# Algorithms for computations in Jacobian group of $C_{ab}$ curve and their application to discrete-log-based public key cryptosystems

ARITA Seigo
C&C Media Research Laboratories, NEC, Japan

**Abstract**

Nowadays, elliptic curve cryptosystems(ECC) receive attention and much efforts are being dedicated to make it more and more practical. However, on the other hand, there are opinions that elliptic curves are too special objects in the mathematical background to be used for cryptosystems without anxiety. To step up the safety of ECC, it is worthwhile to try discrete-log based cryptosystems using more general algebraic curves beyond elliptic or hyperelliptic curves. In the presented paper, we introduce $C_{ab}$ curves, which compose a wide class of algebraic curves, including elliptic and hyperelliptic curves. And we construct an efficient algorithm for addition in Jacobian group of $C_{ab}$ curves. That means the class of algebraic curves which can be used for discrete-log based cryptosystems could be widely enlarged.

**Keywords:** Discrete logarithm problem, Jacobian group, Elliptic curve cryptosystems, $C_{ab}$ curve

# 1 Introduction

Nowadays, elliptic curve cryptosystems(ECC) receive attention and much efforts are being dedicated to make it more and more practical. In ECC, we can use much shorter keys than RSA. Some people say that ECC would occupy the dominating position in public key cryptosystems in the near future. However, there are opposite opinions. They say, elliptic curves are too special objects in the mathematical background, involve so much mathematical theories that we cannot explore all of the possible attacks using those theories in the near future. In fact, recently, we encountered new attacks against ECC successively in September '97 and July '98[Semaev 98, SA 98, Smart, US 98].

ECC is a public key cryptosystem based on the discrete-log problem on Jacobian groups of elliptic curves. General algebraic curves accompany Jacobian groups as well as elliptic curves. Compared with general algebraic curves, elliptic or hyperelliptic curves are very special algebraic curves. To step up the safety of ECC, it is worthwhile to try discrete-log based cryptosystems using Jacobian group of more general algebraic curves beyond elliptic or hyperelliptic curves.

Suppose a class $C$ of algebraic curves is given. To construct a discrete-log based cryptosystems using the class $C$, we need to solve the following two basic problems.

Problem 1 Give an efficient algorithm for addition in the Jacobian group of any curve in the class $C$.

Problem 2 Give an efficient algorithm to find a curve with Jacobian of almost prime order in the class $C$.

The above two problems have been solved for elliptic curves and we are now to serve them for practical use. For hyperelliptic curves, Problem 1 is solved in [Cantor 87, Koblitz 89], and Problem 2 is partially solved in [Koblitz 97, MCT 96, Spallek 94].

[MK 93] has found a class of algebraic curves named "$C_{ab}$ curve" in the development of algebraic geometry codes. $C_{ab}$ curves compose a wide class of algebraic curves, including elliptic and hyperelliptic curves. Moreover, $C_{ab}$ curves has good properties to do computations on computers.

In the presented paper, we solve Problem 1 for $C_{ab}$ curves. That is, we construct an efficient algorithm for addition in the Jacobian of $C_{ab}$ curves, and show the possibility of a discrete-log based cryptosystems using $C_{ab}$ curves.

# 2 Preliminaries

We arrange facts about Jacobian group of an algebraic curve and about Gröbner basis of an ideal in a polynomial ring.

## 2.1 Jacobian group of an algebraic curve

Take an algebraic curve $C$ defined over a finite field $K$. Let $\overline{K}$ be an algebraic closure of $K$. For integers $m_i$ and rational points(strictly, places) $P_i$ of $C$ over $\overline{K}$, a formal sum $D = \sum m_i P_i$ is called a divisor. A divisor $D = \sum m_i P_i$ is called positive when $m_i \geq 0$ for all $i$. The integer $m = \sum m_i$ is called degree of a divisor $D = \sum m_i P_i$, denoted by $deg(D)$. All divisors on a curve $C$ become an abelian group $\boldsymbol{D}$ under formal additions, and all divisors of degree zero is a subgroup $\boldsymbol{D^0}$ of $\boldsymbol{D}$. Let $\boldsymbol{D_K}$ and $\boldsymbol{D_K^0}$ be invariant subgroups of $\boldsymbol{D}$ and $\boldsymbol{D^0}$, respectively, under the action of $Gal(\overline{K} \mid K)$. Elements of $\boldsymbol{D_K}$ are called divisors defined over $K$.

For a rational function $f$ on a curve $C$, let $v_P(f) = n(or, -n)$ be the order $n$ of zero(or, pole) of $f$ at a point $P$. Then, $(f) := \sum_P v_P(f)P$ becomes a divisor of degree 0, called a principal divisor of $f$. The partial sums $(f)_0 := \sum_{P, v_P(f) \geq 0} v_P(f)P$ and $(f)_\infty := \sum_{P, v_P(f) \leq 0} -v_P(f)P$ are called a zero divisor and a pole divisor of $f$, respectively. Note both $(f)_0$ and $(f)_\infty$ are positive divisors, and $(f) = (f)_0 - (f)_\infty$. All principal divisors $\{(f) \mid f \in \overline{K}(C)\}$ become an subgroup $\boldsymbol{P}$ of $\boldsymbol{D^0}$. The residue group of $\boldsymbol{D^0}$ by $\boldsymbol{P}$ is called Jacobian group of $C$, denoted by $J(C)$. The invariant subgroup $J_K(C)$ of $J(C)$ under the action of $Gal(\overline{K} \mid K)$ is called Jacobian group of $C$ defined over $K$.

For a divisor $D$ defined over $K$,

$$L(D) = \{f \in K(C) \mid (f) + D \geq 0\} \cup \{0\}$$

is a finite dimensional vector space over $K$. From Riemann's theorem, we see $dim\ L(D) \geq deg(D) + 1 - g$, where $g$ denotes the genus of $C$.

For details, see [Silverman].

## 2.2 Monomial order and Gröbner bases

Let $\boldsymbol{Z}_{\geq 0}$ denote the set of non-negative integers. For a $n$-variable monomial $x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$, a n-tuple of integers $\alpha = (\alpha_1, \cdots, \alpha_n) \in \boldsymbol{Z}_{\geq 0}^n$ is called a multi-degree of $x^\alpha$, denoted by $\mathrm{MD}(x^\alpha)$. A well-order $<$ on $\boldsymbol{Z}_{\geq 0}^n$ is called monomial order if $\alpha + \gamma < \beta + \gamma$ holds whenever $\alpha < \beta$ and $\gamma \in \boldsymbol{Z}_{\geq 0}^n$. A monomial order on $\boldsymbol{Z}_{\geq 0}^n$ determines a well-order on the set of all monomials through multi-degrees, called a monomial order, too. Suppose a monomial order is given. For a $n$-variable polynomial $f$, the maximum term appearing in $f$ with respect to the monomial order is called a leading term of $f$, denoted by $\mathrm{LT}(f)$. By using monomial orders, we can describe the division algorithm for n-variable polynomials, in which a polynomial $f$ is divided by a set of polynomials $G$.

The ideal generated by polynomials $g_1, \cdots, g_m$ is denoted by $\langle g_1, \cdots, g_m \rangle$. Fix a monomial order. A subset $G = \{g_1, \cdots, g_m\}$ of an ideal $I$ of $n$-variable polynomial ring $K[x_1, \cdots, x_n]$ is called a Gröbner basis of $I$ when we have $\langle \mathrm{LT}(I) \rangle = \langle \mathrm{LT}(g_1), \cdots, \mathrm{LT}(g_m) \rangle$. Any ideal of $n$-variable polynomial ring $K[x_1, \cdots, x_n]$ has a Gröbner basis. If $G = \{g_1, \cdots, g_m\}$ is a Gröbner basis of an ideal $I$, $G$ generates $I$.

For an ideal $I$, all of the multi-degrees of monomials outside $\mathrm{LT}(I)$ is called $\Delta$-set of $I$, denoted by $\Delta(I)$:

$$\Delta(I) = \{\alpha \in \boldsymbol{Z}_{\geq 0} \mid x^\alpha \notin \mathrm{LT}(I)\}.$$

Let $\delta(I)$ denote the number of elements in $\Delta(I)$. On the other hand, for a polynomial set $G = \{g_1, \cdots, g_m\}$, we set

$$\delta(g_1, \cdots, g_m) := \sharp\ (\boldsymbol{Z}_{\geq 0}^n - \cup_{i=1}^m (\mathrm{MD}(\mathrm{LT}(g_i)) + \boldsymbol{Z}_{\geq 0}^n)).$$

($\sharp S$ denotes the number of elements in the set $S$.) Then, for an ideal $I$ satisfying $\delta(I) < \infty$ and for its subset $G = \{g_1, \cdots, g_m\}$, we have

$$\mathrm{G\ is\ a\ Gröbner\ basis\ of\ I} \Longleftrightarrow \delta(I) = \delta(g_1, \cdots, g_m). \tag{1}$$

Use of a Gröbner basis justifies the division algorithm for $n$-variable polynomials. That is, a polynomial $f$ is a member of an ideal $I$ if and only if the remainder of $f$ divided by the Gröbner basis of $I$ is equal

to zero. Although there are several Gröbner bases for a given ideal, the reduced Gröbner basis is uniquely determined up by a given ideal. A Gröbner basis $G$ of an ideal $I$ is called reduced when 1) the coefficient of $\mathrm{LT}(p)$ is 1 for all $p \in G$, 2) any term appearing in $p$ doesn't belong to $\langle \mathrm{LT}(G - \{p\}) \rangle$ for all $p \in G$.

For details, see [CLO].

# 3 $C_{ab}$ curve

Following [Miura], we introduce $C_{ab}$ curves.

Let $K$ be a finite field and $C$ be an algebraic curve over $K$. Suppose $C$ has at least one place of degree one over $K$. Let $P$ be a place of degree one over $K$. We take the ring $L(\infty P)$ of functions on $C$ which are holomorphic away from $P$:

$$L(\infty P) = \{f \in K(C) \mid v_Q(f) \geq 0 \ (\forall Q \neq P)\}.$$

All of the pole number $-v_P(f)$ at $P$ of $f$ in $L(\infty P)$ become a monoid $M_P$:

$$M_P = \{-v_P(f) \mid f \in L(\infty P)\}.$$

**Definition 1 ($C_{ab}$ curve)** *When $M_P$ is generated by two natural numbers $a$ and $b$, we call $(C, P)$ $C_{ab}$ curve.*

Let $(C, P)$ be $C_{ab}$ curve. Then, there is a function $x \in L(\infty P)$ with the pole number $a$ at $P$ and a function $y \in L(\infty P)$ with the pole number $b$ at $P$. Using these functions $x$ and $y$, we can construct an affine model of $C_{ab}$ curve of the form:

$$\sum_{0 \leq i \leq b, 0 \leq j \leq a, ai+bj \leq ab} \alpha_{i,j} X^i Y^j = 0. \tag{2}$$

Here, $\alpha_{i,j}$ belongs to $K$ and both $\alpha_{b,0}$ and $\alpha_{0,a}$ are not equal to zero.

The affine model in the equation (2) is called Miura model of $C_{ab}$ curve $(C, P)$. In Miura model, $C_{ab}$ curve is nonsingular on affine-plane, and the place $P$ becomes the place $P_\infty$ centered on the unique infinite point on the curve. An elliptic curve is nothing but $C_{23}$ curve, and a hyperelliptic curve is nothing but $C_{2b}$ curve. The genus of $C_{ab}$ curve is equal to $(a-1)(b-1)/2$. From now on, we deal with $C_{ab}$ curve always in Miura model.

For computations in $C_{ab}$ curves, $C_{ab}$ order $>_{cab}$, which is a monomial order, plays a fundamental role.

**Definition 2 ($C_{ab}$ order)** *For $\alpha = (\alpha_1, \alpha_2)$, and $\beta = (\beta_1, \beta_2) \in \boldsymbol{Z}^2_{\geq 0}$, let $\alpha >_{cab} \beta$ when the following either (i) or (ii) holds:*

$$\begin{array}{ll} (i) & a\alpha_1 + b\alpha_2 > a\beta_1 + b\beta_2 \\ (ii) & a\alpha_1 + b\alpha_2 = a\beta_1 + b\beta_2, \ \alpha_1 < \beta_1. \end{array}$$

Note that monomials $X^\alpha Y^\beta$ are holomorphic away from $P_\infty$, supposed to be functions on $C_{ab}$ curve. In $C_{ab}$ order, monomials $X^\alpha Y^\beta$, being supposed to be functions on $C_{ab}$ curve, are put in order by the pole number $-v_{P_\infty}(x^\alpha y^\beta) = a\alpha + b\beta$ at $P_\infty$, and when two monomials have the same pole number at $P_\infty$, the monomial with larger degree in $X$ should be smaller.

# 4 Addition algorithm for Jacobian group in divisor representation

[Volcheck 94] has given an algorithm for addition in the Jacobian of a general curve over a finite field. However, the algorithm in [Volcheck 94] needs factorization of polynomials over finite fields, so its efficiency is not enough for use in cryptosystems. Moreover, the algorithm in [Volcheck 94] needs many divisions of labor, so its space complexity is too high for real use.

We will construct much more efficient algorithm for addition in the Jacobian of $C_{ab}$ curve in this and the next section. Our algorithm has the complexity $O(g^3 \log(q)^2)$ for $C_{ab}$ curve of genus $g$ defined over $GF(q)$, and it could be tolerable for use in real cryptosystems.

Take a $C_{ab}$ curve $C$ defined over a finite field $K$. Recall $\boldsymbol{D_K}$,$\boldsymbol{D_K^0}$, and $\boldsymbol{P_K}$ denotes the divisor group of $C$ over $K$, its subgroup of divisors of degree zero, and the principal divisor group of $C$ over $K$, respectively. Jacobian group $J_K(C)$ of $C$ over $K$ is $\boldsymbol{D_K^0/P_K}$. When an element $j$ in the Jacobian $J_K(C)$ has a representative $D$ in $\boldsymbol{D_K^0}$, we set $j = [D]$;

$$\begin{array}{rcl} J_K(C) & \simeq & \boldsymbol{D_K^0/P_K} \\ j & \mapsto & [D]. \end{array}$$

**Definition 3** *A divisor $D$ of degree zero is called "semi-normal" when we have $D = E - n\,\infty$ with a positive divisor $E$ prime to $\infty$ and with a positive integer $n$ between 0 and $g$.*

**Lemma 4** *For any element $j$ in the Jacobian $J_K(C)$, there is a semi-normal divisor $D$ such that $j = [D]$.*

**Proof** There is an divisor $D$ of degree zero such that $j = [D]$. Because we have $\dim L(D+g\,P_\infty) \geq g+1-g = 1$ from Riemann's theorem, there is a nonzero function $f \in L(D + g\,P_\infty)$ such that $D + g\,P_\infty + (f) \geq 0$. Then, setting $E = D + g\,P_\infty + (f)$, we have $j = [E - g\,P_\infty]$ □

By Lemma 4, we can express any element $j$ in the Jacobian $J_K(C)$ by a semi-normal divisor $D$. However, the expression is ambiguous, because there are several semi-normal divisor $D_i$'s such that $j = [D_i]$. In order to choose the unique semi-normal divisor, we use the following algorithm.

**Algorithm 1**
*Input: a divisor $D = E - nP_\infty$ of degree zero, where $E$ is a positive divisor prime to $P_\infty$*
*Output: a semi-normal divisor $G$ equivalent to $-D$*
1° *Find a nonzero function $f \in L(\infty P_\infty)$ satisfying $(f)_0 \geq E$ with the least pole number $-v_{P_\infty}(f)$.*
2° $G \leftarrow -D + (f)$

Algorithm 1 chooses the unique divisor associated to a given element $j \in J_K(C)$. That is,

**Proposition 5** *For any two equivalent semi-normal divisors $D_1$ and $D_2$, the outputs of Algorithm 1 for $D_1$ and $D_2$ are the same as divisors.*

**Proof** Let $D_1 = E_1 - nP_\infty$ and $D_2 = E_2 - nP_\infty$ be equivalent semi-normal divisors. Then there is a nonzero function $\lambda$ such that

$$E_1 - n_1 P_\infty = E_2 - n_2 P_\infty + (\lambda).$$

Take a function $f_1$ for $D_1$ just as in Algorithm 1 such that

$$\mathrm{Supp}((f_1)_\infty) \subseteq \{P_\infty\}, \quad (f_1)_0 \geq E_1. \tag{3}$$

(Here, $\mathrm{Supp}(D)$ denotes the support of a divisor $D$.) Then, putting the pole number of $f_1$ at $P_\infty$ as $k_1$, we have

$$\begin{array}{rcl} (f_1\lambda^{-1}) & = & (f_1) - (\lambda) \\ & = & (f_1)_0 - E_1 + E_2 + (n_1 - k_1 - n_2)\infty. \end{array}$$

Here, because $(f_1)_0 - E_1 + E_2 \geq E_2$, setting $f_2 = f_1\lambda^{-1}$, $f_2$ satisfies that

$$\mathrm{Supp}((f_2)_\infty) \subseteq \{P_\infty\}, \quad (f_2)_0 \geq E_2. \tag{4}$$

Because $\lambda$ is independent on the choices of $f_1$ and $f_2$, the correspondence $f_1 \mapsto f_2 = f_1\lambda^{-1}$ assigns $f_1$ with the least pole order at $P_\infty$ to $f_2$ with the least pole order at $P_\infty$. Then the relation

$$\begin{array}{rcl} -E_2 + n_2 P_\infty + (f_2) & = & -E_2 + n_2 P_\infty + (f_1) - E_1 + E_2 + (n_1 - n_2)P_\infty \\ & = & -E_1 + n_1 P_\infty + (f_1). \end{array}$$

shows that the output of Algorithm 1 for $D_1$ is the same as the output for $D_2$ as divisors. □

**Definition 6** *We call semi-normal divisors obtained as outputs of Algorithm 1 "normal divisors."*

Algorithm 1 outputs a normal divisor which is equivalent to -1 times the input semi-normal divisor. So, by applying Algorithm 1 twice repeatedly to any semi-normal divisor, we get the equivalent normal divisor. From Lemma 4 and Proposition 5, we get

**Theorem 7** *Any element $j$ in Jacobian $J_K(C)$ is expressed uniquely by a normal divisor.*

From Theorem 7, we get the following algorithm for addition in the Jacobian:

**Algorithm 2**
*Input: semi-normal divisors $D_1 = E_1 - n_1 P_\infty$ and $D_2 = E_2 - n_2 P_\infty$*
*Output: a normal divisor $D_3 = E_3 - n_3 P_\infty$ which is equivalent to $D_1 + D_2$*
1° *By applying Algorithm 1 to $D_1 + D_2 = (E_1 + E_2) - (n_1 + n_2)P_\infty$, get a normal divisor $D' = E' - n' P_\infty$.*
2° *By applying Algorithm 1 to a normal divisor $D' = E' - n' P_\infty$, get a normal divisor $D_3 = E_3 - n_3 P_\infty$ and output $D_3$.*

To perform Algorithm 1 and 2 on computers, we need to encode divisors in some way. The most straightforward way is to encode divisors as point sets with multiplicities as in [Volcheck 94]. But to encode divisors involved in Algorithm 1 and 2 as point sets, we need to factor polynomials of degree $3g$ into irreducible components and it hurts the efficiency of the algorithms.

In $C_{ab}$ curves, Jacobian group is naturally isomorphic to the ideal class group of the coordinate ring. In the next section, appealing to this fact, we realize Algorithm 1 and 2 by ideal computations in the coordinate ring.

# 5 Addition algorithm for Jacobian group in ideal representation

Using the natural isomorphism between the Jacobian of $C_{ab}$ curve and the ideal class group of its coordinate ring, we realize Algorithm 1 and 2 by ideal computations.

## 5.1 Construction of the addition algorithm

Let $C$ be a $C_{ab}$ curve defined over a finite field $K$ with a defining equation

$$F(X,Y) = \sum_{0 \le i \le b, 0 \le j \le a, ai+bj \le ab} \alpha_{i,j} X^i Y^j = 0.$$

Because $C_{ab}$ curve $C$ is nonsingular on affine plane, its coordinate ring $K[x,y]=K[X,Y]/(F(X,Y))$ is a Dedekind domain.

In general, for an algebraic curve $C$ of which coordinate ring $A_K$ is a Dedekind domain, its Jacobian group $J_K(C)$ is naturally isomorphic to the ideal class group $H(A_K)$ of its coordinate ring[Hartshorne]. For a $C_{ab}$ curve $C$, the isomorphism $\Phi$ is given by

$$\Phi : \begin{array}{ccc} J_K(C) & \xrightarrow{\sim} & H(A_K) \\ [\sum_P n_P P - nP_\infty] & \mapsto & [L(\infty P_\infty - \sum_P n_P P)]. \end{array}$$

We call ideals, corresponding to normal divisors by the isomorphism $\Phi$, normal ideals (Note any ideal of $A_K$ corresponds to semi-normal divisors). Recall $C_{ab}$ order is based on the pole order at $P_\infty$ of monomials as functions on $C_{ab}$ curve. Applying the isomorphism $\Phi$ to the Algorithm 1 and 2, we get the following Algorithm 3 and 4 for addition in Jacobian group $J_K(C)$.

**Algorithm 3**
*Input: an ideal $I$ of the coordinate ring $A$*
*Output: a normal ideal $J$ equivalent to the inverse ideal of $I$*
1° *Find the minimum polynomial $f \neq 0$ w.r.t. $C_{ab}$ order, belonging to the ideal $I$.*
2° *Find an ideal $J$ satisfying $(f) = I \cdot J$.*

**Algorithm 4**
*Input: ideals $I_1$ and $I_2$ of the coordinate ring $A$*
*Output: a normal ideal $I_3$ equivalent to the ideal product $I_1 \cdot I_2$*
1° *By applying Algorithm 3 to the ideal product $I_1 \cdot I_2$, get a normal ideal $J$.*
2° *By applying Algorithm 3 to the ideal $J$, get a normal ideal $I_3$.*

In Algorithm 4, we must call Algorithm 3 twice. Now, we are merging two calls of Algorithm 3 in Algorithm 4. Suppose we get a polynomial $f$ and the ideal $J$ in the first call of Algorithm 3. Then, they must satisfy

$$(f) = I_1 \cdot I_2 \cdot J.$$

Similarly, suppose we get a polynomial $g$ and the ideal $I_3$ in the second call of Algorithm 3. Then,

$$(g) = J \cdot I_3.$$

Using these relations, we have

$$\begin{aligned} I_1 \cdot I_2 \cdot (g) &= I_1 \cdot I_2 \cdot J \cdot I_3 && (5) \\ &= (f) \cdot I_3. && (6) \end{aligned}$$

So, the output $I_3$ of Algorithm 4 should satisfy

$$I_3 = g/f \cdot I_1 \cdot I_2.$$

Thus, we get the following Algorithm 5 for addition in the Jacobian group of $C_{ab}$ curve.

**Algorithm 5 (Addition in the Jacobian)**
*Suppose $C_{ab}$ curve is given by $F(X,Y) = 0$.*
*Input: ideals $I_1$ and $I_2$ of $K[X,Y]/(F(X,Y))$*
*Output: a normal ideal $I_3$ of $K[X,Y]/(F(X,Y))$*

1° $J \leftarrow I_1 \cdot I_2$
2° $f \leftarrow$ *the minimum polynomial* $f(\neq 0) \in J$ *w.r.t* $C_{ab}$ *order*
3° $g \leftarrow$ *the minimum polynomial* $g(\neq 0)$ *w.r.t.* $C_{ab}$ *order, satisfying* $g \cdot J \subseteq \langle f, F \rangle$
4° $I_3 \leftarrow g/f \cdot J$

## 5.2 Gröbner bases with respect to $C_{ab}$ order

We express ideals in inputs or outputs for Algorithm 5 by reduced Gröbner bases w.r.t. $C_{ab}$ order. For reduced Gröbner bases w.r.t. $C_{ab}$ order, the following proposition is important.

**Proposition 8** *For any semi-normal divisor $E - n\infty$, and the corresponding ideal $I$ by the isomorphism $\Phi$, we have*

$$\deg(E) = \delta(I).$$

*Here, $\delta(I)$ denotes the number of elements in the $\Delta$-set of ideal $I$.*

**Proof** Because two sides of the equation to be proved are unchanged under base field extensions, we can assume the definition field $K$ is algebraically closed. So, for $E = \sum n_P P$, we have $I = \Phi(E) = \prod I_P^{n_P}$, where $I_P$ is the maximum ideal at the point $P$. By [CLO]Chap5.Sec3.Prop.4, $\delta(I) = \dim_K A/I$. On the other hand, $A/I = \sum_P A/I_P^{n_P}$ and as the $C_{ab}$ curve $C$ is nonsingular on the affine plane, we have $\dim_K A/I_P^{n_P} = n_P$. Therefore we obtain that $\dim A/I = \sum_P n_P = \deg(E)$. □

Using Proposition 8, we can see the form of Gröbner bases of various ideals w.r.t. $C_{ab}$ order. For example, as for $C_{34}$ curve,

**Proposition 9** *An ideal corresponding to a general element in Jacobian group $J_K(C)$ for $C_{34}$ curve $C$ has the following form of reduced Gröbner basis:*

$$\{a_0 + a_1 X + a_2 Y + X^2, b_0 + b_1 X + b_2 Y + XY, c_0 + c_1 X + c_2 Y + Y^2\}.$$

*Here, $a_i, b_i, c_i \in K$.*

**Proof** As the genus of $C$ is $g(C) = (3-1)(4-1)/2 = 3$, a general element $j = [P_1 + P_2 + P_3 - 3P_\infty]$ in Jacobian $J_K(C)$ corresponds to three points $\{P_1, P_2, P_3\}$ on the curve $C$. With respect to $C_{34}$ order, the fourth monomial is $X^2$, the fifth is $XY$ , and the sixth is $Y^2$. So, the ideal $I$ should contain three polynomials of the forms $X^2 + \cdots, XY + \cdots, Y^2 + \cdots$ (Here, $\cdots$ denotes the lower terms). As we can see easily, $\delta(X^2 + \cdots, XY + \cdots, Y^2 + \cdots) = 3$. However, from Proposition 8, we see $\delta(I) = 3$. So, (1) in section 2.2 shows $\{X^2 + \cdots, XY + \cdots, Y^2 + \cdots\}$ is in fact a Gröbner basis of $I$. □

Although we dealt with the ideal corresponding to three points on $C_{34}$ curve in the above proposition, the situation is similar for ideals corresponding to any number of points on general $C_{ab}$ curve.

## 5.3 Details of addition algorithm

We show an example of performing Algorithm 5 and explain details of the algorithm. For instance, we take $C_{34}$ curve $C : Y^3 + X^4 + 1 = 0$ over a prime field $GF(17)$ and compute twice of an ideal $I = \{f_1 = X^2 + 14Y + 4X + 5, f_2 = XY + 3Y + 4X + 9, f_3 = Y^2 + 9Y + 16X + 2\}$ which corresponds to an element of Jacobian $J_{GF(17)}(C)$(ref. Proposition 9). In $C_{34}$ order, monomials stand in a row

$$1, X, Y, X^2, XY, Y^2, X^3, X^2Y, XY^2, \cdots$$

in ascending order.

1° Compute the ideal product $I \cdot I$.

As $I \cdot I$ corresponds to six elements on the curve $C$ and the seventh monomial in $C_{34}$ order is $X^3$, we can see that the Gröbner basis of $I \cdot I$ has the form of $\{X^3 + \cdots, X^2Y + \cdots, XY^2 + \cdots\}$, just as in the proof of Proposition 9. From now on, $\overline{f}^G$ denotes the remainder of $f$ divided by a polynomial set $G$.

Now, we begin computations:

$$\begin{aligned}
g_1 &\leftarrow \overline{f_1^2}^{\{F\}} = X^4 + \cdots \\
g_2 &\leftarrow \overline{f_1 \cdot f_2}^{\{g_1, F\}} = X^3Y + \cdots \\
g_3 &\leftarrow \overline{f_2^2}^{\{g_2, g_1, F\}} = X^2Y^2 + \cdots \\
g_4 &\leftarrow \overline{f_1 \cdot f_3}^{\{g_3, g_2, g_1, F\}} = XY^2 + \cdots \\
g_5 &\leftarrow \overline{f_2 \cdot f_3}^{\{g_4, g_3, g_2, g_1, F\}} = X^2Y + \cdots \\
g_6 &\leftarrow \overline{f_3^2}^{\{g_5, g_4, g_3, g_2, g_1, F\}} = X^3 + \cdots
\end{aligned}$$

So, we have

$$J \leftarrow I \cdot I = \{g_6, g_5, g_4\}.$$

This is the expression of $J$ by the Gröbner basis.

2° $f \leftarrow g_6 = X^3 + 10Y^2 + 5XY + 7Y + 11X + 4$

3° Find the minimum polynomial $h$ satisfying $h \cdot J \subset \langle f, F \rangle$.

Note $\{f, F\}$ is a Gröbner basis of the ideal $\langle f, F \rangle$ as $LT(f) = X^3$ and $LT(F) = Y^3$ are prime to each other. We compute products of $g_5$ and monomials modulo $\{f, F\}$ in ascending order:

$$\begin{aligned}
\overline{g_5}^{\{f,F\}} &= X^2Y + \cdots \\
\overline{Xg_5}^{\{f,F\}} &= XY^2 + \cdots \\
\overline{Yg_5}^{\{f,F\}} &= X^2Y^2 + \cdots
\end{aligned}$$

Now, computing $\overline{X^2g_5}^{\{f,F\}}$, we encounter $4X^2Y^2 + \cdots$. But its leading monomial $X^2Y^2$ is the same as the one of $\overline{Yg_5}^{\{f,F\}}$. So, we have $X^2g_5 \equiv 4Yg_5 + 12XY^2 + \cdots \pmod{\{f, F\}}$. Moreover, we note $XY^2$ is the leading term of $\overline{Xg_5}^{\{f,F\}}$, and repeat similar computations as above. We get

$$X^2g_5 \equiv 4Yg_5 + 12Xg_5 + 2g_5 \pmod{\{f, F\}}.$$

Thus, we obtain $h \leftarrow X^2 + 13Y + 5X + 15$.

4°

$$\begin{aligned}(h/f)\cdot J &= (h/f)\cdot\{g_6, g_5, g_4\}\\ &= \{h, (hg_5)/f, (hg_4)/f\}\end{aligned}$$

So, we compute remainders of $hg_5$ and $hg_4$ divided by $\{f, F\}$ and suppose we get $\{a_5, b_5\}$ and $\{a_4, b_4\}$, respectively. Then,

$$\begin{aligned}I_3 &\leftarrow \{h, (hg_5)/f, (hg_4)/f\}\\ &\equiv \{h, a_5, a_4\} \pmod{\{F\}}\\ &= \{X^2+13Y+5X+15, XY+13Y+5X+11,\\ &\quad Y^2+5Y+12X+6\}\end{aligned}$$

This is the expression of $I_3$ by the Gröbner basis.

From the above argument, we obtain Algorithm 6 for addition in Jacobian, giving details of Algorithm 5. In the following, "$\{\{c_1, c_2, \cdots, c_a\}, r\} \leftarrow$ Division$(g, G)$" denotes that we get the quotient $\{c_1, c_2, \cdots, c_a\}$ and the remainder $r$ by dividing the polynomial $g$ by the polynomial set $G$(see [CLO] for details). "$\{\{a_1, \cdots, a_i\}, r\} \leftarrow$ Coefficients$(f, r_1, \cdots, r_i)$" denotes that we get coefficients $\{a_1, \cdots, a_i\}$ and the remainder $r$ to express $f$ as a linear combination of $r_1, \cdots, r_i$. $Mono_i$ denotes the $i$-th monomial in $C_{ab}$ order ($Mono_1 = 1, Mono_2 = X, \cdots$).

**Algorithm 6 (Addition algorithm for Jacobian)**
**algorithm JacobianSum**(inputs $I_1, I_2$, output $I_3$)
$I_3 \leftarrow$ Compose$(I_1, I_2)$
$f \leftarrow$ the minimum element of $I_3$
$I_3 \leftarrow$ Reduce$(f, I_3)$
**RETURN** $I_3$

**subroutine Compose**(inputs $I_1 = \{f_1, f_2, \cdots, f_a\}, I_2 = \{g_1, g_2, \cdots, g_a\}$, output $I_3$)
$I_3 \leftarrow \{F\}$
**FOR** $i = 1$ **TO** $a$, $j = 1$ **TO** $a$ **DO**
$g \leftarrow \overline{f_i \cdot g_j}^{I_3}$
$I_3 \leftarrow \{g\} \cup I_3$
**IF** $\delta(I_3) > \delta(I_1) + \delta(I_2)$ **THEN** $I_3 \leftarrow$ Buchberger$(\delta(I_1) + \delta(I_2), I_3)$
$I_3 \leftarrow$ the set of the minimum $a$ elements of $I_3$
**RETURN** $I_3$

**subroutine Reduce**(inputs $f, I = \{f_1, f_2, \cdots, f_a\}$, output $J$)
$G \leftarrow \{f, \overline{f \cdot y}^{\{F\}}, \cdots, \overline{f \cdot y^{a-1}}^{\{F\}}\}$
**LABEL**(retry)
$J \leftarrow \{\}$
$h \leftarrow$ (random number) $\cdot f_1 +$ (random number) $\cdot f_2 + \cdots +$ (random number) $\cdot f_a$
$g \leftarrow$ Divide$(G, h)$
**FOR** $i = 1$ **TO** $a$
$\{\{c_1, c_2, \cdots, c_a\}, r\} \leftarrow$ Division$(g \cdot f_i, G)$
**IF** $r \neq 0$ **THEN GOTO** retry
$k \leftarrow c_1 + c_2 \cdot y + \cdots + c_a \cdot y^{a-1}$
$J \leftarrow J \cup \{k\}$
**RETURN** $J$

**subroutine Divide**(inputs $G, h$, output $s$)
$r_1 \leftarrow \overline{Mono_1 \cdot h}^{G}$
$s_1 \leftarrow Mono_1$
$i \leftarrow 1$
**WHILE** $r_i \neq 0$ **DO**

$i \leftarrow i+1$
$r_i \leftarrow \overline{Mono_i \cdot h}^G$
$\{\{A_1, \cdots, A_{i-1}\}, r_i\} \leftarrow \text{Coefficients}(r_i, \{r_1, \cdots, r_{i-1}\})$
$s_i \leftarrow Mono_i - \sum_{j=1}^{i-1} A_j s_j$
**RETURN** $s_i$

**subroutine Buchberger**(inputs $m, I = \{f_1, \cdots, f_s\}$, output $G = \{g_1, \cdots, g_t\}$)
$B \leftarrow \{(i,j) \mid 1 \leq i < j \leq s\}$
$G \leftarrow F$
$t \leftarrow s$
**WHILE** $B \neq \phi$ **AND** $\delta(G) > m$ **DO**
Select $(i,j) \in B$
**IF** $\text{LCM}(\text{LT}(f_i), \text{LT}(f_j)) \neq \text{LT}(f_i)\text{LT}(f_j)$ **THEN**
$S \leftarrow \overline{\text{S}(f_i, f_j)}^G$
**IF** $S \neq 0$ **THEN**
$t \leftarrow t+1; f_t \leftarrow S$
$G \leftarrow G \cup \{f_t\}$
$B \leftarrow B \cup \{(i,t) \mid 1 \leq i \leq t-1\}$
$B \leftarrow B - \{(i,j)\}$
**RETURN** $G$

How about the time complexity of Algorithm 6? It is clear that the complexity is dominated by the subroutine Buchberger. In Algorithm 6, we always know the order of $\Delta$-set of ideals before knowing their Gröbner bases, using Proposition 8. So, subroutine Buchberger in Algorithm 6 monitors the order of the $\Delta$-set of $G$ and when the order becomes equal to the input $m$, the subroutine can finish at once. Therefore, if we make a suitable scheduling for selecting $(i,j)$ from $B$ in the subroutine Buchberger, we can see that the complexity of the subroutine Buchberger, as well as the complexity of Algorithm 6, evaluated by the number of multiplications on the definition field, is $O(g^3)$.

However, our experimental results show, when the size of the definition field is large enough, e. g. scores of bits, Gröbner bases are already obtained before performing subroutine Buchberger in subroutine Compose, almost always. Therefore, even if without such scheduling, the complexity of Algorithm 6, evaluated by the number of multiplications on the finite field, is actually $O(g^3)$ with large definition fields.

# 6 Timing Results

This section shows timing results of our implementation of Algorithm 6 by C language. Table 1,Table 2 and Table 3 show running times in the case of $C_{35}$ curve,$C_{37}$ curve and $C_{2,13}$ curve, respectively.

We have chosen $C_{ab}$ curves having 160 bits Jacobian group in every case. In tables, 'simple' denotes $C_{ab}$ curves with defining equations of the form $Y^a + \alpha X^b + \beta$, and 'random' denotes randomly chosen $C_{ab}$ curves. 'Sum', 'Double' and 'Scalar' denotes addition, doubling and scalar multiplication of random point, respectively.

In discrete-log based public key cryptosystems, running times of scalar multiplication are about half of those of encryption, and are almost equal to those of decryption. From Tables 2 and 3, we see that scalar multiplication is performed in 300 milliseconds in the case of random $C_{37}$ curves, and in 167 milliseconds in random $C_{2,13}$ curves. From these results, we see that $C_{ab}$ curve cryptosystems are enough practical.

# 7 Encryption/decryption function

For convenience, we take $C_{34}$ curve $C$ over a finite field $GF(q)$ and construct encryption and decryption functions based on the Jacobian group of $C$. As the genus of $C_{34}$ curve $C$ is three, the Jacobian $J_{GF(q)}(C)$ has the order about $q^3$. So, to construct about 160 bits Jacobian, we take $q$ of about 53 bits.

From Proposition 9, a general element of $J_{GF(q)}(C)$ has the form:

$$\{a_0 + a_1 X + a_2 Y + X^2, b_0 + b_1 X + b_2 Y + XY, c_0 + c_1 X + c_2 Y + Y^2\}. \tag{7}$$

Table 1: Performance for $C_{35}$ curve.(milliseconds on 266 MHZ, PentiumII).

| | simple | random |
|---|---|---|
| Sum | 3.39 | 3.65 |
| Double | 3.76 | 4.21 |
| Scalar | 862 | 958 |

Table 2: Performance for $C_{37}$ curve.(milliseconds on 266 MHZ, PentiumII).

| | simple | random |
|---|---|---|
| Sum | 1.15 | 1.24 |
| Double | 1.15 | 1.28 |
| Scalar | 273 | 300 |

Table 3: Performance for $C_{2,13}$ curve.(milliseconds on 266 MHZ, PentiumII).

| | simple | random |
|---|---|---|
| Sum | 0.70 | 0.73 |
| Double | 0.65 | 0.68 |
| Scalar | 158 | 167 |

As we can see easily, an ideal of the form (7) is generated by the first two polynomials $a_0+a_1X+a_2Y+X^2$ and $b_0+b_1X+b_2Y+XY$, whenever $a_2 \neq 0$. When an ideal of the form (7) is randomly chosen, the probability for $a_2 = 0$ is about $1/q$. So, when the definition field is large enough, we can suppose any element of the Jacobian $J_{GF(q)}(C)$ is expressed by a vector $(a_0, a_1, a_2, b_0, b_1, b_2)$ of $6\log_2(q)$ bits, actually.

We fix an (adequate) element $j_0 = (a_0, a_1, a_2, b_0, b_1, b_2) \in J_{GF(q)}(C)$. We construct a function $C(n) = n \cdot j_0$ which, given an integer $n$, outputs $n$-times of $j_0$ using Algorithm 6. Using the function $C(n)$ as a one-way function, we get encryption and decryption function in Figure 1, just as in the case of ECC. In Figure 1, a function $X$ outputs $X(j) = a0 \mid a_1 \mid a_2$ for $j = (a_0, a_1, a_2, b_0, b_1, b_2) \in J_{GF(q)}(C)$ ('|' denotes concatenation).

Figure 1: Encryption and decryption functions on $C_{34}$ curve

Table 4 shows timings of 164 bits $C_{37}$ curve cryptosystems. In the implementation, we suppose all users share a single $C_{37}$ curve as well as a base point on it, and use precomputation table for scalar multiplication of the unique base point.

Table 4: Performance for $C_{37}$ curve cryptosystems.(milliseconds on 333 MHZ, PentiumII).

| | |
|---|---|
| encryption | 207 |
| decryption | 159 |
| sign | 48 |
| verify | 203 |

# 8 Conclusion

We have shown an efficient algorithm for addition in Jacobian group of $C_{ab}$ curves. $C_{ab}$ curves compose a wide class of algebraic curves, including elliptic and hyperelliptic curves. We hope our algorithm would enlarge the class of algebraic curves which can be used for discrete-log based cryptosystems, and step up the safety of ECC.

We are now trying to find an answer for Problem 2, that is, the algorithm to construct $C_{ab}$ curves with Jacobian of almost prime order.

# References


[Cantor 87] D.G.Cantor, "*Computing in the Jacobian of a hyperelliptic curve*," Mathematics of Computation, 48(177), pp.95-101,1987

[Koblitz 89] N.Koblitz, "*Hyperelliptic cryptosystems*," J.Cryptography,1(1989), pp.139-150

[Koblitz 97] N.Koblitz, "*A Very Easy Way to Generate Curves over Prime Fields for Hyperelliptic Cryptosytems*," Rump Talk, Crypto '97

[MCT 96] N.Matsuda,J.Chao,S.Tsujii, "*Efficient construction algorithms of secure hyperelliptic discrete logarithm problems*," IEICE ISEC96-18(1996)

[MK 93] S.Miura,N.Kamiya, "*Geometric Goppa codes on some maximal curves and their minimum distance*," in Proc. IEEE Workshop on Information Theory (Susono-shi,Japan,June 1993), pp.85-86

[Miura] S.Miura, "*Research on error correcting codes based on algebraic geometry*," Doctor thesis, University of Tokyo, 1997

[SA 98] T.Satoh, K.Araki, "*Fermat Quotients and the Polynomial Time Discrete Log Algorithm for Anomalous Elliptic Curves*," COMMENTARII MATHEMATICI UNIVERSITATIS SANCTI PAULI, vol. 47, No. 1, 81-92, 1998

[Semaev 98] I.A.Semaev, "*Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curves in characteristic p*," Math. Comp. 67, 353-356 (1998)

[Silverman] J.H.Silverman, "*The Arithmetic of Elliptic Curves*," Springer-Verlag

[Smart] P.N.Smart, "*The discrete logarithm problem on elliptic curves of trace one*," To appear in J. Cryptology

[Spallek 94] A.-M.Spallek, "*Kurven vom Geshlecht 2 und ihre Anwendung in Publick-Key-Kryptosystemen*," Doctor thesis, Universität GH Essen, 1994

[US 98] S.Uchiyama, T.Saitoh, "A Note on the Discrete Logarithm Problem on Elliptic Curves of Trace Two," IEICE Technical Report, ISEC98(1998)

[Volcheck 94] E.J.Volcheck, "*Computing in the Jacobian of a plane algebraic curve*," ANTS-I, Lecture Notes in Computer Science, vol **877**(1994), Springer-Verlag, pp. 221-233

[Hartshorne] R.Hartshorne, "*Algebraic Geometry*," Springer-Verlag

[CLO] D.Cox, J.Little, D.O'Shea, "*Ideals, Varieties, and Algorithms*," Springer-Verlag