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

Palash Sarkar and Shashank Singh

Applied Statistics Unit
Indian Statistical Institute
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.

Keywords: Discrete Log, Index calculus, Hyperelliptic curve, Cryptography

MSC: 11Y16, 11T71, 94A60

1 Introduction

Elliptic curve cryptography was independently introduced by Koblitz [12] and Miller [15] and was soon followed by hyperelliptic curve (HEC) cryptography which was introduced by Koblitz [13]. For hyperelliptic curves, cryptography is carried out in a suitably 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 the DLP in different cryptographic groups has been the use of index calculus algorithms. The structure of such algorithms identifies a factor basis which consists of a small subset of elements of the group and a method to decompose elements of the group over the factor basis. Any such decomposition gives rise to a relation among the element that is decomposed and some of the elements of the factor basis. After obtaining sufficiently many relations, sparse linear algebra techniques are used to recover the discrete logs of the factor basis elements. From this, it is possible to obtain the discrete log of a target element either directly or by a further computation. The main tasks in designing an index calculus algorithm are 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 curves, this provided a sub-exponential time algorithm for the DLP in the Jacobian of such curves. Later developments along this line have been reported in [6, 4, 5, 21].

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 rho algorithm. This was described by Gaudry in [8]. The work is important and provided a

practical algorithm for solving the DLP over small genus curves. In particular, the work [8] actually solved some HEC-DLP challenges. Subsequently, variants of this algorithm called the large prime variant [19] and the double large prime variant [10] were introduced to improve the efficiency, especially for genus 3 and 4.

Let q be a prime power and $\mathbb{F}_q$ be the field of q elements having characteristic greater than 2. Suppose $C : y^2 = f(x)$ is a HEC of genus $g \geq 2$ where $f(x)$ is a polynomial over $\mathbb{F}_q$ of degree $2g + 1$. The notation $J_C(\mathbb{F}_q)$ denotes the Jacobian of C restricted to the set of divisor classes which can be represented by $\mathbb{F}_q$-rational divisors. Gaudry's algorithm works over $J_C(\mathbb{F}_q)$ by defining a factor basis and providing a method for decomposing a divisor over the factor basis. Nagao [16] proposed a decomposition method for curves C defined over $\mathbb{F}_{q^n}$ where $n \geq 2$. Subsequently, Joux and Vitse [11] proposed a modification of Nagao's method for such curves and combined this method with the Weil descent method [7, 9, 2] to solve the 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 [8] (and the double large prime variant [10]) 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 [11] 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 and Gaudry's method in Magma and report some experimental results and timings. These indicate that using the new method results in some practical reduction in the time for obtaining a decomposition.

2 Notation

We mention some notation on hyperelliptic curves. A basic description on hyperelliptic curves can be found in [14]. Let $K = \mathbb{F}_q$ be the field of definition of the curve $C : y^2 = f(x)$ and L be an extension of K. We consider reduced divisors by their Mumford representations: i.e., a divisor D will be written as $D = \mathrm{div}(u(x), v(x))$ where $u(x), v(x) \in L[x]$, $\deg(v) < \deg(u) \leq g$ and $u(x)$ divides $v^2(x) - f(x)$. If $u(x) \in L[x]$, then D is said to be an L-rational divisor. A divisor class is said to be L-rational if it can be represented by an L-rational divisor. The set of all divisor classes of C will be denoted by J_C and the set of all L-rational divisor classes will be denoted by $J_C(L)$.

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 the 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 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.

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 S. The degree of D is the degree of $u(x)$; D is said to be prime if $u(x)$ is irreducible. Further, D 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 $v_i(x)$. 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, we have $\mathrm{div}(G) \equiv 0$. 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, checking whether $N(G)$ is S-smooth can be avoided using a sieving technique due to Flassenberg and Paulus [6]. 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 [21] modified it to work over even characteristic fields.

3.2 Gaudry's Decomposition [8]

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

$$\mathrm{div}(u(x), v(x)) = \sum_{\alpha} ((P_\alpha) - (\infty)). \tag{3}$$

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 at most g, we expect to obtain a decomposition in about $g!$ trials. Each decomposition consists of g points of the factor basis along with the elements D_1 and D_2.

For each smooth relation that is obtained, the right side of (3) is written as a row in a matrix having $\mathbb{F}_q$ columns. The corresponding values of a_1 and a_2 are stored separately. Once $q + 1$ independent rows are obtained, it becomes possible to use linear algebraic techniques to obtain a linear combination of the rows which sum to 0. From this it becomes possible to obtain $\log_{D_1} D_2$. We refer to [8] for the details. 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. Each row of the matrix for the linear algebra step has g non-zero entries. Using sparse matrix techniques (Lanczos or block Weidemann) the linear algebra step requires $O(gq^2)$ time. So, the overall complexity is $O(qg! + gq^2)$ 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} and $g \leq 9$, the quantity $g!$ is at most q and so the overall complexity is $O(q^2)$. We again refer to [8] for further details.

3.3 Nagao [16] and Joux-Vitse [11] Decompositions

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

$$\mathcal{FB} = \{D \in J_C(\mathbb{F}_{q^n}) : D = (P) - (\infty), P \in C(\mathbb{F}_{q^n}), x(P) \in \mathbb{F}_q\}. \tag{4}$$

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 basis is obtained.

In a later work, Joux and Vitse [11] modified this method to obtain a decomposition method for the divisor of a rational function. 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 or free. In the second phase, these two variables are varied over $\mathbb{F}_q$ and for different combinations 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 a 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 [10]. 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. 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

As in Gaudry's method, we consider the Jacobian $J_C(\mathbb{F}_q)$ of a curve $C : y^2 = f(x)$ of genus $g \geq 2$ over a field $\mathbb{F}_q$ having characteristic greater than 2. The factor basis remains the same as that in Gaudry's algorithm i.e., the factor basis 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:

$$G(x, y) = u(x)\lambda_0 - (y - v(x)). \tag{5}$$

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 solutions 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 at most 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 $y^2 - f(x)$. 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 $\mathrm{div}(G) \equiv 0$. From this, we can write:

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

where P_i is an $\mathbb{F}_q$-rational point of C. This gives the decomposition of D over the factor basis.

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. The variable a_i can range from 0 to $\mathrm{ord}(D_i) - 1$. Since $\mathrm{ord}(D_i)$ would be about q^g, by varying the control variables (a_1, a_2) it is possible to obtain about q^{2g} divisors D. About q relations will be obtained if $q^{2g} > (g+1)!$ which holds whenever $q > g+1$.

To generate the successive D's, we adopt the random walk technique as in Gaudry's method. 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 $J_C(\mathbb{F}_q)$.

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 [11] 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:

$$\frac{-v(\alpha) \pm (f(\alpha))^{1/2}}{u(\alpha)}. \tag{6}$$

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 (6) 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 $\mathsf{ctr}[\lambda_{01}]$. Note that it is possible to have $f(\alpha) = 0$ in which case λ_{00} and λ_{01} are equal.

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 (6). 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 (6), 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 (6) for this α. After the sieving step for a divisor is complete, the list ctr is sorted according to λ_0. Since the size of ctr will be about $O(q)$, sorting will require $O(q \log q)$ 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 independently 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 polynomial of degree at most g. Each relation in Gaudry's method involves D_1, D_2 and g elements of the factor basis. As mentioned in Section 3.2, the number of non-zero entries in each row of the matrix for the linear algebra step is g.

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 (at most) 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 lead to faster decompositions in the new method. Later we report experimental results which 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 basis. The linear algebra step is carried out as in Gaudry's method. The only difference is that for the new method the number of non-zero entries per row of the matrix is $g+1$. As a result, the linear algebra step for the new method is expected to require $O((g+1)q^2)$ time which is about the same time as in Gaudry's algorithm.

4.3 Double Large Prime Variant

In Gaudry's method, the costs of relation collection and linear algebra are respectively $O(qg!)$ and $O(gq^2)$. On the other hand, the cost of Pollard rho is $O(q^{g/2})$. So, for g equal to

3 or 4, Gaudry's method is not faster than Pollard rho. The idea of the double large prime variant [10] is to reduce the size of the factor basis so that the linear algebra step takes less time. Using a factor basis of size q^r for $r < 1$, the time for linear algebra is $q^{2r} = o(q^2)$. On the other hand, reducing the size of the factor basis leads to an increase in the time for finding a single relation. The value of r is chosen so as to balance the cost for relation collection and linear algebra and to ensure that this overall cost is less than Pollard rho algorithm. The work [10] shows that for $r = 1 - 1/g$, the times for relation collection and linear algebra are balanced.

For the new decomposition method, the sieving technique that we use makes it simple to apply the double large prime variant. This was mentioned by Joux and Vitse in the context of sieving for quadratic extensions, but, applies equally well to the current context.

Suppose q is a prime. This assumption is not necessary and it is easy to modify the method described below to work for non-prime q. We briefly mention this later.

In the double large prime variant, the factor basis consists 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 (6) 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 (6) 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 for some $r < 1$ and so for a fixed genus, the linear algebra step will take time $O(q^{2r})$. 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 basis elements. A graph based approach is used to achieve this [10]. The algorithm and its analysis are given in [10] and we do not propose any modification of these details. Our focus here is only the average time for obtaining a single relation with at most two large primes.

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 a significantly smaller number of relations. So, the average time required for obtaining a single relation actually goes up.

Suppose that q is not a prime and that $q = p^m$ where p is a prime and $m > 1$. Then any element α of $\mathbb{F}_q$ can be written as a polynomial $\alpha(y) \in \mathbb{F}_p[y]$ of degree at most $m-1$. Denote by $\|\alpha\|$ the quantity $\alpha(p)$ evaluated over the integers. The bound $B = q^r$ is defined as above. Let $D = (P) - (\infty)$ be a divisor and $\alpha = x(P)$. Then D is said to be a "small" prime if $\|\alpha\| \leq B$. With this modification, the sieving process described above goes through without any other change. We note, though, that we have not implemented the case of non-prime q.

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 [20] on a single core of Intel Xeon CPU @3.07GHz.

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

$$\begin{aligned} D_1 = (x^7 + 361878x^6 + 853622x^5 + 966112x^4 + 379368x^3 + 578236x^2 \\ +369465x + 201503, 983227x^6 + 37594x^5 + 655264x^4 + \\ 27833x^3 + 886828x^2 + 931655x + 25374); \\ D_2 = (x^7 + 616043x^6 + 290099x^5 + 162688x^4 + 204670x^3 + 551267x^2 \end{aligned}$$

$$+390226x + 747247, 905210x^6 + 983958x^5 + 329094x^4 + 1003866x^3 + 225827x^2 + 817769x + 456719).$$

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.

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

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.

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

5.2 Some Timing Results

The efficiency of obtaining decompositions in both the new method and Gaudry's method depend only on g and q. So, for the experiments we have fixed g and q and have run the decomposition methods on randomly generated hyperelliptic curves. Further, for simplicity we have chosen q to be a prime.

We provide timing results of the new decomposition method and Gaudry's method for various parameters in Table 1. As discussed earlier, using a factor basis of size q results in linear algebra requiring time $O((g+1)q^2)$. For $g = 2$ or 3, the resulting complexity is more than that required for Pollard rho. It is due to this reason that we report timing results for $g \geq 4$ in Table 1. The data given in Table 1 is the average of timings of more than a hundred thousand decompositions for genus up to 8. For genus 9, we have taken the the average of more than ten thousand decompositions.

For conducting the experiments using the new method, we have used a pre-computed table to obtain the values of $(f(\alpha))^{1/2}$ and another pre-computed table to obtain the inverses of $u(\alpha)$. We have used an array based implementation of `ctr` as mentioned in Section 4.1. The smoothness checking required for Gaudry's method was performed in two stages as suggested in [8]. To check a polynomial $u(x)$ for smoothness, Swan's [18] necessary condition was first applied as a filter; if it passes this test, then the criterion $x^p \bmod u(x) \stackrel{?}{\equiv} x$ was applied. Factoring was carried out only after this criterion was satisfied.

For the first two primes, the speed-up of the new method is about two times that of Gaudry's method. For the third prime, the speed-up for low values of g is also about two but, increases as g increases.

We next consider the timing results of the new method for obtaining decompositions with at most two double large primes. These are given in Tables 2, 3 and 4. 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

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

$y^2 = x^{2g+1} + 26412x + 15471$ over $\mathbb{F}_p$, $p = 1048583$ ($\approx 2^{20}$)						
Method	$g = 4$	$g = 5$	$g = 6$	$g = 7$	$g = 8$	$g = 9$
Our	0.00037	0.0017	0.0125	0.098	0.90	9.28
Gaudry	0.00059	0.0032	0.0204	0.161	1.42	14.08
Speedup	1.59	1.88	1.63	1.64	1.58	1.51

$y^2 = x^{2g+1} + 14212x + 47156$ over $\mathbb{F}_p$, $p = 8388593$ ($\approx 2^{23}$)						
Method	$g = 4$	$g = 5$	$g = 6$	$g = 7$	$g = 8$	$g = 9$
Our	0.00041	0.0019	0.0126	0.103	0.93	9.30
Gaudry	0.00087	0.0045	0.0286	0.223	1.90	18.76
Speedup	2.12	2.36	2.26	2.16	2.04	2.01

$y^2 = x^{2g+1} + 26412x + 15471$ over $\mathbb{F}_p, p = 33554467$ ($\approx 2^{25}$)						
Method	$g = 4$	$g = 5$	$g = 6$	$g = 7$	$g = 8$	$g = 9$
Our	0.00043	0.0020	0.0140	0.112	1.01	10.51
Gaudry	0.00083	0.0051	0.0371	0.344	4.98	53.53
Speedup	1.93	2.55	2.65	3.07	4.93	5.09

$y^2 = x^{2g+1} + 26412x + 15471$ over $\mathbb{F}_p, p = 268435459$ ($\approx 2^{28}$)						
Method	$g = 4$	$g = 5$	$g = 6$	$g = 7$	$g = 8$	$g = 9$
Our	0.00060	0.0024	0.0146	0.118	1.04	10.99
Gaudry	0.00153	0.0093	0.0677	0.591	5.32	58.14
Speedup	2.55	3.87	4.63	5.00	5.11	5.29

steps. The timings that we provide are indicative of the time required to obtain a single decomposition using at most two large primes.

As mentioned in [10], the double large prime variant is more relevant for genus 3 and 4 cases. So, we have done our experiments for these two cases only. We note that for $g = 3$, in certain cases (18.57% of all genus 3 hyperelliptic curves) the DLP on hyperelliptic curves can be transferred to that for non-hyperelliptic curves [17] where more efficient algorithms can be used to solve the DLP [3]. The double large prime variant, on the other hand, is applicable to all genus 3 hyperelliptic curves and so it is reasonable to still consider timings for decompositions on such curves.

In practice, the value of $B = q^r$ will be chosen so as to balance the costs of relation collection and linear algebra steps. As mentioned earlier, [10] shows that choosing $r = 1-1/g$ balances these two costs. In our experiments, for each curve, we use two values of the bound B determining the size of the factor basis. These values enclose the value $q^{1-1/g}$.

For these experiments, we have used the implementations of `ctr` which have been described in Section 4.3. From the results, it can be seen that for the double large prime variant, in comparison to Gaudry's method the new method performs even better. This is due to the fact that obtaining a double large prime relation requires many more trials and the requirement of smoothness check at each trial in Gaudry's method has a more pronounced effect.

For $g = 4$, comparing Table 1 and Tables 2, 3 and 4, it is to be noted that obtaining a single double large prime relation takes considerably 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 $p = 8388593 \approx 2^{23}$ and B is the bound determining 'small primes'.

$y^2 = x^7 + 14212x + 47156$ over $\mathbb{F}_p$			$y^2 = x^9 + 14212x + 47156$ over $\mathbb{F}_p$		
Method	$B = 2^{15}$	$B = 2^{16}$	Method	$B = 2^{17}$	$B = 2^{18}$
Our	0.0028	0.0016	Our	0.140	0.039
Gaudry	0.0152	0.0076	Gaudry	0.403	0.106
Speedup	5.43	4.75	Speedup	2.88	2.71

Table 3. Average time in seconds for one decomposition with at most two large primes. Here $p = 33554467 \approx 2^{25}$ and B is the bound determining 'small primes'.

$y^2 = x^7 + 26412x + 15471$ over $\mathbb{F}_p$			$y^2 = x^9 + 26412x + 15471$ over $\mathbb{F}_p$		
Method	$B = 2^{16}$	$B = 2^{17}$	Method	$B = 2^{18}$	$B = 2^{19}$
Our	0.0064	0.0033	Our	0.646	0.176
Gaudry	0.0343	0.0171	Gaudry	2.281	0.573
Speedup	5.35	5.18	Speedup	3.53	3.25

Table 4. 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'.

$y^2 = x^7 + 26412x + 15471$ over $\mathbb{F}_p$			$y^2 = x^9 + 26412x + 15471$ over $\mathbb{F}_p$		
Method	$B = 2^{18}$	$B = 2^{19}$	Method	$B = 2^{20}$	$B = 2^{21}$
Our	0.0134	0.0074	Our	3.16	0.96
Gaudry	0.1260	0.0642	Gaudry	15.65	4.22
Speedup	9.40	8.67	Speedup	4.95	4.39

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.

Acknowledgement

We would like to thank the reviewers of a previous version for providing comments which have helped in improving the paper.

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. Claus Diem and Emmanuel Thomé. Index calculus in class groups of non-hyperelliptic curves of genus three. *J. Cryptology*, 21(4):593–611, 2008.

4. Andreas Enge. Computing discrete logarithms in high-genus hyperelliptic Jacobians in provably subexponential time. *Math. Comput.*, 71(238):729–742, 2002.
5. Andreas Enge and Pierrick Gaudry. A general framework for subexponential discrete logarithm algorithms. *Acta Arithmetica*, 102:83–103, 2002.
6. Ralf Flassenberg and Sachar Paulus. Sieving in function fields. *Experimental Mathematics*, 8(4):339–349, 1999.
7. Gerhard Frey. How to disguise an elliptic curve (Weil descent). Talk at the 2nd Elliptic Curve Cryptography (ECC) Workshop, 1998.
8. 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.
9. 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.
10. 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.
11. 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.
12. Neal Koblitz. Elliptic curve cryptosystesm. *Math. Comp.*, 48(177):203–209, 1987.
13. Neal Koblitz. Hyperelliptic cryptosystems. *J. Cryptology*, 1(3):139–150, 1989.
14. Alfred Menezes, Yi-Hong Wu, and R. Zuccherato. An elementary introduction to hyperelliptic curves. Appendix in 'Algebraic Aspects of Cryptography' by Neal Koblitz, 1998.
15. 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.
16. 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.
17. Benjamin A. Smith. Isogenies and the discrete logarithm problem in Jacobians of genus 3 hyperelliptic curves. *J. Cryptology*, 22(4):505–529, 2009.
18. Richard G. Swan. Factorization of polynomials over finite fields. *Pacific Journal of Mathematics*, 12(3):1099–1106, 1962.
19. 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.
20. Magma v2.19 7. `http://magma.maths.usyd.edu.au/magma/`.
21. 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.