Factorization of RSA-180

S. A. Danilov, I. A. Popovyan
Moscow State University, Russia

May 9, 2010*

Abstract

We present a brief report on the factorization of RSA-180, currently smallest unfactored RSA number. We show that the numbers of similar size could be factored in a reasonable time at home using open source factoring software running on a few Intel Core i7 PCs.

1 Introduction

In 1991 RSA Labs published a list of semiprime numbers of different size and announced a reward for their factorization. The numbers from that list called RSA numbers became a measure of the quality of the factorization tools.

We began working on our factorization project on November 2009. We started with the smallest unfactored RSA number for that moment, RSA-170, written in 170 decimal digits. The factorization was finished on 31 December 2009, then we found out that Dominik Bonenberger and Martin Krone [1] were ahead of us for two days and had already presented the RSA-170 prime decompostion. Meanwhile the new world record in factorization was set [2] – the international team of scientists managed to factor the RSA-768, a 232 decimal digits long RSA number.

On January 2010 after a short break we decided to continue the project and took the number

$$\begin{aligned}\text{RSA-180} = {} & 191147927718986609689229466631454649812986246\\ & 276667354864188503638807260703436799058776201\\ & 365135161278134258296128109200046702912984568\\ & 752800330221777752773957404540495707851421041,\end{aligned}$$

the next smallest RSA number with unknown factorization.

*Revised at 13.04.2010

2 Factorization of RSA-180

Our tools for factorization are essentially based on two open source implementations of General Numebr Field Sieve (GNFS) algorithm – the community maintained *GGNFS* suite [3] and Jason Papadopoulos's *msieve* [4]. We adopted them for parallel computations using MPI and tested on two independent platforms:

1. 24 virtual processors of 3 Intel Core i7 4GHz PCs with 6Gb RAM connected over Gigabit Etherenet and running open source MPI implementation MPICH2 [5],
2. 100 virtual processors of Moscow State University supercomputer SKIF MSU 'Chebyshov' [6].

The GNFS algorithm has 4 major steps: polynomial selection, sieving and post-processing, solving sparse linear system, square root extraction. Let us make a few comments on each of them in our parallel GNFS implementation.

2.1 Polynomial selection

Although J. Papadopoulos implemented a very nice polynomial selection algorithm on nVidia CUDA architecture we decided to parallelize the J.Franke and T. Kleinjung's *pol5* tool from the *GGNFS* suite so we were able to run it on both platforms.

The parallelization was trivial, basicly we've rewritten the perl script coming with the *GGNFS* suite and equally divided the highest polynomial coefficient search interval between parallel processors. This technique is not very efficient since the 'good' polynomials are not equally distributed on each subinterval and therefore many of the processors are underloaded. However this was a good start and allowed us to find after 12 days on platform 1 and 13 days on platform 2 a few polynomial pairs of almost equal quality, the best of which was

$$\begin{aligned} a(x) &= 7563318480x^5 \\ &\quad +17595930596689122x^4 \\ &\quad -396454273559134931547 51x^3 \\ &\quad -17198395720703347534221482658x^2 \\ &\quad +17747480317181638340745152993046310x \\ &\quad -71799751714919877679765254028304 1093812, \\ r(x) &= 6387961227166713233237x \\ &\quad -75950610864009984992932215005522 75. \end{aligned}$$

with skew 1090750.

2.2 Sieving and post-processing

A modified version of *GGNFS*'s lattice siever *lasieve4I15* was used for sieving stage. The parallelization was again very simple – the special-q interval $[5135 \cdot$

$10^4, 35135 \cdot 10^4]$ was equally splitted between the processors. Both large prime bounds were set to 2^{31}. The stage took 63 days on platform 1 and 26 days on platform 2 and resulted in 192 and 198 million of partial relations stored in about 21,5 Gb file.

The post-processing stage was done by the latest *msieve* and after 2 hours on single processor of platform 1 and 4 hours on single processor of platform 2 reduced the problem on platform 1 to the linear system of size 21414654×21414832 and weight 2028585084. There was a small oversieving on platform 2 resulted in slightly different matrix size 20572576×20572753 and weight 1955919707.

2.3 Solving linear system

We implemented a parallel version of P. Montgomery block Lanczos algorithm [7] based on the implementation from the *msieve*. It was taking some time to write it and remembering the previous experience with the RSA-170 we decided to launch the matrix step on the single processor of platform 1 using the 4-thread *msieve* while debugging the parallel one on the platform 2.

The matrix step finished in 33 days on platform 1 and gave us 31 nontrivial dependencies. The parallel version on platform 2 finished in 24 days and showed that the implementation wasn't good enough since the expected speedup was a square root of a number of processors. Currently we are working on the further speedup of the parallel version of Montgomery's block Lanczos algorithm.

2.4 Square root extraction

The square root extraction stage was done purely by *msieve* and took 10 hours on the single processor of platform 1 and 19 hours on the single processor of platform 2.

After all the stages we found that RSA-180 $= pq$, where

$$p = \begin{array}{l} 400780082329750877952581339104100572526829317 \\ 815807176564882178998497572771950624613470377 \end{array}$$

and

$$q = \begin{array}{l} 476939688738611836995535477357070857939902076 \\ 027788232031989775824606225595773435668861833. \end{array}$$

The factorizations of $p \pm 1$ and $q \pm 1$ can be found in Appendix A.

3 Conclusions

We used freely available open source tools to factor RSA-180 on 3 Intel Core i7 PCs and the supercomputer SKIF MSU 'Chebyshov'. The whole process took a comparable time on both platforms: although the sieving stage was more efficient on supercomputer due to a bigger number of processors, all the other stages were more efficient on the Intel Core i7 PC. This means that nowdays

for about $3000 one can buy a computational power comparable to the small supercomputer and factor numbers of about 180 decimal digits in 3 months.

Acknowledgements

The authors are grateful to J. Papadopoulos for the brilliant implementation of the GNFS in his *msieve* software: it was a pleasure to read and understand your code. We thank all the maintainers of the *GGNFS* and Chris Monico personally for giving everybody a good starting point in all factorization projects. The second author also thanks the Research Computing Center of M.V. Lomonosov Moscow State University for the access to the supercomputer SKIF MSU 'Chebyshev'.

References

[1] Dominik Bonenberger, Martin Krone, *Factorization of RSA-170*, `http://public.rz.fh-wolfenbuettel.de/~kronema/pdf/rsa170.pdf`

[2] Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen Lenstra, Emmanuel Thome, Joppe Bos, Pierrick Gaudry, Alexander Kruppa, Peter Montgomery, Dag Arne Osvik, Herman te Riele, Andrey Timofeev and Paul Zimmermann, *Factorization of a 768-bit RSA modulus*, `http://eprint.iacr.org/2010/006`

[3] C. Monico et al., *GGNFS suite*, `http://sourceforge.net/projects/ggnfs/`

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

[5] *MPICH2* `http://www.mcs.anl.gov/research/projects/mpich2/`

[6] *Moscow State University supercomputer SKIF MSU 'Chebyshov'* `http://parallel.ru/cluster/` (in russian)

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

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

The factorizations of $p \pm 1$ and $q \pm 1$ are as follows

$$
\begin{aligned}
p-1 &= 2^3\times \\
&\quad 74051\times \\
&\quad 571409\times \\
&\quad 11839630232137682225268639851533677805502814092536714550 4 \\
&\quad 7486308276274179178583, \\
p+1 &= 2\times \\
&\quad 3\times \\
&\quad 7\times \\
&\quad 11\times \\
&\quad 631\times \\
&\quad 757661\times \\
&\quad 80415217\times \\
&\quad 43164234793001183523080176706701\times \\
&\quad 52275425687740552060697206765047 3564077, \\
q-1 &= 2^3\times \\
&\quad 277\times \\
&\quad 751\times \\
&\quad 47779\times \\
&\quad 88291435965578199481003\times \\
&\quad 67935712535668043985232693389634831578450559709946498371, \\
q+1 &= 2\times \\
&\quad 3\times \\
&\quad 13\times \\
&\quad 29\times \\
&\quad 40780001\times \\
&\quad 9488340193\times \\
&\quad 36601837387\times \\
&\quad 14887798995502551010395541052456872321039885951355136372877.
\end{aligned}
$$