Index Calculus Attack for Jacobian of Hyperelliptic Curves of Small Genus Using Two Large Primes

Koh-ichi Nagao*

* *Dept. of Engineering, Kanto-Gakuin Univ.*
E-mail: nagao@kanto-gakuin.ac.jp

Received October 31, 2006
Revised March 27, 2007

This paper introduces a fast algorithm for solving the DLP of Jacobian of hyperelliptic curve of small genus. To solve the DLP, Gaudry first shows that the idea of index calculus is effective, if a subset of the points of the hyperelliptic curve of the base field is taken by the smooth elements of index calculus. In an index calculus theory, a special element (in our case it is the point of hyperelliptic curve), which is not a smooth element, is called a large prime. A divisor, written by the sum of several smooth elements and one large prime, is called an almost smooth divisor. By the use of the almost smooth divisor, Thériault improved this index calculus. In this paper, a divisor, written by the sum of several smooth elements and two large primes, is called a 2-almost smooth divisor. By use of the 2-almost smooth divisor, we are able to give more improvements. The algorithm of this attack consists of the following seven parts: 1) Preparing, 2) Collecting reduced divisors, 3) Making sufficiently large sets of almost smooth divisors, 4) Making sufficiently large sets of smooth divisors, 5) Solving the linear algebra, 6) Finding a relation of collected reduced divisors, and 7) Computing a discreet logarithm. Parts 3) and 4) need complicated eliminations of the large prime, which is the key idea presented within this paper. Before the tasks in these parts are completed, two sub-algorithms for the eliminations of the large prime have been prepared. To explain how this process works, we prove the probability that this algorithm does not work to be negligible, and we present the expected complexity and the expected storage of the attack.

Key words: index calculus attack, Jacobian, hyperelliptic curve, DLP

1. Introduction

DLP of the Jacobian group of hyperelliptic curve C over finite field $\mathbb{F}_q$, is a problem finding integer n such that $D_1 = nD_2$, where D_1, D_2 are given reduced divisors of the curve. When the base curve is a hyperelliptic curve, the additions of the Jacobian group are easily computable and it has fruitful application to public key cryptology. Gaudry [4] first present a variation of index calculus attack for the DLP of the Jacobian of hyperelliptic curve of small genus. In this attack, $B_0 = \{(P - \infty) \mid P \in C(\mathbb{F}_q)\}$ is taking as smooth elements of index calculus. Gaudry and Harley [3] improved this result by the restriction of the smooth elements (B is taken as some subset of B_0) and by attaining the rebalance of the computation of the group laws and linear algebra. An element in $B_0 \setminus B$ is called large prime and a divisor, written by the sum of several smooth elemments and one large prime, is called alomst smooth divisor. Thériault improved this attack by the use of alomst smooth divisor. These algorithms work in time $O\big(q^{2-\frac{2}{g+1}+\epsilon}\big)$ and $O\big(q^{2-\frac{4}{2g+1}+\epsilon}\big)$ respectively, where g is the genus of the curve C. When $g \geq 3$, the complexity of Thériault's method is less than the complexity of the attack using square root methods such as Pollard's rho method or baby step giant step method.

Similarly, a divisor, written by the sum of several smooth elemments and two large primes, is called 2-alomst smooth divisor. In this paper, we propose further improved index calculus attack against the DLP of the Jacobian by the use of 2-alomst smooth divisor.

Note that the same kind of the attack is done by Gaudry, E. Thomé, N. Thériault, C. Diem [5] independently. (The method of Gaudry et al. interprets the elimination of large prime by the connection of the graph whose vertexes are large primes.)

First, we will explain the outline of the elimination of the large prime, which is a key of the algorithm. Note that an almost smooth divisor is written by the form $\sum$ terms of $B+(P-\infty)$, and a 2-almost smooth divisor is written by the form $\sum$ terms of $B+(Q-\infty)+(R-\infty)$, where $P, Q, R \in B_0 \backslash B$. Let $v_1 = \sum$ terms of $B+(P_1-\infty)$ and $v_2 = \sum$ terms of $B+(P_1-\infty)$ be the almost smooth divisors, which have the same large prime. Then, a new smooth divisor $v_1 - v_2 = \sum$ terms of B is obtained by the elimination of large prime P_1 and by using such new smooth divisors, Thériault realized a faster attack. Let $v = \sum$ terms of $B + (P_0 - \infty)$ be an almost smooth divisor and $v_1 = \sum$ terms of $B + (P_0 - \infty) + (P_1 - \infty), v_2 = \sum$ terms of $B+(P_1-\infty)+(P_2-\infty), \ldots, v_n = \sum$ terms of $B+(P_{n-1}-\infty)+(P_n-\infty)$, be the two-almost smooth divisors written by these forms. Then, new almost smooth divisors $v - v_1 + v_2 - v_3 + \cdots + (-1)^i v_i = \sum$ terms of $B + (-1)^i(P_i - \infty)$, $(0 \leq i \leq n)$ are obtained by the elimination of large prime. In this paper, we propose a faster attack by using such new almost smooth divisors and show the following theorem.

THEOREM 1. *Let C be a hyperelliptic curve of genus ≥ 3 over finite field $\mathbb{F}_q$. Then the DLP of $\mathbf{Jac}_C(\mathbb{F}_q)$ can be solved in expected time $O(q^{2-2/g+\varepsilon})$.*

Further, Diem [2] presents an index calculus to the Jacobian of non-hyperelliptic curve. The expected complexity of genus 3 non-hyperelliptic curve is $O(q^{1+\epsilon})$ which is smaller than that of genus 3 hyperelliptic curve. Further, a variant of the index calculus of using two large primes is applied to the attack of XTR [6].

Further in this paper, we will use the symbols $\doteq$ and $\gg$ by the notations $a \doteq b \Leftrightarrow a/b = 1 + o(1)$ and $a \gg b \Leftrightarrow (a - b)/a = 1 + o(1)$.

2. Jacobian arithmetic

In this section, we will prepare the definitions and the lemmas of the Jacobian arithmetic. Let C be a hyperelliptic curve of genus g over $\mathbb{F}_q$ of the form $y^2+h(x)y = f(x)$ with $\deg f = 2g+1$ and $\deg h \leq g$. Further, use the notation J_q for $\mathbf{Jac}_C(\mathbb{F}_q)$. Moreover, we will assume that $|J_q|$ is odd prime number, for simplicity.

DEFINITION 1. *Given $D_1, D_2 \in J_q$ such that $D_2 \in \langle D_1 \rangle$, DLP for (D_1, D_2) on J_q is computing λ such that $D_2 = \lambda D_1$.*

For an element $P = (x, y)$ in $C(\bar{\mathbb{F}}_q)$, put $-P := (x, -h(x) - y)$.

LEMMA 1. *$C(\mathbb{F}_q)$ is written by the union of disjoint sets $\mathcal{P} \cup -\mathcal{P} \cup \{\infty\}$, where $-\mathcal{P} := \{-P \mid P \in \mathcal{P}\}$.*

Proof. Since $|J_q|$ is odd prime, we have $2 \nmid |J_q|$ and there are no point $P \in C(\mathbb{F}_q)$ such that $P = -P$. $\square$

Further, we will fix $\mathcal{P}$.

DEFINITION 2.

1) *A subset B of $\mathcal{P}$ is called factor base.*
2) *A point $P \in \mathcal{P} \setminus B$ is called large prime.*

Note that the factor base B is used to define the smoothness of index calculus. Point of $\mathbf{Jac}_C$ can be represented uniquely by the reduced divisor of the form

$$\sum_{i=1}^{k} n_i P_i - \sum_{i=1}^{k} n_i \infty, \quad P_i \in C(\bar{\mathbb{F}}_q), \quad P_i \neq -P_j \quad \text{for} \quad i \neq j$$

with $n_i \geq 0$ and $\sum n_i \leq g$. Thus in this paper, a point of Jacobian will be called by using the expression "reduced divisor". Let $D(P) := P - \infty$. Note that $P + (-P) \sim 2\infty$. From Lemma 1, a reduced divisor v of J_q can be represented by the form

$$v = \sum_{P \in C(\bar{\mathbb{F}}_p)} n_P^{(v)} D(P)$$

with $n_P^{(v)} \in \mathbb{Z}$ and $\sum_{P \in C(\bar{\mathbb{F}}_q)} \left| n_P^{(v)} \right| \leq g$.

DEFINITION 3. *Let v be a reduced divisor of Jacobian J_q.*

1) *If v is written by the elements of $C(\mathbb{F}_q)$ i.e.*

$$v = \sum_{P \in C(\mathbb{F}_q)} n_P^{(v)} D(P),$$

it is called potentially smooth reduced divisor.

2) *If v is written by the elements of factor base B i.e.*

$$v = \sum_{P \in B} n_P^{(v)} D(P),$$

it is called smooth reduced divisor.

3) *If v is written by the elements of factor base B except one large prime $P' \in \mathcal{P} \setminus B$ i.e.*

$$v = n_{P'}^{(v)} D(P') + \sum_{P \in B} n_P^{(v)} D(P),$$

it is called almost smooth reduced divisor.

4) *If v is written by the elements of factor base B except two large primes $P', P'' \in \mathcal{P} \setminus B$ i.e.*

$$v = n_{P'}^{(v)} D(P') + n_{P''}^{(v)} D(P'') + \sum_{P \in B} n_P^{(v)} D(P),$$

it is called 2-almost smooth reduced divisor.

In this paper, we treat linear sums of reduced divisors whose coefficients are considerd modulo $|Jq|$. So, we define the notation of smoothness to the general divisor of the form $\sum_{P \in C(\bar{\mathbb{F}}_q)} n_P D(P)$ where n_P's are integers modulo $|J_q|$.

DEFINITION 4.

1) *A divisor v of the form*

$$\sum_{P \in B} n_P^{(v)} D(P)$$

is called smooth divisor.

2) *A divisor v of the form*

$$n_{P'}^{(v)} D(P') + \sum_{P \in B} n_P^{(v)} D(P),$$

where P' is a large prime, is called almost smooth divisor.

3) *A divisor v of the form*

$$n_{P'}^{(v)} D(P') + n_{P''}^{(v)} D(P'') + \sum_{P \in B} n_P^{(v)} D(P),$$

where P', P'' are large primes, is called 2-almost smooth divisor.

For a smooth (resp. almost smooth, resp. 2-almost smooth) divisor v, put

$$l(v) := \#\{P \in B \mid n_P^{(v)} \neq 0\}.$$

LEMMA 2. *Let v_1, v_2 be smooth (resp. almost smooth, resp. 2-almost smooth) divisors and let r_1, r_2 be integers modulo $|J_q|$. Then the cost for computing $r_1 v_1 + r_2 v_2$ is $O(g^2 (\log q)^2 (l(v_1) + l(v_2))$.*

Proof. It requires $l(v_1) + l(v_2)$-time products and additions modulo $|J_q|$. Note that $|J_q| \doteq q^g$. Since the cost of one elementary operation modulo $|J_q|$ is $O((\log |J_q|)^2) = O(g^2 (\log q)^2)$, we have this estimation. □

3. Outline of algorithm

In this section, we present the outline of the proposed algorithm. Let k be a real number satisfying $0 < k < \frac{1}{2g}$. Note that in §12, we will take $k = \frac{1}{\log q}$ and optimize the algorithm. Further in this paper, we will use k as a parameter of this algorithm. Put

$$r := r(k) = \frac{g-1+k}{g}.$$

We will fix a set of factor base B with $|B| = q^r$.

The main algorithm shown in Algorithm 1 consists of the following 7 parts 1) Preparing, 2) Collecting reduced divisors, 3) Making a sufficiently large set of almost smooth divisors, 4) Making a sufficiently large set of smooth divisors, 5) Solving the linear algebra, 6) Finding a relation of collected reduced divisors, and 7) Computing the discreet logarithm. Note that the number of collected 2-almost smooth reduced divisors in Part 2 is bigger than q^{1+k}, which is the meaning of the parameter k.

Algorithm 1. Main algorithm

Input: $C/\mathbb{F}_q$ hyper elliptic curve of small genus g, $D_1, D_2 \in J_q$ such that $D_2 \in \langle D_1 \rangle$.
Output: Integer λ modulo $|J_q|$ such that $D_2 = \lambda D_1$.

1: **Part 1** Computing all points of $C(\mathbb{F}_q)$ and making $\mathcal{P}$ and fix $B \subset \mathcal{P}$ with $|B| = q^r$.
2: **Part 2** *Collecting 2-almost smooth divisors and almost smooth divisors*
Computing a set V_2 of 2-almost smooth reduced divisors and a set V_1 of almost smooth reduced divisors of J_q, of the form $\alpha D_1 + \beta D_2$ with $|V_1| > q^{\frac{(g-1)+(g+1)k}{g}}$ and $|V_2| > q^{1+k}$.
3: **Part 3** Computing a set of almost smooth divisor H_m with $|H_m| > q^{(1+r)/2}$.
4: **Part 4** Computing a set of smooth divisor H with $|H| > q^r$.
5: **Part 5** *Solving linear algebra of the size $q^r \times q^r$*
Computing integers $\{\gamma_h\}_{h\in H}$ modulo $|J_q|$, satisfying $\sum_{h\in H} \gamma_h h \equiv 0 \bmod |J_q|$.
6: **Part 6** Computing integers $\{s_v\}_{v\in V_1\cup V_2}$ modulo $|J_q|$, satisfying $\sum_{v\in V_1\cup V_2} s_v v = 0$.
7: **Part 7** Computing λ.

4. Collecting 2-almost smooth reduced divisors and almost smooth reduced divisors

In order to collect enough 2-almost smooth divisors and almost smooth divisors (Part 2 of the main algorithm), the following Algorithm 2 can be used.

Further, we will estimate the cost of this algorithm.

Lemma 3. *The probability that a reduced divisor in J_q is almost smooth is*

$$\frac{1}{(g-1)!}q^{(-1+r)(g-1)}$$

and the probability that a reduced divisor is 2-almost smooth is

$$\frac{1}{2(g-2)!}q^{(-1+r)(g-2)}.$$

Algorithm 2. Collecting the 2-almost smooth and almost smooth reduced divisors

Input: $C/\mathbb{F}_q$ curve of genus g, $D_1, D_2 \in \mathbf{Jac}_C(\mathbb{F}_q)$
Output: V_1 a set of almost smooth reduced divisors, V_2 a set of 2-almost smooth reduced divisors such that $|V_2| > q^{1+k}$, $|V_1| > q^{\frac{(g-1)+(g+1)k}{g}}$, Integers $\{(\alpha_v, \beta_v)\}_{v \in V_1 \cup V_2}$ such that $v = \alpha_v D_1 + \beta_v D_2$

$V_1 \leftarrow \{\}$, $V_2 \leftarrow \{\}$
repeat
  Let α, β be random numbers modulo $|J_q|$
  Compute $v = \alpha D_1 + \beta D_2$
  if v is almost smooth **then**
    $V_1 \leftarrow V_1 \cup \{v\}$
    $(\alpha_v, \beta_v) \leftarrow (\alpha, \beta)$
  end if
  if v is 2-almost smooth **then**
    $V_2 \leftarrow V_2 \cup \{v\}$
    $(\alpha_v, \beta_v) \leftarrow (\alpha, \beta)$
  end if
until $|V_2| > q^{1+k}$ and $|V_1| > q^{\frac{(g-1)+(g+1)k}{g}}$
return $V_1, V_2, \{(\alpha_v, \beta_v)\}_{v \in V_1 \cup V_2}$

Proof. The first formula is from Propositions 3, 4, 5 in [8]. By the use of the similar argument, the probability of a reduced divisor being 2-almost smooth is roughly estimated by

$$\frac{(2|B|)^{g-2}(2|\mathcal{P} \setminus B|)^2}{2!\,(g-2)!\,|J_q|} \doteq \frac{(q^r)^{g-2}q^2}{2!\,(g-2)!\,q^g} = \frac{1}{2(g-2)!}q^{(-1+r)(g-2)},$$

and the second formula is obtained. $\square$

From this lemma, the number of the loops that $|V_2| > q^{1+k}$ is estimated by

$$q^{(1+k)} \cdot 2(g-2)!\,q^{(1-r)(g-2)} = 2(g-2)!\,q^{2r},$$

and the number of the loops that $|V_1| > q^{\frac{(g-1)+(g+1)k}{g}}$ is estimated by

$$q^{\frac{(g-1)+(g+1)k}{g}} \cdot (g-1)!\,q^{(1-r)(g-1)} = (g-1)!\,q^{2r}.$$

Since the cost of computing Jacobian $v = \alpha D_1 + \beta D_2$ is $O(g^2(\log q)^2)$ and the cost of judging whether v is potentially smooth or not is $O(g^2(\log q)^3)$, the total cost of this part is estimated by

$$O(g^2(g-1)!\,(\log q)^3 q^{2r}).$$

Here, we will estimate the required storage. Note that the bit-length of one potentially smooth reduced divisor is $2g \log q$. So, the storage for V_1 is

$O\Big(gq^{\frac{(g-1)+(g+1)k}{g}} \log q\Big)$ and the storage for V_2 is $O(gq^{(1+k)} \log q)$. Since $1+k > \frac{(g-1)+(g+1)k}{g}$, we have $gq^{(1+k)} \log q \gg gq^{\frac{(g-1)+(g+1)k}{g}} \log q$. So the total required storage can be estimated by

$$O(gq^{(1+k)} \log q).$$

5. Elimination of large prime

In this section, we give sub-algorithms of the elimination of large prime, which are needed Part 3 and Part 4 of Main Algorithm. Let E be a set of almost smooth divisors. Also, let F be 1) a set of 2-almost smooth divisors or 2) a set of almost smooth divisors. Note that elements $e \in E$ and $f \in F$ are written by

$$\begin{aligned} e &= n_{P_1}^{(e)} D(P_1) + \sum_{P \in B} n_P^{(e)} D(P), \\ f &= n_{P_2}^{(f)} D(P_2) + \sum_{P \in B} n_P^{(f)} D(P), \quad \text{if } F \text{ is a set of almost smooth divisors,} \\ f &= n_{P_2}^{(f)} D(P_2) + n_{P_3}^{(f)} D(P_3) + \sum_{P \in B} n_P^{(f)} D(P), \\ &\qquad\qquad\qquad\qquad\qquad\qquad \text{if } F \text{ is a set of 2-almost smooth divisors.} \end{aligned}$$

Put $\sup(e) := \{P_1\}$ and

$$\sup(f) := \begin{cases} \{P_2\} & \text{if } F \text{ is a set of almost smooth divisors,} \\ \{P_2, P_3\} & \text{if } F \text{ is a set of 2-almost smooth divisors.} \end{cases}$$

When $P \in \sup(e) \cap \sup(f)$, also put

$$\phi(e, f, P) := n_p^{(f)} e - n_p^{(e)} f.$$

Note that $\phi(e, f, P)$ is a new divisor obtained by once large prime elimination. So, if F is a set of 2-almost smooth divisors, $\phi(e, f, P)$ is an almost smooth divisor. If F is a set of almost smooth divisors and e is not of the form constant times f, $\phi(e, f, P)$ is a smooth divisor.

First, we treat the case that E being a set of almost smooth divisors and F being a set of 2-almost smooth divisors. By using Algorithm 3, we construct another set of almost smooth divisors, named E', by once elimination of large prime.

Here, we explain the meanings of E' and F' in Algorithm 3. A set of almost smooth divisors $\bigcup \phi(e, f, P)$, where e, f, and P moves $e \in E$, $f \in F$, and $P \in \sup(e) \cap \sup(f)$, is made by once large prime elimination from E and F. The set of almost smooth divisors E' made by Algorithm 3 is a subset of $\bigcup \phi(e, f, P)$ and has the following properties: if $\phi(e_1, f, P_1)$ and $\phi(e_2, f, P_2)$ are distinct elements of E', e_1, e_2 are distinct. This property will be needed in Lemma 8. The set of 2-almost smooth divisors F' made by Algorithm 3 is a subset of F, consist of the 2-almost smooth divisors that dose not used to the eliminations.

Algorithm 3. Elimination of large primes

Input: E almost smooth divisors, F 2-almost smooth divisors
Output: E' almost smooth divisors, F' 2-almostsmooth divisors

set $\mathcal{P} \setminus B = \{R_1, R_2, \ldots, R_{|\mathcal{P}\setminus B|}\}$ (pre-computation)
for $i = 1, 2, \ldots, |\mathcal{P} \setminus B|$ **do**
  $st[i] \leftarrow \{\}$
od
for all $e \in E$ **do**
  $P = \sup(e)$
  Compute i s.t. $P = R_i$
  $st[i] \leftarrow st[i] \cup \{e\}$
od
$E' \leftarrow \{\}$, $F' \longleftarrow F$
for all $f \in F$ **do**
  $P_1, P_2 := \sup(f)$
  Compute i s.t. $P_1 = R_i$
  if $st[i] \neq \emptyset$ **then**
    Take some $e \in st[i]$
    $E' \leftarrow E' \cup \{\phi(e, f, P)\}$, $F' \leftarrow F' \setminus \{f\}$
    break
    break (return to the loop of next $f \in F$)
  end if
  Compute i s.t. $P_2 = R_i$
  if $st[i] \neq \emptyset$ **then**
    Take some $e \in st[i]$
    $E' \leftarrow E' \cup \{\phi(e, f, P)\}$, $F' \leftarrow F' \setminus \{f\}$
    break
    break (return to the loop of next $f \in F$)
  end if
od
return E', F'

Definition 5. *Further, put*

$$E \cdot F := E', \quad E \odot F := F'.$$

We will estimate the size of $E \cdot F$ and $E \odot F$.

Lemma 4. *Let E be a set of randomly chosen almost smooth divisors and F be a set of randomly chosen 2-almost smooth divisors. Assume $|E| \ll q < |F|$. The size of $E \cdot F$ is estimated by*

$$|E \cdot F| \doteq \frac{2|E|\,|F|}{|\mathcal{P} \setminus B|} \doteq \frac{4|E|\,|F|}{q}.$$

Further, $|E \odot F| = |F| - |E \cdot F|$.

Proof. Let $e \in E$, $f \in F$ be randomly chosen elements. Put $P := \sup(e)$. Since F is a set of 2-almost smooth divisors, the probability that $P \in \sup(f)$ is $\frac{2}{|\mathcal{P}\setminus B|} \doteq \frac{4}{q}$ and the size is estimated by $\frac{2}{|\mathcal{P}\setminus B|}|E|\,|F| = \frac{4}{q} \times |E|\,|F|$. Second formula is trivial. □

We will estimate the cost and the storage for computing $E \cdot F$ and $E \odot F$ by Algorithm 3.

Lemma 5. *Put* $c_1 := \max\{l(e) \mid e \in E\}$ *and* $c_2 := \max\{l(f) \mid f \in F\}$. *Assume that* $|E| \ll q$. *Then the cost of computing* $E \cdot F$ *and* $E \odot F$ *is*

$$O(c_1(\log q)^2|E|) + O((\log q)^2|F|) + O((c_1 + c_2)(g\log q)^2|E|\,|F|/q)$$

and the required storage is

$$O(c_1 \log q|E|) + O((c_1 + c_2)\log q|E|\,|F|/q).$$

Proof. The required storage for $st[i]$ is $O(c_1 \log q|E|)$ and the required storage for E' is $O((c_1+c_2)\log q|E|\,|F|/q)$, since $|E'| \doteq |E|\,|F|/q$ and $\max\{l(v) \mid v \in E'\} = c_1 + c_2$. Note that the cost of the routine "Computing index i" is $\log q \log|\mathcal{P}\setminus B| = O((\log q)^2)$. Also note that $|E \cdot F| = O(|E|\,|F|/q)$ and remark that the probability of $st[i] \neq \emptyset$ is very small, since $|E| \ll q$. Thus, we see that the cost of the 1st loop is $O(c_1(\log q)^2|E|)$, the cost of the part "Computing index i" of the 2nd loop is $O((\log q)^2|F|)$, and the cost of the part "Computing the elements of E' and F'" of the 2nd loop is $O((c_1 + c_2)(g\log q)^2|E|\,|F|/q)$ from Lemma 2. □

Now, let E be a set of almost smooth divisors. A set of smooth divisors E' is constructed from E by Algorithm 4.

Similarly, the set of smooth divisors $\bigcup \phi(e_1, e_2, P)$, where e, f and P moves $e_1, e_2 \in E$, $e_1 \neq \text{Const} \times e_2$, and $P = \mathbf{sup}(e_1) \cap \mathbf{sup}(e_2)$ is made by once large prime elimination from E. The set of smooth divisors E' made by Algorithm 4 is a subset of $\bigcup \phi(e, f, P)$ and has the following property: if $\phi(e, e_1, P_1)$, $\phi(e, e_2, P_2)$, $\phi(e_3, e, P_3)$ and $\phi(e_4, e, P_4)$ are distinct elements of E', then e_1, e_2, e_3 and e_4 are distinct. Note that if $e_1, e_2 \in E$ are used once, e_1, e_2 are never used to the construction of E'. This property will be needed in Lemma 8.

Definition 6. *Also put*

$$E \cdot E := E'.$$

We will estimate the size of $E \cdot E$ and the cost of this computation.

Lemma 6. *Let E be a set of randomly chosen almost smooth divisors. Assume* $|E| \ll q$. *The size of* $E \cdot E$ *is estimated by*

$$|E \cdot E| \doteq \frac{|E|^2}{2|\mathcal{P}\setminus B|} \doteq \frac{|E|^2}{q}.$$

Algorithm 4. Elimination of large primes

Input: E almost smooth divisors
Output: E' smooth divisors

```
set P \ B = {R_1, R_2, ..., R_{|P\B|}} (pre-computation)
for i = 1, 2, ..., |P \ B| do
    st[i] ← {}
od
for all e ∈ E do
    P = sup(e)
    Compute i s.t. P = R_i
    st[i] ← st[i] ∪ {e}
od
E' ← {}
for all f ∈ E do
    P := sup(f)
    Compute i s.t. P = R_i
    if st[i] ≠ ∅ then
        for all e ∈ st[i] s.t. e ≠ Const × f do
            E' ← E' ∪ {φ(e, f, P)}, st[i] ← st[i] \ {e, f}
            break (return to the loop of next f ∈ E)
        od
    end if
    od
od
return E'
```

Further, put $c_1 := \max\{l(e) \mid e \in E\}$, *then the cost of computing* $E \cdot E$ *is*

$$O(c_1(\log q)^2|E|) + O(c_1(g \log q)^2|E|^2/q),$$

and the required storage is

$$O(c_1 \log q|E|) + O(c_1 \log q|E|^2/q).$$

Proof. Let $e_1, e_2 \in E$ be randomly chosen elements. Put $P := \sup(e_1)$. The probability that $P \in \sup(e_2)$ is $\frac{1}{|\mathcal{P}\setminus B|} \doteq \frac{2}{q}$ and the size is estimated by $\binom{|E|}{2} \times \text{prob.} = \frac{1}{2|\mathcal{P}\setminus B|}|E|^2 = \frac{1}{q} \times |E|^2$. Cost estimations are similarly done by the previous case. □

6. Computing a large enough set of almost smooth divisors

In this section, we construct a set of almost smooth divisors H_m such that $|H_m| > q^{(1+r)/2}$ using the following Algorithm 5.

Note that the set of almost smooth divisors H_i is obtained by $(i-1)$-th large prime eliminations from V_1 and V_2 and that 2-almost smooth divisors in $V_{2,i}$ are

Algorithm 5. Computing H_m

Input: V_1 a set of almost smooth divisors s.t. $|V_1| > q^{\frac{(g-1)+(g+1)k}{g}}$, V_2 a set of 2-almost smooth divisors s.t. $|V_2| > q^{(1+k)}$
Output: Integer $m > 0$ and $H_1, H_2, \ldots, H_m$ sets of almost smooth divisors s.t. $|H_m| > q^{(1+r)/2}$

$H_1 \leftarrow V_1$, $V_{2,1} \leftarrow V_2$
$i \leftarrow 1$
repeat
  $i{+}{+}$
  $H_i \leftarrow H_{i-1} \cdot V_{2,i-1}$, $V_{2,i} \leftarrow H_{i-1} \odot V_{2,i-1}$,
until $|H_i| > q^{(1+r)/2}$
$m \leftarrow i$
return $m, H_1, H_2, \ldots, H_m$

---

not used to the construction of $H_2, \ldots, H_i$. Now, we estimate the size of m. In order to estimate the sizes $|H_i|$ and $|V_{2,i}|$, we use the size estimation of Lemma 4 as a heuristics. From Lemma 4, the size of H_i is estimated by

$$|H_i| \doteq |H_1| \times (q^k)^{i-1} = q^{\frac{(g-1)+(gi+1)k}{g}}.$$

So, solving the equation $\frac{(g-1)+(gi+1)k}{g} = (1 + r(k))/2$ for i, we have the following.

Lemma 7. *m is estimated by*

$$\frac{1-k}{2gk}.$$

Then, we can assume $m = O\big(\frac{1}{gk}\big)$, which is needed for the cost estimation in §12. Note that $\big\{l(v) \mid v \in \bigcup_{i \leq m} H_i\big\} \leq mg$. From Lemma 5, the cost for computing H_m is

$$m \times (O((\log q)^2 q^{(1+k)}) + O(mg(g \log q)^2 q^{(1+r)/2})))$$

and the required storage is

$$O(mgq^{(1+r)/2} \log q).$$

7. Computing a large enough set of smooth divisors

In this section, we construct a set of smooth divisors H such that $|H| > q^r$ using the following Algorithm 6.

Note that one can put $H' = H_m$. If we assume $H' = H_m$, the arguments of this paper also hold. Moreover, the proof of Lemma 8 becomes easier. However, form an experimental point of view, not using the almost smooth divisors $\bigcup_{i=1}^{m-1} H_i$, Algorithm 6. Computing H

Input: $H_1, H_2, \dots, H_m$ sets of almost smooth divisors s.t. $|H_m| > q^{(1+r)/2}$
Output: H a set of smooth divisors s.t. $|H| > q^r$.
Put $H' := \bigcup_{i=1}^m H_i$
$H \leftarrow H' \cdot H'$
return H

difficultly obtained, is wasteful. Then we ought to use $\bigcup_{i=1}^m H_i$. From this construction, $|H'| > |H_m| \geq q^{(1+r)/2}$. Similarly, we use the size estimation of Lemma 6 as heuristics and the size of H is estimated by

$$|H| = |H'|^2/q \geq q^r.$$

Note that $\left\{l(v) \mid v \in \bigcup_{i \leq m} H\right\} \leq 2mg$ and from Lemma 6, the cost for computing H is estimated by

$$O((\log q)^2 q^{(1+r)/2}) + O(mg(g \log q)^2 q^r)$$

and the required storage is estimated by

$$O(mg \log q q^{(1+r)/2}).$$

8. Two-way representation of $h \in H$

An element $h \in H_i$ is written by the form

$$h = n_{P_1} D(P_1) + \sum_{P \in B} a_P^{(h)} D(P),$$

since it is a almost smooth divisor. Moreover, from its construction, we easily see that

$$l(h) = \#\left\{P \in B \mid a_P^{(h)} \neq 0\right\} \leq ig.$$

Similarly, an element $h \in H$ is written by the form

$$h = \sum_{P \in B} a_P^{(h)} D(P),$$

since it is a smooth divisor. Moreover, from its construction, we see easily that

$$l(h) = \#\left\{P \in B \mid a_P^{(h)} \neq 0\right\} \leq 2mg.$$

Set $B = \{R_1, R_2, \dots, R_{|B|}\}$.

Definition 7. *For any $h \in H_i$ or H, put* $\mathbf{vec}(h) := \left(a_{R_1}^{(h)}, a_{R_2}^{(h)}, \ldots, a_{R_{|B|}}^{(h)}\right)$.

The computation of h ($= \mathbf{vec}(h)$) means the set of pairs $\left\{\left(a_{R_i}^{(h)}, R_i\right)\right\}$ for non-zero $a_{R_i}^{(h)}$. Note that the required storage for one h is $O(mg \log q)$.

On the other hands, from its construction, $h \in H_i$ is written by linear sum of at most i elements of $V_1 \cup V_2$. i.e.

$$h = \sum_{v \in V_1 \cup V_2} b_v^{(h)} v, \quad \#\left\{v \mid b_v^{(h)} \neq 0\right\} \leq i.$$

Similarly, $h \in H$ is written by linear sum of at most $2m$ elements of $V_1 \cup V_2$. i.e.

$$h = \sum_{v \in V_1 \cup V_2} b_v^{(h)} v, \quad \#\left\{v \mid b_v^{(h)} \neq 0\right\} \leq 2m.$$

Definition 8. *For any $h \in H_i$ or H, put* $\mathbf{v}(h) := \left\{\left(b_v^{(h)}, v\right) \mid b_v^{(h)} \neq 0\right\}$.

Note that the required storage for one $\mathbf{v}(h)$ is $O(m \log q)$.

By slightly modifying Algorithms 2, 3, 4, 5, 6, we can obtain both representations of h of the forms $\mathbf{vec}(h)$ and $\mathbf{v}(h)$. Note that the order of the cost and the order of the storage for computing H is essentially the same.

Further, we will assume that the computations of $\mathbf{vec}(h)$ and $\mathbf{v}(h)$ for each $h \in H_i$ or H are done.

9. Linear algebra

In this section, we will solve linear algebra and finding a linear relation of H by the following Algorithm 7.

Algorithm 7. Linear algebra

Input: H a set of smooth divisors such that $|H| > q^r$
Output: Integers $\{\gamma_h\}_{h \in H}$ modulo $|J_q|$ s.t. $\sum_{h \in H} \gamma_h h \equiv 0 \bmod |J_q|$
1: Set $H = \{h_1, h_2, \ldots, h_{|H|}\}$
2: Set matrix $M = ({}^t\mathbf{vec}(h_1), {}^t\mathbf{vec}(h_2), \ldots, {}^t\mathbf{vec}(h_{|H|}))$
3: Solve linear algebra of M and compute $(\gamma_1, \gamma_2, \ldots, \gamma_{|H|})$ such that $\sum_{i=1}^{|H|} \gamma_i \, \mathbf{vec}(h_i) \equiv \overrightarrow{0} \bmod |J_q|$
4: **return** $\{\gamma_i\}$

Note that the elements of matrix is integers modulo $|J_q| \doteq q^g$ and that the cost of an elementary operation modulo J_q is $O(g^2(\log q)^2)$.

M is a sparse matrix of the size $q^r \times q^r$. Note that the number of non-zero elements in one column is $2mg$. So, using [7, 9], the cost of computing $\{\gamma_i\}$ is estimated by

$$O(g^2(\log q)^2 \cdot 2mg \cdot q^r q^r) = O(mg^3(\log q)^2 q^{2r}).$$

The required storage for sparse linear algebra is essentially the storage for non-zero data. Note that the bit length of integer modulo $|J_q|$ is $\log(q^g)$ and that the number of nonzero elements of one row is mg. Thus the required storage is estimated by

$$O(\log(q^g)mg \cdot q^r) = O(mg^2 q^r \log q).$$

10. Nontrivial relation of the divisors in $V_1 \cup V_2$

In the previous section, we found $\{\gamma_h\}$ such that $\sum_{h\in H} \gamma_h h \equiv 0 \bmod |J_q|$. On the other hands, in order to solving DLP, the relation of collected reduced divisors $V_1 \cup V_2$ is desired. $h \in H$ is written by some linear sum $h = \sum_{v\in V_1\cup V_2} b_v^{(h)} v$. So, put

$$s_v := \sum_{h\in H} \gamma_h b_v^{(h)} \bmod |J_q| \quad \text{for all} \quad v \in V_1 \cup V_2$$

and we have the relation of the reudced divisors $V_1 \cup V_2$

$$\sum_{v\in V_1\cup V_2} s_v v = 0.$$

Algorithm 8. Computing s_v

Input: $V_1, V_2, H, \{\gamma_h\}_{h\in H}$ s.t. $\sum_{h\in H} \gamma_h h \equiv 0 \bmod |J_q|$
Output: $\{s_v\}_{v\in V_1\cup V_2}$
for all $v \in V_1 \cup V_2$ **do**
    $s_v \leftarrow 0$
od
for all $h \in H$ **do**
    for all $v \in V_1 \cup V_2$ s.t. $b_v^{(h)} \neq 0$ **do**
        $s_v \leftarrow s_v + \gamma_h b_v^{(h)}$
    od
od
return $\{s_v\}$

The cost of this part is

$$O(gq^{1+k} \log q) + O(mg^2 (\log q)^2 q^{(1+r)/2})$$

and the storage is

$$O(gq^{1+k} \log q).$$

Here, we will show that the obtained relation $\sum s_v v = 0$ is non-trivial.

Lemma 8. *$\{s_v\}_{v\in V_1\cup V_2}$ contains at least one non-zero element.*

Proof of this lemma is complicated, so we prepare the following two lemmas.

Lemma 9. *For any* $h \in H_i$*, there exists some* $v \in V_{2,i-1}$ *satisfying*

1) $b_v^{(h)} \neq 0$ *and*
2) $b_v^{(h')} = 0$ *for any* $h' \in \bigcup_{k=1}^{i} H_k \setminus \{h\}$.

Proof. h is written by the form $\phi(h[1], v, *)$ for $h[1] \in H_{i-1}$ and $v \in V_{2,i-1}$. We will show that this v satisfies the conditions of the lemma. Form the construction, we see $b_v^{(h)} \neq 0$. Further, we see that $b_v^{(h')} = 0$ for all $h' \in \bigcup_{k=1}^{i-1} H_k$, since this v is not used to the construction of $H_1, H_2, \ldots, H_{i-1}$. So, we have to show that $b_v^{(h')} = 0$ for all $h' \in H_i$. $h' \in H_i$ is written by the form $\phi(h'[1], v', *)$ for $h'[1] \in H_{i-1}$ and $v' \in V_{2,i-1}$. From the construction, we see $v \neq v'$, since H_i does not contains both elements of the form $\phi(h_1, v, *)$ and $\phi(h_2, v, *)$ $(h_1 \neq h_2)$. Then h' is written by the linear sum of $V_2 \setminus V_{2,i-1} \cup \{v'\}$, which does not contains the term of v (The 2-almost smooth divisors in $V_2 \setminus V_{2,i-1}$ are used the construction of $H_1, H_2, \ldots, H_{i-1}$.). Thus we have $b_v^{(h')} = 0$. □

Lemma 10. *Let* G *be a non-empty subset of* H*. Then there exists some* $g \in G$ *and some* $v \in V_2$ *satisfying*

1) $b_v^{(g)} \neq 0$ *and*
2) $b_v^{(g')} = 0$ *for all* $g' \in G \setminus \{g\}$.

Proof. $h \in H$ is written by the form $\phi(h[1], h[2], *)$ with $h[1] \in H_{i1}$, $h[2] \in H_{i2}$. Put $d(h) := \max(i_1, i_2)$. Take $g \in G$ whose $d = d(g)$ is maximal; i.e., $d = d(g) \geq d(g')$ for any $g' \in G$. g is written by the form $\phi(g[1], g[2], *)$ with $g[1] \in H_{d_1}$, $g[2] \in H_{d_2}$ and $\max(d_1, d_2) = d$. Without loss of generality, we can assume $d = d_1 \geq d_2 = d'$; i.e. $g[1] \in H_d$, $g[2] \in H_{d'}$ and $d \leq d'$. Let $g' \in G \setminus \{g\}$. g' is also written by the form $\phi(g'[1], g'[2], *)$. Then we see that $g[1] \neq g[2]$, $g[1] \neq g'[1]$, and $g[1] \neq g'[2]$, since form the construction of H, any 2 elements of the form $\phi(e_1, f, *)$, $\phi(e_2, f, *)$, $\phi(f, e_3, *)$, and $\phi(f, e_4, *)$ (f and e_i's are distinct) are not in H. Thus, we have $g[2], g'[1], g'[2] \in \bigcup_{k=1}^{d} H_k \setminus \{g[1]\}$. From the previous lemma, there exists some $v \in V_2$ satisfying

1) $b_v^{(g[1])} \neq 0$, $b_v^{(g[2])} = 0$ and
2) $b_v^{(g'[1])} = b_v^{(g'[2])} = 0$ for any $g' \in G \setminus \{g\}$.

Since g' is written by the linear sum of $g'[1]$ and $g'[2]$, we see that $b_v^{(g')} = 0$. Similarly, since g is written by the linear sum of $g[1]$ and $g[2]$, we see that $b_v^{(g)} \neq 0$. □

Now, return to the proof of Lemma 8.

Proof. Take $G := \{h \in H \mid \gamma_h \neq 0\}$. Then we see easily $s_v = \sum_{g \in G} \gamma_g b_v^{(g)}$. Applying the previous Lemma, there exists some $v \in V_2$ and some $g \in G$ satisfying

1) $b_v^{(g)} \neq 0$ and
2) $b_v^{(g')} = 0$ for any $g' \in G \setminus \{g\}$.

Thus we have $s_v = \gamma_g b_v^{(g)} \neq 0$. □

11. Finding discrete log

In the previous section, we found $\{s_v\}$ such that $\sum s_v v \equiv 0 \bmod |J_q|$. In the Part 2 of the algorithm, we computed (α_v, β_v) such that

$$v = \alpha_v D_1 + \beta_v D_2.$$

So, we have

$$\sum_{v\in V_1\cup V_2} s_v(\alpha_v D_1 + \beta_v D_2) = \left(\sum_{v\in V_1\cup V_2} s_v\alpha_v\right) D_1 + \left(\sum_{v\in V_1\cup V_2} s_v\beta_v\right) D_2 \equiv 0 \bmod |J_q|.$$

So, $-\left(\sum_{v\in V_1\cup V_2} s_v\alpha_v\right)/\left(\sum_{v\in V_1\cup V_2} s_v\beta_v\right) \bmod |J_q|$ is required discrete log. Since $\{s_v\}$ contains non-zero elements (Lemma 8), the probability $\sum_{v\in V_1\cup V_2} s_v\beta_v = 0 \bmod |J_q|$ is $1/|J_q|$ and can be omitted.

Algorithm 9. Computing λ

Input: $V_1, V_2, \{\alpha_v, \beta_v\}, \{s_v\}$
Output: Integer $\lambda \bmod |J_q|$ s.t. $D_1 = \lambda D_2$
1: **return** $-\left(\sum_{v\in V_1\cup V_2} s_v\alpha_v\right)/\left(\sum_{v\in V_1\cup V_2} s_v\beta_v\right) \bmod |J_q|$

Note that the cost of this part is $O(g^2 q^{1+k}(\log q)^2)$.

12. Cost estimation and optimization

In this section, we will estimate the cost and the required storage of the main algorithm under the assumption of

$$k = \frac{1}{\log q}.$$

First, remember that $m = O\left(\frac{1}{gk}\right) = O\left(\frac{\log q}{g}\right)$ (Lemma 7). By a direct computation, we have

$$r = r(k) = \frac{g-1+k}{g} = 1 - \frac{1}{g} + \frac{1}{g\log q},$$

and

$$q^{2r} = q^{2-\frac{2}{g}} \times \exp\left(\frac{2}{g}\right) = O\left(q^{2-\frac{2}{g}}\right).$$

From our cost estimation, the cost of the routine except Part 2 and Part 5 is written by the form

$$O(g^a(\log q)^b q^c) \quad a, b \le 4,\ c \le 1+k.$$

On the other hands, the cost of the routine Part 2 and Part 5 is written by

$$O(g^2(g-1)!\,(\log q)^3 q^{2r}) \quad \text{and} \quad O(mg^3(\log q)^2 q^{2r}).$$

From the definion of r, we see $1+k<2r$ and the cost of the whole parts can be estimated by

$$O(g^2(g-1)!\,(\log q)^3 q^{2r}) = O\big(g^2(g-1)!\,(\log q)^3 q^{2-\frac{2}{g}}\big).$$

Similarly, we see that the required storage (dominant part is Part 2 and Part 7, since $1+k>1>(1+r)/2$ from the definition of r) is

$$O(gq^{1+k}\log q) = O(gq^{1+k}\log q) = O(gq\exp(1)\log q) = O(gq\log q).$$

13. Conclusion

Thériault presented a variant of index calculus for the Jacobian of hyperelliptic curve of small genus, using almost smooth divisors. Here, we improve Thériault's result, using 2-almost divisors and propose an attack for DLP of the Jacobian of hyperelliptic curves of small genus, which works $O\big(q^{2-\frac{2}{g}+\epsilon}\big)$ running time.

Acknowledgment. The author would like to thank Professor Kazuto Matsuo in Institute of Information Security for useful comments and fruitful discussions, Professor Lisa Bond in Kanto-Gakuin University for English writing. The author also would like to thank the referees and the editor for useful coments and pointing out several mistakes.

References

[1] M. Adleman, J. DeMarrais and M.-D. Huang, A subexponential algorithm for discrete logarithms over the rational subgroup of the Jacobians of large genus hyperelliptic curves over finite fields. Algorithmic Number Theory, ANTS-I, LNCS, **877**, Springer-Verlag, 1994, 28–40.

[2] C. Diem, An Index Calculus Algorithm for Plane Curves of Small Degree. Algorithmic Number Theory—ANTS VII, LNCS, **4076**, Springer-Verlag, 2006, 543–557.

[3] A. Enge and P. Gaudry, A general framework for subexponential discrete logarithm algorithms. Acta Arith., **102** (2002), 83–103.

[4] P. Gaudry, An algorithm for solving the discrete log problem on hyperelliptic curves. Eurocrypt 2000, LNCS, **1807**, Springer-Verlag, 2000, 19–34.

[5] P. Gaudry, E. Thomé, Thériault and C. Diem, A double large prime variation for small genus hyperelliptic index calculus. Math. Comp., **76** (2007), 475–492.

[6] R. Granger and F. Vercauteren, On the Discrete Logarithm Problem on Algebraic Tori. Advances in Cryptology, CRYPTO 2005, LNCS, **3621**, Springer-Verlag, 2005, 66–85.

[7] B.A. LaMacchia and A.M. Odlyzko, Solving large sparse linear systems over finite fields. Crypto '90, LNCS, **537**, Springer-Verlag, 1990, 109–133.

[8] N. Thériault, Index calculus attack for hyperelliptic curves of small genus. ASIACRYPT 2003, LNCS, **2894**, Springer-Verlag, 2003, 75–92.

[9] D.H. Wiedemann, Solving sparse linear equations over finite fields. IEEE Trans. Inform. Theory, **32** (1986), 54–62.