A fast graph algorithm for genus-2 hyperelliptic curve discrete logarithm problems

Chia-Wei Lin and Rong-Jaye Chen
Institute of Computer Science and Engineering
College of Computer Science
National Chiao Tung University
{chiawei, rjchen}@cs.nctu.edu.tw

Abstract

In 1989, Koblitz proposed using the Jacobian of a hyperelliptic curve defined over a finite field to implement discrete logarithm cryptographic protocols. The discrete logarithm problem of the Jacobian is called hyperelliptic curve discrete logarithm problem (HCDLP). For a hyperelliptic curve of genus $g$ over the finite field $F_q$, the group order of the Jacobian is $O(q^g)$, which is larger than that of the additive group, which is $O(q)$, in an elliptic curve over $F_q$. Since there is no subexponential algorithm to solve HCDLP of small genus, hyperelliptic curve cryptosystem under applicable setting requires shorter key length than elliptic curve cryptosystem to achieve the same security level.

When genus $g$ is large enough, the index calculus attack has subexponential time complexity. For small genus HCDLP, the algorithms based on birthday paradox is of time complexity $O(q^g)$, and the basic index calculus attack is $O(q^{g/2})$. Thériault improves it by using the large prime method, and get a running time of $O(q^{g/2+1})$. Furthermore, Gaudry et al use a double large prime variation for small genus hyperelliptic index calculus, and the time complexity is $O(q^{g/2+1})$. In this thesis, we focus on the hyperelliptic curve discrete logarithm problem of small genus, implement and improve index calculus and its variations. We propose a fast graph algorithm for solving genus 2 HCDLP which time complexity is $O(q)$.

1 Introduction

In public-key cryptography, elliptic curve cryptography (ECC) [6] is an approach based on the algebraic structure of elliptic curves over finite fields. The use of elliptic curves in cryptography was suggested independently by Neal Koblitz and Victor S. Miller in 1985. The main advantage of ECC is its smaller key length, because there is no sub-exponential time algorithm to solve elliptic curve DLP (ECDLP). A 160-bit key in ECC is considered to be as secure as 1024-key in RSA.

However, hyperelliptic curve cryptosystems offer even smaller key length. In 1989, Koblitz [7] proposed using the Jacobian of a hyperelliptic curve defined over a finite field to implement discrete logarithm cryptographic protocols. Hyperelliptic curves are a special class of algebraic curves and can be viewed as generalizations of elliptic curves. There are hyperelliptic curves of every genus $g \geq 1$. A hyperelliptic curve of genus $g = 1$ is an elliptic curve. There is no known subexponential algorithm for hyperelliptic curves of small genus, and the Jacobian of a hyperelliptic curve of genus $g$ defined over a finite field $F_q$ has group order $O(q^g)$. Hence, the advantage of hyperelliptic curves over elliptic curves is that a smaller base field can be used in order to obtain the same level of security. This makes hyperelliptic curves suitable when only limited memory and computing power is available.

2 Definitions

An introduction to hyperelliptic curve cryptography can be found in [9].

We use $K$ to denote a field and $\overline{K}$ to denote the algebraic closure of $K$ in this chapter.

Definition 1 (Hyperelliptic curve)

A hyperelliptic curve of genus $g$ over $K$ is an equation of the form $C: y^2 + h(x)y = f(x)$ in $K[x, y]$, where $\deg(h(x)) \leq g$, $\deg(f(x)) = 2g+1$, $f(x)$ is a monic polynomial, and the integer $g \geq 1$. A hyperelliptic curve $C$ should be non-singular, that is, there are no solutions $(x, y) \in \overline{K} \times \overline{K}$ on curve $C$ which satisfy both partial derivative $2y + h(x) = 0$ and $h'(x)y - f'(x) = 0$.

Definition 2 (K-rational points)

The set $C(K) = \{(x,y) | x, y \in K$, where $h(x) = 0$, and $f(x)$ is a monic polynomial.
y² + h(x)y = f(x) \cup \{\infty\} is called the set of K-rational points on C. The point $\infty$ is called the point at infinity.

**Definition 3 (Divisor)**
A divisor D is a formal sum of points in C:
\[ D = \sum_{P \in C} m_P P, \quad m_P \in \mathbb{Z}, \text{ where only a finite number of } m_P \text{ is non-zero.} \]

**Definition 4 (Divisor group)**
The set of all divisors, denoted by D, forms an additive group under the addition rule:
\[ \sum_{P \in C} m_P P + \sum_{P \in C} n_P P = \sum_{P \in C} (m_P + n_P) P. \]
The set of all divisors of degree 0, denoted D₀, is a subgroup of D.

**Definition 5 (Principal divisor)**
Let \( R \in K(C) \). The divisor of R is called a principal divisor \( \text{div}(R) = \sum_{P \in C} \text{ord}_P(R) P \).

**Definition 6 (Principal divisor group)**
The group of principal divisors is a subgroup of D₀ and is defined by:
\[ P = P(C) = \{ \text{div}(R) \mid R \in K(C) \} \]. We have that \( P \subset D^0 \subset D \).

**Definition 7 (Jacobian)**
The Jacobian of the curve C is defined by the quotient group:
\[ J = J(C) = D_0/P. \]
If \( D_1, D_2 \in D_0 \) then we write \( D_1 \sim D_2 \) if \( D_1 - D_2 \in P \); \( D_1 \) and \( D_2 \) are said to be equivalent divisors.

**Definition 8 (HCDLP)**
Let C be a hyperelliptic curve over a finite field \( \mathbb{F}_q \) and \( I_2(F_q) \) its Jacobian with order \# I_2(F_q) = n. Given two reduced divisors \( D_1, D_2 \in I_2(F_q) \) and \( D_2 \in <D_1> \). The hyperelliptic curve discrete logarithm problem is to find the integer \( \lambda \in [0, n-1] \), such that \( \lambda D_1 = D_2 \).

In the following we introduce reduced divisor to uniquely represent the divisors in the same equivalence class of J.

**Definition 9 (Semi-reduced divisor)**
A semi-reduced divisor is a degree zero divisor of the form \( D = \sum_{P \in C} m_P P - \sum_{P \in C} m_P \infty \) with the following properties:
(i) \( m_P > 0 \),
(ii) if \( P \neq \overline{P} \) and \( m_P > 0 \) then \( m_{\overline{P}} = 0 \),
(iii) if \( P = \overline{P} \) and \( m_P > 0 \) then \( m_{\overline{P}} = 1 \).

**Definition 10 (Reduced divisor)**
Let \( D = \sum_{P \in C} m_P P - \sum_{P \in C} m_P \infty \) be a semi-reduced divisor. If \( \sum m_i \leq \text{genus} \) then D is the unique reduced divisor.

**Fact 1 (Mumford representation)**
For a hyperelliptic curve \( C: y^2 + h(x)y = f(x) \) in \( K[x,y] \), and
\[ D = \sum_{P \in \{x_i\} \in \mathbb{C}} m_P P - \sum_{P \in \mathbb{C}} m_P \infty \] be a semi-reduced divisor, we can use two polynomials \( a(x), b(x) \in K[x] \) to uniquely represent D. Let \( a(x) = \prod (x - x_i)^{m_i} \). Let \( b(x) \) be the unique polynomial satisfying:
- \( \deg(a) < \deg(b) \),
- \( b(x_i) = y_i \) for all \( i \) which \( m_i \neq 0 \),
- \( a(x) \) divides \( (b(x)^2 + b(x) h(x) - f(x)) \).

Then \( D = \text{gcd}(\text{div}(a(x))), \text{div}(b(x)-y)) \); we usually simplify the notation as \( \text{div}(a, b) \).
If \( D = \text{div}(a, b) \) is a reduced divisor, then \( \deg(a) = \sum m_i \leq \text{genus} \).

By using Mumford representation, Cantor’s algorithm [2] can effectively compute the group operation. It can be used to decompose a divisor into sum of divisors of smaller degree. It’s the key to use index calculus algorithm for HCDLP.

### 3 State of the Art
The best known algorithm for solving the DLP in generic groups is Pollard’s rho algorithm. Pollard’s algorithm has an exponential expected running time of \( \sqrt{n^2} \) group operations and negligible storage requirements. In order to prevent such square-root attacks, the group order \( n \) must have a large prime factor. There are faster algorithms for the DLP than Pollard’s rho method. The most powerful is the index calculus method which yields subexponential-time algorithms for the DLP in some groups.

The first subexponential-time algorithm to compute discrete logarithms over hyperelliptic curves of large genus is introduced by Adleman, DeMassais and Huang [1] in 1994. This algorithm was rather theoretical, and some improvements on it were done by other researchers. When the index calculus algorithm is applied on the small genus HCDLP, even the fastest variation is not faster than Pollard’s rho method for the genus less than 3. In order to analyze the security of such systems, we need to know how the index calculus method works for solving small genus HCDLP.
In 2000, Gaudry [4] first presented a variation of index calculus attack for a hyperelliptic curve of genus g over $F_q$ that could solve the HCDLP in time $O(q^2)$. And Harley improved this algorithm with reduced factor base such that HCDLP can be solved in time $O(q^{\frac{2}{6}+\epsilon})$ [4].

Furthermore, Thériault improved it by using the almost-smooth divisor which contains exactly one large prime. Theriault’s algorithm [11] works in time $O(q^{\frac{4}{9}+\epsilon})$.

By considering double large prime, the time complexity of hyperelliptic index calculus algorithm can be reduced to $O(q^{\frac{2}{9}+\epsilon})$. This idea was proposed independently by Gaudry et al. [5] and Naogo [10] in 2004. They used different tricks to handle large primes, but got the same time complexity.

However, the double large prime variation cannot be applied on genus 2 hyperelliptic curves. We propose an algorithm that can solve the genus 2 HCDLP with time complexity $O(q)$ which can be comparable to Pollard’s rho method.

Here is a guideline to the index calculus algorithm for solving HCDLP.

---

**Algorithm 1**

Hyperelliptic index calculus algorithm

**Input:** A divisor $D_1$ in $J\mathbb{C}(F_q)$ with known order $n = \text{ord}(D_1)$, and a divisor $D_2 \in <D_1>$.  

**Output:** An integer $\lambda$ such that $D_2 = \lambda D_1$.

- Fix smoothness bound $B$ and construct the factor base $F$.
- While not enough relations have been found do:  
  - Pick a random element $R = \alpha D_1 + \beta D_2$.  
  - If $R$ is smooth, record the corresponding relation.  
- Solve the linear algebra system over $\mathbb{Z}_n$.
- Return $\lambda$.

Table 1 shows the comparison between these algorithms described above. Our algorithm has the same time complexity as Pollard’s rho method but smaller hiding constant term.

### Table 1 Time complexity of algorithms solving HCDLP

<table>
<thead>
<tr>
<th>Genus g</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pollard’s rho</td>
<td>$q$</td>
<td>$q^{3/2}$</td>
<td>$q^2$</td>
<td>$q^{5/2}$</td>
<td>$q^3$</td>
</tr>
<tr>
<td>Original index calculus</td>
<td>$q^2$</td>
<td>$q^2$</td>
<td>$q^2$</td>
<td>$q^2$</td>
<td>$q^2$</td>
</tr>
<tr>
<td>with reduced factor base</td>
<td>$q^{4/3}$</td>
<td>$q^{3/2}$</td>
<td>$q^{8/5}$</td>
<td>$q^{5/3}$</td>
<td>$q^{12/7}$</td>
</tr>
<tr>
<td>with single large prime</td>
<td>$q^{6/5}$</td>
<td>$q^{10/7}$</td>
<td>$q^{14/9}$</td>
<td>$q^{18/11}$</td>
<td>$q^{22/13}$</td>
</tr>
<tr>
<td>with double large prime</td>
<td>—</td>
<td>$q^{4/3}$</td>
<td>$q^{3/2}$</td>
<td>$q^{8/5}$</td>
<td>$q^{5/3}$</td>
</tr>
<tr>
<td>Our algorithm</td>
<td>$q$</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

---

4 The Fast Graph Algorithm

Our algorithm first uses a pseudo random walk to create random reduced divisors of the form $R_i = \alpha D_1 + \beta D_2$. Then, create a graph $G$ with $|F|$ vertices corresponding to the elements in the factor base $F$, each edge specifying a relation written as the sum of the points. Initially the graph $G$ contains no edges. If $R_i$ is smooth then write $R_i$ as the sum of at most 2 points in $\mathbb{C}(F_q)$, and then add the corresponding edge between these two points. Notice that if $R_i$ is written as $R_i = cP_j - c\infty$ where $c=1$ or 2 and some $P_j \in \mathbb{C}(F_q)$ then it is an edge of self-loop of the point $P_j$.

For this graph algorithm, the data structure of the graph can be implemented as an array to represent trees with a union-find algorithm. In other words, we only need to record the parent node of each element in the array.
To test if adding an edge $(P_i, P_j)$ would create a cycle, we can traverse the trees from vertices $P_i$ and $P_j$ to see if they have the same root. If adding an edge would create an even length cycle then we get a relation $R = \sum \gamma_i R_i = (\sum \gamma_i \alpha_i) D_i + (\sum \gamma_i \beta_i) D_j = 0$ for some $\gamma_i$, such that the discrete logarithm of $D_i=\lambda D_j$ can be computed as $\lambda = -\sum\gamma_i \alpha_i / \sum\gamma_i \beta_i \mod n$. If adding an edge would create an odd length cycle then we can compute a relation $R=cP_i$ for some $\gamma_i$.

3.3 If $R$ is smooth and $R=cP_i$, we can compute $dR-cS=0$ which implies the self-loop in the roots of the trees without creating information of odd length cycles (including self-loop) in the roots of the trees without creating cycles in the graph $G$. If later we have another odd length cycle within the same tree then we can compute a relation $S=dP_i-d\infty$. With the information of the root $P_i$, the relations $R$ and $S$, we can compute $dR-cS=0$ which implies the discrete logarithm.

Here is our algorithm in detail.

**Algorithm 2**

A fast graph algorithm for genus 2 HCDLP

**Input:** A divisor $D_1$ in $J_0(F_q)$ with known order $n = \text{ord}(D_1)$, and a divisor $D_2 \in <D_1>$.

**Output:** An integer $\lambda$ such that $D_2=\lambda D_1$.

1. /* Build the factor base $F$ */
   
   For each $x_i \in F_q$, solve $v_2+h(x_i) \equiv f(x_i) \mod n$ to find
   
   $y_i \in F_q$ such that $(x_i, y_i) \in C(F_q)$, and store $P_i = (x_i, y_i)$ in $F$.

2. /* Initialization of the random walk */
   
   For $j$ from 1 to 20, select $a_i$ and $b_i$ at random in $[0, n-1]$, and compute
   
   $T_j := a_i D_1 + b_i D_2$.

   Select $c_i$ and $d_i$ at random in $[0, n-1]$ and compute
   
   $R_0 := c_i D_1 + d_i D_2$.

   $G \leftarrow \emptyset$ empty graph

3. /* Main loop */
   
   While $G$ contains no even length cycles or no component with 2 odd length cycles do
   
   3.1 $R_{i+1} \leftarrow R_i + T_i$ for some randomly chosen $j$.

   3.2 If $R_i$ is smooth and $R_i = P_i + P_j - 2\infty$, we first traverse the tree roots of $P_i$ and $P_j$. If they have the same root, then a cycle has been found. Otherwise, we merge the two trees by adding an edge between these two roots with corresponding relation.

**Example 1:**

Suppose we have collected an even length cycle in step 3 as shown in Figure 1.

$R_1 = P_1 + P_2 - 2\infty$,

$R_2 = P_2 + P_3 - 2\infty$,

$R_3 = P_3 + P_4 - 2\infty$,

$R_4 = P_4 + P_1 - 2\infty$.

Then we can compute the relation $R_1 - R_2 + R_3 - R_4 = 0$ in step 4. The discrete logarithm is also solved.

There is an example shows the steps to merge two trees.

**Example 2:**

Suppose we have $R_1$, $R_2$, and $R_3$ appear in the pseudo random walk.

$R_1 = P_1 + P_2 - 2\infty$,

$R_2 = P_2 + P_3 - 2\infty$,

$R_3 = P_3 + P_4 - 2\infty$.

We can link two trees by compute the relations of these two roots $P_1$ and $P_2$. Let $R_1 = R_1 + R_2 - R_3 = P_1 + P_4 - 2\infty$. Add the edge $R_1$ into the graph and discard $R_3$.

**5 Time Complexity**

In order to analyze the time complexity of this algorithm, we refer to Flajolet, Knuth and Pittel’s work[3] which provides comprehensive knowledge of the cycle appearance in random graphs. We quote some of their results here.

![Figure 1 An even length cycle.](image1)

![Figure 2 merge two trees.](image2)
Definition 11 (Uniform model)

The uniform model is a procedure to enrich an initially empty graph on the vertices \{1, 2, ..., n\}. At each step we generate an ordered pair \(<x, y>\), where \(x\) and \(y\) are uniformly distributed between 1 and \(n\), and all \(n^2\) pairs are equally likely. The (undirected edge) \(x - y\) is then added to the graph. In this way we obtain a multi-graph, which may have duplicate edges or self-loops \(x - x\).

A bicyclic component in a graph is a component with more than one cycle.

Corollary 1 (Expected time) [3]

In the uniform model, the first cycle appears at the expected time \(m = \frac{n^3}{3}\) steps. And at this time, the expected cycle length is of order \(n^{1/3}\), and the size of the component containing the first cycle will be \(\Theta\left(\frac{n^{1/3}}{\sqrt{\ln n}}\right)\). The waiting time for the first bicyclic component is approximately \(n/2\).

The graph constructed in our algorithm can be viewed as the uniform model with \(|F| = O(q)|\) vertices. At each step of pseudo random walk, the relation \(R = aD_1 + bD_2\) is smooth with probability 1/2. In other words, it is half chance to add an edge into the graph at each step. By Corollary 1, the first bicyclic component will appear in the graph after about \(q/2\) edges have been added. This requires about \(q\) steps of the pseudo random walk. Hence, we conclude our algorithm solving the genus 2 HCDLP in expected time of \(O(q)\) Jacobian operations.

A practical comparison between Pollard’s method and our algorithm is given in next section.

6 Computational Comparison

| Field size \(|F_q|\) | \(2^{11}\) | \(2^{13}\) | \(2^{17}\) | \(2^{19}\) |
|----------------|---------|---------|---------|---------|
| \(\text{Pollard's rho}\) | | | | |
| Average time (sec) | 1.238 | 5.502 | 113.391 | 827.459 |
| Average iterations | 923.4 | 2642.8 | 50239.5 | 236119.6 |
| Average number of useless collisions | 1.7 | 0.7 | 1.3 | 2.2 |
| \(\text{Our algorithm}\) | | | | |
| Average time (sec) | 0.236 | 1.018 | 17.394 | 80.809 |
| Average iterations | 699.4 | 2350.4 | 40222 | 137832 |
| Average number of smooth divisors | 351.4 | 1169.2 | 20222.3 | 74338.8 |
| Graph size | 1024 | 4071 | 65792 | 261993 |
| Average number of cycles | 2.1 | 2.2 | 2.9 | 2.9 |
References


