Factorization of a 512–bit RSA Modulus [⋆]

Stefania Cavallar[3], Bruce Dodson[8], Arjen K. Lenstra[1], Walter Lioen[3], Peter L. Montgomery[10], Brian Murphy[2], Herman te Riele[3], Karen Aardal[13], Jeff Gilchrist[4], Gérard Guillerm[11], Paul Leyland[9], Joël Marchand[5], François Morain[6], Alec Muffett[12], Chris and Craig Putnam[14], and Paul Zimmermann[7]

[1] Citibank, 1 North Gate Road, Mendham, NJ 07945–3104, USA
`arjen.lenstra@citicorp.com`

[2] Computer Sciences Laboratory, ANU, Canberra ACT 0200, Australia
`murphy@cslab.anu.edu.au`

[3] CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
`{cavallar,walter,herman}@cwi.nl`

[4] Entrust Technologies Ltd., 750 Heron Road, Suite E08, Ottawa, ON, K1V 1A7, Canada
`Jeff.Gilchrist@entrust.com`

[5] Laboratoire Gage, École Polytechnique/CNRS, Palaiseau, France
`Joel.Marchand@medicis.polytechnique.fr`

[6] Laboratoire d'Informatique, École Polytechnique, Palaiseau, France
`morain@lix.polytechnique.fr`

[7] Inria Lorraine and Loria, Nancy, France
`Paul.Zimmermann@loria.fr`

[8] Lehigh University, Bethlehem, PA, USA
`bad0@Lehigh.edu`

[9] Microsoft Research Ltd, Cambridge, UK
`pleyland@microsoft.com`

[10] 780 Las Colindas Road, San Rafael, CA 94903–2346 USA
Microsoft Research and CWI
`pmontgom@cwi.nl`

[11] SITX (Centre of IT resources), École Polytechnique, Palaiseau, France
`Gerard.Guillerm@polytechnique.fr`

[12] Sun Microsystems, Riverside Way, Watchmoor Park, Camberley, UK
`alec.muffett@uk.sun.com`

[13] Dept. of Computer Science, Utrecht University, P.O. Box 80089, 3508 TB Utrecht, The Netherlands
`aardal@cs.uu.nl`

[14] 59 Rangers Dr., Hudson, NH 03051, USA
`craig.putnam@swift.mv.com`

[⋆] Breakdown of individual contributions to this project:
Management: Te Riele; *polynomial selection algorithm*: Montgomery, Murphy; *polynomial selection computations*: Dodson, Lenstra, Montgomery, Murphy; *sieving codes*: Lenstra, Montgomery; *sieving*: Aardal, Cavallar, Dodson, Gilchrist, Guillerm, Lenstra, Leyland, Lioen, Marchand, Montgomery, Morain, Muffett, Putnam, Zimmermann; *filtering*: Cavallar, Montgomery; *linear algebra*: Leyland, Montgomery; *square root*: Montgomery; *data collection, analysis of data and running the NFS code at CWI and SARA*: Cavallar, Lioen, Montgomery; *technical support*: Lioen.
This is a slightly abridged version of the paper which was originally submitted to Eurocrypt 2000: `http://www.cwi.nl/~herman/RSA155/EuCr2000orig.ps` .

Abstract. This paper reports on the factorization of the 512–bit number RSA–155 by the Number Field Sieve factoring method (NFS) and discusses the implications for RSA.

Keywords: Public-key cryptosystems, RSA, factoring, number field sieve

1 Introduction

On August 22, 1999, we completed the factorization of the 512–bit 155–digit number RSA–155 by NFS. The number RSA–155 was taken from the RSA Challenge list [34] as a representative 512–bit RSA modulus. Our result is a new record for factoring general integers. Because 512–bit RSA keys are frequently used for the protection of electronic commerce—at least outside the USA—this factorization represents a breakthrough in research on RSA–based systems.

The previous record, factoring the 140–digit number RSA–140 [8], was established on February 2, 1999, also with the help of NFS, by a subset of the team which factored RSA–155. The amount of computing time spent on RSA–155 was about 8400 MIPS years[1], roughly four times that needed for RSA–140; this is about half of what could be expected from a straightforward extrapolation of the computing time spent on factoring RSA–140 and about a quarter of what would be expected from a straightforward extrapolation of the computing time spent on RSA–130 [11]. The speed-up is due to a new polynomial selection method for NFS of Murphy and Montgomery which was applied for the first time to RSA–140 and now, with improvements, to RSA–155.

Section 2 discusses the implications of this project for the practical use of RSA–based cryptosystems. Section 3 has the details of our computations which resulted in the factorization of RSA–155.

2 Implications for the practice of RSA

RSA is widely used today [17]. The best size for an RSA key depends on the security needs of the user and on how long his/her information needs to be protected.

The amount of CPU time spent to factor RSA–155 was about 8400 MIPS years, which is about four times that used for the factorization of RSA–140. On the basis of the heuristic complexity formula [7] for factoring large N by NFS:

$$\exp\Big((1.923 + o(1))\,(\log N)^{1/3}(\log\log N)^{2/3}\Big)\,, \tag{1}$$

one would expect an increase in the computing time by a factor of about seven.[2] This speed-up has been made possible by algorithmic improvements, mainly in

[1] One *MIPS year* is the equivalent of a computation during one full year at a sustained speed of one **M**illion **I**nstructions **P**er **S**econd.

[2] By "computing time" we mean the *sieve* time, which dominates the total amount of CPU time for NFS. However, there is a trade-off between polynomial search time and sieve time which indicates that a non-trivial part of the total amount of computing

the polynomial generation step [26, 29, 30], and to a lesser extent in the filter step of NFS [9].

The complete project to factor RSA–155 took seven calendar months. The polynomial generation step took about one month on several fast workstations. The most time-consuming step, the sieving, was done on about 300 fast PCs and workstations spread over twelve "sites" in six countries. This step took 3.7 calendar months, in which, summed over all these 300 computers, a total of 35.7 years of CPU-time was consumed. Filtering the relations and building and reducing the matrix corresponding to these relations took one calendar month and was carried out on an SGI Origin 2000 computer. The block Lanczos step to find dependencies in this matrix took about ten calendar days on one CPU of a Cray C916 supercomputer. The final square root step took about two days calendar time on an SGI Origin 2000 computer.

Based on our experience with factoring large numbers we estimate that within three years the algorithmic and computer technology which we used to factor RSA–155 will be widespread, at least in the scientific world, so that by then 512–bit RSA keys will certainly not be safe any more. This makes these keys useless for authentication or for the protection of data required to be secure for a period longer than a few days.

512–bit RSA keys protect 95% of today's E-commerce on the Internet [35]—at least outside the USA—and are used in SSL (Secure Socket Layer) handshake protocols. Underlying this undesirable situation are the old export restrictions imposed by the USA government on products and applications using "strong" cryptography like RSA. However, on January 12, 2000, the U.S. Department of Commerce Bureau of Export Administration (BXA) issued new encryption export regulations which allow U.S. companies to use larger than 512–bit keys in RSA–based products [38]. As a result, one may replace 512–bit keys by 768–bit or even 1024–bit keys thus creating much more favorable conditions for secure Internet communication.

In order to attempt an extrapolation, we give a table of factoring records starting with the landmark factorization in 1970 by Morrison and Brillhart of $F_7 = 2^{128} + 1$ with help of the then new Continued Fraction (CF) method. This table includes the complete list of factored RSA–numbers, although RSA–100 and RSA–110 were not absolute records at the time they were factored. Notice that RSA–150 is still open. Some details on recent factoring records are given in Appendix A to this paper.

Based on this table and on the factoring algorithms which we currently know, we anticipate that within ten years from now 768–bit (232–digit) RSA keys will become unsafe.

Let D be the number of decimal digits in the largest "general" number factored by a given date. From the complexity formula for NFS (1), assuming

time should be spent to the polynomial search time in order to minimize the sieve time. See Subsection *Polynomial Search Time vs. Sieving Time* in Section 3.1. When we use (1) for predicting CPU times, we neglect the $o(1)$–term, which, in fact, is proportional to $1/\log(N)$. All logarithms have base e.

Table 1. Factoring records since 1970

# decimals	date or year	algorithm	effort (MIPS years)	reference
39	Sep 13, 1970	CF		$F_7 = 2^{2^7} + 1$ [27, 28]
50	1983	CF		[6, pp. xliv–xlv]
55–71	1983–1984	QS		[12, Table I on p. 189]
45–81	1986	QS		[36, p. 336]
78–90	1987–1988	QS		[37]
87–92	1988	QS		[32, Table 3 on p. 274]
93–102	1989	QS		[21]
107–116	1990	QS	275 for C116	[22]
RSA–100	Apr 1991	QS	7	[34]
RSA–110	Apr 1992	QS	75	[14]
RSA–120	Jun 1993	QS	835	[13]
RSA–129	Apr 1994	QS	5000	[2]
RSA–130	Apr 1996	NFS	1000	[11]
RSA–140	Feb 1999	NFS	2000	[8]
RSA–155	Aug 1999	NFS	8400	this paper

Moore's law (computing power doubles every 18 months), Brent [5] expects $D^{1/3}$ to be roughly a linear function of the calendar year Y. From the data in Table 1 he derives the linear formula

$$Y = 13.24D^{1/3} + 1928.6.$$

According to this formula, a general 768–bit number (D=231) will be factored by the year 2010, and a general 1024–bit number (D=309) by the year 2018.

Directions for selecting cryptographic key sizes now and in the coming years are given in [23].

The vulnerability of a 512–bit RSA modulus was predicted long ago. A 1991 report [3, p. 81] recommends:

> *For the most applications a modulus size of 1024 bit for RSA should achieve a sufficient level of security for "tactical" secrets for the next ten years. This is for long-term secrecy purposes, for short-term authenticity purposes 512 bit might suffice in this century.*

3 Factoring RSA–155

We assume that the reader is familiar with NFS [19], but for convenience we briefly describe the method here. Let N be the number we wish to factor, known to be composite. There are four main steps in NFS: polynomial selection, sieving, linear algebra, and square root.

The *polynomial selection step* selects two irreducible polynomials $f_1(x)$ and $f_2(x)$ with a common root $m \bmod N$. The polynomials have as many smooth values as practically possible over a given factor base.

The *sieve step* (which is by far the most time-consuming step of NFS), finds pairs (a, b) with $\gcd(a, b) = 1$ such that both

$$b^{\deg(f_1)} f_1(a/b) \text{ and } b^{\deg(f_2)} f_2(a/b)$$

are smooth over given factor bases, i.e., factor completely over the factor bases. Such a pair (a, b) is called a *relation*. The purpose of this step is to collect so many relations that several subsets S of them can be found with the property that a product taken over S yields an expression of the form

$$X^2 \equiv Y^2 \pmod{N}. \tag{2}$$

For approximately half of these subsets, computing $\gcd(X - Y, N)$ yields a non-trivial factor of N (if N has exactly two distinct factors).

The *linear algebra step* first filters the relations found during sieving, with the purpose of eliminating duplicate relations and relations containing a prime or prime ideal which does not occur elsewhere. In addition, certain relations are *merged* with the purpose of eliminating primes and prime ideals which occur exactly k times in k different relations, for $k = 2, \ldots, 8$. These merges result in so-called relation–sets, defined in Section 3.3, which form the columns of a very large sparse matrix over $\mathcal{F}_2$. With help of an iterative block Lanczos algorithm a few dependencies are found in this matrix: this is the most time– and space–consuming part of the linear algebra step.

The *square root step* computes the square root of an algebraic number of the form

$$\prod_{(a,b) \in S} (a - b\alpha),$$

where α is a root of one of the polynomials $f_1(x), f_2(x)$, and where for RSA–155 the numbers a, b and the cardinality of the set S can all be expected to be many millions. All $a - b\alpha$'s have smooth norms. With the mapping $\alpha \mapsto m \bmod N$, this leads to a congruence of the form (2).

In the next four subsections, we describe these four steps, as carried out for the factorization of RSA–155.

3.1 Polynomial selection

This section has three parts. The first two parts are aimed at recalling the main details of the polynomial selection procedure, and describing the particular polynomials used for the RSA–155 factorization.

Relatively speaking, our selection for RSA–155 is approximately 1.7 times better than our selection for RSA–140. We made better use of our procedure for RSA–155 than we did for RSA–140, in short by searching longer. This poses a new question for NFS factorizations—what is the optimal trade-off between increased polynomial search time and the corresponding saving in sieve time? The third part of this section gives preliminary consideration to this question as it applies to RSA–155.

The Procedure Our polynomial selection procedure is outlined in [8]. Here we merely restate the details. Recall that we generate two polynomials f_1 and f_2, using a base-m method. The degree d of f_1 is fixed in advance (for RSA–155 we take $d = 5$). Given a potential a_5, we choose an integer $m \approx (N/a_d)^{1/d}$. The polynomial

$$f_1(x) = a_d x^d + a_{d-1} x^{d-1} + \ldots + a_0 \tag{3}$$

descends from the base-m representation of N, initially adjusted so that $|a_i| \leq m/2$ for $0 \leq i \leq d-1$.

Sieving occurs over the homogeneous polynomials $F_1(x,y) = y^d f_1(x/y)$ and $F_2(x,y) = x - my$. The aim for polynomial selection is to choose f_1 and m such that the values $F_1(a,b)$ and $F_2(a,b)$ are simultaneously smooth at many coprime integer pairs (a,b) in the sieving region. That is, we seek $F_1,\ F_2$ with good *yield*. Since F_2 is linear, we concentrate on the choice of F_1.

There are two factors which influence the yield of F_1, *size* and *root properties*, so we seek F_1 with a good *combination* of size and root properties. By size we refer to the magnitude of the values taken by F_1. By root properties we refer to the extent to which the distribution of the roots of F_1 modulo small p^n, for p prime and $n \geq 1$, affects the likelihood of F_1 values being smooth. In short, if F_1 has many roots modulo small p^n, the values taken by F_1 "behave" as if they are much smaller than they actually are. That is, on average, the likelihood of F_1-values being smooth is increased.

Our search is a *two stage* process. In the first stage we generate a large sample of good polynomials (polynomials with good combinations of size and root properties). In the second stage we identify *without* sieving, the best polynomials in the sample. We concentrate on *skewed* polynomials, that is, polynomials $f_1(x) = a_5 x^5 + \ldots + a_0$ whose first few coefficients (a_5, a_4 and a_3) are small compared to m, and whose last few coefficients (a_2, a_1 and a_0) may be large compared to m. Usually $|a_5| < |a_4| < \cdots < |a_0|$. To compensate for the last few coefficients being large, we sieve over a skewed region, i.e., a region that is much longer in x than in y. We take the region to be a rectangle whose width-to-height ratio is s.

The first stage of the process, generating a sample of polynomials with good yield, has the following main steps ($d = 5$):

- Guess leading coefficient a_d, usually with several small prime divisors (for projective roots).
- Determine initial m from $a_d m^d \approx N$. If the approximation $(N - a_d m^d)/m^{d-1}$ to a_{d-1} is not close to an integer, try another a_d. Otherwise use (3) to determine a starting f_1.
- Try to replace the initial f_1 by a smaller one. This numerical optimization step replaces $f_1(x)$ by

  $$f_1(x+k) + (cx + d) * (x + k - m)$$

  and m by $m - k$, sieving over a region with skewness s. It adjusts four real parameters c, d, k, s, rounding the optimal values (except s) to integers.

– Make adjustments to f_1 which cause it to have exceptionally good root properties, *without* destroying the qualities inherited from above. The main adjustment is to consider integer pairs j_1, j_0 (with j_1 and j_0 small compared to a_2 and a_1 respectively) for which the polynomial

$$f_1(x) + (j_1 x - j_0) \cdot (x - m)$$

has exceptionally good root properties modulo many small p^n. Such pairs j_1, j_0 are identified using a sieve-like procedure. For each promising (j_1, j_0) pair, we revise the translation k and skewness s by repeating the numerical optimization on these values alone.

In the second stage of the process we rate, without sieving, the yields of the polynomial pairs F_1, F_2 produced from the first stage. We use a parameter which quantifies the effect of the root properties of each polynomial. We factor this parameter into estimates of smoothness probabilities for F_1 and F_2 across a region of skewness s.

At the conclusion of these two stages we perform short sieving experiments on the top-ranked candidates.

Results Four of us spent about 100 MIPS years on finding good polynomials for RSA–155. The following pair, found by Dodson, was used to factor RSA–155:

$$\begin{aligned} F_1(x,y) = {} & 11\,93771\,38320\,x^5 \\ & -80\,16893\,72849\,97582\,x^4y \\ & -66269\,85223\,41185\,74445\,x^3y^2 \\ & +1\,18168\,48430\,07952\,18803\,56852\,x^2y^3 \\ & +745\,96615\,80071\,78644\,39197\,43056\,x\,y^4 \\ & -40\,67984\,35423\,62159\,36191\,37084\,05064\,y^5 \\[1em] F_2(x,y) = {} & x - 3912\,30797\,21168\,00077\,13134\,49081\,y \end{aligned}$$

with $s \approx 10800$.

For the purpose of comparison, we give statistics for the above pair similar to those we gave for the RSA–140 polynomials in [8]. Denote by a_{max} the largest $|a_i|$ for $i = 0, \ldots, d$. The un-skewed analogue, $F_1(104x, y/104)$, of F_1 has $a_{max} \approx 1.1 \cdot 10^{23}$, compared to the typical case for RSA–155 of $a_{max} \approx 2.4 \cdot 10^{25}$. The un-skewed analogue of F_2 has $a_{max} \approx 3.8 \cdot 10^{26}$. Hence, F_1 values have shrunk by approximately a factor of 215, whilst F_2 values have grown by a factor of approximately 16. F_1 has real roots x/y near -11976, -2225, 1584, 12012 and 672167.

With respect to the root properties of F_1 we have $a_5 = 2^4 \cdot 3^2 \cdot 5 \cdot 11^2 \cdot 19 \cdot 41 \cdot 1759$. Also, $F_1(x,y)$ has 20 roots x/y modulo the six primes from 3 to 17 and an additional 33 roots modulo the 18 primes from 19 to 97. As a result of its root properties, F_1-values have smoothness probabilities similar to those of random integers which are smaller by a factor of about 800.

Polynomial Search Time vs. Sieving Time The yield of our two RSA–155 polynomials is approximately 13.5 times that of a skewed pair of average yield for RSA–155 (about half of which comes from root properties and the other half from size). The corresponding figure for the RSA–140 pair is approximately 8 (about a factor of four of which was due to root properties and the remaining factor of 2 to size). From this we deduce that, relatively speaking, our RSA–155 selection is approximately 1.7 times "better" than our RSA–140 selection.

Note that this is consistent with the observed differences in sieve time. As noted above, straightforward extrapolation of the NFS asymptotic run-time estimate (1) suggests that sieving for RSA–155 should have taken approximately 7 times as long as RSA–140. The actual figure is approximately 4. The difference can be approximately reconciled by the fact that the RSA–155 polynomial pair is, relatively, about 1.7 times "better" than the RSA–140 pair.

Another relevant comparison is to the RSA–130 factorization. RSA–130 of course was factorized *without* our improved polynomial selection methods. The polynomial pair used for RSA–130 has a yield approximately 3.2 times that of a random (un-skewed) selection or RSA–130. Extrapolation of the asymptotic NFS run-time estimate suggests that RSA–140 should have taken about 4 times as long as RSA–130, whereas the accepted difference is a factor of about 2. The difference is close to being reconciled by the RSA–140 polynomial selection being approximately 2.5 times better than the RSA–130 selection. Finally, to characterize the overall improvement accounted for by our techniques, we note that the RSA–155 selection is approximately 4.2 times better (relatively) than the RSA–130 selection.

Since the root properties of the non-linear polynomials for RSA–140 and RSA–155 are similar, most of the difference between them comes about because the RSA–155 selection is relatively "smaller" than the RSA–140 selection. This in turns comes about because we conducted a longer search for RSA–155 than we did for the RSA–140 search, so it was more likely that we would find good size and good root properties coinciding in the same polynomials. In fact, we spent approximately 100 MIPS years on the RSA–155 search, compared to 60 MIPS years for RSA–140.

Continuing to search for polynomials is worthwhile only as long as the saving in sieve time exceeds the extra cost of the polynomial search. We have analyzed the "goodness" distribution of all polynomials generated during the RSA–155 search. Modulo some crude approximations, the results appear in Table 2. The table shows the expected benefit obtained from κ times the polynomial search effort we actually invested (100 MY), for some useful κ. The second column gives the change in search time corresponding to the κ-altered search effort. The third column gives the expected change in sieve time, calculated from the change in yield according to our "goodness" distribution. Hence, whilst the absolute benefit may not have been great, it would probably have been worthwhile investing up to about twice the effort than we did for the RSA–155 polynomial search. We conclude that, in the absence of further improvements, it is worthwhile using our

Table 2. Effect of varying the polynomial search time on the sieve time

κ	**change** in search time (in MY)	**change** in sieve time (in MY)
0.2	-80	$+260$
0.5	-50	$+110$
1	0	0
2	$+100$	-110
5	$+400$	-260
10	$+900$	-380

method to find polynomials whose yields are approximately 10–15 times better than a random selection.

3.2 Sieving

Two sieving methods were used simultaneously: lattice sieving and line sieving. This is probably more efficient than using a single sieve, despite the large percentage of duplicates found (about 14%, see Section 3.3): both sievers deteriorate as the special q, resp. y (see below) increase, so we exploited the most fertile parts of both. In addition, using two sievers offers more flexibility in terms of memory: lattice sieving is possible on smaller machines; the line siever needs more memory, but discovers each relation only once.

The lattice siever fixes a prime q, called the special q, which divides $F_1(x_0, y_0)$ for some known nonzero pair (x_0, y_0), and finds (x, y) pairs for which both $F_1(x, y)/q$ and $F_2(x, y)$ are smooth. This is carried out for many special q's. Lattice sieving was introduced by Pollard [31] and the code we used is the implementation written by Arjen Lenstra and described in [18, 11], with some additions to handle skewed sieving regions efficiently.

The line siever fixes a value of y (from $y = 1, 2, \ldots$ up to some bound) and finds values of x in a given interval for which both $F_1(x, y)$ and $F_2(x, y)$ are smooth. The line siever code was written by Peter Montgomery, with help from Arjen Lenstra, Russell Ruby, Marije Elkenbracht-Huizing and Stefania Cavallar.

For the lattice sieving, both the rational and the algebraic factor base bounds were chosen to be $2^{24} = 16\,777\,216$. The number of primes was about one million in each factor base. Two large primes were allowed on each side in addition to the special q input. The reason that we used these factor base bounds is that we used the lattice sieving implementation from [18] which does not allow larger factor base bounds. That implementation was written for the factorization of RSA–130 and was never intended to be used for larger numbers such as RSA–140, let alone RSA–155. We expect that a rewrite of the lattice siever that would allow larger factor base bounds would give a much better lattice sieving performance for RSA–155.

Most of the line sieving was carried out with two large primes on both the rational and the algebraic side. The rational factor base consisted of $2\,661\,384$ primes $< 44\,000\,000$ and the algebraic factor base consisted of $6\,304\,167$ prime ideals of norm $< 110\,000\,000$ (including the seven primes which divide the leading coefficient of $F_1(x, y)$). Some line sieving allowed *three* large primes instead of *two* on the algebraic side. In that case the rational factor base consisted of 539 777 primes $< 8\,000\,000$ and the algebraic factor base of 1 566 598 prime ideals of norm $< 25\,000\,000$ (including the seven primes which divide the leading coefficient of $F_1(x, y)$).

For both sievers the large prime bound 1 000 000 000 was used both for the rational and for the algebraic primes.

The lattice siever was run for most special q's in the interval $[2^{24}, 3.08 \times 10^8]$. Each special q has at least one root r such that $f_1(r) \equiv 0 \bmod q$. For example, the equation $f_1(x) \equiv 0 \bmod q$ has five roots for $q = 83$, namely $x = 8$, 21, 43, 54, 82, but no roots for $q = 31$. The total number of special q–root pairs (q, r) in the interval $[2^{24}, 3.08 \times 10^8]$ equals about 15.7M. Lattice sieving ranged over a rectangle of 8192 by 5000 points per special q–root pair. Taking into account that we did not sieve over points (x, y) where both x and y are even, this gives a total of 4.8×10^{14} sieving points. With lattice sieving a total of 94.8M relations were generated at the expense of 26.6 years of CPU time. Averaged over all the CPUs on which the lattice siever was run, this gives an average of 8.8 CPU seconds per relation.

For the line sieving with two large primes on both sides, sieving ranged over the regions[3]:

$$\begin{aligned} |x| &\le 1\,176\,000\,000, & 1 &\le y \le 25\,000, \\ |x| &\le 1\,680\,000\,000, & 25\,001 &\le y \le 110\,000, \\ |x| &\le 1\,680\,000\,000, & 120\,001 &\le y \le 159\,000, \end{aligned}$$

and for the line sieving with three large primes instead of two on the algebraic side, the sieving range was:

$$|x| \le 1\,680\,000\,000, \quad 110\,001 \le y \le 120\,000.$$

Not counting the points where both x and y are even, this gives a total of 3.82×10^{14} points sieved by the line siever. With line sieving a total of 36.0M relations were generated at the expense of 9.1 years of CPU time. Averaged over all the CPUs on which the line siever was run, it needed 8.0 CPU seconds to generate one relation.

Sieving was done at twelve different locations where a total of 130.8M relations were generated, 94.8M by lattice sieving and 36.0M by line sieving. Each incoming file was checked at the central site for duplicates: this reduced the total number of useful incoming relations to 124.7M. Of these, 88.8M (71%) were found by the lattice siever and 35.9M (29%) by the line siever. The breakdown of the 124.7M relations (in %) among the twelve different sites[4] is given in Table 3.

Calendar time for the sieving was 3.7 months. Sieving was done on about 160 SGI and Sun workstations (175–400 MHz), on eight R10000 processors

[3] The somewhat weird choice of the line sieving intervals was made because more contributors chose line sieving than originally estimated.

[4] Lenstra sieved at two sites, viz., Citibank and Univ. of Sydney.

Table 3. Breakdown of sieving contributions

%	number of CPU days sieved	La(ttice) Li(ne)	Contributor
20.1	3057	La	Alec Muffett
17.5	2092	La, Li	Paul Leyland
14.6	1819	La, Li	Peter L. Montgomery, Stefania Cavallar
13.6	2222	La, Li	Bruce Dodson
13.0	1801	La, Li	François Morain and Gérard Guillerm
6.4	576	La, Li	Joël Marchand
5.0	737	La	Arjen K. Lenstra
4.5	252	Li	Paul Zimmermann
4.0	366	La	Jeff Gilchrist
0.65	62	La	Karen Aardal
0.56	47	La	Chris and Craig Putnam

(250 MHz), on about 120 Pentium II PCs (300–450 MHz), and on four Digital/Compaq boxes (500 MHz). The total amount of CPU-time spent on sieving was 35.7 CPU years.

We estimate the equivalent number of MIPS years as follows. For each contributor, Table 4 gives the number of million relations generated (rounded to two decimals), the number of CPU days d_s sieved for this and the estimated average speed s_s, in million instructions per seconds (MIPS), of the processors on which these relations were generated. In the last column we give the corresponding number of MIPS years $d_s s_s/365$. For the time counting on PCs, we notice that on PCs one usually get *real times* which may be higher than the CPU times.

Summarizing gives a total of 8360 MIPS years (6570 for lattice and 1790 for line sieving). For comparison, RSA–140 took about 2000 MIPS years and RSA–130 about 1000 MIPS years.

A measure of the “quality” of the sieving may be the average number of points sieved to generate one relation. Table 5 gives this quantity for RSA–140 and for RSA–155, for the lattice siever and for the line siever. This illustrates that the sieving polynomials were better for RSA-155 than for RSA–140, especially for the line sieving. In addition, the increase of the linear factor base bound from 500M for RSA–140 to 1000M for RSA–155 accounts for some of the change in yield. For RSA–155, the factor bases were much bigger for line sieving than for lattice sieving. This explains the increase of efficiency of the line siever compared with the lattice siever from RSA–140 to RSA–155.

3.3 Filtering and finding dependencies

The filtering of the data and the building of the matrix were carried out at CWI and took one calendar month.

Table 4. # MIPS years spent on lattice (La) and line (Li) sieving

Contributor	# relations	# CPU days sieved	average speed of processors in MIPS	# MIPS years
Muffett, La	27.46M	3057	285	2387
Leyland, La	19.27M	1395	300	1146
Leyland, Li	4.52M	697	300	573
CWI, La	1.60M	167	175	80
CWI, Li, 2LP	15.64M	1160	210	667
CWI, Li, 3LP	1.00M	492	50	67
Dodson, La	10.28M	1631	175	782
Dodson, Li	7.00M	591	175	283
Morain, La	15.83M	1735	210	998
Morain, Li	1.09M	66	210	38
Marchand, La	7.20M	522	210	300
Marchand, Li	1.11M	54	210	31
Lenstra, La	6.48M	737	210	424
Zimmermann, Li	5.64M	252	195	135
Gilchrist, La	5.14M	366	350	361
Aardal, La	0.81M	62	300	51
Putnam, La	0.76M	47	300	39

Table 5. Average number of points sieved per relation

	lattice siever	line siever
RSA–140	1.5×10^6	3.0×10^7
RSA–155	5.1×10^6	1.1×10^7

Filtering Here we describe the filter strategy which we used for RSA–155. An essential difference with the filter strategy used for RSA–140 is that we applied k-way merges (defined below) with $2 \leq k \leq 8$ for RSA–155, but only 2- and 3-way merges for RSA–140.

First, we give two definitions. A *relation–set* is one relation, or a collection of two or more relations generated by a merge. A *k-way merge* ($k \geq 2$) is the action of combining k relation–sets with a common prime ideal into $k-1$ relation–sets, with the purpose of eliminating that common prime ideal. This is done such that the weight increase is minimal by means of a *minimum spanning tree* algorithm [9].

Among the 124.7M relations collected from the twelve different sites, 21.3M duplicates were found generated by lattice sieving, as well as 17.9M duplicates caused by the simultaneous use of the lattice and the line siever.

During the first filter round, only prime ideals with norm > 10M were considered. In a later stage of the filtering, this 10M-bound was reduced to 7M, in order to improve the possibilities for merging relations. We added 0.2M *free relations* for prime ideals of norm > 10M (cf. [16, Section 4, pp. 234–235]). From the resulting 85.7M relations, 32.5M singletons were deleted, i.e., those relations with a prime ideal of norm > 10M which does not occur in any other undeleted relation.

We were left with 53.2M relations containing 42.6M different prime ideals of norm > 10M. If we assume that each prime and each prime ideal with norm < 10M occurs at least once, then we needed to reserve at least $(2 - \frac{1}{120})\pi(10^7)$ excess relations for the primes and the prime ideals of norm smaller than 10M, where $\pi(x)$ is the number of primes below x. The factor 2 comes from the *two* polynomials and the correction factor 1/120 takes account of the presence of free relations, where 120 is the order of the Galois group of the algebraic polynomial. With $\pi(10^7) = 664\,579$ the required excess is about 1.3M relations, whereas we had 53.2M $-$ 42.6M $=$ 10.6M excess relations at our disposal.

In the next *merging* step 33.0M relations were removed which would have formed the heaviest relation–sets when performing 2-way merges, reducing the excess from 10.6M to about 2M relations. So we were still allowed to discard about 2.0M $-$ 1.3M $=$ 0.7M relations. The remaining 20.1M non-free relations[5] having 18.2M prime ideals of norm > 10M were used as input for the merge step which eliminated prime ideals occurring in up to eight different relation–sets. During this step we looked at prime ideals of norm > 7M. Here, our approach differs from what we did for RSA–140, where only primes occurring twice or thrice were eliminated. Applying the new filter strategy to RSA–140 would have resulted in a 30% smaller (3.3M instead of 4.7M columns) but only 20% heavier matrix than the one actually used for the factorization of RSA–140 and would have saved 27% on the block Lanczos run time. The k ($k \leq 8$) relations were combined into the lightest possible $k - 1$ relation–sets and the corresponding prime ideal (row in the matrix) was "balanced" (i.e., all entries of the row were made 0). The overall effect was a reduction of the matrix size by one row and one column while increasing the matrix weight when $k > 2$, as described below. We did not perform all possible merges. We limited the program to only do merges which caused a weight increase of at most 7 original relations. The merges were done in ascending order of weight increase.

Since each k-way merge causes an increase of the matrix weight of about $(k - 2)$ times the weight of the lightest relation–set, these merges were not always executed for higher values of k. For example, 7- and 8-way merges were not executed if all the relation–sets were already-combined relations. We decided to discard relation–sets which contained more than 9 relations and to stop merging (and discarding) after 670K relations were discarded. At this point we should have slightly more columns than rows and did not want to lose any more columns. The maximum discard threshold was reached during the 10th pass through the 18.6M prime ideals of norm > 7M, when we allowed the maximum weight increase to be about 6 relations. This means that no merges with weight increase of 7 relations were executed. The filter program stopped with 6.7M relation sets.

For more details and experiments with RSA–155 and other numbers, see [9].

[5] The 0.1M free relations are not counted in these 20.1M relations because the free relations are generated during each filter run.

Finding dependencies From the matrix left after the filter step we omitted the small primes < 40, thus reducing the weight by 15%. The resulting matrix had 6 699 191 rows, 6 711 336 columns, and weight 417 132 631 (62.27 non–zeros per row). With the help of Peter Montgomery's Cray implementation of the block Lanczos algorithm (cf. [25]) it took 224 CPU hours and 2 Gbytes of central memory on the Cray C916 at the SARA Amsterdam Academic Computer Center to find 64 dependencies among the rows of this matrix. Calendar time for this job was 9.5 days.

In order to extract from these 64 dependencies some dependencies for the matrix *including the primes* < 40, quadratic character checks were used as described in [1], [7, §8, §12.7], and [15, last paragraph of Section 3.8 on pp. 30–31]. This yielded a dense 100×64 homogeneous system which was solved by Gaussian elimination. That system turned out to have 14 independent solutions, which represent linear combinations of the original 64 dependencies.

3.4 The square root step

On August 20, 1999, four different square root (cf. [24]) jobs were started in parallel on four different 300 MHz processors of an SGI Origin 2000, each handling one dependency. One job found the factorization after 39.4 CPU-hours, the other three jobs found the trivial factorization after 38.3, 41.9, and 61.6 CPU-hours (different CPU times are due to the use of different parameters in the four jobs).

We found that the 155–digit number

RSA–155 =
10941738641570527421809707322040357612003732945449205990913842131476349984288\
34784717997257891267332497625752899781833797076537244027146743531593354333897

can be written as the product of two 78-digit primes:

$p =$
102639592829741105772054196573991675900716567808038066803341933521790711307779

and

$q =$
106603488380168454820927220360012878679207958575989291522270608237193062808643.

Primality of the factors was proved with the help of two different primality proving codes [4, 10]. The factorizations of $p \pm 1$ and $q \pm 1$ are given by

$p - 1 = 2 \cdot 607 \cdot$
$\cdot 305999 \cdot 276297036357806107796483997979900139708537040550885894355659143575473$
$p + 1 = 2^2 \cdot 3 \cdot 5 \cdot$
$\cdot 5253077241827 \cdot 325649100849833342436871870477394634879398067295372095291531269$

$q - 1 = 2 \cdot 241 \cdot$
$\cdot 430028152261281581326171 \cdot 514312985943800777534375166399250129284222855975011$
$q + 1 = 2^2 \cdot 3 \cdot 130637011 \cdot$
$\cdot 237126941204057 \cdot 10200242155298917871797 \cdot 28114641748343531603533667478173$

Acknowledgements. Acknowledgements are due to the Dutch National Computing Facilities Foundation (NCF) for the use of the Cray C916 supercomputer at SARA, and to (in alphabetical order)

the Australian National University (Canberra),
Centre Charles Hermite (Nancy, France),
Citibank (Parsippany, NJ, USA),
CWI (Amsterdam, The Netherlands),
École Polytechnique/CNRS (Palaiseau, France),
Entrust Technologies Ltd. (Ottawa, Canada),
Lehigh University (Bethlehem, PA, USA),
the Magma Group of John Cannon at the University of Sydney,
the Medicis Center at École Polytechnique (Palaiseau, France),
Microsoft Research (Cambridge, UK),
the Putnams (Hudson, NH, USA),
Sun Microsystems Professional Services (Camberley, UK), and
Utrecht University (The Netherlands),

for the use of their computing resources.

References

1. L.M. Adleman. Factoring numbers using singular integers. In *Proc. 23rd Annual ACM Symp. on Theory of Computing (STOC)*, pages 64–71, ACM, New York, 1991.
2. D. Atkins, M. Graff, A.K. Lenstra, and P.C. Leyland. THE MAGIC WORDS ARE SQUEAMISH OSSIFRAGE. In J. Pieprzyk and R. Safavi-Naini, editors, *Advances in Cryptology – Asiacrypt '94*, volume 917 of *Lecture Notes in Computer Science*, pages 265–277, Springer-Verlag, Berlin, 1995.
3. Th. Beth, M. Frisch, and G.J. Simmons, editors. *Public–Key Cryptography: State of the Art and Future Directions*, volume 578 of *Lecture Notes in Computer Science.* Springer-Verlag, Berlin, 1992. Report on workshop at Oberwolfach, Germany, July, 1991.
4. Wieb Bosma and Marc-Paul van der Hulst. *Primality proving with cyclotomy.* PhD thesis, University of Amsterdam, December 1990.
5. Richard P. Brent. Some parallel algorithms for integer factorisation. *Proc. Europar'99 (Toulouse, Sept. 1999)*, volume 1685 of *Lecture Notes in Computer Science*, pages 1–22, Springer-Verlag, Berlin, 1999.
6. J. Brillhart, D.H. Lehmer, J.L. Selfridge, B. Tuckerman, and S.S. Wagstaff, Jr. *Factorizations of* $b^n \pm 1, b = 2, 3, 5, 6, 7, 10, 11, 12$ *up to high powers*, volume 22 of *Contemporary Mathematics.* American Mathematical Society, second edition, 1988.

7. J.P. Buhler, H.W. Lenstra, Jr., and Carl Pomerance. Factoring integers with the number field sieve. Pages 50–94 in [19].
8. S. Cavallar, B. Dodson, A. Lenstra, P. Leyland, W. Lioen, P. L. Montgomery, B. Murphy, H. te Riele, and P. Zimmermann. Factorization of RSA–140 using the number field sieve. In Lam Kwok Yan, Eiji Okamoto, and Xing Chaoping, editors, *Advances in Cryptology – Asiacrypt '99 (Singapore, November 14–18)*, volume 1716 of *Lecture Notes in Computer Science*, pages 195–207, Springer-Verlag, Berlin, 1999.
9. S. Cavallar. Strategies for filtering in the Number Field Sieve. Preprint, to appear in the Proceedings of ANTS-IV (*Algorithmic Number Theory Symposium IV*, Leiden, The Netherlands, July 2–7, 2000), *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 2000.
10. H. Cohen and A.K. Lenstra. Implementation of a new primality test. *Mathematics of Computation*, 48:103–121, 1987.
11. James Cowie, Bruce Dodson, R.-Marije Elkenbracht-Huizing, Arjen K. Lenstra, Peter L. Montgomery, and Jörg Zayer. A world wide number field sieve factoring record: on to 512 bits. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology – Asiacrypt '96*, volume 1163 of *Lecture Notes in Computer Science*, pages 382–394, Springer-Verlag, Berlin, 1996.
12. J.A. Davis, D.B. Holdridge, and G.J. Simmons. Status report on factoring (at the Sandia National Laboratories). In T. Beth, N. Cot, and I. Ingemarsson, editors, *Advances in Cryptology, Eurocrypt '84*, volume 209 of *Lecture Notes in Computer Science*, pages 183–215, Springer-Verlag, Berlin, 1985..
13. T. Denny, B. Dodson, A.K. Lenstra, and M.S. Manasse, On the factorization of RSA–120. In D.R. Stinson, editor, *Advances in Cryptology – Crypto '93*, volume 773 of *Lecture Notes in Computer Science*, pages 166–174, Springer-Verlag, Berlin, 1994.
14. B. Dixon and A.K. Lenstra. Factoring using SIMD Sieves. In Tor Helleseth, editor, *Advances in Cryptology, Eurocrypt '93*, volume 765 of *Lecture Notes in Computer Science*, pages 28–39, Springer-Verlag, Berlin, 1994.
15. Marije Elkenbracht-Huizing. *Factoring integers with the number field sieve.* PhD thesis, Leiden University, May 1997.
16. R.-M. Elkenbracht-Huizing. An implementation of the number field sieve. *Experimental Mathematics*, 5:231–253, 1996.
17. *Frequently Asked Questions about today's Cryptography 4.0.* Question 3.1.9, see `http://www.rsa.com/rsalabs/faq/html/3-1-9.html`.
18. R. Golliver, A.K. Lenstra, and K.S. McCurley. Lattice sieving and trial division. In Leonard M. Adleman and Ming-Deh Huang, editors, *Algorithmic Number Theory, (ANTS-I, Ithaca, NY, USA, May 1994)*, volume 877 of *Lecture Notes in Computer Science*, pages 18–27, Springer-Verlag, Berlin, 1994.
19. A.K. Lenstra and H.W. Lenstra, Jr., editors. *The Development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1993.
20. A.K. Lenstra, H.W. Lenstra, Jr., M.S. Manasse, and J.M. Pollard. The factorization of the Ninth Fermat number. *Mathematics of Computation*, 61(203):319–349, July 1993.
21. A.K. Lenstra and M.S. Manasse. Factoring by Electronic Mail. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology – Eurocrypt '89*, volume 434 of *Lecture Notes in Computer Science*, pages 355–371, Springer-Verlag, Berlin, 1990.
22. A.K. Lenstra and M.S. Manasse. Factoring with two large primes. In I.B. Dåmgard, editor, *Advances in Cryptology – Eurocrypt '90*, volume 473 of *Lecture Notes in Computer Science*, pages 72–82, Springer-Verlag, Berlin, 1991.

23. Arjen K. Lenstra and Eric R. Verheul. Selecting Cryptographic Key Sizes. In H. Imai and Y. Zheng, editors, *Public Key Cryptography*, volume 1751 of *Lecture Notes in Computer Science*, pages 446–465, Springer-Verlag, Berlin, 2000.
24. Peter L. Montgomery. Square roots of products of algebraic numbers. In Walter Gautschi, editor, *Mathematics of Computation 1943–1993: a Half-Century of Computational Mathematics*, pages 567–571. Proceedings of Symposia in Applied Mathematics, American Mathematical Society, 1994.
25. Peter L. Montgomery. A block Lanczos algorithm for finding dependencies over GF(2). In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology – Eurocrypt '95*, volume 921 of *Lecture Notes in Computer Science*, pages 106–120, Springer-Verlag, Berlin, 1995.
26. Peter L. Montgomery and Brian Murphy. Improved Polynomial Selection for the Number Field Sieve. Extended Abstract for the Conference on the Mathematics of Public-Key Cryptography, June 13–17, 1999, The Fields Institute, Toronto, Ontario, Canada.
27. Michael A. Morrison and John Brillhart. The factorization of F_7. *Bull. Amer. Math. Soc.*, 77(2):264, 1971.
28. Michael A. Morrison and John Brillhart. A method of factoring and the factorization of F_7. *Mathematics of Computation*, 29:183–205, January 1975.
29. B. Murphy. Modelling the Yield of Number Field Sieve Polynomials. J. Buhler, editor, *Algorithmic Number Theory, (Third International Symposium, ANTS-III, Portland, Oregon, USA, June 1998)*, volume 1423 of *Lecture Notes in Computer Science*, pages 137–151, Springer-Verlag, Berlin, 1998.
30. Brian Antony Murphy. Polynomial Selection for the Number Field Sieve Integer Factorisation Algorithm. PhD thesis, The Australian National University, July 1999.
31. J.M. Pollard. The lattice sieve. Pages 43–49 in [19].
32. Herman te Riele, Walter Lioen, and Dik Winter. Factoring with the quadratic sieve on large vector computers. *J. Comp. Appl. Math.*, 27:267–278, 1989.
33. R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM*, 21:120–126, 1978.
34. RSA Challenge Administrator. In order to obtain information about the RSA Factoring Challenge, send electronic mail to `challenge-info@rsa.com`. The status of the factored numbers on the RSA Challenge List can be obtained by sending electronic mail to `challenge-honor-rolls@majordomo.rsasecurity.com`. Also visit `http://www.rsa.com/rsalabs/html/factoring.html`.
35. A. Shamir. Factoring large numbers with the TWINKLE device. In C.K. Koc and C. Paar, editors, *Cryptographic Hardware and Embedded Systems (CHES)*, volume 1717 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1999.
36. Robert D. Silverman. The multiple polynomial quadratic sieve. *Mathematics of Computation*, 48:329–339, 1987.
37. Robert D. Silverman. Private communication.
38. URL: `http://www.bxa.doc.gov/Encryption/Default.htm`.

A Details of recent absolute and SNFS factoring records

Table 6. Absolute factoring records

# digits	129	130	140	155
method	QS	GNFS	GNFS	GNFS
code	Gardner	RSA–130	RSA–140	RSA–155
factor date	Apr 2, 1994	Apr 10, 1996	Feb 2, 1999	Aug 22, 1999
size of p, q	64, 65	65, 65	70, 70	78, 78
sieve time (in MIPS years)	5000	1000	2000	8400
total sieve time (in CPU years)	?	?	8.9	35.7
calendar time for sieving (in days)	∼270	120	30	110
matrix size	0.6M	3.5M	4.7M	6.7M
row weight	47	40	32	62
Cray CPU hours	n.a.	67	100	224
group	Internet	Internet	CABAL	CABAL

Table 7. Special Number Field Sieve factoring records

# digits	148[20]	167	180	186	211
code	2,512+	3,349−	12,167+	NEC	10,211−
factor date	Jun 15, 1990	Feb 4, 1997	Sep 3, 1997	Sep 15, 1998	April 8, 1999
size of p, q	49, 99	80, 87	75, 105	71, 73	93, 118
total sieve time (in CPU years)	340[a]	?	1.5	5.1	10.9
calendar time for sieving (in days)	83	?	10	42	64
matrix size	72K	?	1.9M	2.5M	4.8M
row weight	dense	?	29	27	49
Cray CPU hours	3[b]	?	16	25	121
group	Internet	NFSNET	CWI	CWI	CABAL

[a] MIPS years

[b] carried out on a Connection Machine