Baltic J. Modern Computing, Vol. 5 (2017), No. 3, 269-274\
http://dx.doi.org/10.22364/bjmc.2017.5.3.02

Efficiency of RSA Key Factorization by Open-Source Libraries and Distributed System Architecture

Edgar Jan VUICIK, Dmitrij ŠEŠOK, Simona RAMANAUSKAITĖ

Vilnius Gediminas Technical University, Sauletekio al. 11, LT–10223, Vilnius, Lithuania

edgar-jan.vuicik@stud.vgtu.lt, dmitrij.sesok@vgtu.lt,
simona.ramanauskaite@vgtu.lt

Abstract. The security of the RSA algorithm relies on the difficulty to factorize large numbers. However the computational power of information technologies is increasing all the time, while open-source factoring libraries are developed at similar pace. Therefore, the possibility to factorize large numbers also increases. In this paper we analyze the efficiency of open-source libraries to factor RSA numbers by using it in computer cluster to decrease the calculation time. To achieve this, we analyze efficiency of Msieve, GGNFS and CADO-NFS libraries for 81 decimal digit number factorization with varying number of cluster nodes (cores). By choosing the best solution (Msieve library with GGNFS library integration for sieving method) we analyze the possibility to factorize different size RSA numbers, and discuss exact conditions to achieve it.

Keywords: Factoring, RSA, Msieve, GGNFS, CADO-BFS, efficiency.

1. Introduction

Factoring of large numbers is one of the oldest branches of number theory. It gained its importance and popularity after RSA (Rivest et al., 1978) was proposed in 1977. RSA is a public-key cryptosystem where two large primary numbers are multiplied to get a part of a key. Different opinions on the best way to break the RSA exist. D. R. Brown (2005) claims that it may be as difficult as factoring while Boneh and Venkatesan (1998) believe it can be done easier than factoring. Despite the contradiction, the possibility of large number factorization is still one of the main measures to evaluate the security of RSA key length.

Currently, the largest factored RSA number is RSA-768 (768 bit or 232 decimal digit lengths), factored in 2009 by Kleinjung (2010). The factorization of RSA-768 number took more than two years and 80 single core 2.2 GHz AMD Opteron processors with 2 GB RAM. It would be difficult to achieve the same results given domestic computational power, however the RSA-155 (512 bit length) is much easier to deal with. Open-source projects as well as computer clusters and distributed computing can be adopted for the task. The aim of this paper is to evaluate the efficiency of open source factoring libraries to factor RSA numbers in computer cluster. This will show the level of vulnerability of

www.manaraa.com

RSA against modern approaches with physical resources similar to average domestic conditions.

In this paper we have two research questions and consequentially two different approaches:

- Which open-source factoring library is the most suitable for large number factoring in computer cluster?
- How long it would take to factorize up to 155 decimal digit RSA numbers by using 8 computer cluster (32 cores)?

2. Factoring of Large Numbers

Factoring of large numbers can be divided into two broad categories, according to the purpose (Duta et al., 2016):

- Special-purpose algorithms. Here algorithm runtime depends on properties of the number being factored, or on its unknown factors, such as size, special form, etc. Trial Division (Bressoud, 2012), Fermat's (Lehman, 1974), Euler's (Oystein, 1976), Pollards p-1 (Pollard, 1974), Pollard's Rho (Pollard, 1975), William's p+1 (Williams, 1982), Elliptic curve method (ECM) (Lenstra, 1985) algorithms belong to this category.
- General-purpose algorithms. Here, the speed does not depend on the size of the prime factors, the number of prime factors or on the form of the number. Algorithms of this category are Dixon's (Dixon, 1981), continued fraction factorization (CFRAC) (Morrison, 1975), quadratic sieve (Pomerance, 1984), number field sieve (NFS) (Lenstra, 1993), special number field sieve (SNFS) (Silverman, 2007), general number field sieve (GNFS) (Pomerance, 2008), Shanks square forms factorization algorithm (SQUFOF) (Shanks, 1975).

While Special-purpose algorithms are useful for solving specific problems, the general-purpose algorithms are more suitable for RSA number factoring. As seen in results of Duta (2016) experiment, the number field sieve algorithms (NFS, SNFS, GNFS) are the most efficient in the category of general-purpose algorithms – the speed is more than twice as fast compared to other algorithms.

For the number field sieve algorithm, different open-source libraries exist. Currently three most popular ones are: Msieve (Papadopoulos, 2014), GGNFS (Monico, 2005) and CADO-NFS (Bai et al.). All these libraries have all five main methods for factorization: polynomial selection, sieving, filtering, linear algebra, square root. Msieve was used to factor 1061-bit Mersenne number (Childers, 2012), GGNFS – RSA-180 number (Danilov et al., 2010), CADO-NFS – SRA-704 number (Bai et al., 2012). As well it is worth mentioning that CADO-NFS is optimized for operation in computer clusters.

Comparative information of these three libraries very limited, as Winograd uses Msieve and GGNFS to extend CrypTool functionality and states the suitability of these libraries for the purpose, excluding any information about the efficiency. Valenta (2016) explains why CADO-NFS and Msieve are suitable to be used on the Amazon Elastic Compute Cloud platform, yet again - without efficiency testing. According to the results of our method research - there are no direct Msieve, GGNFS and CADO-NFS efficiently comparison experiments. Therefore, it is unclear which library is the most suitable for distributed factoring of large numbers.

www.manaraa.com

3. Efficiency Comparison of Msieve, GGNFS and CADO-NFS Libraries

For evaluation of the efficiency of number field sieve libraries, two primary numbers *P* of 40 and *Q* of 41 decimal digits were multiplied and analyzed. As a result - number *N* of 81 decimal digit was obtained.

```
P=6075380529345458860144577398704761614649
Q=66610666966686667666666656664666366626661
N=4046851491361229174697420993794004015930049965312154307580
97470298767813731556989
```

The number *N* was factorized by using computer cluster named "Vilkas" which is located in Vilnius Gediminas Technical University. For the experiment the cluster was configured to use 8 Intel® Core™ i7-860 @ 2.80 GHz with 4 cores, 4 GB DDR3-1600 RAM, 500 GB HDD SATA each. To define how the efficiency depends on the number of parallel processes, we have analyzed six situations with 1, 2, 4, 8, 16 and 32 processes. In each of the situation, three analogues are tested: Msieve (A1), Msieve + GGNFS (A2), CADO-NFS (A3). GGNFS was not analyzed as it is, only as a combination with Msieve. It due to the complications to use all five GGNFS library methods for factoring in a cluster. In tested factoring solution A2 only GGNFS sieving method was used, while the rest four methods were implemented by Msieve library. We calculated the elapsed time for each of factoring steps (see table 1) and the overall time (see Fig. 1) for the number *N* factoring.

Table 1. Time of Msieve, Msieve+GGNFS and CADO-NFS libraries in each factoring method when different number of processes is used.

Pro-ces-ses	Polynomial selection, s			Sieving, s			Filtering, s			Linear algebra, s			Square root, s		
	A1	A2	A3	A1	A2	A3	A1	A2	A3	A1	A2	A3	A1	A2	A3
1	240	252	44	14280	540	631	180	36	12	60	1	27	60	1	3
2	240	36	48	8280	360	461	180	36	12	60	1	21	60	1	5
4	240	36	51	5520	216	241	180	36	12	60	1	13	60	1	3
8	240	36	48	3480	144	473	180	36	13	60	1	13	60	1	3
16	240	36	48	2100	90	144	180	36	10	60	1	13	60	1	3
32	240	36	48	1080	36	139	180	36	11	60	1	12	60	1	6

The results prove the Msieve method is not optimized for sieving method, therefore the combination of Msieve and GGNFS shows better results. Meanwhile the comparison between Msieve+GGNFS and CADO-NFS shows similar results. However, combination of Msieve and GGNFS is more stable, as in some situations (number of processes) the factoring time increases unexpectedly. We assume it might be related to poor task distribution for cluster nodes as the jumps of factoring time are noticed when more than one computer is used.

www.manaraa.com

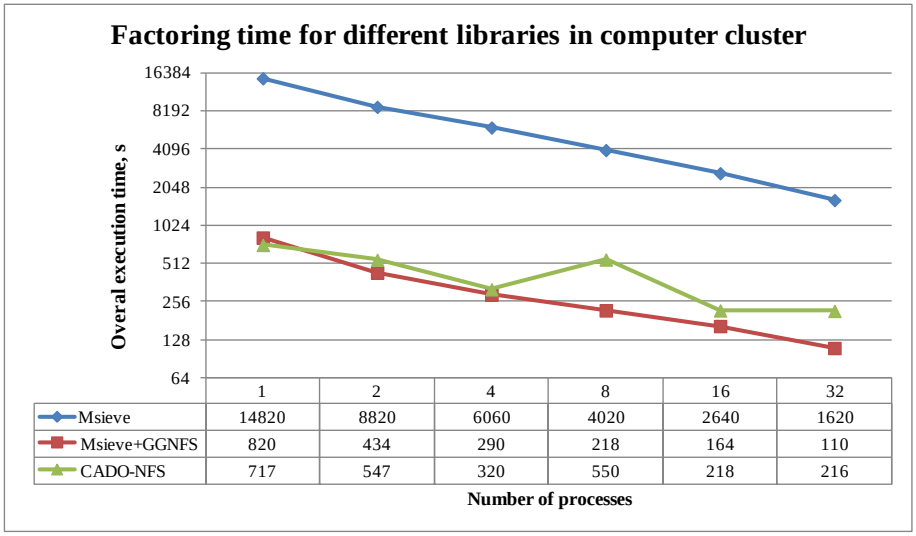

Fig. 1. Comparison of factoring time of different libraries when different number of processes is used

4. Factoring of RSA Keys with Msieve and GGNFS for Sieving

In efficiency research we executed experiments with 81 decimal digit number *N*. An interesting discovery is that the Msieve library can be used for larger number factorization along with GGNFS for sieving. In order to experiment with larger numbers we used eight numbers from RSA-100 to RSA-155 for factoring. Each of those numbers were factored in computer cluster "Vilkas" with 32 cores (the same as in previous experiments) and execution time was estimated for each method.

The experiment results are shown in Table 2 and illustrate that the sieving method plays the key role in the factoring speed. It also shows that all methods require more time to execute by increasing the size of factored number. By analyzing the overall factoring time, it exponentially increases according to the length of the number (see Fig. 2).

Table 2. Factoring time in 32 core (8 node) computer cluster achieved by using with Msieve + GGNFS libraries

RSA number	Polynomial selection, h	Sieving, h	Filtering, h	Linear algebra, h	Square root, h
RSA-100	0.01	0.09	0.04	0.03	0.04
RSA-110	0.01	0.34	0.07	0.06	0.02
RSA-120	0.03	0.70	0.10	0.27	0.06
RSA-129	0.11	1.63	0.18	0.66	0.10
RSA-130	0.14	1.83	0.18	0.90	0.13
RSA-140	0.34	6.85	0.27	2.28	0.41
RSA-150	0.79	21.58	0.55	11.23	0.48
RSA-155	1.34	42.61	0.72	10.35	0.86

www.manaraa.com

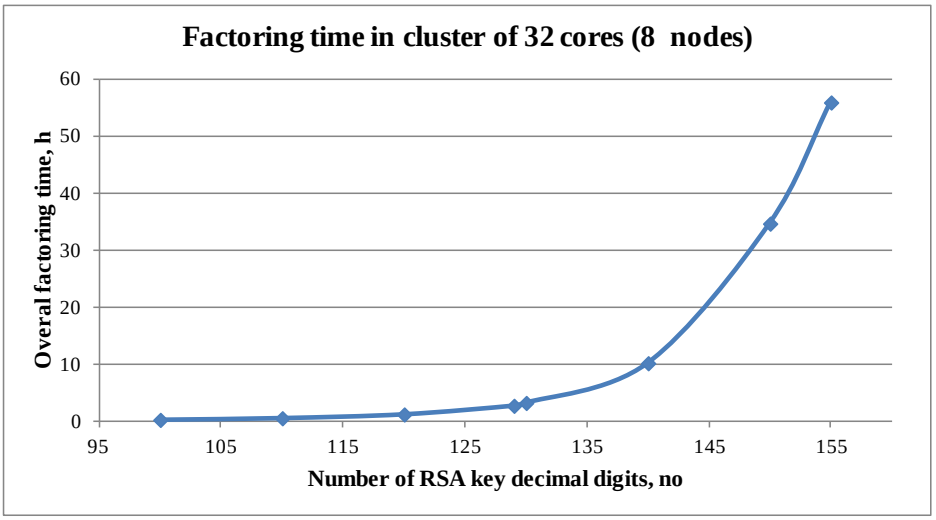

Fig. 2. Factoring time for different length of RSA numbers by using Msieve and GGNFS libraries in 32 core (8 node) computer cluster "Vilkas"

Important notice – the RSA number of 155 decimal digits, or 512 bit can be factorized in less than 60 hours by using computer cluster of 8 Intel® Core™ i7-860 @ 2.80 GHz computers. This proves the RSA-512 is critically unsafe as the key can be broken within three days. If we would follow the exponential curve and would analyze what would be the factoring time for RSA-1024 number we would need more than 15000 years. This RSA number is not factorized yet by any other author and could not be factored by our computer cluster in near future as well.

Conclusions

The research cases of large RSA number factorization were performed by using number field sieve algorithms. In most of cases, open-source libraries were used to factor the RSA number. As the computational capacity of modern computers is increasing and open-source libraries are optimized, the possibility to factor large RSA-512 numbers by an individual person, rather than a scientific laboratory, is increasing rapidly.

By adapting existing open-source libraries for large number factorization in computer cluster we noticed Msieve is lacking efficiency in sieving stage. However, by integrating Msieve with GGNFS for sieving stage, the factoring time decreased up to 20 times. However, the difference is decreasing along with the increasing core number. This shows the computer cluster can be used in order to decrease the factoring time, however it is not able to provide linear increase.

Used combination of Msieve and GGNFS libraries in 32 core computer cluster is able to factor RSA number of 512 bit in less than 60 hours. This time is too short to consider RSA-512 to be safe or even decent for usage. By increasing the length of the number, the factoring time increases exponentially. The RSA-1024 is safe against the attacks against within our used solution. The fact that RSA-1024 was not factored yet by any other author proves the key length is currently sufficient.

www.manaraa.com

References

Bai, S., Filbois, A., Gaudry, P., Kruppa, A., Morain, F., Thomé, E., Zimmermann, P. CADO-NFS, Crible Algébrique: Distribution, Optimisation-Number Field Sieve.

Bai, S., Thomé, E., Zimmermann, P. (2012). Factorisation of RSA-704 with cado-nfs.

Boneh, D., Venkatesan, R. (1998). Breaking RSA may not be equivalent to factoring. Advances in Cryptology—EUROCRYPT'98, 59-71.

Bressoud, D. (2012). Factorization and primality testing. Springer Science & Business Media.

Brown, D. R. (2005). Breaking RSA May Be As Difficult As Factoring. IACR Cryptology ePrint Archive, 2005, 380.

Childers, G. (2012). Factorization of a 1061-bit number by the Special Number Field Sieve. IACR Cryptology ePrint Archive, 2012, 444.

Danilov, S. A., Popovyan, I. A. (2010). Factorization of RSA-180. IACR Cryptology ePrint Archive, 2010, 270.

Dixon, J. D. (1981). Asymptotically fast factorization of integers. Mathematics of computation, 36(153), 255-260.

Duta, C. L., Gheorghe, L., Tapus, N. (2016, July). Framework for evaluation and comparison of integer factorization algorithms. In SAI Computing Conference (SAI), 2016 (pp. 1047-1053). IEEE.

Kleinjung, T., Aoki, K., Franke, J., Lenstra, A., Thomé, E., Bos, J.,.., Te Riele, H. (2010, August). Factorization of a 768-bit RSA modulus. In CRYPTO 2010 (Vol. 6223, pp. 333-350). Springer Verlag.

Lehman, R. S. (1974). Factoring large integers. Mathematics of Computation, 28(126), 637-646.

Lenstra, A. K., Lenstra Jr, H. W., Manasse, M. S., Pollard, J. M. (1993). The number field sieve. In The development of the number field sieve (pp. 11-42). Springer Berlin Heidelberg.

Lenstra, H. W. (1985). Elliptic curve factorization personal communication via samuel wagstaff jr.

Monico, C. (2005). GGNFS-A number field sieve implementation. http://www. math. ttu. edu/cmonico/software/ggnfs/.

Morrison, M. A., Brillhart, J. (1975). A method of factoring and the factorization of F_7. Mathematics of computation, 29(129), 183-205.

Oystein, O. (1976). Euler's Factorization Method. Number Theory and Its History, 59-64.

Papadopoulos, J. (2014). Msieve (2010). Project site: http://msieve. sourceforge. net.

Pollard, J. M. (1974, November). Theorems on factorization and primality testing. In Mathematical Proceedings of the Cambridge Philosophical Society (Vol. 76, No. 3, pp. 521-528). Cambridge University Press.

Pollard, J. M. (1975). A Monte Carlo method for factorization. BIT Numerical Mathematics, 15(3), 331-334.

Pomerance, C. (1984, April). The quadratic sieve factoring algorithm. In Workshop on the Theory and Application of of Cryptographic Techniques (pp. 169-182). Springer, Berlin, Heidelberg.

Pomerance, C. (2008). A tale of two sieves. Biscuits of Number Theory, 85, 175.

Rivest, R. L., Shamir, A., Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2), 120-126.

Shanks, D. (1975). Analysis and improvement of the continued fraction method of factorization. American Mathematical Society Notices, 22, 1.

Silverman, R. D. (2007). Optimal parameterization of SNFS. Journal of Mathematical Cryptology JMC, 1(2), 105-124.

Valenta, L., Cohney, S., Liao, A., Fried, J., Bodduluri, S., Heninger, N. (2016, February). Factoring as a Service. In International Conference on Financial Cryptography and Data Security (pp. 321-338). Springer, Berlin, Heidelberg.

Williams, H. C. (1982). A $p+1$ method of factoring. Mathem. of Computation, 39(159), 225-234.

Winograd, T. CrypTool Number Field Sieve Extensions.

Received August 27, 2017, accepted September 4, 2017

www.manaraa.com

Reproduced with permission of copyright owner.
Further reproduction prohibited without permission.

www.manaraa.com