Experiments Using an Analogue of the Number Field Sieve Algorithm to Solve the Discrete Logarithm Problem in the Jacobians of Hyperelliptic Curves

Nigel P. Smart
Networked Systems Department
HP Laboratories Bristol
HPL-97-130
October, 1997

discrete logarithm, hyperelliptic curves, cryptography

In this paper we report on an implementation of the algorithm of Aldeman, De Marrais and Huang for the solution of the discrete logarithm problem on Jacobians of hyperelliptic curves. The method of Aldeman, De Marrais and Huang is closely related to the Number Field Sieve factoring method which leads us to consider a "lattice sieve" version of the original method.

Internal Accession Date Only

EXPERIMENTS USING AN ANALOGUE OF THE NUMBER FIELD SIEVE ALGORITHM TO SOLVE THE DISCRETE LOGARITHM PROBLEM IN THE JACOBIANS OF HYPERELLIPTIC CURVES

N.P. SMART

Abstract. In this paper we report on an implementation of the algorithm of Aldeman, De Marrais and Huang for the solution of the discrete logarithm problem on jacobians of hyperelliptic curves. The method of Aldeman, De Marrais and Huang is closely related to the Number Field Sieve factoring method which leads us to consider a "lattice sieve" version of the original method.

The supposed intractability of the discrete logarithm (DLOG) problem in the Jacobians of curves defined over a finite field can be used as the basis of public key cryptosystems. For curves of genus one, elliptic curves, this was proposed in [11] and such systems are now in use. For hyperelliptic curves the analogous system was proposed in [12]. The reason for preferring the use of Jacobians of curves as the underlying group, rather than finite fields, is due to the fact that there is no known subexponential algorithm for solving the DLOG problem in the Jacobian of a general curve. This should be contrasted to the case of finite fields where there exists conjectural subexponential time methods (based on the number field sieve factoring algorithm) to solve the DLOG problem. There are provably subexponential time methods for solving the discrete logarithm problem in finite fields, however in practice these are not as fast as the number field sieve algorithm.

In [1], Adleman, De Marrais and Huang (ADH), proposed a conjectural subexponential method for the DLOG problem in Jacobians of hyperelliptic curves of large genus. This method was based on the ideas of the function field sieve algorithm which can be used to solve the discrete logarithm problem in $\mathbb{F}_{2^n}$, [2]. The function field sieve is itself based on Pollard's Number Field Sieve, NFS, algorithm for factoring integers, [14].

The ADH method appears to be only of theoretical interest as for practical systems the genus is usually chosen to be small so that the underlying group operations can be performed quickly. Indeed the group is usually the genus one case of the group of points on an elliptic curve. There is however some, at least theoretical, interest in studying where the cross over point comes between the various methods to solve the DLOG problem such as the exponential methods of Pollard, Pohlig and Hellman and the method of ADH.

Recently Paulus, [16], and Flassenberg and Paulus, [8], have carried out such a comparison for imaginary quadratic function fields (or hyperelliptic curves with one ramified point above infinity). Flassenberg and Paulus did not, however, use the method of ADH directly. Instead they made use of the fact that hyperelliptic curves correspond to degree two function field extensions. Then using the analogy between

quadratic function fields and quadratic number fields they adapted the class group method of Hafner and McCurley, [10] (see also [6]). This combined with a sieving operation provided a working method which could be applied to hyperelliptic curves of small genus. It should be pointed out that although Flassenberg and Paulus did not solve discrete logarithm problems their methods are such that they can be easily extended so that they do.

The NFS algorithm can itself be used to compute class groups of arbitrary degree number fields. There are many similarities between the NFS method and that of Hafner-McCurley (and its generalizations). They are both relation generation methods, in that one chooses a set of factor base elements and then one uses some method to produce random relations between elements of the factor base. These relations are stored in a relation matrix which is finally used to compute the group structure or solve a DLOG problem by application of the Smith and Hermite normal form algorithms.

The main difference is in how the relations are computed. In the Hafner-McCurley method the relations are obtained by reducing random power products of the factor base. This can lead to the production of very dense relation matrices. In the NFS method the relations are obtained by sieving to find elements of smooth norm. This leads to generally sparse relation matrices.

Generalizing Hafner-McCurley to non-quadratic number fields can be done but is rather one has to be careful as to how one reduces the power products of elements of the factor base, see [6] for how this can be done. The NFS method clearly does not suffer from this problem and recent experiments have shown it to be rather efficient for computing class groups of general number fields, [4].

For function fields of degree greater than two the merits of the NFS type approach are immediate as reducing power products of divisors appears to be at least as complicated as the number field case. Hence, one could argue that, the analogue of Hafner-McCurley would be relatviely hard to implement for non-hyperelliptic curves, whilst the analogue of NFS would be relatively easy. Jacobians of non-hyperelliptic curves may be of interest in the distant future if a sub-exponential algorithm for solving the DLOG problem on Jacobians of hyperelliptic curves is ever found.

In this paper we describe an implementation of the analogue of NFS for hyperelliptic function fields. We essentially adapt the method of ADH, however we use the sieving techniques of Flassenberg and Paulus, [8], and, unlike ADH, we include the ramified primes in the factor base. We also make use of optimizations and experience from the factoring problem. For instance we make use of Pollard's Lattice Sieve, [17]. It remains to note that one can probably generalize the method of this paper to Jacobians of non-hyperelliptic curves. This should be especially easy when the curve has only one ramified divisor lying above infinity.

The author would like to thank S. Paulus for making available preprints of his work and some helpful electronic conversations.

1. The Basic Method

Let C denote a hyperelliptic curve of genus g defined over $\mathbb{F}_q$, with imaginary quadratic function field K. For simplicity we shall assume that the characteristic of $\mathbb{F}_q$ is not equal to 2. In particular this means that C can be given in the form

$$C : Y^2 = F(X)$$

where $F(X)$ is a square free, monic polynomial of degree $2g+1$, with coefficients in $\mathbb{F}_q$. A divisor on C will be called semi-reduced if it is effective and if when a point, P, occurs in the support of the divisor then the point $\tilde{P}$ does not, where $\tilde{P}$ denotes the image of P under the hyperelliptic involution. A semi-reduced divisor, which is defined over $\mathbb{F}_q$, can be represented by two polynomials $a, b \in F_q[x]$ which satisfy

i) $\deg b < \deg a$.
ii) b is a solution of the equation $b^2 - F \pmod{a}$.

Such a divisor we shall denote as $\mathrm{div}(a,b)$, and it represents the $\mathbb{F}_q$-rational divisor

$$\sum_{x_i} m_i(x_i, b(x_i))$$

where the sum is over all roots x_i of a, where each root has multiplicity m_i.

The Jacobian of C, which we shall denote by $J(C)$, can be represented uniquely by reduced divisors. A reduced divisor is a semi-reduced divisor as above but of degree less than or equal to g. Hence the polynomial a above will have degree less than or equal to g. The identity of the group law on $J(C)$ is given by $\mathcal{O} = \mathrm{div}(1,0)$, and addition can be performed using the well known algorithm of Cantor, [5]. Cantor's algorithm contains a subprocedure which takes a semi-reduced divisor and reduces it to a reduced divisor; we shall denote this procedure by **reduce** and will return to it below.

The Jacobian we recall is equal to the group of divisors of degree zero modulo principal divisors. An element of $J(C)$ we shall represent by $\mathrm{div}(a,b)$, which will represent the $\mathbb{F}_q$-rational degree zero divisor

$$\sum_{x_i} m_i(x_i, b(x_i)) - (\deg a)\infty,$$

with x_i and m_i as above.

As K is a quadratic function field, prime divisors, P, in K come in one of three varieties. We let p denote the prime of $\mathbb{F}_q[x]$ which lies below P in which case we have:

P ramifies
In this case p divides F and there is only one ramified prime divisor, P, lying above p. We shall denote this prime divisor by $\mathrm{div}(p,0)$.

P is inert
In this case p does not divide F and there is no solution to the equation

$$Y^2 \equiv F(x) \pmod{p}$$

in the field $L = \mathbb{F}_q[x]/(p)$. Whether such a solution exists can either be determined using a standard generalization of the legendre symbol or by factoring $Y^2 - F$ over the field L. Such prime divisors we shall ignore in our algorithm.

P splits
As in the inert case p does not divide F but now the equation

$$Y^2 \equiv F(x) \pmod{p}$$

has two solutions, r_1 and r_2 both of degree less than p. The prime, p, then splits into the two divisors

$$P = \mathrm{div}(p, r_1) \text{ and } \tilde{P} = \mathrm{div}(p, r_2).$$

The values of the polynomial r_1 (and hence r_2) can either be determined by factoring $Y^2 - F$ over the field $L = \mathbb{F}_q[x]/(p)$, or using an obvious generalization of the *ressol* algorithm of Shanks, see [6][Algorithm 1.5.1]

The method of ADH generates random elements of the function field of the form

$$f = a(x)y + b(x),$$

with coprime $a(x), b(x) \in \mathbb{F}_q[x]$. The method then tries to factor the divisor $\mathrm{div}(f)$ over a predetermined set of prime divisors (the factor base). In the original presentation the factor base is chosen to be the set of all split prime divisors of small degree in K. The small degree is the drawback to curves of small genus, for elliptic curves the factor base would essentially consist of half of the points on the curve over $\mathbb{F}_q$.

The decision as to whether an element of the required form factored over the factor base was decided, in [1], using the fact that in random polynomial time one can factor polynomials over finite fields. In the standard NFS factorizations are expensive so one replaces them by a sieving procedure. Factoring polynomials over finite fields is on the other hand cheap so for a complexity theoretic answer one does not need to use a sieving technique. However in practice a sieving technique for function fields, developed by Flassenberg and Paulus, has proved to be particularly useful.

Determining the prime divisor decomposition of the function f can be done via the following proposition, once we have found the factorization of $b^2 - a^2F$.

Proposition 1. *Let $a(x), b(x) = \mathbb{F}_q[x]$ be coprime polynomials, let f denote the function $a(x)y + b(x)$ and set*

$$N_f = N_{K/\mathbb{F}_q[x]}(a(x)y + b(x)) = b(x)^2 - a(x)^2F(x) = \prod_{i=1}^{r} p_i(x)^{m_i},$$

where $p_i(x) \in \mathbb{F}_q[x]$ are irreducible. Then $\mathrm{div}(f)$ *has only ramified or split primes in its support and we have*

$$\mathrm{div}(f) = \sum_{i=1}^{r} m_i \mathrm{div}(p_i, r_i) - (\sum_{i=1}^{r} m_i)\infty,$$

where r_i is the unique polynomial of degree less than the degree of p_i such that

$$a(x)r_i(x) + b(x) \equiv 0 \pmod{p_i(x)} \text{ or } -a(x)r_i(x) + b(x) \equiv 0 \pmod{p_i(x)}.$$

2. The Implementation

In this section we describe our basic implementation of the NFS-type method of ADH. In our description we assume that q denotes a rational prime, small changes need to be made when q is a prime power. We assume we are given two reduced

divisors D_1 and D_2 and we wish to determine a solution (if one exists) to the equation

$$D_2 = [m]D_1.$$

2.1. **Choosing the Factor Base.** We first enter into the factor base all prime divisors of K which lie in the support of our two divisors D_1 and D_2. We then add in all ramified divisors in K, although this is not done in ADH, we have found that this increases the yield of the sieve quite dramatically for very little extra work. Then we add in a random set of split prime divisors of various degrees up to g. A parameter B is entered to the factor base generation algorithm and we choose at most B degree one split prime divisors, $B/2$ of degree two, $B/3$ of degree three and so on. For the curves with Jacobians with orders greater than around ten million we also limited the number of large degree primes in the factor base, this was to ensure the size of the factor base was not too large and consisted mainly of small degree primes. Practical experiments have shown this to be a reasonable strategy for choosing the factor base, however more research needs to be carried out on what an optimal choice would be. We note in passing that if a prime divisor P is in the factor base then we need not include its conjugate $\tilde{P}$ in the factor base as we have the trivial relation;

$$P + \tilde{P} - 2\infty \equiv \mathrm{div}(1,0) = \mathcal{O} \text{ in } J(C).$$

In what follows we let N denote the number of elements in the factor base we have chosen.

2.2. **Setting up the matrix.** The idea of the NFS/ADH method is to generate random relations on the elements of the factor base and then perform an index calculus method to solve the discrete logarithm problem. The relations that we find are stored in the columns of a matrix with $N + 2$ rows. The first 2 rows correspond to the virtual elements of the factor base given by D_1 and D_2 whilst each of the last N rows correspond to an element in the factor base. We can instantly generate three relations;

1. Place a -1 in row one, column one and then enter the factorization of the divisor D_1 over the factor base in the relevant rows of column one.
2. We do the same in column two with the divisor D_2, except now we place the -1 in row two, column two.
3. In column three we can enter the free relation given by factorizing $\mathrm{div}(F)$ over the factor base. This gives us a relation between the ramified primes.

2.3. **The Sieving Condition.** We wish to find polynomials $a, b \in \mathbb{F}_q[x]$ such that the divisor of the function

$$f = ay + b$$

has support on the factor base only. Just as in the number field sieve we notice that if an element of the factor base lies in the support of f then we can derive a congruence condition between a and b. This we described in Proposition 1 above.

We organize a sieve in the function field case as is described in [8]. To every polynomial $g(x) \in \mathbb{F}_q[x]$ we associate a code given by $g(q) \in \mathbb{N}$. This is a unique integer which we use to index a sieving array. The sieving array is a two dimensional matrix indexed by the polynomial codes. Each array element is initialized at the start of the sieve to the value of

$$\deg(N_{K/\mathbb{F}_q[x]}(ay + b)) = \deg(b^2 - a^2F),$$

where a and b are the polynomials whose codes represent the row and column index of the array.

The sieve proceeds by taking every element, $P = \mathrm{div}(p, r)$, of the factor base in turn. We decrease the sieving array element by the degree of p if either

$$ar + b \equiv 0 \pmod{p}$$

or

$$-ar + b \equiv 0 \pmod{p}.$$

So we take every polynomial, $a_0 \pmod{p}$, in the a-direction and compute $b_0 = -a_0 r \pmod{p}$. We then subtract the degree of p from every array element which satisfies

$$(a, b) = (a_0 + e_1 p, \pm b_0 + e_2 p)$$

where e_1 and e_2 are polynomials. This can be done rather efficiently but care needs to be taken as to how we jump through the array. Details of how this can be done can be found in [8].

In our implementation we did not use polynomial arithmetic to compute the jumps. This would mean to deduce the next array element we would need to convert the current array position to polynomials, perform the polynomial addition or left shift and then convert back to two polynomial codes. It is far more efficient to implement polynomial addition and left shift directly on the codes themselves. A left shift is nothing but a multiplication by q, while an addition can be carried out efficiently by computing a base q expansion of the codes of the polynomials which need to be added.

2.4. **The Lattice Sieve.** One way to increase the yield of relations in the standard NFS algorithm is by replacing the standard sieve by the lattice sieve of Pollard, [17]. In the lattice sieve we first choose a special element of the factor base, $\mathrm{div}(p, v)$. We then sieve in the sub-lattice of the (a, b)-plane which is given by all values of (a, b) which give rise to a divisor whose support contains $\mathrm{div}(p, v)$.

We assume that $\mathrm{div}(p, v)$ is not a ramified divisor (a small change to the following needs to be made if it is). Under this assumption we know that $\gcd(p, v) = 1$ and so by the extended euclidean algorithm we can find polynomials s and t such that

$$sp + tv = 1.$$

The sublattice we are interested in are those values of (a, b) such that

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} -t & p \\ 1 & 0 \end{pmatrix} \begin{pmatrix} c \\ d \end{pmatrix}$$

where c and d are any two polynomials. For this any choice of (c, d) the pair (a, b) will always satisfy the sieving condition, as

$$\begin{aligned} av + b &= -vtc + pdv + c = c(1 - tv) + pdv \\ &= p(sc + dv) \equiv 0 \pmod{p}. \end{aligned}$$

If $f = ay + b$ is in this sublattice, called the (c, d)-plane from now on, then we know that p divides N_f. This should increase the chance of factoring N_f over the factor base.

The above basis of the lattice may not be suitable, as its elements may be too large. For real two dimensional lattices there is a reduction method, due to Gauss, which turns a large lattice basis into a small one. This is closely linked

to reduction theory of definite binary quadratic forms and general lattice basis reduction algorithms like LLL, [15].

In our situation we need a similar reduction procedure. For a vector $\mathbf{b}$ in our lattice we let $|\mathbf{b}|_\infty$ denote the maximum value of the degree of the constituent polynomials. Our reduction procedure then proceeds as follows,

---

2×2 matrix reduction algorithm

INPUT: Two basis vectors $\mathbf{b_1}$ and $\mathbf{b_2}$ which generate the lattice.
OUTPUT: Two basis vectors which are reduced.

1. Repeat
2. If $|\mathbf{b_1}|_\infty > |\mathbf{b_2}|_\infty$ then swap the vectors $\mathbf{b_1}$ and $\mathbf{b_2}$ around.
3. Set $d = \mathbf{b_1}^t\mathbf{b_1}$.
4. If $\deg d = 2|\mathbf{b_1}|_\infty$ then choose m and r such that
5. $\mathbf{b_1}^t\mathbf{b_2} = md + r$, $\deg r < \deg d$.
6. else choose m and r such that
7. $\mathbf{b_2}^{(1)} = m\mathbf{b_1}^{(1)} + r$, $\deg r < \deg \mathbf{b_1}^{(1)}$.
8. Put $\mathbf{b_2} = \mathbf{b_2} - m\mathbf{b_1}$.
9. Until $(|\mathbf{b_1}|_\infty \leq |\mathbf{b_2}|_\infty)$.

Clearly this algorithm will terminate and produce a sort of "reduced" basis if we can show the new value of $\mathbf{b_2}$, which we shall denote $\mathbf{b_2^*}$, satisfies

Lemma 2. *After executing Step* 8 *of the above algorithm we have*

$$|\mathbf{b_2^*}|_\infty \leq |\mathbf{b_2}|_\infty.$$

Proof. We know $|\mathbf{b_1}|_\infty \leq |\mathbf{b_2}|_\infty$ and $\Delta = \det(\mathbf{b_1}, \mathbf{b_2})$ satisfies $|\Delta|_\infty \leq |\mathbf{b_1}|_\infty + |\mathbf{b_2}|_\infty$. If we put $d = \mathbf{b_1}^t\mathbf{b_1}$ then there are two cases to consider:

i) $\deg d = 2|\mathbf{b_1}|_\infty$
Here we have

$$\begin{aligned}
\mathbf{b_2^*} &= \frac{1}{d}\left(d\mathbf{b_2} - md\mathbf{b_1}\right) \\
&= \frac{1}{d}\left(d\mathbf{b_2} + r\mathbf{b_1} - (\mathbf{b_1}^t\mathbf{b_2})\mathbf{b_1}\right) \\
&= \frac{1}{d}\begin{pmatrix} r\mathbf{b_1}^{(1)} - \mathbf{b_1}^{(2)}\Delta \\ r\mathbf{b_1}^{(2)} + \mathbf{b_1}^{(1)}\Delta \end{pmatrix}.
\end{aligned}$$

So

$$\begin{aligned}
|\mathbf{b_2^*}|_\infty &\leq \max(\deg(r\mathbf{b_1}^{(1)} - \mathbf{b_1}^{(2)}\Delta), \deg(r\mathbf{b_1}^{(2)} + \mathbf{b_1}^{(1)}\Delta)) - \deg d \\
&\leq \max(|\mathbf{b_1}|_\infty + \deg\Delta, \deg r + |\mathbf{b_1}|_\infty) - 2|\mathbf{b_1}|_\infty \\
&\leq \max(\deg\Delta - |\mathbf{b_1}|_\infty, 0) \leq |\mathbf{b_2}|_\infty.
\end{aligned}$$

ii) $\deg d \neq 2|\mathbf{b_1}|_\infty$
In this case, which can only happen when $p \equiv 1 \pmod 4$, we have that

$$\deg \mathbf{b_1}^{(1)} = \deg \mathbf{b_1}^{(2)} = |\mathbf{b_1}|_\infty.$$

Clearly we have

$$\deg \mathbf{b_2}^{*(1)} = \deg r < |\mathbf{b_1}|_\infty \leq |\mathbf{b_2}|_\infty.$$

For the other component of $\mathbf{b_2^*}$ we notice that

$$\mathbf{b_2^*}^{(2)} = \frac{1}{\mathbf{b_1}^{(1)}} \left(\Delta + r\mathbf{b_1}^{(2)} \right)$$

and so

$$\deg \mathbf{b_2^*}^{(2)} \leq \max(\deg \Delta, \deg(r\mathbf{b_1}^{(2)})) \leq |\mathbf{b_2}|_\infty.$$

□

For our purposes the reduction algorithm above is sufficient, however we note that Lenstra, [13], has given an analogue of the LLL algorithm for lattices over function fields. This would lead to a slightly better behaved basis but with the expense of slightly more work. Lenstra of course gives the algorithm for arbitrary dimensional lattices and not just those of dimension two.

We then perform a sieve as above in the (c, d)-plane. Suppose we have

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \begin{pmatrix} c \\ d \end{pmatrix}$$

and we wish to sieve with respect to an element of the factor base given by $\mathrm{div}(q, w)$. The condition in the (a, b)-plane is $aw + b \equiv 0 \pmod{q}$, there is also the condition $-aw + b \equiv 0 \pmod{q}$ but for clarity we look at one sign only. The condition in the (c, d)-plane becomes

$$\begin{aligned} aw + b &= (a_{1,1}c + a_{1,2}d)w + (a_{2,1}c + a_{2,2}d) = c(a_{1,1}w + a_{2,1}) + d(a_{1,2}w + a_{2,2}) \\ &= u_1 c + u_2 d \equiv 0 \pmod{q}. \end{aligned}$$

So we have three cases to consider:

$u_1 \equiv u_2 \equiv 0 \pmod{q}$
In this case every entry in the sieving array in the (c, d)-plane can be reduced by the degree of q.

$u_1 \not\equiv 0 \pmod{q}$ and $u_2 \equiv 0 \pmod{q}$
Here the sieving condition is that we can reduce all those elements in the sieving array whose position represent a polynomial, c, such that $c \equiv 0 \pmod{q}$.

$u_1 \not\equiv 0 \pmod{q}$ and $u_2 \not\equiv 0 \pmod{q}$
For every possible value of the polynomial $c_0 \pmod{q}$ in the c-direction we compute

$$d_0 = (-u_1 c_0 / u_2) \pmod{q}.$$

Then we need to reduce all array elements, by the degree of q, of the form

$$(c, d) = (c_0 + e_1 q, d_0 + e_2 q)$$

where e_1 and e_2 are polynomials. This can be done using the method of [8] mentioned above.

2.5. **Solving the DLOG problem.** When computing the group structure using a relation generation method, such as the one here, it is crucial that the factor base generates the whole group. It is also crucial that the production of relations proceeds until at least the resulting matrix has full rank. When solving a discrete logarithm problem we do not need to be quite so rigorous. Of course if our factor base and matrix satisfy the above conditions then we will be more than satisfied.

Suppose we at some point column reduce our matrix, using either the Hermite Normal Form algorithm or fraction free gaussian elimination. We assume that this algorithm produces an upper triangular matrix with the first column being non-zero.

If the only non-zero element, t, in the first column is in row one then this tells us that the divisor D_1 has order dividing t in $J(C)$. If the second column has only two non-zero entries which lie in rows one and two (which we shall call r and s) then our discrete logarithm problem can be solved from the linear equation

$$sm \equiv r \pmod{t}.$$

It is a trivial matter to deduce the solutions to this equation. Each solution can then be tested to see if it gives a solution to the discrete logarithm problem. There may be many such solutions but any one will do if, say, we are trying to solve a Diffie-Hellman type problem.

3. Numerical Results

The above method was implemented and we produced the run times in Table 1, which are all given in seconds. Note that these times appear much worse than the times needed by the Hafner-McCurley variant in [8]. In the ADH method to achieve the smallest run times we required much larger factor bases and sieving arrays than in the Hafner-McCurley variant. On the other hand the relation matrix we produced was sparse as opposed to dense. Hence for large scale implementation the method of ADH may be preferable.

Table 1. The standard method

		Time to produce			Size Of	
Genus	$\mathbb{F}_q$	Factor Base	Relation Matrix	HNF	Factor Base	Sieving Array
1	$\mathbb{F}_{11}$	0.05	0.34	0.04	7	4 x 15
2	$\mathbb{F}_{11}$	0.18	1.01	0.13	17	4 x 18
3	$\mathbb{F}_{11}$	1.74	1.8	0.53	41	4 x 400
4	$\mathbb{F}_{11}$	6.16	45.96	3.67	95	20 x 700
1	$\mathbb{F}_{101}$	0.34	5.43	0.47	37	5 x 45
2	$\mathbb{F}_{101}$	1.85	59.3	1.13	70	8 x 400

We found it difficult with the implementation to attempt problems in Jacobians in larger orders than the ones indicated above. It is for this reason that we attempted to modify the method to work better with quadratic function fields. The technique we used, which is also used in the method of Hafner-McCurley, and which is special for imaginary quadratic function fields, is that from a semi-reduced divisor we can quickly produce a reduced divisor using the function **reduce** mentioned earlier. As such this second method is a hybrid version of the ADH and Hafner-McCurley methods.

The sieving stage remained the same but we created more reports from the sieve by using the following strategy. Suppose using the sieve we know we can write the divisor of $f = ay + b$ as

$$\operatorname{div}(f) = D_1 + D_2 - (\deg D_1 + \deg D_2)\infty$$

where D_1 and D_2 are both semi-reduced, D_1 has support entirely in the factor base and the support of D_2 does not contain any intersection with the factor base. If the degree of D_2 is zero then we have a relation just as before. If however D_2 does not have degree zero then the standard algorithm will reject this value of a and b. However if D_1 and D_2 both have degree greater than g we can quickly proceed in the following way to hopefully produce a new relation:

We know D_1 already factorizes over the factor base. We can reduce the divisor D_2 to an equivalent divisor D_3 using the function **reduce**, so we have

$$\mathrm{div}(f) \equiv D_1 + D_3 - (\deg D_1 + \deg D_3)\infty.$$

We then may be lucky that D_3 factorized over the factor base, a property which can be detected by factoring a polynomial. We however found it more efficient to just use trial division by a subset of around half the elements in the factor base. Using this trick the number of reports increased dramatically, as Table 2 demonstrates.

TABLE 2. The hybrid-method

		Time to produce			Size Of	
Genus	$\mathbb{F}_q$	Factor Base	Relation Matrix	HNF	Factor Base	Sieving Array
1	$\mathbb{F}_{11}$	0.01	0.15	0.02	7	2 x 15
2	$\mathbb{F}_{11}$	0.08	0.45	0.08	15	2 x 50
3	$\mathbb{F}_{11}$	0.45	1.43	0.11	19	4 x 250
4	$\mathbb{F}_{11}$	0.94	7.31	0.35	34	5 x 400
5	$\mathbb{F}_{11}$	4.31	15.42	1.17	73	3 x 1000
6	$\mathbb{F}_{11}$	4.12	29.74	0.95	83	30 x 5000
7	$\mathbb{F}_{11}$	4.83	61.76	1.43	88	50 x 5000
8	$\mathbb{F}_{11}$	31.9	857.85	18.84	237	40 x 5000
9	$\mathbb{F}_{11}$	27.39	1822.48	108.24	282	50 x 5000
10	$\mathbb{F}_{11}$	26.29	3875.67	110.88	287	50 x 5000
1	$\mathbb{F}_{101}$	0.13	6.78	0.19	19	4 x 150
2	$\mathbb{F}_{101}$	1.06	23.64	1.1	63	10 x 500
3	$\mathbb{F}_{101}$	5.31	56.22	4.09	137	10 x 900
1	$\mathbb{F}_{1009}$	0.37	164.53	0.5	42	10 x 9000

4. CONCLUSIONS

Our numerical experiments have shown that the method of ADH is practical, when combined with the lattice sieve of Pollard and other tricks, to solve the discrete logarithm problem in the Jacobian of certain hyperelliptic curves. However these Jacobians do not have group orders anywhere near the size that would be required for use in a practical cryptosystem.

It appears that the analogue of the method of Hafner-McCurley is better suited to quadratic function fields than the analogue of the number field sieve method. This should come as no surprise as this is what happens for quadratic number fields. It remains an open question as to how efficient in practice is the NFS type method at solving the DLOG problem in Picard groups of non-quadratic function fields ?

At present this is only of academic interest as no cryptosystem has been proposed which makes use of non-hyperelliptic curves. Clearly the protocols go over to the

non-hyperelliptic case, the only problem being the existence of fast and efficient group law algorithms. For instance one could probably use the method without much alteration to solve DLOG problems in Jacobians of non-hyperelliptic curves of genus three with one $\mathbb{F}_q$-rational hyper flex. Such curves are given by equations of the form

$$y^3 + f_1 y + f_2 = 0$$

where f_1 and f_2 are polynomials in x of degree 2 and 4 respectively.

One should note that the algorithm used can only be proved to be of sub-exponential type when the genus is very large in comparison to the modulus. We also have not implemented some tricks which would allow the consideration of large groups; for instance a large prime variation or implementation in parallel over a network of computers. A major improvement would result from the development of algorithms to compute the Hermite Normal Form for use on large sparse integer matrices. If one was only interested in the group structure then there exist probabilistic algorithms to compute the Smith Normal Form of sparse integer matrices, [9], which would be of use.

References

[1] L. Adleman, J. De Marrais, and M.-D. Huang. A subexponential algorithm for discrete logarithms over the rational subgroup of the Jacobians of large genus hyperelliptic curves over finite fields. In [3], pages 28–40.

[2] L. Adleman. The function field sieve. In [3], pages 108–121.

[3] L.M. Adleman and M-D. Huang, editors. *ANTS-1 : Algorithmic Number Theory*. Springer-Verlag, LNCS 877, 1994.

[4] J. Buchmann and S. Neiss. Private communication. 1997.

[5] D.G. Cantor. Computing in the Jacobian of a hyper-elliptic curve. *Math. Comp.*, Vol 48, 95–101, 1987.

[6] H. Cohen. *A Course In Computational Algebraic Number Theory*. Springer-Verlag, GTM 138, 1993.

[7] H. Cohen, editor. *ANTS-2 : Algorithmic Number Theory*. Springer-Verlag, LNCS 1122, 1996.

[8] R. Flassenberg and S. Paulus. Sieving in function fields. *Preprint*, 1997.

[9] M. Giesbrecht. Probabilistic computation of the Smith Normal Form of a sparse integer matrix. In [7], pages 173–186.

[10] J.L. Hafner and K.S. McCurley. A rigorous subexponential algorithm for computation of class groups. *J. AMS*, Vol 2, 837–850, 1989.

[11] N. Koblitz. Elliptic curve cryptosystems. *Math. Comp.*, Vol 48, 203–209, 1987.

[12] N. Koblitz. Hyperelliptic cryptosystems. *J. of Cryptography*, Vol 1, 139–150, 1989.

[13] A. Lenstra. Factoring multivariate polynomials over finite fields. *J. of Computer and System Sciences*, Vol 30, 235–248, 1985.

[14] A.K. Lenstra and H.W. Lenstra, editors. *The development of the number field sieve*. Springer-Verlag, LNM 1554, 1993.

[15] A.K. Lenstra, H.W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, Vol 261, 515–534, 1982.

[16] S. Paulus. An algorithm of sub-exponential type computing the class group of quadratic orders over principal ideal domains. In [7], pages 243–257.

[17] J.M. Pollard. The lattice sieve. In [14], pages 43–49.

Hewlett-Packard Laboratories,Filton Road, Stoke Gifford, Bristol BS12 6QZ, U.K.
E-mail address: nsma@hplb.hpl.hp.com