Number Theory and its Applications, , 1–10 ()

A Survey on the Number Field Sieve

KEN NAKAMULA nakamula@tnt.math.metro-u.ac.jp
Department of Mathematics, Tokyo Metropolitan University, Minami-Ohsawa 1-1, Hachioji, Tokyo, 192-0397 JAPAN

Abstract. The Number Field Sieve is expected to be the fastest among the recent major factoring methods.

Keywords: Facoring Method, Sieving Method, NFS, SNFS, GNFS, DLP

Recent developments of computational number theory have been very much influenced by cryptography. In particular, several methods have been introduced to factor a large natural number. In this note, a survey on the Number Field Sieve, or NFS, will be given as one of the most important of these methods. In § 1, we shall see that the NFS is expected to be the fastest known factoring method by summarizing the run time of the major factoring methods. In § 2, we shall explain the main steps of the procedure which is common to all sieving methods including the NFS. In § 3, we shall give a description of the essential part of the NFS. In § 4, we shall list several problems related to the NFS. In § 5, we shall quote a few records obtained by the NFS recently.

Notation and Basic References

Let n denote a large natural number to be factored. By the run time of a method of factoring n, we mean its worst case time complexity for $n \to \infty$, in many cases the heuristic, conjectured or expected one. To estimate the time complexity, we define

$$L_x[u, v] = \exp(v(\ln x)^u (\ln \ln x)^{1-u})$$

for real numbers x, u, v with $x > e$, where $\ln x = \log_e x$ is the natural logarithm. For $0 \leq u \leq 1$, $v \geq 0$, the function $L_x[u, v]$ is a monotonically increasing function of x with $L_x[u, v] \geq 1$. We have

$$L_x[u, v] = L_x[u, 1]^v,$$

and $L_x[u, v]$ is an increasing function of the parameter v. We also have

$$L_x[0, v] = (\ln x)^v, \quad L_x[1, v] = x^v,$$

$$\ln \ln L_x[u, v] = u \ln \ln L_x[1, v] + (1 - u) \ln \ln L_x[0, v],$$

and $L_x[u, v]$ is an increasing function of the more important parameter u. As $L_x[u, v]$ interpolates between powers of $\ln x$ and powers of $\exp(\ln x) = x$, the order of $L_x[u, v]$ is said to be *subexponential* in the size $\ln x$ of x if $0 < u < 1$.

A good introduction to the NFS is given in [10]. Basic facts, improvements, related topics, references and a history of the NFS can be found in the lecture notes [9]. A historical description of the NFS including its recent developments can be found in [23]. A variation of the GNFS with four large primes is introduced in [5]. A multiple polynomial variation of the GNFS is introduced in [6]. An application of the NFS to solve the Discrete Logarithm Problem, or DLP, for prime fields is discussed in [27]. For the actual implementation of the General NFS, or GNFS, we refer to [2] and [7]. For further bibliographies, see these papers.

1. A Summary of Factoring Methods

A factoring method is classified either as a deterministic one or a probabilistic one. The latter usually expects either some "smooth" prime factors of n or some "smooth" non-trivial congruences modulo n, where the word "smooth" is used to mean "within a range" not so large in a certain sense. We summarize here the run time of the major methods.

It is well-known that the run time of the trial division, which is of course deterministic, is $L_n[1,\ 1/2 + o(1)]$. As an application of the Fast Fourier Transform, there is a deterministic factoring method of run time $L_n[1,\ 1/4 + o(1)]$ as described in [22, pp. 107–109]. The run time of the Class Group Method in [26] is also $L_n[1,\ 1/4 + o(1)]$, and it is decreased to $L_n[1,\ 1/5 + o(1)]$ assuming that certain Extended Riemann Hypotheses are true as is analyzed in [25]. There are several other deterministic factoring methods as in [15], but the problem to refine the run time has made not so much progress.

One of the simplest probabilistic factoring methods is the Pollard ρ-method in [21] which computes the greatest common divisors of n and gaps of a sequence generated by a polynomial. If we assume that the polynomial used to generate the sequence behave like a random map, then the run time of this method is $L_n[1,\ 1/4 + o(1)]$.

Expecting a "smooth" prime factor p of n, several probabilistic factoring methods such as $(p-1)$-method in [20], $(p+1)$-method in [29], etc. are introduced. We should note that the run time of these methods depend not on n but on the smoothness of $p \mp 1$, etc. The Elliptic Curve Method, or ECM, in [12] searches for an elliptic curve modulo p so that the number of rational points is smooth. It is the most significant among the earlier known ones. Its run time is $L_p[1/2,\ 2^{1/2} + o(1)]$ assuming a certain plausible conjecture, therefore it is $L_n[1/2,\ 1 + o(1)]$ in the worst case. This method is useful to find out a prime factor much smaller than $n^{1/2}$. However, the run time of the NFS below is still better for a general n.

All probabilistic factoring methods expecting a "smooth" congruence modulo n have the same algorithmic scheme which will be described in the next section. The Rational Sieve in [10, § 2.7] is a typical example and its run time is $L_n[1/2,\ 2^{1/2} + o(1)]$. The run time of the Continued Fraction Method in [8] comes down to $L_n[1/2,\ (5/4)^{1/2} + o(1)]$ assuming the heuristic hypotheses in [22, §§ 5 and 8]. The run time of the Quadratic Sieve, or QS, is now $L_n[1/2,\ 1 + o(1)]$ by [22, §§ 7 and 8] and [28]. Several variations of the QS are tested in [1] predicting the run time.

By an idea of J. M. Pollard in 1988 to use the rings of integers of number fields, the Special NFS, or SNFS, in [9, pp. 11–42] is first introduced for factoring a special form of $n = r^k - s$ with small natural numbers r and $|s|$, and its run time is $L_n[1/3,\ (32/9)^{1/3} + o(1)]$. The GNFS for an arbitrary n is introduced in [9, pp. 50–94], and its run time is first estimated to be $L_n[1/3,\ (64/9)^{1/3} + o(1)]$, which has been slightly refined to $L_n[1/3,\ ((92 + 26\sqrt{13})/27)^{1/3} + o(1)]$ by [3] at least theoretically. Thus the GNFS is expected to be the fastest factoring method for a general n at present.

2. A General Scheme for Factoring

The following general procedure is common to all probabilistic factoring methods expecting a "smooth" congruence modulo n.

Step 0: Reduce to the case where n is odd and is not a power of a prime number by some simpler preliminary computations. It is an easy task to make n odd, in fact to make it with no prime factor smaller than a reasonable bound. Then a standard prime power test as in [10, §§ 2.4 and 2.5] only requires almost the same order of run time as that of primality test. This step assures that more than 50% of $x + n\mathbb{Z}$ in (1) satisfy (2) in Step 3 below.

Step 1: Select the factor base as a subset of the representatives modulo n. Namely, we take a subset J of $\mathbb{Z}/n\mathbb{Z}$. We may assume that all elements of J are invertible in $\mathbb{Z}/n\mathbb{Z}$, i.e. $J \subseteq (\mathbb{Z}/n\mathbb{Z})^\times$, because otherwise n can be factored with a non-trivial factor $\gcd(x, n)$ for a certain $x \in j \in J$. We usually require each member of J to have a small representative in some sense. An integer expressed as a finite product of those representatives is said to be "smooth" in this situation.

Step 2: Collect relations among the members of the factor base until the number of relations exceed the cardinality of the factor base. Namely, take a set V of integer vectors with a little bit larger cardinality than that of J, i.e. $\#V > \#J$, so that

$$V \subseteq \{\ (v_j)_{j\in J} \in \mathbb{Z}^{\#J} \mid \prod_{j\in J} j^{v_j} = 1\ \}.$$

In order to find out the relations, we usually search for "smooth" integers by means of the sieving technique eliminating inappropriate ones from a list of candidates similarly as the classical Eratosthenes Sieve. That is the reason why this step is called the sieving stage and the method itself is called a sieving method. The sieving tasks are divided into several disjoint subtasks, which can be executed by distributed processing through network following the approach proposed in [11].

Step 3: Find dependencies modulo 2 among the vectors in V. Namely take subsets W of V such that

$$\sum_{v\in W} v = 2(w_j)_{j\in J} \in 2\mathbb{Z}^{\#J}.$$

Since $\#V > \#J$, there exists at least one non-trivial dependency W. For each dependency W, we can calculate $x \in \mathbb{Z}$ so that

$$x + n\mathbb{Z} = \prod_{j \in J} j^{w_j}.$$

Then this x satisfies

$$x^2 \equiv 1 \pmod{n}, \tag{1}$$

and every $x \notin \pm 1 + n\mathbb{Z}$ gives a non-trivial factor $\gcd(x-1, n)$ of n. Therefore the probability of

$$1 < \gcd(x-1, n) < n \tag{2}$$

is at least $1 - 2^{1-s} \geq 1/2$, where s is the number of distinct prime divisors of n with $s > 1$ by Step 0, provided x satisfying (1) is randomly chosen. This step is reduced to a problem of solving large sparse linear equations over $\mathbb{Z}/2\mathbb{Z}$, and the subject is discussed in a recent paper [18].

3. The Idea of the GNFS

The GNFS is based on the fact that it is possible to construct a monic irreducible polynomial $f \in \mathbb{Z}[X]$ and a ring homomorphism

$$\phi : \mathbb{Z}[X]/f\mathbb{Z}[X] \longrightarrow \mathbb{Z}/n\mathbb{Z} \tag{3}$$

so that the coefficients of f and the representative for $\phi(X + f\mathbb{Z}[X])$ in the open interval $(-n/2, n/2)$ have small absolute values compared to n.

We can employ a straightforward way called the "base m" method to construct f and ϕ in (3). Choose a natural number $d > 1$ for the degree of the polynomial f so that $n > 2^{d^2}$, and put $m = \lfloor n^{1/d} \rfloor$. Then the m-ary expansion of n is of the form

$$n = m^d + \sum_{k=0}^{d-1} a_k m^k \quad \text{with} \quad a_k \in \{0, \ldots, m-1\} \ \ (0 \leq k \leq d-1)$$

as in [9, p. 54]. Let

$$f = X^d + \sum_{k=0}^{d-1} a_k X^k.$$

If f is reducible over $\mathbb{Q}$, then the factorization of f produces a non-trivial factor of $n = f(m)$ as in [9, p. 54]. On the other hand, if f is irreducible over $\mathbb{Q}$, then we can define ϕ in (3) by $\phi(X + f\mathbb{Z}[X]) = m + n\mathbb{Z}$. As is required, the coefficients a_k $(0 \leq k \leq d-1)$ of f and the representative m of $\phi(X + f\mathbb{Z}[X])$ are all small compared to n.

Step 0 in the preceding section does not depend on the way we select for sieving. We first describe the idea of Steps 1 and 2 for the GNFS. After that, we explain some problems arising from it and an additional difficulty in Step 3. In oder to make the story clear, we cut down detailed techniques usually used for the efficiency of the algorithm. Let

$$\alpha = X + f\mathbb{Z}[X], \qquad m + n\mathbb{Z} = \phi(\alpha) \quad \text{with} \quad m \in \mathbb{Z}.$$

For simplicity, we assume that $\mathbb{Z}[\alpha] = \mathbb{Z}[X]/f\mathbb{Z}[X]$ is a unique factorization domain, therefore it is integrally closed, and hence it is the ring of integers of its quotient field $\mathbb{Q}(\alpha)$. We fix a prime element $g_\mathfrak{p} \in \mathbb{Z}[\alpha]$ for each prime ideal $\mathfrak{p}$ of $\mathbb{Z}[\alpha]$ so $\mathfrak{p} = g_\mathfrak{p}\mathbb{Z}[\alpha]$.

Step 1: The factor base is selected as the image $J = \phi(P \cup G \cup U)$ of three sets P, G and U with some smoothness bound B chosen appropriately. Here P is the set of all rational prime numbers not exceeding B,

$$G = \{\, g_\mathfrak{p} \mid \mathfrak{p} \text{ is a prime ideal of } \mathbb{Z}[\alpha] \text{ with norm in } P\},$$

and U is a set of generators for the group of units of $\mathbb{Z}[\alpha]$. A finite product of elements of P or G is said to be P-smooth or G-smooth respectively.

Step 2: Search for pairs (a, b) of coprime integers such that $a + bm$ is P-smooth and $a + b\alpha$ is G-smooth at the same time:

$$a + bm = \prod_{p\in P} p^{e(p)} \quad \text{with} \quad e(p) \in \mathbb{Z},\ e(p) \geq 0,$$
$$a + b\alpha = \prod_{u\in U} u^{e(u)} \prod_{g\in G} g^{e(g)} \quad \text{with} \quad e(u),\ e(g) \in \mathbb{Z},\ e(g) \geq 0.$$

Since $\phi(a + bm) = \phi(a + b\alpha)$, we get the relations

$$\prod_{p\in P} \phi(p)^{e(p)} = \prod_{u\in U} \phi(u)^{e(u)} \prod_{g\in G} \phi(g)^{e(g)}.$$

Collecting them we obtain a set V of relation vectors.

A general method to determine G and U may be applied, but it will be an obstacle to speed up our factoring method itself. Practically, it is better to compute G and U simultaneously. We should also take into account of the possible case where $\mathbb{Z}[\alpha]$ is not a unique factorization domain. Furthermore $\mathbb{Z}[\alpha]$ may not be the ring of integers of $\mathbb{Q}(\alpha)$. In addition, it is possible that there are no "small" generating units, which causes a serious problem in practice. All these subjects related to computing G and U are studied in [9, pp. 20–25].

A new obstacle in Step 3 for the GNFS is that we need to detect and compute the square root of a huge algebraic integer given as a product of a great deal of small ones. This subject is analyzed and studied in [9, pp. 58–76 and 95–102].

4. Related Problems

There are many problems to be solved for theoretical treatment and/or practical use of sieving methods. We summarize here some important ones:

1. On the general scheme in § 2:

   (A) Does the factor base J generate $(\mathbb{Z}/n\mathbb{Z})^\times$? Namely, is the map

   $$\Phi : \mathbb{Z}^{\#J} \ni (v_j)_{j \in J} \mapsto \prod_{j \in J} j^{v_j} \in (\mathbb{Z}/n\mathbb{Z})^\times$$

   surjective? Practically, how should we select J in Step 1 so that $\mathrm{Im}(\Phi)$ is large enough?

   (B) Does the set V of collected relations contain fundamental ones? Namely, does V generate $\mathrm{Ker}(\Phi)$? Practically, how should we choose the members of V in Step 2 so that many of them are independent over $\mathbb{Z}$.

   If Φ is surjective and V generates $\mathrm{Ker}(\Phi)$, then all the prime power factors of n are computed in Step 3.

2. Are there generalized or other completely different formulations of a scheme of factoring method similar to that in § 2?

3. On the GNFS:

   (A) How are f and ϕ (or m) to be constructed? The condition $f(m) \equiv 0 \pmod{n}$ is enough. The "base m" method constructs f and m with the condition $f(m) = n$. Are there any other good idea?

   (B) The process of computing G and U contains many questions to be solved as is described in the last section. Further, we can also ask how to solve (3A) so that G and U are determined more easily.

   (C) As is described in the last section, the square root test in $\mathbb{Z}[\alpha]$ is also needed. Given a G-smooth number $\gamma \in \mathbb{Z}[\alpha]$, we should know whether it is a square or not in $\mathbb{Z}[\alpha]$. If it is a square, then we should find an element $\beta \in \mathbb{Z}[\alpha]$ such that $\beta^2 = \gamma$? A new idea utilizing the LLL reduction algorithm is introduced in [17].

   (D) Many of the run time results are not rigorous ones. Therefore a more precise analysis of computational complexity is required. See the paper [13] about this subject.

   (E) Consider the problem to refine the sieving stage related to the above (3C). Namely, find an algorithm to put the square root test in the sieving stage.

   (F) Apply the GNFS to the DLP. Recall that the NFS itself was inspired by a discrete logarithm algorithm for a prime using Gaussian integers as is written in [9, p. 13].

   (G) It may be also important to solve apparently minor obstructions in the GNFS. For example, we should know how to select the parameters for the smoothness in the GNFS, we should know whether the variation of the GNFS with four large primes is really efficient, and we should know whether the multiple polynomial variation of the GNFS is really efficient.

(H) From a practical point of view, it will be the most important to gain much more experience through actual implementations and experiments by computers.

5. A World Wide NFS Record

Factoring the 130-digits natural number RSA130 in [24] by the GNFS through a World Wide Web sieving project is reported in [4]. The following is extracted from the electoronic mail to the Number Theory List [16], or NTL, by A. K. Lenstra.

```
On April 10, 1996, we found that

RSA-130 = 18070820886874048059516561644059055662781025167694013491701270214\
          500566625402440483873411275908123033717818879665631820132148805 57

has the following factorization

RSA-130 = 39685999459597454290161126162883786067576449112810064832555157243
        * 45534498646735972188403686897274408864356301263205069600999044599

This factorization was found using the Number Field Sieve (NFS) factoring
algorithm, and beats the 129-digit record that was set on April, 2, 1994,
by the Quadratic Sieve (QS) factoring algorithm .....................

We used the polynomial

     5748,30224,87384,05200 X^5 +  9882,26191,74822,86102 X^4
  - 13392,49938,91281,76685 X^3 + 16875,25245,88776,84989 X^2
  +  3759,90017,48552,08738 X   - 46769,93055,39319,05995

and its root 125,74411,16841,80059,80468 modulo RSA-130................
```

Another record is factoring the 180-digits number $n = (12^{167}+1)/13$ by the SNFS in [19]. The following is extracted from the e-mail to the NTL on September 3, 1997, by H. te Riele.

```
...............................................................
       CWI (Centrum voor Wiskunde en Informatica) announces the
factorization of N = (12^167 + 1)/13 by the
Special Number Field Sieve (SNFS).  This 180-digit number

        N = 128624807452920064108902728584786650470522446417 13\
            578041943468146981010437930160098645877829367241 94\
            244003676117518184671309490252184868543062502487 62\
            10479447880653257911424439 4693

is the largest ever factored by SNFS.  The previous record was the
167-digit number (3^349 - 1)/2, achieved by NFSNET
(Number Field Sieve NETwork) in February, 1997.

        We used the polynomials

            f(X) = X^5 - 144
            g(X) = 12^33 * X + 1
```

```
with common root m = 12^134 (mod N)..............................

        By the way, the prime factors of N are

            78853915247995992358347387072972515879664753888371\
            88632621814133473912364 69
            (75 decimal digits)
and
            16311784525650292068755854365669702024741136446212\
            03858931609458045532501874726047433264764355226803\
            78897
            (105 decimal digits)
...............................................................
```

The DLP challenge DLP-129 in [14] by K. S. McCurley for a 129-digits prime finite field was solved on 25 January, 1998. The following is extracted from the e-mail to the NTL by D. Weber.

```
..................................................................
The setup is as follows

a=7
b_A=127402180119973946824269244334322849749382042586 93
    16216545577352903229146790959986818609788130465951
    66455458144280588076766033781
b_B=18016228528745310244478283483679989501596704669534
    66973130251217340599537720584759581769106253806921
    01651848662362137934026803049

p=(739*7^{149}-736)/3  prime
q=(p-1)/(2*739)        prime

|(Z/pZ)^*|=2*739*q

Alice computes (using her secret key x_A): 7^x_A = b_A mod p
Bob   computes (using his secret key x_B): 7^x_B = b_B mod p

Kevin McCurley now asks for the common secret key
K=7^(x_A*x_B) mod p.

Here it is:

   K=38127280411190014138078391507929634193998643551018670285056375615
     0455239669294039221021725140532709288726394263700635327977408 08

K has been obtained after computing the secret key x_A of Alice
by means of the Number Field Sieve Discrete Log algorithm (NFS-DL).

 x_A=6185869085965188327359333165203790426798764306952171345914622218 4
     9525998156144877820757492182909777408338791850457946749734 (mod p-1)
.........................................................................
```

The 140-digits natural number RSA140 in [24] is factored by the GNFS on 2 February, 1999. The following is extracted from the electoronic mail to the NTL, by Herman J. J. te Riele.

```
.....................................................................
On February 2, 1999, we found that

RSA-140 =
2129024631825875754749788201627151749780670396327721627823338321538194\
9984056495911366573853021918316783107387995317230889569230873441936471

can be written as the product of two 70-digit primes:

3398717423028438554530123627613875835633986495969597423490929302771479
*
6264200187401285096151654948264442219302037178623509019111660653946049
.....................................................................
This factorization was found using the Number Field Sieve (NFS) factoring
algorithm, and beats the 130-digit record that was set on April 10, 1996,
.....................................................................
We used the two polynomials

F_1(x,y) =                        43 96820 82840     x^5
                            + 39031 56785 38960 y  *x^4
                       - 7387 32529 38929 94572 y^2*x^3
                  - 19 02715 324374 29887 14824 y^3*x^2
              - 6 34410 25694 46461 79139 30613 y^4*x
          + 31855 39170 71474 35039 22235 07494 y^5

F_2(x,y)  =  x - 3 44356 57809 24253 69517 79007 y .....................
```

Such news can be found on the NTL if you issue the command

```
echo 'subscribe nmbrthry' | mail listserv@vm1.nodak.edu
```

on a UNIX workstation. You can get it in Japanese by

```
echo 'subscribe tnt' | mail tnt-request@tnt.math.metro-u.ac.jp
```

from the mailing list of Tools on Number Theory. Related programs, data and papers are stored in the URL

```
ftp://tnt.math.metro-u.ac.jp/pub/
```

including the proceedings of "Algebra and Computation" 1995 and 1997 at Tokyo Metropolitan University in the directories `ac95` and `ac97`.

References

1. H. Boender, H. J. J. te Riele, *Factoring integers with large-prime variations of the quadratic sieve*, Experiment. Math. **5** (1996), 257–273.
2. J. Buchmann, J. Loho, J. Zayer, *An implementation of the general number field sieve*, in CRYPTO '93, Lecture Notes in Comput. Sci. **773**, Springer-Verlag, 1994, 159–165.
3. D. Coppersmith, *Modifications to the number field sieve*, J. Cryptology **6** (1993), 169–180.
4. J. Cowie, B. Dodson, R. M. Elkenbracht-Huizing, A. K. Lenstra, P. L. Montgomery, J. Zayer, *A world wide number field sieve factoring record: on to 512 bits*, in Asiacrypt '96, Lecture Notes in Comput. Sci. **1163**, Springer-Verlag, 1996, 382–394.
5. B. Dodson, A. K. Lenstra, *NFS with four large primes: an explosive experiment*, in CRYPTO '95, Lecture Notes in Comput. Sci. **963**, Springer-Verlag, 1995, 372–385.

6. M. Elkenbracht-Huizing, *A multiple polynomial general number fields sieve*, in ANTS-II, Lecture Notes in Comput. Sci. **1122**, Springer-Verlag, 1996, 99–114.
7. M. Elkenbracht-Huizing, *An implementation of the number field sieve*, Experiment. Math. **5** (1996), 231–253.
8. D. H. Lehmer, R. E. Powers, *On factoring large numbers*, Bull. Amer. Math. Soc. **37** (1931), 770–776.
9. A. K. Lenstra, H. W. Lenstra, Jr. (eds.), *The development of the number field sieve*, Lecture Notes in Mathematics **1554**, Springer-Verlag, 1993.
10. A. K. Lenstra, H. W. Lenstra, Jr., M. S. Manasse, J. M. Pollard, *The factorization of the ninth Fermat number*, Math. Comput. **61** (1993), 319–349.
11. A. K. Lenstra, M. S. Manasse, *Factoring by electronic mail*, in Eurocrypt '89, Lecture Notes in Comput. Sci. **434**, Springer-Verlag, 1990, 355–371.
12. H. W. Lenstra, *Factoring integers with elliptic curves*, Ann. Math. **126** (1987), 649–673.
13. H. W. Lenstra, C. Pomerance, *A rigorous time bound for factoring integers*, J. Amer. Math. Soc. **5** (1992), 483–516.
14. K. S. McCurley, *The discrete logarithm problem*, Proc. Symp. in Appl. Math. **42**, Amer. Math. Soc., Providence, 1990, 49–74.
15. J. McKee, R. Pinch, *Old and new deterministic Factoring Algorithms*, in ANTS-II, Lecture Notes in Comput. Sci. **1122**, Springer-Verlag, 1996, 217–224.
16. V. Miller, Number Theory List, `nmbrthry@ndsuvm1.nodak.edu`, mailing list.
17. P. L. Montgomery, *Square Roots of Products of Algebraic Numbers*, Proc. Symp. in Appl. Math. **48**, Amer. Math. Soc., Providence, 1994, 567-571.
18. P. L. Montgomery, *A block Lanczos algorithm for finding dependencies over* GF(2), in Eurocrypt '95, Lecture Notes in Comput. Sci. **921**, Springer-Verlag, 1995, 106–120.
19. P. Montgomery, S. Cavalla and H. te Riele, *A new world record for the special number field sieve factoring method*, CWI Quarterly **10** (1997), 105–107.
20. J. Pollard, *Theorems on factorization and primality testing*, Proc. Cambridge Phil. Soc. **76** (1974), 521–528.
21. J. Pollard, *A Monte-Carlo method for factorization*, BIT **15** (1975), 331–334.
22. C. Pomerance, *Analysis and comparison of some integer factoring algorithms*, in Computational Methods in Number Theory, Math. Cent. Tracts **154/155**, Math. Cent., Amsterdam 1982, 89–139.
23. C. Pomerance, *A tale of two sieves*, Notices Amer. Math. Soc. **43** (1996), 1473–1485.
24. RSA Data Security Corp. Inc., `sci.crypt`, May 18, 1991; information available by sending electronic mail to `challenge-rsa-list@rsa.com`.
25. R. Schoof, *Quadratic fields and factorization*, in Computational Methods in Number Theory, Math. Cent. Tracts **154/155**, Math. Cent., Amsterdam 1982, 235–286.
26. D. Shanks, *Class Number, a theory of factorization and genera*, Proc. Symp. in Pure Math. **20**, Amer. Math. Soc., Providence, 1971, 415–440.
27. D. Weber, *Computing discrete logarithms with the general number field sieve*, in ANTS-II, Lecture Notes in Comput. Sci. **1122**, Springer-Verlag, 1996, 391–403.
28. D. Wiedemann, *Solving sparse linear equations over finite fields*, IEEE Trans. Inform. Theory **IT-32** (1986), 54–62.
29. H. C. Williams, *A* $p+1$ *method of factoring*, Math. Comput. **39** (1982), 225–234.