Tatra Mt. Math. Publ. **41** (2008), 79–91

TATRA MOUNTAINS
Mathematical Publications

REMARKS ON THE NFS COMPLEXITY

Pavol Zajac

ABSTRACT. We investigate practical issues with implementing the NFS algorithm to solve the DLP arising in XTR-based cryptosystems. We can transform original XTR-DLP to a DLP instance in $\mathbb{F}_{p^6}$, where p is a medium sized prime.

Unfortunately, for practical ranges of p, the optimal degree of an NFS polynomial is less than the required degree 6. This leads to a problem to find enough smooth equations during the sieve stage of the NFS algorithm. We discuss several techniques that can increase the NFS output, i.e., the number of equations produced during the sieve, without increasing the smoothness bound.

1. Introduction

This research was motivated by the problem of finding discrete logarithms in XTR based systems [7]. XTR uses a subgroup of $\mathbb{F}_{p^6}^*$ with prime order q dividing $p^2 - p + 1$ (called XTR group). Its elements are represented by their traces, and efficient arithmetic is developed to allow fast exponentiation. Thus XTR-DL problem is to find the unknown exponent d from $Q = G^d$, where G, Q are known XTR traces.

The XTR-DL problem can be solved in XTR group by generic methods of asymptotic complexity $O\big(q^{\frac{1}{2}}\big)$. If q is chosen as large as possible, i.e., $p^2 - p + 1$, then the complexity becomes $O(p)$. The computation becomes quickly infeasible with growing p. On the other hand, XTR-DL can be transformed (in a polynomial time) to an instance of the discrete logarithm problem in the finite field $\mathbb{F}_{p^6}$. Then it can be solved by the Number Field Sieve (NFS) in subexponential time with complexity $L_{p^6}\big(\frac{1}{3}, c\big)$, where

$$L_x(\alpha, c) = \exp\big((c + o(1))(\log x)^{\alpha}(\log\log x)^{1-\alpha}\big)\,. \tag{1}$$

2000 Mathematics Subject Classification: 94A60, 11Y40, 11R04.
Keywords: algebraic numbers, number field sieve, complexity.
This work was supported by Grant VEGA 1/3115/06 and ESF SORO/JPD3-038/2005.

The constant c in asymptotic complexity estimate is connected to a choice of optimal smoothness bound B in the NFS algorithm. Using the complexity estimates of [4], the complexity of NFS becomes smaller than complexity of general methods for $p \approx 2^{40}$.

In practical experiments, the situation becomes more complicated. The real NFS is in fact a rather general method (or a set of related algorithms) than an exact algorithm with exactly defined parameters. This leads to many implementation and parameterization options that affect the actual performance.

Other problems can arise from the fact that the polynomial of degree 6 is actually too large for smaller p. Joint degree of two polynomials used in NFS for $\mathbb{F}_{p^6}$ is at least 12 (two polynomials of degree 6 are required). Method of [3] uses two polynomials of degrees $d+1, d$. Degree $d = 6$ is optimal for fields of sizes 2^{780}, corresponding to our $p \approx 2^{130}$. For smaller primes, real performance is affected by a faster growth of the norms of the sieved algebraic numbers.

NFS implementation for DLP in $\mathbb{F}_{p^6}$ gains optimal asymptotical performance only for p's that are too large to consider actual logarithms computable. In practical ranges, NFS parameters must be chosen in a suboptimal (in terms of asymptotic complexity) manner. The main problem during the computation is to find enough smooth equations[1]. We literally strive for every single smooth equation we can get. In this paper we present some heuristics that can be used to increase the number of smooth equations gained from the sieve stage (NFS yield) without considerable effect on the size of the factor base used during the sieve stage.

In the Section 2 we summarize the basic steps of the NFS method. We also show where possible changes can be made that affect the NFS output, i.e., the number of equations produced during the sieve. In Section 3 we analyze how we can influence the NFS output by the polynomial selection. More equations can also be gained by using multiple sieves as it is described in Section 4. In Section 5 an idea of [4] to sieve the space of higher dimension is elaborated. We present a possible change of the line sieving algorithm and consider the actual sieving results. In Section 6 we present some results considering the use of the large factors. Section 7 summarizes the results and recommendations.

2. The Number Field Sieve

The description of the Number Field Sieve and its use to find discrete logarithms in finite fields can be found in [4]. The complexity of this method is

[1]Smooth equation is an equation in the form (3), but in the context of the NFS algorithm the notion "smooth equation" can also denote a pair of smooth algebraic numbers, as well as the corresponding point of the sieve region.

described there as well. Another good complexity analysis of the NFS is in [1]. For the purpose of this article, we present a short overview of the NFS without going into any details of its mathematical aspects. Some specific algorithmic details are further elaborated in the appropriate sections of the article.

Let $\mathbb{Z}_K$ be a ring of integers of the number field K. Let $\xi \in \mathbb{Z}_K$ has B-smooth norm $N(\xi) = \prod p_i^{e_i}$, i.e., $N(\xi)$ has only prime divisors $p_i < B$. We will call ξ a B-smooth algebraic number. The principal ideal $(\xi) = \xi\mathbb{Z}_K$ can be factored as a product of prime ideals lying over primes p_i.

Basic NFS principle 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 t be a common root of f, g in $\mathbb{F}_{p^d}$. Then there exist two homomorphisms $\phi : K_1 \to \mathbb{F}_{p^d}$, and $\psi : K_2 \to \mathbb{F}_{p^d}$, defined by sending α, resp. β, to t.

Let g be a generator of $G = \mathbb{F}_{p^d}^*$ and q is a (large) prime dividing order of G. Let algebraic number $\xi \in K_1$ be B-smooth with the corresponding prime ideal decomposition

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

Further let $\pi_j \in \mathfrak{p}_\mathfrak{j}$, and let h be a class number of K. Using Schirokauer logarithmic maps λ, we can transform this equation to

$$\log_g\big(\phi(\xi)\big) \equiv \sum_{j=0}^{r} \lambda_j(\xi)\Lambda_j + \sum_j v_j x_j \pmod q, \tag{2}$$

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

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 (2), we can write

$$\sum_{j=0}^{r_1} \lambda_j^{(1)}(\xi_1)\Lambda_j^{(1)} + \sum_j v_i x_j^{(1)} \equiv \sum_{j=0}^{r_2} \lambda_j^{(2)}(\xi_2)\Lambda_j^{(2)} + \sum_j v_i x_j^{(2)} \pmod q, \tag{3}$$

with unknown "virtual logarithms" $\Lambda_j^{(1)}, \Lambda_j^{(2)}, x_j^{(1)}$, and $x_j^{(2)}$. We call any equation in the form (3) a smooth equation.

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, we can try to find a non-trivial solution of the corresponding linear system. By substituting to equations (2) we can compute logarithms of the corresponding elements of $\mathbb{F}_{p^d}^*$. Logarithms of other $\mathbb{F}_{p^d}^*$ elements can be

further computed by the means of descent computation [1], [4], which is beyond the scope of this article.

The goal of the NFS based algorithm is to find enough smooth equations in an efficient manner. Equations are sought by the means of a sieve. A linear subspace (a sieving region) representing elements of both K_1, and K_2 corresponding to a same image in $\mathbb{F}^*_{p^d}$, is mapped to a memory of the computer. A sieving region is usually a space $(a, b) \in \mathbb{Z}^2, 0 < |a|, b < M$, corresponding to a pair of algebraic numbers $(a - b\alpha, a - b\beta)$. We mark points corresponding to algebraic numbers belonging to prime ideals from factor base. After marking points from every ideal we are able to identify the smooth ones. More details of the sieve can be found in papers mentioned in [5].

Any NFS implementation thus consists of the four basic steps:

(1) *Parameter selection.* We select polynomials f, g, estimate B and size of the sieving region, construct a factor base and do any required preprocessing that can speed up sieving.
(2) *Sieving.* This is the most time consuming part, where smooth equations are identified (usually in parallel on many computers).
(3) *Linear algebra.* Solve the linear systems constructed from smooth equations.
(4) *Individual logarithms.* Compute logarithms of elements of $\mathbb{F}^*_{p^d}$.

The NFS is a complex method that allows many further parameterizations, and different variants of algorithm implementation. Some of the optional heuristics that can influence the actual algorithm performance are:

(1) Choice of the polynomials.
(2) Choice of the smoothness bound.
(3) Choice of the sieving region.
(4) Use of more than two algebraic fields.
(5) Use large primes and partially smooth equations.
(6) Use of special-q sieve.
(7) Use of a sieve region with higher dimension.
(8) Use of a different smoothness detection and factorization techniques.

In the following sections we present our findings from experiments involving computations of discrete logarithms in $\mathbb{F}_{p^6}$.

3. A choice of the polynomials

In the first step of the NFS, we must choose (at least) two (monic) polynomials $f, g \in \mathbb{Z}[x]$ irreducible over $\mathbb{Z}$, with a common root in $\mathbb{F}_{p^d}$. In our case, when p is not too large, the recommended choice is [4]:

- f is a monic polynomial of degree d irreducible over $\mathbb{F}_p$ with small coefficients (in absolute value);
- $g(x) = f(x) \pm p$.

3.1. A choice of f

There are no strict limits on the choice of the polynomial f, although according to [4] it is possible to exploit automorphism group of certain number fields if the class number of the field is known. Thus we consider only[2] the absolute value of coefficients in the selection of the polynomial f.

One of the most important parameters that influence the NFS performance is the smoothness probability (in the sieve region). This is a probability that an algebraic number from the sieve region has B-smooth norm (for a fixed B). The smoothness probability in the NFS thus depends on the smoothness bound B and the sieving region. Larger sieving region means higher NFS output, but requires more work in sieving. Furthermore, the norms of elements further from origin are higher, and thus less likely B-smooth (with B fixed).

Another way to increase smoothness probability is to increase smoothness bound B. However, this leads to an increased number of elements in the factor base, increased sieving time, and increased size of the linear system respectively. We also need to find even more equation, thus increasing B can lead even to sieve deterioration.

A choice of the polynomial is the third possibility to influence the smoothness probability. The number of irreducible polynomials we can choose from is large enough to consider statistical effects of the polynomial choice on the smoothness probability in a fixed sieve region with fixed smoothness bound B. Our experimental results are described in [10]. The distribution of smoothness probability w.r.t. polynomial choice is almost normal. The polynomials with a higher smoothness probability for some size of the sieving region and B tend to give us higher smoothness probability for other choices of the region size and B. If computing discrete logarithms for different p's and fixed d, it is possible to prepare a list of suitable polynomials sorted in descending order by smoothness probability.

The standard deviation of smoothness probability distribution w.r.t. polynomial selection is high in comparison to the sensitivity of smoothness probability

[2]Additional condition is that the group order q does not divide discriminant of f or g, but this happens very rarely, so we can ignore this case.

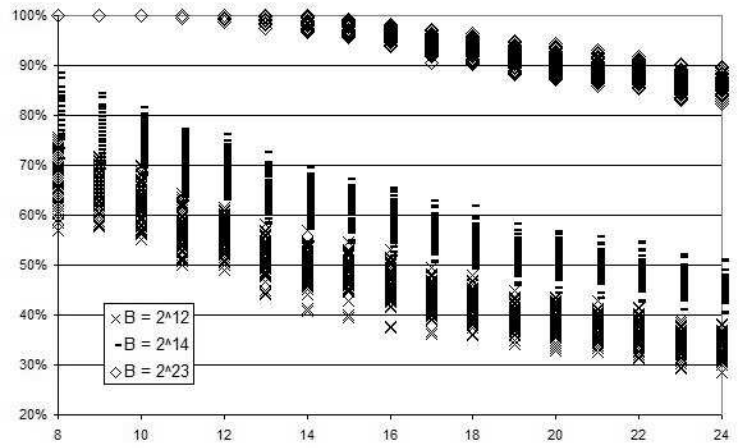

FIGURE 1. The fraction of B-smooth algebraic numbers $a + b\alpha_i + c\alpha_i^2$, with $0 < a, b, c \leq M$, for $M = 8, 9, \ldots 24$. The α_i's are roots of 100 distinct degree 6 irreducible polynomials (over $\mathbb{Z}$), with coefficients $|a_i| \leq 2$.

to the changed B. Experimentally, we can gain the same increase in smoothness probability by changing the polynomial, as by increasing B four times (thus increasing factor base correspondingly), as shown in Figure 1.

The smoothness probability w.r.t. polynomial selection seems to be positively correlated with the number of first degree ideals in the factor base (with constant B), as depicted in Figure 2. This could also mean that an increase in smoothness probability is compensated by requiring a larger factor base. However, the correlation is small, and for any size of the factor base (with fixed B) we have detected both better and worse polynomials (in terms of smoothness probability) than average.

When choosing polynomial we can consider both maximizing the smoothness probability and minimizing the number of ideals in the factor base to gain the best results of the sieve.

3.2. A choice of g

Since we have only two polynomials $g(x) = f(x) \pm p$ in the standard polynomial selection, it is difficult to compare the effect of their selection statistically (it mostly depends on p itself). Thus we should test the smoothness probability in both possible fields, or just choose randomly one of them. It is also possible to restrict selection according to other criteria, e.g., the discriminant of the finite field.

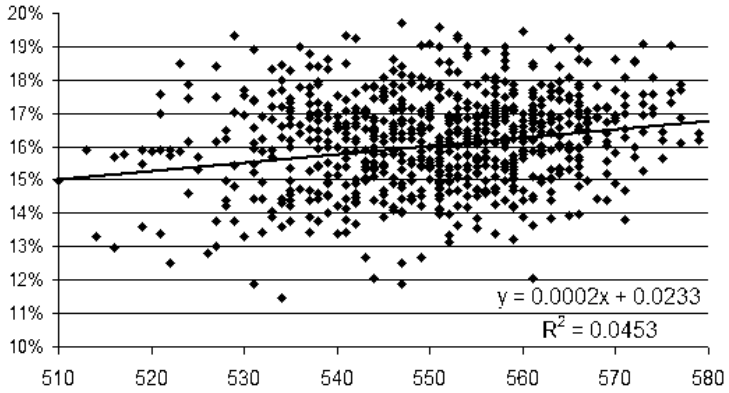

FIGURE 2. The fraction of B-smooth algebraic numbers $a + b\alpha_i$, with $0 < |a|, b \leq 32$ as a function of factor base size. The α_i's are roots of 1000 distinct degree 6 irreducible polynomials (over $\mathbb{Z}$), with coefficients $|a_i| \leq 2$.

Since the absolute coefficient of $g(x)$ is significantly larger than other coefficients, norms of the principal ideals of $\sum a_i \beta^i$ are growing faster with increasing a_i with higher i (dimension). Thus it is usually a good idea to use a skewed sieving region, longer along a_0 axis.

4. Multiple polynomials

The two polynomials $g(x) = f(x) \pm p$ are not the only possible choice with comparative size of the coefficients. We can consider the choice of any $g(x) = f(x) + ph(x)$, with $\deg h < \deg f$, where coefficients of $h(x)$ are small in absolute value (typically equal to ± 1). On the other hand, the norms of algebraic numbers in fields defined by $h(x)$ with higher degree are larger (in absolute value).

However, these polynomials are useful to implement a multiple polynomial variant of NFS. Classical multipolynomial NFS to solve discrete logarithms in $\mathbb{F}_p$ is described in [1]. It is possible to adapt the method also for $\mathbb{F}_{p^d}$. General outline of the new method is as follows:

(1) Let $f(x) \in \mathbb{Z}[x]$ be a monic polynomial of degree d irreducible over $\mathbb{F}_p$. Let $K_0 = \mathbb{Q}(\alpha)$, with $\alpha \in \mathbb{C}$, $f(\alpha) = 0$.

(2) Let $g_i(x) = f(x) + ph_i(x), i = 1, 2, \ldots$ with $\deg h_i < \deg f$; all h_i's are distinct with small coefficients in absolute value. Let $g_i(\beta_i) = 0, \beta_i \in \mathbb{C}$, and let $K_i = \mathbb{Q}(\beta_i)$.

(3) For each $i = 1, 2, \ldots$ find points in the sieve region that correspond to smooth algebraic numbers in field K_0 as well as in the field K_i.

(4) If the number of smooth points accumulated using K_i is smaller than the size of the factor base in this field (plus small constant for logarithmic maps), the results of sieving K_i should be discarded.

(5) Create the system of equations similar to equation (3) with left-hand side corresponding to factorization of elements from K_0, and right-hand side to factorization of corresponding elements from K_i's which were not discarded in step 4.

(6) After computing the solution of the system of equations, individual logarithms of $\mathbb{F}_{p^d}$ can be computed using descent method.

In the linear system of equations, unknowns correspond to prime ideals in K_0 and in every K_i used plus $O(1)$ logarithmic maps or virtual unit logarithms (we need at most d of them). Let the size of factor base for K_i be c_i and let r_i points $(a_0, \ldots, a_t)$ correspond to elements $\sum a_j \alpha^j$, $\sum a_j \beta_i^j$ smooth in both K_0 and K_i. Then we can create r_i new equations at the cost of $c_i + d$ unknowns. If $r_i < c_i + d$, the NFS with the given K_i is unsuccessful. If for some k we get $r_k > c_k + c_0 + 2d$, we do not need more K_i's, just K_k. Thus using multiple polynomials is suitable only if the expected NFS output is above maximal c_i but below minimal $c_i + c_0$.

As it is difficult to precisely estimate the NFS output, we should start with sieving K_1 given by one of the $g(x) = f(x) \pm p$. Using the number of smooth found equations, we can estimate the number of required K_i's. It should be taken into account that for $h_i(x)$ with higher degree, the norms of algebraic numbers are larger. Thus the expected number of found equations is lower. This can be compensated by setting different smoothness bound B for each K_i, or K_0, respectively.

5. Sieving higher dimensions

A classical NFS sieves only a planar region, i.e., points (a_0, a_1) corresponding to a pair of algebraic numbers $(a_0 + a_1\alpha, a_0 + a_1\beta)$. When p is small in comparison with p^d, norms of elements in sieving region get too high without enough of smooth equations collected.

As described in [4], we can gain more equations by sieving $(t+1)$-tuples $(a_0, \ldots, a_t)$ corresponding to pairs of algebraic numbers $\left(\sum_{i=0}^{t} a_i \alpha^i, \sum_{i=0}^{t} a_i \beta^i\right)$. The norms can be bounded by $(d+t)^{(d+t)} M_a^d M_f^t$, where M_a is the upper bound of absolute values of a_i, and M_f is the upper bound of absolute values of coefficients of f, or g, respectively. As the polynomial g is skewed with large absolute

coefficient, norms of elements with higher dimension are larger (for comparable sizes of a_i's).

Besides problems with larger norms in higher dimensions, we encounter also some new implementation problems. The prime ideals in corresponding factorizations can be of a degree as high as t. On the other hand, the higher degree factors are very rare. A prime ideal over some p_i of degree t has norm p_i^t. The chance of the factor with norm n to appear in factorization is $\sim \frac{1}{n}$. Thus an ideal over p_i of degree t has $1/p_i^{t-1}$ smaller chance to appear than a degree one ideal over p_i. Usually only higher degree ideals over small primes appear in smooth equations. As it is useful to exclude small primes from the sieve (for efficiency), we can also exclude all higher degree prime ideals as well. Thus we sieve only with degree one prime ideals.

In a multidimensional sieve we must check that a sieved element, when taken as a polynomial $\sum_{i=0}^{t} a_i x^i \in \mathbb{Z}[x]$, is irreducible over $\mathbb{Z}$. Otherwise we can take the corresponding factors and write equations directly for them (if their product is smooth, then they are certainly smooth as well). If we fix t, we can leave out the irreducibility check at the expense of more equations required to be able to solve the linear system.

There are two main types of the sieve used in the NFS, namely the line sieve and the lattice sieve. The lattice sieve requires additional computation of the base of prime ideals (from within some part of the factor base) within special-q lattice. The complexity of this computation is growing with higher dimensions. Within the lattice sieve, line sieve is used with recomputed ideal bases.

Algorithm described in [9] is a general outline of the line sieve algorithm for any dimension and for any degree of used prime ideals. However, this algorithm is ineffective for fixed t, when using degree one ideals only.

In this case we can use the Hermite Normal Form of the ideal base (either in $\mathbb{Z}[\alpha]$ or in special-q lattice). Excluding small finite number of cases, the bases of $(t+1)$-dimensional subspace have form (base vectors in rows):

$$\begin{pmatrix} p_i & 0 & 0 & \dots & 0 \\ r_1 & 1 & 0 & \dots & 0 \\ r_2 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_t & 0 & 0 & \dots & 1 \end{pmatrix}.$$

Let some $(a_0, a_1, \dots, a_t)$ lie in the ideal $\mathfrak{p}_i$ with the bases above. All points $(a_0 + kp_i, a_1, \dots, a_t)$ lie also in $\mathfrak{p}_i$. Thus we can mark positions on the single line for fixed $a_1, \dots, a_t$ for various $\mathfrak{p}_i$ by finding the "starting point" and making "jumps" of size p_i. Consider the change of some a_j to $a_j + 1$. Then we must certainly mark points $(a_0 + r_j + kp_i, a_1, \dots, a_j + 1, \dots, a_t)$. If the starting point on the line given by $a_1, \dots, a_j, \dots, a_t$ was $m_0 + s$, then the new starting point on

$a_1, \ldots, a_j + 1, \ldots, a_t$ is $m_0 + (s + r_j) \bmod p_i$, where m_0 is the smallest possible a_0-coordinate in the sieving region. Another speedup in the case of cuboid region that can be used is to precompute the change of "starting point" from the line $M_1, \ldots, M_j, a_{j+1}, \ldots, a_t$ to $m_1, \ldots, m_j, a_{j+1} + 1, \ldots, a_t$, where m_j, M_j is the smallest, and highest coordinate in dimension j, respectively.

The algorithm can be summarized as follows:

(1) For every ideal $\mathfrak{p}_i$ compute starting point $m_0 + s_i$ on line $a_1 = m_1, \ldots, a_t = m_t$.
(2) Sieve line $a_1, \ldots, a_t$ by marking $m_0 + s_i + kp_i$ within sieve region.
(3) If $a_1 < M_1$, update $s_i = (s_i + r_1) \bmod p_1$.
(4) If $a_1 = M_1$, find first $a_k < M_k$, increment $a_k = a_k + 1$, update s_i and reset $a_1 = m_1, \ldots, a_{k-1} = m_{k-1}$.
(5) If $a_t = M_t$, stop sieve, otherwise, go to step 2.

The algorithm can be distributed by partitioning the sieve region into smaller subregions. The size of each region along the a_0 should be always the same and equal to B (so that every prime appears at least once on each iteration).

We have used the sieve with $t = 2$ (using coordinate system x, y, z) in the NFS computations in $\mathbb{F}_{p^6}$ with region $[-M, M] \times [-M, M] \times [1, z_{\max}]$. The z coordinate should not be negative, as multiplying the elements by -1 leads to the same equation. Results of one of the experiments are summarized in Table 1. As expected, the NFS output is the highest for $z = 1$. It decreases with the factor of $\left(\frac{1}{2}\right) \log z$. The NFS output is lower, when z has small prime factors, because we have removed points with $\gcd(x, y, z) > 1$. Interesting fact is that we get significantly more equations for plane with $z = 1$ than for plane with $z = 0$, i.e., the classical 2D sieve. This could lead to some optimizations even in existing sieves.

6. Using large primes

In the classical NFS we collect smooth equations using elements with B-smooth norms with some precomputed fixed B. Smooth elements are identified by the sieve, i.e., we mark points in every prime ideal (of degree one) with norm $p_i < B$. Marking the point means that we add the logarithm of p_i (or its approximation) to some counter associated with the point. After marking points in every prime ideal, we compare the counters with the estimated logarithm of the norm. Thus we can easily see whether the associated algebraic number is smooth.

In some cases we have "almost" reached the norm up to some large factor n. If all primes below B were used in the sieve, then certainly $n > B$. If also $n < B^2$,

TABLE 1. NFS output by z. NFS parameters were: $B = 80000$, $f(x) = x^6 - 2x + 2$, $g(x) = x^6 - 2x - 529041$. Sieving region was $[-2^{16}, 2^{16}] \times [-2^{12}, 2^{12}] \times [1, 256]$, and for comparison corresponding (x, y)-halfplane with $z = 0$, $y > 0$. Total NFS output was 29477 equations in 15642 unknowns.

z	NFS output
0	303
1	1103
2	584
3	724
4	463
5	654
6	323

z	NFS output
250	31
251	46
252	22
253	50
254	35
255	40
256	23

then it is clearly a prime. Let $B < B_1 \leq B^2$. If $B_1 \leq n < B_1^2/B$, then either n is prime or n has two (not necessarily distinct) prime factors $B < p_1, p_2 < B_1$. We can thus identify some additional factors almost for free. Single large prime requires one additional comparison. Composite factors require primality testing, and factoring of relatively small number, which are both fast.

These large factors are obtained as a side effect almost for free, but their effective use requires some further post-processing. Every large factor represents one new unknown in the linear system. Thus only large factors that occur more than once are usable. Large primes have been used in many known factoring records using NFS. On the other hand, discrete logarithm records were computed without large primes using larger factor bases than usual [3].

The method that combines both large primes variant and large factor base is to use two-stage sieving. We use two smoothness bounds $B, B_1 < B^2$. First, a classical sieve is applied for every (degree one) prime ideal over $p_i < B$. Large primes below B_1 are identified (both single and double). If we do not have enough B-smooth equations, large primes that occur at least twice are added to the factor base, along with corresponding equations. If we still do not have enough equations, we use special-q lattice sieve for every large prime. This means, we construct a lattice corresponding to a prime ideal over large q_i, and sieve elements on this lattice. Norm of every algebraic number associated with points on this lattice certainly has q_i as its factor. We can even skip the medium step and directly use special-q sieve for every prime $B < q_i < B_1$.

For an easy detection of large primes we should use $B_1 < B^2$ for single large prime, or $B_1^2 < B^3$ for double large prime variant. Practical experiments show that the upper bound is too large if we want to avoid special-q sieve. With larger B_1 we gain more partial equations, but most of them are useless, since

the corresponding prime ideals appear only in a single equation. Recommended practical choice is $B^{1.2} < B_1 < B^{1.4}$ [6].

7. Conclusions

We presented several techniques that can influence the running time of the NFS implementation and the size of the linear system we need to solve.

Choice of the polynomial have significant impact on the smoothness probability in the sieve region, thus influencing the NFS output and running time. An effect of the best polynomial choice in comparison to the worst one is comparable to increasing the factor base by a factor of four.

If the NFS output is comparable with factor base size on one side, but smaller than required, we can create a solvable linear system by employing multiple algebraic number fields. This leads to a larger linear system.

Use of large primes and partially smooth equations can lead to a significantly higher NFS output, at the price of a larger linear system. An effective filtering and a careful selection of parameters should be employed when using large primes, otherwise the performance deteriorates.

A significant increase in the number of smooth equations can be obtained by increasing the dimension of the sieve region. This however leads to an increased sieve complexity, as we must sieve more points. However, using distributed computing, the cost of sieving can be significantly smaller than the cost of solving the linear system.

An interesting side result of higher dimensional sieve, is its possible application to increase the effectiveness of classical NFS with two dimensional sieve. Instead of mapping $(a, b) \in \mathbb{Z}^2$ to a pair of algebraic numbers $(a + b\alpha, a + b\beta)$, we can map them to a pair $(a + b\alpha + \alpha^2, a + b\beta + \beta^2)$. Norms of algebraic numbers in this sieving region are comparable to norms in the original sieving region (depending on the region size and polynomial coefficients). We can however use also points with $\gcd(a, b) \neq 1$, and $b \leq 0$, respectively. Thus we can increase the number of equations that can be found by sieve without increasing the sieve region. If we want to avoid (trivially) linearly dependent equations, we should check whether $b^2 - 4a$ is a square in $\mathbb{Z}$ (in this case the equation can be split according to the factorization of the polynomial $a + bx + x^2$).

REFERENCES

[1] COMMEINE, A.—SEMAEV, I.: *An algorithm to solve the discrete logarithm problem with the number field sieve,* in: Public Key Cryptography—PKC '06 (Y. Moti et al., eds.), Lecture Notes in Comput. Sci., Vol. 3958, Springer-Verlag, Berlin, 2006, pp. 174–190.

[2] DODSON, B.—LENSTRA, A. K.: *NFS with four large primes: an explosive experiment,* in: Advances in Cryptology—CRYPTO '95 (D. Coppersmith, ed.), Lecture Notes in Comput. Sci., Vol. 963, Springer-Verlag, Berlin, 1995, pp. 372–385.

[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.), 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: 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.), de Gruyter, Berlin, 2001, pp. 151–180.

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

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

[10] ZAJAC, P.: *Smoothness probability in degree six number fields*, J. Electr. Engr. **57** (2007), (to appear).

Received September 28, 2007

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