MAT199: Math Alive
Cryptography – Part 2

1 Public key cryptography: The RSA algorithm

After seeing several examples of "classical" cryptography, where the encoding procedure has to be kept secret (because otherwise it would be easy to design the decryption procedure), we turn to more modern methods, in which one can make the encryption procedure public, without sacrifice of security: knowing how to *encrypt* does not enable you to *decrypt* for these public key systems.

To understand how the algorithm was designed, and why it works, we shall need several mathematical ingredients drawn from a branch of mathematics known as *Number Theory*, the study of whole numbers. This branch of mathematics has been studied from antiquity because it was (and is) found to be profound and beautiful, even if people could not find many uses for it. In recent times it has been found very useful, as we shall see. Here are the ingredients we will draw from number theory:

- Modular arithmetic
- Fermat's "little" theorem
- The Euclidean Algorithm

After explaining these ingredients, we turn to

- How the RSA algorithm works
- About factoring and finding prime numbers
- Uses of the RSA algorithm

1.1 Modular arithmetic

Recall from elementary school how to divide a positive integer a (the dividend) by another positive integer b (the divisor). You get a quotient and a remainder. That is, $a = qb + r$ where

q, the quotient, is an integer and r, the remainder, is an integer from 0 to $b-1$. Example, if we want to divide 91 by 17, we find $91 = 5 \times 17 + 6$. The dividend is 91, the divisor is 17, the quotient is 5, and the remainder is 6. This is what we mean by division in this unit; there are no fractions involved.

Definition: the number a is equivalent, or *congruent*, to the number b modulo n if a differs from b by an exact multiple of n. (That is, a number nk where k is an integer.) In shorthand notation we write $a \equiv b \,(\text{mod } n)$.

Examples:

- Even numbers $\equiv 0 \pmod 2$, *i.e.*, when we divide any even number by 2 we always get the same (zero) remainder, so we say that even numbers are congruent.
- Odd numbers $\equiv 1 \pmod 2$.
- $6 \equiv 0 \pmod 2$ because 6 divides into 2 exactly three times and there is no remainder, *i.e.*, $6 = 3 \times 2$ with no remainder.
- $7 \equiv 1 \pmod 2$ because 6 divides into 2 to leave 2 remainder 1, *i.e.*, $7 = 3 \times 2 + 1$.
- $15 \equiv 3 \pmod{12}$ because $15 = 12 \times 1 + 3$.

For this last example we can make a connection between modular arithmetic and clocks – if you count 15 hours round a clock (starting at 12) you get to 3 o'clock. Taking any number mod 12 is equivalent to counting round a clock that many times and looking at the result. Indeed, modular arithmetic is sometimes called *clock arithmetic* for this reason. We can write the idea of modular arithmetic in a more mathematical way and introduce the idea of *equivalent*, or *congruent*, numbers:

It follows from the above definition that if

$$a \equiv b \pmod n \qquad \text{and} \qquad c \equiv d \pmod n \,,$$

then

$$a + c \equiv b + d \pmod n.$$

This is because if $a \equiv b (\text{mod } n)$ then this means that we can write $a = nq + b$ for some integer value q. Similarly, we can write $c = nr + d$. So, adding these two results together means that

$$a + c = n(q + r) + b + d$$

and so indeed

$$a + c \equiv b + d \pmod n.$$

Similarly,

$$ac \equiv bd \pmod n$$

(again, because if you work out the product of a and c, you recover bd by removing multiples of n).

Often we are given the value of a (which can be large) and we want to find the value b which is as small as possible, that is, between 0 and $n-1$. In particular, if we need to make multiple modular calculations, we simplify them after each step, so that we won't need to multiply or add numbers bigger than $n-1$. This process of replacing a number with the remainder you get when you divide it by n is called *reduction modulo* n.

Examples:

- $321 \times 714 \equiv 4 \pmod 5$
  because $321 \equiv 1 \pmod 5$, and $714 \equiv 4 \pmod 5$
  (you can also check that $321 \times 714 = 229,414$, and this $\equiv 4 \pmod 5$).
- $321 \times 714 \equiv 0 \pmod 7$
  because $714 \equiv 0 \pmod 7$.
- $321 \times 715 \equiv 6 \pmod 7$
  because $321 = 280 + 41 \equiv 41 \pmod 7 \equiv 6 \pmod 7$ and $715 = 714 + 1 \equiv 1 \pmod 7$.
- $715^3 = 715 \times 715 \times 715 \equiv 1 \pmod 7$.
- $715^{984} \equiv 1 \pmod 7$.
- $321^3 \equiv 6^3 \pmod 7 \equiv 36 \times 6 \pmod 7 \equiv 1 \times 6 \pmod 7 \quad \Rightarrow \quad 321^3 \equiv 6 \pmod 7$.
  $\uparrow$ because $36 \equiv 1 \pmod 7$
- $321^{984} \equiv 6^{984} \equiv (-1)^{984} \equiv 1 (\mathrm{mod}\ 7)$.

In this way you can find the remainder after division by 7 (or more general n) of huge numbers without doing a lot of work.

What if you want to find $320^{984} \pmod 7$? We know $320^{984} \equiv 5^{984} \pmod 7$, but 5^{984} is still enormous! It turns out we can still do it easily by using the following trick. We start by writing the exponent 984 as a sum of powers of 2. The successive powers of 2 are

$$1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, \ldots$$

We then write 984 as a sum of some of these numbers by first finding the largest power of 2 that does not exceed 984 (that is 512 in this case), and "peeling off" successively smaller powers of 2 as follows:

$$984 = 512 + 472, \qquad 472 = 256 + 216,\ 216 = 128 + 88,\ 88 = 64 + 24,\ 24 = 16 + 8.$$

So this means that we can write

$$\begin{aligned} 984 &= 512 + 256 + 128 + 64 + 16 + 8 \\ &= 2^9 + 2^8 + 2^7 + 2^6 + 2^4 + 2^3. \end{aligned}$$

Comment: to represent a number as a sum of powers of two is actually equivalent to finding the binary representation of a number. For example, the binary representation of 984 is 1111011000. The last digit corresponds to the zero-th power of two, the next to last corresponds to the first power etc. A "one" in the binary representation means that we should include the corresponding power of 2, a "zero" means that we shouldn't include the corresponding power. Let's check: our decomposition $984 = 2^9 + 2^8 + 2^7 + 2^6 + 0 \times 2^5 + 2^4 + 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$ *translates into 1111011000, the binary representation of 984.*

We thus have

$$\begin{aligned} 5^{984} &= 5^{512+256+128+64+16+8} \\ &= 5^{512} \times 5^{256} \times 5^{128} \times 5^{64} \times 5^{16} \times 5^8. \end{aligned}$$

Now

$$\begin{aligned} &5^2 = 25 \equiv 4 \pmod 7 \\ &5^4 = 5^2 \times 5^2 \equiv 4 \times 4 \pmod 7 = 16 \pmod 7 \equiv 2 \pmod 7 \\ &5^8 = 5^4 \times 5^4 \equiv 4 \pmod 7 \\ &5^{16} = 5^8 \times 5^8 \equiv 2 \pmod 7 \\ &5^{32} \equiv 4 \pmod 7, 5^{64} \equiv 2 \pmod 7, 5^{128} \equiv 4 \pmod 7, \\ &5^{256} \equiv 2 \pmod 7, 5^{512} \equiv 4 \pmod 7 \ . \end{aligned}$$

$$\begin{aligned} 5^{984} &= 5^{512} \times 5^{256} \times 5^{128} \times 5^{64} \times 5^{16} \times 5^8 \\ &\equiv \underbrace{4 \times 2} \times \underbrace{4 \times 2} \times \underbrace{2 \times 4} \pmod 7 \\ &\quad 8 \equiv 1 \pmod 7 \\ &\equiv 1 \pmod 7 \end{aligned}$$

In this example, we have shown that $5^{984} \equiv 1 \pmod 7$. In fact, one could show that $a^{984} \equiv 1 \pmod 7$ for any integer a. (Since you can reduce the number before taking all the powers, you only need to really check for $a = 0, 1, 2, 3, 4, 5$, and 6. This would be a good exercise to see that you have mastered modular arithmetic.) The next theorem will show us why this curious fact is true.

Fermat's Little Theorem

Before we talk about the theorem, recall from elementary school that every positive integer n has a unique factorization into prime numbers, for example, $77 = 7 \times 11$ and $120 = 2 \times 2 \times 2 \times 3 \times 5$. We say that an integer n is *divisible* by another integer m to mean that $n = km$ for some integer k. This is the same as saying that n divided by m leaves no remainder, or equivalently, that $n \equiv 0 \pmod m$. Note that n is divisible by a prime number p precisely when p is one of the prime factors of n. Furthermore, it is a special property of primes that if m and n are integers and if mn is divisible by p, then either m is divisible by p or n is divisible by p, because p must

be in the prime factorization of m or of n. For example $14 \times 10 = 140$ is divisible by 7, which means that either 14 or 10 is divisible by 7. (In fact 14 is.) This works because 7 is prime. On the other hand, note that $10 \times 15 = 150$ is divisible by 6 (namely $150 = 6 \times 25$) but neither 10 nor 15 is divisible by 6. This is because $6 = 2 \times 3$ is composite, that is, not prime, and the factor of two is found in 10 while the factor of three is found in 15.

Fermat's "little" theorem states that if p is prime, then $a^p \equiv a \pmod{p}$ for all a. An alternative form states that $a^{p-1} \equiv 1 \pmod{p}$ when p is prime and a is any integer not divisible by p. (This last condition is needed for the alternative form, but not for the usual form.) Let's look at the usual version of Fermat's little theorem in several steps. Try it out:

$$p = 3 \Rightarrow a^3 \stackrel{?}{\equiv} a \pmod{3}?$$

Check: if a is itself a multiple of 3, then both a^p and a are equal to 0 (mod 3), and the identity is true. So we need check only the cases where a is not a multiple of 3. We can start by casting out all 3-s from a, so that we need to check only the cases $a = 1$ and $a = 2$. For $a = 1$, it is obvious that $a^3 = 1 \equiv a \pmod{3}$. For $a = 2$, we have $a^3 = 8 \equiv 2 \pmod{3}$ and so this works.

Our next example is $p = 5$. Is it true that

$$p = 5 \Rightarrow a^5 \stackrel{?}{\equiv} a \pmod{5}$$

for all $a \equiv 1$, 2, 3, or 4 (mod 5)? (For $a \equiv 0 \pmod{5}$, we again don't need to check anything, because then the identity is obviously true.)

Check:

- $a = 1 \rightarrow$ Okay
- $a = 2 \Rightarrow a^5 = 32 \equiv 2 \pmod{5}$
- $a = 3 \Rightarrow a^2 = 9 \equiv 4 \pmod{5}$
  $\Rightarrow a^5 = a \times a^2 \times a^2 \equiv 48 \pmod{5} \equiv 3 \pmod{5}$
- $a = 4 \Rightarrow a^2 = 16 \equiv 1 \pmod{5}$
  $\Rightarrow a^4 \equiv 1 \pmod{5}$
  $\Rightarrow a^5 \equiv a \times a^4 \equiv a \equiv 4 \pmod{5}$
  $\Rightarrow$ Okay again.

We could keep checking for one prime number after the other, but this would never be quite enough—there could always be a nagging doubt that somewhere, beyond the largest number that we checked, there was one p for which the statement would not be true. We need a different argument, that establishes the truth of Fermat's little theorem without **any** doubt. To see how we can do that, let's look at a different way of checking whether a^7 is $\equiv a \pmod{7}$, for all a between 1 and 6. We'll try to do it in such a way that we can hope that a similar argument would work for other p.

Here is the idea: we will let p be our prime number and a be any integer. We want to prove that $a^p \equiv a \pmod{p}$. If $a \equiv 0 \pmod{p}$, then clearly $a^p \equiv 0 \pmod{p}$ also, and so $a^p \equiv a \pmod{p}$. So it only remains to prove that $a^p \equiv a \pmod{p}$ in the case where $a \not\equiv 0 \pmod{p}$, i.e., where a is not divisible by p. This will require a devious excursion of a few paragraphs rather than a frontal assault. So please be patient.

Consider the list of the first p nonnegative numbers, that is 0, 1, 2, ..., $p-2$, $p-1$. Also consider the list we get by multiplying each element of the first list by a. This new list is 0, a, $2a$, ..., $(p-2)a$, $(p-1)a$. Now we claim that if you reduce these new numbers modulo p, you will get the original list, but perhaps in a scrambled order. Let's do it when $p = 7$ and $a = 4$. The original list is just 0, 1, 2, 3, 4, 5, 6. The new list is 0, 4, 8, 12, 16, 20, 24. If we reduce modulo 7, the new list becomes 0, 4, 1, 5, 2, 6, 3, which is just the original list in scrambled order!

Let's see why this is true for *any* prime p and *any* number a which is not divisible by p. We want to show that the elements of 0, a, $2a$, ..., $(p-2)a$, $(p-1)a$ reduce modulo p to 0, 1, 2, ..., $p-2$, $p-1$, although not necessarily in ascending order. We claim it will be sufficient to show that no element of the of the new list is equivalent to any other element of the new list modulo p. This will be enough because all numbers leave a remainder from 0 to $p-1$ when we divide by p, and if no two numbers on the new list are equivalent modulo p, then the p numbers on the new list will leave p different remainders, which must be all the numbers from 0 to $p-1$ in some order.

Here's the proof that no two elements on the new list are equivalent modulo p: Take two elements of the new list, say ja and ka with $0 \leq j < k \leq p-1$. We are going to suppose that $ja \equiv ka \pmod{p}$ (exactly the opposite of what we claim is true!) and show that this leads to a contradiction. If we assume this, then $ka - ja \equiv 0 \pmod{p}$. Then $(k-j)a \equiv 0 \pmod{p}$, so that $(k-j)a$ is divisible by the prime p. Since a is assumed to not be divisible by p, this forces $k-j$ to be divisible by p. But then note that $0 \leq j < k \leq p-1$, which means that $0 < k-j \leq p-1$. But there is no positive number that is less than or equal to $p-1$ and at the same time is divisible by p. So our assumption that $ja \equiv ka \pmod{p}$ leads inevitably to a contradiction, so it must not be true! So in fact, all numbers on the new list are different modulo p.

OK. So now we know that 0, a, $2a$, ..., $(p-2)a$, $(p-1)a$ reduces modulo p to a list of p different remainders, so each possible remainder 0, 1, 2, ..., $p-2$, $p-1$, must appear once. That is 0, a, $2a$, ..., $(p-2)a$, $(p-1)a$ reduces modulo p to the list 0, 1, 2, ..., $p-2$, $p-1$, perhaps in a scrambled order. We remove the zero entries from the lists and conclude that a, $2a$, ..., $(p-2)a$, $(p-1)a$ reduces modulo p to 1, 2, ..., $p-2$, $p-1$, not necessarily in order. Since the two lists have the same elements modulo p, they have the same products modulo p:

$$a \times 2a \times \ldots \times (p-2)a \times (p-1)a \equiv 1 \times 2 \times \ldots \times (p-2) \times (p-1) \pmod{p},$$

which we can rearrange by putting all $p-1$ factors of a up front:

$$a^{p-1} \times 1 \times 2 \times \ldots \times (p-2) \times (p-1) \equiv 1 \times 2 \times \ldots \times (p-2) \times (p-1) \pmod{p},$$

or equivalently, by subtracting

$$a^{p-1} \times 1 \times 2 \times \ldots \times (p-2) \times (p-1) - 1 \times 2 \times \ldots \times (p-2) \times (p-1) \equiv 0 \pmod{p},$$

or, just as well,

$$(a^{p-1} - 1) \times 1 \times 2 \times \ldots \times (p-2) \times (p-1) \equiv 0 \pmod{p},$$

which is the same as saying that $(a^{p-1}-1)\times 1\times 2\times\ldots\times(p-2)\times(p-1)$ is divisible by the prime p. Well, none of the factors 1, 2, ..., $p-1$ are divisible by p since these are numbers smaller than p. So we must have $a^{p-1}-1$ divisible by the prime p, i.e., $a^{p-1}-1\equiv 0 \pmod{p}$, i.e., $a^{p-1}\equiv 1 \pmod{p}$. This is the alternative form of Fermat's little theorem, which only applies when a is not divisible by p (as we have been assuming). To get the usual form, multiply both sides by a to get $a^p\equiv a \pmod{p}$ and we have proved Fermat's little theorem!

The Euclidean Algorithm

So now we know that $a^p \equiv a \pmod{p}$ when p is a prime and a is any integer. When $a\equiv 0 \pmod{p}$, this is obvious, and when a is not divisible by p, we showed this by showing that $a^{p-1}\equiv 1 \pmod{p}$, and then multiplying both sides by a. One consequence of the latter case is that if a is not divisible by p, then $a\times a^{p-2}\equiv 1 \pmod{p}$. We say that a^{p-2} is a *multiplicative inverse of a modulo p*. More generally, if m is any number (prime or not) and $ab\equiv 1 \pmod{m}$, we say that b is a multiplicative inverse of a modulo m.

It turns out that whenever a and m have no common factors (i.e., any prime appearing in the prime factorization of a does not appear in the prime factorization of m), then we can always find a multiplicative inverse of a modulo m. Furthermore, there is an algorithm (procedure guaranteed to work) that finds the inverse very quickly! The algorithm is named after Euclid, an ancient Greek mathematician, in whose book *Elements* it appears (although he was likely documenting something known before his time).

We shall illustrate with an example rather than a proof. Suppose that we want to find a multiplicative inverse of a modulo m with $a=20$ and $m=63$. That is, we want a number b so that $20\times b\equiv 1 \pmod{63}$. Note that $a=20=2^2\times 5$ and $m=63=7\times 9$, so they have no common factor. The Euclidean algorithm works like this:

We consider our two elements 63 and 20. We put them on a list, with the larger element first.

$$\begin{array}{c} 63 \\ 20 \end{array}$$

Our list will also contain bookkeeping information in the form of congruences modulo the first number. (In this case, modulo 63.) Record the reductions of the first two numbers modulo 63, i.e., $63\equiv 0 \pmod{63}$ and $20\equiv 20 \pmod{63}$. So our list is actually

$$\begin{array}{l} 63\equiv 0 \pmod{63} \\ 20\equiv 20 \pmod{63} \end{array}$$

This is the start of the list.

Now we build up the list step-by-step. We list the various things you need to do to complete a single step, illustrating these actions with our ongoing example.

1. Divide the second-to-last element in the list by the last element in the list.

So we divide 63 by 20 to get the remainder: $63 = 3 \times 20 + 3$.

2. Write the remainder as the dividend (63) minus the quotient (3) times the divisor (20).

   That is, write $3 = 63 - 3 \times 20$.

3. Now we change this equation to an equivalence modulo the first element on the list (i.e., modulo 63).

   If two numbers are equal, they are equivalent modulo 63, so $3 \equiv 63 - 3 \times 20 \pmod{63}$.

4. Now we modify this equivalence: we use previous equivalences in our list to replace the dividend (63) and divisor (20).

   From our list, we know that $63 \equiv 0 \pmod{63}$ and $20 \equiv 20 \pmod{63}$, so our equivalence $3 \equiv 63 - 3 \times 20 \pmod{63}$ becomes $3 \equiv 0 - 3 \times 20 \equiv -3 \times 20 \pmod{63}$.

   Notice that this new equivalence gives our remainder 3 as a multiple of 20 modulo 63. We should always get our remainder as a multiple of the second element on the list (20) modulo the first element on the list (63).

5. Now append the new remainder plus this congruence to the end of the list to get

$$\begin{aligned} 63 &\equiv 0 \pmod{63} \\ 20 &\equiv 20 \pmod{63} \\ 3 &\equiv -3 \times 20 \pmod{63} \end{aligned}$$

   This concludes a single step.

Now keep doing these five-part steps. Second step:

1. Divide the second-to-last element by the last: $20 = 6 \times 3 + 2$.
2. Write it as $2 = 20 - 6 \times 3$.
3. Make it a congruence: $2 \equiv 20 - 6 \times 3 \pmod{63}$.
4. Replace dividend (20) and divisor (3) using congruences that are already on the list: $2 \equiv 20 - 6 \times (-3 \times 20) \pmod{63}$. This should work out to some multiple of 20 modulo 63: in fact, $2 \equiv 20 + 18 \times 20 \equiv 19 \times 20 \pmod{63}$.
5. The new list:

$$\begin{aligned} 63 &\equiv 0 \pmod{63} \\ 20 &\equiv 20 \pmod{63} \\ 3 &\equiv -3 \times 20 \pmod{63} \\ 2 &\equiv 19 \times 20 \pmod{63} \end{aligned}$$

Third step:

1. Divide the second-to-last element by the last: $3 = 1 \times 2 + 1$.
2. Write it as $1 = 3 - 1 \times 2$.
3. Make it a congruence: $1 \equiv 3 - 1 \times 2 \pmod{63}$.
4. Replace dividend (3) and divisor (2) using congruences that are already on the list: $1 \equiv -3 \times 20 - 1 \times (19 \times 20) \pmod{63}$. This should work out to some multiple of 20 modulo 63: in fact, $1 \equiv -22 \times 20 \pmod{63}$.
5. The new list:

$$\begin{aligned} 63 &\equiv 0 \pmod{63} \\ 20 &\equiv 20 \pmod{63} \\ 3 &\equiv -3 \times 20 \pmod{63} \\ 2 &\equiv 19 \times 20 \pmod{63} \\ 1 &\equiv -22 \times 20 \pmod{63} \end{aligned}$$

Fourth step:

1. Divide the second-to-last element by the last: $2 = 2 \times 1 + 0$.
2. Write it as $0 = 2 - 2 \times 1$.
3. Make it a congruence: $0 \equiv 2 - 2 \times 1 \pmod{63}$.
4. Replace dividend (2) and divisor (1) using congruences that are already on the list: $0 \equiv 19 \times 20 - 2 \times (-22 \times 20) \pmod{63}$. This should work out to some multiple of 20 modulo 63: in fact, $0 \equiv 63 \times 20 \pmod{63}$.
5. The new list:

$$\begin{aligned} 63 &\equiv 0 \pmod{63} \\ 20 &\equiv 20 \pmod{63} \\ 3 &\equiv -3 \times 20 \pmod{63} \\ 2 &\equiv 19 \times 20 \pmod{63} \\ 1 &\equiv -22 \times 20 \pmod{63} \\ 0 &\equiv 63 \times 20 \pmod{63} \end{aligned}$$

Now we must stop, because we cannot divide 1 by 0.

Recall that we were looking for a multiplicative inverse of 20 modulo 63, i.e., we wanted some number b such that $20 \times b \equiv 1 \pmod{63}$. If you have been very alert, you would see that we did this, even before the Euclidean algorithm terminated!

Look at the penultimate (second-to-last) line of our list, which says $1 \equiv -22 \times 20 \pmod{63}$. So -22 is a multiplicative inverse of 20 modulo 63. Of course $-22 \equiv -22 + 63 \equiv 41 \pmod{63}$, so if we want to use a positive number as the multiplicative inverse of 20 modulo 63, we can use 41.

So the Euclidean algorithm gave us the multiplicative inverse of 20 modulo 63. How can we be sure that it will work for to find the multiplicative of *any* number a modulo *any* number m when a and m have no common factor? (We always insist on the no-common-factor condition for reasons beyond the scope of this lesson.) Is it just a freak accident that the information we wanted appeared on our list? Is it even necessary that the Euclidean algorithm come to a stop? Might we not just go on dividing forever?

Let's analyze the last question. Note that each number is smaller than its predecessor since it is the remainder of some calculation where we divide by the predecessor. For example the remainder of 63 divided by 20 must be from 0 to 19 (in fact, it is 3). So we know the third entry of our list must be less than 20, that is, less than the second entry of our list. Since each number is less than the previous one, it is inevitable that the list end in zero. In fact the numbers get small quite quickly, so the algorithm is quite fast in coming to a halt.

Okay, so the process stops, but how can we be sure that one line in the list will show us the multiplicative inverse of a modulo m? Were we just "lucky" to find the equivalence $1 \equiv -22 \times 20 \pmod{63}$ as the second-to-last line of our list in the example above? A typical line of the list (after the initial two lines) is of the form

$$r \equiv k \times a \pmod{m},$$

i.e., some remainder is equal to a multiple of a modulo m. If r happens to be 1, then we have $1 \equiv k \times a \pmod{m}$, so that k is the multiplicative inverse of a modulo m. So all we need to do is show that 1 is one of the numbers appearing in our list. If this is so, then we shall always obtain the multiplicative inverse by reading off this entry on our list.

In fact, we claim that the number right before 0 in our list will *always* be 1. This is a little harder to prove, but here is the key idea:

Claim: *Any two consecutive numbers in the list have no common factor, i.e., there is no prime p such that both numbers are divisible by p.*

You can check that it is true on our list 63, 20, 3, 2, 1, 0. Note that 63 is divisible by the primes 3 and 7, but 20 is not. And 20 is divisible by the primes 2 and 5, but 3 is not. And 3 and 2 have no common prime factor, nor do 2 and 1, nor 1 and 0. Suppose for the moment that the above claim is true in all applications of our algorithm (we will prove this shortly). Then it is not hard to prove that the second-to-last (penultimate) entry on the list must be 1. This is because the last element, 0, is divisible by *every prime p*, in a very stupid but very genuine way, namely, $0 = p \times 0$. So if the penultimate entry were divisible by *any prime p whatsoever*, then the last two consecutive entries would be divisible by p, violating our claim. So the penultimate entry must not be divisible by any prime. So it must be 1.

So we just need to prove the claim above that each consecutive pair in our list has no common factor; this shows that the penultimate entry must be 1; if the penultimate entry in the list is 1, the congruence for that entry gives the multiplicative inverse of a modulo m. So let's prove the claim. The first two elements in our list are m and a, and they have no common factor because we never run the Euclidean algorithm here unless m and a have no common factor. (It is done in other circumstances, but not in this class.)

Now let's show that the second and third element have no common factor. The third element on our list (which we shall call r) is obtained as the remainder when we divide the first element m by the second element a. Say $m = qa + r$, where q is the quotient. If *both* a (the second element) and r (the third element) had a common prime factor p, then $m = qa + r$ would also be divisible by p. But then m and a would both be divisible by p—preposterously violating our initial assumption that m and a have no common factor. So a and r cannot have any common factors—that would lead straight to a contradiction.

So the second and third elements of the list, a and r, have no common factor. The fourth element (which we shall call s) is obtained as the remainder when we divide the second element by the third. So $a = kr + s$, where k is the quotient. Again, if the third and fourth elements (r and s) had a common factor p, then $a = kr + s$ would also have p as a factor. Then a and r would have a common factor, contrary to what we just showed. So r and s have no common factor.

We can keep on going like this: if two consecutive entries u and v have no common factor, and if w is the remainder that you get when you divide u by v, then v has no factor in common with w. We can continue this argument until we reach the end of the list, and thus prove that each consecutive pair in the list has no common factor.

We conclude this section with one more example of the Euclidean algorithm, just to help you get the hang of it. Suppose we want to find the multiplicative inverse of 69 modulo 25, i.e., a number b such that $69 \times b \equiv 1 \pmod{25}$. Note that $69 \equiv 19 \pmod{25}$, so it is just as well to say that we are looking for a multiplicative inverse of 19 modulo 25. Note that 19 is prime and $25 = 5^2$, so the numbers have no common factors, so we can apply our algorithm. We begin the list

$$\begin{aligned} 25 &\equiv 0 \pmod{25} \\ 19 &\equiv 19 \pmod{25} \end{aligned}$$

First step:

1. Divide the second-to-last element by the last: $25 = 1 \times 19 + 6$.
2. Write it as $6 = 25 - 1 \times 19$.
3. Make it a congruence: $6 \equiv 25 - 1 \times 19 \pmod{25}$.
4. Replace dividend (25) and divisor (19) using congruences that are already on the list: $6 \equiv 0 - 1 \times 19 \equiv -1 \times 19 \pmod{25}$.
5. The new list:

$$\begin{aligned} 25 &\equiv 0 \pmod{25} \\ 19 &\equiv 19 \pmod{25} \\ 6 &\equiv -1 \times 19 \pmod{25} \end{aligned}$$

Second step:

1. Divide the second-to-last element by the last: $19 = 3 \times 6 + 1$.
2. Write it as $1 = 19 - 3 \times 6$.
3. Make it a congruence: $1 \equiv 19 - 3 \times 6 \pmod{25}$.
4. Replace dividend (19) and divisor (6) using congruences that are already on the list: $1 \equiv 19 - 3 \times (-1 \times 19) \equiv 4 \times 19 \pmod{25}$.
5. The new list:

$$\begin{aligned} 25 &\equiv 0 \pmod{25} \\ 19 &\equiv 19 \pmod{25} \\ 6 &\equiv -1 \times 19 \pmod{25} \\ 1 &\equiv 4 \times 19 \pmod{25} \end{aligned}$$

We can actually abandon the Euclidean algorithm before we get the zero remainder, because our list now has the entry $1 \equiv 4 \times 19 \pmod{25}$, which tells us that 4 is a multiplicative inverse of 19 modulo 25. (Hence 4 is a multiplicative inverse of 69 modulo 25, since $19 \equiv 69 \pmod{25}$.) We only continued the Euclidean algorithm beyond this point in our first example to prove that the algorithm halts and that it always works properly. Now that we know this, there is no need to go to the end!

How the RSA algorithm works

As a warm-up, let us first recall Fermat's little theorem:

$$a^p \equiv a \pmod{p}$$

when p is a prime and a is any integer.

If we multiply the two sides of our last equation with a^{p-1}, then we obtain

$$a^{p-1} \times a^p \equiv a^{p-1} \times a \equiv a^p \equiv a \text{ (mod p)}.$$

We can repeat this multiplication with a^{p-1} as many times as we want, leading to

$$a^{K(p-1)} \times a^p \equiv a \text{ (mod p) .}$$

Writing $a^p = a^{p-1} \times a$, and regrouping all the $p - 1$ in the exponent, we obtain

$$a^{(K+1)(p-1)+1} \equiv a \text{ (mod p) .}$$

Writing the constant $K + 1$ as N, we can rewrite this as

$$a^{N(p-1)+1} \equiv a \text{ (mod p),} \tag{1}$$

which is true for any a and any N.

The RSA algorithm uses Fermat's little theorem in a very ingenious way. Suppose that p and q are two primes. We compute their product: $n = p \times q$. Now we pick a positive integer r that has no common factor with $(p-1)(q-1)$. Then we find the multiplicative inverse of r modulo $(p-1)(q-1)$, that is we find a number s such that

$$rs \equiv 1 \pmod{(p-1)(q-1)}.$$

So $rs = 1 +$ a multiple of $(p-1)(q-1)$, or $rs = L(p-1)(q-1)+1$. The Euclidean algorithm finds s rapidly.

Now let's describe the encryption. To encrypt, one only needs to know n and r. For now, let us suppose that our message is a number x smaller than n (any text message can be turned into a sequence of small numbers). How do we encrypt? We compute

$$y \equiv x^r \pmod{n} .$$

The number y will be the encrypted version of the number x. To decrypt, we compute $z \equiv y^s \pmod{n} \equiv x^{rs} \pmod{n}$. We then claim that z is our original x again. To show that this is the case, we first show that $z \equiv x \pmod{n}$. By the equivalence labeled (1) above, and using that $rs = L(p-1)(q-1)+1$, we see that

$$x^{rs} = x^{L(p-1)(q-1)+1} \equiv x \pmod{p} .$$

Equivalently, we can say that

$$x^{rs} - x = \text{ multiple of } p .$$

For the same reason, we have

$$x^{rs} = x^{L(p-1)(q-1)+1} \equiv x \pmod{q} ,$$

or

$$x^{rs} - x = \text{ multiple of } q .$$

Since p and q are two different primes, $x^{rs} - x$ can be a multiple of both p and q only if it is a multiple of their product $n = p \times q$, which implies, as claimed above,

$$x^{rs} \equiv x \pmod{n} .$$

This equivalence establishes only that x^{rs} and x have the same remainders when divided by n. But remember that we took care to pick x positive and less than n itself, so the remainder of dividing x^{rs} by n is always x; so we can recover x by computing the smallest positive number that is equivalent to $y^s \pmod{n}$.

The idea to use this for encryption is now the following: Let x be a block of "plain text" (but in the form of numbers). Then compute

$$y \equiv x^r \pmod{n} .$$

The pair of values n, r is the **public encryption key**. This information is publicly available, so anyone can compute y if they are given x. To decrypt, you also need to know s, the **private decryption key**; to decrypt, you simply compute

$$y^s \pmod{n} \equiv x \ .$$

That is, you need to know s to decrypt. Now s is the multiplicative inverse of r modulo $(p-1)(q-1)$. The outsiders know r, and if they knew $(p-1)(q-1)$, then it would be easy (with the Euclidean Algorithm) to compute s. But they don't know $(p-1)(q-1)$. They know n, which is equal to pq, but they don't have n factored into p and q. To find $(p-1)(q-1)$, they would need to know the prime factors p and q of n, and factoring large numbers is not easy.

In 1994, a team of many mathematicians made the headlines (in the science section) because they factored a number of 129 digits. The effort took 8 months of computation by about 600 volunteers from more than 20 countries, on all continents except Antarctica. The number was known as an RSA-129 challenge; this challenge had been published in Scientific American in August 1977. One of the inventors of RSA, Ronald Rivest, had then made an estimation of the time that would be needed to break RSA-129, based on 1977 knowledge and technology. It was equal to 4 million lifetimes of the universe. (A lot of progress had been made between 1977 and 1994, mostly in the speed of the technology, but the 1994 factorization was still a heroic effort, for a number with only 129 digits.)

Here is that factorization:

RSA-129 and Its Factors

114381625757888867669235779976146612010218296721 2 (number continues on next line)
042362562561842935769352457338978305971235639587050589890751475992900268795435 41

=

349052951084765094914784961990389813341776463849338784399082057 7

×

327691329932667095499619881908344614131776429679929425397982885 33

Once the factorization was obtained, it could be used to decrypt a message that Rivest, Shamir, and Adelman had given as part of the challenge. The decrypted message was

The magic words are squeamish ossifrage

This sentence was hidden in a 128-digit ciphertext. In this case, n had "only" 129 digits. With every digit that you add, the problem becomes harder. (You'll see this in the on-line Lab!) In practical RSA schemes, n has 400 or more digits ....

How then do we construct n? We have to find p and q first.

About factoring and finding prime numbers

RSA is hard to break because factoring large numbers is believed to be **hard**.

How do you factor a number? Here is a straightforward and somewhat naive approach:

$69 = 3 \times 23$
$72 = 2 \times 36 = 2^2 \times 18 = 2^3 \times 9 = 2^3 \times 3^2$
$143 = \ ?$ not divisible by $2, 3, 5, 7$ but divisible by 11 !
$\Rightarrow 143 = 11 \times 13.$

You run through the primes, starting from 2 and up from there, and check every time whether this is a divisor. If you haven't found any divisors smaller than $\sqrt{n}$, then you can stop: the number n is prime.

$667 = \ ?$ not divisible by $2, 3, 5, 7, 11, 13, 17, 19$ but divisible by 23
$\sqrt{667} \approx 25.8$
$667 = 23 \times 29$

$661 = \ ?$ not divisible by $2, 3, 5, 7, 11, 13, 17, 19, 23$
$\sqrt{661} \approx 25.6$ no other primes below 26
$\Rightarrow 661$ is prime.

Factoring a number with three digits is not so hard, although not as immediate as factoring a number of 2 digits. How hard is it to factor a number of 100 digits?

To factor a number of 100 digits by the method just proposed, we would have to run through the different primes, checking each time whether we have a divisor or not, and this until we either find a divisor or until we reach the square root of our number n. This square root will have about 50 digits. How many primes are there that have 50 digits or less? After all, primes become less and less common as you move up: between 1 and 100 there are 28 primes (more than 1 out of 4, on average!), but between 1000 and 1100 there are only 16, a much smaller number. So maybe they become so rare in those regions of very large numbers, that the factoring task is sped up there! And maybe there is only a finite number of primes to worry about anyway?

This last hopeful statement turns out to be mere wishful thinking. The Greeks already knew that there are infinitely many primes.

> The argument is the following "proof by contradiction":
>
> > (A "proof by contradiction" is like an alibi in a detective story. First you suppose that x **is** the culprit. But then it would follow that y could not have seen x at a time where y did, in fact, see x. Contradiction. So x cannot be the culprit.)
>
> Suppose that there **were** only a finite number of primes $P_1, P_2, \ldots, P_n$. This would mean that every number that is larger than all the P_j is necessarily divisible by one of them, since it cannot be prime itself. Now compute $x = P_1 \times P_2 \times P_3 \times s \times P_n + 1$. It is larger than all the P_j, but not divisible by any of them. Contradiction. So it can't be true that there are only finitely many primes.

Not only are there infinitely many primes, but their distribution is rather unpredictable (in the sense that nobody has ever found a simple formula that produces all of the primes). On the other hand, they are not totally unpredictable: for instance, each prime is less than 2 times its predecessor. (You can check on the first few: $2, 3, 5, 7, 11, 13, 19, 23, 29, 31, 37, \ldots$ see `http://primes.utm.edu` for a larger list of primes and lots of other information on primes). Prime numbers become less frequent as we keep enumerating all the integers. We have pretty accurate estimates of how many primes there are, say, between Y and $10Y$. The "prime number theorem" gives a formula for the number of primes with N or fewer digits, that is pretty accurate when N is larger than, say, ten:

$$\begin{aligned} P_N &= \text{number of primes with } N \text{ digits or less} \\ &\simeq 4.3 \times \tfrac{10^{N-1}}{N} \ . \end{aligned}$$

Example: for $N = 4$ the correct number is 1229; the formula gives $\frac{4.3\times 10^3}{4} \simeq 1075$, which is about 13% off. For $N = 5$, the correct number is 9592, and the formula gives about 8600, which is already more accurate, less than 11% off. The accuracy increases with N.

That means that the number of primes with exactly N digits is approximately given by

$$P_N - P_{N-1} \simeq 4.3 \times \frac{10^{N-1}}{N} - 4.3 \times \frac{10^{N-2}}{N-1}$$

For $N = 100$, say, this gives

$$P_{100} - P_{99} \simeq 3.9 \times 10^{97} \ .$$

So there are indeed much fewer prime numbers here: on average, only one out of every 238 numbers is prime in this region, but the region is so vast, that it still makes for a highly respectable number of primes (much larger than the number of atoms that physicists estimate to be in the visible universe ...). So, even though primes do become less frequent, there are still plenty of them.

In practice, there are much smarter approaches than our method of successive trials. Mathematicians have developed algorithms, with the intriguing names of the quadratic sieve, or the number field sieve, which are much more efficient.

The factoring of the 193-digit number RSA-640 (so called because its binary representation has 640 binary digits) in 2005 took about five months of computing by a whole team of mathematicians using 80 computers. The total run time would have been about 30 years if run on a single desktop computer. Here is the exact quote describing this end of the series of RSA-challenges (quoted from `http://www.rsa.com/rsalabs/node.asp?id=2092`):

> "The factoring research team of F. Bahr, M. Boehm, J. Franke, T. Kleinjung continued its productivity with a successful factorization of the challenge number RSA-640, reported on November 2, 2005. The factors [verified by RSA Laboratories] are:
> 1634733645809253848443133883865090859841783670033 0
> 9231218111085238933310010450815121211816751157 9
> and
> 1900871281664822113126851573935413975471896789968
> 5154936666385390880271038021044989571912614655 71
> The effort took approximately 30 2.2GHz-Opteron-CPU years according to the submitters, over five months of calendar time."

Originally, the RSA challenge listed an even larger number to factor, with 212 digits. In practical cryptographic applications, people use 400-digit numbers, harder still. Unless some major breakthroughs are made, those larger RSA codes are secure for the foreseeable future. (Unless someone has already made the breakthrough and isn't telling us...)

But wait, ... if factoring is hard, won't it be hard to find prime numbers as well? After all, a number is prime only if you can't factor it into two smaller factors, and we have seen that factoring is hard? The answer is that mathematicians have found ways to establish that a number is prime with a high degree of certainty (but not absolute mathematical certainty).

For instance, we can turn the alternative version of Fermat's little theorem on its head and say:

> If, for some a between 1 and $n-1$, we find that $a^{n-1} \not\equiv a \pmod{n}$, then n is necessarily composite (i.e., not prime).

This test is called the Fermat test for primality. By itself, this criterion is not strong enough for our purposes, because it does not tell us when a number **is** prime.

So the idea is that we choose a lot of different values of a. If ever we get $a^{n-1} - 1$ not divisible by n, we can stop: our number n is certainly not prime. It fails. But if the number does not fail after a large number of such tests, then we can be almost certain that it is prime (without perfect certainty). However, for practical purposes this is usually enough; picking a composite number which somehow doesn't fail a battery of tests of this kind is more unlikely than a lot of other possibilities that are not worth worrying about. Given that about one in every 260 of the 100-digit numbers is prime, it is therefore easy to find 100-digit primes. You just pick a random string of 100 digits, and you test it. If it is not prime, you just pick another one, and you continue until you have found one.

Comment: in fact there are composite numbers n for which $a^{n-1} - 1$ is a multiple of n for almost any a between 1 and $n-1$; such numbers n are rather rare themselves; a number n for which $a^{n-1} \equiv 1 \pmod{n}$ whenever a shares no common factors with n is called a Carmichael number.

Remark: Note that the existence of these criteria means that it is possible to say conclusively that a number can be factored without being able to produce the factors! (In fact, this is a feat that we can already achieve just by applying Fermat's little theorem.)

Note: The largest known prime as of February 1, 2009 is:

$$2^{43,112,609} - 1$$

discovered by the GIMPS PrimeNet network on August 23rd, 2008. This is a 12,978,189 digit number (!) and the 45th known Mersenne prime.

More information can be found at `http://www.mersenne.org/` . The GIMPS PrimeNet is a network of "prime-searchers," volunteers who download small pieces of one gigantic computation that run quietly in the background on their PCs, and of which they report the results back to the GIMPS central hub.

For more information about large prime numbers, see `http://primes.utm.edu/` , and links you find there.

Uses of RSA algorithm

As secure encryption, private key cryptography is used a lot in financial transactions, or in computer security, in communications where security is needed.

For example, a bank publishes a pair (n, r). Anybody can send a message to the bank, and only the bank can decrypt it. This way the secrecy of the transaction is supported in a sense that nobody can decrypt a message. But there is a drawback of this method. Suppose it is known that we send only three types of messages to our bank: Transfer \$1000, withdraw \$1000, deposit \$1000. Anyone can pre-compute the encrypted versions of these messages. Then when a malicious person intercepts our message, he/she can compare with the three precomputed messages and violate the secrecy of the transaction. There are easy methods to avoid this problem. For example, we can add a very big random number at the end of the message and then encrypt.

In a similar manner, any individual can publish his/her (n, r) pair and can receive encrypted messages from anybody.

Other than secrecy, we would like our message transaction to have many other properties. For example, we would like to authenticate the message - to have a guarantee that the message was sent by a particular person. Also, we would like to have a non-repudiation property - that the person who sent a message can't deny sending it. Both of these properties can be achieved using a digital signature scheme based on the RSA algorithm, as follows. Suppose I would like to send a message to a bank; to do this I need to know the bank's public key. I also make my own public key available. I encrypt my message using my bank's public key. I also encrypt the resulting message using my own private key. Then I send both messages to the bank. The bank decrypts the first part using its private key, then the bank encrypts the second part of the message using my public key, and checks that it is the same as the first part. As only I could have encrypted the second part of the message using my private key, the signature is accepted.

Comment: in reality, I encrypt not the whole message, but only a message digest. I also have to add a time stamp. Can you figure out why I would need to add a time stamp?