# Efficient Quintuple Formulas for Elliptic Curves and Efficient Scalar Multiplication Using Multibase Number Representation

**Abstract**

In the current work we propose two efficient formulas for computing the 5-fold ($5P$) of an elliptic curve point $P$. One formula is for curves over finite fields of even characteristic and the other is for curves over prime fields. Double base number systems (DBNS) have been gainfully exploited to compute scalar multiplication efficiently in ECC. Using the proposed point quintupling formulas one can use 2, 5 and 3, 5 (besides 2, 3) as bases of the double base number system. In the current work we propose a scalar multiplication algorithm, which uses a representation of the scalar using three bases 2, 3 and 5 and computes the scalar multiplication very efficiently. The proposed scheme is faster than all sequential scalar multiplication algorithms reported in literature.

**Keywords** Elliptic Curve Cryptosystems, Scalar Multiplication, Quintupling, Efficient Curve Arithmetic.

## 1 Introduction

Undoubtedly, the papers [24, 27], which independently proposed the elliptic curve cryptography (ECC), are among the most cited papers in cryptology. In ECC, elliptic curves over finite fields are used to generate finite abelian groups to implement public key cryptographic primitives. The advantage of using elliptic curve groups is: there is no known subexponential algorithm to solve the elliptic curve discrete logarithm problem (ECDLP). This means that a desired security level can be achieved with a much smaller key size in comparison to other public key schemes. This, in turn, leads to efficient implementation and efficient use of transmission bandwidth. Another advantage of ECC is the flexibility it offers in the choice of various security parameters (like group order and representation of its elements, group arithmetic, underlying field and its representation etc) used in its implementation.

Scalar multiplication of elliptic curve points is one of the most researched operation in cryptography. If $P$ is a point on an EC and $d$ is an integer, the operation computing the $d$-fold of $P$, namely the point $dP$, is called scalar multiplication. Several methods have been reported in literature to compute scalar multiplication efficiently and securely from prying eyes (side-channel attackers). The strategies used for enhancement of efficiency are: (1) efficient group arithmetic in the elliptic curve group, (2) using a "nice" representation for the scalar (the sparser, the better), (3) use of precomputation to precompute some points required later (4) using efficient algorithms like sliding window method, comb methods or use of efficient addition chains, like Montgomery's ladder etc.

In the current work, we propose a new scalar multiplication algorithm, the essence of whose efficiency comes from two new efficient point quintupling formulas for curves over arbitrary prime and binary fields and use of a very sparse representation of the scalar using three bases. For last couple of years, double base number system (DBNS) has been proposed to be used in this context by several authors [7, 11, 2, 12, 15]. For general curves, a DBNS representation of the scalar using 2 and 3 as bases has been proved quite efficient [11]. In search of sublinear scalar multiplication algorithms, authors of [2] have used complex bases, 3 and $\tau$ for Koblitz curves. However, their proof of sublinearity has some flaws. In

[12], the authors have proved that a sublinear algorithm is indeed possible using three bases, namely $\tau$, $\tau - 1$ and $\tau^2 + \tau + 1$. Their software and hardware implementations using two bases $\tau$ and $\tau - 1$ are fast enough to give the feeling of a sublinear algorithm, but it lacks a theoretical proof. In [15], authors have used the precomputations to obtain further speed-ups. In this work, we represent the scalar using a generalization of DBNS representation, namely, multibase number representation. The exponent scalar is represented as a sum/difference of products of powers of 2, 3 and 5.

**Our Contributions:** The main contribution of this work are two formulas for computing 5-fold ($5P$) of an elliptic curve point $P$, one for curves over binary fields and the other for curves over prime fields. These formulas can be used to compute the scalar multiplication using quinary or DBNS expansion (using 2,5 or 3,5 as bases) of the scalar. We also generalize the algorithm used to compute scalar multiplication in double base [11] to accommodate a third base, namely, 5. Thus, the proposed scalar multiplication algorithms use a representation of the scalar as sum/difference of product of powers of 2, 3 and 5. Experimental results indicate it to be faster than all scalar multiplication algorithms known so far for general curves over binary and large prime fields.

## 2 Background

In this section, we briefly outline the materials used as a prerequisite for this work. Interested readers can consult the cited works to check details.

### 2.1 ECC

In this section, we give a brief overview of elliptic curve cryptography. Details can be found in [1, 3, 4, 19].
**Definition** An elliptic curve $E$ over a field $K$ is defined by an equation

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{1}$$

where $a_1, a_2, a_3, a_4, a_6 \in K$, and $\Delta \neq 0$, where $\Delta$ is the discriminant of $E$. Applying admissible changes of variables, the Weierstrass equation (1) can be simplified. Over prime fields, $K = F_p$, of large characteristic, the equation (1) can be simplified to

$$y^2 = x^3 + ax + b, \tag{2}$$

where $a, b \in K$ and $\Delta = 4a^3 + 27b^2 \neq 0$.

Over binary fields $K = F_{2^m}$, the non-supersingular curves are used for cryptography, whose Weierstrass equation can be simplified to the form:

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

where $a, b \in K$ and $\Delta = b \neq 0$.

The set $E(K)$ of rational points on an elliptic curve $E$ defined over a field $K$ forms an abelian group, under the operation (denoted additively) defined by the secant and tangent law. The special point $\mathcal{O}$, called *point at infinity* plays the role of identity in this group.

The most natural representation for a point 0n an elliptic curve group is the affine representation, i.e., by an ordered pair of field elements satisfying the equation of the curve. However, group operations in affine representation require field inversions, which are the most expensive among field operations. To avoid inversions, several point representations in homogeneous (projective) coordinates have been proposed in the literature. The choice of a coordinate system for point representation in the elliptic curve group largely depends upon the so-called $[i]/[m]$-ratio, the ratio between the cost of a field inversion to that of a field multiplication. It is generally assumed that for binary fields $3 \leq [i]/[m] \leq 10$ and but it is significantly higher (30 or more) for prime fields [17]. In this paper we consider affine ($\mathcal{A}$) coordinates for curves defined over binary fields and Jacobian ($\mathcal{J}$) coordinates, where the point $P = (X : Y : Z)$ corresponds to the point $(X/Z^2, Y/Z^3)$ on the elliptic curve for curves defined over prime fields.

To denote cost of field operations, we will use $[i]$, $[s]$ and $[m]$ to denote the cost of one inversion, one squaring and one multiplication respectively. We shall always neglect the cost of field additions. Also, over binary fields, we will neglect squarings as they are almost free (if normal bases are used) or of negligible cost (linear operation) (see [20] for more details). Moreover, over large prime fields, we will assume that $[s] = 0.8[m]$.

For curves over binary fields, we will use several elliptic curve group operations along with the quintupling operation presented in Section 3. These formulas have been listed in Table 2.1. We have included only those algorithms which will be used in this work. One operation needs a special mention: a repeated doubling formula ($w$-DBL) for these curves, originally proposed by Guajardo and Paar in [18] and subsequently improved by Lopez and Dahab in [10], which requires just one inversion to compute $2^w P, w \geq 1$.

Table 1: Costs of various Elliptic Curve group operations. The costs for curves over binary fields ($E(F_{2^m})$) are in affine coordinates. Those for curves over prime fields ($E(F_p)$) are in Jacobian coordinates.

| Operation | Output | For $E(F_{2^m})$ | | For $E(F_p)$ | |
|---|---|---|---|---|---|
| | | proposed | Cost | proposed | Cost |
| DBL($P$) | $2P$ | – | $1[i] + 2[m]$ | – | $6[s] + 4[m]$ |
| ADD($P$, $Q$) | $P + Q$ | – | $1[i] + 2[m]$ | – | $4[s] + 12[m]$ |
| mADD($P$, $Q$) | – | – | – | [8] | $3[s] + 8[m]$ |
| $w$-DBL($P$) | $2^w P$ | [10] | $1[i] + (4w - 2)[m]$ | [21] | $4w[m] + (4w + 2)[s]$ |
| DA($P, Q$) | $2P \pm Q$ | [6] | $1[i] + 9[m]$ | – | - |
| TPL($P$) | $3P$ | [6] | $1[i] + 7[m]$ | [11] | $10[m] + 6[s]$ |
| $w$-TPL | $3^w P$ | – | – | [11] | $10w[m] + (6w - 5)[s]$ |
| TA($P$, $Q$) | $3P \pm Q$ | [6] | $2[i] + 9[m]$ | - | - |

For curves over prime fields of large characteristics, we will use Jacobian coordinates ($\mathcal{J}$). The following formulas for group arithmetic are available to us: DBL, $w$-DBL, TPL, $w$-TPL and ADD, which compute $2P, 2^w P, 3P, 3^w P$ and $P + Q$ respectively. Also, if the base point is given in affine coordinates ($Z = 1$), then the cost of the so-called *mixed addition* (mADD) ($\mathcal{J} + \mathcal{A} \rightarrow \mathcal{J}$) requires fewer computation than generic addition. Also, DBL and TPL are less expensive when $a = -3$ in (2). In Table 2.1, we summarize the complexity of these different elliptic curve formulas.

All ECDLP based cryptographic primitives, like encryption, decryption, signature generation and verification, need the operation of scalar multiplication. Given an integer $d$ and an elliptic curve point $P$, it is the operation of computing $dP$. Efficiency of the scalar multiplication depends largely upon efficiency of the algorithms used for group arithmetic and representation of the scalar. In this work, we present two new algorithms for efficient group arithmetic and a new representation of the scalar using three bases. This combination considerably accelerates the computation of scalar multiplication in ECC.

## 2.2 Multibase Representation of an Integer

Let $k$ be an integer and let $\mathcal{B} = \{b_1, \cdots, b_l\}$ be a set of "small" integers. A representation of $k$ as a sum of powers of elements of $\mathcal{B}$ ($\Sigma_{j=1}^{m} s_j b_1^{e_{j1}} \cdots b_l^{e_{jl}}$, where $s_j$ is sign) is called a multibase representation of $n$ using the base $\mathcal{B}$. The integer $m$ is the length of the representation. Double base representation or double base number system (DBNS) [13, 14, 11] is a special case with $|\mathcal{B}| = 2$. In the current article we are particularly interested in multibase representations with $\mathcal{B} = \{2, 3, 5\}$.

Table 2: Number of multibase representation of small numbers using various bases.

| $n$ | $\mathcal{B} = \{2,\ 3\}$ | $\mathcal{B} = \{2,\ 5\}$ | $\mathcal{B} = \{2,\ 3,\ 5\}$ | $\mathcal{B} = \{2,\ 3,\ 5,\ 7\}$ |
|---|---|---|---|---|
| 10 | 5 | 3 | 8 | 10 |
| 20 | 12 | 5 | 32 | 48 |
| 50 | 72 | 18 | 489 | 1266 |
| 100 | 402 | 55 | 8425 | 43777 |
| 150 | 1296 | 119 | 63446 | 586862 |
| 200 | 3027 | 223 | 316557 | 4827147 |
| 300 | 11820 | 569 | 4016749 | 142196718 |

The double base number system is highly redundant. Also, these representations are very short in length. The multibase representations are even shorter and more redundant than the DBNS. The number of representations of $n$ grows very fast in the number of base elements. For example, 100 has 402 DBNS representation (base 2 and 3), 8425 representations using the bases 2, 3 and 5 and has 43777 representations using the bases 2, 3, 5, and 7 (considering only positive summands, i.e. $s_j = 1$). The number of representations for some small integers $n$ have been provided in Table 2.2. The multibase representation are very sparse too. One can represent a 160 bit integer using around 23 terms using $\mathcal{B} = \{2,\ 3\}$ and around 15 terms using $\mathcal{B} = \{2,\ 3,\ 5\}$ (see [13] for a result on length of DBNS representations).

In this article, unless otherwise stated, by a multibase representation of $n$ we mean a representation of the form

$$n = \Sigma_i s_i 2^{b_i} 3^{t_i} 5^{q_i}$$

where $s_i = \pm 1$. We will refer to terms of the form $2^a 3^b 5^c$ as 3-integers. A general multibase representation, although very short, is not suitable for a scalar multiplication algorithm. So we are interested in a special representation with restricted exponents.

**Definition:** A multibase representation $n = \Sigma_i s_i 2^{b_i} 3^{t_i} 5^{q_i}$ using the bases $\mathcal{B} = \{2,\ 3,\ 5\}$ is called a *step* multibase representation (SMBR) if the exponents $\{b_i\}$, $\{t_i\}$ and $\{p_i\}$ form three separate monotonic decreasing sequences.

Needless to mention, an integer $n$ has several SMBR, the simplest one being the binary representation. If $n$ is represented in SMBR, then we can write it using Horner's rule and an addition chain (like Double-base chain in [11]) for scalar multiplication can easily be developed.

### 2.2.1 Conversion to SMBR

An integer can be converted to a multibase representation using Greedy Algorithm:

**Greedy Algorithm**
```
while(k >0)
  let z be the largest number 2^b 3^t 5^p ≤ k;
  output (b,t,p)
  replace k by k − z
endwhile
```

The greedy algorithm produces near canonical (shortest) representations. We can implement the approximation step of the algorithm by using a three index array $T[0..max2, 0..max3, 0..max5]$, where the array element $T[i, j, k]$ is $2^i 3^j 5^k$ and $max2$ $max3$ $max5$ are maximum possible powers of 2, 3, and 5 respectively. Although, representations returned by Greedy are very sparse, they are not in SMBR. Algorithm mGreedy as described in below converts an integer into SMBR. mGreedy terminates because $k$ gets reduced in each iteration.

**Algorithm 1** mGreedy Algorithm for Conversion into SMBR

**Input:** $k$ a positive integer; $max2, max3, max5 > 0$, the largest allowed binary, ternary and quinary exponents and the array $T[0..max2;\ 0..max3;\ 0..max5]$.

**Output:** The sequence $(s_i, b_i, t_i, p_i)_{i>0}$ such that $k = \sum_{i=1}^{m} s_i\, 2^{b_i}\, 3^{t_i}\, 5^{p_i}$, with $b_1 \geq \ldots \geq b_m \geq 0$, $t_1 \geq \ldots \geq t_m \geq 0$, $p_1 \geq \ldots \geq p_m \geq 0$.

$s \leftarrow 1$
**while** $k > 0$ **do**
  for(b=0 to max2, t=0 to max3, p=0 to max5)
  $z = Tab[b, t, p]$, the best approximation of $k$
  **print** $(s, b, t, p)$
  $max2 \leftarrow b, \quad max3 \leftarrow t, \quad max5 \leftarrow p$
  **if** $k < z$ **then**
    $s \leftarrow -s$
  $k \leftarrow |k - z|$

### 2.2.2 Improving Performance of mGreedy

Algorithm mGreedy can be improved on two fronts: we can modify it to (1) obtain shorter representations and (2) run faster.

Looking at the outputs of mGreedy, one observes that the average length of the representations become higher because some of the representations are lengthy. In these lengthy representation, one observes that, one or two of the three exponents in the leading term is (are) very small. If a particular exponent in the leading term is small, it becomes zero in next very few term and the representation reduces to a DBNS representation there after. If two of the exponents become 0 very soon, then the representation even degenerates to a single base representation. We can overcome this shortcoming of mGreedy as described below.

Let $c_1$, $c_2$ and $c_3$ be three fractions less than 1. Let $x = 2^b 3^t 5^p$ be the best approximation for $k$ in some iteration. Then in the next iteration mGreedy replaces $max2$ by $b$, $max3$ by $t$ and $max5$ by $p$ and searches for the best approximation for $k - x$ in $T[0..max2; 0..max3; 0..max5]$ again. Instead of searching the array $T[\,]$, from $[0, 0, 0]$ to $[max2, max3, max5]$ we now restrict the lower limit to $[c_1 \times max2, c_2 \times max3, c_3 \times max5]$. This does not allow the any exponent to become very small at once and prevents the representation from being degenerate into a single or double base format. Also, the new algorithm runs faster as the search space in each iteration is smaller than the unrestricted version. With this restriction, the mGreedy (with $max2 = 160,\ max3 = 103,\ max5 = 70, c_1 = .4.c_2 = .3; c3 = .25$) returns an SMBR of average length less than 30 terms for integers of 160 bits (almost 20 % shorter).

Algorithms Greedy and mGreedy are search-based algorithms. They work by searching for the best approximation for the current value of $k$ in the table $T[\,]$. The table $T[\,]$ contains $max2 \times max3 \times max5$ entries. In each iteration, Greedy searches the whole static table $T[\ ]$. mGreedy dynamically changes the table size making it smaller from above (by lowering upper limits of the table indices). The modifications to mGreedy suggested in the last paragraph changes the table size in each iteration to provide the best running time.

Although, the choice $max2 = 160, max3 = 103$ and $max5 = 70$ returns very short representation, the table $T[\,]$ becomes very large. Construction of the table and table look-up make the conversion slow. But, it is observed that smaller choices like $max2 = 84, max3 = 36, max5 = 16$ speed up the conversion process dramatically and the the length of the representation goes up by 1 or 2 terms. Therefore, in all practical purposes, smaller values can be used. If the table $T[\ ]$ can be precomputed and stored, the conversion becomes almost instantaneous.

*However, one may not need the conversion algorithm at all. If the scalar $d$ for which $dP$ is to be computed, is to be randomly chosen at the time of scalar multiplication (as in many elliptic curve* *primitives), then one can generate a random integer directly in SMBR form to get rid of the conversion process altogether. Also, in situation, where d is known beforehand, (like elliptic curve digital signature generation), then the conversion can be done offline.* At least, in such applications, our scalar multiplication algorithm will perform much better than others.

#### 2.2.3 Other Useful Multibase Representations

We investigated on two more types of representations using three bases 2, 3 and 5: (1) SMBR with small anomalies and (2) SMBR with non-trivial digits.

**SMBR with small anomalies:** In this type of representation, the powers of 2, 3 and 5 form monotonic decreasing sequences except for some small deviations in some terms. Let $w_1$, $w_2$ and $w_3$ be the small permissible anomalies for the binary, ternary and quinary exponents respectively. Then a multibase representation $\Sigma_i s_i 2^{b_i} 3^{t_i} 5^{p_i}$ is a step representation with $(w_1, w_2, w_3)$-anomalies if $\{b_i\}$, $\{t_i\}$, and $\{p_i\}$ form monotonic decreasing sequences with a few exceptional terms for which $|b_i - b_{i-1}| \leq w_1$ or $|t_i - t_{i-1}| \leq w_2$ , $|p_i - p_{i-1}| \leq w_3$ hold good. Such representations can be used for scalar multiplication if the points $2^a 3^b 5^c$ for $0 \leq a \leq w_1, 0 \leq b \leq w_2, 0 \leq c \leq w_3$ can be precomputed and stored (see [15]). By choosing $w_i$'s to be as small as 2, it was seen that the length of a MBNS representation can be made quite shorter (24-25 terms).

**SMBR with non-trivial digits** So far we have considered representations $\Sigma_i s_i 2^{b_i} 3^{t_i} 5^{p_i}$, where $s_i \in \{1, 0, -1\}$. Let $\mathcal{D} = \{7, 11, 13, 17, 19, 23, 29, 31, \cdots\}$ be set of integers relatively prime to 2, 3, and 5. Let $\mathcal{D}_j$ be the set of first $j$ integers from $\mathcal{D}$. Let us consider the MBNS representation of the type $\Sigma_i s_i 2^{b_i} 3^{t_i} 5^{p_i}$ where $\pm s_i \in \mathcal{D}_j$. Such representations are also very short and can be used for scalar multiplication if the points $sP$ for $s \in \mathcal{D}_j$ can be precomputed (see [15]).

## 3 Efficient Formulas to Compute $5P$

In this section we present two new quintupling formulas for elliptic curve points, one for curves over prime fields of large characteristics and the other for curves over fields of characteristic 2. The proof of correctness of these formulas have been presented in Appendix A

### 3.1 Point Quintupling in Curves over Binary Fields

As the $[i]/[m]$ ratio in binary fields is known to be quite smaller in binary fields, affine elliptic curve group arithmetic is preferable. Hence we propose the new quintupling formula for such curves in affine coordinates. Let $P(x, y)$ be a point on an elliptic curve given by Equation (3) over a binary field. Let the 5-fold of $P$ be given by, $5P = (x_5, y_5)$. $x_5$ and $y_5$ can be computed as follow:

Let us define the following polynomials: $A = x^4 + x^3 + b$, $B = x^2(A + x^3)$, $C = A^3 + Bx^3$, $D = A^2(A^2 + B)$ Then,

$$\begin{aligned} x_5 &= x + \frac{xBD}{C^2} \\ y_5 &= y + x_5 + \frac{xAD^2}{C^3} + (x^2 + y)\frac{BD}{C^2} \end{aligned} \quad (4)$$

Given $P(x, y)$, let us check how much of computation is required to compute $5P$ using the above formula. Below we list the subexpressions (and costs) required to compute $x_5$ and $y_5$. 1. $A$ $(2[s] + 1[m])$, 2. $B$ $(1[m])$, 3. $C$ $(1[s] + 2[m])$, 4. $D$ $(1[m])$, 5. $1/C$ $(1[i])$, 6. $1/C^2$, $1[s]$ 7. $\frac{BD}{C^2}$ $(2[m])$ 8. $x_5$ $(1[m])$, 9. $1/C^3$ $(1[m])$ 10. $\frac{xAD^2}{C^3}$ $(3[m] + 1[s])$ 11. $y_5$ $(1[m])$ 10. Total: $1[i] + 5[s] + 13[m]$.

Let us consider the efficiency of the proposed formula. As this is the first point quintupling formula for curves over binary fields, we do not have any previous formula to compare performance. We can

Table 3: Cost of the quintupling formulas for various types of elliptic curves

| Curve | Condition | Cost |
|---|---|---|
| $y^2 = x^3 + ax + b$ | general | $10[s] + 15[m]$ |
| over | $a = -3$ | $8[s] + 15[m]$ |
| $K = F_p$ | after a QPL | $9[s] + 15[m]$ |
| $y^2 = x^3 + ax^2 + b$ over $K = F_{2^m}$ | general | $1[i] + 5[s] + 13[m]$ |

compute $5P$ as $2(2P)+P$. Using the generic ADD and DBL, it will cost 3 inversions. We can reduce one inversion by using composite formula double-and-add (DA) (see [6]). Using DA, computing $5P$ costs $2[i] + 11[m]$. If we compute $2P$ first and apply triple-and-add (TA)(see [6]) to $P$ and $2P$, $(3 \times P + 2P)$, then the cost would again involve 3 inversions, as TA requires 2 inversions. Using the repeated doubling formula proposed in [10] and ADD, it costs $2[i] + 8[m]$. So, the proposed formula is better than all these methods if $[i]/[m]$ ratio is 5 or more.

### 3.2 Point Quintupling in Curves over Large Prime Fields

In this section we present the point quintupling formula for elliptic curves over large prime fields. Due to space constraints we present the proof of this quintuple formula in the Appendix.

Let $P(X : Y : Z)$ be a point on the elliptic curve (2) over a prime field. Let $5P$ have coordinates $(X_5 : Y_5 : Z_5)$. Then $X_5$, $Y_5$ and $Z_5$ can be computed as follows:

$$\begin{aligned} X_5 &= XV^2 - 2YUW, \\ Y_5 &= Y(E^3(12VL^2 - V^2 - 16L^4) - 64TL^5), \\ Z_5 &= ZV, \end{aligned} \tag{5}$$

where, $T = 8Y^4$ $(2[s])$, $M = 3X^2 + aZ^4$ $(3[s]+1[m])$, $E = 12XY^2 - M^2$ $(1[s]+1[m])$, $L = ME - T$ $(1[m])$, $U = 4YL$ $(1[m])$, $V = 4TL - E^3$ $(1[s] + 2[m])$, $N = V - 4L^2$ $(1[s])$, $W = EN$ $(1[m])$.

The quantities in the braces are the cost of computing the corresponding subexpressions. Besides, computing $X_5$, $Y_5$ and $Z_5$ from these subexpressions require $1[s]+3[m]$, $4[m]+1[s]$ and $1[m]$. Hence, the cost of computing $5P$ by these formulas are $8[s] + 13[m] \approx 19.4[m]$ (if $Z = 1$) or $10[s] + 15[m] \approx 23[m]$ (if $Z \neq 1$).

This is the first explicit formula in literature to compute the multiplication-by-5 mapping for generic curves over arbitrary finite fields of characteristics $> 3$. Hence, we have no other formula to compare efficiency. Let us check the its efficiency vis-a-vis methods for computing $5P$. We can compute $5P$ by $2(2P) + P$ or by $3P + 2P$. We can compute $5P$ by $2(2P) + P$ with $9[s] + 17[m] \approx 24.2[m]$ (if $P$ is in affine) or $14[s] + 20[m] \approx 31.2[m]$ (if $P$ is in Jacobian). Using the formula $2P + 3P$, we can compute $5P$ with $22[m] + 12[s] \approx 31.6[m]$ or $26[m] + 16[s] \approx 38.8[m]$ according as $P$ is in affine or in Jacobian coordinates.

We will refer to the formula computing $5P$ as QPL. If $a = -3$, then $M = 3X^2 + aZ^4$ can be computed as $3(X+Z^2)(X-Z^2)$ with a cost of $1[s]+1[m]$ saving $2[s]$. Hence like DBL and TPL, QPL is also cheaper over special curves with $a = -3$. Also, just as in case of ($u$-)DBL and ($v$-)TPL, an algorithm $w-$QPL to compute $5^wP$ can be designed which will be cheaper than $u$ invocation of QPL. That is because for every invocation of QPL, one has to compute $Z_i = V_{i-1}Z_{i-1}$ and then compute $aZ_i^4 = aV_{i-1}^4 Z_{i-1}^4$. This step should normally take $1[m] + 2[s]$. But as $aZ_{i-1}^4$ and $V_{i-1}^2$ are already computed in the last QPL operation, by saving these subexpressions, one can compute $aZ_i^4 = aV_{i-1}^4 Z_{i-1}^4$ by just one $[m]$ and one $[s]$, saving one $[s]$. We have summarized the cost of QPL in Table 3.2.

## 4 The Scalar Multiplication Algorithms

The scalar multiplication algorithms used in this work are generalizations to 3 bases of the algorithms used in [11]. Without going into routine details, we add that the computation can be immunized against side-channel attacks using standard techniques proposed in the literature. Algorithm 2 for curves over binary fields uses the group operations like ADD, DBL, $w$-DBL, DA (double-and-add), TA (triple-and-add) for efficient computation.

In Algorithm 2 we describe the proposed scalar multiplication method to be used in conjunction with multibase representation for curves over binary fields. Note that Algorithm 2 requires $b_1$ doublings, $t_1$

**Algorithm 2** Scalar Multiplication for Curves over Fields of Even Characteristic.

**Input:** An integer $k = \sum_{i=1}^{m} s_i\, 2^{b_i} 3^{t_i} 5^{p_i}$, with $s_i \in \{-1, 1\}$, and such that $b_1 \geq b_2 \geq \ldots \geq b_m \geq 0$, $t_1 \geq t_2 \geq \ldots \geq t_m \geq 0$ and $p_1 \geq p_2 \geq \ldots \geq p_m \geq 0$ and a point $P \in E(F_q)$

**Output:** the point $kP \in E(F_q)$

$Z \leftarrow s_1 P$
**for** $i = 1, \ldots, m-1$ **do**
  $u \leftarrow b_i - b_{i+1}$
  $v \leftarrow t_i - t_{i+1}$
  $x \leftarrow p_i - p_{i+1}$
  **if** $u = 0$ **then**
    $Z \leftarrow (5^x Z)$
    **if** $v \neq 0$ **then**
      $Z \leftarrow 3(3^{v-1} Z) + s_{i+1} P$ //(TA used here)
    **else**
      $Z \leftarrow Z + s_{i+1} P$
  **else**
    $Z \leftarrow 5^x Z$
    $Z \leftarrow 3^v Z$
    $Z \leftarrow 2^{u-1} Z$
    $Z \leftarrow 2Z + s_{i+1} P$ //(DA used here)
**Return** $Z$

triplings and $p_1$ quintuplings. The number of additions is precisely the number of terms in the expansion of $k$ in which both the binary and ternary exponents are zero. Otherwise, the addition is always carried out by invoking a composite operation like double-and-add (DA) or triple-and-add (TA). Thus we need a very few number of additions for the computations.

We do not present the algorithm for scalar multiplication for curves over prime fields here. It is a generalization of the algorithm presented in [11] to the case of 3 bases.

## 5 Scalar Multiplication Results

A theoretical analysis of double (or higher) base number system is still alluding the researchers. Therefore, it is not possible to theoretically prove the efficiency of our scalar multiplication algorithms. We will present their average performance seen in applying them to huge number ($10^3$ to $10^6$) of randomly generated scalars.

We randomly generated 1 million 160-bit integers and stored them in a file. All the experiments were conducted by retrieving integers from this file, so that the same integers were used for all the experiments. This minimizes the bias in estimates due the use of different sets of integers for different scenarios. We present the results of our experiments in this section.

We present the experimental results in the Tables 4, 5, 6 below. In these tables, (i) *max2, max3, max5:* stand for maximum powers for 2, 3 and 5 allowed to occur in SMBR expansions, (ii) *alen:* means the average length of the SMBR expansions found. (iii) *cost:* means average cost of scalar multiplication for the randomly generated integers.

It is observed that choosing smaller values for $max2$, $max3$, $max5$ does not affect the cost drastically. Hence we recommend smaller values like $(85, 40, 20)$ to be used instead of $(160, 103, 70)$.

## 5.1 Scalar Multiplication without Precomputation

Let us first consider the cost of scalar multiplication using 3 bases without any precomputation. We conducted several experiments using various values of $max2$, $max3$ and $max5$ and also various values of $c_1$, $c_2$ and $c_3$. In Table 4, we have presented some of the results. Observe that for both kinds of curves, the best results were obtained when the highest possible powers of $max2$, $max3$ and $max5$, i.e. 160, 103 and 70, were chosen. However for these values the conversion from binary to MBNS is the slowest as the search space for the greedy algorithm is very big. Also, it was found that the maximum powers of 2, 3 and 5 observed in these expansions were much smaller. So, we choose smaller values for $max2$, $max3$ and $max5$ and observed that in these cases not only the conversion is very fast, but also the results are also quite competitive.

Table 4: Costs of elliptic curves Scalar Multiplication for 160-bit multipliers. The values of $c_1, c_2, c_3$ have been chosen as $0.4, 0.3$ and $0.25$ respectively.

| max2 | max3 | max5 | alen | $F_p$-Cost | $F_{2^m}$-Cost |
|---|---|---|---|---|---|
| 160 | 103 | 69 | 30.35 | 1673.89[m] | 96.67[i]+ 693.7[m] |
| 100 | 85 | 45 | 31.56 | 1679.19[m] | 101.6[i]+731.5[m] |
| 90 | 75 | 35 | 32.52 | 1691.73[m] | 108.8[i]+704.2[m] |
| 85 | 60 | 25 | 32.78 | 1694.16[m] | 112.6[i]+691.1[m] |
| 85 | 38 | 18 | 31.44 | 1671.34[m] | 113.0[i]+677.2[m] |

## 5.2 Scalar Multiplication with Precomputations

We conducted a huge number of experiments for scalar multiplication in 3-base expansion using precomputations. As mentioned earlier, we considered two kinds of precomputations: (1) SMBR with small anomalies and (2) MBNS with non-trivial digits. In former case, we choose $w_1$, $w_2$, $w_3$ to be 0 or 1, requiring storage of 1 to 7 precomputed points. The representations obtained in this case are very sparse (24.4 terms with a storage of 7 points). Some typical results have been presented in Table 5.

Also, we conducted experiments using SMBR representations with non-trivial larger coefficients. We allowed the SMBR to use various digit sets, starting from $\mathcal{D}_1 = \{7\}$ to $\mathcal{D}_8 = \{7, 11, 13, 17, 19, 23, 29, 31\}$. Use of $\mathcal{D}_i$ requires storage of $i$ points. It was found that the representations are even sparser than SMBR with anomalies. For example, with storage of 7 points, the multibase representation of a 160 bit integer could be 19.87 terms on average. Also, the computation scalar multiplication is quite cheaper than previous cases. Some typical results have been presented in Table 6.

## 5.3 Comparison

Let us compare the performance of the proposed scalar multiplication scheme to some of the schemes existing in literature. Some of the most recent scalar multiplication algorithms for general curves have been proposed in [6, 11, 15, 16, 22].

In [22], the authors have proposed an efficient scalar multiplication algorithm based on Montgomery's ladder. Their scheme does not require any precomputation and is secure against side-channel attacks.

Table 5: Costs of elliptic curves Scalar Multiplication for 160-bit multipliers in SMBR with small anomalies. The values of $c_1, c_2, c_3$ have been chosen as $0.4, 0.3$ and $0.25$ respectively.

| $max2$ | $max3$ | $max5$ | $w_1$ | $w_2$ | $w_3$ | #Points | alen | $F_p$-Cost | $F_{2^m}$-Cost |
|---|---|---|---|---|---|---|---|---|---|
| 84 | 36 | 16 | 1 | 0 | 0 | 1 | 31.01 | 1658.5[m] | 97.7[i] + 676.5[m] |
| 84 | 36 | 16 | 0 | 0 | 1 | 1 | 29.4 | 1616.4[m] | 90.7[i] + 681.3[m] |
| 84 | 36 | 16 | 1 | 0 | 1 | 3 | 28.2 | 1597.5[m] | 88.8[i] + 681.8[m] |
| 84 | 36 | 16 | 1 | 1 | 0 | 3 | 28.5 | 1615.1[m] | 87.5[i] + 681.3[m] |
| 84 | 36 | 16 | 0 | 1 | 1 | 3 | 25.9 | 1582.3[m] | 85.5[i] + 680.3[m] |
| 84 | 36 | 16 | 1 | 1 | 1 | 7 | 24.4 | 1571.4[m] | 83.8[i] + 680.4[m] |

Table 6: Costs scalar multiplication for 160-bit multipliers represented in three bases with larger digits. The values of $c_1, c_2, c_3$ have been chosen as $0.3, 0.3$ and $0.25$ respectively. Column # Points indicates the number of points to be precomputed and stored.

| $max2$ | $max3$ | $max5$ | #Points | alen | $F_p$-Cost | $F_{2^m}$-Cost |
|---|---|---|---|---|---|---|
| 84 | 36 | 16 | 1 | 25.67 | 1585.23[m] | 87.3[i] + 672.07[m] |
| 84 | 36 | 16 | 2 | 24.5 | 1547.22[m] | 83.18[i] + 666.26[m] |
| 84 | 36 | 16 | 3 | 22.86 | 1524.81[m] | 81.17[i] + 662.75[m] |
| 84 | 36 | 16 | 4 | 21.76 | 1510.29[m] | 79.25[i] + 661.34[m] |
| 84 | 36 | 16 | 5 | 21.14 | 1502.37[m] | 77.98[i] + 660.74[m] |

In [6], several formulas for efficient arithmetic have been proposed and a novel representation of the scalar in powers of 2 and 3 has been proposed, which is used for scalar multiplication. We refer to this scheme as binary/ternary scheme. In [11], the authors have proposed two schemes based on double base number systems and have obtained very good results. [15] has extended that work by considering the use of DBNS with precomputations. The authors have computed efficiency of their scheme and compared with several schemes with scalars of 200, 300, ... bits. In [16] the authors have proposed a new point tripling formula based on decomposition to 2 isogenies. They have pointed out efficiency of scalar multiplication schemes. We will compare our schemes with the methods proposed in these works.

Although our scheme in the current form is not secure against side-channel attacks [25, 26], side-channel resistance can be attained by some routine work. For example, attacks like simple power analysis attacks can be resisted by using some schemes like side-channel atomicity proposed in [5] which has almost no performance penalty. Also, attacks similar to differential power attacks can be resisted using curve randomization [23] or point randomization [9] countermeasures, which have a fixed cost (less than $50[m]$).

Let us first consider the scalar multiplication schemes for curves over prime fields without precomputation. Let the size of the elliptic curve group be of $2^{160}$-order. For such scenario, the scheme proposed in [22] requires $2638[m]$ to compute the scalar multiplication. The best scheme proposed in [11] requires $1863[m]$ using double base number system. Of the several schemes proposed in [16] using the efficient tripling formula proposed in same work, the best scheme for this scenario is sextuple and add method. This method requires $1957[m]$ for the special curves used by them and almost the same amount of computation for arbitrary curves. Our best scheme ($max2 = 160,\ max3 = 103,\ max5 = 70$) (see Table 4), takes only $1673[m]$ on average. Even for smaller values of $max2,\ max3,\ max5$, our schemes are clear winners.

If the system admits precomputation and storage of a few points, then we can resort to the two methods using SMBR with small anomalies or SMBR with non-trivial digit sets. In [16], the best performance reported for this scenario is $1623[m]$ with 8 points of storage with $3 - NAF_3$ method. As can be checked from Tables 5 and 6 the methods proposed in this work invariably perform better even with lesser storage.The best method proposed in [15] for 200 bit scalars is the DBChain method with

8 non-trivial coefficients ($\mathcal{S}_8$-DBChain). To compare our scheme with their method, we experimented with 200 bit scalars. With $max2 = 105$, $max3 = 60$, $max5 = 35$ and $c_1 = 0.4$, $c_2 = 0.3$, $c_3 = 0.25$, length of SMBR was seen to be 23.58. While length of the proposed DBNS representation was reported to be 25.9, Also, for 200 bit scalars, SMBR based scalar multiplication took 1917.26$[m]$ computation on average in comparison to 2019$[m]$ reported in [15].

For curves over binary fields, the proposed schemes perform even better. We summarize the comparisons in Table 7. In the table, binary and NAF refer to the traditional Binary and NAF based double-and-add algorithms. The DB-chain method refers to the one proposed in [11] and binary/ternary refer to a method proposed in [6]. The last column of the table is approximate cost obtained by the

Table 7: Average number of field operations using the binary, NAF, ternary/binary and DB-chain approaches for $n = 160$ bits, and $[i]/[m] = 8$.

| Algorithm | $[i]/[m]=8$ | | | Algorithm | $[i]/[m]=8$ | | |
|---|---|---|---|---|---|---|---|
| | $[i]$ | $[m]$ | $\approx [m]$ | | $[i]$ | $[m]$ | $\approx [m]$ |
| binary | 240 | 480 | 2400 | ternary/binary | 129 | 787 | 1819 |
| NAF | 213 | 426 | 2130 | DB-chain | 114 | 789 | 1701 |
| 3-NAF | 200 | 400 | 2000 | This work[1] | 97 | 693 | 1469 |
| 4-NAF | 192 | 384 | 1920 | This work[2] | 113 | 677 | 1581 |

number of inversion to the $[i]/[m]$-ratio and adding the number of multiplication to it. The method proposed in this work outperforms every known scheme even without any precomputations or storage for precomputed points. The scheme can be further improved using precomputations (see Tables 4 and 5).

# 6 Conclusion

In this work we have presented two efficient formulas for point quintupling in ECC over binary and prime fields. Also, we have proposed two scalar multiplication algorithms to take advantage of the proposed formulas. These algorithms use a multibase representation of the scalar using 2, 3 and 5 as bases. Also, we have dealt with the situation where the system admits precomputation and storage of some precomputed points. Our empirical results indicate that all the proposed schemes, with or without precomputation, outperform the corresponding best scalar multiplication schemes previously known.

# References


[1] R. M. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography.* CRC Press, 2005.

[2] R. M. Avanzi and F. Sica. Scalar Multiplication on Koblitz Curves using Double Bases. Tech Report. Available at `http://eprint.iacr.org/2006/067`

[3] I. F. Blake, G. Seroussi, and N.P. Smart. Elliptic Curves in Cryptography. Cambridge University Press, 1999.

[4] I. F. Blake, G. Seroussi, and N.P. Smart (Eds). Advances in Elliptic Curves Cryptography. Cambridge University Press, 2005.

[5] B. Chevalier-Mames, M. Ciet, and M. Joye. Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity. *IEEE Transactions on Computers*, 53(6):760–768, June 2004.

[6] M. Ciet, K. Lauter, M. Joye and P. L. Montgomery Trading inversions for multiplications in elliptic curve cryptography In *Designs, Codes and Cryptography*, 39(2):189-206, 2006.

[7] M. Ciet and F. Sica. An Analysis of Double Base Number Systems and a Sublinear Scalar Multiplication Algorithm. In E. Dawson, S. Vaudenay (Eds.): *Progress in Cryptology - Mycrypt 2005*, vol. 3715 of Lecture Notes in Computer Science, pp.171–182, Springer-Verlag, 2005.

[8] H. Cohen, A. Miyaji, and T. Ono. *Efficient Elliptic Curve Exponentiation Using Mixed coordinates*, In ASIACRYPT'98, LNCS 1514, pp. 51-65, Springer-Verlag, 1998.

[9] J.-S. Coron. Resistance against differential power analysis for elliptic curve cryptography. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 1999*, volume 1717 of *Lecture Notes in Computer Science*, pages 292–302. Springer-Verlag, 1999.

[10] R. Dahab and J. Lopez, An Improvement of Guajardo-Paar Method for Multiplication on non-supersingular Elliptic Curves. In Proceedings of the XVIII International Conference of the Chilean Computer Science Society (SCCC'98), IEEE CS Press, November 12-14, Antofagasta, Chile, pp. 91-95, 1998.

[11] V. Dimitrov, L. Imbert, and P. K. Mishra. Efficient and Secure Elliptic Curve Point Multiplication Using Double Base Chain. In B. Roy Ed., *Asiacrypt 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 59–79. Springer-Verlag, 2005.

[12] V. Dimitrov, K. U. Järvinen, M. J. Jacobson, W. F. Chan and Z. Huang. FPGA Implementation of Point Multiplication on Koblitz Curves Using Kleinian Integers In L. Goubin, M.Matsui (Eds.) Cryptographic Hardware and Embedded Systems - CHES 2006 Vol. 4249 of *Lecture Notes in Computer Science*, pp 445-459, Springer-Verlag, 2006.

[13] V. S. Dimitrov, G. A. Jullien, and W. C. Miller. An algorithm for modular exponentiation. *Information Processing Letters*, 66(3):155–159, May 1998.

[14] V. S. Dimitrov, G. A. Jullien, and W. C. Miller. Theory and applications of the double-base number system. *IEEE Transactions on Computers*, 48(10):1098–1106, Oct. 1999.

[15] C. Doche and L. Imbert. Extended Double-Base Number System with applications to Elliptic Curve Cryptography. Tech Report, Available at `http://eprint.iacr.org/2006/330`. Conference version to appear in *Indocrypt 2006.*

[16] C. Doche, T. Icart and D. Kohel Efficient Scalar Multiplication by Isogeny Decompositions, In Proceedings of *PKC 2006*, LNCS 3958, 191-206, Springer-Verlag, 2006.

[17] K. Fong, D. Hankerson, J. Lòpez, and A. Menezes. Field inversion and point halving revisited. *IEEE Transactions on Computers*, 53(8):1047–1059, Aug. 2004.

[18] J. Guajardo and C. Paar. Efficient Algorithms for Elliptic Curve Cryptosystems over binary fields. In *Advances in Cryptology – Crypto 1997*, volume 1965 of *Lecture Notes in Computer Science*, volume 1294, pages 342–356. Springer-Verlag, 1997.

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

[20] D. Hankerson, J. Lòpez Hernandez, and A. Menezes. Software implementation of elliptic curve cryptography over binary fields. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 1–24. Springer-Verlag, 2000.

[21] K. Itoh, M. Takenaka, N. Torii, S. Temma, and Y. Kurihara. Fast implementation of public-key cryptography on a DSP TMS320C6201. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES '99*, volume 1717 of *Lecture Notes in Computer Science*, pages 61 – 72. Springer-Verlag, 1999.

[22] T. Izu and T. Takagi. Fast elliptic curve multiplications resistant against side channel attacks. *IEICE Transactions Fundamentals*, E88-A(1):161–171, Jan. 2005.

[23] M. Joye and C. Tymen. Protections against differential analysis for elliptic curve cryptography – an algebraic approach. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 377 – 390. Springer-Verlag, 2001.

[24] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, Jan. 1987.

[25] P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In N. Koblitz, editor, *Advances in Cryptology - CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer-Verlag, Aug. 1996.

[26] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag, Aug. 1999.

[27] V. S. Miller. Uses of elliptic curves in cryptography. In H. C. Williams, editor, *Advances in Cryptology – CRYPTO '85*, volume 218 of *Lecture Notes in Computer Science*, pages 417–428. Springer-Verlag, 1986.

[28] R. Tijdeman. On the maximal distance between integers composed of small primes. *Compositio Mathematica*, 28:159–162, 1974.

# A Proof of Quintupling Formulas

We prove the correctness of the proposed quintupling formulas in this section.

## A.1 Proof of Quintupling in Curves over Binary Fields

For nonsupersingular curves over fields of characteristic 2, the division polynomials are given by

$$\begin{aligned}
\psi_1 &= 1 \\
\psi_2 &= x \\
\psi_3 &= x^4 + x^3 + a_6 \\
\psi_4 &= x^6 + a_6 x^2 \quad (= x^2(x^4 + a_6))
\end{aligned} \tag{6}$$

The higher degree division polynomials can be obtained by applying the the following recurrence relations:

$$\begin{aligned}
\psi_{2n+1} &= \psi_{n+2}\psi_n^3 - \psi_{n-1}\psi_{n+1}^3 \\
\psi_2\psi_{2n} &= \psi_{n+2}\psi_n\psi_{n-1}^2 - \psi_{n-2}\psi_n\psi_{n+1}^2
\end{aligned} \tag{7}$$

Using first of these recurrences with $n = 2$ and second one with $n = 3$, we get,

$$\begin{aligned} \psi_5 &= \psi_4\psi_2^3 - \psi_1\psi_3^3 \\ &= \psi_4 x^3 - \psi_3^3 \\ \psi_6 &= (\psi_5\psi_3\psi_2^2 - \psi_1\psi_3\psi_4^2)/\psi_2 \\ &= (\psi_5\psi_3 x^2 - \psi_3\psi_4^2)/x \end{aligned} \tag{8}$$

Using the above division polynomials, we can derive the expressions for 5-fold of a point $P(x, y)$ on the curve using the following relation with $n = 5$:

$$[n]P = (x + \frac{\psi_{n+1}\psi_{n-1}}{\psi_n^2}, y + \psi_2 on + \frac{\psi_{n+1}^2\psi_{n-2}}{\psi_2\psi_n^3} + h_4\frac{\psi_{n+1}\psi_{n-1}}{\psi_2\psi_n^2})$$

where,

$$\psi_2 on = x + \frac{\psi_{n+1}\psi_{n-1}}{\psi_n^2}$$

and

$$h_4 = (x^2 + y)$$

If the point $5P$ has the coordinates $(x_5, y_5)$, then it is an simple exercise to see that

$$\begin{aligned} x_5 &= x + \frac{\psi_6\psi_4}{\psi_5^2} \\ y_5 &= y + x_5 + \frac{\psi_6\psi_6'\psi_3}{\psi_5^3} + (x^2 + y)(\frac{\psi_6'\psi_4}{\psi_5^2}) \end{aligned} \tag{9}$$

where, $\psi_6' = \psi_6/x$. If we define polynomials as

$$\begin{aligned} A &= x^4 + x^3 + b \\ B &= x^2(A + x^3) \\ C &= A^3 + Bx^3 \\ D &= A^2(A^2 + B) \end{aligned} \tag{10}$$

then as can be checked, one has, $\psi_3 = A$, $\psi_4 = B$, $\psi_5 = C$ and $\psi_6 = xD$. Substituting these values in the Equations (9), we get the quintupling formula (4).

## A.2 Proof of quintupling formula for curves over prime fields

The division polynomials $\psi_n$, $0 \le n \le 4$, for elliptic curves (2) over prime fields are given by,

$$\begin{aligned} \psi_0 &= 0 \\ \psi_1 &= 1 \\ \psi_2 &= 2y \\ \psi_3 &= 3x^4 + 6ax^2 + 12bx - a^2 \\ \psi_4 &= 4y(x^6 + 5ax^4 + 20bx^3 - 5a^2x^2 - 4abx - 8b^2 - a^3) \end{aligned} \tag{11}$$

For $n \geq 5$, the division polynomials are given by the recursions,

$$\begin{aligned} 2y\psi_{2n} &= \psi_n(\psi_{n+2}\psi_{n-1}^2 - \psi_{n-2}\psi_{n+1}^2) \\ \psi_{2n+1} &= \psi_{n+2}\psi_n^3 - \psi_{n+1}^3\psi_{n-1} \end{aligned} \tag{12}$$

Using the recursion formulas given in Eq 12, we get,

$$\begin{aligned} \psi_5 &= \psi_4\psi_2^3 - psi_3^3\psi_1 \\ 2y\psi_6 &= \psi_3(\psi_5\psi_2^2 - \psi_1\psi_4^2) \\ \psi_7 &= \psi_5\psi_3^3 - \psi_4^3\psi_2 \end{aligned} \tag{13}$$

Let $P(x, y)$ be a point on the elliptic curve. We define the following polynomials. Let $\mathbf{t} = 8y^4$, $\mathbf{m} = 3x^2 + a$, $\mathbf{e} = 12xy^2 - \mathbf{m}^2$, $\mathbf{l} = \mathbf{me} - \mathbf{t}$, $\mathbf{u} = 4y\mathbf{l}$, $\mathbf{v} = 4\mathbf{tl} - \mathbf{e^3}$, $\mathbf{n} = \mathbf{v} - 4\mathbf{l}^2$ and $\mathbf{w} = \mathbf{en}$. We see that,

$$\begin{aligned} \mathbf{e} &= 12xy^2 - \mathbf{m}^2 \\ &= 12x(x^3 + ax + b) - (3x^2 + a)^2 \\ &= 3x^4 + 6ax^2 + 12bx - a^2 \\ &= \psi_3 \end{aligned} \tag{14}$$

Again, we have,

$$\begin{aligned} \mathbf{l} &= \mathbf{me} - \mathbf{t} \\ &= (3x^2 + a)(3x^4 + 6ax^2 + 12bx - a^2) - 8y^4 \\ &= (3x^2 + a)(3x^4 + 6ax^2 + 12bx - a^2) - 8(x^3 + ax + b)^2 \\ &= x^6 + 5ax64 + 20bx^3 - 5a^2x^2 - 4abx - 8b^2 - a^3 \end{aligned} \tag{15}$$

Comparing this with the expression for $\psi_4$, we get,

$$\begin{aligned} \psi_4 &= 4y\mathbf{l} \\ &= \mathbf{u} \end{aligned}$$

Also, it is simple to check that

$$\begin{aligned} \psi_5 &= \mathbf{v} \\ \psi_6 &= 2y\mathbf{en} \\ \psi_7 &= \mathbf{ve}^3 - 16\mathbf{tl}^3 \end{aligned} \tag{16}$$

We know that for any positive integer $n$, $nP$ is given by,

$$nP = (x - \frac{\psi_{n-1}\psi_{n+1}}{\psi_n^2}, \frac{\psi_{n+2}\psi_{n-1}^2 - \psi_{n-2}\psi_{n+1}^2}{4y\psi_n^3})$$

Table 8: Cost of computing $5P$ for curves over fields of odd characteristic.

| subexpression | cost | subexpression | cost |
|---|---|---|---|
| $\mathbf{t} = 8y^4$ | $2[s]$ | $\mathbf{m} = 3x^2 + a$ | $[s]$ |
| $\mathbf{e} = 12xy^2 - m^2$ | $[m] + [s]$ | $\mathbf{l} = \mathbf{me} - \mathbf{t}$ | $[m]$ |
| $\mathbf{u} = 4y\mathbf{l}$ | $[m]$ | $\mathbf{v} = 4\mathbf{tl} - \mathbf{e}^3$ | $2[m] + [s]$ |
| $\mathbf{n} = \mathbf{v} - 4\mathbf{l}^2$ | $[s]$ | $\mathbf{w} = \mathbf{en}$ | $[m]$ |
| $1/\mathbf{v}$ | $[i]$ | $1/\mathbf{v}^2,\ 1/\mathbf{v}^3$ | $[s] + [m]$ |
| $x_5 = x - (2y).\mathbf{u}.\mathbf{w}.1/\mathbf{v}^2$ | $3[m]$ | | |
| $y_5 = y.\{\mathbf{e}^3.(4\mathbf{vl^2} - \mathbf{n}^2) - 64(\mathbf{tl}.(\mathbf{l^2})^\mathbf{2})\}.1/\mathbf{v}^3$ | $5[m] + [s]$ | | |
| TOTAL | | $1[i] + 8[m] + 15[s]$ | |

Hence, if $5P = (x_5, y_5)$ then, substituting the values above we get,

$$\begin{aligned} x_5 &= x - \frac{2y\mathbf{uw}}{\mathbf{v}^2} \\ y_5 &= y\frac{\mathbf{e}^3(4\mathbf{vl^2} - \mathbf{n}^2) - 64\mathbf{tl}^5}{\mathbf{v}^3} \end{aligned} \tag{17}$$

Thus we obtain a formula for quintupling a point on the curve (2) in affine coordinates. In Table A.2 it is shown that cost of this formula is $1[i] + 8[s] + 15[m]$. Since inversions are too costly in a prime field, the affine formula does not look very attractive. Suppose, the point $P(x, y)$ is given in affine, but we compute the result in Jacobian coordinates, say $5P = (X_5 : Y_5 : Z_5)$, then its is simple to check that,

$$\begin{aligned} X_5 &= x\mathbf{v}^2 - 2y\mathbf{uw} \\ Y_5 &= y(\mathbf{e}^3(4\mathbf{vl^2} - \mathbf{n}^2) - 64\mathbf{tl}^5) \\ Z_5 &= \mathbf{v} \end{aligned} \tag{18}$$

In this case, one $[s]$ can be saved by using $4\mathbf{vl}^2 - \mathbf{n}^2 = 12\mathbf{vl}^2 - \mathbf{v}^2 - 16\mathbf{l}^4$. So, the cost of this computation (to compute $5P$ in Jacobian from $P$ in affine) is only $8[s] + 13[m] \approx 19.4[m]$. If we compute $5P$ by $2(2P) + P$, then the cost will be $13[s] + 14[m] \approx 24.4[m]$. The quintupling formula requires almost $6[m]$ less. Instead if one computes $5P$ as $2P + 3P$, then the cost will be: $2[m] + 4[s]$ for $2P$, $9[m] + 4[s]$ for $3P$ and $12[m] + 4[s]$ for '+', a total cost of $23[m] + 12[s] \approx 32.6[m]$. Again, the quintuple formula "wins big".

If the point $P$ is given in Jacobian coordinates, then the same approach can be used to compute 5 fold of $P$. If $P = (X : Y : Z)$ and $5P = (X_5 : Y_5 : Z_5)$, then

$$\begin{aligned} X_5 &= XV^2 - 2YUW \\ Y_5 &= Y(E^3(12VL^2 - V^2 - 16L^4) - 64TL^5) \\ Z_5 &= ZV \end{aligned} \tag{19}$$

where, $T = 8Y^4$, $M = 3X^2 + aZ^4$, $E = 12XY^2 - M^2$, $L = ME - T$, $U = 4YL$, $V = 4TL - E^3$, $N = V - 4L^2$, $W = EN$. Also, it can be checked that the cost of computation is almost the same except for $M = 3X^2 + aZ^4$ and $Z_5 = ZV$, which takes $2[s] + 2[m]$ extra. Hence the cost of computation for $\text{QPL}^{\mathcal{J}}$ is $15[m] + 10[s]$. It is routine to check that this is much cheaper than computing $5P$ as $2(2P) + P$ or $2P + 3P$ in Jacobian arithmetic.