Tatra Mt. Math. Publ. **45** (2010), 161–172

TATRA MOUNTAINS
Mathematical Publications

ON THE USE OF THE LATTICE SIEVE IN THE 3D NFS

Pavol Zajac

ABSTRACT. An adaptation of the Number Field Sieve (NFS) algorithm to solve a discrete logarithm problem in degree 6 finite fields (DLP6) requires a modified sieving procedure to find smooth elements of the three dimensional sieve space. In our successful solution [P. Zajac: *Discrete Logarithms and Degree Six Numbere Field Sieve: A practical Approach.* VDM Verlag Dr. Müller, Saarbrücken, 2009] we have used a modified line sieving to process a box-shaped region using a large factor base. In this contribution, we compare the results with an alternative approach based on the lattice sieving, which was used in most of the classical factorization and DLP record solutions. Results indicate that this approach does not scale to the 3D-case, making DLP6 more difficult in practice than comparable classical DLP cases.

1. Introduction

The Number Field Sieve (NFS) can be used to solve the integer factorization problem [6] and the discrete logarithm problems in finite fields [4] in a subexponential time. In 2008 we were able to solve a specific instance of the discrete logarithm problem in degree six finite field with approximately 2^{240} elements [12]. Our own custom implementation of the (NFS) method was used, because a special 3D sieving algorithm is required to apply the NFS to the degree six finite fields (within the range of parameters of computationally feasible instances). We are not aware of any (publicly available) solvers of this type, although the idea behind multidimensional sieving was already formulated by Joux et al. in 2006 [4].

In our "record solution" we have solved DLP in the degree 6 field with characteristic $p = 1081034284409$, which is a 40-bit prime number (the field size is

2010 Mathematics Subject Classification: 94A60.
Keywords: algebraic numbers, number field sieve, lattice sieve.
This material is based upon work supported under the grant NIL-I-004 from Iceland, Liechtenstein and Norway through the EEA Financial Mechanism and the Norwegian Financial Mechanism.

a 240-bit number). The smoothness bound was set to $B = L_{p^6}\left(1/3; (8/9)^{1/3}\right) \approx 6532326$, leading to a sieve with a factor base consisting of 893707 elements (degree 1 ideals with norm above 128). We have sieved the region with approximately 2^{40} points in 3 days on 8 computers in parallel. After the post-processing we have obtained a system of 1077984 linear equations in 854821 unknowns (modulo an 80-bit prime). The system was reduced to a denser and smaller one (226059 equations in 223474 unknowns) using a structured Gaussian elimination, which was solved by the Lanczos algorithm in nearly 13 days on a single computer.

To extend our computations to a larger field, we need to increase the size of smoothness bound B along with the sizes of the factor base and the sieving region. The computational time used by the "record solution" is not very large, thus we have some freedom in the time complexity area. Moreover, the computation can be easily distributed among independent computational nodes. However, the sieve requires a lot of memory on a single node to store the whole factor base and necessary data structures. It is also difficult to process and compute the solution of such a large system of linear equations.

It can be noted that parameters of our solution are comparable to experiments in a much larger finite field of degree one [3], and similar holds for factorization results [1]. The main difference is the use of higher degree sieve polynomials, and the application of the 3D sieve. Furthermore, in our experiments we only used a 3D sieve based on the line sieve algorithm without the use of large primes or lattice sieving (as is usual in the classical NFS). To extend our results we wanted to investigate further these techniques, and their possible application in our siever. As we show later, the use of these techniques is limited, and their possible application would require a completely different approach from the existing solutions.

2. Preliminaries

The basic Discrete Logarithm Problem (DLP) can be defined in a number theoretic sense as follows: Given a prime p, and two numbers a, b such that $a \equiv b^x \pmod{p}$, where x is an integer, find x. There exist fast algorithms to compute modular exponentiation. Thus, if we are able to find difficult instances of the DLP, we get a one-way function which can be (and is) used to build asymmetric cryptosystems. The definition of the DLP can be extended to any cyclic group, as follows:

Definition 1. Let $(G, \cdot)$ be an arbitrary finite cyclic group of order n. Let α be the generator of G. The *generalized discrete logarithm problem* is the following: Given an element $\beta \in G$, find the unique integer x, $0 \leq x < n$, such that $\alpha^x = \beta$.

The integer x is called the discrete logarithm of β to the base α, denoted by $x = \log_\alpha \beta$.

Let p be a (medium sized) prime, and let $\mathbb{F}_{p^6}$ denote a finite field of degree six with p^6 elements. Let G be a subgroup of the multiplicative group of $\mathbb{F}_{p^6}$ with large prime order q. According to the present state of the art we can solve the DLP in G (denoted by DLP6) either with generic algorithms with the exponential complexity $O\left(q^{1/2}\right)$, or with the modified version of the NFS algorithm with the subexponential complexity $O\left(L_{p^6}(1/3, c)\right)$ [4], where

$$L_q(\alpha; c) = \exp\left(c(\log q)^\alpha (\log\log q)^{1-\alpha}\right).$$

We use a very specific case of degree six fields because of two main reasons. Our primary motivation was the solution of the so called XTR-discrete logarithm problem, which is the basis of the security of the XTR based cryptosystems [7]. The solution of this problem can be found, if we are able to find the solution of the DLP6. Other motivation is the specific parameter setting and implementation challenges to solve the DLP6 with NFS. In the asymptotic complexity estimates of the NFS [2], [4] the fixed degree of the finite field does not make a difference as the problem size grows to infinity. However, the instances of DLP6 that can be solved in the present require a different handling of the polynomial selection, and specific sieving algorithms, which seem more difficult in practice than the classical NFS [12].

The general NFS algorithm to solve DLP (in the field $\mathbb{F}_{p^d}$ of low degree d) is as follows: Let $\alpha, \beta \in \mathbb{C}$ be the roots of two distinct monic polynomials $f, g \in \mathbb{Z}[x]$ irreducible over $\mathbb{Z}$. Then $K_1 = \mathbb{Q}(\alpha)$, $K_2 = \mathbb{Q}(\beta)$ are two algebraic number fields. Let $\mathcal{O}_K$ denote a ring of integers of the field K. Let t be a common root of f, g in $\mathbb{F}_{p^d}$. Then there exist two homomorphisms $\phi\colon \mathcal{O}_{K_2} \to \mathbb{F}_{p^d}$, and $\psi\colon \mathcal{O}_{K_2} \to \mathbb{F}_{p^d}$, defined by sending α, resp. β, to t.

Let g be a generator of $G = \mathbb{F}^*_{p^d}$ and let q be a (large) prime dividing order of G. Let algebraic number $\xi \in K_1$ be B-smooth, i.e., the corresponding principal ideal can be factored to ideals with norm below B. Furthermore, let the prime ideal decomposition of this principal ideal be:

$$(\xi) = \prod \mathfrak{p}_j{}^{v_j}.$$

Let $\pi_j \in \mathfrak{p}_j$, and let h be a class number of K. Using Schirokauer's logarithmic maps λ [9], we can transform this equation to

$$\log_g(\phi(\xi)) \equiv \sum_{j=0}^{r} \lambda_j(\xi)\Lambda_j + \sum_j v_j x_j \pmod q, \tag{1}$$

where $\Lambda_j = \log_g \phi(\omega_j)$ is an unknown "virtual logarithm" of the unit ω_j, and $x_j = h^{-1} \log_g \phi(\pi_j)$ is an unknown "virtual logarithm" of prime ideal $\mathfrak{p}_j$.

Let $\xi_1 \in \mathbb{Z}_{K_1}$ and $\xi_2 \in \mathbb{Z}_{K_2}$ be two B-smooth algebraic numbers, and let $\phi(\xi_1) = \phi(\xi_2)$. We call (ξ_1, ξ_2) a smooth pair. Using homomorphisms ϕ, ψ and equations (1), we can write

$$\sum_{j=0}^{r_1} \lambda_j^{(1)}(\xi_1)\Lambda_j^{(1)} + \sum_j v_j^{(1)} x_j^{(1)} \equiv \sum_{j=0}^{r_2} \lambda_j^{(2)}(\xi_2)\Lambda_j^{(2)} + \sum_j v_j^{(2)} x_j^{(2)} \pmod{q}, \qquad (2)$$

with unknown “virtual logarithms” $\Lambda_j^{(1)}, \Lambda_j^{(2)}, x_j^{(1)}$, and $x_j^{(2)}$. We call any equation in the form (2) a smooth equation. In an efficient NFS implementation smooth equations are found by the application of the sieving algorithm (see Section 3).

A set of all prime ideals in $\mathbb{Z}_{K_1}$, and $\mathbb{Z}_{K_2}$ respectively, lying over primes $p_j < B$, is called an (algebraic) factor base. If the cardinality of the factor base is $c_1 + c_2$, we can have at most $C = c_1 + c_2 + r_1 + r_2$ unknown “virtual logarithms” in any smooth equation. If we are able to find $R > C$ linearly independent smooth equations (in the so called sieving phase), we can try to find a non-trivial solution of the corresponding linear system (the linear algebra phase). By substituting to equations (1) we can compute logarithms of the corresponding elements of $\mathbb{F}_{p^d}^*$. Using the descent method [2], [4],then we can compute logarithms of other elements of $\mathbb{F}_{p^d}^*$ (with complexity lower than that of sieving and linear algebra steps).

The most important parameter of the algorithm is the size of the smoothness bound B. For the asymptotically optimal performance of DLP6 solver, B should be chosen near $L_{p^6}\left(1/3, (8/9)^{1/3}\right)$. The complexity of the algorithm is then proportional to B^2. Moreover, if the bound B is chosen incorrectly (both too small and too large), we cannot find enough linear equations, and the algorithm terminates without success. It should be noted, that in the reported NFS computations [1] the actual bound used is much smaller than the one given by the asymptotic estimate, but the experiments use semismooth numbers with a higher bound for a single large factor. However, in our experiments we were only able to complete the linear system with B as high as prescribed by the asymptotic estimate.

To provide more background on the sieve algorithm, we need to provide some basic facts from the algebraic number theory. Let $f(x)$ be a monic irreducible polynomial over $\mathbb{Z}$ with (a complex) root α. Then $\mathbb{Z}[\alpha]$ is a $\mathbb{Z}$-module (lattice) with dimension $\deg f$. Every element of $\mathbb{Z}[\alpha]$ is an algebraic integer in the field $K = \mathbb{Q}(\alpha)$. Let p_i be a prime number and let $f(x) = \prod_j f_{i,j}(x) \pmod{p_i}$.[1] Then $\mathfrak{p}_{i,j} = p_i\mathbb{Z}[\alpha] + f_{i,j}(\alpha)\mathbb{Z}[\alpha]$ are prime ideals (of $\mathbb{Z}[\alpha]$) of degree $\deg f_{i,j}$. We say that $\mathfrak{p}_{i,j}$ lies over p_i. Prime ideal $\mathfrak{p}_{i,j} \subset \mathbb{Z}[\alpha]$ is also a $\mathbb{Z}$-module whose basis are

[1]In practice, situation is similar if $f(x)$ has a multiple root modulo p_i. But this case is more complicated from the theoretical point of view, so we do not go into the details.

the entries of

$$\begin{pmatrix} p_i & 0 & 0 & \dots & 0 & \dots \\ 0 & p_i & 0 & \dots & 0 & \dots \\ \vdots & \vdots & \ddots & \dots & \vdots & \dots \\ a_0 & a_1 & \dots & a_e & 0 & \dots \\ 0 & a_0 & a_1 & \dots & a_e & \dots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix} \cdot \begin{pmatrix} 1 \\ \alpha \\ \vdots \\ \alpha^e \\ \alpha^{e+1} \\ \vdots \end{pmatrix},$$

where $e = \deg f_{i,j}$, and $f_{i,j}(x) = \sum a_k x^k$. The norm of $\mathfrak{p}_{i,j} = p^e$ divides the norm of any algebraic integer $\xi \in \mathfrak{p}_{i,j}$. Let q denote the largest prime factor of the norm $N(\xi)$. Algebraic integer ξ is B-smooth, if $q < B$. This also means, that ξ belongs to ideals which lie over primes $p_i < B$ that divide $N(\xi)$ (and the principal ideal generated by ξ is in the intersection of these prime ideals).

Let $\mathcal{B}$ be a set of all ideals lying over primes $p_i < B$. Let $\mathcal{S} \subset \mathbb{Z}[\alpha]$ be a chosen set of points. We call $\mathcal{S}$ a sieving region. In our case it is a bounded three-dimensional sublattice of $\mathbb{Z}[\alpha]$. For each $\mathfrak{p}_i \in \mathcal{B}$ let us define a function

$$v_i \colon \mathcal{S} \to \mathbb{R}, v_i(\xi) = \begin{cases} \log p_i & \text{if } \xi \in \mathfrak{p}_i, \\ 0, & \text{otherwise.} \end{cases}$$

Furthermore, let

$$v \colon \mathcal{S} \to \mathbb{R}, v(\xi) = \frac{\sum v_i(\xi)}{\log N(\xi)}.$$

For each ξ we have $v(\xi) \leq 1$. The equality holds for B-smooth algebraic integers ξ with squarefree norms, thus v can be used to detect all such integers from $\mathcal{S}$ (if we can evaluate the map efficiently). We can also use v in the cases when we want to consider smooth algebraic integers with square factors, or when we exclude some of the prime ideals from $\mathcal{B}$. In these cases, smooth algebraic integers tend to have higher values of $v(\xi)$ than non-smooth ones (although it is not exactly 1).

To compute v, we use an incremental algorithm. We start with assigning $v(\xi) = 0$ for each ξ. Then for each $\mathfrak{p} \in \mathcal{B}$ we increment value of v for each $\xi \in \mathfrak{p} \cap \mathcal{S}$. In the next section we describe the sieving in a more algorithmic way.

3. Description of sieving algorithms

3.1. 3D line sieving

In our original experiments we have used a 3D variant of the line sieving algorithm. It is derived from the original generalization of the line sieving as presented in [10]. The original idea was to recursively contract the dimension of the sieve space along with selection of the factor base ideal with points in the contracted space. In a fixed 3D variant, a more efficient version was presented in [11], and in more details in [12]. We use the box shaped sieving region $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \times [z_{\min}, z_{\max}]$, with $x_{\min} = -x_{\max}$, $y_{\min} = -y_{\max}$, $z_{\min} = 1$.

The algorithm works as follows:

(1) Store every degree 1 ideal with norm $B_{\min} < p_i < B$ in a factor base. We do not sieve with ideals with norms below $B_{\min}$ for efficiency reasons (they contain too many points, and contribute only with a small fraction of the total norm). Instead a small constant contribution is expected in every sieve point (denoted as a sieve tolerance).

We do not use degree 2 ideals as they occur too infrequently in smooth equations. This allows for a more effective line update mechanism.

(2) Compute the modified HNF representation of every ideal $(p_i, r_i + \alpha)$ in the factor base, i.e., get the ideal base in the form (vectors in rows):

$$\begin{pmatrix} p_i & 0 & 0 \\ r_1 & 1 & 0 \\ r_2 & y_{\min} - y_{\max} & 1 \end{pmatrix}. \tag{3}$$

Values $y_{\min}$, and $y_{\max}$ are bounds of the sieve region in the second dimension. Compute r_2 as $r_2 = (y_{\min} - y_{\max} - r_1)r_1 \mod p_i$. The first base vector denotes the update on a single sieve line (the first dimension). The second one represents an update to the next line in the fixed plane. The third one is used to move to the next sieve plane with a simultaneous reset of the sieve line to the beginning of the plane.

(3) Perform the inner sieving algorithm:
   (a) For every ideal $\mathfrak{p}_i$ compute the starting point $x_{\min} + s_i$ on line with $y = y_{\min}, z = z_{\min}$.
   (b) Sieve the line with fixed y, z by marking $x_{\min} + s_i + kp_i$ within the sieve region.
   (c) If $y < y_{\max}$ update $s_i = (s_i + r_1) \mod p_1$.
   (d) If $y = y_{\max}$, increment z, update $s_i = (s_i + r_2) \mod p_2$, and reset $y = y_{\min}$.
   (e) Stop the sieve, if $z = z_{\max}$, or when enough equations is collected, otherwise goto step 3b.

Step 3b is the sieve core. We need an array of the same size as the length of the sieve line $(x_{\max} - x_{\min})$. By marking the point we mean, that we add $\log p_i$ (or its approximation) to a counter in the corresponding memory cell (this is the incremental update of v from section 2). After we process every ideal in the factor base, we examine the stored counter values. If we accumulated the value near $\log N$, where N is the estimated norm of the point examined, then the point is most likely smooth and we report it for further postprocessing.

It is possible to optimize the algorithm by removing inner loops, and by partitioning sieve lines to smaller blocks. The exact algorithm (we call it block sieving) is formally presented as Algorithm 3 in [12]. Both the line sieving algorithm, and its block variant need to place marks on every point in every ideal, and furthermore, examine every point in the sieve region. The difference is only in a constant factor. However, most of the points are not useful, e.g., in our experiments we have examined nearly 2^{40} points to identify approximately 2^{20} points corresponding to smooth equations.

3.2. 3D lattice sieving

The main idea behind the lattice sieve was already formulated by Pollard [8]. Instead of sieving the whole region in $\mathbb{Z}[\alpha]$, we select as a sieve region a sublattice of the original region defined by a chosen "special" prime ideal $\mathfrak{q}$. Norm of every point in $\mathfrak{q}$ has a guaranteed factor $q = N(\mathfrak{q})$, which is useful in some applications, especially in the descent method.

Let $\mathcal{B}$ denote the factor base, and let $\mathfrak{p}_{i,j} \in \mathcal{B}$. To sieve the lattice $\mathfrak{q}$ by $\mathfrak{p}_{i,j}$ we need to efficiently enumerate the points in $\mathcal{L} = \mathfrak{p}_{i,j} \cap \mathfrak{q}$. $\mathcal{L}$ is again a sublattice of $\mathfrak{q}$. The base of $\mathcal{L}$ within $\mathfrak{q}$ can be effectively computed. Let Q be a matrix representing the base of the ideal $\mathfrak{q}$, and P be a matrix representing the base of the ideal $\mathfrak{p}_{i,j}$. We compute[2] the new base by taking the first 3 rows and columns of the image of the matrix

$$\begin{pmatrix} Q \\ P \end{pmatrix}.$$

We will call the set of all new bases of ideals relative to $\mathfrak{q}$ a ($\mathfrak{q}$-)transformed factor base, denoted by $\mathcal{B}_{\mathfrak{q}}$.

Instead of sieving all points in the (large) region at once, we sieve higher number of smaller sublattices defined by different $\mathfrak{q}$'s. If the special-$\mathfrak{q}$ has a large norm, the sieve region contains a relatively small number of points. If the whole number points is M, then the transformed region has approximately M/q points. We should note, that it is not necessary for the transformed sieve region to be just the subset of the original sieve region. However, from the statistical point of view, the norms (respectively the average expected norm) of algebraic integers grow as we move further from the origin, and it is less probable to find a smooth algebraic integer.

If the number of points is too small, we can mark them in the sieving algorithm directly by enumerating points using the bases from $\mathcal{B}_{\mathfrak{q}}$. In this case, it is useful if transformed bases are stored in the LLL-reduced form, instead of Hermite-like form. If we have enough points in the transformed region, we can again apply

[2]Using the NTL library, `http://www.shoup.net/ntl/`.

the line sieving algorithm, but using the bases in $\mathcal{B}_{\mathrm{q}}$. Then these bases should be used in the form (3) (with appropriate new $y_{\min}$, $y_{\max}$).

To consider the efficiency of the sieve, let us choose primes $B/2 < q < B$ as special primes for the sieve. We expect approximately $B/(2\ln B)$ such primes. For each special prime we sieve at most $2M/B$ points. In total we sieve approximately only $M/\ln B$ points. However, we can only detect those smooth equations, that have at least one factor higher than $B/2$. Moreover, we need to compute the transformed base for each special prime, which imposes too large penalty, as shown further in the experimental results section.

4. Experimental results

In the current version of the implementation it is difficult to extend much further our DLP6 computations beyond the already done case of 240-bit equivalent finite field. In our largest experiment we have used a factor base with 893707 ideals (only degree 1, $128 < p < 6500000$). The whole factor base is stored in the memory, and is used in the block sieving algorithm. However, most of the ideals from the factor base are used only in a very small number of equations (as the probability of prime p being a factor of a random integer of is $\sim 1/p$). The goal of our experiments was to find a way to efficiently reduce the size of the used factor base during the sieve.

4.1. Using large primes

The first possible solution is to use only a part of a factor base for sieving. Instead of reporting only points corresponding to smooth numbers, we can also find points with a single large prime factor (or more of them). When post-processing the results from the sieve, we have sorted the factor base by the number of occurrences of a given prime ideal from the most frequent to unused ones. If we split the factor base after the 400000th ideal (arbitrarily chosen) in this ordering (into $\mathcal{B}_1, \mathcal{B}_2$), we get the following distribution of the equations:

- 328621 equations have factors only from $\mathcal{B}_1$;
- 424553 equations have only a single "large" factor from $\mathcal{B}_2$;
- 234823 equations have two "large" factors from $\mathcal{B}_2$;
- only 89987 equations have more than two "large" factors from $\mathcal{B}_2$ (6 equations have the maximum of 7 large factors).

It is not possible to solve the linear system using only the 400000 most frequent ideals (unknowns), but it is possible to solve it using only the equations with at most 2 least frequent ideals.

It is possible to sieve only with a part of the factor base, and to identify the remaining single large factor by the difference between the sieve counter and the estimated norm. We have conducted the sieving experiment by sieving the same region with the factor base consisting of the first 400000 ideals ordered by the ideal norm (corresponding to the smoothness bound $B' = 2757229$). Prior to the sieving, we do not know the exact order according to frequency. However, we can base the ordering on the expected probability of occurrence, which decreases as the norm of the ideal grows.

We have obtained the following results:

- 119053 B'-smooth equations were detected (36.2 % of the expected number);
- 306198 equations with a single large factor $B' < p < B$ were detected;
- we have been able to find at most 18 % percent of the equation with a single large factor by increasing the sieve tolerance (but at the 75 % time impact), which still yields only 85 % of the expected number of such equations.

The difference between the results of the experiments is caused by the different selection of the ideals for the reduced factor base. More frequently occurring ideals, which were missing in the second experiment, were not marked in the sieve. Thus some smooth equations were missed. We do not know a priori which ideals will occur more frequently, so it is not possible to simply reduce the factor base and to use just the line sieve with a single large prime limit.

4.2. Special-q sieve

The complementary experiment (to detecting large primes as a remainder of the sieving) is to use the large primes as a special-q in a lattice sieve. We have used the same bound B' as in the previous experiment (so the factor base has exactly 400000 elements).

We have conducted the experiment in batches of 10000 special-q's from different starting positions in the factor base. The initialization time to transform the bases took approximately 9 % of the total sieve time, but we have used much larger sieve region than necessary. Due to the time complexity we have only sieved with 180000 large primes. Even then just the time needed for base transformation was 2.4-time longer than the whole sieving time in the original solution of the DLP6. This is certainly too large a penalty, and we need a more efficient base transformation routine.

In the experiments we were interested in the number of equations obtained from the special-q sieve. As we can see in Table 1, we were unable to find a single equation for a lot of special-q's (from 45 % to 57 % of special-q's). Unfortunately, this is not necessarily caused by the lack of existing equations, but also by the

Table 1. The results of special-q sieve.

Starting Index	No eqs. found	Good q's	Eqs. per special-q	Yield per special-q	Yield per good prime
400000	45.12 %	22.75 %	0.897	0.334	1.532
500000	48.82 %	19.47 %	0.795	0.283	1.452
600000	52.69 %	17.09 %	0.717	0.244	1.429
700000	55.28 %	14.88 %	0.655	0.207	1.394
800000	56.50 %	14.00 %	0.628	0.193	1.377

stochastic detection of the smooth numbers in our implementation, which is optimized for the line sieve. A more precise detection leads to even slower speed of the lattice sieve, which is undesirable.

The average number of equations per each special-q used decreases from 0.897 to 0.628. Each new special-q leads to a new unknown in the linear system. Thus, if we find only a single equation for a given special-q, we can discard it, as it only increases the system size without adding to the final solution. Only those large primes, for which we are able to find and least 2 smooth equations are interesting. We called them "good primes". The fraction of good primes decreases from 23 % to 14 %.

The yield from the special-q sieve is the number of new equations minus the number of new unknowns. The average yield per each sieved special-q in our implementation and parametric setting is decreasing from 0.334 to 0.193, i.e., we need to run 3–5 special-q sieves to obtain a single additional good equation. Which means that if we want to obtain all missing B'-smooth equations (see the previous experiment), we should run at least around 1.5 million special-q sieves. On the other hand, we can consider only the yield from good primes. It drops from 1.532 new equations to 1.377 per each good special-q prime. This means we only need around 200000 good special-q sieves to complete the system. Unfortunately, we again do not know, how to distinguish such good primes a priori (without the sieving).

4.3. Lattice sieve with small primes

The last experiment we have conducted was to use the lattice sieve with small special-q's. This experiment was based on the observation that just 200 small primes (0.02 % of the factor base) used in the line sieve ($128 < q < 760$) are unknowns in 80 % of the found equations. If we sieve only the points from the sieve region on these special-q's, then the total number of points is slightly lower than in the full sieve. However, if we use small primes, we have a higher chance of sieving through the same points more than once.

The preliminary sieving with the reduced region was only able to produce 3.6 % of expected equations (39383) in 5.6 % expected sieve time. Thus a full experiment was aborted, as it would yield only 64 % of equations (if we are optimistic) in the allotted sieve time, instead of the expected 80 %. This is again caused by the performance loses due to base transformations. Moreover, about 1 % of the equations were reported duplicately (although their detection and removal is very fast).

5. Conclusions

The integer factorization and DLP record solutions do not have similar problems as described in the article. One important difference between these solutions is a much higher probability of a random equation in the sieve region to be smooth. These differences do not play any role in the asymptotic estimates, but do seem to be critical in the actual implementation, and in the effect of some heuristic optimizations of the core NFS. Thus the solution of DLP6 in practice is more costly then the equivalent DLP/IF problems.

A direct application of the 3D lattice sieve seems too slow to provide some performance gain. This is mostly caused by the slow factor base transformation, and less efficient estimation of the norm in the 3D lattice sieve. The use of large primes and special-q sieve still have a potential be used as a trade-off in requirements for the storage of factor base and the sieve time, if it is possible identify good special-q's a priori the application of the lattice sieve. We are not aware of some specific properties of such primes, and it is an interesting open question whether there can be a fast algorithm to identify them a priori to the sieving.

REFERENCES

[1] AOKI, K.—KIDA, Y.—SHIMOYAMA, T.—UEDA, H.: *GNFS factoring statistics of RSA-100, 110, ..., 150,* April 16, 2004, `http://eprint.iacr.org/2004/095.pdf`.

[2] COMMEINE, A.—SEMAEV, I.: *An algorithm to solve the discrete logarithm problem with the number field sieve*, in: Public Key Cryptography—PKC '06 (M. Yung et al., eds.), 9th International Conference on Theory and Practice of Public-Key Cryptography, New York, NY, USA, 2006, Lecture Notes in Comput. Sci., Vol. 3958, Springer-Verlag, Berlin, 2006, pp. 174–190.

[3] JOUX, A.—LERCIER, R.: *Improvements to the general number field sieve for discrete logarithms in prime fields: a comparison with the Gaussian integer method*, Math. Comp. **72** (2003), 953–967.

[4] JOUX, A.—LERCIER, R.—SMART, N.—VERCAUTEREN, F.: *The number field sieve in the medium prime case,* in: Advances in Cryptology—CRYPTO '06 (C. Dwork, ed.), 26th Annual International Cryptology Conference, Santa Barbara, California, USA, 2006, Lecture Notes in Comput. Sci., Vol. 4117, Springer-Verlag, Berlin, 2006, pp. 326–344.

[5] *The Development of the Number Field Sieve* (A. K. Lenstra, H. W. Lenstra, Jr., eds.), Lecture Notes in Math., Vol. 1554, Springer-Verlag, Berlin, 1993.

[6] LENSTRA, A. K.—LENSTRA, H. W., JR.—MANASSE, M. S.—POLLARD, J. M.: *The number field sieve,* in: The Development of the Number Field Sieve (A. K. Lenstra, H. W. Lenstra, Jr., eds.), Lecture Notes in Math., Vol. 1554, Springer-Verlag, Berlin, 1993, pp. 11–42.

[7] LENSTRA, A. K.—VERHEUL, E. R.: *An overview of the XTR public key system,* in: Public-Key Cryptography and Computational Number Theory (K. Alster et al., eds.) Proc. of the Internat. Conference Organized by the Stefan Banach Internat. Math. Center, Warsaw, Poland, 2000, de Gruyter, Berlin, 2001, pp. 151–180.

[8] POLLARD, J.: *The lattice sieve,* in: The Development of the Number Field Sieve (A. K. Lenstra et al., eds.), Lecture Notes in Math., Vol. 1554, Springer-Verlag, Berlin, 1993, pp. 43–49.

[9] SCHIROKAUER, O.: *Virtual logarithms,* J. Algorithms **57** (2005), 140–147.

[10] ZAJAC, P.: *Generalized line sieve algorithm,* in: Proc. of ELITECH '07, STU Bratislava, 2007.

[11] ZAJAC, P.: *Remarks on the NFS complexity,* Tatra Mt. Math. Publ. **41** (2008), 79–91.

[12] ZAJAC, P.: *Discrete Logarithms and Degree Six Numbere Field Sieve: A practical Approach.* VDM Verlag Dr. Müller, Saarbrücken, 2009.

Received April 30, 2010

Department of Applied Information and Information Technology
Slovak University of Technology
Ilkovičova 3
SK–812-19 Bratislava
SLOVAKIA
E-mail: pavol.zajac@stuba.sk