Evaluation of Complexity of Mathematical Algorithms

Thorsten Kleinjung

1 Introduction

The security of many cryptographic applications bases on the difficulty of factoring integers. One of the two major steps in the best factoring algorithm known so far is the collection of relations. The purpose of this report is to estimate the cost of this step for integers of size 1536-bit and 2048-bit using existing software and hardware. This will give an upper bound for the difficulty of this step.

The next section gives a brief overview of the factoring algorithm and a more detailed description of the relation collection step. In section 3 we discuss the choice of polynomial pairs which are a prerequisite for the collection of relations. The main part of this report consists of the fourth section which describes the sieving experiments, how parameters were chosen and how estimates were derived from the results of the experiments. Moreover, several improvements leading to better estimates are discussed. The report concludes with a summary of the results.

2 The general number field sieve

First, we give a brief overview of the General Number Field Sieve (GNFS) (see [LL]) and fix the notation. Then we explain the collection of relations in more detail.

Let N be a composite number to be factored which is not a power of a prime. We assume that N is not divisible by small primes, in particular not by auxiliary primes which are used in the algorithm below. The GNFS algorithm consists of the following five steps:

1. Polynomial selection
   In this step two coprime, irreducible polynomials, $f_1, f_2 \in \mathbb{Z}[X]$, are chosen such that they share a common root m modulo N. Since the runtime of the

subsequent steps depends on the quality of the chosen polynomial pair, many pairs should be considered and the best one chosen (see also next section). We denote by $F_1, F_2 \in \mathbb{Z}[X, Y]$ the corresponding homogeneous polynomials, i. e., $F_i(X, Y) = f_i(\frac{X}{Y}) \cdot Y^{\deg(f_i)}$, $i = 1, 2$. Let also $K_i = \mathbb{Q}[X]/(f_i(X))$ be the associated number fields, $K_{i,(N)} = \{\frac{x}{y} | x, y \in \mathcal{O}_{K_i}, \gcd(y, N) = 1\}$ and $\phi_i : K_{i,(N)} \longrightarrow \mathbb{Z}/N\mathbb{Z}$ defined by $X \mapsto m$.

2. Collection of relations
   The aim of this step is to find sufficiently many coprime pairs $(a, b) \in \mathbb{Z} \times \mathbb{N}$ such that $F_i(a, b)$ is L_i-smooth for $i = 1, 2$, where L_1 and L_2 are parameters. A number is said to be L-smooth if it splits completely into primes $\leq L$. Each pair (a, b) such that $F_i(a, b)$ is L_i-smooth, $i = 1, 2$, is called a relation and the parameters L_1 and L_2 are called large prime bounds.

   The lattice sieving procedure to find such relations is described below in more detail.

3. Construction of the matrix
   This is a combination of several steps. First, duplicate relations, which for example may arise by lattice sieving, are removed. Then a preliminary matrix M_0 over $\mathbb{F}_2$ is constructed whose rows resp. columns correspond to prime ideals of norm $\leq L_i$ in K_i, $i = 1, 2$, resp. to relations. The entry in a row corresponding to the prime ideal $\mathcal{P}_i$ of K_i and a column corresponding to (a, b) is the valuation modulo 2 of $a - bX \bmod f_i$ at $\mathcal{P}_i$.

   Next the matrix is preprocessed by eliminating zero rows, zero columns and columns containing a non-zero entry which is the only non-zero entry in its row. Furthermore elementary column operations can be done. This preprocessing results in a denser, but smaller matrix M whose rows correspond to prime ideals of K_1 or K_2 and whose columns correspond to sets of relations.

4. Linear algebra
   In this step some solutions of the linear system of equations $Mv = 0$ over $\mathbb{F}_2$ will be found. This is usually done by the block Lanczos or block Wiedemann algorithm which usually give several (64, say) solutions.

5. Final computations
   A solution of $Mv = 0$ corresponds to a subset $\mathcal{S}$ of the set of relations such that $\alpha_i = \prod_{(a,b)\in\mathcal{S}} a - bX \bmod f_i$, $i = 1, 2$, has an even valuation at almost all prime ideals of K_i. Using quadratic characters one searches for a linear combination of solutions of $Mv = 0$ such that the corresponding α_i are squares

in K_i (with very high probability). Then square roots $\beta_i^2 = \alpha_i$ are calculated and the computation of $\gcd(\phi_1(\beta_1) - \phi_2(\beta_2), N)$ will heuristically split N with probability $\geq \frac{1}{2}$.

Heuristically the runtime of GNFS is

$$O(e^{((\frac{64}{9}+o(1))(\log N)(\log\log N)^2)^{\frac{1}{3}}}).$$

Since this report only considers the second step (collection of relations) we now present it in more detail.

There are several methods for finding relations, in practice one uses line sieving and lattice sieving. For smaller numbers line sieving is faster than lattice sieving, but for larger numbers (say, above 512 bit) only a small fraction of relations is usually produced by line sieving. Therefore we focus on lattice sieving.

For $i = 1, 2$ let $\mathcal{F}_i$ be the set of prime ideals of degree 1 of K_i of norm less than some bound $B_i < L_i$ and whose norm is coprime to the discriminant and the leading coefficient of f_i. These sets are called factor bases and their elements can be represented by $(p, X - r)$ where p is a prime number and r is a root of f_i modulo p. The numbers B_i are called factor base bounds.

In lattice sieving we first choose an ideal $(q, X - s)$ of K_1 called a special-q where s is a root of f_1 modulo q. For technical reasons we assume that q is square-free and coprime to the discriminant and the leading coefficient of f_1.

For a special-q as above we consider pairs (a, b) such that $a - bX \in (q, X - s)$. This is done by choosing two vectors v and w which generate the lattice $(q, X - s) \cap (\mathbb{Z} \oplus \mathbb{Z}X)$. Then the sieving area $A_{q,s} \subset \mathbb{Z}^2$ consists of the pairs (a, b) corresponding to $iv + jw$ for $-I \leq i < I$, $0 \leq j < J$. The vectors v and w are chosen in such a way that the sizes of $|F_1(a, b)|$ and $|F_2(a, b)|$ become as small as possible.

A simplified description of lattice sieving is the following:

1. Initialize two arrays $\mathcal{A}_1$ and $\mathcal{A}_2$ each of size $2IJ$ with zero. The elements of $\mathcal{A}_k$ correspond to $(a, b) \in A_{q,s}$ and are denoted by $\mathcal{A}_k[(a, b)]$.

2. For each element $(p, X - r) \in \mathcal{F}_1$ add $\log p$ to $\mathcal{A}_1[(a, b)]$ if $a \equiv br \pmod{p}$. Optionally do analogous operations for small powers of p.

3. For each element $(p, X - r) \in \mathcal{F}_2$ add $\log p$ to $\mathcal{A}_2[(a, b)]$ if $a \equiv br \pmod{p}$. Optionally do analogous operations for small powers of p.

4. Identify all (a, b) such that $\mathcal{A}_k[(a, b)] \geq \log(|F_k(a, b)|) - t_k$ for $k = 1, 2$ where t_k are parameters. The set of these (a, b) is called $\mathcal{S}$.

5. For each $(a, b) \in \mathcal{S}$ compute $F_k(a, b)$, do trial division up to B_k (this can use information of the preceding steps) and find a splitting $|F_k(a, b)| = S_k^{(a,b)} R_k^{(a,b)}$ such that all prime factors of $S_k^{(a,b)}$ are below B_k and all prime factors of $R_k^{(a,b)}$ are above B_k.

6. For each $(a, b) \in \mathcal{S}$ such that $R_k^{(a,b)} \leq C_k$, $k = 1, 2$, try to factor $R_k^{(a,b)}$. The numbers C_k are parameters called cofactor bounds. If $R_1^{(a,b)}$ is L_1-smooth and $R_2^{(a,b)}$ is L_2-smooth output (a, b) as a relation.

The parameters t_k should be adjusted to C_k, i. e., a little bit bigger than $\log C_k$.

3 Polynomial selection

The aim of the polynomial selection step is to chose a polynomial pair which minimizes the total runtime. The time spent in collecting relations depends on the smoothness properties of $F_i(a, b)$, $i = 1, 2$, for typical values of (a, b). These smoothness properties heuristically depend on the size of $|F_i(a, b)|$ and the number of roots modulo small primes of f_i. The size of $|F_i(a, b)|$ depends on the degree of f_i and the size of its coefficients if the sizes of a and b are fixed. For given degrees of f_i one tries to find a polynomial pair whose coefficients are small and which has many roots modulo small primes.

Except for one method, so far there are only methods which produce acceptable polynomial pairs where one polynomial is linear. The exceptional method works only for $\deg(f_1) = \deg(f_2) = 2$, and for larger numbers, at least for numbers larger than 512 bit, the polynomial pairs produced are worse than those of other methods.

Let $\deg(f_1) = d > 1$, $\deg(f_2) = 1$, $f_1 = \sum_{i=0}^{d} a_i X^i$ and $f_2 = b_1 X + b_0$. In the following we will call f_1 algebraic polynomial and f_2 rational polynomial and, more generally, refer to many quantities belonging to f_1 resp. f_2 as algebraic resp. rational. The polynomial selection method we used (see [Mur] and [Kle]), has an outer loop over a_d, then a loop over suitable b_1 for this a_d and an inner loop over b_0. Moreover, a_d should be divisible by many small primes and b_1 should be a product of many primes $\equiv 1 \pmod{d}$.

For the 1536-bit number RSA-1536 (see [RSA]) we have chosen $d = \deg(f_1) = 7$ which seems to be better than degree six or eight. For the 2048-bit number RSA-2048 the degree f_1 was also set to 7. The best polynomial pairs for these two numbers for various degrees and parameters used in their selection can be found in appendix A.

Now we try to estimate which improvement for the quality of the polynomial pair we might get if a reasonable amount of time (e. g. 1 % of the total time) were spent for polynomial selection. We assume that, spending one time unit for the selection, we get on average one algebraic polynomial f_1 with coefficient size c. Spending 2^{d+1} time units we heuristically should get on average one algebraic polynomial with coefficient size $\frac{c}{2}$ (assuming that the coefficients behave "randomly"). This is the expected coefficient size for one time unit if the input number were 2^{d+1} times smaller. So increasing the time for polynomial selection by a factor f has the same effect as reducing the input number by a factor f. Thus the quality of the polynomial pair should be improved by a factor $\frac{T(N)}{T(\frac{N}{f})}$ where $T(x)$ is the runtime to factor a number of size x. Since we consider a quotient of T we replace it by the function T_0 defined by

$$T_0(x) = e^{(\frac{64}{9}(\log x)(\log\log x)^2)^{\frac{1}{3}}}.$$

For RSA-1536 a reasonable polynomial selection effort would be about 2^{46} times longer than that used for this estimate. This would lead to an improvement of the total runtime by a factor $\frac{T_0(2^{1536})}{T_0(2^{1490})} \approx 2.5$. For RSA-2048 a reasonable polynomial selection effort would be about 2^{60} times longer which leads to an improvement by a factor $\frac{T_0(2^{2048})}{T_0(2^{1988})} \approx 2.8$. Note that these arguments and estimates are very heuristic.

4 Estimating the sieving step

4.1 Estimating the number of relations

We first describe the general outline of sieving experiments and how we can estimate the runtime of the complete sieving step.

In the sieving step we do lattice sieving for many special-q until the number of (unique) relations is sufficiently large for the subsequent matrix step. If the number of relations exceeds the sum of the number of prime ideals in K_i with norm less than L_i, $i = 1, 2$, by a small amount, this is sufficient. So we strive for approximately $\pi(L_1) + \pi(L_2)$ relations where $\pi(x)$ is the number of primes less than x. In factorizations of smaller numbers it is usually sufficient to collect less relations, say, $0.8\cdot(\pi(L_1)+\pi(L_2))$ relations. However, the parameters in this context differ much from those chosen usually, such that we cannot rely on past experience. So we use the more conservative bound $\pi(L_1) + \pi(L_2)$.

The number of relations generated by one special-q depends mainly on the size of q and on the lengths and angle of the two vectors chosen in the lattice sieving procedure for the special-q. The second dependence can probably be reduced by adapting the shape of the sieving area, but this is not done by the program used for the experiments.

To estimate the number of (non-unique) relations for lattice sieving for a set $\mathcal{Q}$ of special-q, we choose a smaller subset $\mathcal{Q}_0 \subset \mathcal{Q}$, do lattice sieving for $\mathcal{Q}_0$ and extrapolate the number of relations. In order to get a more precise estimate the subset $\mathcal{Q}_0$ should be a "random" subset of $\mathcal{Q}$. This can be done by ordering the special-q by size and choosing every n-th special-q for some n. If the set $\mathcal{Q}$ consists of special-q in an interval $[q_0, q_1]$ which are almost uniformly distributed over the interval, it is convenient to choose special-q near the locations $q_0 + iw$, $i \in \mathbb{Z} \cap [0, \frac{q_1 - q_0}{w}]$. The width w controls the accuracy of the estimate.

Since lattice sieving produces duplicate relations, we also need to estimate the number of duplicates. Let $r_1, \ldots, r_k$ be k relations generated as described above. For each r_i we count the number n_i of special-q in $\mathcal{Q}$ which would generate the same relation r_i if lattice sieving for this special-q were done with the same parameters. This can be done easily since we get a list of candidates for special-q from the factorizations of $F_1(a, b)$ where (a, b) is the pair corresponding to r_i. For each of these candidates we check whether (a, b) is in its sieving area and whether the bounds C_j for cofactors are kept. Then the estimated fraction of unique relations is $\frac{1}{k} \sum_{i=1}^{k} \frac{1}{n_i}$.

For very large numbers like 1536-bit or 2048-bit numbers we may not be able to generate a sufficient number of relations for an estimate, since the average time to generate one relation might be several CPU days or CPU years. But we can keep track of the (a, b) pairs whose cofactors are below the bounds C_j and compute the probability that both cofactors are smooth. The sum of these probabilities is the expected number of relations. How to compute these probabilities will be explained in section 4.3.

Since we will generate only a few relations or even no relation, we cannot use the procedure above to estimate the number of duplicates. Therefore we use the Dickman ρ-function and some simplifications for getting an upper bound on the number of duplicates.

Let $g_i(x)$ be a function which roughly approximates the maximum of $|F_i(a, b)|$ where (a, b) ranges over a sieving area of size x. If, for example, the absolute size of the coefficients of F_i is about c one may choose $g_i(x) = c \cdot x^{\frac{\deg(F_i)}{2}}$. Let $n_{rel,q}$ be the number of relations obtained from one special-q $(q, X - s)$ (we suppress $X - s$ in the following) and let $n_{dup,q,q'}$ be the number of relations obtained from q and from q', i. e. , the number of duplicates between q and q'. Then the number of relations is $n_{rel} = \sum_{q \in \mathcal{Q}} n_{rel,q}$ and the number of duplicates is $n_{dup} \leq \frac{1}{2} \sum_{q \in \mathcal{Q}} \sum_{q \neq q' \in \mathcal{Q}} n_{dup,q,q'}$ ($\leq$ because some relations can arise from three or more special-q). We will now approximate $n_{rel,q}$ and $n_{dup,q,q'}$ in such a way that the quotient $\frac{n_{dup}}{n_{rel}}$ heuristically only increases.

Let A be the size of the sieving area for one special-q. The size of the intersection of the sieving areas for q and q' is heuristically less than $\frac{A \cdot \gcd(q,q')}{\max(q,q')}$. The values of $F_1(a,b)$ for (a,b) in this intersection are divisible by $\frac{qq'}{\gcd(q,q')}$. We now assume that the lattice siever finds all relations with $F_1(a,b)$ being L_1-smooth and $F_2(a,b)$ being L_2-smooth. If we replace the value of $|F_i(a,b)|$ by its maximum over all (a,b) of the sieving area, we get:

$$n_{rel} = \sum_{q \in \mathcal{Q}} A\rho\left(\frac{\log g_1(qA) - \log q}{\log L_1}\right)\rho\left(\frac{\log g_2(qA)}{\log L_2}\right)$$

and

$$\begin{aligned} n_{dup} &= \frac{1}{2}\sum_{q \in \mathcal{Q}} \sum_{q \neq q' \in \mathcal{Q}} \frac{A \cdot \gcd(q,q')}{\max(q,q')}\rho\left(\frac{\log g_1(\min(q,q')A) - \log \frac{qq'}{\gcd(q,q')}}{\log L_1}\right) \times \\ &\quad \times \rho\left(\frac{\log g_2(\min(q,q')A)}{\log L_2}\right). \end{aligned}$$

Replacing the value of $|F_i(a,b)|$ by its maximum over all (a,b) should increase the quotient $\frac{n_{dup}}{n_{rel}}$ since the arguments of the ρ-function in the formula for n_{dup} are not bigger than the arguments in the formula for n_{rel} and the function $\frac{\rho(x)}{\rho(x+c)}$, $c > 0$, is increasing.

In our application the special-q consist of two prime factors, giving a fourfold summation in the formula for n_{dup}. This can be evaluated by grouping primes in intervals of the form $[h^n, h^{n+1}[$, $n \in \mathbb{N}$, for some $h > 1$ and approximating the number of primes in such an interval by the prime number theorem.

Note that heuristically the estimate for the fraction of duplicates is higher than the actual fraction. But if we choose parameters in such a way that the estimated fraction of duplicates is $\frac{1}{2}$, the error will be at most a factor 2.

4.2 Selecting parameters

In this section we describe the interdependence between various parameters of the lattice siever and how the parameters for the sieving experiments have been chosen. The main restriction from hardware is limited memory. We mainly consider PCs with 2 GB memory.

The lattice siever uses 20 byte per factor base element, 4 byte for the prime, 4 byte for its root, 4 byte for the current position in the sieving area and 8 byte for two vectors needed for calculating the position of the next sieving event. Moreover, some buffers whose sizes depend on the sizes of the factor bases and the size of the sieving area are necessary, but the total size of these buffers is usually smaller than that for the factor base data. For a PC with 2 GB memory this restricts the total size of the factor bases to about 70 to 80 million elements. Since the values of $F_1(a,b)$ are bigger than those of $F_2(a,b)$ for most of the sieving area, it is better to choose the factor base bound B_1 bigger than B_2. Therefore the values $B_1 = 11.5 \cdot 10^8$ and $B_2 = 2.5 \cdot 10^8$ were chosen.

Asymptotically the factor base sizes grow like the square root of the runtime. If costs of memory are not considered, optimal factor base sizes for a 200-digit number require about 1 GB. Extrapolating this, using the function T_0 of section 3, yields a memory size of about 60 TB for 1536-bit numbers (or 6 PB for 2048-bit numbers). This is only a very rough estimate, not considering an actual implementation. Since our factor base is much smaller than an optimal one, the number of relations in a given sieving area is much smaller than for an optimal factor base. One can compensate this a bit by increasing the cofactoring bounds C_i which also increases the time spent in the cofactorization step. This is discussed in the next section.

In lattice sieving the actual sieving phase for a special-q is preceded by a phase in which a transformation of coordinates has to be done for each factor base element. If the sieving area is chosen too small, initialization costs will dominate the runtime. So it should not be chosen too small.

Choosing a larger sieving area will reduce the number of special-q needed for generating enough relations. It also leads to less duplicates but to slightly higher values $F_i(a,b)$, thus less relations for a fixed number of (a,b) pairs. From factorizations of smaller numbers it seems to be better to use a sieving area which is not too big, so we choose it to be of size $2^{16} \times 2^{15}$. Another reason for this choice is that the current lattice sieving program cannot handle sieving areas of size above 2^{31}.

For the number under consideration and the parameter choices above, the size of the largest special-q will be larger than the large prime bounds L_i. So a relation for a special-q will only be useful if at least two relations were generated for this special-q. Since this will usually not be the case, we only consider composite special-q. To reduce the number of duplicates we restrict ourselves to special-q consisting of exactly two prime factors and also set a bound p_{min} on the minimal size of the prime factors.

For the large prime bounds L_i we set $L = L_1 = L_2$ and extrapolate L from completed factorizations of smaller numbers. Then p_{min} and the interval $[q_{min}, q_{max}]$ with $q_{max} \approx 2q_{min}$ for the special-q are chosen, such that the following two conditions are satisfied:

- The expected number of duplicates is less than 50%.

- The expected number of relations is about $4\pi(L)$, $\pi(x)$ being the number of primes less than x.

This would give us enough relations to complete the factorization if the sieving step were carried out.

4.3 Cofactoring

Here we consider the problem of finding good strategies for doing the cofactoring. We also discuss how to compute the expected cost and yield of a certain strategy.

In the following we will consider three factoring algorithms (see for example [Coh]) for doing the cofactoring, namely the Multiple Polynomial Quadratic Sieve (MPQS), Pollard's $p-1$ method and the Elliptic Curve Method (ECM). For the purposes of cofactoring this selection of algorithms seems to be most appropriate.

MPQS is an algorithm which (heuristically) always factors the input number, and its runtime depends mainly on the size of the input number. The $p-1$ algorithm has additional parameters $B_1^{(p-1)}, B_2^{(p-1)} \in \mathbb{N}$. For fixed parameters $B_1^{(p-1)}$, $B_2^{(p-1)}$ this algorithm may or may not give a splitting of the input number. The probability that a number is split depends on $B_1^{(p-1)}$, $B_2^{(p-1)}$ and the size of the prime factors of this number. The runtime depends on the parameters $B_1^{(p-1)}$, $B_2^{(p-1)}$ and on the size of the input number. The ECM algorithm behaves similarly with parameters $B_1^{(ECM)}, B_2^{(ECM)} \in \mathbb{N}$. It has an additional parameter E, an elliptic curve. When carefully changing the elliptic curve E for ECM, the runtime and the probability of splitting will not depend on E. The difference between $p-1$ and ECM is that $p-1$ with fixed parameters will always find the same prime divisors, whereas ECM with fixed parameters but a different E will find different prime divisors.

Using MPQS usually gives a complete factorization of the input number, but using $p-1$ or ECM only gives a splitting into smaller factors, possibly composite. We make the following
Assumption: When a splitting of a cofactor has been found, the time to complete the factorization is negligible.
This assumption will be discussed below.

We will call a factoring method one of the following three algorithms:

- MPQS,
- $p-1$ with fixed parameters $B_1^{(p-1)}$ and $B_2^{(p-1)}$ or
- ECM with fixed parameters $B_1^{(ECM)}$ and $B_2^{(ECM)}$.

Note that ECM with different parameters will be considered as different factoring methods. We say that a factoring methods F is applied to a number, if the number is used as input for the algorithm F.

A finite sequence of factoring methods together with the information whether the factoring method shall be applied to the algebraic or to the rational cofactor will be called a strategy. For simplicity we assume that there are no two factoring methods of type $p-1$ which are applied to the same cofactor. Applying a strategy to a pair of cofactors shall mean that the factoring methods of the sequence are successively applied to their respective target numbers, avoiding unnecessary work. This means that we abort if a prime factor $> L$ is found and that a factoring method is skipped if its target has already been split.

For $i = 1, 2$ and $m > 1$ let $M_i(n; n_1, \ldots, n_m)$, $n_1 \leq n_2 \leq \ldots \leq n_m$ be the number of n-bit composites which have no prime divisor $\leq B_i$ and which are a product of m prime numbers of size $n_1, \ldots, n_m$-bit. These numbers can be approximated using the prime number theorem. For a fixed n there are only finitely many m-tuples $(n_1, \ldots, n_m)$ such that $M_i(n; n_1, \ldots, n_m)$ is non-zero. For simplicity we make the
Assumption: The large prime bounds L_i are exact powers of 2.
We call $(n; n_1, \ldots, n_m)$ L_i-smooth if $2^{n_k} \leq L_i$ for $k = 1, \ldots, m$.

To compute the cost $\text{cost}(S; r_1, r_2)$ and yield $\text{yield}(S; r_1, r_2)$ for a strategy S applied to a pair of cofactors of size r_1-bit and r_2-bit, we also need the (average) runtimes of the factoring methods in the strategy and the probabilities that these factoring methods find an r-bit factor for $r \leq \max(r_1, r_2)$. We denote these data by $c(F, n)$ for the cost for an n-bit input number and by $p(F, r)$ for the probability to find an r-bit prime factor where F is a factoring method. These data can be precomputed. Then the calculation of the cost and yield for a strategy $S = (FM_j, s_j)$, $j = 1, \ldots, l$, FM_j: factoring methods, $s_j \in \{1, 2\}$, applied to a pair of cofactors of size r_1-bit and r_2-bit is done as follows:

1. Let k_i be the number of non-zero $M_i(r_i; n_1, \ldots, n_m)$ for $i = 1, 2$. Initialize two arrays A_1 and A_2 of length k_1 resp. k_2 with the non-zero numbers $M_i(r_i; n_1, \ldots, n_m)$. We refer to the entry of A_i containing $M_i(r_i; n_1, \ldots, n_m)$ as $A_i[n_1, \ldots, n_m]$.

2. Let a_i be the sum of the entries of A_i and $a = a_1 a_2$. Set $c \leftarrow 0$, $g_1 \leftarrow 0$ and $g_2 \leftarrow 0$ (in the steps below c will be the cost up to that step and g_i will be the number of smooth r_i-bit cofactors detected up to that step).

3. For $j = 1, \ldots, l$ do the following:

(a) Let b_i be the sum of the elements of A_i. Set $b_{3-s_j} \leftarrow b_{3-s_j} + g_{3-s_j}$ and $b \leftarrow b_1 b_2$ (b is the number of pairs of cofactors for which we do not know at this point whether they are smooth or not).

(b) Set $c \leftarrow c + b \cdot c(FM_j, r_{s_j})$.

(c) Depending on the type of FM_j do:

- MPQS: Set

$$g_{s_j} \leftarrow g_{s_j} + \sum_{(r_{s_j}; n_1, \ldots, n_m) \text{ is } L_{s_j}\text{-smooth}} A_{s_j}[n_1, \ldots, n_m]$$

  and set all entries of A_{s_j} to zero.

- $p-1$: For all entries $A_{s_j}[n_1, \ldots, n_m]$ of A_{s_j} compute

$$q = \prod_{k=1}^{m} (1 - p(p-1, n_k)),$$

  add $A_{s_j}[n_1, \ldots, n_m] \cdot (1-q)$ to g_{s_j} if $(r_{s_j}; n_1, \ldots, n_m)$ is L_{s_j}-smooth and set $A_{s_j}[n_1, \ldots, n_m] \leftarrow A_{s_j}[n_1, \ldots, n_m] \cdot q$.

- ECM: Analogous to $p-1$, replacing $p-1$ by ECM.

4. Set $\text{cost}(S; r_1, r_2) \leftarrow \frac{c}{a}$ and $\text{yield}(S; r_1, r_2) \leftarrow \frac{g_1 g_2}{a}$.

For fixed bit sizes (r_1, r_2) of the cofactors the procedure above can be used to compute cost and yield for several strategies. To select the best of these strategies one chooses a target rate t and selects the strategy S such that

$$\text{yield}(S; r_1, r_2) - t \cdot \text{cost}(S; r_1, r_2)$$

is maximal for S. This is done for all (r_1, r_2) and gives a collection of strategies depending on t. In practice, the best t can be found by sieving experiments. In an ideal situation t will be the quotient of the total yield by the total runtime.

For the calculation of the cost and the yield of a strategy we made the assumption that almost all of the time is spent in the first splitting of a cofactor. This assumption is true for cofactors consisting of two prime factors. For cofactors consisting of three prime factors the assumption also holds most of the time since the effort for finding the first splitting is usually bigger than the subsequent splitting of the smaller composite. However, for cofactors consisting of more than three prime factors the assumption seems to be no longer true. To reduce the time for subsequent splittings we use an ad hoc strategy consisting of ECM and MPQS such that the probability that a smooth cofactor is not found is less than 10%. Using this ad hoc strategy some relations may be missed, in the worst case 19%. Even with this ad hoc strategy, for RSA-1536 twice as much time is spent for subsequent splittings as for the first splitting (and for RSA-2048 the quotient is about 7).

4.4 Results

This section describes the results of the sieving experiments and gives estimates about the complexity of the complete sieving step. A PC always refers to an Athlon64 CPU with 2.2 GHz (or an Opteron) and the specified amount of memory (see appendix B for technical data of the PCs used in the sieving experiments).

The sieving experiments were conducted as described in the previous sections: for a small, uniformly distributed subset of special-q lattice sieving was done and the expected yield was obtained by summing up smoothness probabilities. Also the time spent by the sieving program was measured. We summarize the main restrictions for the choice of parameters:

- The set of special-q is chosen such that the expected fraction of duplicates will be less than $\frac{1}{2}$.
- The expected number of relations shall be at least $4\pi(L)$.
- The expected number of relations is obtained as 0.81 multiplied by the sum of the smoothness probabilities of cofactor pairs passing the cofactor bounds C_i (0.81 is a lower bound on the probability that a smooth pair of cofactors is detected if both cofactors are split).

From the expected yield and the time measurements of the sieving experiments one can compute the number of PCs needed to complete the collection of relations within one year.

For RSA-1536 two sets of parameters were chosen, one for PCs with 2 GB of memory and one for PCs with 3.5 GB of memory. In the second variant the factor base bounds are roughly twice as large as in the first variant. There is a limitation of 2^{31} for factor base bounds, so the algebraic factor base bound in the second variant was set to 2147480000.

The parameters were as follows:

	variant 1	variant 2
(B_1, B_2)	$(11.5 \cdot 10^8, 2.5 \cdot 10^8)$	$(\approx 21.5 \cdot 10^8, 5 \cdot 10^8)$
(L_1, L_2)	$(2^{50}, 2^{50})$	$(2^{50}, 2^{50})$
(C_1, C_2)	$(2^{320}, 2^{240})$	$(2^{320}, 2^{240})$
$[q_{min}, q_{max}]$	$[1 \cdot 10^{20}, 1.85 \cdot 10^{20}]$	$[0.9 \cdot 10^{20}, 1.65 \cdot 10^{20}]$
p_{min}	$2^{27} \approx 1.3 \cdot 10^8$	$2^{27} \approx 1.3 \cdot 10^8$
# special-q	$\approx 71.9 \cdot 10^{16}$	$\approx 62.9 \cdot 10^{16}$

The sieving experiment gave the following result:

	variant 1	variant 2
time per special-q	207.8 s	212.9 s
relations per special-q	$0.00023863 \cdot 0.81$	$0.0002693 \cdot 0.81$
necessary number of special-q	$69.4 \cdot 10^{16}$	$61.5 \cdot 10^{16}$
number of PC years	$4.6 \cdot 10^{12}$	$4.2 \cdot 10^{12}$

For RSA-2048 the following parameters were used for the sieving experiment for PCs with 2 GB memory:

(B_1, B_2)	$(11.5 \cdot 10^8, 2.5 \cdot 10^8)$
(L_1, L_2)	$(2^{57}, 2^{57})$
(C_1, C_2)	$(2^{400}, 2^{260})$
$[q_{min}, q_{max}]$	$[9 \cdot 10^{24}, 15 \cdot 10^{24}]$
p_{min}	$2^{35} \approx 340 \cdot 10^8$
# special-q	$\approx 335 \cdot 10^{20}$

The sieving experiment gave the following result:

time per special-q	245 s
relations per special-q	$59 \cdot 10^{-8} \cdot 0.81$
necessary number of special-q	$315 \cdot 10^{20}$
number of PC years	$25 \cdot 10^{16}$

4.5 Possible improvements

In this section we discuss possible improvements. Algorithmic improvements or the use of special purpose hardware are not considered.

First of all, in the estimates above some partial estimates were quite conservative:

- Spending more time for the selection of polynomial pairs will very probably give an improvement, heuristically of a factor 2 or 3 (see section 3).

- The estimate for the fraction of duplicates is quite pessimistic. By generating a few thousand relations (which already takes about 100 CPU years for RSA-1536) one can get a better estimate. However, since the fraction of duplicates is at most 50%, the improvement will be less than a factor 2.

- In factorizations of smaller numbers it is usually sufficient to generate about $1.6\pi(L)$ relations. If this is also true for the numbers and parameters described above, this will give an improvement by a factor 1.25.

- A more careful analysis of cofactoring strategies, also addressing the problem of subsequent splittings (see section 4.3), will probably improve the performance. The gain of such an improvement is difficult to estimate, but it seems to be less than a factor 4.

- In general, one can search for better parameters. In this estimate the parameters were chosen with the conservative estimates (see above) in mind. If one of the previous points leads to an improvement, a better choice of parameters will probably give another improvement.

Next, there are some improvements by changing parameters:

- Sieving over prime special-q below $L = 2^{50}$ resp. $L = 2^{57}$ and increasing the sieving area per special-q might reduce the number of duplicates. However, the average size of $|F_i(a, b)|$ might increase. Moreover, the lattice sieving program has to be adapted to this situation and there are some technical problems which might reduce the performance of the sieving program. Hence it is not clear whether this will give an improvement.

- One might also collect relations where a few prime factors of the factorizations of $|F_i(a, b)|$ are bigger than L. This is usually only a small extra effort. By combining the additional relations generated in this way into L-smooth relations a big fraction of the sieving time might be saved. However, it is difficult to estimate the number of relations needed.

The main restriction for the experiments in this report is the limited amount of memory and the limitations of the software (the factor base bounds have to be below 2^{31} and the maximal size of the sieving area is 2^{31}). Here we circumvented the problem of small memory by spending more time in cofactoring. However, there are other possibilities:

- One could use a hard disk to increase the amount of memory. In this case most of the factor base information will be stored on the hard disk. Accessing these information will be slow but still faster than generating them on the fly (generating a table of primes and computing the roots of F_i modulo these primes). However, using a hard disk seems to be slower than the next approach.
- One could use a parallelized lattice siever, distributing the factor base elements among the nodes. Each node generates sieving events for the part of prime ideals residing on the node and sends the events to their destination node. For this approach fast communication between the nodes might be very important. The approach seems to be suitable for multi-core processors.

The effect of increasing the available memory is hard to estimate without doing sieving experiments. Therefore the following should be considered more as a guess than an estimate.

For RSA-1536 we increase the factor base bounds to $B_1 = 16 \cdot 10^{12}$ and $B_2 = 4 \cdot 10^{12}$, keep the large prime bounds at $L_1 = L_2 = 2^{50}$ and set the cofactor bounds to $C_1 = C_2 = 2^{150}$. The sieving area size will be increased to $2^{23} \times 2^{22}$ and the special-q will be chosen as prime numbers near $60 \cdot 10^{12}$. Using the Dickman ρ-function and integrating smoothness probabilities we estimate that about 32 relations are generated per special-q. We think that this yield is sufficient since there are less duplicates when using prime special-q and since the parameters are not exceptional, thus less than $\pi(L_1) + \pi(L_2)$ relations should be sufficient. Comparing the number of relations per sieving area and taking into account that we should have less duplicates, we estimate that the size of the total sieving area has decreased by a factor 10 to 15.

The time per area will change in two ways. First, for this variant with smaller cofactor bounds there will be less time spent in cofactoring, probably giving a speed-up by a factor 2. Second, much more time is spent in the sieving part since the factor bases are much bigger. Here the factor is harder to estimate since many implementation specific details and probably also the cache structure play a role. We optimistically assume that the factor is 2 so that it cancels the other 2 above.

In summary we guess that one might gain a factor 10 to 15 by providing enough memory. If the memory is made available via a network (parallelized siever), one has to take communication costs into account, so the factor is smaller. For RSA-2048 similar arguments give a factor 20 to 30.

5 Summary

From the sieving experiments described in this report one can derive several estimates for the complexity of the relation collection step for 1536-bit and 2048-bit numbers. The first variant called "upper bound" is a pessimistic estimate which makes quite conservative assumptions at several points. By considering improvements from a reasonable time for polynomial selection and taking into account another factor 2 for several improvements described at the beginning of section 4.5, we get the next variant. Finally, we consider an optimistic variant in which we assume that a parallel siever is available which improves the performance by a factor 10 for RSA-1536 resp. 20 for RSA-2048 (see the last part of the previous section). Moreover, the improvements from polynomial selection are assumed in this variant.

The number of PCs needed to complete the collection of relations within one year for these three variants is as follows:

variant	RSA-1536	RSA-2048
upper bound	$4.6 \cdot 10^{12}$	$25 \cdot 10^{16}$
improved upper bound	$0.92 \cdot 10^{12}$	$4.4 \cdot 10^{16}$
parallel siever	$0.18 \cdot 10^{12}$	$0.4 \cdot 10^{16}$

Notice that these estimates are not lower bounds for the complexity since improved implementations, a better understanding of the involved parameters or algorithmic improvements can lower them further.

References

[Coh] H. Cohen, *A Course in Computational Algebraic Number Theory*, GTM 138, Springer, 1993.

[Kle] T. Kleinjung, *On Polynomial Selection for the General Number Field Sieve*, Mathematics of Computation 76, 2006, p. 2037-2047.

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

[Mur] B. A. Murphy, *Polynomial selection for the Number Field Sieve Integer Factorization Algorithm*, PhD thesis, The Australian National University, 1999.

[RSA] *RSA Challenge*, see `http://www.rsasecurity.com/rsalabs/challenges/factoring/index.html`

6 Appendix A: Polynomial pairs

For the 1536-bit number RSA-1536 polynomial pairs have been selected for degrees $d = \deg(f_1) = 6$, $d = 7$ and $d = 8$. Comparing the yield of the best polynomial pairs for each degree, it seems that $d = 7$ is optimal for this number. For the 2048-bit number RSA-2048 polynomial selection was done for degrees $d = 7$ and $d = 8$. For the sieving parameters used in this report degree $d = 7$ seems to be better.

The best polynomial pairs for these numbers and degrees are listed below. Moreover, some parameters of the polynomial selection are given, namely

- the search interval for the leading coefficient a_d of f_1
- the multiplier μ for a_d, i. e., the search was restricted to leading coefficients a_d which are divisible by μ
- the number k of prime factors of the leading coefficient b_1 of f_2.

Notice that not all a_d of the search interval divisible by μ have been considered, but only sufficiently many for a one day search on an 1 GHz Pentium III.

RSA-1536, $d = 6$:

interval	$[10^{44}, 10^{44} + 10^{20}]$
μ	$18410000400 = 2^4 \cdot 3^2 \cdot 5^2 \cdot 11 \cdot 17 \cdot 23 \cdot 29 \cdot 41$
k	18

$$\begin{aligned} f_1 = {} & 100000000000000000000000000000363596066133200x^6 \\ & +13711306793107134761462289661826225486538486127 62130x^5 \\ & -570766617961064364687337561437435320448189186003746 1369533x^4 \\ & -16096370008727112517414312648727659841841247443587052967 2688061x^3 \\ & +16444781945684723170780958743074454415194164537981037428 7951485299879x^2 \\ & +126951073807261611810741318083567227802680760657702282929404 1326844917810x \\ & -617348491598850203388187460000376982421861762957544759773443922 61897665398 03 \end{aligned}$$

$$\begin{aligned} f_2 = {} & 2699084276818025737646973987324783468507 3761353x \\ & -5141682217651853177521549768125122593221948973547321520298700609576629 \end{aligned}$$

RSA-1536, $d = 7$:

interval	$[10^{36}, 10^{36} + 10^{20}]$
μ	$26771144400 = 2^4 \cdot 3^2 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 \cdot 23$
k	16

$$\begin{aligned} f_1 \;=\;& 1000000000000000037811142945142568000x^7 \\ & -4371436535027159521885410729553145159955614 4x^6 \\ & +303287546715162928320354682357442025586744759016 16x^5 \\ & -2549632694806349159734389882357104975174959388666079090x^4 \\ & -743476621449070705426118043952107910813296873947226569967 6x^3 \\ & +26941031676370465249694738904702978153213780731967578381868353x^2 \\ & +4586821874001273393055958425616599274775638515656677576074764078 6x \\ & -4886779488245100844142426433573527357308824126462414280437158010608 0 \end{aligned}$$

$$\begin{aligned} f_2 \;=\;& 205334469089201645309341276310576122905417473x \\ & -7856571290567634945150691614513931635402668625054838502550321 \end{aligned}$$

RSA-1536, $d = 8$:

interval	$[10^{29}, 10^{29} + 10^{20}]$
μ	$45668422800 = 2^4 \cdot 3^2 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 19 \cdot 23 \cdot 29$
k	14

$$\begin{aligned} f_1 \;=\;& 10000000000005510522854803968 00x^8 \\ & -1013339316952850682823095101191079189 0x^7 \\ & +901780358656075872817255177277099971864270x^6 \\ & -906702884423786095887754389464962675306322995 54x^5 \\ & -1433483579437420552603953601635897546272831045868 2x^4 \\ & +35831061779132358936685717720236112782208200557413449 2x^3 \\ & +3311004971311054585028507972984542046769718717335926642 4x^2 \\ & -2873709152308757606196763595234926070605854472799205723744 85x \\ & -1324769005836618431207183749939005477858996733887187196433842 5 \end{aligned}$$

$$\begin{aligned} f_2 \;=\;& 837891974558999977192248015447971052 73x \\ & -14398887016497119757518789075346667020278828780321978 02 \end{aligned}$$

RSA-2048, $d = 7$:

interval	$[10^{56}, 10^{56} + 10^{20}]$
μ	$26771144400 = 2^4 \cdot 3^2 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 \cdot 23$
k	16

$$\begin{aligned}
f_1 \quad = \quad & 100000000000000000000000000000000000073338147424598156000x^7 \\
& +83958826335886062384725839675320017089308353422946100747992593111x^6 \\
& -162432384658718261794866293473493276645975758781703095785325697861958 6x^5 \\
& -16297922667449801701018887260891713344933788688216935280929872819397554 24x^4 \\
& +33842219052921654312275628323473891052030445427581764495124943311088430533903x^3 \\
& +971691263975100658276949833182782247501733674148880145256942024691408926689 68x^2 \\
& -127658905721988738113694551290560677881909461129950694195199704883206539032920354252x \\
& +707807432440667114666929289400773218280007663247633921344449396149362951158751254813 44
\end{aligned}$$

$$\begin{aligned}
f_2 \quad = \quad & 41123412601722865693107152959526055267629x \\
& -114112405465044179083292898071688380999575821246316812608380320780980683979836071
\end{aligned}$$

RSA-2048, $d = 8$:

interval	$[10^{46}, 10^{46} + 10^{20}]$
μ	$45668422800 = 2^4 \cdot 3^2 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 19 \cdot 23 \cdot 29$
k	14

$$\begin{aligned}
f_1 \quad = \quad & 10000000000000000000000000001981831814793220400x^8 \\
& -2215229217222143068192667641819120750038383822406401934x^7 \\
& +19166092335521434095569066969479762212934154322946374583356 63x^6 \\
& +26661673829180097195647792070006982037560728914378888764588753212x^5 \\
& -125680460720267648430568998956020777206751911537366072091101816865 31x^4 \\
& -30332231701156503137832844039609655625808443207936237027382905221701472x^3 \\
& +8778624409291627225152939964712907763932508250115493296758218503675017729x^2 \\
& +69617883549877454258318960136408802583265382413980559550630978428816600900 04x \\
& -756525676219717966880555107720645406038893474949813849975176062414350072004951
\end{aligned}$$

$$\begin{aligned}
f_2 \quad = \quad & 32279094371463783323986398433012330 73x \\
& -199602626981280818393607284895114464259910689488947173899674283165812114
\end{aligned}$$

7 Appendix B: Technical data of the PC

Below are listed some technical data of the PCs used for the sieving experiments.

CPU	AMD Athlon(tm) 64 X2 Dual Core Processor 4200+
Clock rate	2.2 GHz
L2 cache size	512 kB
Memory	DDR2 533 MHz
Memory size	4 GB (3.5 GB available)
Mainboard	Asus M2N-VM