A double large prime variation for small genus hyperelliptic index calculus

P. Gaudry and E. Thomé

July 4, 2004

Abstract

An algorithm for solving discrete logarithms in jacobians of small genus hyperelliptic curves is presented and analyzed. This is a classical index-calculus algorithm, where Thériault's large-prime algorithm is extended to a double large prime variation algorithm. In order to give a rigorous analysis, a simplified model is introduced, whose behavior can be described by simple differential equations. The resulting complexity improves on the best known algorithms. The theoretical result is validated by computer experiments, that demonstrate that for genus 3 curves, this algorithm is faster than Pollard Rho method even for rather small sizes.

1 Introduction

The discrete logarithm problem in jacobians of hyperelliptic curves is known to be solvable in subexponential time if the genus is large compared to the base field size [1, 11, 3, 4]. The corresponding index-calculus algorithm still works for small fixed genus. The running time becomes exponential [6] but can be better than Pollard's Rho algorithm. Introducing a large prime variation, Thériault [13] manages to obtain an index calculus algorithm that is asymptotically faster than Pollard's Rho algorithm already for genus 3 curves.

In the present work, we go one step further in the direction of Thériault's algorithm, and introduce a double large prime variation for the small genus hyperelliptic index calculus. Our algorithm is indeed a very simple extension to Thériault's algorithm. However, making a rigorous analysis is not that easy: double large prime variations are commonly used in factorization algorithm, and analyzed via heuristics and experiments. Here our goal is to give a proven upper bound on the complexity. For that, we introduce a simplified model for the double large prime variation, that does not run faster than the real algorithm, and for which a rigorous analysis based on differential equations is feasible.

The main contribution of our work is a proof that there exists a probabilistic algorithm that solves a discrete logarithm problem in the jacobian of a hyperelliptic curve of genus $g \geq 3$ in time bounded by $O(q^{2-\frac{2}{g}})$, up to logarithmic factors. In this complexity estimate, g is supposed to be fixed and q tends to infinity. This improves on Thériault's $O(q^{2-\frac{2}{g+1/2}})$ algorithm.

The improvement is negligible for large genus, and therefore the case of genus 3 curves is of special interest. Therefore we gave it a special study and did practical experiments. For genus 3 curves, Pollard's Rho method has a running time in $O(q^{1.5})$, whereas Thériault's algorithm is in $O(q^{1.42})$ and ours is in $O(q^{1.33})$. The experiments validate our analysis, since the measured running times are in accordance with the theoretical predictions. Furthermore, we demonstrate than even for rather small jacobian sizes, our algorithm is much faster than Pollard's Rho algorithm.

As an application, when designing a hyperelliptic cryptosystem [8] based on a genus 3 curve, it is necessary to take into account our attack, and not only Pollard's Rho attack. The key-sizes should then be enlarged by about 12% to maintain the security level.

Another important contribution of our paper is the introduction of a new approach for analyzing double large prime variations in various contexts. The situation in hyperelliptic discrete logarithm

is quite favorable, since the probabilities are uniform in the sense that all the large primes have the same chance of occurring in a relation. This is no longer the case in the context of factorization or discrete logarithm in finite fields. Still, our model can probably be used to at least predict the number of relations that are needed before getting enough cycles.

The paper is organized as follows: in section 2 we recall Thériault's algorithm and present our variant. In section 3 we describe the graph modelization for the double large prime variation, and introduce the simplifications that allow the analysis that is made in section 4. In section 5 we put the results together to get the main theorem for the hyperelliptic discrete logarithm; and in section 6 we describe our computer experiments that validate our approach and show that it outperforms Pollard's Rho method rather early.

2 A double large prime variant of Thériault's algorithm

2.1 Setting

Let $\mathcal{C}$ be a hyperelliptic curve of genus $g \geq 3$ over a finite field $\mathbb{F}_q$ with q elements, given by an imaginary Weierstrass equation. The elements of the jacobian $\mathrm{Jac}(\mathcal{C})$ of $\mathcal{C}$ over $\mathbb{F}_q$ are handled via their Mumford's representation.

A discrete logarithm problem in $\mathrm{Jac}(\mathcal{C})$ is to be solved. Namely, a divisor D_1 of known order N and a divisor D_2 in the subgroup generated by D_1 are given. The goal is to compute the integer λ in $[0, N-1]$ such that $D_2 = \lambda D_1$.

2.2 Thériault's algorithm

There is a canonical injection of the curve $\mathcal{C}$ into its jacobian, that is used to define a factor base and a set of large primes.

Definition 1 *Let r be a real such that $0 < r < 1$. A subset $S_{\mathcal{B}}$ of $\mathbb{F}_q$ of size q^r is fixed arbitrarily. The* **factor base** $\mathcal{B}$ *is the set*

$$\mathcal{B} = \{P = (x, y) \in \mathcal{C} \subset \mathrm{Jac}(\mathcal{C}); \quad x \in S_{\mathcal{B}}\}.$$

The set of **large primes** $\mathcal{L}$ *is the set*

$$\mathcal{L} = \{P \in \mathcal{C} \subset \mathrm{Jac}(\mathcal{C})\} \setminus \mathcal{B}.$$

If the set $S_{\mathcal{B}}$ is chosen in a way which is independent of the arithmetical properties of $\mathrm{Jac}(\mathcal{C})$, heuristically we have

$$\#\mathcal{B} \approx q^r \quad \text{and} \quad \#\mathcal{L} \approx q.$$

Then Thériault's algorithm proceeds like any index-calculus algorithm with a large prime variation. Random linear combinations of D_1 and D_2 are computed and tested for smoothness, allowing at most one large prime. The smoothness test relies on the properties of Mumford's representation $\langle u(t), v(t) \rangle$ for divisors. The polynomial $u(t)$ is factored ; if it splits completely and all the roots but perhaps one lie in S_B, then we have a relation perhaps involving one large prime.

The analysis is made for a fixed value of g and letting q grow to infinity. At the heart of the analysis is the birthday paradox that tells that after having collected k relations involving large primes, they can be combined to form an expected number of $\frac{k^2}{2q}$ relations involving only elements of the factor base $\mathcal{B}$. Then estimating the probability of getting a relation with one large prime and balancing everything with the linear algebra step, the optimal value for r is $1 - \frac{1}{g+1/2}$, and the overall complexity is $O\left(q^{2-\frac{2}{g+1/2}}\right)$.

Hence Thériault obtains the following result [13].

Theorem 1 (Thériault) *A discrete logarithm problem in the jacobian of a hyperelliptic curve of genus $g \geq 3$ over a finite field $\mathbb{F}_q$ can be solved in time*

$$O\left(q^{2-\frac{2}{g+1/2}}\right),$$

where the constant depends on the genus g, and logarithmic factors in q are omitted.

2.3 A double large prime variation

To go beyond Thériault's result, it is natural to consider a double large prime variation. We keep the same factor base and large prime set as in Thériault's algorithm, but allow the parameter r to be set to some other value.

As usual, we form random linear combinations of D_1 and D_2, and test them for smoothness, but this time, we allow up to 2 large primes in each relation.

Definition 2 *A relation is said to be* **Full** *if it involves only elements of the factor base $\mathcal{B}$. A relation is said to be* **FP** *if it involves elements of $\mathcal{B}$ and exactly one large prime. A relation is said to be* **PP** *if it involves elements of $\mathcal{B}$ and exactly two large primes.*

Simple combinatorial arguments give good estimates for the probabilities of obtaining relations.

Proposition 1 *The probability for a random divisor to yield a Full relation is approximately*

$$q^{g(r-1)}/g!.$$

The probability for a random divisor to yield an FP relation is approximately

$$q^{(g-1)(r-1)}/(g-1)!.$$

The probability for a random divisor to yield a PP relation is approximately

$$q^{(g-2)(r-1)}/2(g-2)!.$$

Clearly, PP relations can be found much more quickly than FP relations. The problem is now to combine them in order to obtain more Full relations. For that, an adaptation of the *union-find* algorithm is used. It allows to solve this question in time almost linear in the number of PP relations we have.

A double large prime variation is profitable as long as the smoothness test is not substantially slowed down by the fact that more large primes are allowed. In our case, the cost of the smoothness test is unchanged by the double large prime variation, which implies that the overall complexity cannot be worse than with the single large prime variation. In the sequel, our goal will be to propose a way to analyze this double large prime variation, thus showing that the complexity of Thériault's algorithm can be improved.

The classical model of double large prime variation algorithms that is based on graph theory will be recalled and then simplified in order to get an upper bound on the expected running time.

3 Graph models for double large prime variation algorithms

In this section, we try to provide a general presentation of what a *double large prime variation* is, going beyond the context of integer factorization. When double large primes are used for index calculus algorithms, relations involve multiplicities, and squares can no longer be canceled out. This implies a more elaborate description than if we were to restrict ourselves to integer factorization.

3.1 Description of the LP-graph and its evolution

The **graph of large prime relations** (LP-graph, in short) is an undirected acyclic graph with $1 + \#\mathcal{L}$ vertices, corresponding to the elements of $\mathcal{L}$ and the special vertex 1. All edges of the LP-graph are labeled with a relation.

At the beginning of the algorithm there are no edges in the LP-graph, and a counter C is set to zero. The algorithm stops when C reaches some prescribed value $C_{\max}$. We typically have $C_{\max} = \#\mathcal{B} + O(1)$, hence $C_{\max} \ll \#\mathcal{L}$. The counter C must first be regarded as the number of independent cycles that *would* appear in the LP-graph in the course of its evolution. However no cycle is actually created.

We start our relation search. Each time we find a relation R, the LP-graph is modified according to the following procedure.

- If R is **Full**, the LP-graph is unchanged and the counter C is incremented.
- Otherwise we consider a new edge E, labeled by R, for potential inclusion into the LP-graph. If R is **FP**, the vertices of E are 1 and p_1 (the large prime appearing in R), while if R is **PP**, the vertices of E are the two large primes p_1 and p_2 appearing in R.

  We consider the following exclusive cases:

  - If E is already present (presumably with another label), the graph is unchanged and the counter C is incremented.
  - If adding E would not create any cycle, E is added to the LP-graph.
  - If adding E would create a cycle Γ, we are led to a technical distinction. Let $k = \#\Gamma$ be the number of edges that form Γ, $V(\Gamma)$ their vertices, and $R(\Gamma)$ their attached relations. $V(\Gamma)$ has cardinality k, and depending on whether $1 \in V(\Gamma)$ or not, the relations in $R(\Gamma)$ involve $k-1$ or k large primes, respectively. By linear algebra, we can obtain a linear combination of the relations in $R(\Gamma)$ which has the contribution of at least $k-1$ large primes canceled. Hence:
    - If $1 \in V(\Gamma)$, a Full relation can be obtained. C is increased, and the LP-graph is unchanged (note that a Full relation may also be obtained in lucky cases even when $1 \notin V(\Gamma)$; this "luck" is automatic in the classical case of the factorization of integers by the quadratic or number field sieve, because the linear algebra involved takes place over $\mathbb{F}_2$).
    - Otherwise, an FP relation can be obtained. The counter C is unchanged, and the procedure described is now applied to this FP relation.

It is now apparent that the counter C in fact represents the number of independent Full relations that can be obtained from the input relations considered. While this is clearly linked to the number of cycles, the last sub-case states the distinction between the two.

Implementing the LP-graph as described here together with its evolution process is efficiently done with the so-called *union-find* algorithm [12]. The processing time obtained is then essentially constant, and tiny, for each relation. As a result, the complexity of the double large prime variation algorithm is the average time to build a relation times the number of relations to build before the counter C reaches $C_{\max}$.

3.2 Observed behavior of the algorithm

An analysis of the double large prime variation, predicting precisely the number of relations to build before $C \geq C_{\max}$, is a challenging task. It is hard, even though our situation here is quite favorable: the large primes are all equiprobable. Studies such as [5] provide good knowledge of the cycle appearance in *random graphs*, but cannot be easily adapted to the present situation because of the bias introduced by the special vertex 1.

In order to tackle the analysis problem, we focus on the traits of the LP-graphs typically handled. The special vertex 1 attracts many relations around it. In a typical application, the

expected degree of the vertex 1 is several orders of magnitude greater than the average degree of the other vertices. Therefore, a huge connected component is quickly built around 1, much sooner than would occur in a random graph. The probability for other connected components to be "swallowed" by the huge connected component grows with their size. Eventually, the vast majority of the cycles yielding full relations (i.e. incrementing C) belong to the connected component of the vertex 1.

Based on this observation, we are tempted to concentrate the analysis on the giant connected component and forget about other parts of the graph.

3.3 A simplified algorithm

We propose a simplified algorithm which will be easier to analyze. Each time the simplified algorithm increments the counter C, the original algorithm would have done so, therefore the simplified algorithm runs no faster than the real algorithm. Its analysis will therefore yield an **upper bound** on the running time of the double large prime variation.

The simplified algorithm is the same as the algorithm described in subsection 2.3, except that no edge is added that is not connected to 1: if a PP relation arrives for which no large prime belongs to the connected component of 1 in the LP-graph, then the relation is just ignored (not even saved for later use). The technical discussion with cycles yielding or not yielding a Full recombined relation is useless here, and we can forget the labels on the edges.

We give the precise rules for completeness:

- If the relation is **Full**, the LP-graph is unchanged and the counter C is incremented.
- Otherwise we consider a new edge E whose vertices are either 1 and p_1, or p_1 and p_2, depending on whether the incoming relation is **FP** or **PP** (the p_i's are the large primes appearing in the relation).
  - If E would not be connected to the 1 vertex, do nothing.
  - If E is already present, the LP-graph is unchanged and the counter C is incremented.
  - If adding E would not create any cycle, the edge is added to the LP-graph (thus enlarging the connected component of 1).
  - Otherwise, adding E would create a cycle. The LP-graph is unchanged and the counter C is incremented.

Hence, the LP-graph is always a tree containing 1, modeling some subset of the giant connected component constructed in the previous algorithm.

At the beginning of the computation, all the PP relations are thrown away, until the first FP relation is encountered. As long as the number of FP relations built is small, most PP relations are useless, since the probability that they meet the tree is tiny. However, at a certain point, the giant component is large enough so that a reasonable proportion of the PP relations are useful and eventually yield an increase of the counter C.

4 Complexity analysis

In order to simplify the analysis, we switch from a discrete time to a continuous time. The unit of time we choose is the running time of each step in the attempt to produce relations. At the implementation level, this unit of time therefore correspond to an operation on the jacobian, and a smoothness test. The continuous point of view gives rise to first-order differential equations for key parameters, and their resolution leads to asymptotic running times.

4.1 Differential equations describing the simplified model

The two main quantities we will focus on are functions of the time denoted by $u(t)$ and $f(t)$:

- $u(t)$ is the proportion of large primes met at time t, expressed as a number between 0 and 1. In other words, this is the proportion of vertices connected to the vertex 1, at time t. Obviously, at $t = 0$, we have $u(0) = 0$.
- $f(t)$ denotes the value of the counter C at time t. The value $f(t)$ comprises genuine Full relations which occur (rarely) in the course of the relation search stage, as well as the relations obtained by recombining two or more FP or PP relations.

Since a relation (possibly Full, FP, PP, or not smooth at all) is produced during each unit of time, an interval of time $\mathrm{d}t$ yields:

- $a\,\mathrm{d}t$ Full relations,
- $b\,\mathrm{d}t$ FP relations,
- $c\,\mathrm{d}t$ PP relations,

where a, b, and c are the respective probabilities to produce a relation that is Full, FP or PP.

Hence it is possible to evaluate the evolution of $u(t)$ between t and $t + \mathrm{d}t$. Among the $b\,\mathrm{d}t$ FP relations produced in the interval of time, only $b(1-u)\,\mathrm{d}t$ are expected to bring a "new" large prime, and to enlarge the connected component related to 1.

The PP relations can be of three forms: the probabilities for such relations to have respectively 0, 1, or 2 of its large primes already in the giant component are respectively written as $(1-u)^2$, $2u(1-u)$, and u^2. Since we decide, in our algorithm, to drop relations with 0 known large primes, only those involving a single unknown large prime add a new large prime to the system. As many as $2cu(1-u)\,\mathrm{d}t$ such relations appear during the interval of time $\mathrm{d}t$.

This yields the following differential system for $u(t)$:

$$\begin{aligned} (\#\mathcal{L})\,u' &= b(1-u) + 2cu\,(1-u)\,, \\ u(0) &= 0. \end{aligned} \tag{1}$$

As for $f(t)$, an expression of $f'(t)$ happens to be easy as well. The number of "new" relations occurring within the interval of time $\mathrm{d}t$ is split into three contributions:

- Full relations: We can count on $a\,\mathrm{d}t$ of these.
- FP relations: We know that $b\,\mathrm{d}t$ such relations appear, but only those matching a large prime already met increments the counter C. Therefore the contribution is $bu\,\mathrm{d}t$.
- PP relations: Only the PP relations involving two large primes already met increment the counter C. This gives $cu^2\,\mathrm{d}t$.

The expression for $f'(t)$ is then:

$$\begin{aligned} f' &= a + bu + cu^2, \\ f(0) &= 0. \end{aligned} \tag{2}$$

4.2 Solution of the system

Solving these equations is not hard. First a closed form for $u(t)$ is obtained, and then integrating the expression for $f'(t)$ yields a formula for $f(t)$. In order to obtain a manageable expression of the result, we use a couple of abbreviations. Let

$$T = \frac{\#\mathcal{L}}{2c+b}, \qquad w(t) = \frac{b}{2c+b}\Big(\exp(t/T) - 1\Big),$$

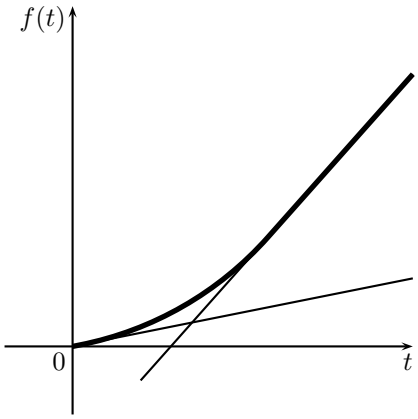

Figure 1: General form of $f(t)$

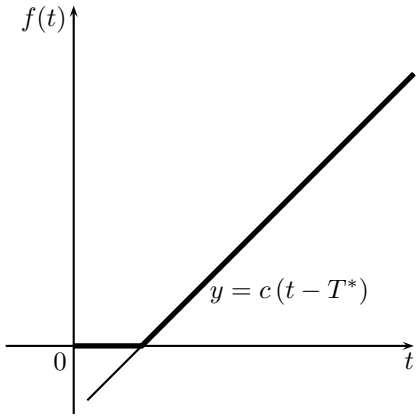

Figure 2: Large scale view of $f(t)$ when $a \ll b \ll c \ll 1$

then it is easy to check that an expression for $u(t)$ is

$$u(t) = 1 - \frac{1}{1 + w(t)}.$$

We see that $u(t)$ is an increasing function from 0 to 1, when t grows from 0 to infinity. This is consistent with the fact that at the beginning of the computation, no large prime has been met, whereas after a long time, we have found a relation involving each large prime.

The quantity $T^* \stackrel{\text{def}}{=\!=} T \log\left(\frac{2c}{b}\right)$ can be viewed as a **critical time**, since this is around this time that u jumps from 0 to 1. Indeed, assuming that $b \ll c$, for $t = T^* - 2T$, we have $u \approx 0.12$, whereas for $t = T^* + 2T$, we have $u \approx 0.88$.

From the expression of $u(t)$, we get the following expression for $f(t)$:

$$f(t) = \left(a - \frac{b^2}{4c}\right) t + \frac{\#\mathcal{L}}{2} \left(\left(1 + \frac{b}{2c}\right) \log(1 + w(t)) - \frac{w(t)}{1 + w(t)} \right).$$

When t grows to infinity, the expression $\log(1 + w(t))$ becomes equivalent to t/T, whereas $w(t)/(1+w(t))$ tends to 1, so that $f(t)$ becomes equivalent to $(a+b+c)t$. This is consistent, since after a long time, the graph has become a huge tree containing all the vertices, and each new Full, FP or PP relation yields a new cycle that increments the counter.

Hence the shape of the function $f(t)$ is as shown in figure 1, where the slope at 0 is a and the asymptotic line at infinity is close to $y = c\,(t - T^*)$, if we assume that $a \ll b \ll c \ll 1$.

But in fact, since a is tiny compared to c, at a large scale the first slope looks horizontal. At the transition point, the behavior of the function is locally exponential, which explains the "explosive growth" or "phase transition" terminologies encountered in the literature [9, 10]. The picture is then more like in figure 2, where the function $f(t)$ looks very angular.

4.3 Asymptotic analysis

The factor base is usually smaller than the set of large primes by an order of magnitude, so we can expect to have a relationship of the form $\#\mathcal{B} = (\#\mathcal{L})^r$, with $0 < r < 1$. The double large prime variation algorithm finishes when the number of genuine or recombined full relations is larger than the size of the factor base. Hence we want to evaluate the time t_f such that

$$f(t_f) = \#\mathcal{B}.$$

We make an asymptotic analysis for t_f, when $\#\mathcal{L}$ grows to infinity. The probabilities a, b and c are functions of $\#\mathcal{L}$. We make the following assumptions, which are verified in the case of the hyperelliptic curve discrete logarithm *à la* Thériault:

- $a \ll b \ll c \ll 1$;
- $a - b^2/4c > 0$;
- The contribution at_f of "genuine" Full relations is negligible in the final count of relations.

With these assumptions, the expressions for T, $w(t)$ and $f(t)$ can be simplified. In the following, we shall write $\alpha \sim \beta$ when α and β are functions of $\#\mathcal{L}$ and that α/β tends to 1 when $\#\mathcal{L}$ tends to infinity. Hence we have

$$\begin{aligned} T &\sim \frac{\#\mathcal{L}}{2c} \\ w(t_f) &\sim \frac{b}{2c}\left(\exp(t_f/T) - 1\right) \\ f(t_f) &\sim \left(a - \frac{b^2}{4c}\right) t_f + \frac{\#\mathcal{L}}{2}\left(\log(1 + w(t_f)) - \frac{w(t_f)}{1 + w(t_f)}\right). \end{aligned}$$

The first term in the equivalent of $f(t_f)$ is considered to be negligible, since it is bounded by the contribution of "genuine" Full relations. To analyze the behavior of the second term, we introduce the function

$$\tau(w) = \log(1 + w) - \frac{w}{1 + w},$$

where w takes positive values. Then we can write

$$f(t_f) \sim \frac{\#\mathcal{L}}{2}\,\tau(w(t_f)).$$

Since we want to evaluate t_f such that $f(t_f) = \#\mathcal{B}$, we have to solve

$$\tau(w(t_f)) \sim 2\frac{\#\mathcal{B}}{\#\mathcal{L}}.$$

The value of $\frac{\#\mathcal{B}}{\#\mathcal{L}}$ is close to zero by assumption. Thus, $w(t_f)$ must also be close to zero, because τ is a strictly increasing function from 0 to infinity. In the neighborhood of zero, $\tau(w)$ is equivalent to $w^2/2$, and we obtain that

$$w(t_f) \sim 2\sqrt{\frac{\#\mathcal{B}}{\#\mathcal{L}}}.$$

Replacing $w(t_f)$ by its equivalent $\frac{b}{2c}\left(\exp(t_f/T) - 1\right)$, we get

$$\boxed{t_f \sim \tfrac{\#\mathcal{L}}{2c}\,\log\left(1 + \tfrac{4c}{b}\sqrt{\tfrac{\#\mathcal{B}}{\#\mathcal{L}}}\right).} \tag{3}$$

In the special case of our extension of Thériault's algorithm, we shall give a more accurate estimate, taking into account several terms in the asymptotic expansion of t_f and not just the equivalent.

5 Back to the hyperelliptic curve discrete logarithm problem

5.1 Main result

Let $\mathcal{C}$ be a hyperelliptic curve of genus g over a finite field with q elements. Let $0 < r < 1$ to be fixed later, and let $\mathcal{B}$ and $\mathcal{L}$ be as in definition 1. We apply the double large prime variation algorithm to that curve. The probabilities a, b, c to get a Full, FP or PP relation are given by proposition 1. Before applying the analysis of the previous section, we take into account the

classical improvement which consists in keeping in the factor base and in the set of large primes only one representative for each class modulo the hyperelliptic involution. The probabilities a, b, c are unchanged, but now we have $\#\mathcal{L} \sim \frac{q}{2}$ and $\#\mathcal{B} \sim \frac{q^r}{2}$.

Putting these values inside the formula 3, we obtain an estimate of

$$\frac{(g-2)!}{2} q^{1-(g-2)(r-1)} \log\left(1 + 2(g-1)q^{(1-r)/2}\right)$$

elements to explore before finding enough relations with the simplified model. Since the true computation is not slower than the simplified model, this gives an upper bound on the running time. Note that this hides the complexity for arithmetic operations in the jacobian and smoothness tests, since these operations represent a unit of time.

We want to make an analysis with fixed genus and q growing to infinity. Therefore the upper bound is equivalent to

$$\frac{(1-r)(g-2)!}{4} q^{1-(g-2)(r-1)} \log q \quad \in \quad O(q^{1-(g-2)(r-1)} \log q).$$

The end of the discrete logarithm computation is the linear algebra step: finding a non-trivial vector in the kernel of a sparse matrix of size $\#\mathcal{B}$ using Lanczos or Wiedemann algorithm. This last step has complexity $O(q^{2r})$.

Ignoring the influence of the logarithmic factors, we can balance the two phases by taking $r = 1 - \frac{1}{g}$. Finally, we get the following result.

Theorem 2 *A discrete logarithm problem in the jacobian of a hyperelliptic curve of genus $g \geq 3$ over a finite field $\mathbb{F}_q$ can be solved in time*

$$O\left(q^{2-\frac{2}{g}}\right),$$

where the constant depends on the genus g, and logarithmic factors in q are omitted.

For small genera we obtain the following complexities:

g	3	4	5
Pollard's algo	$q^{3/2}$	q^2	$q^{5/2}$
Basic index calculus	q^2	q^2	q^2
Thériault's algo	$q^{10/7}$	$q^{14/9}$	$q^{18/11}$
Our algo	$q^{4/3}$	$q^{3/2}$	$q^{8/5}$

Obviously, when the genus gets large, the improvement is marginal, not to say invisible. On the other hand, for genus 3 curves, the $O(q^2)$ complexity of the basic index calculus drops to $O(q^{1.428})$ with Thériault's algorithm and to $O(q^{1.333})$ with our double large prime variant. The constants involved are small enough so that even for small sizes our algorithm should be faster than Pollard's Rho algorithm.

5.2 First terms in the asymptotic expansion of the runtime for $g = 3$

In section 6 below, we make computer experiments for genus 3 curves. In order to obtain an accurate validation of our analysis of the simplified model, we calculate here a few terms in the Taylor expansion of t_f in this particular case.

Let us denote by Q the quantity $\sqrt{\frac{\#\mathcal{B}}{\#\mathcal{L}}} \sim q^{-1/6}$, which is close to zero by assumption. We want to find the value of $w(t_f)$ such that $\tau(w(t_f)) = 2Q^2$. In other words, we need the Taylor expansion in Q of the function $w(t_f)$ satisfying the above equation. A simple calculation gives

$$\begin{aligned} w(t_f) &= 2Q + \frac{8}{3}Q^2 + \frac{26}{9}Q^3 + \frac{368}{135}Q^4 + O(Q^5) \\ &= 2q^{-1/6} + \frac{8}{3}q^{-1/3} + \frac{26}{9}q^{-1/2} + \frac{368}{135}q^{-2/3} + O(q^{-1}). \end{aligned}$$

Now, the expression for $w(t_f)$ in terms of t_f is

$$w(t_f) = \frac{q^{-1/3}}{2}\left(\exp(2t_f/q^{4/3}) - 1\right).$$

Equating those two expressions, and solving for t_f yields:

$$\begin{aligned} t_f &= \frac{q^{4/3}}{2}\log\left(1 + 2q^{1/3}\left(2q^{-1/6} + \frac{8}{3}q^{-1/3} + \frac{26}{9}q^{-1/2} + \frac{368}{135}q^{-2/3} + O(q^{-1})\right)\right) \\ &= \frac{q^{4/3}}{2}\log\left(4q^{1/6} + \frac{16}{3} + \frac{52}{9}q^{-1/6} + \frac{736}{135}q^{-1/3} + O(q^{-2/3})\right) \quad (4) \\ &= \frac{1}{12}q^{4/3}\log q + q^{4/3}\log 2 + \frac{2}{3}q^{7/6} + \frac{5}{18}q + O(q^{5/6}). \quad (5) \end{aligned}$$

We see that for relatively small values of q (for which experiments are feasible), at least the second term will not be negligible, since $\frac{1}{12}\log q$ could be comparable to $\log 2$.

There is not much sense in computing the expansion further, since we started with the approximation that $\#\mathcal{L} \approx q$ and $\#\mathcal{B} \approx q^r$. By Hasse-Weil's bound, these approximations are tight up to roughly the square root of the value. Hence our result cannot be more precise than roughly $\sqrt{q^{4/3}} = q^{2/3}$.

6 Computer experiments, validation of the model

6.1 Validation of the analysis

We implemented the double large prime algorithm described for hyperelliptic curves of genus 3 defined over prime fields. The first series of experiments shown in table 1 aims at validating our analysis. Table 1 lists the final value of t for several experiment sizes. We provide data for the simplified model for which we have done the analysis above, as well as for the real algorithm (as described in section 2.3). We recall that t represents the number of trial relations to test for smoothness before sufficiently many recombined relations can be obtained. The running time of the whole algorithm (without linear algebra) is t times a polynomial expression in $\log q$ accounting for jacobian arithmetic and smoothness tests.

The third column of table 1 gives the ratio of the experimental value obtained in the simplified model versus the theoretical value predicted in section 5.2. We compared t with the expression $\frac{q^{4/3}}{2}\log\left(4q^{1/6} + \frac{16}{3}\right)$, which is derived directly from equation 4. As the data shows, the accordance of experimentation and theory is striking, since the experimental value is in most cases within 1% of its theoretical expectation. It should be noted that for each experiment size, only one run of the algorithm was done. This accounts most certainly for the small variance observed for $q \approx 2^{15}$ and $q \approx 2^{17}$.

The *real* algorithm gives of course shorter running times. We give the comparison of t with $q^{4/3}$ in the fifth column. The ratio seems to be constant, which suggests that the asymptotic value to be expected for this model is $\Theta(q^{4/3})$, with *no* logarithmic factors (except of course for those related to the jacobian arithmetic and smoothness tests).

6.2 Running times and comparison with Pollard Rho

The implementation of the algorithm was done in `C/C++`. We programmed the jacobian arithmetic using explicit formulae, based on the work of [16]. We give timings for a random curve defined over a prime field of cardinality roughly 2^{27}. On a Pentium-M processor clocked at 1.7 GHz, our implementation performs 200 000 jacobian additions or doublings per second (i.e. 5 microseconds each). A step of the algorithm, corresponding to the unit of time t chosen in the analysis above, is performed in 20 microseconds. This includes the time for the smoothness test.

q	simplified model		real algorithm		# cycles
	t	t/theory	t	$t/q^{4/3}$	
$\approx 2^{15}$	2000667	1.15	815473	0.78	512
$\approx 2^{16}$	4500744	0.99	1811672	0.69	812
$\approx 2^{17}$	12551636	1.07	4705192	0.71	1290
$\approx 2^{18}$	30904671	1.02	11253002	0.67	2047
$\approx 2^{19}$	79709007	1.01	27776102	0.66	3250
$\approx 2^{20}$	200813292	0.99	66834647	0.63	5160
$\approx 2^{21}$	532137499	1.01	170327927	0.63	8191
$\approx 2^{22}$	1373241051	1.01	417044579	0.62	13003
$\approx 2^{23}$	3543302265	1.01	1036566361	0.61	20642
$\approx 2^{24}$	9143409061	1.00	2576921045	0.60	32767
$\approx 2^{25}$	23727506320	1.01	6430349490	0.59	52015
$\approx 2^{27}$	157537516376	1.00	39993810485	0.58	131071

Table 1: Final value of t for the two LP-graph models

q	Relation search	Linear algebra	Total	Pollard Rho (estim.)
2^{24}	0.6 days	0.2 days	0.8 days	3.5 days
2^{27}	9 days	5.8 days	14.8 days	79 days

Table 2: Total time for our algorithm and Pollard Rho

The Pollard Rho algorithm is known to have $O(\sqrt{\#G})$ complexity. More precisely, in the case of a prime order jacobian of a hyperelliptic curve of genus three, $\sqrt{\pi \#J}/2$ jacobian operations are needed (we take advantage of the hyperelliptic involution). Instantiated with the parameters for a genus three curve over $\mathbb{F}_q$, where $q \approx 2^{27}$, this yields $1.37 * 10^{12}$ operations in the jacobian, or, at the pace quoted above, 79 days of computation on a Pentium-M processor.

In comparison, the index calculus algorithm described here, with the double large prime variation, requires only $4 * 10^{10}$ jacobian operations and smoothness tests on the same curve as above. This corresponds to 9 days of computation. We performed the corresponding linear algebra computation, using as a linear system solver the block Wiedemann implementation described in [2, 14, 15]. This linear algebra computation required 5.8 days of computation on the same processor. Therefore, the algorithm presented here induces a speed-up of 5.3 compared to Pollard Rho for this problem size. For a curve defined over a field of size 2^{24}, the corresponding speed-up is already of 4.4. Using our implementation, a definition field of size 2^{27} corresponds roughly to the crossover point between Pollard Rho and Thériault's algorithm.

Note that because of the linear algebra step, the index calculus approach cannot enjoy the same amount of parallelization as Pollard's algorithm and its variants. Partial distribution of the linear algebra is possible through the use of multi-processor machines, and taking advantage of the blocking capabilities of the block Wiedemann algorithm. We have been able to reduce the linear algebra time to 1.9 days this way, with room for further improvement since we have not ported yet the asymptotically fast algorithm presented in [14].

7 Conclusion

We have described an algorithm for solving discrete logarithms in hyperelliptic curves of small genus at least 3, that is faster than previously known methods. The tricky part was to provide with a rigorous analysis of a double large prime variation of Thériault's algorithm. This theoretical analysis was validated by computer experiments, that also demonstrated that even for rather small sizes, our method is better than Pollard Rho algorithm.

The direct application to cryptography is that a genus 3 hyperelliptic cryptosystem should have a key size about 12% larger than an elliptic curve cryptosystem for an equivalent security. Furthermore, our method also applies to the Weil descent algorithm of [7] that attacks elliptic curves defined over small extension fields. Hence, this 12% penalty also applies to elliptic curve cryptosystems defined over extension finite fields whose degree is a multiple of 3.

Finally, we believe that the simplified model we used for the analysis could be used in other contexts such as factorization or classical discrete logarithm algorithms in order to have a better understanding of the double large prime variations that are often used. Even though we have resorted to the simplified model in order to achieve the analysis, the experimental evidence gathered in table 1 supports the idea that a fairly reliable prediction of the number of relations needed is possible even for the real algorithm, and not limited to the context studied in this paper.

References

[1] L. 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. In L. Adleman and M.-D. Huang, eds., *ANTS-I*, vol. 877 of *Lecture Notes in Comput. Sci.*, pp. 28–40. Springer–Verlag, 1994.

[2] D. Coppersmith. Solving linear equations over GF(2) via block Wiedemann algorithm. *Math. Comp.*, 62(205):333–350, Jan. 1994.

[3] A. Enge. Computing discrete logarithms in high-genus hyperelliptic Jacobians in provably subexponential time. *Math. Comp.*, 71:729–742, 2002.

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

[5] P. Flajolet, D. Knuth, and B. Pittel. The first cycles in an evolving graph. *Discrete Math.*, 75:167–215, 1989.

[6] P. Gaudry. An algorithm for solving the discrete log problem on hyperelliptic curves. In B. Preneel, ed., *Advances in Cryptology – EUROCRYPT 2000*, vol. 1807 of *Lecture Notes in Comput. Sci.*, pp. 19–34. Springer–Verlag, 2000. Proceedings.

[7] P. Gaudry. Index calculus for abelian varieties and the elliptic curve discrete logarithm problem. Cryptology ePrint Archive: Report 2004/073, 2004.

[8] N. Koblitz. Hyperelliptic cryptosystems. *J. of Cryptology*, 1:139–150, 1989.

[9] A. K. Lenstra and M. S. Manasse. Factoring with two large primes. *Math. Comp.*, 63(208):785–798, Oct. 1994.

[10] P. Leyland, A. K. Lenstra, B. Dodson, A. Muffett, and S. S. Wagstaff, Jr. MPQS with three large primes. In C. Fieker and D. R. Kohel, eds., *ANTS-V*, vol. 2369 of *Lecture Notes in Comput. Sci.*, pp. 448–462. Springer–Verlag, 2002. Proceedings.

[11] V. Müller, A. Stein, and C. Thiel. Computing discrete logarithms in real quadratic congruence function fields of large genus. *Math. Comp.*, 68(226):807–822, 1999.

[12] R. Sedgewick. *Algorithms*. Addison–Wesley, second edition, 1988.

[13] N. Thériault. Index calculus attack for hyperelliptic curves of small genus. In C. Laih, ed., *Advances in Cryptology – ASIACRYPT 2003*, vol. 2894 of *Lecture Notes in Comput. Sci.*, pp. 75–92. Springer–Verlag, 2003. Proceedings.

[14] E. Thomé. Subquadratic computation of vector generating polynomials and improvement of the block Wiedemann algorithm. *J. Symbolic Comput.*, 33(5):757–775, Jul. 2002.

[15] E. Thomé. *Algorithmes de calcul de logarithme discret dans les corps finis*. Thèse, École polytechnique, May 2003.

[16] T. Wollinger, J. Pelzl, and C. Paar. Cantor versus Harley: Optimization and analysis of explicit formulae for hyperelliptic curve cryptosystem. Technical report, Universität Bochum, 2004. Available at `http://www.crypto.rub.de/`.