Improvement of Thériault Algorithm of Index Calculus for Jacobian of Hyperelliptic Curves of Small Genus

Koh-ichi Nagao*
Dept. of Engineering, Kanto-Gakuin Univ.

May 20, 2004

Abstract

Gaudry present a variation of index calculus attack for solving the DLP in the Jacobian of hyperelliptic curves. Harley and Thérialut improve these kind of algorithm. Here, we will present a variation of these kind of algorithm, which is faster than previous ones.
Keywords Index calculus attack, Jacobian, Hyperelliptic curve, DLP,

1 Introduction

Gaudry [3] first present a variation of index calculus attack for hyperelliptic curves that could solve the DLP on the Jacobian of an hyperelliptic curve of small genus. Later, Harley(cf. [2]) and Thériault [1] improve this algorithm. In [1], these algorithms work in time $O(q^{2-\frac{2}{g+1}+\epsilon})$, and $O(q^{2-\frac{4}{2g+1}+\epsilon})$ respectively. Thériault's algorithm uses the almost-smooth divisor $D = \sum D(P_i)$ that all but one of the P_i's are in the set B called factor base. This technique was often used in the number field sieve factorization algorithm, which uses the almost-smooth integer $n = \prod p_i$, that all but one of the p_i's are in the factor base B, which is the set of small primes. In factorization algorithm, the cost of factorizing integer is larger than that of primary testing. So, the cost of factorizing almost-smooth integer is larger than that of normal integer of the same size, and the number that $p_i \notin B$ must be one. However, for the index calculus for the Jacobian of curves, we first compute the point of Jacobian and later consult whether it is almost smooth or not. So that, the new algorithm that use the 2-almost smooth divisors, that all but 2 of the P_i's are in the set B, is useful. For example, the almost smooth divisor of the form $v_1 = \sum$terms of $B+D(P_1)$, and the 2-almost smooth divisors of the form $v_2 = \sum$terms of $B+D(P_1)+D(P_2)$,

*nagao@kanto-gakuin.ac.jp

$v_3 = \sum$terms of $B+D(P_2)+D(P_3)$ are given, $v_1-v_2 = \sum$terms of $B-D(P_2)$, $v_1 - v_2 + v_3 = \sum$terms of $B + D(P_3)$ are other almost smooth divisors. So, we can get much more almost smooth divisors from gathering 2-almost smooth divisors. From this improvement, we get an attack of a running time of $O(q^{2-\frac{2}{g}+\epsilon})$.

2 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 $\deg h \leq g$.

Notation 1. *Use J_q for $\mathbf{Jac}_C(\mathbb{F}_q)$.*

Further, we will assume that $|J_q|$ is odd prime number, for simplicity.

Definition 1. *Given D_1,$D_2 \in J_q$ such that $D_2 \in < D_1 >$, 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 $\mathcal{P} := \{-P | P \in \mathcal{P}\}$.*

Proof. Since $|J_q|$ is odd prime, we have $2 \not| |J_q|$ and there are no point $P \in C(\mathbb{F}_q)$ such that $P = -P$. □

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

Point of $\mathbf{Jac}_C$ can be represented uniquely by the reduced divisor of the form

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

with $n_i \geq 0$ and $\sum n_i \leq g$.

Definition 2. *The reduced divisor of a point of Jacobian J_q is written by the elements of $C(\mathbb{F}_q)$ i.e.*

$$\sum_{i=1}^{k} n_i P_i - \sum_{i=1}^{k} n_i \infty, \quad P_i \in C(\mathbb{F}_q).$$

Then the point is said to be potentially smooth point.

Let $D(P) := P - \infty$. Note that $P + (-P) \sim 2\infty$. From lemma 1, potentially smooth point v of J_q can be represented of the form

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

with $n_P^{(v)} \in \mathbb{Z}$ and $\sum_{P \in \mathcal{P}} |n_P^{(v)}| \leq g$. Further, we will use this representation to potentially smooth points.

Definition 3. *A subset B of $\mathcal{P}$ used to define smoothness is called factor base.*

Definition 4. *A point $P \in \mathcal{P} \backslash B$ is called large prime.*

Definition 5. *A divisor v of the form*

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

is called smooth divisor.

Definition 6. *A divisor v of the form*

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

where P' is a large prime, is called 1-almost smooth divisor or almost smooth divisor.

Definition 7. *A divisor v of the form*

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

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

Definition 8. *An element $J \in J_q$ is called c point, if the reduced divisor representing J is smooth (resp. almost smooth, resp. 2-almost smooth) divisor.*

Further, we will consider the coefficients n_P of a smooth (resp. almost smooth, resp. 2-almost smooth) divisor modulo $|J_q|$. For a smooth (resp. almost smooth, resp. 2-almost smooth) divisor v, put

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

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 $\log |J_q| = (g \log q)^2$, 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 $0 < k < 1/2g$. 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 $|B| = q^r$.

Lemma 3.

$$2r > 1 + k > 1 > \frac{1+r}{2} = \frac{2g+k-1}{2g} > \frac{(g-1)+(g+1)k}{g}.$$

Proof. trivial. □

The whole algorithm consists of the following 7 parts.
Input $C/\mathbb{F}_q$ hyper elliptic curve of small genus g, $D_1, D_2 \in J_q$ such that $D_2 \in < D_1 >$.
Output Integer λ modulo $|J_q|$ such that $D_2 = \lambda D_1$.
1 Computing all points of $C(\mathbb{F}_q)$ and making $\mathcal{P}$ and fix $B \subset \mathcal{P}$ with $|B| = q^r$.
2 *Gathering 2-almost smooth divisors and almost smooth divisors*
Computing a set V_2 of 2-almost smooth points and a set V_1 of almost smooth points of J_q, of the form $\alpha D_1 + \beta D_2$ with $|V_1| > 2q^{\frac{(g-1)+(g+1)k}{g}}$ and $|V_2| > q^{1+k}$.
3 Computing a set of almost smooth divisor H_m with $|H_m| > q^{(1+r)/2}$.
4 Computing a set of smooth divisor H with $|H| > q^r$.
5 *Solving linear algebra of the size* $q^r \times q^r$
Computing integers $\{r_h\}_{h \in H}$ modulo $|J_q|$, satisfying $\sum_{h \in H} r_h h \equiv 0 \bmod |J_q|$.
6 Computing integers $\{s_v\}_{v \in V_1 \cup V_2}$ modulo $|J_q|$, satisfying $\sum_{v \in V_1 \cup V_2} s_v v \equiv 0 \bmod |J_q|$.
7 Computing λ.

4 Gathering 2-almost smooth points and almost smooth points

Algorithm 1 Gathering the 2-almost smooth pts and almost smooth pts

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 divisors, V_2 a set of 2-almost smooth divisors such that $|A_2| > q^{1+k}$, $|V_1| > 2q^{\frac{(g-1)+(g+1)k}{g}}$, Integers $\{(\alpha_v, \beta_v)\}_{v \in V_1 \cup V_2}$ such that $v = \alpha_v D_1 + \beta_v D_2$

1: $V_1 \leftarrow \{\}$, $V_2 \leftarrow \{\}$
2: **repeat**
3: Let α, β be random numbers modulo $|J_q|$
4: Compute $v = \alpha J_1 + \beta J_2$
5: **if** v is almost smooth **then**
6: $V_1 \leftarrow V_1 \cup \{v\}$
7: $(\alpha_v, \beta_v) \leftarrow (\alpha, \beta)$
8: **end if**
9: **if** v is 2-almost smooth **then**
10: $V_2 \leftarrow V_2 \cup \{v\}$
11: $(\alpha_v, \beta_v) \leftarrow (\alpha, \beta)$
12: **end if**
13: **until** $|A_2| > q^{1+k}$ and $|V_1| > 2q^{\frac{(g-1)+(g+1)k}{g}}$
14: **return** V_1,V_2,$\{(\alpha_v, \beta_v)\}_{v \in V_1 \cup V_2}$

Lemma 4. *The probability that a point in J_q is almost smooth is*

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

and the probability that a point is 2-almost smooth is

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

Proof. We can get above lemma similarly from proposition 3,4,5 in [1]. For example, the probability of 2-almost smooth points is roughly estimated by

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

□

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 $|V_1| > 2q^{\frac{(g-1)+(g+1)k}{g}}$ is estimated by

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

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 relative smooth point is $2g \log q$. So, the storage for V_1, the set of almost smooth divisors, is $O(g \log q\, q^{\frac{(g-1)+(g+1)k}{g}})$ and the storage for V_2, the set of 2-almost smooth divisors, is $O(g \log q\, q^{(1+k)})$. From lemma 3, we have $g \log q\, q^{(1+k)} >> g \log q\, q^{\frac{(g-1)+(g+1)k}{g}}$. So the total required storage can be estimated by

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

5 Elimination of large prime (Flame work)

Let E be a set of smooth divisors, and let F be a set of 2-almost smooth divisors or a set of smooth divisors. Note that element $e \in E$ and $f \in F$ are written by

$$e = \sum_{P \in B} n_P^{(e)} P + n_{P_1}^{(e)} P_1,$$

$$f = \sum_{P \in B} n_P^{(f)} P + n_{P_2}^{(f)} P_2 (+ n_{P_3}^{(f)} P_3).$$

Put $\sup(e) := \{P_1\}$ and $\sup(f) := \{P_2, (P_3)\}$. When $P \in \sup(e) \cap \sup(f)$, put

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

Trivially, $\phi(e, f, P)$ is almost smooth divisor, if F is a set of 2-almost smooth divisors and $\phi(e, f, P)$ is smooth divisor, if F is a set of almost smooth divisors and e is not of the form constant times f.

Definition 9.

$$e \heartsuit f := \begin{cases} \phi(e, f, P) & \textit{if } P \in \mathbf{sup}(a) \cap \mathbf{sup}(b) \textit{ and } e \neq \textit{Const} \times f \\ \emptyset & \textit{otherwise.} \end{cases}$$

$$E \heartsuit F := \cup_{e \in E, f \in F}\ e \heartsuit f.$$

Lemma 5. *$E \heartsuit F$ is a set of almost smooth (resp. smooth) divisors, if F is a set of 2-almost smooth (resp. almost smooth) divisors.*

Proof. Trivial! □

We will estimate the size of $E\heartsuit F$.

Lemma 6. *The size of $E\heartsuit F$ is estimated by*

$$|E\heartsuit F| = \begin{cases} |E||F|/q & \text{if } F \text{ is 2-almost smooth} \\ \frac{1}{2}|E||F|/q & \text{if } F \text{ is almost smooth} \end{cases}$$

Proof. Let $e \in E$, $f \in F$ be randomly chosen elements. Put $P := \sup(e)$. if F is a set of 2-almost smooth divisors (resp. almost smooth divisors), the probability that $P \in \sup(f)$ is $\frac{2}{|\mathcal{P}\backslash B|} \doteq \frac{1}{q}$ (resp. $\frac{1}{|\mathcal{P}\backslash B|} \doteq \frac{1}{2q}$), and the size is estimated by $\frac{1}{q} \times |E||F| (resp. \frac{1}{2q} \times |E||F|)$. □

In order to compute $E\heartsuit F$, we use this algorithm.

Algorithm 2 Heartsuit operator

Input: E, F
Output: $E\heartsuit F$
1: **set** $\mathcal{P}\backslash B = \{R_1, R_2, .., R_{|\mathcal{P}\backslash B|}\}$
2: **for** $i = 1, 2, .., |\mathcal{P}\backslash B|$ **do**
3: $st[i] \leftarrow \{\}$
4: **od**
5: **for all** $e \in E$ **do**
6: $P = \sup(e)$
7: Compute i s.t. $P = R_i$
8: $st[i] \leftarrow st[i] \cup \{e\}$
9: **od**
10: $V \leftarrow \{\}$
11: **for all** $f \in F$ **do**
12: **for all** $P \in \sup(f)$ **do**
13: Compute i s.t. $P = R_i$
14: **if** $st[i] \neq \emptyset$ **then**
15: **for all** $e \in st[i]$ s.t. $e \neq Const \times f$ **do**
16: $V \leftarrow V \cup \{\phi(e, f, P)\}$
17: **od**
18: **end if**
19: **od**
20: **od**
21: **return** H

We will estimate the cost and the storage for computing $E\heartsuit F$.

Lemma 7. *Put $c_1 := \max\{l(e)|e \in E\}$ and $c_2 := \max\{l(f)|f \in F\}$. Assume that $|E| << q$. Then the cost of computing $E\heartsuit F$ is*

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

. 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 V is $O((c_1+c_2)\log q\,|E||F|/q)$, since $|V| \doteq |E||F|/q$ and $\max\{l(v)|v \in V\} = c_1 + c_2$ from lemma 2.

Note that the cost of the routine "Computing index i" is $\log q \log|\mathcal{P}\backslash B| = O((\log q)^2)$. Also note that $|E \heartsuit F| = O(|E||F|/q)$ and remark that the probability of $st[i] \neq \emptyset$ is very small, since $|E| << q$. 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 V" of the 2nd loop is $O((c_1+c_2)(\log q)^2|E||F|/q)$ from lemma 2.

□

6 Computing H_m

In this section, we will construct H_m a set of almost smooth divisors $|H_m| > 2q^{(1+r)/2}$.

Algorithm 3 Computing H_m

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

1: $H_1 \leftarrow V_1$
2: $i \leftarrow 1$
3: **repeat**
4: $\quad i++$
5: $\quad H_i \leftarrow H_{i-1} \heartsuit V_2$
6: **until** $|H_i| > 2q^{(1+r)/2}$
7: $m \leftarrow i$
8: **return** m,H_m

From lemma 6, the size of H_i is estimated by

$$|H_i| = |H_1| \times (q^k)^{i-1} = 2q^{\frac{(g-1)+(g\,i+1)k}{g}}.$$

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

Lemma 8. *m is estimated by*

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

Further, we will assume $m = O(\frac{1}{gk})$. Note that $\{l(v)|v \in \cup_{i \leq m} H_i\} \leq mg$. From lemma 7, the cost for computing H_m is

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

and the required storage is

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

7 Computing H

In this section, we compute H a set of smooth divisors for $|H| > q^r$.

Algorithm 4 Computing H

Input: H_m a set of almost smooth divisors s.t. $|H_m| > 2q^{(1+r)/2}$
Output: H a set of smooth divisors s.t. $|H| > q^r$.
1: $H \leftarrow H_m \heartsuit H_m$
2: **return** H

From lemma 6, the size of H is estimated by

$$|H| = |H_m|^2/2q = 2q^r.$$

Note that $\{l(v) | v \in \cup_{i \leq m} H\} \leq 2mg$. From lemma 7, the cost for computing H is

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

and the required storage is

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

8 Two ways representation of $h \in H$

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, form its construction, we see easily that

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

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

Definition 10.

$$Put \qquad \mathbf{vec}(h) := (a_{R_1}^{(h)}, a_{R_2}^{(h)}, ..., a_{R_{|B|}}^{(h)}).$$

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

On the other hands, form its construction, h is written by linear sum of $2m$ elements of $V_1 \cup V_2$. i.e.

$$h = \sum_{v \in V_1 \cup V_2} b_v^{(h)} v, \qquad \#\{v \,|\, b_v^{(h)} \neq 0\} = 2m.$$

Definition 11.

$$Put \qquad \mathbf{v}(h) := \{(b_v^{(h)}, v) \,|\, b_v^{(h)} \neq 0\}.$$

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

Important Remark By little modifying the algorithm **3,4** , we can obtain both representations of h of the forms $\mathbf{vec}(h)$ and $\mathbf{v}(h)$. (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 $\mathbf{v}(h)$ are done.

9 Linear algebra

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

Algorithm 5 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 $H = \{h_1, h_2, ..., h_{|H|}\}$
2: Set matrix $M = ({}^t\mathbf{vec}(h_1), {}^t\mathbf{vec}(h_2), ..., {}^t\mathbf{vec}(h_{|H|}))$
3: Solve linear algebra of M and compute $(\gamma_1, \gamma_2, ..., \gamma_{|H|})$ such that $\sum_{i=1}^{|H|} \gamma_i \mathbf{vec}(h_i) = \vec{0}$
4: **return** $\{\gamma_i\}$

Note that the elements of matrix is integers modulo $|J_q| \doteq q^g$. So the cost of 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 [4] [5], we can compute $\{ \gamma_i\}$. Its cost is

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

and the required storage is

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

(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 $\log(q^g)$, the number of nonzero elements of one row is mg.)

10 Computing s_v

Remember that each element $h \in H$ is of the form $h = \sum_{v \in V_1 \cup V_2} b_v^{(h)} v$. In the previous section, we found $\{\gamma_h\}$ such that $\sum_{h \in H} \gamma_h h \equiv 0$ mod

$|J_q|$. So, put

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

and we have

$$\sum_{v \in V_1 \cup V_2} s_v v \equiv 0 \bmod |J_q|.$$

Algorithm 6 Computing s_v

Input: V_1,V_2,H,$\{\gamma_h\}_{h\in H}$ s.t. $\sum_{h\in H} \gamma_h h \equiv 0$
Output: $\{s_v\}_{v\in V_1\cup V_2}$
1: **for all** $v \in V_1 \cup V_2$ **do**
2: $s_v \leftarrow 0$
3: **od**
4: **for all** $h \in H$ **do**
5: **for all** $v \in V_1 \cup V_2$ s.t $b_v^{(h)} \neq 0$ **do**
6: $s_v \leftarrow s_v + \gamma_h b_v^{(h)}$
7: **od**
8: **od**
9: **return** $\{s_v\}$

The cost of this part is

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

and the storage is

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

11 Finding discreet 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 have computed (α_v, β_v) such that

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

So, we have

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

So, $-(\sum_{v\in V_1\cup V_2} s_v\alpha_v)/(\sum_{v\in V_1\cup V_2} s_v\beta_v) \bmod |J_q|$ is required discreet log.

Algorithm 7 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** $-(\sum_{v\in V_1\cup V_2} s_v\alpha_v)/(\sum_{v\in V_1\cup V_2} s_v\beta_v) \bmod |J_q|$

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

12 Cost estimation

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

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

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

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

and

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

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

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

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

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

From lemma 3, we see $1 + k < 2r$ and the cost of the whole parts can be estimated by

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

Similarly, we see that the required storage (dominant part is part 2 and part 7, since $1 + k > 1 > (1 + r)/2$ from lemma 3) is

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

13 Conclusion

In ASIACRYPT2003, 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 $O(q^{2-\frac{2}{g}+\epsilon})$ running time.

References

[1] N. Thériault, Index calculus attack for hyperelliptic curves of small genus, ASIACRYPT2003, LNCS 2894, Springer-Verlag, 2003, pp. 75–92.

[2] A. Enge, P. Gaudry, A general framework for subexponential discrete logarithm algorithms, *Acta Arith.,* **102**, no. 1, pp. 83–103,2002.

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

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

[5] D. H. Wiedemann, Solving sparse linear equations over finite fields, *IEEE Trans. Inform. Theory*, **IT-32**, no.1, pp.54–62, 1986.