An Addition Algorithm in Jacobian of C_{ab} Curves

Seigo Arita

Internet Systems Research Laboratories, NEC, Kanagawa, 216-8555, Japan

Abstract

Nowadays, elliptic curve cryptosystems receive attention and much effort is being dedicated to make it more and more practical. It is worthwhile to construct discrete logarithm based cryptosystems using more general algebraic curves, because it supplies more security sources for public key cryptosystems. The presented paper introduces C_{ab} curves. Roughly speaking, a curve is C_{ab} if it is non-singular in its affine part and if its singularity at infinity is "nice". C_{ab} curves compose a large family of algebraic curves, including elliptic, hyperelliptic and superelliptic curves. The paper shows an addition algorithm in Jacobian group of C_{ab} curves in three steps: firstly with a geometrical point of view, which is impractical, secondly by translating the algorithm in the language of ideals, and finally, the final algorithm in which some costly steps are removed. The paper also gives experiments that prove that the algorithm behaves well in practice.

1 Introduction

Nowadays, elliptic curve cryptosystems(ECC) receive attention and much effort is being dedicated to make it more and more practical. ECC is a public key cryptosystem based on the discrete logarithm problem on a group of points on an elliptic curve. A general algebraic curve also has a group, Jacobian group, roughly speaking, which is a group of point sets on a curve. It is worthwhile to construct discrete logarithm based cryptosystems using Jacobian group of general algebraic curves beyond elliptic or hyperelliptic curves, because it supplies more security sources for public key cryptosystems.

Suppose a family F of algebraic curves (which is bigger than a family of elliptic curves) is given. To construct a discrete logarithm based cryptosystems using the family F, we need to solve the following two basic problems.

Problem 1 Find an efficient algorithm for addition in Jacobian group of any curve in the family F.

Problem 2 Find an efficient algorithm to find a curve with Jacobian group of almost prime order in the family F.

Problem 1 is solved in the case of hyperelliptic curves [1,6], and can be dealt with rather easily in superelliptic curves [4]. Problem 2 is partially solved in hyperelliptic curves [7,11,2].

Miura[9] has found a family of algebraic curves named "C_{ab} curve" in the development of algebraic geometry codes. Roughly speaking, a curve is C_{ab} if it is non-singular in its affine part and if its singularity at infinity is "nice", in the sense that there is only one place at infinity and it is of degree 1. C_{ab} curves compose a large family of algebraic curves, including elliptic, hyperelliptic and superelliptic curves.

This paper gives a solution for Problem 1 in the case of C_{ab} curves. An algorithm for addition in Jacobian group of C_{ab} curves is given in three steps: firstly with a geometrical point of view, which is impractical, secondly by translating the algorithm in the language of ideals, and finally, the final algorithm in which some costly steps are removed. This paper also gives experiments that prove that the algorithm behaves well in practice.

2 Preliminaries

This section gives preliminaries for Jacobian group of an algebraic curve and for a Groebner basis of an ideal in a polynomial ring.

2.1 Jacobian group of an algebraic curve

Take an algebraic curve C defined over a field K. Let $\overline{K}$ be its algebraic closure. A divisor D is defined to be a formal sum $D = \sum m_i P_i$ for integers m_i and rational points(strictly, places) P_i of C over $\overline{K}$. When $m_i \geq 0$ for all i, the divisor $D = \sum m_i P_i$ is called positive. The integer $\deg(D) = \sum m_i$ is called degree of a divisor $D = \sum m_i P_i$. All divisors on a curve C compose an abelian group $\boldsymbol{D}$ under the formal addition, and all divisors of degree zero become a subgroup $\boldsymbol{D}^0$ of $\boldsymbol{D}$. Let $\boldsymbol{D}_K$ and $\boldsymbol{D}_K^0$ be invariant subgroups of $\boldsymbol{D}$ and $\boldsymbol{D}^0$ under the action of $\mathrm{Gal}(\overline{K} \mid K)$, respectively. 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 on C. 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 $\{(f) \mid f \in \overline{K}(C)\}$ compose a subgroup $\boldsymbol{P}$ of $\boldsymbol{D}^0$. The quotient group $J(C) = \boldsymbol{D}^0/\boldsymbol{P}$ is called Jacobian group of C. The invariant subgroup $J_K(C) = \boldsymbol{D}^0_K/\boldsymbol{P}_K$ of $J(C)$ under the action of $\mathrm{Gal}(\overline{K} \mid K)$ 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. By Riemann's theorem, dim $L(D) \geq \deg(D) + 1 - g$, where g denotes the genus of C.

For details, see chapter 2 of [10].

2.2 Monomial order and Groebner bases

Let $\boldsymbol{N}_0$ 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 $\alpha = (\alpha_1, \cdots, \alpha_n) \in \boldsymbol{N}_0^n$ is called a multi-degree of x^α, denoted by $\mathrm{MD}(x^\alpha)$. A well-order $<$ on $\boldsymbol{N}_0^n$ is called monomial order if $\alpha + \gamma < \beta + \gamma$ holds whenever $\alpha < \beta$ and $\gamma \in \boldsymbol{N}_0^n$. A monomial order on $\boldsymbol{N}_0^n$ 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 largest term (monomial) appearing in f with respect to the monomial order is called a leading term (monomial) of f, denoted by $\mathrm{LT}(f)(\mathrm{LM}(f))$. 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.

Fix a monomial order. A subset $G = \{g_1, \cdots, g_m\}$ of an ideal I of n-variable polynomial ring $R = K[x_1, \cdots, x_n]$ is called a Groebner basis of I when leading monomials $\mathrm{LM}(g_1), \cdots, \mathrm{LM}(g_m)$ generate an ideal $\mathrm{LM}(I)$ of leading monomials in I:

$$\mathrm{LM}(I) = (\mathrm{LM}(g_1), \cdots, \mathrm{LM}(g_m)).$$

Any ideal of n-variable polynomial ring $K[x_1, \cdots, x_n]$ has a Groebner basis. If $G = \{g_1, \cdots, g_m\}$ is a Groebner basis of an ideal I, G generates I.

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

$$\Delta(I) = \{\alpha \in \boldsymbol{N}_0^n \mid x^\alpha \notin \mathrm{LM}(I)\}.$$

Let $\delta(I)$ denote the number of elements in $\Delta(I)$. Obviously, when $\delta(I)$ is finite, $\delta(I) = \dim_K(R/I)$. So, when $\delta(I)$ is finite, it is equal to the number of points which are zeros of I, including multiplicities.

For a polynomial set $G = \{g_1, \cdots, g_m\}$, we set

$$\delta(g_1, \cdots, g_m) := \sharp\ (\boldsymbol{N}_0^n - \bigcup_{i=1}^{m} (\mathrm{MD}(\mathrm{LM}(g_i)) + \boldsymbol{N}_0^n)).$$

($\sharp S$ 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

$$G \text{ is a Groebner basis of } I \Longleftrightarrow \delta(I) = \delta(g_1, \cdots, g_m). \qquad (1)$$

Use of a Groebner 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 Groebner basis of I is equal to zero. Although there are several Groebner bases for a given ideal, the reduced Groebner basis is uniquely determined up by a given ideal. A Groebner 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 $(\mathrm{LT}(G - \{p\}))$ for all $p \in G$. In this paper, a Groebner basis is always considered in the reduced form.

For details, see chapter 2 of [3].

3 C_{ab} curve

This section, after Miura, defines C_{ab} curves and shows some properties of them[9,8].

Let C be an algebraic curve with a place P of degree one over a perfect field K. 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 numbers $-v_P(f)$ at P of $f \in L(\infty P)$ become a monoid M_P:

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

Take a minimum system $A = \{a_1, a_2, \ldots, a_t\}$ $(a_1 < a_2 \cdots < a_t)$ of generators of M_P as a monoid:

$$M_P = \boldsymbol{N}_0 a_1 + \boldsymbol{N}_0 a_2 + \cdots + \boldsymbol{N}_0 a_t = \langle A \rangle,$$

where $\boldsymbol{N}_0$ denotes the set of non-negative integers. Note $\gcd(a_1, \ldots, a_t) = 1$, since M_P is co-finite in $\boldsymbol{N}_0$.

For $A = \{a_1, \ldots, a_t\}$, define a function Ψ_A on $\boldsymbol{N}_0^t$ as

$$\Psi_A(n_1, \ldots, n_t) = \sum_{i=1}^{t} a_i n_i \quad (n = (n_i) \in \boldsymbol{N}_0^t).$$

Definition 1 (C_{ab} order) *For $m = (m_1, \ldots, m_t)$, and $n = (n_1, \ldots, n_t) \in \boldsymbol{N}_0^t$, define an order $>_A$ as*

$$\begin{aligned} m >_A n \overset{\text{def}}{\Longleftrightarrow}\ & \Psi_A(m) > \Psi_A(n) \\ & \textit{or}\ \ \Psi_A(m) = \Psi_A(n),\ m_1 = n_1, \ldots, m_{i-1} = n_{i-1}, m_i < n_i \textit{ for a } i \in [1..t]. \end{aligned}$$

Then, the order $>_A$ becomes a monomial order, called "C_{ab} order of type A".

For each $a \in \langle a_1, \ldots, a_t \rangle$, take the smallest m with respect to C_{ab} order of type A satisfying $\Psi_A(m) = a$, and put those as $B(A)$:

$$B(A) = \{\text{the smallest } m \in \boldsymbol{N}_0^t \text{ w.r.t } C_{ab} \text{ order of type } A \text{ satisfying } \Psi_A(m) = a \mid a \in \langle A \rangle\}.$$

Take a set $V(A)$ of 'minimum' elements not belonging to $B(A)$:

$$V(A) = \{l \in \boldsymbol{N}_0^t \setminus B(A) \mid l = m + n, m \in \boldsymbol{N}_0^t \setminus B(A), n \in \boldsymbol{N}_0^t \Rightarrow n = (0, 0, \ldots, 0)\}.$$

$V(A)$ is a finite set as seen later.

Theorem 2 *Let C be an algebraic curve defined over a perfect field K with a place P of degree one. Suppose M_P has a minimum system $A = \{a_1, \ldots, a_t\}$ $(a_1 < \cdots < a_t)$ of generators as a monoid. Then, the curve C has a nonsingular affine model in t-dimensional affine space defined by the equations*

$$F_m = X^m + \alpha_l X^l + \sum_{n \in B(A), \Psi_A(n) < \Psi_A(m)} \alpha_n X^n \quad (m \in V(A)) \tag{2}$$

with a unique $l \in B(A)$ satisfying $\Psi_A(m) = \Psi_A(l)$, and $\alpha_l (\neq 0), \alpha_n \in K$. The affine model has a unique point P_∞ at infinity, which corresponds to the place P.

Conversely, for $A = \{a_1, \dots, a_t\}$ *such that* $\gcd(a_1, \dots, a_t) = 1, a_1 < \cdots < a_t$, *if the affine curve defined by equations (2) is nonsingular, and equations (2) compose a Groebner basis w.r.t.* C_{ab} *order of type* A*, putting*

$$x_i = X_i \bmod \{F_m \mid m \in V(A)\} \quad (i = 1, \dots, t),$$

we have

$$\begin{aligned} -v_{P_\infty}(x_i) &= a_i \quad (i = 1, \dots, t) \\ M_{P_\infty} &= \langle A \rangle. \end{aligned}$$

In particular

$$\Psi_A(n_1, \dots, n_t) = -v_{P_\infty}(x_1^{n_1} \cdots x_t^{n_t}).$$

The affine curve $F_m = 0$ $(m \in V(A))$ obtained from $A = \{a_1, \dots, a_t\}$ $(\gcd(a_1, \dots, a_t) = 1, a_1 < \cdots < a_t)$, is called a "$C_{ab}$ curve of type A".

It is not trivial to determine $V(A)$ for a given A. For $i = 0, 1, \dots, a_1 - 1$, put

$$b_i = \min\{b \in \langle a_2, a_3, \dots, a_t \rangle \mid b \equiv i \pmod{a_1}\} \tag{3}$$

and put

$$T(A) = \{\text{the smallest } m \in N_0^t \text{ with } \Psi_A(m) = b_i \mid i = 0, 1, \dots a_1 - 1\}.$$

The set $V(A)$ is easily determined by the following proposition.

Proposition 3 *We have*

$$V(A) \subset T(A) + \{(0, \dots, 0, \check{1}^{i}, 0, \dots, 0) \mid i = 2, \dots, t\} \setminus T(A). \tag{4}$$

Proposition 4 *The genus* $g(A)$ *of a* C_{ab} *curve of type* $A = \{a_1, \dots, a_t\}$ *is given by*

$$g(A) = \sum_{i=1}^{a_1 - 1} \left[\frac{b_i}{a_1} \right].$$

In particular, when $A = \{a, b\}$,

$$g(a, b) = (a - 1)(b - 1)/2.$$

Example: $C_{3,4}$ curve

Let $A = \{3, 4\}$. By Proposition 3, we have

$$\begin{aligned}
B(A) &= \{(0,0),(1,0),(0,1),(2,0),\ldots\},\\
T(A) &= \{(0,0),(0,1),(0,2)\},\\
V(A) &= \{(0,3)\}.
\end{aligned}$$

Since $\Psi_A((0,3)) = 12$ is equal to the value of Ψ_A for $(4,0) \in B(A)$, we see that a $C_{3,4}$ curve (i.e. C_{ab} curve of type {3,4}) is a plane curve defined by a polynomial of the form

$$\begin{aligned}
Y^3 = {} & a_0X^4 + a_1XY^2 + a_2X^2Y + a_3X^3 + a_4Y^2 + a_5XY\\
& + a_6X^2 + a_7Y + a_8X + a_9 \qquad (5)
\end{aligned}$$

by Theorem 2.

Similarly a C_{ab} curve of type $A = \{a, b\}$ ($\gcd(a, b) = 1$) is a plane curve defined by a polynomial of the form

$$F(X,Y) = \sum_{0\le i\le b, 0\le j\le a, ai+bj\le ab} \alpha_{i,j}X^iY^j. \qquad (6)$$

Galbraith et al. [4] call a nonsingular plane curve with an equation of the form

$$Y^n = a_\delta X^\delta + \cdots + a_0,$$

superelliptic curves, where n is coprime with the characteristic of the definition field, and n and δ are prime to each other. Obviously, superelliptic curves are special cases of the plane C_{ab} curves.

Example: $C_{3,5,7}$ curve

Let $A = \{3, 5, 7\}$. By Proposition 3, we have

$$\begin{aligned}
B(A) &= \{(0,0,0),(1,0,0),(0,1,0),(2,0,0),(0,0,1),\ldots\},\\
T(A) &= \{(0,0,0),(0,1,0),(0,0,1)\},\\
V(A) &= \{(0,2,0),(0,1,1),(0,0,2)\}.
\end{aligned}$$

$\Psi_A((0,2,0)) = 10$ is equal to the value of Ψ_A for $(1,0,1) \in B(A)$, $\Psi_A((0,1,1))$ is equal to the value of Ψ_A for $(4,0,0) \in B(A)$, and $\Psi_A((0,0,2))$ is equal to the value of Ψ_A for $(3,1,0) \in B(A)$. So, by Theorem 2, $C_{3,5,7}$ curve is a space curve defined by three equations of the form

$$\begin{aligned} Y^2 &= a_0XZ + a_1X^3 + a_2XY + a_3Z + a_4X^2 + a_5Y + a_6X + a_7, \\ YZ &= b_0X^4 + b_1X^2Y + b_2XZ + b_3X^3 + b_4XY + b_5Z + b_6X^2 \\ &\quad + b_7Y + b_8X + b_9, \\ Z^2 &= c_0X^3Y + c_1X^2Z + c_2X^4 + c_3X^2Y + c_4XZ + c_5X^3 + c_6XY \\ &\quad + c_7Z + c_8X^2 + c_9Y + c_{10}X + c_{11}. \end{aligned} \tag{7}$$

4 An addition algorithm in Jacobian with divisors

Let C be an algebraic curve of genus g with a rational point P_∞ over a perfect field K. As seen in section 2.1, Jacobian group $J_K(C)$ of C over K is the quotient group $\boldsymbol{D}_K^0/\boldsymbol{P}_K$. When an element j in $J_K(C)$ has a representative D in $\boldsymbol{D}_K^0$, we denote $j = [D]$;

$$\begin{aligned} J_K(C) &= \boldsymbol{D}_K^0/\boldsymbol{P}_K \\ j &= [D]. \end{aligned}$$

Definition 5 *We call a divisor D of the form $D = E - m\,P_\infty$ with an effective divisor E prime to P_∞ and with some integer m between 0 and g, a semi-normal divisor.*

Lemma 6 *Every element in $J_K(C)$ is represented by a semi-normal divisor.*

Proof Let $j = [D]$ be any element in $J_K(C)$. By Riemann's theorem,

$$\dim L(D + g\,P_\infty) \geq g + 1 - g = 1.$$

So, with some nonzero function f,

$$D + g\,P_\infty + (f) \geq 0.$$

Letting $E = D + g\,P_\infty + (f)$, we have $j = [E - g\,P_\infty]$. □

In general, there are several semi-normal divisors, which represent the same element in Jacobian. However, we can determine a unique representative by using the following algorithm.

Algorithm 1
Input: a divisor $D = E - nP_\infty$ of degree 0 with an effective divisor E prime to P_∞, Output: a semi-normal divisor G equivalent to $-D$.

```
1° Find $f \in L(\infty P_\infty)$ satisfying $(f)_0 \geq E$ with the smallest pole order
   $-v_{P_\infty}(f)$ at $P_\infty$.
2° $G \leftarrow -D + (f)$
```

Proposition 7 *Algorithm 1 outputs a constant divisor for equivalent divisors.*

Proof Let $D_1 = E_1 - n_1 P_\infty$ and $D_2 = E_2 - n_2 P_\infty$ be equivalent divisors of degree 0 with effective divisors E_i $(i = 1, 2)$ prime to P_∞.

With some nonzero function λ, we have

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

For D_1, take the function $f_1 \in L(\infty P_\infty)$ satisfying $(f_1)_0 \geq E_1$ as in Algorithm 1. Then we have

$$\begin{aligned}(f_1 \lambda^{-1}) &= (f_1) - (\lambda) \\ &= (f_1)_0 - E_1 + E_2 + (n_1 + v_{P_\infty}(f_1) - n_2)\infty.\end{aligned}$$

Since $(f_1)_0 - E_1 + E_2 \geq E_2$, letting $f_2 = f_1 \lambda^{-1}$, we have

$$f_2 \in L(\infty P_\infty), \quad (f_2)_0 \geq E_2.$$

Because λ is independent from the choice of f_1 and f_2, the f_1 with the smallest pole number at P_∞ corresponds to the f_2 with the smallest pole number at P_∞. Then,

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

shows that outputs of Algorithm 1 for D_1 and D_2 are the same. □

Definition 8 *We call divisors obtained as outputs of Algorithm 1 normal divisors.*

As Algorithm 1 outputs a divisor equivalent to -1 times the input divisor, we can normalize any semi-normal divisor by applying Algorithm 1 twice. So, Lemma 6 and Proposition 7 show that

Theorem 9 *Any element in Jacobian is represented by a unique normal divisor.*

Now, addition in Jacobian can be done by normalizing the added divisors;

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 equivalent to $D_1 + D_2$.

1° *Applying Algorithm 1 for $D_1 + D_2 = (E_1 + E_2) - (n_1 + n_2) P_\infty$, get a normal divisor D'.*
2° *Applying Algorithm 1 for D', get a normal divisor 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 [12]. But to encode divisors involved in our algorithms as point sets, we need to deal with $g!$-th degree extension field of the definition field K, 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. The next section, due to this fact, realizes Algorithm 1 and 2 by ideal computations in the coordinate ring.

5 An addition algorithm in Jacobian with ideals

Let C be a C_{ab} curve of type

$$A = \{a_1, \ldots, a_t\} \quad (\gcd(a_1, \ldots, a_t) = 1, a_1 < \cdots < a_t)$$

defined over a perfect field K with equations

$$F_m = X^m + \alpha_l X^l + \sum_{n \in B(A), \Psi_A(n) < \Psi_A(m)} \alpha_n X^n \quad (m \in V(A)) \tag{8}$$

and let P_∞ be the unique point on C at infinity.

By the definition of a C_{ab} curve, we have

$$\begin{aligned} L(\infty P_\infty) &= K[x_1, x_2, \ldots, x_t] \\ &\simeq K[X_1, X_2, \ldots, X_t]/(F_m \mid m \in V(A)). \end{aligned}$$

So, the coordinate ring $R_K = K[X_1, X_2, \ldots, X_t]/(F_m \mid m \in V(A))$ is a Dedekind domain.

In general, for a Dedekind domain R, Jacobian group of Spec(R) is just the ideal class group $H(R)$ of R (Example 6.3.2 on p132 of [5]). In this case, the isomorphism Φ is given by

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

Remember that C_{ab} order is defined by the order function Ψ_A, and by Theorem 2,

$$\Psi_A(n_1, \ldots, n_t) = -v_{P_\infty}(x_1^{n_1} \cdots x_t^{n_t}).$$

So, we see that C_{ab} order puts monomials in order by pole numbers at P_∞ of them. Hence, by applying the isomorphism Φ for Algorithm 1, we get

Algorithm 3
Input: an ideal I of the coordinate ring R_K of a C_{ab} curve of type A, Output: an ideal J equivalent to the inverse ideal of I

```
1° $f \leftarrow$ the smallest $f(\neq 0) \in I$ with respect to the $C_{ab}$ order of type $A$
2° $J \leftarrow (f) : I$
```

Using Algorithm 3 twice as in the case of Algorithm 2, we get the following addition algorithm in Jacobian of C_{ab} curve.

Algorithm 4
Input: ideals I_1 and I_2 of the coordinate ring R_K of a C_{ab} curve of type A, Output: an ideal I_3 equivalent to the ideal product $I_1 \cdot I_2$.

```
1° $I \leftarrow I_1 \cdot I_2$
2° $f \leftarrow$ the smallest $f(\neq 0) \in I$ with respect to the $C_{ab}$ order of type $A$
3° $J \leftarrow (f) : I$
4° $g \leftarrow$ the smallest $g(\neq 0) \in J$ with respect to the $C_{ab}$ order of type $A$
5° $I_3 \leftarrow (g) : J$
```

Algorithm 4 can be implemented using Groebner basis with respect to C_{ab} order. For example, we show a sample code by Mathematica Ver. 3 in the appendix.

Computation of ideal quotients costs Algorithm 4 its efficiency. We remove computation of them from the addition algorithm. In Algorithm 4, we have

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

So,

$$I_1 \cdot I_2 \cdot (g) = I_1 \cdot I_2 \cdot J \cdot I_3 = (f) \cdot I_3,$$

and we see

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

Thus, we get the following addition algorithm without ideal quotients.

Algorithm 5
Input: ideals I_1 and I_2 of the coordinate ring R_K of a C_{ab} curve of type A,
Output: an ideal I_3 equivalent to the ideal product $I_1 \cdot I_2$.

```
1° $I \leftarrow I_1 \cdot I_2$
2° $f \leftarrow$ the smallest polynomial $f(\neq 0) \in I$ with respect to the $C_{ab}$ order of type $A$
3° $g \leftarrow$ the smallest polynomial $g(\neq 0)$ with respect to the $C_{ab}$ order of type $A$ s.t. $g \cdot I \subseteq (f)$
4° $I_3 \leftarrow g \cdot I / f$
```

Remember g was the smallest member of $J = (f) : I$. So, by the definition of the ideal quotient, g is the smallest polynomial satisfying $gI \subseteq (f)$.

6 Details of implementation of the addition algorithm

This section explains the details of implementation of Algorithm 5, showing an example of performing the algorithm. For a C_{34} curve on the prime field $K = \boldsymbol{F}_{17}$ with equation

$$F = Y^3 + X^4 + 1,$$

we compute the double of

$$I_1 = \{f_1 = X^2 + 14Y + 4X + 5, f_2 = XY + 3Y + 4X + 9, f_3 = Y^2 + 9Y + 16X + 2\}$$

in its Jacobian group $J_K(C)$.

In C_{34} order, monomials are put in the ascending order as follows;

$$1, X, Y, X^2, XY, Y^2, X^3, X^2Y, XY^2, \cdots. \tag{9}$$

In the below, for a polynomial f and an ideal G, let $\overline{f}^G$ denote the remainder of a polynomial f divided by an ideal G.

1° At the first step, we compute the Groebner basis of the ideal product $I = I_1 \cdot I_1$ with respect to the C_{34} order. Remeber that $\delta(I)$ means the number of points which are zeros of I, including multiplicities (section 2.2). So, $\delta(I) = \delta(I_1) + \delta(I_1) = 6$. Then, if the ideal I contains a subset of the form

$$\{X^3 + \cdots, X^2Y + \cdots, XY^2 + \cdots\},$$

it must be a Groebner basis of I (see Equation (1)).

We compute members in $I = I_1 \cdot I_1$ as follows;

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

Then,

$$I \leftarrow \{g_6, g_5, g_4\}$$

is a Groebner basis of the ideal I by the above remark.

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

3° We find the smallest polynomial $g (\neq 0)$ such that $g \cdot I \subset (f, F)$.

Computing the remainder of the product of g_5 and monomials divided by $\{f, F\}$ in the ascending order, we get

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

Moreover, computing $\overline{X^2g_5}^{\{f,F\}}$, we get the result $4X^2Y^2 + \cdots$, which leading monomial X^2Y^2 is equal to the one of $\overline{Yg_5}^{\{f,F\}}$. So,

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

Noting XY^2 is a leading monomial of $\overline{Xg_5}^{\{f,F\}}$, and repeating the similar computation, we get

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

So, we have

$$g \leftarrow X^2 + 13Y + 5X + 15.$$

4°

$$\begin{aligned}(g/f) \cdot J &= (g/f) \cdot \{g_6, g_5, g_4\} \\ &= \{g, (gg_5)/f, (gg_4)/f\}\end{aligned}$$

Getting the quotient $\{a_5, b_5\}$ and $\{a_4, b_4\}$ by dividing gg_5 and gg_4 by $\{f, F\}$, respectively, we have

$$\begin{aligned}I_3 \leftarrow &\{g, (gg_5)/f, (gg_4)/f\} \\ &\equiv \{g, a_5, a_4\} \pmod{\{F\}} \\ &= \{X^2 + 13Y + 5X + 15, XY + 13Y + 5X + 11, \\ &\quad Y^2 + 5Y + 12X + 6\}\end{aligned}$$

The right-hand side is the Groebner basis of I_3, the result of the double of I_1.

Remark

In the above, a polynomial g is computed such that gg_5 is divisible by f. Then we get such an g with $\mathrm{LM}(g) = X^2$. In this case, gg_4 is automatically divisible by f. The reason is as follows. Note that $\delta(J) = \delta((f) : I) = -v_\infty(X^3) - \delta(I) =$

3. So the smallest monomial in LM(J) (w.r.t. C_{34} order) is not greater than X^2 since X^2 is the fourth smallest monomial (See Eq.(9)). Then, if gg_4 is not divisible by f, LM(g) becomes larger than X^2, and this is impossible since g must be the smallest member in J.

Now we get Algorithm 6 giving the details of Algorithm 5. For simplicity, Algorithm 6 treats only plane C_{ab} curve. In Algorithm 6, "$\{\{c_1, c_2, \cdots, c_a\}, r\} \leftarrow \mathrm{Division}(g, G)$" 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 section 3 of chapter 2 in [3] for details). "$\{\{a_1, \cdots, a_i\}, r\} \leftarrow \mathrm{Coefficients}(f, r_1, \cdots, r_i)$" 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 ($\mathrm{Mono}_1 = 1, \mathrm{Mono}_2 = X, \cdots$).

Algorithm 6

```
algorithm JacobianSum(inputs $I_1, I_2$, output $I_3$)
    $I_3 \leftarrow \mathrm{Compose}(I_1, I_2)$
    $f \leftarrow$ the smallest element of $I_3$
    $I_3 \leftarrow \mathrm{Reduce}(f, I_3)$
    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 $I_3 \leftarrow \mathrm{Buchberger}(\delta(I_1) + \delta(I_2), I_3)$
    $I_3 \leftarrow$ the set of the smallest $a$ elements of $I_3$
    RETURN $I_3$

subroutine Reduce(inputs $f, I = \{f_1, f_2, \cdots, f_a\}$, output $J$)
    $G \leftarrow \{f, \overline{f \cdot Y}^{\{F\}}, \cdots, \overline{f \cdot Y^{a-1}}^{\{F\}}, F\}$
    LABEL(retry)
    $J \leftarrow \{\}$
    $h \leftarrow \sum_{i=1}^{a}(\text{random number}) \cdot f_i$
    $g \leftarrow \mathrm{Divide}(G, h)$
    FOR $i = 1$ TO $a$
        $\{\{c_1, c_2, \cdots, c_a\}, r\} \leftarrow \mathrm{Division}(g \cdot f_i, G)$
        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 $s$)
```

```
    $r_1 \leftarrow \overline{\mathrm{Mono}_1 \cdot h}^G$
    $s_1 \leftarrow \mathrm{Mono}_1$
    $i \leftarrow 1$
    **WHILE** $r_i \neq 0$ **DO**
        $i \leftarrow i+1$
        $r_i \leftarrow \overline{\mathrm{Mono}_i \cdot h}^G$
        $\{\{A_1, \cdots, A_{i-1}\}, r_i\} \leftarrow \mathrm{Coefficients}(r_i, \{r_1, \cdots, r_{i-1}\})$
        $s_i \leftarrow \mathrm{Mono}_i - \sum_{j=1}^{i-1} A_j s_j$
    **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** $\mathrm{LCM}(\mathrm{LT}(f_i), \mathrm{LT}(f_j)) \neq \mathrm{LT}(f_i)\mathrm{LT}(f_j)$ **THEN**
            $S \leftarrow \overline{\mathrm{S}(f_i, f_j)}^G$
            **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** reduced $G$
```

The subroutine Compose corresponds to the first step of Algorithm 5. It computes the Groebner basis of the ideal product $I_3 = I_1 \cdot I_2$ for ideals I_1 and I_2 of the coordinate ring $K[X,Y]/(F(X,Y))$. In the subroutine, we compute the order of Δ-set $\delta(I)$ of an ideal I (for a definition of Δ-set, see section 2.2). Only in the case that $\delta(I_1)+\delta(I_2) \neq \delta(I_3)$, we need the subroutine Buchberger to obtain Groebner basis of I_3.

In general, the complexity of Buchberger algorithm may be quite huge. However, in our situation, the monomial order C_{ab} order has the order function $\Psi_{a,b}$, and forms of ideals involved are quit simple. So, Buchberger algorithm works quite efficiently. Let g be the genus of C_{ab} curve defined over a finite field of q elements. First, I_3 is composed of polynomials whose leading monomials are the about $3g$-th monomials among all the monomials. To obtain Groebner basis of I_3, we need to find polynomials in I_3, whose leading monomials are the about $2g$-th monomials. In WHILE loop of Buchberger algorithm, using notations in subroutine Buchberger, if we can make an ideal schedule for the choice of polynomials f_i, f_j for the computation $S(f_i, f_j)$, $S = \overline{S(f_i, f_j)}^G$ is not trivial and its leading monomial is strictly smaller than those of members in G. This means that WHILE loop should finish in $O(g)$ repeats. So, the complexity of Buchberger algorithm in our situation with the ideal schedule is $O(q^2g^3)$. However, our experimental results show, when the size of the definition field is large enough and the genus is small enough, the complexity of Buchberger algorithm is not far from $O(q^2g^3)$ with a hand-made schedule.

The subroutine Divide corresponds to the third step of Algorithm 5. For a random linear combination h of f_i, it compute the smallest polynomial s with respect to C_{ab} order, such that $hs \in \langle G \rangle$ just as in the above example. The subroutine Divide is essentially same as the Gaussian elimination among g variables. So, the complexity of Divide is $O(q^2g^3)$.

The subroutine Reduce corresponds to the fourth step of Algorithm 5. The reason why we choose a random linear combination of f_i for h is just a heuristic. As seen in the above example, it seems that the fact

$$g \cdot (\text{a random linear combination of } f_i) \in (f)$$

is sufficient for $gI \subset (f)$ when the size of the definition field is large enough and the genus is small enough. Our experimental results support the heuristic.

Note that the Groebner basis of the principal ideal (f) in the coordinate ring $K[X,Y]/(F)$ is given by

$$\{f, \overline{f \cdot Y}^{\{F\}}, \cdots, \overline{f \cdot Y^{a-1}}^{\{F\}}, F\},$$

since the leading term Y^a of $F(X,Y)$ is prime to X.

Finally, we show timing results of our implementation of Algorithm 6 by C language. Table 1, Table 2 and Table 3 show running times on 266MHz Pentium II, for C_{35} curve, C_{37} curve and $C_{2,13}$ curve, respectively. In each case, C_{ab} curve has the real size parameter in cryptographical applications, that is, C_{ab} curves have 160 bits Jacobian groups. 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 of two random elements, doubling a random element and multiplication of a random element by a 160 bits random integer, respectively. These results prove that the algorithm behaves well in practice.

Table 1
Timing result for C_{35} curve (ms on 266MHZ,PentiumII)

	simple	random
Sum	3.39	3.65
Double	3.76	4.21
Scalar	862	958

Table 2
Timing result for C_{37} curve (ms on 266MHZ,PentiumII)

	simple	random
Sum	1.15	1.24
Double	1.15	1.28
Scalar	273	300

Table 3
Timing result for $C_{2,13}$ curve (ms on 266MHZ,PentiumII)

	simple	random
Sum	0.70	0.73
Double	0.65	0.68
Scalar	158	167

Acknowledgements

I wish to thank S. Miura and R. Matsumoto for their helpful and suggesting discussions with me. I also wish to thank anonymous referees for their a lot of efforts to help me arrive at the final version of the paper.

References

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

[2] J.Chao,K.Matsuo,H.Kawashiro,S.Tsujii "Construction of Hyperelliptic Curves with CM and Its Application", Asiacrypt 2000, Advances in Cryptology, LNCS 1976, Springer, 2000.

[3] D.Cox, J.Little, D.O'Shea, "Ideals, Varieties, and Algorithms", Springer-Verlag, 1992.

[4] S.D.Galbraith, S.M.Paulus, and N.P.Smart "Arithmetic on Superelliptic Curves", J. Cryptology (1999) 12, 193-196.

[5] R.Hartshorne, “Algebraic Geometry”, Springer-Verlag, 1977.

[6] N.Koblitz, “Hyperelliptic cryptosystems”, J.Cryptography,1(1989), pp.139-150

[7] N.Koblitz, “A Very Easy Way to Generate Curves over Prime Fields for Hyperelliptic Cryptosystems”, Rump Talk, Crypto ’97

[8] R. Matsumoto, “The Cab Curve — a generalization of the Weierstrass form to arbitrary plane curves”, http://www.rmatsumoto.org/cab.html

[9] S. Miura, “Linear Codes on Affine Algebraic Curves”, Trans. of IEICE, vol. J81-A, No. 10, 1398-1421, Oct. 1998.

[10] J.H.Silverman, “The Arithmetic of Elliptic Curves”, Springer-Verlag

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

[12] 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

A Sample code

Here is a sample code of Algorithm 4 with Mathematica Ver. 3.

```
(* Addition Algorithm in Jacobian of C_{a,b} curve *)

(* parameter *)

Var = {X,Y,Z};
p = 83;
OrderMatrix = {{3,5,7},{-1,0,0},{0,-1,0}};  (* C_{357} *)
OrderMatrix1 = {{1,0,0,0},{0,3,5,7},{0,-1,0,0},{0,0,-1,0}};  (* C_{357} *)
n = 650496;

DefIdeal = {
    64 + 4 X + 30 X^2 + 30 X^3 + 76 Y + 75 X Y + Y^2 + 52 Z + 4 X Z,
10 + 27 X + 16 X^2 + 6 X^3 + 44 X^4 + 69 Y + 16 X Y + 27 X^2 Y
    + 31 Z + X Z + Y Z,
22 + 32 X + 77 X^2 + 30 X^3 + 11 X^4 + 17 Y + 25 X Y + 3 X^2 Y
    + 72 X^3 Y + 45 Z + 32 X Z + 76 X^2 Z + Z^2 };

G = {X-2,Y-33, Z+21}; (* a point on the curve *)
```

```
(* polynomial library *)

LT[f_] :=
    MonomialList[f, Var, Modulus->p, MonomialOrder->OrderMatrix][[1]]

LC[f_] := LT[f] /. ((#->1)& /@ Var)

LM[f_] := LT[f] / LC[f]

GBasis[J_List] :=
    GroebnerBasis[J,Var,Modulus->p,MonomialOrder->OrderMatrix]

GDivision[f_, J_List] :=
    PolynomialReduce[f, J, Var,Modulus->p,MonomialOrder->OrderMatrix]

GRemainder[f_, J_List] := GDivision[f,J][[2]];

IdealIntersection[I_List, J_List] :=
    Module[{t,G},
        G = GroebnerBasis[Union[t*I,(1-t)*J],Prepend[Var,t],Modulus->p,
                 MonomialOrder->OrderMatrix1];
        G = Select[G, (# == (# /.{t->0}))&]
    ]

IdealQuotient[I_List, f_] :=
    Module[{G},
        G = IdealIntersection[I, {f}];
        G = GDivision[#, {f}]& /@ G;
        G = First /@ G;
        Union @@ G
    ] /; PolynomialQ[f]

IdealQuotient[I_List, J_List] :=
    Module[{R},
        R = IdealQuotient[I, First[J]];
        Do[
            R = IdealIntersection[R, IdealQuotient[I, J[[i]]]],
            {i, 2, Length[J]}
        ];
        R
    ]

(* main *)
```

```
JPower[n_Integer, I1_List] :=
    Module[{i=n,J1=I1,R={1}},
        While[i > 0,
            If[OddQ[i], R=JSum[R,J1];i=(i-1)/2, i=i/2];
            If[i>0, J1=JSum[J1,J1]]
        ];
        R
    ]

JSum[I1_List, I2_List] :=
    Module[{I3,f},
        I3 = JCompose[I1,I2];
        f = First[I3];
        I3 = JReduce[f, I3];
        f = First[I3];
        JReduce[f, I3]
    ]

IdealProduct[I1_List, I1_List] :=
    Module[{f, I3=DefIdeal},
        Do[
            f = I1[[i]] I1[[j]] // PolynomialMod[#,p]&;
            f = GRemainder[f, I3];
            If[ f =!= 0, I3 = Prepend[I3, f] ],
            {i, 1, Length[I1]}, {j, 1, i}
        ];
        I3
    ]

IdealProduct[I1_List, I2_List] :=
    Module[{f, I3=DefIdeal},
        Do[
            f = I1[[i]] I2[[j]] // PolynomialMod[#,p]&;
            f = GRemainder[f, I3];
            If[ f =!= 0, I3 = Prepend[I3, f] ],
            {i, 1, Length[I1]}, {j, 1, Length[I2]}
        ];
        I3
    ]

JCompose[I1_List, I2_List] :=
    Module[{I3},
        I3 = IdealProduct[I1, I2];
        I3 = GBasis[I3]
    ]
```

```
JReduce[f_, I_List] := IdealQuotient[Prepend[DefIdeal,f], I] /; PolynomialQ[f]
```

In the code, we treat a $C_{3,5,7}$ curve on a prime field for $p = 83$ with equations

$$\begin{aligned}
0 &= 64 + 4X + 30X^2 + 30X^3 + 76Y + 75XY + Y^2 + 52Z + 4XZ,\\
0 &= 10 + 27X + 16X^2 + 6X^3 + 44X^4 + 69Y + 16XY + 27X^2Y + 31Z\\
&\quad + XZ + YZ,\\
0 &= 22 + 32X + 77X^2 + 30X^3 + 11X^4 + 17Y + 25XY + 3X^2Y + 72X^3Y\\
&\quad + 45Z + 32XZ + 76X^2Z + Z^2.
\end{aligned}$$

The order of the Jacobian is $n = 650496$. For example, computing n times the point $G = \{X - 2, Y - 33, Z + 21\}$ on the curve, we get

```
JPower[n,G] = {1}
```