New Formulae for Efficient Elliptic Curve Arithmetic

Huseyin Hisil, Gary Carter, and Ed Dawson

Information Security Institute, Queensland University of Technology
{h.hisil, g.carter, e.dawson}@qut.edu.au

Abstract. This paper is on efficient implementation techniques of Elliptic Curve Cryptography. In particular, we improve timings[1] for Jacobi-quartic (3M+4S) and Hessian (7M+1S or 3M+6S) doubling operations. We provide a faster mixed-addition (7M+3S+1d) on modified Jacobi-quartic coordinates. We introduce tripling formulae for Jacobi-quartic (4M+11S+2d), Jacobi-intersection (4M+10S+5d or 7M+7S+3d), Edwards (9M+4S) and Hessian (8M+6S+1d) forms. We show that Hessian tripling costs 6M+4C+1d for Hessian curves defined over a field of characteristic 3. We discuss an alternative way of choosing the base point in successive squaring based scalar multiplication algorithms. Using this technique, we improve the latest mixed-addition formulae for Jacobi-intersection (10M+2S+1d), Hessian (5M+6S) and Edwards (9M+1S+1d+4a) forms. We discuss the significance of these optimizations for elliptic curve cryptography.

Keywords: Elliptic curve, efficient point multiplication, doubling, tripling, DBNS.

1 Introduction

One of the main challenges in elliptic curve cryptography is to perform scalar multiplication efficiently. In the last decade, much effort has been spent in representing the elliptic curves in special forms which permit faster point doubling and addition. In particular,

- Cohen, Miyaji and Ono [1] showed fast implementation in Weierstrass form on Jacobian coordinates.
- Smart [2], Joye and Quisquater [3], Liardet and Smart [4], Billet and Joye [5] showed ways of doing point multiplication to resist side channel attacks.
- Doche, Icart and Kohel [6] introduced the fastest doubling[2] and tripling in Weierstrass form on two different families of curves.

[1] **M:** Field multiplication, **S:** Field squaring, **C:** Field cubing on characteristic 3 fields, **d:** Multiplication by a curve constant. **a:** Addition. For simplicity in our analysis, we fix 1S=0.8M, 1C=0.1M, 1d=0M.

[2] With the improvements of Bernstein, Birkner, Lange and Peters [7,8].

K. Srinathan, C. Pandu Rangan, M. Yung (Eds.): Indocrypt 2007, LNCS 4859, pp. 138–151, 2007.

- Negre [9], Kim, Kim and Choe [10], Smart [11] investigated the case of efficient arithmetic for low odd characteristic curves.
- Dimitrov, Imbert and Mishra [12] showed the first efficient inversion free tripling formula on Jacobian coordinates. Meloni [13] showed how special addition chains can be used in point multiplication.
- Avanzi, Dimitrov, Doche and Sica [14] and Doche and Imbert [15] provided an extended way of using DBNS[3] in elliptic curve cryptography.
- Edwards [16] introduced a new representation of elliptic curves. Bernstein and Lange [17,18] showed the importance of this new system for providing fast arithmetic and side channel resistance. They have also built a database [7] of explicit formulae that are reported in the literature together with their own optimizations. Our work greatly relies on the formulae reported in this database.

This paper is composed of several optimizations regarding elliptic curve arithmetic operations. We improve some of the previously reported elliptic curve group operations namely Jacobi-quartic doubling, Jacobi-quartic mixed addition and Hessian doubling. As well, we introduce elliptic curve point tripling formulae for Jacobi-quartic, Jacobi-intersection, Hessian and Edwards forms. We introduce a technique for successive squaring based point multiplication algorithms which speeds-up mixed addition in some forms. This technique enables faster mixed-addition for Jacobi-intersection, Hessian and Edwards curves. The optimizations in this paper are solely efficiency oriented. Therefore, these results do not cover side channel resistance. Some immediate outcomes are;

- Jacobi-quartic form becomes competitive in efficiency oriented applications. For instance, successive squaring based point multiplications on modified/ extended Jacobi-quartic coordinates can be performed faster than standard[4] Edwards coordinates for all S/M values.
- The tripling formulae that are introduced provide a wider background for new comparisons on DBNS based point multiplications in different systems.
- Point multiplication in Hessian form can be performed faster than Jacobian form whenever S/M is near to 1.
- Hessian tripling formula enables efficient implementation of DBNS based point multiplication algorithms with Hessian curves defined over fields of characteristic 3.

This paper is organized as follows. We show faster doubling formulae in Section 2. We introduce new tripling formulae in Section 3. We provide a faster Jacobi-quartic mixed addition in Section 4. We describe an alternative strategy for the selection of base point for point multiplication in Section 5. We draw our conclusions in Section 6.

[3] DBNS: Double Base Number System.

[4] Very recently, Bernstein and Lange [7] developed the inverted-Edwards coordinates which requires less memory and bandwidth.

2 New Doubling Formulae

We observe that output coordinates of an inversion-free formula can be represented alternatively by selecting congruent elements from $K[E]$, i.e. the coordinate ring of E over K. In other words, an inversion-free formula (which is originally derived from its affine version) can be modified using the curve equation. We do not report the detail for finding all new explicit formulae since the process is composed of fairly tedious steps. Nevertheless, a step-by-step derivation of doubling a formula for Jacobi-quartic form is given in the Appendix as an example. Tripling formulae in Section 3 can be derived building on the same ideas.

Let K be a field with $\mathrm{char}(K) \neq 2, 3$. An elliptic curve in Jacobi-quartic form [5] is defined by $E(K)\colon y^2 = x^4 + 2ax^2 + 1$ where $a^2 - 1$ is nonzero. The identity element is the point $(0, 1)$. We do not give background material on curves since it is well documented in the literature. Our optimization leads to the following strategy for inversion-free Jacobi-quartic doubling. (See the Appendix).

$$X_3 = Y_1((X_1 - Z_1)^2 - (X_1^2 + Z_1^2)), \qquad Z_3 = (X_1^2 + Z_1^2)(X_1^2 - Z_1^2),$$

$$Y_3 = 2(Y_1(X_1^2 + Z_1^2))^2 - (X_3^2 + Z_3^2).$$

The operation count shows that Jacobi-quartic doubling costs **3M+4S** (previous best [7], 1M+9S) when the points are represented on the modified coordinates, $(X_1 : Y_1 : Z_1 : X_1^2 : Z_1^2)$. This redundant representation will also be used to improve the mixed-addition for Jacobi-quartic form in Section 4. The operations can be scheduled as follows. We denote the input cache registers as (X_1^2) and (Z_1^2). Re-caching is done on (X_3^2) and (Z_3^2).

$X_3 \leftarrow X_1 + Z_1$	$Y_3 \leftarrow Z_3 \times Y_1$	$Y_3 \leftarrow Y_3^2$	$(Z_3^2) \leftarrow Z_3^2$
$X_3 \leftarrow X_3^2$	$X_3 \leftarrow Y_3 - X_3$	$Y_3 \leftarrow 2 * Y_3$	$Y_3 \leftarrow Y_3 - (Z_3^2)$
$X_3 \leftarrow X_3 \times Y_1$	$R_0 \leftarrow (X_1^2) - (Z_1^2)$	$(X_3^2) \leftarrow X_3^2$	
$Z_3 \leftarrow (X_1^2) + (Z_1^2)$	$Z_3 \leftarrow Z_3 \times R_0$	$Y_3 \leftarrow Y_3 - (X_3^2)$	

Most applications overwrite (X_1^2) and (Z_1^2). In this case, temporary register R_0 can be replaced with (X_1^2) or (Z_1^2). In addition, (X_3^2) and (Z_3^2) can be the same registers as (X_1^2) and (Z_1^2), respectively. This scheduling method uses 6 additions and 1 multiplication by 2. If the caching is performed more redundantly as $(X_1 : Y_1 : Z_1 : X_1^2 : Z_1^2 : (X_1^2 + Z_1^2))$, it is possible to save one addition and to avoid the use of R_0.

Let K be a field with $\mathrm{char}(K) \neq 2$. An elliptic curve in Hessian form [2] is defined by $E(K)\colon x^3 + y^3 + 1 = cxy$ where $(c/3)^3 - 1$ is nonzero. The identity element is the point at infinity. The cost of inversion-free doubling was reported as 6M+3S with respect to the following formula where each coordinate is cubed and used in the obvious way [7].

$$X_3 = Y_1(Z_1^3 - X_1^3), \quad Y_3 = X_1(Y_1^3 - Z_1^3), \quad Z_3 = Z_1(X_1^3 - Y_1^3).$$

However, the same formula costs **7M+1S** when the following strategy is used.

$$A \leftarrow X_1^2, \quad B \leftarrow Y_1(X_1 + Y_1), \quad C \leftarrow A + B, \quad D \leftarrow Z_1(Z_1 + X_1),$$
$$E \leftarrow A + D, \quad F \leftarrow C(X_1 - Y_1), \quad G \leftarrow E(Z_1 - X_1),$$
$$X_3 \leftarrow G\,Y_1, \quad Y_3 \leftarrow -(F + G)\,X_1, \quad Z_3 \leftarrow F\,Z_1.$$

Furthermore, it is possible to save 1 reduction by delaying the reduction steps after computing $X_1^2, Y_1(X_1 + Y_1)$ and $Z_1(Z_1 + X_1)$ whenever desired. Lazy reductions effect the timings when arbitrary moduli are used for field reductions. If fast reduction moduli (such as NIST primes) are used then this advantage vanishes. Note, we do not include this optimization in our complexity analysis. The operations for Hessian doubling can be scheduled as follows.

$R_0 \leftarrow X_1^2$	$R_2 \leftarrow Z_1 \times R_2$	$R_0 \leftarrow R_1 \times R_0$	$X_3 \leftarrow R_1 \times Y_1$
$R_1 \leftarrow X_1 + Y_1$	$R_2 \leftarrow R_0 + R_2$	$Z_3 \leftarrow R_0 \times Z_1$	$R_2 \leftarrow -(R_0 + R_1)$
$R_1 \leftarrow Y_1 \times R_1$	$R_1 \leftarrow R_0 + R_1$	$R_1 \leftarrow Z_1 - X_1$	$Y_3 \leftarrow R_2 \times X_1$
$R_2 \leftarrow Z_1 + X_1$	$R_0 \leftarrow X_1 - Y_1$	$R_1 \leftarrow R_2 \times R_1$	

An alternative layout is as follows.

$$\begin{aligned}
X_3 &= (((X_1 + Y_1)^2 - (X_1^2 + Y_1^2)) - ((Y_1 + Z_1)^2 - (Y_1^2 + Z_1^2))) \cdot \\
&\quad (((X_1 + Z_1)^2 - (X_1^2 + Z_1^2)) + 2(X_1^2 + Z_1^2)) \\
Y_3 &= (((X_1 + Z_1)^2 - (X_1^2 + Z_1^2)) - ((X_1 + Y_1)^2 - (X_1^2 + Y_1^2))) \cdot \\
&\quad (((Y_1 + Z_1)^2 - (Y_1^2 + Z_1^2)) + 2(Y_1^2 + Z_1^2)) \\
Z_3 &= (((Y_1 + Z_1)^2 - (Y_1^2 + Z_1^2)) - ((X_1 + Z_1)^2 - (X_1^2 + Z_1^2))) \cdot \\
&\quad (((X_1 + Y_1)^2 - (X_1^2 + Y_1^2)) + 2(X_1^2 + Y_1^2))
\end{aligned}$$

This strategy costs **3M+6S**. There are no lazy reduction possibilities. It requires more additions and more temporary registers. However, it will be faster whenever $1\mathrm{S} < 0.8\mathrm{M}$. It is known that $1\mathrm{S} \approx 0.66\mathrm{M}$ when fast reduction moduli are used. The operations can be scheduled as follows.

$R_0 \leftarrow X_1^2$	$R_0 \leftarrow R_0^2$	$R_2 \leftarrow R_2^2$	$Y_3 \leftarrow R_0 - R_1$
$R_1 \leftarrow Y_1^2$	$R_0 \leftarrow R_0 - R_3$	$R_4 \leftarrow 2 * R_4$	$R_5 \leftarrow R_2 + R_5$
$R_2 \leftarrow Z_1^2$	$R_1 \leftarrow X_1 + Z_1$	$R_2 \leftarrow R_2 - R_5$	$Y_3 \leftarrow Y_3 \times R_5$
$R_3 \leftarrow R_0 + R_1$	$R_1 \leftarrow R_1^2$	$R_5 \leftarrow 2 * R_5$	$Z_3 \leftarrow R_1 - R_2$
$R_4 \leftarrow R_0 + R_2$	$R_1 \leftarrow R_1 - R_4$	$X_3 \leftarrow R_2 - R_0$	$R_0 \leftarrow R_0 + R_3$
$R_5 \leftarrow R_1 + R_2$	$R_2 \leftarrow Y_1 + Z_1$	$R_4 \leftarrow R_1 + R_4$	$Z_3 \leftarrow Z_3 \times R_0$
$R_0 \leftarrow X_1 + Y_1$	$R_3 \leftarrow 2 * R_3$	$X_3 \leftarrow X_3 \times R_4$	

The comparison of doubling costs in different systems is depicted in Table 1.

Table 1. Cost comparison of elliptic curve point doubling operations in different coordinate systems. The bold values are the old and the new timings that are explained in this section. We assume 1S=0.8M.

System	Cost analysis	Total
Hessian (OLD) [3]	6M + 3S	**8.4M**
Jacobi-quartic (OLD) [5]	1M + 9S + 1d	**8.2M**
Hessian (NEW-1)	7M + 1S	**7.8M**
Hessian (NEW-2)	3M + 6S	**7.8M**
Doche/Icart/Kohel(3) [7,6]	2M + 7S + 2d	7.6M
Jacobian [1,7]	1M + 8S + 1d	7.4M
Jacobian, $a = -3$ [1,7]	3M + 5S	7.0M
Jacobi-intersections [7,4]	3M + 4S	6.2M
Inverted Edwards [7,8]	3M + 4S + 1d	6.2M
Edwards [7,17,18]	3M + 4S	6.2M
Jacobi-quartic (NEW)	3M + 4S	**6.2M**
Doche/Icart/Kohel(2) [7,6]	2M + 5S + 2d	6.0M

3 New Tripling Formulae

Since DBNS based point multiplication algorithms [12,14,15] have been introduced, there has been a demand for fast tripling formulae. We introduce tripling formulae for Jacobi-quartic, Hessian, Jacobi-intersection and Edwards forms in this section. Tripling formulae can be derived by the composition of doubling and addition formulae. However, a straight forward derivation yields expensive expressions. Nevertheless, it is possible to do simplifications using the curve equation. To the best of our knowledge, no algorithm is known to guarantee the best strategy. Therefore, one can expect further improvements in these formulae in the future. At least, there should be multiplication/squaring tradeoffs.

Following the same notation in Section 2, we introduce Jacobi-quartic tripling. The formula is as follows.

$$\begin{aligned} X_3 &= X_1(X_1^8 - 6X_1^4Z_1^4 - 8aX_1^2Z_1^6 - 3Z_1^8) \\ Y_3 &= Y_1(X_1^{16} + 8aX_1^{14}Z_1^2 + 28X_1^{12}Z_1^4 + 56aX_1^{10}Z_1^6 + 6X_1^8Z_1^8 + \\ &\quad 64a^2X_1^8Z_1^8 + 56aX_1^6Z_1^{10} + 28X_1^4Z_1^{12} + 8aX_1^2Z_1^{14} + Z_1^{16}) \\ Z_3 &= Z_1(3X_1^8 + 8aX_1^6Z_1^2 + 6X_1^4Z_1^4 - Z_1^8) \end{aligned}$$

The terms can be organized as follows.

$$A \leftarrow (X_1^2)^2, \quad B \leftarrow (Z_1^2)^2, \quad C \leftarrow 2(((X_1^2) + (Z_1^2))^2 - (A + B)),$$

$$D \leftarrow (a^2 - 1)C^2, \quad E \leftarrow 4(A - B), \quad F \leftarrow 2(A + B) + aC, \quad G \leftarrow E^2,$$

$$H \leftarrow F^2, \quad J \leftarrow (E + F)^2 - (G + H), \quad K \leftarrow 2(H - D),$$

$$X_3 \leftarrow X_1(J - K), \quad Y_3 \leftarrow Y_1(K^2 - 4GD), \quad Z_3 \leftarrow Z_1(J + K),$$

Jacobi-quartic tripling costs 4M+10S+2d in the modified Jacobi-quartic coordinates. The formulae that will take advantage of the fast Jacobi-quartic addition formula [19] require the addition of XY coordinate to modified coordinates

$$(X_1 : Y_1 : Z_1 : X_1^2 : Z_1^2 : (X_1^2 + Z_1^2)).$$

This modification fixes the complexity to **4M+11S+2d**. We refer the reader to EFD [7] for compatible versions developed by Bernstein and Lange.

Temporary registers R_5 and R_6 can be replaced with volatile cache registers (X_1^2) and (Z_1^2). As well, (X_3^2) and (X_3^2) can be the same registers as (X_1^2) and (Z_1^2), respectively.

In the same fashion, we introduce Hessian tripling. We follow the same notation for Hessian curves that is given in Section 2. Set $k = c^{-1}$. One can treat k as the curve constant confidently since addition and doubling formulae do not depend on c. An efficient tripling can be performed using the following formula.

$$\begin{aligned}
X_3 &= X_1^3(Y_1^3 - Z_1^3)(Y_1^3 - Z_1^3) + Y_1^3(X_1^3 - Y_1^3)(X_1^3 - Z_1^3)\\
Y_3 &= Y_1^3(X_1^3 - Z_1^3)(X_1^3 - Z_1^3) - X_1^3(X_1^3 - Y_1^3)(Y_1^3 - Z_1^3)\\
Z_3 &= k(X_1^3 + Y_1^3 + Z_1^3)((X_1^3 - Y_1^3)^2 + (X_1^3 - Z_1^3)(Y_1^3 - Z_1^3))
\end{aligned}$$

The operations are organized as follows.

$$A \leftarrow X_1^3, \quad B \leftarrow Y_1^3, \quad C \leftarrow Z_1^3, \quad D \leftarrow A - B, \quad E \leftarrow A - C,$$

$$H \leftarrow D^2, \quad J \leftarrow E^2, \quad K \leftarrow F^2, \quad X_3 \leftarrow 2AK - B(K - H - J),$$

$$Y_3 \leftarrow 2BJ - A(J - H - K), \quad Z_3 \leftarrow k(A + B + C)(J + H + K).$$

This formula costs **8M+6S+1d**. Furthermore, there exists 2 lazy reduction points. (First, delay reduction when computing AK and $B(K - H - J)$, then delay reduction when computing BJ and $A(J - H - K)$). If the Hessian curve is defined over a field of characteristic 3, the tripling formula simplifies to the following. Note, it is enough to choose a nonzero k in this case.

$$\begin{aligned}
X_3 &= (X_1(Y_1 - Z_1)(Y_1 - Z_1) + Y_1(X_1 - Y_1)(X_1 - Z_1))^3\\
Y_3 &= (Y_1(X_1 - Z_1)(X_1 - Z_1) - X_1(X_1 - Y_1)(Y_1 - Z_1))^3\\
Z_3 &= k((X_1 + Y_1 + Z_1)^3)^3
\end{aligned}$$

It is easy to see that this formula costs **6M+4C+1d**. (Reuse $X_1(Y_1 - Z_1)$ and $Y_1(X_1 - Z_1)$). Furthermore, 2 additional lazy reductions can be done in the computation of X_3 and Y_3. It is interesting to note that the cost of 5P=2P+3P is less than a point addition. Recently, Kim, Kim and Choe [10] introduced 4M+5C+2d tripling formula in Jacobian/ML coordinates which is faster than the tripling introduced here.

Next, we introduce the tripling formula for Jacobi-intersection form [4]. Let K be a field with $\mathrm{char}(K) \neq 2, 3$ and let $a \in K$ with $a(1 - a) \neq 0$. The elliptic curve in Jacobi-intersection form is the set of points which satisfy the equations

$s^2 + c^2 = 1$ and $as^2 + d^2 = 1$ simultaneously. The identity element is the point $(0, 1, 1)$. The inversion-free tripling formula is as follows. With $k = a - 1$,

$$\begin{aligned}
S_3 &= S_1(k(kS_1^8 + 6S_1^4C_1^4 + 4S_1^2C_1^6) - 4S_1^2C_1^6 - 3C_1^8)\\
C_3 &= C_1(k(3kS_1^8 + 4kS_1^6C_1^2 - 4S_1^6C_1^2 - 6S_1^4C_1^4) - C_1^8)\\
D_3 &= D_1(k(-kS_1^8 + 4S_1^6C_1^2 + 6S_1^4C_1^4 + 4S_1^2C_1^6) - C_1^8)\\
T_3 &= T_1(k(-kS_1^8 - 4kS_1^6C_1^2 + 6S_1^4C_1^4) - 4S_1^2C_1^6 - C_1^8)
\end{aligned}$$

The operations can be organized as follows.

$$E \leftarrow S_1^2, \quad F \leftarrow C_1^2, \quad G \leftarrow E^2, \quad H \leftarrow F^2, \quad J \leftarrow G^2, \quad K \leftarrow H^2,$$

$$L \leftarrow ((E+F)^2 - H - G), \quad M \leftarrow L^2, \quad N \leftarrow (G+L)^2 - J - M,$$

$$P \leftarrow (H+L)^2 - K - M, \quad R \leftarrow 2k^2J, \quad S \leftarrow 2kN,$$

$$T \leftarrow 3kM, \quad U \leftarrow 2P, \quad V \leftarrow 2K, \quad W \leftarrow (k+1)U, \quad Y \leftarrow (k+1)S,$$

$$S_3 \leftarrow S_1((R-V)+(T+W)-2(U+V)), \quad C_3 \leftarrow C_1((R-V)-(T-Y)+2(R-S)),$$

$$D_3 \leftarrow D_1((T+W)-(R-S)-(U+V)), \quad T_3 \leftarrow T_1((T-Y)-(R-S)-(U+V)).$$

This formula costs **4M+10S+5d**. An alternative strategy costs **7M+7S+3d**. Here is the alternative organization of operations.

$$E \leftarrow S_1^2, \quad F \leftarrow C_1^2, \quad G \leftarrow E^2, \quad H \leftarrow F^2, \quad J \leftarrow 2H, \quad K \leftarrow 2J,$$

$$L \leftarrow (2F+E)^2 - G - K, \quad M \leftarrow kG, \quad N \leftarrow K + J, \quad P \leftarrow M^2,$$

$$R \leftarrow NM, \quad U \leftarrow ML, \quad V \leftarrow H^2, \quad W \leftarrow HL,$$

$$S_3 \leftarrow S_1(R+kW+2(P-V)-W-P-V), \quad C_3 \leftarrow C_1(2(P-V)-U+P+V-R+kU),$$

$$D_3 \leftarrow D_1(U - P - V + R + kW), \quad T_3 \leftarrow T_1(R - kU - W - P - V).$$

Finally, we introduce the tripling formula for Edwards curves [16,17,18]. Let K be a field with $\mathrm{char}(K) \neq 2$ and let $c, d \in K$ with $cd(1 - c^4d) \neq 0$. Then, the Edwards curve, $(x^2+y^2) = c^2(1+dx^2y^2)$, is birationally equivalent to an elliptic curve [17,18]. The identity element is the point $(0, c)$. The same formula was independently developed by Bernstein, Birkner, Lange and Peters [8]. Edwards tripling costs **9M+4S**. For further results, we refer the reader to Bernstein, Birkner, Lange and Peters [8]. The inversion-free tripling formula is as follows.

$$\begin{aligned}
X_3 &= X_1(X_1^4 + 2X_1^2Y_1^2 - 4c^2Y_1^2Z_1^2 + Y_1^4)(X_1^4 - 2X_1^2Y_1^2 + 4c^2Y_1^2Z_1^2 - 3Y_1^4)\\
Y_3 &= Y_1(X_1^4 + 2X_1^2Y_1^2 - 4c^2X_1^2Z_1^2 + Y_1^4)(3X_1^4 + 2X_1^2Y_1^2 - 4c^2X_1^2Z_1^2 - Y_1^4)\\
Z_3 &= Z_1(X_1^4 - 2X_1^2Y_1^2 + 4c^2Y_1^2Z_1^2 - 3Y_1^4)(3X_1^4 + 2X_1^2Y_1^2 - 4c^2X_1^2Z_1^2 - Y_1^4)
\end{aligned}$$

The cost comparison of tripling formulae in different systems is depicted in Table 2. (Also see Table 3 in the Appendix).

Table 2. Cost comparison of elliptic curve point tripling formulae in different coordinate systems. The bold lines correspond to the complexities of the formulae that are introduced in this section. We assume 1S=0.8M.

System	Tripling Cost	Total
Jacobian [12,7]	5M+10S+1d	13M
Hessian	8M+6S+1d	**12.8M**
Jacobi-quartic	4M+11S+2d	**12.8M**
Jacobi-intersection-2	7M+7S+3d	**12.6M**
Jacobian, $a = -3$ [12,7]	7M+7S	12.6M
Edwards [7,8]	9M+4S+1d	**12.2M**
Inverted Edwards [7,8]	9M+4S+1d	12.2M
Jacobi-intersection-1	4M+10S+5d	**12M**
Doche/Icart/Kohel(3) [6]	6M+6S+2d	10.8M
Hessian, char$= 3$	6M+4C+2d	**6.4M**

4 Mixed-Addition for Modified Jacobi-Quartic Coordinates

Following the outline on modified Jacobi-quartic doubling (see Section 2), we provide a mixed-addition which is faster than the previous best. The updated formula [19,7] is as follows.

$$\begin{aligned} X_3 &= (Y_1 + (X_1 + Z_1)^2 - (X_1^2 + Z_1^2))(2X_2 + Y_2) - \\ &\quad 2X_2((X_1 + Z_1)^2 - (X_1^2 + Z_1^2)) - (Y_1Y_2) \\ Y_3 &= 4X_2((X_1 + Z_1)^2 - (X_1^2 + Z_1^2))(X_1^2 + a(Z_1^2 + X_1^2X_2^2) + Z_1^2X_2^2) + \\ &\quad 4(Z_1^2 + X_1^2X_2^2)(Y_1Y_2) \\ Z_3 &= 2(Z_1^2 - X_1^2X_2^2) \end{aligned}$$

The operations can be organized as follows.

$$A \leftarrow (X_1^2) + (Z_1^2), \quad B \leftarrow (X_1 + Z_1)^2 - A, \quad C \leftarrow B + Y_1, \quad D \leftarrow (X_1^2)(X_2^2),$$
$$E \leftarrow 2BX_2, \quad F \leftarrow (Z_1^2) + D, \quad G \leftarrow 2E, \quad H \leftarrow Y_1Y_2,$$
$$X_3 \leftarrow C(2X_2 + Y_2) - E - H, \quad Y_3 \leftarrow 4FH + ((Z_1^2)(X_2^2) + aF + (X_1^2))G,$$
$$Z_3 \leftarrow 2((Z_1^2) - D), \quad (X_3^2) \leftarrow X_3^2, \quad (Z_3^2) \leftarrow Z_3^2.$$

This formula costs **7M+3S+1d** (previous best, 8M+3S+1d). Let C_0 and C_1 be static registers. $C_0 \leftarrow X_2^2$ and $C_1 \leftarrow 2X_2 + Y_2$ are precomputed and stored permanently. If C_1 is not used, an extra addition and a multiplication by 2 is to be performed for each mixed-addition. The operations can be scheduled as follows.

The formulae that will take the advantage of the fast Jacobi-quartic addition formula that is described by Duquesne [19] require the addition of XY coordinate to modified coordinates,

$$(X_1 : Y_1 : Z_1 : X_1^2 : Z_1^2 : (X_1^2 + Z_1^2)).$$

$$
\begin{array}{l|l|l|l}
X_3 \leftarrow X_1 + Z_1 & Z_3 \leftarrow (Z_1^2) - R_1 & R_1 \leftarrow R_1 + (Z_1^2) & R_1 \leftarrow R_1 \times R_0 \\
X_3 \leftarrow X_3^2 & Z_3 \leftarrow 2 * Z_3 & Y_3 \leftarrow Y_3 \times R_1 & R_1 \leftarrow 2 * R_1 \\
X_3 \leftarrow X_3 - (X_1^2) & X_3 \leftarrow X_3 \times C_1 & Y_3 \leftarrow 4 * Y_3 & Y_3 \leftarrow Y_3 + R_1 \\
R_0 \leftarrow X_3 - (Z_1^2) & X_3 \leftarrow X_3 - Y_3 & R_1 \leftarrow a * R_1 & (X_3^2) \leftarrow X_3^2 \\
X_3 \leftarrow R_0 + Y_1 & R_0 \leftarrow R_0 \times X_2 & R_1 \leftarrow R_1 + (X_1^2) & (Z_3^2) \leftarrow Z_3^2 \\
Y_3 \leftarrow Y_1 \times Y_2 & R_0 \leftarrow 2 * R_0 & R_2 \leftarrow (Z_1^2) \times C_0 & \\
R_1 \leftarrow (X_1^2) \times C_0 & X_3 \leftarrow X_3 - R_0 & R_1 \leftarrow R_1 + R_2 &
\end{array}
$$

We refer the reader to EFD [7] for compatible versions developed by Bernstein and Lange. This coordinate system is named as extended Jacobi-quartic coordinates.

5 Alternative Base Points

In this section, we introduce a technique that is useful for successive squaring based point multiplication algorithms. Our technique improves the mixed-addition timings reported in the literature. We show how an affine point can be represented alternatively in its projective version. Point addition with these alternative points is faster for some of the forms. We will abuse the terminology and call this type of addition as mixed-addition too since these points require the same amount of storage as affine points and they are kept fixed during the point multiplication.

We follow the same notation for Jacobi-intersection curves in Section 3. Let $(S_1 : C_1 : D_1 : T_1)$ and (s_2, c_2, d_2) with $s_2 \neq 0$ be two points to be added. We can observe that representing the base point (s_2, c_2, d_2) as $(1 : (c_2/s_2) : (d_2/s_2) : (1/s_2))$ leads to a faster formulation. We rename this new representation as $(1 : C_2 : D_2 : T_2)$. With this setup we have,

$$
\begin{aligned}
S_3 &= (T_1C_2 + D_1)(C_1T_2 + S_1D_2) - T_1C_2C_1T_2 - D_1S_1D_2 \\
C_3 &= T_1C_2C_1T_2 - D_1S_1D_2 \\
D_3 &= T_1D_1T_2D_2 - aS_1C_1C_2 \\
T_3 &= (T_1C_2)^2 + D_1^2
\end{aligned}
$$

This formula is from Liardet and Smart [4]. The operations can be organized as follows.

$$
\begin{gathered}
E \leftarrow T_1C_2, \quad F \leftarrow S_1D_2, \quad G \leftarrow C_1T_2, \quad H \leftarrow EG, \quad J \leftarrow D_1F, \\
S_3 \leftarrow (E + D_1)(G + F) - H - J, \quad C_3 \leftarrow H - J, \\
D_3 \leftarrow T_1D_1(T_2D_2) - aS_1C_1C_2, \quad T_3 \leftarrow E^2 + D_1^2,
\end{gathered}
$$

If (T_2D_2) is cached permanently, this formula costs **10M+2S+1d** (previous best, 11M+2S+1d). The cost of computing the alternative base point $(1 : C_2 : D_2 : T_2)$ can be omitted if it is directly selected as the base point itself.

We follow the same notation for Hessian curves in Section 2. Let $(X_1 : Y_1 : Z_1)$ and (x_2, y_2) with $x_2 \neq 0$ be two points to be added. We can observe that

representing the base point (x_2, y_2) as $(1 : (y_2/x_2) : (1/x_2))$ leads to a faster formulation. We rename this new representation as $(1 : Y_2 : Z_2)$. Then, we have

$$\begin{aligned} X_3 &= Y_1^2 Z_2 - X_1 Z_1 Y_2^2 \\ Y_3 &= X_1^2 (Y_2 Z_2) - Y_1 Z_1 \\ Z_3 &= Z_1^2 Y_2 - X_1 Y_1 Z_2^2 \end{aligned}$$

This formula costs **5M+6S**. The operations can be organized as follows where $S_0 \leftarrow Y_2^2$, $S_1 = Z_2^2$ and $S_2 \leftarrow 2(Y_2 + Z_2)^2 - S_0 - S_1$ are cached permanently.

$$A \leftarrow X_1^2, \quad B \leftarrow Y_1^2, \quad C \leftarrow Z_1^2, \quad D \leftarrow A + B, \quad E \leftarrow A + C, \quad F \leftarrow B + C,$$

$$G \leftarrow (X_1 + Y_1)^2 - D, \quad H \leftarrow (X_1 + Z_1)^2 - E, \quad J \leftarrow (Y_1 + Z_1)^2 - F,$$

$$X_3 \leftarrow 2BZ_2 - HS_0, \quad Y_3 \leftarrow AS_2 - J, \quad Z_3 \leftarrow 2CY_2 - GS_1.$$

The same idea works for Edwards curves reducing the number of additions in the mixed-addition formula. We follow the same notation for Edwards curves in Section 3. The Edwards mixed-addition formula that is described by Bernstein and Lange [17,18,7] costs 9M+1S+1d+7a. Let $(X_1 : Y_1 : Z_1)$ and (x_2, y_2) with $x_2 \neq 0$ be two points to be added. We can observe that representing the base point (x_2, y_2) as $(1 : (y_2/x_2) : (1/x_2))$ leads to a slightly faster formulation. We rename this new representation as $(1 : Y_2 : Z_2)$. With this setup we have,

$$\begin{aligned} X_3 &= (X_1 Y_2 + Y_1)(Z_1 Z_2)((Z_1 Z_2)^2 - d(X_1 Y_1 Y_2)) \\ Y_3 &= (Y_1 Y_2 - X_1)(Z_1 Z_2)((Z_1 Z_2)^2 + d(X_1 Y_1 Y_2)) \\ Z_3 &= c((Z_1 Z_2)^2 + d(X_1 Y_1 Y_2))((Z_1 Z_2)^2 - d(X_1 Y_1 Y_2)) \end{aligned}$$

The operation count shows that the alternative Edwards mixed-addition costs **9M+1S+1d+4a**. This formula invokes 3 fewer field additions. Note, the curve parameter c can always be made 1. In this case, multiplication by c is eliminated naturally. The operations for Edwards mixed-addition can be scheduled as follows.

$R_0 \leftarrow X_1 \times Y_2$	$Y_3 \leftarrow Y_3 - X_1$	$R_1 \leftarrow d * R_1$	$X_3 \leftarrow X_3 \times R_0$
$R_0 \leftarrow R_0 + Y_1$	$Z_3 \leftarrow Z_1 \times Z_2$	$Z_3 \leftarrow Z_3^2$	$Y_3 \leftarrow Y_3 \times Z_3$
$Y_3 \leftarrow Y_1 \times Y_2$	$X_3 \leftarrow R_0 \times Z_3$	$R_0 \leftarrow Z_3 - R_1$	$Z_3 \leftarrow Z_3 \times R_0$
$R_1 \leftarrow Y_3 \times X_1$	$Y_3 \leftarrow Y_3 \times Z_3$	$Z_3 \leftarrow Z_3 + R_1$	$Z_3 \leftarrow c * Z_3$

6 Conclusion

We provided several optimizations for doing arithmetic on some special elliptic curve representations. In particular, we have improved the group operations of the Jacobi-quartic form which was initially recommended for providing side channel resistance. With our improvements, Jacobi-quartics became one of the fastest special curves in the speed ranking. For instance, successive squaring based point multiplication can be performed faster than standard Edwards coordinates for

all possible scenarios because both coordinates shares the same complexity for doubling and Jacobi-quartics is faster in addition and mixed-addition. Staying in the same context, if the curve constants are large, extended Jacobi-quartic coordinates provide better timings than inverted-Edwards coordinates.

We have developed tripling formulae for Jacobi-quartic, Jacobi-intersection, Hessian and Edwards forms. These tripling formulae provide a wider background for studying DBNS based applications.

Hessian curves were initially used for providing side channel resistance. We improved Hessian doubling and mixed-addition formulae. With these improvements, point multiplication in Hessian form can be performed faster than Jacobian form if S/M is near to 1. In addition, we showed that the tripling can be performed very efficiently in characteristic 3 case. This improvement enables efficient implementation of DBNS based point multiplication with Hessian (char=3) curves.

We described how the mixed-additions can be done faster in Jacobi-intersection, Hessian and Edwards forms.

One should expect further results in the near future. For example, not all tripling formulae have been developed for all known systems yet. The quintupling formulae are also likely to appear for various forms shortly. Furthermore, the formulae that we introduced might be further improved in time.

Acknowledgements

The authors wish to thank Daniel Bernstein and Tanja Lange for announcing our formulae on EFD [7]. This study would not have been possible without their support and freely available scripts. The authors also wish to thank Christophe Doche for his corrections and suggestions on the preprint version of this paper.

References

1. Cohen, H., Miyaji, A., Ono, T.: Efficient elliptic curve exponentiation using mixed coordinates. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 51–65. Springer, Heidelberg (1998)
2. Smart, N.P.: The Hessian form of an elliptic curve. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 118–125. Springer, Heidelberg (2001)
3. Joye, M., Quisquater, J.J.: Hessian elliptic curves and side-channel attacks. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 402–410. Springer, Heidelberg (2001)
4. Liardet, P.Y., Smart, N.P.: Preventing SPA/DPA in ECC systems using the Jacobi form. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 391–401. Springer, Heidelberg (2001)
5. Billet, O., Joye, M.: The Jacobi model of an elliptic curve and side-channel analysis. In: Fossorier, M.P.C., Høholdt, T., Poli, A. (eds.) AAECC. LNCS, vol. 2643, pp. 34–42. Springer, Heidelberg (2003)

6. Doche, C., Icart, T., Kohel, D.R.: Efficient scalar multiplication by isogeny decompositions. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 191–206. Springer, Heidelberg (2006)
7. Bernstein, D.J., Lange, T.: Explicit-formulas database (2007), Accessible through: http://hyperelliptic.org/EFD
8. Bernstein, D.J., Birkner, P., Lange, T., Peters, C.: Optimizing double-base elliptic-curve single-scalar multiplication. In: INDOCRYPT. LNCS, Springer, Heidelberg (2007)
9. Negre, C.: Scalar multiplication on elliptic curves defined over fields of small odd characteristic. In: Maitra, S., Madhavan, C.E.V., Venkatesan, R. (eds.) INDOCRYPT 2005. LNCS, vol. 3797, pp. 389–402. Springer, Heidelberg (2005)
10. Kim, K.H., Kim, S.I., Choe, J.S.: New fast algorithms for arithmetic on elliptic curves over fields of characteristic three. Cryptology ePrint Archive, Report, 2007/179 (2007), http://eprint.iacr.org/
11. Smart, N.P., Westwood, E.J.: Point multiplication on ordinary elliptic curves over fields of characteristic three. Applicable Algebra in Engineering, Communication and Computing 13(6), 485–497 (2003)
12. Dimitrov, V.S., Imbert, L., Mishra, P.K.: Efficient and secure elliptic curve point multiplication using double-base chains. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 59–78. Springer, Heidelberg (2005)
13. Meloni, N.: Fast and secure elliptic curve scalar multiplication over prime fields using special addition chains. Cryptology ePrint Archive, Report, 2006/216 (2006), http://eprint.iacr.org/
14. Avanzi, R.M., Dimitrov, V., Doche, C., Sica, F.: Extending scalar multiplication using double bases. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 130–144. Springer, Heidelberg (2006)
15. Doche, C., Imbert, L.: Extended double-base number system with applications to elliptic curve cryptography. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 335–348. Springer, Heidelberg (2006)
16. Edwards, H.M.: A normal form for elliptic curves. Bulletin of the AMS 44(3), 393–422 (2007)
17. Bernstein, D.J., Lange, T.: Faster addition and doubling on elliptic curves. Cryptology ePrint Archive, Report, 2007/286 (2007), http://eprint.iacr.org/
18. Bernstein, D.J., Lange, T.: Faster addition and doubling on elliptic curves. In: ASIACRYPT 2007. LNCS, vol. 4833, pp. 29–50. Springer, Heidelberg (2007)
19. Duquesne, S.: Improving the arithmetic of elliptic curves in the Jacobi model. Inf. Process. Lett. 104(3), 101–105 (2007)

Appendix

We give a step by step derivation of the new doubling formula for Jacobi-quartic form. The original formula, described by Billet and Joye [5], is as follows.

$$\begin{aligned} X_3 &= 2X_1Y_1Z_1 \\ Y_3 &= 2aX_1^2Z_1^6 + 4X_1^4Z_1^4 + Y_1^2Z_1^4 + 2aX_1^6Z_1^2 + X_1^4Y_1^2 \\ Z_3 &= Z_1^4 - X_1^4 \end{aligned}$$

Step 1: Modify the point $(X_3 : Y_3 : Z_3)$ to $(-X_3 : Y_3 : -Z_3)$. These two points correspond to the same affine point.

$$\begin{aligned} X_3 &= -2X_1Y_1Z_1 \\ Y_3 &= 2aX_1^2Z_1^6 + 4X_1^4Z_1^4 + Y_1^2Z_1^4 + 2aX_1^6Z_1^2 + X_1^4Y_1^2 \\ Z_3 &= X_1^4 - Z_1^4 \end{aligned}$$

Step 2: Organize X_3 and Z_3. Here, Y_3 should be computed after X_3 and Z_3.

$$\begin{aligned} X_3 &= Y_1((X_1^2 + Z_1^2)) - Y_1(X_1 + Z_1)^2 \\ Z_3 &= (X_1^2 + Z_1^2)(X_1^2 - Z_1^2) \\ Y_3 &= 2aX_1^2Z_1^6 + 4X_1^4Z_1^4 + Y_1^2Z_1^4 + 2aX_1^6Z_1^2 + X_1^4Y_1^2 \end{aligned}$$

Step 3: Use the curve equation, $Y_1^2 = X_1^4 + 2aX_1^2Z_1^2 + Z_1^4$, to find a suitable polynomial representation for Y_3.

$$X_3 = Y_1((X_1^2 + Z_1^2)) - Y_1(X_1 + Z_1)^2$$

$$Z_3 = (X_1^2 + Z_1^2)(X_1^2 - Z_1^2)$$

$$\begin{aligned} Y_3 &= 2aX_1^2Z_1^6 + 4X_1^4Z_1^4 + Y_1^2Z_1^4 + 2aX_1^6Z_1^2 + X_1^4Y_1^2 \\ &= 2aX_1^2Z_1^6 + 4X_1^4Z_1^4 + 2aX_1^6Z_1^2 + (X_1^4 + Z_1^4)Y_1^2 \\ &\equiv 2aX_1^2Z_1^6 + 4X_1^4Z_1^4 + 2aX_1^6Z_1^2 + (X_1^4 + Z_1^4)(Z_1^4 + 2aX_1^2Z_1^2 + X_1^4) \\ &\equiv Z_1^8 + 4aX_1^2Z_1^6 + 6X_1^4Z_1^4 + 4aX_1^6Z_1^2 + X_1^8 \\ &\equiv 2(Y_1Z_1^2 + X_1^2Y_1)^2 - (Z_1^8 - 2X_1^4Z_1^4 + X_1^8) - (4X_1^2Y_1^2Z_1^2) \\ &\equiv 2(Y_1(X_1^2 + Z_1^2))^2 - X_3^2 - Z_3^2 \end{aligned}$$

Table 3. This table contains the best speeds in different systems. The rows are sorted with respect to doubling costs (then mixed-addition). Total complexity is computed for commonly accepted ratios; 1S=0.8M and 1d=0M. **I** is used to describe the cost of a field inversion. The bold values are our contributions. The underlined values are the fastest speeds. The double lines are used for the alternative formulae.

System	Double	Total	Triple	Total	Add	Total	Mixed-Add	Total
Doche/Icart/Kohel(3) [6, 7]	4M+5S+2d	8M	6M+6S+2d	10.8M	11M+6S+1d	15.8M	7M+4S+1d	10.2M
Hessian [3]	7M+1S	**7.8M**	8M+6S+1d	**12.8M**	12M	12M	10M	10M
	3M+6S	**7.8M**					5M+6S	**9.8M**
Jacobian [1, 12, 7]	1M+8S+1d	7.4M	5M+10S+1d	13M	10M+4S	13.2M	7M+4S	10.2M
Jacobian, $a = -3$ [1, 12, 7]	3M+5S	7M	7M+7S	12.6M	10M+4S	13.2M	7M+4S	10.2M
Jacobi-intersection [4, 7]	3M+4S	6.2M	4M+10S+5d	**12M**	13M+2S+1d	14.6M	11M+2S+1d	12.6M
			7M+7S+3d	**12.6M**			10M+2S+1d	**11.6M**
Edwards [7, 17, 18]	3M+4S	6.2M	9M+4S	**12.2M**	10M+1S+1d	10.8M	9M+1S+1d+7a	9.8M
			7M+7S	12.6M			9M+1S+1d+4a	**9.8M**
Extended Jacobi-quartic [5, 19, 7]	3M+4S	**6.2M**	4M+11S+2d	**12.8M**	8M+3S+1d	10.4M	7M+3S+1d	**9.4M**
Inverted Edwards [7, 8]	3M+4S+1d	6.2M	9M+4S+1d	12.2M	9M+1S+1d	9.8M	8M+1S+1d	8.8M
Doche/Icart/Kohel(2) [6, 7]	2M+5S+2d	6M	-	-	12M+5S	16M	8M+4S+1d	11.2M
Hessian, char= 3 [11]	3M+2C	3.2M	6M+4C+1d	**6.4M**	12M	12M	9M+1C	9.1M