COMPARISON OF ALGORITHMS FOR FACTORIZATION OF LARGE NUMBERS HAVING SEVERAL DISTINCT PRIME FACTORS

Tomasz Kijko, Michał Wroński

Institute of Mathematics and Cryptology, Cybernetics Faculty,
Military University of Technology, Warsaw, Poland
E-mail: [tkijko,mwronski]@wat.edu.pl

Abstract. We present analysis of security of the most known assymetric algorythm RSA and its modern version MultiPrime RSA. We focused on more precisious estimations of time complexity of two factorization algorithms: Elliptic Curve Method and General Number Field Sieve. Additionally for the MultiPrime RSA algorithm we computed the maximal number of prime factors for given modulus length which does not decrease the security level.
Keywords: factorization, MultiPrimeRSA ECM GNFS smoothness

1. Introduction

Many public key algorithms base on the factoring problem. The fastest known algorithm of factorization for large numbers is GNFS (General Number Field Sieve). The expected running time of the algorithm depends only on the size of the number and it is not sensitive for small factors (small in comparison with the size of the number).

Analysis of security the public key algorithms often base on the expected time of running GNFS. If the number has more factors than two, then the other factorization algorithms may be faster, like ECM (Elliptic Curve Method).

In this article we will show a comparison between expected time of running ECM and GNFS depending on the size of the number and the number of prime factors. However, parallel hardware implamentations are considered (see [8]) we focused on single processor implementations.

This paper devoted to the "Cryptology and Cyberdefence" was supported by The National Centre for Research and Development under the project number DOBR/0001/RON/ID1/2013/01.

2. GNFS (General Number Field Sieve)

2.1. Introduction to GNFS

The biggest numbers being the product of two distinct prime factors have been factorized during the RSA Challenge. Many of these numbers have been factorized using very advanced solutions based on parallel architectures what in the result makes hard to compare the time of factoring these numbers.

We used the data included on the website of RSA Challenge, so we were able to estimate the number of all instructions needed to make a sieve step in GNFS Algorithm.
We were not able to estimate the number of instructions needed to make all steps of GNFS algorithm but we knew that the sieve step is the longest of all steps in GNFS algorithm and takes from 60% to 80% of amount of time. In these circumstances we estimated the running time of GNFS by the time of sieving step.

2.2. The approximating function

It is well known that expected running time of GNFS algorithm is subexponential given by the formula:

$$O\left(e^{\sqrt[3]{\frac{64}{9}}(\ln n)^{\frac{1}{3}}\cdot(\ln(\ln n))^{\frac{2}{3}}}\right).$$

Because of using the big O notation, we are not able to estimate expected time so properly if we do not know the leading coefficient of

$$e^{\sqrt[3]{\frac{64}{9}}(\ln n)^{\frac{1}{3}}\cdot(\ln(\ln n))^{\frac{2}{3}}}.$$

So we are obligated to looking for a function:

$$A\cdot e^{\sqrt[3]{\frac{64}{9}}(\ln n)^{\frac{1}{3}}\cdot(\ln(\ln n))^{\frac{2}{3}}} = e^{\sqrt[3]{\frac{64}{9}}(\ln n)^{\frac{1}{3}}\cdot(\ln(\ln n))^{\frac{2}{3}}+a},$$

where $a = \ln A$.

To find out the constant a, we make the following transformations: Let $n_1, n_2, \ldots n_k$ be RSA modules. For each n_i we denote its factorization time by y_i. Then we can denote: $y_i = e^{\sqrt[3]{\frac{64}{9}}(\ln n_i)^{\frac{1}{3}}\cdot(\ln(\ln n_i))^{\frac{2}{3}}+a}$. Then:

$$\sqrt[3]{\frac{64}{9}}(\ln n_i)^{\frac{1}{3}}\cdot(\ln(\ln n_i))^{\frac{2}{3}} + a = \ln y_i.$$

We are looking for the minimum of the function:

$$f(a) = \sum_{i=1}^{k} \left(\sqrt[3]{\frac{64}{9}} (\ln n_i)^{\frac{1}{3}} \cdot (\ln (\ln n_i))^{\frac{2}{3}} + a - \ln y_i \right)^2 .$$

Afler substituting $\sqrt[3]{\frac{64}{9}} (\ln n_i)^{\frac{1}{3}} \cdot (\ln (\ln n_i))^{\frac{2}{3}}$ by x_i we have got:

$$f(a) = \sum_{i=1}^{k} (x_i + a - \ln y_i)^2.$$

Using the data from RSA Challenge presented in Table 1 we estimated $a = -27.5622$.

Table 1: Numbers broken in RSA Challenge (Based on [4] and [6])

Name of the number	Number of decimal digits	Number of bits	Number of instructions	Estimated time on 3.3 GHz[s]
RSA 140	140	465	$2.72 \cdot 10^{16}$	$8.25 \cdot 10^{6}$
RSA 155	155	515	$2.52 \cdot 10^{17}$	$7.65 \cdot 10^{7}$
RSA 160	160	532	$1.88 \cdot 10^{16}$	$5.70 \cdot 10^{6}$
RSA 576	173	576	$3.82 \cdot 10^{18}$	$1.16 \cdot 10^{9}$
RSA 640	193	640	$2.08 \cdot 10^{18}$	$6.31 \cdot 10^{8}$
RSA 768	231	768	$1.39 \cdot 10^{20}$	$4.20 \cdot 10^{10}$

We also made computations[1] of factoring large numbers with the GNFS algorithm using Cado-NFS[2] implementation. The results are based on factorization numbers from 260 to 350 bits.

The results we got are very similar to these that we got from RSA Challenge. We obtained $a = -26.7220$. The difference is small and is caused by taking much smaller numbers to computation than in RSA Challenge and because we had results of a sieving step from RSA Challenge only.

[1] Computations were made on 3.3 GHz chipset.

[2] See [7].

3. ECM (Elliptic Curve Method)

3.1. Introduction to ECM

The Elliptic Curve Method is based on the idea used firstly in Pollard $p-1$ algorithm.
Let's consider elliptic curve over the ring $\mathbb{Z}/n\mathbb{Z}$:

$$C_{a,b} \ : \ Y^2Z = X^3 + aXZ^2 + bZ^3,$$

where $GCD\left(n,\ 4a^3+27b^2\right)=1$.
Let's define the point of this elliptic curve in projective coordinates:

$$P = (x, y, z) \in C_{a,b}(\mathbb{Z}/n\mathbb{Z}).$$

Let's denote the smallest prime factor of factorized number n by p.

From Hasse's theorem it is well known that $|N_p - (p+1)| < 2\sqrt{p}$, where N_p is the order of group of F_p rational points on $C_{a,b}$ curve. Then we are supposed to choose the number B and to compute $k = \prod_{r=2}^{B} r^{e(r)}$, where $e(r) = \lfloor \log_{\mathrm{r}}(p+2\sqrt{p}+1) \rfloor$ and the point $(x_k, y_k, z_k) = [k]P \ (mod\ n)$.

Finally, if $N_p|k$, then $p|z_k$ and the divisor of number N may be found by counting $GCD\,(z_k, n)$.

From the facts described above, it is obvious that the algorithm will be successful with big probability if we find the elliptic curve with B – smooth order.

3.2. ECM algorithm complexity

Estimation of ECM expected running time is not trivial. There are used several hypothesis, some of them partially proved.

In the first article about Elliptic Curve Method (see [14]), Hendrik W. Lenstra described estimation of expected running time. In this article we try to show the estimation in slightly different way.

Hypotesis 1 (Sato – Tate Hypothesis). *By e_p we denote deviation from the center of the interval which consists the order of elliptic curve. By $a_p = \frac{e_p}{2\sqrt{p}}$ we denote normalized deviation. Then $a_p \in \langle -1, 1\rangle$ and density of probability distribution of finding out the elliptic curve with given deviation t is equal:*

$$\frac{2}{\pi}\sqrt{1-t^2}.$$

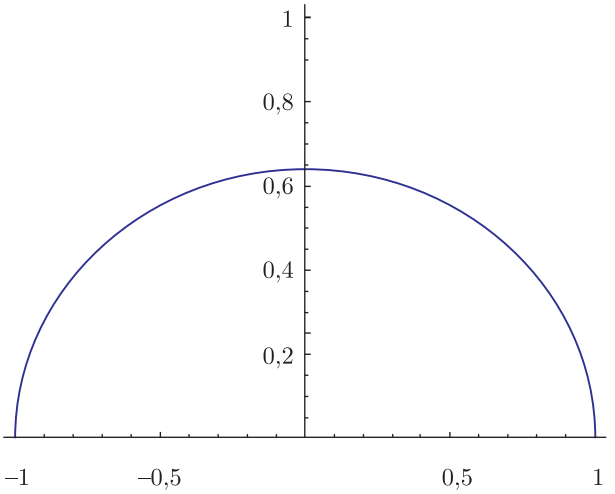

Figure 1. The distribution of probability density of finding elliptic curve with given normalized deviation

This hypothesis has been proved for almost all elliptic curve without complex multiplication.

Assuming the hypothesis as true, we get the following conclusion:

The probability of finding out the elliptic curve with the given deviation $t \in \langle -\frac{1}{2}, \frac{1}{2} \rangle$ is equal to the definite integral: $\int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{2}{\pi}\sqrt{1-t^2}\, dt \approx 0.608998$. Moreover, we can assume uniform distribution in this interval.

3.2.1. The number of B – smooth numbers in the given interval and consequences

To estimate the ECM time complexity we have to find out how many B – smooth numbers are in interval $(x - \sqrt{x} + 1, x + \sqrt{x} + 1)$ for the given smoothness bound B.

Let's denote by $\Psi(x,\ B)$ the number of B – smooth numbers in the interval $\langle 1, \ldots, x \rangle$. Then we know[3], that $\Psi\left(x, x^{\frac{1}{u}}\right) = xu^{-u+O(u)} \approx xu^{-u}$. For large numbers x we can simplify interval to $(x - \sqrt{x}, x + \sqrt{x})$, which gives the

[3] See [15], page 77.

same probability of finding B – smooth number:

$$\frac{\Psi\left(x+\sqrt{x},(x+\sqrt{x})^{\frac{1}{u}}\right)-\Psi\left(x-\sqrt{x},(x-\sqrt{x})^{\frac{1}{u}}\right)}{2\sqrt{x}}$$
$$=\frac{(x+\sqrt{x})\,u^{-u}-(x-\sqrt{x})\,u^{-u}}{2\sqrt{x}}$$
$$=u^{-u}>\frac{\Psi\left(x+\sqrt{x},x^{\frac{1}{u}}\right)-\Psi\left(x-\sqrt{x},x^{\frac{1}{u}}\right)}{2\sqrt{x}}.$$

Finally we would like to count the probability of getting in interval $(x-\sqrt{x},x+\sqrt{x})$ the B – smooth number:

$$\frac{\Psi\left(x+\sqrt{x},\ x^{\frac{1}{u}}\right)-\Psi\left(x-\sqrt{x},\ x^{\frac{1}{u}}\right)}{2\sqrt{x}}.$$

The solution of this problem is not obvious. There are some theorems that help to estimate the number of B – smooth numbers under some conditions: $\Psi\left(x+x^{\beta},x^{\frac{1}{u}}\right)-\Psi\left(x,x^{\frac{1}{u}}\right)\gg x^{\beta}u^{-u}$, where $\beta>\frac{1}{2},\ u>0$. Unfortunately, the case for $\beta=\frac{1}{2}$ has not been proved yet.
Moreover, even assuming the Riemann Hypothesis is true, we are not able to prove it.
In this case we assume that:

$$\Psi\left(x+\sqrt{x},x^{\frac{1}{u}}\right)-\Psi\left(x,x^{\frac{1}{u}}\right)\gg\sqrt{x}u^{-(u+o(u))}.$$

Then the probability of getting the B – smooth number in interval $(x-c\sqrt{x},x+c\sqrt{x})$ is $\gg u^{-(u+o(u))}$.
Under presented conditions we can expect, that the proper elliptic curve will be found after about $\frac{1}{0{,}608998}u^{u}=1.642042u^{u}$ trials.

3.2.2. Estimating of expected running time of algorithm

Every step requires $\ln M(B)$ additions of points on elliptic curve and every addition takes $O\left(\ln^2 n\right)$ of operations, where $M(B)=LCM(1,\ldots,B)$

Let's suppose that $B=p^{\frac{1}{u}}$. Now we are looking for such u, for which the expected running time is the smallest:

$$T(u,n,p)=O\left(\ln^2 n\right)O\left(\ln M(B)\right)O\left(u^{u}\right)$$
$$=A\cdot\ln^2 n\cdot\ln M(B)\cdot u^{u},$$

where A is a constant.

The second Tshebyshev's function helps us to find out the estimation of $\ln(LCM(1,\ldots,B))$[4]

$$\psi(B) = \ln(LCM(1,\ldots,B)) \approx B.$$

Then:

$$T(u,n,p) = A \cdot \ln^2 n \cdot B \cdot u^u = A \cdot \ln^2 n \cdot p^{\frac{1}{u}} \cdot u^u.$$

Let's denote by $S(u,n,p) = \ln(T(u,n,p))$. We have:

$$\begin{aligned} S(u,n,p) &= \ln\left(A \cdot \ln^2 n \cdot p^{\frac{1}{u}} \cdot u^u\right) \\ &= \ln A + \ln\ln^2 n + \frac{1}{u}\ln p + u\ln u. \end{aligned}$$

Then:

$$\frac{\partial S(u,n,p)}{\partial u} = -\frac{\ln p}{u^2} + \ln u + 1 = 0$$

and

$$\begin{aligned} u^2(1+\ln u) &= \ln p, \\ u^2(1+\ln u) &\approx u^2 \ln u \approx \ln p, \\ u^2 \ln u^2 &\approx 2\ln p. \end{aligned}$$

Finally we get $u = \frac{\sqrt{2\ln p}}{\sqrt{\ln u^2}}$, but $u^2 \ln u = \ln p$ and from this equation we get:

$$\ln u^2 + \ln\ln u = \ln\ln p.$$

Because $\ln u^2 \gg \ln\ln u$, we estimate $\ln u^2 = \ln\ln p$, what gives

$$u = \frac{\sqrt{2\ln p}}{\sqrt{\ln\ln p}}.$$

Then

$$p^{\frac{1}{u}} = e^{u\ln u} = e^{\frac{\ln p}{\sqrt{2\ln p}}\sqrt{\ln\ln p}} = e^{\frac{1}{\sqrt{2}}\sqrt{\ln p \cdot \ln\ln p}}$$

and

$$u^u = e^{u\ln u} = e^{\frac{\ln p}{u}} = e^{\frac{\ln p}{\sqrt{2\ln p}}\sqrt{\ln p \cdot \ln\ln p}} = e^{\frac{1}{\sqrt{2}}\sqrt{\ln p \cdot \ln\ln p}}.$$

Finally we have

$$T(u,n,p) = A \cdot \ln^2 n \cdot e^{\sqrt{2\ln p \cdot \ln\ln p}}.$$

[4] See [2].

4. Main results

We made the estimation of factorization running time of GNFS which is given by function

$$e^{\sqrt[3]{\frac{64}{9}}(\ln n\ )^{\frac{1}{3}}\cdot(\ln(\ln n)\ )^{\frac{2}{3}}-27.5622}. \quad (1)$$

We made load of computations of factoring by ECM numbers being products from 2 to 6 factors, each the same length, from 80 to 130 bits. We used the algorithm for point addition on elliptic curve with complexity $O(\ln n\ (\ln\ln n)^2\ \ (\ln\ln\ln n))$ that lead us to the formula of time complexity for ECM:

$$e^{\sqrt{2\ln p\cdot\ln\ln p}+\ln(\ln n\ \ (\ln\ln n)^2\ \ (\ln\ln\ln n))\ +a}. \quad (2)$$

From practical and theoretical obsrevations, which will be described below, we consider that constant a should be different for different numbers of factors. So we can describe the time complexity of factorisation by ECM by formula:

$$e^{\sqrt{2\ln p\cdot\ln\ln p}+\ln(\ln n\ \ (\ln\ln n)^2\ \ (\ln\ln\ln n))\ +a_i}, \quad (3)$$

where a_i is equal:

- $a_2 = -27.1957$ for 2 factors;
- $a_3 = -27.7505$ for 3 factors;
- $a_4 = -28.1881$ for 4 factors;
- $a_5 = -28.4320$ for 5 factors;
- $a_6 = -28.6344$ for 6 factors.

The differences between these values are the result of probability of finding any non–trivial divisor of modulus. The more factors the number has, the bigger is probability of finding out the proper elliptic curve with good parameters.
If the probability of finding a proper elliptic curve for given smallest factor is equal to P, then if there are k factors and all are the same length, the probability of finding the non–trivial factor is given by:

$$\begin{aligned} d(k) &= 1 - P^k - (1-P)^k \\ &= kP - \binom{k}{2}P^2 + \binom{k}{3}P^3 - \ldots + \left((-1)^{n+1-1}\right)P^n. \end{aligned}$$

Because in our computations n is small (not bigger than 6) and P is very small, we can consider that:

$$\begin{aligned} d(k) &= 1 - P^k - (1-P)^k \\ &= kP - \binom{k}{2}P^2 + \binom{k}{3}P^3 - \ldots + \left((-1)^{n+1-1}\right)P^n \approx kP. \end{aligned}$$

Then it is easy to see, that $\frac{d(k+1)}{d(k)} \approx \frac{k+1}{k}$. The probability has a direct impact for the expected running time of the alhorithm. The bigger probability, the smaller is constant a_i.
So we may expect that $a_i - a_{i-1} \approx \ln\left(\frac{i-1}{i}\right)$, what gives:

$$\begin{aligned} a_3 - a_2 &= -0.4055, \quad a_4 - a_3 = -0.2877, \\ a_5 - a_4 &= -0.2231, \quad a_6 - a_5 = -0.1823. \end{aligned}$$

Constants we got in our computations gave results:

$$\begin{aligned} a_3 - a_2 &= -0.5548, \quad a_4 - a_3 = -0.4376, \\ a_5 - a_4 &= -0.2439, \quad a_6 - a_5 = -0.2024. \end{aligned}$$

Basing on our estimated formulas (1) and (3) we were able to compare time required to factorization each number by GNFS and ECM algorithms. We consider that secure MultiPrime RSA modulus should not be factorized faster by ECM than GNFS algorithm. The Table 2 presents the smallest length of secure modules which may consist from given number of cofactors of similar length.

Table 2: Length of number for which GNFS and ECM have the same expected running time

Number of factors	The length of number in bits
2	8
3	724
4	4004
5	11155
6	23909

In 2000 year the COMPAQ made a comparison (Compaq 2000) between speed of factorization large numbers being the product of several primes by GNFS and ECM algorithms.

The calculations were done for expected running times given by following formulas:

- $e^{1.923\cdot\sqrt[3]{\ln n\cdot\ln^2(\ln n)}}$ for GNFS algorithm.
- $2(\log_{10}n)^2 e^{\sqrt{2\ln n\cdot\ln(\ln n)}}$ for ECM algorithm.

These formulas cannot result in precisious comparision of ECM and GNFS algorithms.

We have estimated the number of prime factors for numbers of distinct length, for which the GNFS is the fastest known factorization algorithm. Similar analyzis, using the same functions and techniques as COMPAQ, was presented by Martin Hinek in his article from 2006 ([10]). We present the comparision of our results with results computed by M. Hinek in Table 3.

Table 3: Maximal number of distinct prime factors for given length of factorized number.

Length of number in bits	1024	2048	4096	8192
Maximal number of prime factors (our approach)	3	3	4	4
Maximal number of prime factors (M. Hinek)	3	3	4	5

5. Conclusion

Our computations may help to choose the better factorization algorithm in particular situation. We have shown that the comparison between GNFS and ECM cannot be done properly if expected running times of these algorithms are not described by precise formulas. Our description of expected running time of ECM and some observations about the probability in situation when the large number is being the product of many small factors resulted in formulas that can be used by everyone who want to choose the fastest algorithm of factorization in particular situation.

References

[1] I. F. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography*, Cambridge University Press, (1999).

[2] J. Cilleruelo, J. Rue, P. Sarka, and A. Zumalacarregui, The least common multiple of sets of positive integers, *ArXiv e-prints* arXiv:1112.3013v1 [math.NT], (2011).

[3] Compaq. Cryptography Using Compaq MultiPrime Technology in a Parallel Processing Environment, Compaq, (2000).

[4] http://www.crypto-world.com/FactorRecords.html.

[5] J. H. Ellis, The story of non-secret encryption, Available from http://cryptome.org/jya/ellisdoc.htm, (1997).

[6] http://www.emc.com/emc-plus/rsa-labs/historical/the-rsa-factoring--challenge.htm.

[7] P. Gaudry, A. Kruppa, F. Morain, L. Muller, E. Thom and P. Zimmermann, cado-nfs, An Implementation of the Number Field Sieve Algorithm, Release 1.1, available from http://cado-nfs.gforge.inria.fr/.

[8] W. Geiselmann and R. Steinwandt, A Dedicated Sieving Hardware, In *Public Key Cryptography, 6th International Workshop on Practice and Theoryin Public Key Cryptography, PKC 2003 Proceedings*, LNCS 2567, pp. 254–266, (2002).

[9] A. Granville Smooth numbers: computational number theory and beyond, In *Algorithmic Number Theory MSRI Publications*, no. 44: pp. 267–323 (2008).

[10] M. Jason Hinek, On the security of Multi – prime RSA. In *J. Mathematical Cryptology*, no. 2(2), pp 117–147 (2008).

[11] A. K. Lenstra, Unbelievable Security. Matching AES Security Using Public Key Systems, In *Advances in Cryptology - ASIACRYPT 2001*, pp 67–86 (2001).

[12] A. K. Lenstra, H. W. Lenstra, M. S. Manasse, and J. M. Pollard, The number field sieve, In *Proc 22nd Annual ACM Symposium on the Theory of Computing*, pp. 564–572 (1990).

[13] A. K. Lenstra, A. Shamir, J. Tomlinson, and E. Tromer, Analysis of Bernstein's Factorization Circuit. In *Advances in Cryptology – ASIACRYPT 2002*, pp. 1–26, (2002).

[14] H. W. Lenstra, Factoring Integers with Elliptic Curves, In *The Annals of Mathematics 126* : pp. 649–673 (1987).

[15] C. Pomerance, Smooth numbers and the quadratic sieve, In *Algorithmic Number Theory MSRI Publication*s, no. 44 : pp. 69–81 (2008).

[16] S. Singh, Code book. Delacorte Press, pp. 21–220 (2002).

[17] I. Tolkov, Counting points on elliptic curves: Hasse's theorem and recent developments, http://igor.tolkov.com/essays/336paper.pdf, (2009).

PORÓWNANIE ALGORYTMÓW FAKTORYZACJI DUŻYCH LICZB POSIADAJĄCYCH KILKA RÓŻNYCH CZYNNIKÓW PIERWSZYCH

Streszczenie. W artykule przedstawiamy analizę bezpieczeństwa powszechnie znanego algorytmu klucza publicznego RSA oraz jego następcy MultiPrime RSA. Skupiliśmy się na dokładniejszym wyznaczeniu oczekiwanego czasu faktoryzacji dużych liczb za pomocą dwóch algorytmów: Metody Krzywych Eliptycznych (ECM) i Ogólnego Sita Ciała Liczbowego (GNFS). Dodatkowo dla algorytmu MultiPrime RSA została obliczona maksymalna liczba czynników pierwszych dla danej długości modułu, która nie powoduje zmniejszenia bezpieczeństwa.

Słowa kluczowe: faktoryzacja, MultiPrimeRSA, Metoda Krzywych Eliptycznych, Ogólne Sito Ciała Liczbowego, B-gładkość