A New Method for Decomposition in the Jacobian of Small Genus Hyperelliptic Curves

Palash Sarkar and Shashank Singh

Indian Statistical Institute
Kolkata, India
palash@isical.ac.in, sha2nk.singh@gmail.com

Abstract. Decomposing a divisor over a suitable factor basis in the Jacobian of a hyperelliptic curve is a crucial step in an index calculus algorithm for the discrete log problem in the Jacobian. For small genus curves, in the year 2000, Gaudry had proposed a suitable factor basis and a decomposition method. In this work, we provide a new method for decomposition over the same factor basis. The advantage of the new method is that it admits a sieving technique which removes smoothness checking of polynomials required in Gaudry's method. Also, the total number of additions in the Jacobian required by the new method is less than that required by Gaudry's method. The new method itself is quite simple and we present some example decompositions and timing results of our implementation of the method using Magma.

1 Introduction

Elliptic curve cryptography was independently introduced by Koblitz [11] and Miller [14] and was soon followed by hyperelliptic curve (HEC) cryptography which was introduced by Koblitz [12]. For hyperelliptic curves, cryptography is carried out in a suitable large subgroup of the Jacobian. A fundamental assumption required for HEC based cryptography to be secure is that the discrete logarithm problem (DLP) in the Jacobian should be computationally hard.

In the last few decades, the main approach to tackling DLP in different cryptographic groups has been the use of index calculus algorithms. The structure of such algorithms identifies a factor base which consists of a small subset of elements of the group and a method to decompose elements of the group over the factor base. Any such decomposition gives rise to a relation among the element that is decomposed and some of the elements of the factor base. After obtaining sufficiently many relations, sparse linear algebra techniques are used to recover the discrete logs of the factor base elements. From this, it is possible to obtain the discrete log of a target element either directly or by a further computation. The main task in designing an index calculus algorithm is to identify a factor basis and a method for decomposition over the factor basis and if applicable, a method to calculate the discrete log of the target element.

The first index calculus algorithm for hyperelliptic curves was introduced in [1]. For large genus curve, this provided a sub-exponential time algorithm for

DLP in the Jacobian of such curves. Later developments along this line have been reported in [5, 3, 4, 18].

The index calculus algorithm also works for small genus curves but, the running time is no longer sub-exponential. Even so, it can be better than the running time of the Pollard's rho algorithm. This was described by Gaudry in [7]. The work is important and provided a practical DLP algorithm for small genus curves. It was used to solve some HEC-DLP challenges. Subsequently, variants of this algorithm called the large prime variant [16] and the double large prime variant [9] were introduced to improve the asymptotic efficiency.

Suppose $C : y^2 = f(x)$ is a HEC where $f(x)$ is a polynomial in $\mathbb{F}_q[x]$. The notation $J_C(\mathbb{F}_q)$ denotes the Jacobian of C restricted to the set of divisors which have representations in $\mathbb{F}_q$. Gaudry's algorithm works over $J_C(\mathbb{F}_q)$ by defining a factor basis and a method of decomposing a divisor over the factor basis. Nagao [15] proposed a decomposition method for curves C defined over $\mathbb{F}_{q^n}$ where $n \geq 2$. Subsequently, Joux and Vitse [10] proposed a modification of Nagao's method for such curves and combined this method with the Weil descent method [6, 8, 2] to solve DLP for certain elliptic curves defined over $\mathbb{F}_{p^6}$ for p to be a 25-bit prime.

Our contributions: We consider small genus hyperelliptic curves. For such curves, Gaudry's method [7] is presently the only known practical algorithm for solving the DLP on the Jacobian of such curves. This method has three parts: identification of a factor basis, a method for decomposition and the linear algebra step.

In this work, we provide a new method for decomposition. The factor basis remains the same as in Gaudry's method and also the linear algebra step remains unchanged. Our main contribution is to show an alternative method for decomposing divisors over the factor basis proposed by Gaudry. The advantage of the alternative method is that it allows to apply a sieving technique to eliminate the smoothness checking of polynomials required in Gaudry's method of decomposition. The sieving technique that we use is based on the technique used by Joux and Vitse [10] for quadratic extension fields. On the flip side, a relative disadvantage of the new method is that a decomposition is obtained in about $(g+1)!$ trials whereas in Gaudry's method a decomposition is obtained in about $g!$ trials.

We have implemented the new decomposition method in Magma and report some experimental results and timings. Comparison of timings to Gaudry's method is currently ongoing work. Initial results, however, indicate that the new method compares favourably to Gaudry's method.

2 Notation

We mention a few notation on hyperelliptic curves. Basic description on hyperelliptic curves can be found in [13]. We will consider the underlying field K to be of characteristic greater than 2 and so a curve C will be given by the equation

$C : y^2 = f(x)$, where $f(x)$ is in $K[x]$. If L is an extension of K, then the set of all L-rational points of C will be denoted as $C(L)$. The notation J_C denotes the Jacobian of C and $J_L(C)$ denotes the subset of $J(C)$ consisting of all divisors of C which have a representation over L. Reduced divisors having representations in $J_L(C)$ will be denoted by their Mumford representation, i.e., a divisor D will be written as $D = (u(x), v(x))$ where $u(x), v(x) \in L[x]$, $\deg(v) < \deg(v) \leq g$ and $u(x)$ divides $v^2(x) - f(x)$.

3 Index Calculus Algorithms

Let $G = \langle g \rangle$ be a cyclic group and h be an element of G. The DLP in G is to compute i such that $h = g^i$. An index calculus algorithm to solve DLP in G has several steps. The first step is to identify a factor basis which is a subset of G. The second step consists of identifying relations between the elements of the factor basis. These relations can be converted into linear relations between the discrete logs of the elements of the factor basis. The system of linear equations which arises is exteremly sparse and is solved using either the Lanczos or the block Wiedemann algorithm. This step provides the discrete logs of the elements of the factor basis. The third step is to decompose the target element over the factor basis. With the discrete logs of factor basis elements already known, the discrete log of the target element is obtained. For some algorithms, the relation collection step itself ensures that relations are obtained between the target element and the elements of the factor basis. In such cases, the discrete log of the target element is obtained immediately after the linear algebra step.

For hyperelliptic curves, the DLP problem is defined over $J_C(K)$. Elements of the Jacobian are divisors. To implement index calculus algorithms over $J_C(K)$ two things are required. First, one has to identify a set of divisors as the factor basis and second, it is required to have a method to decompose a divisor in $J_C(K)$ over the factor basis. Each such decomposition provides a relation among the divisors in the factor basis. Note that it is not required to be able to decompose every divisor in $J_C(K)$ over the factor basis. It is only required to generate enough relations so that it is possible to obtain the discrete logs of the elements in the factor basis.

In the next few subsections, we briefly recall some of the developments in index calculus algorithms for hyperelliptic curves which are relevant to our work.

3.1 Adleman-DeMarrais-Huang Decomposition [1]

The general description of the algorithm is in terms of function fields. Here we provide a description that applies to the Jacobian of a hyperelliptic curve.

Let $\mathbb{F}_q$ be a finite field of characteristic greater than 2 and assume that a curve C of genus g is given by an equation of the form $y^2 = f(x)$ where the degree of $f(x)$ is $2g + 1$. For a smoothness bound S, a divisor over $\mathbb{F}_q$ is said to be S-smooth if all its points are defined over an extension $\mathbb{F}_{q^k}$ with $k \leq S$.

Equivalently, $D = \mathrm{div}(u(x), v(x))$ is S-smooth if and only if the irreducible factors of $u(x)$ have degrees at most equal to S.

A divisor $\mathrm{div}(u(x), v(x))$ is said to be prime if $u(x)$ is irreducible. Further, a divisor $\mathrm{div}(u(x), v(x))$ is equal to $\sum \mathrm{div}(u_i(x), v_i(x))$ where $u_i(x)$ are the irreducible factors of $u(x)$ and for suitable polynomials $v_i(x)$. For a smoothness bound S, the factor basis is defined to be the following set:

$$\mathcal{FB} = \{D \in J_C(K) : D \text{ is prime and is of degree at most } S\}. \tag{1}$$

Relations among the elements of the factor basis are obtained in the following manner. Consider a polynomial function $G(x, y) = \lambda(x) + y\mu(x)$ in $K[C]$. Since G is a rational function, $\mathrm{div}(G)$ is (a representative of) the additive identity of the Jacobian of the curve and we write $\mathrm{div}(G) = 0$. The degree of $G(x, y)$ is the degree of its norm which is $N(G) = \lambda^2(x) - f(x)\mu^2(x)$. If $N(G)$ is S-smooth, then a decomposition is obtained.

In the above approach, the check on $N(G)$ to be S-smooth can be avoided using a sieving technique due to Flassenberg and Paulus [5]. This is based on the following fact: a polynomial $\rho(x)$ divides $\lambda^2(x) - f(x)\mu^2(x)$ if and only if $\rho(x)$ divides $\lambda_1^2(x) - f(x)\mu_1^2(x)$ where $\lambda_1(x) = \lambda(x) + r(x)\rho(x)$ and $\mu_1(x) = \mu(x) + s(x)\rho(x)$ for any polynomials $r(x)$ and $s(x)$. This was originally proposed over odd characteristic fields. A later work [18] modified it to work over even characteristic fields.

3.2 Gaudry's Decomposition [7]

Gaudry provided a method for decomposition in the Jacobian $J_C(\mathbb{F}_q)$ of a curve $C : y^2 = f(x)$, with $f(x) \in \mathbb{F}_q[x]$. The factor basis is defined to be of the following form:

$$\mathcal{FB} = \{D \in J_C(\mathbb{F}_q) : D = (P) - (\infty), P \in C(\mathbb{F}_q)\}. \tag{2}$$

Given D_1 and D_2 in $J_C(\mathbb{F}_q)$, the task is to compute $\log_{D_1} D_2$. For integers a_1, a_2, consider the divisor $a_1D_1 + a_2D_2$ given by its reduced representation $\mathrm{div}(u(x), v(x))$. Suppose $u(x)$ factors into linear terms over $\mathbb{F}_q$. If α is a root of $u(x)$, then the point $P_\alpha = (\alpha, v(\alpha))$ is on the curve and it is possible to write

$$\mathrm{div}(u(x), v(x)) = \sum_\alpha ((P_\alpha) - (\infty)).$$

This shows that $a_1D_1 + a_2D_2$ can be written as a sum of the factor basis elements which is a desired decomposition. Since the degree of $u(x)$ is g, we expect to obtain a decomposition in about $g!$ trials. Each decomposition consists of g points of the factor base along with the elements D_1 and D_2.

Each row of the resulting system of linear equations has $g+2$ non-zero entries. Since the elements D_1 and D_2 are part of the factor base, solving the system of linear equations directly provides the desired discrete log. The third step of the index calculus algorithm is not required.

Direct computation of $a_1D_1 + a_2D_2$ requires two scalar multiplications and an addition in the Jacobian of the curve. This computation can be reduced by generating the integers a_1 and a_2 using a random walk where the two scalar multiplications and the addition can be replaced with only one addition.

Obtaining q decompositions requires about $qg!$ trials and linear algebra requires $O(q^2)$ time. So, the overall complexity is $O(qg! + q^2)$ which for a fixed g is $O(q^2)$. More precisely, the $O(q^2)$ complexity holds if $g!$ is $O(q)$. The size of the Jacobian is about q^g. For q^g around 2^{160}, the relation $g!$ is around q for $g \leq 9$. We refer to [7] for the details.

3.3 Nagao [15] and Joux-Vitse [10] Decomposition

Nagao proposed a method for decomposing a divisor D in $J_C(\mathbb{F}_{q^n})$ for $n \geq 2$. The factor base is the following:

$$\mathcal{FB} = \{D \in J_C(\mathbb{F}_{q^n}) : D = (P) - (\infty), P \in C(\mathbb{F}_{q^n}), x(P) \in \mathbb{F}_q\}. \quad (3)$$

Given a divisor D in $J_C(\mathbb{F}_{q^n})$, the method involves using the Riemann-Roch theorem to set up a system of $ng(n-1)$ nonlinear equations in as many variables. Solution of this system yields a polynomial of degree ng over $\mathbb{F}_q$. If the polynomial is smooth over $\mathbb{F}_q$, then a decomposition of D over the factor base is obtained.

In a later work, Joux and Vitse [10] modified this system to obtain a decomposition method for the identity element of $J_C(\mathbb{F}_{q^n})$ with $n \geq 2$. Their decomposition method consists of several steps. In an initial phase, they set up a system of $n(n-1)g + 2(n-1)$ nonlinear equations in $n(n-1)g + 2n$ variables. This system is solved to obtain a LEX Grobner basis where two of the variables are undetermined. In the second phase, these two variables are varied over $\mathbb{F}_q$ and for different combination of values of these two variables a much simpler system of equations is solved. The solution provides a polynomial of degree $ng + 2$ which is then checked for smoothness. If smoothness is achieved, then a decomposition is obtained. In the case of $n = 2$, Joux and Vitse provide an interesting sieving technique to provide significant practical speed-up. Further, they also point out that their sieving technique can be very easily used to obtain relations with double large primes [9]. We discuss this issue in more details later.

In Nagao's method, the number of equations and variables is $ng(n-1)$. So, for $n = 1$ this is a vacuous system and the decomposition method does not apply. Similarly, for the Joux-Vitse method the number of equations is $n(n-1)g + 2(n-1)$ and again for $n = 1$ the system is vacuous and the decomposition method does not apply. Perhaps, it is for this reason that both these papers have the constraint $n \geq 2$.

Gaudry's method applied to $\mathbb{F}_{q^n}$ will result in a factor basis of size q^n, whereas in Nagao's approach and also in the Joux-Vitse decomposition, the factor basis will be of size q. This will lead to faster linear algebra provided the decomposition can be carried out. On the other hand, Gaudry's method will be applicable when the underlying field is $\mathbb{F}_q$ with q a prime, whereas, as mentioned above, Nagao's and the Joux-Vitse methods will not apply to this case.

4 A New Decomposition Method

Let $\mathbb{F}_q$ be the field of q elements. We consider the characteristic of q to be greater than 2, but, the method described below can be modified to work even for the characteristic 2 case. Let $C : y^2 = f(x)$ with $f(x) \in \mathbb{F}_q[x]$, be a hyperelliptic curve of genus g and we consider divisors in $J_C(\mathbb{F}_q)$. The factor base remains the same as that in Gaudry's algorithm i.e., the factor base is given by (2).

Suppose D is a reduced non-degenerate divisor. We write $D = \mathrm{div}(u(x), -v(x))$ and $-D = (u(x), v(x))$ with $\deg_x(u) = g$. The x-coordinates of the points on C determining D are the roots of $u(x)$; suppose these are $\delta_1, \ldots \delta_g$, then the corresponding y-coordinates are given by $-v(\delta_1), \ldots, -v(\delta_g)$. The y-coordinates of $-D$ are $v(\delta_1), \ldots, v(\delta_g)$. So, $-D = \sum_i (Q_i) - g(\infty)$ where $Q_i = (\delta_i, v(\delta_i))$. Note that the δ's are not necessarily over $\mathbb{F}_q$; the divisor D is in $J_C(\mathbb{F}_q)$ if and only if $u(x)$ is in $\mathbb{F}_q[x]$.

Given $D = (u(x), -v(x))$ and an element λ_0 in $\mathbb{F}_q$, define a bivariate polynomial $G(x, y)$ as follows:

$$G(x, y) = u(x)\lambda_0 - (y - v(x)). \tag{4}$$

The parameter λ_0 plays an important role in obtaining decompositions. By its definition, we have $G(\delta_i, v(\delta_i)) = 0$ for $i = 1, \ldots, g$, i.e., the points determining $-D$ are also zeros of $G(x, y)$. This polynomial has more zeros on C and our aim is to find them. So, we wish to look for common solution to $G(x, y) = 0$ and the curve equation $y^2 = f(x)$. Eliminating y between these two equations we obtain the equation $S(x) = 0$ where $S(x)$ is as follows:

$$S(x) = (u(x)\lambda_0 + v(x))^2 - f(x).$$

From the propery of the Mumford representation, $u(x)$ divides $v^2(x) - f(x)$ and so $S(x)/u(x)$ is also a polynomial. We define

$$H(x) = S(x)/u(x) = ((u(x)\lambda_0 + v(x))^2 - f(x))/u(x).$$

Since $u(x)$ is of degree g, $\deg(v) < \deg(u)$ and the degree of $f(x)$ is $2g + 1$, the degree of $S(x)$ is $2g + 1$. So, the degree of $H(x)$ is $g + 1$.

Suppose $H(x)$ is smooth over $\mathbb{F}_q$ and its roots are $\alpha_1, \ldots, \alpha_{g+1}$ and further assume that these roots are distinct from the roots of $u(x)$. Define $\beta_j = u(\alpha_j)\lambda_0 + v(\alpha_j)$, $i = 1, \ldots, g + 1$, i.e., β_j is the value for y when $G(\alpha_j, y) = 0$ is solved for y. Further, since α_j is a root of $S(x)$, we have $f(\alpha_j) = (u(\alpha_j)\lambda_0 + v(\alpha_j))^2 = \beta_j^2$. So, $P_j = (\alpha_j, \beta_j)$ are zeros of both $G(x, y)$ and $y^2 - f(x)$. Further, since the α's are distinct from the δ's, the P_j's are distinct from the Q_i's. Note that $\deg(G(x, y)) = 2g + 1$ and so the Q_i's and the P_j's together account for all the zeros of $G(x, y)$. So, we can write:

$$\begin{aligned} \mathrm{div}(G) &= -D + \sum_{i=1}^{g+1} (P_i) - (g + 1)(\infty) \\ &= -D + \sum_{i=1}^{g+1} ((P_i) - (\infty)). \end{aligned}$$

Since $G(x, y)$ is a polynomial (and hence a rational) function, its divisor $\mathrm{div}(G)$ is a representative of the identity of $J_C(\mathbb{F}_q)$. From this, we can write:

$$D = \sum_{i=1}^{g+1}((P_i) - (\infty))$$

where $x(P_i)$ is in $\mathbb{F}_q$. This gives the decomposition of D over the factor base.

Since the degree of $H(x)$ is $g + 1$, we expect the smoothness condition to be attained in about $(g + 1)!$ trials. There is only one control variable, namely λ_0, which ranges over $\mathbb{F}_q$. So, we expect to obtain about $q/(g + 1)!$ relations by varying λ_0. This does not provide sufficient number of relations to complete the linear algebra step. Additional control variables can be obtained as discussed below.

As in Gaudry's method suppose D_1 and D_2 are given and the requirement is to compute $\log_{D_1} D_2$. For integers a_1 and a_2, set $D = a_1D_1 + a_2D_2$ and consider the decomposition of D. For each such D, by randomly varying λ_0 it is possible to obtain $q/(g + 1)!$ relations. The variables a_1 and a_2 provide two additional control variables leading to q^2 possible divisors each of which provides about $q/(g+1)!$ relations. So, we will obtain sufficiently many relations if $q^2 > (g+1)!$.

To generate the successive D's, we adopt the following procedure. First fix a_1 to 0 and vary a_2 over its q possible values; then increment a_1 and again vary a_2 over its q possible values and so on. By this method, the cost of generating the next D from the present one is exactly one addition in $J_C(\mathbb{F}_q)$. We will require about $(g+1)!$ such D's and the total cost of generating all these divisors will also be about $(g + 1)!$ additions. The amortised cost per relation is then $(g + 1)!/q$ additions in $J_C(\mathbb{F}_q)$. If the values of q and g are such that $q > (g + 1)!$, then it is sufficient to set a_1 to 0 and consider only decompositions of $D = a_2D_2$.

The main cost is the $(g + 1)!$ trials required to obtain a single relation. Each such trial consists of obtaining $H(x)$ (of degree $g + 1$) and checking it for smoothness. So, the cost of obtaining a single relation is the smoothness check of $(g + 1)!$ polynomials each of degree $g + 1$. We next show how to use sieving to avoid this smoothness checking.

4.1 Sieving to Improve Efficiency

We adapt the sieving method proposed by Joux and Vitse in [10] for quadratic extensions to the present case. Write $S(x, \lambda_0) = (u(x)\lambda_0 + v(x))^2 - f(x)$ and $H(x, \lambda_0) = ((u(x)\lambda_0 + v(x))^2 - f(x))/u(x)$ to denote the dependence of S and H on λ_0. For a fixed λ_0, suppose α is such that $u(\alpha) \neq 0$ but, $S(\alpha, \lambda_0)$ is equal to 0. Then such an α is a root of $H(x, \lambda_0)$. It is possible that $H(x, \lambda_0)$ and $u(x)$ have a common root, but, we will not be interested in such $H(x, \lambda_0)$.

The different possible divisors D are generated from the given divisors D_1 and D_2 as described above. For each such divisor $D = \mathrm{div}(u(x), -v(x))$, the sieving step is performed as follows. We use an array $\mathsf{ctr}[0, \ldots, q - 1]$ of length q. Each entry of ctr is initialised to 0. For the divisor D, the sieving step runs

over all the elements of $\mathbb{F}_q$. For each $\alpha \in \mathbb{F}_q$, such that $u(\alpha) \neq 0$, consider the polynomial $S(\alpha, \lambda_0)$ which is quadratic in λ_0. Here we are considering λ_0 as a variable. We wish to solve the equation $S(\alpha, \lambda_0) = 0$ for λ_0. The solutions are the following:

$$\frac{-v(\alpha) \pm (f(\alpha))^{1/2}}{u(\alpha)}. \tag{5}$$

By ensuring $u(\alpha) \neq 0$ and solving for λ_0 in $S(\alpha, \lambda_0) = 0$, we are actually obtaining λ_0 such that $H(\alpha, \lambda_0) = 0$.

If $f(\alpha)$ is a perfect square over $\mathbb{F}_q$, then (5) gives two values of λ_0 which are in $\mathbb{F}_q$. Denote these values as λ_{00} and λ_{01}. Increment $\mathsf{ctr}[\lambda_{00}]$ and $\mathsf{ctr}[\lambda_{01}]$. There are two ways in which an $O(q)$ pre-computation helps in speeding up.

1. Prepare a table of square roots of $f(\alpha)$ for all $\alpha \in \mathbb{F}_q$ and use this table to solve (5). This table is independent of the divisor D and will be used in all the sieving steps. This avoids computing $(f(\alpha))^{1/2}$ during actual sieving.
2. Prepare a table of inverses of all the non-zero elements of $\mathbb{F}_q$. After computing $\gamma = u(\alpha)$ use this table to obtain γ^{-1}. This avoids computing the multiplicative inverse of $u(\alpha)$ during actual sieving.

After the loop over all elements $\alpha \in \mathbb{F}_q$ has been completed, we check each entry of ctr. If $\mathsf{ctr}[\lambda_0] = g + 1$, then this λ_0 results in the polynomial $H(x, \lambda_0)$ in x which is smooth over $\mathbb{F}_q$. To see this note that if $\mathsf{ctr}[\lambda_0]$ is equal to $g+1$, then the sieving has encountered $g + 1$ roots of $H(x)$ in $\mathbb{F}_q$; since $H(x)$ is of degree $g + 1$, it must actually be smooth over $\mathbb{F}_q$.

Once a value of λ_0 for which $H(x)$ is smooth has been identified, the roots of $H(x)$ for this value of λ_0 are obtained by factorisation. These roots have already been encountered in the sieving phase and could be stored and used later. We comment more on this issue later. On the other hand, even if we do not store the roots, only one factoring is required per decomposition and the efficiency loss for this may not be significant.

Each sieving step consists of an $O(q)$ loop and at each iteration of the loop, it is required to compute $u(\alpha)$, $v(\alpha)$ and a small number of additional $\mathbb{F}_q$-operations and table look-ups. Computing $u(\alpha)$ and $v(\alpha)$ requires $O(g)$ $\mathbb{F}_q$-operations. The sieving step is repeated for about $(g + 1)!$ divisors D to obtain about q relations. At no point is it required to perform a smoothness check.

A matrix view of the sieving process: For a fixed divisor, we provide an alternate view of the sieving process. Consider a $q \times q$ matrix M. Further consider that α varies over the columns and λ_0 varies over the rows of M. Entries of M are either 0 or 1. An entry at the position (λ_0, α) of M is 1 if λ_0 is a solution in $\mathbb{F}_q$ of (5), i.e., if $(f(\alpha))^{1/2}$ is in $\mathbb{F}_q$; otherwise the entry at the position (λ_0, α) of M is 0. So, for every α, the column indexed by α has either two 1's or zero 1's. If we assume that $(f(\alpha))^{1/2}$ is in $\mathbb{F}_q$ for about half of the α's, the total number of 1's in the matrix M is about q. For any λ_0, the sum of all elements in the row of M indexed by λ_0 is the value $\mathsf{ctr}[\lambda_0]$. Hence, $\sum_{\lambda_0 \in \mathbb{F}_q} \mathsf{ctr}[\lambda_0]$ is also about q.

As a result, if for some λ_0 the value $\mathsf{ctr}[\lambda_0]$ is greater than 1, then for some other λ_0, the value $\mathsf{ctr}[\lambda_0]$ will be zero.

The above description explains that the number of pairs (λ_0, α) for which $M[\lambda_0, \alpha] = 1$ is about q. This suggests an alternative implementation of ctr. It can be implemented as a list of pairs (λ_0, α). Whenever an α is obtained such that $(f(\alpha))^{1/2}$ is in $\mathbb{F}_q$, then (λ_0, α) is appended to ctr, where λ_0 is a solution to (5) for this α. After the sieving step for a divisor is complete, the list ctr is sorted on λ_0. Since the size of ctr will be about $O(q)$, sorting will require $Oq \log q)$ time. One can then perform a pass over ctr obtaining all possible λ_0 such that there are $g+1$ pairs (λ_0, α) in ctr. Note that this directly provides the $g+1$ α's which are the roots of $H(x)$ for this value of λ_0.

Parallelism: The loop of α over $\mathbb{F}_q$ is completely parallelisable. The computations for two different α's can be carried out independent of each other, but, the array ctr will be shared memory. The only point where a conflict may arise is if two different α's give rise to the same value for λ_0 leading to a situation where the same position of ctr needs to be updated. This issue can be tackled using standard techniques for ensuring consistent writes. The other way in which parallelism can be exploited is by executing the different sieving steps for different divisors in parallel. This will require separate copies of ctr to be available for the different sieving steps which will increase the memory requirement. Depending upon available resources, a suitable method for exploiting parallelism may be determined. We have not tried to exploit parallelism in our experiments.

4.2 Comparison to Gaudry's Method

A relation in Gaudry's method is obtained by decomposing $a_i D_1 + b_i D_2$ over the factor basis, where a_i and b_i are obtained using a random walk. Each such decomposition involves g elements of the factor basis and a decomposition is obtained in about $g!$ trials. So, obtaining q decompositions require about $qg!$ trials. Generating $a_{i+1} D_1 + b_{i+1} D_2$ from $a_i D_1 + b_i D_2$ can be done using a single addition. So, the cost of obtaining a single decomposition consists of $g!$ additions and $g!$ checking of smoothness of a degree g polynomial. Each relation in Gaudry's method involves D_1, D_2 and g elements of the factor basis. So, the number of non-zero entries in each row of the matrix for the linear algebra step is $g+2$.

In the new method, each sieving step on a divisor results in about $q/(g+1)!$ decompositions and to obtain about q decompositions one requires to perform the sieving step on about $(g+1)!$ divisors. This leads to a total of about $(g+1)!$ additions in $J_C(\mathbb{F}_q)$. Each sieving step has q iterations where each iteration involves $O(g)$ $\mathbb{F}_q$-operations and a small number of table look-ups.

The relative efficiencies of the two methods in obtaining q relations is as follows. In Gaudry's method, a total of about $qg!$ additions in $J_C(\mathbb{F}_q)$ and $qg!$ smoothness checking of degree g polynomials over $\mathbb{F}_q$ are required. The total number of operations required by the new method consists of about $(g+1)!$ additions in $J_C(\mathbb{F}_q)$ and about $O(qg(g+1)!)$ $\mathbb{F}_q$-operations. The reduction in the number of additions in $J_C(\mathbb{F}_q)$ and the non-requirement of smoothness checking

should lead to faster decompositions in the new method. Our initial experimental results indicate that this is indeed the case.

Each decomposition in the new method results in a relation involving D and $g+1$ elements of the factor base. In general $D = a_1 D_1 + a_2 D_2$ and so a relation involves $g+3$ divisors. As mentioned above, if $q > (g+1)!$, then one can set $a_1 = 0$ so that each relation involves $g+2$ divisors which is the same as in Gaudry's decomposition. The matrix for the linear algebra step has $g+2$ non-zero elements per row if $q > (g+1)!$; and has $g+3$ non-zero elements per row if $q \leq (g+1)! < q^2$. In both cases, the linear algebra step is expected to require about the same time as in Gaudry's algorithm.

4.3 Double Large Prime Variant

The sieving technique makes it simple to apply the double large prime [9] variant. This was mentioned by Joux and Vitse in the context of sieving for quadratic extensions, but, applies equally well to the current context. The idea of the double large prime variant is to reduce the size of the factor basis so that the linear algebra step takes lesser time.

Suppose q is a prime. In the double large prime variant, the factor base will consist of "small" primes which are divisors $(P) - (\infty)$ with $x(P)$ to be at most some pre-determined bound B. Large primes are divisors $(P) - (\infty)$ with $x(P) > B$. The main idea of the double large prime variant is to decompose a divisor D into a sum of several "small" primes and at most two large primes. Ensuring this in general is difficult.

With the sieving method that we have described this becomes easy. For a divisor, the sieving varies α over all possible q. To implement the double large prime variant, we simply vary α up to the bound B and increment $\mathsf{ctr}[\lambda_0]$ corresponding to the solutions for λ_0 as before. Later, we select λ_0 for which $\mathsf{ctr}[\lambda_0]$ is $g-1$ or more. If this value is $g+1$, then as before, we obtain a relation consisting of only small primes; if the value is g, then since the degree of $H(x)$ is $g+1$, the other root must also be in $\mathbb{F}_q$ and this leads to a relation with a single large prime. If the value of $\mathsf{ctr}[\lambda_0]$ is $g-1$, then again since the degree of $H(x)$ is $g+1$, for this λ_0, the corresponding $H(x)$ has $g-1$ roots in $\mathbb{F}_q$. The other factor of $H(x)$ is quadratic. If this is smooth (which happens with probability $1/2$), then we obtain a decomposition of D consisting of $g-1$ small primes and at most 2 large primes.

Consider the matrix M mentioned in Section 4.1 which describes the sieving process. With the double large prime variant, the number of columns in M reduces from q to B. As a result, many of the row sums turn out to be zero. So, it will be advantageous to have some method to ensure that we only check the positions where ctr has positive values. We discuss two methods to do this.

Indirection: One method is to use indirection. Apart from ctr, we use an additional array val which is of maximum length q, but the actual length is lesser. The initial length of val is 0. When (5) results in two solutions λ_{00} and λ_{01},

these values are appended to val and its current length increases by 2. The entries $\mathsf{ctr}[\lambda_{00}]$ and $\mathsf{ctr}[\lambda_{01}]$ are incremented as before. Suppose the final length of val is N. After the pass of α over $\mathbb{F}_q$ is over, for each i in $1, \ldots, N$, we compare the value of $\mathsf{ctr}[\mathsf{val}[i]]$ with $g-1$. This loops over the N positions of ctr having positive values. Since, N will be much smaller than q, this saves time.

Associative array: The other method is to implement ctr as an associative array indexed by elements of $\mathbb{F}_q$, instead of a fixed array of size q. The entries of ctr are of the type (γ, i), where γ is an element of $\mathbb{F}_q$ and i is a positive integer. We use the notation $\mathsf{ctr}[\gamma] = i$ to denote that the pair (γ, i) is present in the array. By incrementing $\mathsf{ctr}[\gamma]$ we mean the following: if (γ, i) is present in the array, then it is replaced by $(\gamma, i+1)$; and if γ is not equal to the first component of any pair already in ctr, then the pair $(\gamma, 1)$ is inserted into ctr. During the sieving process, suppose (5) gives two values λ_{00} and λ_{01} which are in $\mathbb{F}_q$. Increment $\mathsf{ctr}[\lambda_{00}]$ and $\mathsf{ctr}[\lambda_{01}]$. After the pass of α over $\mathbb{F}_q$ is over, let N be the length of the associative array ctr. By construction, if (λ_0, i) is in ctr, then $\mathsf{ctr}[\lambda_0] > 0$. Now, a pass is made over the entries of ctr comparing each value with $g-1$ as before. Compared to the indirection method, no array of size q is required, but, in this case, an index structure is required to implement ctr.

The efficiency of generating relations depends crucially on the value of B. The value of B in turn determines the value N of the maximum length of ctr. Experimental results indicate that the associative array based approach is faster if B is small, whereas the indirection based approach is faster when B is comparatively larger. In a concrete setting, it is advisable to use the method which is faster.

In general B will be q^r such that q^{2r} is $o(q^{g/2})$. This will ensure that the linear algebra step involving B elements runs in time B^2 and is still faster than the Pollard's rho method. During relation generation, it will be required to obtain much more than B relations so that the large primes in the relations can be eliminated to obtain relations involving only the factor base elements. A graph based approach is used to achieve this [9]. We do not discuss these details, since our focus here is the average time for obtaining a single relation with at most two large primes. For an actual discrete log computation, sufficiently many decompositions will have to be carried out to obtain the required number of double large prime relations.

The sieving loop runs over B values of α. This ensures that each sieving loop runs much faster. Further, while checking the values of ctr, the loop runs over N values which is also significantly smaller than q. So, overall each sieving step runs much faster than the case for obtaining relations where only small primes are involved. The catch, however, is that now each sieving step yields significantly less number of relations. So, the average time required for obtaining a single relation actually goes up. We provide results of some practical experiments later.

5 Experimental Results

In this section, we report the results of some experiments that we conducted with the new method.

For the experiments, we used the Magma Computer Algebra System [17] on a single core of Intel Xeon CPU @ 3.07GHz. The efficiency of obtaining decompositions in the new method does not depend on the order of the Jacobian and depends only on the genus and the underlying field $\mathbb{F}_q$. So, for the experiments we have fixed the genus and the value of q and run the decomposition method on randomly generated hyperelliptic curves. Further, for simplicity we have chosen q to be a prime.

5.1 Examples of Decompositions Obtained Using the New Method

We first provide some examples of decompositions using the new method. Consider a hyperelliptic curve C of genus $g = 7$, defined by $y^2 = x^{15}+26412x+15471$ over the field $\mathbb{F}_q$, where $q = 1048583$.
Let

$$\begin{aligned} D_1 = (x^7 &+ 361878x^6 + 853622x^5 + 966112x^4 + 379368x^3 + 578236x^2 \\ &+369465x + 201503, 983227x^6 + 37594x^5 + 655264x^4 + \\ &27833x^3 + 886828x^2 + 931655x + 25374); \\ D_2 = (x^7 &+ 616043x^6 + 290099x^5 + 162688x^4 + 204670x^3 + 551267x^2 \\ &+390226x + 747247, 905210x^6 + 983958x^5 + 329094x^4 + \\ &1003866x^3 + 225827x^2 + 817769x + 456719). \end{aligned}$$

Suppose $a = 672611$ and $b = 529480$. Then by varying λ_0, it is possible to obtain $q/(g+1)!$ decompositions of $aD_1 + bD_2$. Two such examples are given below.

$$\begin{aligned} -aD_1 - bD_2 &= (x + 404553, 819523) + (x + 476821, 73840) \\ &+ (x + 607178, 332244) + (x + 608877, 68511) + (x + 647811, 676561) + \\ &(x + 898698, 42974) + (x + 958676, 247112) + (x + 1041752, 736564); \\ -aD_1 - bD_2 &= (x + 122108, 276972) + (x + 178013, 962779) \\ &+ (x + 189540, 1018873) + (x + 202334, 504402) + (x + 658095, 911545) \\ &+ (x + 726744, 503834) + (x + 989490, 958207) + (x + 1046320, 202759). \end{aligned}$$

Consider another pair of values for a and b, say, $a = 2405771$ and $b = 1403025$. Then an example of a decomposition of $aD_1 + bD_2$ is as follows.

$$\begin{aligned} -aD_1 - bD_2 &= (x + 185559, 22966) + (x + 192011, 527282) \\ &+ (x + 262101, 183920) + (x + 335423, 773936) + (x + 393421, 741757) \\ &+ (x + 432914, 706326) + (x + 589633, 749516) + (x + 750866, 142288). \end{aligned}$$

5.2 Some Timing Results

We provide timing results of the new decomposition method for various parameters in Tables 1, 2 and 3. For these experiments, we have used a pre-computed table to obtain the values of $(f(\alpha))^{1/2}$. We did not, however, use a pre-computed table to obtain the inverses of $u(\alpha)$. Doing this should further reduce the timings. We have used an array based implementation of ctr as mentioned in Section 4.1. Presently, we are working on implementing Gaudry's method and hope to have comparative timings soon. Some initial results indicate that the timings for the new method are better.

Table 1. Average time in seconds per decomposition for some example hyperelliptic curves.

$y^2 = x^{2g+1} + 26412x + 15471$ over $\mathbb{F}_p$, $p = 1048583$ ($\approx 2^{20}$)						
$g = 4$	$g = 5$	$g = 6$	$g = 7$	$g = 8$	$g = 9$	$g = 10$
0.00039	0.00188	0.0129	0.099	0.95	9.4	103

$y^2 = x^{2g+1} + 14212x + 47156$ over $\mathbb{F}_p$, $p = 8388593$ ($\approx 2^{23}$)						
$g = 4$	$g = 5$	$g = 6$	$g = 7$	$g = 8$	$g = 9$	$g = 10$
0.00041	0.00192	0.0131	0.104	0.96	9.6	105

$y^2 = x^{2g+1} + 26412x + 15471$ over $\mathbb{F}_p$, $p = 33554467$ ($\approx 2^{25}$)						
$g = 4$	$g = 5$	$g = 6$	$g = 7$	$g = 8$	$g = 9$	$g = 10$
0.00043	0.00210	0.0141	0.114	1.04	10.5	118

The data given in Table 1 is the average of timings of more than ten thousand decompositions for genus up to 8. For genus 9, we have taken the the average of more than thousand decompositions. For $g = 10$, as it takes more time, we could consider only a few hundred decompositions.

We next consider the timing results of the new method for obtaining decompositions with at most two double large primes. The total time for relation collection in the double large prime algorithm involves the cost of maintaining an LP graph and the cost of finding a cycle in that graph. We have not implemented these steps.

The timings that we provide are indicative of the time required to obtain a single decomposition using at most two large primes. Since double large prime variant is more relevant for genus 3 and 4 cases, we have done our experiments for these two cases only. For these experiments, we have used the implementations of ctr which have been described in Section 4.3; specifically, we have used the associative array implementation when $B \leq 2^{20}$ and the indirection-based implementation for larger values of B. We have found the respective techniques to be faster for the corresponding cases.

Comparing Table 1 and Tables 2, 3 for $p = 33554393$ and $g = 4$, it is to be noted that obtaining a single relation double large prime relation takes consid-

erably more time. Further, this time goes up as the value of B goes down. This behaviour is to be expected and our results only confirm the behaviour.

Table 2. Average time in seconds for one decomposition with at most two large primes. Here $p = 33554393 \approx 2^{25}$ and B is the bound determining 'small primes'.

$y^2 = x^7 + 5412x + 84711$ over $\mathbb{F}_p$		$y^2 = x^9 + 56241x + 7141$ over $\mathbb{F}_p$	
$B = 2^{16}$	$B = 2^{18}$	$B = 2^{20}$	$B = 2^{22}$
0.00747	0.00219	0.05533	0.00385

Table 3. Average time in seconds for one decomposition with at most two large primes. Here $p = 268435399 \approx 2^{28}$ and B is the bound determining 'small primes'.

$y^2 = x^7 + 14572x + 94347$ over $\mathbb{F}_p$		$y^2 = x^9 + 63441x + 7453$ over $\mathbb{F}_p$	
$B = 2^{18}$	$B = 2^{20}$	$B = 2^{22}$	$B = 2^{25}$
0.01544	0.00483	0.27116	0.00515

6 Conclusion

In this paper, we have described a new method for decomposing a divisor in the Jacobian of a small genus hyperelliptic curve. In practical terms, the method is faster than the decomposition method proposed earlier by Gaudry. The speed-up is obtained by using a sieving method which is based on a method suggested by Joux and Vitse in the context of curves over fields of extenstion degree two. The sieving method combines well with the double large prime variant.

References

1. Leonard M. Adleman, Jonathan DeMarrais, and Ming-Deh A. Huang. A subexponential algorithm for discrete logarithms over the rational subgroup of the jacobians of large genus hyperelliptic curves over finite fields. In Leonard M. Adleman and Ming-Deh A. Huang, editors, *Algorithmic Number Theory, First International Symposium, ANTS-I, Ithaca, NY, USA, May 6-9, 1994, Proceedings*, volume 877 of *Lecture Notes in Computer Science*, pages 28–40. Springer, 1994.
2. Claus Diem. The GHS attack in odd characteristic. *J. Ramanujan Math. Soc.*, 18(1):1–32, 2003.
3. Andreas Enge. Computing discrete logarithms in high-genus hyperelliptic jacobians in provably subexponential time. *Math. Comput.*, 71(238):729–742, 2002.
4. Andreas Enge and Pierrick Gaudry. A general framework for subexponential discrete logarithm algorithms. *Acta Arithmetica*, 102:83–103, 2002.
5. Ralf Flassenberg and Sachar Paulus. Sieving in function fields. *Experimental Mathematics*, 8(4):339–349, 1999.

6. Gerhard Frey. How to disguise and elliptic curve (Weil descent). Talk at the 2nd Elliptic Curve Cryptography (ECC) Workshop, 1998.
7. Pierrick Gaudry. An algorithm for solving the discrete log problem on hyperelliptic curves. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*, pages 19–34. Springer, 2000.
8. Pierrick Gaudry, Florian Hess, and Nigel P. Smart. Constructive and Destructive Facets of Weil Descent on Elliptic Curves. *J. Cryptology*, 15(1):19–46, 2002.
9. Pierrick Gaudry, Emmanuel Thomé, Nicolas Thériault, and Claus Diem. A double large prime variation for small genus hyperelliptic index calculus. *Math. Comput.*, 76(257):475–492, 2007.
10. Antoine Joux and Vanessa Vitse. Cover and decomposition index calculus on elliptic curves made practical - application to a previously unreachable curve over f_{p^6}. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 9–26. Springer, 2012.
11. Neal Koblitz. Elliptic curve cryptosystesm. *Math. Comp.*, 48(177):203–209, 1987.
12. Neal Koblitz. Hyperelliptic cryptosystems. *J. Cryptology*, 1(3):139–150, 1989.
13. Alfred Menezes, Yi-Hong Wu, and R. Zuccherato. An elementary introduction to hyperelliptic curves. Appendix in 'Algebraic Aspects of Cryptography' by Neal Koblitz, 1998.
14. Victor S. Miller. Use of elliptic curves in cryptography. In Hugh C. Williams, editor, *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer, 1985.
15. Koh-ichi Nagao. Decomposition attack for the Jacobian of a hyperelliptic curve over an extension field. In *Algorithmic number theory*, volume 6197 of *Lecture Notes in Comput. Sci.*, pages 285–300. Springer, Berlin, 2010.
16. Nicolas Thériault. Index calculus attack for hyperelliptic curves of small genus. In Chi-Sung Laih, editor, *ASIACRYPT*, volume 2894 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2003.
17. Magma v2.19 7. `http://magma.maths.usyd.edu.au/magma/`.
18. M. D. Velichka, Michael J. Jacobson Jr., and Andreas Stein. Computing discrete logarithms in the jacobian of high-genus hyperelliptic curves over even characteristic finite fields. *Math. Comput.*, 83(286), 2014.