supported in hardware). Some experimental values for r_1 are given in Table 1 [6], [10], [11], [12].

Tables 2 and 3 give rough estimates for addition formulae and scalar multiplication methods. The values are calculated based on the following considerations:

1. $a_2 = 1$, $Z_1 = 1$, and $n = 163$.
2. Total cost in field multiplications assuming $1I = 10M$.
3. If projective coordinates are used, then one inversion and two multiplications are required to convert back to affine coordinates.
4. If projective coordinates are used with Window $NAF_{w=4}$ or Window $TNAF_{w=4}$, then three additions and one doubling are computed using affine coordinates formulae.

Table 3 shows that if the projective coordinates are used, then the enhancement in window method using (7) is approximately 2 percent for general curves and approximately 10 percent for Koblitz curves.

6 Conclusion

We have shown that the implementation of the elliptic curve addition formula can be done more efficiently using our new formula in projective coordinates. The comparison shows that the addition formula in projective coordinates is improved by about 10 percent for general field elements and 12 percent for restricted coefficients ($a_2 = 0$ or 1). The enhancement in the window method is approximately 2 percent for general curves and approximately 10 percent if Koblitz curves are used. Therefore, elliptic curve protocols and applications can be implemented with better performance using the suggested formula.

References

[1] http://www.certicom.com, Jan. 2001.
[2] I. Blake, G. Seroussi, and N. Smart, *Elliptic Curve in Cryptography*, pp. 60- 72. Cambridge, U.K.: Cambridge Univ. Press, 1999.
[3] J. López and R. Dahab, "Fast Multiplication on Elliptic Curves over GF(2^n) without Precomputation," *Proc. Cryptographic Hardware and Embedded Systems—CHES '99*, pp. 316-327, 1999.
[4] IEEE P1363, "Standard Specifications for Public Key Cryptography,"Draft 9, June 2001. http://grouper.ieee.org/groups/1363/.
[5] J. López and R. Dahab, "Improved Algorithms for Elliptic Curve Arithmetic in $GF(2^n)$," *Proc. Selected Areas in Cryptography—SAC '98*, pp. 201-212, 1998.
[6] D. Hankerson, J.L. Hernandez, and A. Menezes, "Software Implementation of Elliptic Curve Cryptography over Binary Fields," *Proc. Cryptographic Hardware and Embedded Systems—CHES 2000*, pp. 1-24, 2000.
[7] J. Solinas, "Efficient Arithmetic on Koblitz Curves," *Designs, Code, and Cryptography*, vol. 19, pp. 195-249, 2000.
[8] E. Knudsen, "Elliptic Scalar Multiplication Using Point Halving," *Proc. Advances in Cryptology—Asiacrypt '99*, pp. 135-149, 1999.
[9] J. López and R. Dahab, "An Improvement of the Guajardo-Paar Method for Multiplication on Non-Supersingular Elliptic Curves," *Proc. 18th Int'l Conf. Chilean Computer Science Soc.*, vol. 1, pp. 1-10, 1998.
[10] R. Schroeppel, H. Orman, S. O'Malley, and O. Spatscheck, "Fast Key Exchange with Elliptic Curve Systems," *Proc. Advances in Cryptology—Crypto '95*, pp. 43-56, 1995.
[11] 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*, K. Kim and T. Matsumoto, eds., pp. 65-76, 1996.
[12] A. Weimerskirch, C. Paar, and S.C. Shantz, "Elliptic Curve Cryptography on a Palm OS Device," *Proc. Sixth Australasian Conf. Information Security and Privacy (ACISP 2001)*, p. 502, July 2001.