Algorithms Seminar 2000–2001,
F. Chyzak (ed.), INRIA, (2002), pp. 77–80.

Available online at the URL
http://algo.inria.fr/seminars/.

Thirty Years of Integer Factorization

François Morain

LIX, École polytechnique (France)

February 5, 2001

Summary by Marianne Durand

Abstract

Factoring integers is quite an old challenge. Thirty years ago, two researchers factored the mythic number $F_7 = 2^{2^7} + 1$. A few years later public-key cryptography was born, and with it the famous RSA algorithm. Even if the security of RSA is not equivalent to integer factorization, factoring the RSA key is the simplest way to decode everything, so a lot of people tried to factor. In 1990, $F_9 = 2^{2^9} + 1$, the ninth Fermat number was factored, with the help of hundreds of computers. In august 1999, it was the turn of the first ordinary 512-bit integer. What follows is a survey of thirty years of factorization, describing the different methods used and the technical problems met.

1. Introduction

Factoring is of great interest since it allows to use the properties of prime number in arithmetic. It is the keystone of the RSA algorithm, the mostly used encryption algorithm. RSA is an asymmetric public key algorithm that is based on the fact that the product of two very large prime numbers can not be easily factored, whereas to check if a number is prime can be done quickly. The complexity class of testing the primality of an integer is $NP \cap$ co-NP. Factoring a number is in NP, but can be done in polynomial time on a quantum computer!

Method	Complexity
sieve	p
ρ	$\sqrt{p}$
elliptic curve method	$L_p[1, 1/2]$
quadratic sieve (QS)	$L_N[1/2, c]$
number field sieve (NFS)	$L_N[1/3, c]$

TABLE 1. Complexity of factorization methods (N is the integer to be factored, p its smallest factor)

A lot of different methods exist to factor a number, starting from the linear sieve up to the algebraic sieve, including methods based on elliptic curves. Their complexity can be expressed in terms of the function

$$L_x[\alpha, c] = e^{c \log^\alpha x (\log\log x)^{1-\alpha}}.$$

Some complexities are given in Table 1. The smallest factor p of N is usually of order $\sqrt{N}$. The letter c stands for a constant and is not specified as it depends on the algorithm and its implementation. These methods are detailed in the next section.

2. Combination of Congruences

The method of combination of congruences is an extension of Kraitchik's method. The latter aims at finding an integer x such that $x^2 \equiv 1 \bmod N$ and $x \neq \pm 1 \bmod N$, then at testing if $\text{pgcd}(x-1, N)$ is non-trivial. If so, it is a factor of N. The quadratic congruence approach refines the way the square root of 1 is found. The first step consists in finding pairs of integers $(u_i, v_i)_{i \in I}$ such that $u_i^2 \equiv v_i \bmod N$ and $u_i^2 \neq \pm v_i$. The second step is to find a subset $J \subset I$ such that $\prod_{j \in J} v_j$ is a square, noted V_J^2. This step is detailed later. If we note $\prod_{j \in J} u_j = U_J$ then step 2 implies $U_J^2 \equiv V_J^2 \bmod N$. As we also assume that V_J and N are together prime (otherwise we have a factor of N) then $x = U_J/V_J \bmod N$ is well defined and is a square root of 1. There is a probability greater than $1/2$ that it gives a non trivial factorization of N. This extension is interesting because in order to find the pairs (u_i, v_i), we can use an algorithm that eventually rejects or ignore some valid pairs, to go faster. One solution for this is Dixon's method. The idea is to restrict the search to integers v_i that can be factored on a small set of given small prime integers $P_k = (p_1, \ldots, p_k)$. To find pairs (u_i, v_i) according to Dixon's method, we choose an integer u_i, and try to factor u_i^2 on the set P_k. If we succeed, then we keep the pair (u_i, u_i^2). The integer u_i has to be greater than $\sqrt{N}$, so as to give a non-trivial pair.

Once the pairs (u_i, v_i) are found, the second step is to find a subspace J such that $\prod_{j \in J} v_j$ is a square. As the factorization of each v_i is already known, this can be seen as a linear algebra problem. Assume that there are $k+1$ valid pairs available. Consider the matrix M of size $(k, k+1)$ with coefficients 0 and 1 viewed in the field $\mathbb{Z}/2\mathbb{Z}$ and such that $M[i,j]$ is equal to the exponent of p_i in the factorization of v_j. This matrix has a rank smaller than k, so there exists a linear combination of the colums equals to 0. The subset J corresponds to the non-zero coefficients in the linear combination, and we can check that $\prod_{j \in J} v_j$ is a square, because all its factors are of even degree. To exhibit a concrete linear combination equal to zero is made easier by the sparsity of the matrix M. As a matter of fact, the techniques of Wiedemann or of Lanczos have complexity $O(k^{2+\epsilon})$ on sparse matrices, whereas the Gauss pivot has complexity $O(k^3)$. Then we have the expression of V_J easily, and a square root of 1 that may give a factorization of N. This algorithm has a complexity $L_N[1/2, c]$, where c is a constant that depends on the algorithm.

3. Sieves

A sieve algorithm searches a lot of candidates satisfying a certain property. Then it makes some tests systematically on all candidates, and at the end keeps the ones that have passed all the tests successfully. One of the first sieves concerning primality and factorization is the Erastothene sieve. The sieve technique is useful in factorization for the search of the set of pairs (u, v) such that $u^2 \equiv v \bmod N$.

The basic quadratic sieve, found by Pomerance in 1981 is an extension of the combination of congruence, with a specific choice algorithm for the pairs (u_i, v_i). The idea is to choose $u_i = i + \lfloor \sqrt{N} \rfloor$, which implies

$$v_i = \left(i + \lfloor \sqrt{N} \rfloor\right)^2 - N. \tag{1}$$

The advantage is that v_i is close to $2i\sqrt{N}$, and thus $v_i \ll N$, this increases the probability that the prime factors of v_i are small. To check that these factors are in the prime number basis P_k we use a sieve algorithm. This sieve algorithm can be described as follows. First fill an array S such that $S[i] = v_i$ for i from 1 to a bound L, then for every p in the prime number basis P_k, for the two roots of the equation $\left(i+\lfloor\sqrt{N}\rfloor\right)^2 \equiv N \bmod p$ noted $i_\pm(p)$, do $i \leftarrow i_\pm(p)$, and while $i < L$ do $S[i] \leftarrow S[i]/p$ and $i \leftarrow i + p$. This algorithm is justified by the equivalence $p|v_i \iff \left(i + \lfloor\sqrt{N}\rfloor\right)^2 \equiv N \bmod p$. Then at the end of the loops, for every i such that $S[i] = 1$, v_i is factored on P_k. The complexity of this algorithm is $L_N[1/2, 3/\sqrt{8}]$, and the cost in memory space is $L_N[1/2, 1/\sqrt{8}]$. The algorithm can be optimized in many ways, for example the large prime or double large prime variation that we are going to detail in the next paragraph.

The large prime variation owes its name to the use of large primes, not in the prime factor basis, and smaller than the square of the largest prime in the basis P_k. The sieving stage of the algorithm can easily be modified to find new relations $v_i = q\prod p^{\alpha_p}$, where q is a large prime. Now we can combine two relations using the same large prime q, namely $v_1 = q\prod p^{\alpha_p}$ and $v_2 = q\prod p^{\beta_p}$, and see that v_1v_2/q^2 is factored on P_k. This large prime technique allows us to search for more "good" pairs (u_i, v_i) and so to get more candidates to factor N. In practice it means a speed-up by a factor of approximatly 2.5 [5]. The double large prime variation is quite similar, the difference is that two large primes are allowed in the factorization of the integers v_i. For example if $v_1 = q_1q_2\prod p_j^*$, $v_2 = q_2q_3\prod p_j^*$, and $v_3 = q_1q_3\prod p_j^*$ (p^* stands for any power of p), then $v_1v_2v_3/(q_1q_2q_3)^2$ is factored on the prime basis. The choice of v_i, v_j and v_k such that their product can be factored upon the prime basis P_k modulo squares of large primes can be modelled by a graph problem. Let G be the graph with vertex q_i and multiple edges q_i, q_j labelled by the multiples v_k of q_iq_j. A useful relation corresponds to a cycle in the graph G. This technique was used for the sieving step of a 138-digit number in 1990, as the non-optimized sieve was too big to be handled [5] (see also [4]).

The algebraic sieve [2] or number field sieve (NFS) algorithm is based on the factorization in a number field. Given a polynomial $P \in \mathbb{Z}[X]$ irreducible over $\mathbb{Q}$, we will work in the number field $\mathbb{Q}[X]/\big(P(X)\big) = \mathbb{Q}(\theta)$ where θ is a root of P. In the ring $\mathbb{Z}[\theta]$ we can talk about the primality or the prime decomposition of an element, and the norm of the number $a - b\theta$ is $\prod(a - b\theta_i)$ where θ_i are all the roots of the polynomial P. In particular the norm does not depend on the particular choice of θ. The description of the algorithm requires the following notation. First let m be an integer such that $P(m) \equiv 0 \bmod N$, then consider the ring homomorphism ϕ that maps $\mathbb{Z}[\theta]$ onto $\mathbb{Z}/N\mathbb{Z}$ and that satisfies $\phi(\theta) = m$. We are now looking for a set $\mathcal{A}$ of pairs (a, b) such that $\prod_{\mathcal{A}}(a - b\theta) = (A - B\theta)^2$ and $\prod_{\mathcal{A}}(a - bm) = Z^2$. These properties give $\phi\left((A - B\theta)^2\right) \equiv (A - Bm)^2 \equiv Z^2 \bmod N$. Then $(A - Bm)/Z$ is a square root of 1, that provides a candidate to factor N. The choice of the polynomial P plays a large part in the efficiency of the algorithm [6]. If the degree of P is $O\left((\log N)^{1/3}(\log\log N)^{2/3}\right)$ then the complexity is $L_N[1/3, c]$, where c is a constant.

The way the factorization is done in $\mathbb{Z}[\theta]$ needs to be explained as it is a non trivial part of the algorithm. The idea is to factor first the norm of $a - b\theta$, $\mathrm{Norm}(a - b\theta) = \pm\prod p^{\alpha_p(a,b)}$. This helps because the factorization of $a - b\theta$ follows the factorization of its norm. If p is a factor of $N(a - b\theta)$, and p does not divide b (this being a pathological case), then there exists an integer r such that $a - br \equiv 0 \bmod p$ and $P(r) \equiv 0 \bmod p$. We denote by $[p, r]$ the ideal of $\mathbb{Z}[\theta]$ such that any element $x - y\theta$ of $[p, r]$ satisfies $\mathrm{Norm}(x - \theta y) \equiv 0 \bmod p$ and $x - yr \equiv 0 \bmod p$. This family of ideals is very interesting because $(a - b\theta) = \prod[p, r]^{\alpha_p(a,b)}$, where $(a - b\theta)$ is the ideal generated by $a - b\theta$.

Now that we know how to factor a number in $\mathbb{Z}[\theta]$, we apply the sieve algorithm over the pairs (a, b). The factorization algorithm can be optimized by a good choice of the polynomial P [1]. The variant SNFS, Special Number Field Sieve, targets the numbers $b^n \pm 1$ by the choice of P. The

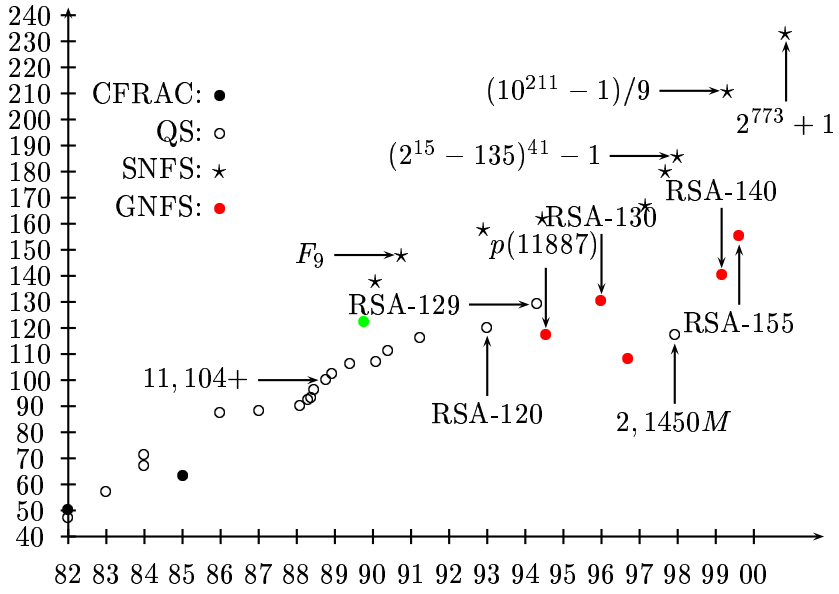

FIGURE 1. Size in bits of the factored numbers depending on the year.

general NFS algorithm becomes better than the quadratic sieve with large primes optimizations for numbers of size around 130 digits.

4. Records and Conclusion

Figure 1 shows the evolution of the factorization records. For each specific algorithm, the progress follows Moore's law that states that the speed of computers double every 18 months. Then for each change of algorithm, there is a jump. Remark that the SNFS algorithm factors specific numbers, that are thus larger than for GNFS that factors general numbers [3]. The linear algebra is often the limiting factor, and unless there is a new idea on the subject, RSA can still be used for some times if used with a key big enough.

Bibliography

[1] Bernstein (Daniel J.) and Lenstra (A. K.). – A general number field sieve implementation. In Lenstra (A.) and Lenstra (H.) (editors), *The development of the number field sieve*, pp. 103–126. – Springer, Berlin, 1993.

[2] Buhler (J. P.), Lenstra, Jr. (H. W.), and Pomerance (Carl). – Factoring integers with the number field sieve. In Lenstra (A.) and Lenstra (H.) (editors), *The development of the number field sieve*, pp. 50–94. – Springer, Berlin, 1993.

[3] Cavallar (Stefania), Dodson (Bruce), Lenstra (Arjen K.), Lioen (Walter M.), Montgomery (Peter L.), Murphy (Brian), te Riele (Herman), Aardal (Karen), Gilchrist (Jeff), Guillerm (Gerard), Leyland (Paul C.), Marchand (Joël), Morain (François), Muffett (Alec), Putnam (Chris), Putnam (Craig), and Zimmermann (Paul). – Factorization of a 512-bit RSA modulus. In Preneel (B.) (editor), *Advances in cryptology—EUROCRYPT'00 (Bruges, 2000)*, pp. 1–18. – Springer, Berlin, 2000.

[4] Lenstra (A. K.) and Manasse (M. S.). – Factoring with two large primes. *Mathematics of Computation*, vol. 63, n° 208, 1994, pp. 785–798.

[5] Lenstra (Arjen K.) and Manasse (Mark S.). – Factoring with two large primes (extended abstract). In Damgå rd (I. B.) (editor), *Advances in cryptology—EUROCRYPT '90 (Aarhus, 1990)*, pp. 72–82. – Springer, Berlin, 1991.

[6] Murphy (Brian). – Modelling the yield of number field sieve polynomials. In Buhler (J. P.) (editor), *Algorithmic number theory (Portland, OR, 1998)*, pp. 137–150. – Springer, Berlin, 1998. Proceedings of the Third International Symposium ANTS-III.