PAPER *Special Section on Discrete Mathematics and Its Applications*

Baby Step Giant Step Algorithms in Point Counting of Hyperelliptic Curves

Kazuto MATSUO[†a)], **Jinhui CHAO**[††], ***Regular Members,***
and **Shigeo TSUJII**[†††], ***Honorary Member***

SUMMARY Counting the number of points of Jacobian varieties of hyperelliptic curves over finite fields is necessary for construction of hyperelliptic curve cryptosystems. Recently Gaudry and Harley proposed a practical scheme for point counting of hyperelliptic curves. Their scheme consists of two parts: firstly to compute the residue modulo a positive integer m of the order of a given Jacobian variety, and then search for the order by a square-root algorithm. In particular, the parallelized Pollard's lambda-method was used as the square-root algorithm, which took 50CPU days to compute an order of 127 bits. This paper shows a new variation of the baby step giant step algorithm to improve the square-root algorithm part in the Gaudry-Harley scheme. With knowledge of the residue modulo m of the characteristic polynomial of the Frobenius endomorphism of a Jacobian variety, the proposed algorithm provides a speed up by a factor m, instead of $\sqrt{m}$ in square-root algorithms. Moreover, implementation results of the proposed algorithm is presented including a 135-bit prime order computed about 15 hours on Alpha 21264/667 MHz and a 160-bit order.

key words: *hyperelliptic curve cryptosystems, hyperelliptic curves, point counting algorithms, baby step giant step algorithms, Gaudry-Harley scheme*

1. Introduction

Security of hyperelliptic curve cryptosystems depends in an essential way on the orders of Jacobian varieties of the hyperelliptic curves used in the systems. In particular, it is believed that if a small genus curve is used, and the order of its Jacobian variety contains a prime number with a small cofactor, coprime to the characteristic of the definition field and immune to the Weil/Tate pairing reduction [9], then the hyperelliptic curve cryptosystem can resist all known attacks except maybe the Weil-Descent attack [10].

Therefore, the order counting for Jacobian varieties of random hyperelliptic curves is one of the most important problems in construction of hyperelliptic curve cryptosystems.

Manuscript received August 26, 2002.
Manuscript revised November 11, 2002.
Final manuscript received January 9, 2003.

[†]The author is with the Research and Development Initiative, Chuo University, Tokyo, 162-8473 Japan.

[††]The author is with the Department of Electrical, Electronic and Communication Engineering, Chuo University, Tokyo, 112-8851 Japan.

[†††]The author is with the Department of Information and System Engineering, Chuo University, Tokyo, 112-8851 Japan.

a) E-mail: mats@m.ieice.org

Recently several researches have been reported on point counting algorithms. In particular, efficient algorithms [6], [12], [13], [15], [19], [20] have been proposed for curves over small characteristic finite fields. Using these algorithms, it is possible to compute orders of Jacobian varieties in sizes for cryptographic usage (e.g. 160 bits) over such fields,

On the other hand, the situation is quite different for point counting of curves over finite fields with arbitrary characteristics. Although a number of theoretical results such as [1], [16], [17], [26] have been known, it is only until very recent that a practical point counting scheme for curves over large characteristic finite fields is proposed and implemented by Gaudry and Harley [11], [14].

The Gaudry-Harley scheme consists of two parts: the first part is to compute the residue modulo a positive integer m of the order of a given Jacobian variety. The second part is to search for the order by an algorithm square root complexity using its modulo m residue. (We will misuse "square-root algorithm" referring to these search algorithms hereafter.) This is a natural generalization of the point counting scheme of elliptic curves proposed in [25] to hyperelliptic curves. In particular, the parallelized Pollard's lambda-method was used as the searching algorithm. It seemed that the square-root algorithm is the most time consuming part in the Gaudry-Harley scheme. For an example, this part took actually 50CPU days to compute an order of 127 bits.

This paper proposes an improvement of the baby step giant step algorithm, and applies it as the square-root algorithm in the Gaudry-Harley scheme. It is shown that with knowledge of the residue modulo m of the characteristic polynomial of the Frobenius endomorphism of a Jacobian variety, the proposed search algorithm provides a speed up by a factor m, instead of $\sqrt{m}$ in the usual square-root algorithms. (All complexity estimates hereafter are in terms of operations in the Jacobians.) Moreover, implementation results of the proposed algorithm are presented, including a 135-bit order is computed about 15 hours on Alpha 21264/667 MHz and a 160-bit order.

2. Hyperelliptic Curves over Finite Fields and the Orders of their Jacobian Varieties

Let p be an odd prime number, $\mathbf{F}_q$ a finite field of order q with $\mathrm{char}(\mathbf{F}_q) = p$. Let g be a positive integer. Then a genus g hyperelliptic curve $C/\mathbf{F}_q$ is defined as follows:

$$C : Y^2 = F(X), \quad F(X) = X^{2g+1} + f_{2g}X^{2g} + \cdots + f_0, \tag{1}$$

where $f_i \in \mathbf{F}_q$, $\mathrm{disc}\,(F) \neq 0$. We will restrict ourselves to $g = 2$ cases hereafter.

Let $\mathbf{J}_C$ be the Jacobian variety of C, $\mathbf{J}_C(\mathbf{F}_q)$ its $\mathbf{F}_q$-rational points. It is known that $\mathbf{J}_C(\mathbf{F}_q)$ is a finite Abelian group, so that discrete logarithm based cryptosystems can be constructed on it.

The characteristic polynomial $\chi_q(X)$ of the q-th power Frobenius endomorphism of $\mathbf{J}_C$ is given as follows [17], [29]:

$$\chi_q(X) = X^4 - s_1X^3 + s_2X^2 - s_1qX + q^2, \tag{2}$$

where $s_i \in \mathbf{Z}$. Then the order $\#\mathbf{J}_C(\mathbf{F}_q)$ of $\mathbf{J}_C(\mathbf{F}_q)$ can be obtained as

$$\#\mathbf{J}_C(\mathbf{F}_q) = \chi_q(1) = q^2 + 1 - s_1(q+1) + s_2 \tag{3}$$

from $\chi_q(X)$ [29]. It is well known that $\#\mathbf{J}_C(\mathbf{F}_q)$ is bounded within the Hasse-Weil range:

$$L_o \leq \#\mathbf{J}_C(\mathbf{F}_q) \leq H_o, \quad L_o = \left\lceil (\sqrt{q}-1)^4 \right\rceil, H_o = \left\lfloor (\sqrt{q}+1)^4 \right\rfloor. \tag{4}$$

3. The Gaudry-Harley Scheme

In this section a rough description of the Gaudry-Harley scheme and its implementation results are given according to [11], [14]. (See [11], [14] for further details.)

In Algorithm 1, we show an outline of the Gaudry-Harley scheme.

Algorithm 1: Gaudry-Harley point counting scheme.

```
Input: A genus 2 HEC $C/\mathbf{F}_q$
Output: $\#\mathbf{J}_C(\mathbf{F}_q)$
 1: Compute $\#\mathbf{J}_C(\mathbf{F}_q) \bmod 2^e$ by the halving algorithm
 2: for prime numbers $l = 3, 5, \dots, l_{max}$ do
 3:     Compute $\chi_q(X) \bmod l$ by a Schoof-like algorithm
 4:     Compute $\#\mathbf{J}_C(\mathbf{F}_q) \bmod l$ from $\chi_q(X) \bmod l$
 5: end for
 6: Compute $\chi_q(X) \bmod p$ by using the Cartier-Manin operator
 7: Compute $\#\mathbf{J}_C(\mathbf{F}_q) \bmod p$ from $\chi_q(X) \bmod p$
 8: Compute $\#\mathbf{J}_C(\mathbf{F}_q) \bmod m, m = 2^e \cdot 3 \cdots l_{max} \cdot p$ by CRT
 9: Compute $\#\mathbf{J}_C(\mathbf{F}_q)$ by a square root algorithm using
    $\#\mathbf{J}_C(\mathbf{F}_q) \bmod m$
```

The capability of Schoof-like algorithm therefore the largest value of l_{max} which can be computed in Step 2 of Algorithm 1 is subject to many factors, such as the size of $\mathbf{F}_q$. To construct secure curves, $l_{max} = 13$ is a reasonable estimate at the present. Since the computation of the Cartier-Manin operator in Step 6 costs exponential time in $\log p$, it can not be applied to curves over prime fields of large sizes in cryptographic usage. Therefore, Step 6 and 7 will be skipped when the scheme is applied to curves over prime fields. For the details of the Cartier-Manin operator and its computation, see [11], [14], [22], [23], [31].

Gaudry and Harley computed the orders of the Jacobian varieties of hyperelliptic curves over a 64-bit prime field and a degree 3 extension of a 16-bit prime field. The orders are 127 bits and 128 bits respectively, and these results seem to be present records of point counting of hyperelliptic curves over prime fields and over large characteristic fields.

However, construction of secure hyperelliptic curve for cryptographic usage is still out of reach. For an example, it took 50CPU days in Step 9 when the 127-bit order is computed. Besides, point counting algorithms usually have to be repeated before a secure curve is found.

4. A Baby Step Giant Step Algorithm Using $\#\mathbf{J}_C(\mathbf{F}_q) \bmod m$

It is known that compared with the baby step giant step algorithm, the parallelized Pollard's lambda-method used in Step 9 of Algorithm 1 has merits such as it can be parallelized and needs only constant amount of memories. Both algorithms have essentially the same computational complexities in CPU time and can be used in Step 9 in Algorithm 1.

In this section, as a preliminary to the following sections, we describe a modified version of the standard baby step giant step algorithm which can be used in Step 9 in Algorithm 1. This algorithm is an extension of the algorithm that has been applied for point counting of elliptic curves in [25].

From Step 8 of Algorithm 1, one knows $N_r \in \mathbf{Z}$ such that

$$\#\mathbf{J}_C(\mathbf{F}_q) = N_r + mN_m, 0 \leq N_r < m. \tag{5}$$

Therefore, $\#\mathbf{J}_C(\mathbf{F}_q)$ can be obtained by searching for N_m among

$$\lfloor L_o/m \rfloor \leq N_m \leq \lfloor H_o/m \rfloor. \tag{6}$$

Now we set $n \approx \sqrt{R_o}$, where

$$R_o \approx (H_o - L_o)/m = 8q^{3/2}/m + O(q/m). \tag{7}$$

Then the candidates of $N_m = i + nj$ can be obtained by finding (i, j) such that

$$(N_r + mi)D = -mnjD \tag{8}$$

for all $D \in \mathbf{J}_C(\mathbf{F}_q)$ by searching for a collision between the LHS and the RHS of (8) among

$$0 \le i < n, \tag{9}$$
$$\left\lfloor \frac{L_o}{mn} \right\rfloor - 1 \le j \le \left\lfloor \frac{H_o}{mn} \right\rfloor. \tag{10}$$

Assuming $\#\mathbf{J}_C(\mathbf{F}_q)$ is a prime order and q is large enough, one can compute $\#\mathbf{J}_C(\mathbf{F}_q)$ from the pair (i, j) obtained by the above computation as follows:

$$\#\mathbf{J}_C(\mathbf{F}_q) = N_r + m(i + nj). \tag{11}$$

When $\#\mathbf{J}_C(\mathbf{F}_q)$ is not a prime number, to compute $\#\mathbf{J}_C(\mathbf{F}_q)$ one may have to calculate $N_r + m(i+nj)$ for different divisors $D \in \mathbf{J}_C(\mathbf{F}_q)$ and their least common multiple [5], [14]. However, it is possible to test whether $\#\mathbf{J}_C(\mathbf{F}_q)$ is a prime order or not once the $N_r+m(i+nj)$ is found for a single $D \in \mathbf{J}_C(\mathbf{F}_q)$. Thus, in construction of secure curves, one can abandon the curves with non-prime orders and only looking for prime orders.

This algorithm costs $O(q^{3/4}/\sqrt{m})$ according to (7).

5. An Improved Baby Step Giant Step Algorithm

It can be noticed that in Algorithm 1 in Sect. 3, the residues modulo m of s_i for $\chi_q(X)$ in (2) as well as $\#\mathbf{J}_C(\mathbf{F}_q) \bmod m$ can also be obtained by using either Schoof-like algorithm or the Cartier-Manin operator.

In this section, we propose an improved baby step giant step algorithm which makes effectively use of the residues $s_i \bmod m$. In such a way, this algorithm speeds up the baby step giant step algorithm in Sect. 4 and other square-root algorithms by a factor $O(\sqrt{m})$.

First of all, the boundaries of s_i are given by Lemma 1.

Lemma 1. s_1 is bounded by

$$s_{1l} \le s_1 \le s_{1u}, \tag{12}$$
$$s_{1l} = -\lfloor 4\sqrt{q} \rfloor, \ s_{1u} = \lfloor 4\sqrt{q} \rfloor,$$

and s_2 is bounded by

$$s_{2l} \le s_2 \le s_{2u}, \tag{13}$$
$$s_{2l} = \lceil 2\sqrt{q}|s_1| - 2q \rceil, \ s_{2u} = \left\lfloor \frac{1}{4}s_1^2 + 2q \right\rfloor.$$

Proof. The boundaries in (12) are well known, cf. [29]. The upper bound in (13) is shown by Elkies [7] without proof, and the lower bound was pointed out to the authors by Fumiyuki Momose. Bellow is a proof follows him for (13).

Let $\alpha, \alpha^\rho, \beta, \beta^\rho$ be the eigenvalues of the q-th power Frobenius endomorphism of $\mathbf{J}_C$, where ρ is complex conjugate. Let $a_1 = \alpha + \alpha^\rho$ and $a_2 = \beta + \beta^\rho$. Then

$$s_1 = a_1 + a_2,$$
$$s_2 = a_1 a_2 + 2q,$$

$a_i \in \mathbf{R}$, and $|a_i| \le 2\sqrt{q}$, because $|\alpha|=|\beta|= \sqrt{q}$ [29]. Therefore,

$$\begin{aligned} 0 \le \frac{1}{4}(a_1 - a_2)^2 &= \frac{1}{4}(a_1 + a_2)^2 - a_1 a_2 \\ &= \frac{1}{4}s_1^2 - s_2 + 2q \end{aligned} \tag{14}$$

then $s_2 \le s_{2u}$.

If $a_1 \ge 0$ and $a_2 \ge 0$,

$$s_2 - 2\sqrt{q}s_1 + 2q = (a_1 - 2\sqrt{q})(a_2 - 2\sqrt{q}) \ge 0$$

then $s_2 \ge s_{2l}$. If $a_1 < 0$ and $a_2 < 0$,

$$s_2 + 2\sqrt{q}s_1 + 2q = (a_1 + 2\sqrt{q})(a_2 + 2\sqrt{q}) \ge 0$$

then $s_2 \ge s_{2l}$ also. Finally, assume $a_1 \ge 0$ and $a_2 < 0$. Then

$$-s_2 + 2\sqrt{q}s_1 - 2q = (a_1 - 2\sqrt{q})(-a_2 + 2\sqrt{q}) \le 0$$

which leads to $s_2 \ge s_{2l}$ again, if $s_1 \ge 0$. Moreover we also have

$$-s_2 - 2\sqrt{q}s_1 - 2q = -(a_1 + 2\sqrt{q})(a_2 + 2\sqrt{q}) \le 0$$

which leads to $s_2 \ge s_{2l}$, if $s_1 \le 0$. □

Remark 1. This sharper bound of (13) is used in the algorithm 2 shown belows. The practical speed of the algorithm is roughly three times faster than using the well-known bound $|s_2| \le 6q$, cf. [17].

Now let $s_i' \in \mathbf{Z}$ such that

$$0 \le s_i' < m, \tag{15}$$
$$s_1 = s_1' + mt_1, t_1 \in \mathbf{Z}, \tag{16}$$
$$s_2 = s_2' + mt_2', t_2' \in \mathbf{Z}. \tag{17}$$

Then t_1 in (16) is bounded by

$$\left\lfloor \frac{s_{1l}}{m} \right\rfloor \le t_1 \le \left\lfloor \frac{s_{1u}}{m} \right\rfloor \tag{18}$$

due to (12) and t_2' in (17) is bounded by

$$\left\lfloor \frac{s_{2l}}{m} \right\rfloor \le t_2' \le \left\lfloor \frac{s_{2u}}{m} \right\rfloor \tag{19}$$

due to (13). Moreover let t_2, t_3 be integers such that

$$t_2' = t_2 + nt_3, \ t_2, t_3 \in \mathbf{Z}, \tag{20}$$
$$0 \le t_2 < n \tag{21}$$

for a positive integer n, then t_3 is bounded by

$$\left\lfloor \frac{s_{2l}}{mn} \right\rfloor - 1 \le t_3 \le \left\lfloor \frac{s_{2u}}{mn} \right\rfloor \tag{22}$$

due to (19).

Consequently we have

$$\begin{aligned} \#\mathbf{J}_C(\mathbf{F}_q) = q^2&+1-s_1'(q+1) \\ &+ s_2'-m(q+1)t_1+mt_2+mnt_3 \end{aligned} \tag{23}$$

by substituting (16), (17), (20) into (3). Hence $\#\mathbf{J}_C(\mathbf{F}_q)$ can be computed by finding (t_1, t_2, t_3) satisfying

$$\begin{aligned}(q^2+1-s_1'(q+1)+s_2'-m(q+1)t_1+mnt_3)D \\ = -mt_2D \end{aligned} \quad (24)$$

for all $D \in \mathbf{J}_C(\mathbf{F}_q)$ in the ranges of (18), (21), (22). These computations are carried out by collision searching between the LHS and the RHS of (24).

Next we determine the most effective value of n.

The number of the pairs (s_1, s_2) satisfying (12) and (13) is roughly $32q^{3/2}/3$ because

$$\begin{aligned}&\int_{s_{1l}}^{s_{1u}} \frac{1}{4}s_1^2 + 2q - (2\sqrt{q}|s_1| - 2q)\mathrm{d}s_1 \\ &= \left[s_1(\frac{1}{12}s_1^2 - \sqrt{q}|s_1| + 4q) \right]_{s_{1l}}^{s_{1u}} \approx 32q^{3/2}/3. \end{aligned} \quad (25)$$

Therefore the number S of the triples (t_1, t_2, t_3) is

$$S \approx \frac{32q^{3/2}}{3m^2}. \quad (26)$$

Now we set n as

$$n_0 \approx \sqrt{S} = \frac{4\sqrt{6}q^{3/4}}{3m} \quad (27)$$

then the number of point additions for all (t_1, t_2, t_3) satisfying (12) and (13) is roughly n on both the LHS and the RHS of (24). The algorithm under this setting works most efficiently. Therefore the computational complexity of the algorithm is $O(q^{3/4}/m)$ according to (27). Thus, using this algorithm in the square-root algorithm part of the Gaudry-Harley scheme, the computation of $\#\mathbf{J}_C(\mathbf{F}_q)$ is $O(\sqrt{m})$ times faster than by the original baby step giant step algorithm in Sect. 4 and other square-root algorithms.

Algorithm 2 shows an application of the proposed algorithm in the Gaudry-Harley point counting scheme to compute prime orders.

Remark 2. If $q \le 131$ then $H_0 > 2L_0$ and it is possible that Algorithm 2 outputs a wrong answer. Namely if one chooses $D \in \mathbf{J}_C[r]$ in Step 3 from $\mathbf{J}_C$ and $\#\mathbf{J}_C(\mathbf{F}_q) = rs$ where $r > L_0$ is a prime number and $s > 1$ then Algorithm 2 outputs r. However, this situation can be easily checked out.

Remark 3. Although s_i obtained from the proposed algorithm are not always correct, the correct values can be easily obtained by an additional test using a quadratic twist of the curve. See [7], [18].

Remark 4. The algorithm proposed in this section costs $O(q^{g(g+1)/8}/m^{g/2})$ time for genus g curves. On the other hand the original algorithm shown in the section 4 costs $O(q^{(2g-1)/4}/\sqrt{m})$. Therefore, for genus $g > 2$ curves, the proposed algorithm is faster than the original algorithm in the cases of $m = \Omega(q^{\epsilon+(g-2)/4})$ for any $\epsilon > 0$. In the other cases, for the proposed algorithm to outperform the original algorithm, one needs to e.g. make further usage of the properties described in the Remark 3.

Algorithm 2: An improved baby step giant step algorithm for finding prime order curves.

```
Input: A genus 2 HEC $C/\mathbf{F}_q$ with $q \ge 137$, $m, s_1', s_2' \in \mathbf{Z}_{>0}$
    such that $s_i \equiv s_i' \bmod m$ and $0 \le s_i' < m$
Output: $\#\mathbf{J}_C(\mathbf{F}_q)$, if it is a prime number
 1: $n \leftarrow \left\lfloor 4\sqrt{6}q^{3/4}/(3m) \right\rceil$
 2: $l \leftarrow q^2+1-s_1'(q+1)+s_2'$
 3: Choose a random $D \in \mathbf{J}_C(\mathbf{F}_q)\backslash\{0\}$
 4: $B \leftarrow \{(b_j = -jmD, j) \mid 0 \le j < n\}$
 5: Sort the table $B$ by the entry $b_j$
 6: $D_1 \leftarrow lD$
 7: for $i = -\lceil \lfloor 4\sqrt{q} \rfloor /m \rceil \ldots \lfloor 4\sqrt{q}/m \rfloor$ do
 8:   $D_2 \leftarrow D_1 - im(q+1)D$
 9:   $s_1 \leftarrow s_1' + im$
10:   for $k = \lfloor (\lceil 2\sqrt{q}|s_1| \rceil - 2q)/(mn) \rfloor - 1 \ldots \lfloor (\lfloor s_1^2/4 \rfloor + 2q)/(mn) \rfloor$ do
11:     $D_3 \leftarrow D_2 + kmnD$
12:     if $^{\exists}j$ such that $b_j = D_3$ then
13:       $l \leftarrow l + (-i(q+1)+j+kn)m$
14:       if $l =$ a prime number then
15:         Output $l$ as $\#\mathbf{J}_C(\mathbf{F}_q)$ and terminate
16:       else
17:         $\#\mathbf{J}_C(\mathbf{F}_q)$ is not a prime number and terminate
18:       end if
19:     end if
20:   end for
21: end for
```

6. Implementation and Construction of Prime Order Curves

Algorithm 2 is implemented to construct genus 2 prime order hyperelliptic curves. In order to obtain $s_i \bmod m$, we also implemented the computation of both $s_i \bmod 2$ and $s_i \bmod p$ by the Cartier-Manin operator.

6.1 Computation of $s_i \bmod 2$

For construction of prime order curves, one can actually use only curves of which the residues of s_i modulo 2 are determined by $2 \nmid \#\mathbf{J}_C(\mathbf{F}_q)$, according to the following lemma.

Lemma 2. The following conditions are equivalent:

1. $\#\mathbf{J}_C(\mathbf{F}_q)$ is odd.
2. F is irreducible over $\mathbf{F}_q$.
3. Each s_i is odd.

Proof. 1. $\Leftrightarrow$ 2.: See [18].
3. $\Rightarrow$ 1.: $\#\mathbf{J}_C(\mathbf{F}_q)$ is odd iff s_2 is odd from (3), so that $\#\mathbf{J}_C(\mathbf{F}_q)$ is odd, if each s_i is odd.
1. $\Rightarrow$ 3.: If $2 \mid s_1$ then $2 \mid \#\mathbf{J}_C(\mathbf{F}_{q^3})$ from

$$\begin{aligned}&\#\mathbf{J}_C(\mathbf{F}_{q^3}) \\ &\quad = \#\mathbf{J}_C(\mathbf{F}_q) \times (q^4-q^2+1+2s_2^2-\#\mathbf{J}_C(\mathbf{F}_q)s_2 \\ &\qquad + s_1(q^3+1-2q(q+1)+s_1(q^2-q+1))).\end{aligned}$$

Then there exists $\alpha \in \mathbf{F}_{q^6}$ such that $F(\alpha) = 0$, so that α belongs to either $\mathbf{F}_{q^2}$ or $\mathbf{F}_{q^3} \backslash \mathbf{F}_q$ because $\deg F = 5$. On one hand, if $\alpha \in \mathbf{F}_{q^2}$ then $2 \mid \#\mathbf{J}_C(\mathbf{F}_q)$. On the other hand, if $\alpha \in \mathbf{F}_{q^3} \backslash \mathbf{F}_q$ then F can be factored as $F = GH$, where $G, H \in \mathbf{F}_q[X]$ with $G(\alpha) = 0$ and $\deg H = 2$, and a root of H belong to $\mathbf{F}_{q^2}$, so that $2 \mid \#\mathbf{J}_C(\mathbf{F}_q)$ also. □

Below, we choose irreducible F in the definition equation (1) of curves and set $s_i \equiv 1 \bmod 2$.

6.2 Computation of $s_i \bmod p$

We compute $s_i \bmod p$ by using the Cartier-Manin operator [11], [14], [22], [23], [31].

When the characteristic p is large, the dominant part of the Cartier-Manin operator computation is to compute

$$U = \sum u_i X^i = F^{(p-1)/2} \tag{28}$$

for F in (1). This computation itself can be efficiently carried out using FFT multiplication. It is sufficient to compute only u_{p-2}, u_{p-1}, u_{2p-2}, u_{2p-1} to determine $s_i \bmod p$.

One of the anonymous referees suggests that this computation can be done more efficiently by Euler's recurrence formula described in Chapter 4 of [8]. This method needs mod p^2 operations besides the usual mod p operations. On the other hand, its overall cost is $O(p)$ instead of $O(p \log p)$ and it requires only constant amount of memory. Thus we used this method in computation of u_{p-2}, u_{p-1}, u_{2p-2}, u_{2p-1}.

6.3 Implementation of Algorithm 2

Algorithm 2 is speeded up by using the following techniques in implementation.

1. Both the computational time and the required memory can be reduced by a factor roughly $1/\sqrt{2}$ using of the property that $-D$ can be obtained easily from a given $D \in \mathbf{J}_C(\mathbf{F}_q)$. This is done by choosing $n = \sqrt{2}n_0$, the boundaries of t_2 to be $-n \le t_2 \le n-1$, and the condition of j in Step 12 to be $B_j = \pm D_3$ and so on. See [25], [28] for further details.
2. Although Algorithm 2 is designed to minimize the cost of the worst case computation, it is more appropriate to design an algorithm minimizing the average cost for computation of prime order curves. One can minimize the average cost by choosing $n = (1/\sqrt{2})n_0$ [2], [30]. This reduces the average time by a factor roughly $2\sqrt{2}/3$ and the required memory by a factor roughly $1/\sqrt{2}$.
3. We use a 32-bit hash value of b_j in the table B and the precomputation table described in [21] to reduce the required memory.
4. Since practical speed of Algorithm 2 depends on the addition speed on $\mathbf{J}_C(\mathbf{F}_q)$, we use an improved Harley addition algorithm shown in [24].
5. The algorithm is terminated once one checked out that $\mathbf{J}_C(\mathbf{F}_q)$ has a non-prime order.

Remark 5. The average time is reduced by a factor roughly 2/3 and the required memory is reduced roughly to a half, by using the techniques of 1. and 2. simultaneously, here $n = n_0$ is used.

6.4 Implementation Results

The results shown in this section include: two examples of genus 2 hyperelliptic curves with prime orders constructed by Algorithm 3 an example of 160 bit order of a genus 2 hyperelliptic curve computed by applying the Gaudry-Harley's Schoof-like algorithm and the proposed algorithm simultaneously, and timings to compute these orders. NTL [27] is used for finite field and polynomial operations.

Algorithm 3: Construction of a prime order genus 2 hyperelliptic curve.

```
Input: A finite field $\mathbf{F}_q$ and $p = \mathrm{char}(\mathbf{F}_q)$
Output: A prime order curve $C$ and $\#\mathbf{J}_C(\mathbf{F}_q)$
1: repeat
2:     Choose a monic irreducible polynomial $F/\mathbf{F}_q, \deg F = 5$ randomly
3:     $C : Y^2 = F$
4:     Compute $s_{CMi} \equiv s_i \bmod p, 0 \le s_{CMi} < p$ by using the Cartier-Manin operator
5:     $m \leftarrow 2p$, $s'_i \leftarrow s_{CMi}$ if $2 \nmid s_{CMi}$, else $s'_i \leftarrow s_{CMi} + p$
6:     Compute $\#\mathbf{J}_C(\mathbf{F}_q)$ by Algorithm 2
7: until $\#\mathbf{J}_C(\mathbf{F}_q) =$ a prime number
8: Output $C$ and $\#\mathbf{J}_C(\mathbf{F}_q)$
```

Example 1. A 123-bit prime order Jacobian variety of the following curve

$$\begin{aligned}
&C_1/\mathbf{F}_q : Y^2 = F_1(X),\\
&F_1 = X^5 + (1154721\alpha^2 + 240985\alpha + 1084256)X^4\\
&\quad +(737339\alpha^2 + 426915\alpha + 410309)X^3\\
&\quad +(432186\alpha^2 + 1175381\alpha + 162117)X^2\\
&\quad +(1082439\alpha^2 + 231901\alpha + 16392)X\\
&\quad +670097\alpha^2 + 295934\alpha + 569191
\end{aligned}$$

is obtained by Algorithm 3, where

$$\begin{aligned}
&p = 1342181,\ \mathbf{F}_q = \mathbf{F}_p(\alpha),\\
&\alpha^3 + 808659\alpha^2 + 445314\alpha + 844247 = 0.
\end{aligned}$$

The order of $\mathbf{J}_{C_1}(\mathbf{F}_q)$ is

$$\begin{aligned}\#\mathbf{J}_{C_1}(\mathbf{F}_q) = {}& 5846103767676896833614385889373\\ & 401461.\end{aligned}$$

Table 1 Timing of computing $\#\mathbf{J}_{C_1}(\mathbf{F}_q)$ on Athlon XP 2000+.

Algorithm	Step	Time
Algorithm 3	Step 4	1m 14s
Algorithm 2	Step 4	28m 18s
	Step 5	1m 32s
	Step 6–Step 21	32m 42s
Total		1h 3m 46s

Table 2 Timing of computing $\#\mathbf{J}_{C_2}(\mathbf{F}_q)$ on Alpha 21264/667 MHz.

Algorithm	Step	Time
Algorithm 3	Step 4	3m 19s
Algorithm 2	Step 4	5h 29m 57s
	Step 5	19m 56s
	Step 6–Step 21	9h 17m 5s
Total		15h 10m 17s

The computation of $\#\mathbf{J}_{C_1}(\mathbf{F}_q)$ took about 64 minutes on Athlon XP 2000+ (1667 MHz) and less than 1GB memory. Table 1 shows the timing of main parts of Algorithm 3 and Algorithm 2 for computing $\#\mathbf{J}_{C_1}(\mathbf{F}_q)$.

Example 2. A 135-bit prime order Jacobian variety of the following curve

$$\begin{aligned}
&C_2/\mathbf{F}_q : Y^2 = F_2(X),\\
&F_2 = X^5 + (2817153\alpha^2 + 3200658\alpha + 1440424)X^4\\
&\quad +(3310325\alpha^2 + 481396\alpha + 1822351)X^3\\
&\quad +(108275\alpha^2 + 120315\alpha + 469800)X^2\\
&\quad +(2168383\alpha^2 + 1244383\alpha + 5010679)X\\
&\quad +4682337\alpha^2 + 53865\alpha + 2540378
\end{aligned}$$

is obtained by Algorithm 3, where

$$\begin{aligned}
&p = 5491813,\ \mathbf{F}_q = \mathbf{F}_p(\alpha),\\
&\alpha^3 + 4519302\alpha^2 + 3749080\alpha + 607603 = 0.
\end{aligned}$$

The order of $\mathbf{J}_{C_2}(\mathbf{F}_q)$ is

$$\begin{aligned}
&\#\mathbf{J}_{C_2}(\mathbf{F}_q)\\
&= 27434335457581234045473311611818187339271.
\end{aligned}$$

The computation of $\#\mathbf{J}_{C_2}(\mathbf{F}_q)$ took about 15 hours on Alpha 21264/667 MHz and less than 2GB memory. Table 2 shows the timing of main parts of Algorithm 3 and Algorithm 2 for computing $\#\mathbf{J}_{C_2}(\mathbf{F}_q)$.

Example 3. A 160-bit (but non-prime) order is obtained by using both the Gaudry-Harley's Schoof-like algorithm and the proposed algorithm simultaneously. For the computation of the Gaudry-Harley's Schoof-like algorithm, we used Magma V.2.8 [3] and its inner package of the Gaudry-Harley's Schoof-like algorithm written by Gaudry.

For the curve

Table 3 Timing of computing $\#\mathbf{J}_{C_3}(\mathbf{F}_q)$ on Athlon XP 2000+.

Algorithm	l / Step	Time
Schoof-like	3	13s
	5	9m 39s
	7	2h 30m 29s
	11	11d 14h 22m 3s
	13	9d 1h 10m 19s
Algorithm 3	Step 4	1m 1s
Algorithm 2	Step 4	58m 7s
	Step 5	2m 52s
	Step 6–Step 21	33m 34s
Total		20d 19h 48m 17s

$$\begin{aligned}
&C_3/\mathbf{F}_q : Y^2 = F_3(X),\\
&F_3 = X^5\\
&\quad +(917060\alpha^3 + 614005\alpha^2 + 1015600\alpha + 259417)X^3\\
&\quad +(762193\alpha^3 + 746826\alpha^2 + 86760\alpha + 91163)X^2\\
&\quad +(479517\alpha^3 + 775547\alpha^2 + 362123\alpha + 634715)X\\
&\quad +730866\alpha^3 + 778219\alpha^2 + 936773\alpha + 106583
\end{aligned}$$

the order of the Jacobian variety is obtained as

$$\begin{aligned}
\#\mathbf{J}_{C_3}(\mathbf{F}_q) =&1461445886399501592450126870336\\
&3804446894515782 97\\
=&1667 \times 3323 \times 1109653\\
&\times 1453706062552087\\
&\times 163550925035311435747
\end{aligned}$$

where

$$\begin{aligned}
&p = 2^{20} - 5,\ \mathbf{F}_q = \mathbf{F}_p(\alpha),\\
&\alpha^4+278680\alpha^3+445675\alpha^2+218811\alpha+653340=0.
\end{aligned}$$

The computation of $\#\mathbf{J}_{C_3}(\mathbf{F}_q)$ took about 21 days on Athlon XP 2000+ and less than 500 MB memory. Table 3 shows the timing of main parts to compute $\#\mathbf{J}_{C_3}(\mathbf{F}_q)$.

Remark 6. In the Schoof-like algorithm of Example 3, the computation for $l = 11$ is slower than for $l = 13$. This is because that Gaudry-Harley's Schoof-like algorithm needs to find l-torsion points defined over certain extension of $\mathbf{F}_q$, so that the cost depends on the minimal degree of the extension over which the l-torsion points exist. In Example 3, the minimal degree is 2 for $l = 13$ and 305 for $l = 11$.

Remark 7. All above examples are only to show the practical performance of the proposed algorithm. The curves C_i obtained in the examples should not be used in cryptosystems. In fact, the orders of both $\mathbf{J}_{C_1}$ and $\mathbf{J}_{C_2}$ are not large enough for cryptographic usage, the order of $\mathbf{J}_{C_3}$ does not contain a prime factor which is large enough for cryptographic usage.

The definition fields of all C_i are small extensions of prime fields therefore they may not secure against certain potential Weil descent attacks.

7. Conclusion and Outlook

This paper proposed an improvement of the baby step giant step algorithm for point counting of hyperelliptic curves over finite fields. In construction of secure hyperelliptic curves of genus 2, with knowledge of the residues modulo m of the characteristic polynomials of the Frobenius endomorphisms, the new algorithm can speed up searching by a factor m, instead of $\sqrt{m}$ in original square-root algorithms. Moreover the algorithm is implemented to find a 135-bit prime order curve.

Computation of an order of a hyperelliptic curve in the size of cryptographic usage is possible if both the algorithm proposed in this paper and the Gaudry-Harley's Schoof-like algorithm are used simultaneously. However, computation of the residues modulo a prime l by the Schoof-like algorithm becomes impractical for large l. Thus, it seems difficult at present to use these algorithms to find a curve which is cryptographically interesting.

On the other hand, if one could somehow compute the residues modulo l of s_i for l up to 31, then combining with the proposed algorithm, it will be possible to efficiently compute 160-bit orders for curves over prime fields. In fact, the memory required by the proposed algorithm will be less than 150 MB.

Acknowledgement

The authors would like to thank Fumiyuki Momose for Lemma 1. We would also like to thank Seigo Arita and Koh-ichi Nagao for their interesting discussions, the anonymous referees for their comments to improve this paper and on Euler's recurrence formula, and Andreas Stein for his helpful comments on an early draft of this paper.

A part of this research was supported by Telecommunications Advancement Organization of Japan (TAO).

References

[1] L.M. Adleman and M.D. Huang, "Counting rational points on curves and Abelian varieties over finite fields," ANTS-II, in Lecture Notes in Computer Science, ed. H. Cohen, no.1122, pp.1–16, Springer-Verlag, 1996.
[2] S.R. Blackburn and E. Teske, "Baby–step giant–step algorithms for non-uniform distributions," ANTS-IV, in Lecture Notes in Computer Science, ed. W. Bosma, no.1838, pp.153–168, Springer-Verlag, 2000.
[3] W. Bosma and J. Cannon, Handbook of Magma functions, University of Sydney. http://www.maths.usyd.edu.au:8000/u/magma/, 2001.
[4] J.W.S. Cassels and E.V. Flynn, Prolegomena to middlebrow arithmetic of curves of genus 2, London Mathematical Society, Lecture Note Series, no.230, Cambridge University Press, 1996.
[5] H. Cohen, A Course in Computational Algebraic Number Theory, Graduate Text in Mathematics, no.138, Springer-Verlag, 1993.
[6] J. Denef and F. Vercauteren, "An extension of Kedlaya's algorithm to Artin-Schreier curves in characteristic 2," ANTS-V, in Lecture Notes in Computer Science, ed. C. Fieker and D. Kohel, no.2369, pp.308–323, Springer-Verlag, 2002.
[7] N.D. Elkies, "Elliptic and modular curves over finite fields and related computational issues," in Computational perspectives on number theory, ed. D.A. Buell and J.T. Teitlbaum, pp.21–76, AMS, 1995.
[8] L. Eulero, Introductio in analysin infinitorum, Lausannæ: Marcum–Michaelem Bousquet, 1748.
[9] G. Frey and H.G. Rück, "A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves," Math. Comput., vol.62, pp.865–874, 1994.
[10] S.D. Galbraith, "Weil descent of Jacobians," Electronic Notes in Discrete Mathematics, ed. D. Augot and C. Carlet, Elsevier Science Publishers, 2001.
[11] P. Gaudry, Algorithmique des courbes hyperelliptiques et applications à la cryptologie, Ph.D. Thesis, École Polytechnique, 2000.
[12] P. Gaudry, "Algorithms for counting points on curves," Talk at ECC 2001, The Fifth Workshop on Elliptic Curve Cryptography, University of Waterloo, 2001.
[13] P. Gaudry and N. Gürel, "An extension of Kedlaya's point-counting algorithm to superelliptic curves," Advances in Cryptology—ASIACRYPT2001, in Lecture Notes in Computer Science, ed. C. Boyd, no.2248, pp.480–494, Springer-Verlag, 2001.
[14] P. Gaudry and R. Harley, "Counting points on hyperelliptic curves over finite fields," ANTS-IV, in Lecture Notes in Computer Science, ed. W. Bosma, no.1838, pp.297–312, Springer-Verlag, 2000.
[15] R. Harley, "Counting points with the arithmetic-geometric mean," Rump talk at EUROCRYPT 2001, 2001. (joint work with J.-F. Mestre and P. Gaudry)
[16] M.D. Huang and D. Ierardi, "Counting rational point on curves over finite fields," J. Symbolic Computation, vol.25, pp.1–21, 1998.
[17] W. Kampkötter, Explizite Gleichungen für Jacobische Varietäten hyperelliptischer Kurven, Ph.D. Thesis, GH Essen, 1991.
[18] N. Kanayama, K. Nagao, and S. Uchiyama, "Generating hyperelliptic curves of genus 2 suitable for cryptography," Proc. 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002), 2002.
[19] K.S. Kedlaya, "Counting points on hyperelliptic curves using Monsky-Washinitzer cohomology," J. Ramanujan Math. Soc., vol.16, no.4, pp.323–338, 2001.
[20] A. Lauder and D. Wan, "Computing zeta functions of Artin-Schreier curves over finite fields," LMS J. Comput. Math., vol.5, pp.33–55, 2002.
[21] F. Lehmann, M. Maurer, V. Müller, and V. Shoup, "Counting the number of points on elliptic curves over finite fields of characteristic greater than three," ANTS-I, in Lecture Notes in Computer Science, ed. L. Adleman and M.D. Huang, no.877, pp.60–70, Springer-Verlag, 1994.
[22] J.I. Manin, "The theory of commutative formal groups over fields of finite characteristic," Russian Mathematical Surveys, vol.18, pp.1–83, 1963.
[23] J.I. Manin, "The Hasse-Witt matrix of an algebraic curve," Trans. AMS, vol.45, pp.245–264, 1965.
[24] K. Matsuo, J. Chao, and S. Tsujii, "Fast genus two hyperelliptic curve cryptosystems," IEICE Technical Report, ISEC2001-31, 2001.
[25] A. Menezes, S. Vanstone, and R. Zuccherato, "Counting

points on elliptic curves over $\mathbf{F}_{2^m}$," Math. Comput., vol.60, pp.407–420, 1993.

[26] J. Pila, "Frobenius maps of Abelian varieties and finding roots of unity in finite fields," Math. Comput., vol.55, pp.745–763, 1990.

[27] V. Shoup, `http://www.shoup.net/ntl/`, 2001.

[28] A. Stein and E. Teske, "Optimized baby step-giant step methods and applications to hyperelliptic function fields," Tech. Rep. CORR 2001-62, Department of Combinatorics and Optimization, University of Waterloo, 2001.

[29] H. Stichtenoth, Algebraic function fields and codes, Universitext, Springer-Verlag, 1993.

[30] E. Teske, "Square-root algorithms for the discrete logarithm problem (A survey)," in Public-Key Cryptography and Computational Number Theory, pp.283–301, Walter de Gruyter, Berlin, New York, 2001.

[31] N. Yui, "On the Jacobian varieties of hyperelliptic curves over fields of characteristic $p > 2$," J. Algebra, vol.52, pp.378–410, 1978.

Kazuto Matsuo received the B.E., M.E., and D.E. from Chuo University, Tokyo, Japan in 1986, 1988, and 2001, respectively. He joined Toyo Communication Equipment Co., LTD from 1988 to 2001. He has been an associate professor in the Research and Development Initiative at Chuo University since 2002. His current research interests include cryptography.

Jinhui Chao received the B.E. from the Department of Electronics Engineering, Xidian University, China in 1982, and the M.E. and D.E. from Tokyo Institute of Technology, Tokyo, Japan in 1985 and 1988, respectively both in electrical engineering. He was an assistant professor of Tokyo Institute of Technology in 1989. From 1992 he became an associate professor and from 1996 a professor in the Department of Electrical and Electronic Engineering at Chuo University, Tokyo, Japan. His current research interests include cryptography theory, artificial neural networks, nonlinear adaptive signal processing, 3D images, and color science. He received the Excellent Paper Awards from IEICE in 1988 and 1990, and the Shinoda Academic Award from IEICE in 1991.

Shigeo Tsujii was born in Kyoto, Japan on September 13, 1933. He received the B.E. and the D.E. in electrical engineering from Tokyo Institute of Technology, Tokyo, Japan in 1958 and 1970, respectively. From 1958 to 1965 he worked for Nippon Electric Company. Between 1965 and 1971 he was an associate professor at Yamanashi University, Kofu, Japan and since 1971, after seven years as an associate professor, he was a professor at Tokyo Institute of Technology, Tokyo, Japan until 1994. He is now working as a professor in the Department of Information Systems, Chuo University, Tokyo, Japan. His research interests include communication networks, and information security and cryptology. He served as a president of IEICE Japan from 1996 to 1997 and of the Radio Regulatory Council, Ministry of Public Management, Home Affairs, Posts and Telecommunications Japan from 2001 to 2002. He is now acting as chair person of Japan Council of IEEE.