Fast Scalar Multiplication for Elliptic Curves over Binary Fields by Efficiently Computable Formulas

Saud Al Musa and Guangwu Xu *

Department of EE & CS,
University of Wisconsin-Milwaukee, USA,
{salmusa,gxu4uwm}@uwm.edu

Abstract. This paper considers efficient scalar multiplication of elliptic curves over binary fields with a twofold purpose. Firstly, we derive the most efficient $3P$ formula in λ-projective coordinates and $5P$ formula in both affine and λ-projective coordinates. Secondly, extensive experiments have been conducted to test various multi-base scalar multiplication methods (e.g., greedy, ternary/binary, multi-base NAF, and tree-based) by integrating our fast formulas. The experiments show that our $3P$ and $5P$ formulas had an important role in speeding up the greedy, the ternary/binary, the multi-base NAF, and the tree-based methods over the NAF method. We also establish an efficient $3P$ formula for Koblitz curves and use it to construct an improved set for the optimal pre-computation of window TNAF.

Keywords: binary elliptic curves, point multiplication, lambda coordinates, efficient formulas, DBNS, MBNS

1 Introduction

Koblitz and Miller first introduced the use of an elliptic curve in public key cryptography [19, 26]. What makes an elliptic curve cryptosystem attractive for use is that it has a shorter key length, and it is as secure as the larger key length in other public key cryptosystems. For instance, the shorter key length 283 bits in an elliptic curve cryptosystem is regarded as secure as the larger key length 3072 bits in an RSA cryptosystem [21].

The dominant operation in elliptic curve cryptography (ECC) cryptographic schemes is the scalar multiplication, which is an operation that adds a point to itself a large number of times. The research to increase the speed of this operation has attracted considerable attention ever since the discovery of the ECC. Many proposed methods have improved general exponentiation algorithms. The idea of presenting a scalar in a non-adjacent form (NAF) with signed coefficients has been a basic method to use. The sparse property of NAF participates

* Research supported in part by the National 973 Project of China (No. 2013CB834205).

for minimizing the total number of point addition operations. Furthermore, the NAF method can be used with window width. The binary width-w NAF allows coefficients to be in $\{0, \pm 1, \pm 3, \cdots, \pm 2^{w-1} - 1\}$, which makes the representation sparser. Hence, the binary width-w NAF needs to pefrom an online pre-computation for $2^{w-2} - 1$ points. For a length l scalar, the average running time for this method is approximately $\frac{l}{w+1}$ point addition and l point doubling. See [6, 17] for more details of the NAF and the window NAF methods.

Another faster method for scalar multiplication is to convert a scalar to double-base number system (DBNS). The DBNS with ternary and binary bases for scalar n is represented such that $n = \sum s\ 2^a\ 3^b$ where $a, b \geqslant 0, s \in \{-1, +1\}$. A natural extension of DBNS is multi-base number system (MBNS). The main advantage of using MBNS is that the scalar has a shorter average expansion length than its single-base average expansion length. As a result, the total number of point additions are minimized and that leads to a faster scalar multiplication operation. A greater computation speed can be achieved if an efficient formula is available for scalar multiplication by an integer in the base. There are several methods for representing an integer in MBNS, including greedy method first proposed by Dimitrov, Imbert, and Mishra [8], ternary/binary method developed by Ciet, Joye, Lauter, and Montgomery [7], tree-based method given by Doche and Habsieger [10], and multi-base NAF introduced by Longa in [22]. A scalar multiplication using MBNS expansion has been further researched in [3, 2, 9, 11, 23, 27, 25, 33].

In [18], Koblitz proposed a class of binary curves, what are now called Koblitz curves, for cryptographic use. [18] also initiated a study of NAF of some algebraic integers using the Frobenius map τ. A very important extension to Koblitz's result was the window TNAF by Solinas [29] which reduces the computation for scalar multiplication dramatically. More computational properties of the window TNAF have been revealed by Blake, Murty and Xu [4, 5]. Recently, Trost and Xu formulated and constructed an optimal pre-computation for window TNAF [30]. Some new formulas have been derived in [30] that require a fewer number of field operations. For some curves over prime fields, Gallant, Lambert, and Vanstone proposed the GLV method for efficient scalar multiplication which makes use of endomorphisms that can be computed efficiently [14]. Working over the field $\mathbb{F}_{p^2}$, Galbraith, Lin and Scott constructed efficiently computable endomorphisms for a large family of elliptic curves by using twists. They also demonstrated that the GLV method for these curves is much faster, see [13]. This GLS construction was extended to binary fields by Hankerson, Karabina, and Menezes in [15] and showed a great speed improvement as well.

Inversion operation is a very expensive operation in finite fields. The inversion to multiplication ($\mathbf{I}/\mathbf{M}$) ratio over binary fields is not fixed, and it gets effected by the used inversion algorithm on different computing platforms [16]. We usually assume the low $\mathbf{I}/\mathbf{M}$ ratio is 5, and the high $\mathbf{I}/\mathbf{M}$ ratio is 8 as suggested in [17]. Much effort on elliptic curve arithmetic has been made in working on different coordinate systems to avoid field inversion and achieve efficiency. A common way of avoiding expensive division is to change to projective coordi-

nate systems. Besides standard projective coordinates, Jacobian and Chudnosky projective coordinate systems are also used for general curves. For curves over binary fields, López-Dahab (LD) coordinates [24] is a very efficient alternative. Recently, Oliveira, López, Aranha, and Rodríguez-Henríquez proposed a more efficient coordinate system for binary elliptic curves—the lambda representation (λ- coordinates) [28].

Devising efficient elliptic curve operations with small scalars have been of significant interest, e.g., reducing number of field operations for computing $3P$ and $5P$ for an elliptic curve point P. These operations are key to the fast computation by using DBNS and multi-base number representation. In [7], Ciet, Joye, Lauter, and Montgomery presented an efficient formula for $3P$ for both prime curves and binary curves, and it takes 1 field inversion ($\mathbf{I}$), 4 squarings ($\mathbf{S}$), and 7 multiplications ($\mathbf{M}$): $1\mathbf{I}+7\mathbf{M}+4\mathbf{S}$. A ternary/binary algorithm was also designed in [7] that utilizes $3P$ with an improved speed over the NAF method. For curves over a binary field, Dimitrov, Imbert, and Mishra gave an improved $3P$ formula that requires $1\mathbf{I}+6\mathbf{M}+3\mathbf{S}$ [8]. The most efficient formulas for computing $3P$ were given by Yu, Kim, and Jo in [32], their formula for binary field only needs $1\mathbf{I}+5\mathbf{M}+2\mathbf{S}$. In [27], Mishra and Dimitrov proposed the Multi-Base Number Representation for scalar multiplication. They derived an efficient $5P$ formula for binary curves with a small number of operations: $1\mathbf{I}+13\mathbf{M}+5\mathbf{S}$. Some concise formulas for Koblitz curves can be found in [30], e.g., for $(1-\tau)P$ and $(1+\tau)P$.

1.1 Our Contribution

In this paper, we consider the problem of fast scalar multiplication for binary elliptic curves. The main contribution of the paper is twofold. In the first part, we derive $3P$ and $5P$ efficient formulas for binary elliptic curves. In affine coordinates, our improved $5P$ formula uses $1\mathbf{I}+11\mathbf{M}+6\mathbf{S}$. Under the very promising λ-projective coordinate systems, we are able to set up efficient computation for $3P$ and $5P$ and their formulas cost $8\mathbf{M}+1\mathbf{M}_a+5\mathbf{S}$ and $13\mathbf{M}+1\mathbf{M}_a+8\mathbf{S}$ respectively, here $\mathbf{M}_a$ denotes the cost of multiplication of a general field element with a fixed field element a (which is usually a coefficient of elliptic curve and has a small size). The derivation techniques for our $3P$ and $5P$ efficient formulas in λ-projective coordinates are not based on the $3P$ and $5P$ efficient formulas in affine coordinates. λ-coordinates system has its own affine coordinates, which is called λ-affine coordinates. Thus, it is necessary to find first a formula in λ-affine coordinates that leads to an efficient formula in λ-projective coordinates. The derived $3P$ and $5P$ efficient formulas in this paper are state of the art, and to the best of our knowledge, we are the first to present them in λ-projective coordinates. More precisely, our $3P$ λ-projective coordinates greatly improves that using LD projective coordinates [31] and Jacobian projective coordinates [8]. A projective coordinate formula for $5P$ seems not available in literature.

The second part of our contribution is conducting extensive performance comparison tests for the MBNS methods in λ-coordinates. The MBNS methods are one of the best applications that shows the importance of our $3P$ and $5P$ formulas in speeding up scalar multiplication operations. The investigated MBNS

methods are the greedy, the ternary/binary, the multi-base NAF, and the tree-based [8, 7, 23, 10]. Our tests compare these methods using our efficient formulas with respect to three characteristics: the expansion length, the total number of multiplications, and the running time. Other comparison tests in [8, 7, 23, 10] emphasize the expansion length and the total number of multiplications. We include the running time test since it takes into account the time of converting integer n to a multi-base chain. To the best of our knowledge, we are the first study that compares the performance of the MBNS methods with the NAF method in λ-coordinates.

Our comparison test in terms of the total number of multiplications shows the greedy, the ternary/binary, the multi-base NAF, and the tree-based methods speed up to $10\%, 8\%, 12\%$, and 15% over the NAF method. Our running time test shows they speed up to $7\%, 9\%, 12\%$, and 15% over the NAF method. The running time test of the greedy method gives less percentage of improvement than the comparison test in terms of the total number of multiplications. The reason for that is the running time test considers the time of converting integer n to a multi-base chain, which implies the greedy method has a higher conversion cost than other MBNS methods.

Some of the ideas for computing $3P$ also lead an improvement of the optimal pre-computation of window TNAF for Koblitz curves [30]. By efficient formulas for the pre-computed points in the forms of $P - \tau(P), P + \tau(P), P - \tau^2(P)$ and $3P$ and working with the λ-projective coordinates, we show the performance of the optimal pre-computation of window TNAF with these efficient formulas gets $48\%, 24\%$ and 11% faster for window width $4, 5$ and 6 respectively.

The rest of the paper is organized into five sections. Efficient formulas for $3P$ and $5P$ are given in section 2. In section 3, we briefly review several existing MBNS methods, and we conduct comparison tests for these methods using our efficient formulas. In section 4, we briefly review the optimal pre-computation of window TNAF for Koblitz curves, and we propose $3P$ efficient formula to the improved set of the optimal pre-computation of window TNAF with experiments. Finally, the paper is concluded in section 5.

2 Formulas for $3P$ and $5P$ on Binary Elliptic Curves

An elliptic curve E over a binary field $\mathbb{F}_{2^m}$ can be represented by the simplified Weierstrass equation

$$E : y^2 + xy = x^3 + ax^2 + b, \tag{1}$$

where $a, b \in \mathbb{F}_{2^m}$ and $b \neq 0$. We denote $E(\mathbb{F}_{2^m})$ to be the set of all points (x, y) with $x, y \in \mathbb{F}_{2^m}$ that satisfy the equation (1) together with the point at infinity $\mathcal{O}$. $E(\mathbb{F}_{2^m})$ forms an abelian group under "+" operation. The identity of the abelian group is the point at infinity $\mathcal{O}$. The point addition can be computed by the chord and tangent method.

One of the main advantages of using binary elliptic curves over prime elliptic curves is that squaring is a linear operation in binary fields. This is the reason for having a low squaring to multiplication ($\mathbf{S}/\mathbf{M}$) ratio in binary fields, and it is

close to a free operation. In prime fields, the $\mathbf{S}/\mathbf{M}$ ratio is higher, and it is close to the cost of one multiplication operation [16]. It is noted that the most expensive operation for binary fields as well as prime fields is the inversion operation. The ratio ($\mathbf{I}/\mathbf{M}$) can be quite big (e.g. 8). One solution for reducing the cost of the inversion operation is to use projective coordinates over affine coordinates. We shall choose to derive efficient formulas in λ-projective coordinate systems as it has been proved to be better than other projective coordinates.

Another solution for reducing the cost of the inversion operation is to use efficient formulas. Efficient formulas in affine coordinates are based on the idea of trading an inversion with multiplication operations for faster performance. Several efficient formulas for Weierstrass equation in affine coordinates have been proposed. In our case, we emphasize the $3P$ and $5P$ efficient formulas since these formulas are frequently used with MBNS methods as discussed in the next section. For affine coordinate systems, the $3P$ formula given in [32] has a very small cost of $1\mathbf{I}+5\mathbf{M}+2\mathbf{S}$ and seems hard to be further improved. In the first subsection, we are able to derive an efficient affine formula for $5P$. For λ-projective coordinate systems, fast formulas for $3P$ and $5P$ seem not available in the literature. In the second subsection of the paper, we develop efficient λ-projective coordinate formulas for $3P$ and $5P$, and these formulas will be incorporated later into MBNS methods to achieve a greater efficiency for scalar multiplication operations.

2.1 A $5P$ Formula in Affine Coordinates

As mentioned earlier, an efficient $5P$ formula under affine coordinate systems for binary elliptic curves has been proposed in [27], with a cost of $1\mathbf{I}+13\mathbf{M}+5\mathbf{S}$ [27]. We propose an improved efficient computation of $5P$ in affine coordinate system. The precise formula of $5P$ is presented by the following theorem.

Theorem 1 *Let $P=(x_P,y_P)\in E(\mathbb{F}_{2^m})$ and $6P\neq\mathcal{O}$. Set*

$$\begin{cases}\alpha=x_P^4+x_P^3+b\\ \beta=\alpha^2+x_P^2(x_P^4+b)\\ \gamma=\alpha^2(x_P^4+b)+x_P^3\beta\end{cases}.$$

Then $5P=(x_{5P},y_{5P})$ is given by

$$x_{5P}=x_P+\frac{x_P^3\beta}{\gamma}+\left(\frac{x_P^3\beta}{\gamma}\right)^2$$
$$y_{5P}=y_P+x_P+(x_{5P}+x_P)\left(\frac{x_P^3\beta}{\gamma}+x_P^2+a\right)+\frac{x_P\beta\alpha^2(\beta+(x_P^4+b)(x_P^4+b+y_P^2+x_P^2))}{\gamma^2}.$$

Remark 1. 1. The proof of the Theorem 1 is presented in Appendix 5.1.

2. Our $5P$ efficient formula in affine coordinates costs $1\mathbf{I}+11\mathbf{M}+6\mathbf{S}$. Operation counts for our $5P$ efficient formula are given in Table 1. A less costly way for computing $5P$ without using such a formula is through $4P+P$ and this way costs $2\mathbf{I}+8\mathbf{M}+6\mathbf{S}$. With our $5P$ efficient formula, we trade $1\mathbf{I}$ with $3\mathbf{M}$ for a faster performance. Our $5P$ formula in affine coordinates saves $2\mathbf{M}$ (but with an extra $\mathbf{S}$ whose cost is low for binary fields) over the proposed $5P$ efficient formula in [27].

Table 1. Operation Counts for our $5P$ in Affine Coordinates

Computing term	Operation counts
$\alpha = x_P^4 + x_P^3 + b$	1**M**+2**S**
$\beta = \alpha^2 + x_P^2(x_P^4 + b)$	1**M**+1**S**
$\gamma = \alpha^2(x_P^4 + b) + x_P^3\beta$	2**M**
$x_{5P} = x_P + \frac{x_P^3\beta}{\gamma} + \left(\frac{x_P^3\beta}{\gamma}\right)^2$	1**I**+1**M**+1**S**
$y_{5P} = y_P + x_P + (x_{5P} + x_P)\left(\frac{x_P^3\beta}{\gamma} + x_P^2 + a\right)$	1**M**
$+\frac{x_P\beta\alpha^2(\beta+(x_P^4+b)(x_P^4+b+y_P^2+x_P^2))}{\gamma^2}$	5**M**+2**S**
	1**I**+11**M**+6**S**

2.2 $3P$ and $5P$ Formulas in λ-Projective Coordinates

The λ-coordinates system is introduced in [28] for elliptic curves over binary fields. λ-coordinates represent affine point $(x, y) \in E(\mathbb{F}_{2^m})$ by (x, λ) where $\lambda = x + \frac{y}{x}$. λ-coordinates represent a projective point by (X, L, Z) and $Z \neq 0$. λ-affine point (x_P, λ_P) is converted to λ-projective point (X_P, L_P, Z_P) by using the relation $(x_P, \lambda_P) = (\frac{X_P}{Z_P}, \frac{L_P}{Z_P})$. This representation for λ-coordinates led to an efficient $P+Q$ formula. The Weierstrass equation for λ-projective coordinates is given in [28] by

$$(L^2 + LZ + aZ^2)X^2 = X^4 + bZ^4.$$

The authors in [28] presented $2P, P+Q$, and $2Q+P$ formulas for λ-coordinates system. In this subsection, we derive efficient formulas for $3P$ and $5P$ in λ-projective coordinates.

Table 2. The Cost for Efficient Formulas in Different Projective Coordinates over Binary Fields

	λ-projective	LD projective	Jacobian projective
$2P$	4**M**+1**M**$_a$+4**S** [28]	4**M**+1**M**$_a$+4**S** [20]	4**M**+1**M**$_b$+5**S** [1]
$P+Q$	11**M**+2**S** [28]	13**M**+4**S** [20]	15**M**+1**M**$_a$+3**S** [1]
$3P$	8**M**+1**M**$_a$+5**S** (*this work*)	10**M**+2**M**$_a$+7**S** [31]	13**M**+2**M**$_{a,b}$+7**S** [8]
$5P$	13**M**+1**M**$_a$+8**S** (*this work*)	n/a	n/a

Theorem 2 *Let $P = (X_P, L_P, Z_P) \in E(\mathbb{F}_{2^m})$. Then $3P = (X_{3P}, L_{3P}, Z_{3P})$ using λ-projective coordinates is given by*

$$\begin{aligned}
T &= L_P^2 + L_P Z_P + aZ_P^2 \\
A &= (T + X_P Z_P)^2 \\
B &= TZ_P^2 + A \\
X_{3P} &= X_P Z_P B^2 \\
Z_{3P} &= Z_P^2 AB \\
L_{3P} &= T(A+B)^2 + (L_P Z_P + Z_P^2)AB.
\end{aligned}$$

Remark 2. 1. The proof of Theorem 2 is presented in Appendix 5.2.
2. The cost of our $3P$ efficient formula in λ-projective coordinates, as Table 3 shows, is $8\mathbf{M}+1\mathbf{M}_a + 5\mathbf{S}$. The less costly way for computing $3P$ in λ-projective coordinates without the $3P$ efficient formula is through $2P + P$ with cost $15\mathbf{M}+1\mathbf{M}_a + 6\mathbf{S}$. With our concise $3P$ formula, we save $7\mathbf{M}$ over $2P + P$. Our $3P$ formula saves $3\mathbf{M}$ over the proposed $3P$ efficient formula in LD-projective coordinates in [31]. It saves $6\mathbf{M}$ over the proposed $3P$ efficient formula in Jacobian projective coordinates in [8]. Table 2 compares the cost of $3P$ in different coordinate systems over binary fields.

Table 3. Operation Counts for our $3P$ in λ-projective Coordinates

Computing term	Operation counts
$T = L_P^2 + L_P Z_P + aZ_P^2$	$1\mathbf{M}+1\mathbf{M}_a+2\mathbf{S}$
$A = (T + X_P Z_P)^2$	$1\mathbf{M}+1\mathbf{S}$
$B = TZ_P^2 + A$	$1\mathbf{M}$
$X_{3P} = X_P Z_P B^2$	$1\mathbf{M}+1\mathbf{S}$
$Z_{3P} = Z_P^2 AB$	$2\mathbf{M}$
$L_{3P} = T(A + B)^2 + (L_P Z_P + Z_P^2)AB$	$2\mathbf{M}+1\mathbf{S}$
	$8\mathbf{M}+ 1\mathbf{M}_a+5\mathbf{S}$

Theorem 3 *Let* $P = (X_P, L_P, Z_P) \in E(\mathbb{F}_{2^m})$. *Then* $5P = (X_{5P}, L_{5P}, Z_{5P})$ *using* λ*-projective coordinates is given by*

$$
\begin{aligned}
&T = L_P^2 + L_P Z_P + aZ_P^2\\
&A = (T + X_P Z_P)^2\\
&B = TZ_P^2 + A\\
&C = (T(A + B))^2 + AB^2\\
&D = A^2 B + AB^2 + C\\
&X_{5P} = X_P Z_P D^2\\
&Z_{5P} = Z_P^2 CD\\
&L_{5P} = T(C + D)^2 + (L_P Z_P + Z_P^2)CD + Z_P^2 (AB)^3.
\end{aligned}
$$

Remark 3. 1. The proof of Theorem 3 is presented in Appendix 5.3.
2. The cost of our $5P$ efficient formula in λ-projective coordinates, as Table 4 shows, is $13\mathbf{M}+1\mathbf{M}_a + 8\mathbf{S}$. The less costly way for computing $5P$ in λ-projective coordinates without the $5P$ efficient formula is through $4P + P$ with cost $19\mathbf{M}+2\mathbf{M}_a+10\mathbf{S}$. With our $5P$ efficient formula, we save $6\mathbf{M}+1\mathbf{M}_a$ over $4P+P$ in λ-projective coordinates. To the best of our knowledge, this is the first proposed $5P$ efficient formula in projective coordinates over binary fields, and it is the most efficient $5P$ formula for binary elliptic curves.

3 MBNS Methods

The simplest and most studied form of the MBNS is the DBNS with $\{2, 3\}$-integers. A positive integer n is represented in the DBNS with $\{2, 3\}$-integers in

Table 4. Operation Counts for our $5P$ in λ-projective Coordinates

Computing term	Operation counts
$T = L_P^2 + L_P Z_P + aZ_P^2$	1**M**+1$\mathbf{M}_a$+2**S**
$A = (T + X_P Z_P)^2$	1**M**+1**S**
$B = TZ_P^2 + A$	1**M**
$C = (T(A + B))^2 + AB^2$	2**M**+2**S**
$D = AB^2 + A^2B + C$	1**M** +1**S**
$X_{5P} = X_P Z_P D^2$	1**M**+1**S**
$Z_{5P} = Z_P^2 CD$	2**M**
$L_{5P} = T(C + D)^2 + (L_P Z_P + Z_P^2)CD + Z_P^2 AB^2 A^2 B$	4**M**+1**S**
	13**M**+1$\mathbf{M}_a$+8**S**

the form of

$$n = \sum_{i=1}^{l} s_i \; 2^{a_i} \; 3^{b_i}$$

where $a_i, b_i \geqslant 0, s_i \in \{-1, +1\}$, and l is the length of the expansion. The MBNS with $\{2, 3, 5\}$-integers is a natural extension to the DBNS with $\{2, 3\}$-integers. A positive integer n in the MBNS with $\{2, 3, 5\}$-integers is represented by

$$n = \sum_{i=1}^{l} s_i \; 2^{a_i} \; 3^{b_i} \; 5^{c_i}$$

where $a_i, b_i, c_i \geqslant 0, s_i \in \{-1, +1\}$, and l is the expansion length. An MBNS expansion for integer n always exists, but it is not unique [8]. What is important to ECC is the property that under MBNS, an integer n has a short average expansion length compared to that of its single-base average expansion length, hence it minimizes the total number of point addition during the point multiplication operation.

In application, when an integer n is represented in MBNS, it has to be represented as a multi-base chain for efficiency reasons. The double-base chain with $\{2, 3\}$-integers is decreasing sequences of the exponents a_i and b_i such that $a_1 \geqslant a_2 \geqslant \cdots \geqslant a_l \geqslant 0$ and $b_1 \geqslant b_2 \geqslant \cdots \geqslant b_l \geqslant 0$. The highest exponents term $2^{a_{max}} \; 3^{b_{max}}$ of a double-base chain is called a leading factor. The leading factor and the expansion length of a double-base chain determine the total number of operations. Thus, they have an important role for minimizing the total number of operations.

Doche in [9] defines an optimal double-base chain with $\{2, 3\}$-integers by the following three requirements. It represents given integer n. It has a leading factor that divides given $2^{a_{max}} \; 3^{b_{max}}$. It has minimum length. For example, let $n = 935811$ and $2^{20} \; 3^{13}$ is given. Then these chains

$$n = 2^{12} \; 3^5 - 2^8 \; 3^5 + 2^5 \; 3^4 + 2^5 \; 3 + 3$$
$$n = 2^7 \; 3^8 + 2^7 \; 3^6 + 2^5 \; 3^4 + 2^5 \; 3 + 3$$
$$n = 2^4 \; 3^{10} - 2^2 \; 3^7 - 3^5 + 3^3 - 3^2$$

are optimal for the following reasons. They have leading factors that divide the given $2^{20}\ 3^{13}$. The length 5 is the shorter double-base chain with $\{2,3\}$-integers that represents n according to the enumeration approach in [9]. One of these optimal chains has a leading factor that is less costly in a particular coordinates system. For example, the leading factor $2^{12}\ 3^5$ is less costly in λ-coordinates. We see that Doche in his definition of an optimal chain emphasizes the optimal length. The optimal length is an important aspect to consider since we generally assume the cost of $P+Q$ formula is high and the cost of $2P$ and $3P$ formulas are low. However, it is necessary to consider the leading factor for the optimal cost. Converting integer n to a double-base chain that has an optimal length on-the-fly is still an open problem [3, 9]. However, efforts were made to propose methods that convert integer n to a shorter double-base chain.

3.1 Review of MBNS Methods

Greedy Method. One of the early methods that convert an integer n to a double-base chain is the greedy method with restricted exponents [8]. The greedy method with $\{2,3\}$-integers has two critical steps. The first step is finding the best approximation for integer n in term of a $\{2,3\}$-integer. A proposed practical solution for finding the best approximation is to use a line search algorithm as presented in [33]. The second critical step is the selection of an upper bound $(a_{max},\ b_{max})$ that minimizes the total number of operations. The selection of these values relies on the cost of $2P, P+Q$, and $3P$ formulas in a particular coordinates system. The best way to get these values is to try different values of $(a_{max},\ b_{max})$ such that $a_{max} + b_{max}\log_2 3 \approx \log_2 n$. Then, the best values that minimized overall cost are selected. For example, Table 7 shows the best upper bound is $(a_{max} = 245,\ b_{max} = 104)$ for the irreducible polynomial of degree 283 in λ-coordinates since this upper bound has lesser cost. The average length of the greedy method without restricted exponents is $\mathcal{O}(\frac{\log n}{\log\log n})$, but with the restricted exponents, the average length is still unproven [10].

Multi-base NAF Method. The multi-base NAF was proposed in [23], and it is a generalization of the single-base NAF method. The multi-base NAF relies on non-adjacent property for a shorter multi-base chain. When integer n is represented in its multi-base NAF, the non-zero digit density becomes less than its single-base NAF [23]. As a result, the total number of point addition in multi-base NAF is minimized, and this leads to a faster scalar multiplication operation. The multi-base NAF has similar properties to the single-base NAF. An integer n in its multi-base NAF is represented in a unique way. Non-zero digits are not consecutive. The average length of the multi-base NAF chain with $\{2,3\}$-integers is approximately $\mathcal{O}(\frac{\log_2 n}{4.1887})$ [23]. The average cost of the multi-base NAF chain with $\{2,3\}$-integers is approximately

$$\log_2 n\ (\frac{1}{4.1887}A + 0.7162\ D + 0.179\ T),$$

where A , D , and T are the cost of $P+Q$, $2P$, and $3P$ formulas respectively [23]. The average length of the multi-base NAF chain with $\{2,3,5\}$-integers is approximately $\mathcal{O}(\frac{\log_2 n}{4.9143})$ [23]. The average cost of the multi-base NAF chain with $\{2,3,5\}$-integers is approximately

$$\log_2 n \ (\frac{1}{4.9143} \ A + 0.6104 \ D + 0.1526 \ T + 0.0635 \ Q),$$

where A, D, T, and Q are the cost of $P+Q$, $2P$, $3P$, and $5P$ formulas respectively [23].

Ternary/Binary Method. The ternary/binary method was proposed in [7] as an efficient scalar multiplication method that outperforms the NAF method. Later in [8, 10, 23], the ternary/binary was studied in the context of a method that converts integer n to a double-base chain with $\{2,3\}$-integers. The ternary/binary method based on the idea of keeping divide integer n by 2 or 3 until n is coprime with 6. Then this method solves this by either $n-1$ or $n+1$, so that 6 divides $n-1$ or $n+1$. This method keeps repeating the process until it reaches 1. The average length of the ternary/binary chain is approximately $\mathcal{O}(\frac{\log_2 n}{4.3774})$[10].The average cost of the ternary/binary chain is approximately

$$\log_2 n \ (\frac{1}{4.3774} \ A + 0.4569 \ D + 0.3427 \ T),$$

where A , D , and T are the cost of $P+Q$, $2P$, and $3P$ formulas respectively [10].

Tree-Based Method. The tree-based method is generalized of the ternary/binary method as proposed in [10]. The key difference between the ternary/binary and the tree-based with $\{2,3\}$-integers methods, when n is coprime with 6, the tree-based method considers both options $n-1$ and $n+1$. While the ternary/binary method considers only one option either $n-1$ or $n+1$. Another difference is that the tree-based method uses the bound size B to control the cost of converting integer n to a multi-base chain. The conversion cost is approximately $2B\log_2(2B)l$ since it needs to sort $2B$ elements, which costs $2B\log_2(2B)$, throughout the expansion length l. For example, the conversion cost for the tree-based with $B=4$ method is $24l$.

Let the bound size $B=1$ in the tree-based method. The average length of the tree-based chain with $\{2,3\}$-integers is approximately $\mathcal{O}(\frac{\log_2 n}{4.6419})$, which is approximately 6% improvement over the length of the ternary/binary method [10]. The average cost of the tree-based chain with $\{2,3\}$-integers is approximately

$$\log_2 n \ (\frac{1}{4.6419} \ A + 0.5569 \ D + 0.2795 \ T),$$

where A , D , and T are the cost of $P+Q$, $2P$, and $3P$ formulas respectively [10]. The average length of the tree-based chain with $\{2,3,5\}$-integers is approximately $\mathcal{O}(\frac{\log_2 n}{5.6142})$ [32]. The average cost of the tree-based chain with $\{2,3,5\}$-integers

is approximately

$$\log_2 n \ (\frac{1}{5.6142} \ A + 0.454 \ D + 0.216 \ T + 0.0876 \ Q),$$

where A, D, T, and Q are the cost of $P+Q$, $2P$, $3P$, and $5P$ formulas respectively [32].

3.2 Experiments

Our goal in these experiments is to compare the MBNS methods with the NAF method in λ-coordinates. The tested MBNS methods are the greedy, the ternary/binary, the multi-base NAF, and the tree-based [8, 7, 23, 10]. Our concise $3P$ and $5P$ formulas in λ-coordinates are utilized in all the tested methods. We denote (2,3)greedy to be the greedy method with restricted exponents in terms of $\{2, 3\}$-integers [8]. (2,3)NAF is the multi-base NAF method with $\{2, 3\}$-integers, and (2,3,5)NAF is the multi-base NAF method with $\{2, 3, 5\}$-integers [23]. (2,3)tree is the tree-based method with $\{2, 3\}$-integers, (2,3,5)tree is the tree-based method with $\{2, 3, 5\}$-integers, and B is the bound size [10].

The environment specifications are in the following descriptions. We used C programming language with GNU C Compiler (GCC) version 4.2. We used Intel Core i7 processor with speed 2.3 GHz. We utilized the binary field operations including: squaring, fast reduction modulo, Extended Euclidean inversion, and right-to-left comb multiplication in [17]. We used GNU Multiple Precision (GMP) library version 6.1 to generate random integers of different sizes [35]. For better accuracy, we recorded the average after trying 1000 random integers in each reading result. We used the NIST binary elliptic curves B-283, B-409, and B-571 [34].

Table 5 and Table 6 show that the tested MBNS methods with our efficient formulas succeed in outperforming the NAF method. Table 5 shows the greedy, the ternary/binary, the multi-base NAF, and the tree-based methods speed up to 10%, 8%, 12%, and 15% over the NAF method. These speed-up results in Table 5 are achieved by comparing only the total number of multiplications with the NAF method. It does not consider the cost of converting integer n to a multi-base chain. The conversion cost may affect the overall performance for some methods. The running time test in Table 6 considers the cost of converting n to a multi-base chain. Table 6 shows the running time of the greedy, the ternary/binary, the multi-base NAF, and the tree-based methods are up to 7%, 9%, 12%, and 15% faster than the NAF method. It shows only the running time of the greedy method has less percentage of improvement than the comparison test in Table 5. It implies converting integer n to a multi-base chain in the greedy method has a higher cost than the ternary/binary, the multi-base NAF, and the tree-based methods.

Table 5 also shows if a method has a shorter expansion length, that does not guarantee it has a lesser number of multiplications. For example, let $n = 1118848774838$, the ternary/binary method returns this chain that represents n

$$2^{16}3^{15} + 2^{15}3^{14} + 2^{14}3^{13} - 2^{13}3^{12} - 2^{10}3^9 + 2^9 3^8 - 2^8 3^7 + 2^7 3^4 - 2^3 3^3 + 2^2 3 + 2.$$

Table 5. Theoretical Comparison between NAF and MBNS Methods in λ-coordinates

	B-283			B-409			B-571		
	l	m	%	l	m	%	l	m	%
NAF	94.77	2173.13		136.57	3136.67		190.77	4381.11	
(2,3)greedy	64.72	1945.81	10.46	92.66	2810.05	10.41	128.32	3925.59	10.39
ternary/binary	65.03	1990.21	8.42	93.92	2883.57	8.06	130.53	4028.01	8.05
(2,3)NAF	67.89	1960.33	9.79	98.09	2834.3	9.63	136.7	3956.86	9.68
(2,3,5)NAF	57.77	1903.77	12.39	83.56	2752.62	12.24	116.81	3846.46	12.2
(2,3)tree$_{B=1}$	61.52	1923.25	11.49	88.39	2779.69	11.38	123.43	3889.15	11.22
(2,3,5)tree$_{B=1}$	50.81	1869.46	13.99	73.07	2707.51	13.68	101.81	3784.64	13.61
(2,3,5)tree$_{B=2}$	47.89	1839.99	15.32	68.85	2662.99	15.1	95.79	3723.41	15.01

l: The average length of the scalar expansion.
m: The average of the total number of multiplications.
%: The speed-up percentage in term of m.

Table 6. Running Time Comparison between NAF and MBNS Methods in λ-coordinates

	B-283		B-409		B-571	
	Time in ms	%	Time in ms	%	Time in ms	%
NAF	32.31		78.96		198.69	
(2,3)greedy	29.83	7.67	72.87	7.71	184.24	7.27
ternary/binary	29.23	9.53	71.73	9.15	179.55	9.63
(2,3)NAF	29.17	9.71	70.57	10.62	177.03	10.9
(2,3,5)NAF	28.16	12.84	69.03	12.57	173.21	12.82
(2,3)tree$_{B=1}$	28.43	12.01	69.86	11.52	174.62	12.11
(2,3,5)tree$_{B=1}$	27.83	13.86	68.52	13.22	171.54	13.66
(2,3,5)tree$_{B=2}$	27.36	15.32	66.81	15.38	168.11	15.39

The length of this chain is 11, and it costs in λ-coordinates $16\times 4+15\times 8+10\times 11 = 294\mathbf{M}$. See Table 2 for the cost of $P+Q$, $2P$, and $3P$ efficient formulas in λ-coordinates. The multi-base NAF with $\{2,3\}$-integers returns this chain that represents n

$$2^{32}3^5 + 2^{30}3^4 - 2^{27}3^4 - 2^{25}3^3 - 2^{21}3^2 - 2^{19}3^2 - 2^{14}3^2 - 2^{11}3^2 - 2^9 3 + 2^6 3 - 2^3 - 2.$$

The length of this chain is 12, and it costs in λ-coordinates $289\mathbf{M}$. This example explains, as Table 5 shows, the ternary/binary method has a shorter average length than the multi-base NAF method with $\{2,3\}$-integers. However, the multi-base NAF with $\{2,3\}$-integers method has a lesser average number of multiplications than the ternary/binary method. In Table 5, the tree-based succeeds in generating a shorter average length than other tested MBNS methods. The tree-based method with bound size $B=1$ does not always produce an optimal chain. For example, let $n = 1118848774838$, the tree-based with $\{2,3\}$-integers and $B=1$ returns this chain that represents n

$$2^{21}3^{12} + 2^{18}3^9 - 2^{17}3^8 + 2^{14}3^7 + 2^8 3^7 + 2^7 3^4 - 2^3 3^3 + 2^2 3 + 2.$$

The length of this chain is 9, and it costs in λ-coordinates 268**M**. According to the enumeration approach in [9], the optimal chain with $\{2,3\}$-integers that represents n is

$$2^{21}3^{12} + 2^{13}3^{12} - 2^{13}3^7 + 2^8 3^7 + 2^7 3^4 - 2^3 3^3 + 2^2 3 + 2.$$

The optimal chain length is 8, and it costs in λ-coordinates 257**M**.

Table 7. The Cost of Greedy Method with Different Values of (a_{max}, b_{max}) in λ-coordinates

B-283				B-409				B-571			
a_{max}	b_{max}	l	m	a_{max}	b_{max}	l	m	a_{max}	b_{max}	l	m
140	91	75.95	2100.73	205	129	108.02	3020.44	285	181	153.14	4251.62
160	79	65.42	1964.21	230	113	94.14	2841.52	325	155	130.04	3954.89
170	72	64.72	1945.81	245	104	92.66	2810.05	345	143	129.09	3923.85
180	65	66.29	1951.83	260	95	95.16	2822.14	365	130	133.01	3945.73
200	53	71.22	1985.8	290	75	103.78	2884.58	405	104	144.96	4034.8
220	40	77.06	2028.37	320	57	111.42	2938.09	445	79	155.84	4113.04
240	27	82.43	2066.38	350	38	119.57	2996.4	485	54	166.89	4192.71
260	15	87.83	2105.48	380	18	128.26	3059.8	525	28	177.83	4270.8

l: The average length of the scalar expansion.
m: The average of the total number of multiplications.

Table 5 also shows the greedy method with $\{2,3\}$-integers has a shorter average length and a lesser average number of multiplications than the ternary/binary and the multi-base NAF method with $\{2,3\}$-integers. However, the greedy method result in Table 5 does not consider the conversion cost nor the effort to select the best upper bound (a_{max}, b_{max}) as Table 7 shows. In Table 7, we tried values from $\frac{\log_2 n}{2}$ to $\log_2 n$ for a_{max} such that $a_{max} + b_{max} \log_2 3 \approx \log_2 n$. We selected $(a_{max} = 170,\ b_{max} = 72)$, $(a_{max} = 245,\ b_{max} = 104)$, and $(a_{max} = 345,\ b_{max} = 143)$ for the irreducible polynomials of degree 283, 409, and 571 respectively. We used a line search algorithm to find the best approximation for integer n in term of a $\{2,3\}$-integer [32]. We did not find it a practical to use a look-up table as proposed in [11]. The look-up table contains off-line pre-computation for all integers n and their corresponding in term of a $\{2,3\}$-integer.

4 The Window TNAF for Koblitz Curves

Koblitz introduced in [19] an efficiently computable endomorphism with a special class of elliptic curves. Koblitz defined the special class of curves E_a over binary fields $\mathbb{F}_{2^m}$ by

$$E_a : y^2 + xy = x^3 + ax^2 + 1, \tag{2}$$

where $a \in \{0, 1\}$. We denote $E_a(\mathbb{F}_{2^m})$ to be the set of all points (x, y) that satisfy the equation (2), plus the point of infinity $\mathcal{O}$. The properties of Koblitz curves allow a scalar multiplication to use the *Frobenius* map instead of point doubling. The *Frobenius* map $\tau : E_a(\mathbb{F}_{2^m}) \to E_a(\mathbb{F}_{2^m})$ is defined by

$$\tau(x, y) = (x^2, y^2),$$
$$\tau(\mathcal{O}) = \mathcal{O}.$$

One property of E_a is that $\tau^2(P) + 2P = \mu\tau(P)$, for all $P \in E_a(\mathbb{F}_{2^m})$, where $\mu = (-1)^{1-a}$. This means τ can be considered to be a complex number that satisfies $\tau^2 + 2 = \mu\tau$. By solving $\tau^2 - \mu\tau + 2 = 0$, there is a choice for

$$\tau = \frac{\mu + \sqrt{-7}}{2}.$$

Let $\mathbb{Z}[\tau]$ be a ring of polynomials in τ with integer coefficients. Let element $\kappa \in \mathbb{Z}[\tau]$ and $P \in E_a(\mathbb{F}_{2^m})$. Then κ can be represented by $u_{l-1}\tau^{l-1}+\cdots+u_1\tau+u_0$ where $u_i \in \mathbb{Z}[\tau]$. A scalar multiplication can be performed by $\kappa P = (u_{l-1}\tau^{l-1} + \cdots + u_1\tau + u_0)P = u_{l-1}\tau^{l-1}(P) + \cdots + u_1\tau(P) + u_0(P)$. Koblitz demonstrated in [18] a method that converts scalar κ to a unique base-τ expansion. The base-τ expansion can be represented by $u_{l-1}\tau^{l-1} + \cdots + u_1\tau + u_0$ where $u_i \in \{0, 1\}$ and $u_{l-1} \neq 0$.

Later, Solinas in [29] showed an improved method that converts a scalar to a unique signed digits representation called TNAF. The TNAF can be represented by $u_{l-1}\tau^{l-1} + \cdots + u_1\tau + u_0$ where $u_i \in \{0, \pm 1\}$ and $u_{l-1} \neq 0$. When a scalar is represented with its reduced TNAF, the average density of nonzero digits becomes less than its base-τ expansion [29]. As a result of Solinas's improvement, the total number of point additions are minimized, and that significantly increases the speed of a scalar multiplication operation.

Solinas showed the window TNAF method can be used with Koblitz curves. It needs to perform an online pre-computation for $2^{w-2} - 1$ points where w is the selected window width. According to [30], working on the main subgroup of $E_a(\mathbb{F}_{2^m})$, the reduced window TNAF method can be briefly described as

1. *Reduction.* Find some suitable $\rho = r_1 + r_2\tau \in \mathbb{Z}[\tau]$ with $|r_1|, |r_2|$ being roughly $\sqrt{n}$, such that

$$\rho \equiv n \pmod{\frac{\tau^m - 1}{\tau - 1}}.$$

   Then the computing nP is equivalent to computing ρP.
2. *Window TNAF.* Fix a positive integer w, choose $\mathcal{C}_{min} = \{c_1, c_3, \cdots, c_{2^{w-1}-1}\}$ with c_j being an element with the least norm from the odd congruence class $\overline{j} = \{c \in \mathbb{Z}[\tau] : c \equiv j \pmod{\tau^w}\}$, $(j = 1, 3, \cdots, 2^{w-1} - 1)$. The width-$w$ τ non-adjacent form of ρ is

$$\rho = \sum_{i=0}^{l-1} \varepsilon_i u_i \tau^i,$$

where $\varepsilon_i \in \{-1, 1\}$ and $u_i \in \mathcal{C}_{min} \cup \{0\}$ with the properties that in any segment $\{u_k, u_{k+1}, \cdots, u_{k+w-1}\}$ of length w , there is at most one nonzero u_i. We denote the above expression of ρ by $\mathrm{TNAF}_w(n)$.

3. *Pre-Computation*: Compute $Q_j = c_j P$ for each $j = 1, 3, \cdots, 2^{w-1} - 1$. Note that $c_1 = 1$, so $Q_1 = P$ needs no calculation.
4. *Computing* nP: Evaluate ρP by Horner's rule, using $\mathrm{TNAF}_w(n)$ and pre-computed $Q_1, Q_3, \cdots, Q_{2^{w-1}}$. Discarding the zero coefficients, the $\mathrm{TNAF}_w(n)$ of ρ can be written as

$$\rho = \underbrace{\varepsilon_0 c_{k_0} \tau^{k_0} + \varepsilon_1 c_{k_1}}_{k_1 - k_0 \geq w} \underbrace{\tau^{k_1} + \varepsilon_2 c_{k_2}}_{k_2 - k_1 \geq w} \tau^{k_2} + \cdots + \varepsilon_{s-1} c_{k_{s-1}} \underbrace{\tau^{k_{s-1}} + \varepsilon_s c_{k_s}}_{k_s - k_{s-1} \geq w} \tau^{k_s}$$

with $\varepsilon_j \in \{-1, 1\}$ and $c_{k_j} \in \mathcal{C}_{min}$. So $nP = \rho P$ can be computed through

$$nP = \tau^{k_0}(\tau^{k_1 - k_0}(\cdots(\tau^{k_s - k_{s-1}} \varepsilon_s Q_{j_{k_s}} + \varepsilon_{s-1} Q_{j_{k_{s-1}}}) + \cdots + \varepsilon_1 Q_{j_{k_{i_1}}}) + \varepsilon_0 Q_{j_{k_0}}).$$

Trost and Xu in [30] established an optimal arrangement setting for the pre-computed points of window TNAF. The optimal pre-computation of window TNAF, as Table 8 shows, costs one point addition and two evaluations of τ at most for each pre-computed point.

4.1 A 3*P* Formula for the Optimal Pre-computation of Window TNAF

Trost and Xu in [30] proposed improvements for the optimal pre-computation of window TNAF by replacing point additions with the efficient formulas in λ-coordinates. The efficient formulas are for the pre-computed points in the forms of $P - \mu\tau(P)$, $P + \mu\tau(P)$, and $P - \tau^2(P)$. Our contribution in this section has two parts. Frist, we propose a 3P efficient formula that can be used together with the already proposed efficient formulas for further speed-up of the optimal pre-computation of window TNAF. Secondly, we conduct experiments to measure the achieved improvement for the optimal pre-computation of window TNAF by using these proposed efficient formulas.

Recall that the proposed efficient formulas for the pre-computed points in the forms of $P - \mu\tau(P)$ and $P + \mu\tau(P)$ are given in [30] by

$$\begin{aligned} A &= X_P(X_P + Z_P)^2 \\ B &= X_P^4 + X_P Z_P + Z_P^4 \\ X_{P-\mu\tau(P)} &= (X_P + Z_P)^4 \\ L_{P-\mu\tau(P)} &= L_P A + X_P^3 Z_P \\ Z_{P-\mu\tau(P)} &= Z_P A \\ X_{P+\mu\tau(P)} &= B^2 \\ L_{P+\mu\tau(P)} &= X_P^7 Z_P + L_P AB \\ Z_{P+\mu\tau(P)} &= Z_P AB. \end{aligned}$$

The pre-computed point in the form of $P - \tau^2(P)$ can be computed by letting $Q = P + \mu\tau(P)$. Then, we have $P - \tau^2(P) = Q - \mu\tau(Q)$. The pre-computed point $3P$ can be computed by Theorem 2 and it can be recognized in the optimal pre-computation of window TNAF by the following proposition.

Proposition 1 *Let $P = (x_P, y_P) \in E_a(\mathbb{F}_{2^m})$ for Koblitz curve E_a. Then*

$$3P = P - \tau^2(P) + \mu\tau(P).$$

Proof. We know $(\tau^2+2)P = \mu\tau(P)$ for all $P \in E_a(\mathbb{F}_{2^m})$. It means $2P = \mu\tau(P) - \tau^2(P)$. It implies $2P + P = \mu\tau(P) - \tau^2(P) + P$. Thus, $3P = P - \tau^2(P) + \mu\tau(P)$.

For examples, consider the optimal pre-computation of window TNAF, as Table 8 shows, with $w = 4$. Then, the pre-compute point Q_3 can be computed by the inverse of $P - \mu\tau(P)$ efficient formulas. To explain, $Q_3 = -(P - \tau^2(P)) = -(Q_7 - \mu\tau(Q_7))$. Consider the optimal pre-computation of window TNAF with $w = 6$. Then, the pre-computed point Q_3 can be computed by the $3P$ efficient formulas. To explain, $Q_3 = Q_{29} + \mu\tau(P) = P - \tau^2(P) + \mu\tau(P) = 3P$.

As mentioned earlier, the improvement of the optimal pre-computation of window TNAF is achieved by replacing point additions with the efficient formulas. Recall that the point addition in λ-projective coordinates costs $11\mathbf{M}+2\mathbf{S}$. The efficient formulas for the pre-computed point in the form of $P - \mu\tau(P)$ costs $5\mathbf{M}+3\mathbf{S}$. The efficient formulas for the pre-computed point in the form of $P + \mu\tau(P)$ costs $7\mathbf{M} + 5\mathbf{S}$. The efficient formulas for the pre-computed point $3P$ costs $8\mathbf{M}+6\mathbf{S}$. Thus, the pre-computed points for the above cost less than point addition in λ-projective coordinates.

4.2 Experiments

The goal for these experiments are to measure the improvement for the optimal pre-computation of window TNAF with the efficient formulas. We replaced the pre-computed points in the forms of $P - \mu\tau(P), P + \mu\tau(P), P - \tau^2(P)$, and $3P$

Table 8. The Optimal Pre-computation of Window TNAF when $a = 0$.

Width	**Pre-computed points**		
4	$Q_3 = -P + \tau^2 P$	$Q_5 = -P - \tau P$	$Q_7 = P - \tau P$
5	$Q_3 = -P + \tau^2 P$	$Q_5 = -P - \tau P$	$Q_7 = P - \tau P$
	$Q_9 = Q_3 - \tau P$	$Q_{11} = Q_5 - \tau P$	$Q_{13} = Q_7 - \tau P$
	$Q_{15} = -Q_{11} + \tau P$		
6	$Q_{29} = P - \tau^2 P$	$Q_3 = Q_{29} - \tau P$	$Q_{31} = Q_3 - \tau^2 P$
	$Q_5 = Q_{31} - \tau P$	$Q_7 = -Q_{31} - \tau P$	$Q_9 = -Q_{29} - \tau P$
	$Q_{27} = P + \tau P$	$Q_{11} = -Q_{27} - \tau P$	$Q_{25} = -P + \tau P$
	$Q_{13} = -Q_{25} - \tau P$	$Q_{15} = -Q_{11} + \tau P$	$Q_{17} = -Q_9 + \tau P$
	$Q_{19} = -Q_7 + \tau P$	$Q_{21} = -Q_{17} - \tau P$	$Q_{23} = -Q_3 + \tau P$

When $a = 1$, Q_j can be obtained by changing only the sign of τ.

of the optimal pre-computation of window TNAF with the efficient formulas. For simplicity, we denote OPT to be the optimal pre-computation of window TNAF without the efficient formulas. We denote OPT+ to be the optimal pre-computation of window TNAF with the efficient formulas. We used two tests to measure the performance of OPT and OPT+. In the first test, we compare OPT and OPT+ in terms of the number of multiplications, as Table 9 shows. In the second test, we did a software implementation, as Table 10 shows, for OPT and OPT+. We used the NIST Koblitz curves K-283, K-409, and K-571 [34]. The environment specifications for these experiments are similar to the experiments in section 3.2.

Table 9. Theoretical Comparison in Terms of the Number of Multiplications

	affine coordinates		λ-projective coordinates		
	OPT: I/M $= 5$	**OPT: I/M** $= 8$	**OPT**	**OPT+**	%
$w = 4$	21	30	33	17	48.48
$w = 5$	49	70	77	58	24.67
$w = 6$	105	150	165	146	11.51

OPT: The optimal pre-computation without the efficient formulas.
OPT+: The optimal pre-computation with the efficient formulas.

Table 9 shows OPT+ speeds up to 48%, 24% and 11% over OPT for window width 4, 5 and 6 respectively. In Table 9, we counted the number of inversions, multiplications of OPT and OPT+ in different window width. We converted an inversion to multiplication based on the ratio **I/M** assumption. We presented two cases for the **I/M** ratio in affine coordinates. The first case is the number of multiplications for OPT when the **I/M** ratio $= 5$. The second case is the number of multiplications for OPT when the **I/M** ratio $= 8$. A squaring operation was ignored in this method since squaring is almost a free operation over binary fields.

Table 10. Running Time Comparison between OPT and OPT+

coordinates		affine	λ-projective		
		OPT	**OPT**	**OPT+**	%
K-283	$w = 4$	0.58 ms	0.517 ms	0.281 ms	45.64
	$w = 5$	1.34 ms	1.113 ms	0. 901 ms	19.04
	$w = 6$	2.92 ms	2.401 ms	2.188 ms	8.87
K-409	$w = 4$	1.05 ms	0.844 ms	0.461 ms	45.37
	$w = 5$	2.54 ms	1.615 ms	1.993 ms	18.96
	$w = 6$	5.34 ms	4.101 ms	3.694 ms	9.92
K-571	$w = 4$	1.94 ms	1.553 ms	0.834 ms	46.29
	$w = 5$	4.55 ms	3.565 ms	2.893 ms	18.84
	$w = 6$	9.74 ms	7.488 ms	6.794 ms	9.26

OPT: The optimal pre-computation without the efficient formulas.
OPT+: The optimal pre-computation with the efficient formulas.

Table 10 shows the running time of OPT+ in λ-projective coordinates performs faster than OPT. However, the percentage of improvement is different for each window width. It shows OPT+ in λ-projective coordinates gives up to a 46% speed-up over OPT if the selected window width is 4. It shows OPT+ gives up to a 19% speed-up over OPT if the selected window width is 5. OPT+ gives up to a 9% speed-up over OPT if the selected window width is 6.

5 Conclusion

In this paper, we present the most efficient $3P$ and $5P$ formulas for binary elliptic curves. We are the first to derive efficient formulas for $3P$ and $5P$ in λ-projective coordinates. We also derived the most efficient $5P$ formula in affine coordinates. Our efficient formulas have an important role in speeding up scalar multiplication operations based on MBNS. We investigated the following MBNS methods: the greedy, the ternary/binary, the multi-base NAF, and the tree-based. We conducted performance comparison tests to these methods using our formulas with respect to the expansion length, the total number of multiplications, and the running time. The total number of multiplications test shows the greedy, the ternary/binary, the multi-base NAF, and the tree-based methods speed up to 10%, 8%, 12%, and 15% over the NAF method. Our running time test shows that the greedy method has a lower percentage of improvement since this test considers the time of converting integer n to a multi-base chain. It implies the greedy method has a higher conversion cost than other MBNS methods.

We proposed a $3P$ efficient formula for the optimal pre-computation of window TNAF for Koblitz curves. Our $3P$ formula can be used with the already proposed efficient formulas to the pre-computed points in the forms of $P-\mu\tau(P)$, $P+\mu\tau(P)$, and $P-\tau^2(P)$. Our experiments show the optimal pre-computation of window TNAF using the efficient formulas speed up to 48%, 24%, and 11% if the used window width is 4, 5, and 6 respectively.

References

1. Avanzi, R.M., Cohen, H., Doche, C., Frey, G., Nguyen, K., Lange, T., Vercauteren, F.: Handbook of Elliptic and Hyperelliptic Curve Cryptography. Chapman & Hall/CRC, Boca Raton (2005)
2. Bernstein, D.J., Birkner, P., Lange, T., Peters, C.: Optimizing double-base elliptic curve single-scalar multiplication. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 167-182. Springer, Heidelberg (2007)
3. Berthe, V., Imbert, L.: On converting numbers to the double-base number system. Adv. Signal Pro. Algo. Archit. Implement. XIV. **5559**, 7078 (2004)
4. Blake, I. F., Murty, V. K., Xu, G.: A note on window τ-NAF algorithm. Info. Processing Letters. **95**(5), 496-502 (2005)
5. Blake, I. F., Murty, V. K., Xu, G.: Nonadjacent radix-τ expansions of integers in euclidean imaginary quadratic number fields. Canadian J. of Math. **60**, 1267-1282 (2008)
6. Blake, I.F., Seroussi, G., Smart, N.P.: Elliptic Curves in Cryptography. Cambridge University Press, Cambridge (1999)
7. Ciet, M., Joye, M., Lauter, K., Montgomery, P.L.: Trading inversions for multiplications in elliptic curve cryptography. Designs Codes Crypto. **39**(2), 189-206 (2006)
8. Dimitrov, V., Imbert, L., Mishra, P.K.: The double-base number system and its application to elliptic curve cryptography. Math. Comput. **77**(262), 1075-1104 (2008)
9. Doche, C.: On the enumeration of double-base chains with applications to elliptic curve cryptography. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 297-316. Springer, Heidelberg (2014)
10. Doche, C., Habsieger, L.: A tree-based approach for computing double-base chains. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 433-446. Springer, Heidelberg (2008)
11. Doche, C., Imbert, L.: Extended double-base number system with applications to elliptic curve cryptography. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 335-348. Springer, Heidelberg (2006)
12. Doche, C., Kohel, D.R., Sica, F.: Double-base number system for multi-scalar multiplications. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 502-517. Springer, Heidelberg (2009)
13. Galbraith, S.D., Lin, X., Scott, M.: Endomorphisms for faster elliptic curve cryptography on a large class of curves. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 518-535. Springer, Heidelberg (2009)
14. Gallant, R.P., Lambert, R.J., Vanstone, S.A.: Faster point multiplication on elliptic curves with efficient endomorphisms. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 190-200. Springer, Heidelberg (2001)
15. Hankerson, D., Karabina, K., Menezes, A.: Analyzing the Galbraith-Lin-Scott point multiplication method for elliptic curves over binary fields. IEEE Trans. Comput. **58**(10), 1411-1420 (2009)
16. Hankerson, D., López Hernandez, J., Menezes, A.: Software implementation of elliptic curve cryptography over binary fields. In: Koc, C.K., Paar, C. (eds.) CHES 2000. LNCS, vol 1965, pp 1-24. Springer, Heidelberg (2000)
17. Hankerson, D., Menezes, A., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer-Verlag, New York (2004)

18. Koblitz, N.: CM-curves with good cryptographic properties. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 279-287. Springer, Heidelberg (1992)
19. Koblitz, N.: Elliptic Curve Cryptosystems. Math. of Computation. **48**(177), 203-209 (1987)
20. Lange, T.: A note on López-Dahab coordinates. Cryptology ePrint Archive, Report 2004/323 (2004). https://eprint.iacr.org
21. Lenstra, A., Verheul, E.: Selecting cryptographic key sizes. J. Crypto. **14**, 255-293 (2001)
22. Longa, P.: Accelerating the scalar multiplication on elliptic curve cryptosystems over prime fields. Cryptology ePrint Archive, Report 2008/100 (2008). https://eprint.iacr.org
23. Longa, P., Gebotys, C.: Fast multibase methods and other several optimizations for elliptic curve scalar multiplication. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 443-462. Springer, Heidelberg (2009)
24. López, J., Dahab, R.: Improved algorithms for elliptic curve arithmetic in GF(2^n). In: Tavares, S., Meijer, H. (eds.) SAC 1998. LNCS, vol 1556, pp 201-212. Springer, Heidelberg (1999)
25. Méloni, N., Hasan M.: Efficient double bases for scalar multiplication. IEEE Trans. Comput. 64, 2204-2212 (2015)
26. Miller, V.S.: Use of elliptic curves in cryptography. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417-426. Springer, Heidelberg (1986)
27. Mishra, P.K., Dimitrov, V.S.: Efficient quintuple formulas for elliptic curves and efficient scalar multiplication using multibase number representation. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 390-406. Springer, Heidelberg (2007)
28. Oliveira, T., López, J., Aranha, D.F., Rodriguez-Henriquez, F.: Two is the fastest prime: lambda coordinates for binary elliptic curves. J. Crypto. Eng. **4**(1), 3-17 (2014)
29. Solinas, J.A.: Efficient arithmetic on Koblitz curves. Designs Codes Crypto. **19**(23), 195-249 (2000)
30. Trost, W., Xu, G.: On the optimal pre-computation of window tNAF for Koblitz curves. IEEE Trans. Comput. **65**, 2918-2924 (2016)
31. Yasin, S., Muda, Z.: Tripling formulae of elliptic curve over binary field in Lopez-Dahab model. J. Theoretical Applied Info. Technology. **75**(2), (2015)
32. Yu, W., Kim, K.H., Jo, M.S.: New fast algorithms for elliptic curve arithmetic in affine coordinates. In: Tanaka, K., Suga, Y. (eds.) IWSEC 2015. LNCS, vol. 9241, pp 56-64. Springer, Cham (2015)
33. Yu, W., Wang, K., Li, B., Tian, S.: Triple-base number system for scalar multiplication. In: Youssef, A., Nitaj, A., Hassanien, A.E. (eds.) AFRICACRYPT 2013. LNCS, vol. 7918, pp. 433-451. Springer, Heidelberg (2013)
34. Digital Signature Standard (DSS). FIPS PUB. 186-4 (2013)
35. The GNU Multiple Precision Arithmetic Library. http://www.gmplib.org

Appendix: Proofs

5.1 Theorem 1

Proof. We shall prove Theorem 1 by the fact

$$(x_{5P}, \lambda_{5P}) = (x_{2P}, \lambda_{2P}) + (x_{3P}, \lambda_{3P}).$$

By using the $P+Q$ λ-affine formula given in [28], we have

$$x_{5P} = \frac{x_{3P}x_{2P}}{(x_{3P}+x_{2P})^2}(\lambda_{3P}+\lambda_{2P}). \tag{3}$$

$$\lambda_{5P} = \frac{x_{3P}(x_{5P}+x_{2P})^2}{x_{5P}x_{2P}} + \lambda_{2P} + 1. \tag{4}$$

We apply $x_{3P} = x_P + \frac{x_P^3}{\alpha} + \left(\frac{x_P^3}{\alpha}\right)^2$ and $x_{2P} = \frac{x_P^4+b}{x_P^2}$ in equation (3). We have

$$x_{5P} = \frac{x_P^3(\alpha^2 + x_P^2(x_P^4+b))\alpha^2(x_P^4+b)}{\left(\alpha^2(x_P^4+b) + x_P^3(\alpha^2+x_P^2(x_P^4+b))\right)^2}(\lambda_{3P}+\lambda_{2P}) \tag{5}$$

$$= \frac{x_P^3\beta\alpha^2(x_P^4+b)}{\gamma^2}(\lambda_{3P}+\lambda_{2P}) \tag{6}$$

We note that

$$\lambda_{3P}+\lambda_{2P} = \frac{x_P\gamma^2}{x_P^3\beta\alpha^2(x_P^4+b)} + 1. \tag{7}$$

By applying equation (7) in equation (6), we have

$$x_{5P} = x_P + \frac{x_P^3\beta\alpha^2(x_P^4+b)}{\gamma^2} \tag{8}$$

$$= x_P + \frac{x_P^3\beta}{\gamma} + \left(\frac{x_P^3\beta}{\gamma}\right)^2. \tag{9}$$

We have derived x_{5P} . Next, we want to derive y_{5P}. From equation (4), we have

$$\lambda_{5P} = \frac{x_{3P}}{x_{2P}}x_{5P} + \frac{x_{3P}x_{2P}}{x_{5P}} + \lambda_{2P} + 1. \tag{10}$$

We apply equation (10) to the fact $y_{5P} = x_{5P}(\lambda_{5P}+x_{5P})$. We have

$$y_{5P} = x_{5P}\left(\frac{x_{3P}}{x_{2P}}x_{5P} + \lambda_{2P} + 1 + x_{5P}\right) + x_{3P}x_{2P}. \tag{11}$$

We apply x_{3P}, x_{2P} , and $\lambda_{2P} = \frac{x_P^4}{x_P^4+b} + \lambda_P^2 + a + 1$ in equation (11). We have

$$\begin{aligned}
y_{5P} &= x_{5P}\Big(\frac{x_P^3\beta}{\alpha^2(x_P^4+b)}x_{5P} + \frac{x_P^4}{x_P^4+b} + \frac{y_P^2}{x_P^2} + x_P^2 + a + x_{5P}\Big) + \frac{(x_P^4+b)\beta}{x_P\alpha^2} \\
&= x_{5P}\Big(\Big(\frac{x_P^3\beta}{\gamma}\Big)^2 + x_P^2 + a + x_{5P}\Big) + \frac{(x_P^4+b)\beta}{x_P\alpha^2} + \frac{x_P^4\beta}{\alpha^2(x_P^4+b)}x_{5P} + \frac{x_P^6 + y_P^2(x_P^4+b)}{x_P^2(x_P^4+b)}x_{5P} \\
&= y_P + x_P + (x_{5P}+x_P)\Big(\Big(\frac{x_P^3\beta}{\gamma}\Big)^2 + x_P^2 + a + x_{5P} + x_P\Big) + \frac{x_P\beta\alpha^2(x_P^6 + y_P^2(x_P^4+b))}{\gamma^2}.
\end{aligned}$$

We note that $x_P^6 = \beta + (x_P^4+b)^2 + x_P^2(x_P^4+b)$ and $\left(\frac{x_P^3\beta}{\gamma}\right)^2 = \frac{x_P^3\beta}{\gamma} + x_{5P} + x_P$.
We have

$$y_{5P} = y_P + x_P + (x_{5P}+x_P)\Big(\frac{x_P^3\beta}{\gamma} + x_P^2 + a\Big) + \frac{x_P\beta\alpha^2(\beta+(x_P^4+b)(x_P^4+b+y_P^2+x_P^2))}{\gamma^2}.$$

5.2 Theorem 2

Proof. We shall prove Theorem 2 by the fact

$$(x_{3P}, \lambda_{3P}) = (x_P, \lambda_P) + (x_{2P}, \lambda_{2P}). \tag{12}$$

By using the $P + Q$ λ-affine formula given in [28], we have

$$x_{3P} = \frac{x_p x_{2P}}{(x_P + x_{2P})^2}(\lambda_P + \lambda_{2P}). \tag{13}$$

$$\lambda_{3P} = \frac{x_{2P}(x_{3P} + x_P)^2}{x_{3P}x_P} + \lambda_P + 1. \tag{14}$$

We apply the relation $\lambda_P + \lambda_{2P} = \frac{(x_P + x_{2P})^2}{x_{2P}} + 1$ in equation (13) . We have

$$x_{3P} = x_P + \frac{x_p x_{2P}}{(x_P + x_{2P})^2} \tag{15}$$

$$= \frac{x_P\big(x_{2P} + (x_{2P} + x_P)^2\big)}{(x_P + x_{2P})^2}. \tag{16}$$

We convert λ-affine point (x_P, λ_P) to λ-projective point (X_P, L_P, Z_P) by using the relation $(x_P, \lambda_P) = (\frac{X_P}{Z_P}, \frac{L_P}{Z_P})$. Thus, the equations above become

$$\begin{aligned}
x_{2P} &= \frac{L_P^2 + L_P Z_P + aZ^2}{Z_P^2} = \frac{T}{Z_P^2}.\\
x_{3P} &= \frac{\frac{X_P}{Z_P}\big(\frac{T}{Z_P^2} + \frac{(T + X_P Z_P)^2}{Z_P^4}\big)}{\frac{(T + X_P Z_P)^2}{Z_P^4}}\\
&= \frac{X_P\big(TZ_P^2 + (T + X_P Z_P)^2\big)}{Z_P(T + X_P Z_P)^2}\\
&= \frac{X_P B}{Z_P A}.\\
\lambda_{3P} &= \frac{\frac{T}{Z_P^2}\big(\frac{X_P B}{Z_P A} + \frac{X_P}{Z_P}\big)^2}{\frac{X_P^2 B}{Z_P^2 A}} + \frac{L_P Z_P + Z_P^2}{Z_P^2}\\
&= \frac{T(A + B)^2}{Z_P^2 AB} + \frac{L_P Z_P + Z_P^2}{Z_P^2}\\
&= \frac{T(A + B)^2 + (L_P Z_P + Z_P^2)AB}{Z_P^2 AB}.
\end{aligned}$$

5.3 Theorem 3

Proof. We shall proof x_{5P} by the fact

$$(x_{5P}, \lambda_{5P}) = (x_{2P}, \lambda_{2P}) + (x_{3P}, \lambda_{3P}).$$

By using the $P+Q$ λ-affine formula given in [28], we have

$$x_{5P} = \frac{x_{2P}x_{3P}}{(x_{2P}+x_{3P})^2}(\lambda_{2P}+\lambda_{3P}). \tag{17}$$

We apply the relation $\lambda_{2P}+\lambda_{3P} = \frac{x_P(x_{2P}+x_{3P})^2}{x_{2P}x_{3P}} + 1$ to equation (17) . We have

$$x_{5P} = x_P + \frac{x_{2P}x_{3P}}{(x_{2P}+x_{3P})^2} \tag{18}$$

$$= \frac{x_P(x_{2P}+x_{3P})^2 + x_{2P}x_{3P}}{(x_{2P}+x_{3P})^2}. \tag{19}$$

Next, we shall derive λ_{5P} by the fact

$$(x_{5P},\lambda_{5P}) = (x_P,\lambda_P) + (x_{4P},\lambda_{4P}).$$

By using the $P+Q$ λ-affine formula , we have

$$\lambda_{5P} = \frac{x_{4P}(x_{5P}+x_P)^2}{x_{5P}x_P} + \lambda_P + 1. \tag{20}$$

We convert λ-affine point (x_P,λ_P) to λ-projective point (X_P,L_P,Z_P) by using the relation $(x_P,\lambda_P) = (\frac{X_P}{Z_P},\frac{L_P}{Z_P})$. Thus, the equations above become

$$\begin{aligned}
x_{2P} &= \frac{L_P^2 + L_PZ_P + aZ^2}{Z_P^2} = \frac{T}{Z_P^2}.\\
x_{3P} &= \frac{X_P\big(TZ_P^2 + (T+X_PZ_P)^2\big)}{Z_P(T+X_PZ_P)^2} = \frac{X_PB}{Z_PA}.\\
x_{4P} &= \frac{L_{2P}^2 + L_{2P}TZ_P^2 + a(TZ^2)^2}{(TZ_P^2)^2} = \frac{T_2}{(TZ_P^2)^2}.\\
x_{5P} &= \frac{\frac{X_P}{Z_P}\big(\frac{T}{Z_P^2}+\frac{X_PB}{Z_PA}\big)^2 + \frac{TX_PB}{Z_P^3A}}{(\frac{T}{Z_P^2}+\frac{X_PB}{Z_PA})^2}\\
&= \frac{X_P\big((TA+X_PZ_PB)^2 + TZ_P^2AB\big)}{Z_P(TA+X_PZ_PB)^2}\\
&= \frac{X_PD}{Z_PC}.\\
\lambda_{5P} &= \frac{\frac{T_2}{(TZ_P^2)^2}\big(\frac{X_PD}{Z_PC}+\frac{X_P}{Z_P}\big)^2}{\frac{X_P^2D}{Z_P^2C}} + \frac{L_PZ_P+Z_P^2}{Z_P^2}\\
&= \frac{T_2(C+D)^2}{(TZ_P^2)^2CD} + \frac{L_PZ_P+Z_P^2}{Z_P^2}\\
&= \frac{Z^2T_2(AB)^2 + (L_PZ_P+Z_P^2)CD}{Z_P^2CD}.
\end{aligned}$$

We note the following relations

$$\begin{aligned} Z_P^2 T_2 &= T(A+B)^2 + Z_P^2 AB \\ C &= (TA + X_P Z_P B)^2 = (T(A+B))^2 + AB^2 \\ D &= TZ_P^2 AB + C = A^2 B + AB^2 + C \end{aligned}$$

Thus , we have

$$L_{5P} = T(C+D)^2 + (L_P Z_P + Z_P^2)CD + Z_P^2 (AB)^3.$$