Primality Testing*

Richard P. Brent
Computing Laboratory
University of Oxford
clemson@rpbrent.co.uk

April 4, 2003

*Presented at Clemson, South Carolina, 4/4/2003.

 clemsont

Abstract

We consider the classical problem of testing if a given (large) number n is prime or composite. First we outline some of the efficient randomised algorithms for solving this problem.

For many years it has been an open question whether a deterministic polynomial time algorithm exists for primality testing, i.e. whether

"PRIMES is in P".

Recently Agrawal, Kayal and Saxena answered this question in the affirmative. They gave a surprisingly simple deterministic algorithm. We describe their algorithm, mention some improvements by Bernstein and Lenstra, and consider whether the algorithm is useful in practice.

Finally, as a topic for future research, we mention a conjecture that, if proved, would give a fast and practical deterministic primality test.

Outline

- Notation
- Complexity Classes
- Primes and Factorisation
- Fermat test
- Certificates
- Rabin-Miller
- Jabobi Sums
- ECPP and Adelman-Huang
- AKS
- Reliability
- The Ultimate Primality Test ?
- Generating Primes for Cryptography
- Conclusions

"O" and "O-tilde" Notation

As usual, we say that

$$f(n) = O(n^k)$$

if, for some c and n_0, for all $n \ge n_0$,

$$f(n) \le cn^k .$$

We say that

$$f(n) = \widetilde{O}(n^k)$$

if, for all $\varepsilon > 0$,

$$f(n) = O(n^{k+\varepsilon}) .$$

The "$\widetilde{O}$" notation is useful to avoid terms like $\log n$. For example, we can multiply n-bit numbers in $O(n^2)$ operations with the classical algorithm, but this can be reduced to $\widetilde{O}(n)$ if we use the Schönhage-Strassen algorithm.
It is easier to write

$$\widetilde{O}(n)$$

than the (more precise)

$$O(n \log n \log\log n) .$$

Complexity Classes

We give an informal definition of some complexity classes. See Motwani and Raghavan [21, §1.5] for the formal definitions.

P is the class of problems that have a *deterministic polynomial time algorithm*, i.e. an algorithm running in time $O(\lambda^k)$ on inputs of *length* λ, for some exponent k.

ZPP is the class of problems that have an error-free randomised algorithm running in expected time $O(\lambda^k)$.

RP is the class of problems that have a randomised algorithm running in expected time $O(\lambda^k)$, with (one-sided) error probability at most 1/2. co-RP is the same as RP, but permitting an error on the other side.

BPP is similar but allows errors (with probability at most 1/4) on both sides.

Reducing Probability of Error

Given a problem in RP, co-RP, or BPP, we can reduce the probability of error below any given $\varepsilon > 0$ by iterating $O(\log(1/\varepsilon))$ times with independent random choices at each iteration. In this context an iteration is called a "trial".

Containment Relationships

$$P \subseteq ZPP = RP \cap \text{co-}RP \subseteq \left\{ \begin{array}{c} RP \\ \text{co-}RP \end{array} \right\} \subseteq BPP \; .$$

It is not known if any of the inclusions is strict. It is *conjectured* that $BPP \subseteq NP \neq P$.

Unique Factorisation

We *never* encounter prime factorisations like

$$43235479271 = 8059 \times 5364869 = 4337 \times 9968983$$

because factorisation into prime powers is unique[1].

[1] Except for this example ?

Factoring Primes ?

> "The obvious mathematical breakthrough would be development of an easy way to factor large prime numbers."
>
> – Bill Gates[2]

By definition, primes do not have a nontrivial factorisation over the integers $\mathbf{Z}$. However, if we consider a larger domain, e.g. $\mathbf{Z}[i]$, then some primes *can* be factored, e.g.

$$5 = (2+i)(2-i) \; .$$

In fact, Gauss proved that an odd prime p can be written as a sum of two (integer) squares iff $p = 1 \mod 4$, and in this case

$$p = a^2 + b^2 = (a+bi)(a-bi)$$

gives a factorisation of p into a product of *Gaussian integers*.

[2] Bill Gates, Nathan Myhrvold and Peter M. Rinearson, *The Road Ahead*, Viking Press, 1995, p. 265.

An Algorithm for "Factoring" Primes

There is a nice ZPP algorithm for finding a and b, given any prime $p = 1 \mod 4$. This gives the factorisation into Gaussian integers:

$$p = (a + bi)(a - bi) .$$

It is usually called *Cornacchia's Algorithm* although it was discovered by H. Smith in 1885 – see Crandall and Pomerance [12, §2.3.4].

Randomisation is used to find a *quadratic non-residue* mod p. This can be done in expected polynomial time by random sampling, since half of the numbers $\{1, 2, \ldots, p-1\}$ are quadratic residues and the other half are non-residues. We can easily check if a number is a quadratic residue, using Gauss's law of quadratic reciprocity.

There is a deterministic polynomial time algorithm (due to Schoof) for computing a quadratic non-residue mod p. However, this algorithm takes time $O((\log p)^9)$.

Fermat's Little Theorem

If n is prime and a is any integer, then

$$a^n = a \mod n.$$

Thus, if we find a such that $a^n \neq a \mod n$, we can be sure that n is composite. We say that

"a is a *witness* to the compositeness of n".

Note: we can guarantee that n is composite *without knowing the factors of* n.

The converse of Fermat's little theorem is false: if $a^n = a \mod n$ we can not be sure that n is prime. There are infinitely many examples (called *Carmichael numbers*) of composite n for which a^n is always $a \mod n$. The smallest example is

$$561 = 3 \cdot 11 \cdot 17$$

and the number of Carmichael numbers up to N is at least of order $N^{2/7}$ (Alford, Granville and Pomerance [5]).

Certificates

A *certificate* for a property is some information that proves the property and can be verified in polynomial time.
For example, a "witness" a such that $a^n \neq a \mod n$ is a certificate of the compositeness of n.
Pratt [25] showed that every prime has a certificate. The idea is to write

$$p - 1 = p_1^{\alpha_1} \cdots p_\nu^{\alpha_\nu}$$

and give a *primitive root* a of p. This can be verified by checking that

$$a^{p-1} = 1 \mod p$$

and

$$a^{(p-1)/p_\beta} \neq 1 \mod p \text{ for } \beta = 1, \ldots, \nu .$$

We (recursively) give certificates for $p_1, \ldots, p_\nu$ unless they are sufficiently small (say in $\{2, 3, 5, 7\}$).

First Extension of Fermat

A slight extension of Fermat's little Theorem is useful for primality testing.

If $n = 2^k q + 1$ is an odd prime, and $0 < a < n$, then either $a^q = 1 \mod n$, or the sequence

$$\left(a^{2^j q} \mod n\right)_{j=0,1,\ldots,k}$$

ends with 1, and the value just preceding the first appearance of 1 must be $-1 \mod n$.

That is, the sequence looks like

$$\begin{array}{c} 1, 1, \ldots 1 \\ \text{or } -1, 1, \ldots, 1 \\ \text{or } ?, \ldots, ?, -1, 1, \ldots, 1 \end{array}$$

Proof: If $y^2 = 1 \mod n$ then $n|(y-1)(y+1)$.
Since n is prime, $n|(y-1)$ or $n|(y+1)$.
Thus $y = \pm 1 \mod n$. $\square$

This has been known for a long time.
Rabin [26, 27] proved its usefulness in a randomised algorithm.

The Rabin-Miller Algorithm

The extension of Fermat's little Theorem gives a *necessary* (but not sufficient) condition for primality of n. The [Artjuhov]-Rabin-Miller algorithm [15] checks if this condition is satisfied for a random choice of a, and returns "yes" if it is, "no" otherwise.

If n is prime, the answer is always "yes" (correct).

If n is composite, the answer is "no" with probability greater than 3/4, i.e. the probability of error is $< 1/4$ (this is a theorem of Rabin).

Thus we have an RP-algorithm for testing compositeness. We can say that compositeness is in RP, or (equivalently)

<u>primality is in co-RP</u>

The Rabin-Miller algorithm gives a certificate for compositeness, but not a certificate for primality.

Popular Primality-Testing Algorithms

Popular algorithms include:

- The *Jacobi Sums* algorithm of Adleman, Pomerance and Rumely.
- The *Elliptic Curve Primality Proving* algorithm ECPP of Atkin and Morain, based on a proposal by Goldwasser and Kilian [13].
- The Rabin-Miller algorithm.

These algorithms all have their good (and bad) points. We'll look at each more closely.

Jacobi Sums

The *Jacobi Sums* algorithm of Adleman, Pomerance and Rumely [2] runs in deterministic time

$$(\log n)^{O(\log\log\log n)}\ .$$

This is *almost* polynomial time[3].

In fact, we can be more precise: Odlyzko and Pomerance have shown that, for all large n, the running time is in

$$[(\log n)^{A\log\log\log n},\ (\log n)^{B\log\log\log n}]\ ,$$

where A, B are positive constants.

The Jacobi sums algorithm is deterministic and practical – it has been used for numbers of at least 3395 decimal digits (Mihailescu – 6.5 days on 500 Mhz Dec Alpha).

[3] While it has been *proved* that $\log\log\log n$ tends to infinity with n, it has never been *observed* doing so [Pomerance].

ECPP

The *Elliptic Curve Primality Proving* algorithm ECPP of Atkin and Morain [7, 20], runs in *expected* polynomial time under some plausible assumptions, but the time bound has not been proved rigorously (there may be a thin exceptional set). Empirically the time is $\widetilde{O}(\log^5 n)$.

ECPP produces a certificate of primality that can be checked in deterministic polynomial time $\widetilde{O}(\log^3 n)$.

ECPP is a Las Vegas algorithm – the running time is random but the result is error-free.

ECPP is practical and has been used to prove primality of numbers of at least 5878 decimal digits (Gomez – 22 weeks on AMD XP1800+).

In practice ECPP is comparable to the Jacobi Sums algorithm, but ECPP has the advantage of producing an easily-checked certificate.

The Adelman-Huang Algorithm

There is a complicated algorithm[4], due to Adleman and Huang [1], that gives a certificate for primality in *expected* polynomial time. Thus

<u>primality is in RP</u>

It follows from Adelman-Huang and Rabin-Miller that

<u>primality is in $RP \cap$ co-$RP = ZPP$</u>.

The reason why we can prove that the Adelman-Huang algorithm runs in expected polynomial time, but can not do this for ECPP, is that the Hasse-Weil interval for curves of genus two is sufficiently large ($\Theta(p^{3/4})$ versus $\Theta(p^{1/2})$ for elliptic curves).

In practice no one uses the Adelman-Huang algorithm – it is of theoretical interest only.

[4] Using abelian varieties, which are generalisations of elliptic curves.

Rabin-Miller

The Rabin-Miller algorithm is fast: one trial takes time $\widetilde{O}(\log^2 n)$.

If we assume a certain Riemann Hypothesis (ERH), then $2\ln^2 n$ trials with $a = 2, 3, \ldots$ are sufficient to guarantee a correct result [Ankeny-Montgomery-Lenstra-Bach], so we have a deterministic polynomial-time algorithm. However, no one knows how to prove ERH.

Rabin-Miller is a Monte Carlo algorithm – there is a nonzero probability of error. In practice the probability of error is negligible if we take at last ten independent trials.

Rabin-Miller is feasible for numbers of 10^5 (maybe even 10^6) decimal digits.

Rabin-Miller produces a certificate of compositeness, but not a certificate of primality (compare ECPP).

Extension of Fermat to Polynomials

If n is prime and a is fixed, then

$$(x+a)^n = x^n + a \mod n \qquad (1)$$

holds, where each side of the equality is a polynomial in x. Formally, we are working in the ring $(\mathbf{Z}/n\mathbf{Z})[x]$ of polynomials whose coefficients are in the ring $\mathbf{Z}/n\mathbf{Z}$ of integers mod n..

Agrawal and Biswas [3] noticed that (1) is both necessary and sufficient for the primality of n.

In general, we can not compute $(x+a)^n \mod n$ in time polynomial in $\log n$.

Nevertheless Agrawal and Biswas [3] were able to give an RP compositeness test based on (1). The idea (similar to hashing) is to compute each side of (1) modulo some polynomial of low degree. The resulting test is slower than the Rabin-Miller test, so it did not attract much attention at the time.

The AKS Algorithm

In August 2002, Agrawal, Kayal and Saxena [4] announced a *deterministic* polynomial-time primality test based on (1). Thus,

<u>primality is in P</u>.

The idea is to compute $(x+a)^n \mod (x^r - 1, n)$ for $1 \le a \le s$ and sufficiently large r, s. The not-so-obvious fact is that it is sufficient to choose

$$r = O(\log n)^6, \quad s = O(\log n)^4 .$$

Thus, we can do everything in time $\widetilde{O}(\log n)^k$.

The precise value of the exponent k depends on details of the implementation. In the original AKS paper $k = 19$ (if classical algorithms are used for multiplication and division), or $k = 12$ if faster algorithms are used. k can be reduced further, as I shall mention later.

Example

Take $n = 1729 = 7 \times 13 \times 19$. This is a Carmichael number, so

$$a^n = a \mod n$$

for all integers a. However,

$$(x+1)^n \neq x^n + 1 \mod n \ .$$

In fact, working mod n (i.e. in $(\mathbf{Z}/n\mathbf{Z})[x]$),

$$(x+1)^n = x^{1729} + 247x^{1722} + \cdots + 247x^7 + 1 \ .$$

We can more easily verify that $(x+1)^n \neq x^n + 1 \mod n$ by working mod $(x^5 - 1)$ as well as mod n: we find that

$$(x+1)^n = 134x^4 + 1330x^3 + 532x^2 + 1330x + 134$$

in $(\mathbf{Z}/n\mathbf{Z})[x]/(x^5 - 1)$.

Here $x^5 - 1$ acts rather like a hash function – it lets us sum every fifth term in the binomial expansion of $(x+1)^n$, thus reducing $n+1$ terms to five. The computation involves polynomials of degree at most eight.

The Key Theorem

Theorem (AKS-Bernstein [8, Thm 1]; see also Morain [20, Thm 4.1])

Suppose that $n, r, s > 0$, where r is prime and q is the largest prime factor of $r - 1$. Suppose that

$$\binom{q+s-1}{s} > n^{2\lfloor\sqrt{r}\rfloor} \ ,$$

that n has no prime factor $\leq s$, and that $n^{(r-1)/q} \mod r \notin \{0, 1\}$. Finally, suppose that

$$(x-a)^n = x^n - a$$

in $Z/nZ[x]/(x^r - 1)$ for $1 \leq a \leq s$. Then n is a prime power.

Remarks

The time bound is $\widetilde{O}(rs \log^2 n)$.

The original AKS result [4] has $r = O(\log^6 n)$, $q \geq 4\sqrt{r} \log n$, $s \geq 2\sqrt{r} \log n$, giving time $\widetilde{O}(\log^{12} n)$.

Reducing the Exponent and Constant

A *Sophie-Germain prime* is a prime q such that $r = 2q + 1$ is also prime, e.g. $q = 11$, $r = 23$. It is conjectured that there are infinitely many Sophie-Germain primes, and that the number up to N is asymptotic to $2C_2 N / \log^2 N$, where $C_2 \approx 0.66016$.

If the conjecture about Sophie-Germain primes is true, then there is a suitable $r = O(\log^2 n)$, much better than the unconditional result $r = O(\log^6 n)$.

If $r = O(\log^2 n)$ then $s = O(\sqrt{r} \log n) = O(\log^2 n)$ and the time bound is

$$\widetilde{O}(rs \log^2 n) = \widetilde{O}(\log^6 n) \ .$$

Lenstra & Bernstein [8, Thm 2] reduced the implicit constant in the time bound by a factor of 60,000 over that in the original paper [4] (from 1024 to 0.01764).

Recent Improvements

Improvements by Bernstein, Berrizbeitia, Cheng, H. W. Lenstra, Mihailescu, Pomerance, Poonen, Vaaler, Voloch etc come out almost daily!

For example, Bernstein [9, §4] claims to have reduced the constant to 0.0005027 (a factor of 2.037×10^6 better than the original).

The best *deterministic* exponent is $6 + \varepsilon$. If we are willing to accept randomization, Bernstein claims an AKS-like algorithm that finds a certificate of primality in expected time $\widetilde{O}(\log^2 n)$. The certificate can be verified in (deterministic) time $\widetilde{O}(\log^4 n)$.

Comparing Bernstein's new result with ECPP, we see that finding the certificate is [asymptotically] faster than for ECPP, but verifying it is slower. (These are theoretical comparisons, ignoring the constant factors, which are important in practice.)

Experimental Results

The following table gives some times for a Magma implementation of the AKS algorithm (with Lenstra & Bernstein's improvements of August 2002) on a 300 Mhz Pentium II.

Times marked "(est)" are estimated from the time taken for one of the s iterations.

Times marked "(?)" are estimated by extrapolation, assuming the exponent $k = 6$.

n	r	s	time
$10^9 + 7$	43	315	3 sec
$10^{19} + 51$	67	5427	2500 sec
$10^{49} + 9$	491	28801	107 hours (est)
$10^{100} + 267$	3541	58820	3.3 years (est)
$2^{511} + 111$			44 years (?)
$2^{1023} + 1155$			2800 years (?)

Table 1: Times for the (slightly improved) AKS algorithm

Comparison with Other Algorithms

The following table gives times for Magma implementations[5] of the Rabin-Miller, ECPP and AKS algorithms on a 300 Mhz Ultrasparc 10. In all cases the number tested was $10^{100} + 267$.

Algorithm	trials	time
Rabin-Miller	1	0.01 sec
Rabin-Miller	10	0.10 sec
Rabin-Miller	100	1.00 sec
ECPP		6.68 sec
AKS		2.36 years (est)

Table 2: Times for various algorithms

[5] Our Magma programs are rather inefficient, but they are not biased towards any one algorithm. Thanks to Paul Zimmermann for his assistance with the Magma implementation of the AKS algorithm.

Bernstein's Implementation

Bernstein has proved $2^{1024} + 643$ prime by his improved AKS-like algorithm in 13 hours on an 800 Mhz PC. This is *much* faster than the original AKS algorithm.

Rabin-Miller takes less than one second, and ECPP takes about 50 seconds (Morain, on a 450 Mhz PC). Thus, ECPP is about 1660 times faster than Bernstein for numbers of this size (ignoring implementation differences).

In the race between ECPP and AKS-like algorithms, ECPP is well ahead, but the AKS-like algorithms are catching up fast, so it is hard to predict the eventual winner.

If the exponents for Bernstein's algorithm and ECPP are 4 and 5 respectively, then by extrapolation we expect Berstein's to be faster for numbers of 500,000 decimal digits or more. However, a number this large would take about 10^{10} years to prove prime by either algorithm!

Reliability of the Result

ECPP gives a certificate of primality, and the certificate can be checked quickly. It *should* be checked, to guard against hardware and/or programming errors. It is conjectured that the expected time to find a certificate is $\widetilde{O}(\log^5 n)$. The time to check a certificate is $\widetilde{O}(\log^3 n)$.

Rabin-Miller does not give a certificate of primality. It takes time $\widetilde{O}(\log^2 n)$ per trial. T trials give probability of error less than 4^{-T} if n is composite; there is no error if n is prime.

Theoretically, AKS is error-free. However, in a long computation there is a significant probability of a hardware error. Such an error would in most cases make a prime n "appear" composite; a composite n would usually still "appear" to be composite.

To be safe, one should make an independent check of the certificate (if available), or use at least two different algorithms.

The Ultimate Primality Test ?

Conjecture (Bhattacharjee and Pandey [11])

If r is an odd prime that does not divide $n(n^2 - 1)$, and

$$(x - 1)^n = x^n - 1$$

in $(\mathbf{Z}/n\mathbf{Z})[x]/(x^r - 1)$, then n is prime.

Remarks

1. We can find an odd prime $r = O(\log n)$ that does not divide $n^2 - 1$ simply by checking $r = 3, 5, 7, 11, \ldots$. (If $r|n$ then we are finished.)
2. The time for the test is $\widetilde{O}(r \log^2 n) = \widetilde{O}(\log^3 n)$.
3. The conjecture has been verified for $r < 100$, $n < 10^{10}$, and also for Carmichael numbers up to 10^{16} (checking the smallest applicable r). For partial results and connections with other conjectures, see Kayal and Saxena [14].

Generating Primes for Cryptography

In cryptographic applications, we may want to generate large, "random" primes. In practice Rabin-Miller is quite adequate for this. However, it is possible to generate a random prime at the same time as constructing a proof of its primality. This is analogous to constructing a correctness proof at the same time as writing a program.

The idea is to (recursively) generate provable odd primes $q_1, \ldots, q_\nu$ say, such that

$$p = 2q_1 \cdots q_\nu + 1$$

is prime. Since we know the factorisation of $p - 1$, we can easily prove that p is prime.

This procedure does not generate the primes in a given interval with equal probability, but all we need is to generate primes uniformly from a sufficiently large set of possibilities.

Conclusions

- The AKS algorithm is theoretically significant, since it shows without doubt that primality is in P.
- The AKS algorithm is quite "elementary". There could be other elementary algorithms for interesting problems (e.g. integer factorisation) waiting to be discovered by clever undergraduates.
- The AKS algorithm is too slow in practice. ECPP is *much* faster. Rabin-Miller is even faster, at the price of a *minute* probability of error. However, faster AKS-like algorithms keep appearing; it's not yet clear if they will be able to compete with ECPP.

References

[1] L. M. Adleman and M. A. Huang, Primality testing and abelian varieties over finite fields, in *Lecture Notes in Mathematics*, vol. 1512, Springer-Verlag, 1992.

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

[3] M. Agrawal and S. Biswas, Primality and identity testing via chinese remaindering, *Proc. IEEE Symposium on Foundations of Computer Science*, 1999, 202–209.

[4] M. Agrawal, N. Kayal and N. Saxena, PRIMES is in P, preprint, August 6, 2002. Available from `http://www.cse.iitk.ac.in/primality.pdf`.

[5] W. Alford, A. Granville and C. Pomerance, There are infinitely many Carmichael numbers, *Ann. of Math.* 139 (1994), 703–722.

[6] M. Artjuhov, Certain criteria for the primality of numbers connected with the little Fermat theorem (in Russian), *Acta Arith.* 12 (1966/67), 355–364.

[7] A. Atkin and F. Morain, Elliptic curves and primality proving, *Math. Comp.* 61 (1993), 29–68.

[8] D. Bernstein, An exposition of the Agrawal-Kayal-Saxena primality-proving theorem, version of 20 Aug 2002. Superseded by [9].

[9] D. Bernstein, Proving primality after Agrawal-Kayal-Saxena, version of 25 Jan 2003, available from `http://cr.yp.to/papers.html#aks` .

[10] D. Bernstein, Proving primality in essentially quartic expected time, 6 March 2003, available from `http://cr.yp.to/papers.html#quartic` .

[11] R. Bhattacharjee and P. Pandey, *Primality Testing*, Tech. Report, IIT Kanpur, 2001. Available from `http://www.cse.iitk.ac.in/research/btp2001/primality.html` .

[12] R. Crandall and C. Pomerance, *Prime Numbers: A Computational Perspective*, Springer-Verlag, New York, 2001.

[13] S. Goldwasser and J. Kilian, Almost all primes can be quickly certified, *Proc. 18th Annual ACM Symposium on Theory of Computing*, 1986, 316–329. Newer version in *J. ACM* 46 (1999), 450–472.

[14] N. Kayal and N. Saxena, *Towards a Deterministic Polynomial-time Primality Test*, Tech. Report, IIT Kanpur, 2002. Available from `http://www.cse.iitk.ac.in/research/btp2001/primality.html` .

[15] D. E. Knuth, *The Art of Computer Programming*, Vol. 2, 3rd edition, Addison-Wesley, Menlo Park, 1997, §4.5.4.

[16] J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, Elsevier, Amsterdam, 1990.

[17] A. K. Lenstra and H. W. Lenstra, Jr., Algorithms in number theory, in [16], 675–715.

[18] G. L. Miller, Riemann's hypothesis and tests for primality, *J. Comp. System Sci.* 13 (1976), 300–317.

[19] L. Monier, Evaluation and comparison of two efficient probabilistic primality testing algorithms, *Theoret. Comput. Sci.* 12 (1980), 97–108.

[20] F. Morain, Primalité théorique et primalité pratique ou AKS vs. ECPP, preprint, 4 October 2002. Also paper and transparencies for séminaire Bourbaki. Available from `http://www.lix.polytechnique.fr/Labo/Francois.Morain/`

[21] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.

[22] R. G. E. Pinch, Some primality testing algorithms, *Notices of the AMS* 40 (1993), 1203–1210. `http://www.chalcedon.demon.co.uk/publish.html`

[23] R. G. E. Pinch, The Carmichael numbers up to 10^{15}, *Math. Comp.* 61 (1993) 381–391. Extension to 10^{16} at `http://www.chalcedon.demon.co.uk/publish.html`

[24] C. Pomerance, *Primality testing: variations on a theme of Lucas*, to appear. `http://cm.bell-labs.com/cm/ms/who/carlp/pub.html`

[25] V. Pratt, Every prime has a succinct certificate, *SIAM J. Computing* 4 (1975), 214–220.

[26] M. O. Rabin, "Probabilistic algorithms", in *Algorithms and Complexity* (J. F. Traub, editor), Academic Press, New York, 1976, 21–39.

[27] M. O. Rabin, Probabilistic algorithms for testing primality, *J. Number Theory* 12 (1980), 128–138.

[28] H. Riesel, *Prime numbers and computer methods for factorization*, second edition, Birkhäuser, Boston, 1994, Chapter 4.

[29] R. L. Rivest, A. Shamir and L. Adleman, A method for obtaining digital dignatures and public-key cryptosystems, *Comm. ACM* 21 (1978), 120–126.

[30] R. Schoof, Elliptic curves over finite fields and the computation of square roots mod p, *Math. Comp.* 44 (1985), 489–494.

[31] R. Solovay and V. Strassen, Fast Monte Carlo test for primality, *SIAM J. on Computing* 6 (1977), 84–85; *erratum* 7 (1978), 118.

[32] H. C. Williams, *Édouard Lucas and Primality Testing*, Canadian Math. Soc. Monograph, Vol. 22, John Wiley & Sons, New York, 1998.