DISTRIBUTED PRIMALITY PROVING AND THE PRIMALITY OF $(2^{3539} + 1)/3$

François Morain *†
morain@inria.inria.fr

Abstract

We explain how the Elliptic Curve Primality Proving algorithm can be implemented in a distributed way. Applications are given to the certification of large primes (more than 500 digits). As a result, we describe the successful attempt at proving the primality of the 1065-digit $(2^{3539}+1)/3$, the first *ordinary* Titanic prime.

1 Introduction

For cryptographical purposes [7], it is desirable to generate large primes as fast as possible. This can be done via *ad hoc* techniques [30, 12, 14, 4] or by means of a general purpose primality testing algorithm such as that described in [1, 11, 10, 6] or the Elliptic Curve Primality Proving (ECPP) algorithm due to Atkin [2, 26, 24] (For a survey of primality testing, see [18]).

Another point is to certify large primes, such as the Cunningham numbers [8], which sometimes have more than 400 digits. The purpose of this paper is to explain how the ECPP algorithm has been implemented on a network of workstations and used to test some numbers with more than 500 digits for primality. In particular, it is now routine to test 800-digit numbers and it is not too hard to test 1000-digit numbers.

We first begin by a short introduction to ECPP and then, we explain the distributed process *à la* Lenstra-Manasse [19]. These ideas are exemplified by the certification of large primes and we also give the history of the primality of the record breaking $\mathcal{N}_{3539} = (2^{3539} + 1)/3$, which has 1065 digits.

*Institut National de Recherche en Informatique et en Automatique (INRIA), Domaine de Voluceau, B. P. 105 78153 LE CHESNAY CEDEX (France) / Département de Mathématiques, Université Claude Bernard, 69622 Villeurbanne CEDEX (France).

†On leave from the French Department of Defense, Délégation Générale pour l'Armement.

I.B. Damgard (Ed.): Advances in Cryptology - EUROCRYPT '90, LNCS 473, pp. 110-123, 1991.

2 A brief description of ECPP

2.1 Elliptic curves

Let $\mathbf{K}$ be a field of characteristic prime to 6. An elliptic curve E over $\mathbf{K}$ is a non singular algebraic projective curve of genus 1. It can be shown [9, 34] that E is isomorphic to a curve with equation:

$$y^2z = x^3 + axz^2 + bz^3, \tag{1}$$

with a and b in $\mathbf{K}$. The *discriminant* of E is $\Delta = -16(4a^3 + 27b^2)$ and the *invariant* is

$$j = 2^8 3^3 \frac{a^3}{4a^3 + 27b^2}.$$

We write $E(\mathbf{K})$ for the set of points with coordinates $(x : y : z)$ which satisfy (1) with $z = 1$, together with the point at infinity: $O_E = (0 : 1 : 0)$. We will use the well-known *tangent-and-chord* addition law on a cubic [16] over a finite field $\mathbf{Z}/p\mathbf{Z}$ as well as over a ring $\mathbf{Z}/N\mathbf{Z}$ with N composite (see [21] for a justification).

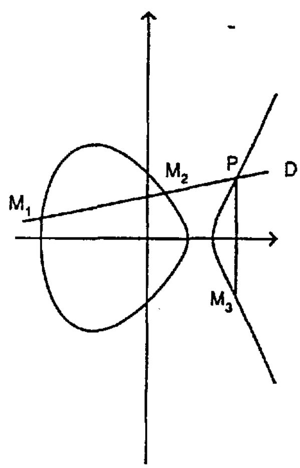

Figure 1: An elliptic curve over $\mathbf{R}$

In order to add two points $M_1 = (x_1, y_1)$ and $M_2 = (x_2, y_2)$ on E resulting in $M_3 = (x_3, y_3)$, the equations are

$$\begin{cases} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1 \end{cases}$$

where

$$\lambda = \begin{cases} (y_2 - y_1)(x_2 - x_1)^{-1} & \text{if } x_2 \neq x_1 \\ \\ (3x_1^2 + a)(2y_1)^{-1} & \text{otherwise.} \end{cases}$$

We can compute kP using the binary method [17] (see also [10]) or addition-subtraction chains [29].

2.2 Primality proving

Let us recall one of the converses of Fermat's theorem.

Theorem 1 ([31]) *Let s be a divisor of $N-1$. Let a be an integer prime to N such that*

$$a^{N-1} \equiv 1 \bmod N \text{ and } \gcd(a^{(N-1)/q} - 1, N) = 1,$$

for each prime divisor q of s. Then each prime divisor p of N satisfies $p \equiv 1 \bmod s$.

Corollary 1 *If $s > \sqrt{N} - 1$ then N is prime.*

A similar theorem can be stated for elliptic curves.

Theorem 2 ([13, 20]) *Let N be an integer greater than 1 and prime to 6. Let E be an elliptic curve over $\mathbf{Z}/N\mathbf{Z}$, m and s two integers such that $s \mid m$. Suppose we have found a point P on E that satisfies $m\,P = O_E$, and that for each prime factor q of s, we have verified that $\frac{m}{q}P \neq O_E$. Then if p is a prime divisor of N, $\#E(\mathbf{Z}/p\mathbf{Z}) \equiv 0 \bmod s$.*

Corollary 2 *If $s > (\sqrt[4]{N} + 1)^2$, then N is prime.*

In order to use the preceding theorem, we need to compute the number of points m. This process is far from trivial in general (see [32]). From a practical point of view, it is desirable to use deep properties of elliptic curves over finite fields. This involves the theory of complex multiplication and class fields and requires a lot of theory [26]. We can summarize the principal properties.

Theorem 3 *Let p be an odd prime. Every elliptic curve $E \bmod p$ has complex multiplication by an order of an imaginary quadratic field $K = \mathbf{Q}(\sqrt{-D})$.*

From a very down-to-earth point of view, this comes down to saying

- p splits completely in K as $(p) = (\pi)\,(\pi')$ in K;
- $H_D(j(E)) \equiv 0 \bmod p$ for a fixed polynomial $H_D(X)$ in $\mathbf{Z}[X]$;
- $m = \#E(\mathbf{Z}/p\mathbf{Z}) = (\pi - 1)(\pi' - 1) = p + 1 - t$, where $|t| \leq 2\sqrt{p}$ (Hasse's theorem).

The computation of the polynomials H_D is dealt with in [26] and [27].

2.3 Outline of ECPP

We now explain how the preceding theorems are used in a *factor and conquer* algorithm similar to the DOWNRUN process of [37]. The first phase of the algorithm consists in finding a sequence $N_0 = N > N_1 > \cdots > N_k$ of probable primes such that N_{i+1} prime $\Longrightarrow N_i$ prime. The second then proves that each number is prime, starting from N_k.

Procedure SearchN

1. $i := 0$; $N_0 := N$;
2. find a fundamental discriminant $-D$ such that (N_i) splits as the product of two principal ideals in $\mathbb{Q}(\sqrt{-D})$;
3. for each solution of $(N_i) = (\pi)(\pi')$, find all factors of $m_\pi = (\pi - 1)(\pi' - 1)$ less than a given bound B and let N_π be the corresponding cofactor;
4. if one of the N_π is a probable prime then set $N_{i+1} := N_\pi$, store $\{N_i, D, \pi, m_\pi\}$ set $i := i + 1$, and go to step 2 else go to step 3.
5. end.

The second phase consists in proving that the numbers N_i are indeed primes. This is done as follows.

Procedure Proof

for $i = k..0$

1. compute a root j of $H_{D_i}(X)$ mod N_i as described in [27, 28];
2. find an equation of the curve E_i whose invariant is j and cardinality m_i;
3. verify the condition of theorem (2).

end.

For more details, the reader is referred to [2].

3 Large primes

The author used ECPP to test about fifty numbers from the Cunningham tables [8] and some others, namely $S_p = ((1 + \sqrt{2})^p + (1 - \sqrt{2})^p)/2$ for $p \in \{1493, 1901\}$ with respectively 572 and 728 digits, in 30 and 40 days on a single SUN 3/60. Indeed, a simple extrapolation shows that testing a 1000-digit number would require about 6 months (at least). We must do something else to increase the bound on the largest number ECPP can test.

4 Distributed computations

From the preceding description, it is easy to see that this algorithm is very well suited for distributed computations. We can do the first phase in parallel and then the second one too. Let us see how I did this.

First of all, I implemented ECPP using the `Le_Lisp` language and the multiprecision described in [15]. Then the computations were done using a star network *à la* Caron-Silverman [33]. There are a *master* ($\mathfrak{M}$) and an indefinite number of workstations, called *slaves* ($\mathfrak{S}$).

The idea is that when dealing with very large numbers, the crucial part of ECPP is the first one, because it requires the factorization of very large numbers. There are basically two ways of doing that. The first one is to try to factor a single number using all the stations. The second is to let each station work on a different number. Actually, I use the latter scheme, because the first one would require more communications and also because it is not the *right* philosophy of the test: The less factoring power we use, the better.

We now describe the conditions required to do an optimal job.

4.1 Constraints

We want to use the idle time of a network of workstations. We do this in a way similar to that of [19]. We start a process on a machine in such a way that a legitimate user is not (too much) disturbed: If a user types on a console (in UNIX words, he changes the date of one of the `tty`'s), then the program is stopped (by means of a `kill -STOP`) and restarted 10 minutes after the last action of the user (with a `kill -CONT`). The process is also stopped whenever the load climbs up some prescribed value (typically 1.5) and is subjected to the same restart conditions. All this is done with the shell scripts distributed by Mark Manasse for integer factorization. Another important feature of these programs is the ability to restart themselves after a small crash such as a `Connection timed out` from a server. Also, they do not depend on a particular machine (at least running UNIX or ULTRIX) or a particular language. It is possible to use a `C` program on a DEC station and a `Le_Lisp` program on a SUN.

4.2 The first phase

4.2.1 Role of the master

On $\mathfrak{M}$ (typically the author's own workstation), the program used does the following things for each N_i of the first phase

1. put in the file WHICHN the number to be tested;

2. find all fundamental discriminants D (from a finite subset $\mathcal{D}$) for which N_i is represented by a form of G_0 and put them in the file DSET;

3. initialize the rank of the next D to be examined to 1 in the file DRANK;

4. start finding a suitable D.

4.2.2 Role of the slaves

On $\mathfrak{S}$, the program looks like

1. read the number to be tested from WHICHN and call it N;
2. while N is equal to the content of WHICHN, select a new D in DSET, update DRANK and try to factor any of the m_π.

4.2.3 Tasks performed by every machine

Each machine does the following

1. find a D such that (N) splits completely in $\mathbb{Q}(\sqrt{-D})$;
2. try to factor each m_π using first trial division, then Pollard's ρ method, and finally the $p-1$ method.

Inside each factoring algorithm, the program periodically tests whether something has happened. When this is so, it gives up on N_i and begins a new work on N_{i+1}. When using the ρ method [23], the test is done at each gcd (for our purposes, there are 10^4 iterations and a gcd each 1000 iterations). During $p-1$, only once.

4.2.4 Communications between $\mathfrak{M}$ and $\mathfrak{S}$

The files DSET, DRANK and WHICHN have just been described. All this supposes the use of a distributed file system: Here it is NFS that does all the job. Special code has been written to handle the problems arising when one machine wants to read a file while another tries to write in it or to test whether the file can be accessed through NFS.

4.3 The second phase

For each N_i, it remains to check the primality conditions. Using a file containing the next number to be certified, each station takes the useful data and does its job. It should be noted that this phase can be started even if the first one is not complete.

4.4 Problems encountered

One of the major problem is the reliability of the NFS protocols, especially when using machines not depending from the same file server. The program is very well suited for testing the reliability of the network. Each time there is a connection problem, the process simply crashes.

Also, using a `Le_Lisp` executable requires a lot of memory and, sometimes, this resulted in a swap problem and also a crash.

5 Establishing a new frontier: the history of $\mathcal{N}_{3539}$

Last year, the 100-digit line was crossed for the first time for integer factorization [19]. In 1983, Yates [38] introduced the concept of *Titanic* primes, that is primes with at least 1000 decimal digits. This seemed to make a distinction between the real world of small primes and that of large primes. The frontier for primality testing was thus 1000 digits. The aim of this section is to describe how we went far beyond the line, thus making the testing of 1000-digit numbers a routine.

5.1 Entomology of a Record

The first thing to do was finding a good candidate. It had to be greater than the repunit R_{1031}, whose primality was proven by Williams and Dubner [36]. During their setting of the new Mersenne's conjecture [3], Bateman, Selfridge and Wagstaff tested some numbers of the form $\mathcal{N}_p = (2^p + 1)/3$ for primality. They found that $\mathcal{N}_p$ was a probable prime for $p \in \{1709, 2617, 3539\}$.

During EUROCRYPT '89 (April 10-13, 1989), it appeared that both ECPP and the Jacobi Sums test [11, 10, 6] were able to attack numbers as large as 1000 digits. This was the very start of a stimulating competition with W. Bosma and M.-P. van der Hulst.

Indeed, the first of these numbers ($p = 1709$, $\mathcal{N}_p$ with 514 digits) was the first number proven prime using ECPP in its distributed version. This was done on April 19, 1989 with three SUN's and four days of CPU.

Then, I decided to skip $p = 2617$ and try $\mathcal{N}_{3539}$. As shown by in the following figure, the factorization of $\mathcal{N}_{3539} - 1$ is not complete (up to now).

$$2^{3538} - 1$$

$$2^{1769} - 1 \qquad\qquad\qquad\qquad 2^{1769} + 1$$

$$2^{29} - 1 \quad \Phi_{1680}(2) \quad 2^{61} - 1 \qquad\qquad 3 \times 59 \times 3539 \times 39232883$$

$$233 \times 1103 \times 2089 \qquad p_{19} \qquad\qquad \times 768614336404564651 \times C_{502}$$

$$10199969089 \times C_{496}$$

Using ECPP, I first launch the process on April 20, 1989. The set $\mathcal{D}$ mentioned above consisted of all D's with $h(-D) \leq 20$, sorted according to $(h/g, h, D)$. Following [2], the *difficulty* of testing N may be defined as

$$\Phi(N) = e^{-\gamma} \frac{\log N}{M(N)}$$

where γ is Euler's constant and $M(N)$ is defined as follows. Put

$$M(N) = \sum_{\substack{D \in \mathcal{D} \\ N \in G_0(-D)}} w(-D) \frac{g(-D)}{h(-D)},$$

where the summation is on all D for which N can be represented by a form of the principal genus of quadratic forms of discriminant $-D$, $g(-D) = 2^{t-1}$ with t the number of prime factors of D and $h(-D)$ the class number. As a matter of fact, $\Phi(N)$ yields the value ($\log B$) of the upper bound on the largest factor of a number of points m we must factor in order to find a good candidate.

Coming back to $\mathcal{N}_{3539}$, I found that $M(\mathcal{N}_{3539}) = 55$ yielding $\log_{10} B = 11$. This implied in turn that the only way to achieve this was using ECM. At that time, I hadn't implemented this and so the program started using only Pollard ρ and the $p-1$ method *. This first attempt lasted till May 13, without any result: I couldn't even find a good N_1. There was something to be done. Moreover, some problems seemed to arise in the $p-1$ method, where the routine seemed to loop forever in some cases.

When looking at

$$\log B = \Phi(N), \tag{2}$$

there are two distinct ways of solving the problem. The first one is to use sophisticated factoring routines, the other one is to increase the value of $M(N)$. I used the second and decided to enlarge $\mathcal{D}$ with all D less than 2^{15} †. This increased $M(N)$ to the value of 174, yielding $\log_{10} B = 3.44$. This clearly said that ECM was no more necessary and that ρ was enough. After fixing some stupid bug in my ρ routine, I re-started the program on June 5 and it lasted till July 10, yet without any result.

Clearly, there was a problem. Using induction, it seemed clear that there was, somewhere, a deep bug that only appeared when dealing with large numbers, but not with small ones. So I decided to stop working on $\mathcal{N}_{3539}$, and began to reassure myself with a smaller one, namely $\mathcal{N}_{2617}$ (788 digits). Although this number could have been done by simply factoring $\mathcal{N}_{2617} - 1$ (as remarked by Atkin), this attempt was designed to find this bug. So, the process started on July 21 and ended on August 19, proving the number to be prime, but without revealing any bug.

At this point, I decided to implement ECM, just to see if something would happen to change. I had problems with this, since it was only possible to use the first phase of the algorithm, all the second phases requiring too much memory (they all need about $(\log N)^2$ storage, making it infeasible for 1000-digit numbers). Moreover, I could only use 20 curves or so, again because the storage was making it prohibitive to use on workstations with not too much memory, such as a standard SUN 3/50 (4 Mo). This was quite a disappointment. The third attempt on $\mathcal{N}_{3539}$ was then started on September 12 and took two weeks. Nothing observable happened.

I decided then to replace all these D's less than 2^{15} with all D's with h/g small, irrespective of the size of D as soon as D fitted in a 32-bit word. More precisely, I computed and stored all D with $h(-D) \leq 50$ (plus some with $h = 64$) and ordered them according to $(h/g, h, D)$. This yielded $M(\mathcal{N}_{3539}) = 291$ and $\log_{10} B = 2.05$. What would appear as the last attempt then began on September 29.

A feeling of deep personal gratification came over me when, on October 5, 1989, I finally confirmed my initial impression that a part of the program was irretrievably

---

*As suggested by Atkin, $p-1$ is worth using when dealing with Cunningham numbers, because they have non-trivial arithmetic properties.

†This limitation comes from the language I used, Le_Lisp, which does not accept 32-bit integers.

bug ridden. This occurred when I thoroughly checked my factoring routines. I had simply forgotten to reduce the parameters of $\rho, p-1, \ldots$ after a factor was discovered ! The program was thus *asymptotically bugged*: When dealing with small numbers, I need maybe one large factor, but with large numbers, maybe two or more. This explained also the above mentioned problem with $p-1$ (because of the way the exponentiation routine was programmed, it wanted to find the first 1 in the binary expansion of a zero word).

And (not surprisingly ?), it began to work. The breakthrough occurred on October 6, when N_1 was reached, using $D = 97507$ (with $h = 36, g = 2$). After that, this was quite a quiet work, except that there happened to be a difficult client at one stage (namely N_{11}), requiring a D with $h = 56$ and $g = 2$. The building of the tower of primes was finally completed on November 8 at 830 pm (INRIA-Paris time).

Meantime, I had used one workstation for the second part of the process, proving the numbers to be primes. One week before the end of the first phase, I also used one of the SUN's on this $h/g = 28$ business, that is finding a root of a polynomial of degree 28 over a finite field with about 10^{991} elements. For that, I chose the most resistant SUN I could find. By this, I mean a station that was able to resist all network problems that could appear. Actually, this was a period of time where there was quite a lot of those. This computation took one week.

When the first phase ended, about 40 proving steps were done and I was able to launch the workstations on the remaining cases. On November 11, it was over, even the 28 degree stuff. It was it, I had sunk the Titanic, this time with an *ordinary* prime (as opposed to the Elliptic Mersenne Primes of [25]): The problem of testing 1000-digit numbers for primality was solved. Looking at the whole story, it took only one month and a half to do that. Moreover, it took only one week to come down from 700 digits to 10: This means that one can routinely test such numbers for primality. Some further experiments confirmed this [2].

The final result is a file of 500 kbytes consisting of the certificate of primality for $\mathcal{N}_{3539}$. This file can be sent to anyone who wants to check it using the protocols described in [26].

5.2 Technical details

In order to prove the primality of $\mathcal{N}_{3539}$, I used 12 SUN workstations, among which four 3/50, seven 3/60 and one 3/160 with a special chip designed for 512-bit multiplication [5]. Using the full power of the chip was done by using Montgomery's ideas on modular multiplications [22]: These ideas were only used for Pollard ρ, $p-1$ and pseudoprimality tests. The speedup for a modular exponentiation of 110 words of 32 bits is about 8.

The first phase took approximately 288 days of CPU (only one month and a half in real time). The second one 31 days of CPU. The total time is thus less than one year of CPU. The tower of primes consists of 162 numbers. In Figure 2, we print the number of digits of N_i versus the real time from the start of the job. The distribution of the *gains*, that is the number of digits we win in finding the following member is displayed in Figure 3: The mean value is 6.5, with minimum 0 and maximum 34. In

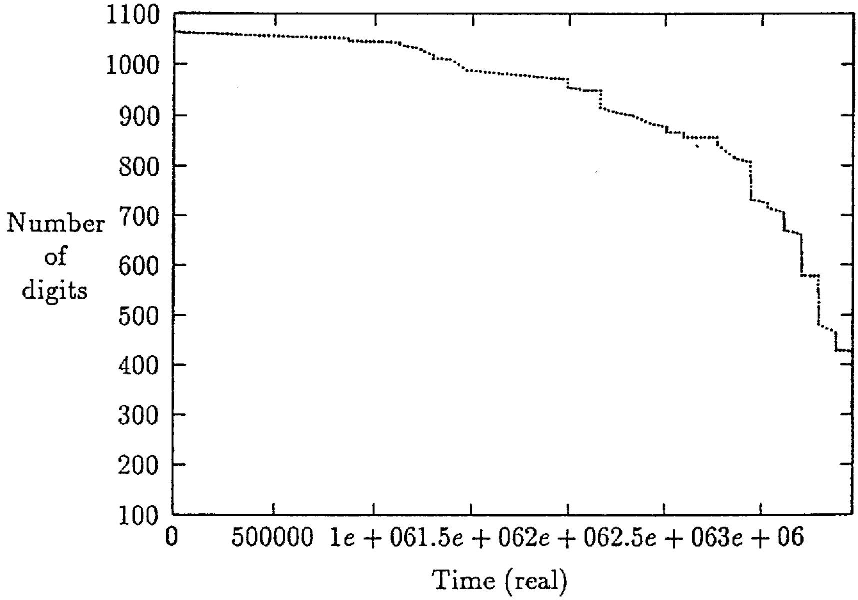

Figure 2: Number of digits reached vs. real time

Figure 4, we put the distribution of the values of h/g, the mean value being 3.49.

6 Conclusions

We see that ECPP in its distributed implementation is a very powerful tool to test arbitrary large numbers for primality. It should be able to deal with somewhat larger numbers (maybe with 1200 digits or so). The problem that is bound to arise is that there is a point where we need powerful factoring routines such as ECM. However, this would slow down the running time of the whole process. So it seems not possible to deal with 2000-digit numbers.

It should be noted that van der Hulst and Bosma finally succeeded in proving the same number to be prime (hopefully!). It took them [35] about three weeks and a half on a DEC 3100 (about five times faster than a SUN). They have made some improvements and now, it should just require one week and a half to do that size of number.

Acknowledgments. First of all, I thank the owners of the workstations that contributed to my record, namely L. Albert, L. Audoire, J.-J. Codani, V. Collette, P. Flajolet, P. Jacquet, P. Le Chenadec, M. Régnier, B. Serpette and P. Zimmermann. W. Bosma and M.-P. van der Hulst (and also A. K. Lenstra) must be thanked for their stimulating work to attack the supremacy of ECPP using the Jacobi Sums test.

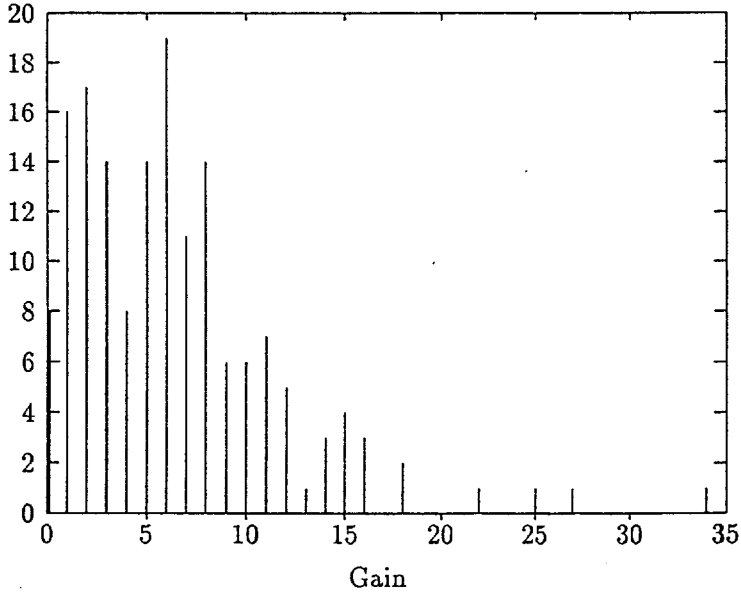

Figure 3: Number of digits gained at each step

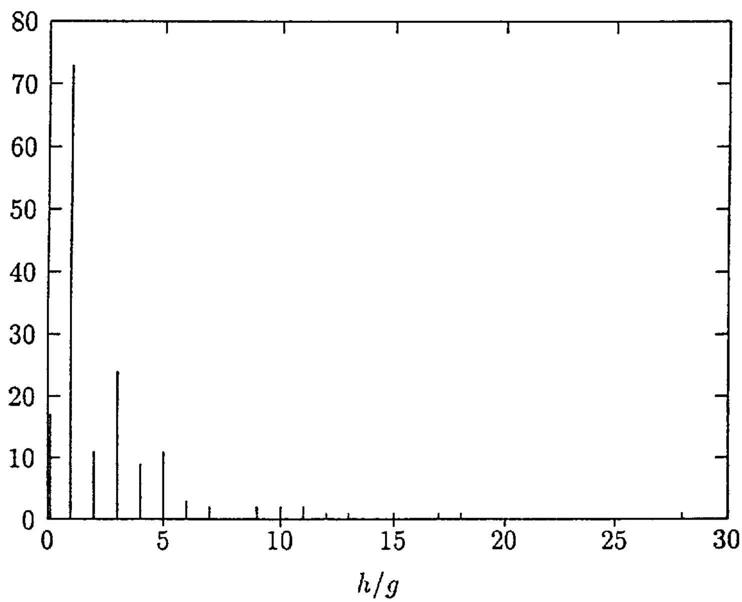

Figure 4: Distribution of h/g

Without the script-shells of M. Manasse, this job would have been less easy: special thanks to him, then. Thanks to R. Ehrlich who helped me modifying the above scripts and explained to me some of the magic properties of NFS. Thanks also to I. Vardi for (helpful or stylistic) comments about my manuscript.

Lastly, I'd like to thank the technical staff in Scanticon (where EUROCRYPT took place) for the wonderful job they did for me in doing my slides concerning some typical French comic character.

References

[1] L. M. ADLEMAN, C. POMERANCE, AND R. S. RUMELY. On distinguishing prime numbers from composite numbers. *Annals of Math. 117* (1983), 173–206.

[2] A. O. L. ATKIN AND F. MORAIN. Elliptic curves and primality proving. Rapport de Recherche 1256, INRIA, Juin 1990.

[3] P. T. BATEMAN, J. L. SELFRIDGE, AND S. S. WAGSTAFF, JR. The new Mersenne conjecture. *American Mathematical Monthly 96*, 2 (1989), 125–128.

[4] P. BEAUCHEMIN, G. BRASSARD, C. CRÉPEAU, C. GOUTIER, AND C. POMERANCE. The generation of random numbers that are probably prime. *J. Cryptology 1* (1988), 53–64.

[5] P. BERTIN, D. RONCIN, AND J. VUILLEMIN. Introduction to programmable active memories. In *Proc. of the Internat. Conf. on Systolic Arrays* (1989).

[6] W. BOSMA AND M.-P. VAN DER HULST. Faster primality testing. To appear in *Proc.* Eurocrypt '89.

[7] G. BRASSARD. *Modern Cryptology*, vol. 325 of *Lect. Notes in Computer Science*. Springer-Verlag, 1988.

[8] J. BRILLHART, D. H. LEHMER, J. L. SELFRIDGE, B. TUCKERMAN, AND S. S. WAGSTAFF, JR. *Factorizations of* $b^n \pm 1$, $b = 2,3,5,6,7,10,11,12$ *up to high powers*, 2 ed. No. 22 in Contemporary Mathematics. AMS, 1988.

[9] J. W. S. CASSELS. Diophantine equations with special references to elliptic curves. *J. London Math. Soc. 41* (1966), 193–291.

[10] H. COHEN AND A. K. LENSTRA. Implementation of a new primality test. *Math. Comp. 48*, 177 (1987), 103–121.

[11] H. COHEN AND H. W. LENSTRA, JR. Primality testing and Jacobi sums. *Math. Comp. 42*, 165 (1984), 297–330.

[12] C. COUVREUR AND J. QUISQUATER. An introduction to fast generation of large prime numbers. *Philips J. Research 37* (1982), 231–264.

[13] S. Goldwasser and J. Kilian. Almost all primes can be quickly certified. In *Proc. 18th STOC* (Berkeley, 1986), pp. 316–329.

[14] D. Gordon. Strong primes are easy to find. In *Proc. Eurocrypt '84* (1984), Springer, pp. 216–223.

[15] J.-C. Hervé, F. Morain, D. Salesin, B. Serpette, J. Vuillemin, and P. Zimmermann. Bignum: A portable and efficient package for arbitrary precision arithmetic. Rapport de Recherche 1016, INRIA, avril 1989.

[16] D. Husemöller. *Elliptic curves*, vol. 111 of *Graduate Texts in Mathematics*. Springer, 1987.

[17] D. E. Knuth. *The Art of Computer Programming: Seminumerical Algorithms*. Addison-Wesley, 1981.

[18] A. K. Lenstra. *Cryptology and computational number theory*. AMS, 1989, ch. Primality testing. Lecture Notes, August 6–7, 1989, Boulder, Colorado.

[19] A. K. Lenstra and M. S. Manasse. Factoring by electronic mail. To appear in Proc. Eurocrypt '89, 1989.

[20] H. W. Lenstra, Jr. Elliptic curves and number theoretic algorithms. Tech. Rep. Report 86-19, Math. Inst., Univ. Amsterdam, 1986.

[21] H. W. Lenstra, Jr. Factoring integers with elliptic curves. *Annals of Math. 126* (1987), 649–673.

[22] P. L. Montgomery. Modular multiplication without trial division. *Math. Comp. 44*, 170 (April 1985), 519–521.

[23] P. L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Math. Comp. 48*, 177 (January 1987), 243–264.

[24] F. Morain. Atkin's test: news from the front. To appear in Proc. Eurocrypt '89.

[25] F. Morain. Elliptic curves, primality proving and some Titanic primes. To appear in Actes des Journées Arithmétiques, Luminy 1989.

[26] F. Morain. Implementation of the Atkin-Goldwasser-Kilian primality testing algorithm. Rapport de Recherche 911, INRIA, Octobre 1988.

[27] F. Morain. Construction of Hilbert class fields of imaginary quadratic fields and dihedral equations modulo p. Rapport de Recherche 1087, INRIA, Septembre 1989.

[28] F. Morain. Résolution d'équations de petit degré modulo de grands nombres premiers. Rapport de Recherche 1085, INRIA, Septembre 1989.

[29] F. MORAIN AND J. OLIVOS. Speeding up the computations on an elliptic curve using addition-subtraction chains. Rapport de Recherche 983, INRIA, Mars 1989.

[30] D. A. PLAISTED. Fast verification, testing and generation of large primes. *Theoretical Computer Science 9* (1979), 1–16.

[31] H. C. POCKLINGTON. The determination of the prime and composite nature of large numbers by Fermat's theorem. *Proc. Cambridge Philos. Soc. 18* (1914-1916), 29–30.

[32] R. SCHOOF. Elliptic curves over finite fields and the computation of square roots mod p. *Math. Comp. 44* (1985), 483–494.

[33] R. D. SILVERMAN. Parallel implementation of the quadratic sieve. *The Journal of Supercomputing 1*, 3 (1987).

[34] J. T. TATE. The arithmetic of elliptic curves. *Inventiones Math. 23* (1974), 179–206.

[35] M.-P. VAN DER HULST. Timings for $(2^{3539}+1)/3$. Email, February 1990.

[36] H. C. WILLIAMS AND H. DUBNER. The primality of $R1031$. *Math. Comp. 47*, 176 (1986), 703–711.

[37] M. C. WUNDERLICH. A performance analysis of a simple prime-testing algorithm. *Math. Comp. 40*, 162 (1983), 709–714.

[38] S. YATES. Titanic primes. *J. Recr. Math. 16* (1983/84), 250–260.