Novel Precomputation Schemes for Elliptic Curve Cryptosystems

Patrick Longa, and Catherine Gebotys

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

Abstract. We present an innovative technique to add elliptic curve points with the form $P \pm Q$, and discuss its application to the generation of precomputed tables for the scalar multiplication. Our analysis shows that the proposed schemes offer, to the best of our knowledge, the lowest costs for precomputing points on both single and multiple scalar multiplication and for various elliptic curve forms, including the highly efficient Jacobi quartics and Edwards curves.

Keywords: Elliptic curve cryptosystem, scalar multiplication, multiple scalar multiplication, precomputation scheme, conjugate addition.

1 Introduction

In mid 80's, Miller and Koblitz independently proposed the use of elliptic curves for cryptographic purposes [16,8]. Since then, Elliptic Curve Cryptography (ECC) has gained increasing research and commercial interest.

Scalar multiplication, denoted by *kP*, where *k* is a scalar and *P* is a point on the elliptic curve, is the central operation of most elliptic curve cryptosystems. A plethora of methods exist in the literature to execute this operation efficiently, mainly exploiting some efficient representation of the scalar. For instance, the Non-Adjacent Form (NAF) is a standard representation with the fewest nonzero terms using digits from the set $\{-1, 0, 1\}$.

In some settings, however, it is required to compute a multiple scalar multiplication with the form *kP*+*lQ*, where *k* and *l* are scalars and *P* and *Q* are points on the curve. In this scenario, well-known methods are Interleaving [17] and the Joint Sparse Form (JSF) [19].

A practical strategy that reduces further the number of required additions at the expense of some extra memory is the use of precomputations. In this case, a table of points is built and stored in advance (precomputation stage) for later use during the execution of the scalar multiplication itself (evaluation stage). Although these window-based methods effectively reduce the number of nonzero terms in most representations, a potential drawback is the cost of computing such a table, which grows with the window size.

Thus, it is an important research effort to minimize the cost of the precomputation stage to reduce the total cost of scalar multiplication. Further, although improved elliptic curve

This manuscript is also available at: http://patricklonga.bravehost.com/publications.html.

shapes with faster explicit formulae are currently the focus of intense research [2,6], there is still a lack of analysis of precomputation schemes that are efficient for these settings.

In that direction, this work proposes efficient precomputation schemes and analyzes their performance on *three* relevant elliptic curve settings: standard elliptic curves using Jacobian coordinates, Jacobi quartics using an extended coordinate system [6,7] and Edwards curves using inverted Edwards coordinates [3].

The proposed schemes are based on the following simple idea: if *P*+*Q* has been computed for two distinct points *P*, *Q*, the subtraction of those points only requires a few additional field operations. In the remainder, we will refer to this operation, namely $P-Q\ (=P+(-Q))$, as "conjugate" addition. It will turn out that this operation will allow computing precomputed tables very efficiently. We apply the strategy of the conjugate addition to calculate tables of the form d_iP and $c_iP\pm d_iQ$, which are commonly found in most single and multiple scalar multiplication algorithms.

Further, our precomputation schemes are compared and analyzed for *three* possible cases, which are basically determined by the system used to represent points: projective coordinates, affine coordinates with restriction to *one* inversion per point, and affine coordinates (without restriction in the number of inversions). Our extensive analysis allows us to determine which case is the most efficient for a particular scenario and for determined *I*/*M* (field inversion/multiplication) ratios.

Our work is organized as follows. In Section 2, we detail some background about ECC over prime fields. Then, in Section 3 we describe our strategy to derive low-cost formulas for the conjugate addition in the different settings under study. In Section 4, we introduce the new schemes for precomputing points for tables with the forms d_iP and $c_iP\pm d_iQ$, and discuss their costs. In Section 5, we analyze and compare the performance of the proposed schemes with the previously most efficient methods. A discussion of some other applications of the strategy of the conjugate addition follows in Section 6. Some conclusions summarizing the contributions of this work are presented at the end.

2 Preliminaries

An elliptic curve *E* over a prime field $\mathbb{F}_p$ is defined by the short Weierstrass equation $E:\ y^2=x^3+ax+b$, where $a,b\in\mathbb{F}_p$ and $\Delta=4a^3+27b^2\neq 0$, and which will be referred in the remainder as the standard elliptic curve form. The points on the curve *E* and the identity element $\mathcal{O}$, known as the point at infinity, form an abelian group whose group law mainly consists of two basic operations: doubling (2*P*) and addition (*P*+*Q*) of points.

The main operations in most elliptic curve-based cryptosystems have the forms *kP* and *kP*+*lQ*, known as (single) scalar multiplication and multiple scalar multiplication.

Affine coordinates (referred to as $\mathcal{A}$ in the remainder) uses (x, y) to represent points. However, since this system requires field inversions, it is generally expensive over prime fields. When using efficient forms for the prime *p* (as recommended by [4]), it has been observed that the cost of inversion can be as high as $1I > 30M$. For example, benchmarks by [10] and [1] show *I/M* ratios between 30-40 and 50-100, respectively.

In efficient implementations, point representations with the form $(X : Y : Z)$, known as projective coordinates, were introduced to replace inversions. For example, an efficient case of this projective representation is given by Jacobian coordinates (referred to as $\mathcal{J}$), where each projective point $(X_i : Y_i : Z_i)$ corresponds to the affine point $(X_i / Z_i^2, Y_i / Z_i^3)$. In this case, equation E acquires the form $Y^2 = X^3 + aXZ^4 + bZ^6$, and the negative of an element $P = (X_i, Y_i, Z_i)$ is given by $-P = (X_i, -Y_i, Z_i)$.

In recent years, other curve forms with faster group laws have appeared in the literature. In this work, we focus on two of them: Jacobi quartics and Edwards curves, whose explicit formulas have been found to be particularly fast. We briefly describe both curve shapes in the following. Note that we consider that constant curve parameters are fixed to small values so that the cost of performing any operation with them is negligible.

Jacobi quartic curve. It is defined by the curve $y^2 = x^4 + 2ax^2 + 1$, where $a \in \mathbb{F}_p$, $a^2 \neq 1$. The projective curve is $Y^2 = X^4 + 2aX^2Z^2 + Z^4$, where a given projective point $(X_i : Y_i : Z_i)$ corresponds to the affine point $(X_i / Z_i, Y_i / Z_i^2)$. In this case, the negative of an element $P = (X_i, Y_i, Z_i)$ is represented by $-P = (-X_i, Y_i, Z_i)$. The most efficient formulae for these curves have been developed by Hisil et al. [6,7] using an extended coordinate system of the form $(X_i : Y_i : Z_i : X_i^2 : Z_i^2)$ that will be referred as $\mathcal{JQ}$.

Edwards curve. It is defined by the curve $x^2 + y^2 = 1 + dx^2y^2$, where $d \notin \{0,1\}$. In [2], the authors presented explicit formulas for point operations on this curve using standard projective coordinates. Later in [3], the same authors introduced a more efficient coordinate system, known as inverted Edwards coordinates (denoted by $\mathcal{IE}$), where each projective point $(X_i : Y_i : Z_i)$ corresponds to $(Z_i / X_i, Z_i / Y_i)$ in affine. In this case, the curve equation is given by $(X^2 + Y^2)Z^2 = X^2Y^2 + dZ^4$, where $XYZ \neq 0$, and the negative of a point $P = (X_i, Y_i, Z_i)$ is given by $-P = (-X_i, Y_i, Z_i)$.

In Table 1, we summarize the costs of the most efficient formulas in projective coordinates for the three curve forms under consideration. For complete details about formulas using $\mathcal{J}$ coordinates the reader is referred to [11]. Following the common practice in the literature, costs are expressed by the number of field multiplications (*M*) and squarings (*S*) that are required to perform certain operation, neglecting cheaper operations as field addition/subtraction (*A*) and multiplication/division by small constants.

Table 1 includes efficient operations using *mixed* coordinates, which are useful if input point(s) are represented in affine ($\mathcal{A}$) coordinates but the result is required in some projective system $\mathcal{P}$. Also, note that we have included efficient formulas exploiting pre-stored values. If, for instance, values Z_1^2, Z_1^3, Z_2^2 and Z_2^3 are available when computing an addition in $\mathcal{J}$ coordinates then we can saved up to $2M + 2S$. Similarly, in the case of Jacobi quartics it is possible to reduce the original cost of $7M + 4S$ of the addition formula to $7M + 3S$ (see [6] for complete details) by noting that $(X_i + Z_i)^2$ can be precomputed.

Finally, Table 1 also includes the highly efficient doubling-addition operation (DA) developed by [13], which involves the recurrent operation 2*P*+*Q* and is more efficient than performing a traditional doubling followed by an addition using $\mathcal{J}$.

Table 1. Costs of point operations in projective coordinates using Jacobian ($\mathcal{J}$), inverted Edwards ($\mathcal{IE}$) and extended Jacobi quartic ($\mathcal{JQ}$) coordinates.

Point Operation	Cost		
	Jacobian ($\mathcal{J}$, $a = -3$)	InvEdw ($\mathcal{IE}$)	JQuartic ($\mathcal{JQ}$)
Doubling (D), $2(\mathcal{P}) \to \mathcal{P}$	$3M + 5S$	$3M + 4S$	$2M + 5S$
Mixed doubling (mD), $2(\mathcal{A}) \to \mathcal{P}$	$1M + 5S$	$3M + 3S$	$7S$
Tripling (T), $3(\mathcal{P}) \to \mathcal{P}$	$7M + 7S$	$9M + 4S$	$8M + 4S$
Mixed tripling (mT), $3(\mathcal{A}) \to \mathcal{P}$	$5M + 7S$	$7M + 3S$	$5M + 6S$
Addition with stored values, $\mathcal{P} + \mathcal{P} \to \mathcal{P}$	$10M + 4S$ / $9M + 3S$	-	$7M + 3S$
Addition (A), $\mathcal{P} + \mathcal{P} \to \mathcal{P}$	$11M + 5S$	$9M + 1S$	$7M + 4S$
Mixed addition (mA), $\mathcal{P} + \mathcal{A} \to \mathcal{P}$	$7M + 4S$	$8M + 1S$	$6M + 3S$
Mixed addition (mmA), $\mathcal{A} + \mathcal{A} \to \mathcal{P}$	$4M + 2S$	$7M$	$4M + 3S$
DA with stored values, $2(\mathcal{P}) + (\mathcal{P}) \to \mathcal{P}$	$13M + 8S$	-	-
DA, $2(\mathcal{P}) + (\mathcal{P}) \to \mathcal{P}$	$14M + 9S$	-	-
Mixed DA, $2(\mathcal{P}) + (\mathcal{A}) \to \mathcal{P}$	$11M + 7S$	-	-

$\mathcal{P}$: projective coordinates ($\mathcal{J}$, $\mathcal{JQ}$ or $\mathcal{IE}$ coordinates).

3 Our Strategy: Conjugate Addition

Our strategy to yield efficient precomputation schemes is based on the similarities between adding and subtracting two points. Basically, if the addition *P*+*Q* takes place, then it is expected that, when subtracting the same points (i.e., *P*−*Q*), most of the intermediate field operations are identical simply because *P*−*Q* = *P*+(−*Q*) and the negative of a point only involves the change of at most one of the coordinate values in the point representation, as described in the previous section.

Let us illustrate the latter with the point addition formula using $\mathcal{J}$. Let $P = (X_1, Y_1, Z_1)$ and $Q = (X_2, Y_2, Z_2)$ be two points on an elliptic curve *E*. If the addition $P + Q = (X_3, Y_3, Z_3)$ is performed using [12, formula (15)] as follows

$$X_3 = \alpha^2 - (4\beta^3 + 8Z_2^2 X_1 \beta^2)\,, \quad Y_3 = \alpha(Z_2^2 X_1 \beta^2 - X_3) - Z_2^3 Y_1 \beta^3\,, \quad Z_3 = \theta\beta\,, \tag{1}$$

where $\alpha = 2(Z_1^3 Y_2 - Z_2^3 Y_1)$, $\beta = Z_1^2 X_2 - Z_2^2 X_1$ and $\theta = (Z_1 + Z_2)^2 - Z_1^2 - Z_2^2$, then $P - Q$ can be computed as $P + (-Q) = (X_1, Y_1, Z_1) + (X_2, -Y_2, Z_2) = (X_4, Y_4, Z_4)$ reusing the partial values $(4\beta^3 + 8Z_2^2 X_1 \beta^2)$, $Z_2^2 X_1 \beta^2$, $-Z_2^3 Y_1 \beta^3$, Z_3, $Z_1^3 Y_2$ and $Z_2^3 Y_1$. The latter can be performed with the following formula for the conjugate addition

$$X_4 = \gamma^2 - (4\beta^3 + 8Z_2^2 X_1 \beta^2)\,,\ \ Y_4 = \gamma(Z_2^2 X_1 \beta^2 - X_4) - Z_2^3 Y_1 \beta^3\,,\ \ Z_4 = Z_3\,, \tag{2}$$

where $\gamma = -2(Z_1^3 Y_2 + Z_2^3 Y_1)$. Note that the cost of the conjugate addition (2) using $\mathcal{J}$ is only 1*M*+1*S*, which is significantly less than the cost of a general addition (1) (i.e., 11*M*+5*S*). If we also consider other usually neglected operations, then the cost drops from $11M{+}5S{+}9A{+}2\,(\times 2)+1\,(\times 4)$ to only $1M{+}1S{+}4A{+}1\,(\times 2)$.

It may seem that performing this conjugate operation would involve several extra registers to store partial values temporarily. However, memory requirements can be minimized by performing *P*+*Q* and *P*−*Q* in “parallel”. For instance, a possible execution sequence for computing $P \pm Q$ using formulas (1) and (2) would be as follows:

INPUT: $T_1 \leftarrow X_1$, $T_2 \leftarrow Y_1$, $T_3 \leftarrow Z_1$, $T_4 \leftarrow X_2$, $T_5 \leftarrow Y_2$, $T_6 \leftarrow Z_2$
OUTPUT: $T_1 \leftarrow X_3$, $T_2 \leftarrow Y_3$, $T_3 \leftarrow Z_3$, $T_4 \leftarrow X_4$, $T_5 \leftarrow Y_4$

1. $T_7 = T_3^2$	$\{Z_1^2\}$	13. $T_7 = T_4 - T_8$	$\{\beta\}$	25. $T_1 = T_6^2$	$\{\alpha^2\}$
2. $T_4 = T_4 \times T_7$	$\{Z_1^2 X_2\}$	14. $T_3 = T_3 \times T_7$	$\{Z_3 = Z_4\}$	26. $T_1 = T_1 - T_4$	$\{X_3\}$
3. $T_8 = T_3 \times T_7$	$\{Z_1^3\}$	15. $T_6 = T_7^2$	$\{\beta^2\}$	27. $T_7 = T_2 \times T_7$	$\{Z_2^3 Y_1 \beta^3\}$
4. $T_5 = T_5 \times T_8$	$\{Z_1^3 Y_2\}$	16. $T_7 = T_6 \times T_7$	$\{\beta^3\}$	28. $T_2 = T_8 - T_1$	$\{Z_2^2 X_1 \beta^2 - X_3\}$
5. $T_8 = T_6^2$	$\{Z_2^2\}$	17. $T_8 = T_6 \times T_8$	$\{Z_2^2 X_1 \beta^2\}$	29. $T_2 = T_2 \times T_6$	$\{\alpha(Z_2^2 X_1 \beta^2 - X_3)\}$
6. $T_7 = T_7 + T_8$	$\{Z_1^2 + Z_2^2\}$	18. $T_4 = 2T_8$	$\{2Z_2^2 X_1 \beta^2\}$	30. $T_2 = T_2 - T_7$	$\{Y_3\}$
7. $T_3 = T_3 + T_6$	$\{Z_1 + Z_2\}$	19. $T_4 = T_4 + T_7$	$\{\beta^3 + 2Z_2^2 X_1 \beta^2\}$	31. $T_6 = T_5^2$	$\{\gamma^2\}$
8. $T_3 = T_3^2$	$\{(Z_1 + Z_2)^2\}$	20. $T_4 = 4T_4$	$\{4\beta^3 + 8Z_2^2 X_1 \beta^2\}$	32. $T_4 = T_6 - T_4$	$\{X_4\}$
9. $T_3 = T_3 - T_7$	$\{\theta\}$	21. $T_6 = T_5 - T_2$	$\{Z_1^3 Y_2 - Z_2^3 Y_1\}$	33. $T_8 = T_8 - T_4$	$\{Z_2^2 X_1 \beta^2 - X_4\}$
10. $T_6 = T_6 \times T_8$	$\{Z_2^3\}$	22. $T_6 = 2T_6$	$\{\alpha\}$	34. $T_8 = T_5 \times T_8$	$\{\gamma(Z_2^2 X_1 \beta^2 - X_4)\}$
11. $T_2 = T_2 \times T_6$	$\{Z_2^3 Y_1\}$	23. $T_5 = -T_5 - T_2$	$\{-(Z_1^3 Y_2 + Z_2^3 Y_1)\}$	35. $T_5 = T_8 - T_7$	$\{Y_4\}$
12. $T_8 = T_1 \times T_8$	$\{Z_2^2 X_1\}$	24. $T_5 = 2T_5$	$\{\gamma\}$		

The previous execution requires 8 registers only (including temporary registers and registers storing input coordinates). It is easy to verify that the memory requirement is the same as that of the addition formula alone. Thus, executing the conjugate addition does not increase the memory requirements in this case.

We have derived the conjugate addition formulas in projective coordinates (i.e., $\mathcal{J}$, $\mathcal{JQ}$ and $\mathcal{IE}$ coord.), and also in affine for the three curves of interest. The costs of these new formulas are summarized in Table 2. We have also included the costs of the traditional addition operations that accompany the execution of our formulas. Note that, in some cases, the traditional operations have been modified slightly so that the cost of the pair addition/conjugate addition is minimized. Refer to Appendices A-C for complete details.

As it can be seen in Table 2, the new conjugate formulas introduce significant cost reductions in comparison to traditional operations (see Table 1). In the following section, we take advantage of the latter to develop low-cost precomputation schemes.

Table 2. Costs of new conjugate additions for standard, Edwards and Jacobi quartic curves using projective ($\mathcal{J}$, $\mathcal{IE}$ and $\mathcal{JQ}$) and affine coordinates.

Point Operation	Cost		
	Standard curve	Edwards curve	Jacobi quartic
Conjugate addition (A') , $\mathcal{P}-\mathcal{P}\to\mathcal{P}$	$1M+1S$	$4M$	$2M+1S$
Addition (A), $\mathcal{P}+\mathcal{P}\to\mathcal{P}$	$11M+5S$	$9M+1S$	$7M+3S$
Conjugate mixed addition (mA'), $\mathcal{P}-\mathcal{A}\to\mathcal{P}$	$1M+1S$	$4M$	$2M+1S$
Mixed addition (mA), $\mathcal{P}+\mathcal{A}\to\mathcal{P}$	$7M+4S$	$8M+1S$	$6M+3S$
Conjugate mixed addition (mmA'), $\mathcal{A}-\mathcal{A}\to\mathcal{P}$	$1M+1S$	$3M$	$1M+1S$
Mixed addition (mmA), $\mathcal{A}+\mathcal{A}\to\mathcal{P}$	$4M+2S$	$8M$	$5M+3S$
Conjugate addition (A'), $\mathcal{A}-\mathcal{A}\to\mathcal{A}$	$2M+1S$	$4M$	$3M$
Addition (A), $\mathcal{A}+\mathcal{A}\to\mathcal{A}$	$1I+2M+1S$	$1I+9M+1S$	$1I+7M+4S$

4 New Precomputation Method for Scalar Multiplication

In this Section, we apply the concept of conjugate addition to derive highly efficient precomputation schemes first for tables of the form d_iP and then for tables of the form $c_iP \pm d_iQ$, where $c_i, d_i \in \{1,3,5,\ldots,m\}$. We consider *three* scenarios: precomputed points are left in projective coordinates (referred to as *case 1*), precomputed points are calculated in projective coordinates and then converted to affine using one inversion (referred to as *case 2*), and precomputed points are computed and left in affine (referred to as *case 3*).

4.1 Precomputation Scheme for Table of the form d_iP , $d_i \in \{1,3,5,\ldots,m\}$

Well-known methods to compute scalar multiplication using a precomputed table with points d_iP, where $d_i \in D_i = \{1,3,5,...,m\}$, are Window-*w* NAF (*w*NAF) and Fractional Window-*w* NAF (Frac-*w*NAF), in the case of single scalar multiplication, and the Interleaving method, in the case of multiple scalar multiplication.

We propose a recursive scheme that first tries to reach a "strategic" point and then applies efficiently the conjugate addition technique described in Section 3. In the following, we define as "strategic" to those points that can be efficiently computed and from which is possible to calculate the maximum possible number of precomputed points at the lowest cost. The steps of our scheme are detailed in the following.

Step 1: Computation of precomputed points. This is the main body of our scheme, and is presented in Algorithm 4.1. In this step, points can be computed in projective coordinates using operations from Table 1 (*case 1*), or directly in $\mathcal{A}$ (*case 3*). If projective points are to be converted to $\mathcal{A}$ (*case 2*), then Step 2 should be executed right after.

Basically, Algorithm 4.1 first reaches certain "strategic" point and then computes all

Algorithm 4.1 Computation of precomputed points

INPUT: a point P in affine ($\mathcal{A}$) coordinates, and

an odd value $m \geq 7$ to build a table of the form d_iP, where $d_i \in \{1, 3, 5, 7, \ldots, m\}$

OUTPUT: the precomputed table $T = \{T_0 = P, T_1 = 3P, \ldots, T_{(m-1)/2} = mP\}$ in projective or affine coord.

```
1. $r = 3,\ l = 1,\ i = 2,\ n = v = 0$
2. $T_0 = P,\ T_1 = rP$
3. $R = T_1$
4. While $n < (m-3)/2$
   4.1. If $m < 2r$
        4.1.1. While $n < (m-3)/2$
                   $T_s = R + T_l$
                   $n = n + 1,\ l = l + 1,\ s = s + 1$
   4.2. Else
        4.2.1. $t = 2^v$
        4.2.2. $v = v + 1$
        4.2.3. $R = 2R$
        4.2.4. $r = 2r,\ j = t - 1,\ first = 1$
        4.2.5. While $j \geq 0$ do
                   $T_i = R - T_j,\ n = n + 1$
                   If $first = 1$, then $l = j + 1,\ s = r - i,\ first = 0$
                   $i = i + 1$
                   If $m \geq r + 2j + 1$, then
                       $T_{(r+2j)/2} = R + T_j,\ n = n + 1$
                       If $T_j = T_0$, then $i = i + 1$
                   $j = j - 1$
5. Return $T = \{T_0, T_1, \ldots, T_{(m-1)/2}\}$
```

the points that are close to it by efficiently performing additions and conjugate additions. The "strategic" points proposed in our scheme have the form $P_{i+1} = 2P_i$, for $i \in \mathbb{Z} \geq 0$ and $P_0 = 3P$ (i.e., $6P$, $12P$, $24P$, and so on), which are computed using a combination of one tripling (performed at the beginning, Step 2) and a sequence of doublings (Step 4.2.3). Note that there is a minimum number of close points that makes worthwhile the computation of the following "strategic" point. If that minimum is not fulfilled (evaluation in Step 4.1) then the algorithm calculates the remaining points from the previous "strategic" point (loop beginning in Step 4.1.1). The value of such a minimum depends on the particular costs of point operations. For $\mathcal{J}$, $\mathcal{JQ}$ and $\mathcal{IE}$, we have determined that the lowest cost is achieved if the next "strategic" point is computed always that the *m* value is greater or equal to such a "strategic" point (condition in Step 4.1).

Let us illustrate the proposed scheme with the following example.

Example 1. If m = 13, Alg. 4.1 computes the first points as $P \to 3P \to 6P$, where $6P$ is the first "strategic" point. From this, 5*P* and 7*P* (*close* points) are calculated by adding $6P+(-P)$ and $6P+P$. Note that the latter operation can be calculated with a conjugate addition, requiring a very low number of operations. Then, Alg. 4.1 calculates the following "strategic" point (since $m > 12$) by doubling $6P \rightarrow 12P$, and finally computes close points 9*P*, 11*P* and 13*P* by performing $12P+(-3P)$, $12P+(-P)$ and $12P+P$, respectively. Note again that the latter operation is also a low-cost conjugate addition.

In Appendix D, we have sketched the derivation of points for tables with different values *m*. Note that the method described does not include cases $m = 3, 5$. Computing the table for $m = 3$ only requires *one* mixed tripling. For case $m = 5$, $\mathcal{JQ}$ and $\mathcal{J}$ coordinates, it is more efficient to compute points by performing $P \rightarrow 2P \rightarrow 4P$, and then obtaining 3*P* and 5*P* with an addition/conjugate addition pair (i.e., $4P+(-P)$ and $4P+P$). For case $\mathcal{IE}$, we suggest to compute the table following the sequence $P \rightarrow 2P \rightarrow 3P \rightarrow 5P$.

In the following, we describe the procedure to convert points to $\mathcal{A}$ for *case 2*.

Step 2: Conversion to affine coordinates (if required). If mixed addition (or mixed DA) is significantly more efficient than general addition (or general DA) in a given setting, then it would be convenient to express the precomputed table in $\mathcal{A}$.

It is known that conversion to $\mathcal{A}$ can be achieved by calculating $(X_i / Z_i^2, Y_i / Z_i^3)$, $(X_i / Z_i, Y_i / Z_i^2)$ and $(Z_i / X_i, Z_i / Y_i)$ for $\mathcal{J}$, $\mathcal{JQ}$ and $\mathcal{IE}$ coordinates, respectively.

For each setting, calculation of denominators (denoted by u_i) can be efficiently carried out by using the well-known Montgomery's method of simultaneous inversion. In this way, the number of expensive inversions can be limited to only *one*.

First, we compute the inverse $U = (u_1 u_2 \ldots u_n)^{-1}$, where u_i are all distinct denominators of the expressions above from all the non-trivial points in the table {3*P*, 5*P*, …, *mP*}. For $\mathcal{J}$ and $\mathcal{JQ}$, the number of such denominators is reduced to only $n = (m-1)/2 - c$, where *c* is the number of points computed via conjugate addition, since points computed with addition/conjugate addition pairs share the same coordinate *Z*. For $\mathcal{IE}$, $n = m-1$ as each point has *two* distinct denominators, namely X_i and Y_i.

Then, individual denominators u_i are recovered from *U*, and the results multiplied to their corresponding numerator following the conversion expressions.

As it can be seen the use of conjugate additions reduces the cost of the Montgomery's method for the cases of $\mathcal{J}$ and $\mathcal{JQ}$ coordinates. Following our explanation above, it can be easily verified that one saves $3M + 1S$ per point computed with a conjugate addition.

Cost Analysis. The cost of the scheme proposed mainly depends on the value *m* in the precomputed table and the curve form selected. We list in Table 3 the costs in terms of number of operations for various values *m*. As operations in $\mathcal{A}$ coordinates are relatively expensive in Jacobi quartic and Edwards curves (see Table 2), we only show the performance of case 3 in the setting of the standard curve. Note that, in this case, listed point operations using *mixed* coordinates should be read as standard operations (e.g., for *m* = 7, case 3, the proposed method requires 1T+1D+1A+1A').

Table 3. Costs of the proposed scheme for case 1 in projective coord. using Jacobian ($\mathcal{J}$) and extended Jacobi quartic ($\mathcal{JQ}$) coord.; case 2 using one inversion; and case 3 in affine ($\mathcal{A}$) coord.

m	Point Operations	Case 1		Case 2		Case 3
		$\mathcal{J}$	$\mathcal{JQ}$	$\mathcal{J}$	$\mathcal{JQ}$	Standard curve
7	1mT + 1D + 1mA+ 1mA’	$17M + 17S$	$15M + 17S$	$1I + 28M + 18S$	$1I + 24M + 20S$	$3I + 13M + 7S$
9	1mT + 1D + 1mA+ 1mA’ + 1A	$27M + 21S$	$22M + 20S$	$1I + 43M + 22S$	$1I + 36M + 25S$	$4I + 15M + 8S$
11	1mT + 1D + 1mA+ 1mA’ + 2A	$37M + 25S$	$29M + 23S$	$1I + 59M + 27S$	$1I + 48M + 30S$	$5I + 17M + 9S$
13	1mT + 2D + 2mA+ 2mA’ + 1A	$39M + 31S$	$32M + 30S$	$1I + 63M + 32S$	$1I + 53M + 35S$	$6I + 21M + 11S$
15	1mT + 2D + 2mA+ 2mA’ + 1A + 1A’	$40M + 32S$	$34M + 32S$	$1I + 67M + 33S$	$1I + 57M + 37S$	$6I + 23M + 12S$

Depending on the curve form selected, some additional considerations are necessary. In the case of the standard curve using $\mathcal{J}$, if the evaluation stage uses the efficient addition with *two* stored values, then values Z_i^2 and Z_i^3 should be computed during the precomputation stage. Naively, the latter would require $(1M + 1S)(m-1)/2$. However, some additional cost reductions are possible. First, the initial tripling computes the required values for point $3P$ (i.e., Z_{3P}^2 and Z_{3P}^3) without requiring extra operations. Also, one squaring can be saved every time a doubling is performed to get any “strategic” point since values Z_i^2 are cached. Moreover, it is easy to see that addition and conjugate addition formulas share the same coordinate Z (see Appendix A). Hence, we only require $1M$+$1S$ to get Z_i^2 and Z_i^3 for two points computed with an addition/conjugate addition pair. Finally, when performing additions using a “strategic” point Q, its values Z_Q^2 and Z_Q^3 are calculated in the first mixed addition, say $Q+P = (X_Q, Y_Q, Z_Q) + (x_1, y_1)$. Thus, following general additions of the form $Q+P = (X_Q, Y_Q, Z_Q) + (X_R, Y_R, Z_R)$ can be executed using an addition with *four* stored values.

Similarly, in $\mathcal{JQ}$, if the evaluation stage uses the efficient addition with the stored value $(X_i + Z_i)^2$, then these values should be included in the precomputation cost. We now describe a few optimizations to minimize this cost. First, one squaring can be saved every time a doubling is performed to get any “strategic” point by noting that $(X_i + Z_i)^2$ can be cached from a previous mixed tripling or mixed addition. Also, when performing additions with a “strategic” point Q, the value $(X_Q + Z_Q)^2$ is calculated in the first mixed addition. Then, following general additions with the same point Q save one extra squaring.

The costs including the savings described above are detailed in Table 3, case 1. For the case where points are converted to $\mathcal{A}$ (case 2), we have to also consider the cost of performing the Montgomery’ simultaneous inversion method (Step 2). The cost of the latter in $\mathcal{J}$ and $\mathcal{JQ}$ is given by $\text{Cost}_{\mathcal{J}\to\mathcal{A}} = 1I + (6L-3)\,M + (L)\,S$ and $\text{Cost}_{\mathcal{JQ}\to\mathcal{A}} = 1I + (5L-3)\,M + (2L)\,S$, respectively, where $L = (m-1)/2$ and m odd ≥ 5. However, as described in Section 4.1, Step 2, the proposed scheme allows for some extra savings since points obtained through an addition/conjugate addition pair share the same coordinate Z. The reduced costs including these savings are given by

$$\text{Cost}_{proposed\,\mathcal{J}\to\mathcal{A}} = 1I + (6L - 3c - 3)\,M + (L - c)\,S\ , \tag{6}$$

$$\text{Cost}_{proposed\ \mathcal{JQ}\to\mathcal{A}} = 1I + (5L - 3c - 3)\,M + (2L - c)\,S\ , \tag{7}$$

respectively, where *c* denotes the number of points obtained using a conjugate addition. In the case of $\mathcal{IE}$, the cost of the Montgomery's method is as follows

$$\text{Cost}_{\mathcal{IE}\to\mathcal{A}} = 1I + (6L + \lceil (L-2)/L \rceil - 1)\,M\ , \tag{8}$$

The total costs including conversion to $\mathcal{A}$ are given in Table 3, case 2. Note that in this case addition operations with stored values do not apply.

4.2 Precomputation Scheme for Table of the form $c_iP \pm d_iQ$, $c_i, d_i \in \{1,3,5,\ldots,m\}$

This scenario mainly applies to methods for computing multiple scalar multiplications such as those based on JSF. In this case, the application of our strategy of conjugate additions is straightforward since precomputed points have the form $c_iP \pm d_iQ$, where $c_i, d_i \in \{1,3,5,\ldots,m\}$, and each two points $cP \pm dQ$ can be computed with an addition/conjugate addition pair.

In the following, we analyze the cost involved when precomputing points for the specific case of the efficient JSF-based algorithm by Kuang et al. [9]. Extension of the method to similar table forms easily follows.

Cost Analysis. If both *P* and *Q* are unknown before the scalar multiplication is executed, the points $3P$, $3Q$, $P \pm Q$, $3P \pm Q$, $P \pm 3Q$, $3P \pm 3Q$ required by the method by [9] need to be computed on the fly. The latter costs 2mT+2mmA+4mA+2A for case 1 (when points are left in projective coordinates). With the strategy of conjugate additions, that cost reduces to 2mT+1mmA+1mmA'+2mA+2mA'+1A+1A'. Note that the advantage increases for case 2 as our approach allows saving some operations during conversion to $\mathcal{A}$, as shown in Section 4.1.

In Table 4, we show the cost performance of the proposed scheme for the considered curve shapes. Note that, in the setting of $\mathcal{J}$ and $\mathcal{JQ}$, we use again the efficient addition formulas with stored values and, following the same procedure described in Section 4.1, we have minimized the impact of the computation of those partial values for case 1. For case 2 the conversion to $\mathcal{A}$ coord. is similar to that of the scheme from Section 4.1 and, hence, it follows the costs given by (6), (7) and (8) for $\mathcal{J}$, $\mathcal{JQ}$ and $\mathcal{IE}$, respectively.

Table 4. Costs of the proposed scheme for case 1, in projective coord. using Jacobian ($\mathcal{J}$) and extended Jacobi quartic ($\mathcal{JQ}$) coord.; case 2 using one inversion; and case 3, in affine ($\mathcal{A}$) coord.

Curve form	Point Operations	Case 1	Case 2	Case 3
Jacobi Quartic ($\mathcal{JQ}$)	2mT + 1mmA + 1mmA' + 2mA+ 2mA' + 1A + 1A'	$41M + 35S$	$1I + 76M + 44S$	-
Edwards ($\mathcal{IE}$)		$47M + 24S$	$1I + 107M + 24S$	-
Standard ($\mathcal{J}$)		$42M + 32S$	$1I + 84M + 35S$	$6I + 30M + 16S$

5 Performance Comparison

In this section, we analyze and compare the proposed approach with the most efficient precomputation schemes available in the literature.

In the case of $\mathcal{J}$, [13] recently proposed a highly efficient scheme, which has been shown to achieve the lowest cost among methods using only *one* inversion (case 2). The cost of this method (referred to as LM method in the remainder) is given by ($1M = 0.8S$)

$$\text{Cost}_{\text{LM, case 2}} = 1I + (9L)M + (2L+6)S = 1I + (10.6L + 4.8)M \ , \tag{9}$$

We now derive the cost of the LM method for case 1 using the traditional chain $P \rightarrow 2P \rightarrow 3P \rightarrow 5P \rightarrow \ldots \rightarrow mP$ and the special addition due to [15], but avoiding the final conversion to $\mathcal{A}$. This involves one mixed doubling and L special additions that cost $5M$+$2S$. Also, the use of additions with pre-stored values during the evaluation stage requires precalculating values Z_i^2 and Z_i^3 with a cost of $L(1M$+$1S)$. Then the total cost is

$$\text{Cost}_{\text{LM, case 1}} = (6L+1)M + (3L+5)S = (8.4L+5)M \ , \tag{10}$$

Regarding, $\mathcal{IE}$ and $\mathcal{JQ}$, we could not find literature related to precomputation schemes in these settings. Hence, we have analyzed the performance of the straightforward implementation using the same chain above. The costs for case 1 are as follows

$$\text{Cost}_{\mathcal{IE},\text{ case 1}} = (9L+2)M + (1L+3)S = (9.8L+4.4)M \ , \tag{11}$$

$$\text{Cost}_{\mathcal{JQ},\text{ case 1}} = (7L-1)M + (3L+7)S = (9.4L+4.6)M \ , \tag{12}$$

for $\mathcal{IE}$ and $\mathcal{JQ}$ coordinates, respectively. For case 2, the costs are given by

$$\text{Cost}_{\mathcal{IE},\text{ case 2}} = 1I + (15.8L + \lceil (L-2)/L \rceil + 3.4)M \ , \tag{13}$$

$$\text{Cost}_{\mathcal{JQ},\text{ case 2}} = 1I + (12L-4)M + (5L+7)S = 1I + (16L+1.6)M \ , \tag{14}$$

In Table 5, we compare the costs of the described schemes to that of the proposed scheme from Section 4.1 (see Table 3 and Appendix D) for different windows w. As it can be seen, the new approach outperforms every other method in cases 1 and 2 for both $\mathcal{IE}$ and $\mathcal{JQ}$. Note that the advantage increases with the window size. For instance, if $1I = 30M$, the cost reduction can be as high as 25% ($w = 6$, $\mathcal{JQ}$).

In the case of standard curves, the LM scheme still achieves the highest performance. Nevertheless, for case 1, the modified LM scheme (10) and the new approach achieve similar performance.

In settings where inversions are not so expensive (low *I/M* ratios), it could be attractive the implementation of case 3. In this case, Table 6 shows the performance of the traditional approach and the proposed method on a standard curve form. Also, the *I/M* ratios for which the traditional, the proposed and the LM method achieve the lowest cost are shown at the bottom of the table. As it can be observed, the LM method offers the highest performance for a wide range of high *I/M* ratios on a standard curve, whereas the

Table 5. Costs of different schemes in projective (case 1) and affine coord. (case 2); $1M = 0.8S$.

Case	Method	Curve form	$w = 3$	$w = 4$	$w = 5$	$w = 6$
case 1	Proposed scheme	$\mathcal{JQ}$	$10.6M$	$28.6M$	$59.6M$	$116.6M$
	Method (12)	$\mathcal{JQ}$	–	$32.8M$	$70.4M$	$145.6M$
	Proposed scheme	$\mathcal{IE}$	$9.4M$	$28.4M$	$61.2M$	$121.6M$
	Method (11)	$\mathcal{IE}$	–	$33.8M$	$73.0M$	$151.4M$
	Proposed scheme	$\mathcal{J}$	$10.6M$	$30.6M$	$65.6M$	$130.6M$
	LM Method (10)	$\mathcal{J}$	–	$30.2M$	$63.8M$	$131.0M$
case 2	Proposed scheme	$\mathcal{JQ}$	–	$1I + 40.0M$	$1I + 86.6M$	$1I + 173.6M$
	Method (14)	$\mathcal{JQ}$	–	$1I + 49.6M$	$1I + 113.6M$	$1I + 241.6M$
	Proposed scheme	$\mathcal{IE}$	–	$1I + 46.4M$	$1I + 103.2M$	$1I + 211.6M$
	Method (13)	$\mathcal{IE}$	–	$1I + 46.8M$	$1I + 102.0M$	$1I + 212.4M$
	Proposed scheme	$\mathcal{J}$	$1I + 10.2M$	$1I + 42.4M$	$1I + 93.4M$	$1I + 194.0M$
	LM Method (9), [13]	$\mathcal{J}$	–	$1I + 36.6M$	$1I + 79.0M$	$1I + 163.8M$

Table 6. Costs of different schemes in affine coordinates (case 3) and *I/M* ranges for which each scheme achieves the lowest cost on the standard curve form; $1M = 0.8S$

Method	$w = 4$	$w = 5$	$w = 6$
Proposed scheme	$3I + 18.6M$	$6I + 32.6M$	$11I + 57.8M$
Traditional	$4I + 11.2M$	$8I + 22.4M$	$16I + 44.8M$
I/M range (LM Method (9), [13])	$I > 9M$	$I > 9.3M$	$I > 10.6M$
I/M range (Proposed, case 3)	$7.4M < I < 9M$	$5.1M < I < 9.3M$	$2.6M < I < 10.6M$
I/M range (Traditional)	$I < 7.4M$	$I < 5.1M$	$I < 2.6M$

proposed method is convenient for low/intermediate values *I/M*.

Let us now compare the performance of our scheme for cases 1 and 2, to determine the best scheme for each scenario. For this analysis, we should also consider the scalar multiplication cost since different point operations apply to different cases. Note that we only analyze the performance on Edwards and Jacobi quartic curves, as these are the settings where our method has been shown to attain the lowest costs (see Table 5).

Let us consider the standard *w*NAF method. In this case, the cost of a scalar multiplication is approximately

$$\left[n\mathrm{D} + \left(\frac{(2^{w-2}-1)(n-1)}{2^{w-2}(w+1)} \right) \mathrm{A} + \left(\frac{(n-1)}{2^{w-2}(w+1)} \right) \mathrm{mA} \right] + \mathrm{Cost}_{\mathrm{Proposed,\,case\,1}}\,, \quad \left[n\mathrm{D} + \left(\frac{n-1}{w+1} \right) \mathrm{mA} \right] + \mathrm{Cost}_{\mathrm{Proposed,\,case\,2}}\,,$$

for cases 1 and 2, respectively. Table 7 shows the performance of the scalar multiplication including the costs of the precomputation schemes proposed in this work, cases 1 and 2. As it can be seen, case 1 achieves the best performance for most common *I/M* ratios for *n*

= 160 bits. However, this advantage reduces if n = 512 bits. Ultimately, the particular *I/M* ratio of a given implementation would define which case is the most efficient one.

From another viewpoint, case 1 would be largely preferred if there is no restriction in the number of precomputations.

Table 7. Costs of scalar multiplication using *w*NAF and the proposed precomputation schemes (cases 1 and 2); and *I/M* range for which case 1 achieves the lowest cost on $\mathcal{JQ}$; $1M = 0.8S$

Method	n = 160 bits		n = 512 bits		
	$w = 4$	$w = 5$	$w = 4$	$w = 5$	$w = 6$
Proposed, case 1	$1279.6M$	$1265.4M$	$4035.7M$	$3921.5M$	$3867.4M$
Proposed, case 2	$1I + 1267.1M$	$1I + 1269.2M$	$1I + 3970.5M$	$1I + 3874.0M$	$1I + 3858.8M$
I/M range (case 1)	$I > 12.5M$	$I > 0M$	$I > 65.2M$	$I > 47.5M$	$I > 8.6M$

Finally, we analyze the performance of the proposed scheme for the table $c_i P \pm d_i Q$. In this case, a multiple scalar multiplication [9] costs approximately $[n\text{D}+0.3083(n-1)\text{A}+0.0617(n-1)\text{mA}] + \text{Cost}_{\text{Proposed, case 1}}$ and $[n\text{D}+0.37(n-1)\text{mA}] + \text{Cost}_{\text{Proposed, case 2}}$ for cases 1 and 2, respectively. The latter can be reduced in the case of $\mathcal{J}$ coordinates if we consider the efficient DA operation [13]. The cost in this case can be expressed as follows

$$\left[(0.63n + 0.37)\text{D} + 0.3083(n-1)\text{DA} + 0.0617(n-1)\text{mDA}\right] + \text{Cost}_{\text{Proposed, case 1}}, \tag{15}$$

$$\left[(0.63n + 0.37)\text{D} + 0.37(n-1)\text{mDA}\right] + \text{Cost}_{\text{Proposed, case 2}}. \tag{16}$$

Table 8 shows the performance of the scalar multiplication including the costs of our precomputation scheme, cases 1 and 2.

Table 8. Costs of multiple scalar multiplication using the JSF$_3$ method [9] and the proposed precomputation schemes (cases 1 and 2); and *I/M* ranges for which case 1 achieves the lowest cost on Jacobi quartic ($\mathcal{JQ}$), inverted Edwards ($\mathcal{IE}$) and Jacobian ($\mathcal{J}$) coordinates; $1M = 0.8S$

Method	n = 160 bits			n = 512 bits		
	$\mathcal{JQ}$	$\mathcal{IE}$	$\mathcal{J}$	$\mathcal{JQ}$	$\mathcal{IE}$	$\mathcal{J}$
Proposed, case 1	$1572.2M$	$1624.9M$	$1889.6M$	$4886.7M$	$5062.0M$	$5840.2M$
Proposed, case 2	$1I + 1565.4M$	$1I + 1635.9M$	$1I + 1796.8M$	$1I + 4771.4M$	$1I + 4964.4M$	$1I + 5511.1M$
I/M range (case 1)	$I > 6.8M$	$I > 0M$	$I > 92.8M$	$I > 115.3M$	$I > 97.6M$	$I > 329.1M$

In Table 8, case 1 again achieves the best performance for most common *I/M* ratios for n = 160 bits. However, if n = 512 bits, the range of *I/M* ratios for which case 2 is more efficient increases. Also, note that case 2 appears to be the best choice for $\mathcal{J}$ coordinates for a wide range of *I/M* ratios.

As reference, the costs for $\mathcal{JQ}$ and $\mathcal{J}$ using a traditional chain for precomputation would be $1598M$ or $1I + 1612M$, and $1922M$ or $1I + 1849M$, respectively ($n = 160$ bits).

6 Other Applications

We have discussed the application of the strategy of the conjugate addition to build efficient precomputation tables with the forms d_iP and $c_iP \pm d_iQ$. However, this technique can be easily applied to other table forms such as the one required by the generalized JSF [18], which requires the computation of $(3^k - 1)/2 - k$ non-trivial points. For instance, for $k = 3$ scalars, the previous algorithm requires the precomputation of $P \pm Q$, $P \pm R$, $Q \pm R$, $P + Q \pm R$, $P - Q \pm R$, which costs about 10 general additions. With our strategy, the latter is reduced to only 5 addition/conjugate addition pairs (case 1). Note that the advantage grows exponentially with the number of scalars.

Other obvious application is the extension of our strategy to other settings such as binary fields. Let us illustrate the latter with the addition formula due to [14] and later refined by [5]. The cost of adding two points $P + Q$ with the latter formula takes $13M + 4S$. Then, if we need the value $P - Q$ right after, we can store most partial results from the original addition and obtain the previous value with a cost of only $5M$ by noticing that $-Q = (X_2, X_2Z_2 + Y_2, Z_2)$ in Lopez-Dahab coordinates. Note that the partial term $Y_2Z_1^2$ from the original formula is replaced by $-Y_2Z_1^2 = (X_2Z_2 + Y_2)Z_1^2 = X_2Z_2Z_1^2 + Y_2Z_1^2$, which only cost *one* extra multiplication. Straightforward generalizations of this technique (and also of the proposed precomputation schemes) can be applied to other coordinate systems and/or elliptic curve forms.

7 Conclusions

We have introduced an innovative technique based on conjugate additions that can be efficiently exploited to reduce costs in a scalar multiplication. The relevant formulas on three different settings (namely, standard, Jacobi quartic and Edwards curves) over prime fields have been derived and shown to attain significant cost reductions in comparison with traditional formulae. In particular, we have proposed novel precomputation schemes based on this technique. Our analysis shows that the new schemes are especially attractive on the highly efficient Jacobi quartic and Edwards curves, enabling even faster implementations. Finally, we have also discussed other applications of the introduced strategy to binary fields and other precomputation tables.

Acknowledgments. This research was supported in part by grants from the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Ontario Centres of Excellence (OCE). Also, after finishing this work, we became aware of a similar idea proposed by Michael Scott during a presentation in ECC2008. He rediscovered the idea of exploiting similarities between $P + Q$ and $P - Q$ and suggested a slightly different sequence to precompute points.

References

1. M. Brown, D. Hankerson, J. Lopez and A. Menezes, “Software Implementation of the NIST elliptic curves over prime fields,” in *Progress in Cryptology CT-RSA 2001*, LNCS Vol. 2020, pp. 250-265, Springer, 2001.
2. D. Bernstein and T. Lange, “Faster Addition and Doubling on Elliptic Curves,” in *Advances of Cryptology - Asiacrypt 2007*, LNCS Vol. 4833, pp. 29–50, Springer, 2007.
3. D. Bernstein and T. Lange, “Inverted Edwards Coordinates,” in *Applied Algebra, Algebraic Algorithms, and Error Correcting Codes Symposium (AAECC 2007)*, 2007.
4. FIPS PUB 186-2. Digital Signature Standard (DSS). *National Institute of Standards and Technology (NIST)*, 2000.
5. A. Higuchi and N. Takagi, “A Fast Addition Algorithm for Elliptic Curve Arithmetic in GF(2^n) using Projective Coordinates,” in *Information Processing Letters*, Vol. 76, No 3, pp. 101-103, 2000.
6. H. Hisil, K. Wong, G. Carter and E. Dawson, “Faster Group Operations on Elliptic Curves,” in *Cryptology ePrint Archive*, Report 2007/441, 2007.
7. H. Hisil, K. Wong, G. Carter and E. Dawson, "An Intersection Form for Jacobi-Quartic Curves", *personal communication*, 2008.
8. N. Koblitz, “Elliptic Curve Cryptosystems,” in *Mathematics of Computation*, Vol. 48, pp. 203–209, 1987.
9. B. Kuang, Y. Zhu and Y. Zhang, “An Improved Algorithm for uP+vQ using JSF3,” in *ACNS 2004*, LNCS Vol. 3089, pp. 467-478, Springer, 2004.
10. C.H. Lim and H.S. Hwang, “Fast implementation of Elliptic Curve Arithmetic in GF(p^n),” in *Public Key Cryptography (PKC’00)*, LNCS Vol. 1751, pp. 405-421, Springer, 2000.
11. P. Longa, “ECC Point Arithmetic Formulae (EPAF),” available online at http://patricklonga.bravehost.com/jacobian.html.
12. P. Longa and A. Miri, “Fast and Flexible Elliptic Curve Point Arithmetic over Prime Fields,” in *IEEE Transactions on Computers*, Vol. 57, No 3, pp. 289-302, 2008.
13. P. Longa and A. Miri, “New Composite Operations and Precomputation Scheme for Elliptic Curve Cryptosystems over Prime Fields,” in *Public Key Cryptography (PKC’08)*, LNCS Vol. 4939, pp. 229-247, Springer, 2008.
14. J. López and R. Dahab, “Improved Algorithms for Elliptic Curve Arithmetic in GF(2^n),” *Technical Report IC-98-39*, Relatorio Técnico, 1998.
15. N. Meloni, “New Point Addition Formulae for ECC Applications,” in *International Workshop on the Arithmetic of Finite Fields (WAIFI 2007)*, LNCS Vol. 4547, pp. 189-201, 2007.
16. V. Miller, “Use of Elliptic Curves in Cryptography,” in *Advances in Cryptology - Crypto’85*, LNCS Vol. 218, pp. 417-426, Springer, 1986.
17. B. Möller, “Algorithms for Multi-exponentiation,” in *Selected Areas in Cryptography (SAC 2001)*, LNCS Vol. 2259, pp. 165-180, Springer, 2001.
18. J. Proos, “Joint Sparse Forms and Generating Zero Columns when Combing,” *Technical Report CORR 2003-23*, University of Waterloo, 2003.
19. J. Solinas, “Low-Weight Binary Representations for Pairs of Integers,” *Technical Report CORR 2001-41*, University of Waterloo, 2001.

A Conjugate (Mixed) Addition in Jacobian Coordinates

Let $P=(X_1,Y_1,Z_1)$ and $Q=(X_2,Y_2,Z_2)$ be two points on an elliptic curve *E*. If the general addition $P+Q$ is performed using [12, formula (15)] and the partial values $(4\beta^3+8Z_2^2X_1\beta^2)$, $Z_2^2X_1\beta^2$, $-Z_2^3Y_1\beta^3$, Z_3, $Z_1^3Y_2$ and $Z_2^3Y_1$ are temporarily stored, the conjugate addition $P-Q=P+(-Q)=(X_1,Y_1,Z_1)+(X_2,-Y_2,Z_2)=(X_4,Y_4,Z_4)$ can be performed with the following:

$$X_4=\gamma^2-(4\beta^3+8Z_2^2X_1\beta^2)\,,\ Y_4=\gamma(Z_2^2X_1\beta^2-X_4)-Z_2^3Y_1\beta^3\,,\ Z_4=Z_3\,, \quad (17)$$

where $\gamma=-2(Z_1^3Y_2+Z_2^3Y_1)$. This formula only requires $1M + 1S + 4A + 1\,(\times 2)$.

In the case of mixed addition, let $P=(X_1,Y_1,Z_1)$ and $Q=(x_2,y_2)$ be two points on an elliptic curve *E*. If the mixed addition $P+Q$ is performed using [12, formula (16)] and the partial values $(4\beta^3+8X_1\beta^2)$, $4X_1\beta^2$, $-8Y_1\beta^3$, Z_3 and $Z_1^3y_2$ are temporarily stored, the conjugate mixed addition $P-Q=P+(-Q)=(X_1,Y_1,Z_1)+(x_2,-y_2)=(X_4,Y_4,Z_4)$ can be performed as follows:

$$X_4=\gamma^2-(4\beta^3+8X_1\beta^2)\,,\ Y_4=\gamma(4X_1\beta^2-X_4)-8Y_1\beta^3\,,\ Z_4=Z_3\,, \quad (18)$$

where $\gamma=-2(Z_1^3y_2+Y_1)$. This formula only costs $1M + 1S + 4A + 1\,(\times 2)$.

B Conjugate (Mixed) Addition in $\mathcal{JQ}$ Coordinates

Let $P=(X_1,Y_1,Z_1,X_1^2,Z_1^2)$ and $Q=(X_2,Y_2,Z_2,X_1^2,Z_1^2)$ be two points on a Jacobi quartic curve. If the addition $P+Q$ is performed using the following formula due to [6]

$$X_3=(\alpha+2Y_1)(\beta+2Y_2)-\alpha\beta-4Y_1Y_2\,,\ Z_3=4Z_1^2Z_2^2-4X_1^2X_2^2\,,\ X_3^2=(X_3)^2\,,\ Z_3^2=(Z_3)^2\,, \quad (19)$$
$$Y_3=(4X_1^2X_2^2+4Z_1^2Z_2^2+2\alpha\beta)[4(X_1^2+Z_1^2)(X_2^2+Z_2^2)+a\,\alpha\beta+4Y_1Y_2]-16(X_3^2+Z_3^2)\,,$$

where $\alpha=(X_1+Z_1)^2-(X_1^2+Z_1^2)$, $\beta=(X_2+Z_2)^2-(X_2^2+Z_2^2)$, and the partial values β, $(\alpha+2Y_1)$, $2Y_2$, $\alpha\beta$, $-4Y_1Y_2$, $(4X_1^2X_2^2+4Z_1^2Z_2^2)$, $2\alpha\beta$, $4(X_1^2+Z_1^2)(X_2^2+Z_2^2)+4Y_1Y_2$, $a\,\alpha\beta$, Z_3 and Z_3^2 are temporarily stored, then the conjugate addition $P-Q=P+(-Q)=(X_1,Y_1,Z_1,X_1^2,Z_1^2)+(-X_2,Y_2,Z_2,X_2^2,Z_2^2)=(X_4,Y_4,Z_4)$ can be performed with only $2M + 1S + 7A + 1\,(\times 16)$ as follows :

$$X_4=(\alpha+2Y_1)(-\beta+2Y_2)+\alpha\beta-4Y_1Y_2\,,\ Z_4=4Z_1^2Z_2^2-4X_1^2X_2^2=Z_3\,,\ X_4^2=(X_4)^2\,,\ Z_4^2=Z_3^2\,, \quad (20)$$
$$Y_4=(4X_1^2X_2^2+4Z_1^2Z_2^2-2\alpha\beta)[4(X_1^2+Z_1^2)(X_2^2+Z_2^2)-a\,\alpha\beta+4Y_1Y_2]-16(X_4^2+Z_4^2)\,,$$

In the case of mixed addition, let $P=(X_1,Y_1,Z_1,X_1^2,Z_1^2)$ and $Q=(x_2,y_2,x_2^2)$ be two points on a Jacobi quartic curve. If the mixed addition $P+Q$ is performed using the following formula due to [6]

$$X_3 = (\alpha + 2Y_1)(x_2 + y_2) - \alpha x_2 - 2Y_1 y_2\,,\ Z_3 = 2(Z_1^2 - X_1^2 x_2^2)\,,\ X_3^2 = (X_3)^2\,,\ Z_3^2 = (Z_3)^2\,, \quad (21)$$
$$Y_3 = 2((X_1^2 x_2^2 + Z_1^2 + \alpha x_2)[2(X_1^2 + Z_1^2)(x_2^2 + 1) + a\,\alpha x_2 + 2Y_1 y_2] - 2(X_3^2 + Z_3^2))\,,$$

where $\alpha = (X_1 + Z_1)^2 - (X_1^2 + Z_1^2)$, and the partial values $(\alpha + 2Y_1)$, αx_2, $-2Y_1 y_2$, $(X_1^2 x_2^2 + Z_1^2)$, $(2(X_1^2 + Z_1^2)(x_2^2 + 1) + 2Y_1 y_2)$, $a\,\alpha x_2$, Z_3 and Z_3^2 are temporarily stored, then the conjugate mixed addition $P - Q = P + (-Q) = (X_1, Y_1, Z_1, X_1^2, Z_1^2) + (-x_2, y_2, x_2^2) = (X_4, Y_4, Z_4)$ can be performed with $2M + 1S + 7A + 2\,(\times 2)$ as follows:

$$X_4 = (\alpha + 2Y_1)(-x_2 + y_2) + \alpha x_2 - 2Y_1 y_2\,,\ Z_4 = 2(Z_1^2 - X_1^2 x_2^2) = Z_3\,,\ X_4^2 = (X_4)^2\,,\ Z_4^2 = Z_3^2\,, \quad (22)$$
$$Y_4 = 2((X_1^2 x_2^2 + Z_1^2 - \alpha x_2)[2(X_1^2 + Z_1^2)(x_2^2 + 1) - a\,\alpha x_2 + 2Y_1 y_2] - 2(X_4^2 + Z_4^2))\,.$$

C Conjugate (Mixed) Addition in $\mathcal{IE}$ Coordinates

Let $P = (X_1, Y_1, Z_1)$ and $Q = (X_2, Y_2, Z_2)$ be two points on Inverted Edwards coordinates. If the general addition $P + Q$ is performed using the following formula due to [3] (note that some terms have been rearranged to save a few field additions):

$$X_3 = [X_1 X_2 Y_1 Y_2 + d(Z_1 Z_2)^2](X_1 X_2 - Y_1 Y_2)\,,\ Y_3 = [X_1 X_2 Y_1 Y_2 - d(Z_1 Z_2)^2](X_1 Y_2 + X_2 Y_1)\,,$$
$$Z_3 = Z_1 Z_2 (X_1 X_2 - Y_1 Y_2)(X_1 Y_2 + X_2 Y_1)\,, \quad (23)$$

and the partial values $[X_1 X_2 Y_1 Y_2 + d(Z_1 Z_2)^2]$, $X_1 X_2$, $Y_1 Y_2$, $[X_1 X_2 Y_1 Y_2 - d(Z_1 Z_2)^2]$, $X_1 Y_2$, $X_2 Y_1$ and $Z_1 Z_2$ are temporarily stored, then the conjugate addition $P - Q = P + (-Q) = (X_1, Y_1, Z_1) + (-X_2, Y_2, Z_2) = (X_4, Y_4, Z_4)$ can be performed with the following (with a cost of only $4M + 2A$):

$$X_4 = [X_1 X_2 Y_1 Y_2 - d(Z_1 Z_2)^2](X_1 X_2 + Y_1 Y_2)\,,\ Y_4 = -[X_1 X_2 Y_1 Y_2 + d(Z_1 Z_2)^2](X_1 Y_2 - X_2 Y_1)\,,$$
$$Z_4 = -Z_1 Z_2 (X_1 X_2 + Y_1 Y_2)(X_1 Y_2 - X_2 Y_1)\,, \quad (24)$$

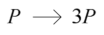
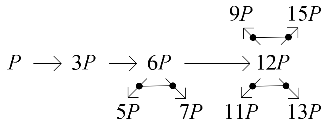
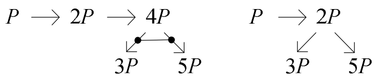
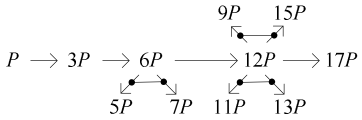
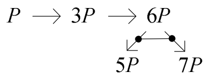
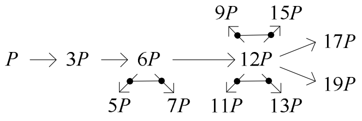
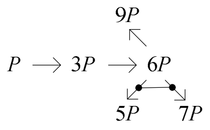
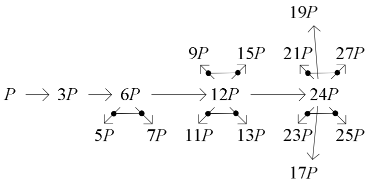
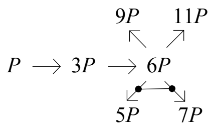
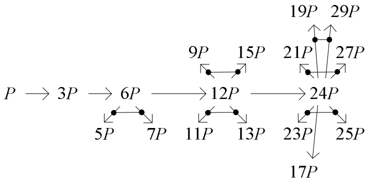
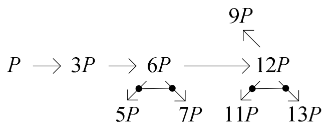
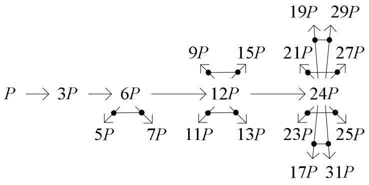
The formula for mixed addition can be obtained by setting $Z_2 = 1$ in formula (23) and has a cost of $9M + 1S + 4A$. Then, if the partial values $(X_1 x_2 Y_1 y_2 + dZ_1^2)$, $X_1 x_2$, $Y_1 y_2$, $(X_1 x_2 Y_1 y_2 - dZ_1^2)$, $X_1 y_2$ and $x_2 Y_1$ are temporarily cached, then the conjugate mixed addition $P - Q = P + (-Q) = (X_1, Y_1, Z_1) + (-x_2, y_2) = (X_4, Y_4, Z_4)$ can be performed by:

$$X_4 = [X_1 x_2 Y_1 y_2 - dZ_1^2](X_1 x_2 + Y_1 y_2)\,,\ Y_4 = -[X_1 x_2 Y_1 y_2 + dZ_1^2](X_1 y_2 - x_2 Y_1)\,,$$
$$Z_4 = -Z_1 (X_1 x_2 + Y_1 y_2)(X_1 y_2 - x_2 Y_1)\,, \quad (25)$$

which only costs $4M + 2A$. We remark that memory requirements of the new conjugate formulas can be minimized by performing P+Q and P−Q in "parallel".

D Calculation of precomputed points for different values *m*

The following table shows the proposed precomputing sequences for different values *m*. For $m = 5$, the first sequence corresponds to $\mathcal{J}$ and $\mathcal{JQ}$, and the second one to $\mathcal{IE}$ coordinates. Tied arrows denote an addition/conjugate addition pair (or mixed addition/conjugate mixed addition pair if addition is performed with affine point *P*).

m	Precomputation Scheme	*m*	Precomputation Scheme
3		15	
5		17	
7		19	
9		27	 19P 9P 15P 21P 27P P → 3P → 6P → 12P → 24P 5P 7P 11P 13P 23P 25P 17P
11		29	 19P 29P 9P 15P 21P 27P P → 3P → 6P → 12P → 24P 5P 7P 11P 13P 23P 25P 17P
13		31	 19P 29P 9P 15P 21P 27P P → 3P → 6P → 12P → 24P 5P 7P 11P 13P 23P 25P 17P 31P