FASTER PRIMALITY TESTING

extended abstract

Wieb Bosma and Marc-Paul van der Hulst

Mathematisch Instituut
Universtiteit van Amsterdam
Roetersstraat 15
1018 WB Amsterdam
The Netherlands

Acknowledgement. Research was done while the authors were supported by the Nederlandse organisatie voor wetenschappelijk onderzoek NWO.

Abstract

Several major improvements to the Jacobi sum primality testing algorithm will speed it up in such a way that proving primality of primes of up to 500 digits will be a matter of routine. Primes of about 800 digits will take at most one night on a Cray.

Primality Testing and Factoring

Primality testing is one of two closely related classical problems in computational number theory, the other being that of *factoring integers*. Usually, if a positive integer n is composite, it is easy to find a proof for that. Since such a proof generally does not provide factors of n, for composite numbers the problem of factoring n remains. But if a number does not seem to be composite, one would like to find a *proof* for its primality; this is the object of primality testing and the subject of this paper. A *primality test* is an algorithm that gives a rigorous proof for the primality of prime numbers; one inputs an integer and the algorithm either yields a proof that n is prime, or it fails, indicating that n must be composite.

In this paper we describe several major improvements to the Jacobi sum primality test. As a consequence we will soon be able to prove the primality of prime numbers of up to many hundreds of digits routinely. Our estimates show that in the worst case proofs for 800 digit primes will take at most one night on a Cray. Furthermore, our implementation allows distribution on almost any number of processors; in this way we can achieve an m fold speedup by running the test on m identical machines.

There exist very fast *compositeness tests*, also called pseudo-prime tests, that on input n either give a proof for the fact that n is composite, or tell you that n is probably prime. A proof of compositeness usually consists of exhibiting an integer, called a *witness to the compositeness* of n, which has a special property (for instance concerning its order in the multiplicative group modulo n) that no integer can satisfy if n were prime. In particular, these proofs for compositeness do not give any clue as to the *divisors* of n; finding these is very hard in general, which is the raison d'être of the 'factoring industry'.

In the case of Rabin's compositeness test [R] the probability that a random integer is a witness to the compositeness of some composite number, is at least $\frac{3}{4}$. Since such a test can be repeated independently as many times as one would like, the probability that a composite integer is declared

J.J. Quisquater and J. Vandewalle (Eds.): Advances in Cryptology - EUROCRYPT '89, LNCS 434, pp. 652-656, 1990.

probably prime can be made arbitrarily small.

So the main problem in primality testing is that of proving the correctness of an answer that is easily obtained; in factoring it is just the other way around: there it is very hard to obtain the result (a factorization), but one checks its validity immediately (by multiplying out the factors).

But if one is willing to accept a small probability of giving the wrong answer, it is easy to answer the question whether a given integer n is a prime or a composite number. Thus the following paradoxical situation arises: for most practical purposes, one is perfectly happy with pseudo-prime tests, but this is mathematically unsatisfactory; on the other hand, it is easy to state sufficient conditions for primality, but it is much harder to make these criteria practical, i.e., to devise an efficient test! It should be remarked however, that primality testing is commonly regarded as "easier" than factoring, in terms of computational complexity.

To substantiate this belief, we mention here that under the assumption of some unproved hypotheses from analytic number theory, viz. sufficient generalized Riemann hypotheses, for every composite n there exists a witness a for the compositeness of n smaller than $2(\log n)^2$ (cf. [Ba]). Thus, under these hypotheses, primality testing is "polynomial time" [Mi].

Primality Tests

Two types of primality test ought to be distinguished: firstly, those tests that work only for primes with special arithmetic properties, and secondly, the general purpose type tests. Usually tests of the first type exploit divisibility properties of $n-1$ or $n+1$, more generally of $n^w - 1$ for small values of w. The classical examples give criteria for Fermat primes (i.e., primes of the form $2^{2^k}+1$) and Mersenne primes (of the form $2^k - 1$) respectively. In both cases a property is used that is necessary as well as sufficient and that can be checked very quickly. It is these type of test that make headlines, because they are used to find gigantic primes, of up to tens of thousands of digits.

We will call these tests of *Lucas-Lehmer* type. In general they give a sufficient criterion for the primality of n that is applicable in case enough factors of $n-1, n^2-1, \ldots$ can be found; here "enough factors" means that their product exceeds $\sqrt{n}$. Therefore these tests will only work for primes with very special arithmetic properties. Since they depend on the (hard) problem of factoring, there scope for general purposes is limited; in particular problems arose for certain primes of around 80 digits, for which not enough information on divisors could be gathered to complete a proof.

We turn to general purpose primality tests. The straightforward method of proving the primality of n by showing that no prime number smaller than $\sqrt{n}$ divides n, using trial division, rapidly becomes too time-consuming with increasing n, and a table of all primes up to $\sqrt{n}$ is needed. In fact, the trial division method can be employed quite efficiently for sieving out all composite integers up to a given bound, thus producing tables of primes; this is known as the sieve of Eratosthenes.

The first practical general purpose algorithm for primality proving was the *Jacobi sum test*. Based on observations made by Adleman, Pomerance and Rumely [APR], it was made practical by improvements of Cohen and H.W. Lenstra [CL]; in the implementation of A.K. Lenstra and Cohen [LC] it can routinely handle primes up to 212 digits and yields primality proofs for such numbers within 2 minutes on a Cray. Basically one restricts the possible divisors of the integer n to at most t different residue classes modulo s, for certain auxiliary integers t and s. If $s > \sqrt{n}$, then at least one divisor of n must be among these residues; if also their number t is not too large, one can thus prove the primality of n by showing that *none* of these residues does in fact divide n. It was proved that his gives rise to a subexponential algorithm. Below we will give a somewhat more detailed description of this algorithm.

Here we should also mention an important idea of H.W. Lenstra [L]. He proved that there can be at most 11 divisors of n in any given residue class modulo s if s exceeds $\sqrt[3]{n}$. Moreover there is an efficient algorithm for finding them. Therefore both the bound to be exceeded by the product of

the factors found in the Lucas-Lehmer type tests, and the bound to be exceeded can be lowered to $\sqrt[3]{n}$, if one is willing to spend a little more time on checking the possibilities in each residue class. This idea was for instance used in proving the primality of the number all of whose 1031 decimal digits are equal to 1 (by means of Lucas-Lehmer type tests) [W].

In recent years, the theory of elliptic curves has been successfully applied to the problem of primality testing (as well as to factoring). Analogues of the Lucas-Lehmer type tests were devised using factorizations in the ring of integers of a quadratic number field that is the complex multiplication ring of an elliptic curve [Bo]. One should think of this as replacing the multiplicative group of integers modulo n by the group of points on certain elliptic curves .

An algorithm of Goldwasser and Kilian [GK] gives primality proofs for almost all primes in expected polynomial time. But it relies on an algorithm of Schoof [S] to compute the number of elements on an elliptic curve that – though polynomial – is considered to be too slow for practical purposes. A variant of this idea, working on hyperelliptic curves, by Adleman and Huang, [AH] yields primality proofs in expected polynomial time for all primes.

It seems that Atkin's method of using elliptic curves with complex multiplication for a general purpose test has proven to be practical (recently it was reported that it has been applied to primes of up to 564 digits [Mo]); in fact this formed a mayor incentive to improve the Jacobi sum algorithm as reported here, in order to let it maintain it's leading rôle. Very roughly, Atkin tries a list of elliptic curves until one is found for which the number of points on it, defined modulo n, is of the form kq, with k small and with q a number that is proven prime recursively. Although heuristically Atkin's algorithm is polynomial time, a rigorous analysis has not been given yet.

The Jacobi sum test

The Jacobi sum test can be roughly described as follows.

Select integers t and s such that $s = \prod q$ is the product of primes q with the property that $q - 1 | t$ and such that $s > \sqrt{n}$. For every pair (p^k, q), with q a prime divisor of s and p^k the highest power of the prime p dividing $q - 1$, perform a Jacobi sum test, which consists roughly of raising an element in $\mathbf{Z}[\zeta_{p^k}]/n\mathbf{Z}[\zeta_{p^k}]$ to the power n. Finally check that the integers $1 < r_i \leq \sqrt{n}$ do not divide n, where $r_i \equiv n^i \bmod s$ for $i = 1, 2, \ldots, t$.

It can be shown that, in order to get $s > \sqrt{n}$, it suffices to take $t = (\log n)^{O(\log\log\log n)}$. For instance, for proving the primality of integers up to 212 digits, one could take t equal to (a divisor of) 55440.

We have made practical improvements on this algorithm in several directions. In the first place, the Jacobi sum test can be combined with the Lucas-Lehmer type tests; roughly speaking, this means that for every factor found in $n^w - 1$ the bound that the auxiliary number s for the Jacobi sum part of the combined test need to exceed, can be lowered by the same factor. Since the (modified) Lucas-Lehmer type tests are usually much cheaper than the Jacobi sum tests, this can be a tremendous gain. Of course one should compare this gain to the time needed to find more factors in $n^w - 1$, for small values of w. Using heuristics on the expected size of the factors that are to be found, a reasonable decision can be made here.

Secondly, it is possible to reduce the amount of work done in carrying out the Jacobi sum tests. Instead of doing the n-th powerings in the extension rings $\mathbf{Z}[\zeta_{p^k}]/n\mathbf{Z}[\zeta_{p^k}]$ of degree $\phi(p^k)$ over $\mathbf{Z}/n\mathbf{Z}$, where ϕ is Euler's function, it is possible to work in ring extensions of degree equal to $order(n \in \mathbf{Z}/p^k\mathbf{Z})$, which is a divisor of $\phi(p^k)$, and which may be considerably smaller.

Thirdly, it is possible to combine several tests for pairs (p^k, q) into one larger test, provided that the primes p are different. The tests consist of n-th powerings of elements in a ring extension of degree u, which will be represented as polynomials of degree u on integer coordinates modulo n. Suppose that one test has to be done in an extension of degree u_1 and another in an extension of degree u_2; then they can be combined into one test in an extension of degree $\mathrm{lcm}(u_1, u_2)$. Of course that only makes sense if the combined test is cheaper; making the realistic assumption that multiplication is quadratic in the number of coordinates, combining the two tests is only advantageous if $\mathrm{lcm}^2(u_1, u_2) < u_1^2 + u_2^2$. But we are only able to deal with extensions of relatively small degree, and then it is easily seen that combining is only profitable if u_1 divides u_2 (or the other way around). In general, combinations should be made for degrees $u_1, u_2, \ldots, u_k$ with the property that every u_i divides $\max(u_1, u_2, \ldots, u_k)$. There is an easy, efficient procedure for finding the optimal combination, once the collection of all pairs (p^k, q) is known.

This combination method introduces another interesting optimalization problem: which choice of auxiliary numbers t and s leads to the least expensive collection of tests, i.e., of pairs (p^k, q)? Although some NP-complete parts of this problem prevented us from efficiently finding a solution that is guaranteed to be optimal, a procedure was found for generating a solution that is within a few percents of the optimal solution, in an amount of time that is negligible compared to the time saved in performing the rest of the algorithm using this solution.

Finally, Lenstra's idea of divisors in residue classes modulo $\sqrt[3]{n}$ applies here as well, which means that with some care the bound for s can be lowered to $\sqrt[3]{n}$.

Predictions

Although there has been no time yet to experiment extensively with the improved primality testing algorithm, some predictions can be made. We expect that in the very worst case, testing an integer of 800 digits for primality would take one *night* on a Cray. Here the worst case means that no factors are found for the Lucas-Lehmer part of the algorithm and moreover that the order of n is maximal modulo every divisor of the auxiliary number t. Also, the $\sqrt[3]{n}$ idea is not used here.

Experiments have shown that on the average, numbers of the same size will require about one third of the time to test the worst possible case; we expect that these experiments are reliable, even though they are only based on the optimalization part of the algorithm, since only the *size* of n and its residue class modulo the divisors of t determine the time needed for the Jacobi sum part of the test.

Taking into account that the algorithm is very well suited for parallelization (both of the time consuming steps, the Jacobi sum tests and the final trial divisions, can be performed in parallel), we predict that it will be possible, using this algorithm, to give primality proofs for random primes of up to 1000 digits in a few days, using either supercomputers or a network of small processors.

Thus the improved Jacobi sum test will once more prove to be the most powerful general purpose primality testing algorithm.

References

[AH] L.M. Adleman, M.A. Huang, *Recognizing primes in random polynomial time*, Proceedings of the nineteenth annual ACM symposium on the theory of computing (STOC), (1987), pp. 462-469.

[APR] L.M. Adleman, C. Pomerance and R. Rumely, *On distinguishing prime numbers from composite numbers*, Annals of Mathematics, 117 (1983), pp. 173-206.

[Ba] E. Bach, *Analytic methods in the analysis and design of number-theoretic algorithms*, MIT Press, (1985),

[Bo] W. Bosma, *Primality testing using elliptic curves*, Report 85-12, Universiteit van Amsterdam, (1985),

[CL] H. Cohen, H.W. Lenstra, Jr., *Primality testing and Jacobi sums*, Mathematics of Computation, **42** (1984), pp. 297-330.

[GK] S. Goldwasser, J. Kilian, *Almost all primes can be certified quickly*, Proceedings of the eighteenth annual ACM symposium on the theory of computing (STOC), (1986), pp. 316-329.

[LC] A.K. Lenstra, H. Cohen, *Implementation of a new primality test*, Mathematics of Computation, **48** (1987), pp. 103-121.

[L] H.W. Lenstra, Jr., *Divisors in residue classes*, Mathematics of Computation, **42** (1984), pp. 331-340.

[Mi] G.L. Miller, *Riemann's hypothesis and tests for primality*, J. Comp. Sys. Sci., **13** (1976), pp. 300-317.

[Mo] F. Morain, see update 2.2 to: *Factorizations to* $b^n \pm 1$, by Brillhart, Lehmer, Selfridge, Tuckerman and Wagstaff.

[R] M.O. Rabin, *Probabilistic algorithms for primality testing*, Journal of Number Theory, **12** (1980), pp. 128-138.

[W] H. Williams, H. Dubner, *The primality of R1031*, Mathematics of Computation, **47** (1986), pp. 703-711.