Index calculus in class groups of non-hyperelliptic curves of genus 3 from a full cost perspective

– Extended Abstract –

Claus Diem

University of Leipzig

Abstract

We consider the discrete logarithm problem (DLP) in degree 0 class groups of non-hyperelliptic curves of genus 3 over finite fields $\mathbb{F}_q$. Using a recent index calculus algorithm with double large prime variation by the author, heuristically, one can solve this problem in a time of $\tilde{O}(q)$.

In this work, we study this problem from a full cost perspective. We argue that heuristically, using a 3-dimensional mesh, an instance of the problem can be solved with a full cost of $\tilde{O}(q^{17/12})$ (whereas the ρ-method has a full cost of $\tilde{O}(q^{3/2})$, and a full cost of $\Omega(q^{3/2})$ if the group order is nearly prime). In fact, we argue that for $n \longrightarrow \infty$, one can even solve $n^{1/6}$ instances of this DLP with $q \leq n$ with a full cost of $\tilde{O}(n^{17/12})$. Moreover, we argue that one can solve $n^{1/4}$ instances of this DLP with $q \leq n$ with a full cost of $\tilde{O}(n^{3/2})$.

1 Introduction and results

Additionally to the discrete logarithm problem (DLP) in elliptic curves and degree 0 class groups (a.k.a. Picard groups, Jacobian groups) of hyperelliptic curves, it has recently been proposed by various authors to use the DLP in degree 0 class groups of *non-hyperelliptic genus 3 curves* over finite fields as a primitive for public-key cryptographic protocols (see e.g. [2], [3], [4], [9], [11], [12]).

In [7], an index calculus algorithm with double large prime variation which is well-suited for the solution of the DLP in class groups of curves over finite fields represented by plane models of small degree has been introduced. Using that any non-hyperelliptic genus 3 curve is isomorphic to a plane quartic, the heuristic analysis of the algorithm given in [7] gives rise to:

Asymptotically for $q \longrightarrow \infty$, the DLP in degree 0 class groups of non-hyperelliptic genus 3 curves over $\mathbb{F}_q$ (given by a homogeneous equation of degree 4) can be solved in a time of $\tilde{O}(q)$.

Here, the $\tilde{O}$-notation captures logarithmic factors. The result is based on the computational model of a random access machine (RAM).

The full cost perspective

Given this result, one might conclude that non-hyperelliptic genus 3 curves do not provide a larger security level than genus 2 curves *over the same finite field* (but in the former case the key-length is 1.5 times larger). This conclusion would however be to hasty. Indeed, as argued by by Amirazizi and Hellman ([1]), Bernstein ([5]), Wiener ([13]) and other cryptographers, a better cost estimate than the mere running time of a cryptanalytic algorithm (on a RAM) might be its *full cost.*

As the algorithm presented in [7] has a storage requirement of $\Theta(q)$ field elements, a direct (unparallelized) application of the algorithm in [7] is comparable to an (unparallelized) application of Baby-Step-Giant-Step to genus 2 curves over the same field but not to the preferable ρ-method.

In this work, we first present a variant of the algorithm in [7] which also has a running time of $\tilde{O}(q)$ but has reduced storage requirements of $\Theta(q^{3/4})$ field elements. Then we adapt this algorithm to operate on a 3-dimensional mesh and analyze the resulting algorithm from a full cost perspective.

We are thereby concerned with the asymptotic full cost and not the actual cost required to build and run a device to solve a particular instance of the DLP.

We note that to our knowledge, no *formal* (mathematical) definition of "full cost" has yet been given. We choose a mesh-based architecture as a basis for the device because this architecture has an extremely simple structure, and it seems to be relatively easy to give a formal definition of "full cost" using this architecture.

The reader should keep the following semi-formal, intuitive, definition in mind:

Up to factors polynomial in the input length, the full cost *of a randomized algorithm running on a mesh, whose nodes each have a storage polynomial in the input length, is the number of nodes of the mesh multiplied with the expected running time.*

In the following, we always work with meshes whose nodes each have a storage polynomial in the input length.

The main results

The main results of this work are as follows:

Heuristic Result 1

- *One can perform the relation generation part of the algorithm on a 3-dimensional mesh with $O(q^{3/4})$ nodes and an expected running time* *of $\tilde{O}(q^{1/2})$, leading to a full cost of $\tilde{O}(q^{5/4})$.*

- *One can perform the linear algebra part of the algorithm on a 3-dimensional mesh with $\tilde{O}(q^{1/2})$ nodes and an expected running time of $\tilde{O}(q^{2/3})$, leading to a full cost of $\tilde{O}(q^{7/6})$.*

By multiplying the (dominating) storage requirement for the first part with the (dominating) running time of the second part, we obtain:

Heuristic Result 2 *Using a 3-dimensional mesh, one can solve the DLP in degree 0 class groups of non-hyperelliptic curves of genus 3 over $\mathbb{F}_q$ with a full cost of $\tilde{O}(q^{17/12})$.*

In comparison, the ρ-method has a full cost of $\tilde{O}(q^{3/2})$, and it has a full cost of $\Omega(q^{3/2})$ if the group order is nearly prime. Note that this means that we have obtained a reduction of the full cost comparable to an asymptotic reduction of the key-length by 1/18th.

An economic interpretation

It follows an *economic interpretation* of Heuristic Result 1.

Let us assume that meshes of sufficient size have already been built and are owned by a particular institution. Now the question arises what the *variable costs* for the solution of one particular instance of the DLP are. Under the assumption that the variable costs are proportional to the number of nodes times the running time, we can conclude that the variable costs are $\tilde{O}(q^{5/4} + q^{7/6}) = \tilde{O}(q^{5/4})$, whereas the ρ-method leads to variable costs of $\Omega(q^{3/2})$ if the group order is nearly prime.

The assumption that the variable costs are proportional to the number of nodes times the running time can be justified as follows: The main cost block for the variable costs is arguably the energy consumption. It is realistic to assume that (for fixed clock rate) the energy consumption is proportional to the number of nodes times the running time. (The administration costs are usually part of the fixed costs but the argument still holds if part of the administration costs are variable and grow proportionally to the size of the mesh times the operating time.)

Further results and another interpretation

If one performs the two parts of the algorithm on the same mesh, during the linear algebra part, only a tiny fraction of the mesh is actually used. Because of this it is possible to perform calculations of several instances of the DLP in (possibly) different groups in essentially the same full cost as one instance. This is the content of the first item of the following result. The second item can serve as a comparison with the ρ-method.

Heuristic Result 3

- *For $n \longrightarrow \infty$, one can perform $\lceil n^{1/6} \rceil$ instances of the DLP in degree 0 class groups of non-hyperelliptic genus 3 curves over $\mathbb{F}_q$ with $q \leq n$ with a 3-dimensional mesh with $\tilde{O}(n^{3/4})$ nodes and an expected running time of $\tilde{O}(q^{2/3})$, leading to a full cost of $\tilde{O}(n^{17/12})$.*
- *For $n \longrightarrow \infty$, one can perform $\lceil n^{1/4} \rceil$ instances of the DLP with $q \leq n$ with a 3-dimensional mesh with $\tilde{O}(n^{3/4})$ nodes and an expected running time of $\tilde{O}(n^{3/4})$, leading to a full cost of $\tilde{O}(n^{3/2})$.*

Note that in contrast, the calculation of $\lceil n^{1/4} \rceil$ DLPs in *different* groups of size $\Theta(n)$ and prime order has a full cost of $\Omega(n^{7/4})$ if one uses the ρ-method.

Conclusion

The results of this work confirm the implication of the results of [7] that the DLP in degree 0 class groups of non-hyperelliptic genus 3 curves is not as secure as the DLP in the groups of rational points of (carefully chosen) elliptic curves of the same group order over prime fields or over $\mathbb{F}_{2^n}$ (n a prime).

The results give further indication that – in the absence of any particular advantage of non-hyperelliptic genus 3 curves – the usage of such curves in cryptographic applications is questionable.

Related results

We mention some related results which can be obtained by adapting the algorithm in [10] to a mesh-based architecture.

For any fixed $g \geq 4$, the DLP in degree 0 class groups of any curve of genus g (represented by a plane model of bounded degree) over a finite field can on heuristic grounds be solved with an asymptotically lower full cost than possible with the ρ-method. This result even holds if one operates on a 2-dimensional mesh.

In contrast, for *hyperelliptic genus 3 curves* one obtains a full cost (for 3-dimensional meshes) which is asymptotically larger than that of the ρ-method. In fact, the full cost of the linear algebra part alone is larger than the full cost of the ρ-method.

This work

In this extended abstract we omit various details of the analysis. In particular, we omit the proof of Proposition 9 which is quite long and technical.

The work is organized as follows: In the next section we give a variant of the algorithm in [7] (in the special case of non-hyperelliptic genus 3 curves) which also has a heuristic expected running time of $\tilde{O}(q)$ but smaller storage requirements ($\Theta(q^{3/4})$ instead of $\Theta(q)$ field elements). In the third section, we discuss the operation of the algorithm on a 3-dimensional mesh, and we derive the heuristic results stated above.

We note that just as the analysis in [7], the analyses performed in this work are heuristic. However, the analysis of the variant of the algorithm we present here is "loaded with substantially more heuristic assumption" than the one in [7]. This applies in particular to the results on full cost we derive.

Acknowledgments

I thank W. König and E. Thomé for discussions and comments. I also thank the anonymous referees for comments.

2 Reduction of the storage requirements

This work should be seen as an addendum to [7], and should be read in conjunction with [7]. We use the same notation as in [7], and we assume that the reader is familiar with the index calculus algorithm in [7, Section 3]. We use this algorithm as a basis and point out the necessary changes in order to reduce the storage requirements.

We start off with a non-hyperelliptic genus 3 curves $\mathcal{C}$ over $\mathbb{F}_q$, given by a homogeneous (non-singular) equation of degree 4 of the form

$$F(X, Y, Z) = 0 \ ,$$

as well as two elements $a, b \in \mathrm{Cl}^0(\mathcal{C})$ (the degree 0 class group over $\mathbb{F}_q$) with $b \in \langle a \rangle$. The goal is to compute an $x \in \mathbb{N}$ with $x \cdot a = b$.

Let D_∞ be the intersection of the line $Z = 0$ with $\mathcal{C}$ (with multiplicities), let $\mathcal{F} = \{F_1, F_2, \ldots\} \subseteq \mathcal{C}(\mathbb{F}_q)$ be the *factor base*, and let $\mathcal{L} := \mathcal{C}(\mathbb{F}_q) - \mathcal{F}$ be the set of *large primes*. Recall that the main new idea of the algorithm in [7] is to generate relations of the form

$$[F_i] + [F_j] + [D_{i,j}] - [D_\infty] = 0 \tag{1}$$

(with $D_{i,j} \geq 0$) by intersecting the curve with lines passing through elements F_i, F_j of the factor base. Let us recall Definition 1 of [7] in our context.

Definition 4 A relation of the form (1) is called a *Full*, a *FP* or a *PP* relation if $D_{i,j}$ splits over $\mathbb{F}_q$ and contains no, one or two large primes respectively.

Recall also that in the algorithm in [7], FP and PP relations are stored in a so-called *graph of large prime relations*, which is a graph on the set $\mathcal{L} \dot\cup \{*\}$. Then at a later stage this graph is used to obtain recombined relations over the factor base.

We modify the algorithm in [7] in three ways to reduce the storage requirements.

1. We start off by searching for a multiple αa which is represented by a completely split divisor by trying different integers α. Likewise we search for a multiple βb which is represented by a completely split divisor. After having achieved this, fix the factor base $\mathcal{F} \subseteq \mathcal{C}(\mathbb{F}_q)$, thereby inserting the points necessary to express αa and βb. Apart from the initial two relation we only consider relations obtained by intersecting the curve with lines running through two points of the factor base.
2. We do not construct the full graph but only a tree, thereby discarding relations which would lead to other connected components than that of $*$. We stop the construction of the tree if we have $\lceil q^{3/4} \rceil$ vertices.
3. We do not enumerate the elements of $\mathcal{C}(\mathbb{F}_q)$.

Note that Modification 1 concerns mainly Step 5 of the algorithm in [7] but also affects the construction of the factor base (Step 1 in the algorithm in [7]). Modifications 2 mainly affects Step 2 of the algorithm in [7] but affects the construction of the factor base too: Indeed, in order to cope with the fact that the construction of the tree is less efficient than the construction of the full graph, we have to increase the factor base by a logarithmic factor. Modification 3 affects only the construction of the factor base.

Modification 2 means that we follow the approach of the "simplified algorithm" in [10]. It has already been pointed out in [10] that with this approach one can decrease the storage requirements. Modification 1 is made in order to further decrease the storage requirements.

We do not enumerate the elements in $\mathcal{C}(\mathbb{F}_q)$ in order to save the storage. Note that the enumeration was used to generate a factor base which is uniformly randomly distributed among all factor bases of the appropriate size. In order to generate the factor base in a "maximally random" way, we choose the (X, Z)-coordinates uniformly at random and then choose (if possible) one of the corresponding points on the curve uniformly at random.

The vertices of the tree are then stored / found via a balanced search tree.

We note again that each of these modifications means that the analysis will be loaded with "more heuristic assumptions" than the analysis in [7].

The algorithm is now as follows.

An algorithm with reduced storage requirements

Input: A non-hyperelliptic genus 3 curve $\mathcal{C}/\mathbb{F}_q$, given by a homogeneous equation $F(X, Y, Z) = 0$ of degree 4, the group order $\ell := \#\mathrm{Cl}^0(\mathcal{C})$ and two elements $a, b \in \mathrm{Cl}^0(\mathcal{C})$ with $b \in \langle a \rangle$.

Let $P_0 \in \mathcal{C}(\mathbb{F}_q)$ be a point used to represent the elements in $\mathrm{Cl}^0(\mathcal{C})$, that is, the elements in $\mathrm{Cl}^0(\mathcal{C})$ are represented by along P_0 maximally reduced divisors.

```
1. Compute multiples $\alpha a$ (with $\alpha \in (\mathbb{Z}/\ell\mathbb{Z})^*$) of $a$ until such a multiple $\alpha a$ is represented by an (along $P_0$ maximally reduced) completely split divisor. Analogously compute a multiple $\beta b$ of $b$ represented by a completely split divisor.

2. Choose a factor base $\mathcal{F} = \{F_1, F_2, \ldots\} \subseteq \mathcal{C}(\mathbb{F}_q)$ with $\lceil \log(q) \cdot q^{1/2} \rceil$ elements as described above.
   Insert the points $\mathcal{C}(\mathbb{F}_q)$ used to represent $\alpha a$ and $\beta b$ (including $P_0$) in the factor base.
   (If $\mathcal{C}(\mathbb{F}_q)$ has fewer elements, terminate.)

3. Construct a tree $T$ whose vertices are in $\mathcal{L} \,\dot{\cup}\, \{*\}$ as follows:
   Iterate over the pairs $(i, j)$ with $i < j \leq \#\mathcal{F}$:
       Let $F_i + F_j + D_{i,j}$ be the intersection divisor
       of the line through $F_i$ and $F_j$ with $\mathcal{C}$.
       If $D_{i,j}$ splits into points of $\mathcal{C}(\mathbb{F}_q)$, then
           if it defines an FP relation or it defines a PP relation and
           exactly one large prime lies in the tree,
               then insert the edge in the tree.
   Until the tree has $\lceil q^{3/4} \rceil$ vertices.

4. Construct a sparse matrix $R$ over $\mathbb{Z}/\ell\mathbb{Z}$ as follows:
   Store the relation for $\alpha a$ in the first row and the relation for $\beta b$ in second row.
   Iterate over the remaining pairs $(i, j)$ with $i < j$ not used in Step 3:
       Let $F_i + F_j + D_{i,j}$ be the intersection of the line through $F_i$ and $F_j$ with $\mathcal{C}$.
       If $D_{i,j}$ splits into elements of $\mathcal{F} \cup T$,
         Use the tree $T$ to substitute these elements
         by sums of elements of $\mathcal{F} \cup \{D_\infty\}$
         and store the relation obtained as a new for of the matrix $R$.
   Until $R$ has $\mathcal{F} + 1$ rows.

5. Compute a vector $\gamma = (\gamma_i)_i$ over $\mathbb{Z}/\ell\mathbb{Z}$ which is uniformly randomly distributed over $\ker(R^t)$. If $\gamma_2$ (the entry for the row for $\beta b$) is not invertible, compute some more relations as in the previous step and try again.

6. Let
       $$x := -\frac{\gamma_1 \alpha}{\gamma_2 \beta} .$$
   Output $x$.
```

If one runs out of pairs (i, j) before the calculation is finished, one should restart the algorithm (with another factor base chosen as described above the presentation of the algorithm).

Heuristic analysis

Clearly the algorithm above has a storage requirement of $\tilde{O}(q^{3/4})$. We will now argue that heuristically one should expect that with this algorithm one can also solve the DLP in an expected time of $\tilde{O}(q)$.

Provided that the group is cyclic, Step 1 can be performed with asymptotically 6 choices for α and β respectively. Heuristically, this also holds in general. (Note that this task is so easy because we include the points necessary to express αa and βb into the factor base rather than first fixing the factor base and then trying to express multiples of a and b over it.)

Step 2 has a running time of $\tilde{O}(q^{1/2})$.

We come to Step 3.

Note that as pointed out in [7], Heuristic Assumption 1 in Section 4 of [7] is satisfied in the special case of non-hyperelliptic genus 3 curves. Indeed we have (for $q \longrightarrow \infty$):

Proposition 5

- *The probability that a uniformly randomly chosen divisor in $|D_\infty|$ (over $\mathbb{F}_q$) is completely split is $\sim 1/4!$.*
- *If we choose $P_1, P_2 \in \mathcal{C}(\mathbb{F}_q)$ $(P_1 \neq P_2)$ uniformly at random, the probability that the intersection of the line through P_1 and P_2 with $\mathcal{C}$ defines a completely spit divisor is $\sim 1/2$.*

A *proof* of this proposition based on an effective Chebotarev theorem will be given in [8]. (Briefly, one fixes any point $P \in \mathcal{C}(\mathbb{F}_q)$ and studies the splitting behavior of points in $\mathbb{P}^1(\mathbb{F}_q)$ under the map $\mathcal{C} \longrightarrow \mathbb{P}^1_{\mathbb{F}_q}$ given by the complete linear system $|D_\infty - P|$.)

To analyze the growth of the tree, we rely on a heuristic comparison with the growth of a "random tree". The following proposition follows with the calculations in [10, Section 4]. (The first statement follows with the calculation in [10, 4.4], and the second statement follows with the calculation above Equation (2) in [10, 4.2]. Note that the notations in the calculations in [10] are different from the ones used below.)

Proposition 6 *Let $N \in \mathbb{N}$ and $c \in (0, 1]$. Let us consider the following random process:*

We consider a set of N elements ("vertices") with a distinguished element $*$. We construct a tree on this set as follows: We start with the tree consisting only of $*$ (the root) and insert edges by repeating the following procedure:

First we independently of the previous choices choose 1 with a probability of c and 0 with a probability of $1-c$. If we have chosen 0, we do nothing. If we have chosen 1, we uniformly randomly (and independently of the previous choices) choose an (unordered) pair of vertices. If this pair of vertices leads to an edge which is connected to exactly one point in the tree, we insert the edge in the tree.

Let $M \in \mathbb{N}$, $M \leq N$, and let us stop if we have obtained a tree with M vertices. Then the expected average distance of an element of the tree to the root of the tree is $\leq \log(M) + 1$.

Let c and $\alpha \in (0,1)$ be fixed and let us stop if we have obtained a tree with $\lceil N^{\alpha} \rceil$ vertices. Then for $N \longrightarrow \infty$, the expected stopping time is $\sim \frac{\alpha}{2c} \cdot \log(N) \cdot N$.

Remark 7 One can modify this process to model the occurrence of FP relations by increasing the probability that an edge connected to the root is drawn. Then the first statement of the proposition still holds, the expected stopping time for $N \longrightarrow \infty$ is $\lesssim \frac{\alpha}{2c} \cdot \log(N) \cdot N$.[1]

Our analysis relies on Proposition 5 and a *heuristic comparison* of the growth of the tree constructed in Step 3 of the algorithm with a "random tree" as described in Proposition 6.

Recall that we construct a tree whose vertices lie in $\mathcal{L} \,\dot\cup\, \{*\}$ and stop the construction if we have a tree with $\lceil q^{3/4} \rceil$ vertices. Moreover, heuristically and on the basis of Proposition 5, the probability that a pair of elements of the factor base leads to a split divisor $D_{i,j}$ is $\sim 1/2$.

We thus apply Proposition 5 with

$$N := \#\mathcal{L} \,\dot\cup\, \{*\} \sim q \,, \quad c \sim 1/2 \,, \quad \alpha := 3/4 \;.$$

Note that for these values we have

$$\frac{\alpha}{2c} \cdot \log(N) \cdot N \sim 3/4 \cdot \log(q) \cdot q \;.$$

Using this heuristic comparison, we obtain that heuristically one should expect that an expected number of $\sim 3/4 \cdot \log(q) \cdot q$ lines (unordered pairs of distinct elements of the factor base) should suffice to construct the tree in Step 3. (Note that the term "expected number" refers to the expected number of lines which have to be considered if one varies the factor base as described.)

We now come to Step 4.

As the tree has $\lceil q^{3/4} \rceil$ vertices, heuristically the probability that a divisor $D_{i,j}$ splits over $\mathbb{F}_q$ and consists of vertices of the tree or elements of the factor

[1] The symbol $\lesssim$ should be understood analogously to the symbol $\gtrsim$ defined in [7, Definition 3]: For sequences $(a_n)_{n\in\mathbb{N}}$ and $(b_n)_{n\in\mathbb{N}}$ of real numbers we write $a_n \lesssim b_n$ if $\limsup \frac{a_n}{b_n} \leq 1$.

base is $\sim 1/2 \cdot q^{-1/2}$. We conclude that heuristically, the expected number of distinct lines through elements of the factor base which have to be considered in Step 4 is $2q$.

All in all, we have argued that heuristically, the expected number of lines through pairs of points of the factor base one has to consider in Steps 3 and 4 is $\lesssim (3/4 \cdot \log(q) + 2) \cdot q$.

We have however $\sim 1/2 \cdot \log^2(q) \cdot q$ unordered pairs of elements of the factor base at our disposal. We therefore conclude with Markov's bound that with a very high probability we do not run out of pairs of elements of the factor base before reaching Step 5 of the algorithm. (Note that by our analysis a factor base of size $2\log(q)^{1/2} \cdot q^{1/2}$, say, should actually suffice in order that the expected number of restarts of the algorithm is in $O(1)$. We have chosen a larger factor base in order to have some "safety margin", and because logarithmic factors do not matter for our analysis.)

An important heuristic assumption is now that that one needs only slightly more rows than columns in R such that *the second row of the matrix is linearly dependent of the other rows*. We note that this is a mayor difference to the algorithm in [7]. Indeed, in [7], if the degree 0 class group is generated by a, we do not need to make any assumption after an appropriate tree has been constructed.

Under the assumption that the second row of the matrix is indeed linearly dependent on the other rows, for a uniformly randomly distributed vector $\gamma = (\gamma_i)_i \in \ker(R^t)$, the coefficient γ_2 is invertible with a probability of $\varphi(\ell)/\ell$.

On the basis of this heuristic analysis we conclude again that with a very high probability we do not run out of pairs of elements of the factor base.

Finally, heuristically and by comparison with Proposition 6, the expected average number of non-zero entries per row of the matrix R is in $\tilde{O}(1)$. Under this heuristic assumptions, one can derive a vector $\gamma = (\gamma_i)_i$ (over $\mathbb{Z}/\ell\mathbb{Z}$) such that γ_2 is invertible in an expected time of $\tilde{O}(q)$.

This completes the heuristic analysis, and we have the following heuristic result (based on a RAM).

Heuristic Result 8 *One can solve the DLP in degree 0 class groups of non-hyperelliptic genus 3 curves in a time of $\tilde{O}(q)$, using a storage of $\tilde{O}(q^{3/4})$.*

3 Parallelization

Let us fix the following notation: If $f, g : \mathbb{N} \longrightarrow \mathbb{R}$ are two functions, then $f = \tilde{\Theta}(g)$ means that $f(n) \leq p(\log(n)) \cdot g(n)$ and $g(n) \leq q(\log(n)) \cdot f(n)$ for all $n \gg 0$ with two polynomial functions p, q.

We come to the analysis of the algorithm from a full cost perspective. As announced in the introduction we analyze the operation of the algorithm on 3-dimensional meshes. We first derive Heuristic Result 1 on the relation generation and the linear algebra part (and thus establish our main result, Heuristic Result 2). Then we show how to obtain Heuristic Result 3.

We start off with a 3-dimensional mesh with $\lceil 2q^{3/4} \rceil$ nodes. On each node we generate relations and store vertices of the tree (for more details on the latter see the description of Step 3 below).

In order to avoid a "congestion" at the lower part of a search tree, the addresses of the vertices of the tree constructed in Step 3 are now found via a hash function rather than a search tree. More concretely, we fix a hash function $\mathcal{C}(\mathbb{F}_q) \longrightarrow \lceil 2q^{3/4} \rceil$. The vertices of the tree are stored on the nodes given by their hash values. Every time in Step 3 a point (vertex) is to be stored it is first checked if the corresponding node is not yet occupied and only if this is the case the vertex is stored. Otherwise the new edge of the tree is discarded. Note that as we stop with a tree of size $\lceil q^{3/4} \rceil$, heuristically the probability that the node where an incoming vertex is to be stored is still empty is always $\geq 1/2$.

Step 1 has a negligible running time.

Heuristically, the factor base as in Step 2 can be generated on a single node and stored in the mesh in a time of $\tilde{O}(q^{1/2})$. After this is done, $\tilde{\Theta}(q^{1/4})$ copies of each element are produced and stored in the mesh, such that on each node an element of the factor base is stored.

Step 3 of the algorithm is now modified as follows: Two elements F_i and F_j of the factor base are routed to each node (such that the unordered pairs $\{F_i, F_j\}$ are distinct for each node). This means that in total $2 \cdot \lceil q^{3/4} \rceil$ elements are routed. Heuristically this takes a time of $\tilde{\Theta}(q^{1/4})$. (We use that we have already stored $\tilde{\Theta}(q^{1/4})$ copies of each element of the factor base.) Then on each of the $\lceil q^{3/4} \rceil$ nodes a divisor $D_{i,j}$ is generated. The two points occurring in a split $D_{i,j}$ are routed to their destinations (i.e. to the nodes given by their hash values). At the destination it is checked whether the node is occupied. If this is the case, it is checked if the point equals the stored point, and then this information is routed back to the node where the point was generated. Heuristically, this procedure takes $\tilde{\Theta}(q^{1/4})$ operations.

A straightforward implementation of the insertion of the new edges into the tree would be as follows: For all $D_{i,j}$ which are split as $P + Q$ such that P is contained in the tree and no point having the hash value of Q is contained in the tree, Q is inserted in the tree. That is, the coordinates of P (on $\mathcal{C}$) are routed to the node given by the hash value of Q and the coordinates of Q are routed to the node given by the hash value of P.

This procedure would however lead to a potential problem during the later Step 4: The construction of the recombined relations might lead to a "congestion" at the root of the tree.

Because of this problem, we not only route the coordinates of P to the node given by the hash value of Q but in fact route the list of all vertices connecting P to the root to the node given by the hash value of Q. Note that we assumed from the beginning one that the storage of each node is polynomial in $\log(q)$ (see the "informal definition" of full cost in the introduction). This means that we have to guarantee that the lengths of these lists are polynomial in $\log(q)$, or with other words that the maximal distance of an element in the tree to the root is polynomial in $\log(q)$. We discuss this point below (see Conjecture 11) and make the heuristic assumption that this is the case.

Then again heuristically, the routing takes $\tilde{\Theta}(q^{1/4})$ operations.

Recall that we have shown in the previous section that heuristically, one can – with an unparallelized algorithm – construct a tree of size $\lceil q^{3/4} \rceil$ by considering $\tilde{O}(q)$ lines. Assuming that this construction can be parallelized in a manner which is "optimal except polynomial factors", we obtain that we have to apply the above procedure $\tilde{O}(q^{1/4})$ times to construct a tree of appropriate size, leading to a running time of $\tilde{O}(q^{1/2})$.

Under the assumption that the growth of the tree can be modeled by comparison with a random tree, this is indeed the case. Indeed, we have the following proposition. As mentioned in the introduction, the proof of this proposition is long and technical and is omitted here.

Proposition 9 *Let $\beta < 1$ be fixed.*

Let V be a set with N elements and a distinguished element $*$. Let us consider the following random process to construct a tree whose vertices are contained in V.

At time 0, the tree consists merely of the root $*$.

Let us assume that after the i-th step we have already constructed a tree T_i. Then we choose $\lceil N^{\beta} \rceil$ unordered pairs of elements from V $e_{i+1,j}$ $(j = 1, \ldots, \lceil N^{\beta} \rceil)$ uniformly and independently at random.

We define T_{i+1} as follows: For every $P \in V - T_i$ which is connected to T_i via an $e_{i+1,j}$, we choose the smallest j such that $e_{i+1,j}$ connects T_i and P and insert $e_{i+1,j}$ into the tree.

Let $M \leq N$ and let us stop have obtained a tree of size $\geq M$. Then the expected average distance of an element of the tree to the root of the tree is $\leq \min\{\log(M + N^{\alpha}) + 1, \log(N) + 1\}$.

Let us stop if we have obtained a tree which contains all elements from V. Then for $N \longrightarrow \infty$, the expected stopping time is in $\tilde{O}(N^{1-\beta})$.

Remark 10 Note also that the second statement also holds if we stop if the tree has reached an arbitrary number of elements from V.

The interpretation of this proposition is as follows: Again let $N := \#\mathcal{L} \,\dot{\cup}\, \{*\} \sim q$, and let $\beta := 3/4$. Then for each i, j, the $e_{i,j}$ represents a new pair of large primes which is obtained at time i by the j^{th} node. For simplicity we ignored the fact that heuristically only $\sim 1/2$ of the lines lead to a pair of large primes. We also ignored FP relations. The tree is constructed in such a way that if two nodes at the same moment of time generate pairs of vertices leading to a new edge of the tree which would end at the same vertex, then the pair of vertices coming from the node with the smaller number is used to generate a new edge of the tree.

As stated above, we need a result on the maximal distance of a vertex in the tree to its root. Again we would like to make an argument which relies on a comparison with a "random tree". This time, we are however not aware of an appropriate result. On the basis of Propositions 6 and 9 and because of similar results on random graphs in the literature (cf. e.g. [6]), we make the following conjecture.

Conjecture 11 *In the context of Propositions 6 or 9, let us stop the construction of the tree if all N elements are exhausted. Then with a probability converging to 1 for $N \longrightarrow \infty$, the maximal distance to the root is in* $\Theta(\log(N))$*.*

Remark 12 For our purposes it would suffice if for $N \longrightarrow \infty$, with probability converging to 1, the maximal distance to the root is in $\tilde{O}(\log(N))$.

We conclude that indeed heuristically we can perform Step 3 of the algorithm with a running time of $\tilde{O}(q^{1/4} \cdot q^{1/4}) = \tilde{O}(q^{1/2})$.

We now come to Step 4. We proceed similarly to Step 3: On each node we generate divisors $D_{i,j}$. The two points of each split divisor are routed to its destinations. If they occur in the tree, they are substituted with the corresponding addition chain of elements of the factor base which is routed back to the node where $D_{i,j}$ has been generated. Then they are routed to a part of the mesh where the linear algebra is going to be performed. Heuristically, this takes $\tilde{O}(q^{1/4})$ operations. On the basis of the arguments of the previous section, we have to consider $\sim 2q$ lines in total, leading to a running time of $\tilde{O}(q^{1/4} \cdot q^{1/4}) = \tilde{O}(q^{1/2})$.

So far we have not discussed the size of the factor base. By Proposition 9 it is reasonable to assume that a factor base of size $\tilde{O}(q^{1/2})$ suffices. More explicitly, let $\kappa > 0$ be such that for $\alpha = 3/4$ in Proposition 9 the expected stopping time is in $O(\log(q)^{\kappa} \cdot q^{1-\alpha})$. Note that this means that the expected number of pairs of vertices $e_{i+1,j}$ considered is in $O(\log(q)^{\kappa} \cdot q)$. If we then choose a factor base of size $\log(q)^{\kappa} \cdot q^{1/2}$, similarly to Section 2 we have a "safety margin" of $\Omega(\log(q)^{\kappa/2})$.

For the linear algebra, one can use for example a sorting algorithm as described in [5]. We note that in [5], the description of the algorithm is only done over $\mathbb{F}_2$. The generalization to other fields is however straightforward. (In the notation of [5], one has to replace the j's by (j, v_j) and the $(i,j)'s$ by $(i, j, A_{i,j})$. Then after the first snakelike sorting one has to continue by replacing the $(j, v_j)'$s by $(i, A_{i,j} \cdot v_j)$ for $A_{i,j} \cdot v_j \neq 0$. After the second snakelike sorting one has to perform the additions $\sum_j A_{i,j} \cdot v_j$.)

As the factor base has a size of $\tilde{\Theta}(q^{1/2})$ elements, the linear algebra then has a running time of $\tilde{\Theta}(q^{2/3})$, leading to a full cost of $\tilde{\Theta}(q^{7/6})$.

We have now derived Heuristic Results 1 and 2. It remains to argue that Heuristic Result 3 holds too.

Let us start with the first of the two results: We are given some natural number n and $\lceil n^{1/6} \rceil$ instances of the DLP in degree 0 class groups of some genus 3 curves $\mathcal{C}$ over $\mathbb{F}_q$ with $q \leq n$.

We again fix a 3-dimensional mesh with $\lceil 2q^{3/4} \rceil$ nodes. We perform the relation generation for the $\lceil n^{1/6} \rceil$ instances we want to consider in sequence. Each time, we store the matrix in a different cube inside the mesh. Note heuristically, each cube has a size of $\tilde{O}(q^{1/2})$, thus all cubes together have size $\tilde{O}(q^{2/3})$. This means that we can expect that we do not run out of storage to store all linear algebra problems. On the basis of the analysis above, the relation generation can be achieved in a time of $\tilde{O}(n^{1/6} \cdot n^{1/2}) = \tilde{O}(q^{2/3})$. After this, we solve all linear algebra problems in parallel. The expected time for this is again in $\tilde{O}(n^{2/3})$.

This gives the first result.

Now we assume we are given $\lceil n^{1/6} \rceil$ instances of the DLP.

We again use a mesh as above. We start off by performing the relation generation in sequence, and again we store the linear algebra problems in cubes. After we have exhausted the storage, we solve all linear algebra problems. We repeat the procedure until we have solved all $\lceil n^{1/4} \rceil$ instances.

The heuristic expected running time for the relation generation is now in $\tilde{O}(q^{3/4})$. Heuristically, we can expect that we can solve $\tilde{\Theta}(q^{1/4})$ linear algebra problems in parallel. This means that the solution of all $\lceil q^{1/4} \rceil$ linear algebra problems takes a time of $\tilde{O}(q^{2/3})$. The running time is thus dominated by the time of $\tilde{O}(q^{3/4})$ for the relation generation.

References

[1] H. Amirazizi and M. Hellman. Time-Memory-Processor Trade-Offs. *IEEE Transactions on Information Theory*, IT-34(3):505–512, 1988.

[2] A. Basiri, A. Enge, J.-C. Faugère, and N. Gürel. Implementing the Arithmetic of $C_{3,4}$-Curves. In *Algebraic Number Theory – ANTS VI*, LNCS, pages 87–101, Berlin, 2004. Springer-Verlag.

[3] A. Basiri, A. Enge, J.-C. Faugère, and N. Gürel. The arithmetic of Jacobian groups of superelliptic cubics. *Math. Comp.*, 249:389–410, 2005.

[4] M. Bauer, E. Teske, and A. Weng. Point counting on Picard curves in large characteristic. *Math. Comp.*, (74):1983–2005, 2005.

[5] D. Bernstein. Circuits for integer factorization: a proposal. available under cr.yp.to/papers/nfscircuit.pdf, Nov. 2001.

[6] F. Chung and L. Lu. The diameter of random sparse graphs. *Adv. in Apply. Math.*, 26:257–279, 2001.

[7] C. Diem. Index Calculus with Double Large Prime Variation in Class Groups of Plane Curves of Small Degree. Submitted, available under http://www.math.uni-leipzig.de/∼diem.

[8] C. Diem and E. Thomé. Index Calculus in Class Groups of Non-Hyperelliptic Curves of Genus 3. Forthcomming.

[9] S. Flon and R. Oyono. Fast arithmetic on Jacobians of Picard curves. In *Advances in Cryptology — PKC 2004*, volume 2947 of *LNCS*, pages 55–68, Berlin, 2004. Springer-Verlag.

[10] P. Gaudry, E. Thomé, N. Thériault, and C. Diem. A double large prime variation for small genus hyperelliptic index calculus. 2005. Accepted for publication in Mathematics of Computation, available under http://eprint.iacr.org/2204/153.

[11] K. Koyke and A. Weng. Construction of CM-Picard curves. *Math. Comp.*, 72:499–518, 2005.

[12] A. Weng. Generation of random Picard curves for crytography. 2005. accepted for publication in Designs, Codes and Cryptography.

[13] M. Wiener. The Full Cost of Cryptanalytic Attacks. *J. Cryptology*, 17(2):105–124, 2004.