A New Approach to Low Complexity Binary Elliptic Curve Arithmetic

Murat Cenk, Ahmad Salam Alrefai, Christophe Negre, and M. Anwar Hasan

Abstract

The well known formulas for point addition and doubling on binary elliptic curves involve instances of simultaneous finite field multiplications like AB and AC, where A is common. In this paper, we present a technique which, compared to the conventional approach, has a lower arithmetic complexity for two binary polynomial multiplications AB and AC. This can potentially reduce arithmetic complexities of various systems. Here, we apply this technique to reduce the cost of a point addition and doubling on binary elliptic curves by giving new formulas for Weierstrass, Edwards and Hessian curves. Then we successfully apply this technique to point multiplication in projective coordinate system for Weierstrass curves to show the improvements ranging from 7.65% to 33.85%. Moreover, we implement the point multiplication for NIST curves in projective coordinates for Weierstrass curves and the results show that the proposed algorithm performs better than the previous one.

Index Terms

Elliptic curves, point addition, point doubling, point multiplication, polynomial multiplication, field multiplication.

1 Introduction

Miller [15] and Koblitz [12] independently proposed the use of elliptic curves in cryptography (ECC) in the mid 1980s. The robustness of the protocols in ECC are based on the so-called discrete logarithm problem on elliptic curve (ECDLP). We refer the reader to [9], [18] and references therein for further details. The main advantage of ECC is its significantly shorter key length and its efficiency compared to RSA [14], which is based on the factorization of integer.

The main operation in elliptic curve cryptography is the point multiplication that uses point addition and doubling on the curve. In order to perform point addition and doubling efficiently, there is a need for fast finite field arithmetic and suitable coordinate systems to represent a point on the curve. The best known formulas for point addition and doubling on an elliptic curve over finite fields can be found in [3]. In Crypto 2009, Bernstein [2] proposed new binary polynomial multiplication formulas leading to speed

records for elliptic curve point multiplication. One of the techniques used in [2] is the new three-way binary polynomial multiplication with five multiplications by using interpolation/evaluation approach at five elements $0, 1, \infty, X, X+1$. The recursive use of this technique results in an arithmetic complexity of $O(n^{\log_3 5})$ for three-way multiplication. On the other hand, the authors of [4] have proposed a three-way split binary algorithm for polynomial multiplication with five multiplications by using the techniques in [19]. The algorithm in [4] uses elements $0, 1, \infty, \alpha, \alpha+1$ as evaluation points where $\mathbb{F}_4 = \mathbb{F}_2[\alpha]/(\alpha^2+\alpha+1)$ is the finite field with four elements. The method in [4] has the additional advantage to work for the multiplication of two polynomial in $\mathbb{F}_2[X]$ or even in $\mathbb{F}_4[X]$.

In this paper, we will use the binary polynomial multiplication algorithms of [4] by combining a method given in [19] to improve the complexities for the simultaneous products AB and AC for binary polynomials A, B and C. This method is useful when we simultaneously perform finite field operations that contains products of type AB and AC, where one operand is common. Fortunately, this is often the case in elliptic curve point addition formulas. In this paper, we derive improved point addition formulas and study the impact on NIST recommended binary curves.

The rest of the paper is organized as follows: in Section 2, we set the notations used throughout the paper. In Section 3, we review three-way split methods for polynomial multiplication and their complexities. In Section 4, we present our approach to reduce two $\mathbb{F}_2[X]$ multiplications to one $\mathbb{F}_4[X]$ multiplication. The improved formulas for point addition and doubling are given in Sections 5 and 6, respectively. We give the complexity comparison in Section 7. In Section 8, we briefly present the resulting improvement obtained for the point multiplication on an elliptic curve and give some concluding remarks in Section 9.

2 BASIC NOTATIONS

Throughout the paper we will use the following notations:

- $\mathbb{F}_q$: the finite field with q elements.
- $\mathbb{F}_q[X]$: the ring of polynomials over $\mathbb{F}_q$.
- $\mathbf{M}_q(n)$: the total cost of multiplication of degree $n-1$ polynomials in $\mathbb{F}_q[X]$.
- $\mathbf{M}_{2,4}(n)$: the total cost of multiplication of a degree $n-1$ polynomials in $\mathbb{F}_2[X]$ by a degree $n-1$ polynomial in $\mathbb{F}_4[X]$.
- $\mathbf{R}_2(n)$: the cost of reduction of a degree $(2n-2)$ polynomial in $\mathbb{F}_2[X]$ modulo a degree n irreducible polynomial in $\mathbb{F}_2[X]$.(See Remark 1.)
- $\mathbf{R}_4(n)$: the cost of reduction of a degree $(2n-2)$ polynomial in $\mathbb{F}_4[X]$ modulo a degree n irreducible polynomial in $\mathbb{F}_2[X]$. (See Remark 1.)
- $\mathbf{S}(n)$: the cost of squaring in $\mathbb{F}_{2^n}$.
- $\mathbf{add}(n)$: the cost of addition of two degree $n-1$ polynomials in $\mathbb{F}_2[X]$.

- $\mathbf{d}(n)$: the cost of multiplying a degree $n-1$ polynomial by the curve parameter that is generally a small degree polynomial (for NIST binary curves those are all one).
- Let $\mathbb{F}_4 = \mathbb{F}_2[\alpha]/(\alpha^2+\alpha+1)$ and $T_i \in \mathbb{F}_4[X]$ for some positive integer i. We will use $T_{i,0}$ for α free part of T_i and $T_{i,1}$ for α part of T_i, i.e. $T_i = T_{i,0} + \alpha T_{i,1}$ where $T_{i,0}, T_{i,1} \in \mathbb{F}_2[X]$.
- $\mathbf{A}(n)$: the cost of elliptic curve point addition.
- $\mathbf{D}(n)$: the cost of elliptic curve point doubling.

We will simply represent $\mathbf{M}_q$ instead of $\mathbf{M}_q(n)$ if n is clear from the context. Similarly, we will drop n in the other notations.

Remark 1. In order to multiply finite field elements, we first apply a polynomial multiplication algorithm followed by a reduction modulo an irreducible polynomial in $\mathbb{F}_2[X]$. Consequently, the complexity of a multiplication in $\mathbb{F}_{2^n}$ is equal to $\mathbf{M}_2(n) + \mathbf{R}_2(n)$ and the multiplication of degree $n-1$ polynomials in $\mathbb{F}_4[X]$ modulo a degree n irreducible polynomial in $\mathbb{F}_2[X]$ is equal to $\mathbf{M}_4(n) + \mathbf{R}_4(n)$. For a squaring in $\mathbb{F}_{2^n}$, we remark that $(\sum_{i=0}^{n-1} a_i x^i)^2 = \sum_{i=0}^{n-1} a_i x^{2i}$ which is free of computations. The cost of a squaring in $\mathbb{F}_{2^n}$ is thus simply equal to $\mathbf{R}_2(n)$. Now, since there is always an irreducible pentanomial of degree n in $\mathbb{F}_2[X]$, the cost of the reduction can be assumed to be at worst $4(n-1)$ additions (for further details we refer to [20]). Consequently, in the remaining of this paper, we will use the following expressions of $\mathbf{R}_2(n), \mathbf{R}_4(n)$ and $\mathbf{S}(n)$

$$\mathbf{R}_2(n) = 4n-4, \quad \mathbf{R}_4(n) = 8n-8, \quad \mathbf{S}(n) = 4n-4. \tag{1}$$

Moreover, an addition in finite field $\mathbb{F}_{2^n}$ requires n additions at the bit level and thus $\mathbf{add}(n) = n$. We will also consider in the sequel a multiplication by a constant in $\mathbb{F}_{2^n}$ which has a small degree and small number of non zero coefficients, i.e., roughly 10. We will thus assume that the cost of such multiplication is less than $10n$, i.e., we will have $\mathbf{d}(n) = 10n$.

3 THREE-WAY BINARY POLYNOMIAL MULTIPLICATIONS

Let $\mathrm{O}(n^\omega)$ be the arithmetic complexity of the binary polynomial multiplication where n is the size of the polynomials being multiplied. The standard schoolbook method has $\omega = 2$. Karatsuba [10] showed that two degree one polynomials can be multiplied with three multiplications of half sizes. The recursive use of this algorithm for $n = 2^\ell$ gives $\omega = \log_2 3 \approx 1.58$. This algorithm is called two-way split algorithm. For $n = 3$, Karatsuba like multiplication algorithms can be obtained by using the Chinese remainder theorem [17], [19] with six multiplications; hence one can obtain $\omega = \log_3 6 \approx 1.56$ for three way algorithms which can be used for polynomial multiplication of size $n = 3^\ell$. If n is not power of two or three, one can use padding technique and combinations of two or three-way split algorithms. Note that further results were obtained in [16], [7], [6] for binary polynomial multiplications.

In Crypto 2009, Bernstein proposed a three-way split binary polynomial multiplication with five multiplications [2] by using the interpolation method with evaluation points $0, 1, \infty, X$ and $X+1$. This leads

to $\omega = \log_3 5 \approx 1.46$. The computation of the exact complexities of formulas including hidden coefficients can be found in [4]. The complexities are also summarized in Table 1.

TABLE 1: Complexities of three-way split multiplications in $\mathbb{F}_2[X]$

Operation	Formula	$\#\oplus$	$\#\otimes$
Pol. Mul. with six rec. mult.	[17]	$5.33n^{\log_3(6)} - 7.33n + 2$	$n^{\log_3(6)}$
	[5]	$5.27n^{\log_3(6)} - 6.67n + 1.4$	$n^{\log_3(6)}$
Pol. Mul. with five rec. mult.	[2]	$18.5n^{\log_3(5)} - 21.5n + 3$	$7n^{\log_3(5)} - 4n - 2$
	[4]	$27.75n^{\log_3(5)} - 9.67n\log_3(n) - 28.5n + 0.75$	$3n^{\log_3(5)} - 2n$

Reference [4] presents an algorithm for three-way split polynomial multiplication with five multiplications of one-third sizes. This algorithm will be used throughout the paper. Therefore, we give this algorithm briefly. First, binary polynomials $A = \sum_{i=0}^{n-1} a_i X^i$ and $B = \sum_{i=0}^{n-1} b_i X^i$ with $n = 3^\ell$ are divided into three parts $A = A_0 + A_1 X^{n/3} + A_2 X^{2n/3}$ and $B = B_0 + B_1 X^{n/3} + B_2 X^{2n/3}$ where A_i and B_i have degree $< n/3$ each. Since there are not enough elements in $\mathbb{F}_2$ for applying the interpolation method, the polynomials are evaluated at 0, 1, α, $\alpha + 1$ and ∞ and the result is interpolated where $\mathbb{F}_4 = \mathbb{F}_2[\alpha]/(\alpha^2+\alpha+1)$. The resulting evaluations and recursive multiplications of smaller size polynomials are given below:

$$\begin{aligned}
P_0 &= A_0 B_0 && \text{in } \mathbb{F}_2[X],\\
P_1 &= (A_0 + A_1 + A_2)(B_0 + B_1 + B_2) && \text{in } \mathbb{F}_2[X],\\
P_2 &= (A_0 + A_2 + \alpha(A_1 + A_2))(B_0 + B_2 + \alpha(B_1 + B_2)), && \text{in } \mathbb{F}_4[X],\\
P_3 &= (A_0 + A_1 + \alpha(A_1 + A_2))(B_0 + B_1 + \alpha(B_1 + B_2)), && \text{in } \mathbb{F}_4[X],\\
P_4 &= A_2 B_2 && \text{in } \mathbb{F}_2[X].
\end{aligned} \tag{2}$$

The reconstruction of $C = A \times B$ is given below

$$\begin{aligned}
C = {} & (P_0 + X^{n/3} P_4)(1 + X^n) + (P_1 + (1 + \alpha)(P_2 + P_3))(X^{n/3} + X^{2n/3} + X^n)\\
& + \alpha(P_2 + P_3)X^n + P_2 X^{2n/3} + P_3 X^{n/3}
\end{aligned} \tag{3}$$

The arithmetic complexity of the recursive use of this algorithm is $O(n^{\log_3 5})$. When the hidden coefficients are computed it is seen that the total arithmetic complexities (i.e., the number of bit level additions and multiplications combined) of the algorithms in [4] are higher than those in [2]; however the multiplication and delay complexities are better. In the next section, we show that formulas given in (2) and (3) can be used to compute AB and AC for $A, B, C \in \mathbb{F}_2[X]$ at the cost of only one multiplication in the extension field $\mathbb{F}_4[X]$ rather than two direct multiplications in the base field $\mathbb{F}_2[X]$.

4 REDUCTION OF TWO $\mathbb{F}_2[X]$ MULTIPLICATIONS TO ONE $\mathbb{F}_4[X]$ MULTIPLICATION

In [19], it was observed that the computation of the products AB and AC for polynomials $A, B, C \in \mathbb{F}[X]$, the ring of polynomials over an arbitrary field $\mathbb{F}$, can be computed with only one multiplication in $\mathbb{F}(\alpha)[X]$, the ring of polynomials over $\mathbb{F}(\alpha)$, the extension field of $\mathbb{F}$. When we apply this to the binary polynomials with the algorithm given in (2) and (3), we see that AB and AC for $A, B, C \in \mathbb{F}_2[X]$ can be computed with only one multiplication in $\mathbb{F}_4[X]$:

$$P = A(B + \alpha C) = P_0 + \alpha P_1. \tag{4}$$

This gives us $P_0 = AB$ and $P_1 = AC$ at the cost of only one multiplication in $\mathbb{F}_4[X]$ rather than two multiplications in $\mathbb{F}_2[X]$. This method works better if the cost of one polynomial multiplication in $\mathbb{F}_4[X]$ is better than the cost of two polynomial multiplications in $\mathbb{F}_2[X]$. Therefore, we need the complexity of the product $A(B + \alpha C)$ where $A \in \mathbb{F}_2[X]$ and $(B + \alpha C) \in \mathbb{F}_4[X]$. The arithmetic complexity of such products can be computed by using the algorithm given in (2) and (3). The exact complexity of the algorithm in (2) and (3) for computing AB where $A, B \in \mathbb{F}_2[X]$ or $A, B \in \mathbb{F}_4[X]$ are given in [4].

One can compute the complexity of the product AB for $A \in \mathbb{F}_2[X]$ and $B \in \mathbb{F}_4[X]$ similarly. Here, we need $4n/3$ additions for computing the evaluations in $A(X) \in \mathbb{F}_2[X]$ and $11n/3$ additions for computing the evaluations in $B(X) \in \mathbb{F}_4[X]$ (see [4] for details). The recursive products require three multiplications of one-third sizes where one operand is from $\mathbb{F}_2[X]$ and other operand is from $\mathbb{F}_4[X]$; and two multiplications of one-third sizes in $\mathbb{F}_2[X]$. Finally, the reconstruction complexity is the same with the computations for $\mathbb{F}_4[X]$ in [4]. Therefore, we obtain

$$\begin{cases} M_{2,4,\oplus}(n) &= 3M_{2,4,\oplus}(n/3) + 2M_{4,\oplus}(n/3) + 51n/3 - 21,\ M_{2,4,\oplus}(1) = 0, \\ M_{2,4,\otimes}(n) &= 3M_{2,4,\otimes}(n/3) + 2M_{4,\otimes}(n/3),\ M_{2,4,\otimes}(1) = 2, \end{cases} \tag{5}$$

where $M_{2,4,\oplus}(n)$ and $M_{2,4,\otimes}(n)$ are the number additions and multiplications in order to compute the product of a degree $n-1$ polynomial over $\mathbb{F}_2$ by a degree $n-1$ polynomial over $\mathbb{F}_4$. Then we get

$$\begin{cases} M_{2,4,\oplus}(n) &= 26.75n^{\log_3(5)} - 2.33n\log_3(n) - 32n + 10.5, \\ M_{2,4,\otimes}(n) &= 4n^{\log_3(5)} - 2n. \end{cases} \tag{6}$$

So the total arithmetic complexity for the multiplication of a polynomial in $\mathbb{F}_2[X]$ by a polynomial $\mathbb{F}_4[X]$ is

$$\mathbf{M}_{2,4}(n) = 30.75n^{\log_3(5)} - 2.33n\log_3(n) - 34n + 10.5 \tag{7}$$

On the other hand, the direct computation of AB and AC for $A, B, C \in \mathbb{F}_2[X]$ requires two polynomial multiplications. If we use the complexity of [2] given in Table 1, we obtain the total arithmetic complexity $51n^{\log_3 5} - 52n - 6$. Note that the dominant term in the complexities is the term with $n^{\log_3 5}$. The linear terms can be ignored when n increases. Therefore, we can say that the method in (4) improves the total arithmetic complexity by approximately 40% for computing the products AB and AC in $\mathbb{F}_2[X]$.

This method is useful when we perform simultaneous finite field operations that contains products of type AB and AC. Fortunately, this kind of computation exists in elliptic curve point addition and doubling formulas. We will apply this method to the elliptic curve addition and doubling in the following sections.

5 IMPROVED POINT ADDITION FORMULAS

In this section, we apply the algorithm described in Section 4 to the elliptic curve point addition and we propose new elliptic curve point addition formulas for Weierstrass, Edwards and Hessian curves. We refer the reader to references like [9], [18] for more details on elliptic curves in cryptography.

5.1 Short Weierstrass curves

A Weierstrass form of an elliptic curve over $\mathbb{F}_{2^n}$ is

$$y^2 + xy = x^3 + a_2x^2 + a_6 \tag{8}$$

where $a_2, a_6 \in \mathbb{F}_{2^n}$ and $a_6 \neq 0$. Note that $a_2 = 1$ for NIST binary curves. We will give improved formulas for elliptic curve point addition in López-Dahab coordinates, projective coordinates and Jacobian coordinates.

López-Dahab coordinates. The point (x, y) on the curve as defined in (8) is represented by (X, Y, Z) in López-Dahab coordinates where $x = X/Z$ and $y = Y/Z^2$ satisfying

$$Y^2 + XYZ = X^3Z + a_2X^2Z^2 + a_6Z^4. \tag{9}$$

Let $P = (X_1, Y_1, Z_1)$, $Q = (X_2, Y_2, Z_2)$ and $R = (X_3, Y_3, Z_3)$ be on (12) such that $P + Q = R$. The formula in [3] and [13] for the efficient point addition in this coordinate is given in (10) which requires 13$\mathbf{M}_2$, 13$\mathbf{R}_2$, 4$\mathbf{S}$, and 9**add**.

$$\begin{cases} A = X_1Z_2,\ B = X_2Z_1,\ C = A^2,\ D = B^2,\ E = A + B, \\ F = C + D,\ G = Y_1Z_2^2,\ H = Y_2Z_1^2,\ I = G + H,\ J = IE, \\ Z_3 = FZ_1Z_2,\ X_3 = A(H + D) + B(C + G) \\ Y_3 = (AJ + FG)F + (J + Z_3)X_3. \end{cases} \tag{10}$$

Note that the products $B = X_2Z_1$ and $Z_3 = FZ_1Z_2$ in (10) have both Z_1 as input. In order to compute X_2Z_1 and Z_1Z_2 with only one multiplication in $\mathbb{F}_4$ we compute $T_1 = Z_1(X_2 + \alpha Z_2)$. Then we obtain $T_{1,0} = Z_1X_2$ and $T_{1,1} = Z_1Z_2$ from $T_1 = T_{1,0} + \alpha T_{1,1}$. Similar situations are observed in the products $F(Z_1Z_2)$ and FG, where F is the common operand, and also in $A(H + D)$ and AJ, where A is the common operand. The new formula is given below that uses 7$\mathbf{M}_2$, 3$\mathbf{M}_{2,4}$, 8$\mathbf{R}_2$, 3$\mathbf{R}_4$, 4$\mathbf{S}$, and 9**add**.

$$\begin{cases} T_1 = Z_1(X_2 + \alpha Z_2), \quad T_2 = T_{1,0}, \quad T_3 = T_{1,1}, \quad T_4 = X_1 Z_2, \\ T_5 = T_4^2, \quad T_6 = T_2^2, \quad T_7 = T_2 + T_4, \quad T_8 = T_5 + T_6, \\ T_9 = Y_1 Z_2^2, \quad T_{10} = Y_2 Z_1^2, \quad T_{11} = T_9 + T_{10}, \quad T_{12} = T_7 T_{11}, \\ T_{13} = T_8(T_3 + \alpha T_9), \quad T_{14} = T_{13,0}, \quad T_{15} = T_{13,1}, \\ T_{16} = T_4(T_6 + T_{10} + \alpha T_{12}), \quad T_{17} = T_{16,0}, \quad T_{18} = T_{16,1}, \\ Z_3 = T_{14}, \quad X_3 = T_{17} + T_2(T_5 + T_9), \\ Y_3 = (T_{18} + T_{15})T_8 + (T_{12} + Z_3)X_3 \end{cases} \tag{11}$$

Extended López-Dahab coordinates with $a_2 = 1$. The point (x, y) on the curve as defined in (8) is represented by (X, Y, Z, S) in extended López-Dahab coordinates with $a_2 = 1$ where $x = X/Z$, $y = Y/Z^2$ and $S = Z^2$ satisfying

$$Y^2 + XYZ = X^3 Z + X^2 Z^2 + a_6 Z^4. \tag{12}$$

Let $P = (X_1, Y_1, Z_1, S_1)$, $Q = (X_2, Y_2, 1, 1)$ and $R = (X_3, Y_3, Z_3, S_3)$ be on (12) such that $P + Q = R$. The formula in [3] and [11] for the efficient point addition in this coordinate is given in (13) which requires $8\mathbf{M}_2$, $8\mathbf{R}_2$, $4\mathbf{S}$, and $8\mathbf{add}$.

$$\begin{cases} A = X_1 + X_2 Z_1,\ B = Y_1 + Y_2 S_1,\ C = A Z_1,\ D = C(B + C),\ Z_3 = C^2, \\ S_3 = Z_3^2,\ X_3 = B^2 + CA^2 + D,\ Y_3 = (X_3 + X_2 Z_3)D + (X_2 + Y_2)S_3. \end{cases} \tag{13}$$

Note that the products $D = C(B + C)$ and CA^2 in X_3 in (13) have both C as a operand. In order to compute $C(B + C)$ and CA^2 with only one multiplication in $\mathbb{F}_4$ we compute $T_1 = C((B + C) + \alpha A^2)$. Then we obtain $T_{1,0} = C(B+C)$ and $T_{1,1} = CA^2$ from $T_1 = T_{1,0} + \alpha T_{1,1}$. The new formula is given below that uses $6\mathbf{M}_2$, $1\mathbf{M}_{2,4}$, $6\mathbf{R}_2$, $1\mathbf{R}_4$, $4\mathbf{S}$, and $8\mathbf{add}$.

$$\begin{cases} A = X_1 + X_2 Z_2,\ B = Y_1 + Y_2 S_1,\ C = A Z_1,\ T_1 = C((B + C) + \alpha A^2),\ D = T_{1,0}, \\ T_2 = T_{1,1},\ Z_3 = C^2,\ S_3 = Z_3^2,\ X_3 = B^2 + T_2 + D,\ Y_3 = (X_3 + X_2 Z_3)D + (X_2 + Y_2)S_3. \end{cases} \tag{14}$$

Projective coordinates. An elliptic curve point (x, y) satisfying (8) is represented in projective coordinates as (X, Y, Z), where $x = X/Z$ and $y = Y/Z$, and the projective representation satisfies the following:

$$Y^2 Z + XYZ = X^3 + a_2 X^2 Z + a_6 Z^3. \tag{15}$$

Let $P = (X_1, Y_1, Z_1)$, $Q = (X_2, Y_2, Z_2)$ and $R = (X_3, Y_3, Z_3)$ be on (15) such that $P + Q = R$. The formula in [3] for the efficient point addition in this coordinate system is given below.

$$\begin{cases} S_1 = Y_1 Z_2, \quad S_2 = X_1 Z_2, \quad A = S_1 + Z_1 Y_2, \\ B = S_2 + Z_1 X_2, \quad S_3 = A + B, \quad C = B^2 \\ D = Z_1 Z_2, \quad E = BC, \quad F = (AS_3 + a_2 C)D + E \\ X_3 = BF, \quad Y_3 = C(AS_2 + BS_1) + S_3 F, \quad Z_3 = ED \end{cases} \tag{16}$$

The formula for point addition as given in (16) requires $14\mathbf{M}_2$, $14\mathbf{R}_2$, $1\mathbf{S}$, $1\mathbf{d}$, and $7\mathbf{add}$.

Note that the products $S_1 = Y_1Z_2$ and $S_2 = X_1Z_2$ have the common operand Z_2; Z_1Y_2 and Z_1X_2 have the common operand Z_1; AS_3 and AS_2 have the common operand A; $(AS_3 + a_2C)D$ and ED have the common operand D, and finally FB and FS_3 have the common term F. So we can propose the following formula by using the method in Section 4:

$$\begin{cases} T_1 = Z_2(Y_1 + \alpha X_1), \quad T_2 = T_{1,0}, \quad T_3 = T_{1,1}, \\ T_4 = Z_1(Y_2 + \alpha X_2), \quad T_5 = T_{4,0}, \quad T_6 = T_{4,1}, \\ T_7 = T_2 + T_5, \quad T_8 = T_3 + T_6, \quad T_9 = T_7 + T_8, \quad T_{10} = T_8^2, \\ T_{11} = Z_1Z_2, \quad T_{12} = T_8T_{10}, \quad T_{13} = T_7(T_9 + \alpha T_3), \\ T_{14} = T_{13,0}, \quad T_{15} = T_{13,1}, \quad T_{16} = T_{14} + a_2T_{10}, \\ T_{17} = T_{11}(T_{16} + \alpha T_{12}), \quad T_{18} = T_{17,0}, \quad T_{19} = Z_3 = T_{17,1}, \\ T_{20} = T_{18} + T_{12}, \quad T_{21} = T_{20}(T_8 + \alpha T_9), \quad T_{22} = T_{21,0}, \\ T_{23} = T_{21,1}, \quad X_3 = T_{22}, \quad Y_3 = T_{10}(T_{15} + T_8T_2) + T_{23}. \end{cases} \tag{17}$$

Formula in (17) uses $4\mathbf{M}_2$, $5\mathbf{M}_{2,4}$, $4\mathbf{R}_2$, $5\mathbf{R}_4$, $1\mathbf{S}$, $1\mathbf{d}$, and $7\mathbf{add}$.

Jacobian coordinates. An elliptic curve point (x, y) satisfying (8) is represented in Jacobian coordinates as (X, Y, Z) where $x = X/Z^2$ and $y = Y/Z^3$. The corresponding curve equation is

$$Y^2 + XYZ = X^3 + a_2X^2Z^2 + a_6Z^6. \tag{18}$$

Let $P = (X_1, Y_1, Z_1)$, $Q = (X_2, Y_2, Z_2)$ and $R = (X_3, Y_3, Z_3)$ be on (18) such that $P + Q = R$. The formula in [3] for the efficient point addition in this coordinate system is given below.

$$\begin{cases} O_1 = Z_1^2, \quad O_2 = Z_2^2, \quad A = X_1O_2, \quad B = X_2O_1 \\ C = Y_1O_2Z_2, \quad D = Y_2O_1Z_1, \quad E = A + B \\ F = C + D, \quad G = EZ_1, \quad H = FX_2 + GY_2, \quad Z_3 = GZ_2 \\ I = F + Z_3, \quad X_3 = a_2Z_3^2 + FI + EE^2, \quad Y_3 = IX_3 + G^2H. \end{cases} \tag{19}$$

The formula of (19) requires $14\mathbf{M}_2$, $14\mathbf{R}_2$, $5\mathbf{S}$, $1\mathbf{d}$, and $7\mathbf{add}$.

Note that in (19) the products O_2X_1 and O_2Z_2 have the common operand O_2; O_1X_2 and O_1Z_1 have the common operand O_1; FX_2 and FI have the common operand F, EZ_1 and EE^2 have the common operand E and finally GY_2 and GZ_2 have the common operand G. Using the method in Section 4, we can modify (19) as follows:

$$\begin{cases} T_1 = Z_1^2,\ T_2 = Z_2^2,\ T_3 = O_2(X_1 + \alpha Z_2),\ T_4 = T_{3,1}, \\ T_5 = T_{3,0},\ T_6 = O_1(X_2 + \alpha Z_1),\ T_7 = T_{6,1},\ T_8 = T_{6,0}, \\ T_9 = Y_1 T_4,\ T_{10} = Y_2 T_7,\ T_{11} = T_5 + T_8,\ T_{12} = T_9 + T_{10}, \\ T_{13} = T_{11}^2,\ T_{14} = T_{11}(Z_1 + \alpha T_{13}),\ T_{15} = T_{14,0},\ T_{16} = T_{14,1}, \\ T_{17} = T_{15}(Y_2 + \alpha Z_2),\ T_{18} = T_{17,0},\ T_{19} = Z_3 = T_{17,1}, \\ T_{20} = T_{12} + Z_3,\ T_{21} = T_{12}(X_2 + \alpha T_{20}),\ T_{22} = T_{21,0}, T_{23} = T_{21,1}, \\ T_{24} = T_{18} + T_{22},\ T_{25} = Z_3^2,\ X_3 = a_2 T_{25} + T_{23} + T_{16},\ T_{26} = T_{20} X_3, \\ T_{27} = T_{15}^2,\ T_{28} = T_{27} T_{24},\ Y_3 = T_{26} + T_{28}. \end{cases} \tag{20}$$

The proposed formula in (20) uses $4\mathbf{M}_2$, $5\mathbf{M}_{2,4}$, $4\mathbf{R}_2$, $5\mathbf{R}_4$, $5\mathbf{S}$, $1\mathbf{d}$, and $7\mathbf{add}$.

5.2 Edwards curves

An elliptic curve in Edwards form is represented by the following equation:

$$d_1(x + y) + d_2(x^2 + y^2) = (x + x^2)(y + y^2), \tag{21}$$

where $d_1, d_2 \in \mathbb{F}_{2^n}$.

Affine coordinates. Although no z-coordinate is involved in (21), to be onsistent with other formulas, we let $x = X$ and $y = Y$. Elliptic curve point addition in affine coordinates [1] can be performed by using the following formula:

$$\begin{cases} W_1 = X_1 + X_2, \quad W_2 = X_2 + Y_2, \quad A = X_1^2 + X_1, \\ B = Y_1^2 + Y_1, \quad C = d_2 W_1 W_2, \quad D = X_2 Y_2, \\ X_3 = Y_1 + (C + d_1(W_1 + X_2) + A(D + X_2))/(d_1 + A W_2), \\ Y_3 = X_1 + (C + d_1(W_1 + Y_2) + B(D + Y_2))/(d_1 + B W_2). \end{cases} \tag{22}$$

The formula in (22) requires $2\mathbf{I}$, $8\mathbf{M}_2$, $8\mathbf{R}_2$, $2\mathbf{S}$, $3\mathbf{d}$, and $16\mathbf{add}$, where $\mathbf{I}$ is the cost of inversion.

Note that A is the common operand in products $A(D + X_2)$ and AW_2; and B is common operand in $B(D + Y_2)$ and BW_2. So we can propose the following formula by using the method in Section 4:

$$\begin{cases} W_1 = X_1 + X_2, \quad W_2 = X_2 + Y_2, \quad A = X_1^2 + X_1, \\ B = Y_1^2 + Y_1, \quad C = d_2 W_1 W_2, \quad D = X_2 Y_2, \\ T_1 = A(D + X_2 + \alpha W_2), \quad T_2 = T_{1,0}, \quad T_3 = T_{1,1}, \\ T_4 = B(D + Y_2 + \alpha W_2), \quad T_5 = T_{4,0}, \quad T_6 = T_{4,1}, \\ X_3 = Y_1 + (C + d_1(W_1 + X_2) + T_2)/(d_1 + T_3), \\ Y_3 = X_1 + (C + d_1(W_1 + Y_2) + T_5)/(d_1 + T_6). \end{cases} \tag{23}$$

The formula in (23) requires $2\mathbf{I}$, $4\mathbf{M}_2$, $4\mathbf{R}_2$, $2\mathbf{M}_{2,4}$, $2\mathbf{R}_4$, $2\mathbf{S}$, $3\mathbf{d}$, and $16\mathbf{add}$.

Projective coordinates. An elliptic curve point (x, y) satisfying (21) is represented in projective ordinates as (X, Y, Z) where $x = X/Z$ and $y = Y/Z$. Elliptic curve point addition in projective coordinates [1], [3]

can be performed by using the following formula:

$$\begin{cases} W_1 = X_1 + X_2, \quad W_2 = X_2 + Y_2, \quad A = X_1(X_1 + Z_1), \\ B = Y_1(Y_1 + Z_1), \quad C = Z_1Z_2, \quad D = W_2Z_2, \\ E = d_1C^2, \quad H = (d_1Z_2 + d_2W_2)W_1C, \quad I = d_1CZ_1, \\ U = E + AD, \quad V = E + BD, \quad S = UV, \\ X_3 = SY_1 + (H + X_2(I + A(Y_2 + Z_2)))VZ_1, \\ Y_3 = SX_1 + (H + Y_2(I + B(X_2 + Z_2)))UZ_1, \quad Z_3 = SZ_1. \end{cases} \tag{24}$$

The formula in (24) requires 21$\mathbf{M}_2$, 21$\mathbf{R}_2$, 1**S**, 4**d**, and 15**add**.

Note that in (24) the products Z_1Z_2 and W_2Z_2 have the common operand Z_2; CW_1 and CZ_1 have the common operand C; AD and BD have the common operand D, SY_1 and SX_1 have the common operand S and VZ_1 and UZ_1 have common operand Z_1. Using the method in Section 4, we can modify (24) as follows:

$$\begin{cases} T_1 = X_1 + X_2, \quad T_2 = X_2 + Y_2, \quad T_3 = X_1(X_1 + Z_1), \\ T_4 = Y_1(Y_1 + Z_1), \quad T_5 = Z_2(Z_1 + \alpha T_2), \quad T_6 = T_{5,0}, \\ T_7 = T_{5,1}, \quad T_8 = d_1T_6^2, \quad T_9 = T_6(T_1 + Z_1\alpha), \\ T_{10} = T_{9,0}, \quad T_{11} = T_{9,1}, \quad T_{12} = (d_1Z_2 + d_2T_2)T_{10}, \\ T_{13} = d_1T_{11}, \quad T_{14} = T_7(T_3 + \alpha T_4), \quad T_{15} = T_{14,0}, \\ T_{16} = T_{14,1}, \quad T_{17} = T_8 + T_{15}, \quad T_{18} = T_8 + T_{16}, \\ T_{19} = T_{17}T_{18}, \quad T_{20} = T_{19}(Y_1 + \alpha X_1), \quad T_{21} = T_{19,0}, \\ T_{22} = T_{19,1}, \quad T_{23} = Z_1(T_{17} + \alpha T_{18}), \quad T_{24} = T_{23,0}, \\ T_{25} = T_{23,1}, \quad X_3 = T_{21} + (T_{12} + X_2(T_{13} + T_3(Y_2 + Z_2)))T_{25}, \\ Y_3 = T_{22} + (T_{12} + Y_2(T_{13} + T_4(X_2 + Z_2)))T_{24}, \quad Z_3 = T_{19}Z_1. \end{cases} \tag{25}$$

The formula in (25) requires 11$\mathbf{M}_2$, 5$\mathbf{M}_2$, 11$\mathbf{R}_2$, 5$\mathbf{R}_4$, 1**S**, 4**d**, and 15**add**.

5.3 Hessian curves

An elliptic curve in the Hessian form is represented by the following equation:

$$x^3 + y^3 + 1 = dxy, \tag{26}$$

where $d \in \mathbb{F}_{2^n}$ is a parameter.

Projective coordinates. An elliptic curve point (x, y) satisfying (26) is represented in projective ordinates as (X, Y, Z) where $x = X/Z$ and $y = Y/Z$. Elliptic curve point addition in projective coordinates [3] can be performed by using the formula in (27)

$$\begin{cases} A = X_1Y_2, \quad B = X_1Z_2, \quad C = Y_1Z_2, \\ D = Y_1X_2, \quad E = Z_1X_2, \quad F = Z_1Y_2, \\ X_3 = DC + FA, \quad Y_3 = BA + DE, \quad Z_3 = FE + BC. \end{cases} \tag{27}$$

The formula in (27) requires $12\mathbf{M}_2$, $12\mathbf{R}_2$, and $3\mathbf{add}$.

Note that in (27) the products X_1Y_2 and X_1Z_2 have the common operand X_1; Y_1Z_2 and Y_1X_2 have the common operand Y_1; Z_1X_2 and Z_1Y_2 have the common operand Z_1; DC and DE have the common operand D; BC and BA have common operand B; and FA and FE have the common operand F. Using the method in Section 4, we can modify (27) as follows:

$$\begin{cases} T_1 = X_1(Y_2 + \alpha Z_2), \quad T_2 = T_{1,0}, \quad T_3 = T_{1,1}, \\ T_4 = Y_1(Z_2 + \alpha X_2), \quad T_5 = T_{4,0}, \quad T_6 = T_{4,1}, \\ T_7 = Z_1(X_2 + \alpha Y_2), \quad T_8 = T_{7,0}, \quad T_9 = T_{7,1}, \\ T_{10} = T_6(T_5 + \alpha T_8), \quad T_{11} = T_{10,0}, \quad T_{12} = T_{10,1}, \\ T_{13} = T_9(T_2 + \alpha T_8), \quad T_{14} = T_{13,0}, \quad T_{15} = T_{13,1}, \\ T_{16} = T_3(T_2 + \alpha T_5), \quad T_{17} = T_{16,0}, \quad T_{18} = T_{16,1}, \\ X_3 = T_{11} + T_{14}, \quad Y_3 = T_{17} + T_{12}, \quad Z_3 = T_{15} + T_{18}. \end{cases} \tag{28}$$

The proposed formula in (28) uses $6\mathbf{M}_{2,4}$, $6\mathbf{R}_4$, and $3\mathbf{add}$.

6 IMPROVED POINT DOUBLING FORMULAS

In this section, we will apply the algorithm described in Section 4 to the elliptic curve point doubling and we will propose new elliptic curve point doubling formulas for Weierstrass, Edwards and Hessian curves.

6.1 Short Weierstrass curves

The only improvement for point doubling is obtained in projective coordinates for Weierstrass curves since no multiplications of type AB and AC are observed in López-Dahab and Jacobian coordinates.

Projective coordinates. For point doubling on the Weierstrass curves using projective coordinates, we start with the best known and then apply the common operand idea to it. The following formula is from [3]:

$$\begin{cases} A = X_1^2,\ B = A + Y_1Z_1,\ C = X_1Z_1, D = B + C,\ E = C^2, \\ F = BD + a_2E,\ X_3 = CF,\ Y_3 = DF + A^2C,\ Z_3 = CE. \end{cases} \tag{29}$$

The formula in (29) requires $7\mathbf{M}_2, 7\mathbf{R}_2, 3\mathbf{S}, 1\mathbf{d}$, and $4\mathbf{add}$. Note that in (29) the products Y_1Z_1 and X_1Z_1 have the common operand Z_1; CF and DF have the common operand F; similarly A^2C and CE have the common operand C. Using the method in Section 4, we can modify (29) as follows:

$$\begin{cases} T_1 = X_1^2,\ T_2 = Z_1(Y_1 + \alpha X_1),\ T_3 = T_{2,0},\ T_4 = T_{2,1}, T_5 = T_1 + T_3,\ T_6 = T_4 + T_5, \\ T_7 = T_4^2,\ T_8 = T_5T_6 + a_2T_7, T_9 = T_8(T_4 + \alpha T_6),\ T_{10} = T_{9,0},\ T_{11} = T_{9,1}, \\ T_{12} = T_1^2,\ T_{13} = T_4(T_{12} + \alpha T_7),\ T_{14} = T_{13,0},\ T_{15} = T_{13,1}, \\ X_3 = T_{10},\ Y_3 = T_{11} + T_{14},\ Z_3 = T_{15}. \end{cases} \tag{30}$$

The proposed formula for point doubling of projective coordinates in (30) uses $1\mathbf{M}_2$, $3\mathbf{M}_{2,4}$, $1\mathbf{R}_2$, $3\mathbf{R}_4$, $3\mathbf{S}, 1\mathbf{d}$, and 4**add**.

6.2 Edwards curves

In this section, we give improved point addition and doubling formulas for Edwards curves in affine and projective coordinates.

Affine coordinates Elliptic curve point doubling in affine coordinates [1] can be performed by using the formula (31) where $d_2d_1 = d_2/d_1$ is assumed.

$$\begin{cases} A = X_1^2, \quad B = A^2, \quad C = Y_1^2, \quad D = C^2, \\ E = A + C, \quad F = B + D, \quad G = 1/(d_1 + E + d_2d_1F), \\ X_3 = 1 + (d_1 + d_2E + C + D)G, \quad Y_3 = X_3 + (E + F)G. \end{cases} \tag{31}$$

The formula 31 requires $1\mathbf{I}$, $2\mathbf{M}_2$, $2\mathbf{R}_2$, $4\mathbf{S}$, $2\mathbf{d}$, and 10**add**.

Note that G is the common operand in products $(d_1+d_2E+C+D)G$ and $(E+F)G$. So we can propose the following formula by using the method in Section 4:

$$\begin{cases} A = X_1^2, \quad B = A^2, \quad C = Y_1^2, \quad D = C^2, \\ E = A + C, \quad F = B + D, \quad G = 1/(d_1 + E + d_2d_1F), \\ T_1 = G((d_1 + d_2E + C + D) + \alpha(E + F)), \quad T_2 = T_{1,0}, \quad T_3 = T_{1,1}, \\ X_3 = 1 + T_2, \quad Y_3 = X_3 + T_3. \end{cases} \tag{32}$$

The formula in (32) requires $1\mathbf{I}$, $1\mathbf{M}_{2,4}$, $2\mathbf{R}_4$, $4\mathbf{S}$, $2\mathbf{d}$, and 10**add**.

Projective coordinates. Elliptic curve point doubling in projective coordinates [1] can be performed by using the formula (33) where $d_2d_1 = d_2/d_1$ is assumed.

$$\begin{cases} A = X_1^2, \quad B = A^2, \quad C = Y_1^2, \quad D = C^2, \quad E = Z_1^2, \quad F = d_1E^2, \\ G = d_2d_1(B + D), \quad H = AE, \quad I = CE, \quad J = H + I, \\ K = G + d_2J, \quad Z_3 = F + J + G, \quad X_3 = K + H + D, \quad Y_3 = K + I + B. \end{cases} \tag{33}$$

The formula in (33) requires $2\mathbf{M}_2$, $2\mathbf{R}_2$, $6\mathbf{S}$, $2\mathbf{d}$, and 9**add**.

Note that E is the common operand in products AE and CE. So we can propose the following formula by using the method in Section 4:

$$\begin{cases} A = X_1^2, \quad B = A^2, \quad C = Y_1^2, \quad D = C^2, \quad E = Z_1^2, \quad F = d_1E^2, \\ G = d_2d_1(B + D), \quad T_1 = E(A + \alpha C), \quad , T_2 = T_{1,0}, \quad T_3 = T_{1,1}, \\ K = G + d_2J, \quad Z_3 = F + J + G, \quad X_3 = K + T_2 + D, \quad Y_3 = K + T_3 + B. \end{cases} \tag{34}$$

The formula in (36) requires $1\mathbf{M}_{2,4}$, $1\mathbf{R}_4$, $6\mathbf{S}$, $2\mathbf{d}$, and 9**add**.

6.3 Hessian curves

In this section, we give the point doubling formula for Hessian curves in projective coordinates.

Projective coordinates. Elliptic curve point doubling in projective coordinates for Hessian curves [3] can be performed by using the following formula with $6\mathbf{M}_2$, $6\mathbf{R}_2$, $3\mathbf{S}$, and $3\mathbf{add}$:

$$\begin{cases} A = X_1^2, \quad B = X_1 A, \quad C = Y_1^2, \quad D = Y_1 C, \quad E = Z_1^2, \quad F = Z_1 E, \\ X_3 = Y_1(F + B), \quad Y_3 = X_1(D + F), \quad Z_3 = Z_1(B + D). \end{cases} \tag{35}$$

Note that Y_1 is the common operand in products $Y_1 C$ and $Y_1(F + B)$. So we can propose the following formula by using the method in Section 4:

$$\begin{cases} A = X_1^2, \quad B = X_1 A, \quad C = Y_1^2, \quad D = Z_1^2, \quad E = Z_1 D, \quad T_1 = Y_1(E + B + \alpha C), \\ X_3 = T_{1,0}, \quad T_2 = T_{1,1}, \quad Y_3 = X_1(T_2 + E), \quad Z_3 = Z_1(B + T_2) \end{cases} \tag{36}$$

The formula in (36) requires $4\mathbf{M}_2$, $4\mathbf{R}_2$, $\mathbf{M}_{2,4}$, $\mathbf{R}_4$, $3\mathbf{S}$, and $3\mathbf{add}$.

7 Complexity comparison

The cost of point addition and doubling are summarized in Table 2.

TABLE 2: Cost of point addition and point doubling

Operation	Curve	Coordinate	Old complexity	New complexity
Point addition	Weierstrass	López-Dahab	$13\mathbf{M}_2, 13\mathbf{R}_2, 4\mathbf{S}, 9\mathbf{add}$	$7\mathbf{M}_2, 3\mathbf{M}_{2,4}, 7\mathbf{R}_2, 3\mathbf{R}_4, 4\mathbf{S}, 9\mathbf{add}$
		Extended López-Dahab	$8\mathbf{M}_2, 8\mathbf{R}_2, 4\mathbf{S}, 8\mathbf{add}$	$6\mathbf{M}_2, 1\mathbf{M}_{2,4}, 6\mathbf{R}_2, 1\mathbf{R}_4, 4\mathbf{S}, 8\mathbf{add}$
		Projective	$14\mathbf{M}_2, 14\mathbf{R}_2, 1\mathbf{S}, 1\mathbf{d}, 7\mathbf{add}$	$4\mathbf{M}_2, 5\mathbf{M}_{2,4}, 4\mathbf{R}_2, 5\mathbf{R}_4, 1\mathbf{S}, 1\mathbf{d}, 7\mathbf{add}$
		Jacobian	$14\mathbf{M}_2, 14\mathbf{R}_2, 5\mathbf{S}, 1\mathbf{d}, 7\mathbf{add}$	$4\mathbf{M}_2, 5\mathbf{M}_{2,4}, 4\mathbf{R}_2, 5\mathbf{R}_4, 5\mathbf{S}, 1\mathbf{d}, 7\mathbf{add}$
	Edwards	Affine	$2\mathbf{I}, 8\mathbf{M}_2, 8\mathbf{R}_2, 2\mathbf{S}, 3\mathbf{d}, 16\mathbf{add}$	$2\mathbf{I}, 4\mathbf{M}_2, 2\mathbf{M}_{2,4}, 4\mathbf{R}_2, 2\mathbf{R}_4, 2\mathbf{S}, 3\mathbf{d}, 16\mathbf{add}$
		Projective	$21\mathbf{M}_2, 21\mathbf{R}_2, 1\mathbf{S}, 4\mathbf{d}, 15\mathbf{add}$	$11\mathbf{M}_2, 5\mathbf{M}_2, 11\mathbf{R}_2, 5\mathbf{R}_{2,4}, 1\mathbf{S}, 4\mathbf{d}, 15\mathbf{add}$
	Hessian	Projective	$12\mathbf{M}_2, 12\mathbf{R}_2, 3\mathbf{add}$	$6\mathbf{M}_{2,4}, 6\mathbf{R}_4, 3\mathbf{add}$
Point doubling	Weierstrass	Projective	$7\mathbf{M}_2, 7\mathbf{R}_2, 3\mathbf{S}, 1\mathbf{d}, 4\mathbf{add}$	$1\mathbf{M}_2, 3\mathbf{M}_{2,4}, 1\mathbf{R}_2, 3\mathbf{R}_4, 3\mathbf{S}, 1\mathbf{d}, 4\mathbf{add}$
	Edwards	Affine	$1\mathbf{I}, 2\mathbf{M}_2, 2\mathbf{R}_2, 4\mathbf{S}, 2\mathbf{d}, 10\mathbf{add}$	$1\mathbf{I}, 1\mathbf{M}_{2,4}, 1\mathbf{R}_4, 4\mathbf{S}, 2\mathbf{d}, 10\mathbf{add}$
		Projective	$2\mathbf{M}_2, 2\mathbf{R}_2, 6\mathbf{S}, 2\mathbf{d}, 9\mathbf{add}$	$1\mathbf{M}_{2,4}, 1\mathbf{R}_4, 6\mathbf{S}, 2\mathbf{d}, 9\mathbf{add}$
	Hessian	Projective	$6\mathbf{M}_2, 6\mathbf{R}_2, 3\mathbf{S}, 3\mathbf{add}$	$4\mathbf{M}_2, 1\mathbf{M}_{2,4}, 4\mathbf{R}_2, 1\mathbf{R}_4, 3\mathbf{S}, 3\mathbf{add}$

Note that to the best of our knowledge the complexity in (7) is the most efficient algorithm for $\mathbf{M}_{2,4}$. On the other hand, the algorithms in [2] and [5] for $\mathbf{M}_2$ are the best known algorithms for three-way split multiplications with five and six multiplications, respectively. The total arithmetic cost of point addition and doubling obtained by using those multiplication formulas are summarized in Table 3 in which the costs are given in terms of $n^{\log_3 5}$ and $n^{\log_3 6}$. Moreover, we assume that $I = 10M$ as in the case [3]. In order to compare the complexity results, we report in Table 4 the complexities for $n = 163, 233, 283, 409, 571$, the values recommended by NIST. As none of the values of n is not divisible by three, we pad one or

two zero bits to polynomials to make the size divisible by three. This process is applied in each recursion as needed.

As it can be seen in Tables 4 and 5, the proposed formula requires fewer bit-level operations than the corresponding previous method for each n. The improvements vary from about 4.43% to 29.65% for the values considered here, and in general they are relatively higher for the three-way scheme with five recursive multiplications than six.

8 POINT MULTIPLICATION

Let k be a positive integer and P be a point on the elliptic curve. The operation kP is called point multiplication and it is the main operation in elliptic curve cryptosystems. One of the widely used efficient point multiplications is the window NAF method that requires point addition and point doubling. The expected cost of this formula is $(1\mathbf{D} + (2^{w-2} - 1)\mathbf{A}) + (n/(w+1)\mathbf{A} + n\mathbf{D})$, where w is the window width. When we choose $w = 4$ as in [8], the total cost of point multiplication is $(\frac{n}{5} + 3)\mathbf{A} + (n+1)\mathbf{D}$.

In order to compare the proposed method with previous ones in the computation of point multiplication, below we consider the projective coordinates for Weierstrass curves. The cost of point multiplication with the previous formulas in (16) and (29) is

$$(49n/5 + 49)\mathbf{M}_2 + (49n/5 + 49)\mathbf{R}_2 + (16n/5 + 6)\mathbf{S} + (6n/5 + 4)\mathbf{d} + (27n/5 + 25)\mathbf{add}. \tag{37}$$

When we use the proposed algorithms in (17) and (30), the cost of point multiplication becomes

$$\begin{aligned}&(9n/5 + 13)\mathbf{M}_2 + (4n + 18)\mathbf{M}_{24} + (4n + 18)\mathbf{R}_2 + (9n/5 + 13)\mathbf{R}_4 + (16n/5 + 6)\mathbf{S}+\\ &(6n/5 + 4)\mathbf{d} + (27n/5 + 25)\mathbf{add}.\end{aligned} \tag{38}$$

In Table 6, we list the exact arithmetic complexities for point multiplication for all five NIST recommended values of n. For comparison, we give the corresponding complexity results for binary NAF based point multiplication assuming windows sizes of 1 and 4. We should note that the complexity results presented in the table can be further reduced by applying various implementation techniques that can be found in [2] (e.g. stopping the recursion earlier and using look-up tables).

The rightmost column of Table 6 shows improvements that we have observed in actual software implementation. Our software ran on Intel Core i7 with 2.8 GHz and 8.00 GB memory. We have used Windows 7 Professional and the C# programming language is used under visual studio 2010 development tool with .Net framework 4.0. The main objective of software of implementation is to compare results under different configuration rather than achieving the maximum speed. In the padding method, during any recursion of the field multiplication algorithm if the size of the polynomials involved is not a multiple of three, then one or two zeros are padded to the polynomials so that their size is increased to the next higher integer divisible by three. For example, in the case of $n = 163$, the sizes are 165, 57, 21 and 9 in iterations 1, 2, 3 and 4, respectively. In our implementation, when the size of the polynomials become 9 or less, instead of the proposed or previous multiplication algorithm, a look-up table is used for polynomial

TABLE 3: Total arithmetic cost of point addition and doubling with three-way split multiplication algorithms

Operation	Curve	Coordinate	Multiplication algorithm	Old	New
Point addition	Weierstrass	López-Dahap	Three-way with 6 multiplications in [5]	$81.51n^{\log_3 6} - 9.71n - 49.8$	$43.89n^{\log_3 6} + 92.25n^{\log_3 5} - 71.69n - 6.99n\log_3 n - 26.7$
			Three-way with 5 multiplications in [2]	$331.5n^{\log_3 5} - 294.5n - 15$	$270.75n^{\log_3 5} - 203.5n - 6.99n\log_3 n - 29.5$
		Extended López-Dahap with $a_2 = 1$	Three-way with 6 multiplications in [5]	$50.16n^{\log_3 6} + 2.64n - 36.8$	$37.62n^{\log_3 6} + 30.75n^{\log_3 5} - 18.02n - 2.33n\log_3 n - 29.1$
			Three-way with 5 multiplications in [2]	$204n^{\log_3 5} - 148n - 40$	$183.75n^{\log_3 5} - 131n - 2.33n\log_3 n - 31.5$
		Projective	Three-way with 6 multiplications in [5]	$87.78n^{\log_3 6} - 16.38n - 40.4$	$25.08n^{\log_3 6} + 153.75n^{\log_3 5} - 119.68n - 11.65n\log_3 n - 26.7$
			Three-way with 5 multiplications in [2]	$357.5n^{\log_3 5} - 280n - 46$	$255.75n^{\log_3 5} - 195n - 11.65n\log_3 n - 3.5$
		Jacobian	Three-way with 6 multiplications in [5]	$87.78n^{\log_3 6} - 0.38n - 56.4$	$25.08n^{\log_3 6} + 153.75n^{\log_3 5} - 103.68n - 11.65n\log_3 n - 17.9$
			Three-way with 5 multiplications in [2]	$357n^{\log_3 5} - 264n - 62$	$255.75n^{\log_3 5} - 179n - 11.65n\log_3 n - 19.5$
	Edwards	Affine	Three-way with 6 multiplications in [5]	$175.56n^{\log_3 6} - 20.76n - 80.8$	$150.48n^{\log_3 6} + 61.50n^{\log_3 5} - 4.66n\log_3 n - 62.08n - 65.4$
			Three-way with 5 multiplications in [2]	$714.0n^{\log_3 5} - 548.0n - 92$	$673.50n^{\log_3 5} - 4.66n\log_3 n - 514n - 75$
		Projective	Three-way with 6 multiplications in [5]	$131.67n^{\log_3 6} + 2.93n - 58.6$	$68.97n^{\log_3 6} + 153.75n^{\log_3 5} - 100.37n - 11.65n\log_3 n - 20.1$
			Three-way with 5 multiplications in [2]	$535n^{\log_3 5} - 392.5n - 67$	$434.25n^{\log_3 5} - 307.5n - 11.65n\log_3 n - 24.5$
	Hessian	Projective	Three-way with 6 multiplications in [5]	$75.24n^{\log_3 6} - 29.04n - 31.2$	$184.50n^{\log_3 5} - 13.98n\log_3 n - 153n - 15$
			Three-way with 5 multiplications in [2]	$306^{\log_3 5} - 255n - 36$	$184.50n^{\log_3 5} - 13.98n\log_3 n - 153n - 15$
Point doubling	Weierstrass	Projective	Three-way with 6 multiplications in [5]	$43.89n^{\log_3 6} + 7.31n - 30.2$	$6.27n^{\log_3 6} + 92.25n^{\log_3 5} - 54.67n - 6.99n\log_3 n - 7.1$
			Three-way with 5 multiplications in [2]	$178.5n^{\log_3 5} - 124.5n - 33$	$117.75n^{\log_3 5} - 6.99n\log_3 n - 73.5n - 7.5$
	Edwards	Affine	Three-way with 6 multiplications in [5]	$75.24n^{\log_3 6} + 13.96n - 47.2$	$62.70n^{\log_3 6} + 30.75n^{\log_3 5} - 2.33n\log_3 n - 0.70n - 35.5$
			Three-way with 5 multiplications in [2]	$306n^{\log_3 5} - 212n - 52$	$285.75n^{\log_3 5} - 2.33n\log_3 n - 189n - 39.5$
		Projective	Three-way with 6 multiplications in [5]	$12.54n^{\log_3 6} + 47.66n - 29.2$	$30.75n^{\log_3 5} - 2.33n\log_3 n - 27n - 21.5$
			Three-way with 5 multiplications in [2]	$51n^{\log_3 5} - 10n - 30$	$30.75n^{\log_3 5} - 2.33n\log_3 n - 27n - 21.5$
	Hessian	Projective	Three-way with 6 multiplications in [5]	$37.62n^{\log_3 6} - 10.98n - 39.6$	$25.8n^{\log_3 6} + 30.7525.8n^{\log_3 5} - 2.33n\log_3 n - 21.68n - 19.9$
			Three-way with 5 multiplications in [2]	$153^{\log_3 5} - 102n - 42$	$132.75n^{\log_3 5} - 2.33n\log_3 n - 97n - 21.5$

multiplication. The experiment for each configuration has been done 100 times and then the number of clock cycles is averaged over 100.

TABLE 4: Total arithmetic cost of point addition for Weierstrass curves with three-way split multiplication algorithms, where A: Three-way with 6 multiplications in [5] and B: Three-way with 5 multiplications in [2].

Curve	Coordinate	n	Multiplication algorithm	Complexities		Improvement %
				Previous	Proposed	
Weierstrass	López-Dahab	163	A	586966	558070	4.92
			B	890971	721765	18.99
		233	A	627976	600130	4.43
			B	967965	783201	19.09
		283	A	2877030	2553915	11.23
			B	3605251	2946034	18.28
		409	A	3078612	2747832	10.74
			B	3937834	3210490	18.47
		571	A	3638646	3252315	10.62
			B	4725355	3837466	18.79
	Extended López-Dahab with $a_2 = 1$	163	A	362608	352976	2.66
			B	549688	493286	10.26
		233	A	388448	379166	2.39
			B	597672	536084	10.30
		283	A	1772912	1665207	6.07
			B	2221048	2001309	9.89
		409	A	1898048	1787788	5.81
			B	2426800	2184352	9.99
		571	A	2244080	2115303	5.74
			B	2912824	2616861	10.16
	Projective	163	A	629535	581375	7.65
			B	956925	674915	29.47
		233	A	672585	626175	6.90
			B	1038727	730787	29.65
		283	A	3093847	2555322	17.41
			B	3878085	2779390	28.33
		409	A	3308929	2757629	16.66
			B	4234245	3022005	28.63
		571	A	3909463	3265578	16.47
			B	5079765	3599950	29.13
	Jacobian	163	A	630187	582027	7.64
			B	957577	675567	29.45
		233	A	673517	627107	6.89
			B	1039659	731719	29.62
		283	A	3094979	2556454	17.40
			B	3879217	2780522	28.32
		409	A	3310565	2759265	16.65
			B	235881	3023641	28.62
		571	A	3911747	3267862	16.46
			B	5082049	3602234	29.12

9 Conclusion

In this paper we have considered point addition and doubling for Weierstrass, Edwards and Hessian curves over binary fields. Specifically, we have showed that binary elliptic curve point addition and doubling formulas can be improved by using the polynomial multiplication algorithm in $\mathbb{F}_4$. We have also given the exact complexities of point addition and point multiplication for NIST curves by using three-way split polynomial multiplication algorithms. Moreover, we have obtained the cost of point multiplication for the Weierstrass curves in projective coordinates by using the proposed method. We have computed complexities by using zero-padding technique for polynomial multiplication until the sizes of polynomials become 1 bit. One can stop the recursion before the sizes become 1 bit and can use

TABLE 5: Total arithmetic cost of point addition for Edwards and Hessian curves with three-way split multiplication algorithms, where A: Three-way with 6 multiplications in [5] and B: Three-way with 5 multiplications in [2].

Curve	Coordinate	n	Multiplication algorithm	Complexities		Improvement %
				Previous	Proposed	
Edwards	Affine	163	A	1264286	1245022	1.52
			B	1919066	1806262	5.88
		233	A	1352626	1334062	1.37
			B	2084910	1961734	5.91
		283	A	6196750	5981340	3.48
			B	7765226	7325748	5.66
		409	A	6630946	6410426	3.33
			B	8481578	7996682	5.72
		571	A	7837198	7579644	3.29
			B	10177802	9585876	5.82
	Projective	163	A	951232	903072	5.06
			B	1442317	1160307	19.55
		233	A	1018782	972372	4.55
			B	1567995	1260055	19.64
		283	A	4652800	4114275	11.57
			B	5829157	4730462	18.85
		409	A	4980778	4429478	11.07
			B	6368752	5156512	19.03
		571	A	5888464	5244579	10.93
			B	7643917	6164102	19.36
Hessian	Projective	163	A	538557	480765	10.73
			B	819177	480765	41.31
		233	A	575007	519315	9.68
			B	888843	519315	41.57
		283	A	2650053	2003823	24.38
			B	3322257	2003823	39.68
		409	A	2833599	2172039	23.35
			B	3626727	2172039	40.11
		571	A	3347301	2574639	23.08
			B	4350417	2574639	40.82

other algorithms like schoolbook or can use look-up tables in order to improve the complexities further. Moreover, we have implemented the point multiplication for NIST curves and have showed that the proposed algorithm reduces the time needed to perform point multiplication.

We note that the proposed method also gives improvements when *two*-way multiplication is used in a polynomial multiplications for larger n, e.g., $n > 512$. The requirement for such large n is due to the higher number of recursions in two-way than three-way split method. It should also be noted that one can obtain even better results over the two-way multiplication by stopping the recursion before the sizes become 1 bit and using other algorithms or using look-up tables. This kind of tricks are especially useful in implementations.

Finally, we would like to note that the idea presented in Section 4 can be applied to other computational applications and, to this end we plan on presenting our work on hyper elliptic curve and pairing based cryptographic systems in a future technical report.

REFERENCES

[1] D. J. Bernstein, Tanja Lange, and Reza Rezaeian Farashahi. Binary edwards curves. In *CHES*, pages 244–265, 2008.

TABLE 6: Point multiplication in projective coordinates for Weierstrass curves, where A: Three-way with 6 multiplications in [5], B: Three-way with 5 multiplications in [2], I: Binary NAF method for point multiplication in [9], and II: Window NAF method for point multiplication with $w = 4$ in [9].

n	Multiplication algorithm	Point multiplication algorithm	Complexities		Improvement % (Theoretical)	Improvement % (Software impl.)
			Previous	Proposed		
163	A	I	8.6×10^7	7.88×10^7	8.50	7.05
		II	7.46×10^7	6.79×10^7	8.96	2.56
	B	I	1.31×10^8	8.77×10^7	32.85	40.23
		II	1.13×10^8	7.51×10^7	33.61	9.86
233	A	I	1.32×10^8	1.22×10^8	7.65	7.41
		II	1.13×10^8	1.04×10^8	8.21	1.9
	B	I	2.03×10^8	1.36×10^8	32.99	57.65
		II	1.74×10^8	1.15×10^8	33.85	20.05
283	A	I	7.31×10^8	5.89×10^8	19.44	16.22
		II	6.25×10^8	5.01×10^8	19.91	4.16
	B	I	9.16×10^8	6.26×10^8	31.66	33.43
		II	7.83×10^8	5.30×10^9	32.33	8.87
409	A	I	1.13×10^9	9.21×10^8	18.59	16.81
		II	9.63×10^8	7.79×10^8	19.10	6.73
	B	I	1.45×10^9	9.84×10^8	31.98	40.43
		II	1.23×10^9	8.28×10^8	32.70	10.45
571	A	I	1.87×10^9	1.52×10^9	18.37	17.62
		II	1.58×10^9	1.28×10^9	18.90	4.16
	B	I	2.42×10^9	1.63×10^9	32.52	58.39
		II	2.05×10^9	1.37×10^9	32.28	14.45

[2] D.J. Bernstein. Batch Binary Edwards. In *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *LNCS*, pages 317–336, 2009.

[3] D.l J. Bernstein and T. Lange. Explicit-formulas database. http://www.hyperelliptic.org/EFD/. (accessed 2012-01-25).

[4] M. Cenk, C. Negre, and M. A. Hasan. Improved three-way split formulas for binary polynomial multiplication. In *Selected Areas in Cryptography*, pages 384–398, 2011.

[5] M. Cenk, C. Negre, and M. Anwar Hasan. Improved three-way split formulas for binary polynomial and toeplitz matrix vector products. Technical Report 30, CACR, Waterloo, ON, Canada, 2011. See http://www.cacr.math.uwaterloo.ca/techreports/2011/cacr2011-30.pdf.

[6] M. Cenk and F. Özbudak. Improved Polynomial Multiplication Formulas over $\mathbb{F}_2$ Using Chinese Remainder Theorem. *IEEE Trans. Computers*, 58(4):572–576, 2009.

[7] Haining Fan and M. A. Hasan. Comments on "Five, Six, and Seven-Term Karatsuba-Like Formulae". *IEEE Trans. Computers*, 56(5):716–717, 2007.

[8] D. Hankerson, J. L. Hernandez, and A. Menezes. Software implementation of elliptic curve cryptography over binary fields. In *CHES*, pages 1–24, 2000.

[9] D. Hankerson, D. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag, 2004.

[10] A. A. Karatsuba. Multiplication of multidigit numbers on automata. *Soviet Physics Doklady*, 7:595–596, 1963.

[11] K. H. Kim and S. I. Kim. A new method for speeding up arithmetic on elliptic curves over binary fields. *IACR Cryptology ePrint Archive*, 2007:181, 2007.

[12] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):pp. 203–209.

[13] T. Lange. A note on López-Dahab coordinates. Cryptology ePrint Archive, Report 2004/323, 2004. urlhttp://eprint.iacr.org/.

[14] A. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[15] V. S. Miller. Use of elliptic curves in cryptography. In *CRYPTO*, pages 417–426, 1985.

[16] P. L. Montgomery. Five, six, and seven-term karatsuba-like formulae. *IEEE Trans. Computers*, 54(3):362–369, 2005.

[17] B. Sunar. A generalized method for constructing subquadratic complexity GF(2^k) multipliers. *IEEE Transactions on Computers*, 53:1097–1105, 2004.

[18] L. C. Washington. *Elliptic Curves: Number Theory and Cryptography*. Chapman & Hall/CRC, 2nd edition, 2008.

[19] S. Winograd. *Arithmetic Complexity of Computations*. Society For Industrial & Applied Mathematics, U.S., 1980.

[20] H. Wu. Low complexity bit-parallel finite field arithmetic using polynomial basis. In *CHES*, pages 280–291, 1999.