# Improved Sieving on Algebraic Curves

Vanessa Vitse, Alexandre Wallet

# Improved Sieving on Algebraic Curves

Vanessa Vitse[1] and Alexandre Wallet[2,3]

[1] Institut Fourier, UJF-CNRS, UMR 5582, 38402 Saint-Martin d'Hères, France
[2] Sorbonnes Universités, UPMC Univ Paris 06, CNRS, INRIA, LIP6 UMR 7606, 4 place Jussieu 75005 Paris, France
[3] Projet POLSYS, INRIA Rocquencourt, 78153 Le Chesnay Cedex, France
vanessa.vitse@ujf-grenoble.fr, alexandre.wallet@lip6.fr

**Abstract.** The best algorithms for discrete logarithms in Jacobians of algebraic curves of small genus are based on index calculus methods coupled with large prime variations. For hyperelliptic curves, relations are obtained by looking for reduced divisors with smooth Mumford representation [4]; for non-hyperelliptic curves it is faster to obtain relations using special linear systems of divisors [2, 3]. Recently, Sarkar and Singh have proposed a sieving technique, inspired by an earlier work of Joux and Vitse, to speed up the relation search in the hyperelliptic case. We give a new description of this technique, and show that this new formulation applies naturally to the non-hyperelliptic case with or without large prime variations. In particular, we obtain a speed-up by a factor approximately 3 for the relation search in Diem and Kochinke's methods.

**Keywords :** discrete logarithm, index calculus, algebraic curves, curve-based cryptography.

## 1 Introduction

Given a commutative group $(G, +)$, and two elements $g, h$ of $G$, the discrete logarithm problem (DLP) consists in finding, if it exists, an integer $x$ such that $x \cdot g = h$. It is considered as a major computational challenge and much work has been done on the principal families of groups in the last decades. Jacobians of algebraic curves defined over finite fields are of particular interest, largely because of their link with elliptic curves (which, as genus 1 curves, are their own Jacobians). Indeed, transfer attacks as introduced in [5] can reduce certain discrete logarithm instances on elliptic curves to instances on higher genus curves, see for instance the record computations of [7]. Consequently, even though for applications in cryptography only genus 1 (i.e. elliptic curves) or genus 2 curves are currently considered, assessing the exact difficulty of the DLP in higher genus is still very important from both the practical and the number-theoretical points of view.

A popular approach to the discrete logarithm problem is given by the index calculus family of algorithms, that rely heavily on the structures surrounding the group. The general picture is always the same. We start by choosing a so-called "factor base" $\mathcal{B}$, i.e. a subset of the elements of the group. Then in a first

phase we search for relations involving elements of the factor base, yielding linear equations between their discrete logarithms; we call this first phase "harvesting" throughout this paper. In a second phase we basically solve the linear system of equations given by the relation matrix and deduce the wanted discrete logarithm. This linear algebra phase does not depend on the structure of the group, but its complexity clearly depends on the cardinality $N$ of the factor base, since it is roughly the size of the relation matrix.

It is possible to simplify the linear algebra at the expense of the harvesting phase by decreasing the size of the factor base. This must be done with added care since less elements in the factor base means lower probability of obtaining a relation. In the so-called "large prime" variants[4] (see [2, 6, 8, 11]), we choose or construct a subset $\mathcal{B}_s$ of the factor base $\mathcal{B}$ called the set of "small primes", of size $O(N^\alpha)$ with $0 < \alpha < 1$. The set $\mathcal{B} \backslash \mathcal{B}_s$ is the set of large primes. Then we accept only relations involving elements in $\mathcal{B}_s$ and at most two large primes. This decreases the probability of a relation, but it is shown in [2, 6] that finding $O(N)$ relations involving at most two large primes is enough to form $O(N^\alpha)$ relations involving only small primes. The linear algebra phase over $\mathcal{B}_s$ then runs in $O(N^{2\alpha})$ and it remains to choose $\alpha$ to balance both main phases of index calculus. More details are given at the end of Sect 2.1.

In this work we focus on improving the known harvesting methods, dedicating to Jacobian varieties of algebraic curves defined over finite fields. We emphasize that all the other aspects of the index calculus method (such as the choice of the factor base, the processing of large prime relations and the linear algebra phase) are not modified. Recently, Sarkar and Singh proposed in [10] to use a sieving technique for harvesting relations in the hyperelliptic case, instead of the standard approach of Gaudry [4] based on smooth reduced divisors. A very similar sieve had actually been used before by Joux and Vitse in [7], but in the different context of curves defined over extension fields and Weil restrictions. It turns out that Sarkar and Singh's sieve has a simpler interpretation, which allows to generalize it to the index calculus introduced by Diem [2] for non-hyperelliptic curves, or more exactly small degree planar curves. In our experiments, the new non-hyperelliptic sieve improves Diem's original method, as well as its development by Diem and Kochinke [3], by a factor approximately 3.

The presentation follows these steps. We begin by the case of hyperelliptic curves, recalling the classical approach of Gaudry based on smoothness check. We present and analyze the sieving variant of Sarkar and Singh before introducing its simpler reformulation. The next section deals with algebraic curves of genus $g$ admitting a plane model of degree $d \leq g+1$. Again we start by the classical ideas of using principal divisors associated to equations of lines to generate relations [2]. We then give the adaptation of our sieve to small degree curves, and compare it to Diem's method. We also briefly present the singularity-based technique of Diem and Kochinke and show that our sieve adapts again to this setting. Experiments and timings are reported in the last section.

[4] The terminology of index calculus stems from the context of integer factorization. In our setting, "large primes" are arbitrary elements, and involve no notion of size.

## 2 Sieving for Hyperelliptic Curves

### 2.1 Gaudry's Relation Search

Index calculus on hyperelliptic curves relies on the search of smooth Mumford representations of divisors. Let $\mathcal{H}$ be a genus $g$ imaginary hyperelliptic curve defined over $\mathbb{F}_q$, with equation $y^2 = h(x)$ (so that $\deg h = 2g+1$) and point at infinity $P_\infty$. For simplicity we assume that we are in the odd characteristic case, but everything can be easily adapted to the characteristic two case. We recall that any element $[D]$ in the Jacobian variety $\mathrm{Jac}_{\mathcal{H}}(\mathbb{F}_q)$ can be uniquely represented by a couple $[u(x), v(x)]$ of polynomials such that:

- $u$ is monic and $\deg u \leq g$;
- $\deg v < \deg u$;
- $u | (h - v^2)$.

More precisely, this is the Mumford representation of the unique reduced divisor $D = (P_1) + \cdots + (P_{\deg u}) - \deg u\,(P_\infty)$ in the class $[D]$. The roots of $u$ are exactly the $x$-coordinates of the points $P_i \in \mathcal{H}(\overline{\mathbb{F}_q})$, and $v$ satisfies $v(x_{P_i}) = y_{P_i}$. The Mumford representation is actually defined for every *semi*-reduced divisor, i.e. of the form $(P_1) + \cdots + (P_w) - w(P_\infty)$ with $P_i \neq \imath P_j$ if $i \neq j$, where $\imath$ stands for the hyperelliptic involution $(x, y) \mapsto (x, -y)$. The integer $w = \deg u$ is called the weight of the divisor, which is reduced if and only if $w \leq g$.

With this setting we see that if the polynomial $u$ splits over $\mathbb{F}_q$, then for each $i$ the point $P_i$ is $\mathbb{F}_q$-rational so that the class of $(P_i) - (P_\infty)$ defines an element of $\mathrm{Jac}_{\mathcal{H}}(\mathbb{F}_q)$. Furthermore, we have $D = \sum_i ((P_i) - (P_\infty))$ (i.e. $D$ is 1-smooth) and the same equality holds when taking linear equivalence classes. Gaudry's algorithm [4] stems from this observation, and its harvesting phase can be summarized as follows.

- The factor base $\mathcal{B}$ is the set $\{(P) - (P_\infty) : P \in \mathcal{H}(\mathbb{F}_q)\}$, or rather a set of representatives of its quotient by the hyperelliptic involution, accounting for the trivial relations $(\imath P) - (P_\infty) \sim -((P) - (P_\infty))$. It contains $\Theta(q)$ elements.
- At each step, we compute (using a semi-random walk) the Mumford representation $[u, v]$ of a reduced divisor $D \sim aD_0 + bD_1$, where $D_0$ and $D_1$ are the entries of the DLP challenge.
- If $u$ splits over $\mathbb{F}_q$ as $\prod_i (x - x_i)$ then a relation is found since we have $aD_0 + bD_1 \sim \sum_i ((P_i) - (P_\infty))$ where $P_i = (x_i, v(x_i))$.

We see that each step of the harvesting phase requires a few operations in $\mathrm{Jac}_{\mathcal{H}}$ followed by the factorization of $u$, which is generically a degree $g$ polynomial. The probability that $u$ actually splits over $\mathbb{F}_q$ is about $1/g!$, so we need about $g!$ trials before finding a relation.

A precise analysis of the complexity is given by the author in [4]. However, asymptotically when $g$ is fixed and $q$ grows to $+\infty$, the cost of finding one relation is in $\tilde{O}(1)$, so the complexity of the harvesting phase is in $\tilde{O}(q)$. By contrast, the linear algebra phase costs $\tilde{O}(q^2)$ and dominates the complexity. Balancing the

two phases has first been proposed by Harley and improved by Thériault [11], who introduced the large prime variants in this context. Asymptotically, the best result[5] is obtained by Gaudry, Thériault, Thomé and Diem [6] using the double large prime variation with a factor base $\mathcal{B}_s$ of size $\approx q^{1-1/g}$, yielding a complexity of $\tilde{O}(q^{2-2/g})$ for both the harvesting and the linear algebra phase. Giving a detailed description of the double large variant is outside the scope of this paper, but we mention that there are two distinct approaches regarding the construction of the small prime factor base. A first possibility is to define directly $\mathcal{B}_s$ as a subset of $\mathcal{B}$ whose elements satisfy an easy to check condition, so that membership testing and enumeration are very fast; alternately, the set $\mathcal{B}_S$ can be constructed progressively from the first relations in order to simplify the future elimination of large primes (as done in by Laine and Lauter [8] in the non-hyperelliptic case).

### 2.2 Sarkar and Singh's Sieve

A recent result of Sarkar and Singh [10] proposes a sieving approach to the relation search for hyperelliptic curves. In this method, with the same factor base $\mathcal{B}$ as in Sect. 2.1, we start from a weight $g$ reduced divisor written as $D = [u, v] = \sum_{i=1}^{g}(P_i) - g(P_\infty)$, usually related to the challenge. We then consider all the weight $g+1$ semi-reduced divisors $D' = [u', v']$ that are linearly equivalent to $-D$; a relation is obtained each time $u'$ is split (the factor base $\mathcal{B}$ is the same as in the previous version). The set of all the decompositions of $-D$ as

$$-D \sim \sum_{i=1}^{g+1}(Q_i) - (g+1)(P_\infty),$$

i.e. the set of all weight $g+1$ semi-reduced divisors linearly equivalent to $-D$, is in one-to-one correspondence with the set of divisors of functions in the Riemann-Roch space $\mathcal{L}(-D + (g+1)(P_\infty)) = \mathcal{L}(-\sum_{i=1}^{g}(P_i) + (2g+1)(P_\infty))$. This space is equal to $Span(u(x), y - v(x))$ (since functions in this space have poles at $P_\infty$ only, of order at most $2g+1$, and vanish at the support of $D$), and thus the decompositions of $-D$ can be parametrized by an element $\lambda \in \mathbb{F}_q$.

We begin with the non-large-prime, non-sieving version of the algorithm. The relation search consists of two main loops, the outer one being simply a semi-random walk iterating through reduced divisors $D = [u, v] \sim aD_0 + bD_1$. The inner loop iterates over the value of the parameter $\lambda \in \mathbb{F}_q$. For each $\lambda$, we consider the function $f_\lambda = y - v(x) + \lambda u(x)$ and the corresponding semi-reduced divisor $D_\lambda = -D + \mathrm{div}(f_\lambda)$. The Mumford representation $[u_\lambda, v_\lambda]$ of $D_\lambda$ is given by the formulae

$$\begin{cases} u_\lambda = c\frac{(\lambda u - v)^2 - h}{u} = c(\lambda^2 u - 2\lambda v + \frac{v^2 - h}{u}) \\ v_\lambda = v - \lambda u \mod u_\lambda \end{cases},$$

[5] This is only true asymptotically. For actual instances of the DLP many other factors have to be taken into account, and large prime variations are not always appropriate.

where $c \in \mathbb{F}_q$ is the constant that makes $u_\lambda$ monic. We obtain a relation each time $D_\lambda$ is 1-smooth, i.e. when $u_\lambda$ is split over $\mathbb{F}_q$; this happens heuristically with probability $1/(g+1)!$.

The main advantage of this relation search is that it admits a sieving version, in the spirit of [7]. The idea is to replace the inner loop in $\lambda$ by an inner loop in $x \in \mathbb{F}_q$.

For each value of $x$, we compute the expression

$$S(x,\lambda) = \lambda^2 u(x) - 2\lambda v(x) + \frac{v(x)^2 - h(x)}{u(x)},$$

which becomes a quadratic polynomial in $\lambda$, and find the corresponding roots (for simplicity we can skip the values of $x$ for which $u(x) = 0$). There are two distinct roots $\lambda_0$ and $\lambda_1$ if and only if $h(x)$ is a square in $\mathbb{F}_q$, and those roots are given by:

$$\lambda_0 = \frac{v(x) + h(x)^{1/2}}{u(x)},\ \lambda_1 = \frac{v(x) - h(x)^{1/2}}{u(x)}.$$

As explained by the authors, this step is very fast if a table containing a square root of $h(x)$ (if it exists) for each $x \in \mathbb{F}_q$ has been precomputed. We then store the corresponding couples $(\lambda_0, x)$ and $(\lambda_1, x)$. At the end of the inner loop, we look for the values of $\lambda$ that have appeared $g+1$ times: this means that the corresponding polynomial $u_\lambda$ has $g+1$ distinct roots, so that $D_\lambda$ yields a relation, i.e. a decomposition of $-D$. In practice, we can either store each value of $x$ in an array $L$ of lists indexed by $\lambda$; each time a value of $\lambda$ is obtained as a root of the quadratic expression, we append $x$ to $L[\lambda]$. When $\#L[\lambda] = g+1$, we directly have the $x$-coordinates of the points in the support of $\mathrm{div}(f_\lambda) - D$, and a last step is then to compute back the $y$-coordinates using $f_\lambda$. Alternatively, we can simply maintain a counter array `Ctr` indexed by $\lambda$ and increment `Ctr`$[\lambda]$ each time $\lambda$ is obtained as a root. When this counter reaches $g+1$, we factorize the corresponding split polynomial $u_\lambda$. This variant has the merit of saving memory at the expense of some duplicate computations, but is more interesting when $g$ increases since the proportion of $\lambda$'s yielding a relation becomes small.

The main speed-up is provided by the fact that at each iteration of the inner loop, we replace the splitting test and the eventual factorization of either the degree $g$ polynomial $u$ (in Gaudry's version) or the degree $g+1$ polynomial $S(\lambda, x)$ evaluated in $\lambda$, by the resolution of the degree 2 equation $S(\lambda, x) = 0$, evaluated in $x$. This comes at the expense of a slightly lower decomposition probability, namely $1/(g+1)!$ instead of $1/g!$, and higher memory requirement.

As already noticed in [7], a second advantage of this sieve is its compatibility with the double large prime variation. Indeed, once the "small prime" factor base $\mathcal{B}_s$ is constructed, it is sufficient to sieve among the values of $x \in \mathbb{F}_q$ corresponding to abscissae of its elements (the full sieving as described above can still be used in the construction steps of $\mathcal{B}_s$ if necessary). Since the cardinality of $\mathcal{B}_s$ is in $\Theta(q^\alpha)$ with $\alpha = 1 - 1/g$, this shortened sieving only costs $\tilde{O}(q^\alpha)$ instead of $\tilde{O}(q)$. We then look for the values of $\lambda$ that have been obtained at least $g-1$ times. The corresponding polynomials $u_\lambda$ have at least $g-1$ roots corresponding

to small primes, and it just remains to test if it is indeed split, which happens with heuristic probability $1/2$ (in the case where $\lambda$ has been obtained exactly $g-1$ times). Note that we cannot simply scan the array $L$ or `Ctr`, as it would cost $\tilde{O}(q)$ (even with a very small hidden constant) and defeat our purpose. So additional care must be taken in the implementation in order to recover the interesting values of $\lambda$ in only $\tilde{O}(q^{\alpha})$, for instance using associative arrays, see [10] for details. Although it is not specified in the original paper, one can show that the asymptotic complexity of this variant is still in $\tilde{O}(q^{2-2/g})$ for fixed $g$, as in the work of Gaudry, Thomé, Thériault and Diem [6], but it is more efficient in practice, and the authors report a significant speed-up.

### 2.3 Sarkar and Singh's Sieve Revisited

As mentioned above, precomputing a table containing an eventual square root of $h(x)$ for each $x \in \mathbb{F}_q$ can significantly speed up the sieving phase (for a $\tilde{O}(q)$ overhead). But this table is actually nothing more than a list of the rational points of $\mathcal{H}$. Indeed, if $y$ is a square root of $h(x)$ then $(x, y)$ and $(x, -y)$ are exactly the two points in $\mathcal{H}(\mathbb{F}_q)$ with abscissa $x$, and this precomputation is actually performed when the factor base $\mathcal{B} = \{(P) - (P_\infty) : P \in \mathcal{H}(\mathbb{F}_q)\}$ is enumerated.

This means that we can modify Sarkar and Singh's sieve as follows. Recall that we are looking for functions $f_\lambda = y - v(x) - \lambda u(x)$ such that $-D + \mathrm{div}(f_\lambda)$ is 1-smooth. Instead of sieving over the value of $x \in \mathbb{F}_q$, or in a small subset corresponding to small primes, we directly sieve over $P = (x_P, y_P) \in \mathcal{B}$ or $\mathcal{B}_s$, and the corresponding value of $\lambda$ is simply recovered as $\frac{y_P - v(x_P)}{u(x_P)}$. We give a pseudo-code of this sieve in Alg. 1.

This pseudo-code corresponds to the non-large-prime version. Details like the management of the list or associative array $L$ and the update of $M$ are omitted. As mentioned above, a simple counter array `Ctr` can be used instead of $L$, requiring the factorization of $S(x, \lambda)$ for the update of $M$. If the double large prime variation is used, then the first inner loop iterates only over the elements of the small factor base $\mathcal{B}_s$, and in the second we test if $\#L[\lambda] \geq g - 1$ and subsequently if the remaining factor of $S(x, \lambda)$ splits.

An easy improvement, not included in the pseudo-code for the sake of clarity, is to use the action of the hyperelliptic involution to divide by two the size of the factor base. We can then compute simultaneously the values of $\lambda$ corresponding to $P = (x_P, y_P)$ and $\imath P = (x_P, -y_P)$. This saves one evaluation of $u$ and of $v$ at $x_P$, and one inversion of $u(x_P)$, although it is also possible to precompute all inverses. It is clear that this is basically a rewriting of Sarkar and Singh's original sieve, so that the performances of both should be similar. However, we will now see that it is easier to adapt to the non-hyperelliptic case.

**Algorithm 1** Sieving in the hyperelliptic case

**Input:** the set of rational points $\mathcal{B}$ of $\mathcal{H}$.
**Output:** the relation matrix $M$.

```
n_rel = 0.
repeat
  Choose a random reduced divisor D = [u, v] ~ aD_0 + bD_1.
  Initialize an array of lists L.
  for P = (x_P, y_P) ∈ B do
    Compute u(x_P) and v(x_P).
    if u(x_P) ≠ 0 then
      Compute λ = (y_P − v(x_P))/u(x_P).
      Append P to L[λ].
    end if
  end for
  for λ ∈ F_q do
    if #(L[λ]) = g + 1 then
      Update M.
      Increment n_rel.
    end if
  end for
until n_rel > #B
return the matrix M.
```

## 3 Sieving for Small Degree Curves

### 3.1 Diem's Relation Search

Gaudry's algorithm can be adapted to non-hyperelliptic curves. Most divisors can still be represented by Mumford coordinates, but computations in the Jacobian variety are not as tractable and checking for 1-smoothness is less obvious. However, all these operations are in $\tilde{O}(1)$ when $g$ is fixed, so that the asymptotic complexity is still in $\tilde{O}(q^{2-2/g})$, albeit with a larger hidden constant than in the hyperelliptic case.

In [2], Diem devised a different harvesting technique for plane curves whose degree is really close to the genus. More precisely, if a curve of genus $g \geq 3$ is general enough (which rules out hyperelliptic curves), we can find in polynomial time a plane model of expected degree $d \leq g+1$ using a probabilistic algorithm. This means that Diem's algorithm applies to almost all non-hyperelliptic curves, with the (then unexpected) consequence that the DLP is easier on non-hyperelliptic curves than on hyperelliptic ones.

Harvesting is done by considering relations coming from principal divisors corresponding to equations of lines. For any couple of rational points $P_1$ and $P_2$ on the curve $\mathcal{C}$, we consider the affine function $f = ax+by+c \in \mathbb{F}_q(\mathcal{C})$ such that the equation of the line $L$ passing through $P_1$ and $P_2$ is $f = 0$. The intersection of $L$ and the affine part of $\mathcal{C}$ contains up to $d$ rational points, of which we already know two; determining the remaining $d-2$ amounts to finding the roots of a degree $d-2$ polynomial. If there are exactly $d-2$ other rational intersection

points $P_3, \ldots, P_d$ then we obtain a relation of the form

$$\mathrm{div}(f) = (P_1) + \cdots + (P_d) - D_\infty \sim 0,$$

where $D_\infty$ is the divisor corresponding to the intersection of $\mathcal{C}$ with the line at infinity. This happens with probability $1/(d-2)!$, which is better than the $1/g!$ probability for hyperelliptic curves as soon as $d \leq g+1$.

We can summarize Diem's harvesting technique as follows. The curve $\mathcal{C}$ is defined by the equation $F(x,y) = 0$, and we denote by $\mathcal{C}_0$ its affine, non-singular part. We no longer consider only (classes of) degree 0 divisors, so technically we are working in the full divisor class group and not only its degree 0 part, but in practice it makes no difference.

- The factor base is $\mathcal{B} = \{(P) : P \in \mathcal{C}_0(\mathbb{F}_q)\} \cup \{D_\infty\}$.
- We choose two points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ in $\mathcal{B}$ such that $x_1 \neq x_2$ (for simplicity) and compute $\lambda = (y_2 - y_1)/(x_2 - x_1)$ and $\mu = y_1 - \lambda x_1$.
- We test if the degree $d-2$ polynomial $\frac{F(x,\lambda x+\mu)}{(x-x_1)(x-x_2)}$ splits over $\mathbb{F}_q$. If it is the case, we compute its roots $x_3, \ldots, x_d$ and the associated $y$-coordinates $y_3 = \lambda x_3 + \mu, \ldots$, and we store the relation

$$(P_1) + (P_2) + (P_3) + \cdots + (P_d) - D_\infty \sim 0$$

  where $P_i = (x_i, y_i)$, provided these points are non-singular.
- We go back to the second step until enough relations are found.

Note that a descent phase is needed to express the entries of the DLP challenge in terms of elements of the factor base, see [2].

The whole routine is particularly well-suited to a two large prime variation (see also [8] for another version differing mainly on the construction of the factor base and the large prime graph). Instead of selecting two points in $\mathcal{B}$, we pick them in the small factor base $\mathcal{B}_s$ and keep only relations involving at most two large primes. The main advantage (as compared to the hyperelliptic case) is that we ensure in this way that each potential relation contains already two small primes; this greatly increases the probability of finding relations with only two large primes. In particular if $d = 4$, every relation found by the above method automatically satisfies the two large prime condition.

A precise complexity analysis is given in [2] and, if $\mathcal{C}$ admits a plane model of degree $g+1$, gives an asymptotic running time of $\tilde{O}(q^{2-2/(d-2)})$, for a small factor base $\mathcal{B}_s$ of size $\Theta(q^{1-1/(d-2)})$. If $d = g+1$ this is $\tilde{O}(q^{2-2/(g-1)})$, which improves over the $\tilde{O}(q^{2-2/g})$ complexity of the hyperelliptic case. Note however that the size of the small factor base is such that in order to find enough relations, almost all lines going through pairs of points of $\mathcal{B}_s$ have to be considered. This is troublesome because each line, and thus each relation, can be obtained several times, namely $n(n-1)/2$ times if it contains $n$ small factor base points. This is not really an issue if $d = 4$, but for higher $d$ some extra care has to be taken in order to prevent duplicate relations.

### 3.2 The Sieving Technique

We can easily adapt our sieving formulation to Diem's setting. The factor base remains the same set of points. Basically, in a first loop we iterate over points $P_1 = (x_1, y_1) \in \mathcal{C}_0(\mathbb{F}_q)$. The equation of a non-vertical line passing through $P_1$ is given by $(y - y_1) - \lambda(x - x_1) = 0$. The task is now to find the values of $\lambda$ such that the line has $d$ rational points of intersection with $\mathcal{C}$ without checking for smoothness. For this we then loop in $P_2 = (x_2, y_2) \in \mathcal{C}_0(\mathbb{F}_q)$ and compute the corresponding $\lambda = (y_2 - y_1)/(x_2 - x_1)$. But instead of looking for the intersection of the line with $\mathcal{C}$, we just append $P_2$ to the list $L[\lambda]$, where $L$ is an array of lists; alternatively, we can simply increment a counter $\texttt{Ctr}[\lambda]$. If this counter reaches $d - 1$, or if $L[\lambda]$ contains $d - 1$ elements, we know that the line contains enough points and yields a relation. This is made precise in the pseudo-code of Alg. 2.

**Algorithm 2** Sieving for small degree curves

**Input:** the list of rational non-singular affine points $\mathcal{B} = \mathcal{C}_0(\mathbb{F}_q)$.
**Output:** the relation matrix $M$.

```
n_rel = 0.
for i = 1 to #B do
  (x_1, y_1) ← B[i]
  Initialize an array of lists L.
  for j = i + 1 to #B do
    (x_2, y_2) ← B[j].
    if x_2 ≠ x_1 then
      Compute λ = (y_2 − y_1)/(x_2 − x_1).
      Append (x_2, y_2) to L[λ].
    end if
  end for
  for λ ∈ F_q do
    if #L[λ] = d − 1 then
      Update M.
      Increment n_rel.
    end if
    if n_rel > #B then
      return the matrix M.
    end if
  end for
end for
```

Note that in the inner loop we do not iterate over the elements of $\mathcal{B}$ that have already been considered in the outer loop. Indeed, after an iteration of the outer loop all the lines passing through the given point $P_1 = \mathcal{B}[i]$ have been surveyed, so there is no reason to scan this point again. In this way no line can be considered twice, and we avoid completely having to check for duplicate relations.

In Diem's version, each step requires the computation and factorization of $\frac{F(x,\lambda x+\mu)}{(x-x_1)(x-x_2)}$. The probability of finding a relation is $1/(d-2)!$, so that after $q$ steps about $q/(d-2)!$ relations are harvested. By comparison, in our sieving each step requires a single division (or multiplication if the inverses are pretabulated). The inner loop ends after about $\#\mathcal{B} \approx q$ steps, and yields $q/(d-1)!$ relations approximately: all the lines through $P_1 = (x_1, y_1)$ have been explored, and contain $d-1$ other points with probability $1/(d-1)!$. Thus we need $d-1$ times as many steps to obtain the same number of relations, but each step is much simpler, and the experiments of the next section confirm the important speed-up.

This sieve can be adapted straightforwardly to the double large prime variation : we just have to restrict both loops to the small factor base $\mathcal{B}_s$ (once it is constructed, if the version of [8] is followed), then we recover the values of $\lambda$ such that $\#L[\lambda] \geq d-3$. When $\#L[\lambda] = d-3$, we still have to check if the remaining two points on the line are rational, which amounts to factorizing a degree 2 polynomial. If $d = 4$, in Diem's version there are at most two remaining points on any line anyway; our new sieve is thus basically equivalent and does not provide a significant speed-up when using double large primes. However as soon as $d \geq 5$ it outperforms Diem's version, but the asymptotic complexity remains in $\tilde{O}(q^{2-2/(d-2)})$.

### 3.3 Sieving with Singularities

An article of Diem and Kochinke [3] tries to improve on the asymptotic complexity of the above method. The basic idea is to consider singular small degree plane models, and use a singular point as one of the points defining the lines cutting out the curve. Indeed, a singular point appears with a multiplicity greater than one in any line passing through it, so that there are fewer remaining points of intersection with $\mathcal{C}$, and the degree of the polynomial to test for smoothness is less than when two regular points are used. Unfortunately in general there are not enough singular points on a given planar curve to obtain sufficiently many relations. Thus an important part of Diem and Kochinke's work is to find a way to compute new singular plane models of degree $d \leq g+1$ for a given genus $g$ curve, but this is outside of the scope of the present article; furthermore, the computation of the maps between the different models is not asymptotically relevant. Using Brill-Noether theory and considerations on special linear systems, they show that this method works for "general enough" non-hyperelliptic curves, of genus $g \geq 5$.

So we assume that we are given a degree $d$ curve $\mathcal{C}$, of equation $F(x, y) = 0$, with a rational singular point $P_1$ of multiplicity $m \geq 2$ (in most cases $m = 2$). The factor base is given by the rational points of the desingularization $\tilde{\mathcal{C}}$ of $\mathcal{C}$, i.e. $\mathcal{B} = \{(P) : P \in \tilde{\mathcal{C}}(\mathbb{F}_q)\}$. In the original version, for each other point $P_2$ in $\mathcal{B}$ or in the small factor base $\mathcal{B}_s$, the intersection of $\mathcal{C}$ with the line passing through $P_1$ and $P_2$ is computed as before. This amounts to finding the roots of

the polynomial

$$\frac{F(x, \lambda x + \mu)}{(x - x_1)^m (x - x_2)},$$

which has degree $d - m - 1$. If it splits, which happens with probability about $1/(d - m - 1)!$, we compute the intersection points $P_3, \ldots, P_{d-m-1}$ and obtain a relation that we can write as

$$D \sim (P_2) + (P_3) + \cdots + (P_{d-m-1}),$$

where $D$ involves the singularity and the points at infinity. In the double large prime variation we keep this relation only if it involves no more than two large primes. To get rid of the divisor $D$ on the left-hand side we would like to subtract one such relation from all the other ones. But in order to do this (using large primes) we need one relation involving only small primes ; if it does not exist a solution is then to add some points to $\mathcal{B}_s$. Since there are less points on the right-hand side than in Diem's first algorithm, the probability of finding a relation increases, and one can show that the overall complexity becomes $\tilde{O}(q^{2-2/(g-2)})$. Note that here again, some care must be taken to avoid duplicate relations, and in particular not all points $P_2$ but only a fraction of the factor base should be considered.

Now it is clear that our sieve can be naturally adapted to this new setting. Indeed, we can keep the inner loops of Alg. 2 ; the point $(x_1, y_1)$ is now the singular point $P_1$, and we look for the values of $\lambda$ that have been obtained $d-m$ times, or $d - m - 2$ times in the double large prime variation. Once again, this replaces the factorization of a degree $d - m - 1$ polynomial by a single division, and avoids checking for duplicate relations.

## 4 Experiments

We have experimented the harvesting techniques presented in this article for several curves of different genera, defined over different finite fields. All computations have been done using the computer algebra system Magma [1] on an AMD Opteron™ 6176 SE@2.3GHz processor. We only implemented the non-large-prime version of the algorithms, the main reason being that we wanted the tests to be as simple as possible[6]. The curves have been generated with the command `RandomCurveByGenus`, which always returned a degree $g$ curve (instead of $g + 1$) for $g \geq 6$; for this reason the results in genus 6 are very close to those in genus 5 and we did not report them. For the non-sieving versions, we used associative arrays and sets to automate the check for duplicate relations, but this is more and more costly as the number of relations grows.

We give in Table 1 the comparison between Diem's method and our sieve; the values are the timings in seconds (on an Intel© Core i5@2.00Ghz processor) to obtain $p \approx \#\mathcal{F}$ relations, averaged over several curves.

[6] More fundamentally, large prime variations are interesting for the asymptotic complexity analysis, but are not always well-suited in practice ; other methods such as the Gaussian structured elimination [9] can be more efficient.

**Table 1.** Comparisons of the new sieve with Diem's classical method

| $p$ | | 78137 | 177167 | 823547 | 1594331 |
|---|---|---|---|---|---|
| Genus 3, degree 4 | Diem | 11.57 | 27.54 | 135.1 | 266.1 |
| | Diem + sieving | 3.65 | 9.38 | 46.96 | 94.60 |
| | Ratio | 3.16 | 2.95 | 2.88 | 2.81 |
| Genus 4, degree 5 | Diem | 51.85 | 122.4 | 595.8 | 1174 |
| | Diem + sieving | 15.58 | 40.01 | 195.1 | 387.6 |
| | Ratio | 3.33 | 3.06 | 3.05 | 3.03 |
| Genus 5, degree 6 | Diem | 229.4 | 535.8 | 2581 | 5062 |
| | Diem + sieving | 75.66 | 199.0 | 969.3 | 1909 |
| | Ratio | 3.03 | 2.69 | 2.66 | 2.65 |
| Genus 7, degree 7 | Diem | 1382 | 3173 | 14990 | 29280 |
| | Diem + sieving | 458.5 | 1199 | 5859 | 11510 |
| | Ratio | 3.02 | 2.65 | 2.56 | 2.54 |

In Table 2 we give timings comparing the new sieve with Diem and Kochinke's method. We did not implement the change of plane models; instead, we simply chose random curves possessing rational singular points, and used one of them as the base point for the relation search. In the sieving version all the relations involving lines passing through the singularity were computed, whereas in the non-sieving case we only iterated through half of the basis, as suggested in [3]. For this reason the values correspond to the timings in seconds needed to obtain 1000 relations, again averaged over several curves.

**Table 2.** Comparisons of the new sieve with Diem and Kochinke's method

| $p$ | | 78137 | 177167 | 823547 | 1594331 |
|---|---|---|---|---|---|
| Genus 5, degree 6 | Diem & Kochinke | 1.58 | 1.60 | 1.69 | 1.76 |
| | DK + sieving | 0.43 | 0.45 | 0.52 | 0.61 |
| | Ratio | 3.67 | 3.60 | 3.23 | 2.90 |
| Genus 7, degree 7 | Diem & Kochinke | 8.59 | 8.68 | 8.97 | 9.20 |
| | DK + sieving | 1.21 | 1.25 | 1.56 | 1.93 |
| | Ratio | 7.13 | 6.96 | 5.74 | 4.77 |

## 5 Conclusion

We have shown in this work that a reformulation of Sarkar and Singh's sieve [10], namely sieving over points instead of $x$-coordinates, gives a simpler presentation of the harvesting phase of the index calculus algorithm on hyperelliptic curves. More importantly, it can be naturally adapted to Diem and Kochinke's index calculus for non-hyperelliptic curves [2, 3]. Our experiments show that the new sieve clearly outperforms the relation search of the other methods in all circumstances and should always be preferred.

*Acknowlegdements.* We would like to thank the anonymous referees for their useful comments during the elaboration of the article.

## References

1. W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).
2. C. Diem. An index calculus algorithm for plane curves of small degree. In *Algorithmic number theory*, volume 4076 of *Lecture Notes in Comput. Sci.*, pages 543–557. Springer, 2006
3. C. Diem and S. Kochinke. Computing discrete logarithms with special linear systems. Preprint, available at http://www.math.uni-leipzig.de/ diem/preprints/dlp-linear-systems.pdf, 2013.
4. P. Gaudry. An algorithm for solving the discrete log problem on hyperelliptic curves. In *Advances in cryptology—EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Comput. Sci.*, pages 19–34. Springer, 2000.
5. P. Gaudry, F. Hess, and N.P. Smart. Constructive and destructive facets of Weil descent on elliptic curves. *J. Cryptology*, 15(1):19–46, 2002.
6. P. Gaudry, E. Thomé, N. Thériault, and C. Diem. A double large prime variation for small genus hyperelliptic index calculus. *Math. Comput.*, 76(257):475–492, 2007.
7. A. Joux and V. Vitse. Cover and decomposition index calculus on elliptic curves made practical: application to a previously unreachable curve over $\mathbb{F}_{q^6}$. In *Advances in cryptology—EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Comput. Sci.*, pages 9–26. Springer, 2012.
8. K. Laine and K. Lauter. Time-memory trade-offs for index calculus in genus 3. To appear in *J. Math. Crypto.*
9. B. A. LaMacchia and A. M. Odlyzko. Computation of discrete logarithms in prime fields. *Des. Codes Cryptogr.*, 1(1):47–62, 1991.
10. P. Sarkar and S. Singh. A new method for decomposition in the Jacobian of small genus hyperelliptic curves. Cryptology ePrint Archive, Report 2014/815, 2014.
11. N. Thériault. Index calculus attack for hyperelliptic curves of small genus. In *Advances in Cryptology – ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Comput. Sci.*, pages 75–92. Springer, 2003.