Revision 2010-01-12
Revision 2010-03-08

Factorization of RSA-170

Dominik Bonenberger and Martin Krone

Email d.bonenberger@ostfalia.de, ma.krone@ostfalia.de

Adress Department of Computer Science
Ostfalia University of Applied Sciences
Salzdahlumer Str. 46/48
38302 Wolfenbüttel, Germany

Abstract We present details on the factorization of RSA-170 – the smallest unfactored semiprime from the obsolete RSA Factoring Challenges. This number is the product of two 85 digit primes and was factored by the General Number Field Sieve in approximately six weeks. It differs from other similar factorizations as the polynomial selection for the Number Field Sieve was solely performed on a graphics card.

1 Introduction

The RSA Factoring Challenge [13] was announced in 1991 and served to measure the state of progress in practical cryptanalysis. It consisted of a list of semiprimes [12] ranging from 100 to 500 decimal digits, which were later replaced by numbers named after their binary length. Until 2007, when the challenges were declared inactive, eleven numbers (RSA-100 – RSA-160, RSA-576, RSA-640 and RSA-200) had been factored; the three smallest by a variation of the Quadratic Sieve [11] and the others by the General Number Field Sieve [3]. As of November 2009, the 563-bit number

$$\begin{aligned}\text{RSA-170} = {} & 2606262368413984492152987926667443219708592 5\\ & 3804864064161647851918599996285420693614502 8\\ & 3931914514618683512198164805919882053057222 9\\ & 7411647806509580983237733651071154575 9\end{aligned}$$

remained the smallest unfactored semiprime from this list.

2 Factorization of RSA-170

2.1 Polynomial selection

During the polynomial selection of the General Number Field Sieve, one usually seeks two polynomials $f(x)$ and $g(x)$ of degree d and 1, respectively. The degree of f is usually chosen as $d \sim (3 \ln n / \ln \ln n)^{\frac{1}{3}}$, where n is the number one wants to factor. For n = RSA-170, we looked for a polynomial of degree five.

Generally, the polynomial search is distributed among several computers. However, for RSA-170, we chose a different approach and used a GPU-implementation of Thorsten Kleinjung's improvements [2] of the ideas presented in [5], implemented in J. Papadopoulos' *msieve* program [7]. This implementation makes use of NVIDIA's parallel computing architecture CUDA [6] and allows for a remarkable speed-up of the first stage of the polynomial search. The second stage, however, is still performed on the CPU.
Polynomial search was started on a GeForce GTX 295 graphics card on November 18th, 2009, and was run for five days. After some sieving experiments, the following pair of polynomials was chosen:

$$\begin{aligned} f(x) = & 13860x^5 \\ & -6284825207568x^4 \\ & +11930382331307760430913x^3 \\ & +29541783562395905332471879106 3x^2 \\ & -10039016083291826481453029464394509407 7x \\ & +41083010956788616645493246462780305324133338 41 \end{aligned}$$

$$g(x) = 1298900392606034887x - 113462154421905191937841181688 2072$$

However, we are confident a better pair of polynomials could have been found if more time was spent on the search.

2.2 Sieving and post-processing

Sieving was started on November 24th on the following set of computers:

- 11 notebooks: Ten of them are equipped with an Intel P8700 dual-core CPU and one uses an Intel T6400 dual-core CPU.
- 2 desktop computers: One uses an Intel E4300 dual-core CPU and one runs on an Intel i7-940 quad-core CPU.

We did lattice sieving [10] on the algebraic side only for most special-q $\in [40e6, 117e6]$ using the software available from [1]. In order to keep track of the available ranges, the ones currently being sieved and to avoid duplicate work, the first author implemented a web-interface that allows exclusive reservations of unsieved ranges for individual PCs. These ranges were chosen to take between 2 and 56 hours to finish on one core of an Intel P8700 processor. Most of the sieving was carried out at the Department of Computer Science at the Ostfalia University. Only a small portion was done on the private computers of the two authors.
As of December 15th, 2009, a total of 168605286 relations had been collected of which 145922935 were unique. From this dataset, a 13484736 $\times$ 13484961 matrix with 1028735710 non-zero entries was optained. The Block Lanczos [4] implementation of J. Papadopoulos would take about 310 hours on four cores of an Intel i7-940 CPU to complete the iteration. Therefore, sieving was extended to December 22nd, 2009.
At that time, 169525698 unique and 34232149 duplicate relations had been collected. Storing all relations required about 21.1GB of disk space. As a result of the oversieving effort, filtering was now able to create a much smaller matrix of size 10462971 $\times$ 10463197 with 794960710 non-zero entries. The Block Lanczos

iteration used about 3.7GB of RAM and finished after 182 hours at 2:45am on December 29th, 2009. Shortly after, three square root jobs were successively started and each required about 90 minutes to complete.

On December 29th, 2009, at 07:21 GMT, we found that RSA-170 can be written as the product of two primes p and q:

$$\begin{aligned} \mathrm{p} = {} & 35864207304285014867998045872685204232914596105997816114023186063394845085804059396 38 \\ \mathrm{q} = {} & 72670290641070190788637977639239462641361378038569966703137089360022815822495874944 93 \end{aligned}$$

Primality of these factors was proved by the APRCL-method as implemented in PARI/GP [8]. The factorizations of $p \pm 1$ as well as $q \pm 1$ can be found in Appendix A.
In hindsight, the entire factorization would have finished slightly (say a few hours) earlier if we had ran Block Lanczos on the 13484736×13484961 matrix. But as it is rather reasonable to run this algorithm for seven or eight days instead of thirdteen, the authors feel confident that oversieving was the right choice.

3 Conclusions

Due to freely available sieving- and post-processing code that is capable of handling inputs such as RSA-170, general numbers with 170 (and possibly quite a few more) decimal digits can be factored on home computers only in relatively short time. For the post-processing, however, it seems reasonable to use a computer with a quad-core CPU which are much more common today than they were a few years ago.
The search for suitable polynomials on GPUs appears to be a good alternative to the regular approach on several CPUs. Powerful graphic cards with CUDA-support are available for as low as $30 USD and are already an order of magnitude faster for such jobs than modern CPUs. As even more powerful GPUs are announced every year, we assume that graphic cards will play a major role in the polynomial selection for future factorizations.

A Factorizations of $p \pm 1$ and $q \pm 1$

Finally, we give the factorizations of $p \pm 1$ as well as of $q \pm 1$. The factorization of $q - 1$ was found by an implementation of the Multiple Polynomial Quadratic Sieve, written by the second author.

$$\begin{aligned} p - 1 &= 2 \cdot 11^2 \cdot 14819920373671493747106630525902976955749833392809827112149718432371687813462977661 \\ p + 1 &= 2^2 \cdot 3 \cdot 1297 \cdot 230430527526889070084798547113114907690276258099458889818827541803774637037910601 \\ q - 1 &= 2^2 \cdot 11 \cdot 17 \cdot 133985428794214885836992816330212720274 89 \cdot \\ &\quad 72509970560983533614308804099133980792626 1 \end{aligned}$$

$$q+1 = 2 \cdot 3 \cdot 7 \cdot 17302450152635759711580470866485586343181280485373801595985021276195908529165684510 7$$

These factorizations clearly indicate that none of the prime factors could have been found by either Pollard's $p-1$ [9] or Williams' $p+1$ [14] algorithm.

Acknowledgments.

The authors are deeply grateful to J. Papadopoulos for his GPU-implementation of the polynomial selection for the General Number Field Sieve as well as his excellent post-processing code. Furthermore, we thank Prof. I. Mengersen for the use of her computing resources that allowed us to perform the sieving step in reasonable amount of time.

References

[1] GGNFS suite, `http://sourceforge.net/projects/ggnfs/`.

[2] T. Kleinjung, Polynomial Selection, Talk presented at the CADO workshop on integer factorization (2008), `http://cado.gforge.inria.fr/workshop/slides/kleinjung.pdf`.

[3] A.K Lenstra and H.W Lenstra, Jr. (editors), *The development of the number field sieve*, Springer, Berlin, 1993.

[4] P.L. Montgomery, *A Block Lanczos Algorithm for Finding Dependencies over GF(2)*, Advances in Cryptology – EUROCRYPT '95, Springer, Berlin, 1995, 106–120.

[5] B.A. Murphy, *Polynomial Selection for the Number Field Sieve Integer Factorisation Algorithm*, Ph.D. thesis, The Australian National University, 1999.

[6] NVIDIA CUDA Zone, `http://www.nvidia.com/object/cuda_home.html`.

[7] J. Papadopoulos, *msieve*, `http://sourceforge.net/projects/msieve/`.

[8] PARI/GP, `http://pari.math.u-bordeaux.fr/`.

[9] J.M. Pollard, *Theorems on Factorization and Primality Testing*, Proc. Cambridge Phil. Soc. 76 (1974), 521–528.

[10] J.M. Pollard, The lattive sieve, 43–49 in [3].

[11] R.D. Silverman, *The Multiple Polynomial Quadratic Sieve*, Mathematics of Computation 48 (1987), 329–339.

[12] The RSA Challenge Numbers, `http://en.wikipedia.org/wiki/RSA_numbers`.

[13] The RSA Factoring Challenge, `http://www.rsa.com/rsalabs/node.asp?id=2092`.

[14] H.C. Williams, *A p+1 Method of Factoring*, Mathematics of Computation 39 (1982), 225–234.