Parabola Volume 28, Issue 2 (1992)

PRIME NUMBERS AND SECRET CODES

David Tacon

"The problem of distinguishing prime numbers from composite numbers and of resolving the latter into their prime factors is known to be one of the most important and useful in arithmetic. It has engaged the industry and wisdom of ancient and modern geometers to such an extent that it would be superfluous to discuss the problem at length ··· The dignity of the science itself seems to require that every possible means be explored for the solution of a problem so elegant and so celebrated." C.F. Gauss (1801).

In the last issue, see "Prime numbers" we discussed such problems as:

(i) how do we check whether or not a 200 digit number is prime;
and

(ii) how do we find the prime factorization of such a digit.

The answers at first sight were very surprising. It turns out to be the case that while it is relatively easy to check for primality it is virtually impossible to find the prime factorization of a randomly chosen 200 digit number. This is reflected in the fact that the largest Mersenne number so far factored is $M_{509} = 2^{509} - 1$ whilst the largest Mersenne number currently known to be prime is $M_{756839} = 2^{756839} - 1$. The magnitude of this result hits home when we realize that we would take about 86 pages of **Parabola** to print this number's decimal expansion. The tests used in checking the primality of large Mersenne numbers provide excellent exercises for testing new supercomputers and this was exactly how the primality of M_{756839} was discovered. Scientists working on a Cray-2 supercomputer at the Atomic Energy Authority Harwell Laboratory were keen to test its reliability. David Slowinski and Paul Gage of Cray Research suggested looking at one hundred Mersenne numbers that might be prime and success was achieved on the 85th number the computer tried. The Cray 2 took just 19 hours of computation before it was proved that M_{756839} is a prime. We might mention here that the Cray-1A supercomputer is reported to have undetected random errors which occur at a frequency of rate about one for every 1000 hours of computing. The main cause of these errors is due to background

radiation from alpha particles. It is impossible to physically shield the memory cells – silicon chips – from this radiation so the memory must be designed under the assumption that at least some of the cells are denoting 0 when they should be denoting 1, and vice versa. Consequently **error correcting codes** must be build into the computer's hardware. These are mathematical elaborations of the **error detecting codes** used on ISBN's : see the article by David Angell in this issue.

We are going to discuss a different type of code: a **secret code** or **cipher**. John Loxton discussed the traditional types of secret codes in the 1982 and 1983 issues of **Parabola** (see 18(3), 19(1) and 19(3)). The basis for the cryptosystem we wish to investigate is the following idea. Suppose we choose a pair of large primes p and q and form the product $n = pq$. (By "large" we mean, say, 100 digits long.) We then know something that no one else can discover: the factorization of n.

Before we can properly understand how we can exploit the above idea we need two more results from number theory. Fortunately neither is very difficult to understand and we can explain them here.

The first goes back to the Greeks and is called the **Euclidean algorithm.** It provides us with a recipe for finding the "greatest common divisor" (a, b) of two positive integers a and b. Using this notation $(12, 18) = 6$ since 6 is the largest, or greatest, integer which divides both 12 and 18. Similarly $(14, 15) = 1$, $(50, 125) = 25$ and so on. When $(a, b) = 1$ we say a and b are relatively prime. You might wonder why we need an algorithm at all when we can so easily read out the greatest common divisor from knowledge of the prime factorization of the numbers, e.g., if $a = 2^3.3.7^5.43^2.61^7$ and $b = 2^2.3^5.7^3.43^4$ then $(a, b) = 2^2.3.7^3.43^2$. There are a number of reasons but certainly a very practical one from our viewpoint is because we cannot effectively find the prime factorization of very big numbers. This afterall is the basis of the idea we are trying to exploit. The Euclidean algorithm, by contrast, is easy to compute! There is also an unexpected payoff from the algorithm which will be more than useful to us.

We begin the Euclidean algorithm by dividing b into a. This allows us to find integers

q_1 and r_1 with

$$a = bq_1 + r_1 \text{ where } 0 \leq r_1 < b.$$

The point is that the remainder r_1 is less than b. The key to understanding why the algorithm works is the observation that the common divisors of a and b are the same as the common divisors of b and r_1. (Keep in mind that if k divides a and b then k divides $r_1 = a - bq_1$). This means that $(a, b) = (b, r_1)$. We now work with b and r_1 to obtain

$$b = r_1 q_2 + r_2 \qquad 0 < r_2 < r_1$$

and we continue to obtain a sequence of equations

$$r_1 = r_2 q_3 + r_3 \qquad 0 < r_3 < r_2$$

$$\vdots$$

$$r_{n-2} = r_{n-1} q_n + r_n \quad 0 < r_n < r_{n-1}$$

$$r_{n-1} = r_n q_{n+1}$$

The remainders continually decrease so that eventually we must achieve a zero remainder r_{n+1} as we have indicated in the last equation. Since $(a, b) = (b, r_1) = (r_1, r_2) = \cdots = (r_{n-1}, r_n)$ and r_n divides r_{n-1} it follows that $(a, b) = r_n$. The extra payoff for us is that we can trace back through the algorithm:

$$r_n = r_{n-2} - r_{n-1} q_n = r_{n-2} - (r_{n-3} - r_{n-2} q_{n-1}) q_n = \cdots .$$

to find integers s and t such that

$$sa + tb = r_n.$$

Here is how this works on a simple example. Suppose we wish to find $(1512, 47)$ and integers s and t such that

$$(1512, 47) = 1512s + 47t.$$

We obtain:

$$1512 = 47.32 + 8$$

$$47 = 8.5 + 7$$

$$8 = 7.1 + 1$$

$$7 = 1.7$$

$$\therefore (1512, 47) = 1. \text{ Furthermore}$$

$$\begin{aligned}(1512, 47) = 8 - 7.1 &= 8 - (47 - 8.5).1 \\ &= 8.6 - 47.1 = (1512 - 47.32).6 - 47.1 \\ &= 1512.6 - 47.193\end{aligned}$$

so that we can set $s = 6,\ t = -193$.

Our second result generalizes Fermat's theorem. Remember in our first article that we proved, assuming p is prime, that

$$a^{p-1} \equiv 1 \bmod p \quad \text{provided } (a, p) = 1.$$

(We wrote $a \equiv b \bmod n$ to indicate that $a - b$ is divisible by n.) We know that this result rarely holds if p is not prime. Nevertheless we can say something constructive in the general case. Euler's theorem states that

$$a^{\phi(n)} \equiv 1 \bmod n \quad \text{provided } (a, n) = 1$$

where $\phi(n)$ is the number of positive integers less than n which are relatively prime to n. For a prime p each of the integers $1, 2, \cdots, p-1$ is relatively prime to p so that $\phi(p) = p-1$. Therefore Fermat's theorem is a special case of Euler's theorem. On the other hand to calculate $\phi(15)$, say, we can proceed by listing the 15-1 numbers $1, 2, 3, \cdots, 14$, and then we can delete the $(5-1) = 4$ numbers divisible by 3 and the $(3-1) = 2$ numbers divisible by 5. We are left with $1, 2, 4, 7, 8, 11, 13, 14$ so that $\phi(15) = (15-1) - (5-1) - (3-1) = 8$. Clearly $\phi(pq) = (pq-1) - (q-1) - (p-1) = (p-1)(q-1)$ whenever p and q are prime. The proof of Euler's theorem is a development of the proof of Fermat's theorem and we can write it down as follows.

Let $r_1 = 1, r_2, \cdots, r_{\phi(n)}$ be the $\phi(n)$ positive integers less than n which are relatively prime to n. Assume $(a, n) = 1$ and for each i let s_i be the remainder when $r_i a$ is divided by n (so $0 \leq s_i < n$ and $r_i a \equiv s_i \bmod n$). Then

(i) each remainder r_i must be one of the r_k (if $(s_i, n) \neq 1$ then $(r_i a, n) \neq 1$ which means either $(a, n) \neq 1$ or $(r_i, n) \neq 1$) and

(ii) $s_i \neq s_j$ if $i \neq j$ (if $s_i = s_j$ then $r_i a \equiv r_j a \bmod n$ which implies $n \mid (r_i - r_j)a$ so that either $r_i = r_j$ or $(a, n) \neq 1$). These two observations imply that the s_j must just be

the r_i rearranged. Consequently

$$\begin{aligned} r_1 r_2 \cdots r_{\phi(n)} a^{\phi(n)} &\equiv r_1 a . r_2 a \cdots r_{\phi(n)} a \bmod n \\ &\equiv s_1 s_2 \cdots s_{\phi(n)} \bmod n \\ &\equiv r_1 r_2 \cdots r_{\phi(n)} \bmod n. \end{aligned}$$

Since $(r_i, n) = 1$ for $i = 1, 2, \cdots, \phi(n)$ it follows that

$$a^{\phi(n)} = 1 \bmod n \quad \text{provided } (a, n) = 1.$$

In fact it follows from Euler's theorem that

$$a^{k\phi(n)+1} \equiv a \bmod n$$

for all a, not just for those a for which $(a, n) = 1$. For the special case where $n = pq$ we can check this easily. If $pq|a$ then the result is trivial so the only cases of real interest are when $a = p$ or $a = q$. If we suppose $a = p$ we need show that $pq|p(p^{k\phi(n)} - 1)$, or equivalently that $q|p^{k(p-1)(q-1)} - 1$, or $q|(p^{k(p-1)})^{q-1} - 1$. But this final statement is true since $(p^{k(p-1)}, q) = 1$.

We can now explain how the RSA public key cryptosystem works. Once set up this cryptosystem allows our agents to send us messages in code, safe in the knowledge that no third party can decipher the code

Step 1: We generate two large primes p and q which we keep secret.

As an illustration we'll let $p = 37$ and $q = 43$. (Remember that if we were setting up a realistic cryptosystem we could find prime numbers with 100 or so digits in their decimal expansion.)

Step 2: We compute $n = pq$.

In our case $n = 1591$. (Even if p and q were large there are no serious difficulties in carrying out the multiplication.)

Step 3: We choose an integer h so that h and $(p-1)(q-1)$ are relatively prime.

We'll let h be 47. (One simple way of ensuring the condition is satisfied is to choose h to be a prime number greater than p and q.)

Step 4: We compute d so that $dh \equiv 1 \mod (p-1)(q-1)$.

We know such an integer d exists when h and $(p-1)(q-1)$ are relatively prime since, by the Euclidean algorithm, there then exist integers s and t such that

$$sh + t(p-1)(q-1) = 1.$$

This means $sh \equiv 1 \mod (p-1)(q-1)$ so that we can let d be congruent to s. In our example $(p-1)(q-1) = 1512$ and we have already found that

$$1 = 1512.6 - 47.193$$

so that $d \equiv -193 \equiv 1512 - 193 \equiv 1319$. (Note that we can also write $1 = 1512(6-47) + 47(-193+1512)$.)

Step 5: We are happy to publish n and h but we keep d secret. At this stage we would be wise to destroy p, q and $(p-1)(q-1)$.

We relax, confident that none will ever factorize 1591! As for $d = 1319$ we keep it under guard in our office.

Why did we destroy $(p-1)(q-1)$ but publish $n = pq$? The secrecy of the cryptosystem depends on a third party, a spy, not being able to work out p and q. We have to release $n = pq$ to the second party with whom we are corresponding so it is no longer secure. If $\phi(n) = (p-1)(q-1)$ and n are intercepted then our spy knows $pq = n$ and $p + q = n - \Phi(n) + 1$ and can immediately find p and q. (Afterall she can write down the quadratic equation which has p and q as roots.)

We have yet to reveal how to code an arbitrary message number m.

According to Euler's theorem we know that, for $k = 1, 2, 3, \cdots$

$$x^{k\phi(n)+1} \equiv x \mod n \quad \text{for any integer } x.$$

But $dh \equiv 1 \bmod \phi(n)$ so that $dh = k\phi(n) + 1$ for some integer k. Consequently we know $m^{dh} \equiv m \bmod n$. This suggests that anyone wishing to send us a message should compute $c = m^h \bmod n$ and send us c. We can then decipher the coded message c by computing

$$c^d \equiv (m^h)^d \equiv m^{dh} \equiv m \bmod n.$$

The question remains as to whether or not this is secure. A spy might have discovered n, h and c and wants to discover m. She can calculate $m = c^d$ if she can discover d or, equivalently, if she can solve

$$dh \equiv 1 \bmod n, \quad \text{given } h \text{ and } n.$$

Fortunately this is a difficult problem, essentially equivalent to factorizing n. Remember trial and error is hardly feasible since there are so many possibilities of d to check!

Finally to complete our example let's suppose that someone wants to use our simple cryptosystem to send us the message $m = 42$. (It's rumoured that this is the answer to the fundamental question concerning the universe!) To transmit this answer to us in secret they should code m as

$$c = 42^{47} \bmod 1591.$$

How do they calculate c efficiently? Remember that in a more realistic situation 47 and 1591 would be "big". The general algorithm requires us to use the binary expansion of the power:

$$47 = 1 \times 2^5 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2 + 1 = (101111)_2$$

so that

$$\begin{aligned}
42^{47} &= \Big(\Big(\Big(\big((42)^2\big)^2 42\Big)^2 42\Big)^2 42\Big)^2 42 \bmod 1591 \\
&= \Big(\Big(\big((173)^2 42\big)^2\Big)^2 42\Big)^2 42 \bmod 1591 \\
&= \Big(\big((128)^2 42\big)^2 42\Big)^2 42 \bmod 1591 \\
&= \big((816)^2 42\big)^2 42 \bmod 1591 \\
&= (945)^2 42 \bmod 1591 = 816 \bmod 1591
\end{aligned}$$

Notice that using this algorithm we needed only 9 multiplications to evaluate 42^{47}. In general if h has α bits in its binary expansion, β of which are 1's then the computation of m^h requires $(\alpha-1)+(\beta-1)$ multiplications. When we receive the coded message $c = 816$ we decipher by calculating

$$816^{1319} \bmod 1591.$$

Since $1319 = 2^{10} + 2^8 + 2^5 + 2^2 + 2 + 1 = (10100100111)_2$ we can decipher using just $10 + 5 = 15$ multiplications. We leave this as an exercise.

The RSA public key cryptosystem was invented by Rivest, Shamir and Adleman in 1978. One of its significant advantages over more traditional cryptosystems is that there is no need to keep the "keys" h and n of the enciphering algorithm secret. In most other systems the secrecy of the message depends upon the secrecy of these keys. This meant that the keys had to be distributed secretly. But how? This was one of the major headaches of conventional cryptography which has been solved by Rivest et al.

WEIGHTY PROBLEMS

1. There are 12 seemingly identical \$2.00 coins. Eleven coins are of equal weight but the one counterfeit coin has a different weight. Find the counterfeit coin with no more than three uses of a beam balance.

2. We have 10 piles with 10 coins in each pile. Each coin in each pile weighs 10g except for the coins in one pile which weigh 9g each. Find the odd pile of coins by using just one weighing with the kitchen scales.