IMPLEMENTING THE ASYMPTOTICALLY FAST VERSION OF THE ELLIPTIC CURVE PRIMALITY PROVING ALGORITHM (VERY PRELIMINARY DRAFT)

F. MORAIN

ABSTRACT. The elliptic curve primality proving algorithm is one of the fastest practical algorithm for proving the primality of large numbers. Its running time cannot be proven rigorously, but heuristic arguments show that it should run in time $\tilde{O}((\log N)^5)$ to prove the primality of N. An asymptotically fast version of it, attributed to J. O. Shallit is supposed to run in time $\tilde{O}((\log N)^4)$. The aim of this article is to describe this version in more details, leading to an implementation that is able to handle number with several thousands decimal digits. The distributed version of the algorithm was able to prove the primality of a number with 7124 decimal digits.

1. INTRODUCTION

From the work of Agrawal, Kayal and Saxena [2], we know that determining the primality of an integer N can be done in proven deterministic polynomial time $\tilde{O}((\log N)^{10.5})$. H.-W. Lenstra and C. Pomerance have announced a version in $\tilde{O}((\log N)^6)$. This algorithm is for the time being too slow to prove the primality of relatively large primes. Building on the work of P. Berrizbeitia [6], D. Bernstein [5] and P. Mihăilescu & R. Mocenigo [18], independently, have given improved probabilistic versions with a claim of proven complexity of $\tilde{O}((\log N)^4)$, reusing classical cyclotomic ideas that originated in the Jacobi sums test [1, 7]. For more on primality before AKS, we refer the reader to [8]. For the recent developments, see [4].

On the other hand, the elliptic curve primality proving algorithm [3] has been used for years to prove the primality of always larger numbers, see for instance the record to 6959 digits by Hans Rosenthal* using the implementation PRIMO of M. Martin†. The algorithm has a heuristic running time of $O((\log N)^{6+\varepsilon})$ in the naive version, and $\tilde{O}((\log N)^5)$ using fast multiplication techniques. In the course of writing [21], the author rediscovered the article [17], in which an asymptotically fast version of ECPP is described. This version, attributed to J. O. Shallit, has a heuristic running time of $\tilde{O}((\log N)^4)$.

To the best of the author's knowledge, this fast version has never been implemented. Due to the large numbers now being dealt with and the stimulating effect of both the AKS thread and the fair-play concurrence of PRIMO, the author realized that implementing this fast version was now at hand, helped by recent work on the computation of defining polynomials of Hilbert class fields. The aim of this paper is to describe the fast version of ECPP, give a heuristic analysis of it and describe its implementation. We also give the timings obtained with some large numbers, starting with the proof of the primality of $2177^{580} + 580^{2177}$, a 6016 decimal digit number taken from P. Leyland's tables‡.

Date: July 16, 2003.

The author is on leave from the French Department of Defense, Délégation Générale pour l'Armement.

*Email to the NMBRTHRY mailing list, July 13th, 2003.

†`http://www.ellipsa.net/`

‡`http://research.microsoft.com/~pleyland/primes/xyyx.htm`

2. The basic ECPP algorithm

2.1. **Presentation of the algorithm.** We present a rough sketch of the algorithm, enough for us to estimate its complexity. We do not insist on what happens if one of the steps fails, revealing the compositeness of N. More details are to be found in [3].

We want to prove that N is prime. The algorithm runs as follows:

[Step 1.] Find an imaginary quadratic field $\mathbf{K} = \mathbb{Q}(\sqrt{-D})$ of discriminant $-D$, $D > 0$, such that

$$4N = U^2 + DV^2 \tag{1}$$

in rational integers U and V.

[Step 2.] Compute $m = N + 1 - U$; if this number is not twice a probable prime N', go to Step 1.

[Step 3.] Build an elliptic curve $\overline{E}$ over $\overline{\mathbb{Q}}$ having complex multiplication by the ring of integers of $\mathbf{K}$.

[Step 4.] Reduce $\overline{E}$ modulo N to get a curve E.
[Step 5.] Find P on E such that $[N']P \neq O_E$. If this cannot be done, then N is composite, otherwise, it is prime.

[Step 6.] Set $N = N'$ and go back to Step 1.

Clearly, we need $\log N$ steps for proving the primality of N. Step 5 is easy to analyze, since an exponentiation on a curve costs $\tilde{O}((\log N)^2)$, and we need very few points on average.

2.2. **Analysis of Step 1.** The choice of D is crucial. Remember that a fundamental discriminant may be written as:

$$-D = \prod_{i=1}^{t} q_i^*$$

where all q_i^*'s are distinct and q_i^* is either -4 or ± 8, or $q_i^* = (-1/q_i)q_i$ for any prime q_i. The probability that a prime p splits in $\mathbf{K}$ is $1/(2h)$ where $h = h(-D)$ is the class number of $\mathbf{K}$. Using Gauss's theory of genera of forms, it is known that if $\left(\frac{q_i^*}{p}\right) = 1$ for all i's, then the probability of p splitting is g/h.

Given D, testing whether (1) can be satisfied costs the computation of $\sqrt{-D} \bmod N$, followed by an application of Cornacchia's algorithm (see [22]). The first task costs $\tilde{O}((\log N)^2)$ operations, the second a mere $\tilde{O}(\log N)$.

The probability that D is such that N splits is $O(1/h) = \tilde{O}(D^{1/2})$ by Siegel's theorem. Putting $D = O((\log N)^a)$, we expect to need $\tilde{O}((\log N)^{a/2})$ discriminants.

2.3. **Analysis of Step 2.** In the event that equation (1) is solvable, then we need check that $m = 2N'$ and test N' for primality, which costs again some $\tilde{O}((\log N)^2)$.

The heuristic probability of m being of the given form is $O(1/\log N)$. Though quite realistic, it is impossible to prove, given the current state of the art in analytical number theory. This means that we expect to need at least $O(\log N)$ splitting D's.

Gathering the results of the preceding sections, we see that we need $\tilde{O}((\log N)^{a/2+1})$ discriminants, which readily implies that $a \geq a/2 + 1$ or $a \geq 2$. The cost of Step 1 and 2 is:

$$\tilde{O}((\log N)^{a/2+1+2}) + \tilde{O}((\log N)^3).$$

Setting the minimal possible value of a to the value 2, we get that this phase takes $\tilde{O}((\log N)^4)$.

2.4. **Analyzing the curve building step.** The original version of Step 3 is to build the Hilbert class field $\mathbf{K}_H$ of $\mathbf{K}$, an extension of degree h that we realize via the computation of the minimal polynomial $H_D(X)$ of the special values of the classical j-invariant at quadratic integers. More precisely, we can view the class group $Cl(-D)$ of $\mathbf{K}$ as a set of primitive reduced quadratic forms of discriminant $-D$. If (A,B,C) is such a form, with $B^2-4AC=-D$, then

$$H_D(X) = \prod_{(A,B,C)\in Cl(-D)} \left(X - j((-B+\sqrt{-D})/(2A))\right).$$

In [9], it is argued that the height of this polynomial is well approximated by the quantity:

$$\pi\sqrt{D} \sum_{[A,B,C]\in Cl(-D)} \frac{1}{A},$$

In other words, the height is $\tilde{O}(h)=\tilde{O}(D^{1/2})$.

Couveignes et al.

Evaluating the roots of $H_D(X)$ and building this polynomial can be done in $\tilde{O}(h^2)$ operations. This could be seen as a precomputation, since it does not depend on N.

Now, we need find a root of $H_D(X)$ modulo N. This can be done with the Cantor-Zassenhaus algorithm. Briefly, we split $H_D(X)$ by computing $\gcd(X^{(N-1)/2}-1, H_D(X)) \bmod N$. We need $\log h$ of these, and each test needs $\tilde{O}((\log N)M(h)(\log N))$ operations (exponential, products of degree h polynomials modulo N).

The overall cost is thus $\tilde{O}(h^2)+\tilde{O}((\log N)M(h)(\log N)) = \tilde{O}((\log N)^2+(\log N)^3) = \tilde{O}((\log N)^3)$.

2.5. **Remarks on the implementation.** In the original version of ECPP, the D's are best considered in decreasing order of g/h, starting with all idoneal numbers for which the ratio is 1. This corresponded to a situation where it was thought better to precompute a large set of defining polynomials, for a large number of D's. Nowadays, with technology helping a bit, it appears best not to precompute anything.

3. The fast version of ECPP

3.1. **Presentation.** When dealing with large numbers, all the time is spent in the finding of D, which means that a lot of squareroots modulo N must be computed. A first way to reduce the computations, alluded to in [3, §8.4.3], is to accumulate squareroots, and reuse them, at the cost of some multiplications. For instance, if one has $\sqrt{-3}$ and $\sqrt{5}=\sqrt{-20}/\sqrt{-4}$, then we can build $\sqrt{-15}$, etc.

A better way that leads to the fast version consists in precomputing a basis of small squareroots and build discriminants from this basis. More formally, we replace Step 1. by Step 1'. as follows:

[Step 1'.]

1.1. Find the r smallest primes q^* such that $\left(\frac{q^*}{N}\right)=1$, yielding $\mathcal{Q}=\{q_1^*,q_2^*,\ldots,q_r^*\}$.

1.2. Precompute all $\sqrt{q^*} \bmod N$ for $q^*\in\mathcal{Q}$.

1.3. For all subsets $\mathcal{S}=\{q_{i_1}^*,q_{i_2}^*,\ldots,q_{i_s}^*\}$ of $\mathcal{Q}$ for which $\prod_{q^*\in\mathcal{S}} q^* = -D < 0$, try to solve equation (1).

Given r, there is at most 2^r subsets, which means that we can take r relatively small. The cost of step 1.3 is now essentially that of Cornacchia's algorithm.

3.2. **Analysis.** Let s be a fixed integer $\leq r$. There are $O(r^s)$ suitable discriminants. A typical discriminant has size $O(q_r^s)$ and class number $O(q_r^{s/2})$. Therefore, the probability of finding one splitting prime is $O(1/q_r^{s/2})$. Since we look for some m to be twice a prime, we expect to have to look at $O(q_r^{s/2}\log N)$ numbers, which implies

$$r^s \gg q_r^{s/2}\log N.$$

Since $q_r = O(r\log r)$, we get that $s \geq 2$.

Now, we precompute r squareroots for a cost of $\tilde{O}(r(\log N)^2)$; for $O(q_r^{s/2}\log N)$ discriminants, we need reconstruct $\sqrt{-D}$ and apply Cornacchia, which gives us

$$\tilde{O}(q_r^{s/2}(\log N)^2).$$

The remaining part of the analysis carries over and we find a cost of

$$\tilde{O}(r(\log N)^2) + \tilde{O}(q_r^{s/2}(\log N)^2) + \tilde{O}((\log N)^3).$$

We see that if $r = O(\log N)$, $s = 2$, then we are done and the cost is $\tilde{O}((\log N)^3)$, leading to the claimed total time of $\tilde{O}((\log N)^4)$.

4. Implementation

4.1. **Updating ECPP.**

4.1.1. *Algorithmic improvements.* In practice, we may use $m = cN'$ for some small c obtained by sieving (see [20] for tricks). Here, we sieve up to 10^7.

Proving larger and larger numbers forces us to use larger and larger D's, leading to larger and larger polynomials H_D. For this to be doable, new invariants had to be used, so as to minimize the size of the minimal polynomials. This task was done using Schertz's formulation of Shimura's reciprocity law [23], with the invariants of [11] as demonstrated in [9] (alternatively see [14, 13]). Galois theory came in handy, for dealing with composite class numbers h that could be split into factors and the corresponding extension towers built, as demonstrated in [16, 10].

Note that replacing j by other functions does not the change the complexity of the algorithm, though it is crucial in practice.

4.1.2. *Improving the program.* The new implementation uses GMP[§] for the basic arithmetic, which enables one to use `mpfr` [15] and `mpc` [12], thus leading to a complete program that can compute polynomial H_D's on the fly, contrary to the author's implementation of ECPP, prior to version 11.0.5. This turned out to be the key for the new-born program to compete with the old one.

Here are some timings on a 629 word number on a Xeon:
$\sqrt{-D}$: 34 sec;
Cornacchia: 0.04 sec;
SieveDWithTabCompactMax: 18 sec;
EcppProbablePrimeTest: 34 sec.

Everything in ECPP can be distributed (see [19] for the first description): Step 1, 3 and 5 are best choices, and the idea is to use a master/slave approach. The present version uses MPICH for this task. Distribution gains a factor proportional to the number of processors, in good cases.

[§] http://www.swox.com/gmp/

5. A compact pseudocertificate

One advantage of ECPP is the certificate, which consists in $\log N$ curves and points, for a total size of $O((\log N)^2)$, and that can be checked in time $\tilde{O}((\log N)^3)$.

We describe here a short compact version of it that could be used to reproduce the sequence of intermediate primes. The idea is just to give the list of D's used, together with the sign of U. Knowing this and the fact that we sieve up to certain bound, then we can imagine reconstructing the whole proof, though with some effort. It can serve as a convincing element that the author of the proof really went through.

Let $D > 4$. Then $D-$ means that $m = N+1+U$ was used, as $D+$ means $m = N+1-U$. There are three special cases: When the $N-1$ test (resp. $N+1$ test) was used, we write $1-$ (resp. $1+$).

When $D = 3$, there are 6 possible cardinalities. In that case, we can write $4N = L^2 + 27M^2$, $L \not\equiv 0 \bmod 3$, $L > 0$, $M > 0$. We decide to code the cardinalities as in:

cardinality	code
$N+1-L$	a
$N+1+L$	b
$N+1-(L+9M)/2$	c
$N+1+(L+9M)/2$	d
$N+1-(L-9M)/2$	e
$N+1+(L-9M)/2$	f

When $D = 4$, we know that there are 4 possible cardinalities. Write $N = x^2 + 4y^2$, with x odd, $x > 0$, $y > 0$. We decide to code the cardinalities as in:

cardinality	code
$N+1-2x$	a
$N+1+2x$	b
$N+1-4y$	c
$N+1+4y$	d

6. Some numbers proven

All computations were done on a cluster of 6 biprocessors Xeon at 2.66 MHz. We took the following numbers from the tables of P. Leyland¶.

¶`http://research.microsoft.com/~pleyland/primes/xyyx.htm`

x	2177	2589	2551
y	580	218	622
#dd	6016	6055	7127
when	030513-030604	030606-030617	030618-030714
# steps	801	736	965
CPU	164 days	103 days	235 days
WCT	164×1.2	103×1.1	235×1.3
$\sqrt{-D}$	81 days	30 days	74 days
Time 2nd phase	28 days	21 days	55 days
Time H_D	2951 sec	1686 sec	18451 sec
Time CZ	26 days	20 days	50 days
Checking	25 hours	22 hours	45 hours
$\max h$	1980	2080	3312
$\max D$	7,749,263	19,076,479	52,396,648

Putting $\theta = 7127/6055 \approx 1.177$, we find that $\theta^4 \approx 1.91$ is close to 2, which is coherent with the claimed complexity.

The largest number required 625 GHz-days, and should be compared to the 738 GHz-days for the 6959 decimal digit number of Hans Rosenthal.

7. Conclusions

We have described in greater details the fast version of ECPP. We have demonstrated its efficiency. We have thus settled some benchmarks for the fast AKS versions to compete and stimulate further work on the topic.

Acknowledgments. The author wants to thank D. Bernstein for stimulating emails on the existence and analysis of fast ECPP. Thanks also to P. Gaudry for never ending discussions on how close we are to infinity, as far as fast algorithms are concerned.

References

[1] L. M. Adleman, C. Pomerance, and R. S. Rumely. On distinguishing prime numbers from composite numbers. *Ann. of Math. (2)*, 117:173–206, 1983.

[2] M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. Preprint; available at `http://www.cse.iitk.ac.in/primality.pdf`, August 2002.

[3] A. O. L. Atkin and F. Morain. Elliptic curves and primality proving. *Math. Comp.*, 61(203):29–68, July 1993.

[4] D. Bernstein. Proving primality after Agrawal-Kayal-Saxena. `http://cr.yp.to/papers/aks.ps`, January 2003.

[5] D. Bernstein. Proving primality in essentially quartic expected time. `http://cr.yp.to/papers/quartic.ps`, January 2003.

[6] P. Berrizbeitia. Sharpening "Primes is in P" for a large family of numbers. `http://arxiv.org/abs/math.NT/0211334`, November 2002.

[7] H. Cohen and H. W. Lenstra, Jr. Primality testing and Jacobi sums. *Math. Comp.*, 42(165):297–330, 1984.

[8] R. Crandall and C. Pomerance. *Prime numbers – A Computational Perspective*. Springer Verlag, 2000.

[9] A. Enge and F. Morain. Comparing invariants for class fields of imaginary quadratic fields. In C. Fieker and D. R. Kohel, editors, *Algorihmic Number Theory*, volume 2369 of *Lecture Notes in Comput. Sci.*, pages 252–266. Springer-Verlag, 2002. 5th International Symposium, ANTS-V, Sydney, Australia, July 2002, Proceedings.

[10] A. Enge and F. Morain. Fast decomposition of polynomials with known Galois group. In M. Fossorier, T. Høholdt, and A. Poli, editors, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, volume 2643 of *Lecture Notes in Comput. Sci.*, pages 254–264. Springer-Verlag, 2003. 15th International Symposium, AAECC-15, Toulouse, France, May 2003, Proceedings.

[11] A. Enge and R. Schertz. Constructing elliptic curves from modular curves of positive genus. Preprint. Available at `http://www.lix.polytechnique.fr/Labo/Andreas.Enge/`, March 2003.

[12] A. Enge and P. Zimmermann. `mpc` — a library for multiprecision complex arithmetic with exact rounding, 2002. Version 0.4.1, available from http://www.lix.polytechnique.fr/Labo/Andreas.Enge.

[13] A. Gee. Class invariants by Shimura's reciprocity law. *J. Théor. Nombres Bordeaux*, 11:45–72, 1999.

[14] A. Gee and P. Stevenhagen. Generating class fields using Shimura reciprocity. In J. P. Buhler, editor, *Algorithmic Number Theory*, volume 1423 of *Lecture Notes in Comput. Sci.*, pages 441–453. Springer-Verlag, 1998. Third International Symposium, ANTS-III, Portland, Oregon, june 1998, Proceedings.

[15] G. Hanrot, V. Lefèvre, and P. Zimmermann et. al. `mpfr` — a library for multiple-precision floating-point computations with exact rounding, 2002. Version contained in GMP. Available from http://www.mpfr.org.

[16] G. Hanrot and F. Morain. Solvability by radicals from an algorithmic point of view. In B. Mourrain, editor, *Symbolic and algebraic computation*, pages 175–182. ACM, 2001. Proceedings ISSAC'2001, London, Ontario.

[17] A. K. Lenstra and H. W. Lenstra, Jr. Algorithms in number theory. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A: Algorithms and Complexity, chapter 12, pages 674–715. North Holland, 1990.

[18] P. Mihăilescu and R. Avanzi. Efficient quasi-deterministic primality test improving AKS. Available from `http://www-math.uni-paderborn.de/~preda/papers/myaks1.ps`, April 2003.

[19] F. Morain. Distributed primality proving and the primality of $(2^{3539}+1)/3$. In I. B. Damgård, editor, *Advances in Cryptology – EUROCRYPT '90*, volume 473 of *Lecture Notes in Comput. Sci.*, pages 110–123. Springer–Verlag, 1991. Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques, Aarhus, Denmark, May 21–24, 1990.

[20] F. Morain. Primality proving using elliptic curves: an update. In J. P. Buhler, editor, *Algorithmic Number Theory*, volume 1423 of *Lecture Notes in Comput. Sci.*, pages 111–127. Springer-Verlag, 1998. Third International Symposium, ANTS-III, Portland, Oregon, june 1998, Proceedings.

[21] F. Morain. La primalité en temps polynomial [d'après Adleman, Huang ; Agrawal, Kayal, Saxena]. Séminaire Bourbaki, exposé 917, March 2003.

[22] A. Nitaj. L'algorithme de Cornacchia. *Exposition. Math.*, 13:358–365, 1995.

[23] R. Schertz. Weber's class invariants revisited. *J. Théor. Nombres Bordeaux*, 14:325–343, 2002.

(F. Morain) LIX École Polytechnique, CNRS/UMR 7650, INRIA/Futurs, F-91128 Palaiseau CEDEX, FRANCE

E-mail address, F. Morain: `morain@lix.polytechnique.fr`