Efficient Elliptic Curve Exponentiation Using Mixed Coordinates

Henri Cohen[1], Atsuko Miyaji[2], and Takatoshi Ono[3]

[1] Laboratoire A2X, Université Bordeaux I
[2] Multimedia Development Center, Matsushita Electric Industrial Co., Ltd.
[3] Matsushita Information Systems Research Laboratory Nagoya Co., Ltd.

Abstract. Elliptic curve cryptosystems, proposed by Koblitz ([12]) and Miller ([16]), can be constructed over a smaller field of definition than the ElGamal cryptosystems ([6]) or the RSA cryptosystems ([20]). This is why elliptic curve cryptosystems have begun to attract notice. In this paper, we investigate efficient elliptic curve exponentiation. We propose a new coordinate system and a new mixed coordinates strategy, which significantly improves on the number of basic operations needed for elliptic curve exponentiation.

`key words:` elliptic curve exponentiation, coordinate system

1 Introduction

Koblitz ([12]) and Miller ([16]) proposed a method by which public key cryptosystems can be constructed on the group of points of an elliptic curve over a finite field instead of a finite field. If elliptic curve cryptosystems satisfy both MOV-conditions ([15,10]) and FR-conditions ([4]), and avoid p-divisible elliptic curves over $\mathbb{F}_{p^r}$ ([23,21,25]), then the only known attacks are the Pollard $\rho-$method ([19]) and the Pohlig-Hellman method ([18]). Hence with current knowledge, we can construct elliptic curve cryptosystems over a smaller definition field than the discrete-logarithm-problem (DLP)-based cryptosystems like the ElGamal cryptosystems ([6]) or the DSA ([5]) and RSA cryptosystems ([20]). Elliptic curve cryptosystems with a 160-bit key are thus believed to have the same security as both the ElGamal cryptosystems and RSA with a 1,024-bit key. This is why elliptic curve cryptosystems have been discussed in ISO/IEC CD 14883-3, ISO/IEC DIS 11770-3, ANSI ASC X.9, X.9.62, and IEEE P1363 ([10]). As standardization advances, fast implementations of elliptic curve cryptosystems has been reported ([9,22,27,8,3]).

There are two approaches for efficient elliptic curve exponentiation. One uses general methods valid for any elliptic curve. The other uses ad-hoc methods for special elliptic curves, which use the complex multiplication field ([26,13]). For security purposes, an elliptic curve should not be fixed and be changed periodically. Therefore an efficient algorithm valid for any elliptic curve and not for a fixed elliptic curve is desirable. This paper explores an efficient algorithm valid for any elliptic curve.

K. Ohta and D. Pei (Eds.): ASIACRYPT'98, LNCS 1514, pp. 51–65, 1998.

Elliptic curve exponentiations involve three different factors: the field of definition, the addition-chains ([11,17,14,22]), and the coordinate systems. For the field of definition, we may choose optimal fields on which modular reduction is efficient ([3]) or on which inversion is efficient ([22]). For the addition-chains, the addition-subtraction method is usually mixed with the window method ([11,17,14,22,3]). On the other hand, the optimal coordinate systems have not been so thoroughly studied, though there have been some proposals ([1]). In this paper, we study optimal coordinates for the case of a field of definition $\mathbb{F}_p$ (with p larger than 3). We propose a new coordinate system and a new mixed coordinates strategy for elliptic curve exponentiation.

1. Coordinates of an elliptic curve
An elliptic curve can be represented using several coordinate systems. For each such system, the speed of additions and doublings is different. Therefore a good choice of coordinate system is an important factor for elliptic curve exponentiations. Affine coordinates and projective coordinates are well known ([24]). Two more coordinate systems, the Jacobian coordinates and the five element Jacobian coordinates (which we will call the Chudnovsky Jacobian coordinates) have been proposed in [1]. The efficiency of Jacobian coordinates for elliptic curve exponentiation is discussed in [3].

In the present paper, we introduce what we call modified Jacobian coordinates, which gives faster doublings than affine, projective, Jacobian and Chudnovsky Jacobian coordinates. Since doublings take the largest part of the time for an elliptic curve exponentiation, this leads to noticeable improvements.

2. Strategy of elliptic curve exponentiation
Although we have at our disposal five coordinate systems including our new one, there is no single system which gives both fast doublings and fast additions: for example, the Jacobian coordinates have faster doublings but slower additions than the Chudnovsky Jacobian coordinates. Up to now, for fast elliptic curve exponentiation, a single coordinate system has been used which minimizes the total computation time ([9,22,27,8,3]). This is not the best method since some coordinates are good at additions and others are good at doublings. In this paper, we propose a new strategy using mixed coordinate systems for efficient elliptic curve exponentiation: for doublings, we use the best possible system for doublings, and for additions, we use the best possible system for additions.

This paper is organized as follows. Section 2 discusses the four known coordinate systems. Section 3 presents our new coordinate system and investigates strategies using mixed coordinate systems. The number of basic field operations for elliptic curve exponentiation using mixed coordinates is also estimated. Section 5 presents an implementation of our strategy.

2 The Coordinate Systems

An elliptic curve can be represented by several coordinate systems. We give here the addition and doubling formulas for affine coordinates ([24]), projective

coordinates ([14]), Jacobian coordinates ([1,3]), and Chudnovsky Jacobian coordinates ([1]), as well as the necessary number of field operations. From now on, we assume that $\mathbb{F}_p$ is a field with $p > 3$.

2.1 The Addition Formulas in Affine Coordinate

Let

$$E : y^2 = x^3 + ax + b \ \ (a, b \in \mathbb{F}_p, 4a^3 + 27b^2 \neq 0).$$

be the equation of an elliptic curve E over $\mathbb{F}_p$.

The addition formulas for affine coordinates are the following. Let $P = (x_1, y_1)$, $Q = (x_2, y_2)$ and $P + Q = (x_3, y_3)$ be points on $E(\mathbb{F}_p)$.

• **Curve addition formulas in affine coordinates ($P \neq \pm Q$)**

$$x_3 = \lambda^2 - x_1 - x_2,\ y_3 = \lambda(x_1 - x_3) - y_1, \tag{1}$$

where $\lambda = (y_2 - y_1)/(x_2 - x_1)$;

• **Curve doubling formulas in affine coordinates ($P = Q$)**

$$x_3 = \lambda^2 - 2x_1,\ y_3 = \lambda(x_1 - x_3) - y_1, \tag{2}$$

where $\lambda = (3x_1^2 + a)/(2y_1)$.

Here we discuss the computation times for these formulas in detail. For simplicity, we neglect addition, subtraction and multiplication by a small constant in $\mathbb{F}_p$ because they are much faster than multiplication and inversion in $\mathbb{F}_p$. Let us denote the computation time of an addition (resp. a doubling) by $t(\mathcal{A} + \mathcal{A})$ (resp. $t(2\mathcal{A})$) and represent multiplication (resp. inverse, resp. squaring) in $\mathbb{F}_p$ by M (resp. I, resp. S). Then we see that $t(\mathcal{A} + \mathcal{A}) = I + 2M + S$ and $t(2\mathcal{A}) = I + 2M + 2S$.

2.2 The Addition Formulas in Projective coordinates

For projective coordinates, we set $x = X/Z$ and $y = Y/Z$, giving the equation

$$E_P : Y^2Z = X^3 + aXZ^2 + bZ^3.$$

The addition formulas in projective coordinates are the following. Let $P = (X_1, Y_1, Z_1)$, $Q = (X_2, Y_2, Z_2)$ and $P + Q = R = (X_3, Y_3, Z_3)$.

• **Curve addition formulas in projective coordinates ($P \neq \pm Q$)**

$$X_3 = vA,\ Y_3 = u(v^2X_1Z_2 - A) - v^3Y_1Z_2,\ Z_3 = v^3Z_1Z_2, \tag{3}$$

where $u = Y_2Z_1 - Y_1Z_2, v = X_2Z_1 - X_1Z_2, A = u^2Z_1Z_2 - v^3 - 2v^2X_1Z_2$;

• **Curve doubling formulas in projective coordinates ($R = 2P$)**

$$X_3 = 2hs,\ Y_3 = w(4B - h) - 8{Y_1}^2s^2,\ Z_3 = 8s^3, \tag{4}$$

where $w = a{Z_1}^2 + 3{X_1}^2, s = Y_1Z_1, B = X_1Y_1s, h = w^2 - 8B$.

The computation times are $t(\mathcal{P} + \mathcal{P}) = 12M + 2S$ and $t(2\mathcal{P}) = 7M + 5S$, where $\mathcal{P}$ means projective coordinates.

2.3 The Addition Formulas in Jacobian and Chudnovsky Jacobian coordinates

For Jacobian coordinates, we set $x = X/Z^2$ and $y = Y/Z^3$, giving the equation

$$E_J : Y^2 = X^3 + aXZ^4 + bZ^6.$$

The addition formulas in the Jacobian coordinates are the following. Let $P = (X_1, Y_1, Z_1)$, $Q = (X_2, Y_2, Z_2)$ and $P + Q = R = (X_3, Y_3, Z_3)$.

• **Curve addition formulas in Jacobian coordinates ($P \neq \pm Q$)**

$$X_3 = -H^3 - 2U_1H^2 + r^2,\ Y_3 = -S_1H^3 + r(U_1H^2 - X_3),\ Z_3 = Z_1Z_2H, \quad (5)$$

where $U_1 = X_1Z_2^2, U_2 = X_2Z_1^2, S_1 = Y_1Z_2^3, S_2 = Y_2Z_1^3, H = U_2 - U_1, r = S_2 - S_1$;

• **Curve doubling formulas in Jacobian coordinates ($R = 2P$)**

$$X_3 = T,\ Y_3 = -8{Y_1}^4 + M(S - T),\ Z_3 = 2Y_1Z_1, \quad (6)$$

where $S = 4X_1Y_1^2, M = 3X_1^2 + aZ_1^4, T = -2S + M^2$.

The computation times are $t(\mathcal{J} + \mathcal{J}) = 12M + 4S$ and $t(2\mathcal{J}) = 4M + 6S$, where $\mathcal{J}$ means Jacobian coordinates.

We see that Jacobian coordinates offer a faster doubling and a slower addition than projective coordinates. In order to make an addition faster, we should represent internally a Jacobian point as the quintuple (X, Y, Z, Z^2, Z^3) ([1]). This is called the Chudnovsky Jacobian coordinate and denoted by $\mathcal{J}^c$. The addition formulas in the Chudnovsky Jacobian coordinates are the following. Let $P = (X_1, Y_1, Z_1, Z_1^2, Z_1^3)$, $Q = (X_2, Y_2, Z_2, Z_2^2, Z_2^3)$ and $P + Q = R = (X_3, Y_3, Z_3, Z_3^2, Z_3^3)$.

• **Curve addition formulas in Chudnovsky Jacobian coordinates ($P \neq \pm Q$)**

$$X_3 = -H^3 - 2U_1H^2 + r^2, Y_3 = -S_1H^3 + r(U_1H^2 - X_3), Z_3 = Z_1Z_2H, Z_3^2 = Z_3^2, Z_3^3 = Z_3^3, \quad (7)$$

where $U_1 = X_1(Z_2^2), U_2 = X_2(Z_1^2), S_1 = Y_1(Z_2^3), S_2 = Y_2(Z_1^3), H = U_2 - U_1, r = S_2 - S_1$;

• **Curve doubling formulas in Chudnovsky Jacobian coordinates ($R = 2P$)**

$$X_3 = T,\ Y_3 = -8{Y_1}^4 + M(S - T),\ Z_3 = 2Y_1Z_1,\ Z_3^2 = Z_3^2,\ Z_3^3 = Z_3^3, \quad (8)$$

where $S = 4X_1Y_1^2, M = 3X_1^2 + a(Z_1^2)^2, T = -2S + M^2$.

The computation times are $t(\mathcal{J}^c + \mathcal{J}^c) = 11M + 3S$ and $t(2\mathcal{J}^c) = 5M + 6S$.

3 A new Strategy for Elliptic Curve Exponentiation

In this section, we investigate a new strategy for elliptic curve exponentiation. Up to now, since only one kind of coordinate system is used, it has been necessary that it should offer both an addition and a doubling with reasonable speed (not the fastest but not too slow) ([8,9,27,26,22,3]). The Chudnovsky Jacobian

coordinate system is a good example: it reduces the computation time of an addition by slightly increasing the doubling time, but this is still worthwhile since Jacobian coordinates have a rather faster doubling but slower addition times than projective coordinates.

On the contrary, here we further improve on the Jacobian coordinate system in order to offer even faster doublings, and there will be no loss in elliptic curve exponentiation since we are going to use a new strategy of mixed coordinate systems.

3.1 The Modified Jacobian Coordinates

Here we modify the Jacobian coordinates in order to obtain the fastest possible doublings. For this, we represent internally the Jacobian coordinates as a quadruple (X, Y, Z, aZ^4). We call this the modified Jacobian coordinate system, and denote it by $\mathcal{J}^m$. The addition formulas in the modified Jacobian coordinates are the following. Let $P = (X_1, Y_1, Z_1, aZ_1^4)$, $Q = (X_2, Y_2, Z_2, aZ_2^4)$ and $P + Q = R = (X_3, Y_3, Z_3, aZ_3^4)$.

• **Curve addition formulas in modified Jacobian coordinates ($P \neq \pm Q$)**

$$X_3 = -H^3 - 2U_1H^2 + r^2, Y_3 = -S_1H^3 + r(U_1H^2 - X_3), Z_3 = Z_1Z_2H, aZ_3^4 = aZ_3^4, \quad (9)$$

where $U_1 = X_1Z_2^2, U_2 = X_2Z_1^2, S_1 = Y_1Z_2^3, S_2 = Y_2Z_1^3, H = U_2 - U_1, r = S_2 - S_1$;

• **Curve doubling formulas in modified Jacobian coordinates ($R = 2P$)**

$$X_3 = T,\ Y_3 = M(S - T) - U,\ Z_3 = 2Y_1Z_1,\ aZ_3^4 = 2U(aZ_1^4), \quad (10)$$

where $S = 4X_1Y_1^2, U = 8{Y_1}^4, M = 3X_1^2 + (aZ_1^4), T = -2S + M^2$.

The computation times are then $t(\mathcal{J}^m + \mathcal{J}^m) = 13M + 6S$ and $t(2\mathcal{J}^m) = 4M + 4S$. Obviously a modified Jacobian coordinate doubling is faster than a projective, Jacobian or Chudnovsky Jacobian coordinate doubling. Furthermore it is faster than an affine coordinate doubling unless $I < 3.6M$ (S is set to $0.8M$), which seems extremely unlikely if p is larger than 100 bits, independently of the field of definition $\mathbb{F}_p$ and of the implementation of inversion.

3.2 Using Mixed Coordinates

It is evidently possible to mix different coordinates, i.e. to add two points where one is given in some coordinate system, and the other point is in some other coordinate system. We can also choose the coordinate system of the result. Since we have five different kinds of coordinate systems (represented by the symbols $\mathcal{A}$, $\mathcal{P}$, $\mathcal{J}$, $\mathcal{J}^c$, and $\mathcal{J}^m$), this gives a large number of possibilities. Generalizing slightly the notation used above, let us denote by $t(\mathcal{C}^1 + \mathcal{C}^2 = \mathcal{C}^3)$ the time for addition of points in coordinates $\mathcal{C}^1$ and $\mathcal{C}^2$ giving a result in coordinates $\mathcal{C}^3$, and by $t(2\mathcal{C}^1 = \mathcal{C}^2)$ the time for doubling a point in coordinates $\mathcal{C}^1$ giving a result in coordinates $\mathcal{C}^2$. Table 1 gives the computation times for additions and doublings in various coordinates (not all possible combinations are given, only the most useful ones).

A small discussion is necessary if we want to compare computation times. The ratio S/M is almost independent of the field of definition and of the implementation, and can be reasonably taken equal to 0.8. On the other hand, the ratio I/M deeply depends on the field of definition and on the implementation: it can be estimated to be between $9M$ and $30M$ in the case of p larger than 100 bits. From Table 1, we see that for a doubling using a fixed coordinate system, $\mathcal{J}^m$ is the best choice. On the other hand, for an addition using a fixed coordinate system, we cannot decide what is the best coordinate system independently of the relative speed of inversion: it will usually be $\mathcal{J}^c$, unless $I/M < 10.6$, in which case it will be $\mathcal{A}$.

doubling		addition	
operation	computation time	operation	computation time
$t(2\mathcal{P})$	$7M + 5S$	$t(\mathcal{J}^m + \mathcal{J}^m)$	$13M + 6S$
$t(2\mathcal{J}^c)$	$5M + 6S$	$t(\mathcal{J}^m + \mathcal{J}^c = \mathcal{J}^m)$	$12M + 5S$
$t(2\mathcal{J})$	$4M + 6S$	$t(\mathcal{J} + \mathcal{J}^c = \mathcal{J}^m)$	$12M + 5S$
$t(2\mathcal{J}^m = \mathcal{J}^c)$	$4M + 5S$	$t(\mathcal{J} + \mathcal{J})$	$12M + 4S$
$t(2\mathcal{J}^m)$	$4M + 4S$	$t(\mathcal{P} + \mathcal{P})$	$12M + 2S$
$t(2\mathcal{A} = \mathcal{J}^c)$	$3M + 5S$	$t(\mathcal{J}^c + \mathcal{J}^c = \mathcal{J}^m)$	$11M + 4S$
$t(2\mathcal{J}^m = \mathcal{J})$	$3M + 4S$	$t(\mathcal{J}^c + \mathcal{J}^c)$	$11M + 3S$
$t(2\mathcal{A} = \mathcal{J}^m)$	$3M + 4S$	$t(\mathcal{J}^c + \mathcal{J} = \mathcal{J})$	$11M + 3S$
$t(2\mathcal{A} = \mathcal{J})$	$2M + 4S$	$t(\mathcal{J}^c + \mathcal{J}^c = \mathcal{J})$	$10M + 2S$
–	–	$t(\mathcal{J} + \mathcal{A} = \mathcal{J}^m)$	$9M + 5S$
–	–	$t(\mathcal{J}^m + \mathcal{A} = \mathcal{J}^m)$	$9M + 5S$
–	–	$t(\mathcal{J}^c + \mathcal{A} = \mathcal{J}^m)$	$8M + 4S$
–	–	$t(\mathcal{J}^c + \mathcal{A} = \mathcal{J}^c)$	$8M + 3S$
–	–	$t(\mathcal{J} + \mathcal{A} = \mathcal{J})$	$8M + 3S$
–	–	$t(\mathcal{J}^m + \mathcal{A} = \mathcal{J})$	$8M + 3S$
–	–	$t(\mathcal{A} + \mathcal{A} = \mathcal{J}^m)$	$5M + 4S$
–	–	$t(\mathcal{A} + \mathcal{A} = \mathcal{J}^c)$	$5M + 3S$
$t(2\mathcal{A})$	$2M + 2S + I$	$t(\mathcal{A} + \mathcal{A})$	$2M + S + I$

Table 1. Computation amount of addition and doubling

3.3 Use of Mixed Coordinate Systems

Elliptic curve exponentiation kP usually combines the addition-subtraction method with the window method ([8,14,26,22,3]). We will set $n = \lfloor \log_2(k) \rfloor + 1$ (i.e. n is the number of bits of k), and we denote the width of a window by w. Some representations in signed binary are reported in [11,14,3]. Since our discussion does not depend on this representation, we restrict here k to be in the following representation,

$$k = 2^{k_0}(2^{k_1}(\cdots 2^{k_{v-1}}(2^{k_v}W[v] + W[v-1])\cdots) + W[0]) \tag{11}$$

where $W[i]$ is an odd integer in the range $-2^w + 1 \leq W[i] \leq 2^w - 1$ for all i, $W[v] > 0$, $k_0 \geq 0$ and $k_i \geq w+1$ for $i \geq 1$. This representation is easy to obtain inductively by looking at the bit pattern of k ([3]). Then kP can be computed using the following procedure: first precompute points $P_i = iP$ for odd integers i and $1 \leq i \leq 2^w - 1$, set $P_{-i} = -P_i$ for each i, and then repeat doublings and addition/subtractions with these precomputed points.

The first stage of computation, that is $2^{k_v} P_{W[v]}$, can be modified in order to reduce the computation amount as follows. In the case of $W[v] = 1$, k_v doublings are reduced to $(k_v - w)$ doublings and 1 addition by setting

$$2^{k_v} P_1 = 2^{k_v - w}(P_{2^w - 1} + P_1).$$

In the case of $W[v] = 3$, k_v doublings are reduced to $(k_v - w + 1)$ doublings and 1 addition by setting

$$2^{k_v} P_3 = 2^{k_v - w + 1}(P_{2^w - 1} + P_{2^{w-1}+1}).$$

Similar modifications can be made for all $W[v] < 2^{w-1}$, and one can show that the most significant doublings $2^{k_v} P_{W[v]}$ can be reduced by $(w^2+5w-2)/(2w+4)$ doublings minus $(w+1)/(w+2)$ additions on average.

Up to now, we have used a single coordinate system in all the procedure. Here we propose to mix different coordinate systems by dividing the computation into three parts: we will use the coordinate system $\mathcal{C}^1$ for repeated *main* doublings (i.e. $2^{k_i - 1} P'$), the coordinate system $\mathcal{C}^2$ for the result of a final doubling (i.e. $2(2^{k_i - 1} P')$) and the coordinate system $\mathcal{C}^3$ for the precomputed points, where P' is an intermediate point in the computation of kP. Summarizing, the computation of kP is done by repeating $2^{k_i} P' + P_{W[i-1]} = 2(2^{k_i - 1} P') + P_{W[i-1]}$, whose computation time is equal to

$$(k_i - 1)t(2\mathcal{C}^1) + t(2\mathcal{C}^1 = \mathcal{C}^2) + t(\mathcal{C}^2 + \mathcal{C}^3 = \mathcal{C}^1).$$

Let us now discuss suitable coordinate systems for $\mathcal{C}^1, \mathcal{C}^2$, and $\mathcal{C}^3$. Since doublings in $\mathcal{C}^1$ are repeated the most frequently, we should choose $\mathcal{C}^1$ such that $t(2\mathcal{C}^1)$ is the fastest, hence we set $\mathcal{C}^1$ equal to $\mathcal{J}^m$.

We now look at the coordinates suitable for $\mathcal{C}^2$ and $\mathcal{C}^3$. In this case, we must also consider the computation time necessary for constructing the table of precomputed points, which requires addition routines. For those, Table 1 says that

$$t(\mathcal{J}^c + \mathcal{J}^c) < t(\mathcal{A} + \mathcal{A}) \iff 9M + 2S < I, \tag{12}$$

where $t(\mathcal{J}^c + \mathcal{J}^c)$ is the fastest of all addition routines with no inversions and a fixed coordinate system. ¿From equation (12), the optimal coordinate system depends on the relative speed of inversion. Roughly speaking, when the relative speed of I to M is fast, we use affine coordinates as $\mathcal{C}^3$. When the relative speed of I to M is slow, we use Chudnovsky Jacobian coordinates as $\mathcal{C}^3$. In the next section, we first discuss each case generally, and then investigate the ratio of I to M in the case where k has 160-bits, 192-bits, and 224-bits.

3.4 Precomputed Points in Affine Coordinates

We assume here that we choose $\mathcal{C}^3$ to be $\mathcal{A}$. For $\mathcal{C}^2$, we search for the coordinate system such that $t(2\mathcal{J}^m = \mathcal{C}^2) + t(\mathcal{C}^2 + \mathcal{A} = \mathcal{J}^m)$ is as small as possible. From Table 1, we see that both $\mathcal{J}^c$ and $\mathcal{J}$ are suitable choices for $\mathcal{C}^2$. Thus, we choose the simplest system $\mathcal{J}$. To summarize, we set $(\mathcal{C}^1, \mathcal{C}^2, \mathcal{C}^3) = (\mathcal{J}^m, \mathcal{J}, \mathcal{A})$.

To compute the table of precomputed points P_i, we have two methods. We can compute it in the straightforward way, which requires a time of

$$2^{w-1}I + 2^w M + (2^{w-1} + 1)S. \tag{13}$$

Or we can use the well known Montgomery trick of simultaneous inversions: the inverses modulo p of m numbers can be computed in time $I + (3m-3)M$ (see for example [2], Algorithm 10.3.4). We compute $(2P)$, then $(3P, 4P)$, $(5P, 7P, 8P)$,... $((2^{w-2}+1)P, ..., (2^{w-1}-1)P, 2^{w-1}P)$, $((2^{w-1}+1)P, ..., (2^w - 1)P)$, giving a computation time of

$$wI + (5 \cdot 2^{w-1} + 2w - 10)M + (2^{w-1} + 2w - 3)S. \tag{14}$$

This will be almost always less than the time given in Equation (13) (for example, if $w = 4$, it will be the case if $I > 6.3M$). Furthermore the memory size necessary for constructing the table in Montgomery's trick is just the same as that in the above straightforward way. Thus, we will use this method for computing the table.

To compute the first stage of doublings, that is $2^{k_v} P_{W[v]}$, we use the modification discussed in Section 3.3: for example if $W[v] = 1$ we compute

$$t(2^{k_v} P_{W[v]}) = t(\mathcal{A} + \mathcal{A} = \mathcal{J}^m) + (k_v - w - 1)t(2\mathcal{J}^m) + t(2\mathcal{J}^m = \mathcal{J})$$

On the other hand, in the final stage, that is $2^{k_0}(P' + P_{W[0]})$, we use $t(\mathcal{J} + \mathcal{A} = \mathcal{J})$ instead of $t(\mathcal{J} + \mathcal{A} = \mathcal{J}^m)$ if $k_0 = 0$, and otherwise we use $t(2\mathcal{J}^m = \mathcal{J})$ instead of $t(2\mathcal{J}^m)$ as the final doubling.

We now discuss the total computation time. From Equations (11) and (14), the total computation time $T_w^1(n)$ including the time for constructing a table of P_i (i odd, $1 \le i \le 2^w - 1$) is equal to

$$T_w^1(n) = wI + (5 \cdot 2^{w-1} - 12 + \frac{11}{w+2} + 4u + 8v)M + (2^{w-1} - 6 + \frac{12}{w+2} + 4u + 5v)S, \tag{15}$$

where u is equal to $\sum_{i=0}^{v} k_i$. It is easily shown that the average interval between two windows is 2 bits ([3]). More precisely, one can show that we have approximately $u = n - w/2 + \theta$ and $v = (n - w/2 - \theta)/(w+2)$, where $\theta = 1/2 - 1/(w+2)$. Thus, if we set $n_1 = n - w/2$, $T_w^1(n)$ is approximately given by the following formula:

$$\begin{aligned} T_w^1(n) = wI + (5 \cdot 2^{w-1} - 7\theta - \frac{13}{2} + 4n_1 + \frac{8}{w+2}(n_1 - \theta))M \\ + (2^{w-1} - 8\theta + 4n_1 + \frac{5}{w+2}(n_1 - \theta))S. \end{aligned} \tag{16}$$

3.5 Precomputed Points in Chudnovsky Jacobian Coordinates

We assume here that we choose $\mathcal{C}^3$ to be $\mathcal{J}^c$. For $\mathcal{C}^2$, we search for the coordinate system such that $t(2\mathcal{J}^m = \mathcal{C}^2) + t(\mathcal{C}^2 + \mathcal{J}^c = \mathcal{J}^m)$ is as small as possible. ¿From Table 1, we see that both $\mathcal{J}^c$ and $\mathcal{J}$ are suitable choices for $\mathcal{C}^2$. Thus, we choose the simplest system $\mathcal{J}$. To summarize, we set $(\mathcal{C}^1, \mathcal{C}^2, \mathcal{C}^3) = (\mathcal{J}^m, \mathcal{J}, \mathcal{J}^c)$.

The computation time for constructing a table of P_i (i odd, $1 \le i \le 2^w - 1$) is

$$t(2\mathcal{A} = \mathcal{J}^c) + (2^{w-1} - 2)t(\mathcal{J}^c + \mathcal{J}^c) + t(\mathcal{A} + \mathcal{J}^c = \mathcal{J}^c) = (11 \cdot 2^{w-1} - 11)M + (3 \cdot 2^{w-1} + 2)S.$$

The first computation of $2P$ can be done instead using affine coordinates. In this case, the computation time for a table is

$$t(2\mathcal{A}) + (2^{w-1} - 2)t(\mathcal{A} + \mathcal{J}^c = \mathcal{J}^c) + t(\mathcal{A} + \mathcal{A} = \mathcal{J}^c) = I + (2^{w+2} - 9)M + (3 \cdot 2^{w-1} - 1)S.$$

However, this is never optimal if $224 \ge k \ge 100$ so we omit this case.

To compute the first stage of doublings, that is $2^{k_v} P_{W[v]}$, we use the modification discussed in Section 3.3: for example if $W[v] = 1$, we compute

$$t(2^{k_v} P_{W[v]}) = t(\mathcal{A} + \mathcal{J}^c = \mathcal{J}^m) + (k_v - w - 1)t(2\mathcal{J}^m) + t(2\mathcal{J}^m = \mathcal{J})$$

On the other hand, in the final stage of addition, that is $2^{k_0}(P' + P_{W[0]})$, we use $t(\mathcal{J} + \mathcal{J}^c = \mathcal{J})$ instead of $t(\mathcal{J} + \mathcal{J}^c = \mathcal{J}^m)$ if $k_0 = 0$, and otherwise we use $t(2\mathcal{J}^m = \mathcal{J})$ instead of $t(2\mathcal{J}^m)$ as the final doubling.

Here we discuss the total computation amount. We obtain a total computation time $T_w^2(n)$ including the time for constructing a table of P_i (i odd, $1 \le i \le 2^w - 1$), given by

$$\begin{aligned} T_w^2(n) = (11 \cdot 2^{w-1} - 2w - 7 - \frac{4}{w+2} + 4u + (11 - 3/2^{w-1})v)M \\ + (3 \cdot 2^{w-1} - 2w - 1 + \frac{12}{w+2} + 4u + 5v)S. \end{aligned} \tag{17}$$

Note that the term $3/2^{w-1}$ comes from the fact that although the P_i for $i > 1$ are in Chudnovsky Jacobian coordinates, P_1 is in affine coordinates so addition with P_1 is faster.

In the same way as in Section 3.4 with $n_1 = n - w/2$, we get approximately

$$\begin{aligned} T_w^2(n) = (11 \cdot 2^{w-1} - 2w + 8\theta - 9 + 4n_1 + \frac{11 - 3/2^{w-1}}{w+2}(n_1 - \theta))M \\ + (3 \cdot 2^{w-1} - 2w - 8\theta + 5 + 4n_1 + \frac{5}{w+2}(n_1 - \theta))S. \end{aligned} \tag{18}$$

4 Time Comparisons Depending on the Ratio I/M

4.1 The Case of $k = 160$ Bits

To fix ideas, we assume here that k has 160 bits and that $S = 0.8M$. In this case, the optimal value of w is equal to 4, u is approximately equal to 158.33, and v is approximately equal to 26.28. We obtain the following results:

1. $I < 30.5M$
   The optimal mixed coordinate system is as in Section 3.4: $(\mathcal{C}^1, \mathcal{C}^2, \mathcal{C}^3) = (\mathcal{J}^m, \mathcal{J}, \mathcal{A})$. In other words, we use affine coordinates for computing the table, modified Jacobian coordinates in the main doublings (i.e. $2^{k_i-1}P'$), and we compute the result of a final doubling (i.e. $2(2^{k_i-1}P')$) using Jacobian coordinates. The computation time is given by $T_4^1(160) = 4I + 1488.4M$ (Equation (15)).
2. $I > 30.5M$
   The optimal mixed coordinate system is as in Section 3.5: $(\mathcal{C}^1, \mathcal{C}^2, \mathcal{C}^3) = (\mathcal{J}^m, \mathcal{J}, \mathcal{J}^c)$. In other words, we use Chudnovsky Jacobian coordinates for computing the table, modified Jacobian coordinate in the main doublings (i.e. $2^{k_i-1}P'$), and we compute the result of a final doubling (i.e. $2(2^{k_i-1}P')$) using Jacobian coordinates. The computation time is given by $T_4^2(160) = 1610.2M$ (Equation (17)).

Let us compare our new method using mixed coordinate systems with the traditional method using a single coordinate system. If we use Jacobian coordinates and addition-subtraction with the window method as above, the computation time for elliptic curve exponentiation is approximately $1869.1M$, which is the best known among projective, Jacobian or Chudnovsky Jacobian coordinate systems. If we use our new modified Jacobian coordinates instead of the Jacobian coordinates, the computation time of elliptic curve exponentiation is improved to approximately $1708.2M$. On the other hand, affine coordinates would be worse. We thus see that the use of modified Jacobian coordinate $\mathcal{J}^m$, together with a clever use of mixed coordinate systems, with a computation time of at most $1610.2M$, gives a very significant improvement.

4.2 The Case of $k = 192$ Bits

We assume here that k has 192 bits and that $S = 0.8M$. In this case, the optimal value of w is equal to 4, u is approximately equal to 190.33, and v is approximately equal to 31.61. We obtain the following results:

1. $I < 33.9M$
   The optimal mixed coordinate system is as in Section 3.4: $(\mathcal{C}^1, \mathcal{C}^2, \mathcal{C}^3) = (\mathcal{J}^m, \mathcal{J}, \mathcal{A})$. The computation time is given by $T_4^1(192) = 4I + 1782.8M$ (Equation (15)).
2. $I > 33.9M$
   The optimal mixed coordinate system is as in Section 3.5: $(\mathcal{C}^1, \mathcal{C}^2, \mathcal{C}^3) = (\mathcal{J}^m, \mathcal{J}, \mathcal{J}^c)$. The computation time is given by $T_4^2(192) = 1918.5M$ (Equation (17)).

Let us compare our new method using mixed coordinate systems with the traditional method using a single coordinate system. If we use Jacobian coordinates and addition-subtraction with the window method as above, the computation time for elliptic curve exponentiation is approximately $2228.6M$. If we use our new modified Jacobian coordinates instead of the Jacobian coordinates, the

computation time of elliptic curve exponentiation is improved to approximately $2030.3M$. We thus see that the use of modified Jacobian coordinate $\mathcal{J}^m$, together with a clever use of mixed coordinate systems, with a computation time of at most $1918.5M$, gives a very significant improvement.

4.3 The Case of $k = 224$ Bits

We assume here that k has 224 bits and that $S = 0.8M$. In this case, the optimal value of w is equal to 4 except for the mixed coordinate system of $(\mathcal{C}^1, \mathcal{C}^2, \mathcal{C}^3) = (\mathcal{J}^m, \mathcal{J}, \mathcal{A})$ in Section 3.4. In the case of $(\mathcal{C}^1, \mathcal{C}^2, \mathcal{C}^3) = (\mathcal{J}^m, \mathcal{J}, \mathcal{A})$, the optimal value of w is determined by the relative speed of I to M: if $I > 17.7M$, then $w = 4$, otherwise $w = 5$. Here we assume that w is equal to 4 since $I > 17.7M$ in our implementation. Then u is approximately equal to 222.33, and v is approximately equal to 36.94. We obtain the following results:

1. $I < 37.4M$
   The optimal mixed coordinate system is as in Section 3.4: $(\mathcal{C}^1, \mathcal{C}^2, \mathcal{C}^3) = (\mathcal{J}^m, \mathcal{J}, \mathcal{A})$. The computation time is given by $T_4^1(224) = 4I + 2077.2M$ (Equation (15)).
2. $I > 37.4M$
   The optimal mixed coordinate system is as in Section 3.5: $(\mathcal{C}^1, \mathcal{C}^2, \mathcal{C}^3) = (\mathcal{J}^m, \mathcal{J}, \mathcal{J}^c)$. The computation time is given by $T_4^2(224) = 2226.8M$ (Equation (17)).

Let us compare our new method using mixed coordinate systems with the traditional method using a single coordinate system. If we use Jacobian coordinates and addition-subtraction with the window method as above, the computation time for elliptic curve exponentiation is approximately $2588.1M$. If we use our new modified Jacobian coordinates instead of the Jacobian coordinates, the computation time of elliptic curve exponentiation is improved to approximately $2352.5M$. We thus see that the use of modified Jacobian coordinate $\mathcal{J}^m$, together with a clever use of mixed coordinate systems, with a computation time of at most $2226.8M$, gives a very significant improvement.

5 Implementation

5.1 Elliptic Curves

Elliptic curves $E/\mathbb{F}_p$ with order divisible by a prime of at least 160-bits are secure if the trace of E ([24]) is equal to neither 0 nor 1 ([15,23]). Here we implement two elliptic curves with 160-bit, 192-bit and 224-bit key size.
Elliptic curve E_1 (160-bit key size)

- a field of definition $\mathbb{F}_{p_1}$: $p_1 = 2^{160} - 2933$

- an elliptic curve E_1: $y^2 = x^3 + a_1x + b_1$, where
  $a_1 = 260304558782498478937947576884532782721650322528$
  $b_1 = 173536372521665652625298384589688521814433548352$,
  $\#E_1(\mathbb{F}_{p_1}) = 3 \cdot 5 \cdot 157 \cdot q_1$, where q_1 is a prime
  $q_1 = 62059517508743223702916552938161116922491333 7$
- a point P_1: $(x_1, y_1) \in E_1(\mathbb{F}_{p_1})$ with order q_1, where
  $x_1 = 1274\ 10436\ 88184\ 50369\ 80533\ 90568\ 22189\ 38631\ 36302\ 30379$
  $y_1 =\ 572\ 21905\ 85804\ 38390\ 03353\ 99912\ 01426\ 54787\ 42865\ 52166$

Elliptic curve E_2 (192-bit key size)

- a field of definition $\mathbb{F}_{p_2}$: $p_2 = 2^{192} - 3345$
- an elliptic curve E_2: $y^2 = x^3 + a_2x + b_2$, where
  $a_2 = 429731083554301521680038274056331893792536022079263215959 7$
  $b_2 = 286487389036201014453358849370887929195024014719508810639 8$,
  $\#E_2(\mathbb{F}_{p_2}) = 5^2 \cdot q_2$, where q_2 is a prime
  $q_2 = 25108406941546723055343157692204617886491928148401033301 9$
- a point P_2: $(x_2, y_2) \in E_2(\mathbb{F}_{p_2})$ with order q_2, where
  $x_2 =\ 523\ 46903\ 86238\ 76826\ 11193\ 52046\ 88411\ 23614\ 71708\ 15234$
  $y_2 =\ 23\ 91039\ 55423\ 03027\ 66388\ 76206\ 81604\ 62176\ 43806\ 46680$

Elliptic curve E_3 (224-bit key size)

- a field of definition $\mathbb{F}_{p_3}$: $p_3 = 2^{224} - 1025$
- an elliptic curve E_3: $y^2 = x^3 + a_3x + b_3$, where
  $a_3 = 124045765741249697014423371828958597533618029996105045924187297616 88$
  $b_3 = 970358006201711339548311860274918061351689428195337859245658699100 2$,
  $\#E_3(\mathbb{F}_{p_3}) = 69 \cdot q_3$, where q_3 is a prime
  $q_3 = 39072386474131362021256543604376277771282351667343224473457378206 1$
- a point P_3: $(x_3, y_3) \in E_3(\mathbb{F}_{p_3})$ with order q_3, where
  $x_3 = 249765308100512709270375849840091210710938852696633500117319681085 24$
  $y_3 = 841302677393235943446120820595866096728965993663923313219342782811 3$

5.2 The Running Time

We present the running times of elliptic curve exponentiation over our 160-bit and 192-bit field of definition using our methods. We compare each strategy of Section 3.4 with the traditional method using a single coordinate. Our modulo arithmetic uses the GNU MP Library GMP ([7]), so as to make easy comparisons possible, since GMP may well be the most popular multiprecision library. The platform is an UltraSPARC (143 MHz/Solaris 2.4). Table 2 shows the running times. We see that our new strategy gives a very significant improvement.

6 Conclusion

In this paper, we have introduced modified Jacobian coordinates $\mathcal{J}^m$, which offer the fastest doubling of all known coordinate systems. The new modified Jacobian

	160 bit key	192 bit key	224 bit key
field operations (μsec)			
160/192/224 bit addition	0.59	0.64	0.71
160/192/224 bit multiplication	6.50	8.93	12.00
160/192/224 bit squaring	5.35	7.22	9.01
reduction (320/384/448 $\rightarrow$ 160/192/224 bit)	2.37	2.77	2.62
160/192/224 bit inverse	166	213	261
elliptic curve operations (msec)			
addition ($t(\mathcal{A}+\mathcal{A})$)	0.203	0.257	0.314
addition ($t(\mathcal{J}^c+\mathcal{J}^c)$)	0.130	0.171	0.215
addition ($t(\mathcal{J}+\mathcal{J})$)	0.144	0.191	0.239
doubling ($t(2\mathcal{J}^m)$)	0.079	0.103	0.127
doubling ($t(2\mathcal{J})$)	0.094	0.122	0.148
elliptic curve exponentiation (msec)			
mixed coordinates (case 1)	16.17	24.93	35.73
mixed coordinates (case 2)	16.66	25.54	37.53
single coordinate (Jacobian coordinate)	18.66	28.79	41.86
single coordinate (projective coordinate)	20.33	30.17	44.79

Table 2. Times for elliptic curve operations (UltraSPARC)

coordinates improve the computation time of 160-bit elliptic curve exponentiation to approximately $1708.2M$ even with the traditional method which uses a single coordinate system: the use of modified Jacobian coordinates reduces the computation time of the best known method by 9%.

Furthermore we have proposed a new method using mixed coordinate systems, which divides elliptic curve exponentiation into three parts, and in each part we choose the optimal system. For these choices we have presented three cases according to the relative speed of inversion to multiplication over $\mathbb{F}_p$. We have seen that the use of modified Jacobian coordinates together with a clever use of mixed coordinate systems, having a computation time of at most $1610.2M$, gives a very significant improvement. Our new strategy with modified Jacobian coordinates reduces the computation time of the best known method by more than 14%.

References

1. D. V. Chudnovsky and G. V. Chudnovsky "Sequences of numbers generated by addition in formal groups and new primality and factorization tests" *Advances in Applied Math.*, **7** (1986), 385–434.
2. H. Cohen, "A course in computational algebraic number theory", Graduate Texts in Math. **138**, Springer-Verlag, 1993, Third corrected printing, 1996.
3. H. Cohen, A. Miyaji and T. Ono, "Efficient elliptic curve exponentiation", *Advances in Cryptology-Proceedings of ICICS'97*, Lecture Notes in Computer Science, **1334** (1997), Springer-Verlag, 282–290.
4. G. Frey and H. G. Rück, "A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves", *Mathematics of computation*, **62**(1994), 865-874.
5. "Proposed federal information processing standard for digital signature standard (DSS)", *Federal Register*, **56** No. 169, 30 Aug 1991, 42980–42982.
6. T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms", *IEEE Trans. Inform. Theory*, **IT-31** (1985), 469–472.
7. Torbjorn Granlund, The GNU MP LIBRARY, version 2.0.2, June 1996. ftp://prep.ai.mit.edu/pub/gnu/gmp-2.0.2.tar.gz
8. Jorge Guajardo and Christof Paar "Efficient algorithms for elliptic curve cryptosystems", *Advances in Cryptology-Proceedings of Crypto'97*, Lecture Notes in Computer Science, **1294** (1997), Springer-Verlag, 342–356.
9. G. Harper, A. Menezes and S. Vanstone, "Public-key cryptosystems with very small key lengths", *Advances in Cryptology-Proceedings of Eurocrypt'92*, Lecture Notes in Computer Science, **658** (1993), Springer-Verlag, 163–173.
10. *IEEE P1363 Working Draft*, June 16, 1998.
11. D. E. Knuth, *The art of computer programming, vol. 2, Seminumerical Algorithms*, 2nd ed., Addison-Wesley, Reading, Mass. 1981.
12. N. Koblitz, "Elliptic curve cryptosystems", *Mathematics of Computation*, **48** (1987), 203–209.
13. N. Koblitz, "CM-curves with good cryptographic properties", *Advances in Cryptology-Proceedings of CRYPTO'91*, Lecture Notes in Computer Science, **576** (1992), Springer-Verlag, 279–287.
14. K. Koyama and Y. Tsuruoka, "Speeding up elliptic cryptosystems by using a signed binary window method", *Advances in Cryptology-Proceedings of Crypto'92*, Lecture Notes in Computer Science, **740** (1993), Springer-Verlag, 345–357.
15. A. Menezes, T. Okamoto and S. Vanstone, "Reducing elliptic curve logarithms to logarithms in a finite field", *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing* (1991), 80–89.
16. V. S. Miller, "Use of elliptic curves in cryptography", *Advances in Cryptology-Proceedings of Crypto'85*, Lecture Notes in Computer Science, **218** (1986), Springer-Verlag, 417–426.
17. F. Morain and J. Olivos, "Speeding up the computations on an elliptic curve using addition-subtraction chains", Theoretical Informatics and Applications **24** No.6 (1990), 531–544.
18. S. C. Pohlig and M. E. Hellman, "An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance", *IEEE Trans. Inf. Theory*, **IT-24** (1978), 106–110.
19. J. Pollard, "Monte Carlo methods for index computation (mod p)", *Mathematics of Computation*, **32** (1978), 918–924.

20. R. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM*, **21** No. 2 (1978), 120–126.
21. T. Satoh and K. Araki "Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves", *Commentarii Math. Univ. St. Pauli.*, vol. **47** (1998), 81-92.
22. R. Schroeppel, H. Orman, S. O'Malley and O. Spatscheck, "Fast key exchange with elliptic curve systems", *Advances in Cryptology-Proceedings of Crypto'95*, Lecture Notes in Computer Science, **963** (1995), Springer-Verlag, 43–56.
23. I. A. Semaev "Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p", *Mathematics of computation*, **67** (1998), 353-356.
24. J. H. Silverman, *The Arithmetic of Elliptic Curves*, GTM **106**, Springer-Verlag, New York, 1986.
25. N. P. Smart "The discrete logarithm problem on elliptic curves of trace one", to appear in J. Cryptology.
26. Jerome A. Solinas "An improved algorithm for arithmetic on a family of elliptic curves", *Advances in Cryptology-Proceedings of Crypto'97*, Lecture Notes in Computer Science, **1294** (1997), Springer-Verlag, 357–371.
27. E. D. Win, A. Bosselaers and S. Vandenberghe "A fast software implementation for arithmetic operations in GF(2^n)", *Advances in Cryptology-Proceedings of Asiacrypt'95*, Lecture Notes in Computer Science, **1163** (1996), Springer-Verlag, 65–76.