Finding large primes and factorising large numbers: is there any alternative to a brute-force search?

by Professor Tim Gowers, 9th October 2006

LaTeXed by Hiro Funakoshi
Please send comments and corrections to tms-sec@srcf.ucam.org

We all know of the standard method of checking whether a number is prime; we divide a number n by primes 2,3,5 and so on up to $\sqrt{n}$. Let's start by considering how long this will actually take; this is an interesting problem as the RSA public key cryptography works by having two prime numbers where knowledge of the product alone is enough to encode a message, but the original primes (i.e. the factorisation of the product) are needed to actually decode and read the message.

Suppose n has 100 digits, then $\sqrt{n}$ will have approximately 50 digits. This means that using the above algorithm, we'll have to look at approximately 10^{50} numbers (or slightly less if we just look at primes), so we clearly need a clever method. In reality even if a clever method is found it will not really matter to banks, as if we can factorise 100 digits in a reasonable time, then we just use 300 digits. Indeed there are "better" methods, but none are currently good enough to jeopardise bank security.

Let's first have a quick recap of modular arithmetic. Pick any number m, and then all numbers in the universe are $0, 1, \ldots, m-1$. We define addition as normal if $a+b < m$, otherwise $a+b = a+b-m$. Multiplication is defined similarly; ab = remainder when ab is divided by m. So, as an example, if we have $m = 13$, then $5 \times 7 = 35 = 2 \times 13 + 9 \equiv 9 \pmod{13}$ - usually we just write $5 \times 7 \equiv 9 \pmod{13}$. It turns out that if m is prime, then all numbers from 1 to $m-1$ have unique multiplicative inverses, so for example when we have $m = 13$:

Number	1	2	3	4	5	6	7	8	9	10	11	12
Inverse	1	7	9	10	8	11	2	5	3	4	6	12

Note here that $12 \times 12 \equiv 1 \pmod{13}$; this is true in general since $(m-1)^2 = m^2 - 2m + 1 \equiv 1 \ (m)$. In modular arithmetic we have something called the

Fermat's Little Theorem (not to be confused of course with Fermat's Last Therorem which has the same abbreviation). This states that if we have a prime p and a number $a \neq 0$, then $a^p \equiv a \pmod{p}$. For example, if $p = 7$ and $a = 3$, then $3^7 = 2187 = 7 \times 312 + 3 \equiv 3\ (7)$. Here is a proof:

Suppose $a \not\equiv 0$.

$$\begin{aligned} ax &\equiv ay \\ bax &\equiv bay \text{ (letting } b = a^{-1}) \\ x &\equiv y \end{aligned}$$

Now consider $1.2.3.\ldots.p-1$ and $(a.1).(a.2).(a.3).\ldots.(a.(p-1))$. We know from the above that $ax \equiv by$ iff $x \equiv y$, so all the numbers in the second expression are distinct. But there are only $p-1$ distinct numbers, so they are the numbers from 1 to $p-1$ in some order (think back to the above table with numbers and their inverses). Thus:

$$\begin{aligned} 1.2.3.\ldots.(p-1) &\equiv a.1.a.2.a.3.\ldots.a.(p-1) \\ 1 &\equiv a.a.a.\ldots.a \\ 1 &\equiv a^{p-1} \\ a &\equiv a^p \quad \square \end{aligned}$$

Note that if we are not working modulo a prime, we do not always have inverses so we cannot carry out the cancellation.

Let's now take a look at the problem of raising powers in modular arithmetic. If for an exponent n, we required n numbers of steps, then our method would not be particularly efficient; if for example we needed to raise something to the power of 10^{200}, and we needed to do 10^{200} steps, then we would very quickly reach the heat death of the universe. If however we could find a method where to raise to an exponent n^{10} we only required on the order of n steps, then we could do this fairly effectively. So, consider the problem of calculating the value of 2^{91} modulo 91. The bad way of doing this would be to work out 2^{91} and then divide through by 91. A better way would be to subtract 91 as you multiply:

$$\begin{aligned} 2^1 &\equiv 2 \\ 2^2 &\equiv 4 \\ 2^5 &\equiv 32 \\ 2^{11} &\equiv 46 \equiv 2^{-1} \end{aligned}$$

$$\begin{aligned} 2^{22} &\equiv 23 \equiv 2^{-2} \\ 2^{45} &\equiv 2^{-2} \times 23 \times 2 \equiv 23 \times 2^{-1} \equiv 57 \\ 2^{91} &\equiv 37 \end{aligned}$$

We can get 23×2^{-1} fairly easily; $23 + 91 = 114$, but $114 \div 2 = 57$. After this we use a similar trick to get from 2^{45} to 2^{91}. This is a fairly evil example, as most people would assume 91 is prime and hence try to use FLT (so as a side note, this also serves as a proof that 91 is not a prime).

This suggests a possible method to check for the primality of a number - that if $a^p \equiv a \pmod{p}$ then p is prime. Unfortunately, this is not the case, for example if we take the number 561, we have $a^{561} \equiv a \pmod{561}$ for all a, but 561 is composite. These numbers are called Carmichael numbers. Let's see how we can use powers to prove that 561 is composite. We have that $a^{561} \equiv a \pmod{561}$, from which we can say that $a^{560} \equiv a \pmod{560}$, as long as a does not share a cofactor with 561; in practice for sufficiently large numbers we can more or less say that $a^{n-1} \equiv 1 \pmod{n}$. Working to mod 561:

$$\begin{aligned} 2^{560} &\equiv 1 \\ 2^{280} &\equiv 1 \\ 2^{140} &\equiv 67 \\ 67^2 &\equiv 1 \\ 67^2 - 1 &\equiv 0 \\ 68 \times 66 &\equiv 0 \end{aligned}$$

Thus 561 is a factor of 68×66, so it cannot be prime. This brings us to another candidate for a primality test; in the above the crucial step was where a square (2^{280}) was congruent to 1 but the square root of this number (2^{140}) was not. In general, working mod p where p is prime:

$$\begin{aligned} x^2 &\equiv 1 \\ (x+1)(x-1) &\equiv 0 \\ x &\equiv \pm 1 \end{aligned}$$

where the last step follows since we are working modulo a prime so we can cancel the terms. From this we can deduce that if there is a non-trivial solution to $x^2 \equiv 1 \pmod{n}$ then n is not prime. It was in fact shown by Miller and Rabin that there is no equivalent of Carmichael numbers for this test; if ± 1 are the only solutions, then n is prime. Furthermore, for any composite n, for at least $\frac{3}{4}$ of the $a < n - 1$ you get some exact square root

of 1 when you work out a^{n-1}. So, if we work out a^{n-1} and it throws out extra roots of 1, then we have confimation that n is composite, else we pick another a and repeat until we are happy that n is prime. Practically, if we pick about 50 different a, then we should be fine.

Like the above test, the best known tests for primality are randomised, and are used on the basis that they are very fast and have a very high probability of working. There is a purely deterministic polynomial time algorithm, due to Agrawal and Saxena, but this is not as fast as the aforementioned tests so is not really used.

Suppose now that we pick a large number at random, say with 5 million digits, and we run the test, and think that it is composite. With a number as large as this it is not likely to be prime; by the prime number theorem the probability that this number is prime is approximately 1/(5 million). However, if we run the test with 50 different a as suggested above, this is roughly equal to the probability that the test fails (i.e. it returns that a number is prime when in fact it is not). So in fact this test is not without problems.

Finally we will quickly cover some tricks for non-brute force factorising. Say we want to factorise 8051 - but this is the difference of two squares and from this we can say that $8051 = 83 \times 97$. In general we have that:

$$\left(\frac{a+b}{2}\right)^2 - \left(\frac{a-b}{2}\right)^2 = ab$$

so we can search through a, b to obtain a factorisation in this way. However, this is still quite slow, but there is a clever trick which we can employ. To factorise 1649:

$$\begin{aligned}
41^2 - 1649 &= 32 \\
42^2 - 1649 &= 115 \\
43^2 - 1649 &= 200 \\
41^2 &\equiv 32 \pmod{1649} \\
43^2 &\equiv 200 \pmod{1649} \\
41^2 \times 43^2 &\equiv 6400 = 80^2 \pmod{1649}
\end{aligned}$$

So, $(41 \times 43 + 80)(41 \times 43 - 80)$ is a multiple of 1649 thus running Euclid on 1843 and 1683, we can get the factors of 1649.