From *SIAM News*, Volume 35, Number 7, September 2002

Researchers Devise Fast Deterministic Algorithm for Primality Testing

By Sara Robinson

One Sunday last month, Manindra Agrawal, a professor of computer science at the Indian Institute of Technology in Kanpur, quietly sent a postscript file to a dozen or so experts in theoretical computer science and computational number theory. He was relieved that three years of hard work—two off and on by himself and one more with his PhD students Neeraj Kayal and Nitin Saxena—had panned out.

Hendrik Lenstra was working in his office at the University of California at Berkeley when he saw the e-mail message from India, with subject header "Primes is in P." The text of the message said merely: "Subject header says it all."

As a number theorist with an interest in computational issues, Lenstra had thought about this problem off and on himself. Just about everyone in the field had. But since he didn't know the authors of the paper, he was initially skeptical, expecting to see gibberish from an amateur. "I get a lot of junk," Lenstra explains.

Once he opened the paper, however, it was immediately clear that this paper was "serious." Still doubtful enough not to read it immediately, he printed it out and took it home with him. That night, over dinner with visiting number theorists Peter Stevenhagen and Manjul Bhargava, he took a closer look. The proof was clever and elegant and, apart from some small, easily fixable glitches, seemed utterly correct. Except for one fairly difficult result from analytic number theory to prove the run-time, Lenstra says, the techniques, a combination of analytic and algebraic ideas, were "perfectly elementary." Stevenhagen departed after dinner, but Lenstra and Bhargava sat in a Berkeley café for hours, talking through the proof and some possible improvements until they were finally thrown out at closing time.

Across the country at Bell Labs, number theorist and cryptographer Carl Pomerance had also been on the mailing list. The IIT Kanpur paper, however, had run afoul of his junk mail filter, so he was initially baffled on Monday morning by a string of messages, from Lenstra and others, referring to a deterministic primality test. But he quickly obtained a copy of the paper and agreed that the result was correct. By lunchtime on Tuesday, he was explaining the proof to his colleagues; he even gave an impromptu seminar on the result late that afternoon. It was that simple to digest.

By Monday afternoon, news of the algorithm had spread through mathematics and computer science departments across the country. By Thursday, an article on the result appeared in *The New York Times*, and in the *Times* of India and other major newspapers. By Friday, there was an Associated Press story.

It's rare that a new theorem generates this kind of attention, particularly if there are no practical implications. Agrawal was astonished at all the hoopla. "I knew that this result would be very well received in the theoretical computer science community and amongst some mathematicians," he says. "However, I never imagined the kind of excitement it has created amongst mathematicians! They seem to find more in it than I can see."

A Brief History

The trio had solved a longstanding open problem: devising an algorithm, running in time polynomial in the length (in binary) of the input, for determining whether a given number is prime. Fast algorithms for primality testing have been of widespread interest to computer scientists since the early 1970s because of the important role they play in many modern cryptosystems.

Keys for the RSA algorithm, for instance, are numbers that are a product of two large primes, generated with a primality test algorithm. The security of the algorithm rests on the fact that while multiplying two large numbers is easy, factoring a number into its prime components is hard. Part of the magic of RSA is that determining whether a number is prime or composite can be done quickly, but, in general, no one knows a quick way to find a composite number's prime factors.

The fast primality tests used in practice are "randomized" algorithms, which make use of random number generators. These algorithms have a high probability of giving a correct answer, and that probability can be made higher still by running the algorithm multiple times. Even so, a tiny probability of error always remains.

In the context of complexity theory, where problems are categorized by their computational hardness, primality testing was not known to be in P, the class of problems solvable by polynomial-time algorithms. To be sure, the existence of fast randomized algorithms suggested to many that a fast deterministic algorithm must exist, but no one had yet been able to find one.

The basic problem of recognizing primes dates back thousands of years, at least to the sieve of Eratosthenes, and probably earlier. But the modern problem of finding a computationally fast method seems to date to a 1956 letter from Gödel to von Neumann. Writing in the letter about the issue of the growth rate of a problem, Gödel referred specifically to primality testing: "It would be interesting to know, for example, what the situation is in the case of determining whether a number is a prime number, and in the case of finite combinatorial problems, how strongly in general the number of steps of the exhaustive search can be reduced" (translation from the original German).

It was not until 20 years later, however, that Gary Miller came up with the first step toward a polynomial-time primality test. Miller's algorithm was deterministic and ran in time bounded by a polynomial of degree 4, but he needed to assume the Extended Riemann Hypothesis to prove his algorithm correct. In 1980, Michael Rabin modified Miller's test to obtain an unconditional but randomized polynomial-time algo-

rithm for primality testing. Based on Fermat's Little Theorem, that algorithm correctly identified composite numbers but had a small probability of error in identifying primes.

Meanwhile, in 1977, Robert Solovay and Volker Strassen used quadratic residues to obtain another randomized polynomial-time algorithm that could be derandomized, assuming the Extended Riemann Hypothesis. A slew of similar algorithms followed.

In 1983, Leonard Adleman, Pomerance, and Robert Rumely came up with a deterministic algorithm that runs in something very close to polynomial time. Their algorithm was then simplified by Henri Cohen and Lenstra and implemented by Cohen and Arjen Lenstra. Three years later, Shafi Goldwasser and Joe Kilian (and, separately, A.O.L. Atkin and François Morain) came up with a randomized algorithm of a different type. Based on elliptic curves, the algorithm always gives correct answers, but the run-time is a random variable that has a polynomial expected value only on almost all inputs (all under a widely believed conjecture). Two years after that, Adleman and Huang modified the Goldwasser-Kilian approach to obtain an algorithm that gives a guaranteed answer and runs in "expected polynomial time" on all inputs.

For all practical purposes, primality testing was at that point a solved problem. Indeed, the method of choice in implementations of RSA is one of the earliest algorithms, Miller–Rabin, which happens to be fast and easy to program. Ivan Damgard, Peter Landrock, and Pomerance showed that the worse-case error estimates of this method hold for only a few numbers that are unlikely to be chosen. In practice, then, the probability of error is not a problem. From a complexity theoretic point of view, however, the results were not satisfactory. Researchers widely believed primality to be in P, but there was no proof.

The AKS Algorithm

The Agrawal–Kayal–Saxena algorithm is based on an identity that Agrawal and his colleague Somenath Biswas used in 1999 to create a randomized polynomial-time primality test: If a is coprime to p, then p is prime if and only if the polynomial $(x - a)^p$ is congruent to the polynomial $(x^p - a) \pmod{p}$.

In general, evaluating the left-hand side of this congruence takes exponential time, but Agrawal, Kayal, and Saxena were able to use a clever trick to simplify the congruence to one that loses some information but can be computed in polynomial time. From there, they use other tricks to show how to extract a definitive answer.

The proven exponent for the running time is about 12, but heuristically, the real exponent is shown to be about half that. Hendrik Lenstra has already improved the researchers' run-time analysis to a polynomial of degree 8.

The most striking thing about the AKS algorithm, according to both Pomerance and Lenstra, is that its methods are relatively elementary. This stands in sharp contrast to some of the earlier results, which used long, hard proofs and a lot of sophisticated machinery. The run-time proof for the AKS algorithm, except for its appeal to a single hard theorem of analytic number theory, requires only simple algebra.

That so elegant a proof eluded researchers for so many decades suggests to Pomerance that other problems believed to be hard, such as factoring, might also fall under an appropriate attack. "When there are surprises in the field it must make cryptographers a bit uneasy," Pomerance says. Factoring is widely believed to be harder than primality testing; still, the only proof of its hardness is that no one has yet found a method.

Agrawal hopes that some of the techniques of the new algorithm might give a new perspective on factoring. But first, he plans to look at a more immediate application of his methods: other number theoretic problems for which there are randomized algorithms but no deterministic ones. Having successfully converted the Agrawal–Biswas probabilistic algorithm into a deterministic algorithm, Agrawal and his students are hopeful that a similar approach might succeed in derandomizing other number theoretic algorithms. "That investigation is next on our agenda," Agrawal says.

Sara Robinson is a freelance writer based in Berkeley, California.