A Method for Distinguishing the Two Candidate Elliptic Curves in the Complex Multiplication Method

Yasuyuki Nogami, Mayumi Obara, and Yoshitaka Morikawa

In this paper, we particularly deal with no F_p-rational two-torsion elliptic curves, where F_p is the prime field of the characteristic p. First we introduce a shift product-based polynomial transform. Then, we show that the parities of (#E − 1)/2 and (#E′− 1)/2 are reciprocal to each other, where #E and #E′ are the orders of the two candidate curves obtained at the last step of complex multiplication (CM)-based algorithm. Based on this property, we propose a method to check the parity by using the shift product-based polynomial transform. For a 160 bits prime number as the characteristic, the proposed method carries out the parity check 25 or more times faster than the conventional checking method when 4 divides the characteristic minus 1. Finally, this paper shows that the proposed method can make CM-based algorithm that looks up a table of precomputed class polynomials more than 10 percent faster.

Keywords: CM method, irreducible cubic polynomial, quadratic power residue/non-residue.

Manuscript received Mar. 15, 2006: revised Sept. 11, 2006.

A part of this paper was presented at the 7th International Conference on Information Security and Cryptology (ICISC2004) held in Seoul, Korea [1].

Yasuyuki Nogami (phone: + 81 86 251 8128, email: nogami@cne.okayama-u.ac.jp), Mayumi Obara (email: obara@trans.cne.okayama-u.ac.jp), and Yoshitaka Morikawa (email: morikawa@cne.okayama-u.ac.jp) are with the Department of Communication Network Engineering, Okayama University, Okayama, Japan.

I. Introduction

In recent years, elliptic curve cryptography (ECC) has received much attention. For ECC, some attacks have been proposed [2], [3]. From the viewpoints of security and efficiency of ECC, it is said that a prime order elliptic curve is suitable for ECC. It should be noted that the prime order must not be the characteristic of the definition prime field itself. In this case, the elliptic curve is especially called an *anomalous curve* and is not suitable for cryptographic use [2]. If the elliptic curve $E(F_q)$ is a prime order curve, every rational point on the curve except for the infinity point is a generator in $E(F_q)$. It is helpful for the implementation of ECC. For security reasons, the order must have a prime factor larger than 160 bits for which it is best that the order itself is a large and secure prime number. As compared to generating prime numbers, generating prime order elliptic curves takes a lot of computation time. Therefore, a fast algorithm is needed. In order to systematically generate prime order elliptic curves, several algorithms have been proposed. We can roughly classify them into two types; one adopts a certain order counting algorithm [4], [5] and the other adopts the complex multiplication (CM) method [6], [7]. This paper is related to the CM method, and particularly deals with no F_p–rational two-torsion elliptic curves defined over the prime field F_p, where p is the characteristic.

Using a CM-based algorithm, we can generate an elliptic curve whose order is a certain prime number. There are several versions of CM-based algorithms depending on which parameters one wants to enforce [8]. For example, first input a prime number as the characteristic p, then determine a small discriminant D, and then construct a certain prime order. After that, construct a class polynomial corresponding to the discriminant D and then obtain the j-invariant from the class polynomial. According to the *j*-invariant, we have two candidate elliptic curves at the last step in the CM-based algorithm; one of these two curves has the constructed prime order. For these two candidate curves, we must check which curve has the constructed prime order. In practice, it is checked by picking a random rational point on the curve and then calculating a scalar multiplication with the rational point. This paper proposes a method to check it faster. In CM-based algorithms, the class polynomial computation is the most time-consuming operation [8]. Therefore, most of the conventional improvements for CM-based algorithms are given for this class polynomial computation; however, recently Atkin and others [9] have proposed an algorithm that prepares a table of precomputed class polynomials. Using this table, the algorithm generates a prime order elliptic curve within several seconds. As an application of the proposed method, this paper deals with a CM-based prime order curve generation algorithm that looks up a table of precomputed class polynomials. Depending on the inputs, there are several different CM-based algorithms. It should be noted that the CM-based algorithm shown in this paper is just one of them and the proposed method can be applied for every version.

In this paper, we introduce shift product-based polynomial transform (SPPT). When the degree of the polynomial is 3, SPPT is carried out by a square root calculation and a modular reduction modulo polynomial over the prime field F_p. Especially in cases in which the order of elliptic curve is an odd number such as a prime number, the parities of $(\#E-1)/2$ and $(\#E'-1)/2$ are reciprocal to each other, where $\#E$ and $\#E'$ are the orders of the two candidate curves obtained at the last step in a CM-based algorithm. Based on this property, we propose a method to check the parity by using SPPT. This parity check method does not need a scalar multiplication for a rational point; however, it needs a square root calculation, a modular reduction modulo polynomial, and a quadratic power residue check instead. Especially when 4 divides the characteristic minus 1, the proposed method does not need the square root calculation and modular reduction modulo polynomial but only needs a quadratic power residue check, where this check is carried out by an exponentiation in the prime field. From the experimental results, we show that the proposed method is superior to the conventional method. For a 160 bits prime number as the characteristic, the proposed method carries out the parity check 25 or more times faster than the conventional method when 4 divides the characteristic minus 1. Finally, this paper shows that the proposed method can make a CM-based algorithm that looks up a table of precomputed class polynomials more than 10 percent faster. In the experiments, we also applied the Montgomery operation with the Montgomery ladder technique [10] for checking which curve has the constructed order. Since the Montgomery operation does not need the *y*-coordinate of the rational point, it does not need any square root calculations. In this paper, the efficiency of the Montgomery operation is also discussed.

In this paper, we deal with a finite field F_q whose characteristic p is an odd prime number larger than 3. The prime field is denoted by F_p; $X \mid Y$ and $X \nmid Y$ mean that X divides Y and does not divide Y, respectively; and $X \| Y$ means that X divides Y but X^2 does not divide Y. Without any additional explanation, polynomials in this paper are monic. This paper especially focuses on CM-based algorithms for generating prime order elliptic curves. This paper is the extended version of our previous works [1] and [11]. Our previous works only compared the computation times between a scalar multiplication and a parity check; therefore, the advantage and efficiency of the proposed method were not clearly shown. In this paper, using a practical example of a CM-based algorithm for generating prime order elliptic curves defined over F_p, we clarify the advantage and examine the efficiency of the proposed method.

II. Fundamentals

In this section, we go over the fundamentals of an elliptic curve, quadratic residue/non-residue, and a CM-based algorithm for generating prime order elliptic curves.

1. Defining Equation

When the characteristic of F_q is not equal to 2 or 3, an elliptic curve over F_q is generally defined by

$$E(x,y)=x^3+ax+b-y^2=0,\ \ a,b\in F_q. \tag{1}$$

The solutions (x, y) to (1) and the point at infinity denoted by O are called F_q–rational points when the coordinates of x and y lie in F_q. On the elliptic curve, F_q–rational points form an additive Abelian group. In this paper, we denote this group and its order by $E(F_q)$ and $\#E(F_q)$, respectively. The following parameter t is called the trace of elliptic curve $E(x, y) = 0$:

$$t=q+1-\#E(F_q). \tag{2}$$

2. No F_q–Rational Two-Torsion Curve

The necessary and sufficient condition for an elliptic curve to have no F_q–rational two-torsion points is that $E(x,0)$ given from its defining equation is irreducible over F_q. An F_q– rational two-torsion point P means that $2P = O$, where O plays a role of the unity in the Abelian group $E(F_q)$. It is necessary for a prime order elliptic curve to have no F_q-rational two-torsion points, and our previous work [5] uses this necessary condition for generating prime order curves. In what follows, we consider no F_q-rational two-torsion curves.

3. The Order of Elliptic Curve

For an arbitrary element $i \in F_q$, if $E(i, 0)$ is a quadratic residue in F_q, then the following two rational points on the curve (1) are given:

$$\left(i,\ \pm\sqrt{E(i,0)}\right), \tag{3}$$

where $E(i, 0) \neq 0$ because $E(x, y) = 0$ is a no F_q-rational two-torsion curve. Therefore, let N be the number of quadratic residues in the following set:

$$\left\{c \mid c = E(i,0), \forall i \in F_q\right\}, \tag{4}$$

the order $\#E(F_q)$ is given by

$$\#E(F_q) = 2N + 1, \tag{5}$$

where 1 in the right-hand side of the above equation corresponds to the point at infinity O. From (5), N is written as

$$N = \frac{\#E(F_q) - 1}{2}. \tag{6}$$

4. Quadratic Residue/Non-Residue

For a non-zero element $c \in F_q$, we can check whether c is a quadratic residue (QR) or quadratic non-residue (QNR) in F_q as

$$c^{(q-1)/2} = \begin{cases} 1 & \text{when c is a QR} \\ -1 & \text{when c is a QNR}. \end{cases} \tag{7}$$

The product of two non-zero QRs and that of two QNRs become QRs in F_q. On the other hand, the product of a QR and a QNR becomes a QNR in F_q.

5. CM-Based Algorithm for Prime Order Curves

We refer to the following algorithm [8] as CM-based prime order elliptic curve generation algorithm:

Input. Characteristic p

Output. Prime order curve $E(x, y) = 0$ over F_p

Step 1. Find a smallest D along with t such that $4p = t^2 + Ds^2,\ D, t, s \in Z$.

Step 2. Check whether any of the candidate orders

$$\#E_{\pm}(F_q) = p + 1 \pm t \tag{8}$$

is a prime number. If both are not prime numbers, then find another D along with t at step 1.

Step 3. Construct the class polynomial $H_D(x)$.

Step 4. Find a root $j \in F_p$ of $H_D(x)$, then set

$$k = j/(1728 - j) \in F_p. \tag{9}$$

Step 5. Check whether the order of

$$E(x, y) = x^3 + 3kx + 2k - y^2 = 0 \tag{10}$$

is $\#E_-(F_p)$ or $\#E_+(F_p)$ by preparing a random rational point P on (10) and checking whether or not $\#E_+(F_p)P = O$. If the order is the constructed prime order, output (10); otherwise, output its twisted curve.

The calculation of step 3 is too time-consuming; therefore, several improvements have been proposed [6]. As compared to step 3, step 5 is carried out much faster. Atkin and others [9] canceled the time-consuming calculation by preparing a table of precomputed class polynomials. By using this table, the algorithm generates a prime order elliptic curve within several seconds. This paper proposes an improvement for step 5. The above CM-based algorithm is just the non-optimized version in which the characteristic p is input. We can consider several versions depending on the input such as the order $\#E$ and the discriminant D. Our proposed method can be applied to every version.

III. Main Idea

In what follows, we consider the prime field F_p as the definition field. We first define SPPT [12]. Then, we show a relation between SPPT and cubic trinomial $E(x, 0)$.

1. SPPT

Let us consider an irreducible polynomial $f(x)$ over F_p of degree m written as

$$f(x) = \sum_{i=0}^{m} f_i x^i,\ \ f_i \in F_p, f_m = 1. \tag{11}$$

As shown in Appendix A, if $f_{m-1} = 0$ and $p \nmid m$, from $f(x)$ we can uniquely determine an irreducible polynomial $\tilde{f}(x)$ that satisfies

$$\tilde{f}(x^p - x) = \prod_{i=0}^{p-1} f(x+i), \tag{12-1}$$

$$\tilde{f}(x)=\sum_{i=0}^{m}\tilde{f}_i x^i,\ \tilde{f}_i\in F_p,\tilde{f}_m=1,\tilde{f}_{m-1}=0. \qquad (12\text{-}2)$$

The two irreducible polynomials $f(x)$ and $\tilde{f}(x)$ are in a one-to-one relation (see Appendix A). We define the SPPT as

$$\text{SPPT}: f(x)\to\tilde{f}(x). \qquad (13)$$

In what follows, we denote the irreducible polynomial after SPPT with ˜ as shown in (13).

2. SPPT for Defining Equation $E(x, y) = 0$

For no F_p– rational two-torsion curve $E(x, y) = 0$, let us consider the following SPPT:

$$\text{SPPT}: E(x,0)\to\tilde{E}(x,0). \qquad (14)$$

Noting that $E(x, 0)$ is an irreducible cubic polynomial over F_p in this paper, as shown in (12-1), we can uniquely determine a cubic irreducible polynomial $\tilde{E}(x,0)$ that satisfies

$$\tilde{E}(x^p-x,0)=\prod_{i=0}^{p-1}E(x+i,0). \qquad (15)$$

By substituting $x = 0$ into (15), we have

$$\tilde{E}(0,0)=\prod_{i=0}^{p-1}E(i,0). \qquad (16)$$

From (16), we have

$$\tilde{E}(0,0)^{(p-1)/2}=\prod_{i=0}^{p-1}E(i,0)^{(p-1)/2}. \qquad (17)$$

By using the number N defined in section II.3 and substituting (7), we obtain

$$\tilde{E}(0,0)^{(p-1)/2}=(-1)^{p-N}=-(-1)^N. \qquad (18)$$

Note that this paper deals with an odd prime number larger than 3 as the characteristic p. Thus we have the following property:

Property 1. Let $\tilde{E}(0,0)$ be the constant term of a cubic irreducible polynomial $\tilde{E}(x,0)$ over F_p that satisfies (15), and let N be the number of QRs in the set (4). Then, N is an odd number if and only if $\tilde{E}(0,0)$ is a QR in F_p.

3. Implementation of SPPT

From property 1 and (18), if we have the constant term $\tilde{E}(0,0)$, then we can check the parity of N by testing whether or not $\tilde{E}(0,0)$ is a QR in F_p. In order to obtain the constant term $\tilde{E}(0,0)$, we consider how to determine the cubic irreducible polynomial $\tilde{E}(x,0)$ over F_p that satisfies (15). In other words, we consider how to implement SPPT defined in section III.1.

Let ω and τ be zeros of $f(x)$ and $\tilde{f}(x)$ introduced in section III.1, respectively, then we have $\tau=\omega^p-\omega$ from (12-1). Therefore, $\tilde{f}(x)$ is the minimal polynomial of $\omega^p-\omega$ with respect to F_p, where we should note that ω and τ belong to F_{p^m} but not to F_p. As shown in Appendix B, from $f(x)$ given by (11) with $m = 3$, we obtain the following two candidates of $\tilde{f}(x)$:

$$\tilde{f}_{\pm}(x)=x^3+3f_1x\pm\sqrt{D(f)}, \qquad (19\text{-}1)$$

$$D(f)=-(4f_1^3+27f_0^2), \qquad (19\text{-}2)$$

where $D(f)$ is the discriminant of $f(x)$. Since $f(x)$ is irreducible over F_p, let three zeros of $f(x)$ be ω, ω^p, and ω^{p^2}, and its discriminant $D(f)$ is given as [13]

$$\begin{aligned}D(f)&=(\omega-\omega^p)^2(\omega^p-\omega^{p^2})^2(\omega-\omega^{p^2})^2\\&=\left\{(\omega-\omega^p)^{1+p+p^2}\right\}^2=c^2,\end{aligned} \qquad (20)$$

where $c=(\omega-\omega^p)^{1+p+p^2}\in F_p$.

Therefore, since the discriminant $D(f)$ becomes a QR in F_p, we can calculate the square roots of $D(f)$ in F_p by a square root computation [8]. Since $f(x)$ and $\tilde{f}(x)$ satisfy (12), we can distinguish them as

$$\tilde{f}(x)=\begin{cases}\tilde{f}_+(x) & \text{when } f(x) \text{ divides } \tilde{f}_+(x^p-x),\\ \tilde{f}_-(x) & \text{when } f(x) \text{ divides } \tilde{f}_-(x^p-x).\end{cases} \qquad (21)$$

As shown in (21), the test whether $f(x)$ divides $\tilde{f}_+(x^p-x)$ or $\tilde{f}_-(x^p-x)$ requires a modular reduction modulo polynomial over F_p.

IV. Distinguishing the Two Candidate Curves

In this section, we consider the CM-based algorithm [8] as an application of property 1. We will not describe the CM method itself in detail. In what follows, the definition field of the elliptic curves is the prime field F_p.

1. Two Candidate Elliptic Curves

As stated in section I, several algorithms for generating prime order elliptic curves have been proposed [5], [6]. In the CM-based algorithm introduced in section II.5, the output is the defining equation $E(x, y) = 0$ whose order is a prime number. In what follows, we suppose that the j-invariant is not 0 or 1728.

Let us consider that X is the constructed order written as

$$X = p + 1 - t. \tag{22}$$

Using the CM-based algorithm, we have a pair of the characteristic p and j-invariant. In other words, we obtain

$$E(x,y) = x^3 + 3kx + 2k - y^2 = 0 \tag{23}$$

where $k = j/(1728 - j),\ k, j \in F_p$.

Let $\#E(F_p)$ be the order of the curve defined by (23), then it is possible for the order $\#E(F_p)$ to be the following two numbers:

$$\#E_{\pm}(F_p) = p + 1 \pm t. \tag{24}$$

From only the j-invariant, we cannot distinguish whether $\#E(F_p)$ is $\#E_{-}(F_p)$ or $\#E_{+}(F_p)$. For this problem, as shown in step 5, we randomly pick an F_p–rational point P on the curve (23), then test whether or not $XP = O$ by a scalar multiplication. If $\#E(F_p)$ is not X, then output the twist of $E(x, y) = 0$ as

$$E'(x,y) = x^3 + 3kc^2x + 2kc^3 - y^2 = 0, \tag{25}$$

where c is a QNR in F_p. In order to solve this problem without the scalar multiplication, our proposed method uses the following property:

Property 2. The parity of $(\#E_{+}(F_p) - 1)/2$ and that of $(\#E_{-}(F_p) - 1)/2$ are reciprocal to each other. It should be noted that this paper is especially dealing with no F_p– rational two-torsion curves; accordingly, the order of the curve is an odd number.

2. Proposed Method

In order to distinguish whether the order $\#E(F_p)$ is $\#E_{+}(F_p)$ or $\#E_{-}(F_p)$, this paper proposes the following step 5′ that uses SPPT.

Step 5′. For the irreducible cubic polynomial

$$E(x,0) = x^3 + 3kx + 2k, \tag{26}$$

calculate $\tilde{E}(x,0)$ that satisfies (15) by using SPPT as introduced in section III.3 and then calculate $T \in F_p$ as

$$T = \tilde{E}(0,0)^{(p-1)/2}. \tag{27}$$

Let $\#E(F_p)$ be the order of $E(x, y)=x^3 + 3kx + 2k - y^2 = 0$, then we have

$$N = \frac{\#E(F_p) - 1}{2} = \begin{cases} \text{odd} & \text{when } T = 1, \\ \text{even} & \text{when } T = -1. \end{cases} \tag{28}$$

Based on this relation, check whether $\#E(F_p)$ is $\#E_{+}(F_p)$ or $\#E_{-}(F_p)$ by using property 2. If the order $\#E(F_p)$ is the constructed prime order, then output $E(x, y) = 0$; otherwise, output the twisted curve $E'(x, y) = 0$.

Step 5′ can be applied instead of step 5 in the conventional CM-based algorithm. In step 5′, we calculate an irreducible cubic polynomial $\tilde{E}(x,0)$ from $E(x, 0)$ by using SPPT as introduced in section III.3.

According to property 1 and property 2, we can check the parity of N from $\tilde{E}(0,0)$ and therefore we can distinguish whether the order $\#E(F_p)$ is $\#E_{+}(F_p)$ or $\#E_{-}(F_p)$, where N is written as (6). If the order is not the constructed order, the twisted elliptic curve (25) has the constructed order.

Step 5 in the conventional CM-based algorithm introduced in section II.5 needs a scalar multiplication for a rational point as introduced in section IV.1. On the other hand, step 5′ only depends on whether or not $\tilde{E}(0,0)$ is a QR in F_p. As shown in (19) and (21), step 5′ needs a square root calculation, a modular reduction modulo polynomial for SPPT, and a quadratic power residue check (27). In other words, we need to distinguish whether $\tilde{f}(x)$ given by SPPT is $\tilde{f}_{+}(x)$ or $\tilde{f}_{-}(x)$. However, when 4 divides $p - 1$, we can easily check (27), that is whether or not $\tilde{E}(0,0)$ is a QR in F_p, by the following calculation:

$$\begin{aligned} \tilde{E}(0,0)^{(p-1)/2} &= \left(\pm\sqrt{-(108k^3 + 108k^2)}\right)^{(p-1)/2} \\ &= \left(-108k^3 - 108k^2\right)^{(p-1)/4}. \end{aligned} \tag{29}$$

Therefore, the result of (29) does not depend on the sign ±. Consequently, when 4 divides $p - 1$, the parity of N can be easily checked without a square root calculation and modular reduction modulo polynomial for SPPT. As compared to a scalar multiplication for a rational point, the calculation of the right-hand side of (29) can be carried out much faster because it only needs an exponentiation in the prime field F_p.

When 4 divides $p - 1$, the following step 5″ can be used instead of Step 5 in the CM-based algorithm introduced in section II.5.

Step 5″. For the irreducible cubic polynomial

$$E(x,0) = x^3 + 3kx + 2k, \tag{30}$$

calculate the following $T \in F_p$:

$$T = \left(-108k^3 - 108k^2\right)^{(p-1)/4}. \tag{31}$$

Let $\#E(F_p)$ be the order of $E(x,y)=x^3+3kx+2k-y^2=0$, then we have

$$N = \frac{\#E(F_p)-1}{2} = \begin{cases} \text{odd} & \text{when } T = 1, \\ \text{even} & \text{when } T = -1. \end{cases} \quad (32)$$

Based on this relation, check whether $\#E(F_p)$ is $E_+(F_p)$ or $\#E_-(F_p)$ by using property 2. If the order $\#E(F_p)$ is the constructed prime order, then output $E(x, y) = 0$; otherwise, output the twisted curve $E'(x, y) = 0$.

Step 5″ needs an exponentiation (31) only. We should note that (31) does not need any square root calculations. Half of the odd prime numbers satisfy the condition that 4 divides $p - 1$. That is to say that we can apply step 5″ to half of the prime numbers.

3. Cost Evaluation

The conventional method, that is step 5, needs not only to calculate a scalar multiplication but also to prepare a random rational point on the curve. The preparation of a random rational point needs a square root calculation. In this section, we consider only the calculation cost of the scalar multiplication of step 5, for instance. It is noted that we do not take additions and subtractions in the definition field into account.

Let I, M, and S be the calculation cost of an inversion, a multiplication, and a square in the definition field F_p, respectively. Let ω be the number of non-zero coefficients of non-adjacent form (NAF) representation of the scalar [14]. Using *affine coordinates* [10], the calculation cost C_5 of a scalar multiplication is given by

$$C_5 \approx (I+2M+2S)\log_2 p+(I+2M+S)\,\omega. \quad (33)$$

We can also apply the generalized Montgomery operation [15] with the Montgomery ladder technique [10], [16] instead of this scalar multiplication (see Appendix F).

On the other hand, step 5′ needs squares of degree 2 polynomial, multiplications between two polynomials of degree 2, and modulo operations for degree 3 or 4 polynomial by $f(x)$ as the modular polynomial over F_p. The number of polynomial squares is about $\log_2 p$ times, and let the number of polynomial multiplications be ω'. By using Karatsuba method [17], a square of polynomial of degree 2 needs 6 squares and a multiplication between two polynomials of degree 2 needs 6 multiplications in the definition field F_p. Since a modular reduction modulo polynomial needs 0 or 1 or 2 multiplications in F_p (see Appendix D), let us consider that 2 multiplications in F_p are constantly needed. Then we calculate an exponentiation (27). Consequently, the calculation cost $C_{5'}$ of step 5′ becomes

$$C_{5'} \approx (6S+2M)\log_2 p+(6M+2M)\omega'+S\log_2 p+M\omega'' \quad (34)$$

where ω'' is the Hamming weight of the binary representation of the exponent $(p-1)/2$ shown in (27).

Step 5″ needs an exponentiation (31). Therefore, the calculation cost $C_{5''}$ of step 5″ is given by

$$C_{5''} \approx S\log_2 p+M\omega''' \quad (35)$$

where ω''' is the Hamming weight of the binary representation of the exponent $(p-1)/4$ shown in (31).

We used Pentium 4 (3.8 GHz) and NTL (A Library for doing Number Theory) [18] in the following experiments. NTL has the following properties:

$$S \approx 0.8M, \quad I \approx 10M. \quad (36)$$

Therefore, the calculation costs become

$$C_5 \approx (13.6 \log_2 p+12.8\omega)M, \quad (37\text{-}1)$$

$$C_{5'} \approx (7.6 \log_2 p+8\omega'+\omega'')M \quad (37\text{-}2)$$

$$C_{5''} \approx (0.8 \log_2 p+\omega''')M \quad (37\text{-}3)$$

When $\log_2 p=160$, $\omega=30$, $\omega'=9$, $\omega''=9$, and $\omega'''=9$, for example, these costs become

$$C_5 \approx 2560M,\ C_{5'} \approx 1297M,\ C_{5''} \approx 137M. \quad (38)$$

In the following experiments, ω is about 30 as shown in Table 2 and Table 3, in addition we adopted prime numbers in the form $2^i + j$, $j < 2^{18}$ as the characteristic p, therefore we consider that ω', ω'', and ω''' are about 9. We can estimate that step 5′ and step 5″ are about twice and 19 times faster than step 5, respectively. It is noted that C_5 and $C_{5'}$ do not contain the calculation cost of square root calculations.

V. Experimental Result

This section first introduces a CM-based prime order curve generation algorithm in which the conventional step 5, the proposed step 5′, or step 5″ is applied. Using this algorithm, some experimental results for generating 100 prime order curves of 160, 180, or 200 bits are shown. Based on the results, we compare step 5, step 5′, and step 5″ and then we discuss the efficiency of the proposed parity check method.

1. CM-Based Algorithm

Referring to Savas and others [7], we prepared a table of class polynomials by using the software library developed by Konstantinou and others [19] for the following discriminants D:

$$\begin{aligned} D = \{&11,19,35,43,51,59,67,83,91,107,115,123,\\ &131,139,155,163,179,187,195,203,211,219,\\ &227,235,251,259,267,283,291,299,307,323,\\ &331,339,347,355,371,379,395,403,411,419,\\ &427,435,443,451,467,483,491,499\}. \end{aligned} \quad (39)$$

In addition, as shown in Table 1, we prepared start prime numbers as the characteristic p.

As shown in (39), we restricted the discriminant D to be less than or equal to 499 and also to satisfy $D \equiv 3 \pmod 8$. The former is because the degree of $H_D(x)$ becomes larger as the discriminant D becomes larger. For the latter we refer to Morain and others [20]. In the following algorithm, the maximum of the discriminant D is particularly related to step 0, step 1b, and step 4. After several experiments, we decided to limit the discriminant to a maximum of 499.

Table 1. Start prime numbers.

	Size of the start prime (bits)		
	160	180	200
$4 \nmid (p-1)$	$2^{160}+7$	$2^{180}+15$	$2^{200}+235$
$4 \mid (p-1)$	$2^{160}+357$	$2^{180}+193$	$2^{200}+697$

We considered the following CM-based algorithm for generating prime order elliptic curves:

Input. A start prime number of 160, 180, or 200 bits

Output. Prime order elliptic curves

Preparation. Consider the discriminant D shown in (39), and prepare the following:

- A table of class polynomials $H_D(x)$.
- For each D, a table of QRs mod$D$.

Step 0. Search the prime number p next to the previous (or the start) prime number as the characteristic of F_p and initialize the discriminant $D = 11$.

Step 1a. Check the following conditions:

- $3 \nmid (D-1)$ when $3 \mid (p-1)$.
- p is a QR mod$D$.

If these two conditions are satisfied, then go to the next step; otherwise, go to step 6.

Step 1b. Using Cornacchia's algorithm, find a pair of t and s such that $4p = t^2 + Ds^2,\ t, s \in Z$. If the pair of t and s is obtained, then go to the next step; otherwise, go to step 6.

Step 2. Check whether any of the orders

$$\# E_{\pm}(F_p) = p + 1 \pm t \tag{40}$$

is a prime number. If both are not prime numbers, then go to step 6. Otherwise, go to the next step.

Step 3. Corresponding to the discriminant D, find the class polynomial $H_D(x)$ from the table.

Step 4. Find a root $j \in F_p$ of $H_D(x)$, then set

$$k = j/(1728 - j) \in F_p. \tag{41}$$

If k cannot be determined, go to step 6; otherwise, go to the next step.

Step 5, 5′, or 5″. Check whether the order of

$$E(x, y) = x^3 + 3kx + 2k - y^2 = 0 \tag{42}$$

is $\#E_-(F_p)$ or $\#E_+(F_p)$. Then, corresponding to the result, output $E(x, y) = 0$ or the twisted curve $E'(x, y) = 0$. Then, return to step 0.

Step 6. If $D < 499$, set the next D and then go to step 1a. Otherwise, return to step 0.

□

At step 0 in this algorithm, the Miller witness algorithm included in the NTL is used for searching for prime numbers, where the number of trials is 5. In this case, the error rate of the primality check becomes about 2^{-5}; however, since most of the prime number candidates are rejected through the following steps, we do not check so strictly at step 0. At step 1a, we check the two conditions. The first check is easy and the second check can be done by looking up the table of precomputed QRs mod D. These conditions are given for $4p = t^2 + Ds^2$ to be solvable at Step 1b. To be more detailed, the second condition and $D \equiv 3 \bmod 4$ guarantee that $-D$ is a QR mod p (see Appendix C). For step 1b, the Cornacchia's algorithm is applied [10]. For the primality check at step 2, the Miller witness algorithm included in the NTL is used, where the number of trials is 20. In this case, the error rate of the primality check becomes about 2^{-20}, which is low enough. At step 3, this paper uses the Hilbert polynomial but not the Weber polynomial [8]. At step 4, we calculate

$$F(x) = \gcd(x^p - x, H_D(x)), \tag{43-1}$$

then we calculate the following greatest common divisor $G(x)$ by changing the parameter s from 0 to 10 until the degree of $G(x)$ becomes 1:

when $4 \nmid (p-1)$

$$G(x) = \gcd((x+s)^{(p-1)/2} \pm 1,\ F(x)),\ s \in F_p, \tag{43-2}$$

when $4 \mid (p-1)$

$$G(x) = \gcd((x+s)^{(p-1)/4} \pm 1,\ F(x)),\ s \in F_p. \tag{43-3}$$

If the degree of $G(x)$ is equal to 1, a zero of $H_D(x)$ in F_p can be easily given as the zero of $G(x)$; however, if we cannot find any zeros of $H_D(x)$ in F_p by (43), then go to step 6. The reason we consider the parameter s up to 10 is described in sectionV.2.A. At step 5, we check whether the obtained curve has the constructed prime order or not. As previously described, the conventional step 5 calculates a scalar multiplication of a rational point on the curve. As fast scalar multiplication algorithms, the binary method, NAF representation method, and window method are well known [10]. For a scalar multiplication at step 5, we used the NAF representation method. In general, the NAF representation method is faster than the binary method and the window method is even faster than the NAF representation method; however, the window method is not so efficient when the elliptic curve and its order vary every time (see Appendix E). For easy understanding and comparison, we adopted the NAF representation method. The window method can be applied not only for step 5 but also step 5′, step 5″. When $4 \nmid (p-1)$, we compared the conventional step 5 and the proposed step 5′. When $4 \mid (p-1)$, we compared the conventional step 5 and the proposed step 5″. After that, in order to be sure, we strictly checked the primality of p by the Miller witness algorithm of which the number of trials was 15 because at step 0 we did not check the primality so strictly.

2. Experimental Results

Tables 2 and 3 show the experimental result of generating 100 prime order elliptic curves defined over prime fields by the CM-based algorithm shown in sectionV.1 when $4 \nmid (p-1)$ and $4 \mid (p-1)$, respectively. We used Pentium 4 (3.8 GHz), C language with NTL.

A. Explanation of Tables 2 and 3

For example, when the size of the characteristic p was equal to 160, 59515 integers were input to step 0. Through step 0, 1082 prime numbers were obtained. By using these prime numbers and changing the discriminant D, 51309 pairs of the characteristic p and the discriminant D were input to step 1a. Then, 14027 pairs of p and D were input to step 1b. In other words, 14027 pairs of p and D satisfied the two conditions at step 1a. Then, 2627 pairs of integers $p+1\pm t$ were input to step 2. In other words, the equation $4p = t^2 + Ds^2$ was solved with respect to t and s by Cornacchia's algorithm for 2627 pairs of p and D. Then, 103 prime orders and pairs of p and D were input to step 3 though step 3 only uses the discriminant D. In other words, among 2627 pairs of integers $p+1\pm t$, 103 integers lead to prime numbers. At step 3, the average of the discriminant D was 225.91 and that of the degree of $H_D(x)$ was 2.29. Then, 100 roots were each obtained from 100 among 103 $H_D(x)$'s at step 4; accordingly, 200 prime order elliptic curve candidates were obtained. In other words, among 103 $H_D(x)$'s, 3 $H_D(x)$'s could not be solved over F_p by changing the parameter s from 0 up to 10. At step 4, there were 39 $H_D(x)$'s of degree 1 and the average of the shift s by which a root of $H_D(x)$ was found was 1.68. From such experimental results, we set the maximum of the parameter $s \in F_p$ to 10. Finally, by step 5 or step 5′, 100 prime order elliptic curves were determined from 200 candidates. For scalar multiplications at step 5, we used NAF representation. The average of the number of non-zero coefficients of the NAF representations was 27.75. For generating 100 prime order curves, it took 9.67 seconds with step 5 or 9.34 seconds with step 5′, where the difference comes only from the difference between step 5 and step 5′.

B. Comparison between Step 5, Step 5′, and Step 5″

From the experimental results shown in Tables 2 and 3, we find that

- step 5′ is about twice as fast as step 5;
- step 5″ is 25 or more times faster than step 5.

It can be roughly understood that these results come from the calculation costs (37); however, the latter result is not so close to the estimation. From Tables 2 and 3, we find that the calculation times of step 1b and step 5 when $4 \mid (p-1)$ are slower than those when $4 \nmid (p-1)$, respectively. For example, when the size of the characteristic p is equal to 160, the computation time at step 5 takes 6.51×10^{-3} and 8.58×10^{-3} seconds on average when $4 \nmid (p-1)$ and $4 \mid (p-1)$, respectively. This is due to square root calculations in F_p at step 1b and step 5. It is known that a square root calculation when $4 \mid (p-1)$ becomes a little more complicated than that when $4 \nmid (p-1)$. For a non-zero QR $a \in F_p$ when $4 \nmid (p-1)$, we can easily calculate its square roots as $\pm a^{(p+1)/4}$; however, when $4 \mid (p-1)$, we must calculate its square roots by some square root calculation algorithm such as the Tonelli-Shanks algorithm which is briefly introduced in [21]. The experimental results for which the Montgomery operation with the Montgomery ladder is applied are shown in Appendix F. It is noted that this paper is particularly dealing with no F_p-rational two-torsion elliptic curves; therefore, the curves cannot be transformed to Montgomery form [10].

3. Efficiency of the Proposed Method

For example, when $4 \nmid (p-1)$ and the size of the characteristic p is 160 bits, generating 100 prime order elliptic curves with step 5 takes 9.67 seconds in total. The breakdown of the total computation time is also shown in the table, for example, it took 6.51×10^{-1} seconds for step 5. When we use the proposed step 5′ instead of the conventional step 5, it took 3.28×10^{-1} seconds for step 5′; accordingly, the total computation time became 9.34 seconds. In this case, it can be said that step 5′ contributes about 3.5% improvement. As shown in Table 2, when $4 \nmid (p-1)$, step 5′ contributes about 3% improvement on average for CM-based prime order curve generation algorithm that looks up a table of precomputed class

Table 2. Computation time for generating 100 prime order curves when $4 \nmid (p - 1)$.

Size of p (bits)	Step	# of inputs	Computation time (s)	Remarks	
160	Step 0	59515	3.79	# of prime numbers as p	1082
	Step 1a	51309	3.58×10^{-2}		
	Step 1b	14027	3.29		
	Step 2	2627	8.00×10^{-1}		
	Step 3	103	5.23×10^{-2}	Average of discriminant D	225.91
				Average of deg($H_D(x)$)	2.29
	Step 4	103	9.39×10^{-1}	# of cases that deg($H_D(x)$) = 1	39
				Average of shift s	1.68
	Step 5/ Step 5′	100	6.51×10^{-1}/ 3.28×10^{-1}	Average of Hamming weight ω †	27.75
	Total	-	9.67 / 9.34	Improvement	3.5%
180	Step 0	80331	5.75	# of prime numbers as p	1282
	Step 1a	60924	4.78×10^{-2}		
	Step 1b	16885	4.70		
	Step 2	3135	9.72×10^{-1}		
	Step 3	102	3.85×10^{-2}	Average of discriminant D	187.47
				Average of deg($H_D(x)$)	2.05
	Step 4	102	8.35×10^{-1}	# of cases that deg($H_D(x)$) = 1	45
				Average of shift s	1.22
	Step 5/ Step 5′	100	7.93×10^{-1} / 3.82×10^{-1}	Average of Hamming weight ω †	31.65
	Total	-	1.33×10 / 1.29×10	Improvement	3.0%
200	Step 0	86469	7.93	# of prime numbers as p	1254
	Step 1a	59839	4.80×10^{-2}		
	Step 1b	16677	6.03		
	Step 2	3156	1.32		
	Step 3	100	4.94×10^{-2}	Average of discriminant D	214.84
				Average of deg($H_D(x)$)	2.08
	Step 4	100	7.12×10^{-1}	# of cases that deg($H_D(x)$) = 1	42
				Average of shift s	1.11
	Step 5/ Step 5′	100	9.92×10^{-1} / 4.99×10^{-1}	Average of Hamming weight ω †	34.98
	Total	-	1.73×10 / 1.68×10	Improvement	2.9%

* CPU: Pentium 4, 3.80 GHz, we used NTL [18] and prepared class polynomials by [19].
† Hamming weight ω means the number of non-zero coefficients of NAF representation of #E.

polynomials. We must say that the contribution of step 5′ is quite small.

On the other hand, when $4 \mid (p - 1)$, as in the case when $4 \nmid (p - 1)$, step 0 and step 1b are the major calculations; however, the third-largest step is step 5. This is because a square root computation in step 5 when $4 \mid (p - 1)$ becomes a little more time-consuming than that when $4 \nmid (p - 1)$. Since the proposed step 5″ is 25 or more times faster than the conventional step 5, the proposed step 5″ contributes about 7% improvement as shown in 3 on average. The improvement is not so great; however, the calculation of step 5″ is quite simple as compared to that of step 5. In addition, step 5″ can be applied for half of odd prime numbers as the characteristic because half of odd prime numbers satisfy $4 \mid (p - 1)$. The efficiency of the use of (43-3) can be seen in the numbers of inputs at step 4 shown in Table 2 and Table 3. For example, when the size of the characteristic p is equal to 180, these numbers are 102 and 100, respectively. This means that the greatest common divisor calculation (43-2) fails to output $G(x)$ of degree 1 twice; however, the detailed greatest common divisor calculation (43-3)

Table 3. Computation time for generating 100 prime order curves when 4 | (p - 1).

Size of p (bits)	Step	# of inputs	Computation time (s)	Remarks	
160	Step 0	56487	3.55	# of prime numbers as p	999
	Step 1a	47252	3.66 × 10^{-2}		
	Step 1b	12877	4.54		
	Step 2	2326	7.69 ×10^{-1}		
	Step 3	100	5.30 ×10^{-2}	Average of discriminant D	230.44
				Average of deg($H_D(x)$)	2.34
	Step 4	100	5.21 ×10^{-1}	# of cases that deg($H_D(x)$) = 1	32
				Average of shift s	1.12
	Step 5/ Step 5′	100	8.58 ×10^{-1}/ 3.16 ×10^{-2}	Average of Hamming weight ω †	28.72
	Total	-	1.05 × 10 / 9.63	Improvement	8.3%
180	Step 0	81510	5.68	# of prime numbers as p	1203
	Step 1a	56985	4.61 ×10^{-2}		
	Step 1b	15794	6.41		
	Step 2	2866	9.48 ×10^{-1}		
	Step 3	100	3.68 ×10^{-2}	Average of discriminant D	184.12
				Average of deg($H_D(x)$)	2.11
	Step 4	100	5.22 ×10^{-1}	# of cases that deg($H_D(x)$) = 1	41
				Average of shift s	0.97
	Step 5/ Step 5′	100	1.02 / 3.61 ×10^{-2}	Average of Hamming weight ω †	31.97
	Total	-	1.49 × 10 / 1.39 × 10	Improvement	6.7%
200	Step 0	82731	7.60	# of prime numbers as p	1201
	Step 1a	57359	4.56 ×10^{-2}		
	Step 1b	15680	7.95		
	Step 2	2871	1.26		
	Step 3	100	5.12 ×10^{-2}	Average of discriminant D	231.72
				Average of deg($H_D(x)$)	2.34
	Step 4	100	8.10 ×10^{-1}	# of cases that deg($H_D(x)$) = 1	31
				Average of shift s	1.12
	Step 5/ Step 5′	100	1.29 / 4.64 ×10^{-2}	Average of Hamming weight ω †	34.96
	Total	-	1.93 × 10 / 1.80 × 10	Improvement	6.7%

* CPU: Pentium 4, 3.80 GHz, we used NTL [18] and prepared class polynomials by [19].

† Hamming weight ω means the number of non-zero coefficients of NAF representation of #E.

Table 4. Average computation time of each step for generating 100 prime order curves when 4 | (p - 1) and the size of the characteristic p is 160 bits.

Size of p (bits)	Step	(A) # of inputs	(B) Computation time (s)	(B) divided by (A) (s)
Add Replace Delete	Step 0	56487	3.55	6.28 ×10^{-5}
	Step 1a	47252	3.66 × 10^{-2}	7.62 ×10^{-7}
	Step 1b	12877	4.54	3.53 ×10^{-4}
	Step 2	2326	7.69 ×10^{-1}	3.31 ×10^{-4}
	Step 3	100	5.30 ×10^{-2}	5.30 ×10^{-4}
	Step 4	100	5.21 ×10^{-1}	5.21 ×10^{-3}
	Step 5/Step 5″	100	8.58 ×10^{-1} / 3.16 ×10^{-2}	8.58 ×10-3 / 3.16 ×10^{-4}

* CPU: Pentium 4, 3.80 GHz, we used NTL [18] and prepared class polynomials by [19].

Table 5. Computation time for generating 100 curves whose order has a large prime factor more than 160 bits when 4 | (p - 1) and the size of the characteristic p is 170 bits.

Size of p (bits)	Step	# of inputs	Computation time (s)	Remarks	
170	Step 0	21970	1.40	# of prime numbers as p	382
	Step 1a	16285	1.13×10^{-2}		
	Step 1b	4427	1.74		
	Step 2	819	7.1×10^{-1}		
	Step 3	100	5.05 ×10-2	Average of discriminant D	219.00
				Average of deg($H_D(x)$)	2.45
	Step 4	100	6.48×10^{-1}	# of cases that deg($H_D(x)$) = 1	27
				Average of shift s	1.16
	Step 5/Step 5″	100	8.45×10^{-1} / 3.36×10^{-2}	Average of Hamming weight ω †	29.88
	Total	-	5.45 / 4.63	Improvement	15.0%

* CPU: Pentium 4, 3.80 GHz, we used NTL [18] and prepared class polynomials by [19].
† Hamming weight ω means the number of non-zero coefficients of NAF representation of #E.

does not. This efficiency contributes to the calculation time at step 4 and the average of shift s in (43-3).

Let us consider the computation time divided by the number of inputs for each step. Table 4 shows the results. From the results, we find that a scalar multiplication at step 5 is more time-consuming than a primality check at step 2, a Cornacchia's algorithm calculation at step 1b, finding a zero of $H_D(x)$ in F_p at step 4, and so on. From this viewpoint, we can conclude that the contribution of the proposed step 5″ is considerable. If we can efficiently restrict the pairs of characteristic p and discriminant D in some way, the inputs for step 0 and step 1b will be decreased, accordingly the percentage of the improvement by the proposed step 5″ will become greater. For example, Table 5 shows an experimental result when the order of elliptic curve is restricted so as to have a large prime factor more than 160 bits but not so as to be a prime number, where the start prime number is $2^{170}+49$. The inputs for step 0 and step 1b were decreased; accordingly, the improvement became 15.0%.

VI. Conclusion

In this paper, we have particularly dealt with no F_p–rational two-torsion elliptic curves, first we have defined a shift product-based polynomial transform (SPPT) that was carried out by a square root computation and a modular reduction modulo polynomial. Then, it was shown that the parities of $(\#E-1)/2$ and $(\#E'-1)/2$ are reciprocal to each other, where $\#E$ and $\#E'$ were the orders of the two candidate curves obtained at the last step in a CM-based algorithm. Based on this property, we proposed a method to check the parity by using SPPT. This parity check method does not need a scalar multiplication for a rational point; it needs a square root calculation, a modular reduction modulo polynomial, and a quadratic power residue check instead. Especially when 4 divides the characteristic minus 1, the proposed method does not need the square root calculation or a modular reduction modulo polynomial. It only needs an exponentiation. For a 160 bit prime number as the characteristic, the proposed method could carry out the parity check 25 or more times faster than the conventional method when 4 divides the characteristic minus 1. Finally, this paper showed that the proposed method could make a CM-based algorithm that looks up a table of precomputed class polynomials several percent faster.

Appendix A. Relation between $f(x)$ and $\tilde{f}(x)$

In what follows, it is important that the characteristic p does not divide the extension degree m. This paper is dealing with the case that p is an odd prime number larger than 3 and m is equal to 3; therefore, this condition is always satisfied.

Let us consider an irreducible polynomial $f(x)$ of degree m over F_p whose zero ω satisfies $\mathrm{Tr}(\omega)=0$ and $\omega \in F_{p^m}$. Then, we have

$$\begin{aligned}\prod_{i=0}^{p-1} f(x+i) &= \prod_{i=0}^{p-1}(x-\omega+i)(x-\omega^p+i)\cdots(x-\omega^{p^{m-1}}+i)\\ &= \prod_{i=0}^{m-1}(x-\omega^{p^i})(x-\omega^{p^i}+1)\cdots(x-\omega^{p^i}+(p-1))\\ &= \prod_{i=0}^{m-1}(x^p-x-(\omega^p-\omega)^{p^i}). \qquad \text{(A1)}\end{aligned}$$

Let $\tilde{f}(x)$ be $\prod_{i=0}^{m-1}(x-(\omega^p-\omega)^{p^i})$, in what follows we show that $\tilde{f}(x)$ is an irreducible polynomial of degree m over F_p. Suppose that $\tau = \omega^p - \omega$ belongs to a proper subfield F_{p^r}, $r \mid m$ of F_{p^m}, then we have

$$\begin{aligned}&\tau+\tau^p+\cdots+\tau^{p^{r-1}}\\ &=(\omega^p-\omega)+(\omega^{p^2}-\omega^p)+\cdots+(\omega^{p^r}-\omega^{p^{r-1}}) \quad \text{(A2)}\\ &=\omega^{p^r}-\omega.\end{aligned}$$

Since ω does not belongs to the proper subfield, F_{p^r}, $\omega^{p^r}-\omega$ is not equal to 0. On the other hand, $\tau+\tau^p+\cdots+\tau^{p^{r-1}}$ is the sum of all conjugates of τ with respect to F_p; therefore, the sum becomes an element in F_p. Let $c \in F_p$ be the element, we have

$$\begin{aligned}&c+c^{p^r}+c^{p^{2r}}+\cdots+c^{p^{m'r}}\\ &=(\omega^{p^r}-\omega)+(\omega^{p^r}-\omega)^{p^r}+(\omega^{p^r}-\omega)^{p^{2r}} \quad \text{(A3)}\\ &+\cdots+(\omega^{p^r}-\omega)^{p^{m'r}},\end{aligned}$$

then we have

$$(m'+1)c=\omega^{p^m}-\omega=0, \quad \text{(A4)}$$

where $m'=m/r-1$. As described at the beginning of this section, this paper deals with the case that the characteristic p does not divide the extension degree m; therefore, c must be 0 because $p \nmid (m'+1)$. Consequently, τ does not belong to the proper subfield F_{p^r}; therefore, $\tilde{f}(x)$ is an irreducible polynomial of degree m over F_p whose zero $\tau = \omega^p - \omega$ satisfies

$$\omega^{p^m}-\omega=\tau+\tau^p+\tau^{p^2}+\cdots+\tau^{p^{m-1}}=\mathrm{Tr}(\tau)=-\tilde{f}_{m-1}=0, \quad \text{(A5)}$$

where $\mathrm{Tr}(x)=x+x^p+x^{p^2}+\cdots+x^{p^{m-1}}$.

Therefore, we can make an irreducible polynomial $\tilde{f}(x)$ of degree m over F_p satisfying (12) from an irreducible polynomial $f(x)$.

Next, let us determine an irreducible polynomial $f(x)$ of degree m over F_p whose zero γ satisfies $\mathrm{Tr}(\gamma) = 0$ from $\tilde{f}(x)$. Let τ and γ be zeros of $\tilde{f}(x)$ and $\tilde{f}(x^p-x)$, respectively, then we have

$$\tau=\gamma^p-\gamma, \quad \text{(A6-1)}$$

$$\tau^p=\gamma^{p^2}-\gamma^p, \quad \text{(A6-2)}$$

$$\tau^{p^2}=\gamma^{p^3}-\gamma^{p^2}, \quad \text{(A6-3)}$$

$$\vdots$$

$$\tau^{p^{m-1}}=\gamma^{p^m}-\gamma^{p^{m-1}}. \quad \text{(A6-4)}$$

It is noted that $\tilde{f}(x)$ satisfying (12) is irreducible over F_p. By adding these equations, we have

$$\gamma^{p^m}-\gamma=\tau+\tau^p+\tau^{p^2}+\cdots+\tau^{p^{m-1}} \quad \text{(A7)}$$

$$=\mathrm{Tr}(\tau)=-\tilde{f}_{m-1}=0. \quad \text{(A8)}$$

Noting that τ belongs to F_{p^m} but not to its proper subfield, we find that γ also belongs to F_{p^m} but not to its proper subfield because γ satisfies (A6-1). In addition, we find that γ, $\gamma+1$, $\gamma+2, \cdots$, $\gamma+(p-1)$ are also zeros of $\tilde{f}(x^p-x)$. In the case that the characteristic p does not divide the extension degree m, there exists an element that satisfies $\mathrm{Tr}(x) = 0$ among these zeros [22]. Supposing that $\omega = \gamma + a$ satisfies $\mathrm{Tr}(\omega) = 0$ with a certain element $a \in F_p$ and denoting the minimal polynomial of ω by $f(x)$, we have $f(x)$ that satisfies (12) and $f_{m-1} = 0$, where f_{m-1} is the coefficient of x^{m-1} of $f(x)$. Therefore, we can make an irreducible polynomial $f(x)$ of degree m over F_p satisfying (12) from an irreducible polynomial $\tilde{f}(x)$. Consequently, we have SPPT and it is shown that $f(x)$ and $\tilde{f}(x)$ hold the one to one relation.

Appendix B. Proof of (19)

Based on the relation $\tau = \omega^p-\omega$, we have the following equations:

$$\tilde{f}_2=-(\tau+\tau^p+\tau^{p^2})=0, \quad \text{(A9-1)}$$

$$\tilde{f}_1=\tau\tau^p+\tau^p\tau^{p^2}+\tau^{p^2}\tau=3f_1, \quad \text{(A9-2)}$$

$$\tilde{f}_0=-\tau\tau^p\tau^{p^2}=A-B, \quad \text{(A9-3)}$$

where

$$A=\omega\omega^{2p}+\omega^p\omega^{2p^2}+\omega^{p^2}\omega^2, \quad \text{(A9-4)}$$

$$B=\omega^2\omega^p+\omega^{2p}\omega^{p^2}+\omega^{2p^2}\omega. \quad \text{(A9-5)}$$

Since $A+B=3f_0$ and $AB=f_1^3+9f_0^2$, we obtain

$$\tilde{f}_0=A-B=\sqrt{D(f)} \text{ or } -\sqrt{D(f)}, \quad \text{(A10-1)}$$

where $D(f)=-(4f_1^3+27f_0^2)$. (A10-2)

As shown above, we can easily obtain two candidates of 0 without any calculations in the extension field F_{p^3} to which ω belongs.

Appendix C. –D is a QR mod p

Let $D \equiv 3 \bmod 4$, p be a QR mod D, and D be factorized as a product of r prime numbers:

$$D = d_1 d_2 d_3 \cdots d_r, \tag{A11}$$

where d_i, $1 \le i \le r$ is an odd prime number.

Since p is a QR mod D, p is also a QR mod d_i. Using Legendre symbol $(x/p) = x^{(p-1)/2}$ mod p, whether $-D$ is a QR mod p or not, that is $(-D/p)$, is given by

$$(-D/p) = (-1/p)\prod_{i=1}^{r}(d_i/p). \tag{A12}$$

According to the quadratic reciprocity law [13], we have

$$(d_i/p) = (-1)^{\frac{(p-1)(d_i-1)}{4}}(p/d_i). \tag{A13}$$

Therefore, using $(p/d_i) = 1$, we have

$$\begin{aligned}(-D/p) &= (-1/p)\prod_{i=1}^{r}(-1)^{\frac{(p-1)(d_i-1)}{4}}(p/d_i) \\ &= (-1/p)\prod_{i=1}^{r}(-1/p)^{\frac{(d_i-1)}{2}}.\end{aligned} \tag{A14}$$

Since $D \equiv 3$ mod 4, the number of d_i's such that $d_i \equiv 3$ mod 4 is odd. Thus, we have $(-D/p) = (-1/p)^2 = 1$.

Appendix D. Modular Reduction Modulo $E(x, 0)$

Consider the product $p(x)$ of two polynomials whose degree is less than or equal to 2. The degree of $p(x)$ is less than or equal to 4. Let $p(x)$ be given as

$$\begin{aligned}p(x) &= p_4x^4 + p_3x^3 + p_2x^2 + p_1x + p_0, \\ &p_4, p_3, p_2, p_1, p_0 \in F_p,\ \ p_4 \ne 0,\end{aligned} \tag{A15}$$

the modular reduction modulo $E(x,0)=x^3+3kx+2k$ is calculated as follows:

$$\begin{aligned}p(x) \bmod E(x, 0) &= p_4x^4 + p_3x^3 + p_2x^2 + p_1x + p_0 \\ &-(p_4x^4 + 3kp_4x^2 + 2kp_4x) \\ &-(p_3x^3 + 3kp_3x + 2kp_3).\end{aligned} \tag{A16}$$

Therefore, we need two multiplications kp_4 and kp_3, where $3kp_4$, for example, is calculated by

$$3kp_4 = kp_4 + kp_4 + kp_4. \tag{A17}$$

In the same way, when the degree of $p(x)$ is 3, we need one multiplication. If the degree of $p(x)$ is less than or equal to 2, we do not need any multiplications.

Appendix E. Window Method

As shown in Table 5, step 5 in which the NAF representation method was used took 8.45×10^{-1} seconds. Under the same condition, by using the window method with the following window width instead of the NAF representation method, the computation time at step 5 became

$$\begin{aligned}\omega &= 2 : 8.43 \times 10^{-1} \text{ seconds} \\ \omega &= 3 : 8.48 \times 10^{-1} \text{ seconds} \\ \omega &= 4 : 8.50 \times 10^{-1} \text{ seconds}\end{aligned} \tag{A18}$$

where ω was the window size. As a result, step 5 with the window method became a little faster than that with the NAF representation method; however, the efficiency was quite low.

Appendix F. Montgomery Operation for Step 5

Using the generalized Montgomery operation with the Montgomery ladder technique [10], [16], we can calculate only the x-coordinate of nP using that of P, where P is a rational point and projective coordinates are used in general. Therefore, we can apply the Montgomery operation with the Montgomery ladder technique for our purpose. It does not need any square root calculations because the y-coordinate of a rational point is not needed. In addition, it does not need any inversions in F_p because projective coordinates are used. In the same way of C_5 shown in (33), we can evaluate the calculation cost at step 5 with the Montgomery operation with the Montgomery ladder technique. Let C_{5M} be the calculation cost given as [10]

$$C_{5M} \approx (14M + 5S)\log_2 p. \tag{A19}$$

In the same way as in section IV.3, when $\log_2 p = 160$,

$$C_{5M} \approx 18M \log_2 p = 2880M. \tag{A20}$$

Therefore, we can estimate that step 5′ and step 5″ are about 2.2 and 21 times faster than step 5 with the Montgomery operation, respectively.

Table A1 and Table A2 show the experimental result in the case in which the generalized Montgomery operation with the Montgomery ladder technique is applied for step 5. From the results, we find that

- Step 5′ is about 2.1 times faster than step 5,
- Step 5″ is about 22 times faster than step 5.

It can be understood that these results come from the calculation costs. As compared to section V.2.B, the experimental result is very close to the estimation described above. This is because the generalized Montgomery operation with the Montgomery ladder technique does not need any square root calculations. Especially when 4 divides the characteristic minus 1, we find that the Montgomery operation

Table A1. Computation time for generating 100 prime order curves.

Size of p (bits)	Step	# of inputs	Computation time [unit:s]	Remarks	
			$4 \nmid (p-1)$		
160	Step 5 / Step 5′	100	6.99×10^{-1} / 3.28×10^{-1}		
	Total	-	9.72 / 9.34	Improvements	4.0%
180	Step 5 / Step 5′	100	8.20×10^{-1} / 3.82×10^{-1}		
	Total	-	1.36×10 / 1.29×10	Improvements	5.2%
200	Step 5 / Step 5′	100	1.04 / 4.99×10^{-1}		
	Total	-	1.78×10 / 1.68×10	Improvements	5.6%
			$4 \mid (p-1)$		
160	Step 5 / Step 5″	100	6.96×10^{-1} / 3.16×10^{-2}		
	Total	-	1.03×10 / 9.63	Improvements	6.5%
180	Step 5 / Step 5″	100	8.13×10^{-1} / 3.61×10^{-2}		
	Total	-	1.47×10 / 1.39×10	Improvements	5.4%
200	Step 5 / Step 5″	100	1.04 / 4.64×10^{-2}		
	Total	-	1.91×10 / 1.80×10	Improvements	5.8%

* CPU: Pentium 4, 3.80 GHz, we used NTL [18] and prepared class polynomials by [19].
** For step 5, the generalized Montgomery operation with Montgomery ladder technique is applied [10],[16] .

Table A2. Computation time for generating 100 curves whose order has a large prime factor more than 160 bits when $4 \mid (p - 1)$ and the size of the characteristic p is 170 bits.

Size of p (bits)	Step	# of inputs	Computation time (s)	Remarks	
170	Step 5 / Step 5″	100	8.25×10^{-1}/ 3.36×10^{-2}		
	Total	-	5.43 / 4.63	Improvements	14.7%

* CPU: Pentium 4, 3.80 GHz, we used NTL [18] and prepared class polynomials by [19].
** For step 5, the generalized Montgomery operation with Montgomery ladder technique is applied [10],[16] .

Table A3. Computation time for generating 100 curves whose order has a large prime factor more than 160 bits when $4 \mid (p - 1)$ and the size of the characteristic p is 170 bits.

Size of p (bits)	Step	# of inputs	Computation time (s)	Remarks	
170	Step 0	22900	1.41	# of prime numbers as p	406
	Step 1a	16973	8.38×10^{-3}		
	Step 1b	4513	1.84		
	Step 2	863	7.26×10^{-1}		
	Step 3	100	3.27×10^{-2}	Average of discriminant D	168.20
				Average of deg($H_D(x)$)	2.12
	Step 4	100	5.18×10^{-1}	# of cases that deg($H_D(x)$) = 1	42
				Average of shift s	0.97
	Step 5	100	9.98×10^{-1} (6.09×10^{-1}) †		
	Total	-	5.59		

* CPU: Pentium 4, 3.80 GHz, we used NTL [18] and prepared class polynomials by [19].
** For step 5, the original and generalized Montgomery operations with Montgomery ladder are applied [10], [16].
*** The curves are restricted to F_p– rational two-torsion curves. *FindRoots*() function in NTL is used at Step 5.
† The data in the parenthesis shows the calculation time needed for the transformation to Montgomery form.

with the Montgomery ladder technique at step 5 works more efficiently for our purpose as compared to the NAF representation method. The original Montgomery operation is applied for elliptic curves in the Montgomery form as

$$Ay^2 = x^3 + Bx^2 + x, \quad A, B \in F_p. \qquad \text{(A21)}$$

Some of the elliptic curves in the form (1) can be transformed into the above Montgomery form [10]. If we have the Montgomery form (A21), the calculation cost $C_{5\overline{M}}$ at step 5 by the original Montgomery operation with the Montgomery ladder technique is evaluated as [10]

$$C_{5\overline{M}} \approx (6M + 4S)\log_2 p. \qquad \text{(A22)}$$

When $\log_2 p = 160$, $C_{5\overline{M}} \approx 9.2M\log_2 p = 1472M$. As compared to (A20), we can easily find that it will become about twice as fast; however, the transformation into the Montgomery form needs the following conditions and calculations. First, (1) must satisfy the following conditions:

(a) The cubic polynomial $E(x, 0) = x^3 + ax + b$ has at least one root α in F_p; accordingly the curve becomes F_p–rational two-torsion elliptic curve.
(b) The element $3\alpha^2 + a$ is a QR in F_p.

Then, put $A = 3\alpha s$, $B = s$, where s is a square root of $(3\alpha^2+a)^{-1}$, we have the Montgomery form (A21). For the former operation (a), when we solve a cubic equation $x^3 + ax + b = 0$ over F_p, a few square and cubic root calculations are needed. Of course we can apply several greatest common divisor polynomial calculations such as (43). For this calculation, we applied *FindRoots*() function in NTL [18] that is almost the same as that of (43). For the latter operation (b), one square root calculation is needed. In addition, the possibility that $E(x, 0) = x^3 + ax + b$ has at least one root α in F_p is about 2/3, it is the case that $x^3 + ax + b$ is reducible over F_p. Moreover, the possibility that the element $3\alpha^2 + a$ is a QR in F_p is about 1/2, in general. Therefore, the possibility that we can transform the curve into the Montgomery form is about 1/3. Consequently, if we use only the original Montgomery operation with Montgomery ladder technique, we can easily guess that generating a lot of secure prime order curves takes much more calculation time as compared to using the generalized Montgomery operation with the Montgomery ladder technique. According to the former condition (a), we can use the original Montgomery operation with the Montgomery ladder only for F_p–rational two-torsion elliptic curves. This paper is particularly dealing with no F_p–rational two-torsion elliptic curves; therefore, it is out of the scope of this paper. However, from the viewpoint of the fast generation of secure elliptic curves, we also carried out a simulation as shown in Table A3. In the simulation, we considered the following restrictions:

- The 170 bits characteristic p satisfies $4 \mid (p - 1)$.
- The curves are F_p-rational two-torsion curves.
- The curves can be transformed into Montgomery form and the calculation time contains the time needed for the transformation.

As shown in Table A3, for the purpose of generating a lot of secure order elliptic curves with a CM-based algorithm, we cannot say that the original Montgomery operation is so efficient because it needs the above introduced transformation at step 5.

References

[1] Y. Nogami and Y. Morikawa, "A Method for Distinguishing the Two Candidate Elliptic Curves in CM Method," *Proc. of the 7th Int'l Conf. Information Security and Cryptology (ICISC2004)*, 2004, pp. 187-198.

[2] T. Sato and K. Araki, "Fermat Quotients and the Polynomial Time Discrete Log Algorithm for Anomalous Elliptic Curve," *Commentarii Math. Univ. Sancti. Pauli*, vol. 47, no. 1, 1998, pp. 81-92.

[3] P. Gaudry, F. Hess, and N. Smart, "Constructive and Destructive Facets of Weil Descent on Elliptic Curves," *Hewlett Packard Tech. Report HPL-2000-10*, 2000.

[4] K. Horiuchi et al., "Construction of Elliptic Curves with Prime Order and Estimation of Its Complexity," *IEICE Trans.*, J82-A, no. 8, 1999, pp. 1269-1277.

[5] Y. Nogami and Y. Morikawa, "Fast Generation of Elliptic Curves with Prime Order over $F_{p^{2c}}$" *Proc. of Workshop on Coding and Cryptography 2003*, 2003, pp. 347-356.

[6] E. Konstantinou, Y. Stamatiou, and C. Zaroliagis, "On the Construction of Prime Order Elliptic Curves," *Indocrypto 2003*, LNCS 2904, 2003, pp. 309-322.

[7] E. Savas, T. Schmidt, and C. Koc, "Generating Elliptic Curves of Prime Order," *CHES2001, LNCS2162*, 2001, pp. 142-158.

[8] I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography*, LNS 265, Cambridge University Press, 1999.

[9] A.O.L. Atkin and F. Morain, "Elliptic Curves and Primality Proving," *Math. Comp.61*, 1993, pp. 29-68.

[10] H. Cohen and G. Frey, *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, *Discrete Mathematics and Its Applications*, Chapman & Hall CRC, 2005, pp. 280-285, p. 458.

[11] M. Obara, Y. Nogami, and Y. Morikawa, "A Method for Checking the Parity of (#E – 1)/2," *Technical Report of IEICE*, vol. 104/ISEC2004-13, 2004, pp. 1-6.

[12] Y. Nogami and Y. Morikawa, "A Consideration on the Order of Genus 2 Hyperelliptic Curve," *The 28th Symp. Information Theory and Its Application* (SITA2005), 2005, pp. 889-892.

[13] E. Berlekamp, *AlgebraicCodingTheory*, McGraw-Hill, 1968.

[14] A.D.Booth, “A Signed Binary Multiplication Technique,” *Quarterly J. Mech. and Appl.Math,* vol. 4, no. 2, 1951, pp. 236-240.

[15] E. Brier and M. Joye, “Weierstra*β* Elliptic Curves and Side Channels Attacks,” *PKC2002, LNCS 2274*, Springer-Verlaq, 2002, pp. 335-345.

[16] P. L. Montgomery, “Speeding the Pollard and Elliptic Curve Methods of Factorization,” *Mathematics of Computation*, vol. 48, no. 177, 1987, pp. 243-264.

[17] D. Bailey and C. Paar, “Optimal Extension Fields for Fast Arithmetic in Public-Key Algorithms,” *Proc. Asiacrypt 2000*, LNCS 1976, 2000, pp. 248-258.

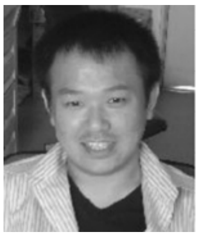
[18] A Library for doing Number Theory, http://www.shoup.net/ntl/.

[19] E. Konstantinou et al., “A Software Library for Elliptic Curve Cryptography,” http://www.ceid.upatras.gr/faculty/zaro/software/ecc-lib/

[20] F. Morain, “Implementing the Asymptotically Fast Version of the Elliptic Curve Primality Proving Algorithm,” http://www.lix.polytechnique.fr/Labo/Francois.Morain/Articles/fastecpp050127.pdf.

[21] H. Cohen, *A Course in Computational Algebraic Number Theory*, Springer, 2000.

[22] R. Lidl and H. Niederreiter, *Finite Fields*, *Encyclopedia of Mathematics and Its Applications*, Cambridge University Press, 1984.



Yasuyuki Nogami graduated from Shinshu University in 1994 and received the Phd degree in 1999 from Shinshu University. He is now a Research Associate of Okayama University. His main fields of research are finite field theory and its applications. He is a member of IEICE and IEEE.

Mayumi Obara graduated from the Department of Communication Network Engineering, the Faculty of Engineering, Okayama University. She is now with the Graduate School of Natural Science and Technology, Okayama University. She is now studying public key cryptographies, especially elliptic curve cryptography.

Yoshitaka Morikawa graduated from the Department of Electronic Engineering, Osaka University in 1969 and obtained the MS degree in 1971. He then joined Matsushita Electric, where he engaged in research on data transmission. In 1972, he became a Research Associate at Okayama University, and subsequently an Associate Professor in 1985. He is now a Professor of the Department of Communication Network Engineering. He has been engaged in research on image information processing. He holds a D.Eng. degree.