Fall 2006
Chris Christensen
MAT/CSC 483

Primality testing

To construct an RSA key, we need two large primes. It would not be a good idea to take primes from a list; it's likely that people might attack our key might try checking to determine whether primes that are on "common lists" are factors of *n*. So we might just, for example, take a large power of 2 and add 1 and see if that number is prime. If not, add 2 and see whether that number is prime. Etc. But how can we determine whether a positive integer is prime or not?

It is rather remarkable that there are tests for primality that do not require factoring. Of course, this is good because factoring is hard.

The basic idea for these tests goes back to Fermat's little theorem. If *n* is prime, then $a^{n-1} = 1 \bmod n$ for all positive integers *a* such that $1 \le a \le n-1$.

So, if we are given a positive integer *n* and we can find an *a* such that $1 \le a \le n-1$ and $a^{n-1} \ne 1 \bmod n$, then *n* is NOT prime.

Unfortunately, if $a^{n-1} = 1 \bmod n$, we cannot conclude that *n* is prime. In particular, there are infinitely many Carmichael numbers (1910) – positive composite integers *n* so that $a^{n-1} = 1 \bmod n$ for all *a* such that $1 \le a \le n-1$. So, there are composite numbers that "pass" this test for all bases *a*.

Probably the most commonly used primality test is the Miller-Rabin test (1976). Like the test we described using Fermat's little theorem, failing this test yields a definite result – the number is composite – while passing the test does not yield a definite result – we can only say that the number is "probably prime." But, we will have a better idea about what "probably " means.

Here's the test. Let *n* be an odd prime and write $n-1 = 2^k m$ where *m* is odd. Choose an integer *a* so that 1 < *a* < *n* - 1. Calculate a^m, a^{2m}, $a^{2^2 m}$, …, $a^{2^k m} = a^{n-1}$ mod n (notice that each term is the square of the preceding term). *n* passes the test if the first occurrence of 1 in this sequence is the first term or if the first occurrence of 1 is preceded by -1. Every odd prime passes this test, but composites can also pass this test. We know however that for a given base *a*, the probability

P(composite passing the test) < 1/4

If an integer *n* fails this test (for some base *a*), *n* is definitely composite. If an integer *n* passes this test (for some base *a*), these is evidence that *n* is prime – the probability that *n* is composite would be < ¼. If *n* passed this test for two different bases *a*, the probability would be < $(1/4)^2$ that *n* is composite. If *n* passed this test for three different bases *a*, the probability would be < $(1/4)^3$ that *n* is composite. … . If *n* passed this test for one hundred different bases *a*, the probability would be < $(1/4)^{100}$ that *n* is composite. If *n* passed this test for 10 bases, the probability that *n* was composite would be less than 0.000000953674; certainly that would mean that *n* is "probably prime." Such tests for primality are probabilistic.

The test described above is the test used by *Mathematica* in the command **PrimeQ**[integer**]**.

Two examples:

Let $n = 8929$. $n-1 = 8929-1 = 2^5 \times 279$. Take $a = 2$.

PowerMod[2, 279, 8929**]**

3355

PowerMod[%, 2, 8929**]**

5485

PowerMod[%, 2, 8929**]**

3424

PowerMod[%, 2, 8929]

8928

PowerMod[%, 2, 8929**]**

1

8929 passes for the base $a = 2$. (Notice that 8928 is -1 mod 8929.) Therefore, 8929 is probably prime with a probability > $1-(1/4)^1$.

Let's try another base. Say, $a = 325$.

PowerMod[325, 279, 8929**]**

308

PowerMod[%, 2, 8929**]**

5574

PowerMod[%, 2, 8929**]**

5485

PowerMod[%, 2, 8929**]**

3424

PowerMod[%, 2, 8929**]**

8928

PowerMod[%, 2, 8929**]**

1

8929 also passes for the base a = 3; 8928 = -1 mod 8929. Therefore, 8929 is probably prime with a probability > $1-(1/4)^2$. Etc.

Ok, let's settle it.

PrimeQ[8929**]**

True

Now try n = 14875873. (Any guesses to primeness?) $n-1=2^5\times 464871$.

PowerMod[2, 464871, 14875873**]**

10350676

PowerMod[%, 2, 14875873**]**

10034786

PowerMod[%, 2, 14875873**]**

3603084

PowerMod[%, 2, 14875873**]**

10192210

PowerMod[%, 2, 14875873]

7853992

PowerMod[%, 2, 14875873**]**

2801884

A good reference for factoring algorithms and primality testing is *Prime Numbers: A Computational Perspective* by Crandall and Pomerance.

In 2002 a polynomial time algorithm was developed that definitely determines whether an integer is prime or not. It has not yet been implemented. See the link for *Primes is in P*.