

Elliptic Curve Cryptography— Good Enough for Government Work

By Barry A. Cipra

“Standards are the products of logic, of analysis and painstaking study; they are evolved on the basis of a problem well stated. In the final analysis, however, a standard is established by experimentation.”

—Le Corbusier, *The Modulor*

Elliptic curves, the key to unlocking the cryptic comment known as Fermat’s Last Theorem at the end of the second millennium, could be the key to *locking* cryptic comments well into the third millennium. The speakers at a mini-tutorial on elliptic curve cryptography at the SIAM 50th Anniversary Meeting in Philadelphia showed how.

Neal Koblitz of the University of Washington, Darrel Hankerson of Auburn University, and Alfred Menezes of the University of Waterloo explained how the theory of elliptic curves gives cryptographers a slew of potent tools for creating “unbreakable” codes that are fast and easy to implement—key concerns in a fast-paced, insecure world. One technique, the Elliptic Curve Digital Signature Algorithm (ECDSA), has already received the seal of approval of the U.S. government. ECDSA is one of three methods for authenticating messages certified so far by the National Institute of Standards and Technology.

The Mod Squad

“Unbreakable” number-theoretic cryptosystems have been around since the 1970s. The most famous is the RSA code, named for its creators, Ronald Rivest, Adi Shamir, and Leonard Adleman. In brief, RSA codes a message M by computing $C = M^e \bmod N$, where $N = pq$ is the product of two large primes and e is a more or less random large number, restricted mainly by the requirement that it not have any divisors in common with $p - 1$ or $q - 1$. The decoding computation requires a number d that satisfies $ed = 1 \bmod (p - 1)(q - 1)$, which is easily found by the euclidean algorithm, provided you know the numbers p and q . Elementary number theory, namely Euler’s generalization of Fermat’s Little Theorem, guarantees that $M = C^d \bmod N$. If M is less than N , the computation returns it intact.

The unbreakability of RSA is an article of faith. It relies on the apparent—and still unproved—difficulty of factoring numbers composed of large primes. “Large” is a relative term, however. In 1977, when the RSA team introduced their code, they posed a challenge that amounted to factoring a 129-digit number, estimating that it would take roughly twenty thousand years to break the code. They were off by three orders of magnitude; the code was broken in 1994.

RSA is still considered secure, as long as it uses numbers well beyond the reach of anticipated attacks. The current reigning champion among factoring algorithms, an approach known as the Number Field Sieve, has a heuristic run time that grows slightly more rapidly than $(\exp(\log N)^{1/3})$. The exponent $1/3$ qualifies the algorithm as “subexponential,” but that’s still a far cry from polynomial. The Number Field Sieve has successfully factored numbers with up to about 150 digits. Moore’s law will undoubtedly push things onward and upward, but hardware advances alone will only make it easier for RSA to outpace potential attackers by using larger and larger primes—unless, of course, the hardware advance is a working quantum computer.

But RSA is just one number-theoretic arrow in the cryptographic quiver. Elliptic curves portend a whole Agincourt’s worth of codes.

Cubism

Roughly speaking, an elliptic curve is the solution set of a cubic equation in two variables. For number-theoretic purposes, the equation can be brought into the form $y^2 = x^3 + Ax + B$, with integer coefficients A and B , although other forms are often used as well. The solution set is relative to some field of definition, such as the field of complex numbers. Of greatest interest are the *rational* solutions of such equations. But much of the theory—and essentially all the cryptographic applications—lie in the solutions mod p (or, more generally, in solutions over finite fields).

One of the nice features of elliptic curves mod p is that the size of the solution set is never too far from p . The exact theorem, proved by Helmut Hasse in 1933, is $|N_p - p| \leq 2p^{1/2}$, where N_p is the number of solutions mod p . This ensures that for large p , there are lots of solutions over the finite field F_p .

But the nicest feature of elliptic curves, over *any* field, is that the solution set, with an extra “point at infinity” tacked on, forms a group, with a group law given by an explicit pair of rational functions. The group turns out to be abelian (hence the additive notation), and the point at infinity, usually denoted O , is its “zero” element.

In many cases, the order of the group (over F_p) is itself a large prime, q , or a small multiple of such a prime. When that happens, the curve is ripe for use in cryptography. In particular, it can be used to attach a digital signature to a message.

Sez Who?

For a digital signature, the main idea is no longer to disguise what a message says, but rather to prove that it originates with a particular sender. RSA, for example, has a simple way of doing this. If the “owner” of a code (N, e) wants to prove that she’s the

sender of a message M , she can use her private “decoding” exponent d to compute $C' = M^d \bmod N$, and then send both M and C' . The receiver can then persuade himself that the message truly originated with the owner of d by computing $C'^e \bmod N$, and checking that it's the same as M . (Caveat: If someone tries “signing” a Yes/No message by sending the pair (0,0) or (1,1), don't believe her!) Indeed, a fancy version of this, known as rDSA, was approved by NIST along with the elliptic curve approach.

The very first NIST-approved digital signature algorithm, DSA, is based on what's called the discrete logarithm problem. To give an unrealistically simple example: What value of x solves $2^x = 10 \bmod 101$?—i.e., in the finite field F_{101} , what is the “log base 2” of 10? More generally, if g is an element of (large) order q in the multiplicative group of a finite field F_p , the problem is to compute x , knowing only g and $g^x \bmod p$ (and, of course, p and q). Elementary number theory, namely Fermat's Little Theorem, says that q divides $p - 1$; DSA requires that q be a 160-bit prime and p a prime with between 512 and 1024 bits.

In terms of computational difficulty, the discrete logarithm problem seems to be on a par with factoring. The basic idea of DSA is for the “signer” of message M —that is, the possessor of the value x behind the publicly known $g^x \bmod p$ —to append a pair of numbers r and s obtained by secretly picking another number k between 1 and q , computing $r = (g^k \bmod p) \bmod q$ (i.e., computing $g^k \bmod p$, and then taking the remainder of that number mod q) and $s = k^{-1}(\text{SHA}(M) + xr) \bmod q$, where k^{-1} is the multiplicative inverse of $k \bmod q$ and SHA is the Secure Hash Algorithm. Another NIST standard, SHA reduces a character string of any length to a 160-bit string of gibberish. (The official acronym is SHA-1, but that looks funny in formulas.)

The receiver of (M,r,s) from “person” g^x computes $u = s^{-1} \text{SHA}(M) \bmod q$ and $v = s^{-1} r \bmod q$, and then checks that $((g^u)(g^{xv}) \bmod p) \bmod q$ equals r . If it doesn't, then, by elementary number theory, something definitely went wrong. If it does, then, according to NIST, you can safely assume that message M came from the presumably unique individual who knows the discrete logarithm of g^x .

The Elliptic Curve Digital Signature Algorithm works in much the same way. It requires an elliptic curve E over a finite field F_p with a point P of (large prime) order q , all of which is public. Each user is identified by a scalar multiple, xP , where x is a (secret) number between 1 and q . To sign a message, the possessor of x again secretly picks a number k between 1 and q and appends a pair of numbers r and s . The number s is exactly as it is for DSA: $k^{-1}(\text{SHA}(M) + xr) \bmod q$. But this time r is obtained by computing the point kP on the elliptic curve E over F_p : If $kP = (k_1, k_2)$ is the result of that computation, then $r = k_1 \bmod q$.

Verifying an elliptic curve signature is also similar to the verification process for DSA: The receiver computes u and v as before, but then computes the point $uP + v(xP)$ on the elliptic curve E over F_p . If the x -coordinate of this point is not equal to $r \pmod q$, the signature is rejected.

Pros and Cons

One advantage of elliptic curve cryptography is that, unlike factoring and the discrete logarithm problem, there is no known algorithm for computing x from xP that runs in less than exponential time. This makes it possible to use relatively small values for p and q . One of the NIST-approved curves, for example, uses 192-bit (68-digit) numbers.

There are also lots of elliptic curves to choose from. But just as factoring is often easier than one would expect from the worst-case run times of the algorithms, some elliptic curves are vulnerable to attack. If, for example, $q = p$ (that is, if the curve has exactly $p - 1$ solutions mod p , along with the “point at infinity”), researchers have shown that E_p can be explicitly equated with the additive group F_p , at which point the extremely efficient euclidean algorithm can be used to recover x . Similarly, if q divides $p^m - 1$ for some exponent m , the group E_p can be identified with the multiplicative group of the finite field F_{p^m} and the task of finding x reverts to the original discrete logarithm problem. If m is large (and Fermat's Little Theorem guarantees one such value, namely $m = q - 1$), that's OK, but if it's small, there could be trouble, security-wise.

The elliptic curves that carry NIST's seal of approval avoid these pitfalls. There are 15 of them, falling into three classes. The first five, labeled P-192, -224, -256, -384, and -521, are curves of the form $y^2 = x^3 - 3x + b$ over a finite field F_p (the labels refer to the number of bits in the prime p). For these curves, the coefficients b were chosen in a pseudo-random fashion.

The other two classes are taken over binary fields F_{2^m} with $m = 163, 233, 283, 409, \text{ and } 571$. The curves are labeled B- m and K- m (“B” for “binary” and “K” for “Koblitz”). In each case, the equations have the form $y^2 + xy = x^3 + ax + b$. For the B class, the coefficient $a = 1$ and the b 's are generated pseudo-randomly. For the K class, $b = 1$ and a is either 0 or 1. The Koblitz curves, also called “anomalous binary curves,” or ABC curves (and early on dubbed “magic curves” by enthusiasts at the National Security Agency), were introduced by Koblitz in 1991. Their special form offers advantages in terms of efficient implementation of the code—that is, doing the computations required by cryptographic algorithms.

The computationally expensive steps in ECDSA are the scalar multiplications kP , uP , and $v(xP)$. (The computation of xP is done only once, by the owner of x .) These computations aren't hard in the P-versus-NP sense. They're just tedious and time-consuming. But the special algebraic nature of elliptic curves presents some cost-cutting opportunities. Scalar multiplication, which a naïve approach does by repeated doubling and straightforward addition (for example, $7P = 4P + 2P + P$, where $2P = P + P$ and $4P = 2P + 2P$), is, from the perspective of abstract algebra, a group endomorphism—that is, a mapping of the group E_p into itself. As such, it can sometimes be re-expressed in terms of other endomorphisms that, while theoretically fancier, are easier to compute, much as an appropriate change of basis can turn an ugly matrix multiplication problem into a computational breeze. The Koblitz

For a digital signature, the main idea is no longer to disguise what a message says, but rather to prove that it originates with a particular sender.

curves, for example, take advantage of arithmetic in the integers with the square root of -7 adjoined.

Security Concerns

How secure is ECDSA? Like all cryptosystems based on the unproved assumptions of complexity theory, that question will have no definitive answer unless some breakthrough shows the answer to be “No.” In 1998, Joseph Silverman, an elliptic curve expert at Brown University, sent shock waves through the elliptic curve crypto community when he proposed a new method for attacking the xP problem. Silverman’s “xedni” algorithm (“xedni” is “index” spelled backward) uses sophisticated techniques in algebraic geometry to pry loose the x . Fortunately (for cryptographers), the xedni algorithm turns out to require exponential time. The analytic demonstration of that is also rooted in heavy-duty algebraic geometry, including such things as the Taniyama–Shimura conjecture, which was the key to the proof of Fermat’s Last Theorem.

Even if complexity theorists prove the assumptions that cryptographers rely on, the security of specific curves will always be suspect and, of course, relative to the algorithms and computing resources of the day. The endomorphisms that streamline implementations of elliptic curve computations, for example, also facilitate attacks. But the methods that speed things up for factorization and the discrete logarithm seem to lag when it comes to elliptic curves.

For anyone who can round up some serious spare cycle time, Certicom, an Ontario-based security company specializing in elliptic curve cryptography, has posed a set of elliptic curve challenges, the largest of which involves a 239-bit (71-digit) prime. The largest solved so far involves a Koblitz curve over a field of size 2^{109} . The computation, led by Robert Harley and colleagues at the French Institut National de Recherche en Informatique et en Automatique (INRIA), took four months and some 9500 computers. The remaining challenges, along with white papers and an elliptic curve tutorial, are available online at http://www.certicom.com/resources/ecc_chall/challenge.html.

Certicom shelled out US\$10,000 for the K-109 computation, or just over a dollar per participating processor. That would barely pay the electric bill. But with billions of dollars soon to be zooming around the Internet signed by cubic equations, you can count on computational number theorists to continue refining their methods.

Even if complexity theorists prove the assumptions that cryptographers rely on, the security of specific curves will always be suspect and, of course, relative to the algorithms and computing resources of the day. . . . But the methods that speed things up for factorization and the discrete logarithm seem to lag when it comes to elliptic curves.

Barry A. Cipra is a mathematician and writer based in Northfield, Minnesota.