Fast Multibase Methods and Other Several Optimizations for Elliptic Curve Scalar Multiplication*

Patrick Longa and Catherine Gebotys

Department of Electrical and Computer Engineering,
University of Waterloo, Canada,
{plonga,cgebotys}@uwaterloo.ca

Abstract. Recently, the new Multibase Non-Adjacent Form (mbNAF) method was introduced and shown to speed up the execution of the scalar multiplication with an efficient use of multiple bases to represent the scalar. In this work, we first optimize the previous method using fractional windows, and then introduce further improvements to achieve additional cost reductions. Moreover, we present new improvements in the point operation formulae. Specifically, we reduce further the cost of composite operations such as quintupling and septupling of a point, which are relevant for the speed up of multibase methods in general. Remarkably, our tests show that, in the case of standard elliptic curves, the refined mbNAF method can be as efficient as Window-w NAF using an optimal fractional window size. Thus, this is the first published method that does not require precomputations to achieve comparable efficiency to the standard window-based NAF method using precomputations. On other highly efficient curves as Jacobi quartics and Edwards curves, our tests show that the refined mbNAF currently attains the highest performance for both scenarios using precomputations and those without precomputations.

Keywords: Elliptic curve cryptosystem, scalar multiplication, multibase non-adjacent form, double base number system, fractional window.

1 Introduction

Scalar multiplication, denoted by kP (where k is a scalar and P a point on the elliptic curve), is the most time consuming operation in Elliptic Curve Cryptosystems (ECC). Although several algorithms to compute kP using efficient representations of k have been proposed and extensively studied in past years, it is still a challenge to improve the performance of this operation for further deployment in embedded systems.

* A version of this paper appears in the 12th International Conference on Practice and Theory in Public Key Cryptography (PKC2009), LNCS 5443, pp. 443-462, S. Jarecki and G. Tsudik (eds.). Springer, Heidelberg, 2009.

In that effort, a strategy that has gained lots of attention in recent years is the use of representations based on double- and multi-base chains. The use of the so-called Double Base Number System (DBNS) for cryptographic applications was first proposed by Dimitrov et al. in [7]. In the setting of ECC, double-base chains were first applied to the computation of scalar multiplication by Dimitrov et al. [8], and later extended to multibase chains by [17] and [9].

Although it was empirically shown that double-base chains reduce the cost of the scalar multiplication, the main drawbacks of the initial approaches using this strategy were their memory penalty and difficulty to analyze performance theoretically [8, 11, 1]. To solve these problems, an improved multibase representation was introduced by Longa in [17]. One of the key features of this representation is the use of the non-adjacency property (as found in NAF), which makes the conversion process (from binary to multibase) simple, efficient and with no memory impact. This new representation is called Multibase NAF (mbNAF). Its window-based version using an extended set of precomputations appears as a natural extension and is referred to as Window-w Multibase NAF (wmbNAF)[1].

Nevertheless, although Multibase NAF is simple and offers high performance, it is still possible to find shorter and more efficient multibase chains. In that direction, this work proposes new algorithms that are able to find highly efficient multibase chains and thus reduce scalar multiplication costs even further.

In addition, we also propose other several optimizations that aim at improving the efficiency of multibase methods (and standard methods in some cases). The contributions of this paper can be summarized as follows[2]:

- New reductions in the cost of composite (point) operations. We present improved formulas for quintupling and septupling in Jacobian coordinates.
- Improved (w)mbNAF-based algorithms (hereinafter referred to as Refined mbNAF methods) that "smartly" trade doublings for triplings/quintuplings and find shorter chains, reducing further the cost of the scalar multiplication.
- Window-based methods, namely wmbNAF and its refined version, are optimized by using fractional windows.
- The theoretical analysis demonstrating mathematically the performance of the Multibase NAF methods (and variants) is presented.
- Finally, we carry out a more comprehensive comparison taking into account most efficient curve shapes and point formulas found in the literature, and recent and most efficient methods to compute the evaluation and precomputation stages in the scalar multiplication.

Note that we focus here on methods that are efficient if the point P used to compute kP is not known in advance. In such a context, we analyze the performance of the proposed methods and compare to that of traditional binary methods such as NAF and wNAF, and another approaches using double-base

[1] In some way, Multibase NAF can be seen as a generalization of the ternary/binary algorithm by Ciet et al. [6]. The original intention of the author [17], however, was to insert the concept of double-base chains proposed by [8] into the NAF algorithm.

[2] This paper presents formally and expands the results of the technical report [19].

chains (methods using a "Greedy" algorithm will be generically referred to as DB [8, 11, 1]). Our analysis includes the standard form of an elliptic curve over prime fields, and other very efficient curve forms such as Jacobi quartics [5] and Edwards curves [12]. It will turn out that the Refined mbNAF is currently the most cost efficient method for both scenarios with and without precomputations and for all the studied curve forms.

Our work is organized as follows. In Section 2, we detail some background about ECC over prime fields, summarizing the state-of-the-art point formulae for Jacobian coord., Jacobi quartics and Edwards curves. Improvements to these operations are discussed in this section. In the following section, we briefly describe the original $(w)mb$NAF methods and discuss their theoretical performance. In Section 4, we optimize the performance of multibase algorithms by using fractional windows and present a detailed theoretical analysis. We then describe new improvements to Multibase NAF in Section 5 and propose the Refined mbNAF method, highlighting its advantages and high performance for computing scalar multiplication. In Section 6, the performance of various methods for scalar multiplication is evaluated through extensive tests. Finally, in Section 7 we present some conclusions summarizing the contributions of this work.

2 Elliptic Curve Cryptography

A brief introduction to ECC is presented in this section. The reader is referred to [14] for extended details. An elliptic curve E over a prime field $\mathbb{F}_p$ (denoted by $E(\mathbb{F}_p)$) can be defined by the simplified Weierstrass equation E: $y^2 = x^3 + ax + b$ (referred to as the standard EC form in the remainder), where $a, b \in \mathbb{F}_p$. The points on the curve E and the point at infinity, denoted by $\mathcal{O}$, form an additive group on top of which the cryptosystem works. Two basic operations exist to perform point computations: doubling ($2P$) and addition ($P + Q$) of points.

The representation of points on the curve E using (x, y), known as affine coordinates, introduces expensive field inversions (I) in the computation of point operations. Hence, most efficient implementations use representations of the form $(X : Y : Z)$, known as projective coordinates. For example, an efficient case of the latter is given by Jacobian coordinates, where each projective point $(X_i : Y_i : Z_i)$ corresponds to the affine point $(X_i/Z_i^2, Y_i/Z_i^3)$. The reader is referred to [18] for complete details about most efficient formulae in this system.

Recently, other curve forms with faster group laws have appeared in the literature. We focus here on two of them: Jacobi quartics and Edwards curves, whose explicit formulas are highly efficient. We briefly described them in the following.

Jacobi quartic curve. It is defined by the projective curve $Y^2 = X^4 + 2aX^2Z^2 + Z^4$, where $a \in \mathbb{F}_p$ and $a^2 \neq 1$. A given projective point $(X_i : Y_i : Z_i)$ corresponds to the affine point $(X_i/Z_i, Y_i/Z_i^2)$. The most efficient formulae in these curves have been developed by Hisil et al. [15, 16] using an extended coordinate system of the form $(X_i : Y_i : Z_i : X_i^2 : Z_i^2)$.

Edwards curve. The projective curve in this setting is given by $(X^2 + Y^2)\, Z^2 =$

$Z^4 + dX^2Y^2$. However, the most efficient explicit formulas in this case correspond to a new coordinate system known as inverted Edwards coord. [4], for which the curve equation takes the form $(X^2 + Y^2)\,Z^2 = X^2Y^2 + dZ^4$ and each projective point $(X_i : Y_i : Z_i)$ corresponds to the affine point $(Z_i/X_i, Z_i/Y_i)$.

Note that the basic doubling and addition operations are sufficient to implement traditional methods relying on (signed) binary representations as NAF. However, new double- and multi-base methods require specialized operations such as tripling and/or quintupling of a point for their efficient realization.

In this work, we present optimized formulas for quintupling and septupling using Jacobian coord. (refer to Appendix A for complete details). Furthermore, certain computations such as those at the beginning of the evaluation stage can benefit from having specialized formulas with *mixed* coordinates that accept the input in affine and output the result in some projective system. Because of page constrains, formulas using mixed coordinates on standard and Edwards curves have not been included (the interested reader is referred to [19]). For mixed Jacobi quartic-affine coord., formulas can be easily derived from doubling, tripling, quintupling and septupling formulas due to [15, 16] by setting $Z_1 = 1$.

In Table 1, we summarize the costs of the state-of-the-art point formulae, including the ones described above, for our three curves of interest: standard elliptic curves using Jacobian coordinates (*Jacobian*, parameter $a = -3$ in equation E), Jacobi quartics using the extended coordinate system (*JQuartic*) and Edwards curves using inverted Edwards coord. (*InvEdw*). For the remainder, doubling ($2P$), tripling ($3P$), quintupling ($5P$), septupling ($7P$), addition ($P + Q$) and doubling-addition ($2P + Q$) are denoted by D, T, Q, S, A and DA, respectively. Operations using *mixed* coordinates are denoted by mD, mT, mQ, mS, mA and mDA, corresponding to each of the aforementioned point operations. For addition, the case in which both inputs are in affine is denoted by mmA. Costs are expressed in terms of field multiplications (M) and squarings (S), disregarding field addition/subtractions (A) and multiplication/divisions by small constants for simplification purposes. We also assume that $1S = 0.8M$.

In some cases, it is possible to reduce the cost of certain operations if some values are precalculated in advance. That is the case of addition and doubling-addition (DA) with stored values in Jacobian coordinates (see Table 1). If, for instance, values Z_i^2 and Z_i^3 are precalculated for each precomputed point in windowed methods the costs of these point operations can be reduced by $1M + 1S$.

Table 1. Cost of elliptic curve point operations

Curve	D/mD	T/mT	Q/mQ	S/mS	A	mA/mmA	DA/mDA
Jacobian	$3M+5S$/ $1M+5S$	$7M+7S$/ $5M+7S$ (1)	$10M+12S$ (1)/ $8M+12S$ (1)	$14M+15S$ (1)/ $12M+15S$ (1)	$10M+4S$ (2) $11M+5S$	$7M+4S$/ $4M+2S$	$13M+8S$ (2) $14M+9S$/ $11M+7S$
InvEdw	$3M+4S$/ $3M+3S$	$9M+4S$/ $7M+3S$ (1)	-	-	$9M+1S$	$8M+1S$/ $7M$	-
JQuartic	$2M+5S$/ $6S$ (1)	$8M+4S$/ $5M+5S$ (1)	$14M+4S$/ $11M+5S$ (1)	$16M+8S$/ $13M+9S$ (1)	$7M+3S$ (2) $7M+4S$	$6M+3S$/ $4M+3S$	-

(1) Introduced in this work (see Appendix A and [19]); (2) cost of operation with stored values.

We use the efficient operations discussed in this section for our comparisons and cost analyses of scalar multiplication methods in Section 6.

3 Multibase Non-Adjacent Form Methods

Following, we briefly describe the original Multibase NAF methods introduced in Longa [17]. Our main contribution in this section is to provide the theoretical analysis of the average density of these methods when using bases {2,3} and {2,3,5} that was deferred in [17].

3.1 Multibase NAF (*mb*NAF) and Window-*w* Multibase NAF (*wmb*NAF)

Determining and finding the "optimal" multibase chain in the setting of ECC seems to be a hard problem, mainly due to the fact that an "optimal" multibase chain is not necessarily the shortest, but the one that requires the "right" balance in the number of additions and all the other point operations. Although finding such an "optimal" multibase chain remains an open problem, Longa [17] proposed a representation that allows a better control of the appearance of point operations in the scalar expansion, and consequently, gets closer to the optimal. Such a generic multibase representation, known as *mb*NAF, has the form:

$$k = \sum_{i=1}^{m} s_i \prod_{j=1}^{J} a_j^{c_i(j)} \tag{1}$$

where $a_1 \neq \ldots \neq a_J$ are prime integers from a set of bases $\mathcal{A} = \{a_1, \ldots, a_J\}$ (a_1: main base),
m is the length of the expansion,
s_i are signed digits from a given set $D\backslash\{0\}$, i.e., $|s_i| \geq 1$ and $s_i \in D\backslash\{0\}$,
$c_i(j)$ are decreasing exponents, s.t. $c_1(j) \geq c_2(j) \geq \ldots \geq c_m(j) \geq 0$ for each j from 2 to J, and
$c_i(1)$ are decreasing exponents for the main base a_1 (i.e., $j = 1$), s.t. $c_i(1) \geq c_{i+1}(1) + 2 \geq 2$ for $1 \geq i \geq m - 1$.

The last two conditions above guarantee that an expansion of the form (1) is efficiently executed by a scalar multiplication using Horner's method as follows:

$$kP = \prod_{j=1}^{J} a_j^{d_m(j)} \left(\prod_{j=1}^{J} a_j^{d_{m-1}(j)} \left(\ldots \left(\prod_{j=1}^{J} a_j^{d_1(j)} (s_1 P) + s_2 P \right) + \ldots + s_{m-1} P \right) + s_m P \right)$$

where $d_m(1) \geq 0$, and $d_i(1) \geq 2$ for $1 \geq i \geq m - 1$. The latter is equivalent to the last condition in (1) and incorporates the non-adjacency property in the multibase representation. Basically, it fixes the minimal number of consecutive operations with the main base (i.e., a_1) between any two additions to 2. Note that an operation with the main base refers to a doubling if $a_1 = 2$ (this will be the case for most scenarios where doubling is the most efficient point operation).

If we relax the previous condition and allow larger window sizes (i.e., allowing 3, 4, or more, consecutive operations with the main base between any two additions) we can reduce further the average number of nonzero terms in the scalar representation at the expense of a larger digit set D and, consequently, a larger precomputed table. The previous technique is known as wmbNAF.

The mbNAF and wmbNAF representations require the following digit set [17]

$$D = \left\{0, \pm 1, \pm 2, \ldots, \pm \left\lfloor \frac{a_1^w - 1}{2} \right\rfloor \right\} \setminus \left\{ \pm 1a_1, \pm 2a_1, \ldots, \pm \left\lfloor \frac{a_1^{w-1} - 1}{2} \right\rfloor a_1 \right\} \quad (2)$$

where $w \geq 2 \in \mathbb{Z}^+$ ($w = 2$ for mbNAF). Without considering $\{\mathcal{O}, P\}$, the digit set (2) involves precomputing d_iP, where $d_i \in D^+ \setminus \{0, 1\}$ (note that only positive values d_iP need to be stored in the table as the inverse of points can be computed on the fly). Thus, the precomputed table consists of $(a_1^w - a_1^{w-1} - 2)/2$ points. Note that if $w = 2$ (mbNAF case), the requirement of precomputations is minimal. For instance, in the case $a_1 = 2$ we need to store nil points besides $\{\mathcal{O}, P\}$.

It is important to remark that, obviously, (1) does not involve unique representations. In [17], Longa provided algorithms (see Alg. 3.1) that efficiently find a multibase chain of the form (1) and, given a window width and set of bases, is unique for each integer. Note that Algorithm 3.1 integrates mbNAF and wmbNAF.

Algorithm 3.1. Computing the mbNAF (wmbNAF) of a positive integer

---

INPUT: scalar k, bases $\mathcal{A} = \{a_1, \ldots, a_J\}$, where $a_j \in \mathbb{Z}^+$ are primes for $1 \geq j \geq J$, window $w = 2$ for mbNAF, and window $w > 2$ for wmbNAF, where $w \in \mathbb{Z}^+$

OUTPUT: the $(a_1, a_2, \ldots, a_J)\text{NAF}_w(k) = (\ldots, k_2^{(a_j)}, k_1^{(a_j)})$

---

1. $i = 1$
2. While $k > 0$ do
   - 2.1. If $k \bmod a_1 = 0$ or $k \bmod a_2 = 0$ or $\ldots$ or $k \bmod a_J = 0$, then $k_i = 0$
   - 2.2. Else:
     - 2.2.1. $k_i = k \bmod s\ a_1^w$
     - 2.2.2. $k = k - k_i$
   - 2.3. If $k \bmod a_1 = 0$, then $k = k/a_1, k_i = k_i^{(a_1)}$
   - 2.4. Elseif $k \bmod a_2 = 0$, then $k = k/a_2, k_i = k_i^{(a_2)}$
   - $\vdots$
   - 2.(J+2). Elseif $k \bmod a_J = 0$, then $k = k/a_J, k_i = k_i^{(a_J)}$
   - 2.(J+3). $i = i + 1$
3. Return $(\ldots, k_2^{(a_j)}, k_1^{(a_j)})$

$k_i^{(a_j)}$ in Algorithm 3.1 represents the digits in the multibase NAF representation, where $k_i \in D$ (see (2)) and the superscript (a_j) represents the base $a_j \in \mathcal{A}$ associated to the digit in position i. The function $mods$ represents the following

$$\begin{cases} \text{If } k \bmod a_1^w \geq a_1^w/2, \text{then } k_i = (k \bmod a_1^w) - a_1^w \\ \text{Else, } k_i = k \bmod a_1^w \end{cases}$$

Let us illustrate the method using Alg. 3.1 with the following example.

Example 1. The *mb*NAF representation of 9750 according to Algorithm 3.1 is $(2,3)\text{NAF}_2(9750) = 1^{(2)}\,0^{(2)}\,0^{(2)}\,0^{(2)}\,1^{(2)}\,0^{(3)}\,0^{(2)}\,\text{-}1^{(2)}\,0^{(2)}\,0^{(2)}\,1^{(2)}\,0^{(3)}\,0^{(2)}$, which would allow to compute $9750P$ as $2 \times 3\ (2^3\,(2^2 \times 3(2^4P + P) - P) + P)$. The latter involves 1mD+9D+2T+3mA. For instance, using Table 1 (JQuartic, $1S = 0.8M$), $9750P$ would cost $107.2M$. Compare this to the cost using NAF, i.e., 1mD+12D+5mA = $119.6M$.

Zero and Nonzero Density of Multibase NAF Methods. One of the attractive properties of Multibase NAF methods is that the average number of operations can be precisely determined by using Markov chains. The following theorems are presented on this regard (please, refer to Appendix B for the proofs).

Theorem 1. *The average densities of additions, doublings and triplings for the (w)mbNAF using bases $\mathcal{A} = \{2,3\}$ are approximately*

$$\delta_x = \frac{2^w}{3(2^{w-2}-s)+2^w(w+1)}\,, \delta_{0^2} = \frac{2^w(w+1)}{3(2^{w-2}-s)+2^w(w+1)} \text{ and } \delta_{0^3} = \frac{3(2^{w-2}-s)}{3(2^{w-2}-s)+2^w(w+1)},$$

respectively, where $s = \lfloor (2^{w-2}+1)/3 \rfloor$ and $w \geq 2 \in \mathbb{Z}^+$ ($w = 2$ for mbNAF).

Theorem 2. *The average densities of additions, doublings, triplings and quintuplings for the (w)mbNAF using bases $\mathcal{A} = \{2,3,5\}$ are approximately*

$$\delta_x = \frac{2^{w+3}}{17\cdot 2^{w-1}-5r-24s-5t+2^{w+3}(w+1)}\,, \delta_{0^2} = \frac{2^{w+3}(w+1)}{17\cdot 2^{w-1}-5r-24s-5t+2^{w+3}(w+1)}\,,$$
$$\delta_{0^3} = \frac{24(2^{w-2}-s)}{17\cdot 2^{w-1}-5r-24s-5t+2^{w+3}(w+1)} \text{ and } \delta_{0^5} = \frac{5(2^{w-1}-r-t)}{17\cdot 2^{w-1}-5r-24s-5t+2^{w+3}(w+1)},$$

respect., where $r = \lfloor (2^{w-2}+2)/5 \rfloor$, $s = \lfloor (2^{w-2}+1)/3 \rfloor$ and $t = \lfloor (2^{w-2}+7)/15 \rfloor$.

Let us determine the average number of operations when using the Multibase NAF method. First, it is known that the expected number of doublings, triplings and additions is given by $\#\text{D} = \delta_{0^2} \cdot digits$, $\#\text{T} = \delta_{0^3} \cdot digits$ and $\#\text{A} = \delta_x \cdot digits$, where *digits* represents the total number of digits in the expansion (note that a nonzero digit involves one doubling and one addition). Also, we can assume that $2^{\#D} \cdot 3^{\#T} \approx 2^{n-1}$, where n represents the average bitlength of the scalar k. Thus, $\#\text{D} \cdot \log 2 + \#\text{T} \cdot \log 3 \approx (n-1)\log 2$, and replacing #D and #T, we can estimate *digits* with the following

$$digits \approx \frac{(n-1)\log 2}{\delta_{0^2} \cdot \log 2 + \delta_{0^3} \cdot \log 3} \tag{3}$$

which allow us to determine #D, #T and #A using the expressions above. A similar procedure easily follows for the case of bases {2,3,5}.

For instance, in the case of mbNAF, bases $\mathcal{A} = \{2,3\}$, $w = 2$ and $n = 160$ bits, the average densities for doublings, triplings and additions derived from Theorem 1 are 4/5, 1/5 and 4/15. Using (3), we determine that $digits = 142.35$. Then, the average cost of a scalar multiplication using Table 1 (JQuartic,$1S = 0.8M$) is approx. 113.88D+28.47T+37.96mA = $1321M$. Similarly, if we use bases $\mathcal{A} = \{2,3,5\}$, the average cost can be estimated as approximately 97.06D+24.27T+ 10.11Q+32.35mA = $1299.82M$. Compare the previous costs to that offered by NAF: 159D+53mA = $1399.2M$ (in this case, $\delta_{NAF} = 1/3$). Hence, theoretically, it is determined that (2,3)NAF and (2,3,5)NAF surpasses NAF (case with no precomputations) by about 5.6% and 7.1%, respectively.

Despite these results, it is still possible to find more efficient multibase chains at the expense of some increment in the complexity of the basic Multibase NAF. The improved multibase algorithms will be discussed in Section 5. Following, we optimize the basic multibase methods using a recoding based on fractional windows.

4 The Fractional Window-w Multibase Non-Adjacent Form (Frac-wmbNAF)

In this section, we apply the concept of "fractional" windows [24] to the multibase NAF method to allow a flexible number of points in the precomputed table. The new representation is called Fractional wmbNAF (denoted by Frac-wmbNAF).

For the remainder, we will assume that the main base a_1 is 2 as this value is expected to achieve the lowest costs with most efficient ECC curve forms. First, let us establish our ideal table with unrestricted number of points d_iP, where $d_i \in D^+\backslash\{0,1\} = \{3,5,\ldots,m\}$, and $m \geq 3 \in \mathbb{Z}^+$ is an odd integer. If we define m in terms of the standard windows w, it would be expressed as

$$m = 2^{w-2} + s, \tag{4}$$

where $2^{w-2} < m < 2^{w-1}$ and $s \geq 1 \in \mathbb{Z}^+$ is odd.

We can now define the rules of our recoding scheme for bases $\mathcal{A} = \{a_1, a_2, \ldots, a_J\}$ in the following:

1. If ($k \bmod 2 = 0$ or $k \bmod a_2 = 0 \ldots$ or $k \bmod a_J = 0$), then $k_i = 0$
2. Elseif $0 < r \leq m$, then $k_i = r$
3. Elseif $m < r < (3m - 4s)$, then $k_i = r - 2^{w-1}$
4. Elseif $(3m - 4s) \leq r < 2^w$, then $k_i = r - 2^w$
5. $k = k - k_i$

where $r = k \bmod 2^w$. Basically, the proposed recoding first detects if k is divisible by one of the bases. Else, it establishes a window w and checks if k can be approximated to the closest extreme of the window using any of the digits d_i available. It can be verified that the latter will be accomplished if steps 2 or 4 are satisfied. Otherwise, the established window is too large and, hence, it is "reduced" to the immediately preceding window size to which k can be approximated (condition in step 3).

An algorithm to convert any integer to Frac-*wmb*NAF representation can be easily derived by replacing steps 1-5 above by steps 2.1 and 2.2 in Algorithm 3.1. In this case, we will denote the Frac-*wmb*NAF of an integer k by $(2, a_2, \ldots, a_J)\text{NAF}_{w,t}(k) = (\ldots, k_2^{(a_j)}, k_1^{(a_j)})$, where t represents the number of precomputed points, i.e., $m = 2t + 1$.

Let us illustrate the new recoding with the following example.

Example 2. If $k = 9750$ and $m = 5$, then $d_i \in D^+ \backslash \{0, 1\} = \{3, 5\}$, and $w = 4$ and $s = 1$ by means of (4). Then, the Frac-*wmb*NAF of 2950 is given by $(2,3)\text{NAF}_{4,2}(9750) = 1^{(2)}0^{(2)}0^{(2)}0^{(2)}\text{-}3^{(2)}0^{(2)}0^{(2)}0^{(2)}\text{-}5^{(2)}0^{(2)}0^{(2)}1^{(2)}0^{(3)}0^{(2)}$, and the conversion process can be visualized as $\frac{9750}{2} \rightarrow \frac{4875}{3} \rightarrow 1625 - 1 \rightarrow \frac{1624}{8} \rightarrow 203 + 5 \rightarrow \frac{208}{16} \rightarrow 13 + 3 \rightarrow \frac{16}{16} \rightarrow 1$.

Observe that, when 1625 is obtained, it requires an addition with 7 to reach 1632 (which is the closest number $\equiv (0 \bmod 2^4)$, as required by a standard window $w = 4$). However, 7 is not part of our precomputed table, so the window size is reduced accordingly to $w = 3$ and the value 1625 is approximated to the closest value in the new window (i.e., 1624) using an addition with -1. We now present the following theorem regarding the average density of this method for the case $\mathcal{A} = \{2, 3\}$.

Theorem 3. *The average densities of nonzero terms, doublings and triplings of the Frac-wmbNAF using bases $\mathcal{A} = \{2, 3\}$, window size w and t available points (represented by (2,3)NAF$_{w,t}$) are approximately*

$$\frac{2^w}{8(t+1)-3(u+v)+2^{w-2}(4w-1)},\ \frac{8(t+1)+2^w(w-1)}{8(t+1)-3(u+v)+2^{w-2}(4w-1)},\ \frac{3(2^{w-2}-(u+v))}{8(t+1)-3(u+v)+2^{w-2}(4w-1)}$$

respectively, where $u = \lfloor (t+2)/3 \rfloor$ and $v = \lfloor (2^{w-2} - t)/3 \rfloor$.

The reader is referred to Appendix C for a proof. With Theorem 3, it is possible to theoretically estimate the expected number of doublings, triplings and additions using this method. For instance, following the procedure detailed in Section 3.1, we can determine the cost of a scalar multiplication (without including precomputation) for $n = 160$ bits using $t = 2$ points ($w = 4$) as 132.7D+16.6T+29.5mA = $1229.9M$ (JQuartic). Compare to the cost achieved by Frac-wNAF, namely 159D+35.3mA = $1250.5M$ ($\delta_{Frac-wNAF} = 1/4.5$ when using $m = 5$; see [25]). Further cost reductions are observed for the case of $\mathcal{A} = \{2, 3, 5\}$.

5 The Refined Multibase Non-Adjacent Form (Refined *mb*NAF)

In this section we present a new methodology to derive algorithms able to find more efficient multibase chains. Similarly to the original Multibase NAF methods, we base our approach on the key observation that point operations such as doublings and triplings have different costs and that any multibase algorithm

with application to scalar multiplication should not only try to reduce the length of the expansion but also (and more importantly) find the right balance between the number of all the point operations involved.

Following the previous criteria, we modify the original Multibase NAF using the next conditional statements (again, we restrict our analysis to the most efficient case $a_1 = 2$. Also, q, r are odd integers and $\{2, a_2, \ldots, a_J\} \nmid q,\ r$):

1. *CONDITION1*: before every nonzero term, approximate the current partial value k to the closest value in the established window with $k - k_i = 2^w \cdot a_2^{w_2} \cdot \ldots \cdot a_J^{w_J} \cdot q$, where $k_i \in D \backslash \{0\} = \{\pm 1, \pm 3, \pm 5, \ldots, \pm m\}$, as usually performed, if and only if there does not exist some value $k - d_i = 2^{w_1'} \cdot a_2^{w_2'} \cdot \ \ldots \ \cdot a_J^{w_J'} \cdot r$ in the established window, with $w_j, w_j' \geq 0$ for each base from the set $\mathcal{A} = \{a_1, a_2, \ldots, a_J\}$ and $d_i \in D \backslash \{0\} \neq k_i$, such that the zero digit sequence to follow is "greater" than that guaranteed by the window w (i.e., for practical purposes, $2^{w_1'} \cdot \ldots \cdot a_J^{w_J'} > 2^w \cdot \ldots \cdot a_J^{w_J} + e$), in which case (*CONDITION1 = true*) the approximation $k - d_i$ is applied instead of $k - k_i$.
2. *CONDITION2*: before each zero term different than $0^{(2)}$, we test if there is a nonzero digit $d_i \in D \backslash \{0\}$ which would allow an approximation $k - d_i = 2^{w'} \cdot a_2^{w_2'} \cdot \ \ldots \ \cdot a_J^{w_J'} \cdot r$ such that $2^{w'} \cdot \ldots \cdot a_J^{w_J'} > a_2^{w_2} \cdot \ldots \cdot a_J^{w_J} + e'$, where $k = a_2^{w_2} \cdot \ \ldots \ \cdot a_J^{w_J} \cdot q$ is a partial scalar value and $w_j, w_j' \geq 0$ for each base from the set $\mathcal{A}$. If the latter happens (*CONDITION2 = true*), the approximation $k - d_i$ replaces the testing and dividing by extra bases.

CONDITION1 aims at fulfilling our first criteria, namely, reducing the length of the expansion. In this case, parameter e guarantees that the new approximation will yield a much shorter chain such that is justifiable to use more expensive point operations instead of the usual sequence of doublings after each nonzero term. Similarly, *CONDITION2* is responsible for fixing a good balance between the different point operations. In particular, it will "smartly" insert more doublings, as these are the most efficient operations in most common ECC settings. In this case, e' is a security parameter that guarantees that the algorithm in fact trades expensive point operations by a large enough sequence of doublings such that is justifiable to introduce an extra nonzero term. Both parameters, e and e', vary according to the relative cost among point operations and even with the value of the scalar. Despite this complexity, we have been able to determine parameter values that are efficient for most cases by performing extensive tests with random numbers.

We have inserted the modifications above to the Frac-wmbNAF algorithm (see Section 4), since this representation generalizes the $(w)mb$NAF methods, and derived Algorithm 5.1 for bases {2,3} and {2,3,5}.

Notice the addition of *CONDITION1* and *2* in steps 2.2.5. and 2.4.1. As can be seen from the descriptions above, these techniques are quite general and give a high degree of freedom to adjust the algorithm to different settings with different constrains in the complexity level. In this work, we have focused on selecting parameters that achieve high performance without increasing excessively the complexity of the Multibase NAF algorithms.

Algorithm 5.1. Computing the Refined mbNAF of a positive integer

INPUT: scalar k, bases $\{2,3\}$ or $\{2,3,5\}$, digit set $D\backslash\{0\}=\{\pm 1, \pm 3, \ldots, \pm(m{=}2t{+}1)\}$
$w \geq 2 \in \mathbb{Z}^+; m = 2^{w-2} + s$ and $2^{w-2} < m < 2^{w-1}$, where $m \geq 3$ and $s \geq 1$ are odd integers ($m = 1, s = 0$ for case without precomputations)

OUTPUT: the Refined $(2,3)\text{NAF}_{w,t}(k)$ or $(2,3,5)\text{NAF}_{w,t}(k) = (\ldots, k_2^{(a_j)}, k_1^{(a_j)})$

1. $i = 1$, *exception* $= 0$
2. While $k > 0$ do
   - 2.1. If *exception* $= 0$ and ($k \bmod 2 = 0$ or $\ldots$ or $k \bmod a_J = 0$), then $k_i = 0$
   - 2.2. Else:
     - 2.2.1. $r = k \bmod 2^w$
     - 2.2.2. If $0 < r \leq m$, then $k_i = r$
     - 2.2.3. Elseif $m < r < (3m - 4s)$, then $k_i = r - 2^{w-1}$
     - 2.2.4. Elseif $(3m - 4s) \leq r < 2^w$, then $k_i = r - 2^w$
     - 2.2.5. If *CONDITION1* $= true$, then $k_i = d_i$
     - 2.2.6. $k = k - k_i$, *exception* $= 0$
   - 2.3. If $k \bmod 2 = 0$, then $k = k/2, k_i = k_i^{(2)}$
   - 2.4. Elseif $k_i = 0$
     - 2.4.1. If *CONDITION2* $= true$, then *exception* $= 1$
     - 2.4.2. Elseif $k \bmod 3 = 0$, then $k = k/3, k_i = k_i^{(3)}$
     - $\vdots$
     - 2.4.J. Elseif $k \bmod a_J = 0$, then $k = k/a_J, k_i = k_i^{(a_J)}$
   - 2.5. $i = i + 1$
3. Return $(\ldots, k_2^{(a_j)}, k_1^{(a_j)})$

The recommended conditional statements for Alg. 5.1 are detailed in Tables 2 and 3 for bases {2,3} and {2,3,5}, respectively. We remark that these are only recommended parameters, and that *CONDITION1* and *2* can be modified to suit the complexity constrains of a specific implementation, leading to different performance levels.

It is clear from Tables 2 and 3 that the original conditions of the Multibase NAF regarding non-adjacency (see (1)) have been relaxed. In particular, according to *CONDITION1*, it can be the case that fewer consecutive doublings are inserted for a particular window size.

Let us illustrate the proposed method with the following example.

Example 3. Using Algorithm 5.1 and Table 2, we find that the Refined mbNAF chain for computing $9750P$ using bases {2,3}, $w = 4, m = 5$, is $9750 = 5 \times 2^3 \times 3^5 + 5 \times 2 \times 3$, which has been derived using the sequence $\frac{9750}{2} \rightarrow \frac{4875}{3} \rightarrow 1625 - 5 \rightarrow \frac{1620}{2} \rightarrow \frac{810}{2} \rightarrow \frac{405}{3} \rightarrow \frac{135}{3} \rightarrow \frac{45}{3} \rightarrow \frac{15}{3} \rightarrow 5$.

Notice that the partial value 1625 is conveniently approximated to 1620, by means of *CONDITION1*, instead of 1624 (see Example 2), allowing the efficient insertion of several triplings to reduce the length of the expansion. If we compare the performance of the refined method when computing $9750P$ against the basic Multibase NAF approach using the same fractional window size (see Example

2), we can observe that the cost reduces significantly from 12D+1T+3mA = $108.4M$ to only 3D+5T+1mA = $82.4M$ (JQuartic, $1S = 0.8M$).

Table 2. Recommended parameters for *CONDITION1* and *2*, bases $\mathcal{A} = \{2, 3\}, d_i \in D\backslash\{0, k_i\}$.

Window w	*CONDITION1*	*CONDITION2*
2 $m = 1$	If ($\mathcal{D}$ mod 9 = 0 and $\mathcal{K}$ mod 8 ≠ 0) or ($\mathcal{D}$ mod 27 = 0 and $\mathcal{K}$ mod 16 ≠ 0) or ($\mathcal{D}$ mod 81 = 0 and $\mathcal{K}$ mod 32 ≠ 0)	If ($\mathcal{D}$ mod 16 = 0 and k mod 9 ≠ 0) or ($\mathcal{D}$ mod 32 = 0 and k mod 27 ≠ 0) or ($\mathcal{D}$ mod 64 = 0 and k mod 81 ≠ 0)
3 $m = 3$	If ($\mathcal{D}$ mod 27 = 0 and $\mathcal{K}$ mod 16 ≠ 0) or ($\mathcal{D}$ mod 81 = 0 and $\mathcal{K}$ mod 32 ≠ 0)	If ($\mathcal{D}$ mod 2^{w+1} = 0 and k mod 9 ≠ 0) or ($\mathcal{D}$ mod 2^{w+2} = 0 and k mod 27 ≠ 0) or ($\mathcal{D}$ mod 2^{w+3} = 0 and k mod 81 ≠ 0)
4 $m = 5, 7$	If ($\mathcal{D}$ mod 216 = 0 and $\mathcal{K}$ mod 32 ≠ 0) or ($\mathcal{D}$ mod 324 = 0 and $\mathcal{K}$ mod 64 ≠ 0)	
5 $m = 9, \ldots, 15$	If ($\mathcal{D}$ mod 144 = 0 and $\mathcal{K}$ mod 64 ≠ 0) or ($\mathcal{D}$ mod 432 = 0 and $\mathcal{K}$ mod 128 ≠ 0)	
6 $m = 17, \ldots, 31$	If ($\mathcal{D}$ mod 288 = 0 and $\mathcal{K}$ mod 128 ≠ 0) or ($\mathcal{D}$ mod 864 = 0 and $\mathcal{K}$ mod 256 ≠ 0)	

Table 3. Recommended parameters for *CONDITION1* and *2*, bases $\mathcal{A} = \{2, 3, 5\}, d_i \in D\backslash\{0, k_i\}$.

Window w	*CONDITION1*	*CONDITION2*

2
$m = 1$
	If ($\mathcal{K}$ mod 5 ≠ 0 and (
	($\mathcal{D}$ mod 9 = 0 and $\mathcal{K}$ mod 8 ≠ 0)
	or ($\mathcal{D}$ mod 27 = 0 and $\mathcal{K}$ mod 16 ≠ 0)
	or ($\mathcal{D}$ mod 81 = 0 and $\mathcal{K}$ mod 32 ≠ 0)))
	or ($\mathcal{D}$ mod 15 = 0 and $\mathcal{K}$ mod 8 ≠ 0)
		If k mod 3 ≠ 0
		If ($\mathcal{D}$ mod 2^{w+2} = 0 and k mod 25 ≠ 0)
		or ($\mathcal{D}$ mod 2^{w+3} = 0 and k mod 125 ≠ 0)
		or ($\mathcal{D}$ mod 2^{w+4} = 0 and k mod 625 ≠ 0)
		else
		If $\mathcal{K}$ mod 25 ≠ 0 and (
		or ($\mathcal{D}$ mod 2^{w+2} = 0 and k mod 9 ≠ 0)
		or ($\mathcal{D}$ mod 2^{w+3} = 0 and k mod 27 ≠ 0)
		or ($\mathcal{D}$ mod 2^{w+4} = 0 and k mod 81 ≠ 0))

3 $m = 3$	If ($\mathcal{K}$ mod 5 ≠ 0 and (($\mathcal{D}$ mod 27 = 0 and $\mathcal{K}$ mod 16 ≠ 0) or ($\mathcal{D}$ mod 81 = 0 and $\mathcal{K}$ mod 32 ≠ 0))) or ($\mathcal{D}$ mod 45 = 0 and $\mathcal{K}$ mod 32 ≠ 0)	
4 $m = 5, 7$	If ($\mathcal{K}$ mod 5 ≠ 0 and (($\mathcal{D}$ mod 108 = 0 and $\mathcal{K}$ mod 32 ≠ 0) or ($\mathcal{D}$ mod 324 = 0 and $\mathcal{K}$ mod 64 ≠ 0)))	
5 $m = 9, \ldots, 15$	If ($\mathcal{K}$ mod 5 ≠ 0 and (($\mathcal{D}$ mod 144 = 0 and $\mathcal{K}$ mod 64 ≠ 0) or ($\mathcal{D}$ mod 432 = 0 and $\mathcal{K}$ mod 128 ≠ 0)))	
6 $m = 17, \ldots, 31$	If ($\mathcal{K}$ mod 5 ≠ 0 and (($\mathcal{D}$ mod 288 = 0 and $\mathcal{K}$ mod 128 ≠ 0) or ($\mathcal{D}$ mod 864 = 0 and $\mathcal{K}$ mod 256 ≠ 0)))	

(*) $\mathcal{D} = k - d_i$; $\mathcal{K} = k - k_i$

As can be observed, the gain in performance with this method is obtained by increasing the complexity in the conversion step. This may or may not be a limiting factor depending on the characteristics of a particular implementation and the chosen platform. In cases where the conversion to multibase becomes non-negligible, the method would still remain practical for settings where the same scalar k is reused several times or the conversion can be carried out during

an *idle* time (e.g., between the first and second phase of the ECDH scheme during data transmission).

To evaluate the performance of this refined methodology for scalar multiplication, we implemented the method and carried out several tests. The results are summarized in the following section.

6 Performance Comparison

We have carried out several tests to demonstrate the high performance of the Multibase NAF methods discussed in this work when applied on standard, Jacobi quartic and Edwards curves. We implemented the traditional Frac-wNAF and the (Refined) Frac-wmbNAF (Algorithm 5.1) and ran the algorithms with different window sizes for 1000 160-bit scalars chosen randomly. In the case of Multibase NAF, we evaluated the methods when using the following sets of bases $\mathcal{A} : \{2, 3\}$ and $\{2, 3, 5\}$.

To estimate costs for each method, we first counted the required number of point operations per scalar, averaged the results and then calculated the cost using Table 1. Also, for windows $w > 2$ we included in the overall cost the cost of calculating the precomputed points. For computing these points, we consider two cases: points are left in projective coordinates (referred to as *case 1*), and points are converted to affine using *one* inversion (referred to as *case 2*). As expected, case 2 is advantageous using Jacobian coordinates, where mDA is significantly more efficient than the general DA version (see Table 1). Ultimately, the particular *I/M* ratio of an implementation will decide which case is more effective on a standard curve. In the case of JQuartic and InvEdw, we only consider case 1 as this scheme should be largely preferred because of the minimal difference of costs between general and mixed additions. Specifically, for Jacobian coordinates, we use the efficient scheme proposed in [22], and for JQuartic and InvEdw we apply the recently proposed scheme by the authors [20].

The costs using the various methods are summarized in Table 4. Costs with the label **Optimized** correspond to methods that have been slightly optimized by saving some initial computations. This technique is similar to that proposed in [13, Section 4.2.2] plus some additional savings gained with the use of composite operations (i.e., tripling, quintupling).

The "basic" operation count (without the aforementioned optimization) is detailed per method. In the case of windowed methods, the count is given separately for 7 and 6 precomputations (the latter case always corresponds to Jacobian coordinates only). Also, note that for Jacobian coord. we use doubling-addition (DA) operations instead of traditional additions. Hence, in this case, the total number of doublings is obtained by subtracting the number of doublings listed by that of additions. Finally, for case 2 (Jacobian), the total number of mDA operations is obtained by adding numbers listed in mDA and DA, as precomputed points are in affine and all the additions involve mixed coordinates.

As can be seen, in scenarios without precomputations, the basic Multibase NAF using bases {2,3} and {2,3,5} achieve better performance than the original

DB method based on the "Greedy" algorithm [8]. That is in addition to the attractive features of Multibase NAF such as simplicity and memory efficiency. More recently, Doche et al. [10] introduced a new method that also finds double-base chains without using the "Greedy" algorithm, although using a somewhat more complex search-based approach in comparison with the basic Multibase NAF. This method's cost is comparable to (2,3)NAF, but slightly higher than that achieved by (2,3,5)NAF. More importantly, the proposed Refined Multibase NAF method presents even lower costs in all the cases, bases {2,3} and {2,3,5}. The improvement is especially significant in the case without precomputations, which makes our method especially interesting for applications on constrained devices. With Jacobi quartics, the advantage of the Refined *mb*NAF using bases {2,3,5} is as large as 9.3% over the traditional NAF. In Jacobian coord., that advantage rises to 9.8%.

Interestingly enough, in the case of windowed methods, we observe that the refined multibase algorithms surpass the performance of traditional binary methods for all the curve shapes analyzed, contradicting conclusions by [1] and [10]. Most remarkably, if we consider the "basic" operation count, the Refined (2,3,5)NAF with no precomputations is comparable and/or surpasses the performance of the fastest NAF method using an optimal window with 7 and 6 precomputed points for Jacobi quartics and Jacobian coordinates, respectively. For the latter, the multibase method is superior always that $1I > 23M$. Even if we consider the optimized version of the NAF method, the multibase method achieves higher performance always that $1I > 40M$.

For the record, we also include results by [2] and [1]. These works use highly optimized radix-2 and double-base (DB) scalar multiplications. We can see that both the basic Multibase NAF using precomputations and the refined version offer lower computing costs for the cases when precomputations include one or nil field inversions. Moreover, our optimized implementations of wNAF and Frac-wNAF are also superior in performance to these works. The latter is due to a combination of improved precomputation schemes, more efficient point formulas and the inclusion of the technique to save initial computations.

In particular, the Refined *mb*NAF using 6 and 7 precomputed points achieves the highest performance using bases {2,3,5} in the case of standard curves and Jacobi quartics. In the case of Edwards curves using inverted Edwards coordinates the lowest cost is achieved by the same method using bases {2,3} and 7 points. (The lowest costs per curve are highlighted in **bold**.) Also, note that for Jacobian the highest speed up is achieved with a table of the form {3,5,...,13} (6 points; 160 bits), which corresponds to a fractional window and, thus, highlights the importance of this recoding for Jacobian coordinates.

7 Conclusion

We have introduced a refined multibase method and other several optimizations, including improved point operation formulas, that have been efficiently applied to speed up (multibase) methods for scalar multiplication. In particular,

Table 4. Comparison of scalar multiplication methods ($n = 160$ bits; $1S = 0.8M$)

Method	# pts	Operation count					JQuartic		InvEdw		Jacobian	
		D	T	Q	mA (mDA)	A (DA)	Precomp	Total	Precomp	Total	Precomp	Total
Refined (2,3,5)NAF (this work)	0	109.9	21.6	6.2	31.0	-	0	**1267.5*M***	-	-	0	**1459.6*M***
Refined (2,3)NAF (this work)	0	120.6	23.9	-	35.6	-	0	1288.4M	0	**1350.5*M***	0	1484.8M
(2,3,5)NAF, Longa [17]	0	96.9	24.3	10.0	32.0	-	0	1292.6M	-	-	0	1485.9M
(2,3)NAF, Longa [17]	0	113.6	28.4	-	37.8	-	0	1314.9M	0	1381.0M	0	1513.3M
Tree-based DB, Doche et al. [10]	0	91.1	42.9	-	32.8	-	0	1302.6M	0	1376.8M	0	1493.1M
DB (Greedy), Dimitrov et al. [8]	0	94.2	40.9	-	36.3	-	0	1328.2M	0	1402.5M	0	1523.2M
NAF	0	158.7	-	-	52.9	-	0	1396.6M	0	1449.5M	0	1618.7M
Optimized	7			-			59.6M	**1214.1*M***	-	-	55.4M	**1425.8*M***
	6 (Jac)			-			-	-	-	-	1I+72.4M	**1I+1393.5*M***
Refined (2,3,5)NAF (this work)	7	132.4	10.0	3.8	2.9	18.2	59.6M	1226.5M	-	-	55.4M	1435.2M
	6 (Jac)	129.8	10.9	4.3	5.0	16.3	-	-	-	-	1I+72.4M	1I+1406.5M
Optimized	7			-			59.6M	1216.0M	61.2M	**1265.6*M***	55.4M	1434.1M
	6 (Jac)			-			-	-	-	-	1I+72.4M	1I+1396.4M
Refined (2,3)NAF (this work)	7	143.9	8.3	-	3.0	20.0	59.6M	1229.2M	61.2M	1277.1M	55.4M	1445.1M
	6 (Jac)	142.2	9.4	-	5.1	18.3	-	-	-	-	1I+72.4M	1I+1410.9M
(2,3,5)NAF, Longa [17]	7	129.5	10.5	5.1	2.7	18.8	59.6M	1240.9M	-	-	55.4M	1450.4M
	6 (Jac)	124.0	12.6	6.0	5.5	16.0	-	-	-	-	1I + 72.4M	1I+1422.4M
(2,3)NAF, Longa [17]	7	140.2	11.0	-	2.9	20.4	59.6M	1240.5M	61.2M	1290.5M	55.4M	1456.0M
	6 (Jac)	136.1	13.6	-	5.8	17.8	-	-	-	-	1I+72.4M	1I+1423.3M
Optimized (this work)	7			-			59.6M	1233.1M	61.2M	1276.2M	55.4M	1463.1M
	6 (Jac)			-			-	-	-	-	1I+72.4M	1I+1419.5M
Frac-wNAF	7	157.3	-	-	3.3	22.8	59.6M	1245.9M	61.2M	1289.5M	55.4M	1476.4M
	6 (Jac)	157.4	-	-	7.0	20.3	-	-	-	-	1I+72.4M	1I+1436.4M
DB (Greedy), Bernstein et al. [1]	7			N/A			N/A	1311.0M	N/A	1290.3M	N/A	1504.3M
NAF-based, Bernstein et al. [2]	7			N/A			N/A	1308.5M	N/A	1287.8M	N/A	1511.9M
	8			N/A			-	-	-	-	-	1I+1434.1M

we have applied the concept of "fractional" windows to the multibase scenario, generalizing Multibase NAF methods to any number of precomputations. Also, we have presented a more comprehensive analysis of scalar multiplications methods and tested their performance in comparison with Multibase NAF methods using different elliptic curve shapes. The conclusion is that currently the proposed Refined *mb*NAF achieves the lowest costs found in the literature among methods without precomputations, independently of the curve selected. Using bases {2,3,5} and {2,3} we can perform a scalar multiplication with costs of only $1459M$ (field multiplications) and $1350M$ in Jacobian and inverted Edwards coordinates (respect.). With Jacobi quartics, that cost can be as low as $1267M$ using bases {2,3,5}. Similar results are attained by the same method when using precomputations. In this case, we present the lowest costs reported in the literature: $1425M$ or $1I + 1393M$ in Jacobian, $1265M$ in inverted Edwards and $1214M$ in extended Jacobi quartic coordinates.

Finally, we have included the theoretical analysis of Multibase NAF and its different variants, detailing the average zero and nonzero density characterizing these representations. This analysis has been confirmed with our extensive tests.

Acknowledgments. We would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Ontario Centres of Excellence (OCE) for partially supporting this work. We would also like to thank the reviewers for their useful comments.

References

1. Bernstein, D., Birkner, P., Lange, T., Peters, C.: Optimizing Double-Base Elliptic-Curve Single-Scalar Multiplication. INDOCRYPT 2007, LNCS, vol. 4859, pp. 167–182. Springer, Heidelberg (2007)
2. Bernstein, D., Lange, T.: Analysis and Optimization of Elliptic-Curve Single-Scalar Multiplication. Cryptology ePrint Archive, Report 2007/455 (2007)
3. Bernstein, D., Lange, T.: Faster Addition and Doubling on Elliptic Curves. ASIACRYPT 2007, LNCS, vol. 4833, pp. 29–50. Springer, Heidelberg (2007)
4. Bernstein, D., Lange, T.: Inverted Edwards Coordinates. AAECC 2007, LNCS, vol. 4851, pp. 20–27. Springer, Heidelberg (2007)
5. Billet, O., Joye, M.: The Jacobi Model of an Elliptic Curve and Side-Channel Analysis. AAECC 2003, LNCS, vol. 2643, pp. 34–42. Springer, Heidelberg (2003)
6. Ciet, M., Joye, M., Lauter, K., Montgomery, P.L.: Trading Inversions for Multiplications in Elliptic Curve Cryptography. Designs, Codes and Cryptography, vol. 39(2), pp. 189–206 (2006)
7. Dimitrov, V., Jullien, G., Miller, W.: Theory and Applications for a Double-Base Number System. ARITH 1997, pp. 44 (1997)
8. Dimitrov, V., Imbert, L., Mishra, P.K.: Efficient and Secure Elliptic Curve Point Multiplication using Double-Base Chains. ASIACRYPT 2005, LNCS, vol. 3788, pp. 59–78. Springer, Heidelberg (2005)
9. Dimitrov, V., Mishra, P.K.: Efficient Quintuple Formulas for Elliptic Curves and Efficient Scalar Multiplication using Multibase Number Representation. ISC 2007, LNCS, vol. 4779, pp. 390–406. Springer, Heidelberg (2007)

10. Doche, C., Habsieger, L.: A Tree-Base Approach for Computing Double-Base Chains. 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. INDOCRYPT 2006, LNCS, vol. 4329, pp. 335–348. Springer, Heidelberg (2006)
12. Edwards, H.: A Normal Form for Elliptic Curves. Bulletin of the American Mathematical Society 44, 393–422 (2007)
13. Elmegaard-Fessel, L.: Efficient Scalar Multiplication and Security against Power Analysis in Cryptosystems based on the NIST Elliptic Curves over Prime Fields. Master Thesis, University of Copenhagen (2006)
14. Hankerson, D., Menezes, A., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer-Verlag (2004)
15. Hisil, H., Wong, K., Carter, G., Dawson, E.: Faster Group Operations on Elliptic Curves. Cryptology ePrint Archive, Report 2007/441 (2007)
16. Hisil, H., Wong, K., Carter, G., Dawson, E.: An Intersection Form for Jacobi-Quartic Curves. Personal communication (2008)
17. Longa, P.: Accelerating the Scalar Multiplication on Elliptic Curve Cryptosystems over Prime Fields. Master Thesis, University of Ottawa (2007). Available at http://patricklonga.bravehost.com/publications.html
18. Longa, P.: ECC Point Arithmetic Formulae (EPAF). Available at http://patricklonga.bravehost.com/jacobian.html
19. Longa, P., Gebotys, C.: Setting Speed Records with the (Fractional) Multibase Non-Adjacent Form Method for Efficient Elliptic Curve Scalar Multiplication. CACR Technical Report, CACR 2008-06, University of Waterloo (2008)
20. Longa, P., Gebotys, C.: Novel Precomputation Schemes for Elliptic Curve Cryptosystems. (To appear) ACNS 2009, LNCS, vol. 5536, pp. 71–88, Springer, Heidelberg (2009)
21. Longa, P., Miri, A.: Fast and Flexible Elliptic Curve Point Arithmetic over Prime Fields. IEEE Trans. Comp. 57(3), 289–302 (2008)
22. Longa, P., Miri, A.: New Composite Operations and Precomputation Scheme for Elliptic Curve Cryptosystems over Prime Fields. PKC 2008, LNCS, vol. 4939, pp. 229–247. Springer, Heidelberg (2008)
23. Meloni, N.: New Point Addition Formulae for ECC Applications. WAIFI 2007, LNCS, vol. 4547, pp. 189–201. Springer, Heidelberg (2007)
24. Möller, B.: Improved Techniques for Fast Exponentiation. ICISC 2002, LNCS, vol. 2587, pp. 298–312. Springer, Heidelberg (2003)
25. Möller, B.: Fractional Windows Revisited: Improved Signed - Digit Representations for Efficient Exponentiation. ICISC 2004, LNCS, vol. 3506, pp. 137–153. Springer, Heidelberg (2005)

A Derivation of composite operations of form *dP*

Consider the following formula due to [23] to add two points $P = (X_1, Y_1, Z)$ and $Q = (X_2, Y_2, Z)$ with the same coordinate Z in Jacobian coordinates:

$$\begin{array}{c} X_3 = (Y_2 - Y_1)^2 - (X_2 - X_1)^3 - 2X_1(X_2 - X_1)^2,\ Z_3 = Z(X_2 - X_1) \\ Y_3 = (Y_2 - Y_1)(X_1(X_2 - X_1)^2 - X_3) - Y_1(X_2 - X_1)^3\ . \end{array} \tag{5}$$

To derive composite operations of the form dP, where $d > 2$ is a small prime, we follow the next scheme using (5) to perform additions:

$$dP = (2P + (\ldots (2P + (2P + P)) \ldots)) \ . \tag{6}$$

According to (6), we first compute $2P$ with the following [21]:

$X_2 = [3(X_1 + Z_1^2)(X_1 - Z_1^2)]^2 - 8X_1Y_1^2,\ Z_2 = 2Y_1Z_1 = (Y_1 + Z_1)^2 - Y_1^2 - Z_1^2,$
$Y_2 = [3(X_1 + Z_1^2)(X_1 - Z_1^2)](4X_1Y_1^2 - X_2) - 8Y_1^4$.

And then, we perform the addition $3P = 2P+P = (X_2, Y_2, Z_2)+(X_1^{(1)}, Y_1^{(1)}, Z_1^{(1)}) = (X_3, Y_3, Z_3)$ using formula (5), where $(X_1^{(1)}, Y_1^{(1)}, Z_1^{(1)}) = (X_1(4Y_1^2), Y_1(8Y_1^3), Z_1(2Y_1)) \equiv (X_1, Y_1, Z_1)$, as follows:

$X_3 = (Y_1^{(1)} - Y_2)^2 - (X_1^{(1)} - X_2)^3 - 2X_2(X_1^{(1)} - X_2)^2,$
$Y_3 = (Y_1^{(1)} - Y_2)[X_2(X_1^{(1)} - X_2)^2 - X_3] - Y_2(X_1^{(1)} - X_2)^3,\ Z_3 = Z_2(X_1^{(1)} - X_2)$.

After scaling and replacement of some multiplications by squarings, computation of $3P$ takes the form:

$$X_3 = \omega^2 - 4\theta^3 - 8X_2\theta^2,\ Y_3 = \omega[4X_2\theta^2 - X_3] - 8Y_2\theta^3,\ Z_3 = 2Z_1^{(1)}\theta, \tag{7}$$

where $\alpha = 3(X_1 + Z_1^2)(X_1 - Z_1^2),\ \theta = 4X_1Y_1^2 - X_2,\ \omega = 16Y_1^4 - 2Y_2,\ X_2 = \alpha^2 - 8X_1Y_1^2,\ 2Y_2 = (\alpha + \theta)^2 - \alpha^2 - \theta^2 - 16Y_1^4,\ Z_1^{(1)} = (Y_1 + Z_1)^2 - Y_1^2 - Z_1^2$. Following the same approach for the next addition in (6), it is easy to derive the formula for the quintupling of a point $5P = (X_5, Y_5, Z_5)$ in Jacobian coordinates (special case $a = -3$). The new formula is given by:

$$\begin{gathered} X_5 = \gamma^2 - 4\phi^3 - 8X_2^{(1)}\phi^2,\ Y_5 = \gamma[4X_2^{(1)}\phi^2 - X_5] - 8Y_2^{(1)}\phi^3, \\ Z_5 = 2Z_2[(\theta + \phi)^2 - \theta^2 - \phi^2], \end{gathered} \tag{8}$$

where $\alpha = 3(X_1 + Z_1^2)(X_1 - Z_1^2),\ \theta = X_1^{(1)} - X_2,\ \omega = 2Y_1^{(1)} - 2Y_2,\ X_2 = \alpha^2 - 2X_1^{(1)},\ 2Y_2 = (\alpha + \theta)^2 - \alpha^2 - \theta^2 - 2Y_1^{(1)},\ Z_2 = (Y_1 + Z_1)^2 - Y_1^2 - Z_1^2,\ \gamma = \omega^2+\phi^2-(\omega+\phi)^2-4Y_2^{(1)},\ X_1^{(1)} = 4X_1Y_1^2,\ 2Y_1^{(1)} = 16Y_1^4,\ X_2^{(1)} = 4X_2\theta^2,\ Y_2^{(1)} = 8Y_2\theta^3,\ \phi = \omega^2 - 4\theta^3 - 3X_2^{(1)}$.

This quintupling formula costs $10M + 12S$. In the general case (random a), the cost is fixed at $9M + 15S$ with the following change of parameters: $\alpha = 3X_1^2 + aZ_1^4,\ X_1^{(1)} = 2[(X_1 + Y_1^2)^2 - X_1^2 - Y_1^4]$.

Again, following the same procedure for the next addition in (6), it is straightforward to derive the formula for the septupling $7P = (X_7, Y_7, Z_7)$ in Jacobian coord. (case $a = -3$). The new septupling formula is given by:

$$X_7 = \varphi^2 - 4\sigma^3 - 8X_2^{(2)}\sigma^2,\ Y_7 = \varphi[4X_2^{(2)}\sigma^2 - X_7] - 8Y_2^{(2)}\sigma^3,\ Z_7 = 2Z_2^{(2)}\sigma, \tag{9}$$

where $\varphi = \gamma^2+\sigma^2-(\gamma+\sigma)^2-4Y_2^{(2)},\ \sigma = \gamma^2-4\phi^3-3X_2^{(2)},\ \gamma = \omega^2+\phi^2-(\omega+\phi)^2-4Y_2^{(1)},\ X_2^{(2)} = 4X_2^{(1)}\phi^2,\ Y_2^{(2)} = 8Y_2^{(1)}\phi^3,\ Z_2^{(2)} = 2Z_2[(\theta + \phi)^2 - \theta^2 - \phi^2],\ \phi =$

$\omega^2 - 4\theta^3 - 3X_2^{(1)},\ \omega = 2Y_1^{(1)} - 2Y_2,\ \theta = X_1^{(1)} - X_2,\ \alpha = 3(X_1 + Z_1^2)(X_1 - Z_1^2),\ X_2^{(1)} = 4X_2\theta^2,\ Y_2^{(1)} = 8Y_2\theta^3,\ X_2 = \alpha^2 - 2X_1^{(1)},\ X_1^{(1)} = 4X_1Y_1^2,\ 2Y_1^{(1)} = 16Y_1^4,\ 2Y_2 = (\alpha+\theta)^2 - \alpha^2 - \theta^2 - 2Y_1^{(1)},\ Z_2 = (Y_1+Z_1)^2 - Y_1^2 - Z_1^2$. This septupling formula costs $14M + 15S$. In the general case (parameter a random), the cost is fixed at $13M + 18S$ with the following change of parameters: $\alpha = 3X_1^2 + aZ_1^4,\ X_1^{(1)} = 2[(X_1+Y_1^2)^2 - X_1^2 - Y_1^4]$.

B Proof of the average zero and nonzero densities of $(w)mb$NAF using bases {2,3} and {2,3,5}

The method can be modeled as a Markov chain with three states (case $\mathcal{A}$=$\{2,3\}$) : "$0^{(2)}$", "$0^{(3)}$" and "$\underbrace{0^{(2)}\ldots0^{(2)}}_{w-1}k_i^{(2)}$", with the following probability matrix:

$$\begin{pmatrix} \text{"}0^{(2)}\text{"} & : & 1/2 & \frac{2^{w-2}-\lfloor(2^{w-2}+1)/3\rfloor}{2^w} & \frac{2^{w-2}+\lfloor(2^{w-2}+1)/3\rfloor}{2^w} \\ \text{"}0^{(3)}\text{"} & : & 0 & 1/3 & 2/3 \\ \text{"}\underbrace{0^{(2)}\ldots0^{(2)}}_{w-1}k_i^{(2)}\text{"} & : & 1/2 & \frac{2^{w-2}-\lfloor(2^{w-2}+1)/3\rfloor}{2^w} & \frac{2^{w-2}+\lfloor(2^{w-2}+1)/3\rfloor}{2^w} \end{pmatrix}$$

This Markov chain is irreducible and aperiodic, and hence, it has stationary distribution, which is given by:

$$\left(\underbrace{0^{(2)}\ldots0^{(2)}}_{w-1}k_i^{(2)}, 0^{(2)}, 0^{(3)}\ :\ \frac{2^w}{2^{w+1}+3(2^{w-2}-s)}\ \ \frac{2^w}{2^{w+1}+3(2^{w-2}-s)}\ \ \frac{3(2^{w-2}-s)}{2^{w+1}+3(2^{w-2}-s)})\right)$$

Thus, nonzero digits k_i appear 2^w out of $w{\cdot}2^w+2^w+3(2^{w-2}-\lfloor(2^{w-2}+1)/3\rfloor)$, which proves our assertion about the nonzero density. Doublings and triplings (i.e., number of zero and nonzero digits with base 2 and 3, respect.) appear $2^w \cdot w+2^w$ and $3(2^{w-2}-\lfloor(2^{w-2}+1)/3\rfloor)$ out of $w{\cdot}2^w+2^w+3(2^{w-2}-\lfloor(2^{w-2}+1)/3\rfloor)$, respectively. This proves our assertion about the average density of doublings and triplings.

In the case of $\mathcal{A} = \{2,3,5\}$, there are four states: "$0^{(2)}$", "$0^{(3)}$", "$0^{(5)}$" and "$\underbrace{0^{(2)}\ldots0^{(2)}}_{w-1}k_i^{(2)}$". The probability matrix in this case is as follows (see Theorem 2 for notation):

$$\begin{pmatrix} \text{"}0^{(2)}\text{"} & : & 1/2 & \frac{2^{w-2}-s}{2^w} & \frac{2^{w-2}-r+s-t}{2^{w+2}} & \frac{3{\cdot}2^{w-2}+r+3s+t}{2^{w+2}} \\ \text{"}0^{(3)}\text{"} & : & 0 & 1/3 & 1/6 & 1/2 \\ \text{"}0^{(5)}\text{"} & : & 0 & 0 & 1/5 & 4/5 \\ \text{"}\underbrace{0^{(2)}\ldots0^{(2)}}_{w-1}k_i^{(2)}\text{"} & : & 1/2 & \frac{2^{w-2}-s}{2^w} & \frac{2^{w-2}-r+s-t}{2^{w+2}} & \frac{3{\cdot}2^{w-2}+r+3s+t}{2^{w+2}} \end{pmatrix}$$

This Markov chain is irreducible and aperiodic with stationary distribution:

$$\left(\text{"}\underbrace{0^{(2)}\ldots0^{(2)}}_{w-1}k_i^{(2)}\text{"},\text{"}0^{(2)}\text{"},\text{"}0^{(3)}\text{"},\text{"}0^{(5)}\text{"} \;:\; \tfrac{2^{w+3}}{\omega} \;\; \tfrac{2^{w+3}}{\omega} \;\; \tfrac{24(2^{w-2}-s)}{\omega} \;\; \tfrac{5(2^{w-1}-r-t)}{\omega}\right)$$

where $\omega = 49 \cdot 2^{w-1} - 5r - 24s - 5t$. Therefore, nonzero digits k_i appear 2^{w+3} out of $2^{w+3} \cdot w + 2^{w+3} + 24(2^{w-2} - s) + 5(2^{w-1} - r - t)$, which proves our assertion about the nonzero density. Doublings, triplings and quintuplings appear $2^{w+3} \cdot w + 2^{w+3}$, $24(2^{w-2} - s)$ and $5(2^{w-1} - r - t)$ out of $2^{w+3} \cdot w + 2^{w+3} + 24(2^{w-2} - s) + 5(2^{w-1} - r - t)$, respectively. This proves our assertion about the average density of the aforementioned operations.

C Proof of the average zero and nonzero densities of Fractional *wmb*NAF using bases {2,3}

Let us consider the following states to model this fractional windows method using Markov chains: "$0^{(2)}$", "$0^{(3)}$", "$\underbrace{0^{(2)}\ldots0^{(2)}}_{w-2}k_i^{(2)}$", and "$\underbrace{0^{(2)}\ldots0^{(2)}}_{w-1}k_i^{(2)}$". Then, the probability matrix is as follows (see Theorem 3 for notation):

$$\begin{pmatrix}
\text{"}0^{(2)}\text{"} & : & 1/2 & \frac{t-\lfloor(t+1)/3\rfloor}{4t} & \frac{(2^{w-2}-t)(t+\lfloor(t+1)/3\rfloor)}{2^w t} & \frac{t+\lfloor(t+1)/3\rfloor}{2^w} \\
\text{"}0^{(3)}\text{"} & : & 0 & 1/3 & \frac{2^{w-2}-t}{3\cdot 2^{w-3}} & \frac{t}{3\cdot 2^{w-3}} \\
\text{"}\underbrace{0^{(2)}\ldots0^{(2)}}_{w-2}k_i^{(2)}\text{"} & : & 0 & \alpha & \beta & 1-\alpha-\beta \\
\text{"}\underbrace{0^{(2)}\ldots0^{(2)}}_{w-1}k_i^{(2)}\text{"} & : & 1/2 & \frac{t-\lfloor(t+1)/3\rfloor}{4t} & \frac{(2^{w-2}-t)(t+\lfloor(t+1)/3\rfloor)}{2^w t} & \frac{t+\lfloor(t+1)/3\rfloor}{2^w}
\end{pmatrix}$$

where $\alpha = \frac{(2^{w-2}-t)-\lfloor(2^{w-2}-t+1)/3\rfloor}{2(2^{w-2}-t)}$ and $\beta = \frac{(2^{w-2}-t)+\lfloor(2^{w-2}-t+1)/3\rfloor}{2^{w-1}}$. This Markov chain is irreducible and aperiodic with the following stationary distribution:

$$\left(\underbrace{0^{(2)}\ldots0^{(2)}}_{w-1}k_i^{(2)},\underbrace{0^{(2)}\ldots0^{(2)}}_{w-2}k_i^{(2)},0^{(2)},0^{(3)} \;:\; \tfrac{16t}{\mu} \;\; \tfrac{12((u+v)-2^{w-2})}{\mu} \;\; \tfrac{16(t-2^{w-2})}{\mu} \;\; \tfrac{16t}{\mu}\right)$$

where $\mu = 16t - 12(u+v) + 7 \cdot 2^w$. Therefore, the nonzero digits k_i appear 2^w out of $8t - 3(u+v) + 2^{w-2}(4w-1)$, which proves our assertion about the nonzero density. Doublings and triplings appear $8t + 2^w(w-1)$ and $3(2^{w-2} - (u+v))$ out of $8t - 3(u+v) + 2^{w-2}(4w-1)$, respectively. This proves our assertion about the average density of doublings and triplings.