The Number Field Sieve

Factorization of Large Numbers

Jens Putzka

B-IT - Bonn-Aachen International Center for Information Technology

Computer Security

Rheinische Friedrich Wilhelms Universität Bonn

Oberseminar CoSec, 2. November 2006
1. Factorization of Large Integers
2. The Idea of Number Field Sieves
3. The Algorithm - An Overview
4. The Different Components of the Algorithm
   - Polynomial Choosing
   - Sieve Step
   - Removing Copies, Pruning, Filtering
   - Solving Linear Equations
   - Calculating Square Roots in Number Fields
5. Further Reading
We know different methods to factorize a given integer $N$, for example:

- **Trial Division**: Try all possible factors up to $\sqrt{N}$
- **Pollard $\rho$ method, Pollard $p - 1$ method**: Need time proportional to $\sqrt{p}$ if $p$ is the smallest prime factor.
- **Elliptic Curve Method (ECM)**: Depends on smallest factor say $p$. Time about $\exp\left(\frac{1}{\sqrt{2}} \sqrt{\log p \log \log p}\right)$
- **Multiple Polynomial Quadratic Sieve (MPQS)**
- **Special Number Field Sieve (SNFS)**
- **General Number Field Sieve (GNFS)**
What is the advantage of SNFS or GNFS?

The algorithms are the fastest known for factoring large integers.
This means:

- **SNFS**: Factors special integers for example \( N = a^b + c \)
  
  Need:
  
  \[
  \exp \left( (c_s + o(1))(\log n)^{\frac{1}{3}}(\log \log n)^{\frac{2}{3}} \right), \quad c_s = \left( \frac{32}{9} \right)^{\frac{1}{3}} \approx 1.5
  \]

- **GNFS**: Factors arbitrary integers \( N \).
  
  Need:
  
  \[
  \exp \left( (c_g + o(1))(\log n)^{\frac{1}{3}}(\log \log n)^{\frac{2}{3}} \right), \quad c_g = \left( \frac{64}{9} \right)^{\frac{1}{3}} \approx 1.9
  \]
Example for the Use of SNFS

The SNFS was introduced 1988 by Pollard to factorize large integers as by Lenstra et al. the 155 digits large ninth Fermat number $F_9$

Factoring $F_9$

The ninth Fermat number $F_9$ is

$$F_9 = 2^{2^9} + 1 = 2^{512} + 1 = 1340780792994259709957402499820584612$$

$$747936582059239337772356144372176403007354697680$$

$$187429816690342769003185818648605085375388281194$$

$$6569946433649006084097$$

$$= 2^{2424833} \cdot \text{prime of 49 digits} \cdot \text{prime of 99 digits}$$
Example for the Use of GNFS

The GNFS was used to factor the 193 digits large $RSA_{640}$ in November 2005 by F. Bahr, M. Boehm, J. Franke, T. Kleinjung.

$RSA_{640} = 310741824049004372135075003588856793003734$
$602284272754572016194882320644051808150455$
$634682967172328678243791627283803341547107$
$310850191954852900733772482278352574238645$
$4014691736602477652346609$

$= 163473364580925384844313388386509085984178$
$367003309231218111085238933310010450815121$
$2118167511579 \cdot 190087128166482211312685157$
$39354139754718967899685154936666385390880$
$27103802104498957191261465571$
The idea goes back to FERMAT and LEGENDRE. We use

\[ N = x^2 - y^2 = (x - y)(x + y). \]

More general it suffices to find solutions of

\[ x^2 \equiv y^2 \mod N. \]

Then \( \gcd(x - y, N) \) gives a nontrivial factor.

One can show that for at least 50% of the pairs \((x, y)\) which satisfy \( x^2 \equiv y^2 \mod N \) and \( \gcd(xy, N) = 1 \) we have

\[ 1 < \gcd(x - y, N) < N. \]

To find solutions one can use random squares, continued fraction, ...
1 **Choose:** \( f \in \mathbb{Z}[X] \) irreducible, monic of degree \( d > 1 \). Take a primitive root \( \alpha \) of \( f \), i.e. \( f \) is the minimal polynomial of \( \alpha \). Let \( K = \mathbb{Q}(\alpha) = \mathbb{Q}[X]/f\mathbb{Q}[X] \) and \( \mathcal{O}_K \) its ring of integers.

2 **Find:** \( m \in \mathbb{Z} \) s.th.

\[
f(m) \equiv 0 \mod N
\]

This defines a ring homomorphism

\[
\phi : \mathbb{Z}[\alpha] \rightarrow \mathbb{Z}/N\mathbb{Z}
\]

by \( \phi(\alpha) = m \mod N \).
Find: $S \subseteq \{(a, b) \in \mathbb{Z} \times \mathbb{Z} : \gcd(a, b) = 1\}$, s.th.
$\forall (a, b) \in S$:
- $\prod_{(a, b) \in S} (a + bm) = x^2$ in $\mathbb{Z}$,
- $\prod_{(a, b) \in S} (a + b\alpha) = \beta^2$ in $\mathbb{Z}[\alpha]$ resp. other subring of $\mathcal{O}_K$.

Find square root: Because $\phi(a + b\alpha) \equiv a + bm \pmod{N}$
and $\phi(\beta^2) \equiv x^2 \pmod{N}$ we can find $y \in \mathbb{Z}$:

$$\phi(\beta) \equiv y \pmod{N}.$$ 

So we find $y^2 \equiv x^2 \pmod{N}$.

Calculate: $\gcd(x - y, N)$
Example (\(N = F_9\))

One can choose

\[
N = 2^{2^9} + 1
\]

\[
f(X) = X^5 + 8
\]

\[
m = 2^{103}
\]

\[
f(m) = m^5 + 8 = 2^{515} + 8 \equiv 0 \mod N
\]
Problems to solve:

There are different fields of problems:

- How to choose the polynomial $f$ and the number $m$? (Choosing Polynomial)
- How to find the set $S = \{(a, b) : \ldots\}$? (Sieve Step)
- How to find $\beta \in \mathcal{O}_k$ s.th. $\beta^2 = \prod_{(a,b)} (a + b\alpha)$? (Calculating square roots)
- How much time we need. (Time)
Problems to solve:

If $N$ has no special structure (for example $N = a^b + 1$) we know nothing about the structure of ring $\mathcal{O}_K$ of integers in $K = \mathbb{Q}(\alpha)$.

Mathematical Problems:

- In general $\mathbb{Z} [\alpha] \neq \mathcal{O}_K$.
- The ring $\mathcal{O}_K$ is in general **not a principal ideal domain** (⇔ the class number is not 1), not even a factorial ring. This means we have no unique decomposition of a given element in $\mathcal{O}_K$ into prime elements.
- But $\mathcal{O}_K$ is always a **Dedekind ring**
- The group of units $\mathcal{O}_K^\times$ can be very complicated in general.

The more we know about $\mathcal{O}_K$ the easier we can calculate in it.
The Difference between GNFS and SNFS

As the name says GNFS covers all cases of SNFS. But you can factorize arbitrary integers \( N \). To use SNFS we make most time the following assumptions:

**Assumptions for SNFS:**

- The integer we want to factorize is of a special form. This means

\[
N = a^b + c \quad \text{or} \quad N = p_1 \cdot p_2 (a^b + c)
\]

for small \( a, b, c \) and small prime factors \( p_1, p_2 \).

- \( \mathbb{Z}[\alpha] = \mathcal{O}_K \) is a principal ideal domain.

- We have a small group of units \( \mathcal{O}_K^\times \) (for example \( \pm 1 \)).
The Difference between GNFS and SNFS

I will explain most time in this talk the case of GNFS.

In Case of GNFS we have most time *(only)* to change from a decomposition into primes to a decomposition into prime ideals:

\[
\beta^2 = \prod_{p \text{ prime}} p^{e_{a,b}(p)} \quad \iff \quad (\beta^2) = \prod_{p \text{ prime}} p^{e_{a,b}(p)}
\]
The Basic Structure of GNFS

We can structure the algorithm of GNFS in the following steps:

1. Polynomial choosing
2. **Sieve Step**: Create Relations (Lattice or Line Sieves)
3. Remove copies and Pruning
4. Filter
5. **Solving Linear Equations** over $\mathbb{F}_2$
6. Calculation of square roots
Definition

We call an integer

\[ N = \prod_{p \text{ prim}} p^{\nu_p} \]

B-smooth if all \( p \) for which \( \nu_p > 0 \) are \( p \leq B \).

Theorem (Canfield, Erdős, Pomerance)

Let \( \psi(x, B) \) be the number of integers \( n < x \) which are B-smooth. For fixed \( \epsilon \in (0, 1) \) and 

\[ u = \frac{\log x}{\log B} \]

we have the following asymptotic formula

\[ \psi(x, B) = x \cdot u^{-(1+o(1))}. \]
Preparation (Choosing Polynomials): In practice as done by FRANKE et al. to factorize RSA_{640} one chooses not only one polynomial, one chooses two coprime polynomials

\[ p_1, p_2 \in \mathbb{Z}[X], \quad p_s = \sum_{i=0}^{d_i} a_i^{(s)} X^i, \]

because this makes it easier to find the factors (doubles relations we can use). Each polynomial defines a number field \( K_s = \mathbb{Q}[\alpha_s]/p_s(\alpha_s) \). The Minimal Polynomial of \( a + b\alpha_s \) is given by

\[ \frac{b^{d_s}}{a_{ds}^{(s)}} p_s \left( \frac{T - a}{b} \right). \]
Then the norm of $a + b\alpha_s$ is given by the zeroth coefficient of the minimal polynomial

$$N_{K_s/\mathbb{Q}}(a + b\alpha_s) = \frac{1}{a_d^{(s)}} \sum_{i=0}^{d_s} a_i^{(s)} (-a)^i b^{d_s-i}.$$ 

Let $\mathcal{O}_{K_s}$ be the ring of integers in $K_s$ and $\mathcal{I}_s \subseteq \mathcal{O}_{K_s}$ the smallest ideal with

$$\mathcal{I}_s \alpha_s \subseteq \mathcal{O}_{K_s}.$$ 

Then $N(\mathcal{I}_s) = \left| a_{d-s}^{(s)} \right|$. 

\[ \text{Overview GNFS} \]
Overview GNFS

Looking for Relations: Fix an upper bound $B_s$ (this can be done empirically) for the factor base, i.e. the primes into which our considered numbers should decompose. Find pairs $(a, b)$ for which $I_s(a + b\alpha_s)$ is $B_s$-smooth. This is equivalent to the fact that

$$a_d^{(s)} N_{K_{s}/\mathbb{Q}}(a + b\alpha_s) = \sum_{i=0}^{d_s} a_i^{(s)} (-a)^i b^{d_s-i}$$

is $B_s$-smooth. We can restrict us to the case that $a$ and $b$ are coprime and $b > 0$. Call the set of prime ideals which really occur in the decomposition $\mathcal{F}_s$ and let $F_s$ be the size of the factor base and $F = F_1 + F_2$. 
We can get such pairs by using **two sieve steps**, which we want to explain later. We have the following relations:

- Each pair \((a, b)\) we got by sieving gives us a relation in \(\mathcal{O}_{K_s}\) resp. (via \(\phi_s\)) in \(\mathbb{Z}/N\mathbb{Z}\).
- Each pair \((a, 0)\) with \(a > 0\) and prime (in \(\mathbb{Z}\)), which decomposes into prime factors in \(\mathcal{O}_{K_1}\) and \(\mathcal{O}_{K_2}\) which are in \(\mathcal{F}_s\). This relations are called **free relations**. Free relations can not give all pairs we need.

**Collect** at least

\[
\Lambda = F + 100 + r
\]

pairs \((a_\lambda, b_\lambda)\), where \(r\) depends only on the rank of the ideal class group and the unit group of the related number fields.
For this Λ pairs, which survived the sieve, we get the following prime ideal decomposition

\[(a_\lambda + b_\lambda \alpha_s)\mathcal{O}_{K_s} = \prod_{N(l) < B_s} l^{e_\lambda,s(l)} \cdot \mathcal{O}_{K_s}\]

**Find Exponents** $f_\lambda$, which solve the following equation system over $\mathbb{F}_2$

\[\sum_{\lambda=1}^{\Lambda} e_{\lambda,s}(l) f_\lambda \equiv 0 \pmod{2} \quad s \in \{1, 2\}, \, l \in \mathcal{F}_s.\]

We need more than $r + 2$ solutions.
Each solution of $\sum_{\lambda=1}^{\Lambda} e_{\lambda,s}(I) f_{\lambda} \equiv 0 \mod 2$ gives an element

$$q_s = \prod_{\lambda=1}^{\Lambda} (a_\lambda + b_\lambda \alpha_s)^{f_\lambda} \in K_s^\times.$$

In the prime ideal decomposition only even multiplicities occur, so we are able to take square roots later on.

The image of $q_s$ in $K_s^\times / K_s^{\times 2}$ lies in a subgroup, which is isomorphic to an $r_s + 1$ dimensional $\mathbb{F}_2$ vectorspace, where $r_s$ is the contribution of $K_s$ to $r$. 
Overview GNFS

Because we have \( r + 2 \) solutions of \( \sum_{\lambda=1}^{\Lambda} e_{\lambda,s}(l)f_{\lambda} \equiv 0 \mod 2 \) we can arrange that \( q_s \in K_s^\times \).

Then \( q_s \) is a square in \( R_s = \bigcup_{k=0}^{\infty} I_s^{-k} \mathcal{O}_{K_s} \).

**Take care:** \( R_s \) and \( \mathcal{O}_{K_s} \) are in general different. But there is an element \( p_s \) s.th. for all \( x \in R_s: p_s^k x \in \mathbb{Z}[\alpha_s] \). We can choose \( f_s = N(I_s) \text{discr}(p_s) \). This is a good choice for example in case of SNFS.

In general we can choose \( f_s \) in such a way that \( \gcd(N, f_s) = 1 \).
If \( \gcd(N, f_s) = 1 \) we can construct (without (!) knowing the prime decomposition of \( N \)) the ring homomorphism

\[
\phi_s : R_s \rightarrow \mathbb{Z}/N\mathbb{Z}
\]

by \( \phi_s(\alpha_s) = m \). Then

\[
\phi_1(a + b\alpha_1) = \phi_2(a + b\alpha_2) \Rightarrow \phi_1(q_1) = \phi_2(q_2).
\]

If \( w_i \in R_s \) is a square root from \( q_i \), then

\[
\phi_1(w_1)^2 \equiv \phi_2(w_2)^2 \mod N.
\]
If such a congruence is distributed accidently, this will give us – as in the case of only one polynomial – with a probability \( \geq \frac{1}{2} \) a non trivial divisor.

But this is at the moment not proved in an exact way in this setting. But for coprime pairs \((a, b)\) this is nearly always true.

I do not discuss the details of this problem in this talk.
As a consequence of this problem we consider only the free relations of the form \((a, 0)\) and such pairs \((a, b)\) where \(\gcd(a, b) = 1\). Therefore we consider in the factor base \(\mathcal{F}_s\) only such ideals \(I\), for which the condition \(a + b\alpha_s \in l\mathcal{I}_s^{-1}\) is given by one single congruence relation

\[ xa \equiv yb \mod p \quad \text{with} \quad \gcd(a, b, p) < p, \]

where \(p = \text{char}(\mathcal{O}_{K_s}/I)\).

Except some divisors of \(f_s\) or \(\mathcal{I}_s\) we have only to consider such prime ideals \(I\) which have residue class field \(\mathbb{F}_p \cong \mathcal{O}_{K_s}/I\).
Calculating Square Roots: Now it is possible to calculate the square roots $w_s$ of $q_s$.

Problem: The polynomials may have large (8 digits) coefficients. So some algorithms are not fast enough.
Example (SNFS)

For $N = a^b + c$ we can choose

$$p_1(X) = X^d + a^{de-b}c \quad p_2(X) = X - a^e,$$

where the degree $d$ of $p_1$ can be chosen arbitrarily and $e$ is determined by $0 \leq de - b < d$.

The common zero mod $N$ is given by $m = a^e$. 
1. Polynomial choosing

The time the algorithm needs depends on the **quality of the chosen** polynomial(s).

So we have to take the choice carefully.

**Problem:**
What are good polynomials?
1. Polynomial choosing

We have to choose \( p_1, p_2 \in \mathbb{Z}[X] \) polynomials of degree \( d_1 \) resp. \( d_2 \). We assume \( d_1 \geq d_2 \).

We associate to \( p_s \) a homogenous polynomial

\[
F_s(X; Y) := Y^{d_s} p_s \left( \frac{X}{Y} \right)
\]

of degree \( d_s \).

The runtime depends how many smooth pairs \((F_1(a, b), F_2(a, b))\) for \((a, b)\) near the origin.
1. Polynomial choosing

At the moment there are only efficient algorithms for the following cases

- \( d_1 = d_2 = 2 \) give coefficients about \( N^{\frac{1}{4}} \)
- \( d_1 = d \, d_2 = 1 \) give coefficients about \( N^{\frac{1}{d+1}} \)

In practice the case \( d_1 = d_2 = 2 \) seems to be always worse than the second case.
It may be interesting to find other possible combinations we good runtime properties.

We need a measure for the quality of polynomial pairs.
1. Polynomial choosing

We want to give a *measure* for the quality of the \((p_1, p_2)\).

**Definition (Quality of Pairs of Polynomials)**

We can define the quality of a pair as

\[
Q(p_1, p_2) = \# \{(a, b) \in \mathbb{Z} : \gcd(a, b) = 1, \\
F_s(a, b) \text{ is } B_s \text{ smooth, } |a| \leq A, 0 < b \leq B \}.
\]

This is not a good definition to use in the algorithm because it is difficult to calculate \(Q\).

\[\Rightarrow\] We have to find a *approximation of \(Q\*) which is better to calculate.
1. Polynomial choosing

Let $\rho$ be the Dickman $\rho$ function then a approximation of $Q$ is given by

$$Q(f_1, f_2) \sim Q_1(f_1, f_2)$$

$$= \frac{6}{\pi^2} \int_{|a| \leq A} \rho \left( \frac{\alpha(F_1) + \log F_1(a, b)}{\log B_1} \right) \rho \left( \frac{\alpha(F_2) + \log F_2(a, b)}{\log B_2} \right).$$

The function $\alpha(F_s)$ depends only on the number of linear factors in which the homogenous polynomial $F_s$ decomposes.

In practice we vary this function s.th. we have only to calculate 1-dimensional integrals.
1. Polynomial choosing (Algorithm)

We look for $p_1(X) = \sum_{i=0}^{d} a_i X^i$ and $p_2(X) = b_1 X + b_0$.

We make the assumption that $d \geq 4$.

### Algorithm to Produce Polynomials

**Input:** $N, d, M$ upper bound for sup norm of $p_1$, $l$ lower bound for number of prime factors of $b_1$, $P_b$ upper bound for the prime factors itself

**Output:** List of pairs of polynomials which satisfies for $a_d$, $a_{d-1}$, $a_{d-2}$, $a_1$ and $a_0$ the bound $|a_i| t^{d-i/2} < M$
1. Polynomial choosing

1. Set $\mathcal{P} = \{ r \equiv 1 \mod d : r < P_b \}$

2. For all $a_d$ in the search interval set

$$Q(a_d) = \left\{ r \in \mathcal{P} : \frac{a_d}{N} \not\equiv 0 \mod r \text{ and } \exists g : \frac{a_d}{N} \equiv g^d \mod r \right\}$$

Calculate $\tilde{d} = \sqrt[\mathcal{P}]{{\frac{N}{a_d}}}$, $a_{d-1,max} = \frac{M^2}{d}$,

$a_{d-2,max} = \left( \frac{M^{2d-6}}{d^d-4} \right)^{\frac{1}{d-2}}$

3. Set $\epsilon = \frac{a_{n-2,max}}{d_0}$. $\forall \mathcal{P}' \subset \mathcal{P}$ with $\#Q(a_d) \cap \mathcal{P}' \geq l$, s.th.

$b_1 = \prod_{r \in \mathcal{P}'} r \leq a_{d-1,max}$ look for such vectors $\mu = (\mu_i)$ for which

$$f_0 + \sum_{i=1}^{l} f_i,\mu_i \mod \mathbb{Z} \in [-\epsilon, \epsilon]$$

holds. Return the associated polynomial pair.
1. Polynomial choosing

All terms in the previous algorithm can be calculated explicitly. For example

\[ x_\mu = \sum_{i=1}^{l} x_{i,\mu_i} \]

\( d \) solutions of \( N \equiv a_d x^d \mod b_0 \)

\[ d_0 = \min \left\{ d' \geq \tilde{d} : b_0 | d' \right\} \]

\( d' \) solutions of \( N \equiv a_d x^d \mod b_0 \) near \( \tilde{d} \)

\[ d_\mu = d_0 + x_\mu \]

\[ f_0 = \frac{N - a_d d_0^d}{p^2 d_0^{d-1}} \]

\[ f_{i,j} = -\frac{a_d d x_{i,j}}{b_0^2} - \frac{e_{i,j}}{b_0} \]
1. Polynomial choosing

We have to reduce the middle coefficients of $f_1$. For this we observe, that the property to have a common root mod $N$ does not depend on:

- Change of $t$
- Replacing $p_1$ by $p_1 + \lambda p_2$ for $\lambda \in \mathbb{Z}[X]$
- Translation of $p_1$ and $p_2$ by the same integer $\kappa \in \mathbb{Z}$

By this operation, we have to minimize the $L^2$-norm

$$
\int_{\left|a\right| \leq \sqrt{t}} F_1(a, b)^2 da \, db.
$$

This can be done by the Root Sieve.
2. Sieve Step

**Given:**

\( p_s \in \mathbb{Z}[X] \) with common root. Upper bounds \( B_s \)

**Find:**

Enough pairs \((a, b) \in \mathbb{Z}^2\) with \( b > 0 \) s.th. \( F_s(a, b) \) are \( B_s \)-smooth and \( \gcd(a, b) = 1 \).
2. Sieve Step

Set as factor base

\[ \mathcal{F}_s = \{(p, r) : p < B_s, r \in \{0, \ldots, p-1, \infty\}, s.t. p_s(r) \equiv 0 \mod p\} \].

We consider \( \infty \) as root of \( p_s \) if the leading coefficient is divisible by \( p \).

Lemma

The value \( F_s(a, b) \) with \( \gcd(a, b) = 1 \) is divisible by \( p \) if and only if \( a \equiv br \mod p \) for a pair \( (p, r) \in \mathcal{F}_s \).

(In case of \( r = \infty \) the condition is \( b \equiv 0 \mod p \))
2. Sieve Step

Such pairs \((a, b)\) can be found by sieves. One can use one of the following (or combinations):

- **Line Sieve** For GNFS the lines can be very long. This can be handled by scheduling but also scheduling can be too slow for very large numbers.

- **Lattice Sieve** Disadvantage: We have to enlarge \(B_s\), so \(F_s(a, b)\) becomes larger.
2. Sieve Step

**Line Sieves:** We look for coprime pairs \((a, b)\) in the rectangle \(|a| \leq A, 0 < b \leq B\), s.th. \(F_s(a, b)\) is \(B_s\)-smooth.

**Testing Coprime Pairs**

- If we test the pairs to be coprime before starting the sieve algorithm, we have to test all possible pairs in the considered rectangle.

- If we test after the sieve step, so we have a sieve of size \((2A + 1)B\). But much less to test after the sieve step. There are two improvements by dividing the sieve into subsieves:
  
- **(Pairs mod 2)** First sieve pairs where \(a\) is even and \(b\) is odd, at next \(a\) odd and \(b\) even, and at last both are odd. \(\Rightarrow\) We have reduced the size of the sieve to \(\frac{3}{4}\).

- **(Pairs mod 6)** Similar. Gives reduction to \(\frac{2}{3}\).

The consideration of for example 5 gives only a reduction to \(\frac{24}{25}\). But enlarges the number of sieves that it cost to much.
2. Sieve Step

The Basic Idea of the Line Sieve: To explain how the Sieve Step works I go back to the case at the beginning, where we consider only one polynomial.

Find:

\[ S = \{ (a, b) \in \mathbb{Z} \times \mathbb{Z} : \gcd(a, b) = 1 \} \], s.th. \( \forall (a, b) \in S: \)

- \( \prod_{(a,b) \in S} (a + bm) = x^2 \) in \( \mathbb{Z} \).
- \( \prod_{(a,b) \in S} (a + b\alpha) = \beta^2 \) in \( \mathbb{Z}[\alpha] \) resp. \( \mathcal{O}_K \)
2. Sieve Step

We start with the set:

\[ U = \{(a, b) \in \mathbb{Z}^2 : \gcd(a, b) = 1, |a| \leq A, 0 < b \leq B\} \]

For simplicity we assume now \( A = B \) sufficiently large that \( S \neq \emptyset \). We can do this in to steps

1. Sieve pairs \((a, b)\), s.th. \((a + bm)\) is \(B\)-smooth.
2. Sieve pairs \((a, b)\), s.th. \((a + b\alpha)\) is \(B\)-smooth.

The first step is easier.
2. Sieve Step

How to find

\[ T_1 = \{(a, b) \in U : a + bm \text{ is } B - \text{smooth}\} \]

**Algorithm:** For each fixed \( b = 1, \ldots, B \) initialize the sieve array with \( a + bm \) for \(|a| \leq B\).

1. Find for all primes \( p < B \) which satisfies \( a \equiv -bm \mod p \) the highest power of \( p \) which divides the entry \( a + bm \) say \( p^l \).
2. Divide \( a + bm \) by \( p^l \), save this in the place where \( a + bm \) came from. Then take the next \( p \).
3. The pairs \((a, b)\), which have now at their places in the sieve array the entries \( \pm 1 \) are \( B \)-smooth.
2. Sieve Step

**Improvement**

Better runtime properties we get if we replace $a + bm$ by its logarithm (for example to the base 2). This changes the multiplicative problem

$$a + bm = \pm \prod_{p < B} p^{e(p)}$$

into an additive problem

$$\log(a + bm) = \sum_{p < B} e(p) \cdot \log(p).$$

We need $\# T_1 > \pi(B) + 1 = F + 1$. 
2. Sieve Step

The Line Sieve for GNFS: For both $s$, for $b = 1, \ldots, B$ do

1. For each fixed $b$ and $r = \infty$ calculate:

$$\sum_{(p, \infty) \in \mathcal{F}_s} \log p$$

(There is some weight before log, for simplicity we skip it:) This can be used to reduce the size of bound for evaluating later sieve entries.

2. Take a bound $L_1$: For each element $p$ in the factor base with $p > L_1$, calculate to which element in which subinterval the give a contribution.
2. Sieve Step

The Line Sieve for GNFS:

3. Initialize a Sieve array of length $L_1$ with 0.
   1. Start sieving with $p \in \mathcal{F}_s : p < L_1$. Add the sieve entries in this subinterval, evaluate the sieve array and save the survivors.
   2. Repeat the previous steps for $\mathcal{F}_{3-s}$, but only apply them to the survivors of the previous sieve.
   3. For the survivors of both sieves do
      1. Evaluate the polynomials
      2. Trial Division
      3. Prime Test for the rest, if necessary factorization
   4. Save the found sieve reports. Start with the next interval of length $L_1$

If all intervals are finished rise $b$ by 1 and renew the starting values of each element of the factor base. (Only one addition mod $p$).

Start again.
2. Sieve Step

The Lattice Sieve for GNFS: Take one \( q \not\in F_s \). (Else remove this for the sieve step from the factor base)

Then the pair \((a, b)\) with \( a \equiv br \mod q \) form a lattice with generators \((q, 0)\) and \((r, 1)\).

Reduce the lattice s.th. the lattice points fit at best into a rectangle with ration \( A : B \). Then take as set to sieve the points spanned the reduced matrix. This can be done similar to the Line Sieve.

**Advantage:** We reduce the length of lines in the Sieve.
**Problem:** If we change line we are sieving, we have to calculate the starting values new. In this case the costs are very high.
Now we want to look at the second condition, that \( a + b\alpha \) is \( B \)-smooth. This means that:

\[
(a + b\alpha) = u \prod_{N(p) < B} p^{\tilde{e}(p)} \quad u \in \mathcal{O}_K^\times,
\]

We can in principle do the same as before, but we have to replace the prime factors by prime ideals. So we have to look at the same congruences but for the norm, not for the ideals itself. For example we have to look at

\[
N(a + b\alpha) \equiv 0 \mod p.
\]
2. Sieve Step

As before one may replace the entries by its logarithm.

In this step there are several other improvements to the basic idea, which uses some result out of algebraic number theory which depends how much we know on $\mathcal{O}_K, \mathcal{O}_K^\times, J_K, \text{Cl}_K, \ldots$.

In this talk I want to skip the details of this step, because there are too much variations. They are explained for example in [?].
In any case we get from both sieve steps two decompositions

\[ \prod_{(a,b) \in S} (a + b\alpha) = u \prod_{N(p) < B} p^{\tilde{e}(p)} \quad u \in \mathcal{O}_K^\times, \]

\[ = u_0^{\tilde{e}(0)} \prod_{N(p) < B} p^{\tilde{e}(p)} \]

\[ \prod_{(a,b) \in S} (a + bm) = (-1)^{e(0)} \prod_{p < B} p^{e(p)}, \]

where \( u_0 \) is a fundamental unit.

Later we want to calculate the square root so all multiplicities \( \tilde{e}(j) \) and \( e(j) \) has to be even.
From the Sieve to Linear Equations

Set

\[ e(a + bm) = (e(0) \mod 2, e(2) \mod 2, \ldots, e(p) \mod 2). \]

This leads to the following system of equations

\[ \sum_{(a,b) \in S} e(a + bm) = 0 \quad \in \mathbb{F}_2^{F+1} \quad (\ast) \]

and a similar system for \( \tilde{e} \).

If \( (\ast) \) is satisfied \( \prod_{(a,b) \in S} (a + bm) \) is a square in \( \mathbb{Z} \).

The for \( \tilde{e} \) gives a square in \( \mathcal{O}_K \).
3. Remove Copies an Pruning

If we get enough pairs \((a, b)\) we could put them together into a matrix to solve a large System of linear equations over \(\mathbb{F}_2\).

But before we start to do this we have to:
If we get enough pairs \((a, b)\) we could put them together into a matrix to solve a large System of linear equations over \(\mathbb{F}_2\).

But before we start to do this we have to:

- Remove Copies
- Pruning
3. Remove Copies an Pruning

- **Remove Copies** Because of using lattice sieves in the previous step, combination of line and lattice sieves together, or simply by mistakes done by humans it may happen that different pairs \((a, b)\) occurs more than one time, which have to be removed because they will produce trivial solutions.

**Removing Copies** can be done by **Hash Tables**. We can reduce the size the hash tables by using a hash function \(h\). It may happen that we lose some pairs more than necessary. So \(h\) has to be choosen carefully.

- **Pruning**
3. Remove Copies and Pruning

- **Remove Copies**
- **Pruning** Without much more effort we can reduce the size of the matrix. We may assume that the different prime ideals corresponds to the rows of the matrix and that the different pairs corresponds to the columns. Say the matrix is called $A$, then we have to solve $Av = 0$. It is knowledge from Linear Algebra that the following does not change the space of solutions.
  - Remove 0 Rows.
  - If in the $i$-th row there is only one 1 in $(i,j)$, then we can remove the $i$-th row and the $j$-th column. Then set the $j$-th entry of $v$ to zero.
3. Remove Copies an Pruning

- **Remove Copies**
- **Pruning** This can be done by:
  1. **Calculate** how often each prime ideal occurs in the sieve reports.
  2. **Mark** each prime ideal, which occurs in exactly one relation.
  3. **Remove** the relation which was marked.
4. Filtering

To reduce the size of the matrix more we can use that column operations does not change the space of solutions.

We can look to rows with **exactly 2** (or more) entries 1.

In case of 2 entries say in row \( j \) and in column \( i \) and \( i' \) add both together and replace column \( i \) by the sum.

Now we have in row \( j \) exactly one 1 and we can proceed as in pruning.
4. Filtering

For more than two entries 1 we can proceed similar.

**Definition**

We call the number of 1s in a row the *weight* of this row.

**Problem:**

Column operations can enlarge the weight of other rows.

To reduce the number of column operations one can use graph theoretic results to find operations of minimal weight.

Because of time we should give a limit to the weight (say 32).
5. Solving Linear Equations

After reducing the system from millions of equations in the best case to several thousands by the method stated before, we have now to solve a large system of linear equations over the finite field $\mathbb{F}_2$.

In general the best algorithm for random matrices with a lot of entries which are different from 0 is the Gauß Elimination together with Fast Matrix Multiplication.
5. Solving Linear Equations

But in this case we have a sparse system of linear equations. So there are much better algorithms.

Algorithms for fast Solving of Linear Equations

- Lonczos Algorithm
- Block-Lonczos Algorithm
- Wiedemann Algorithm
- Block-Wiedemann Algorithm

This will be the subject of the talk of Daniel Loebenberg on 7th of December 2006.

Literature

The Number Field Sieve

Jens Putzka (B-IT Bonn)
5. Calculation of square roots

Calculating roots of squares in $\mathbb{Z}$ is no problem. There are different methods to calculate the square roots in number fields at the end of the algorithm.

The methods explained for example in [?] are out of date, because they are too slow.

New methods go back to MONTGOMERY [?] and can be found for example in the Diploma Thesis of F. BAHR, who worked together with FRANKE while factoring $RSA_{640}$ using the GNFS.
5. Calculation of square roots

How does it work?

Given: Sieve reports, consisting of \((a, b)\), for which 
\(q_s = \prod (a + b\alpha_s)\) are possible squares, together with their prime ideal decomposition.

Find: Square root, resp. their image under \(\phi_s\).

The algorithm divides step by step squares from \(q_s\). After the \(t\)-th round we have

\[ q_s = \gamma_t^2 \delta_t \]

\((\gamma_0 = 1, \delta_0 = q_s)\). We consider the principal ideal \((\delta_t)\) together with one ideal decomposition of it.

One uses bases for the Ideal \((\delta_t)\) and uses LLL algorithm to find short lattice vectors, which span the ideals.

For further details I refer to the literature.
For Further Reading

F. Bahr
Liniensieben und Quadratwurzelberechnung für das Zahlkörpersieb
Mathematisches Institut der Universität Bonn, Diplomarbeit 2005

H. Cohen
A Course in Computational Algebraic Number Theory
Graduate Texts in Mathematics 138, Springer 1996

A.K. Lenstra, H.W. Lenstra (Eds.)
The Development of the Number Field Sieve
Lecture Notes in Mathematics 1554, Springer 1993