Algorithms for factoring integers

Johannes Buchmann

Volker Müller

Universität des Saarlandes
Postfach 15 11 50
66041 Saarbrücken
Germany

Contents

Chapter 1

Introduction

As an example for the application of number theory, we describe the RSA public key crypto system. This example will then serve as a motivation for the discussion of factorization, primality-testing and discrete logarithm algorithms.

1.1 Basic concepts

By $\mathbb{Z}$ we denote the set of rational integers. If $a,\ b \in \mathbb{Z}$, then there is precisely one pair $q,\ r \in \mathbb{Z}$ such that

$$a \;=\; q \cdot b + r \quad \text{with} \;\; 0 \le r < |b|.$$

We say that the division of a by b leaves **remainder** r. If $r = 0$, then b is called **divisor** of a. We write $b \mid a$. A divisor b of a is called **proper divisor** of a, if $b \neq a$ and $b \neq 1$. A number $p \in \mathbb{Z}_{>1}$, which has no proper divisors, is called **prime number**. There are infinitely many prime numbers. The numbers ± 1 are the **units** of $\mathbb{Z}$. Each $a \in \mathbb{Z}$ can be decomposed into a power product of units and prime numbers p_i:

$$a \;=\; (-1)^{l_0} \cdot \prod_{i=1}^{k} p_i^{l_i}$$

with $l_0 \in \{0,\ 1\}$ and $l_i \in \mathbb{Z}_{\geq 0}$. This decomposition is uniquely determined up to the ordering of the prime numbers p_i.

Let $a \in \mathbb{Z}$ and $n \in \mathbb{Z}_{\geq 1}$. Then

$$a + n \cdot \mathbb{Z} \;=\; \{a + n \cdot k \mid k \in \mathbb{Z}\}$$

is called the **residue class** of a modulo n. There are n residue classes modulo n. If n is clear, then we also write $\overline{a} = a + n \cdot \mathbb{Z}$. On the residue classes modulo n we can introduce an addition and a multiplication:

$$\overline{a} + \overline{b} \;=\; \overline{a+b} \qquad \text{and} \qquad \overline{a} \cdot \overline{b} \;=\; \overline{a \cdot b}.$$

With those operations, the residue classes modulo n form a commutative ring with unit element $\overline{1}$, the **residue class ring** modulo n, denoted by $\mathbb{Z}/n\mathbb{Z}$. If $\overline{a} = \overline{b}$, we also write

$$a \equiv b \bmod n.$$

An element $b \in \overline{a}$ is called **representative** of the residue class $\overline{a}$. We can uniquely choose a representative of $\overline{a}$ in the set $\{0, \ldots, n-1\}$. It is called the **least (non negative) residue** of a mod n.

1.2 The RSA crypto system

The encryption in the RSA crypto system ([RSA78]) works as follows: Suppose that Bob wants to receive secret messages from Alice. He chooses a modulus n, say $n = 33$, and an encryption exponent e, say $e = 3$. Both those pieces of information are sent to Alice. Alice wishes to send him the secret message "I love you". Each letter can be encoded by a number $m \in \{1, \ldots, 26\}$. The separating blank is $m = 27$. Thus Alice encrypts her message

$$9\ 27\ 12\ 15\ 22\ 5\ 27\ 25\ 15\ 21$$

by computing for each encoding number m the least residue of m^e modulo n. Then the encrypted message is

$$3\ 15\ 12\ 9\ 22\ 26\ 15\ 16\ 9\ 21.$$

In order to recover the original information, Bob only needs to calculate for each encrypted number m the least residue of m^d modulo n with decryption exponent $d = 7$. Of course, Bob keeps the decryption exponent d secret and hopes that nobody can find it out.

We have to ask the following question: How does one choose the modulus n and the exponents e and d such that the system is secure? To answer this question, we must study the residue class ring $\mathbb{Z}/n\mathbb{Z}$ a little bit more.

For $a, b \in \mathbb{Z}$ the largest $d \in \mathbb{Z}_{\geq 1}$, which divides both a and b, is called the **greatest common divisor** of a and b. We write

$$d = \gcd(a, b).$$

Each common divisor d' of a and b divides $\gcd(a, b)$. If $\gcd(a, b) = 1$, we call a and b **coprime**. The gcd of a and b has a representation

$$\gcd(a, b) = x \cdot a + y \cdot b \tag{1.1}$$

with integers $x, y \in \mathbb{Z}$.

If $\gcd(a, n) = 1$, then $\overline{a}$ is called **primitive residue class** modulo n. Since by (1.1) there is a representation

$$1 = x \cdot a + y \cdot n,$$

we see that $\overline{1} = \overline{x} \cdot \overline{a}$ (or, in other words, $x \cdot a \equiv 1 \bmod n$). Therefore the primitive residue classes are **invertible** in $\mathbb{Z}/n\mathbb{Z}$. This means that the primitive residue classes modulo n form a finite abelian group, the **primitive residue class group** $(\mathbb{Z}/n\mathbb{Z})^*$.

1.1. Example *Let $n = 10$. Then there are 10 residue classes modulo 10. Among those there are 4 primitive ones, namely $\overline{1}$, $\overline{3}$, $\overline{7}$, $\overline{9}$. We have claimed that they are all invertible in $\mathbb{Z}/10\mathbb{Z}$. In fact, we have $3 \cdot 7 = 21 \equiv 1$ mod 10 and $9 \cdot 9 = 81 \equiv 1$ mod 10.*

The number of primitive residues modulo n, i.e. the order of $(\mathbb{Z}/n\mathbb{Z})^*$ is denoted by $\varphi(n)$. The function

$$\begin{array}{rccc} \varphi\colon & \mathbb{Z}_{\geq 1} & \longrightarrow & \mathbb{Z}_{\geq 1} \\ & n & \longmapsto & |\,(\mathbb{Z}/n\mathbb{Z})^*\,| \end{array}$$

is called the **Euler φ-function**. As an application of Lagrange's theorem, we have for every primitive class $\overline{a}$ modulo n

$$\overline{a}^{\varphi(n)} = \overline{1} \qquad \text{or, equivalently} \qquad a^{\varphi(n)} \equiv 1 \mod n. \tag{1.2}$$

The congruence (1.2) show us how to choose the encryption and decryption exponents e and d. e is chosen such that $\gcd(e,\, \varphi(n)) = 1$. Then by (1.1) we can choose d, such that

$$1 = e \cdot d + y \cdot \varphi(n) \tag{1.3}$$

for some $y \in \mathbb{Z}$. If m is the encoded letter (or word), then $\overline{m^e}$ is the residue class of the encrypted letter. We have

$$(m^e)^d = m^{e \cdot d} = m^{1 - y\cdot\varphi(n)} = m \cdot m^{-\varphi(n)\cdot y} = m \cdot (m^{\varphi(n)})^{-y} \equiv m \bmod n.$$

This argument only covers the case where $\gcd(m,\, n) = 1$. In general

$$m^{e\cdot d} \equiv m \bmod n \tag{1.4}$$

is not even true. But if $n = p \cdot q$ is the product of the distinct prime numbers p and q, then (1.4) remains valid.

In order to apply this encryption scheme in practice, Bob must be able to determine $\varphi(n)$ and then solve the equation (1.3). If

$$n = \prod_{i=1}^{k} p_i^{l_i} \tag{1.5}$$

is the decomposition of n into powers of distinct primes, then

$$\varphi(n) = n \cdot \prod_{i=1}^{k} \left(1 - \frac{1}{p_i}\right).$$

Hence $\varphi(n)$ can be computed if the factorization (1.5) of n is known. No other efficient way of computing $\varphi(n)$ is known so far. The efficiency of computing $\varphi(n)$

therefore depends on the efficiency of the factoring algorithm which Bob uses. The equation (1.3), or more generally, the equation (1.1) can be solved by means of the Euclidean algorithm, which we describe now.

If we want to determine the greatest common divisor of a_1, $a_2 \in \mathbb{Z}$, we have to successively carry out the divisions with remainder

$$a_i \;=\; q_i \cdot a_{i+1} + a_{i+2} \quad \text{with} \;\; 0 \le a_{i+2} < |a_{i+1}|. \tag{1.6}$$

Because the sequence (a_i) is strictly decreasing, we find that after a finite number of iterations a remainder a_i is zero. The last non-zero remainder is the gcd of a_1 and a_2. The recursion (1.6) can be used to find the representation of the gcd. For an extensive discussion of Euclid's algorithm we refer to [Kn74, pp. 316-364].

1.2. Example *Let* $a_1 = 605$ *and* $a_2 = 430$. *Then we get*

$$\begin{aligned}
605 &= 1 \cdot 430 + 175 \\
430 &= 2 \cdot 175 + 80 \\
175 &= 2 \cdot 80 + 15 \\
80 &= 5 \cdot 15 + 5 \\
15 &= 3 \cdot 5 + 0
\end{aligned}$$

and so $\gcd(605, 430) = 5$. *The represention of the* gcd *is computed by the sequence*

$$\begin{aligned}
a_1 &= 1 \cdot a_2 + a_3 \\
a_2 &= 2 \cdot a_3 + a_4 \\
a_3 &= 2 \cdot a_4 + a_5 \\
a_4 &= 5 \cdot a_5 + a_6 \\
a_6 &= a_4 - 5 \cdot a_5 = a_4 - 5(a_3 - 2a_4) \\
&= 11 \cdot a_4 - 5 \cdot a_3 = 11 \cdot (a_2 - 2 \cdot a_3) - 5 \cdot a_3 \\
&= 11 \cdot a_2 - 22 \cdot a_3 - 5 \cdot a_3 = 11 \cdot a_2 - 27 \cdot a_3 \\
&= 11 \cdot a_2 - 27 \cdot (a_1 - a_2) = 38 \cdot a_2 - 27 \cdot a_1.
\end{aligned}$$

In order to use the RSA-scheme, Bob will use a modulus $n = p \cdot q$ with two large prime numbers (each 100 digits). He will pick a large encryption exponent e. Since $\varphi(n) = (p-1) \cdot (q-1)$, he can easily determine the decryption exponent d.

Alice only needs to encode her message using numbers $m \in \{0, \ldots, n-1\}$. Then she has to determine the least positive residue in $\overline{m}^e$. Both for the encryption and decryption one must use fast exponentiation techniques in $\mathbb{Z}/n\mathbb{Z}$, which will be described later .

In order to break the scheme, one has to recover $\overline{m}$ from $\overline{m}^e$ and e. For real numbers x we can compute x from x^e and e by

$$x = \exp((\log x^e)/e).$$

Therefore this problem is known as the problem of computing **discrete logarithms** in $\mathbb{Z}/n\mathbb{Z}$. As we have seen, discrete logarithms can be computed if the **factorization** of the modulus n is known. Hence three major problems of computational number theory arise in the context of the RSA-scheme:

- Primality testing: To find an appropriate modulus n.
- Factoring integers: To break the RSA-scheme.
- Computing discrete logarithms in $\mathbb{Z}/n\mathbb{Z}$.

The factorization problem will be the subject of this report.

Chapter 2

Historical methods for factorization

2.1 The Fermat test

Let n be a large odd integer, which you want to feed into a complicated factoring algorithm. It is first of all advisable to check whether n really is composite or not. This is done as follows: If n is a prime number, then we know that $\varphi(n) = n - 1$ and from (1.2) we can conclude that for every $a \in \mathbb{Z}$ with $\gcd(a, n) = 1$ we have

$$a^{n-1} \equiv 1 \bmod n. \tag{2.1}$$

Thus we have the following criterion:

2.1. Proposition *Let $a \in \mathbb{Z}$. If $\gcd(a, n) = 1$, but $a^{n-1} \not\equiv 1 \bmod n$, then n is composite.*

Proposition 2.1 provides a sufficient criterion for testing a number n for compositeness. There are however composite numbers n, which satisfy (2.1) for a certain base a (so called **pseudoprimes** for the base a) or even for all bases a with $\gcd(a, n) = 1$ (**Carmichael numbers**). We note that Carmichael numbers are rather scarce (2163 below $25 \cdot 10^9$) and that below $25 \cdot 10^9$ there are 21853 pseudoprimes for the base 2. The chance that a test constructed on the basis of Proposition 2.1 fails is very small. In the lecture on primality testing we will discuss more sophisticated compositeness tests.

In order to turn Proposition 2.1 into a practical test, the so-called **Fermat test**, we have to carry out two steps:

1. Choose a base a with $1 < a < n$. Compute $\gcd(a, n) = d$. If $d > 1$, then d is a proper divisor of n and thus n is composite.

2. Compute $a^{n-1} \bmod n$. If $a^{n-1} \not\equiv 1 \bmod n$, then n is composite, otherwise go to step 1.

If we have checked (2.1) for "a few" bases, then we suspect that n is prime and we apply a primality test. Otherwise we attempt to factor n.

Step 2 of this procedure can be executed very efficiently if we use a fast exponentiation technique. If we want to determine $a^d \bmod n$, we compute the binary decomposition of the exponent

$$d = \sum_{i=0}^{s} \beta_i \cdot 2^i, \ \beta_i \in \{0, 1\}.$$

Then we have

$$a^d = a^{\sum_{i=0}^{s} \beta_i \cdot 2^i} = \prod_{i=0}^{s} a^{\beta_i \cdot 2^i} = \prod_{\beta_i=1} a^{2^i}. \tag{2.2}$$

Note that the number of factors in this representation of a^d is $s+1 = [\log_2 d] + 1$. The decomposition of d can be done while computing the powers. We have

2.1. Algorithm (Fast exponentiation)

Input: $\overline{a}$, d.
Output: $\overline{b} = \overline{a^d}$.

1. *Initialize* $d' = d$, $\overline{b} = \overline{1}$, $\overline{c} = \overline{a}$.
2. *If* d' *is odd, then set* $\overline{b} = \overline{b} \cdot \overline{c}$ *and* $d' = d' - 1$.
3. *If* $d' = 0$, *terminate with result* $\overline{b}$.
4. *Set* $d' = d'/2$, $\overline{c} = \overline{c} \cdot \overline{c}$ *and go to 2.*

The correctness of this algorithm is easily seen from (2.2). Concerning its running time we have the following proposition:

2.2. Proposition *a^d mod n can be computed in $O(\log d)$ elementary operations on numbers of size $O(\log n)$.*

2.1. Exercise *Using the Fermat Test show that the numbers $n = 2^{20} + 1$ and $n = 10^6 1$ are composite.*

2.2 Trial division

The basis of the trial division method for factoring n is the following trivial observation.

2.3. Proposition *n is prime if it has no prime divisor $p \leq \sqrt{n}$.*

We can therefore either precompute a table of all prime numbers $p \leq C$ and use this table to factor numbers $n \leq C^2$. Since the number of primes $p \leq C$ is roughly $C/\log C$, this is quite an expensive method. It might therefore be more efficient to trial divide n by all numbers $l = 6 \cdot k \pm 1$ for $k \in \mathbb{Z}_{\geq 1}$ and $l \leq \sqrt{n}$. This method avoids an expensive precomputation, but nevertheless we test all prime numbers.

2.4. Proposition *Trial division requires at most $O(n^{1/2})$ elementary operations on numbers of size $O(\log n)$.*

2.3 Fermat's factoring method

Reference: [Fe94]

Suppose that $n > 1$ is odd, non square and composite. Fermat's method tries to represent n in the form

$$n = x^2 - y^2 \tag{2.3}$$

with $x, y \in \mathbb{Z}_{\geq 1}$. Then $n = x^2 - y^2 = a \cdot b$ with $a = x - y$ and $b = x + y$ is a factorization of n. This method always works, since for

$$n = a \cdot b \quad \text{with} \quad a > b, \tag{2.4}$$

we can choose

$$x = \frac{a+b}{2} \quad \text{and} \quad y = \frac{a-b}{2}. \tag{2.5}$$

It follows from (2.3) that $x > \sqrt{n}$. Since by (2.5) $x = (b + n/b)/2$ and because $b < \sqrt{n}$ we know that x is maximal for $b = 1$ and thus we get $x < (n+1)/2$. So we have proved

2.5. Proposition *Each factorization (2.4) of n corresponds uniquely to a representation (2.3) of n with $\sqrt{n} < x < (n+1)/2$.*

How do we practically exhibit the representation of n as a difference of two squares? The first candidate for x is $x_0 = \lfloor \sqrt{n} \rfloor + 1$. We must now find out whether $y_0 = x_0^2 - n$ is a square. If this is not the case, we try recursively $x_{i+1} = x_i + 1$. Then we have to check whether

$$y_{i+1} = x_{i+1}^2 - n = (x_i + 1)^2 - n = x_i^2 - n + 2 \cdot x_i + 1 = y_i + 2 \cdot x_i + 1 \tag{2.6}$$

is a square or not. The sequence $(2 \cdot x_i + 1)$ is just the sequence of odd integers starting with $2\lfloor \sqrt{n} \rfloor + 1$. By (2.6) the sequence y_i is easily calculated.

2.1. Example *Reference: [Br81]*

Let $n = 2027651281$. So $\lfloor \sqrt{n} \rfloor = 45029$. We successively compute $y_i = x_i^2 - n$ for $x_i = 45029 + i$ with $i = 1, 2, 3, \ldots$:

i	x_i	y_i	$2x_i + 1$
0	45030	49619	90061
1	45031	139680	90063
2	45032	229743	90065
3	45033	319808	90067
4	45034	409875	90069
5	45035	499944	90071
6	45036	590015	90073
7	45037	680088	90075
8	45038	770163	90077
9	45039	860210	90079
10	45040	950319	90081
11	45041	1040400	

Since $1040400 = 1020^2$, *we have found the representation*

$$2027651281 \;=\; 45041^2 - 1020^2 \;=\; (45041-1020)\cdot(45041+1020) \;=\; 44021\cdot 46061,$$

which is the complete factorization.

The most laborious part in Fermat's factoring algorithm is the extraction of square roots. A lot of this work can be avoided, if we note that 00, 01, 04, 09, 16, 21, 24, 25, 29, 36, 41, 44, 49, 56, 61, 64, 69, 76, 81, 84, 89 and 96 are the only possible final two digits of squares. In Example 2.1 only the numbers 499944 and 1040400 have square endings. However the first number is divisable by 3 but not by 9. Analogously, if we want to use this trick with the binary representation of n rather than with the decimal representation, we have to observe that squares are congruent to 0, 1, 4 or 9 modulo 16.

The next exercises shows that further refinements of this method are possible.

2.2. Exercise *Let p be an odd prime number, $n \mid 2^p - 1$. Show that every x in (2.3) is of the form $x = (n+1)/2 + k\cdot p^2$, $k \in \mathbb{Z}$. Using this feature, modify Fermat's factoring method.*

2.3. Exercise *Assume that n has no prime factor $p \le B$. Show that for every factorization of n there is a representation (2.3) with $\sqrt{n} < x < (B + n/B)/2$. Use this feature to modify Fermat's factoring method.*

2.4. Exercise *Factor 200819 and 141467 using Fermat's method. (Remark: Replace by better numbers).*

Now we show that in the worst possible case Fermat's method is pretty inefficient. Assume $n = a\cdot b$ is a factorization. This will be found with $x = (a+b)/2$ and $y = (a-b)/2$. So $x = (b + n/b)/2$. We analyse the function $f(b) = (b + n/b)/2$. We find $f'(b) = (1 - n/b^2)/2$. For $b < \sqrt{n}$ we have $f'(b) < 0$. So $f(b)$ is minimum for b

closest to $\sqrt{n}$. Therefore the first factor of n found by Fermat's method is the one, which is closest to $\sqrt{n}$. The at most number of iterations in Fermat' s method is $x - \sqrt{n}$. Write $b = \sqrt{n} - b'$, then we find that the number of iterations is at most

$$\begin{aligned}\frac{\left(\sqrt{n} - b' + \frac{n}{\sqrt{n}-b'}\right)}{2} - \sqrt{n} &= \frac{-\sqrt{n} - b' + \frac{n}{\sqrt{n}-b'}}{2} \\ &= \frac{b'^2 - n + n}{2b} \\ &= \frac{b'^2}{2b}.\end{aligned}$$

In the worst case where $b = 3$ this is asymptotically equal to $n/6$. Even for $b \approx n^{1/3}$ this is still of order of magnitude $n^{2/3}$, which is pretty bad. The algorithm is best suited for numbers n with two factors of almost equal size.

2.6. Proposition *Using a refinement of Fermat's factoring algorithm, n can be factored in $O(n^{1/3})$ elementary operations on numbers of size $O(\log n)$.*

Proof: [Le74] ■

2.4 Gauß's method of exclusion

We consider the equation

$$f \cdot n = x^2 - y^2$$

or

$$y^2 = x^2 - f \cdot n. \tag{2.7}$$

Let $m \in \mathbb{Z}_{\geq 1}$, then it follows from (2.7) that

$$y^2 \equiv x^2 - f \cdot n \mod m. \tag{2.8}$$

This congruence can be used to exclude many possible values for x. We illustrate this in

2.2. Example *(Brillhart)*
Let $n = 2^{41} - 1 = 219902325551$, $f = 12383$. By Exercise 2.2, x is of the form

$$x = 1681 \cdot k + 1099511627776$$

with $k \geq 0$. Therefore

$$y^2 = 2825761 \cdot k^2 + 554788690434 \cdot k + 256141258816 = g(k). \tag{2.9}$$

Choose $m_1 = 32$. The idea is to compute the right hand side of (2.9) and to check, which of the results is a square modulo 32. So we first determine which of the residue *classes modulo 32 is a square by squaring the numbers* 0, 1, ..., 16 *and reducing the result modulo 32.*

k *mod* 32	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
k^2 *mod* 32	0	1	4	9	16	25	4	17	0	17	4	25	16	9	4	1	0.

The only squares in $\mathbb{Z}/32\mathbb{Z}$ *are* $\overline{0}$, $\overline{1}$, $\overline{4}$, $\overline{6}$, $\overline{9}$, $\overline{16}$, $\overline{17}$, $\overline{25}$.

We now determine the value of $g(k)$ *mod 32 for* $0 \leq k \leq 31$. *This yields the following table:*

k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$g(k)$ *mod* 32	0	3	8	15	24	3	16	31	16	3	24	15	8	3	0	31
k	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$g(k)$ *mod* 32	0	3	8	15	24	3	16	31	16	3	24	15	8	3	0	31

From this table we see that only for $k \equiv 0, 6, 8, 14, 16, 22, 24, 30$ *mod* 32 *the value of* $g(k)$ *can be a square. A similar computation with* $m_2 = 27$, $m_3 = 25$ *shows that only for*

$$k \equiv 0, 3, 6, 9, 12, 15, 12, 18, 21, 24 \mod 27$$

and

$$k \equiv 0, 1, 5, 6, 10, 11, 15, 16, 20, 21 \mod 25$$

$g(k)$ *can be a square.*

Finding values for k, which satisfy all the three congruence conditions simultaneously is called a sieving problem.

2.5. Exercise *Using the information obtained so far to factor* $2^{41} - 1$.

In general, one chooses so-called exclusion moduli $m_1, \ldots, m_k$. One determines k congruence conditions for x. Then one determines x by sieving.

In 1927 D.H. Lehmer built a sieving device with bicycle chains that could check 50 values of x per second. A significant factorization obtained on this machine was of the cofactor $n = 20191210335106439 = 2579599 \cdot 78539161$ of $3^{111} + 1$.

Later in 1933, this machine was redesigned by replacing the bicycle chains by carefully designed gears and a photo-electrical system.

In the early 1960's Brillhart programmed this method on the IBM 7090 and achieved a scanning rate of about one hundred thousand values of x per second.

The latest development in this direction is the University of Manitoba Sieving Unit (UMSU).

We conclude the description of this method by mentioning that sieving plays an important role in many other areas of algorithmic number theory.

Chapter 3

Modern methods for factorization

3.1 $x^2 \equiv y^2 \bmod n$

3.1.1 The algorithm

Suppose that for $x,\, y \in \mathbb{Z}$ we know that

$$x^2 \equiv y^2 \bmod n, \tag{3.1}$$

but

$$x \not\equiv \pm y \bmod n. \tag{3.2}$$

Then we have

$$n \mid (x-y)\cdot(x+y),$$

but

$$n \not\mid (x-y) \text{ and } n \not\mid (x+y).$$

Hence $d = \gcd(n,\, x-y)$ is a proper divisor of n. Finding $x,\, y \in \mathbb{Z}$ which satisfy (3.1) and (3.2) therefore means factoring n.

3.1. Proposition *If $n \in \mathbb{Z}_{\geq 1}$ is odd, no prime power and composite, then there are always $x,\, y \in \mathbb{Z}$ satisfying (3.1) and (3.2).*

Proof: Let $n = n_1 \cdot n_2$ with $n_1,\, n_2 \in \mathbb{Z}_{>1}$ coprime. Pick $y \in \mathbb{Z}$ with $\gcd(y,\, n) = 1$. Choose $x \in \mathbb{Z}$ with $x \equiv y \bmod n_1$ and $x \equiv -y \bmod n_2$. Then $x^2 \equiv y^2 \bmod \operatorname{lcm}(n_1,\, n_2) = n$, but $x \not\equiv \pm y \bmod n$. For if $x \equiv y \bmod n$ then $-y \equiv y \bmod n_2$ and so $2 \cdot y \equiv 0 \bmod n_2$. Since $\gcd(y,\, n_2) = 1$, this means that $n_2 = 2$, but n was chosen odd. □

Remark: The proof of Proposition 3.1 shows that the number of pairs $x,\, y$ satisfying (3.1) and (3.2) is at least $\varphi(n)$.

The factorization of a number n can be achieved as shown in the following

3.1. Example *Let $n = 4633$. Assume that we know the following congruences:*

$$\begin{aligned} 67^2 &\equiv -2^4 \cdot 3^2 \mod n, \\ 68^2 &\equiv -3^2 \mod n. \end{aligned}$$

Multiplying those congruences we find that

$$(67 \cdot 68)^2 \equiv (2^2 \cdot 3^2)^2 \mod n,$$

i.e. $77^2 \equiv 36^2 \mod n$. *Since* $77 \not\equiv \pm 36 \mod n$ *we find the factor* $41 = \gcd(77 - 36, 4633)$.

This idea can be formalized as follows: Choose a factor base

$$\mathcal{F} = \{p_0, p_1, \ldots, p_l\}$$

where the p_i are pairwise distinct prime numbers, except that $p_0 = -1$ is allowed. In Example 3.1 we have chosen $\mathcal{F} = \{-1, 2, 3\}$. Determine congruences

$$x_i^2 \equiv z_i \mod n, \tag{3.3}$$

such that the z_i factor completely over $\mathcal{F}$ as

$$z_i = \prod_{j=0}^{l} p_j^{e_{ij}} \tag{3.4}$$

and such that a subproduct of the z_i is a perfect square:

$$z = \prod_{i=1}^{k} z_i^{f_i} = y^2 \quad \text{for } y \in \mathbb{Z},\ f_i \in \{0, 1\}. \tag{3.5}$$

Put $x = \prod_{i=1}^{k} x_i^{f_i}$. If we are lucky, then $x \not\equiv \pm y \mod n$ and we have found a factorization. In order to practically use this method, we have to discuss how to produce solutions of (3.3), (3.4) and how to detect whether (3.5) holds. Let us start with the latter question. First observe that z is a perfect square if and only if all the exponents in the prime decomposition of z are even.

Since we know the prime decomposition of the factors z_i, we can easily determine the prime decomposition of z:

$$z = \prod_{i=1}^{k} z_i^{f_i} = \prod_{i=1}^{k} \prod_{j=0}^{l} p_j^{f_i \cdot e_{ij}} = \prod_{j=0}^{l} p_j^{\sum_{i=1}^{k} f_i \cdot e_{ij}}.$$

z is a perfect square if

$$\sum_{i=1}^{k} f_i \cdot e_{ij} \equiv 0 \mod 2$$

for $0 \leq j \leq l$. We compute with the exponent vectors $\underline{e}_i = (e_{i0}, \ldots, e_{il}) \mod 2$. As soon as we have found more than $l + 1$ such vectors, there must be a linear dependency

$$\sum_{i=1}^{k} f_i \cdot \underline{e}_i \equiv \underline{0} \mod 2$$

and thus, the f_i such that z is a perfect square can be determined by solving a linear system over $\mathbb{F}_2$, the field of two elements.

How to obtain solutions of (3.3) and (3.4)? The first possibility is to randomly generate a solution of (3.3) and to check whether a factorization (3.4) over the factor basis $\mathcal{F}$ can be determined. This means that one chooses randomly $x \in \{0, \ldots, n-1\}$ and computes z by reducing $x^2 \bmod n$. This is called **Dixon's random square method**. Unfortunately, this strategy is not practical because the numbers z that have to be factored over $\mathcal{F}$ are quite large (order of magnitude n).

3.1.2 The continued fraction method (RCFA)

An efficient way of choosing the x_i is given by the **continued fraction method** ([MoBr75]). We briefly introduce some basic facts about continued fractions.

Instead of representing a real number x by its decimal expansion it can also be written as an (infinite) regular continued fraction.

$$x = b_0 + \cfrac{1}{b_1 + \cfrac{1}{b_2 + \cfrac{1}{b_3 + \cfrac{1}{\vdots}}}} \tag{3.6}$$

with $b_0 \in \mathbb{Z}$, $b_i \in \mathbb{Z}_{\geq 1}$ for $i \geq 1$. We also write (3.6) as

$$x = [b_0, b_1, b_2, \ldots]. \tag{3.7}$$

b_0 and the **partial denominators** b_i are calculated via the following recursion

$$x_0 = x,\ b_n = \lfloor x_n \rfloor,\ x_{n+1} = \frac{1}{x_n - b_n}. \tag{3.8}$$

3.2. Example

$$x_0 = x = \sqrt{2},\ b_0 = 1,$$

$$x_1 = \frac{1}{\sqrt{2}-1} = \sqrt{2}+1,\ b_1 = 2,$$

so in general

$$x_n = \sqrt{2}+1,\ b_n = 2$$

and thus

$$\sqrt{2} = [1, 2, 2, \ldots].$$

Note that the continued fraction expansion of $\sqrt{2}$ is periodic with period length 1. In general, the continued fraction expansion of x is periodic if and only if x

is a quadratic irrationality, i.e. x is a root of a quadratic polynomial with integer coefficients, a fact, which we do not prove here.

The n-th **convergent** or the n-th **partial quotient** is the finite part

$$\frac{A_n}{B_n} = [b_0,\, b_1,\, \cdots,\, b_n].$$

3.2. Proposition *If we define*

$$\begin{aligned} A_{-1} &= 1,\ B_{-1} = 0 \\ A_0 &= b_0,\ B_0 = 1, \end{aligned}$$

then we have

$$\begin{aligned} A_s &= b_s \cdot A_{s-1} + A_{s-2} \\ B_s &= b_s \cdot B_{s-1} + B_{s-2} \end{aligned}$$

for every $s \in \mathbb{Z}_{>0}$.

Proof: Exercise. □

We also mention without further discussion that the sequence of convergents of x is in fact convergent with limit x.

For the purpose of factoring n, the regular continued fraction expansion of $\sqrt{n}$ is employed.

3.3. Proposition *For* $n \geq 0$ *we have*

$$A_{n-1} \cdot B_n - A_n \cdot B_{n-1} = (-1)^n.$$

Proof: Exercise. □

3.4. Proposition *For* $i \geq 1$ *we have*

$$x = \frac{A_{i-2} + A_{i-1}x_i}{B_{i-2} + B_{i-1}x_i}.$$

Proof: We have

$$\frac{A_i}{B_i} = [b_0,\, b_1,\, \ldots,\, b_{i-1},\, b_i].$$

Let

$$R(t) = [b_0,\, b_1,\, \ldots,\, b_{i-1},\, t]$$

for $t > 0$. Then we can write

$$R(t) = \frac{a + bt}{c + dt}.$$

If we consider the recursion formulas (3.8) we directly see that

$$\lim_{t\to\infty} R(t) = \frac{A_{i-1}}{B_{i-1}}.$$

In order to evaluate $\lim\limits_{t\to 0} R(t)$ we note that

$$\lim_{t\to 0} \frac{1}{b_{i-1}+\frac{1}{t}} = 0,$$

so we can use the same argument to achieve

$$\lim_{t\to 0} R(t) = \frac{A_{i-2}}{B_{i-2}}.$$

Hence

$$R(t) = \frac{A_{i-2}+k\cdot A_{i-1}t}{B_{i-2}+k\cdot B_{i-1}t}$$

for some $k \neq 0$. But since

$$R(b_i) = \frac{A_i}{B_i} = \frac{A_{i-2}+k\cdot A_{i-1}\cdot b_i}{B_{i-2}+k\cdot B_{i-1}\cdot b_i},$$

it follows from Proposition 3.2 that $k = 1$. Finally, taking $t = x_i$ proves the assertion. □

3.5. Proposition *If $x = \sqrt{n}$, then we can write*

$$x_i = \frac{\sqrt{n}+P_i}{Q_i}$$

with P_i, $Q_i \in \mathbb{Z}$ and

$$\begin{aligned} &P_0 = 0,\ Q_0 = 1,\\ &P_{i+1} = b_i\cdot Q_i - P_i,\\ &Q_{i+1} = (n-P_{i+1}^2)/Q_i. \end{aligned}$$

Moreover, we have the estimates

$$|P_i| < \sqrt{n},\ 0 < Q_i < 2\sqrt{n}.$$

Proof: The proof employs mathematical induction.

$i = 0:$

$$x_0 = \sqrt{n} = (\sqrt{n}+P_0)/Q_0$$
$$x_1 = \tfrac{1}{\sqrt{n}-b_0} = \tfrac{\sqrt{n}+b_0}{n-b_0^2},$$

so

$$\begin{aligned} P_1 &= b_0 = b_0\cdot Q_0 - P_0.\\ Q_1 &= n-b_0^2 = (n-P_1^2)/Q_0. \end{aligned}$$

$i \to i+1$: Suppose that the assumption is true for i. We obtain

$$\begin{aligned} x_{i+1} &= \frac{1}{x_i - b_i} = \frac{1}{\frac{\sqrt{n}+P_i}{Q_i} - b_i} \\ &= \frac{Q_i}{\sqrt{n} + P_i - b_i \cdot Q_i} = \frac{Q_i}{\sqrt{n} - P_{i+1}} \\ &= \frac{Q_i \cdot (\sqrt{n} + P_{i+1})}{n - P_{i+1}^2}. \end{aligned}$$

We must now prove that $Q_i \,|\, (n - P_{i+1}^2)$. Since $Q_i = (n - P_i^2)/Q_{i-1} \in \mathbb{Z}$, we have

$$\frac{n - P_{i+1}^2}{Q_i} = \frac{n - (b_i \cdot Q_i - P_i)^2}{Q_i} \in \mathbb{Z}.$$

Finally, we prove the estimates for P_i, Q_i:
For $i = 0$ the assertion is true. Suppose that the assertion is true for i. We note that

$$x_i - b_i = \frac{\sqrt{n} - P_{i+1}}{Q_i}.$$

Hence

$$0 < \frac{\sqrt{n} - P_{i+1}}{Q_i} < 1$$

or

$$0 < \sqrt{n} - P_{i+1} < Q_i < 2\sqrt{n}$$

implying

$$|P_{i+1}| < \sqrt{n}.$$

Now

$$0 < \frac{1}{x_{i+1}} = \frac{Q_{i+1}}{\sqrt{n} + P_{i+1}} < 1,$$

which shows that

$$0 < Q_{i+1} < \sqrt{n} + P_{i+1} < 2\sqrt{n}.$$

□

Note that from Proposition 3.5 follows that the regular continued fraction algorithm (RCFA) of $\sqrt{n}$ is periodic. The clue is

3.6. Proposition *If $x = \sqrt{n}$, P_i, Q_i as in Proposition 3.5, A_i, B_i as in Proposition 3.2, then we have*

$$A_{i-1}^2 - n \cdot B_{i-1}^2 = (-1)^i \cdot Q_i$$

for $i \geq 0$.

Proof: By Proposition 3.4 we have for $i \geq 1$

$$x = \frac{A_{i-2} + A_{i-1} \cdot x_i}{B_{i-2} + B_{i-1} \cdot x_i}.$$

Since by Proposition 3.5 $x_i = \frac{\sqrt{n}+P_i}{Q_i}$ we find

$$\sqrt{n} = \frac{Q_i \cdot A_{i-2} + P_i \cdot A_{i-1} + A_{i-1} \cdot \sqrt{n}}{Q_i \cdot B_{i-2} + P_i \cdot B_{i-1} + B_{i-1} \cdot \sqrt{n}}.$$

Hence

$$\sqrt{n} \cdot (Q_i \cdot B_{i-2} + P_i \cdot B_{i-1}) + n \cdot B_{i-1} = Q_i \cdot A_{i-2} + P_i \cdot A_{i-1} + A_{i-1} \cdot \sqrt{n}.$$

Comparing coefficients yields the equations

$$\begin{aligned} Q_i \cdot B_{i-2} + P_i \cdot B_{i-1} &= A_{i-1} \\ Q_i \cdot A_{i-2} + P_i \cdot A_{i-1} &= n \cdot B_{i-1}. \end{aligned}$$

Eliminating P_i yields

$$Q_i \cdot (A_{i-1} \cdot B_{i-2} - A_{i-2} \cdot B_{i-1}) = A_{i-1}^2 - n \cdot B_{i-1}^2.$$

The assertion follows from Proposition 3.3. □

If we consider the result of Proposition 3.6 modulo n, then we have

$$A_{i-1}^2 \equiv (-1)^i \cdot Q_i \bmod n.$$

By Proposition 3.5 we also have

$$0 < Q_i < 2\sqrt{n}.$$

The RCFA of $\sqrt{n}$ therefore yields solutions of (3.3) with small $|z_i|$. The amount of computing time for the attempt to factor z_i over the factor base $\mathcal{F}$ is comparatively small.

It might however happen that the period length of the expansion of $\sqrt{n}$ is very small. In this case, one considers the RCFA of $\sqrt{f \cdot n}$ for a suitable factor $f \in \mathbb{Z}_{\geq 1}$ instead. In this way, Brillhart and Morrison were able in 1970 to factor

$$F_7 = 2^{2^7} + 1.$$

They calculated 1330000 Q_i 's in the RCFA of $\sqrt{257 \cdot F_7}$, factoring 2059 Q_i's.

3.1. Exercise *Find an infinite family of non square numbers $n \in \mathbb{Z}_{\geq 1}$ such that the RFCA of $\sqrt{n}$ has period length 1.*

Though the quadratic residues produced by the continued fraction method are quite small, trial dividing them takes a long time. This is why C. Pomerance says that factoring 50 digit numbers is probably the limit of the basic continued fraction factoring algorithm.

3.1.3 The quadratic sieve algorithm (QS)

We now turn to a method which avoids most of the trial divisions, the **quadratic sieve algorithm** ([Po85]). Using this method numbers with more than 100 digits could be factored.

Again, the number to be factored is called n. We calculate $m = \lfloor \sqrt{n} \rfloor$ and we put

$$z(x) = (x+m)^2 - n \tag{3.9}$$

for $x \in \mathbb{Z}$. Note that

$$\begin{aligned} |z(x)| &\leq (|x| + \sqrt{n})^2 - n \\ &= |x|^2 + 2|x| \cdot \sqrt{n} + n - n \\ &= |x| \cdot (|x| + 2\sqrt{n}), \end{aligned}$$

i.e. for small x we have $z(x) = O(\sqrt{n})$ and that

$$z(x) \equiv (x+m)^2 \bmod n.$$

Hence $z(x)$ is a quadratic residue modulo n which is not too large. We therefore try to factor $z(x)$ over the factor base

$$\mathcal{F} \;=\; \{p_0, p_1, \ldots, p_l\}.$$

Suppose that for $p \in \mathcal{F}$

$$p \mid z(x). \tag{3.10}$$

This can be written in the form

$$z(x) \equiv 0 \bmod p. \tag{3.11}$$

Hence x is a zero of the quadratic polynomial $\overline{z}(x)$ in the finite field $\mathbb{Z}/p\mathbb{Z} = \mathbb{F}_p$.

3.7. Proposition *Let p be a prime number and let $g \in \mathbb{F}_p[x]$ be of degree k. Then g has at most k zeros in $\mathbb{F}_p$.*

Proof: For each zero $\overline{\lambda} \in \mathbb{F}_p$ we can divide g by $x - \overline{\lambda}$ ($\mathbb{F}_p[x]$ is euclidean). □

By Proposition 3.7 the quadratic polynomial $\overline{z}(x)$ has at most two zeros modulo p. Assume that we know the zeros x_p and x_p' of $z(x)$ modulo p. Both x_p, x_p' can be found in the interval $\{0, 1, \ldots, p-1\}$. All the other zeros of $z(x)$ modulo p are of the form

$$x = x_p + k \cdot p, \;\; x' = x_p' + k \cdot p, \quad k \in \mathbb{Z}. \tag{3.12}$$

So we do not have to trial divide all the quadratic residues $z(x)$ by p but we can predict by (3.12) for which arguments x the value $z(x)$ is divisible by p.

We briefly exemplify how this observation can be used to avoid trial division of the $z(x)$.

3.3. Example *Let* $n = 323$, $\mathcal{F} = \{2, 3, 5, 7\}$. *Then*

$$m = \lfloor\sqrt{n}\rfloor = 17,$$
$$z(x) = (x+17)^2 - 323 = x^2 + 34x - 34.$$

$p = 2:$ $z(x) \equiv x^2 \bmod 2$.

There is precisely one zero modulo 2, $x_2 = 0$. *All the other zeros are of the form*

$$x = k \cdot 2.$$

$p = 3:$ $z(x) \equiv (x-1)^2 + 1 \bmod 3$.

To solve $z(x) \equiv 0$ *mod* 3 *we must solve*

$$(x-1)^2 \equiv -1 \bmod 3,$$

i.e. we must determine whether -1 *is a square modulo 3. Since the only squares modulo 3 are 0 and 1, it follows that* $z(x)$ *is never divisible by 3 and we may as well remove 3 from the factor base.*

$p = 5:$ *With* $p = 5$ *the same can be done as with* $p = 3$.

$p = 7:$ $z(x) \equiv (x+3)^2 - 1 \bmod 7$

To solve $z(x) \equiv 0$ *mod* 7 *we must solve*

$$(x+3)^2 \equiv 1 \bmod 7 \quad \Leftrightarrow \quad (x+3) \equiv \pm 1 \bmod 7.$$

We get the solutions

$$x_7 = 3,\ x_7' = 5.$$

All zeros of $z(x)$ *modulo 7 are of the form*

$$x = 3 + k \cdot 7,\ x' = 5 + k \cdot 7.$$

In order to use this information, we write down a one-dimensional array of $\mathbb{Z}$*-values:*

x	1	2	3	4	5	6	7	8	9	10	11	12	13
$z(x)$	1	38	77	118	161	206	253	302	353	406	461	518	577

We know that $z(0)$ *is divisible by* 2 *and that we have to increment* $x_2 = 0$ *by multiples of* 2 *to find the other values of* $z(x)$, *which are divisible by* 2.

$z(x)/2$ *1* *19* *77* *59* *161* *103* *253* *151* *353* *203* *461* *259* *577*

The same procedure is applied to $p = 7$.

If we add 11 *to our factor base, then we see that* $z(3)$ *can be completely factored over* $\mathcal{F}$ *:*

$$z(3) = 77 = 7 \cdot 11.$$

$z(x)/7$ 1   19   11   59   23   103   253   151   353   29   461   37   577

7

7

In general, the procedure consists of three steps:

1. Compute the value of $z(x)$ for every $x \in \mathbb{Z}$ with $|x| \leq M$ ($\{-M, -M+1, \ldots, M\}$ is called the **sieving intervall)** and store those values.
2. For every prime number $p \in \mathcal{F}$ compute the numbers $x \in \{0, \ldots, p-1\}$ with $z(x) \equiv 0 \bmod p$. If there is none remove p from $\mathcal{F}$, otherwise call them x_p (and x'_p).
3. For every $p \in \mathcal{F}$:

   For every $x = x_p + k \cdot p$ with $|x| \leq M$ replace $z(x)$ by $z(x)/p$. (Do the same with $x' = x'_p + k' \cdot p$ if applicable).

   If $z(x) = 1$, then $z(x)$ can be decomposed completely over $\mathcal{F}$.

We note that this procedure only yields those numbers x, for which $z(x)$ is a product of pairwise distinct prime numbers in $\mathcal{F}$. It can however be generalized to prime powers. Assume we know x_k with $x_k^2 \equiv n \bmod p^k$. In order to solve $x_{k+1}^2 \equiv n \bmod p^{k+1}$ we take $x_{k+1} = x_k + c \cdot p^k$ where $c \equiv -d \cdot (2 \cdot x_k \cdot p^k)^{-1} \bmod p$ and $d \equiv x_k^2 - n \bmod p^{k+1}$.

If we have obtained $z(x) = 1$ for some x, then we can easily reconstruct the decomposition of $z(x)$ over $\mathcal{F}$ by trial division.

The main problem that has to be dealt with now is: How can we decide whether $z(x) \equiv 0 \bmod p$ is solvable and, in the case of solvability, how can we determine the absolute smallest solutions. Since $z(x) = (x+m)^2 - n$, checking the solvability of $z(x) \equiv 0 \bmod p$ for a prime number p means checking whether n is a square modulo p.

3.1.4 Computing square roots modulo p

We insert a little theory about squares modulo odd primes p. For $a \in \mathbb{Z}$ we define the **Legendre symbol**

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{if } \gcd(a, p) = 1 \text{ and } a \text{ is a square mod } p, \\ -1 & \text{if } \gcd(a, p) = 1 \text{ and } a \text{ is not a square mod } p, \\ 0 & \text{if } p \mid a. \end{cases}$$

If $\left(\frac{a}{p}\right) = 1$, then a is called **quadratic residue** modulo p and if $\left(\frac{a}{p}\right) = -1$, a is called **quadratic non residue** modulo p.

In order to be able to decide whether or not $a \in \mathbb{Z}$ with $\gcd(a, p) = 1$ is a square modulo p and to calculate square roots mod p, we have to study the group $\mathbb{F}_p^* = (\mathbb{Z}/p\mathbb{Z})^*$ of primitive residues mod p.

3.8. Proposition *$\mathbb{F}_p^*$ is a cyclic group of order $p-1$.*

Proof: [Ko87, Prop. II.1.2] □

In general, it is quite easy to find $g \in \mathbb{Z}$ such that $\overline{g}$ generates $\mathbb{F}_p^*$. Such a g is called **primitive root** modulo p.

3.4. Example *Let $p = 13$. We test whether 2 is a primitive root modulo 13.*

$$\overline{2}^0 = \overline{1},\ \overline{2}^1 = \overline{2},\ \overline{2}^2 = \overline{4},\ \overline{2}^3 = \overline{8},\ \overline{2}^4 = \overline{3},\ \overline{2}^5 = \overline{6},$$
$$\overline{2}^6 = \overline{12},\ \overline{2}^7 = \overline{11},\ \overline{2}^8 = \overline{9},\ \overline{2}^9 = \overline{5},\ \overline{2}^{10} = \overline{10},\ \overline{2}^{11} = \overline{7},\ \overline{2}^{12} = \overline{1}.$$

In fact, 2 is a primitive root modulo 13.

3.9. Proposition *(Eulers's criterion)*
For $a \in \mathbb{Z}$ with $\gcd(a, p) = 1$ we have

$$a^{(p-1)/2} \equiv \left(\frac{a}{p}\right) \bmod p.$$

3.2. Exercise *Use Proposition 3.8 to prove Proposition 3.9.*

Proposition 3.9 yields an $O(\log p)$ algorithm for computing $\left(\frac{a}{p}\right)$.

3.10. Proposition *If $p \equiv 3$ mod 4, $\gcd(n, p) = 1$ and $\left(\frac{n}{p}\right) = 1$, then*

$$x \equiv -m \pm n^{(p+1)/4} \bmod p$$

are the two zeros of $z(x)$ mod p.

Proof: By Proposition 3.9 we have

$$\begin{aligned} z(x) &\equiv (\pm n^{(p+1)/4} - m + m)^2 - n \\ &\equiv n^{(p-1)/2} \cdot n - n \\ &\equiv 0 \bmod p. \end{aligned}$$

□

We now introduce Shanks' algorithm RESSOL for computing square roots modulo p. Let p be an odd prime, $a \in \mathbb{Z}$, $\gcd(a, p) = 1$, $\left(\frac{a}{p}\right) = 1$. Our aim is to compute an $r \in \mathbb{Z}$ such that $r^2 \equiv a \bmod p$.

The idea is to calculate a sequence $(r_i,\, n_i)$ such that

$$r_i^2 \equiv a \cdot n_i \bmod p \tag{3.13}$$

and the order of n_i mod p (i.e. the smallest exponent $e_i \in \mathbb{Z}_{\geq 1}$ with $n_i^{e_i} \equiv 1$ mod p) is strictly decreasing. Then we will eventually have $n_i \equiv 1$ mod p and $r_i^2 \equiv a$ mod p as desired.

For this purpose we compute a quadratic non residue $z \in \mathbb{Z}$ mod p by trial. Since half of the primitive residues mod p are non squares, the probability of success is quite high. In fact, it can be shown under the assumption of the generalized Riemann hypothesis that the first quadratic non residue is of magnitude $O((\log p)^2)$.

Now we determine $s_0,\, k \in \mathbb{Z}_{\geq 0}$ such that

$$p = 2^{s_0} \cdot (2k+1) + 1.$$

Then the order of $\mathbb{F}_p^*$ is $2^{s_0} \cdot (2k+1)$. The subgroup G_1 of elements of two power order is of order 2^{s_0} and the subgroup G_2 of elements of odd order is of order $2k+1$. Both these groups are cyclic, because $\mathbb{F}_p^*$ is cyclic.

We now initialize the algorithm by setting

$$r_0 \equiv a^{k+1} \bmod p,\;\; n_0 \equiv a^{2k+1} \bmod p.$$

Then (3.13) is easily verified. Moreover we set $c_0 \equiv z^{2k+1} \bmod p$. Then the order of c_0 is even and, because z is a quadratic non residue, c_0 generates G_1. It is easily verified that the order of n_0 is a power of two, i.e. $n_0 \in G_1$. Finally, we initialize the loop parameter: $i = 0$. If $n_i \equiv 1 \bmod p$, the algorithm terminates.

Assume that $\overline{n}_i \in \langle \overline{c}_i \rangle$ (= cyclic group generated by $\overline{c}_i$), hence we can write

$$n_i \equiv c_i^{2^{t_i}(2m_i-1)} \bmod p.$$

t_i can be easily determined as we will see below, but the computation of m_i would be consuming too much time. Because a is a quadratic residue we can show that if $n_0 \not\equiv 1 \bmod p$ we have $t_0 > 0$. By induction one can prove that for $n_i \not\equiv 1 \bmod p$ the number $t_i > 0$.

We now want to modify r_i:

$$r_{i+1} \equiv r_i \cdot b_i \bmod p.$$

Then

$$n_{i+1} \equiv n_i \cdot b_i^2 \bmod p,$$

and we want to choose b_i, such that the order of n_{i+1} is lower than the order of n_i. Shanks' idea is to choose

$$b_i \equiv c_i^{2^{t_i-1}} \bmod p,$$

so

$$n_{i+1} \equiv c_i^{2^{t_i}(2m_i-1)} \cdot c_i^{2^{t_i}} \equiv c_i^{2^{t_i}\cdot 2m_i} \bmod p.$$

Finally, we note that for

$$c_{i+1} \equiv b_i^2 \bmod p$$

we have $\overline{n}_{i+1} \in \langle \overline{c}_{i+1} \rangle$, hence the recursion can start all over again.

But how can we determine t_i ? Again by induction we can show that the order of c_i is a power of two. Let $\mathrm{ord}(c_i) = 2^{s_i}$. Then we directly get that $\mathrm{ord}(n_i) = 2^{s_i - t_i}$. Thus we can compute the number t_i if we compute the order of n_i by repeated squaring and then set $t_i = s_i - l$ (where l is the number of squarings we need until we have found the order of n_i). Then set $s_{i+1} = s_i - t_i$. It is quickly verified that $2^{s_{i+1}}$ is the order of c_{i+1}.

Because $t_i > 0$ we conclude that $s_{i+1} < s_i$ and thus the order of n_i is strictly decreasing.

3.3. Exercise *Write a program to extract square roots modulo p.*

3.1.5 Modifications of QS

Several modifications of the QS have been suggested. It is, for example, possible to store 8-bit approximations of $\log z(x)$ rather that $z(x)$ and to subtract $\log p$ whenever $p \mid z(x)$. If $\log z(x)$ has become very small, one can attempt to factor the corresponding $z(x)$.

It is also possible to consider a factorization

$$z(x) = L \cdot \prod_{p \in \mathcal{F}} p^{e_p}$$

up to a certain magnitude of L as a complete factorization of $z(x)$ and to hope that the cofactor L shows up several times. This is called the **large prime variant**.

3.1.6 Parallel version of QS

Reference: [Si87]

Instead of trying to factor $z(x) = (x + \lfloor \sqrt{n} \rfloor)^2 - n$ over a sieving interval $[-M, M]$ one may try to factor values of other quadratic polynomials,

$$\begin{aligned} Q(x) &= Ax^2 + Bx + C \\ &= A \cdot (x + B(2A)^{-1})^2 - (B^2 - 4 \cdot A \cdot C)(4A)^{-1} \qquad (3.14) \\ \gcd(A, n) &= 1. \end{aligned}$$

In order to make $Q(x)$ generate quadratic residues modulo n, we ask that

$$B^2 - 4 \cdot A \cdot C = k \cdot n$$

for some $k \in \mathbb{Z}$. Under the assumption that A is a quadratic residue mod n we have $Q(x) \equiv A \cdot (x + B(2A)^{-1})^2 \bmod n$ is a quadratic residue mod n.

It is advantageous to keep the values of $Q(x)$ small. Without loss of generality we assume that A, $B \geq 0$. Then the maximum value of $Q(x)$ is either

$$A \cdot \left(M + \frac{B}{2A}\right)^2 - \frac{k \cdot n}{4A} \qquad \text{or} \qquad \frac{|k| \cdot n}{4A}.$$

Let $k \geq 0$. If $A \cdot \left(M + \frac{B}{2A}\right)^2 - \frac{k \cdot n}{4A} \geq \frac{k \cdot n}{4A}$, then the optimal choice is $B = 0$, $A = \frac{\sqrt{2}}{2}\frac{\sqrt{k \cdot n}}{M}$. The same is true for $A \cdot \left(M + \frac{B}{2A}\right)^2 - \frac{k \cdot n}{4A} \leq \frac{k \cdot n}{4A}$.
Consequently, we have

$$C = \frac{M \cdot \sqrt{k \cdot n}}{2\sqrt{2}}.$$

The idea is to choose values for A, B, C which are close to those optimal choices.

Silverman suggests the following: Let $A = D^2$ where D is a (probable) prime $D \equiv 3 \bmod 4$, $(\frac{k \cdot n}{D}) = 1$, $A \approx \frac{\sqrt{k \cdot n}}{\sqrt{2} \cdot M}$.
To find the coefficients compute

$$\begin{aligned} h_0 &\equiv (k \cdot n)^{(D-3)/4} \bmod D \\ h_1 &\equiv k \cdot n \cdot h_0 \equiv (k \cdot n)^{(D+1)/4} \bmod D. \end{aligned}$$

Then

$$h_1^2 \equiv k \cdot n \cdot (k \cdot n)^{(D-1)/2} \equiv k \cdot n \bmod D$$

since $\left(\frac{k \cdot n}{D}\right) = 1$.
Let

$$h_2 \equiv (2 \cdot h_1)^{-1} \cdot \left[\frac{k \cdot n - h_1^2}{D}\right] \bmod D$$

and

$$B \equiv h_1 + h_2 \cdot D \bmod D^2.$$

Then

$$\begin{aligned} B^2 &\equiv h_1^2 + 2 \cdot h_1 \cdot h_2 \cdot D + h_2^2 \cdot D^2 \bmod D^2 \\ &\equiv k \cdot n \bmod D^2 \end{aligned}$$

because

$$2 \cdot h_1 \cdot h_2 \equiv \frac{k \cdot n - h_1^2}{D} \bmod D$$

and thus

$$2 \cdot h_1 \cdot h_2 \cdot D \equiv k \cdot n - h_1^2 \bmod D^2.$$

With those choices one has by (3.14)

$$Q(x) \equiv \left(\frac{2 \cdot A \cdot x + B}{2D}\right)^2 \bmod k \cdot n$$

which is easy to evaluate and a square modulo $k \cdot n$. The roots of $Q(x) \bmod p$ for a prime p are

$$(-B \pm \sqrt{k \cdot n})(2A)^{-1} \bmod p.$$

The cost of finding the coefficients is dominated by the probable prime and residue tests on D, the computation of h_1 and $(2 \cdot h_1)^{-1} \bmod D$ and $(2 \cdot D)^{-1} \bmod k \cdot n$.

3.1.7 The Number Field Sieve (NFS)

Reference: [LLMP]

Let $f \in \mathbb{Z}[x]$ be a monic and irreducible polynomial and $m \in \mathbb{N}$ be a number with

$$f(m) \equiv 0 \bmod n.$$

3.5. Example *The seventh Fermat number $F_7 = 2^{2^7} + 1 = 2^{128} + 1$, a number with 39 digits, was first factored in 1970 by Brillhard and Morrison. If we look for a suitable polynomial f for this number we get some problems. Thus we use a trick and try to factor the number $n = 2 \cdot F_7 = 2^{129} + 2$. Now we can use*

$$f(x) = x^3 + 2 \qquad \text{and} \qquad m = 2^{43}.$$

Remark: One of the main problems of the general number field sieve is the problem of finding a suitable polynomial f and a suitable number m for a given n. If n has a special form, for example $n = b^r - c$ with small b and c (n is a so called **Cunningham number**), we find f and m easily. But in general this problem generates a lot of difficulties (for example nobody has yet found m and f for the RSA-Challenge-numbers). Nevertheless the following construction always succeeds:

Let d be the degree of f. We look for a polynomial

$$f(x) = f_0 \cdot x^d + f_1 \cdot x^{d-1} + \ldots + f_d$$

and a number m with $f(m) = f_0 \cdot m^d + f_1 \cdot m^{d-1} + \ldots + f_d = n$. Set $m = \lfloor n^{1/d} \rfloor$. Then we can quickly find $f_0, \ldots, f_d$ with $0 \le f_j < m$. This polynomial can in a simple manner be modified such that it gets irreducible and monic and that $f(m) = n$. Unfortunately this construction leads in general to polynomials with great coefficients. But we need a polynomial f with small coefficients as we will see below.

Now let $f(x) = a_0 + a_1 x + \ldots + x^d$. Regard

$$\mathbb{Z}[\rho] = \left\{ a_0 + a_1 \rho + \ldots + a_{d-1} \rho^{d-1} \; ; \; a_j \in \mathbb{Z} \right\}$$

with ρ a formal zero of f. We claim that $\mathbb{Z}[\rho]$ is a ring with 1 and no zero divisors. In addition we assume that $\mathbb{Z}[\rho]$ is a UFD, i.e. every element is a unique power product of prime elements.

Then we introduce the following map:

$$\begin{array}{rccc} \varphi: & \mathbb{Z}[\rho] & \longrightarrow & \mathbb{Z}/n\mathbb{Z} \\ & \sum_{i=0}^{d-1} a_i\rho^i & \longmapsto & \sum_{i=0}^{d-1} a_i m^i \end{array} \tag{3.15}$$

3.1. Lemma *The map φ of (3.15) is a ring homomorphism.*

3.4. Exercise *Proof this lemma.*

Now we describe the idea of the NFS to generate a relation. First we choose a constant $C > 0$. Then we fix two factor bases. The first factor basis consists of all prime numbers of $\mathbb{Z}$ up to the constant C; the second factor basis is build up by special prime elements of $\mathbb{Z}[\rho]$. More precisely, we choose

$$\begin{array}{lcl} F_1 & = & \{p \in \mathbb{P};\, p \leq C\} = \{p_1, \ldots, p_k\} \\ F_2 & = & \{\pi;\, \pi \text{ prime element in } \mathbb{Z}[\rho],\ N(\pi) \in F_1\} = \{\pi_1, \ldots, \pi_l\} \end{array}$$

Hereby $N(x)$ is the norm of the element x; an expression which will be discussed later. In addition we choose a set of units $U = \{u_0, \ldots, u_r\}$ such that U generates the unit group of $\mathbb{Z}[\rho]$. Now we try to factor elements

$$a + b \cdot \rho \in \mathbb{Z}[\rho]$$

over the factor basis F_2 and the set U. If we succeed we try to factor $\varphi(a + b \cdot \rho) = a + b \cdot m$ over F_1. If we are successful in both cases we get

$$a + b \cdot \rho = \prod_{u \in U} u^{e_u} \cdot \prod_{\pi \in F_2} \pi^{e_\pi}$$

and

$$a + b \cdot m = v \cdot \prod_{p \in F_1} p^{e_p}$$

with $v \in \{\pm 1\}$. Thus we have

$$\begin{aligned} v \cdot \prod_{p \in F_1} p^{e_p} = a + b \cdot m &= \varphi(a + b \cdot \rho) \\ &= \varphi\left(\prod_{u \in U} u^{e_u} \cdot \prod_{\pi \in F_2} \pi^{e_\pi}\right) \equiv \prod_{u \in U} \varphi(u)^{e_u} \cdot \prod_{\pi \in F_2} \varphi(\pi)^{e_\pi} \bmod n. \end{aligned}$$

The numbers $\varphi(\pi)$ and $\varphi(u)$ are integers, but in general no prime numbers. Thus the relation

$$v \cdot \prod_{p \in F_1} p^{e_p} \equiv \prod_{u \in U} \varphi(u)^{e_u} \cdot \prod_{\pi \in F_2} \varphi(\pi)^{e_\pi} \bmod n$$

is in general not trivial. If we have "enough" relations we can construct a square by solving a linear system. This proceeds in the same way as in the QS. Note that in the NFS both sides of a relation are nonsquare, i.e. we have to find $|F_1| + |F_2| + |U| + 1$ relations to solve the linear system and to find a nontrivial solution for $x^2 \equiv y^2 \bmod n$.

Now we discuss a few problems. First we have not yet said how to construct the first factor basis F_1. But this is an easy problem, we can do it by using the sieve of Eratosthenes or by using a precomputed table of small prime numbers.

The second problem is the construction of the second factor basis F_2. To do this we have to explain some facts about number fields.

The **norm** of a number x in the number field F is the norm of the endomorphism

$$\begin{array}{ccc} F & \longrightarrow & F \\ \alpha & \longmapsto & x \cdot \alpha. \end{array} \tag{3.16}$$

Hereby F is regarded as d-dimensional vector space over $\mathbb{Q}$.

3.6. Example *The number field $F = \mathbb{Q}(\sqrt[3]{-2})$ has the basis $1, \sqrt[3]{-2}$ and $\sqrt[3]{4}$. If we have to compute the norm of $x = \sqrt[3]{-2}$ we have to compute the norm of the endomorphism $\alpha \longmapsto \sqrt[3]{-2} \cdot \alpha$ for all $\alpha \in F$. We can describe this map by the matrix*

$$\begin{pmatrix} 0 & 0 & -2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

(the matrix describes the images of the basis). The norm of the endomorphism is the determinant of this matrix. In our example we compute that the matrix has determinant -2 and so we get $N(\sqrt[3]{-2}) = -2$.

We can do the same procedure in general. If we have an element $x_0 + x_1 \cdot \rho + \ldots + x_{d-1} \cdot \rho^{d-1}$ we can compute the matrix of the endomorphism (3.16) and the determinant of this matrix in general. If our polynomial has the form $f(x) = x^d - c$ then we have $N(x_0 + x_1 \cdot \rho) = x_0^d - c \cdot (-x_1)^d$.

For the computation of the second factor basis we first have to check whether there indeed exists a prime element $\pi \in \mathbb{Z}[\rho]$ with $N(\pi) = p$. This can be done according to the following proposition.

3.11. Proposition *For a prime number p there exists a prime element π with $N(\pi) = p$ iff $f \bmod p$ has a linear factor. For every linear factor there exists exactly one prime element π.*

3.7. Example *We will illustrate this proposition in the following example. We use the polynomial $f(x) = x^3 + 2$ and the factor basis $F_1 = \{2, 3, 5, 7, 11\}$. According to proposition 3.11 we get*

$$\begin{aligned} x^3 + 2 &\equiv x \cdot x \cdot x \bmod 2 \\ x^3 + 2 &\equiv (x-1)^3 \bmod 3 \end{aligned}$$

$$\begin{aligned} x^3+2 &\equiv (x-2)\cdot(x^2+2x+4) \bmod 5 \\ x^3+2 & \quad \textit{is irreducible} \bmod 7 \\ x^3+2 &\equiv (x-4)\cdot(x^2+4x+5) \bmod 11. \end{aligned}$$

Thus we know that there exists no prime element π with $N(\pi) = 7$. For the other prime numbers $2, 3, 5$ and 11 there exists exactly one prime element of the corresponding norm.

The second step consists of finding the prime elements π with $N(\pi) = p$. The prime elements of $\mathbb{Z}[\rho]$ with norm p are in 1-to-1 correspondence to (p, c_p) where $f(c_p) \equiv 0 \bmod p$. Thus randomly select small $h_i \in \mathbb{Z}$ such that

$$\sum_{i=0}^{d-1} h_i \cdot c_p^i \equiv 0 \bmod p \qquad \text{and} \qquad N\left(\sum_{i=0}^{d-1} h_i \cdot \rho^i\right) = k \cdot p.$$

Mostly we will get $k = 1$ and we have found a generator of the prime ideal of norm p corresponding to (p, c_p). Otherwise divide the generator by an appropriate generator of an ideal with norm k.

3.8. Example *We use trial and error to compute the factor basis F_2. We randomly choose small x_1, x_2, x_3 and compute the norm of $x = x_1 + x_2 \cdot \sqrt[3]{-2} + x_3 \cdot \sqrt[3]{4}$. We have $N(x) = x_1^3 - 2x_2^3 + 4x_3^3 + 6x_1 \cdot x_2 \cdot x_3$. Thus we get*

x_1	x_2	x_3	$N(x)$
0	*1*	*1*	*2*
-1	*0*	*1*	*3*
1	*0*	*1*	*5*
3	*2*	*0*	*11*

Thus we have found $F_2 = \{\sqrt[3]{-2} + \sqrt[3]{4}, -1 + \sqrt[3]{4}, 1 + \sqrt[3]{4}, 3 + 2 \cdot \sqrt[3]{-2}\}$.

The computation of the set U of units can be done by the same way. Let s be the number of real roots of f and $l = \frac{d+s}{2} - 1$. According to Dirichlet's Unit Theorem we know that the group of units is generated by an appropriate root of unity u_0 and l independent elements $u_1, \ldots, u_l$. If $s > 0$ we can choose $u_0 = -1$. By computing the norms of random elements it is not hard to find l multiplicative independent elements of norm ± 1. The elements u_i should be chosen in the way that special embeddings are positive. More details can be found in the part where we describe the computation of the unit decomposition. Note that the computation of the set U can be accomplished during the computation of F_2.

After knowing how to compute the factor bases we have to explain how to do the factorization of an element $a + b \cdot \rho \in \mathbb{Z}[\rho]$ for small a and b.

3.12. Proposition *The prime element π with $N(\pi) = p$ corresponding to (p, c_p) divides $a + b \cdot \rho$ iff*

$$a + b \cdot c_p \equiv 0 \bmod p.$$

In addition we know that if π divides $a + b \cdot \rho$ then p divides the norm of $a + b \cdot \rho$. The exponent of π in the factorization of $a + b \cdot \rho$ is exactly the exponent of p in $N(a + b \cdot \rho)$.

3.9. Example *We want to find a factorization of $\alpha = 2 - 3 \cdot \sqrt[3]{-2}$ over the factor basis F_2. For the prime number 2 we have $2 - 3 \cdot c_2 = 2 - 3 \cdot 0 \equiv 0 \bmod 2$ and we know that π_2 (the prime element of norm 2) is a divisor of α. By computing*

$$N(2 - 3\sqrt[3]{-2}) = 8 - 2 \cdot 27 = -46 = -1 \cdot 2 \cdot 23$$

we see that π_2 has exponent 1 in the factorization of α.

For hastening the factorization of $a + b \cdot \rho$ one can use the same idea as in the QS and use a sieve. The idea for using the sieve is that

$$\pi \mid a + b \cdot \rho \quad \text{iff} \quad a + b \cdot \rho = a_0 + k \cdot p + b_0 \cdot \rho$$

with numbers a_0 and b_0 such that $a_0 + b_0 \cdot c_p \equiv 0 \bmod p$.

So the procedure for finding a relation is the following: Fix a value b_0. Compute an array $T[a] = a + b_0 \cdot m$ for a less the sieving bound. Sieve this array analog to the QS using the first factor basis F_1. Then compute $T[a] = N(a + b_0 \cdot \rho)$ for the numbers a with $a + b \cdot m$ factors completely over F_1 and $T[a] = \infty$ otherwise. Then sieve the norms: for π corresponding to (p, c_p) compute the minimal a_0 such that $a_0 + b_0 \cdot c_p \equiv 0 \bmod p$. Then divide $T[a_0]$ by p. All other numbers $a + b_0 \cdot \rho$ which are divided by π are of the form $a_0 + k \cdot p + b_0 \cdot \rho$. Thus we could sieve this values too. At the end all pairs (a, b_0) with $T[a] = \pm 1$ are good pairs, i.e. $a + b_0 \cdot m$ and $a + b_0 \cdot \rho$ factor completely over F_1, F_2 respectively. Note that we can use logarithms as in the case of the QS.

But how can we find the unit decomposition of the elements $a + b_0 \cdot \rho$? An easy solution is the following: store all units in a table and use table lookup. Another solution would be to wait until you have found two relations with the same unit part and then divide the two relations. Lenstra suggests the following:

Choose l embeddings of $\mathbb{Q}(\rho)$ into $\mathbb{C}$ which are not complex conjugate. Denote by x_i the image of the i-th embedding of x. Then define the map

$$\nu(x) = \left(\log|x_1| - \frac{\log|N(x)|}{d}, \ldots, \log|x_l| - \frac{\log|N(x)|}{d}\right) \in \mathbb{R}^l.$$

Under the mapping ν the group of units form a lattice in $\mathbb{R}^l$, i.e. for a unit u we have $\nu(u^e) = e \cdot \nu(u)$. Let W be the matrix having $\nu(u_i)^t$ as i-th column. Then the columns of W form a basis of this lattice.

Now let $x = \frac{a+b\cdot\rho}{\prod_{\pi \in F_2} \pi^{e_\pi}}$ be the unit part of $a + b \cdot \rho$. Then $x = u_0^{e_0} \cdot \prod_{i=1}^{l} u_i^{e_i}$ and therefore

$$\nu(x) = e_0 \cdot \nu(u_0) + \ldots + e_l \cdot \nu(u_l).$$

If $s > 0$ and we have chosen the u_i such that for the real embedding the images of the u_i for $i = 1, \ldots, l$ are positive, we can compute e_0 as the sign of the real embedding of $a + b \cdot \rho$. Now note that

$$\nu(x)^t = e_0 \cdot \nu(u_0)^t + W \cdot \begin{pmatrix} e_1 \\ \vdots \\ e_l \end{pmatrix}.$$

So we can compute the e_i as

$$\begin{pmatrix} e_1 \\ \vdots \\ e_l \end{pmatrix} = W^{-1} \cdot \left(\nu(x)^t - e_0 \cdot \nu(u_0)^t\right).$$

For $s > 0$ we can compute e_0. Otherwise we should try several e_0 until all other e_i are integers and then check by trial division. In practice we compute the images $\nu(u_i)$ and the matrix W^{-1} once and in limited precision and check the correctness by trial division.

Similar to the QS we can use large primes. In the NFS large primes can occur as primes in $\mathbb{Z}$ and large primes in $\mathbb{Z}[\rho]$. The first sort of large primes can be handled as in the QS. Two relations with the same large prime in $\mathbb{Z}[\rho]$ can be combined by dividing and working with negative exponents.

3.2 Solving large sparse linear systems

To find the solution to the relation $x^2 \equiv y^2 \bmod n$ we have to solve a system of linear equations over the field $\mathbb{F}_2$. For numbers with 100 digits this system is of size $10^5 \times 10^5$. Thus we need 10^{10} bits if we store the matrix in the normal way.

If we use Gaussian Elimination to solve the linear system we need 10^{15} operations. Using a supercomputer (10^8 - 10^9 operations per second) or a vector computer (10^3 operations simultaniously) it might be possible to work with Gaussian Elimination. But in the general NFS and for computing discrete logarithms we have linear systems over a prime field with characteristic $\approx 10^{100}$. Here we need 125 gigabyte storage and 10^{19} bit operations if we would use Gaussian Elimination.

But the linear systems have a speciality: they are sparse and of the following structure:

$$\left(\begin{array}{c|c} \text{small dense part} & \text{large sparse part} \end{array} \right).$$

This is naturally because it is impossible that a lot of great primes out of the factor basis divide the number we want to factor. Only small factor basis primes have a chance to divide a lot of test numbers.

For solving linear systems with matrices of this form there exist the following special methods:

1. Structured Gaussian Elimination
2. Conjugate Gradient Method
3. Coordinate Recurrence method of Wiedemann

We will now explain all this methods.

3.2.1 Structured Gaussian Elimination

We want to solve a linear system with a large matrix which is sparse. Moreover we assume that we have more relations as unknowns (i.e. more lines as columns). If we use the ordinary Gaussian elimination for solving the linear system the matrix rapidly fills with new entries and after a few iterations the matrix is no more sparse. Therefore we change the algorithm and try to work as long as possible with a sparse matrix.

The idea to do this is the following: just do line operations which generate no (or only few) new non zero entries. In this way we generate a system which is more dense, but smaller as the original system. This can be solved with ordinary Gauss elimination. After solving the small system the remaining variables can be computed by backward-solving.

How do we generate this smaller matrix? We give an algorithm:

1. Eliminate all columns which have only one non zero entry. Eliminate the corresponding line.
2. Divide the matrix in a heavy and a light part. The heavy part consists of the columns with the most entries. Odlyzko suggests to choose the heavy part of the matrix as the first $\approx \frac{n}{30}$ columns, where n is the total number of columns.
3. Let a_i be a line which has only one non zero entry a_{ij} in the light part. Without loss of generality we can assume that this entry a_{ij} is equal to ± 1. Subtract a suitable multiple of the line a_i from all lines a_k with $a_{kj} \neq 0$. In this way we produce a new column with only one non zero entry without creating a new entry in the light part of the matrix. New entries are only created in the heavy part of the matrix.

   This should be done until no more lines with only one non zero entry in the light part exist.
4. Go to step 1 and begin a new iteration. In this new iteration we fix a new (i.e. greater) heavy part in step 2.

Experimentelly Odlyzko has determined that the heavy part should be enlarged as minimal as possible to gain the best solution. If the heavy part becomes too large you should replace some of the most bad lines by some not yet used relations. By this way we generate a much smaller matrix which can be solved by ordinary Gaussian elimination.

Odlyzko has used this method to transform a linear system with 288017 relations and 95697 unknowns with only ≈ 18.2 non zero entries per line into a system with 7262 relations of 6006 unknowns, but ≈ 80 non zero entries per line.

3.2.2 The Conjugate Gradient Method

Originally this is a method of numerical analysis. We have to solve the linear system

$$A \cdot x = b \quad \text{with } A \in \mathbb{R}^{n \times n} \text{ symmetric and positive definite, } b \in \mathbb{R}^n.$$

For this special situation we can construct an approximative algorithm for solving the system and the convergence of the algorithm can be proven. These formulas can be carried over to finite fields. But in general we have to solve a system $B \cdot x = u$ with a non symmetric matrix B. Then consider

$$B^t \cdot B \cdot x = B^t \cdot u \qquad \text{and define } A := B^t \cdot B, b := B^t \cdot u.$$

The idea is to look at

$$F(x) \quad = \quad \frac{1}{2} \langle A \cdot x, x \rangle - \langle b, x \rangle .$$

Then we have $\operatorname{grad}(F) = A \cdot x - b$. If we find a minimum of F then we have $\operatorname{grad}(F) = 0$. In contrary A is the Hesse matrix of F and there exists a minimum.

How can we find a minimum of F? The idea is the following:

We start with an arbitrary x. Then we want to find a better x'. We set $x' = x + \lambda \cdot p$ with $\lambda \in \mathbb{R}$ and p the direction we want to change. The optimal direction we get if we use the gradient

$$p = \frac{A \cdot x - b}{||A \cdot x - b||}.$$

Then we have

$$f(x) = F(x + \lambda p) = F(x) + \lambda \langle A \cdot x - b, p \rangle + \frac{\lambda^2}{2} \langle A \cdot p, p \rangle.$$

This term is minimal for $\lambda = -\frac{\langle r,p \rangle}{\langle A \cdot p,p \rangle}$ with $r = A \cdot x - b$.

Over finite fields we get the formulas

$$x_0 \qquad \text{arbitrary}, \qquad p_0 = r_0 = b - A \cdot x_0$$

and

$$\begin{aligned} \alpha_i &= \frac{\langle r_i, r_i \rangle}{\langle p_i, A \cdot p_i \rangle}, \qquad x_{i+1} = x_i + \alpha_i \cdot p_i, \qquad r_{i+1} = r_i - \alpha_i \cdot A \cdot p_i \\ \beta_i &= \frac{\langle r_{i+1}, r_{i+1} \rangle}{\langle r_i, r_i \rangle}, \qquad p_{i+1} = r_{i+1} + \beta_i \cdot p_i \end{aligned}$$

We stop the iteration if $r_i = 0$. Then we have

$$x = x_i.$$

For a proof one should prove by induction that $A \cdot x_i = b - r_i$. But there still is a probleme. If we work over $\mathbb{F}_p$ and compute $\langle p_i, A \cdot p_i \rangle$, we might get the value 0. This would lead to an error. The probability of such an error is very small if the cardinality of the field is very great. Therefore we choose x_0 out of the extension field $\mathbb{F}_{p^l}$ for a suitable l. $\mathbb{F}_{p^l}$ should have " substantially" more elements than $\mathbb{F}_p$.

It can be shown that we need at most n iterations until we have found a solution to the equation $A \cdot x = b$. Therefore the algorithm needs totally $O(n \cdot (n + w))$ operations in $\mathbb{F}_{p^l}$, where w is the number of non zero entries of the matrix A.

3.2.3 The Coordinate Recurrence Method of Wiedemann

Reference: [Wi86]

Suppose $\mathbb{F}$ is a finite field, $m, n \in \mathbb{Z}_{\geq 1}$, $A \in \mathbb{F}^{m \times n}$, $b \in \mathbb{F}^m$ and assume that we want to solve the linear system

$$A \cdot x = b.$$

Wiedemann presents a probabilistic algorithm which solves the system using

$$O(n_1 \cdot (w + n_1) \cdot \log^k n_1)$$

field operations where n_1 is the rank of A, w is the number of non zero entries of A and k is some constant.

First assume that $n = m = n_1$. Hence A is a non singular square matrix. Consider the subspace $S = \operatorname{span}\{A^i \cdot b \mid i \in \mathbb{Z}_{\geq 0}\}$ of $\mathbb{F}^n$. On S define the endomorphism

$$\varphi : S \to S, \ \underline{v} \longmapsto A \cdot \underline{v}.$$

Let $f(x) = c_n + c_{n-1}x + \ldots + c_1 x^{n-1} + x^n$ be the minimal polynomial of φ. Then we have by the theorem of Cayley-Hamilton $f(\varphi)\underline{v} = \underline{0}$ for every $\underline{v} \in S$. In particular we know that

$$f(A)\underline{b} = \underline{0} = c_n \cdot \underline{b} + A \cdot \sum_{i=1}^{n} c_{n-i} \cdot A^{i-1} \cdot \underline{b} \qquad (c_0 := 1).$$

Since A is non singular we must have $c_n \neq 0$ because otherwise $\underline{x} = \sum_{i=1}^{n} c_{n-i} \cdot A^{i-1}\underline{b}$ would be a non trivial solution of the homogeneous equation $A \cdot \underline{x} = \underline{0}$. Therefore

$$\underline{x} = -c_n^{-1} \cdot \sum_{i=1}^{n} c_{n-i} \cdot A^{i-1} \cdot \underline{b}$$

is a solution of $A \cdot \underline{x} = \underline{b}$.

It is therefore sufficient to find the minimal polynomial of φ in order to solve the equation. The number of operations in $\mathbb{F}$ required for determining $\underline{x}$ is $O(n \cdot w)$, because the multiplication of A with $\underline{v} \in S$ requires $O(w)$ operations.

We now discuss the determination of the minimal polynomial f. Pick any $\underline{u} \in \mathbb{F}^n$ and consider the sequence

$$s_j = (\underline{u},\ A^j \cdot \underline{b}) \quad \text{for} \quad j \geq 0$$

(here $(\cdot,\ \cdot)$ denotes the ordinary inner product on $\mathbb{F}^n$). For this sequence we have

$$\begin{aligned} 0 &= (\underline{u},\ f(A)A^j \cdot \underline{b}) \\ &= (\underline{u},\ A^{n+j} \cdot \underline{b} + \sum_{i=1}^{n} c_i \cdot A^{n-i+j} \cdot \underline{b}) \\ &= (\underline{u},\ A^{n+j} \cdot \underline{b}) + \sum_{i=1}^{n} c_i \cdot (\underline{u},\ A^{n-i+j} \cdot \underline{b}) \\ &= s_{n+j} + \sum_{i=1}^{n} c_i \cdot s_{n+j-i}, \end{aligned}$$

which is a linear recurrence relation of length n. We will show that the linear recurrence of minimal length L_∞ which generates (s_j) is uniquely determined and coincides with the recurrence of minimal length L_N which generates $(s_j)_{j \leq N-1}$ for $N > 2n$. Such a linear recurrence of minimal length L_N can be inductively computed by the algorithm of Berlekamp-Massey.

Let $(s_j)_{0 \leq j < k}$ be a sequence of length k and L_k be the minimal length of a linear recurrence which generates $(s_j)_{0 \leq j < k}$, i.e.

$$\forall j = L_k, \ldots, k-1: \qquad s_j = -\sum_{i=1}^{L_k} c_i^{(k)} \cdot s_{j-i}.$$

The generating polynomial of this sequence is defined as

$$p_k(x) = 1 + c_1^{(k)} x + c_2^{(k)} x^2 + \ldots + c_{L_k}^{(k)} x^{L_k}.$$

If we have computed the linear recurrence of minimal length L_N for $N > 2n$, then we are ready. This can be accomplished inductively:

We define $L_{-1} = -1, L_0 = 0$ and $p_{-1}(x) = p_0(x) = 1$. Assume that we have found $p_j(x)$ and L_j for all $j \leq k$ and that we want to compute $p_{k+1}(x)$ and L_{k+1} with $L_{k+1} = \max\{L_k, k+1-L_k\}$. Define

$$d_k = s_k + \sum_{i=1}^{L_k} c_i^{(k)} s_{k-i}.$$

If $d_k = 0$, then $L_{k+1} = L_k$ because we can generated the $(k+1)$-th sequence element s_k with the linear recurrence $p_k(x)$ and we may choose $p_{k+1}(x) = p_k(x)$.

Otherwise let $m < k$ with $L_m < L_{m+1} = L_{m+2} = \ldots = L_k$. Then

$$0 \neq d_m = s_m + \sum_{i=1}^{L_m} c_i^{(m)} s_{m-i}$$

and because $L_{i+1} = \max\{L_i,\, i+1-L_i\}$ for $i < k$ we have

$$L_k = L_{m+1} = m + 1 - L_m. \tag{3.17}$$

Define

$$p_{k+1}(x) = p_k(x) - d_k \cdot d_m^{-1} \cdot x^{k-m} \cdot p_m(x).$$

The degree of this polynomial is at most

$$\max\{L_k,\ k - m + L_m\} = \max\{L_k,\ k + 1 - L_k\},$$

because by (3.17) we have $L_m = m + 1 - L_k$. To see that this polynomial is a valid choice notice that

$$s_j + \sum_{i=1}^{L_k} c_i^{(k)} \cdot s_{j-i} - d_k \cdot d_m^{-1} \cdot \left[s_{j-k+m} + \sum_{i=1}^{L_m} c_i^{(m)} \cdot s_{j-k+m-i} \right] = 0 \quad \text{for } L_{k+1} \leq j \leq k.$$

This equation can be checked as follows:
for $L_{k+1} \leq j \leq k-1$ we have $L_m - i = L_{k+1} - k + m - i \leq j - k + m - i < m - i$ and with $L_k \leq L_{k+1}$ we get

$$\sum_{i=1}^{L_k} c_i^{(k)} \cdot s_{j-i} = -s_j \qquad \text{and} \qquad \sum_{i=1}^{L_m} c_i^{(m)} \cdot s_{j-k+m-i} = -s_{j-k+m}.$$

For $j = k$ we have

$$s_k + \sum_{i=1}^{L_k} c_i^{(k)} \cdot s_{k-i} = d_k \qquad \text{and} \qquad s_{k-k+m} + \sum_{i=1}^{L_m} c_i^{(m)} \cdot s_{k-k+m-i} = d_m.$$

Thus we see that $p_{k+1}(x)$ is a valid choice. Using this inductive formulas we can compute the linear reccurence $(s_j)_{0 \leq j < k}$ for $k > 2 \cdot n$.

3.10. Example *Assume that we have the following sequence of numbers:*

$$s_0 = 1,\ s_1 = 2,\ s_2 = 3,\ s_3 = 6,\ s_4 = 11,\ s_5 = 20,\ s_6 = 37.$$

We want to compute the generating polynomial of this sequence. For doing this we start the algorithm with $k = 0$ and $L_0 = 0$. Then we get

$$d_0 = s_0 + \sum_{i=1}^{L_0} c_i^{(0)} \cdot s_{0-i} = s_0 = 1$$

and with $m = -1$ we compute the generating polynomial

$$p_1(x) \ = \ p_0(x) - d_0 \cdot d_{-1}^{-1} \cdot x^{0-(-1)} \cdot p_{-1}(x) = 1 - x.$$

Thus we get $L_1 = 1$.

In the same way we compute d_1 *as*

$$d_1 = s_1 + \sum_{i=1}^{L_1} c_i^{(1)} \cdot s_{1-i} = 2 - 1 = 1$$

and with $m = 0$ *we achieve*

$$p_2(x) \;=\; p_1(x) - d_1 \cdot d_0^{-1} \cdot x^{1-0} \cdot p_0(x) = 1 - 2x.$$

Using the inductive formulas for p_{k+1} *we can compute the following generating polynomials:*

- $k = 2, m = 0 :\quad d_2 = -1,\ p_3(x) = 1 - 2x + x^2.$
- $k = 3, m = 2 :\quad d_3 = 2,\ p_4(x) = 1 - 3x^2.$
- $k = 4, m = 2 :\quad d_4 = 2,\ p_5(x) = 1 - x^2 - 4x^3.$
- $k = 5, m = 4 :\quad d_5 = 2,\ p_6(x) = 1 - x - x^2 - x^3.$
- $k = 6, m = 4 :\quad d_6 = 0,\ p_7(x) = p_6(x).$

Thus we have computed the following minimal linear recurrence of the sequence (s_j) *for* $0 \le j \le 6$*:*

$$s_{j+3} \;=\; s_j + s_{j+1} + s_{j+2} \quad \textit{for } j = 0, \ldots, 3.$$

Now we will prove that for $k \ge 2n$ the construction must reproduce the same recurrence each time we increase k.

If $k \ge 2n$, then $k + 1 - L_k \ge n + 1$, because $L_k \le L_{k+1} \le k + 1 - L_k$ and therefore $L_k \le \frac{k+1}{2}$. Hence the construction must yield a recursion of the same length for all greater k.

If $c_1, \ldots, c_{L_k},\ c'_1, \ldots, c'_{L_k}$ are the coefficients of two different recursions which generate $(s_j)_{j \le k-1}$, then we consider $c''_i = c_i - c'_i$ $(1 \le i \le L_k)$. We let $c''_{L_k - j}$ the first of those coefficients which is non zero. Define

$$d_i = c''^{-1}_{L_k-j} \cdot c''_{L_k-j+i} \quad \text{for } 0 \le i \le j.$$

Then we have for an appropriate l which will be defined later

$$\begin{aligned}
\sum_{i=0}^{j} d_i \cdot s_{l-i} \;&=\; c''^{-1}_{L_k-j} \cdot \sum_{i=0}^{j} (c_{L_k-j+i} - c'_{L_k-j+i}) \cdot s_{l-i} \\
&=\; c''^{-1}_{L_k-j} \sum_{i=L_k-j}^{L_k} (c_i - c'_i) \cdot s_{L_k-j+l-i} \\
&=\; c''^{-1}_{L_k-j} \sum_{i=0}^{L_k} (c_i - c'_i) \cdot s_{L_k-j+l-i} \\
&=\; 0
\end{aligned}$$

for $L_k - j + l \leq k - 1$, i.e. $l \leq k - 1 - L_k + j = k^* - 1$. We will explain the last steps: the first equation follows just from the definition of the d_i and the c_i''. Then we have changed the running index by $L_k - j$. The next equality follows by the fact that $c_i'' = 0$ for $i = 1, \ldots, L_k - j - 1$.

Hence we have $L_{k^*} \leq j$. But on the other hand

$$L_{k^*} = \max\{L_{k^*-1}, k^* - L_{k^*-1}\} \geq k^* - L_{k^*-1} = k - L_k + j - L_{k^*-1}.$$

For proving this note that

$$L_{k^*-1} \leq L_{k^*} \leq j < L_k \leq n.$$

So we get $L_{k^*} > j$ for $k \geq 2n$. This contradiction shows that the recursion which generates $(s_j)_{j \leq k-1}$ is uniquely determined.

Thus we have shown how to compute the generating equation p_∞ for a sequence $(s_j) = (\underline{u}, A^j \cdot \underline{b})$ for a random element $\underline{u} \in \mathbb{F}^n$.

3.5. Exercise *Show that p_∞ divides f.*

One way of determining f is to compute p_∞ for randomly chosen vector $\underline{u}$ and to hope that $p_\infty = f$. We want to estimate the probability for a random $\underline{u}$ that $p_\infty = f$. We know that

$$\begin{array}{ccc} S = span\{A^i \cdot b \mid i \in \mathbb{Z}_{\geq 0}\} & \to & \mathbb{F}[x]\big/f(x)\mathbb{F}[x] =: R \\ A^i \cdot b & \longmapsto & \rho^i \end{array}$$

with $\rho = x + f(x)\mathbb{F}[x]$ is an isomorphism. If we moreover define

$$\eta(1)\left(\sum_{i=1}^{d} \lambda_i \cdot \rho^{i-1}\right) = \lambda_d$$

and for $\alpha, \beta \in R$

$$\eta(\alpha)(\beta) = \eta(1)(\alpha \cdot \beta),$$

then the map

$$\begin{array}{ccc} R & \to & \mathrm{Hom}_{\mathbb{F}}(R, \mathbb{F}) = R^* \\ \alpha & \longmapsto & \eta(\alpha)(\cdot) \end{array}$$

is an isomorphism. Hence, in order to count the number of linear recurrences $(\underline{u}, A^j \cdot \underline{b})_{j \geq 0}$ with minimal polynomial f we may, as well, count the number of linear recurrences $(\eta(\alpha)(\rho^j))_{j \geq 0}$ with minimal polynomial f. Let h be the minimal polynomial of this recurrence, then

$$\begin{aligned} 0 &= h(\eta(\alpha)(\rho^j)) = \eta(\alpha)(h(\rho^j)) \\ &= \eta(1)(\alpha \cdot h(\rho^j)) \qquad \text{for } j \geq 0, \end{aligned}$$

which means that $\alpha \cdot h(\rho^j) = 0$ and, if α is a unit in R, then $h(\rho^j) = 0$. But this means that $f|h$, hence $f = h$.

One can show that the fraction of units in R is at least $\frac{1}{6 \cdot j}$, where j is determined as the least exponent such that

$$deg(f) \leq 1 + q + q^2 + \ldots + q^j.$$

where $q = \#\mathbb{F}$. Thus we have $j \approx \log(n)$ and the probability for finding a vector $\underline{u}$ which determines the minimal polynomial f is approximately $\frac{1}{\log(n)}$.

We return to our method for solving the equation and analyse its complexity.

We pick $\underline{u} \in \mathbb{F}^n$ at random, we determine the first $2n + 1$ elements of the linear recurrence $(\underline{u},\ A^i \cdot \underline{b})$ which requires $O(n \cdot (w + n))$ operations. Note that the vector u can be of the form $(0, \ldots, 0, 1, 0, \ldots, 0)$. The construction of the minimal polynomial using the Berlekamp-Massey algorithm requires $O(n^2)$ operations. In order to determine the characteristic polynomial of A one must choose several $\underline{u}$ and compute the lcm of the resulting polynomials. Hence we have

3.13. Proposition *Let $A \in \mathbb{F}^{n \times n}$ be non singular with w non zero entries and let $\underline{b} \in \mathbb{F}^n$. There is a probabilistic algorithm which calculates a solution of $A \cdot \underline{x} = \underline{b}$ in expected time $O(n(w + n) \cdot \log n)$.*

Now we assume that A is non-square and singular. Then using a probabilistic method it is possible to quickly find a matrix Q such that $A \cdot Q$ or $Q \cdot A$ is non singular. Then we either solve $(A \cdot Q) \cdot \underline{x} = \underline{b}$ (and then we have the solution $Q \cdot x$) or $(Q \cdot A) \cdot \underline{x} = Q \cdot \underline{b}$.

3.14. Proposition *Let $A \in \mathbb{F}^{m \times n}$, $\underline{b} \in \mathbb{F}^m$. There is an probabilistic algorithm which decides the solvability of $A \cdot \underline{x} = \underline{b}$ using $O(n_1 \cdot (w + n_1) \cdot \log^k n_1)$ operations where $n_1 = \max\{n, m\}$, w is the number of non zero entries of A and k is a constant.*

3.3 The Monte Carlo factoring algorithm

Again we want to factor n. We generate a sequence $(x_i)_{i \geq 0}$ in $\{0, \ldots, n-1\}$. Let p be a proper divisor of n. Of course, there must exist $i, j \in \mathbb{Z}_{\geq 0}$ with

$$x_i \equiv x_j \bmod p. \tag{3.18}$$

If $x_i \not\equiv x_j \bmod n$ (i.e. $x_i \neq x_j$) then $d = \gcd(x_i - x_j, n)$ is a proper divisor of n. It can be shown that a sequence of random numbers in $\{0, \ldots, n-1\}$ has a good chance of producing a factor of n. The problem of turning this into a practical algorithm is that one would have to store all the sequence elements in order to check whether (3.18) holds. We therefore produce the sequence in a systematic way. For this purpose we consider a map

$$F: \{0, \ldots, n-1\} \to \{0, \ldots, n-1\}.$$

We pick a starting value x_0 and we define

$$x_{i+1} = F(x_i).$$

3.15. Proposition *$(x_i)_{i\geq 0}$ is of the periodic form*

$$(x_i)_{i\geq 0} = x_0,\,\ldots,\,x_{m-1},\,x_m,\,\ldots,\,x_{m+k-1},\,x_m,\,\ldots,\,x_{m+k-1}\cdots$$

Proof: Since the sequence is infinite but $\{0,\ldots,n-1\}$ is finite, there must be $m \geq 0$, $k \geq 1$ such that $x_m = x_{m+k}$. But then we have $x_{m+k+1} = F(x_{m+k}) = F(x_m) = x_{m+1}$. □

If in Proposition 3.15 the numbers m and k are chosen minimal, we call m the **preperiod length**, k the **period length**, $\{x_0,\,\ldots,\,x_m\}$ the **preperiod** and $\{x_m,\,\ldots,\,x_{m+k-1}\}$ the **period** of $(x_i)_{i\geq 0}$.

3.16. Proposition *There is an $i \in \mathbb{Z}$, $1 \leq i < m+k$ such that $x_{2i} = x_i$.*

Proof: Let $i = l\cdot k$ be the smallest multiple of k greater than m. Then $i < m+k$ and $x_{2i} = x_{2lk} = x_{lk+lk} = x_{lk} = x_i$. □

Based on the previous statement, one can try to factor n by calculating the sequence of pairs $((x_i,\,y_i))_{i\geq 1}$ with $y_i = x_{2i}$. Note that the calculation of $(x_{i+1},\,y_{i+1})$ from $(x_i,\,y_i)$ requires three applications of F.

The function F should use sequences, which are sufficiently random. Experimentally, the function

$$F(s) = (s^2+c)_n$$

for fixed $c \in \mathbb{Z}$ (where $(\cdot)_n$ means the smallest non negative residue modulo n) has turned out to be a good choice.

3.11. Example *Factor $n = 91643$ with $x_0 = 3$, $F(x) = (x^2-1)_n$.*

$$\begin{array}{lll}
x_1 = 8, & y_1 = x_2 = 63, & \gcd(63-8,n) = 1,\\
x_2 = 63, & y_2 = x_4 = 74070, & \gcd(74070-63,n) = 1,\\
x_3 = 3968, & y_3 = x_6 = 35193, & \gcd(35193-3968,n) = 1,\\
x_4 = 74070, & y_4 = x_8 = 45368, & \gcd(74070-45368,n) = 113.
\end{array}$$

Note that the execution of this algorithm requires a lot of gcd-computations, which are expensive. In order to avoid some of them, we can accumulate the product

$$Q_i = \left(\prod_{j=1}^{i}(y_j - x_j)\right)_n$$

and apply Euclid's algorithm only occasionally, say if i is a multiple of 100.

Example 3.11 also shows that all the x_l with $l \leq i$ and $2\,|\,l$ are computed twice if $y_i - x_i$ yields the factorization. The following statement shows how to avoid this problem

3.17. Proposition *There is an $i \in \mathbb{Z}_{\geq 0}$, $j \in \mathbb{Z}_{\geq 1}$ such that $x_{2^i} = x_{2^i+j}$ with $2^{i-1} < j \leq 2^i$, $2^i < 2\max\{m,\,k\}$.*

Proof: Let i be chosen such that $2^i \geq \max\{m, k\} > 2^{i-1}$. Let $l \leq i$ be chosen such that $2^{i-1} < 2^l k \leq 2^i$. Then set $j = 2^l k$ and verify the result. $\square$

We now compute the sequence $(x_i)_{i \geq 0}$. If we have calculated x_{2^l}, then we store this value and we calculate x_{2^l+j}, $j = 1, 2, \ldots, 2^l$. We also initialize $Q = 1$ and we multiply Q by $(x_{2^l} - x_{2^l+j})$, $2^{l-1} < j \leq 2^l$.

3.12. Example *The sequence and the factors of Q are computed as follows*

x_1, x_2	$x_1 - x_2$
x_3, x_4	$x_2 - x_4$
x_5, x_6, x_7	$x_4 - x_7$
x_8	$x_4 - x_8$
$x_9, x_{10}, x_{11}, x_{12}, x_{13}$	$x_8 - x_{13}$
x_{14}	$x_8 - x_{14}$
x_{15}	$x_8 - x_{15}$
x_{16}	$x_8 - x_{16}$
etc.	

3.6. Exercise *Let $d \in \mathbb{Z}_{\geq 1}, M = \{0, \ldots, d-1\}$ and $F: M \to M$, let $x_0 \in M$ and $x_{i+1} = F(x_i)$. Assume that $(x_i)_{i \geq 1}$ has preperiod length 0. Estimate the number of factors produced in both versions of the Monte Carlo algorithm before two equal sequence elements are found.*

3.7. Exercise *Find an example for which the Monte Carlo factoring algorithm fails.*

We see from Example 3.12 that if x_i is the last sequence element calculated before n is factored then roughly $i/2$ factors of Q have been computed; the same is true for the first variant of the method. Also, if $m \leq k$, then the number of sequence elements computed is roughly the same. The second version has the advantage that none of the sequence elements has to be recomputed.

3.4 The Pollard $p - 1$ method

Again we want to factor n. If p is a prime factor of n, then we have

$$a^{p-1} - 1 \equiv 0 \bmod p$$

for every $a \in \mathbb{Z}$ with $\gcd(a, p) = 1$. And even for exponents $e = k \cdot (p-1)$, $k \in \mathbb{Z}_{\geq 1}$, we have

$$a^e - 1 \equiv 0 \bmod p.$$

On the other hand, if $\gcd(a, n) = 1$ and if the order of a modulo n is not a divisor of the exponent e, then

$$a^e - 1 \not\equiv 0 \bmod n,$$

which means that

$$d = \gcd(a^e - 1,\, n)$$

is a proper divisor of n. Of course, we do not know p. But if $p-1$ factors into a power product of small prime numbers then a multiple of $p-1$ can easily be written down.

3.13. Example *Let* $n = 247,\ e = 2^2 \cdot 3,\ a = 2.$

$$\begin{aligned} a^e = a^{12} &\equiv 144 \bmod n \\ \gcd(144-1,\, n) &= 13. \end{aligned}$$

How to choose the exponent e? If we want to catch all numbers below B_2, which are composed of prime numbers below B_1 we have to determine for any such prime p the minimal exponent e_p such that $p^{e_p} \geq B_2$ and then $e = \prod\limits_{p \leq B_1} p^{e_p}$.

Of course, e is not computed explicitely but we rather store a list of prime numbers $p \leq B_1$ as $p_1, p_2, \ldots$ with $p_{j+1} \geq p_j$ and the multiplicity e_p for each p. Then we set

$$b_1 = a,\ b_{i+1} = (b_i^{p_i^{e_{p_i}}})_n,\ Q_{i+1} = \left(\prod_{j=1}^{i+1}(b_j - 1)\right)_n \equiv (b_{i+1} - 1) \cdot Q_i \bmod n.$$

Again, we calculate $\gcd(Q_i,\, n)$ only occasionally, e.g. if i is a multiple of 100.

3.5 The elliptic curve method (ECM)

Reference: [Le87]

Let us analyse Pollard's $(p-1)$-method a little bit:

We are working in the group $G_n = (\mathbb{Z}/n\mathbb{Z})^*$. For each prime divisor p we can define a homomorphism

$$\phi\colon G_n \longmapsto \mathbb{F}_p^* = G_p$$

(reduction modulo p).

If we find a non zero element of the kernel of ϕ, we find a factor of n. Such an element is found if we find an element $g \in G_n$ such that the order of $\phi(g)$ is composed of small prime numbers only, whereas this is false for the order of g. This can only happen if the order of the prime field $\mathbb{F}_p$ for at least one of the prime divisors p is of the form

3.14. Example *(Koblitz)*
Let $n = 491389$. *We would like to find a non trivial divisor. Now* $n = 383 \cdot 1283,\ 383 - 1 = 2 \cdot 191,\ 1283 - 1 = 2 \cdot 641$. *Both those finite fields have orders, which do not decompose into small prime factors. We are therefore stuck with the* $(p-1)$*-method because we have no other choice for* G_p.

It was Lenstra's idea to replace G_n by the group of points of an elliptic curve over $\mathbb{Z}/n\mathbb{Z}$.
Assume that $6 \not| n$ (a trivial assumption, in our context).
For $d \,|\, n$ consider a polynomial congruence of the form

$$y^2 z \equiv x^3 + axz^2 + bz^3 \bmod d \tag{3.19}$$

with $a,\, b \in \mathbb{Z}$. Assume that for

$$\Delta = 4a^3 + 27b^2$$

we have $\gcd(\Delta,\, n) = 1$.

Consider the set

$$\tilde{E}_d = \{(\overline{x},\, \overline{y},\, \overline{z}) \mid (x,\, y,\, z) \text{ satisfy } (3.19)\}$$

We define an equivalence relation on $\tilde{E}_d$, which identifies the solutions in $\tilde{E}_d$, which are trivially equal:

$$(\overline{x},\, \overline{y},\, \overline{z}) \sim (\overline{x'},\, \overline{y'},\, \overline{z'}) \iff (\overline{x},\, \overline{y},\, \overline{z}) = \overline{u} \cdot (\overline{x'},\, \overline{y'},\, \overline{z'}) \;\text{ with } \overline{u} \in (\mathbb{Z}/d\mathbb{Z})^*$$

Denote by E_d the set of equivalence classes and the elements of E_d by $(\overline{x} : \overline{y} : \overline{z})$.

3.18. Proposition *E_d is an abelian group (usually additively written) with zero element $(\overline{0} : \overline{1} : \overline{0})$. The canonical mapping*

$$E_n \to E_d$$

is a homomorphism.

If we manage to find a point $(\overline{x} : \overline{y} : \overline{z})$ on an elliptic curve defined over $\mathbb{Z}/n\mathbb{Z}$, which is non-zero but whose image in E_d is zero for some proper divisor d of n, then $\gcd(n,\, z)$ is a proper divisor of n.

In order to factorize n by this method, we must first choose an elliptic curve E defined over $\mathbb{Z}/n\mathbb{Z}$ and a point on this curve. Although it is easy to write down such an elliptic curve, it is much harder to find a point on a given curve. One therefore starts by choosing the point: Choose $x,\, y \in \{0,\, \ldots,\, n-1\}$. The point will be $(\overline{x} : \overline{y} : \overline{1})$. Choose also $a \in \{0,\, \ldots,\, n-1\}$.

In order for $(\overline{x} : \overline{y} : \overline{1})$ to be a point on the curve, we must have

$$y^2 \equiv x^3 + ax + b \bmod n, \tag{3.20}$$

i.e. we can choose

$$b = (y^2 - x^3 - ax)_n. \tag{3.21}$$

Then we only must check whether $\gcd(\Delta,\, n) = 1$ where $\Delta = 4a^3 + 27b^2$. If we have $1 < \gcd(\Delta,\, n) < n$, then a proper divisor of n is found. Only if $\gcd(\Delta,\, n) = n$, then we must choose a new triple $(x,\, y,\, a)$. Note that this event is very unlikely.

Denote the curve by E_n and the point by P_n. Correspondingly, for $d \mid n$ write E_d and P_d for the curve and the point considered over $\mathbb{Z}/d\mathbb{Z}$. If there is a divisor d of n, for which P_d has an order, which is a power product of small primes only and whose order is lower than the order of P_n then - as in the Pollard $(p-1)$-method - we have a chance to factor n.

It only remains to explain the addition of points on E_n. The general method is quite complicated and not suited for an efficient computer implementation. We rather use a restricted method, which enables us to treat points of the form $P_1 = (\overline{x}_1 : \overline{y_1} : \overline{1})$, $P_2 = (\overline{x}_2 : \overline{y}_2 : \overline{1})$. If $P_1 + P_2$ is of the form $P = (\overline{x}_3 : \overline{y}_3 : \overline{1})$, then the algorithm will yield P_3. If P_3 is not of this form, then $P_3 = (\overline{x}_3 : \overline{y}_3 : \overline{z}_3)$ where $d = \gcd(z_3, n) > 1$. In this case the algorithm will yield a divisor of n, which in case $d < n$ will be a non trivial one. Here is the method:

3.1. Algorithm
Input: $P_i = (\overline{x}_i : \overline{y}_i : \overline{1})$ $(1 \leq i \leq 2)$
Output: $P_3 = P_1 + P_2 = (\overline{x}_3 : \overline{y}_3 : \overline{1})$ *or if* $P_1 + P_2$ *is not of this form a divisor* $d > 1$ *of* n.

1. *If* $x_1 \not\equiv x_2$ *modulo* n*, then try to calculate the inverse* k *of* $\gcd(x_2 - x_1)$ *modulo* n*. If this calculation fails, then* $\gcd(x_2 - x_1, n) = d > 1$*. Terminate. Else: Set*

$$\begin{aligned} m &= ((y_2 - y_1) \cdot k)_n, \\ x_3 &= (m^2 - x_1 - x_2)_n, \\ y_3 &= (m \cdot (x_1 - x_3) - y_1)_n. \end{aligned}$$

2. *If* $x_1 \equiv x_2$ *modulo* n*,* $y_1 \equiv -y_2$ *modulo* n *then* $P_1 = -P_2$*, so* $P_3 = (\overline{0} : \overline{1} : \overline{0})$.

3. *If* $x_1 \equiv x_2$ *modulo* n*,* $y_1 \equiv y_2$ *modulo* n *(so* $P_1 = P_2$*) then try to calculate the inverse* k *of* $2y_1$ *mod* n*. If this calculation fails, then* $\gcd(2y_1, n) > 1$*. Terminate. Else: Set*

$$\begin{aligned} m &= ((3x_1^2 + a) \cdot k), \\ x_3 &= (m^2 - x_1 - x_2)_n, \\ y_3 &= (m(x_1 - x_3) - y_1)_n. \end{aligned}$$

3.15. Example *(Koblitz)*
Let $n = 5429$*. Consider the congruence*

$$y^2 z \equiv x^3 + axz - az^3 \bmod n. \tag{3.22}$$

Here $\Delta = 4a^3 + 27a^2$ *and if* $\gcd(\Delta, n) = 1$*, then this congruence defines an elliptic curve over* $\mathbb{Z}/n\mathbb{Z}$*. This elliptic curve contains the point* $(\overline{1} : \overline{1} : \overline{1})$*. Let* $e = 2^6 3^4$ *be the multiple of possible orders of points we want to use.*

If we choose $a = 1$*, then the method fails to yield a factor of* n*. Let* $a = 2$*. Then* $\Delta = 140$*. We have* $\gcd(\Delta, n) = 1$*. The computation is successful if we compute* $2^6 \cdot 3^2 \cdot P_1$*.*

Note that the method is extremely easy to implement. Its major advantage over the $(p-1)$-algorithm is the variety of possible groups. The method has been used successfully for large numbers whose prime factors are not too large. If the prime factors are close to $\sqrt{n}$, then the quadratic sieve is the better choice.

In practical implementations of this algorithms often a second stage is used. We will now explain this improvement.

Choose a third bound $B_3 > B_1$ and let $m = \prod_{p \le B_1} p^{e_p}$. After the first stage we have computed the point $m \cdot P$. We store a table of the difference of primes between B_1 and B_3. Now these differences are small, and there will not be too many of them. So we can quickly compute the points $d \cdot P$ for all possible differences d and store them. Let q be the first prime greater than B_1. Start with computing $q \cdot m \cdot P$. Then for all prime numbers q' between B_1 and B_3 the point $q' \cdot m \cdot P$ can be computed by simply adding the point $d \cdot P$ to $q'' \cdot m \cdot P$ where d is the difference between the least prime q'' before q' and q'. Thus we are "successful" if the order of our elliptic curve E_d is of the form $\#E_d = l \cdot q'$, where l is a divisor of m and q' is a prime between B_1 and B_3.

3.6 Factoring with quadratic forms

3.6.1 Quadratic forms and class groups

A polynomial $ax^2 + bxy + cy^2 \in \mathbb{Z}[x, y]$ is called a **binary quadratic form** and $\Delta = b^2 - 4ac$ is called its **discriminant**. We denote a binary quadratic form $ax^2 + bxy + cy^2$ by (a, b, c). It is called **primitive** if $\gcd(a, b, c) = 1$. A form, for which $a > 0$ and $\Delta < 0$ is called **positive**. Two forms (a, b, c) and (a', b', c') are called **equivalent** if there exist $\alpha, \beta, \gamma, \delta \in \mathbb{Z}$ with $\alpha\delta - \beta\gamma = 1$ such that $a'u^2 + b'uv + c'v^2 = ax^2 + bxy + cy^2$, where $u = \alpha x + \gamma y$, $v = \beta x + \delta y$. We write $(a, b, c) \sim (a', b', c')$. Equivalent forms have the same discriminant.

3.8. Exercise *Show that $\sim$ is in fact an equivalence relation and that equivalent forms have the same discriminant.*

Now let $\Delta \in \mathbb{Z}_{\le -1}$, $\Delta \equiv 0, 1 \bmod 4$. We consider the forms of discriminant Δ. Since $c = (b^2 - \Delta)/(4a)$, it is sufficient to write (a, b) for the quadratic form (a, b, c). We study the set C_Δ of equivalence classes of primitive forms of discriminant Δ. First we develop a convenient way to represent the elements of C_Δ: The form (a, b, c) is called **reduced** if $|b| \le a \le c$, and $b \ge 0$ if $|b| = a$ or $a = c$.

3.19. Proposition *Each class in C_Δ contains precisely one reduced form.*

Proof: We first present the reduction algorithm.

3.2. Algorithm *(Reduction of positive binary quadratic forms)*
Input: The form (a, b).
Output: The reduced form (a, b).

(1) Replace (a, b) by $(a, b-2k\cdot a)$, where $k \in \mathbb{Z}$ is such that $-a < b-2k\cdot a \le a$.

(2) If (a, b, c) is reduced, then stop; otherwise replace (a, b, c) by $(c, -b, a)$ and go back to step (1).

After step 1 we have $|b| \le a$ and $b \ge 0$ if $|b| = a$. If (a, b, c) is not reduced after this step then either $a > c$ or $b < 0$ and $a = c$. In the latter case (a, b, c) will be reduced after step 2. In the first case a is decreased by at least 1, which shows the finiteness of the algorithm.

The proof of uniqueness is left as an exercise. □

On C_Δ we define a group structure: We identify the classes with their reduced representatives. To compute $(a_1, b_1)\cdot(a_2, b_2)$ we use the Euclidean algorithm to determine

$$d = \gcd\left(a_1,\ a_2,\ \frac{b_1+b_2}{2}\right)$$

and $r, s, t \in \mathbb{Z}$ such that

$$d = r\cdot a_1 + s\cdot a_2 + t\cdot\frac{b_1+b_2}{2}.$$

The product is obtained from an application of the reduction algorithm to

$$\left(\frac{a_1a_2}{d^2},\ b_2 + \frac{2a_2(s\frac{b_1-b_2}{2} - tc_2)}{d}\right)$$

where $c_2 = (b_2^2 - \Delta)/(4a_2)$. The identity element is

$$1_\Delta = \begin{cases} (1,\ 1) & \text{if } \Delta \text{ is odd} \\ (1,\ 0) & \text{if } \Delta \text{ is even} \end{cases}$$

The inverse of (a, b) is $(a, -b)$.

3.20. Proposition *C_Δ is a finite abelian group.*

A reduced form is called **ambiguous** if $b = 0$ or $a = b$ or $a = c$. If we know ambiguous forms, then we can obtain factorizations of Δ:

If $(a, b) = (a, 0)$	then $\Delta = b^2 - 4ac = -4ac$
If $(a, b) = (a, a)$	then $\Delta = a^2 - 4ac = a(a-4c)$
If $(a, b, c) = (a, b, a)$	then $\Delta = b^2 - 4a^2 = (b-2a)(b+2a)$

Notice that because of the primitivity of the forms the factors must be coprime. Conversely, let $\Delta \equiv 1 \bmod 4$ and let $|\Delta| = p\cdot q$ be a factorization into the coprime factors. We want to obtain an ambiguous form, which corresponds to this factorization. For this purpose we either have to solve the equation

$$-p\cdot q = \Delta = a(a-4c) \tag{3.23}$$

or

$$- p \cdot q = \Delta = (b - 2a)(b + 2a). \tag{3.24}$$

The first type of ambiguous forms is useless in this context because the corresponding discriminant is even.

Since $-p \cdot q \equiv 1 \bmod 4$ we must have $-p \equiv 3 \bmod 4$ and $q \equiv 3 \bmod 4$ or $-p \equiv 1 \bmod 4$ and $q \equiv 1 \bmod 4$, in any case $p + q \equiv 0 \bmod 4$. To satisfy (3.23) we may put $a = p = b,\ c = (p^2 + p \cdot q)/4p = (p + q)/4$.

Let us check when this form is reduced. It must satisfy $|b| \leq a \leq c$, i.e. $p \leq p \leq (p+q)/4$. The first inequality is trivially satisfied. In order for the second inequality to hold, we need $3p \leq q$. Finally, since $|b| = a$ we must have $b \geq 0$, which is also true.

Now we examine ambiguous forms of the third type:

We may choose $a = (p + q)/4,\ b = (q - p)/2$. Again we examine when this form is reduced. In $|b| \leq a \leq c$ the second inequaltiy is trivial. The first one is $(q - p)/2 \leq (p+q)/4$ (we assume without loss of generality that $q > p$). So $2 \cdot q - 2 \cdot p \leq p + q,\ q \leq 3 \cdot p$. This covers the rest of the cases and we have proved

3.21. Proposition *For $\Delta < 0, \Delta \equiv 1 \bmod 4$ there is a 1-1-correspondence between the ambiguous forms and the factorizations of Δ into coprime factors:*
If $-\Delta = p \cdot q,\ p < q$, then the correspondence is given by

$$p \cdot q \longleftrightarrow \begin{cases} (p,\, p) & \text{for } q > 3p \\ (\frac{p+q}{4},\, \frac{q-p}{2}) & \text{for } q < 3p \end{cases}$$

3.22. Proposition *A class in C_Δ containing an ambiguous form has order either 1 or 2 in C_Δ.*

Proof: Consider the form $(a,\, a)$. Apply the composition algorithm:
$d = \gcd(a,\, a,\, a) = a,\ r = 1,\ s = t = 0$. Result $(1,\, 0)$. Other case exercise. □

So the factorization of Δ can be obtained from the elements of order 2 in C_Δ. We describe the probabilistic algorithm from Seysen ([Sey87]) to determine those elements. A system G of primitive forms is called a **generating system** for C_Δ if every element in C_Δ is a power product of the classes of the forms in G.

Let us describe the idea calculating all the ambiguous forms.

The elements of order ≤ 2 form a subgroup $C_{2,\Delta}$ of C_Δ. If $\{f_1,\, \ldots,\, f_r\}$ is a generating system for C_Δ, then a generating system for the subgroup of all elements of 2-power order can be obtained as follows:
Determine a multiple $\tilde{h}$ of the **classnumber** $h_\Delta = |C_\Delta|$. Let h' be the odd part of $\tilde{h}$. Compute $f_i' = f_i^{h'}$ for $1 \leq i \leq r$. Then the f_i' generate the subgroup of all elements of 2-power order in C_Δ.

Now determine for $1 \leq i \leq r$ (if exists) an exponent $m_i \in \mathbb{Z}_{\geq 1}$ such that

$$\begin{aligned} f_i'^{2^{m_i}} &= 1 \\ f_i'^{2^{m_i - 1}} &\neq 1 \end{aligned}$$

(otherwise set $m_i = 0$) and set $\tilde{f}_i = f_i'^{2^{m_i - 1}}$. Then $\{\tilde{f}_1, \ldots, \tilde{f}_r\}$ is a generating system for $C_{2,\Delta}$. So

$$C_{2,\Delta} = \{\tilde{f}_1^{x_1} \ldots \tilde{f}_r^{x_r} \mid x_i \in \{0, 1\}\}.$$

It remains to determine a method for computing a multiple for $h_\Delta = |C_\Delta|$. For this purpose we again let $\{f_1, \ldots, f_r\}$ be a system of generators for the class group C_Δ.

3.23. Proposition *(i) The map*

$$\begin{array}{ccc} \mathbb{Z}^r & \rightarrow & C_\Delta \\ (x_1, \ldots, x_r) & \longmapsto & \prod_{i=1}^r f_i^{x_i} \end{array} \tag{3.25}$$

is an epimorphism.
(ii) The kernel of this map is an r-dimensional lattice in $\mathbb{Z}^r$ of determinant h_Δ.

Proof:
(i) The map (3.25) clearly is a homomorphism. It is surjective because $\{f_1, \ldots, f_r\}$ is a generating system.
(ii) The second assertion will be proved later. □

3.6.2 Integer lattices

Now I briefly describe some fundamental facts about **integer lattices**.

An integer lattice is a subgroup of the (additive abelian) group $\mathbb{Z}^r$ for some $r \in \mathbb{Z}_{\geq 1}$.

3.24. Proposition *Let $L \subseteq \mathbb{Z}^r$ be an integer lattice. Then L is of the form*

$$\begin{aligned} L &= \mathbb{Z}\underline{v}_1 \oplus \ldots \oplus \mathbb{Z}\underline{v}_k \\ &= \left\{ \sum_{j=1}^k x_j \underline{v}_j \mid x_j \in \mathbb{Z},\ 1 \leq j \leq k \right\} \end{aligned}$$

where $\underline{v}_1, \ldots, \underline{v}_k \in \mathbb{Z}^r$ are linearly independent. $\{\underline{v}_1, \ldots, \underline{v}_k\}$ is called a **basis** *of L.*

Proof: For $1 \leq j \leq r$ let $\underline{v}_j$ be a vector in L of the form $\underline{v}_j = (v_1^{(j)}, \ldots, v_j^{(j)}, 0, \ldots, 0)^t$ with minimal $v_j^{(j)} > 0$. If such a vector does not exist, then set $\underline{v}_j = \underline{0}$. We claim that the non-zero $\underline{v}_j$ form a basis of L. In fact, let $\underline{v} \in L$. Then we can find inductively coefficients $\lambda_j \in \mathbb{Z}$ such that for $1 \leq j_0 \leq r$ we have

$$\underline{v} - \sum_{j=1}^r \lambda_j \underline{v}_j = \underline{w}_{j_0} = (w_1^{(j_0)}, \ldots, w_{j_0}^{(j_0)}, 0, \ldots, 0)$$

and this proves the assertion. So assume that we have found $\lambda_{j_0+1}, \ldots, \lambda_r$ for $j_0 \leq r$. If $v_{j_0}^{(j_0)} = 0$, then we also have $w_{j_0}^{(j_0)} = 0$. If $v_{j_0}^{(j_0)} \neq 0$, we divide with remainder: $w_{j_0}^{(j_0)} = \lambda_{j_0} v_{j_0}^{(j_0)} + z$ with $0 \leq z < v_{j_0}^{(j_0)}$. Here $z = 0$ because of the minimality of $v_{j_0}^{(j_0)}$. □

3.25. Proposition *Let $L \subseteq \mathbb{Z}^r$ be an integer lattice. Then all the bases of L have the same cardinality k, which is called the* **dimension** *of L. Let $\underline{v}_1, \ldots, \underline{v}_k$ be a basis of L and let $\underline{w}_1, \ldots, \underline{w}_k \in L$*

$$\underline{w}_j = \sum_{i=1}^{k} \lambda_{ij} \underline{v}_i$$

with $(\lambda_{ij}) \in \mathbb{Z}^{k \times k}$. Then $\{\underline{w}_1, \ldots, \underline{w}_k\}$ is a basis of L if and only if $(\lambda_{ij}) \in$ $\mathrm{GL}(n, \mathbb{Z})$, i.e. $|\det((\lambda_{ij}))| = 1$.

Proof: Exercise. $\square$

Let $L \subseteq \mathbb{Z}^r$ be an integer lattice. Then $\underline{v}_1, \underline{v}_2 \in \mathbb{Z}^r$ are called **congruent modulo** L if $\underline{v}_1 - \underline{v}_2 \in L$. This is an equivalence relation on $\mathbb{Z}^r$. The congruence class of $\underline{v} \in \mathbb{Z}^r$ is $\underline{v} + L$. The set $\{\underline{v} + L \mid \underline{v} \in \mathbb{Z}^r\}$ forms a group with respect to addition:

$$(\underline{v} + L) + (\underline{v}' + L) = (\underline{v} + \underline{v}') + L.$$

This group is the **factor group** $\mathbb{Z}^r/L$. A set $V \subseteq L$ such that $\{\underline{v} + L \mid \underline{v} \in V\} = \mathbb{Z}^r/L$ is called a **system of representatives** of $\mathbb{Z}^r/L$. If for every $\underline{v}_0 \in V$ the set $V \backslash \{\underline{v}_0\}$ is not a system of representatives, then V is called a **minimal system of representatives** of $\mathbb{Z}^r/L$.

The determinant of a k-dimensional integer lattice $L \subseteq \mathbb{Z}^r$ is defined as

$$d(L) = (|\det((\underline{v}_i, \underline{v}_j)_{1 \le i, j \le k})|)^{\frac{1}{2}}$$

where $\underline{v}_1, \ldots, \underline{v}_k$ is any basis of L and $(\underline{v}_i, \underline{v}_j)$ denotes the inner product of $\underline{v}_i$ and $\underline{v}_j$:

$$(\underline{v}_i, \underline{v}_j) = \sum_{l=1}^{k} v_{li} v_{lj}.$$

3.26. Proposition *If $L \subseteq \mathbb{Z}^r$ is a r-dimensional integer lattice, then*

$$d(L) = |\det(\underline{v}_1, \ldots, \underline{v}_r)|$$

for any basis $\underline{v}_1, \ldots, \underline{v}_r$ of L.

Proof: We have

$$d(L)^2 = |\det((\underline{v}_i \underline{v}_j))| = |\det[(\underline{v}_1, \ldots, \underline{v}_r)^t (\underline{v}_1, \ldots, \underline{v}_r)]| = |\det(\underline{v}_1, \ldots, \underline{v}_r)|^2.$$

$\square$

3.27. Proposition *Let $L \subseteq \mathbb{Z}^r$ be an integer lattice. Then $\mathbb{Z}^r/L$ is finite if and only if the dimension of L is r. In this case we have $d(L) = |\mathbb{Z}^r/L|$.*

Proof: Define $\underline{v}_1, \ldots, \underline{v}_r$ as in the proof of Proposition 3.24. Then

$$\{\sum_{i=1}^{r} x_i \underline{v}_i \mid 0 \leq x_i < v_i^{(i)} \text{ for } v_i^{(i)} \neq 0 \text{ and } x_i \in \mathbb{Z} \text{ for } v_i^{(i)} = 0\}$$

is a minimal system of representatives for $\mathbb{Z}^r/L$. This proves the assertion. □

Proof: (of Proposition 3.23 (ii)):
Let L be the kernel of (3.25). Then L is a subgroup of $\mathbb{Z}^r$, hence an integer lattice. By the homomorphism theorem we have $\mathbb{Z}^r/L \cong C_\Delta$. Since C_Δ is finite, the factor group $\mathbb{Z}^r/L$ is also finite and by Proposition 3.27 L is of dimension r. Moreover, we have by Proposition 3.27: $|C_\Delta| = h_\Delta = |\mathbb{Z}^r/L| = d(L)$. □

If $L' \subset L \subset \mathbb{Z}^r$ are integer lattices, then L' is called **sublattice** of L. Again, we can define the factor group L/L'.

3.28. Proposition *If $L' \subseteq \mathbb{Z}^r$ is a sublattice of the integer lattice $L \subseteq \mathbb{Z}^r$ of the same dimension as L, then $d(L') = c \cdot d(L)$ with $c \in \mathbb{Z}_{\geq 1}$.*

Proof: Exercise. □

The class number h_Δ is the determinant of the kernel L of (3.25), which is an integer lattice of dimension r. If we manage to find r linearly independent vectors $\underline{w}_1, \ldots, \underline{w}_r \in L$, then $L' = \oplus_{i=1}^r \mathbb{Z}\underline{w}_i$ is an r-dimensional sublattice of L, i.e. by Proposition 3.28 its determinant $d(L')$ is an integer multiple of $d(L) = h_\Delta$. Hence, in order to find an integer multiple of h_Δ it suffices to determine r linearly independent vectors in L. Those vectors will be generated by a probabilistic algorithm.

3.6.3 Prime forms

For a prime number p with $\left(\frac{\Delta}{p}\right) = 1$ there is $b_p \in \mathbb{Z}_{\geq 1}$ such that (p, b_p) is a form of discriminant Δ. To make b_p unique we choose $b_p = \min\{b \in \mathbb{Z}_{>0} \mid b^2 \equiv \Delta \bmod 4p\}$. The reduced form I_p equivalent to (p, b_p) is called **prime form**.

3.29. Proposition *Assume the GRH. Then the prime forms I_p with $p \leq 2(\log|\Delta|)^2$ generate the class group C_Δ.*

Note that the last proposition enables us to choose a generating system for C_Δ.

3.30. Proposition *Let (a, b) be a form of discriminant Δ and $\gcd(\Delta, a) = 1$. Let $a = \prod_{p \in \mathbb{P}} p^{e_p}$ be the factorization of a. Then we have:*
If $e_p \neq 0$ then $\left(\frac{\Delta}{p}\right) = 1$ and $(a, b) \sim \prod_{p\in\mathbb{P}} I_p^{s_p e_p}$ where $s_p \in \{-1, +1\}$ is determined by $b \equiv s_p \cdot b_p \bmod 2p$.

Proof: Since $\Delta = b^2 - 4ac$ we have $\Delta \equiv b^2 \bmod p$ for every $p \in \mathbb{P}$, which divides a, i.e. $\left(\frac{\Delta}{p}\right) = 1$ for every $p \in \mathbb{P}$, which divides a.

To prove the second assertion we first observe that $I_p^{s_p}$ is equivalent to a form $(p,\, b')$ with $b' \equiv b \bmod 2p$. Next we note that $(I_p^{s_p})^{e_p}$ is equivalent to a form $(p^{e_p},\, b'')$ with $b'' \equiv b \bmod 2p^{e_p}$. In fact, since $\gcd(\Delta,\, a) = 1$, it follows that $p \not| b_p = (b_p + b_p)/2$. Hence, the composition algorithm yields a form equivalent to $(p^{e_p},\, b')$ where $b'^2 \equiv \Delta \bmod 4p^{e_p}$. For odd p the group $(\mathbb{Z}/2p^{e_p}\mathbb{Z})^*$ is cyclic. Hence the congruence $x^2 \equiv \Delta \bmod 2p^{e_p}$ has at most two solutions. Since b is also such a solution and because $b \equiv b'' \bmod 2p$ it follows that $b \equiv b'' \bmod 2p^{e_p}$. A similar argument proves the assertion for $p = 2$. □

The idea of producing random relations in the class group is as follows. We fix $C_1,\, C_2 \in \mathbb{Z}_{>1}$ and let $p_1,\, \ldots,\, p_r$ all prime numbers below C_1 with $\left(\frac{\Delta}{p_i}\right) = 1$, ordered according to their size.

Choose $\underline{y} = (y_1,\, \ldots,\, y_r) \in \mathbb{Z}^r$ at random and calculate the reduced form $f = f(\underline{y}) = (a(\underline{y}),\, b(\underline{y}))$ in the class of $\prod_{i=1}^r I_{p_i}^{y_i}$.

Try to factor $a = a(\underline{y})$ over the factor base $\mathcal{F} = \{p_1,\, \ldots,\, p_r\}$. Suppose this factorization is successful:

$$a(\underline{y}) = \prod_{i=1}^r p_i^{e_i(\underline{y})}.$$

Determine $s_i(\underline{y}) \in \{\pm 1\}$ according to the congruence condition

$$b(\underline{y}) \equiv s_i(\underline{y}) b_{p_i} \bmod 2p_i \;\; (1 \le i \le r).$$

Then we know from Proposition 3.30 that $(a(\underline{y}),\, b(\underline{y}))$ is in the class of

$$\prod_{i=1}^r I_{p_i}^{s_i(\underline{y})e_i(\underline{y})}(\underline{y})$$

and by construction, this form is in the same class as

$$\prod_{i=1}^r I_{p_i}^{y_i}$$

Therefore $\underline{x} = (x_1,\, \ldots,\, x_r)$ with

$$x_i = y_i - s_i(\underline{y}) e_i(\underline{y}) \;\; (1 \le i \le r)$$

belongs to the kernel of the map (3.25).

After having produced sufficiently many random relations we will find r linearly independent vectors in the kernel and thus a multiple of the class number, which allows us to factor Δ.

Although this algorithm yields the complete factorization of n, it has a severe drawback: When calculating the determinant of the relation vectors very large numbers can occur. There is however a modification, which avoids this drawback:

Calculate $r+1$ relation vectors $\underline{x}_1,\, \ldots,\, \underline{x}_{r+1}$. Then there must be coefficients $c_i \in \{0,\, 1\}$ $(1 \le i \le r+1)$ such that

$$\sum_{i=1}^{r+1} c_i \underline{x}_i \equiv \underline{0} \bmod 2$$

Those coefficients can be determined by computing over $\mathbb{F}_2$ only. Once we know all the c_i, we determine the kernel element

$$\underline{x} = \sum_{i=1}^{r+1} c_i \underline{x}_i$$

and write

$$\underline{x} = 2\underline{x}' = 2(x'_1, \ldots, x'_r).$$

Then the reduced form in the class of $\prod_{i=1}^{r} f_i^{x'_i}$ has either order 1 or order 2 in the class group. This means that we have found an ambiguous form and hence a factor of n if we were lucky. If not, we repeat the whole procedure.

Chapter 4

Analysis of some factoring algorithms

We analyse the running time for factoring a composite number n. We set

$$L(n) = \exp\left(\sqrt{\log n \log\log n}\right).$$

When we say that **a probabilistic factoring algorithm has a running time of** $L(n)^{\alpha+o(1)}$, we mean that there is a function $\Theta(n)$ which tends to 0 as $n \to \infty$ through the composites such that for every $\epsilon > 0$ there is an $M > 1$ such that with probability at least $1-\epsilon$ the running time to factor n lies between $M^{-1} \cdot L(n)^{\alpha+\Theta(n)}$ and $M \cdot L(n)^{\alpha+\Theta(n)}$.

That is, on most implementations of the algorithm for n the running time will not be too many times greater or less than $L(n)^{\alpha+\Theta(n)}$. To emphasize this difference, when speaking of a probabilistic algorithm, we shall say it has an **expected running time of** $L(n)^{\alpha+o(1)}$.

An important notion is the notion of **smoothness**. We say that an integer is **smooth with respect to** y or y**-smooth**, if all its prime factors are at most y. In what follows we will often be interested in the probability that a random integer between 1 and x is smooth with respect to some y.

To derive an expression for this probability, we define $\psi(x, y)$ as the number of positive integers $n \leq x$ that are smooth with respect to y.

4.1. Proposition *([Bru66])*
Let $\epsilon > 0$ be arbitrary but fixed. Then we have that for $x \geq 10$ and $u \leq (\log x)^{1-\varepsilon}$

$$\psi(x, x^{1/u}) = x \cdot u^{-u+f(x,u)}$$

for a function f that satisfies $f(x, u)/u \to 0$ for $u \to \infty$ uniformly in x.

4.1. Corollary *For fixed $\alpha, \beta \in \mathbb{R}_{>0}$ we find that for $n \to \infty$*

$$\psi(n^\alpha, L(n)^\beta) = n^\alpha \cdot L(n)^{-\alpha/(2\beta)+o(1)}.$$

4.2. Corollary *A random positive integer* $\leq n^{\alpha}$ *is smooth with respect to* $L(n)^{\beta}$ *with probability* $L(n)^{-\alpha/(2\beta)+o(1)}$*, for* $n \to \infty$.

For $\beta \in \mathbb{R}$ we will often write $L_n[\beta]$ for $L(n)^{\beta}$ and we will abbreviate $L_n[\beta + o(1)]$ to $L_n[\beta]$ for $n \to \infty$. Notice that in this notation

$$L_n[\alpha] + L_n[\beta] = L_n[\max\{\alpha, \beta\}], \quad L_n[a] \cdot L_n[b] = L_n[a+b], \quad L_n[\alpha] \cdot \log^k n = L_n[\alpha]$$

and that the prime counting function π satisfies $\pi(L_n[\beta]) = L_n[\beta]$.

4.1 Analysis of Dixon's random square algorithm

We will start analysing a very simple variant of the quadratic sieve method, Dixon's random square algorithm.

Say we wish to factor n. If m is an integer, denote by $Q(m)$ the least non negative residue of $m^2 \bmod n$. Let α be a positive constant to be chosen later. We choose all prime numbers below $L_n[\alpha]$ as factor basis. Then we perform the following steps:

(1) Randomly choose a number $m \in [1, n-1]$ and form $Q(m)$.

(2) Try to factor $Q(m)$ by trial division with the primes $p \leq L_n[\alpha]$.

(3) If $Q(m)$ completely factors in step (2), store m, $Q(m)$ and its factorization.

These steps are repeated enough times until we have $\pi(L_n[\alpha]) + 1 = L_n[\alpha]$ completely factored $Q(m)$'s. To each factored $Q(m)$ we associate a vector $\underline{v}(m) \in (\mathbb{Z}/2\mathbb{Z})^{\pi(L_n[\alpha])}$, where the i-th coordinate of $\underline{v}(m)$ is 0 or 1 depending on whether the i-th prime appears an even or odd number of times in the prime factorization of $Q(m)$. Perform Gaussian elimination to find a linear dependency among those vectors:

$$\underline{v}(m_1) + \underline{v}(m_2) + \ldots + \underline{v}(m_k) \equiv \underline{0} \bmod 2.$$

Compute $x \in \mathbb{Z}$ with $x^2 = Q(m_1) \cdot \ldots \cdot Q(m_k)$ and $y^2 = m_1^2 \cdot m_2^2 \cdot \ldots \cdot m_k^2$. Then we have found a solution of

$$y^2 \equiv x^2 \bmod n,$$

which might lead to a factorization of n. In fact, we have

4.2. Proposition *(Dixon)*
The probability of a factoring success in the random square algorithm is at least 1/2.

Each run of (1), (2), (3) costs $L_n[\alpha]$ operations. Totally we need $L_n[\alpha]$ many relations to start solving the linear system. But how many iterations are neccessary to find one relation? According to corollary 4.2 the probability for a random number $\leq n$ being $L_n[\alpha]$-smooth is $L_n\left[\frac{-1}{2\alpha}\right]$. We make the assumption that the probability to completely factor $Q(m)$ over the factor basis is the same as the probability to completely factor a random number in the interval $[0,\dots,n-1]$ with primes below $L_n[\alpha]$. Thus we need

$$\frac{1}{L_n\left[\frac{-1}{2\alpha}\right]} = L_n\left[\frac{1}{2\alpha}\right]$$

trials until we find one relation. Therefore we need $L_n[\alpha]\cdot L_n\left[\frac{1}{2\alpha}\right]$ many iterations of (1),(2),(3) until we have found enough relations to factor n. The time for finding all relations is therefore $L_n[2\cdot\alpha+1/(2\alpha)]$. The following Gaussian elimination procedure requires $L_n[3\alpha]$ operations and thus, the total running time is $L_n[\max\{2\cdot\alpha+1/(2\alpha),\ 3\alpha\}]$. So $\alpha=\frac{1}{\sqrt{2}}$ is an optimal choice and we get a running time of $L_n\left[\frac{3}{\sqrt{2}}\right]$. But in the next section we will show that a number $\le n$ can be tested for $L_n[\alpha]$-smoothness in time $L_n[0]$. Thus we get a total running time of $L_n[\max\{\alpha+1/(2\alpha), 3\alpha\}]$.

These arguments can be made precise and in this way one obtains

4.3. Proposition *Dixon's random square algorithm with ordinary Gaussian elimination factors n in expected time $L_n\left[\frac{3}{2}\right]$; the space required is $L_n[1]$. If we use Wiedemann's algorithm to solve the linear system the running time is $L_n[\sqrt{2}]$ and we need space $L_n\left[\frac{1}{\sqrt{2}}\right]$.*

4.2 Analysis of the elliptic curve method

We briefly recall the elliptic curve method (ECM):

Choose B_1, B_2 and precompute $m=\prod\limits_{\substack{p\in\mathbb{P}\\ p\le B_1}} p^{e_p}$ where e_p is the largest exponent such that $p^{e_p}\le B_2+2\sqrt{B_2}+1$. Choose $a,\ x,\ y\in\{0,\dots,n-1\}$.

Construct b according to the congruence

$$y^2\equiv x^3+ax+b \bmod n.$$

Determine the discriminant $\Delta=4a^3+27b^2$, $d=\gcd(\Delta,\ n)$. If $d=n$, then the algorithm fails to factor n.

If $1<d<n$, then d is a proper divisor of n.

If $d=1$, then $P_n=(\overline{x}{:}\,\overline{y}{:}\,\overline{1})\in E_n$. Try to compute $m\cdot P_n$. If this computation fails, then a non trivial divisor of n is found. Otherwise the algorithm fails to factor n.

4.4. Proposition *Suppose that n has prime divisors p and q satisfying the following conditions:*

(i) $p\le B_2$,

(ii) each prime number r dividing $\#E_p$ satisfies $r\le B_1$ and

(iii) $\#E_q$ is not divisible by the largest prime number dividing the order of P_p.

Then the ECM is successful.

In order to prove this statement, we need a theorem of Hasse:

4.5. Proposition *(Hasse)*
If $p \not| n$, then $\#E_p = p + 1 + t$ with $|t| \leq 2\sqrt{p}$.

Proof: (Prop. 4.4)
By Proposition 4.5 we find that for every prime number r the exponent of r in $\#E_p$ is at most e_r. The same is then true for the exponent of r in the order w of P_p. Denote by l the largest prime number dividing w, and by u the exponent of l in w. So $1 \leq u \leq e_l$. Put

$$k_0 = (\prod_{p<l} p^{e_p}) \cdot l^{u-1}.$$

Then $k_0 \not\equiv 0 \bmod w$ but $k_0 \cdot l \equiv 0 \bmod w$. So $k_0 \cdot P_p \neq O_p$, $k_0 \cdot l \cdot P_p = O_p$. Now note that $P = O$ if and only if $P_p = O_p$. If $m \cdot P_n$ is successfully computed, then $k_0 \cdot P_n$ and $k_0 \cdot l \cdot P_n$ are computed along the way. It is therefore sufficient to show that not both of these multiples can be successfully determined. If $k_0 \cdot l \cdot P_n$ is successfully computed, then $k_0 \cdot l \cdot P_n = O$ because $k_0 \cdot l \cdot P_p = O_p$. But this means that $k_0 \cdot l \cdot P_q = O$ and since l does not divide the order of P_q, this also means that $k_0 \cdot P_q = O_q$, i.e. $k_0 \cdot P_n = O$, but this contradicts the fact that $k_0 \cdot P_p \neq O_p$. $\square$

4.1. Theorem *(Lenstra)*
Let N be the fraction of successful triples $(a,\, x,\, y) \in [0, \ldots,\, n-1]^3$. Then

$$N \geq \frac{c_0}{\log p} \cdot u$$

where u is the fraction of integers in $[p - \sqrt{p},\, p + \sqrt{p}]$ which only have prime factors below B_1.

Remark: This theorem reduces the problem of predicting the success of the method to a question of elementary (analytic) number theory.

Proof: (Sketch)
Let p be the smallest prime divisor of n and let q be any other prime divisor of n. Denote by N_1 the number of triples $(a',\, x',\, y') \in [0, \ldots,\, p-1]^3$ which define points on elliptic curves over $\mathbb{F}_p$ satisfying (ii). Let N_2 be the number of triples $(a'',\, x'',\, y'') \in [0, \ldots,\, q-1]^3$ which define points P_q on elliptic curves over $\mathbb{F}_q$ satisfying (iii). The proof shows that

$$\begin{aligned} N_1 &\geq c_1 \cdot \tfrac{p^3}{\log p} \cdot u \\ N_2 &\geq c_2 \cdot q^3. \end{aligned} \tag{4.1}$$

Thus by means of the Chinese remainder theorem we find that

$$N \geq \frac{N_1 \cdot N_2}{p^3 \cdot q^3} = \frac{c_0 \cdot u}{\log p}.$$

In order to see (4.1), one needs two important number theoretic results.

4.6. Proposition *(Deuring)*
Let l be a prime number and $t \in \mathbb{Z}$, $|t| \leq 2\sqrt{l}$. The number of elliptic curves E_l over $\mathbb{F}_l$ with $\#\mathbb{E}_l = l + 1 - t$ exceeds $c_3 \cdot l \cdot h\left(\mathbb{Z}\left[\sqrt{t^2 - 4l}\right]\right)$.

4.7. Proposition *(Brauer-Siegel)*

$$h\left(\mathbb{Z}\left[\sqrt{t^2 - 4l}\right]\right) \geq c_4 \cdot \frac{\sqrt{l}}{\log l}.$$

The number of B_1-smooth orders in $[p - \sqrt{p},\, p + \sqrt{p}]$ is approximately $u \cdot \sqrt{p}$. For each of such an order there exist at least $c_4 \cdot p \cdot \frac{\sqrt{p}}{\log p}$ many elliptic curves, i.e. we have so many possibilities to choose values for a. Using Hasse's theorem we know that each curve has approximately p points, i.e. we have $\approx p$ possibilities to choose x and y. Hence we get

$$N_1 \geq \#orders \cdot \#curves \cdot \#points \geq c_1 \cdot (u \cdot \sqrt{p}) \cdot \left(p \cdot \frac{\sqrt{p}}{\log p}\right) \cdot p.$$

Hendrik Lenstra has shown that the number of elliptic curves E over $\mathbb{F}_q$ with $\#E(\mathbb{F}_q) \not\equiv 0 \bmod l$ modulo isomorphism is at least $c \cdot q$. Thus we similarly can show

$$N_2 \geq c_2 \cdot q^3.$$

Note that by this way we get a better bound for the number of curves as if we would have used proposition 4.6 and 4.7. □

Let $B_1 = L_p[\beta]$ and $B_2 = L_p[\alpha]$. As an analogue of corollary 4.2 we have

4.1. Conjecture *If $p \leq B_1$ we have*

$$u = L_p[-1/(2\beta)] = L_{B_1}[-1/(2\beta)],$$

i.e. we assume that the numbers in the interval $[p - \sqrt{p}, p + \sqrt{p}]$ behave like random integers of size $\leq p + \sqrt{p}$.

The expected number of trials before a successful trial occurs is therefore

$$\frac{1}{N} \leq L_p\left[\frac{1}{2\beta}\right] \cdot \log(p) = L_p\left[\frac{1}{2\beta}\right].$$

One addition on the elliptic curve can be done in $O(\log^2 n)$ bit operations. Because we need

$$\log(m) = \log\left(\prod_{p \le B_1} p^{e_p}\right) \le B_1 \cdot \log(B_1) \cdot \log(B_2) = L_p[\beta]$$

additions (using "fast exponentiation") to compute $m \cdot P$, the computing time per trial is $(\log n)^2 \cdot L_p[\beta]$. The total time is therefore

$$L_p\left[\beta + \frac{1}{2\beta}\right] \cdot \log(n)^2$$

which is optimal for $\beta = 1/\sqrt{2}$.

4.2. Theorem *Assume Conjecture 4.1 and assume that n has a divisor $p \le B_1$. Then the expected time for factoring n by means of the ECM is $(\log n)^2 \cdot L_{B_1}[\sqrt{2}]$ and the space used by the algorithm is $L_{B_1}[0]$.*

4.3. Corollary *The expected time for factoring n by ECM is $L_n[1]$ and the space required is $L_n[0]$.*

The analysis shows that the ECM is sensitive to the size of the prime factors of n. This is not true for QS and this is why ECM works very well for numbers n with prime factors of moderate size.

ECM can also be used to test a given number for $L_n[\beta]$-smoothness. This is necessary for many of the factoring algorithms such as QS, RFCA etc. If we test $L_n[\beta]$-smoothness by trial division, then this procedure requires $L_n[\beta]$ operations.

If we use ECM, we cannot prove anything except we modify the concept of smoothness.

Define for $y \in \mathbb{R}_{>3}$ the set $S(y)$ as the set of primes p, $3 < p \le y$, for which the interval $(p - \sqrt{p}, p + \sqrt{p})$ contains at least $\sqrt{p} \cdot \exp\left(-((\log y)^{1/7} \log\log y)/6\right)$ numbers that are $\exp((\log p)^{6/7})$-smooth. It then follows from a result of Friedlander and Lagarias that

$$\pi(y) - \#S(y) = O(y \cdot \exp(-((\log y)^{1/6})/2)). \tag{4.2}$$

We say that an integer is (x, y)-**smooth** if it is $\le x$ and built up from primes p such that $p \le \exp(64(\log\log x)^6)$ or $p \in S(y)$. Define $\psi_1(x, y)$ as the number of (x, y)-smooth integers.

4.8. Proposition *For fixed $\beta \in \mathbb{R}_{>0}$ and $x \to \infty$ we have*

$$\psi_1(x, L_x[\beta]) = x \cdot L_x[-1/2\beta].$$

Proposition 4.8 shows that $(n, L_n[\beta])$-smooth numbers occur asymtotically about as frequently as $L_n[\beta]$-smooth numbers. Those numbers can be detected very quickly. First remove all the prime factors $\le \exp(64(\log\log n)^6)$. If the resulting quotient a is not equal to 1 apply ECM to find the factors $\le L_n[\beta]$. It can be proved

4.9. Proposition *(Pomerance)*
If x is $(n, L_n[\beta])$-smooth, then the complete factorization of x will be found in time $L_n[0]$ with probability $1 - (\log n)/n$.

4.3 Analysis of the quadratic sieve algorithm

Next we want to analyse QS. We briefly recall the algorithm:

For $x \in \mathbb{Z}$ define $z(x) = (x + \lfloor\sqrt{n}\rfloor)^2 - n$. Choose a factor base $\mathcal{F} = \{p \in \mathbb{P} \mid p \leq L_n[\beta],\ (\frac{n}{p}) = 1\}$ for some $\beta \in \mathbb{R}_{>0}$. Try to factor $z(x)$ for $|x| \leq L_n[\alpha]$ completely over $\mathcal{F}$ for some $\alpha \in \mathbb{R}_{>0}$. If $\pi(L_n[\beta]) + 1$ factorizations are found, then use those factorizations to construct a solution of the congruence $x^2 \equiv y^2 \bmod n$. Note that $x \leq L_n[\alpha]$ and the primes $p \in \mathcal{F}$ satisfy $p \leq L_n[\beta]$.

We must analyse the complexity of the sieving procedure, we must determine an appropriate α which yields sufficiently many completely factored quadratic residues mod n and we must estimate the expected time for solving the linear system over $\mathbb{F}_2$ which leads to a solution of the congruence $x^2 \equiv y^2 \bmod n$. Finally, we must discuss the probability for being able to factor n with the solution of this congruence.

In the first step we must precompute the two solutions of $x^2 \equiv n \bmod p^k$ for every $p \in \mathcal{F}$, $p^k \leq c_0 = \max\{z(x);\ |x| \leq L_n[\alpha]\}$. For the sake of simplicity, we may assume that $n \equiv 3 \bmod 4$. Then we must only consider primes p with $p \equiv 3 \bmod 4$. A solution of the congruence $x_1^2 \equiv n \bmod p$ is then given as $x_1 \equiv n^{(p+1)/4} \bmod p$ since $x_1^2 \equiv n^{(p+1)/2} \equiv n^{(p-1)/2} \cdot n \equiv n \bmod p$ (because n is a quadratic residue mod p). The start values for p^k can be computed according to the formulas we have given in the description of the QS. Thus the calculation of each solution requires $L_n[0]$ operations and the number of congruences to consider is $L_n[\beta] \cdot \log n = L_n[\beta]$. Here we have used the fact that for $x \leq L_n[\alpha]$ we have $z(x) = n^{1/2+o(1)}$ and so $\log(c_0) = O(\log(n))$. The expected running time for the precomputation is therefore $L_n[\beta]$.

The sieving time per prime power p^k is $\frac{L_n[\alpha]}{p^k}$. Thus the total sieving time is

$$L_n[\alpha] \cdot \sum_{\substack{p \in \mathcal{F} \\ p^k \leq c_0}} \frac{1}{p^k} \leq L_n[\alpha] \cdot \sum_{i=1}^{c_0} \frac{1}{i} \approx L_n[\alpha] \cdot \log(c_0) = L_n[\alpha].$$

Again we use $\log(c_0) = O(\log(n))$.

The solution of the $L_n[\beta] \times L_n[\beta]$ system by means of the coordinate recurrence method of Wiedemann requires $L_n[2\beta]$ operations (the number of columns is $L_n[\beta]$ and the number of non zero entries per column is $L_n[0]$). If we assume that the probability for a $z(x)$ to be $L_n[\beta]-$smooth is the same as the probability for a random number of size $n^{1/2+o(1)}$ to be $L_n[\beta]-$smooth (i.e. $L_n\left[\frac{-1}{4\beta}\right]$), then we must try

$$L_n[\beta] \cdot L_n\left[\frac{1}{4\beta}\right] = L_n\left[\beta + \frac{1}{4\beta}\right]$$

many x until we have found enough relations. Thus we should choose $\alpha = \beta + \frac{1}{4\beta}$ and we get a total running time of

$$L_n\left[\max\left\{\beta, \beta + \frac{1}{4\beta}, 2\beta\right\}\right].$$

This term is optimal for $\beta = \frac{1}{2}$ and we get

4.3. Theorem *Assuming that the quadratic residues generated in QS behave with respect to smoothness like random integers of size $O(\sqrt{n})$ the expected running time of QS is $L_n[1]$ and space $L_n\left[\frac{1}{2}\right]$.*

4.4 Analysis of the number field sieve

In the NFS we try to find relations over $\mathbb{Z}$ and over the ring $\mathbb{Z}[\rho]$, where ρ is a zero of an irreducible polynomial of degree d. This can in both times be done by sieving. We will now analyse the running time of the NFS.

Therefore we have to introduce a generalisation of the L_n-functions. We define the function $L_n[\nu, \gamma]$ as

$$L_n[\nu, \gamma] = \exp\left((\gamma + o(1)) \cdot (\log n)^{\nu} \cdot (\log\log n)^{1-\nu}\right).$$

Similar to corollary 4.1 we can derive

4.10. Proposition *The probability for a number $\leq L_n[\nu, \gamma]$ being $L_n[\omega, \tau]$-smooth is*

$$L_n\left[\nu - \omega, \frac{-\gamma \cdot (\nu - \omega)}{\tau}\right].$$

We will now use this generalised L_n-function to analyse the NFS. Let $n = r^e - s$ be the number to be factored and r, s small. We must choose an irreducible polynomial f of degree d and a number m with $f(m) \equiv 0 \bmod n$. Supposed we choose the extension degree d close to $\left(\frac{3\log n}{2\log\log n}\right)^{1/3}$. Then it follows that for small s

$$m \approx n^{1/d} = L_n\left[\frac{2}{3}, \left(\frac{2}{3}\right)^{1/3}\right].$$

Let the size C of the factor basis F_1 and a, b be bounded by $L_n\left[\frac{1}{3}, \left(\frac{2}{3}\right)^{2/3}\right]$. If r, s are assumed to be small we get

$$N(a + b \cdot \rho),\ a + b \cdot m \leq L_n\left[\frac{2}{3}, \left(\frac{2}{3}\right)^{1/3}\right].$$

To prove these statements consider

$$\begin{aligned}
|N(a+b\cdot\rho)| &\leq c\cdot(a^d+b^d)\\
&\leq exp\left(\left(\left(\frac{2}{3}\right)^{2/3}+o(1)\right)\cdot(\log n)^{1/3}\cdot(\log\log n)^{2/3}\right)^{((3\log n)/(2\log\log n))^{1/3}}\\
&= L_n\left[\frac{2}{3},\left(\frac{2}{3}\right)^{1/3}\right]
\end{aligned}$$

and

$$\begin{aligned}
a+b\cdot m &\approx exp\left(\left(\left(\frac{2}{3}\right)^{2/3}+o(1)\right)\cdot(\log n)^{1/3}\cdot(\log\log n)^{2/3}\right)\cdot\\
&\quad exp\left(\left(\left(\frac{2}{3}\right)^{1/3}+o(1)\right)\cdot(\log n)^{2/3}\cdot(\log\log n)^{1/3}\right)\\
&= exp\left(\left(\left(\frac{2}{3}\right)^{1/3}+o(1)\right)\cdot(\log n)^{2/3}\cdot(\log\log n)^{1/3}\cdot C\right)\\
&= L_n\left[\frac{2}{3},\left(\frac{2}{3}\right)^{1/3}\right],
\end{aligned}$$

because $C=\frac{\left(\left(\frac{2}{3}\right)^{2/3}+o(1)\right)\cdot(\log\log n)^{1/3}}{\left(\left(\frac{2}{3}\right)^{1/3}+o(1)\right)\cdot(\log n)^{1/3}}+1=1+o(1)$ for $n\to\infty$.

We make the heuristic assumption that these numbers behave as random numbers of size $L_n\left[\frac{2}{3},\left(\frac{2}{3}\right)^{1/3}\right]$. Therefore by proposition 4.10 each of them is $C=L_n\left[\frac{1}{3},\left(\frac{2}{3}\right)^{2/3}\right]$-smooth with probability $L_n\left[\frac{1}{3},-\left(\frac{1}{18}\right)^{1/3}\right]$. The probability that both are simultaniously C-smooth is therefore

$$L_n\left[\frac{1}{3},-\left(\frac{1}{18}\right)^{1/3}\right]\cdot L_n\left[\frac{1}{3},-\left(\frac{1}{18}\right)^{1/3}\right]=L_n\left[\frac{1}{3},-\left(\frac{2}{3}\right)^{2/3}\right].$$

This can be checked in the same way as the upper bound for $a+b\cdot m$. For finding a solution to $x^2\equiv y^2 \bmod n$ we need $\approx 2\cdot C=L_n\left[\frac{1}{3},\left(\frac{2}{3}\right)^{2/3}\right]$ relations. According to the previous statement one relation can be found if we regard $L_n\left[\frac{1}{3},\left(\frac{2}{3}\right)^{2/3}\right]$ different pairs a,b. Thus we can find all relations if we use

$$L_n\left[\frac{1}{3},\left(\frac{2}{3}\right)^{2/3}\right]\cdot L_n\left[\frac{1}{3},\left(\frac{2}{3}\right)^{2/3}\right]=L_n\left[\frac{1}{3},2\cdot\left(\frac{2}{3}\right)^{2/3}\right]$$

different pairs. Note that so many possibilities exist if we choose a,b as described in the beginning. Note further that we find all relations if we sieve for all $b_0\leq L_n\left[\frac{1}{3},\left(\frac{2}{3}\right)^{1/3}\right]$ all possible values for $a+b_0\cdot m$, $N(a+b_0\cdot\rho)$ respectively. This sieving can be done in time

$$L_n\left[\frac{1}{3},\left(\frac{2}{3}\right)^{1/3}\right]\cdot L_n\left[\frac{1}{3},\left(\frac{2}{3}\right)^{1/3}\right]\cdot\log n\leq L_n\left[\frac{1}{3},2\cdot\left(\frac{2}{3}\right)^{1/3}\right].$$

Using Wiedemann's method we can solve the sparse $C \times C$-linear system in time

$$C^2 = L_n\left[\frac{1}{3}, 2 \cdot \left(\frac{2}{3}\right)^{2/3}\right].$$

Thus the total running time of the NFS is

$$L_n\left[\frac{1}{3}, 2 \cdot \left(\frac{2}{3}\right)^{2/3}\right] \approx L_n\left[\frac{1}{3}, 1.5\right].$$

It was for a long time believed that $L_n[1]$ is the best time for factoring a number n. The number field sieve is the first algorithm which has a better running time. Unfortunately it is only applicable for numbers of special form. Nevertheless the discovery of this algorithm was a great advantage in factoring and the today greatest factored (non trivial) number, the Fermat number F_9, was factored in 1990 by using the NFS.

4.5 Analysis of the class group algorithm

Recall the CGM: Suppose that we want to factor n. Put $\Delta = -n$ if $-n \equiv 1 \bmod 4$ and $\Delta = -3 \cdot n$ otherwise. Consider the group of equivalence classes of quadratic forms $(a, b, c) = ax^2 + bxy + cy^2$ of discriminant $\Delta = b^2 - 4 \cdot a \cdot c$. Each equivalence class contains precisely one reduced form by which the class is represented. Since this group, the class group C_Δ, is of finite order h_Δ we can talk about random reduced forms.

The algorithm produces random reduced forms in the following way. It precomputes the system of all prime forms I_p with $p \leq 8 \cdot \log|\Delta|$, say $I_{p_1}, \ldots, I_{p_k}$. Then an exponent vector $(e_1, \ldots, e_k) \in \mathbb{Z}^k_{|\Delta|}$, $\mathbb{Z}_{|\Delta|} = \{0, ..., |\Delta|-1\}$ is chosen at random and the reduced form (a, b) in the class of $\prod_{i=1}^{k} I_{p_i}^{e_i}$ is determined.

Then the algorithm attempts to factor a over the factor base $\mathcal{F} = \{p \in \mathbb{P} \mid p \leq L_n[\beta], (\frac{\Delta}{p}) = 1\} = \{p_1, \ldots, p_l\}$ for some $\beta \in \mathbb{R}_{>0}$. In case of success, the corresponding factorization $(a, b) \sim \prod_{i=1}^{l} I_{p_i}^{e'_i}$ is determined.

If we define $e_i = 0$ for $k < i \leq l$, then $\underline{v} = (e_1 - e'_1, \ldots, e_l - e'_l)$ belongs to the lattice L of all integer vectors $\underline{w} \in \mathbb{Z}^l$ which represent exponent vectors of relations among the generators $\{I_p \mid p \in \mathcal{F}\}$.

We have seen that L has dimension l and determinant h. If we have found $l+1$ relation vectors, say $\underline{v}_1, \ldots, \underline{v}_{l+1}$, then we solve the linear system

$$\sum_{i=1}^{l+1} \lambda_i \cdot \underline{v}_i \equiv \underline{0} \bmod 2.$$

We determine the exponent vector $\underline{w} = \frac{1}{2}\sum_{i=1}^{l+1} \lambda_i \cdot \underline{v}_i$. Then the reduced form equivalent to $\prod_{i=1}^{l} I_{p_i}^{w_i}$ is an ambiguous form and might lead to a factorization.

In order to analyze this algorithm, we must determine the probability for a form to be decomposable over $\{I_p \mid p \in \mathcal{F}\}$. From this probability we obtain the running time for generating $l+1$ completely decomposed forms. Then we must discuss the time for solving the linear system over $\mathbb{F}_2$ and finally we must estimate the probability of success.

In order to analyse the first part of the algorithm where the reduced forms are generated we first prove that those forms behave like random forms in C_Δ. For this purpose we prove

4.11. Proposition *Let $m \in \mathbb{Z}_{\geq 1}$ and let L be a lattice in $\mathbb{Z}^m$ of determinant h such that the exponent of $\mathbb{Z}^m/L$ is at most d. Then for any $\underline{w} \in \mathbb{Z}^m$ and $B \in \mathbb{Z}_{\geq d}$ we have*

(i)

$$\frac{1}{h}\left(1-\frac{h-1}{B}\right) \leq \frac{\#\{\underline{z} \in \mathbb{Z}_B^m \mid \underline{z}-\underline{w} \in L\}}{B^m} \leq \frac{1}{h}\left(1+\frac{h-1}{B}\right),$$

and

(ii)

$$\frac{1}{h}\left(1-\frac{d-1}{B}\right)^m \leq \frac{\#\{\underline{z} \in \mathbb{Z}_B^m,\ \underline{z}-\underline{w} \in L\}}{B^m} \leq \frac{1}{h}\left(1+\frac{d-1}{B}\right)^m.$$

where $\mathbb{Z}_B = \{0, 1, \ldots, B-1\}$.

Proof: Let $\underline{b}_1, \ldots, \underline{b}_m$ be a basis of L such that the matrix $M = (\underline{b}_1, \ldots, \underline{b}_m)$ is in (upper) Hermite normal form, i.e. M is in upper triangular form and $0 \leq b_{ij} < b_{jj}$ for $1 \leq i < j \leq n$. Then $h = \prod_{i=1}^{m} b_{ii}$ and $b_{ii} \leq d$ $(1 \leq i \leq m)$.

Moreover, there is precisely one $\underline{z} \in \mathbb{Z}^m$ with $\underline{w}-\underline{z} \in L$ and $0 \leq z_i < b_{ii}$ $(1 \leq i \leq m)$. In fact, we can construct such a $\underline{z}$ by reducing $\underline{w}$ with $\underline{b}_m, \underline{b}_{m-1}, \ldots, \underline{b}_1$. On the other hand, if there were two different such vectors, say $\underline{z}$ and $\underline{z}'$, then the difference $\underline{z}-\underline{z}' \neq \underline{0}$ would belong to L but could not be represented on the basis $\underline{b}_1, \ldots, \underline{b}_m$ (Exercise).

It follows from this observation that for any $(k_1, \ldots, k_m) \in \mathbb{Z}_{\geq 0}^m$ the number of vectors $\underline{z} \in \mathbb{Z}^m$ with $0 \leq z_i \leq k_i \cdot b_{ii}$ is precisely $\prod_{i=1}^{m} k_i$.

If we put $k_i = \left\lfloor \frac{B}{b_{ii}} \right\rfloor \geq \frac{B-b_{ii}+1}{b_{ii}}$, then we see that the number of vectors $\underline{z} \in \mathbb{Z}_B^m$ with $\underline{w}-\underline{z} \in L$ is at least $\frac{B^m}{h} \prod_{i=1}^{m}\left(1-\frac{b_{ii}-1}{B}\right)$. Using $b_{ii} \leq d$ we obtain the lower bound in (ii) and the lower bound in (i) is obtained by applying repeatedly the inequality

$$\left(1-\frac{a-1}{B}\right) \cdot \left(1-\frac{b-1}{B}\right) \geq \left(1-\frac{a \cdot b-1}{B}\right)$$

for $a, b \geq 1$.

The upper bounds in (i) and (ii) can be proven analogously. $\square$

We use the previous statement to show that the forms generated in the algorithm behave like random reduced forms. For any $\underline{v} \in \mathbb{Z}^k_{|\Delta|}$ we denote by $\phi(\underline{v})$ the reduced form in the class of $\prod_{i=1}^{k} I_{p_i}^{v_i}$. For any arbitrary reduced form $f \in C_\Delta$ we have $f = \phi(\underline{v})$ with probability

$$\frac{\#\{\underline{v} \in \mathbb{Z}^k_{|\Delta|} \mid \phi(\underline{v}) = f\}}{|\Delta|^k}.$$

Applying Proposition 4.11 and using the fact that $h_\Delta = O\left(\sqrt{|\Delta|} \cdot \log |\Delta|\right)$ we find that this probability is $\frac{1}{h_\Delta} \cdot (1 + o(1))$ for $|\Delta| \to \infty$.

Now we can analyse the running time of the method. Using ECM each of the forms can be tested for (slightly modified) $L_n[\beta]$-smoothness in time $L_n[0]$. The probability for a random form to be $L_n[\beta]$-smooth is $L_n[-1/4\beta]$ because a form is $L_n[\beta]$-smooth iff the corresponding a is $L_n[\beta]$-smooth, but $a \leq \sqrt{|\Delta|}$ (Exercise). So the time for generating sufficiently many forms is $L_n[\beta + 1/4\beta]$, since (by virtue of the GRH) the number of prime forms used for generating C_Δ is $L_n[0]$. We see that the linear system to be solved has only $L_n[\beta]$ non-zero entries. Hence, by means of Wiedemann's algorithm, $L_n[2 \cdot \beta]$ operations suffice to solve the system. Hence we again end up with $\beta = \frac{1}{2}$ and running time $L_n[1]$.

4.4. Theorem *Under the assumption of the GRH the expected running time for the class group algorithm is $L_n[1]$ and the space required is $L_n[\frac{1}{2}]$.*

Next we want to analyse the probability of success:

Let H be the subgroup of the group G of all the ambigous forms which lead to a trivial factorization of n. Since n is composite and because each factorization of n corresponds to an ambiguous form we know that H is a proper subgroup of G, so $[G\colon H] \geq 2$. We will show that the probability for the algorithm to yield a specific form $g \in G$ is $\frac{1+o(1)}{|G|}$. Thus the probability for a random form to be an element of H (and so to lead to a trivial factorization of n) is

$$\frac{|H| \cdot (1 + o(1))}{|G|} \leq \frac{1 + o(1)}{2}.$$

Therefore the probability for the algorithm to yield a nontrivial factorization of n is $\frac{1}{2} - o(1)$.

The algorithm determines $l + 1$ exponent vectors $\underline{e}_1, \ldots, \underline{e}_{l+1} \in \mathbb{Z}^k$ such that the reduced form $f_j(U)$ in the class of $\prod_{i=1}^{k} I_{p_i}^{e_{ij}}$ $(1 \leq j \leq l + 1)$ can be decomposed over the factor base $\mathcal{F} = \{I_{p_i} \mid 1 \leq i \leq l\}$, where we set

$$U = (\underline{e}_1, \ldots, \underline{e}_{l+1}).$$

In the next step, the algorithm finds the decomposition $f_j(U) \sim \prod_{i=1}^{l} I_{p_i}^{e'_{ij}}$ and determines the matrix $A = A(U) = (a_{ij})$ where $a_{ij} = e_{ij} - e'_{ij}$ $(1 \le i \le l,\ 1 \le j \le l+1)$ if we additionally define $e_{ij} = 0$ for $k+1 \le i \le l,\ 1 \le j \le l+1$.

The algorithm solves the linear system $A \cdot \underline{x} \equiv 0 \bmod 2$ and determines the ambiguous form $\psi_x(U)$ equivalent to $\prod_{j=1}^{l+1} f_j^{x_j}$.

Let

$$K = \{V = (\underline{v}_1, \ldots, \underline{v}_{l+1}) \in (2\mathbb{Z})^{k \times (l+1)} \mid \underline{v}_j \in L (1 \le j \le l+1)\},$$

then K is a subgroup of $\mathbb{Z}^{k \times (l+1)}$ and we have for every $V \in K$

$$f_j(U+V) = f_j(U) \qquad (1 \le j \le l+1)$$

and thus

$$A(U+V) \equiv A(U) \bmod 2.$$

Hence, the f_j and A mod 2 depend only on the coset of U modulo K. Fix a coset C of U in K. Also fix a solution $\underline{x}$ of $A \cdot \underline{x} = \underline{0} \bmod 2$. For $\underline{v} = (v_1, \ldots, v_l) \in L \cap (2 \cdot \mathbb{Z})^l$ define $\psi(\underline{v})$ as the ambiguous form equivalent to $\prod_{i=1}^{l} I_{p_i}^{v_i/2}$. Then the homomorphism

$$\psi\colon \quad L \cap (2 \cdot \mathbb{Z})^l \to G$$

is surjective because $\mathcal{F}$ generates the class group.

Analogously, define for $V = (\underline{v}_1, \ldots, \underline{v}_{l+1}) \in K$ by $\psi_x(V)$ the ambiguous form in the class of $\prod_{j=1}^{l+1} \psi(\underline{v}_j)^{x_j}$.

We claim that also the homomorphism

$$\psi_x\colon \quad K \to G$$

is surjective. Assume without loss of generality that $x_1 \neq 0$. So $x_1 = 1$ and thus we can find for every $g \in G$ a matrix $V \in K$ all of whose column vectors except for the first one are zero with $\psi_x(V) = g$.

Denote by J the kernel of ψ_x and consider K and J as m-dimensional lattices with $m = k \cdot (l+1)$. Then we can study, how $\psi_x(U)$ varies with $U \in C$. Two matrices $U, U' \in C$ yield the same ambiguous form if their difference $U - U'$ belongs to J. Hence, the probability for the algorithm to yield any given $g \in G$ is

$$\frac{|\mathbb{Z}^m_{|\Delta|} \cap D|}{|\mathbb{Z}^m_{|\Delta|} \cap C|},$$

where D denotes the coset of J in K belonging to g. To estimate this probability we use

$$\mathbb{Z}^m \supset K \supset J \supset (2 \cdot L)^{l+1}.$$

Thus the exponent d_K of K in $\mathbb{Z}^m$ and the exponent d_J of J in $\mathbb{Z}^m$ are at most $2 \cdot h = \det\left((2 \cdot L)^{l+1}\right)$. From Proposition 4.11 we get

$$\frac{1}{\det(J)} \cdot \left(1 - \frac{d_J - 1}{|\Delta|}\right)^m \leq \frac{|\mathbb{Z}^m_{|\Delta|} \cap D|}{|\Delta|^m} \leq \frac{1}{\det(J)} \cdot \left(1 + \frac{d_J - 1}{|\Delta|}\right)^m .$$

From Proposition 4.7 (Brauer-Siegel) we know

$$d_J \leq 2 \cdot h = O(|\Delta|^{1/2} \cdot \log \Delta).$$

Moreover $m = L_{|\Delta|}[\beta] = L_{|\Delta|}[\frac{1}{2}]$. Then we have

$$|\mathbb{Z}^m_{|\Delta|} \cap D| = \frac{|\Delta|^m}{\det(J)} \cdot (1 + o(1))$$

and according

$$|\mathbb{Z}^m_{|\Delta|} \cap C| = \frac{|\Delta|^m}{\det(K)} \cdot (1 + o(1)) .$$

Hence we get

$$\frac{|\mathbb{Z}^m_{|\Delta|} \cap D|}{|\mathbb{Z}^m_{|\Delta|} \cap C|} = \frac{\det(K)}{\det(J)} \cdot (1 + o(1)) = \frac{1}{|G|} \cdot (1 + o(1)) ,$$

because $\psi : K \to G$ is surjective. Hence $G \cong K/J$ respectively $|G| = \frac{\det(J)}{\det(K)}$. By this way we have proven

4.5. Theorem *The probability of success of CGM is* $\frac{1}{2}(1 - o(1))$ *for* $|\Delta| \to \infty$.

Bibliography

[Bru66] N.G. de Bruijn: *On the number of integers $\leq x$ and free of prime factors $> y$,* Indag. Math., 38, 1966, 239-247

[Br81] J. Brillhart: *Fermat's factoring method and its variants,* Congressus numeratium, 32, 1981, 29-48

[Fe94] P. Fermat: *Oeuvres 2,* 1894, 256

[Kn74] D. Knuth: *The Art of computer programming,* vol. 2, Seminumerical algorithms, Addison Wesley 1973

[Ko87] N. Koblitz: *A course in number theory and cryptography,* Springer-Verlag 1987

[Le74] R. S. Lehman: *Factoring large integers,* Math. Comp. 28,1974, 637-646

[LLMP] A.K. Lenstra, H.W. Lenstra Jr., M.S. Manasse, J.M. Pollard: *The number field sieve,* to appear

[Le87] H.W. Lenstra Jr.: *Factoring integers with elliptic curves,* Ann. of math., 126, 1987, 649-673

[MoBr75] M.A. Morrison, J. Brillhart: *A method for factoring and the factorization of F_7,* Math. Comp., 29, 1975, 183-205

[Po85] C. Pomerance: *The quadratic sieve factoring algorithm,* Advances in Cryptology, SLN, 209, 1985, 169-182

[RSA78] R.L. Rivest, A. Shamir, L. Adleman: *A method for obtaining digital signatures and public-key cryptosystems,* Comm. ACM, 21, 1978, 120-126

[Sey87] M. Seysen: *A probabilistic factorization algorithm with quadratic forms of negative discriminant,* Math. Comp., 48, 1987, 757-780

[Si87] R.D. Silverman: *The multiple quadratic sieve,* Math. Comp., 48, 1987, 329-339

[Wi86] D.H. Wiedemann: *Solving sparse linear equations over finite fields,* IEEE Trans. Inform. Theory, IT 32, 1986, 54-62