Queen Mary
University of London

MAS 335 **Cryptography**

Notes 8: Public-key cryptography **Spring 2008**

Primes and factorisation

Primality testing is easy, but factorisation is hard. This assertion is the basis of the security of the RSA cipher. In this section, we consider it further.

Primality testing

The basic algorithm for primality testing is trial division. In a very crude form it asserts that, if $n > 1$ and n is not divisible by any integer smaller than $\sqrt{n}$, then n is prime.

The first thing to say about this algorithm is that, with minor modification, it leads to a factorisation of n into primes. If n is not prime, then the first divisor we find will be a prime p, and we continue dividing by p while this is possible to establish the exact power of p. The quotient is divisible only by primes greater than p, so we can continue the trial divisions from the point where we left off.

The second thing is that this simple algorithm does not run in polynomial time: the input is the string of digits of n, and the number of trial divisions is about $\sqrt{n}$, roughly $2^{k/2}$ if n has k digits to the base 2.

It is a bit surprising at first that primality testing can be easier than factorisation. This holds because there are algorithms which decide whether or not a number is prime without actually finding a factor if it is composite! Two examples of such theorems are:

Theorem 22 *Little Fermat Theorem: If n is prime then $x^n \equiv x \pmod{n}$ for any integer x.*

Wilson's Theorem: n is prime if and only if $(n-1)! \equiv -1 \pmod{n}$.

We have seen the proof of the little Fermat theorem. Here is Wilson's Theorem.

Suppose that p is prime. We know that every number x in the set $\{1,\dots,p-1\}$ has an inverse y mod p (so that $xy \equiv 1 \pmod{p}$). The only numbers which are equal to their inverses are 1 and $p-1$: for if x is equal to its inverse, then $x^2 \equiv 1 \pmod{p}$, so that p divides $x^2-1=(x-1)(x+1)$, and p must divide one of the factors. The other $p-3$ numbers in the range can be paired with their inverses, so that the product of each pair is congruent to 1 mod p. Now multiplying all these numbers together gives

$$(p-1)! \equiv 1 \cdot 1^{(p-3)/2} \cdot (p-1) \equiv -1 \pmod{p}.$$

Conversely, *suppose that p is composite*. If q is a prime divisor of p, then certainly q divides $(p-1)!$, and so $(p-1)!$ is congruent to a multiple of q mod p, and cannot be -1.

Can either of these results give us a quick test for primality?

As we explained in the last section, there is an efficient way to calculate x^n mod n, involving at most $2\log_2 n$ multiplications of numbers not exceeding n and calculation of the remainder mod n after each multiplication. Thus, for example, $2^{589} \equiv 326 \pmod{589}$, so we know that 589 is composite without finding any of its factors.

Unfortunately, this test doesn't work in the other direction. For example, $2^{341} \equiv 2 \pmod{341}$, even though $341 = 11 \cdot 31$ is not prime. We say that 341 is a pseudoprime to the base 2. In general, n is a *pseudoprime* to the base a if n is not prime but $a^n \equiv a \pmod{n}$. Pseudoprimes are not very common, and if we are prepared to take the small risk that the number we chose is a pseudoprime rather than a prime, then we could simply accept such numbers.

We could feel safer if we checked different values. For example, although 341 is a pseudoprime to base 2, we find that $3^{341} \equiv 168 \pmod{3}$, so that 341 is definitely composite.

Unfortunately even this does not give us complete confidence. A *Carmichael number* is defined to be a number that is a pseudoprime to every possible base but not a prime. It seems unlikely that such numbers exist, but they do!

Proposition 23 *The positive integer n is a Carmichael number if and only if it is composite and $\lambda(n)$ divides $n-1$.*

For a Carmichael number can have no repeated prime divisors: if p^2 divides n then p^n mod n is a multiple of p^2 and so not equal to p. Now for such numbers we know that $x^m \equiv x \pmod{n}$ if and only if $\lambda(n)$ divides $m-1$.

Now consider $n = 561 = 3 \cdot 11 \cdot 17$. We have $\lambda(n) = \mathrm{lcm}(2,10,16) = 80$ and 80 divides 560, so 561 is a Carmichael number.

Refinements of this test due to Solovay and Strassen and to Rabin gave fast algorithms which could conclude either that n is certainly composite or that n is 'probably prime', where our degree of confidence could be made as close to 1 as required.

The Miller–Rabin primality test This is a probabilistic algorithm based on the same idea as we saw in the last chapter, for factorising $N = pq$ when an encryption–decryption pair (e,d) for RSA modulo N is known. We assume that the previous tests have not shown n is composite, so we might as well assume that $x^{n-1} \equiv 1 \bmod n$ for all x we ever come across. Now write $n-1 = 2^a.b$, where b is odd, and pick a random number x. Calculate $y \equiv x^b \bmod n$, so that $y^{2^a} \equiv 1 \bmod n$. As before we let z be the last term not equal to 1 in the sequence $y, y^2, y^4, y^8, \ldots, \bmod n$. If $z \equiv -1 \bmod n$, then the test has failed to prove n is composite, and therefore increases the likelihood that n is actually prime. Applying the test again with another value of x increases the likelihood again. Indeed, if n is composite, the probability of the test failing is less than $1/2$ for each value of x. Therefore, if we apply the test 20 times with 20 independent random values of x, and each time it fails to prove that n is composite, then we conclude that n is very likely to be prime (with a probability of error of less than one in a million).

Wilson's theorem doesn't have the drawback of Fermat's Little Theorem: it is a necessary and sufficient condition for primality. That is, if $(n-1)! \equiv -1 \pmod n$, then n is prime; if not, not. Unfortunately, unlike the situation of calculating powers of integers, nobody has ever discovered a quick method of calculating factorials mod n for given n. (The natural method would require $n-1$ multiplications.)

The method finally used by Agrawal, Kayal and Saxena was a kind of combination of these two approaches, together with some ingenuity. They begin with the remark that n is prime if and only if

$$(x-1)^n \equiv x^n - 1 \pmod n$$

as *polynomials*, rather than integers. This is because the coefficients in $(x-1)^n$ are binomial coefficients $\binom{n}{i}$; if n is prime, then $\binom{n}{i}$ is a multiple of n for $i = 1, \ldots, n-1$, but if n has a prime factor q then $\binom{n}{q}$ is not a multiple of n.

This is no good as it stands; we can raise $x-1$ to the nth power with only $2\log_2 n$ multiplications, but the polynomials we have to deal with along the way have as many as n terms, too many to write down. So the trick is to work mod $(n, x^d - 1)$ for some carefully chosen number d. I refer to their paper for the details.

Factorisation

There is not a lot to say about factorisation: it is a hard problem! There are some special tricks which have been used to factorise huge numbers of some special forms, such as *Fermat numbers* $2^{2^n}+1$ and *Mersenne numbers* $2^p - 1$ (for p prime). Of course, we would avoid such special numbers when designing a cryptosystem.

However, one should not overestimate the difficulty of factorisation. Numbers with well over 100 digits can be factorised with today's technology. The gap between primality testing and factorisation is sufficiently narrow that it is necessary to keep updating an RSA system to use larger primes.

Later we may touch on quantum computing and see why the advent of this technology (if and when it comes) will allow efficient factorisation of large numbers and make the RSA system insecure.

We discuss briefly just one factorisation technique: *Pollard's $p-1$ method*. This method works well if the number N we are trying to factorise has a prime factor p such that $p-1$ has only small prime power divisors. Suppose that we can choose a number b such that every prime power divisor q of $p-1$ satisfies $q \leq b$.

The algorithm works as follows. Choose any number $a > 1$, and by successive powering compute $x = a^{b!} \bmod N$. By assumption, every prime power divisor of $p-1$ is at most b, and hence divides $b!$. Hence $p-1$ divides $b!$. Thus, $a^{b!} \equiv 1 \pmod{p}$ by Fermat's Little Theorem, so that p divides $x-1$. By assumption, p divides N. Hence $\gcd(x-1,N)$ is a multiple of p, and so is a non-trivial divisor of N. (Indeed, in the RSA case, if N is the product of two primes, then $\gcd(x-1,N)$ will be a prime factor of N.)

Here is an example. Let $N = 6824347$ and $b = 10$. Choosing $a = 2$, we find that $x = 5775612$ and $\gcd(x-1,N) = 2521$. Thus, 2521 is a factor of N, and with a bit more work we find that it is prime and that $N = 2521 \cdot 2707$ is the prime factorisation of N.

The method succeeds because

$$2521 - 1 = 2^3 \cdot 3^2 \cdot 5 \cdot 7$$

and all the prime power divisors are smaller than 10. Of course, if this condition were not satisfied, the method would probably fail. If we replace 2521 by 2531 in the above example, we find that $N = 2531 \cdot 2707 = 6851417$, $x = 2^{10!} \bmod N = 6414464$, and $\gcd(x-1,N) = 1$.

Because we have to calculate $a^{b!} \bmod N$ by successively replacing a by $a^i \bmod N$ for $i = 1,\ldots,b$, we have to perform $b-1$ exponentiations mod N. So the method will not be polynomial-time unless $b \leq (\log N)^k$ for some k. So we are only guaranteed success in polynomial time if the prime-power factors of $p-1$ for at least one of the divisors of N are at most $(\log N)^k$ – this is small compared to the magnitudes of the primes involved, which may be roughly $\sqrt{N}$.

Thus, in choosing the primes p and q for an RSA key, we should if possible avoid primes for which $p-1$ or $q-1$ have only small prime power divisors; these are the most vulnerable.