Factoring via Strong Lattice Reduction Algorithms

Technical Report

Harald Ritter
Dept. of Math./Comp. Science
University of Frankfurt
P. O. Box 111932
60054 Frankfurt on the Main
Germany
ritter@cs.uni-frankfurt.de

Carsten Rössner
Dept. of Math./Comp. Science
University of Frankfurt
P. O. Box 111932
60054 Frankfurt on the Main
Germany
roessner@cs.uni-frankfurt.de

June 13, 1997

Abstract

We address to the problem to factor a large composite number by lattice reduction algorithms. Schnorr [Sc93] has shown that under a reasonable number theoretic assumptions this problem can be reduced to a simultaneous diophantine approximation problem. The latter in turn can be solved by finding sufficiently many ℓ_1–short vectors in a suitably defined lattice.

Using lattice basis reduction algorithms Schnorr and Euchner applied the reduction technique of [Sc93] to 40–bit long integers. Their implementation needed several hours to compute a 5% fraction of the solution, i.e., 6 out of 125 congruences which are necessary to factorize the composite.

In this report we describe a more efficient implementation using stronger lattice basis reduction techniques incorporating ideas of [SH95] and [R97]. For 60–bit long integers our algorithm yields a complete factorization in less than 3 hours.

1 Introduction

The security of many public key cryptosystems relies on the hardness of factoring large numbers. In fact, no algorithm is known which given a number N computes its prime factor representation in deterministic polynomial time in the input length $O(\log_2 N)$. The fastest known factoring algorithm is the Number Field Sieve (*NFS*) [LL93, Sch93]. The NFS takes expected running time $O(e^{(1.923+o(1))(\log N)^{1/3}(\log\log N)^{2/3}})$ where log denotes the natural logarithm. Like many other factoring algorithms the goal of the NFS and Schnorr's reduction is to find positive integers x and y such that $x^2 \equiv y^2 (\bmod\ N)$. If $x \not\equiv \pm y (\bmod\ N)$ then $gcd(x \pm y, N)$ is a non–trivial divisor of N. In the NFS this goal is accomplished by determining sufficiently many smooth numbers, i.e., numbers with small prime factors, both in the ring of integers $\mathbb{Z}$ and in an algebraic extension (the number field) $K = \mathbb{Z}[\alpha]$. If f is the minimal polynomial for α and m a small $(\bmod\ N)$–zero of f the integers x^2, y^2 are computed from the product of smooth integers and algebraic numbers $(a + b\,m)$, $(a + b\,\alpha)$, respectively, where the product is taken over pairs (a, b) with relatively prime integers. Similarily, in Schnorr's reduction — the Diophantine Approximation Algorithm (*DAA*) [Sc93] — the numbers x and y are constructed from the product of smooth numbers u and $u - v\,N$ with pairs (u, v) of relatively prime integers. The pairs are generated from lattice vectors which are sufficiently close to a fixed point $\boldsymbol{N}$. Given constants α, $c > 1$ and the first t primes $p_1, \ldots, p_t$ with $p_t = (\log N)^\alpha$ the DAA proceeds as follows:

- first, find at least $t+2$ non–trivial integral vectors $e := (e_1, \ldots, e_t)^\top \in \mathbb{Z}^t$ which are both 'good' and ℓ_1–short approximations of $\log N$, i.e.,

$$\left| \sum_{i=1}^{t} e_i \log p_i - \log N \right| \leq N^{-c}\, p_t^{o(1)} \quad \text{and} \tag{1}$$

$$\sum_{i=1}^{t} |e_i \log p_i| \leq (2\,c-1)\log N + 2\log p_t \quad ; \tag{2}$$

- secondly, for each pair (u, v) set $u = \prod_{a_j>0} p_j^{a_j}$ and $v = \prod_{a_j<0} p_j^{|a_j|}$ and factorize $|u - v\,N|$ over the primes $p_1, \ldots, p_t$; a factorization $|u - v\,N| = \prod_{j=1}^{t} p_j^{b_j}$ yields a non–trivial congruence $u \equiv \pm(u - v\,N) (\mathrm{mod}\ N)$;

- $t+2$ of this congruences suffice to construct the integers x and y which give a non–trivial divisor of N with probability of at least $1/2$.

Schnorr has shown that the diophantine approximation problem, in turn, can be reduced in probabilistic polynomial time to the problem to find at least $t+2$ sufficiently ℓ_1–near vectors to $\boldsymbol{N} = (\underbrace{0, \ldots, 0}_{t}, N^c \log N)$ in the lattice $L_{\alpha,c} \subseteq \mathbb{R}^{t+2}$ generated by the row vectors of the matrix

$$\begin{bmatrix} \log 2 & 0 & \ldots & 0 & N^c \log 2 \\ 0 & \log 3 & \ldots & 0 & N^c \log 3 \\ \vdots & 0 & \ddots & 0 & \vdots \\ 0 & \ldots & \ldots & \log p_t & N^c \log p_t \end{bmatrix} \quad . \tag{3}$$

In order to compute ℓ_1–near lattice vectors it suffices to use strong lattice basis reduction algorithms in the ℓ_2–norm.

Adleman [Ad95] presented a probabalistic polynomial–time reduction from the problem of factoring a composite N to the problem of finding the ℓ_2–nearest vectors of sufficiently many pairs of suitably defined lattices and fixed vectors.

Adleman [Ad95] essentially reduces the problem of factoring the composite N to the problem of enumerating at least $t+2$ sufficiently ℓ_2–short vectors in the lattice generated by the row vectors of

$$\begin{bmatrix} 1 & 0 & \ldots & 0 & N^c \log N \\ 0 & \sqrt{\log 2} & \ldots & 0 & N^c \log 2 \\ \vdots & 0 & \ddots & 0 & \vdots \\ 0 & \ldots & \ldots & \sqrt{\log p_t} & N^c \log p_t \end{bmatrix} \quad . \tag{4}$$

As [Ad95] uses a reduction to the problem of finding ℓ_2–nearest lattice vectors the $\log p_j$'s in the matrix (3) have to be replaced by the square roots of the corresponding $\log p_j$'s. Adleman does not provide an experimental analysis of his method.

We refine Schnorr's factoring algorithm where we use efficient lattice basis reduction algorithms originating from the concept of block reduction as introduced in [Sc87] and improved in subsequent papers [SE94, SH95, R97]. We also provide practical results of our methods.

2 The Factoring Algorithm

Throughout the paper let $\mathbb{R}^n$ denote the real n–dimensional vector space with the ordinary inner product $< x, y > := \sum_{i=1}^{n} x_i\, y_i$, ℓ_2–norm (i.e. Euclidean length) $\|x\|_2 :=< x, x >^{1/2} = (\sum_{i=1}^{n} x_i^2)^{1/2}$, ℓ_1–norm $\|x\|_1 := \sum_{i=1}^{n} |x_i|$ for vectors $x := (x_1, \ldots, x_n)^\top$, $y := (y_1, \ldots, y_n)^\top \in \mathbb{R}^n$. Moreover, let $\log(.)$ denote the natural logarithm function. Let N be an integer with at least two distinct prime factors.

A discrete, additive subgroup $L \subseteq I\!R^n$ is called a *lattice*. Every lattice is generated by some set of linearly independent vectors $b_1, \ldots, b_m$, called *basis* of L, i.e., $L = \sum_{i=1}^{m} \mathbb{Z}\, b_i$. Let $L(b_1, \ldots, b_m)$ denote the lattice with basis $b_1, \ldots, b_m$. Its *rank* or *dimension* is m and its determinant is $det(L) := \det[(< b_i, b_j >)_{1 \le i,j \le m}]^{1/2}$. The rank and the dimension are independent of the choice of the lattice basis. The determinant of a lattice $L(b_1, \ldots, b_m)$ can be geometrically interpreted as the m–dimensional volume of the parallelepiped spanned by the vectors $b_1, \ldots, b_m$, i.e., $det(L) = vol_m(\sum_{i=1}^{m} x_i\, b_i \mid 0 \le x_i \le 1,\ 1 \le i \le m)$.

We briefly give an outline of the factoring method. The algorithm essentially coincides with the one given in [Sc93]. Differences result from using a stronger and more refined enumeration method in step 2 :

INPUT N, rationals $\alpha, c > 1$,

1. Compute the list $p_1, \ldots, p_t$ of the first t primes smaller than $(\log N)^\alpha$ by Trial Division.

2. Let $b_0 := \boldsymbol{N}$ and $b_1, \ldots, b_t$ be the row vectors of the basis matrix of the lattice $L_{\alpha,c}$ as defined in section 1. Enumerate $m \ge t+2$ lattice vectors $z_i := \sum_{j=0}^{t} a_{i,j}\, b_j$, $a_{i,j} \in \mathbb{Z}$, with the following property:

   for $u_i := \prod_{a_{i,j}>0} p_j^{a_{i,j}}$, $v_i := \prod_{a_{i,j}<0} p_j^{|a_{i,j}|}$, the absolute value $|u_i - v_i\, N|$ factorizes completely over the primes $p_1, \ldots, p_t$.

   Define the vectors $\boldsymbol{a}_i$, $1 \le i \le m$ by $\boldsymbol{a}_i := (a_{i,0}, \ldots, a_{i,t})^\top$, where $a_{i,0} := 0$ for $1 \le i \le m$.

3. For every $i = 1, \ldots, t$ factorize $u_i - v_i\, N$ over the primes $p_1, \ldots, p_t$ and $p_0 = -1$.

   Let $u_i - v_i\, N =: \prod_{j=0}^{t} p_j^{b_{i,j}}$, $1 \le i \le t$, and define the vectors $\boldsymbol{b}_i$, $1 \le i \le m$ by $\boldsymbol{b}_i := (b_{i,0}, \ldots, b_{i,t})^\top$.

4. Find a non–trivial 0/1–solution $(c_1, \ldots, c_m)$ of the linear homogeneous system of m congruences

$$\sum_{i=1}^{m} c_i\, (\boldsymbol{a}_i\ + \boldsymbol{b}_i\ ) \ \equiv\ \boldsymbol{0}\ (\mathrm{mod}\, 2) \quad .$$

5.

$$x := \prod_{j=1}^{m} p_j^{\sum_{i=1}^{m} c_i\, (a_{i,j}+b_{i,j})/2} \ (\mathrm{mod}\, N)\ ,$$

$$y := \prod_{j=1}^{m} p_j^{\sum_{i=1}^{t} c_i\, b_{i,j}} \ (\mathrm{mod}\, N) \ = \ \prod_{j=1}^{m} p_j^{\sum_{i=1}^{t} c_i\, a_{i,j}} \ (\mathrm{mod}\, N) \quad .$$

   (The definition of x and y implies that $x^2 = y^2 \ (\mathrm{mod}\, N)$).

6. If $x \neq \pm y \ (\mathrm{mod}\, N)$ then $x + y$ or $x - y$ is a non–trivial factor of N.
   OUTPUT $gcd(x \pm y, N)$ and stop.
   Otherwise go to 4 and generate a different solution $(c_1, \ldots, c_m)$.

Remarks. 1. By the prime number theorem (c.p. [RS62]) the number t of primes less than $(\log N)^\alpha$ is at most

$$0.876\,(\log N)^\alpha/(\alpha\,\log\log N)\ .$$

This bounds the cost of step 1.

2. Steps 4–6 require that for $i = 1,\ldots,t+1$ the numbers $u_i - v_i\,N$ completely factorize over the primes $p_1,\ldots,p_t$. A hypothesis given in [Sc93] essentially states that among all pairs of integers (u,v) with prime factors less than $(\log N)^\alpha$ and $N^{c-1}/2 < v < N^{c-1}$ there is a $(\log N)^{O(1)}$ fraction with $|u - v\,N| = 1$. Using this hypothesis and a theorem of Norton and Canfield, Erdös, Pomerance [No71, CEP83] Schnorr [Sc93] has shown the following: For any $\sigma > 0$ there exists an $\epsilon > 0$ such that there are $N^{\epsilon+o(1)}$ lattice vectors z with

$$\|z - \boldsymbol{N}\ \|_1\ \leq\ (2\,c-1)\,\log N + 2\,\sigma\,\log p_t\ . \tag{5}$$

For the associated pairs of integers (u,v) this bound implies

$$|u - v\,N|\ \leq\ p_t^{1/\alpha+\sigma+o(1)}\ ,\quad \text{where}\ \ \epsilon := c - 1 - (2\,c-1)\log\log N/\log p_t\ .$$

By the detailed analysis given in [Sc93] the parameters α and c have to be chosen such that $\alpha > (2\,c-1)/(c-1)$.

We derive a more efficient method to determine the pairs (u,v). We give a necessary and sufficient condition for $|u - v\,N| \leq p_t^\sigma$ in the following

Lemma. *Let $\alpha, c > 1, \sigma > 0$ be fixed with $(\log N)^\alpha = p_t < N$. For every vector $z \in L_{\alpha,c}$ the associated pair $(u(z), v(z))$ satisfies:*
A necessary condition for $|u(z) - v(z)N| \leq p_t^\sigma$ is

$$\frac{1}{2}\|z - \boldsymbol{N}\|_1 - \left(\frac{N^c+1}{2N^c}\right)|(z-\boldsymbol{N})_{t+1}| + \log(|(z-\boldsymbol{N})_{t+1}|) < (c-\frac{1}{2})\log N + \sigma\log p_t. \tag{6}$$

A sufficient condition for $|u(z) - v(z)N| \leq p_t^\sigma$ is

$$\frac{1}{2}\|z - \boldsymbol{N}\|_1 - \left(\frac{N^c-1}{2N^c}\right)|(z-\boldsymbol{N})_{t+1}| + \log(|(z-\boldsymbol{N})_{t+1}|) < (c-\frac{1}{2})\log N + \sigma\log p_t. \tag{7}$$

The proof is straightforward by evaluating the condition $|u - v\,N| \leq p_t^\sigma$ in terms of $\max(u, v\,N)$ and $\min(u, v\,N)$ and exploiting the concaveness of the log–function. For details we refer to [R97]. For the enumeration process in step 2 of the factoring algorithm we must only enumerate lattice vectors z for which the associated pairs $(u(z), v(z))$ satisfy (6) and (7).

Now the hard task of the algorithm is the enumeration of at least $t+2$ such pairs which in turn amounts to the enumeration of at least $t+2$ sufficiently close lattice vectors to the fixed point $\boldsymbol{N}$. We achieve this by using the concept of block reduction of lattice bases as introduced in [Sc87] and improved in subsequent papers as particulary [R97].

3 Enumeration of Short Lattice Vectors

The lattice basis reduction is done via block reduction as already implemented in [SE94]. However, for the enumeration of candidates for short lattice basis vectors we use a more efficient method proposed by Ritter [R97]. It incorporates ideas of [SE94] and [SH95]. We briefly introduce the concept of block reduction.

With an ordered lattice basis $b_1, \ldots, b_m \in \mathbb{R}^n$ we associate the Gram–Schmidt orthogonalization $\hat{b}_1, \ldots, \hat{b}_m \in \mathbb{R}^n$ which can be computed together with the Gram–Schmidt coefficients $\mu_{i,j} = < b_i, \hat{b}_j > / < \hat{b}_j, \hat{b}_j >$ by the recursion $\hat{b}_1 = b_1,\ \hat{b}_i = b_i - \sum_{j=1}^{i-1} \mu_{i,j} \hat{b}_j$ for $i = 2, \ldots, m$. We define the orthogonal projections $\pi_i : \mathbb{R}^n \to \mathrm{span}(b_1, \ldots, b_{i-1})^\perp$ for $i = 1, \ldots, m$. Clearly, $\pi_i(b_j) = \sum_{t=i}^{j} \mu_{i,t} \hat{b}_t$.

An (ordered) lattice basis $b_1, \ldots, b_m$ is called *size–reduced* if $|\mu_{i,j}| \leq 1/2$ for $1 \leq j < i \leq m$. It is L^3–reduced — according to Lenstra, Lenstra and Lovász [LLL82] — with $\delta \in [1/4, 1)$ if additionally

$$\delta \, \|\pi_{k-1}(b_{k-1})\|^2 \ \leq \ \|\pi_{k-1}(b_k)\|^2 \quad , \ k = 2, \ldots, m \ .$$

[LLL82] presented the first polynomial time algorithm which approximates the shortest non–trivial vector in an m–dimensional lattice up to the factor $1/(\delta - \frac{1}{4})^{m/2}$. A lattice basis $b_1, \ldots, b_m$ is called *block reduced* with $\delta \in [1/4, 1)$ and *blocksize* β if it is size–reduced and

$$\delta \, \|\pi_i(b_i)\| \ \leq \ \|\pi_i(v)\| \ \text{ for all } v \in L(b_i, \ldots, b_{\min\{i+\beta-1, m\}}) \, , \ i = 1, \ldots, m \ ,$$

i.e., $\pi_i(b_i)$ is up to the factor δ the shortest non–trivial vector in the lattice $\pi_i(L(b_i, \ldots, b_{\min\{i+\beta-1, m\}}))$. L^3–reduced bases are a special case of block–reduced bases with $\beta = 2$. The approximation of the shortest lattice vectors by a block–reduced basis becomes tighter by increasing blocksize, i.e., for a block–reduced basis $b_1, \ldots, b_m$ of a lattice L the ratio

$$\max_{1 \leq i \leq m} \|b_i\| / \min\{r > 0 : \ \exists i \text{ lin. indep. } c_1 \ldots, c_i \in L : \ \|c_i\| \leq r\}$$

is bounded by $\gamma_\beta^{\frac{2\,(m-1)}{\beta-1}} \, \frac{m+3}{4}$, where γ_β is the Hermite constant for dimension β. Algorithms for block reduction have been proposed in [SE94, SH95] and are only known to have exponential running time $O(n\,m\,\log B)\,[O(m^2) + O(\beta)\,1/(\delta - \frac{1}{4})^{O(\beta^2)}]$ for L^3–reduced input bases with maximum Euclidean length B [Sc87, R97]. For $k = t, \ldots, j$ we define the following functions $w_t,\ \tilde{c}_t$ with integer arguments $\tilde{u}_t, \ldots, \tilde{u}_k$:

$$\begin{aligned} w_t := w_t(\tilde{u}_t, \ldots, \tilde{u}_k) \ &:= \ \pi_t(\sum_{i=t}^{k} \tilde{u}_i b_i) = w_{t+1} + \left(\sum_{i=t}^{k} \tilde{u}_i \mu_{i,t} \right) \hat{b}_t \\ \tilde{c}_t := \tilde{c}_t(\tilde{u}_t, \ldots, \tilde{u}_k) \ &:= \ \|w_t\|_2^2 = \tilde{c}_{t+1} + \left(\sum_{i=t}^{k} \tilde{u}_i \mu_{i,t} \right)^2 \|\hat{b}_t\|_2^2 \end{aligned}$$

The core of the block reduction algorithm of [SE94] is a procedure ENUM_2 that generates the shortest non–trivial lattice vector $\bar{b}$ in a β–block $\pi_j(b_j), \ldots, \pi_j(b_k)\ \ k = j + \beta - 1$ by complete enumeration in depth first search order. Ritter has shown that the running time of the enumeration is $O(\beta)\,1/(\delta - \frac{1}{4})^{O(\beta^2)}$ [R97].

For fixed $\tilde{u}_{t+1}, \ldots, \tilde{u}_m$ the enumeration order of the $\tilde{u}_t$–values is controlled by the variables Δ_t and δ_t, which are the same as in [SE94].

Procedure $\mathrm{ENUM}_2(j, k)$

INPUT $\ c_i := \|\hat{b}_i\|_2^2, \mu_{i,t} \quad \text{for} \quad j \leq t \leq i \leq k$

1. FOR $i = j, \ldots, k+1$
   $\tilde{c}_i := u_i := \tilde{u}_i := v_i := y_i := \Delta_i := 0,\ \delta_i := 1,\ w_i := (0, \ldots, 0)$
   $u_j := \tilde{u}_j := 1,\ s := t := j,\ \bar{c_j} := \|b_j\|_2^2$
2. WHILE $t \leq k$
   $\tilde{c}_t := \tilde{c}_{t+1} + (y_t + \tilde{u}_t)^2 c_t$
   IF $\tilde{c}_t < \bar{c_j}$
   THEN IF $t > j$

THEN $t := t-1$, $\Delta_t := 0$, $y_t := \sum_{i=t+1}^{s} \tilde{u}_i \mu_{i,t}$
$\tilde{u}_t := v_t := \lceil -y_t \rfloor$, $\Delta_t := 0$
IF $\tilde{u}_t > -y_t$ THEN $\delta_t := -1$
ELSE $\delta_t := 1$
ELSE $(u_j, \ldots, u_k) := (\tilde{u}_j, \ldots, \tilde{u}_k)$
$\bar{c_j} := \tilde{c}_j$
ELSE $t := t+1$
$s := \max(t, s)$
IF $t < s$
THEN $\Delta_t := -\Delta_t$
IF $\Delta_t \delta_t \geq 0$ THEN $\Delta_t := \Delta_t + \delta_t$
$\tilde{u}_t := v_t + \Delta_t$
END while
OUTPUT the minimum $\bar{c}_j$ of $c_j(u_j, \ldots, u_k)$ and the coordinates $(u_j, \ldots, u_k) \in \mathbb{Z}^{k-j+1} - 0^{k-j+1}$

The algorithm enumerates in depth first search order all nonzero integer vectors $(\tilde{u}_t, \ldots, \tilde{u}_k)$ for $t = k, \ldots, j$ satisfying $\tilde{c}_t(\tilde{u}_t, \ldots, \tilde{u}_k) < \bar{c}_j$, where $\bar{c}_j$ is the current minimum for the function $\tilde{c}_j(\tilde{u}_j, \ldots, \tilde{u}_k)$.

In the factoring algorithm we replace the enumeration step 2 by a more powerful variant due to Ritter [R97]. Ritter prunes the enumeration of points $(\tilde{u}_j, \ldots, \tilde{u}_k)$ with $\tilde{c}_t(\tilde{u}_j, \ldots, \tilde{u}_k) < \bar{c}_j$ by a heuristic approach but guarantees on each stage t a fixed probability of finding the shortest vector in $\pi_t(L(b_t, \ldots, b_k))$. The depth first search enumeration of ENUM_2 is cut off if the probability of missing an ℓ_2–shortest vector in $\pi_t(L(b_t, \ldots, b_k))$ on stage t does not exceed a given threshold. The probability on each stage t is independent of the previously visited stages. Using this pruning rule the modfied algorithm becomes exponentially faster while guaranteeing a given success rate of finding the ℓ_2–shortest vector in $\pi_j(L(b_j, \ldots, b_k))$. We give an outline of the method.
We represent enumerated points $(\tilde{u}_t, \ldots, \tilde{u}_k)$ by nodes on different stages of $(k-j+1)$–deep subtrees where the root corresponds to the k–th coordinate $\tilde{u}_k$ and coordinates $\tilde{u}_j$ to leaves. Each enumerated point $(\tilde{u}_t, \ldots, \tilde{u}_k)$ corresponds to a traversed path in the subtree and is assigned a weight $(\bar{c}_j - \widetilde{c}_t)^{\frac{t-j}{2}} / \prod_{i=j}^{k} \|\hat{b}_i\|$. By the Gaussian volume heuristic the weight may be regarded as the number of points $(\tilde{u}_j, \ldots, \tilde{u}_{t-1}, \tilde{u}_t, \ldots, \tilde{u}_k)$ with $\tilde{c}_j(\tilde{u}_j, \ldots, \tilde{u}_k) < \bar{c}_j$ for fixed coordinates $(\tilde{u}_t, \ldots, \tilde{u}_k)$. The Gaussian volume heuristic estimates the number of points of a lattice L in a sphere S *(with unifomly distributed center 'modulo' the lattice L)* as $vol(S)/det(L)$ [SH95, R97]. Given a set of paths $\{(\tilde{u}_t, \ldots, \tilde{u}_k) \in A_t\}$ traversed in depth first search order the probability of finding a remaining path with minimum $\tilde{c}_j(\tilde{u}_j, \ldots, \tilde{u}_k)$ is the fraction of the sum of all weights visited so far and the sum of all weights of all possible paths from the current stage t. The traversion of nodes is cut off if this probability exceeds a given threshold p_t. Hence, the probability of finding a shortest vector in $\pi_j(L(b_j, \ldots, b_k))$ is at least $\prod_{t=j}^{k} p_t$.
Ritter's enumeration variant in step 2 of ENUM_2 can be informally decribed as follows: (For a detailed description and analysis we refer to [R97].)

Pruned $\mathrm{ENUM}_2(j,k)$
1. initialize $A_t := B_t := 0$
2. for $t = j+1, \ldots, s$ and fixed $(\widetilde{u}_{t+1}, \ldots, \widetilde{u}_s)$ traverse all nodes $\widetilde{u}_t$ with $\widetilde{c}_t := \widetilde{c}_t(\widetilde{u}_{t+1}, \ldots, \widetilde{u}_s) < \bar{c}_j$ in ascending order w.r.t. $\widetilde{c}_t$ and add the 'weight' $(\bar{c}_j - \widetilde{c}_t)^{\frac{t-j}{2}}$ to B_t;
3. if $A_t/B_t \geq p_t$ (the probability threshold for stage t) take the next value $\widetilde{u}_t$ (or increment t if all nodes $\widetilde{u}_t$ have already been visited); otherwise add the 'weight' to A_t and decrement t;
4. if $t = j$ determine $\widetilde{u}_j$ with minimum $\widetilde{c}_j$; if $\widetilde{c}_j < \bar{c}_j$ update $\bar{c}_j = \widetilde{c}_j$, $(u_j, \ldots, u_k) = (\widetilde{u}_j, \ldots, \widetilde{u}_s, 0, \ldots, 0)$.

Schnorr and Hörner [SH95] gave a local variant of Ritter's algorithm where the enumeration of the subtree of some node is cut off if the probability of finding a shorter vector in the subtree is less than

a given threshold. However, this method does not provide a lower bound on the probability of finding a shortest vector in the whole enumeration tree for points $(\widetilde{u}_1, \ldots, \widetilde{u}_m) \in \mathbb{Z}^m$.

4 Algorithm Details and Practical Results

For our experiments we randomly generate two distinct prime numbers P and Q with equal bitlength and set $N = P\,Q$. We transform the problem of finding ℓ_2–close vectors $z \in L_{\alpha,c}$ to a fixed point $\boldsymbol{N} = (\underbrace{0, \ldots, 0}_{t}, N^c \log N)$ to the problem of finding ℓ_2–short vectors in the lattice spanned by the vector $b_0 := (1, \underbrace{0, \ldots, 0}_{t}, N^c \log N)^\top$ and the row vectors of the basis matrix of the lattice $L_{\alpha,c}$ each with an additional 0–entry in the first coordinate. Moreover, for experiments we have to approximate the floating point entries of the resulting basis matrix by rationals where we use the log–routine of Mathematica Package 2.0 with a precision of c_1 respectively $c_2 \approx c_1 - c \log_{10} N$ decimals right from the point:

$$\widetilde{B} := \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 10^{c_1} \log N \\ 0 & 10^{c_2} \log 2 & 0 & & 0 & 10^{c_1} \log 2 \\ 0 & 0 & 10^{c_2} \log 3 & & 0 & 10^{c_1} \log 3 \\ \vdots & \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & 10^{c_2} \log p_t & 10^{c_1} \log p_t \end{pmatrix}.$$

For every lattice vector $z = (z_0, \ldots, z_{t+1})^\top = \sum_{i=0}^{t} e_i\, b_i$ with $|\widetilde{z}_0| = 1$ we define the associated pair $(u(z), v(z))$ by

$$u(z) := \prod_{\substack{i=1 \\ e_0 e_i < 0}}^{t} p_i^{|e_i|} \quad \text{and} \quad v(z) := \prod_{\substack{i=1 \\ e_0 e_i > 0}}^{t} p_i^{|e_i|}.$$

Then the necessary and sufficient condition (6),(7) for $|u(z) - v(z)\, N| < p_t^\sigma$, transforms into

$$|z_0| = 1 \quad , \quad \frac{1}{2} 10^{-c_2} \sum_{i=1}^{t} |z_i| - 10^{-c_1} |z_{t+1}| + \log |z_{t+1}| < c_1 \log 10 + \sigma \log p_t - \frac{1}{2} \log N \ , \tag{8}$$

$$|z_0| = 1 \quad , \quad \frac{1}{2} 10^{-c_2} \sum_{i=1}^{t} |z_i| + 10^{-c_1} |z_{t+1}| + \log |z_{t+1}| < c_1 \log 10 + \sigma \log p_t - \frac{1}{2} \log N \,, \text{ resp.} \tag{9}$$

For our practical tests we use integral lattices by omitting the denominators of the columns in the rational basis matrix, i.e.,

$$B := \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & \lceil 10^{c_1} \log N \rfloor \\ 0 & \lceil 10^{c_2} \log 2 \rfloor & 0 & & 0 & \lceil 10^{c_1} \log 2 \rfloor \\ 0 & 0 & \lceil 10^{c_2} \log 3 \rfloor & & 0 & \lceil 10^{c_1} \log 3 \rfloor \\ \vdots & \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & \lceil 10^{c_2} \log p_t \rfloor & \lceil 10^{c_1} \log p_t \rfloor \end{pmatrix}.$$

Enumeration of pairs (u_i, v_i)
INPUT $N, t, c_1, c_2, \beta, \sigma, p, R$
1. Compute the lattice basis $B = (b_0, b_1, \ldots, b_t)^T$;
2. L^3–reduce $b_0, \ldots, b_t$ with $\delta = 0.99$, i.e., block–reduce $b_0, \ldots, b_t$ with $\delta = 0.99$ and blocksize 2 ;
3. Randomly permute the basis vectors $b_0, \ldots, b_t$;
4. Perform a block–reduction of $b_0, \ldots, b_t$ with $\delta = 0.99$ and blocksize β.
Repeat steps 3 and 4 up to R times.

For the enumeration procedure within the block–reduction of step 4 we invoke Pruned ENUM$_2$ (j,k) with fixed success probability p. Setting $p_t = p^{1/(k-j+1)}$, $t = j \ldots, k$, Pruned ENUM$_2$ (j,k) returns the non–trivial vector $\sum_{i=j}^{k} u_i\, b_i$ with minimal $\bar{c}_j = c_j(u_j, \ldots, u_k) = \|\pi_j(\sum_{i=j}^{k} u_i\, b_i)\|^2$. Hereafter we size–reduce the vector $z := \sum_{i=j}^{k} u_i\, b_i$ with respect to all vectors $b_{j-1}, \ldots, b_1$ such that $|< z, \hat{b}_i > |/\|\hat{b}_i\|^2 \leq 1/2$ for $i = 1, \ldots, j-1$. We test whether the pair (u,v) associated with the reduced vector satisfies condition (6) and whether $|u - v\,N|$ factorizes over $p_1, \ldots, p_t$. The algorithm stops if $t+2$ such pairs have been found.

The practical tests have been performed on a HP–Workstation 735 with 125 MHz and 57.4 MFLOPS.

Results for $N = 2131438662079$							
t	c_1	c_2	σ	β	p	run.–time	rounds
125	12	3	3.0	30	0.05	04:18.35	12
125	13	4	3.0	30	0.04	03:57.45	10
125	13	4	3.0	30	0.05	04:17.00	11
125	13	4	3.0	30	0.06	03:04.95	7
125	14	4	3.0	30	0.05	02:06.74	4
125	15	3	3.0	30	0.05	02:23.92	4
125	15	4	3.0	30	0.05	02:10.89	3
125	15	5	3.0	30	0.05	02:48.50	5
125	16	4	3.0	30	0.05	02:32.94	5
125	20	4	2.0	40	0.01	12:50.38	13
125	20	4	2.0	50	0.01	13:35.91	3
125	20	4	3.0	30	0.05	05:20.81	5

Results for $N = 250518388711599163$							
t	c_1	c_2	σ	β	p	run.–time	rounds
160	18	5	5.0	100	0.001	2:59:43.12	27
180	15	1	5.0	80	0.001	2:34:15.52	22

For the factorization of $N = 2131438662079$ Schnorr uses the parameters $t = 125,\ c_1 = 25,\ c_2 = 1,\ \beta = 32,\ \sigma = 1$ and performs a complete enumeration without any pruning [Sc93]. Instead of condition (6) Schnorr tests whether

$$|z_0| = 1 \quad \text{and} \quad \|z\|_1 < 2c_1 \log 10 + 2\sigma \log p_t - \log N \tag{10}$$

which is — without regarding rounding errors — equivalent to condition (8). In order to find a single pair (u,v) Schnorr's algorithm took a couple of hours on a SUN–Sparc 1+ Workstation which are a couple of minutes on a HP–Workstation 735.

For $t = 125,\ c_1 = 15,\ c_2 = 4,\ \beta = 30,\ \sigma = 3.0$ and $p = 0.05$ the new algorithm determines all $t + 2 = 127$ pairs (u_i, v_i) in 2 minutes where steps 3 and 4 are repeated 3 times.

For $N = 250518388711599163$ and parameters $t = 160,\ c_1 = 18,\ c_2 = 5,\ \beta = 100,\ \sigma = 5,\ p = 0.001$ the algorithm terminates after 27 rounds and takes 3 hours to find all $t + 2 = 162$ pairs (u_i, v_i).

5 Conclusion

We have proposed an efficient method to factorize composite numbers up to bitlength 60 by using strong lattice reduction algorithms. Moreover, we have provided a detailed analysis of the practical performance of our techniques. The generation of sufficiently many ℓ_1–close vectors to a fixed point remains the bottleneck of the factoring method. However, the techniques presented in this report give evidence that intricate enumeration algorithms might solve the factoring problem for bitlength up to 100 in reasonable time.

It should be pointed out that the enumeration of lattice vectors in the ℓ_2–norm seems to be most efficient. Experiments where we directly perform the enumeration in the ℓ_1–norm do not reduce the running time of our algorithm (see [R97] for more details).

Since the strategy of the pruned enumeration relies on the traversion of weighted subtrees in depth first search order we have implemented a simple version of the 'Go with the winners'–scheme as

proposed in [AV94]. The few experiments done with the parameters given in the table did not improve the running time of our enumeration. Nevertheless, this topic may be of further interest.

References

[Ad95] L.M. Adleman: Factoring and Lattice Reduction, *Draft*, University of Southern California, CA (1995).

[AV94] D. Aldous and U. Vazirani: "Go with the Winners" Algorithm, 35th Symposium on Foundations of Computer Science, Santa Fé, New Mexico (1994).

[CEP83] E.R. Canfield, P. Erdös and C. Pomerance: On a Problem of Oppenheim Concerning 'Factorisatio Numererorum', *J. Number Theory* **17** (1983), 1–28.

[LL93] A.K. Lenstra and H.W. Lenstra: The Development of the Number Field Sieve, Springer Lecture Notes of Computer Science, No. 1554 (1993).

[LLL82] A.K. Lenstra, H.W. Lenstra, Jr., L. Lovász: Factoring Polynomials with Rational Coefficients, *Math. Ann.* **261** (1982), 515–534.

[No71] K.K. Norton: Numbers with Small Prime Factors, and the Least k–th Power Non–residue, *Mem. Amer. Math. Soc.* **106** (1971).

[R97] H. Ritter: Aufzählung von kurzen Gittervektoren in allgemeiner Norm, *Ph.D. dissertation*, Universität Frankfurt am Main, 1997.

[RS62] J.B. Rosser and L. Schoenfeld: Approximate Formulas for some Functions of Prime Numbers, *Illinois Journal of Mathematics* **6** (1962), 64–94.

[Sch93] O. Schirokauer: Discrete Logarithms and Local Units, *Phil. Trans. R. Soc. Lond. A* **345** (1993), 409–423.

[Sc87] C. P. Schnorr: A Hierarchy of Polynomial Time Lattice Basis Reduction Algorithms, *Theoretical Comp. Science* **53** (1987), 201–224.

[Sc93] C. P. Schnorr: Factoring Integers and Computing Discrete Logarithms via Diophantine Approximations, *AMS DIMACS Series in Disc. Math. and Theoretical Comp. Science* **13** (1993), 171–181.

[SE94] C. P. Schnorr and M. Euchner: Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems, *Mathematical Programming* **66** (1994), 181–194.

[SH95] C.P. Schnorr and H.H. Hörner: Attacking the Chor–Rivest Cryptosystem by Improved Lattice Reduction, *Advances in Cryptology — EUROCRYPT '95, Springer LNCS* **921** (1995), 1–12.