MATH 587/CSCE 557 - SUMMARY OF CLASS, 4/5/07

Class on March 29 was review for the midterm, and class on April 3 was the midterm.

Today I recapped all the ingredients going into successfully using RSA (probabilistic primality tests, Euclid's algorithm, quick exponentiation mod N, ...). I mentioned how this week's homework illustrates some possible pitfalls in implementing RSA. Then I discussed attacks based on factoring N.

The first clever method, that led to the factoring of RSA-129, was the quadratic sieve method. The idea is to fix some bound K. Then for various x just a little bigger than $\sqrt{N}$, look at the factorization of $x^2 - N$. A successful example is one where all the primes dividing $x^2 - N$ are less than K. The search for successful examples can be farmed out to many individuals, using idle cycles on multiple machines (whence the term factoring by email).

Successful examples are sent to a central location. Suppose that there are m primes less than K. Once a little more than m successful examples have been found, the person at the central location can try to find a product of these examples which is a perfect square. This uses linear algebra mod 2 to find the nullspace of a huge matrix.

Suppose $(x_1^2 - N)(x_2^2 - N)...(x_r^2 - N)$ is a perfect square y^2. Then $(x_1 x_2 ... x_r)^2 \pmod N = y^2 \pmod N$, so that $(x_1 x_2 ... x_r)^2 - y^2 = (x_1 x_2 ... x_r - y)(x_1 x_2 ... x_r + y)$ is divisible by N. If $N = pq$, then we are lucky if p and q don't both divide the same factor (this happens about 50% of the time). In that case $\gcd(x_1 ... x_r - y, N)$ will give you a proper factor of N. If you are unlucky, then simply use a different product of successful examples, until you get lucky.

I mentioned how these attacks on RSA are leading to the need to use ever longer keys (2048-bit often recommended). For constrained environments (e.g. handheld PDA's), this won't work because of space and power requirements. An alternative is Diffie-Hellman key exchange.

The idea is the following. Pick a large prime p and an integer $g > 1$. These can be public. Alice and Bob pick secret integers a, b respectively. Alice sends $g^a \pmod p$ to Bob and Bob sends $g^b \pmod p$ to Alice. Each can compute $g^{ab} \pmod p$ and this shared secret can then function as a key (as for a one-time pad). An eavesdropper knows $g^a \pmod p$ and $g^b \pmod p$ but then it is hard to compute $g^{ab} \pmod p$.

Developments such as index calculus and the number field sieve mean that it is not as hard as previously thought. Companies are turning to elliptic curve cryptosystems (ECC). For this pick an elliptic curve $y^2 = f(x)$, where f is a cubic, pick a large prime p, and pick a solution Q to the elliptic curve equation $\pmod p$. There is a natural way to add points on an elliptic curve. Alice sends aQ to Bob and Bob sends bQ to Alice and they use shared secret abQ as a key.