Number Theory and Cryptography

A textbook for MATH2088 and MATH2988

THE UNIVERSITY OF SYDNEY

R. Howlett

Contents

Week 1

Cryptography is the science of secret communication. Starting with a message that anyone could read (the *plaintext*) one disguises or *encrypts* it, so that the resulting message (the *ciphertext*) can only be read by someone who knows how to *decrypt* it. In mathematical parlance, encryption is a function from the set of possible plaintext messages to the set of possible ciphertexts, and decryption is the inverse function. Ideally, it should be very quick and easy to encrypt a message, and very quick and easy to decrypt it if you know how. And it should be next to impossible to figure out how to decrypt a message if you have not been told how.

People have been encrypting and decrypting messages since ancient times. Until some 35 years ago, the methods that were employed were such that if you knew how to encrypt a message then you could easily work out how to decrypt one. But it does not have to be this way: knowing how to compute values of a function does not necessarily tell you how to compute values of the inverse function. The celebrated RSA cryptosystem, invented in 1978 by R. L. Rivest, A. Shamir and L. Adleman, demonstrates this. Using the RSA system, a person can publicly release an encryption process and still keep the decryption process secret. This gives anyone who wants to send a confidential message to the holder of the decryption key the ability to do so.

The RSA system is based on elementary number theory. In essence the idea is that it is much easier to multiply numbers than to factorize numbers. If a, b and c are integers such that $a = bc$ then it is easy to compute a given b and c, but much harder to find b and c given only a. In MATH2088/2988 we shall study number theory, and in due course we shall find out how the RSA system works. But since number theory is an interesting subject in its own right, we shall not be solely concerned with the mathematics behind RSA. Even if one were only interested in cryptography, it would not be sensible to restrict oneself to the mathematics of RSA. There are other cryptosystems in use that are based on other results from number theory. And who is to say what number theory may be used in some yet to be invented cryptosystem!

§1 Set Notation

Modern mathematics is based on set theory. Since it is impossible to define everything and prove everything, even the purest of pure mathematicians has to start with some primitive (or undefined) concepts, and simply trust that other people can understand. And *set* is one of these primitive concepts. Intuitively, a set is a single object that is in some way composed of other objects. These other objects are called the *elements* or *members* of the set. But no further definition of any of these terms is given.

Recall that the symbols { and } are used for sets. Thus the notation $\{5, 7, 9\}$ means the set whose three elements are the numbers 5, 7 and 9.

Notation of the form $\{\ldots \mid \ldots\}$ means "the set of all ... such that ... ". In this context the vertical line | is read as "such that". The ... to the left of the vertical line should be a formula of some kind, and the ... to the right of the vertical line should be a condition of some kind (called the *membership condition*). The elements of the set in question are all things given by those values of the formula for which the membership condition is satisfied.

Here is some more standard notation (which you have no doubt already met). The symbol "$\in$" means "in" or "is in" or "is an element of". Thus the sentence "$3 \in \{1, 2, 3, 4, 5\}$" says that 3 is an element of the set $\{1, 2, 3, 4, 5\}$ (which is true!). The symbol "$\subseteq$" means "is a subset of". So the sentence "$\{3, 5\} \subseteq \{1, 5, 9\}$" says that $\{3, 5\}$ is a subset of $\{1, 5, 9\}$ (which is false). Recall that S is a subset of T if and only if every element of S is also an element of T.

Finally, if A and B are sets then $A \cup B$ is the set of all things that are either elements of A or elements of B (or both), and $A \cap B$ is the set of all things that are elements of both A and B. That is,

$$\begin{aligned} A \cup B &= \{\, x \mid x \in A \text{ or } x \in B \,\}, \\ A \cap B &= \{\, x \mid x \in A \text{ and } x \in B \,\}. \end{aligned}$$

Recall that $A \cup B$ is called the *union* of A and B, while $A \cap B$ is called the *intersection* of A and B.

§2 Natural numbers: foundations

In MATH2088/2988 we use the following notation and terminology.

1) The set of all real numbers is denoted by $\mathbb{R}$.
2) The set of all integers (whole numbers) is denoted by $\mathbb{Z}$. Note that this includes zero and negative integers.
3) The set of all positive integers is denoted by $\mathbb{Z}^+$.
4) The set consisting of the positive integers and zero will be called the set of *natural numbers* and denoted by $\mathbb{N}$.
5) The set of all rational numbers is denoted by $\mathbb{Q}$. That is,

$$\mathbb{Q} = \{\, n/m \mid n \in \mathbb{Z},\ m \in \mathbb{Z}^+ \,\}.$$

Note that many people use the term "natural numbers" to mean exactly the same thing as "positive integers", but since it seems wasteful to use two good names for the same thing, we adopt the convention that zero is a natural number.

Number theory is, first and foremost, the study of the positive integers. But to study positive integers adequately one also needs to consider other kinds of numbers. Hence we make frequent use of $\mathbb{R}$ and $\mathbb{Q}$.

We shall not go into the foundations of mathematics; so we shall not attempt to define $\mathbb{Z}$, $\mathbb{Q}$ or $\mathbb{R}$, and many of the basic properties of numbers will be used without proof. But if we were to try to prove these basic things then our starting point would be Peano's Postulates. These are five properties that completely characterize $\mathbb{N}$; they can be considered as the axioms of the natural numbers. Every theorem of number theory that you will ever learn, as well as all those that you won't, can be proved using only the Peano Postulates.

P1) There is a natural number denoted by 0.
P2) For every natural number a there is another natural number, $S(a)$, called the "successor" of a. The successor of a is unique: no a has more than one successor.
P3) There is no a such that $S(a) = 0$.
P4) If a, b are natural numbers with $a \neq b$ then $S(a) \neq S(b)$.
P5) Suppose that A is a set of natural numbers having the property that $0 \in A$ and the property that $S(a) \in A$ whenever $a \in A$. Then $A = \mathbb{N}$.

Using these axioms one defines $1 = S(0)$, $2 = S(1)$, and so on, and then defines addition so that $a + 0 = a$, and $a + S(b) = S(a + b)$ for all a and b. (It is a consequence of this definition that $a + 1 = S(a)$ for all $a \in \mathbb{N}$.) It is also easy to define multiplication and the order relation ($a < b$). We pretend that we have done all this.

A side remark: it is possible to use the axioms of set theory to construct a set $\mathbb{N}$ satisfying P1 to P5. The usual way is to define 0 to be the empty set, then define 1 to be the single-element set $\{0\}$, then define 2 to be $\{0, 1\}$, and so on. But we certainly are not interested into delving into the foundations of mathematics in such detail.

Observe that P5 above is essentially the Principle of Mathematical Induction. To see this, suppose that $P(n)$ is some statement involving the natural number n, and we want to prove that $P(n)$ is true for all n. Define

$$A = \{\, n \in \mathbb{N} \mid P(n) \text{ is true}\,\}.$$

Then we want to prove that $A = \mathbb{N}$, and according to P5 it suffices to prove that

(*i*) $0 \in A$ (that is, $P(0)$ is true), and
(*ii*) $S(a) \in A$ whenever $a \in A$ (that is, if $P(a)$ is true then $P(a + 1)$ is true).

This is exactly the Principle of Mathematical Induction as you have seen it in previous years.

Another formulation of the same thing is the so-called "Least Integer Principle", which can be stated as follows:

Every nonempty set of natural numbers has a least element.

This is a reasonably obvious fact: a decreasing sequence of natural numbers cannot go on forever. Indeed, if $n_0, n_1, \ldots, n_k$ are natural numbers with $n_0 > n_1 > \cdots > n_k$

then $k \leq n_0$, since there are only n_0 natural numbers less than n_0. Now if S is a nonempty set of natural numbers then we can choose a natural number $n_0 \in S$. If n_0 is not the smallest number in S then we can choose $n_1 \in S$ with $n_0 > n_1$. If n_1 is not the smallest then we can choose $n_2 \in S$ with $n_0 > n_1 > n_2$. Continuing on in this way, we shall encounter the least element of S in at most n_0 steps.

Here is a formal proof of the Least Integer Principle using the Principle of Mathematical Induction.

Let B be a subset of $\mathbb{N}$, and suppose that B has no least element. We shall show that B is empty. Indeed, we use induction on n to prove the following statement, $P(n)$.

$P(n)$: *B does not contain any of the natural numbers* $0, 1, 2, \ldots, n$.

Firstly, 0 is the least natural number, and $B \subseteq \mathbb{N}$; so B cannot contain anything less than 0. So if $0 \in B$ then 0 is the least element of B. Since by assumption B has no least element, it follows that $0 \notin B$. Hence $P(0)$ is true.

Now suppose that $P(a)$ is true for some natural number a. Then none of the numbers $0, 1, 2, \ldots, a$ are in B. Thus no natural number less than $a + 1$ is in B. This means that if $a + 1$ is in B then $a + 1$ is the least element of B. Since B has no least element we conclude that $a + 1 \notin B$, and hence none of $0, 1, 2, \ldots, a, a + 1$ are in B. So $P(a + 1)$ is true.

This completes our induction, showing that $P(n)$ is true for all n. In particular, $n \notin B$, for all natural numbers n. As $B \subseteq \mathbb{N}$ this means that B is empty.

§3 The division algorithm

(1.1) Proposition: *Let b be a positive integer. For every natural number a there is a unique integer r such that*

(*i*) $0 \leq r < b$,

(*ii*) $a = qb + r$ *for some natural number q.*

To see that this is true we can use the fact that a decreasing sequence of natural numbers must terminate. The idea is simple: start at the number a, and repeatedly subtract b so long as this gives a nonnegative answer.

Here is a (very inefficient) computer program to do this.

```
Residue:= function(a,b)
   Q:=0;
   R:=a;
   while R ge b do
      Q:=Q+1;
      R:=R-b;
   end while;
   return R,Q;
end function;
```

This is written in the language of the computer algebra package MAGMA that we shall be using in the computer tutorials. It is reasonably intuitive. Note that "ge" (in the fourth line of the above code) means "greater than or equal to".

Note that the equation $a = Qb + R$ is satisfied for the initial values $Q = 0$ and $R = a$. Moreover, since $a = Qb + R$ implies that also $a = (Q+1)b + (R-b)$, changing the values of Q and R by adding 1 to Q and subtracting b from R will not destroy the equation $a = Qb + R$. So no matter how often the commands between "while" and "end while" are performed, $a = Qb + R$ will still hold. Moreover, these commands are only performed when $R \geq b$, in which case $R - b$ will still be nonnegative. So the successive values of R that we obtain are all natural numbers. And each successive one is smaller than its predecessor (since it is obtained by subtracting the positive number b). Our fact about decreasing sequences of natural numbers therefore implies that after some finite number of iterations a situation will be reached in which the "while" condition ($R \geq b$) is not satisfied. In other words, the natural number R will then be less than b. So the returned values of R and Q satisfy $a = Qb + R$ and $0 \leq R < b$, as required.

This algorithm shows the there do exist numbers q and r satisfying the conditions (1) and (2) of the above proposition, and tells us how to find such numbers. However, the proposition also says that these numbers are unique, and we really ought to prove this. So we should show that if

$$a = qb + r, \qquad 0 \leq r < b$$

and

$$a = q'b + r', \qquad 0 \leq r' < b$$

then we must have $r' = r$ and $q' = q$.

Since $0 \leq r < b$, subtracting r' gives

$$-r' \leq r - r' < b - r'.$$

But $-b < -r'$ (since $r' < b$) and $b - r' \leq b$ (since $r' \geq 0$). So $-b < -r' \leq r - r' < b - r' \leq b$, and, in particular,

$$-b < r - r' < b. \qquad (1)$$

But $qb + r = a = q'b + r'$, and so

$$r - r' = q'b - qb = (q' - q)b.$$

So (1) gives $-b < (q' - q)b < b$. But now $b > 0$, and so we can divide through by b without harming the $<$ signs:

$$-1 < q' - q < 1.$$

Since $q' - q$ is an integer this means that $q' - q = 0$; that is, $q' = q$. And it follows that $r = a - qb = a - q'b = r'$ too, as required.

So the uniqueness claim in the proposition has been proved.

Terminology: We call the number r in Proposition (1.1) the *residue of a modulo b*, and denote it by $\mathrm{res}_b(a)$. The number q is called the *quotient*.

The residue of a modulo b can also be called the *remainder when a is divided by b*.

If the residue of a modulo b is 0 then $a = qb$ for some integer q. Under these circumstances we say that "b divides a". We use the notation "$b|a$" to mean "b divides a". Note that there are several alternative ways of saying it: "a is a multiple of b", "b is a divisor of a" and "b is a factor of a" all mean exactly the same thing as "b divides a".

§4 Greatest common divisors

If a and b are natural numbers then there is a unique natural number d having the following two properties:

(*i*) $d|a$ and $d|b$;

(*ii*) for every integer c such that $c|a$ and $c|b$ we have that $c|d$.

The first of these properties says that d is a common divisor of a and b (that is, a divisor of them both), and the second property says that every common divisor of a and b is a divisor of d.

(1.2) Definition: The natural number d satisfying (*i*) and (*ii*) above is called the *greatest common divisor* of the given natural numbers a and b.

Note that a divisor of a divisor of a number n must also be a divisor of n itself: if $m|k$ and $k|n$ then $m|n$. For if $k = hm$ and $n = lk$ then $n = l(hm) = (lh)m$. So given that d is a common divisor of a and b, it follows that every divisor of d is also a common divisor of a and b. Since the second property above is the converse statement—every common divisor of a and b is a divisor of d—it follows that (*i*) and (*ii*) together mean that the set of all common divisors of a and b must be exactly the set of all divisors of d.

So we can say that the greatest common divisor of a and b is the number d such that the set of divisors of d is the set of common divisors of a and b.

We give an algorithm for finding the number d given a and b. This algorithm is called the *Euclidean Algorithm* since it appears in Book VII of Euclid's *Elements*. We present it as MAGMA code, using the function `Residue` we defined above.

```
GCD:= function(a,b)
   A:=a;
   B:=b;
   while B ne 0 do
      R:=Residue(A,B);
      A:=B;
      B:=R;
   end while;
   return A;
end function;
```

The function starts with $A = a$ and $B = b$, and repeatedly replaces A by B and B by the residue of A modulo B, until B becomes 0. When this happens, A is the gcd of the original a and b.

Before investigating why it works, let us do an example.

(1.3) ***Example:*** *Find the gcd of* 42 *and* 78.

$$78 = 1 \times 42 + 36 \qquad (\text{So Residue}(78, 42) = 36.)$$
$$42 = 1 \times 36 + 6 \qquad (\text{So Residue}(42, 36) = 6.)$$
$$36 = 6 \times 6 + 0 \qquad (\text{So Residue}(36, 6) = 0.)$$

A	B
78	42
42	36
36	6
6	0

When B becomes zero the value of A is 6. So $\gcd(78, 42) = 6$.

The Euclidean Algorithm depends on the following result.

(1.4) Lemma: *Suppose that a, $b \in \mathbb{N}$ with $b > 0$. If $a = gb + s$ for some g, $s \in \mathbb{N}$ then the set of common divisors of a and b is the same as the set of common divisors of b and s.*

Proof. Let A be the set of common divisors of a and b and B the set of common divisors of b and s. We must show that every element of A is also an element of B, and vice versa.

Let $d \in A$. Then d is a common divisor of a and b, and so we have $a = hd$ and $b = kd$ for some integers h and k. Since $s = a - gb$ this gives

$$s = hd - g(kd) = (h - gk)d,$$

so that d is a divisor of s. But d is also a divisor of b; so d is a common divisor of b and s. Thus $d \in B$, and we have shown that every element of A is an element of B.

Conversely, suppose that $d \in B$. Then d is a common divisor of b and s, and so we have $b = hd$ and $s = kd$ for some integers h and k. Since $a = gb + s$ this gives

$$a = g(hd) + kd = (gh + k)d,$$

so that d is a divisor of a. But d is also a divisor of b; so d is a common divisor of a and b. Thus $d \in A$, and we have shown that every element of B is an element of A. □

In particular the lemma shows that the set of common divisors of A and B equals the set of common divisors of B and Residue(A, B) (since $A = QB +$ Residue(A, B) for some $Q \in \mathbb{N}$). Since each step of the algorithm replaces (A, B) by $(B, \text{Residue}(A, B))$, the set of common divisors of A and B does not change as the algorithm proceeds. Note also that since B is replaced by Residue(A, B), which is always less than B, the successive values of B form a decreasing sequence of natural numbers. This sequence must necessarily terminate, which means that the time must come when the "while" loop is not performed. In other words, at some stage B becomes 0, the process stops, and the value of A is then returned.

It remains to observe that when $B = 0$ the set of all common divisors of A and B is just the set of all divisors of A. This is because every integer is a divisor of 0. (Recall that $n|m$ means $m = kn$ for some integer k. If $m = 0$ then we can just take $k = 0$, since it is certainly true that $0 = 0n$. So $n|0$ is true for all n.) So the final value of A is a number whose set of divisors is the set of all common divisors of the numbers a and b that we started with.

§5 The extended Euclidean Algorithm

It is an important fact that if a and b are arbitrary natural numbers and $d = \gcd(a, b)$ then there exist integers s and t such that $as + bt = d$. (Note that, unless it happens that one of a or b is a divisor of the other, d will necessarily be less than both a and b. So one of the integers s, t will have to be negative and the other positive.) The extended Euclidean Algorithm computes s and t as well as d.

Recall that the basic Euclidean Algorithm starts by putting $A = a$ and $B = b$, and then modifies A and B without changing their gcd. The extended Euclidean Algorithm works by keeping track of a 2×2 matrix $\left(\begin{smallmatrix} K & L \\ M & N \end{smallmatrix}\right)$ having the property that, at all stages,

$$\begin{pmatrix} K & L \\ M & N \end{pmatrix}\begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix}.$$

To ensure that this equation holds initially, when $A = a$ and $B = b$, we initially take $\left(\begin{smallmatrix} K & L \\ M & N \end{smallmatrix}\right)$ to be the identity matrix $\left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$.

At each step of the algorithm we divide B into A, obtaining a quotient Q and remainder R satisfying $A = QB + R$, and we replace A by B and B by R. That is,

$$\begin{pmatrix} A \\ B \end{pmatrix} \quad \text{is replaced by} \quad \begin{pmatrix} B \\ R \end{pmatrix} = \begin{pmatrix} B \\ A - QB \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -Q \end{pmatrix}\begin{pmatrix} A \\ B \end{pmatrix}.$$

So to guarantee that we do not change the product

$$\begin{pmatrix} K & L \\ M & N \end{pmatrix}\begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix},$$

we replace $\begin{pmatrix} K & L \\ M & N \end{pmatrix}$ by $\begin{pmatrix} K & L \\ M & N \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & -Q \end{pmatrix}^{-1}$:

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} K & L \\ M & N \end{pmatrix}\begin{pmatrix} A \\ B \end{pmatrix} = \underbrace{\begin{pmatrix} K & L \\ M & N \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & -Q \end{pmatrix}^{-1}}_{= \text{ new}\begin{pmatrix} K & L \\ M & N \end{pmatrix}} \underbrace{\begin{pmatrix} 0 & 1 \\ 1 & -Q \end{pmatrix}\begin{pmatrix} A \\ B \end{pmatrix}}_{= \text{ new}\begin{pmatrix} A \\ B \end{pmatrix}}.$$

Note that

$$\begin{pmatrix} 0 & 1 \\ 1 & -Q \end{pmatrix}^{-1} = \begin{pmatrix} Q & 1 \\ 1 & 0 \end{pmatrix}.$$

Evaluating the product $\left(\begin{smallmatrix} K & L \\ M & N \end{smallmatrix}\right)\left(\begin{smallmatrix} Q & 1 \\ 1 & 0 \end{smallmatrix}\right)$, we find that the new values for K, L, M and N are, respectively, $KQ + L$, K, $MQ + N$ and M.

Note also that $\left(\begin{smallmatrix} Q & 1 \\ 1 & 0 \end{smallmatrix}\right)$ has determinant -1, and so

$$\det\left(\begin{pmatrix} K & L \\ M & N \end{pmatrix}\begin{pmatrix} Q & 1 \\ 1 & 0 \end{pmatrix}\right) = \det\begin{pmatrix} K & L \\ M & N \end{pmatrix}\det\begin{pmatrix} Q & 1 \\ 1 & 0 \end{pmatrix} = -\det\begin{pmatrix} K & L \\ M & N \end{pmatrix}.$$

It follows that at each stage of the algorithm the determinant of $\left(\begin{smallmatrix} K & L \\ M & N \end{smallmatrix}\right)$ is multiplied by -1. Initially this determinant is $+1$ (the determinant of the identity matrix), and so after n steps it will be $(-1)^n$.

Let us do an example, to make this more concrete. We first apply the basic Euclidean Algorithm to find the gcd of 78 and 21. We form a table with two rows: the first row starts with the two given numbers (78 and 21 in this case), and continues with the successive remainders that are obtained, while the successive quotients form the second row. The completed table is

$$\begin{array}{cccccc} 78 & 21 & 15 & 6 & 3 & 0 \\ & & 3 & 1 & 2 & 2, \end{array}$$

corresponding to the following equations:

$$\begin{aligned} 78 &= 3 \times 21 + 15, \\ 21 &= 1 \times 15 + 6, \\ 15 &= 2 \times 6 + 3, \\ 6 &= 2 \times 3 + 0. \end{aligned}$$

The last nonzero remainder is 3; so $\gcd(78, 15) = 3$.

Now we construct a table that will give the successive values of K, L, M and N. Before doing any calculations, the partially filled in table is as follows:

$$\begin{array}{cccccc} & & 3 & 1 & 2 & 2 \\ \hline 0 & 1 & & & & \\ 1 & 0 & & & & \end{array}$$

Here the numbers in the top row are the quotients from the basic Euclidean Algorithm, in the order in which they were obtained. The two numbers in the second row are the initial values of L and K, and the two numbers in the third row are the initial values of N and M. In effect, we have swapped the order of the columns of $\left(\begin{smallmatrix} K & L \\ M & N \end{smallmatrix}\right)$; it turns out that doing this gives a more convenient way of setting out the calculations.

We work from left to right filling in the spaces, writing the new values of K and M alongside the old. Recall that the new L and N are the old K and M.

$$\begin{array}{ccc} & \ldots & \ldots & Q & \ldots \\ \hline \ldots & L & K & & \\ \ldots & N & M & & \end{array} \quad \longrightarrow \quad \begin{array}{ccccc} & \ldots & \ldots & Q & \ldots \\ \hline \ldots & L & K & QK+L & \\ \ldots & N & \underbrace{M \quad QM+N}_{=\text{new} \begin{smallmatrix} L & K \\ N & M \end{smallmatrix}} & & \end{array}$$

So to obtain the new number to put into the next space, multiply the number Q at the top of the column by the number immediately to the left of the space (K or M) and add on the number to the left of that (L or N). In our example the completed table is as follows:

$$\begin{array}{cccccc} & & 3 & 1 & 2 & 2 \\ \hline 0 & 1 & 3 & 4 & 11 & 26 \\ 1 & 0 & 1 & 1 & 3 & 7. \end{array}$$

We have yet to see why this helps us find integers s and t such that $as + bt = d$. The point is that when the algorithm terminates we have $A = d$ and $B = 0$, and so

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} K & L \\ M & N \end{pmatrix}\begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} K & L \\ M & N \end{pmatrix}\begin{pmatrix} d \\ 0 \end{pmatrix} = \begin{pmatrix} Kd \\ Md \end{pmatrix}.$$

Thus $K = a/d$ and $M = b/d$. But we also have that

$$KN - ML = \det\begin{pmatrix} K & L \\ M & N \end{pmatrix} = \pm 1,$$

whence it follows that

$$\frac{a}{d}N - \frac{b}{d}L = \pm 1. \tag{2}$$

Multiplying through by $\pm d$ gives $as + bt = d$, where $s = N$ and $t = -L$ (if the sign is plus) or $s = -N$ and $t = L$ (if the sign is minus).

In our example above the final values of K and M are indeed 78/3 and 21/3, and the final values of L and N, namely 11 and 3, should be t and $-s$ or $-t$ and s. We readily check that $78 \times 3 + 21 \times (-11) = 3$.

We remark that the method used in the example above, namely to go right through the basic Euclidean algorithm, recording the sequence of quotients obtained, and then complete a table for the values of K, L, M and N, is fine for hand calculations, so long as the basic algorithm does not take too many steps to complete and the sequence of quotients to be recorded is therefore not too long. But in truth this way of doing things is inefficient. The real point of the extended Euclidean Algorithm is that it is not necessary to store the sequence of quotients: you can work out everything as you go along.

For hand calculations a convenient procedure is to construct a table with four rows, which for the example above will end up looking like this:

78	21	15	6	3	0
		3	1	2	2
0	1	3^-	4	11^-	26
1	0	1	1^-	3	7^-.

This just combines into one table the two separate tables we constructed before; the superscript minus sign that alternates between the third and fourth rows is just a convenient device to make it easy to determine the correct sign in Eq. (2) above. Initially, before we do any calculation, the table has only two columns, and the successive columns are added, one by one, using the same calculations as above:

78	21
0	1
1	0 ,

78	21	15
		3
0	1	3^-
1	0	1 ,

78	21	15	6
		3	1
0	1	3^-	4
1	0	1	1^-,

and so on. Note that at each step only the numbers in the last two columns are used in the calculation of the new column.

Here is suitable MAGMA code for a function that performs the Extended Euclidean Algorithm. The function takes two natural numbers a and b as arguments, and returns $d = \gcd(a,b)$ and numbers s and t satisfying $sa + tb = d$.

```
extendedEA:= function(a,b)
   A:=a;
   B:=b;
   K:=1;
   L:=0;
   M:=0;
   N:=1;
   det:=1;
   while B ne 0 do
      Q,R:=Quotrem(A,B);
      A:=B;
      B:=R;
      newK:=Q*K+L;
      L:=K;
      K:=newK;
      newM:=Q*M+N;
      N:=M;
      M:=newM;
      det:=-det;
   end while;
   return A,det*N,-det*L;
end function;
```

Note that `Quotrem` is a built-in MAGMA function that does much the same thing as our home-grown function `Residue` defined on p.4, although it does it extremely efficiently, whereas `Residue` is extremely inefficient.*

§6 Prime numbers and composite numbers

(1.5) Definition: A positive integer p is said to be *prime* if $p > 1$ and there is no integer d such that $1 < d < p$ and $d|p$. A positive integer m is said to be *composite* if $m > 1$ and m is not prime. The positive integer 1 is neither prime nor composite.

We are particularly interested in expressing positive integers as products of primes. Our reason for this interest derives from the fact that the security of the RSA cryptosystem relies on the fact that the problem of expressing a given integer as a product of primes is computationally difficult.

Observe that if n is a positive integer and $n = de$ then either $d \leq \sqrt{n}$ or $e \leq \sqrt{n}$ (since $d, e > \sqrt{n}$ would give $de > (\sqrt{n})^2 = n$). The obvious way to find a factor of n is by "trial division": for each integer m such that $1 < m \leq \sqrt{n}$, try dividing m into n, and see if the remainder is 0. If we go all the way from 2 to $\sqrt{n}$ without finding any integer m that is a divisor of n then it must be that n is prime.

This simple-minded approach works well enough when n has a small factor, but otherwise it takes too long. (Of course, given the speed of modern computers, n really has to be very large to make certain that trial division—and other naive factorization methods—is not practicable.)

Another technique for finding a factor of n is "Fermat's factorization method". This proceeds as follows. First of all, check if n is even. If it is, then 2 is a factor of n, and we are finished. Now suppose that n is odd, and suppose also that $n = de$ for some integers d, e with $1 < e \leq d < n$. Then d and e must both be odd. Now $d + e$ and $d - e$ will both be even, and so $m = (d + e)/2$ and $k = (d - e)/2$ are both integers. Furthermore, since $e > 1$ and $d - 1 > 0$ we see that $(d - 1)e > d - 1$, giving

$$d + e < de + 1 = n + 1,$$

and consequently $m = (d + e)/2 < (de + 1)/2 = (n + 1)/2$.

Now $d = m + k$ and $e = m - k$, and it follows that

$$n = de = (m + k)(m - k) = m^2 - k^2.$$

This shows that every odd composite number can be expressed as a difference of two squares. Fermat's method is simply to go through all possible values of m,

* The function `Quotrem(A,B)` returns the quotient and the remainder obtained when `A` is divided by `B`, whereas `Residue(A,B)` returns the same things in the opposite order. (So to use `Residue` instead of `Quotrem` you would need to replace `Q,R:=Quotrem(A,B)` by `R,Q:=Residue(A,B)`.)

checking if $m^2 - n$ is a perfect square. If we find an m with $m^2 - n = k^2$ for some k, then we have a factorization of n as $(m + k)(m - k)$.

Since we want $m^2 - n = k^2 \geq 0$, we see that m must be at least as large as $\sqrt{n}$. We proved above that $m < (n + 1)/2$. So we just check all m in this range. This technique may be useful when trying to factorize a number n that is not too large, using a non-programmable calculator; it works best when n happens to have factors d and e that are reasonably close to $\sqrt{n}$, and in practice it is often better than trial division in such cases. But for factorizing enormous numbers Fermat's method is no use at all, because the number of values of m to test is roughly $\frac{1}{2}(n + 1) - \sqrt{n}$, which is far too large. Trial division, which only requires checking $\sqrt{n}$ cases, is much better.

(1.6) ***Example:*** *Use Fermat's method to factorize* 1769.

Since $42^2 = 1764 < 1769 < 43^2 = 1849$, the first value of m to try is 43.

$43^2 - 1769 = 80$, not a perfect square.

$44^2 - 1769 = (44^2 - 43^2) + (43^2 - 1769) = 87 + 80 = 167$, not a perfect square.

$45^2 - 1769 = (45^2 - 44^2) + (44^2 - 1769) = 89 + 167 = 256$. This is a perfect square!

So we find that $1769 = 45^2 - 16^2 = (45 + 16)(45 - 16) = 61 \times 29$.

Although Fermat's factorization method itself is no use for factorizing very large numbers, it is in some sense the forerunner of one of the best factorization methods known: the quadratic sieve. The details of this factorization method are beyond the scope of MATH2088/2988; however, like Fermat's method it relies on the principle that a solution of $n = m^2 - k^2$ yields a factorization of n.

Week 2

§7 Some definitions

If a, $b \in \mathbb{Z}$ then "$b|a$" means "$a = qb$ for some $q \in \mathbb{Z}$". Note that $a = qb$ is equivalent to $a = (-q)(-b)$ and also to $-a = (-q)b$ and $-a = q(-b)$. So it follows that $b|a$ is equivalent to $\pm b|\pm a$ for any/all choices of the signs $\pm$.

If $a \in N$ and $b \in \mathbb{Z}^+$ we have defined the *residue of a modulo b* to be the integer r that satisfies $0 \leq r < b$ and $b|(a - r)$. We make the same definition if $a < 0$. In this case the algorithm to calculate the residue is to start with a and add on b repeatedly until the result is nonnegative.

If $b < 0$ the residue of a mod b is defined to be the same as the residue of a mod $|b|$. And if $b = 0$ the residue of a mod b is defined to be equal to a. Finally, if m, n are arbitrary integers then we define $\gcd(m, n)$ to be the same as $\gcd(|m|, |n|)$. In particular the gcd is always nonnegative.

Integers m and n are said to be *coprime* or *relatively prime* if $\gcd(a, b) = 1$.

§8 Linear Diophantine equations in two unknowns

A *Diophantine equation* is an equation in which the unknowns are integers. In this section we investigate the simplest kind of Diophantine equation: $as + bt = n$, where a, b and n are given integers and s and t are the unknowns. We have already seen that in the special case $n = \gcd(a, b)$ a solution can be found by means of the extended Euclidean Algorithm; in this section we go a step further, and deal with arbitrary values of n.

We do two examples before stating the general result.

(2.1) ***Example:*** *Find, if possible, integers s and t such that* $171s + 51t = 9$.

Solution: First, apply the extended Euclidean Algorithm to 171 and 51.

$$\begin{array}{cccccc}
171 & 51 & 18 & 15 & 3 & 0 \\
 & & 3 & 2 & 1 & 5 \\
0 & 1 & 3^- & 7 & 10^- & 57 \\
1 & 0 & 1 & 2^- & 3 & 17^-.
\end{array}$$

This tells us that $\gcd(171, 51) = 3$ (the last nonzero number in the top row) and that $171 \times 3 + 51 \times (-10) = 3$. (More generally, if we choose n to be (any) one of the numbers in the top row of the table, then the table gives us a solution of $171s + 51t = n$: put $s = k$ and $t = -h$, or $s = -k$ and $t = h$, where h and k are the numbers below n in the third and fourth rows of the table. The first alternative applies when the superscript minus sign is attached to h, the second alternative applies when it is attached to k. Thus, for example, $171 \times (-2) + 51 \times 7 = 15$ and $171 \times 1 + 51 \times (-3) = 18$.)

The rules used to construct the table are as follows. The first two rows of the table show the steps of the basic Euclidean Algorithm: the successive remainders go in the first row, the quotients go below the remainders in the second row. The third row starts with 0 and 1, the fourth row starts with 1 and 0, and successive numbers in these two rows are calculated in the manner described below.

In general the completed table has the form

$$\begin{array}{ccccccc}
r_{-2} & r_{-1} & r_0 & r_1 & \dots & r_{m-1} & r_m \\
 & & q_0 & q_1 & \dots & q_{m-1} & q_m \\
h_{-2} & h_{-1} & h_0 & h_1 & \dots & h_{m-1} & h_m \\
k_{-2} & k_{-1} & k_0 & k_1 & \dots & k_{m-1} & k_m
\end{array}$$

where r_{-2} and r_{-1} are the numbers a and b whose gcd is to be found (where we assume that $0 \neq a \geq b \geq 0$), and the values of h_{-2}, h_{-1}, k_{-2} and k_{-1} are, respectively, 0, 1, 1 and 0. Now for each natural number i we define q_i and r_i to be the quotient and remainder on dividing r_{i-1} into r_{i-2}, while h_i and k_i are defined by

$$\begin{aligned} h_i &= q_i h_{i-1} + h_{i-2}, \\ k_i &= q_i k_{i-1} + k_{i-2}. \end{aligned}$$

It is a consequence of these rules that $k_i a - h_i b = (-1)^i r_i$, for all i. It is convenient to keep track of the sign $(-1)^i$ by attaching superscript minus signs to the even-numbered h_i and the odd-numbered k_i. The value of m is determined by the condition that $r_m = 0$. (Of course, one has to stop when a zero remainder is obtained: since division by 0 is impossible, one would not be able to continue on to find another quotient and remainder.)

The gcd is r_{m-1}, the second last number in the top row (the last nonzero number in the top row). What is crucial for the purpose of solving an equation of the form $as + bt = n$ is that $d = \gcd(a, b)$ must be a divisor of n. The idea is that the table allows us to write down integers j and l such that $aj + bl = d$, and if $n = df$ then multiplying through by f gives $a(jf) + b(lf) = df = n$. So $s = jf$ and $t = lf$ is a solution of the equation.

Computing only the first two rows of the table, rather than the full table, means performing only the basic Euclidean Algorithm rather than the extended Euclidean Algorithm. This is sufficient to determine whether or not $as + bt = n$ has a solution, but not sufficient to find a solution if one exists. So one might just work through the simple Euclidean Algorithm first, and only do the remaining calculations if it turns out that $d|n$. The practical computational drawback of this approach is that it requires storing (or recalculating) the complete sequence of remainders, whereas if one applies the extended Euclidean Algorithm immediately then one only needs to have stored r_{i-2}, r_{i-1}, q_{i-1}, h_{i-1} and k_{i-1} in order to be able to compute r_i, q_i, h_i and k_i.

In the present example the gcd is 3, which is a divisor of the right hand side of the equation $171s + 51t = 9$. So a solution exists. From the second last column of the table we see that $3 = 171j + 51l$, where $j = k_2 = 3$ (the bottom entry of the column) and $l = -h_2 = -10$ (the second bottom entry of the column, with minus sign attached). The integer $f = n/d$ is $9/3 = 3$. So $s = 3j = 9$ and $t = 3l = -30$ is a solution of $171s + 51t = 9$.

(2.2) ***Example:*** *Find, if possible, integers s and t such that* $182s + 49t = 4$.

Solution: We find $\gcd(182, 49)$.

182	49	35	14	7	0
		3	1	2	2

Since $\gcd(182, 49) = 7$, which does not divide 4, the equation $182s + 49t = 4$ has no solution. Indeed, since $182s + 49t$ is divisible by 7 for all choices of s and t, no choice will make it equal to 4.

(2.3) Proposition: *Suppose that a, $b \in \mathbb{Z}$, and let $d = \gcd(a, b)$. For any given integer n, the equation $as + bt = n$ has a solution in integers s, t if and only if $d|n$.*

Proof. If $d|n$ then a solution can be found by the method explained in Example (2.1) above. On the other hand, suppose that $d\nmid n$. Since $d|a$ and $d|b$ we can write $a = da'$ and $b = db'$ for some integers a', b', and for all $s, t \in \mathbb{Z}$ we have that $as + bt = d(a's + b't)$ is divisible by d. So $as + bt$ can never equal n. □

§9 Congruence notation

Let $n \in \mathbb{Z}$. We say that integers a, b are *congruent modulo n* if $a - b = kn$ for some integer k. We write "$a \equiv b \pmod{n}$" to mean that a is congruent to b modulo n.

In other words,

$$a \equiv b \pmod{n} \quad \text{if and only if} \quad n|(a-b). \tag{3}$$

For example, $12 \equiv (-9) \pmod{7}$ since $12 - (-9) = 21$ is a multiple of 7.

If $n \neq 0$ then $a \equiv b \pmod{n}$ if and only if a and b have the same residue modulo n. Thus, for example, $16 \equiv 51 \pmod{5}$, and both 16 and 51 have residue 1 modulo 5. It is perhaps also worth observing that $a \equiv b \pmod{0}$ if and only if $a = b$.

The congruence notation was invented by Gauss (1777–1855), who was the first person to study number theory systematically. Although it is really a trivial idea, congruence notation is of enormous practical usefulness, basically because of the next theorem.

(2.4) Theorem: *Let $a, a', b, b', m \in \mathbb{Z}$, and suppose that $a \equiv a' \pmod{m}$ and $b \equiv b' \pmod{m}$. Then $a + b \equiv a' + b' \pmod{m}$ and $ab \equiv a'b' \pmod{m}$.*

Proof. Since $m \mid (a - a')$ and $m \mid (b - b')$ we have $a - a' = km$ and $b - b' = hm$ for some integers k and h. Writing these equations as $a = a' + km$ and $b = b' + hm$ we find immediately that

$$a + b = (a' + km) + (b' + hm) = (a' + b') + (k + h)m,$$

so that $(a + b) - (a' + b') = (k + h)m$ is a multiple of m, and similarly

$$ab = (a' + km)(b' + hm) = a'b' + (kb' + a'h + khm)m,$$

so that $ab - a'b' = (kb' + a'h + khm)m$ is a multiple of m. Thus $a + b \equiv a' + b' \pmod{m}$ and $ab \equiv a'b' \pmod{m}$, as required. □

In number theory it is frequently the case that one is not so much interested in the value of some quantity as in its residue with respect to some modulus m. The above theorem shows that the arithmetical operations of addition and multiplication are compatible with reduction modulo m; consequently one often loses nothing by replacing large numbers by their residues modulo m, thereby simplifying arithmetical calculations enormously.

(2.5) Remark: Note that, as a special case of Theorem (2.4), we have that $a \equiv b$ implies $ak \equiv bk$ and $a + k \equiv b + k$, for all integers k.

Our next result is of critical importance in the proof of the so-called "Fundamental Theorem of Arithmetic", which states that every integer greater than 1 can be factorized uniquely as a product of primes.

(2.6) Theorem: *Let a, b, n be integers such that $n \mid ab$. If $\gcd(a, n) = 1$ then it follows that $n \mid b$.*

Proof. Assuming that $\gcd(a, n) = 1$, the proposition in §2 above tells us that there exist $s, t \in \mathbb{Z}$ such that $as + nt = 1$, and multiplying this through by b gives $(ab)s + n(tb) = b$. But we are given that $ab \equiv 0 \pmod{n}$ and it is obvious that $n(tb) \equiv 0 \pmod{n}$; so

$$b = (ab)s + n(tb) \equiv 0s + 0 \equiv 0 \pmod{n}.$$

That is, $n \mid b$, as required. □

(2.7) Corollary: *Let $a, b \in \mathbb{Z}$. If p is a prime such that $p \mid ab$, then $p \mid a$ or $p \mid b$.*

Proof. Suppose that p is a prime such that $p \mid ab$, and let $d = \gcd(a, p)$. Then d is a common divisor of a and p, which means that $d \mid a$ and $d \mid p$. However, since p is a prime the only positive integers that are divisors of p are 1 and p itself. So either $d = 1$ or $d = p$.
Case 1: Suppose that $d = 1$. Then we have that $p \mid ab$ and $\gcd(a, p) = 1$; so Theorem (2.6) tells us that $p \mid b$.
Case 2: Suppose that $d = p$. Since $d \mid a$ (as noted above), this tells us that $p \mid a$.
So either $p \mid b$ (Case 1) or $p \mid a$ (Case 2), as required. □

(2.8) Corollary: *If p is prime and $p \mid (a_1a_2 \cdots a_l)$ for some integers $a_1, a_2, \ldots, a_l$, then $p \mid a_i$ for some $i \in \{1, 2, \ldots, l\}$.*

We remark that the notation $a_1a_2 \cdots a_l$ employed here does not carry the implication that $l \geq 2$. If $l = 1$ then $a_1a_2 \cdots a_l$ should be understood as meaning just a_1. It is even legitimate for l to be zero, in which case $a_1a_2 \cdots a_l$ is an empty product. Empty products are always interpreted as being equal to 1 (as with $3^0 = 1$ and $0! = 1$).*

Proof of Corollary (2.8) (outline only). Since $p > 1$ the hypothesis cannot be satisfied with $l = 0$. If $l = 1$ the hypothesis is just that $p \mid a_1$, and there is nothing to prove. Otherwise we have that $p \mid a_1(a_2 \cdots a_l)$, and Corollary (2.7) tells us that either $p \mid a_1$ or $p \mid a_2 \cdots a_l$. If the former alternative holds we are finished, if the latter holds we repeat the argument with $a_1a_2 \cdots a_l$ replaced by $a_2 \cdots a_l$. Since the number of factors decreases each time, the process must terminate, and this means that $p \mid a_i$ for some i. □

We proceed to outline a proof of the Fundamental Theorem of Arithmetic. You may regard this as a strange thing to do, since perhaps most people regard the Fundamental Theorem as an obvious fact that everybody knows. But whether or not it is really true that everybody knows it, it is certainly not obvious. If challenged to explain it, many people who consider it obvious will find that they do not actually know why it is true.

(2.9) The Fundamental Theorem of Arithmetic: *Let n be an arbitrary positive integer. Then there exist prime numbers $p_1, p_2, \ldots, p_l$ such that $n = p_1p_2 \cdots p_l$. Moreover, if we also have $n = q_1q_2 \cdots q_k$ for some primes $q_1, q_2, \ldots, q_k$ then the numbers k and l are equal, and the q_j's are the same as the p_i's (in some order).*

Proof (outline). The first part, that each positive integer n can be expressed as a product of primes, really is reasonably obvious; the main content of the theorem is that there is essentially only one way to express n as a product of primes.

The number n is allowed to be equal to 1, although this is a rather degenerate case. By the convention on empty products (see above), $1 = p_1p_2 \cdots p_l$ is satisfied if $l = 0$. So 1 can be expressed as a product of primes. Furthermore, since prime numbers are always greater than 1, a nonempty product of primes will always be greater than 1: if $q_1, q_2, \ldots, q_k$ are primes and $k \geq 1$ then $1 < q_1 \leq q_1q_2 \cdots q_k$. So the one and only way to write 1 as a product of primes is given by the empty product. This shows that both assertions of the theorem hold when $n = 1$, and so for the rest of the proof we may assume that $n > 1$.

For the first part, since $n > 1$ either n is prime or else we can factorize n as ml for some integers m, l that are greater than 1 and less than n. If either of the factors m, l is not prime then we can replace it by a product of smaller positive integers; for example, $m = hk$ would give $n = hkl$. We continue like this for as long as possible,

* If n is a natural number then $n!$, called *n factorial*, is the product $1 \times 2 \times 3 \times \cdots \times n$. In the case $n = 0$ this product is empty and therefore takes the value 1.

replacing each non-prime factor by a product of two positive integers smaller than itself. The basic principle that a decreasing sequence of positive integers cannot be infinite means that this process cannot go on forever, and since it only stops when all the factors are prime it follows that n can be expressed as a product of primes, as claimed.

Turning now to the proof of the uniqueness part of the theorem, suppose that $n = p_1p_2\cdots p_l = q_1q_2\cdots q_k$ where the p_i and q_j are prime. Our task is to show that $q_1, q_2, \ldots, q_k$ equal $p_1, p_2, \ldots, p_l$ in some order. Rearranging the order of the factors in both products, and then renumbering the p_i and q_j if necessary, we may assume that $p_1 \le p_2 \le \cdots \le p_l$ and $q_1 \le q_2 \le \cdots \le q_k$. Since $n > 1$ neither product is empty: $l \ge 1$ and $k \ge 1$. So p_1 and q_1 both exist, and $p_1 | n = q_1q_2\cdots q_k$. By Corollary (2.8) it follows that $p_1 | q_j$ for some j. But q_j is prime, and hence is not divisible by any integer greater than 1 apart from itself. Since $p_1 > 1$ we conclude that $p_1 = q_j$. Furthermore, applying this same argument with the roles of the p's and q's interchanged shows that $q_1 = p_i$ for some i. Now we have

$$p_1 \le p_i = q_1 \le q_j = p_1,$$

which shows that $p_1 = q_1$. Thus the factor $p_1 = q_1$ appears on both sides of the equation $p_1p_2\cdots p_l = q_1q_2\cdots q_k$, and cancelling it gives $p_2\cdots p_l = q_2\cdots q_k$. Repeating the argument shows that $p_2 = q_2$, then $p_3 = q_3$, and so on. This cannot continue forever, and only stops when all the p's and q's have been cancelled away to leave the equation $1 = 1$, at which point it will have been shown that $k = l$ and $p_i = q_i$ for all $i \in \{1, 2, \ldots, k\}$. □

To reinforce the point that unique factorization of integers is not a triviality but a serious theorem of number theory, it is worth noting that unique factorization does not hold for many systems of numbers that are otherwise rather similar to the integers. An example will be presented shortly. But first let us contemplate the set of all complex numbers of the form $a + b\sqrt{-1}$, where a and b are integers. Such complex numbers are known as *Gaussian integers*, and they are very much the complex analogue of the ordinary integers. On can define divisibility and factorization for Gaussian integers exactly as for ordinary integers. An ordinary integer n can also be considered as a Gaussian integer, by writing it as $n + 0\sqrt{-1}$. Some prime integers can be factorized nontrivially when regarded as Gaussian integers: for example, 17 can be factorized as $(4 + \sqrt{-1})(4 - \sqrt{-1})$. Others, such as 7, still have no nontrivial factorization when Gaussian integer factors are allowed. (In actual fact the prime 2 and the primes that are congruent to 1 modulo 4 factorize nontrivially as Gaussian integers, whereas the primes that are congruent to 3 modulo 4 do not.) It is an important fact that unique factorization does hold for Gaussian integers, and this can be proved by a reasonably straightforward generalization of the method we have employed to prove unique factorization for $\mathbb{Z}$.

But now consider the set S consisting of all complex numbers of the form $a + b\sqrt{-5}$, where $a, b \in \mathbb{Z}$. The sum of two complex numbers of this form still has this form, and the product of two complex numbers of this form still has this form. It makes perfect sense to talk about divisibility and factorization for numbers in the set S. It is readily proved that every number in S can be expressed as a product of numbers in S that cannot be factorized further in any nontrivial way. But uniqueness of factorization is not valid for S. Specifically, we can factorize 6 as 2×3 and also as $(1 + \sqrt{-5})(1 - \sqrt{-5})$. The numbers 2, 3, $1 + \sqrt{-5}$ and $1 - \sqrt{-5}$ all have no nontrivial factorizations in S; so the analogue of the Fundamental Theorem of Arithmetic simply does not hold for S.

§10 Complete systems and reduced systems

(2.10) Definition: Let m be a positive integer. A set S of integers is called a *complete system modulo m* if for every integer n there is exactly one $s \in S$ such that $n \equiv s \pmod{m}$.

The most obvious example of a complete system modulo m is the set of all natural numbers less than m: the set $\{0, 1, 2, \ldots, m-1\}$. It is also easy to see that any set of m consecutive integers forms a complete system mod m. Thus, for example, $\{-5, -4, -3, -2, -1, 0\}$ and $\{7, 8, 9, 10, 11, 12\}$ are both examples of complete systems mod 6. And one can create a complete system mod m from a given complete system mod m by replacing one of the elements by anything else that is congruent to it mod m. So, for example, $\{7, 8, 9, 64, 11, 12\}$ is a complete system mod 6. A more interesting example, of a kind about which we shall have more to say later, is the set $\{0, 5, 10, 15, 20, 25\}$, which is another complete system mod 6.

(2.11) Definition: Let m be a positive integer. A set S of integers is called a *reduced system modulo m* if all elements of S are coprime to m, and for every integer n coprime to m there is exactly one $s \in S$ such that $n \equiv s \pmod{m}$.

We can obtain a reduced system mod m from any complete system mod m by simply deleting every element s for which $\gcd(m, s) > 1$. For example, consider the set $\{0, 1, 2, 3, 4, 5\}$, a complete system mod 6. We have $\gcd(6, 0) = 6$, $\gcd(6, 2) = \gcd(6, 4) = 2$ and $\gcd(6, 3) = 3$, while $\gcd(6, 1)$ and $\gcd(6, 5)$ are both equal to 1. We conclude that $\{1, 5\}$ is a reduced system mod 6. Of course we can create other reduced systems mod 6 by replacing 1 by anything to which it is congruent mod 6, and 5 by anything to which it is congruent mod 6. For example, $\{17, 19\}$ is a reduced system mod 6.

The most obvious example of a reduced system mod m is the set of all natural numbers that are less than m and coprime to m. For example, $\{1, 5, 7, 11\}$ is a reduced system mod 12, and $\{1, 3, 5, 7\}$ is a reduced system mod 8. Note that when $m = 1$ the complete system $\{0, 1, 2, \ldots, m-1\}$ is just the set $\{0\}$, and this is also a reduced system since $\gcd(0, 1) = 1$. (In no other case does 0 belong to a reduced system mod m, since $\gcd(0, m) = m \neq 1$ if m is any positive integer other than 1.)

It is clear that all complete systems mod m must have precisely m elements. Indeed, if S_1 and S_2 are both complete systems mod m then each element of S_1 is congruent mod m to exactly one element of S_2 (since S_2 is a complete system), and every element of S_2 is congruent mod m to exactly one element of S_1 (since S_1 is a complete system). Thus matching each element of S_1 with the elemeent of S_2 to which it is congruent yields a one to one correspondence between the two sets. So S_1 and S_2 have the same number of elements. And, in particular, all complete systems have the same number of elements as the complete system $\{0, 1, 2, \ldots, m-1\}$, that is, m elements.

Exactly the same kind of reasoning shows that any two reduced systems modulo m have the same number of elements, but in this case it is not at all clear whether or not there is any simple formula for this number. We define $\phi(m)$ to be the number of elements in any reduced system mod m. This function ϕ is known as *Euler's phi function*.* An alternative way to state the definition of $\phi(m)$ is as follows.

(2.12) Definition: Let m be a positive integer. The quantity $\phi(m)$ is the number of natural numbers k such that $k < m$ and $\gcd(m, k) = 1$.

From the examples of reduced systems given above we see that $\phi(6) = 2$ and $\phi(8) = \phi(12) = 4$. Note also that $\phi(1) = 1$. It is easily seen that if p is a prime then $\{1, 2, \ldots, p-1\}$ is a reduced system mod p (since $\{0, 1, 2, \ldots, p-1\}$ is a complete system, $\gcd(i, p) = 1$ for all nonzero i in this set, and $\gcd(0, p) = p \neq 1$). Thus $\phi(p) = p - 1$ whenever p is prime. We also see that $\phi(27) = 27 - 9 = 18$, since there are exactly $27/3 = 9$ natural numbers less than 27 that are multiples of 3, and the numbers that are coprime to 27 are exactly the numbers that are not multiples of 3. More generally, if p is a prime number and k a positive integer then the numbers less than p^k that are coprime to p^k are exactly the numbers less than p^k that are not multiples of p. There are $p^k/p = p^{k-1}$ natural numbers less than p^k that are multiples of p, and so it follows that $\phi(p^k) = p^k - p^{k-1}$.

§11 Cancellation in congruences

We observed above (see Remark (2.5)) that for all integers a, b, m and k, if $a \equiv b \pmod{m}$ then $ak \equiv bk \pmod{m}$. It is natural to ask whether or not the converse statement holds. If $ak \equiv bk \pmod{m}$, does it follow that $a \equiv b \pmod{m}$?

The relevance of the next example to the above question may not be immediately apparent, but it prepares the way for the subsequent proposition, from which the main result of this section follows as an immediate corollary.

(2.13) ***Example:*** *Find an integer k such that $51k \equiv 1$ (mod 245).*

Solution: We apply the extended Euclidean Algorithm.

$$\begin{array}{cccccc}
245 & 51 & 41 & 10 & 1 & 0 \\
 & & 4 & 1 & 4 & 10 \\
0 & 1 & 4^- & 5 & 24^- & 245 \\
1 & 0 & 1 & 1^- & 5 & 51^-
\end{array}$$

* "Euler" is pronounced "Oiler".

It follows from this that $5 \times 245 - 24 \times 51 = 1$, and hence $51 \times (-24) \equiv 1 \pmod{245}$. That is, $k = -24$ is a solution.

Notice that the solution of a problem like this is never unique. If $k_1 \equiv k_2 \pmod{245}$ then certainly $51k_1 \equiv 51k_2 \pmod{245}$. So any integer k that is congruent to -24 (modulo 245) will also be a solution of $51k \equiv 1 \pmod{245}$. For example, $k = 245 + (-24) = 221$ is another solution.

Clearly in Example (2.13) it is crucially important that $\gcd(245, 51) = 1$. And, moreover, the same method would work if 245 and 51 were replaced by any two coprime integers.Hence we obtain the following important result.

(2.14) Proposition: *Let m and n be integers satisfying $\gcd(m,n) = 1$. Then there exists an integer k such that $nk \equiv 1 \pmod{m}$.*

Proof. By the extended Euclidean Algorithm there exist integers k and l satisfying $nk + ml = \gcd(m,n) = 1$. Then $nk - 1 = ml$ is a multiple of m, which by definition says that $nk \equiv 1 \pmod{m}$, as required. □

(2.15) Corollary: *Suppose that m, $n \in \mathbb{Z}$ satisfy $\gcd(m,n) = 1$, and suppose also that a, $b \in \mathbb{Z}$ are such that $an \equiv bn \pmod{m}$. Then $a \equiv b \pmod{m}$.*

Proof. Since $\gcd(m,n) = 1$ the proposition tells us that there exists an integer k such that $nk \equiv 1 \pmod{m}$. Multiplying by k on both sides of the congruence $an \equiv bn \pmod{m}$, we deduce that $(an)k \equiv (bn)k \pmod{m}$, and so

$$a \equiv a1 \equiv a(nk) = (an)k \equiv (bn)k = b(nk) \equiv b1 \equiv b \pmod{m},$$

as required. □

We had already seen that congruence behaves like equality, to some extent, in that it is legitimate to deduce $ak \equiv bk$ from $a \equiv b$. The above corollary shows that it is also legitimate to cancel, but only if the number being cancelled is coprime to the modulus. We refer to this fact as the *coprime cancellation rule*.

If the coprimeness condition is not satisfied, cancellation is not necessarily valid. For example $45 \equiv 81 \pmod{12}$, but it is certainly not legitimate to cancel 9 from both sides of this congruence, since doing so would give $5 \equiv 9 \pmod{12}$, which is false. The problem is that the gcd of 9 and 12 is not 1, but 3.

In general, if $\gcd(m,n) = d \neq 0$ then $m = dm'$ and $n = dn'$ for some integers m' and n' satisfying $\gcd(m',n') = 1$. Now if $an \equiv bn \pmod{m}$ then we have $an - bn = km$ for some integer k, and this gives $(a-b)n'd = km'd$. Since $d \neq 0$ (by hypothesis) we can cancel d from this equation and deduce that $an' - bn' = km'$, whence $an' \equiv bn' \pmod{m'}$. Now we can apply Corollary (2.15), since $\gcd(m',n') = 1$, and conclude that $a \equiv b \pmod{m'}$. In our numerical example above, the congruence $45 \equiv 81 \pmod{12}$ legitimately yields $15 \equiv 27 \pmod{4}$, after which, since $\gcd(3,4) = 1$, we can cancel 3 and obtain $5 \equiv 9 \pmod{4}$.

(2.16) ***Example:*** The aim of this example is to learn what we can about the sequence of numbers $a_0, a_1, a_2, \ldots$ defined by the rule that a_i is the residue of 2^i modulo 41. We start by finding the first few terms.

The key observation here is that since $2^{i-1} \equiv a_{i-1} \pmod{41}$ it follows that $2^i = 2 \times 2^{i-1} \equiv 2a_{i-1} \pmod{41}$. This means that if we compute the a_i recursively, we never have to work with big numbers. Starting with $a_0 = 2^0 = 1$, we compute each successive a_i by doubling the previous one and reducing modulo 41. We quickly build up the following table.

2^i:	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	$\ldots$
a_i:	1	2	4	8	16	32	23	5	10	20	40	39	37	33	25	9	18	36	$\ldots$

Before pursuing this calculation any further, note that since all the numbers in the second row are less than 41, it will take no more than 41 steps for there to be at least one repetition. We are bound to find, for some $j < 41$, that $a_j = a_i$ for some $i < j$. And once the first repetition is encountered, the sequence will start repeating itself. For since a_{j+1} is the residue of $2a_j \pmod{41}$ and a_{i+1} is the residue of $2a_i \pmod{41}$, if $a_j = a_i$ then follows that $a_{j+1} = a_{i+1}$. And so it continues: from $a_{j+1} = a_{i+1}$ it follows that $a_{j+2} = a_{i+2}$, from which it follows that $a_{j+3} = a_{i+3}$, and so on. But the implication works in the other direction too: the only way that a_{j+1} can equal a_{i+1} is if $a_j = a_i$. For if $a_{j+1} = a_{i+1}$ then $2a_j \equiv 2a_i \pmod{41}$, and since $\gcd(2, 41) = 1$ it follows by coprime cancellation (Corollary (2.15)) that $a_j \equiv a_i \pmod{41}$. Since a_j and a_i are both natural numbers less than 41, this forces $a_j = a_i$. In the same way, if a_j is equal to a_i then a_{j-1} must have been equal to a_{i-1}. This would say that it is impossible for there to be any first repetition, were it not for the fact that there is one term in the sequence that is not preceded by any other term. There is nothing that precedes a_0. So if $a_j = a_i$ and it happens that $i = 0$ then it does not in fact follow that $a_{j-1} = a_{i-1}$, because $a_{i-1} = a_{-1}$ is undefined. Hence if j is the smallest positive integer such that $a_j = a_i$ for some $i < j$, the i in question must in fact be 0.

We did not take the calculations far enough to encounter any repetitions in the second row of the table, but from the reasoning above we know that the first repetition will be $a_j = a_0 = 1$, where the number j is at most 40. (Because we started at $j = 0$, the 41st term a_j corresponds to $j = 40$.) So far we have only taken the calculations up to $j = 17$; so let us push on with the calculations, at least as far as is necessary to confirm the truth of the claim just made. At most 23 more steps will be needed!

2^i:	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	$\ldots$
a_i:	36	31	21	1	2	4	$\ldots$

The calculations have shown that $2^{20} \equiv 2^0 \pmod{41}$, and, after that, $2^{20+j} \equiv 2^{0+j}$ for all positive integers j. We can express this by saying that the sequence $(a_i)_{i=0}^{\infty}$ is periodic with period 20. In other words, $a_i = a_j$ whenever $i \equiv j \pmod{20}$. This means that $2^i \equiv 2^j \pmod{41}$ whenever $i \equiv j \pmod{20}$.

In Example (2.16) we found that 20 is the least positive integer j such that $2^j \equiv 1 \pmod{41}$. This number is called the *order of 2 modulo 41*, written $\mathrm{ord}_{41}(2)$.

(2.17) Definition: Let p be a prime number and a any integer that is not a multiple of p. The *order of a modulo p*, written $\mathrm{ord}_p(a)$, is the least positive integer j such that $a^j \equiv 1 \pmod{p}$.

The reasoning applied in our numerical example above applies equally well when 41 is replaced by any prime number p and 2 by any positive integer a coprime to p. The sequence of mod p residues of the successive powers of a must be periodic: the first value repeated must be $1 = a^0$, and the period is $j = \mathrm{ord}_p(a)$. Moreover, this period is at most $p - 1$.

(2.18) ***Example:*** *Define a_i to be the residue of 6^i modulo 23. Find $\mathrm{ord}_{23}\,6$, and hence determine the value of a_{11115}.*

Solution: We start with $a_0 = 1$ and calculate successive terms by multiplying by 6 and reducing modulo 23. The following table gives the values of a_i up to $i = 13$.

6^i:	6^0	6^1	6^2	6^3	6^4	6^5	6^6	6^7	6^8	6^9	6^{10}	6^{11}	6^{12}	6^{13}	...
a_i:	1	6	13	9	8	2	12	3	18	16	4	1	6	13	...

The calculations have shown that $\mathrm{ord}_{23}(6) = 11$. Thus the sequence of a_i's is periodic with period 11; that is, $a_i = a_j$ whenever $i \equiv j \pmod{11}$. In particular, since $11115 \equiv 5 \pmod{11}$, it follows that $a_{11115} = a_5 = 2$. (Consequently, $6^{11115} \equiv 6^5 \equiv 2 \pmod{23}$. If one were to compute 6^{11115} and divide it by 23, the remainder would be 2.)

Week 3

PART 3A: *Some classical cryptography*

Our main objective in MATH2088/2988 is to understand some modern public key cryptosystems and the underlying mathematics, but to gain a proper appreciation of these modern cryptosystems it is necessary to learn some general things about cryptography. Accordingly we shall devote a little time to studying some of the cryptographic techniques that have been used over the centuries.

Our investigation of classical cryptography will be limited and will mainly take place in the computer tutorials, but we shall also devote a few lectures to the topic. This will serve to reinforce the work done in the computer tutorials, as well as to slow down the pace of the number theory.

§12 Terminology

We use the term *alphabet* to mean a set of symbols (of any kind). The elements of this set are called the *letters* of the alphabet. Most of the time we shall use the alphabet $\{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}, \mathsf{F}, \mathsf{G}, \mathsf{H}, \mathsf{I}, \mathsf{J}, \mathsf{K}, \mathsf{L}, \mathsf{M}, \mathsf{N}, \mathsf{O}, \mathsf{P}, \mathsf{Q}, \mathsf{R}, \mathsf{S}, \mathsf{T}, \mathsf{U}, \mathsf{V}, \mathsf{W}, \mathsf{X}, \mathsf{Y}, \mathsf{Z}\}$. We shall frequently use the numbers $\mathsf{0}, \mathsf{1}, \mathsf{2}, \mathsf{3}, \mathsf{4}, \mathsf{5}, \mathsf{6}, \mathsf{7}, \mathsf{8}, \mathsf{9}, \mathsf{10}, \mathsf{11}, \mathsf{12}, \mathsf{13}, \mathsf{14}, \mathsf{15}, \mathsf{16}, \mathsf{17}, \mathsf{18}, \mathsf{19}, \mathsf{20}, \mathsf{21}, \mathsf{22}, \mathsf{23}, \mathsf{24}, \mathsf{25}$ to represent these 26 letters, since this will make calculation easier. Another important alphabet is the two-letter alphabet $\{\mathsf{0}, \mathsf{1}\}$. We sometimes consider alphabets whose "letters" are composite symbols, such as $\mathsf{AB}, \mathsf{AC}, \ldots, \mathsf{ZZ}$, or $\mathsf{000}, \mathsf{001}, \ldots, \mathsf{111}$.

A *message* is a sequence of letters from some alphabet. We imagine that someone wishes to convey a message to someone else, in such a way that no one but the intended recipient will be able to understand it. This is achieved by an encryption process that replaces the original message (the *plaintext*) by another message (the *ciphertext*), which is the one that is actually sent, combined with a decryption process that the recipient should use to recover the plaintext from the ciphertext. Typically the encryption and decryption processes are very similar; indeed, they may even be identical.

In mathematical terminology, encryption and decryption are functions from a set of messages to another set of messages. Often the messages in these two sets will use the same alphabet, but this need not be the case. We may use the term *source alphabet* for the alphabet of the input messages, and *target alphabet* for the alphabet of the output messages.

There are, in fact, just two basic kinds of encryption/decryption processes that people have devised:

1) *transposition ciphers*:— the output message is obtained by rearranging the letters of the input message according to some rule, so that if the input message is $x_1x_2\ldots x_l$ then the output message is $x_{\pi(1)}x_{\pi(2)}\ldots x_{\pi(l)}$, where the numbers $\pi(1),\pi(2),\ldots,\pi(l)$ are just the numbers $1,2,\ldots,l$ in some other order;
2) *substitution ciphers*:— a function f from the source alphabet to the target alphabet is applied to all of the letters of the input message in turn, so that if the input message is $x_1x_2\ldots x_l$ then the output message is $f(x_1)f(x_2)\ldots f(x_l)$;

Could there be some other, fundamentally different, encryption process? Personally, I cannot imagine one. Of course it is possible to complicate things by applying combinations of several transposition and substitution ciphers, and by applying different ciphers to different parts of the message. But, at the basic level, transposition ciphers and substitution ciphers are the only two kinds.*

§13 Transposition ciphers

Most transposition ciphers involve entering the letters of the input message into some geometrical configuration, and then extracting them in a different order to obtain the output message. One of the simplest is to write the input message along the rows of a rectangular array, and obtain the output message by reading down the columns. Here is an example using eight columns and a plaintext message that is probably familiar.

```
A U S T R A L I
A N S A L L L E
T U S R E J O I
C E F O R W E A
R E Y O U N G A
N D F R E E
```

The ciphertext, split into blocks of length 5 for convenience, would be AATCR NUNUE EDSSS FYFTA ROORR LERUE ALJWN ELLOE GIEIA A. Decryption is achieved by using the total length of the message to work out the length of the short last row and then entering the cipher text into the columns. One could make things more complicated by taking the columns in some other order, or by using diagonal lines instead of columns. For example, using the array above we could start at the bottom left corner and work towards the top right corner, going back and forth along diagonals: NRDFE CTEYR EOFUA ANSOU ENRRS USAEW GAEJL TRLOA ILALE I.

* At least, I think that they are the only two kinds. It is perhaps conceivable that the world's intelligence agencies have devised other encryption/decryption processes that they are keeping secret.

Another system involves splitting the message into blocks of some fixed size m and rearranging the letters within each block according to some permutation of $\{1, 2, \ldots, m\}$. We call this a *block transposition cipher*; the permutation is called the *key*. For example, using a block length of six and the key 634125—which means that the 6th letter of the input message becomes the first letter of the output, the third of the input becomes the second, and so on—the message ENCRYPTIONANDDECRYPTIONSHOULDBEEASYIFYOUKNOWHOW is encrypted as PCRENYNONTIAYECDDRSIOPTNBULHODIASEEYNOUFYKHOOWW. The table below shows the plaintext split into blocks of length 6, with the corresponding blocks of ciphertext underneath.

ENCRYP	TIONAN	DDECRY	PTIONS	HOULDB	EEASYI	FYOUKN	OWHOW*
PCRENY	NONTIA	YECDDR	SIOPTN	BULHOD	IASEEY	NOUFYK	*HOOWW

Note that since the length of the plaintext in the above example was not a multiple of the block length, a dummy letter was added to the end when performing the encryption. However, the dummy letter is not explicitly included in the ciphertext. The total length of the message tells the recipient how many dummy letters there were, and the key determines where they were in the ciphertext.

§14 Simple substitution ciphers

The term "simple substitution cipher" means exactly what is described in item 2) above. A particularly simple example is the *Caesar cipher*, so-called because Julius Caesar used it. As foreshadowed above, let us represent the letters A – Z by the numbers 0 – 25. For ease of reference, we display the correspondence in a table.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Caesar's cipher consists of replacing i by $i + 3$, reduced modulo 26 if $i + 3 > 26$. Thus, for example, the message PREPARETOMEETTHYDOOM, which, represented numerically is

15 17 4 15 0 17 4 19 14 12 4 4 19 19 7 24 3 14 14 12

is encrypted as

18 20 7 18 3 20 7 22 17 15 7 7 22 22 10 1 6 17 17 15

or SUHSDUHWRPHHWWKAGRRP (which would scare anyone who intercepted it).

Of course, one could use any number n in place of 3. A *translation cipher* is a substitution cipher for which each i is replaced (in all its occurrences) by $i + n$ (reduced mod 26). The fixed number n is called the *key*. Translation ciphers must be just about the least secure ciphers imaginable. There are only 26 possible keys, and an enemy who intercepted the message could, for example, just try them all. Such a method of attack is known as an *exhaustive key search*. Of course the attacker needs to know what method of encryption was used before it becomes possible to embark on an exhaustive key search, but even so one would surely prefer to use a system for which the number of possible keys is so enormous that an exhaustive key search is never feasible.

For an arbitrary simple substitution cipher, the key is a permutation of the letters A to Z. That is to say, the key is a bijective function f from the set $\{\mathsf{A},\mathsf{B},\ldots,\mathsf{Z}\}$ to itself. There are 26! possibilities, since $f(\mathsf{A})$ can be chosen in 26 ways, and after that there are 25 choices left for $f(\mathsf{B})$, and then 24 for $f(\mathsf{C})$, and so on. Now 26! is a seriously huge number: 403291461126605635584000000, in fact. If an attacker embarked on an exhaustive key search, and could test each possible key in one nanosecond, it would take 12779535235 years, or thereabouts, to do them all. (Of course this does not mean that using a simple substitution with a randomly chosen key is a secure system.)

One practical problem with randomly chosen substitution keys is that the user of the system cannot be expected to remember a random permutation of 26 letters (or several, since presumably the key should be changed from time to time). One method for determining keys was to choose some phrase or piece of text that contains all the letters, and use the first letter of this to represent A, the second to represent B, and so on, ignoring letters that have already been used. Thus the phrase "the quick brown fox jumps over the lazy dog" would yield the key

$$\begin{pmatrix} \mathsf{A} & \mathsf{B} & \mathsf{C} & \mathsf{D} & \mathsf{E} & \mathsf{F} & \mathsf{G} & \mathsf{H} & \mathsf{I} & \mathsf{J} & \mathsf{K} & \mathsf{L} & \mathsf{M} & \mathsf{N} & \mathsf{O} & \mathsf{P} & \mathsf{Q} & \mathsf{R} & \mathsf{S} & \mathsf{T} & \mathsf{U} & \mathsf{V} & \mathsf{W} & \mathsf{X} & \mathsf{Y} & \mathsf{Z} \\ \mathsf{T} & \mathsf{H} & \mathsf{E} & \mathsf{Q} & \mathsf{U} & \mathsf{I} & \mathsf{C} & \mathsf{K} & \mathsf{B} & \mathsf{R} & \mathsf{O} & \mathsf{W} & \mathsf{N} & \mathsf{F} & \mathsf{X} & \mathsf{J} & \mathsf{M} & \mathsf{P} & \mathsf{S} & \mathsf{V} & \mathsf{L} & \mathsf{A} & \mathsf{Z} & \mathsf{Y} & \mathsf{D} & \mathsf{G} \end{pmatrix},$$

or, in a shorter notation, THEQUICKBROWNFXJMPSVLAZYDG.

The most obvious weakness of simple substitution ciphers is that one can make a good guess at which ciphertext letters represent which plaintext letters by examining the frequencies with which the various letters occur in the ciphertext. In any piece of English text it is almost guaranteed that E will be the most frequently occurring letter. For a translation cipher, identifying E determines the key.

The most common letters in English text are E ($\approx$ 12%), T ($\approx$ 9%), A ($\approx$ 8.5%), O, N, I, S, R, H (all about 6% to 7.5%), followed by D ($\approx$ 5%) and L ($\approx$ 4%). Together these account for 75% of the letters in most pieces of text. It is probably easy to identify E, T and A, and the next six most frequently occurring letters in the ciphertext are likely to represent O, N, I, S, R, H in some order. Now although 26! is huge, the same cannot be said of 6!, which is a mere 720. One can easily try all possible permutations of O, N, I, S, R, H, and it is almost guaranteed that one of them will give you something you can instantly recognize as correct.

One can streamline this process considerably by making use of commonly occurring sequences of letters, such as THE and TION. The most common digraphs (pairs of letters) are TH ($\approx$ 3.3%), HE ($\approx$ 2.9%), ER ($\approx$ 1.6%), ED ($\approx$ 1.5%), AN, ND ($\approx$ 1.4%), AR, RE, EN ($\approx$ 1.3%). Then come ES, TO, NT, EA, OU, NG, ST, AS, RO, AT.

One way to slightly improve on translation ciphers is to change the encryption rule from $i \mapsto i + n$ to $i \mapsto mi + n$ (reduced mod 26), where m and n are fixed. Such ciphers are called *affine ciphers*. The key is the pair of numbers (m, n), both of which can be taken to be residues modulo 26. Choosing the multiplier m to be coprime to 26 ensures that $i \mapsto \mathrm{res}_{26}(mi+n)$ yields a permutation of the numbers 0 to 25; this follows easily from the coprime cancellation property we proved last week.

(3.1) Proposition: *Suppose that m, $n \in \mathbb{Z}$, and suppose that* $\gcd(m, 26) = 1$. *Then as i runs from 0 to 25, the numbers $mi + n$ run through a complete system modulo 26, and the mod 26 residues of these numbers are the numbers 0 to 25 in some order.*

Proof. The two assertions—that the numbers comprise a complete system and that their residues are 0 to 25 in some order—are equivalent statements. Since the 26 numbers $0, 1, \ldots, 25$ are the only possible values for a residue modulo 26, if we can show that no two of the 26 numbers $mi + n$ have the same residue then it will follow that each possible residue occurs exactly once, and the proposition will be proved.

So suppose that i_1, $i_2 \in \{0, 1, \ldots, 25\}$, and suppose that $mi_1 + n$ and $mi_2 + n$ have the same residue modulo 26. Then

$$mi_1 + n \equiv mi_2 + n \pmod{26},$$

adding $-n$ to both sides (using the fact that addition preserves congruences) gives

$$mi_1 \equiv mi_2 \pmod{26},$$

and since $\gcd(m, 26) = 1$ the coprime cancellation principle yields

$$i_1 \equiv i_2 \pmod{26}.$$

So $26 \mid (i_1 - i_2)$. But the fact that i_1, $i_2 \in \{0, 1, \ldots, 25\}$ ensures that $-26 < i_1 - i_2 < 26$, and since the only multiple of 26 in this range is 0 it follows that $i_1 = i_2$. Hence distinct values of i in $\{0, 1, \ldots, 25\}$ give distinct residues for $mi + n$, as required. □

The following table illustrates the proposition in the case $m = 3$ and $n = 5$.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$\mathrm{res}_{26}(3i+5)$	5	8	11	14	17	20	23	0	3	6	9	12	15	18	21	24	1	4	7	10	13	16	19	22	25	2

The reader should check that the second row contains everything exactly once.

Frequency analysis cracks an affine cipher almost as easily as a translation cipher, since once you know E and T you know everything.

(3.2) ***Example:*** *A long intercepted message is known to have been encrypted using an affine cipher, and it is found that P is the most frequently occurring letter, at about 12% of the total, and J is next, at about 10%. What is the most likely key?*

Solution: Letter 15 (P) probably represents letter 4 (E) and letter 9 (J) probably represents letter 19 (T). So we need to find numbers m and n such that $4m + n \equiv 15 \pmod{26}$ and $19m + n \equiv 9 \pmod{26}$. Subtracting the first congruence from the second gives $15m \equiv -6 \pmod{26}$, and then cancelling 3 and multiplying by 5 gives $25m \equiv -10 \pmod{26}$. But $25 \equiv -1 \pmod{26}$; so $m \equiv 10 \pmod{26}$. Hence $n \equiv 15 - 4m \equiv 15 - 40 \equiv 1 \pmod{26}$. So, assuming that E and T have been identified correctly, the key is $(m, n) = (10, 1)$.

---

§15 Homophonic substitution ciphers

One way to try to beat frequency analysis is to use a ciphertext alphabet of more than 26 letters, using more than one ciphertext letter for common letters of the plaintext. When encrypting a letter for which more than one ciphertext letter is available, one makes a random choice. If the ciphertext alphabet consists of pairs of letters A – Z then it may be hard for the enemy cryptanalyst to detect that a homophonic system has been used.

For example, representing A – Z as 0 – 25, one might encrypt i by randomly choosing any one of the 26 pairs jk for which $k - j \equiv i \pmod{26}$. Thus the possibilities for A would be AA, BB, CC, ..., ZZ, and the possibilities for D would be AD, BE, CF, ..., ZC. The word DAD could be encrypted in any of 26^3 possible ways, including, for example, ADAAAD, BEEEEH and FIOOYB.

Of course, such a method increases the length of the message.

§16 Polyalphabetic substitution ciphers

The word "polyalphabetic" basically means "many alphabets", but its use in this context is based on a usage of the word "alphabet" that is different from the one we have adopted. Some people use "alphabet" to mean essentially the same thing as "substitution key", namely a permutation of the letters A – Z. Polyalphabetic systems employ m different substitution keys, for some number m, and the encryption process applies the first substitution key to the first letter of the plaintext, the second key to the second, and so on. For the $(m + 1)$-st letter of the plaintext one returns to the first key, and so on. Thus for the i-th letter one uses the j-th key if and only if $j \equiv i \pmod{m}$.[*]

The well known Vigenère cipher is a polyalphabetic translation cipher: that is, the m substitutions are all translations. For a Vigenère cipher, the key is the m-letter sequence made up of the images of the letter A under the m translations. We give an example of encryption using the key MATHS, which, numerically, is 12 0 19 7 18.

S	O	M	E	W	H	E	R	E	O	V	E	R	T	H	E	R	A	I	N	B	O	W
18	14	12	4	22	7	4	17	4	14	21	4	17	19	7	4	17	0	8	13	1	14	22
12	0	19	7	18	12	0	19	7	18	12	0	19	7	18	12	0	19	7	18	12	0	19
4	14	5	11	14	19	4	5	11	6	7	4	10	0	25	16	17	19	15	5	13	14	15
E	O	F	L	O	T	E	F	L	G	H	E	K	A	Z	Q	R	T	P	F	N	O	P

§17 Polygram substitution ciphers

These are cryptosystems in which blocks of letters are substituted in groups. So instead of being a permutation of the 26 letters A – Z the key might be a permutation

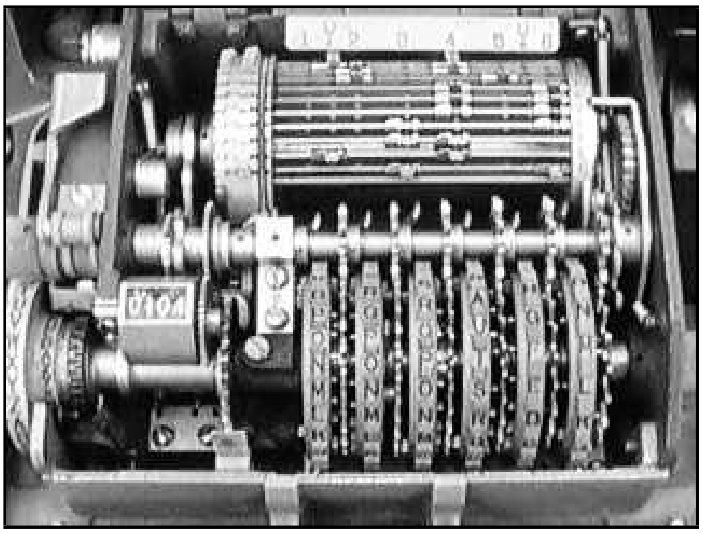
[*] In the same spirit, simple substitution ciphers are traditionally called "monoalphabetic ciphers".

of the 26^2 pairs AA – ZZ (if one uses blocks of length 2), or the 26^3 triples AAA – ZZZ (if one uses triples), and so on. All such strategems make frequency analysis more difficult.

§18 The Hagelin machine

In order to be able to quickly encipher and decipher long messages, it is considerably advantageous to have a machine that can do the job. These days, of course, this means a computer. But in the first half of the 20th century people invented mechanical and mechanical/electrical contraptions for the task. These would resemble typewriters. The operator would key in the plaintext and out would come the ciphertext, or vice versa.

One such machine was invented in the 1930's by a Swedish engineer named Boris Hagelin. It was used by the U.S. armed forces up to about 1950. A picture of the workings of one is shown; there are many other such pictures on the internet.



The machine has 27 rods mounted on a rotatable drum (in the top part of the picture), and each rod has two movable lugs. The lugs can be moved to inactive positions, where they do not touch anything as the drum rotates, or to active positions. On each rod there are six active lug positions; these positions correspond to the positions of the six rotors that can be seen in the bottom half of the picture, alongside the wheels with letters on them. The letters on the wheels serve to identify pins on the rotors. You can see in the picture that the letters are closer together on the leftmost wheel than the rightmost; this is because the leftmost rotor has more pins than the rightmost. In fact, the rotors have 26, 25, 23, 21, 19 and 17 pins, respectively. Each pin can be set either in a raised position or a lowered position. The pins are set and the lugs positioned before encryption begins; indeed, the pin settings and the lug positions together constitute the key for the encryption process.

When the operator types a letter on the keyboard, the drum rotates through one revolution, the rotors remaining fixed. As the drum rotates, the rods all move past the row of rotor pins that are closest to the drum, and if one of these pins is raised it will contact any lug that happens to be positioned above the rotor in question. If this happens then the rod on which the lug is mounted (or something attached to it) will engage a gear wheel that will rotate one position. So as the drum performs one revolution the gear wheel rotates through n positions, where n is the number of rods for which either lug (or both) contacts a raised pin. The effect is that if the operator keyed in letter number i, then the machine types letter number $i' = n - i$ (or the residue of this modulo 26). After each letter is typed, all the rotors move around one place, so that a different collection of pins are now adjacent to the drum, and the value of n for the next letter will be different. Since the lowest common multiple of 26, 25, 23, 21, 19 and 17 is 101405850, the rotors will not return to their original positions until this many letters have been typed.

The recipient of the message must have a Hagelin machine initialized with the same settings as the sender's machine. The operator simply keys in the ciphertext, and since the machine generates the same sequence of values of n as the sender's machine generated, when the operator keys in letter number i', the machine types out letter number $i = n - i'$, exactly the plaintext letter corresponding to the ciphertext i'.

1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	1	0	0	0	0
4	0	0	1	0	0	0
5	1	0	0	0	0	1
6	1	0	0	0	0	0
7	1	0	0	0	0	0
8	1	0	0	0	0	0
9	0	1	0	1	0	0
10	0	1	0	1	0	0
11	0	0	1	1	0	0
12	0	0	0	0	0	1
13	0	0	0	0	0	1
14	0	0	0	0	1	0
15	0	0	1	0	0	0
16	0	1	0	0	0	1
17	1	0	0	0	0	1
18	1	0	0	0	1	0
19	0	1	0	0	1	0
20	0	0	1	0	1	0
21	0	0	0	1	0	0
22	0	0	1	0	0	1
23	1	0	0	0	0	0
24	1	0	0	0	0	0
25	0	1	0	0	0	0
26	0	0	0	1	0	0
27	0	0	1	0	0	1

1	0	0	0	0	1	1
2	0	0	0	0	0	0
3	0	1	0	0	1	1
4	0	0	1	0	0	0
5	1	0	1	1	1	1
6	1	0	1	1	1	0
7	1	0	1	1	1	0
8	1	0	1	1	1	0
9	0	1	1	1	1	0
10	1	1	0	1	0	0
11	1	0	1	1	0	0
12	1	0	0	0	0	1
13	1	1	1	1	0	1
14	1	1	1	1	1	0
15	1	0	1	0	0	1
16	1	1	0	0	1	1
17	1	0	0	0	0	1
18	1	0	0	1	1	
19	0	1	1	0	1	
20	0	0	1	0		
21	0	1	1	1		
22	0	0	1			
23	1	0	0			
24	1	0				
25	1	1				
26	1					

The two tables above combine to show an imaginary Hagelin machine configuration. The table on the left has one row for each rod, and the positions on the rods and above the six rotors are marked with a 1 or a 0 to indicate either the presence or absence of an active lug, each rod having zero, one or two active lugs. The table on the right shows which pins are raised (indicated by a 1) or lowered (indicated by a 0) on the six rotors. The top row of the right hand table corresponds to the rotor pins that are initially next to the drum.

Now imagine that the operator enters the word SYDNEY. When the S is typed the two raised pins (on the fifth and sixth rotors) will contact rods 5, 12, 13, 14, 16, 17, 18, 19, 20, 22 and 27, so that $n = 11$. The value of i is 18 for S, giving $i' = 19$, since $11 - 18 = -7 \equiv 19 \pmod{26}$. So the first letter of the ciphertext is T. For the encryption of the next letter, Y, we move to the second row of the rotor table, in which there are no raised pins. So the value of n will be 0, and $i = 24$ (for Y) will give $i' = 2$, since $0 - 24 \equiv 2 \pmod{26}$. So the second letter of the ciphertext will be C. For the third letter we move to the third row of the rotor table, which is the same as the first but with a raised pin on the second rotor. This will contact lugs on rods 3, 9, 10, 16, 19 and 25, so that contact will be made with 3, 9, 10 and 25 in addition to those listed above for the first letter. So $n = 15$ and $i = 3$ (for D), giving $i' = 15 - 3 = 12$: the letter M. For the next letter the only raised pin is on the third rotor, and it contacts six rods. We have $i = 13$ (N) and $i' \equiv 6 - 13 \pmod{26}$; so $i' = 19$ and the letter is T. For the next letter we get contact with 23 rods (all except 1, 2, 3 and 25); so $n = 23$ and the letter $\mathsf{E} = 4$ is encrypted as $\mathsf{T} = 23 - 4$. For the final letter we get contact with 20 rods—all except 1, 2, 3, 12, 13, 16 and 25—and so the letter $\mathsf{Y} = 24$ is encrypted as $\mathsf{W} = 22$ (since $22 \equiv 20 - 24 \pmod{26}$). The ciphertext is thus TCMTTW.

Part 3B: *Back to number theory*

§19 More on complete systems and reduced systems

Let a, b and c be integers. It is trivial that if $a|b$ and $b|c$ then $a|c$, since $b|c$ gives $c = kb$ for some integer k, and $a|b$ gives $b = la$ for some integer l, and combining these equations gives $c = (kl)a$.

Another basic fact is that if $\gcd(a, c) = 1$ and $\gcd(b, c) = 1$ then $\gcd(ab, c) = 1$. For if $\gcd(ab, c) > 1$ there must be some prime number p such that $p|ab$ and $p|c$. (Any prime divisor of $\gcd(ab, c)$ will have this property.) But we have seen that $p|ab$ implies that either $p|a$ or $p|b$, given that p is prime. Since we additionally know that $p|c$, it follows that either $p|a$ and $p|c$, or $p|b$ and $p|c$. But both alternatives are impossible, the former because $\gcd(a, c) = 1$, the latter because $\gcd(b, c) = 1$. So $\gcd(ab, c)$ cannot, after all, be greater than 1.

Here is another proof of this same result. Given that $\gcd(a, c) = 1 = \gcd(bc)$, it follows from the extended Euclidean Algorithm that there exist integers r, s, t and u such that

$$ra + sc = 1 \tag{4}$$

and

$$tb + uc = 1. \tag{5}$$

Multiplying through by b in Eq. (4) gives $b = rab + sbc$, and substituting this formula for b into Eq. (5) gives $trab + tsbc + uc = 1$. That is,

$$tr(ab) + (tsb + u)c = 1. \tag{6}$$

Now if $d|ab$ and $d|c$ then d divides both terms on the left hand side of Eq. (6), from which it follows that d divides the right hand side: that is, $d|1$. Hence 1 is the only (positive) common divisor of ab and c.

(3.3) Proposition: *Let a, b and c be integers. If a and b are both coprime to c then ab is also coprime to c.*

Perhaps giving two proofs of Proposition (3.3) was excessive, but we certainly do know now that it is true!

Recall that a set S of integers is a complete system modulo m if and only if the mod m residues of the elements of S are 0, 1, 2, …, $m - 1$ in some order. So S is a complete system mod m if and only if the following conditions both hold:

C1) S has precisely m elements,
C2) no two elements of S are congruent modulo m.

This characterization of complete systems combined with the coprime cancellation rule (Corollary (2.15)) readily yields the following proposition.

(3.4) Proposition: *Let $a, m \in \mathbb{Z}$ satisfy $\gcd(a,m) = 1$. If $S = \{s_1, s_2, \ldots, s_m\}$ is a complete system modulo m then $\{as_1, as_2, \ldots, as_m\}$ is also a complete system modulo m.*

Proof. Since $\gcd(a,m) = 1$, if $as_i \equiv as_j \pmod{m}$ then $s_i \equiv s_j \pmod{m}$. But since S is a complete system, $s_i \not\equiv s_j \pmod{m}$ for $i \neq j$. So $as_i \not\equiv as_j \pmod{m}$ for $i \neq j$. So the m numbers $as_1, as_2, \ldots, as_m$ are pairwise distinct modulo m, and thus form a complete system. □

For example, let $m = 9$ and $a = 4$. Then $\gcd(a,m) = \gcd(4,9) = 1$, and so Proposition (3.4) can be applied. So since $\{0,1,2,3,4,5,6,7,8\}$ is a complete system mod 9, it follows that $\{0,4,8,12,16,20,24,28,32\}$ is also a complete system mod 9. The reader should check this.

Observe that the case $n = 0$ of Proposition (3.1) follows from the case $m = 26$ of Proposition (3.4) (and the general case of Proposition (3.1) then follows readily).

Recall that a set R of integers is a reduced system modulo m if and only if the mod m residues of the elements of R give all the natural numbers that are less than m and coprime to m, without repetitions. The number of elements in a reduced system is thus $\phi(m)$ (since by the definition of the Euler phi function, $\phi(m)$ is the number of natural numbers less than m and coprime to m). So a set R is a reduced system mod m if and only if the following three conditions all hold:

R1) R has precisely $\phi(m)$ elements,
R2) $\gcd(r,m) = 1$ for all $r \in R$,
R3) no two distinct elements of R are congruent modulo m.

Proposition (3.4) above, which is concerned with complete systems, has the following analogue for reduced systems.

(3.5) Proposition: *Let a, $m \in \mathbb{Z}$ satisfy* $\gcd(a, m) = 1$. *If* $R = \{r_1, r_2, \dots, r_l\}$ *is a reduced system modulo m then* $\{ar_1, ar_2, \dots, ar_l\}$ *is also a reduced system modulo m.*

Proof. We may assume that the numbers $r_1, r_2, \dots, r_l$ are pairwise distinct, so that l is the number of elements of the set R. Thus $l = \phi(m)$, since R is a reduced system mod m.

By coprime cancellation, which is valid since $\gcd(a, m) = 1$, if $ar_i \equiv ar_j \pmod{m}$ then $r_i \equiv r_j \pmod{m}$. But since R is a reduced system mod m, condition R3 above says that $r_i \not\equiv r_j \pmod{m}$ whenever $i \neq j$. So $ar_i \not\equiv ar_j \pmod{m}$ whenever $i \neq j$, which shows in particular that no two of $ar_1, ar_2, \dots, ar_l$ are equal, and also shows that no two distinct elements of the set $R' = \{ar_1, ar_2, \dots, ar_l\}$ are congruent modulo m. So R' satisfies condition R3, and also satisfies condition R1, since it has $l = \phi(m)$ elements.

It remains to observe that R' also satisfies R2. We know that $\gcd(r_i, m) = 1$ for each i (since R satisfies R2) and $\gcd(a, m) = 1$ by hypothesis. So by Proposition (3.3), $\gcd(ar_i, m) = 1$ for all i. That is, the elements of R' are all coprime to m, as required. □

To illustrate the use we shall make of this proposition, consider again the case $m = 9$ and $a = 4$. The set $\{1, 2, 4, 5, 7, 8\}$ is a reduced system modulo 9, and so it follows that $\{4, 8, 14, 20, 28, 32\}$ is also a reduced system modulo 9. The numbers in the latter set are therefore congruent modulo 9 to the numbers in the former set, in some order. It follows that the product of all the numbers in the latter set is congruent modulo 9 to the product of all the numbers in the former set:

$$4 \times 8 \times 16 \times 20 \times 28 \times 32 \equiv 1 \times 2 \times 4 \times 5 \times 7 \times 8 \pmod{9}.$$

Since each factor on the left hand side can be written as $4i$ for some i, this gives

$$4^6(1 \times 2 \times 4 \times 5 \times 7 \times 8) \equiv 1 \times 2 \times 4 \times 5 \times 7 \times 8 \pmod{9}.$$

But now the number $1 \times 2 \times 4 \times 5 \times 7 \times 8$ is coprime to 9, since all the factors are coprime to 9, and so by the coprime cancellation principle applied to the congruence above we deduce that $4^6 \equiv 1 \pmod{9}$.

The exponent 6 that appears in this comes about because that was the number of elements of the reduced system. In other words, the exponent is 6 because $\phi(9) = 6$. It is not hard to see that exactly the same argument will work for any a and m such that $\gcd(a, m) = 1$, and the conclusion will be that $a^{\phi(m)} \equiv 1 \pmod{m}$. This fact, which is known as the Euler–Fermat theorem, is one of number theoretic facts that form the basis of modern-day public key cryptosystems.

PART 3C: *Extra Comments*

Interested students can download a computer program that simulates a cipher machine used by the United States in World War 2. The program in question is the "M-209 simulator" at http://users.telenet.be/d.rijmenants/en/m209sim.htm.

The description of the Hagelin machine given in §18 above is reasonably compatible with the way the M-209 simulator works, but there are a few points to be made. The first is that the correspondence between letters and numbers that we have used is A ↔ 0, B ↔ 1, . . . , Z ↔ 25, but the M-209 uses the more natural A ↔ 1, B ↔ 2, . . . , Z ↔ 26. The effect of this is that once one finally manages to correctly set the lugs and pins in the simulator to correspond to the example in §18, it will be found that according to the simulator the plaintext SYDNEY gives the ciphertext SBLSSV, whereas in §18 we obtained TCMTTW as the ciphertext. Our answer is obtained from the other by replacing each letter by the one that follows it in the alphabet.

Setting the lugs and pins in the simulator to correspond to our example is a somewhat tedious process. Setting the lugs is relatively straightforward and intuitive, but the pins are harder. The description in §18 said that each pin could be either raised (active) or lowered (inactive). In the actual machines the pins were pushed through to the right hand side of the corresponding key wheel to make them active, or to the left to make them inactive. Again, setting the pins in the simulator is easy enough (if tedious) with a bit of practice. The trickiest bit is figuring out which pins in the simulator should be active and which should be inactive to correspond the the example in §18.

As explained in the simulator's help file (in the "technical details" section), when the key wheels are all in their A positions, the pins that are actally adjacent to the drum are as follows: the P pin on the first key wheel, the O pin on the second key wheel, the N pin on the third key wheel, the M pin on the fourth key wheel, the L pin on the fifth key wheel and the K pin on the sixth key wheel. So the "first row of pins", in the terminology of our example, actually means the P pin on the first key wheel, the O pin on the second key wheel, the N pin on the third key wheel, the M pin on the fourth key wheel, the L pin on the fifth key wheel and the K pin on the sixth key wheel. And then the second row will be Q, P, O, N, M, L, and so on. The upshot is that in our example the active pins are as follows:

1st key wheel:	A,B,C,D,E,F,G,L,M,N,O,T,U,V,W,Y,Z
2nd key wheel:	B,C,E,H,J,N,Q,X,Y
3rd key wheel:	A,C,D,E,I,J,K,L,Q,R,S,T,U,V
4th key wheel:	A,B,D,E,I,L,Q,R,S,T,U
5th key wheel:	A,F,H,J,K,L,N,P,Q,R,S
6th key wheel:	E,F,H,I,J,K,M,O.

Week 4

§20 The Euler–Fermat Theorem

Recall that the Euler phi function is defined as follows: if m is a positive integer then $\phi(m)$ is the number of natural numbers less than m that are coprime to m. This is the same as the number of elements of any reduced system modulo m.

(4.1) The Euler–Fermat Theorem: *Let a, $m \in \mathbb{Z}$, and suppose that* $\gcd(a, m) = 1$. *Then* $a^{\phi(m)} \equiv 1 \pmod{m}$.

Proof. Let $l = \phi(m)$ and let $R = \{r_1, r_2, \ldots, r_l\}$ be a reduced system modulo m. Thus $\gcd(r_i, m) = 1$ for each i.

Recall that the product of two numbers coprime to m is also coprime to m (by Proposition (3.3)). It is easy to see from this that a multiplying any number of factors will give a product that is coprime to m if all the factors are coprime to m. Thus $r_1 r_2 \cdots r_l$ is coprime to m. (The idea is that $r_1 r_2$ is coprime to m since r_1 and r_2 both are; but now we have that $r_1 r_2 r_3$ is coprime to m since $r_1 r_2$ and r_3 both are, after which the same reasoning shows that $r_1 r_2 r_3 r_4$ is coprime to m, and so on.)

Given that $\gcd(a, m) = 1$ and that $R = \{r_1, r_2, \ldots, r_l\}$ is a reduced system mod m, it follows from Proposition (3.5) that $\{ar_1, ar_2, \ldots, ar_l\}$ is also a reduced system mod m, and this means that the numbers $ar_1, ar_2, \ldots, ar_l$ are congruent (modulo m) to the numbers $r_1, r_2, \ldots, r_l$ in some order. Hence

$$(ar_1)(ar_2)\cdots(ar_l) \equiv r_1 r_2 \cdots r_l \pmod{m}.$$

Thus we see that

$$a^l (r_1 r_2 \cdots r_l) \equiv r_1 r_2 \cdots r_l \pmod{m},$$

and by coprime cancellation we deduce that $a^l \equiv 1 \pmod{m}$, as required. □

When p is prime then any number that is not divisible by p is coprime to p, and hence $\{1, 2, \ldots, p-1\}$ is a reduced system modulo p. So $\phi(p) = p - 1$. In this case the Euler–Fermat theorem says that $a^{p-1} \equiv 1 \pmod{p}$ whenever a is coprime to p. This is Fermat's part of Euler–Fermat, and is also known as “Fermat's Little Theorem”.

(4.2) Fermat's Little Theorem: *Let p be a prime, and suppose that $a \in \mathbb{Z}$ and $p \nmid a$. Then* $a^{p-1} \equiv 1 \pmod{p}$.

(4.3) ***Example:***

1) Let $p = 7$, $a = 2$. Then $a^{p-1} = 2^6 = 64 \equiv 1 \pmod{7}$.
2) Let $p = 7$, $a = 3$. Then $a^{p-1} = 3^6 = 729 = 7 \times 104 + 1 \equiv 1 \pmod{7}$.
3) Let $p = 13$, $a = 2$. Then $a^{p-1} = 2^{12} = 4095 + 1 = 5 \times 7 \times 9 \times 13 + 1 \equiv 1 \pmod{13}$.
4) Let $m = 10$. Then $\{1, 3, 7, 9\}$ is a reduced system modulo m, and so $\phi(m) = 4$. Hence the 4th powers of 1, 3, 7 and 9 will all be congruent to 1 modulo 10. That is to say, the final digit of the number will be 1. It is readily checked: $1^4 = 1$, obviously, and $3^4 = 81$, while $7^4 \equiv (-3)^4 = 81$ and $9^4 \equiv (-1)^4 = 1$. (In fact, $7^4 = 2401$ and $9^4 = 6561$.)

By Fermat's Little Theorem, if p is prime and $p \nmid a$ then $a^{p-1} \equiv 1 \pmod{p}$. Multiplying both sides of this by a gives $a^p \equiv a \pmod{p}$ whenever $p \nmid a$. But this congruence is clearly also true when $p \mid a$, since in this case a^p and a are both congruent to 0. In fact, Fermat's Little Theorem is usually stated so as to include this case: if p is prime and a is an arbitrary integer then $a^p \equiv a \pmod{p}$.

It is natural to wonder whether the same idea can be applied with a modulus m that is not prime. For instance, in Example (4.3) above we saw that $a^4 \equiv 1 \pmod{10}$ whenever $\gcd(a, 10) = 1$; so it follows that $a^5 \equiv a \pmod{10}$ whenever $\gcd(a, 10) = 1$. Is the same also true when $\gcd(a, 10) \neq 1$? Let us check!

There is no doubt that if $a \equiv 0 \pmod{10}$ then $a^5 \equiv 0 \equiv a \pmod{10}$. If $a \equiv 2 \pmod{10}$ then $a^5 \equiv 2^5 = 32 \equiv 2 \equiv a \pmod{10}$. The others are equally easy to check: 4^n is congruent mod 10 to 6 if n is even and 4 if n is odd, so that in particular $4^5 \equiv 4 \pmod{10}$; 5^n and 6^n are congruent mod 10 to 5 and 6 (respectively) for all n, including $n = 5$; since $8 \equiv -2$ we have $8^5 \equiv (-2)^5 \equiv -(2^5) \equiv -2 \equiv 8$. (The actual numbers are $4^5 = 1024$, $5^5 = 3125$, $6^5 = 7776$, $8^5 = 32768$.)

At this stage one might conjecture that $a^{\phi(m)+1} \equiv a \pmod{m}$ holds for all m and a. But it is not true, as the following trivial example shows. If $m = 4$ then $\phi(m) = 2$, giving $\phi(m) + 1 = 3$. It is not true that $2^3 \equiv 2 \pmod{4}$; indeed, it is obvious that $2^3 \equiv 0 \pmod{4}$. The same occurs whenever m is the square of a prime p: we find that $p^{\phi(m)+1} \not\equiv p \pmod{m}$, since in fact $p^{\phi(m)+1}$ is divisible by $p^2 = m$. Nevertheless, if m is a *square-free* number, meaning that it is not divisible by the square of any prime, then $a^{\phi(m)+1} \equiv a \pmod{m}$ does hold for all integers a.

Let us confirm this in the simplest—and for us most important—case: when m is the product of two distinct primes p and q. Our first task is to calculate $\phi(pq)$; so we start by considering $S = \{0, 1, 2, \ldots, pq - 1\}$, a complete system mod $m = pq$. Then S has pq elements altogether, and for each $a \in S$ the value of $\gcd(a, m)$ must be one of 1, p, q or pq, since these are the only positive integers that are divisors of m. Of course $\gcd(a, m) = m$ can only occur if $m \mid a$, and so 0 is the one and only $a \in S$ with $\gcd(a, m) = pq$. Thus the $a \in S$ for which $\gcd(a, m) = p$ must be the nonzero multiples of p that are in S. There are exactly $q - 1$ of these, namely p, $2p$, $3p$, …, $(q - 1)p$. Similarly there are exactly $p - 1$ elements $a \in S$ with

$\gcd(a, m) = q$, namely the numbers iq for i in the range 1 to $p - 1$. Subtracting from the total number of elements a in S the number for which $\gcd(a, m) = m$ (namely 1) plus the number for which $\gcd(a, m) = p$ (namely $q - 1$) plus the number for which $\gcd(a, m) = q$ (namely $p - 1$), will leave the number for which $\gcd(a, m) = 1$. Thus

$$\begin{aligned}\phi(m) &= \phi(pq)\\ &= pq - (1 + (q-1) + (p-1))\\ &= pq - p - q + 1\\ &= (p-1)(q-1).\end{aligned}$$

If $\gcd(a, m) = 1$ then the Euler–Fermat Theorem says that $a^{\phi(m)} \equiv 1 \pmod{m}$, giving $a^{\phi(m)+1} \equiv a \pmod{m}$, and if $\gcd(a, m) = m$ then obviously $a^{\phi(m)+1} \equiv 0 \equiv a \pmod{m}$. So it remains to deal with the cases $\gcd(a, m) = p$ and $\gcd(a, m) = q$.

Let a be an integer with $\gcd(a, m) = p$. Then $p|a$, and so $a = ip$ for some i. Certainly $q\nmid i$, since otherwise a would be divisible by pq, making $\gcd(a, m) = pq$, contrary to our assumption. So $q\nmid ip = a$, and applying Fermat's Little Theorem to a and the prime q tells us that $a^{q-1} \equiv 1 \pmod{q}$. Raising both sides to the $(p - 1)$-st power gives $a^{(q-1)(p-1)} \equiv 1^{p-1} \equiv 1 \pmod{q}$. That is, $a^{\phi(m)} \equiv 1 \pmod{q}$, from which we deduce that $a^{\phi(m)+1} \equiv a \pmod{q}$.

The calculations in the above paragraph do not tell us what we wanted to know: they do not tell us that $a^{\phi(m)+1} \equiv a$ modulo m, but merely that $a^{\phi(m)+1} \equiv a$ modulo q. That is, we have shown that q is a divisor of $a^{\phi(m)+1} - a$, but not that pq is. But we had assumed that $p|a$, from which it follows immediately that $p|a^{\phi(m)+1}$, and hence that $p|(a^{\phi(m)+1} - a)$. And thus since p and q are two distinct prime factors of $a^{\phi(m)+1} - a$, the Fundamental Theorem of Arithmetic ensures that pq divides $a^{\phi(m)+1} - a$, as required.

The same argument obviously also applies with the roles of p and q reversed. If $\gcd(a, m) = q$ then by Fermat's Little Theorem we have that $a^{p-1} \equiv 1 \pmod{p}$, yielding that $a^{(p-1)(q-1)} \equiv 1$ and $a^{\phi(m)+1} \equiv a \pmod{p}$. Since also $a^{\phi(m)+1} \equiv 0 \equiv a \pmod{q}$ we conclude that $a^{\phi(m)+1} \equiv a \pmod{pq}$, as required. So we have established, as claimed, that $a^{\phi(m)+1} \equiv a \pmod{m}$ holds for all $a \in \mathbb{Z}$, if m is the product of two distinct primes.

Note as a consequence of this that if k is any positive integer then

$$a^{k\phi(m)+1} \equiv a^{(k-1)\phi(m)}a^{\phi(m)+1} \equiv a^{(k-1)\phi(m)}a \equiv a^{(k-1)\phi(m)+1} \pmod{m},$$

and an easy induction now shows that $a^{k\phi(m)+1} \equiv a \pmod{m}$ for all positive integers a (given that m is a product of two distinct primes). We remark that the RSA cryptosystem relies on this centuries old piece of number theory, which for most of its history was certainly regarded by many as something of no "real world" importance.*

* This "real world" concept varies wildly from individual to individual.

§21 The Chinese Remainder Theorem

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	400	275	150	25	425	300	175	50	450	325	200	75	475	350	225	100	500	375	250	125
1	126	1	401	276	151	26	426	301	176	51	451	326	201	76	476	351	226	101	501	376	251
2	252	127	2	402	277	152	27	427	302	177	52	452	327	202	77	477	352	227	102	502	377
3	378	253	128	3	403	278	153	28	428	303	178	53	453	328	203	78	478	353	228	103	503
4	504	379	254	129	4	404	279	154	29	429	304	179	54	454	329	204	79	479	354	229	104
5	105	505	380	255	130	5	405	280	155	30	430	305	180	55	455	330	205	80	480	355	230
6	231	106	506	381	256	131	6	406	281	156	31	431	306	181	56	456	331	206	81	481	356
7	357	232	107	507	382	257	132	7	407	282	157	32	432	307	182	57	457	332	207	82	482
8	483	358	233	108	508	383	258	133	8	408	283	158	33	433	308	183	58	458	333	208	83
9	84	484	359	234	109	509	384	259	134	9	409	284	159	34	434	309	184	59	459	334	209
10	210	85	485	360	235	110	510	385	260	135	10	410	285	160	35	435	310	185	60	460	335
11	336	211	86	486	361	236	111	511	386	261	136	11	411	286	161	36	436	311	186	61	461
12	462	337	212	87	487	362	237	112	512	387	262	137	12	412	287	162	37	437	312	187	62
13	63	463	338	213	88	488	363	238	113	513	388	263	138	13	413	288	163	38	438	313	188
14	189	64	464	339	214	89	489	364	239	114	514	389	264	139	14	414	289	164	39	439	314
15	315	190	65	465	340	215	90	490	365	240	115	515	390	265	140	15	415	290	165	40	440
16	441	316	191	66	466	341	216	91	491	366	241	116	516	391	266	141	16	416	291	166	41
17	42	442	317	192	67	467	342	217	92	492	367	242	117	517	392	267	142	17	417	292	167
18	168	43	443	318	193	68	468	343	218	93	493	368	243	118	518	393	268	143	18	418	293
19	294	169	44	444	319	194	69	469	344	219	94	494	369	244	119	519	394	269	144	19	419
20	420	295	170	45	445	320	195	70	470	345	220	95	495	370	245	120	520	395	270	145	20
21	21	421	296	171	46	446	321	196	71	471	346	221	96	496	371	246	121	521	396	271	146
22	147	22	422	297	172	47	447	322	197	72	472	347	222	97	497	372	247	122	522	397	272
23	273	148	23	423	298	173	48	448	323	198	73	473	348	223	98	498	373	248	123	523	398
24	399	274	149	24	424	299	174	49	449	324	199	74	474	349	224	99	499	374	249	124	524

The table above has 25 rows indexed by the congruence classes modulo 25, and 21 columns indexed by the congruence classes modulo 21. It was filled in by inserting the number k at the intersection of row a and column b, starting with $k = 0$ in the top left hand corner, where $a = 0$ and $b = 0$, and then increasing k, a and b by 1 for each successive insertion, continuing for as long as is possible without overwriting earlier insertions. The effect is that we move diagonally downwards and to the right as k increases, except that since a and b are meant to be residues modulo 25 and 21 respectively, they must be returned to 0 when they would otherwise move out of range. Thus "increasing" $a = 24$ by 1 means moving to $a = 0$, and "increasing" $b = 20$ by 1 means moving to $b = 0$. That is, when we hit the right edge we wrap around to the left edge, and when we hit the bottom edge we go up to the top edge.

As you can see, the table gets filled in nicely with all $525 = 25 \times 21$ numbers from 0 to 524 inclusive. It is not hard to see that this happens because 25 and 21 are coprime. Trying the same game with the non-coprime pair 4 and 6 comes unstuck halfway through.

	0	1	2	3	4	5
0	0		8		4	
1		1		9		5
2	6		2		10	
3		7		3		11

The first important thing to observe about the 25 by 21 table above is the relationship between k, a and b: with k in row a and column b we have

$$\begin{aligned} k &\equiv a \pmod{25} \\ k &\equiv b \pmod{21} \end{aligned} \tag{7}$$

in every case. For example, the fact that $194 \equiv 19 \pmod{25}$ and $194 \equiv 5 \pmod{21}$ means that 194 is in row 19 and column 5. (Check that $194 = 7 \times 25 + 19$ and $194 = 9 \times 21 + 5$.) Similarly, 7 is the residue of 457 mod 25, and 16 the residue of 457 mod 21; so 457 is in row 7 and column 16.

It is easy to see why this property holds, given the way we constructed the table. We started by putting 0 in row 0 and column 0, since the mod 25 and mod 21 residues of 0 are, of course, both 0. Now at each stage we increase k by 1, and we also increase a by 1 (mod 25) and b by 1 (mod 21), and since $k \equiv a \pmod{25}$ and $k \equiv b \pmod{21}$ certainly imply that $k + 1 \equiv a + 1 \pmod{25}$ and $k + 1 \equiv b + 1 \pmod{21}$, we have an inductive proof that Eq. (7) above remains true as the whole table is completed.

We have yet to prove that if 25 and 21 are replaced by an arbitrary pair of coprime positive integers m and n then the process will still work. We can certainly make a table with rows indexed by $S = \{0, 1, \ldots, m-1\}$ and columns indexed by $T = \{0, 1, \ldots, n-1\}$, put $k = 0$ in row $a = 0$ and column $b = 0$, and move diagonally, as before, as we increase k. This ensures (by induction) that $k \equiv a \pmod{m}$ and $k \equiv b \pmod{n}$ for all k, where a and b are the row and column indices of k. But how do we know that this fills in the whole table without any overwriting taking place? How do we know that the numbers 0, 1, 2, …, $mn - 1$ will all go in different places in the table, with no places left over?

We know because the Chinese Remainder Theorem says so!

(4.4) The Chinese Remainder Theorem: *Let m and n be coprime positive integers. Then for each $a \in \{0, 1, \ldots, m-1\}$ and $b \in \{0, 1, \ldots, n-1\}$ there is a unique $k \in \{0, 1, 2, \ldots, mn-1\}$ such that $k \equiv a \pmod{m}$ and $k \equiv b \pmod{n}$.*

Before we prove this, a lemma is needed.

(4.5) Lemma: *Let m and n be coprime integers, and suppose that l is an integer such that $m \mid l$ and $n \mid l$. Then $mn \mid l$.*

Proof. Since $m|l$ we have $l = mm'$ for some $m' \in \mathbb{Z}$. Our assumption that m and n are coprime and our assumption that $n|l$ therefore combine to tell us that

$$n|mm' \text{ and } \gcd(n,m) = 1.$$

These two facts imply that $n|m'$, by Theorem (2.6). (Recall that this was the key result used in the proof of the unique factorization theorem for $\mathbb{Z}$, the Fundamental Theorem of Arithmetic.) Thus we can write $m' = nn'$ for some $n' \in \mathbb{Z}$, which combines with $l = mm'$ to give $l = mnn'$. So $mn|l$, as claimed. □

Note that it is also easy to use the Fundamental Theorem of Arithmetic to prove the lemma. The point is basically that m and n contribute disjoint sets of prime factors to the prime factorization of l, and so all the primes appearing in the factorization of mn occur at least as often in the prime factorization of l.

Proof of the Chinese Remainder Theorem. Let $P = \{ k \in \mathbb{Z} \mid 0 \le k \le mn - 1 \}$ and $H = \{ (a,b) \mid 0 \le a \le m-1 \text{ and } 0 \le b \le n-1 \}$. Then both P and H have mn elements. So any function from the set P to the set H that is one to one will necessarily be onto. (This is the so-called pigeonhole principle. If we have the same (finite) number of pigeons as pigeonholes, and we put all the pigeons in different holes, then we exactly use up all the holes. In our case the pigeons are the elements of the set P (the numbers from 0 to $mn-1$) and the holes are the places in the table, or the pairs of numbers that comprise the set H.)

We define $f\colon P \to H$ by the rule that for all k,

$$f(k) = (\mathrm{res}_m(k), \mathrm{res}_n(k)),$$

where $\mathrm{res}_m(k)$ means the residue of k mod m and $\mathrm{res}_n(k)$ means the residue of k mod n. The Chinese Remainder Theorem asserts that for each (a,b) in H there is a unique k in P such that $\mathrm{res}_m(k) = a$ and $\mathrm{res}_n(k) = b$; that is, it says that the function f is one to one and onto. By the pigeonhole principle it is sufficient to prove that f is one-to-one, given that P and H both have mn elements.

So suppose that k_1 and k_2 are elements of P such that $f(k_1) = f(k_2)$, and write $(a,b) = f(k_1) = f(k_2)$. Then we have that $\mathrm{res}_m(k_1) = \mathrm{res}_m(k_2) = a$ and $\mathrm{res}_n(k_1) = \mathrm{res}_n(k_2) = b$. Thus

$$\begin{aligned} k_1 \equiv k_2 \equiv a &\pmod{m} \\ k_1 \equiv k_2 \equiv b &\pmod{n}, \end{aligned}$$

and thus $m|(k_1 - k_2)$ and $n|(k_1 - k_2)$. But $\gcd(m,n) = 1$; so Lemma (4.5) implies that $mn|(k_1 - k_2)$. But since k_1 and k_2 are both in $P = \{0, 1, \ldots, mn-1\}$ their difference is at most $mn - 1$. Therefore, being a multiple of mn, this difference must be zero. That is, $k_1 = k_2$. Thus we have shown that $f(k_1)$ can only equal $f(k_2)$ if $k_1 = k_2$; in other words, f is one to one, as required. □

There is a statement that is rather analogous to the pigeonhole principle—indeed, it could be called a variant of the pigeonhole principle—that says that if two finite sets P and H have the same number of elements then any function from P to H that is onto is necessarily one-to-one. If we fill up all the holes with pigeons then we can never put two pigeons in the same hole. We can use this to give a second proof of the Chinese Remainder Theorem, proving directly that the function f defined above is onto, and concluding that it must therefore be one-to-one as well.

Second proof of the Chinese Remainder Theorem. Since $\gcd(m,n) = 1$ it follows from the extended Euclidean Algorithm that there exist integers s, t satisfying

$$sm + tn = 1. \tag{8}$$

Let $E = tn$ and $F = sm$. Then

$$\begin{aligned} E &\equiv 1 \pmod{m} \\ E &\equiv 0 \pmod{n} \end{aligned}$$

the second of these being obvious since E is defined as a multiple of n, and the first following from Eq. (8) since $E - 1 = -sm$ is aa multiple of m. Similarly

$$\begin{aligned} F &\equiv 0 \pmod{m} \\ F &\equiv 1 \pmod{n} \end{aligned}$$

the first from the definition of F and the second from Eq. (8).

With the sets P and H and the function f defined as in the first proof, let (a,b) be an arbitrary element of H, and define k to be the residue of $aE + bF$ modulo mn. Then $k \in P$, and since $k \equiv aE + bF \pmod{mn}$ it follows that $k \equiv aE + bF \pmod{m}$ and $k \equiv aE + bF \pmod{n}$. Thus

$$k \equiv aE + bF \equiv a1 + b0 \equiv a \pmod{m}$$

and

$$k \equiv aE + bF \equiv a0 + b1 \equiv b \pmod{n}.$$

Hence $f(k) = (a,b)$, and since (a,b) was chosen as an arbitrary element of H this shows that the function f is onto, as required. □

Look once more at the 25×21 table from the start of this section. If we take a value of a (a row number) that is a multiple of 5, then every k in that row is a multiple of 5. This is because every such k satisfies $k \equiv a \pmod{25}$, and since a and 25 are multiples of 5 it follows that k is too. Similarly, if we take a value of b (a column number) that is a multiple of 7, then every k in that column is a multiple of 7 (since $k \equiv b \pmod{21}$, and this implies that $k \equiv b \equiv 0 \pmod{7}$). And the same thing holds for the columns indexed by multiples of 3. Furthermore, the converse statements are true also: if $k \in \{0, 1, \ldots, 524\}$ and $5 | k$ then the row index a of k

must be a multiple of 5 (since $k \equiv a \pmod{25}$ and k and 25 are both multiples of 5); similarly, if $7|k$ or $3|k$, and b is the column index of k, then $k \equiv b \pmod{21}$, giving $0 \equiv b$, modulo 7 or 3 as the case may be. Now every k such that $\gcd(k, 525) \neq 1$ must be a multiple of 3, 5 or 7, since these are the only prime divisors of 525, and so every such k lies in a row indexed by a multiple of 5 or a column indexed by a multiple of 3 or 7. And every number k in such a row or such a column will have $\gcd(k, 525) > 1$. So the numbers k for which $\gcd(k, 525) = 1$ are exactly the numbers that would be left if we were to erase the columns indexed by multiples of 3 or 7 and the rows indexed by multiples of 5.

Removing from the set $T = \{0, 1, \ldots, 20\}$ the numbers divisible by 3 or 7 will change T from a complete system mod 21 to a reduced system mod 21; exactly $\phi(21)$ columns will remain. Similarly, removing from the set $S = \{0, 1, \ldots, 24\}$ the numbers divisible by 5 will change S from a complete system mod 25 to a reduced system mod 25; exactly $\phi(25)$ rows will remain. So deleting these rows and columns will leave us with a $\phi(25) \times \phi(21)$ array, and will also remove exactly those numbers k for which $\gcd(k, 525) > 1$. The numbers k that are left will therefore form a reduced system modulo 525, and of course there must be exactly $\phi(525)$ of these. We conclude that $\phi(525) = \phi(25)\phi(21)$.

The same reasoning shows that $\phi(mn) = \phi(m)\phi(n)$ whenever m and n are coprime. In the notation of our proof of the Chinese Remainder Theorem, the set P is a complete system mod mn, and each $k \in P$ corresponds to a unique pair (a, b) in the set H. Now if $\gcd(k, mn) > 1$ then there must be some prime p such that $p|k$ and $p|mn$. Now $p|mn$ implies $p|m$ or $p|n$ (since p is prime), and so we either have that $p|k$ and $p|m$ or that $p|k$ and $p|n$. The former alternative implies that $\gcd(k, m) > 1$ and the latter alternative implies that $\gcd(k, n) > 1$. And $\gcd(k, m) = \gcd(a, m)$ since $k \equiv a \pmod{m}$, and $\gcd(k, n) = \gcd(b, n)$ since $k \equiv b \pmod{n}$. This has shown that if $\gcd(k, mn) > 1$ then either $\gcd(a, m) > 1$ or $\gcd(b, n) > 1$. Conversely, if $\gcd(a, m) > 1$ then there is a prime p that divides both a and m, and hence divides both k (as $k = a + qm$ for some q) and mn, so that $\gcd(k, mn) > 1$. Similarly if $\gcd(b, n) > 1$ then $\gcd(k, mn) > 1$. Thus we have shown that in our one to one correspondence

$$k \leftrightarrow (a, b)$$

between the sets P and H, we have that $\gcd(k, mn) > 1$ if and only if either $\gcd(a, m) > 1$ or $\gcd(b, n) = 1$. Equivalently put, $\gcd(k, mn) = 1$ if and only if both $\gcd(a, m) = 1$ and $\gcd(b, n) = 1$. Therefore we also have a one-to-one correspondence between the set

$$P' = \{\, k \in \mathbb{Z} \mid 0 \leq k \leq mn - 1 \text{ and } \gcd(k, mn) = 1 \,\}$$

(which is a reduced system modulo mn) and the set

$$H' = \{\, (a, b) \mid 0 \leq a \leq m{-}1 \text{ and } \gcd(a, m) = 1, \text{ and } 0 \leq b \leq n{-}1 \text{ and } \gcd(b, n) = 1 \,\}$$

(which is the Cartesian product of a reduced system modulo m and a reduced system modulo n). Since K' has $\phi(mn)$ elements and H' has $\phi(m)\phi(n)$ elements, we conclude that $\phi(mn) = \phi(m)\phi(n)$ whenever m and n are coprime.

§22 Solving simultaneous congruences

We illustrate a number of general principles by considering a sequence of examples.

Problem 1: *Solve the congruence* $56x \equiv 48$ *(mod 91), where* $x \in \mathbb{Z}$*.*

Solution. We want $56x = 48 + 91k$ for some $k \in \mathbb{Z}$. Thus we want

$$48 = 56x - 91k = 7(8x - 13k)$$

for some $k \in \mathbb{Z}$, which obviously cannot be done since $7 \nmid 48$.

Conclusion: *The congruence* $ax \equiv b \pmod{m}$ *has no solution* $x \in \mathbb{Z}$ *if* $\gcd(a, m) \nmid b$*.*

Problem 2: *Solve the congruence* $56x \equiv 49$ *(mod 91), where* $x \in \mathbb{Z}$*.*

Solution. We want $56x = 49 + 91k$ for some $k \in \mathbb{Z}$. Dividing through by 7 we see that this is equivalent to $8x = 7 + 13k$ for some $k \in \mathbb{Z}$, or $8x \equiv 7 \pmod{13}$.

Conclusion: *If* $\gcd(a, m) | b$ *then the congruence* $ax \equiv b \pmod{m}$ *is equivalent to the congruence* $(a/d)x \equiv (b/d) \pmod{(m/d)}$*, where* $d = \gcd(a, m)$*.*

In other words, dividing everything by a common factor does not alter the set of solutions.

We still have not solved Problem 2, but we have actually already seen how to solve congruences of the form $ax \equiv b \pmod{m}$ in the case that $\gcd(a, m) = 1$. You can do it with the extended Euclidean Algorithm. In particular, the first step in solving $8x \equiv 7 \pmod{13}$ is to find integers s and t such that $13s - 8t = \pm 1$.

13	8	5	3	2	1	0
		1	1	1	1	2
0	1	1^-	2	3^-	5	
1	0	1	1^-	2	3^-	

The conclusion is that $13s + 8t = 1$ when $s = -3$ and $t = 5$. Reading the equation modulo 13 we deduce that $k = 5$ is a solution of $8k \equiv 1 \pmod{13}$.

Note that $8k \equiv 1 \pmod{13}$ and $k \equiv 5 \pmod{13}$ are equivalent. Starting from $8k \equiv 1 \pmod{13}$ and multiplying by 5 gives $40k \equiv 5 \pmod{13}$, whence $k \equiv 5 \pmod{13}$, since $40 \equiv 1 \pmod{13}$. And, conversely, starting from $k \equiv 5 \pmod{13}$ and multiplying by 8 gives $8k \equiv 40 \equiv 1 \pmod{13}$.

Conclusion: *If* $\gcd(a, m) = 1$ *then the congruence* $ax \equiv 1$ *(mod m) is equivalent to the congruence* $x \equiv a'$ *(mod m) for some integer* a'*. Moreover,* a' *can be found via the extended Euclidean Algorithm applied to a and m.*

The number a' in this statement will often be referred to as the *inverse of a modulo m*. Note, however, that it is only unique up to congruence modulo m. In our example, where m was 13 and a was 8, we found $a' = 5$. But $a' = 18$ or $a' = -8$ would do just as well, as indeed would anything that is congruent to 5 modulo 13.

We *still* have not solved Problem 2! We reduced it to solving $8x \equiv 7 \pmod{13}$, but then we solved $8k \equiv 1 \pmod{13}$ instead of what we were originally trying to solve. But we solved it by finding that 5 is the inverse of 8 modulo 13, and this information makes it easy to solve $8x \equiv 7 \pmod{13}$ also. Just multiply by the inverse: $8x \equiv 7$ implies $40x \equiv 35$, or $x \equiv 9 \pmod{13}$. Note that multiplying back by 8 recovers the original congruence: $x \equiv 9$ implies $8x \equiv 72 \equiv 7 \pmod{13}$. So we have neither created nor lost any solutions, and the upshot is the the set of solutions to Problem 2 is precisely the set of all integers x that are congruent to 9 modulo 13.

Conclusion: *If* $\gcd(a, m) = 1$ *then the solution of the congruence* $ax \equiv b \pmod{m}$ *is* $x \equiv a'b \pmod{m}$*, where the integer* a' *is the inverse of* a *modulo* m*. The integer* a' *and the solution are unique up to congruence modulo* m*.*

Using the extended Euclidean Algorithm to solve $8x \equiv 7 \pmod{13}$ is really overkill. Since the numbers are so small, you can easily find the solution by trial and error. Notice that replacing 7 by $7 \pm 13k$ doesn't change anything, and trying all the possibilities for k we can quickly find a value for which $7 \pm 13k$ is a multiple of 8. In fact $7 - 13 \times 3 = -32$; so the congruence $8x \equiv 7 \pmod{13}$ can be written as $8x \equiv -32 \pmod{13}$, and by coprime cancellation this is equivalent to $x \equiv -4 \pmod{13}$. (This is the same as our previous answer, since $-4 \equiv 9 \pmod{13}$.)

Conclusion: *A good way to solve* $ax \equiv b \pmod{m}$ *when* $\gcd(a, m) = 1$ *and the numbers are small is to go through the numbers* b*,* $b \pm m$*,* $b \pm 2m$*, … successively until one is found that is divisible by* a*. If* b' *is such a number then* $ax \equiv b \pmod{m}$ *is equivalent to* $ax \equiv b' \pmod{m}$*, and by coprime cancellation the solution is* $x \equiv b'/a \pmod{m}$*.*

Problem 3: *Find all* $x \in \mathbb{Z}$ *simultaneously satisfying*

$$\begin{aligned} x &\equiv 1 \pmod{13}, && (i)\\ x &\equiv 4 \pmod{15}, && (ii)\\ x &\equiv 8 \pmod{19}. && (iii)\end{aligned}$$

Solution. Congruence (*i*) gives $x = 1 + 13k$ for some $k \in \mathbb{Z}$. So (*ii*) becomes $1 + 13k \equiv 4 \pmod{15}$, and reducing everything mod 15 this becomes $-2k \equiv 3 \pmod{15}$. This in turn is the same as $-2k \equiv -12 \pmod{15}$, and since $\gcd(2, 15) = 1$ we may cancel and get $k \equiv 6 \pmod{15}$. So $k = 6 + 15l$ for some $l \in \mathbb{Z}$, and therefore

$$x = 1 + 13k = 1 + 13(6 + 15l) = 79 + 195l.$$

Alternatively expressed, these calculations show that the solution of the simultaneous congruences (*i*) and (*ii*) is $x \equiv 79 \pmod{195}$. This is very much as it should be, since the Chinese Remainder Theorem guarantees that the solution of the simultaneous congruences $x \equiv a \pmod{13}$ and $x \equiv b \pmod{15}$ is bound to be $x \equiv k \pmod{195}$, for some k that is uniquely determined modulo 195. A pair (a, b) consisting of a residue mod 13 and a residue mod 15 yields a unique residue k mod 195 with $k \equiv a \pmod{13}$ and $k \equiv b \pmod{15}$.

So now we have reduced the problem of solving (*i*), (*ii*) and (*iii*) simultaneously to the problem of solving

$$x \equiv 8 \pmod{19} \qquad (iii)$$
$$x \equiv 79 \pmod{195} \qquad (iv)$$

simultaneously, and we can use the same method again.

Using (*iv*) we have $x = 79 + 195l$ for some $l \in \mathbb{Z}$, and substituting this into (*iii*) gives $79 + 195l \equiv 8 \pmod{19}$. Reducing everything mod 19 converts this to $3 + 5l \equiv 8 \pmod{19}$, or $5l \equiv 5 \pmod{19}$, and cancelling (since $\gcd(5, 19) = 1$) gives $l \equiv 1 \pmod{19}$. Thus we have $l = 1 + 19m$ for some $m \in \mathbb{Z}$, and

$$x = 79 + 195(1 + 19m) = 274 + 3705m.$$

So the solution is $x \equiv 274 \pmod{3705}$.

We could obviously go on and deal with any number of simultaneous congruences in the same way. The process consists of solving a succession of pairs of simultaneous congruences of the general form $x \equiv a \pmod{m}$ and $x \equiv b \pmod{n}$. In the case that m and n are coprime the solution is $x \equiv k \pmod{mn}$ for some k that is uniquely determined modulo mn. In the case that m and n are not coprime we can still apply the same method as in our example above: substitute $x = a + mk$ (for some $k \in \mathbb{Z}$) into $x \equiv b \pmod{n}$ and conclude that the problem is equivalent to solving $a + mk \equiv b \pmod{n}$, or $mk \equiv b - a \pmod{n}$. As we observed earlier in this section, this has no solution if $d = \gcd(m, n)$ does not divide $b - a$, whereas if d does divide $b - a$ then we can divide through by d and reduce the problem to the coprime case.

The general statement of the Chinese Remainder Theorem for moduli that are pairwise coprime, is as follows.

Chinese Remainder Theorem: *Let $m_1, m_2, \ldots, m_k$ be integers having the property that $\gcd(m_i, m_j) = 1$ whenever $i \neq j$. Then, for any integers $a_1, a_2, \ldots, a_k$, the system of simultaneous congruences $x \equiv a_i \pmod{m_i}$ has a solution k that is unique modulo $m_1 m_2 \cdots m_k$.*

Problem 4: *Find the integer N that is the residue of 2^{6754} modulo 1155.*

Solution. Observe first that $1155 = 11 \times 105 = 11 \times 7 \times 5 \times 3$. By Fermat's Little Theorem we know that $2^2 \equiv 1 \pmod{3}$, and hence $2^{2k} \equiv 1^k \equiv 1 \pmod{3}$ for all k. In particular, $2^{6754} \equiv 1 \pmod{3}$. Similarly Fermat's Little Theorem gives $2^4 \equiv 1 \pmod{5}$, and so $2^{4k} \equiv 1 \pmod{5}$ for all k. Now 6752 is a multiple of 4; so

$$2^{6754} = 2^{6752} 2^2 \equiv 2^2 \equiv 4 \pmod{5}.$$

Applying the same method again, $2^6 \equiv 1 \pmod{7}$, and $6754 = 6k + 4$ for some k gives

$$2^{6754} = 2^{6k} 2^4 \equiv 2^4 \equiv 16 \equiv 2 \pmod{7},$$

and $2^{10} \equiv 1 \pmod{11}$ and $6754 = 6750 + 4$ gives

$$2^{6754} = (2^{10})^{675}2^4 \equiv 2^4 \equiv 16 \equiv 5 \pmod{11}.$$

So by the form of the Chinese Remainder Theorem stated above, the number N that we seek is the unique residue mod 1155 satisfying the simultaneous congruences

$$\begin{aligned} N &\equiv 1 \pmod{3}, \\ N &\equiv 4 \pmod{5}, \\ N &\equiv 2 \pmod{7}, \\ N &\equiv 5 \pmod{11}. \end{aligned}$$

The first condition gives $N = 1 + 3S$ for some $S \in \mathbb{Z}$, and putting this into the next condition gives $3S \equiv 3 \pmod{5}$. Cancelling we conclude that $S = 1 + 5T$ for some $T \in \mathbb{Z}$, giving $N = 4 + 15T$. Substituting this into the next condition gives $4 + 15T \equiv 2 \pmod{7}$, or $T \equiv -2 \pmod{7}$. So $T = -2 + 7R$ for some $R \in \mathbb{Z}$, and $N = 4 + 15(-2 + 7R) = -26 + 105R$. Finally, putting this is the last condition yields $-26 + 105R \equiv 5 \pmod{11}$, or $6R \equiv 9 \pmod{11}$. Cancelling gives $2R \equiv 3 \equiv 14 \pmod{11}$, and cancelling again gives $R \equiv 7 \pmod{11}$. So $R = 7 + 11V$ for some V, and finally

$$N = -26 + 105(7 + 11V) = 709 + 1155V.$$

Thus the desired residue is in fact 709.

Week 5

§23 Multiplicative functions

(5.1) Definition: A function f defined on the positive integers is said to be *multiplicative* if $f(mn) = f(m)f(n)$ whenever $\gcd(m,n) = 1$.

We proved last week that the Euler phi function is multiplicative (and we prove again below). We shall meet some more examples of multiplicative functions shortly. But first we prove some straightforward things that we need.

(5.2) Proposition: *Let r, $s \in \mathbb{Z}$ and $d = \gcd(r,s)$. Then r/d and s/d are coprime integers.*

Proof. Since d is a common divisor of r and s it is certainly true that r/d and s/d are integers; so we just have to prove that they are coprime. Now if $e = \gcd(r/d, s/d)$ then $(r/d)/e = r/ed$ and $(s/d)/e = s/ed$ are both integers, and so ed is a common divisor of r and s. Hence ed is a divisor of $\gcd(r,s)$ (by condition (*ii*) in Definition (1.2): see Week 1). So $ed|d$, which means that $d/de = 1/e$ is an integer. Since $e = \gcd(r/d, s/d)$ is a positive integer by definition, it follows that $e = 1$, as required. □

(5.3) Proposition: *Let m, $n \in \mathbb{Z}$ with $\gcd(m,n) = 1$. If r, $s \in \mathbb{Z}$ with $r|m$ and $s|n$, then $\gcd(r,s) = 1$.*

Proof. Let $d = \gcd(r,s)$. Then d is a positive integer such that $d|m$ (since $d|r$ and $r|m$) and $d|n$ (since $d|s$ and $s|n$). It follows that $d|\gcd(m,n)$, and since $\gcd(m,n) = 1$ we conclude that $d = 1$, as required. □

(5.4) Proposition: *Let a, b be coprime integers that are both divisors of the integer k. Then ab is also a divisor of k.*

This is Lemma (4.5), proved last week in the lead up to the proof of the Chinese Remainder Theorem. The proof is very short, and the reader is urged to try to prove it without looking back at last week's notes.

We can now prove the following result, which should also seem intuitively reasonable.

(5.5) Proposition: *Let m, $n \in \mathbb{Z}$ with $\gcd(m,n) = 1$. Then every $k \in \mathbb{Z}^+$ that is a divisor of mn can be expressed uniquely as a product ab, where a, $b \in \mathbb{Z}^+$ and $a|m$ and $b|n$. Indeed, the factors a and b are given by $a = \gcd(k,m)$ and $b = \gcd(k,n)$.*

Proof. Let k be a divisor of mn. Let us first show that if $k = ab$ for some a, $b \in \mathbb{Z}^+$ such that $a|m$ and $b|n$, then a must equal $\gcd(k,m)$ and b must equal $\gcd(k,n)$.

Assume that we have such a factorization $k = ab$. Let $r = \gcd(k,m)$, and note first that since $a|k$ and $a|m$ it must certainly be true that $a|r$. But Proposition (5.3) above tells us that $\gcd(r,b) = 1$, since $r|m$ and $b|n$, and now since $r|k = ab$ and $\gcd(r,b) = 1$, it follows that $r|a$. Since $r|a$ and $a|r$ (and they are both positive integers) it follows that $a = r$.

A similar proof shows that $b = \gcd(k,n)$. It therefore remains to show that k does indeed have a factorization ab with $a|m$ and $b|n$.

Let $b = \gcd(k,n)$. Then k/b and n/b are integers, and since $k|mn$ it follows that $(k/b)|m(n/b)$. But $\gcd(k/b, n/b) = 1$, by Proposition (5.2) above, and so we conclude that $(k/b)|m$. Writing $a = k/b$, we have now found integers a and b such that $a|m$ and $b|n$, and $k = ab$, as required. □

Let us review our proof that $\phi(mn) = \phi(m)\phi(n)$ whenever m and n are coprime positive integers. The point is that the Chinese Remainder Theorem gives a one to one correspondence between the set

$$\{\, k \in \mathbb{Z} \mid 0 \le k < mn \,\}$$

and the set

$$\{\, (a,b) \in \mathbb{Z} \times \mathbb{Z} \mid 0 \le a < m \text{ and } 0 \le b < n \,\}$$

such that k corresponds to the pair (a,b) if and only if $k \equiv a \pmod{m}$ and $k \equiv b \pmod{n}$.* And it is not hard to show that in this correspondence, $\gcd(k,mn) = 1$ if and only if $\gcd(a,m) = 1$ and $\gcd(b,n) = 1$. (In fact, it is not hard to prove a little more than this, namely that $\gcd(k,mn) = \gcd(a,m)\gcd(b,n)$; we shall do this shortly.) Thus the set

$$\{\, k \in \mathbb{Z} \mid 0 \le k < mn \text{ and } \gcd(k,mn) = 1 \,\},$$

which has $\phi(mn)$ elements, is in one to one correspondence with the set

$$\{\, (a,b) \in \mathbb{Z} \times \mathbb{Z} \mid 0 \le a < m \text{ and } \gcd(a,m) = 1, \text{ and } 0 \le b < n \text{ and } \gcd(b,n) = 1 \,\},$$

which has $\phi(m)\phi(n)$ elements. Hence $\phi(mn) = \phi(m)\phi(n)$, as required.

Let us prove the parenthetic remark above.

(5.6) Proposition: *Let m and n be coprime positive integers. In the Chinese Remainder Theorem correspondence described above, if k corresponds to (a,b) then $\gcd(k,mn) = \gcd(a,m)\gcd(b,n)$.*

Proof. Since $h = \gcd(k,mn)$ is a divisor of mn, the previous proposition tells us that $h = \gcd(h,m)\gcd(h,n)$. So it will be sufficient to prove that $\gcd(h,m) = \gcd(a,m)$ and $\gcd(h,n) = \gcd(b,n)$.

---

* If A and B are any sets then $A \times B$ is defined to be the set $\{\, (a,b) \mid a \in A \text{ and } b \in B \,\}$. Thus $\mathbb{Z} \times \mathbb{Z} = \{\, (a,b) \mid a \in \mathbb{Z} \text{ and } b \in \mathbb{Z} \,\}$, the set of all ordered pairs of integers.

Let $d = \gcd(h,m)$, and note that $d|k$ since $h|k$. In other words, $k \equiv 0 \pmod{d}$. Moreover, since $d|m$ and $k \equiv a \pmod{m}$, it follows that $k \equiv a \pmod{d}$. Thus $a \equiv 0 \pmod{d}$; that is, $d|a$. Since we also have that $d|m$ (since $d = \gcd(h,m)$), it follows that $d|\gcd(a,m)$. That is, $\gcd(h,m)\,|\gcd(a,m)$.

To complete the proof that in fact $\gcd(h,m) = \gcd(a,m)$, we show that $\gcd(a,m)\,|\gcd(h,m)$. Write $e = \gcd(a,m)$, noting that $a \equiv 0 \pmod{e}$, and also that $k \equiv a \pmod{e}$ since $e|m$ and $k \equiv a \pmod{m}$. Thus $e|k$. But we also have that $e|m$ and $m|mn$; so $e|mn$. Hence $e|\gcd(k,mn) = h$, which combined with the fact that $e|m$ tells us that $e|\gcd(h,m)$. That is, $\gcd(a,m)\,|\gcd(h,m)$, as required.

The proof that $\gcd(h,n) = \gcd(b,n)$ is similar. □

Let f be any multiplicative function defined on $\mathbb{Z}^+$, and let n be a positive integer. By the Fundamental Theorem of Arithmetic we may write $n = p_1^{k_1}p_2^{k_2}\cdots p_r^{k_r}$ for some primes p_i and some positive integers k_i. (Note that when $n = 1$ the number r is zero, giving an empty product.) Repeated application of the fact that f is multiplicative yields

$$\begin{aligned}
f(n) &= f(p_1^{k_1})f(p_2^{k_2}\cdots p_r^{k_r}) && \text{since } \gcd(p_1^{k_1}, p_2^{k_2}\cdots p_r^{k_r}) = 1\\
&= f(p_1^{k_1})f(p_2^{k_2})f(p_3^{k_3}\cdots p_r^{k_r}) && \text{since } \gcd(p_2^{k_2}, p_3^{k_3}\cdots p_r^{k_r}) = 1\\
&= \cdots\\
&= f(p_1^{k_1})f(p_2^{k_2})f(p_3^{k_3})\cdots f(p_r^{k_r}),
\end{aligned}$$

so that the problem of finding a formula for $f(n)$ in terms of the factorization of n is reduced to the problem of finding a formula for $f(p^k)$ whenever p is prime and k is a positive integer.

In the case of the Euler phi function, it is easy to compute the value on prime powers. If p is prime and $k \in \mathbb{Z}^+$, then an integer m is coprime to p^k if and only if m is not divisible by p. There are exactly p^{k-1} natural numbers less than p^k that are multiples of p, namely the numbers ip for $i \in \{0,1,2,\ldots,p^{k-1}-1\}$. So there are $p^k - p^{k-1}$ natural numbers less than p^k and coprime to p^k. In other words,

$$\phi(p^k) = p^k - p^{k-1} = p^{k-1}(p-1) = p^k\Bigl(1 - \frac{1}{p}\Bigr).$$

Hence if $n = p_1^{k_1}p_2^{k_2}\cdots p_r^{k_r}$ as above, we find that

$$\begin{aligned}
\phi(n) &= p_1^{k_1}\Bigl(1-\frac{1}{p_1}\Bigr)p_2^{k_2}\Bigl(1-\frac{1}{p_2}\Bigr)\cdots p_r^{k_r}\Bigl(1-\frac{1}{p_r}\Bigr)\\
&= (p_1^{k_1}p_2^{k_2}\cdots p_r^{k_r})\Bigl(\Bigl(1-\frac{1}{p_1}\Bigr)\Bigl(1-\frac{1}{p_2}\Bigr)\cdots\Bigl(1-\frac{1}{p_r}\Bigr)\Bigr)\\
&= n\Bigl(\Bigl(1-\frac{1}{p_1}\Bigr)\Bigl(1-\frac{1}{p_2}\Bigr)\cdots\Bigl(1-\frac{1}{p_r}\Bigr)\Bigr).
\end{aligned}$$

It is common to use the Greek letter $\prod$ for products in the same way as $\sum$ is used for sums. Thus, for example, $\prod_{i=1}^{n} x_i$ denotes $x_1x_2\cdots x_n$. Using this convention, the formula for $\phi(n)$ becomes

$$\phi(n) = n\prod_{p|n}\Bigl(1-\frac{1}{p}\Bigr)$$

where the product is over all primes p that divide n.

(5.7) ***Example:***

1) There are exactly two prime divisors of 100, namely 2 and 5. So it follows that $\phi(100) = 100(1-\frac{1}{2})(1-\frac{1}{5}) = 100(\frac{1}{2})(\frac{4}{5}) = 40$.
2) The prime divisors of 60 are 2, 3 and 5. So $\phi(60) = 60(\frac{1}{2})(\frac{2}{3})(\frac{4}{5}) = 16$.
3) Let us confirm that $\phi(60) = 16$ by writing down all the natural numers less than 60 and coprime to 60, and confirming that there are exactly sixteen of them. In fact they are $1, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 49, 53, 59$.

One consequence of the above formula for $\phi(n)$ is that the only obvious way to compute $\phi(n)$ is to find all the prime divisors of n. Indeed, if we know somehow that n is the product of two distinct primes p and q, then the problem of finding p and q given n, and the problem of finding $\phi(n)$ given n, really are equivalent. On the one hand, if we know p and q then we can certainly calculate $\phi(n)$ by the above formula; indeed

$$\phi(n) = pq\Bigl(1-\frac{1}{p}\Bigr)\Bigl(1-\frac{1}{q}\Bigr) = (p-1)(q-1) = pq-(p+q)+1.$$

On the other hand, this formula tells us that $\phi(n) = n-(p+q)+1$; so if we know $\phi(n)$ and n then the equations

$$\begin{aligned} p+q &= n+1-\phi(n) \\ pq &= n \end{aligned}$$

mean that we know all the coefficients of the polynomial $x^2-(p+q)x+pq$. We can therefore compute p and q, since they are the roots of this quadratic.

The security of the RSA cryptosystem depends on the fact that there is no fast algorithm known for computing $\phi(n)$ if n is the product of two sufficiently large prime numbers. As we have just shown, the problem of factorizing n is essentially the same problem.

§24 The number of divisors and the sum of the divisors

(5.8) Definition: Let n be a positive integer. We define $\tau(n)$ to be the number of positive integers d such that $d|n$, and we define $\sigma(n)$ to be the sum of all the positive integers d such that $d|n$.

Let m and n be coprime positive integers, and define

$$S = \{\,(a,b) \in \mathbb{Z}\times\mathbb{Z} \bigm| a|m \text{ and } b|n\,\}$$

and

$$T = \{\,k \in \mathbb{Z} \bigm| k|mn\,\}.$$

Propositions (5.3) and (5.4) in §23 above tell us that if $(a,b) \in S$ then $\gcd(a,b) = 1$ and therefore $ab|m$. Consequently $G(a,b) = ab$ defines a function $G\colon S \to T$. Proposition (5.5) from §23 tells us that for each $k \in T$ there exist $(a,b) \in S$ such that $k = ab$; this tells us that the function G is onto. Furthermore, the same proposition tells us that the pair (a,b) is uniquely determined by k, so that G is also one to one. In other words, we have a one to one correspondence between the sets T and S.

The number of elements of the set T is $\tau(mn)$, by the definition of the function τ. Similarly, the number of elements of S is $\tau(m)\tau(n)$, since there is exactly one element of S for each choice of a divisor a of m and a divisor b of n. Thus we have shown that $\tau(mn) = \tau(m)\tau(n)$ whenever $\gcd(m,n) = 1$. In other words, the function τ is multiplicative.

Very similar reasoning shows that the function σ is also multiplicative. If m, n, S and T are as above, then by definition $\sigma(mn) = \sum_{k\in T} k$. But the one to one correspondence between T and S means that each $k \in T$ corresponds to a unique $(a,b) \in S$ such that $k = ab$, and so

$$\sigma(mn) = \sum_{(a,b)\in S} ab = \sum_{a|m}\sum_{b|n} ab = \Bigl(\sum_{a|m} a\Bigr)\Bigl(\sum_{b|n} b\Bigr),$$

since every product ab appears exactly once when this last expression is expanded. Thus $\sigma(mn) = \sigma(m)\sigma(n)$, and σ is multiplicative.

(5.9) *Example:* To illustrate the above proofs, consider the coprime integers 9 and 14. The set of positive integers that are divisors of 9 is $\{1,3,9\}$, and the set of positive integers that are divisors of 14 is $\{1,2,7,14\}$. Thus $\tau(9) = 3$ and $\tau(14) = 4$, and $\tau(9\times 14)$ should be 3×4. It is, because one obtains all the divisors of 9×14, without any repetitions, by multiplying every element of $\{1,3,9\}$ by every element of $\{1,2,7,14\}$. The twelve numbers obtained are 1, 3, 9, 2, 6, 18, 7, 21, 63, 14, 42 and 126, and is is readily checked that these are all the divisors of 126. Similarly, since

$$(1+3+9)(1+2+7+14) = (1+3+9+2+6+18+7+21+63+14+42+126),$$

we confirm that $\sigma(9)\sigma(14) = 13\times 24 = 312 = \sigma(126)$.

As we have seen, it is always possible to find a formula for the value at n of any multiplicative function in terms of the prime factorization of n, provided only that one can deal with the case that n is a prime power. It is clear that if k is a positive integer and $n = p^k$, where p is prime, then the divisors of n are precisely the powers p^i for $i \in \{0, 1, 2, \ldots, k\}$. The number of these is $k + 1$, and their sum is $(p^{k+1} - 1)/(p - 1)$ (since they comprise a geometric series with first term 1 and common ratio p). Thus we find that if $n = p_1^{k_1} p_2^{k_2} \cdots p_r^{k_r}$ is the factorization of n, then

$$\tau(n) = \prod_{i=1}^{r} (k_i + 1)$$

and

$$\sigma(n) = \prod_{i=1}^{r} \frac{p_i^{k_i+1} - 1}{p_i - 1}.$$

§25 Perfect numbers and Mersenne primes

Pythagoras and his followers considered a number to be in some way magical if it is the sum of all its divisors (excluding itself). For example, the divisors of 6 (excluding 6) are 1, 2 and 3, and we see that $1 + 2 + 3 = 6$. Another example is $28 = 1 + 2 + 4 + 7 + 14$.

The fact that prime numbers are discussed in Euclid's *Elements* is undoubtedly due to this Greek fascination with these so-called *perfect numbers*.

Reformulating the concept in terms of the function σ, we make the following definition.

(5.10) Definition: A positive integer n is said to be *abundant* if $\sigma(n) > 2n$, *deficient* if $\sigma(n) < 2n$, and *perfect* if $\sigma(n) = 2n$.

It is not known if there are any odd perfect numbers. It is known, however, that any odd perfect number must be greater than 10^{300}, have at least eight distinct prime divisors, have at least 37 prime divisors counting multiplicities, and have at least one prime factor greater than 10^{20}. Idle doodling with pen and paper will probably not find one for you.

It is, however, possible to give some more positive information about even perfect numbers.

Indeed, suppose that N is an even perfect number, and let 2^k be the highest power of 2 dividing N. Thus $N = 2^k M$, where $k \geq 1$ and M is odd. Then $\gcd(2^k, M) = 1$, and so the multiplicative property of σ tells us that

$$\sigma(2^k)\sigma(M) = \sigma(2^k M) = \sigma(N) = 2N = 2^{k+1}M,$$

where the equation $2N = \sigma(N)$ comes from the fact that N is perfect. Furthermore, we know that $\sigma(2^k) = \sum_{i=0}^{k} 2^i = 2^{k+1} - 1$, and so it follows that

$$(2^{k+1} - 1)\sigma(M) = 2^{k+1}M. \qquad (9)$$

Consecutive numbers are always coprime; so $\gcd(2^{k+1}-1, 2^{k+1}) = 1$. Since the above equation tells us that $(2^{k+1}-1) \mid 2^{k+1}M$, we deduce that $(2^{k+1}-1) \mid M$. So we can write $M = (2^{k+1}-1)Q$ for some positive integer Q.

Putting this into Eq. (9) yields, after cancelling $2^{k+1}-1$,

$$\sigma(M) = 2^{k+1}Q,$$

and adding Q to both sides of the equation $M = (2^{k+1}-1)Q$ gives

$$M + Q = 2^{k+1}Q.$$

These last two equations tell us that $\sigma(M) = M + Q$. But M and Q are divisors of $M = (2^{k+1}-1)Q$, and $M > Q$ since $k \geq 1$. If $Q \neq 1$ then 1 is another divisor of M distinct from both M and Q, and this would mean that $\sigma(M) \geq M + Q + 1$, contradicting $\sigma(M) = M + Q$. We conclude that $Q = 1$, and $\sigma(M) = M + 1$. Moreover, this means that M and 1 are the only divisors of M; so M is prime.

Putting all this together, we have shown that if N is an even perfect number then $N = 2^k(2^{k+1}-1)$, and the number $2^{k+1}-1$ is prime.

It is also easy to see that the converse statement is true: if $2^{k+1}-1$ is prime then $N = 2^k(2^{k+1}-1)$ is perfect. Indeed,

$$\begin{aligned}
\sigma(N) &= \sigma(2^k)\sigma(2^{k+1}-1) && (\text{since } \gcd(2^k, 2^{k+1}-1) = 1)\\
&= (2^{k+1}-1)(1 + (2^{k+1}-1)) && (\text{since } 2^{k+1}-1 \text{ is prime})\\
&= 2^{k+1}(2^{k+1}-1)\\
&= 2N
\end{aligned}$$

as required.

The algebraic identity $(x^k - 1) = (x-1)(x^{k-1} + x^{k-2} + \cdots + x + 1)$ shows that $m-1$ is a divisor of $m^k - 1$ for all integers m and k. Putting $m = a^d$, we conclude that $a^d - 1$ is always a divisor of $a^{dk} - 1$. In particular, $a^m - 1$ is divisible by $a^d - 1$ whenever $d \mid m$. This means that it is very hard for a number of the form $a^m - 1$ to be prime. Indeed, if $a > 2$ then $a^m - 1$ can never be prime: it is divisible by $a - 1$. If $a = 2$ then it is possible for $a^m - 1 = 2^m - 1$ to be prime, but only if m is prime. If m is not prime then there is a number $d > 1$ such that $d \mid m$ and this means that $2^d - 1$ is a nontrivial proper divisor of $2^m - 1$.

Here is a table giving the values of $2^m - 1$ for the first few values of m.

m	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$2^m - 1$	0	1	3	7	15	31	63	127	255	511	1023	2047	4095	8191

The thing to notice about the numbers in the second row is that factors recur at regular intervals. For example, 3 occurs first as a factor of $2^m - 1$ when $m = 2$ (excluding $m = 0$), and after that 3 occurs as a factor whenever m is a multiple of 2. Similarly, 7 first appears as a factor when $m = 3$, and it occurs again at every multiple of 3. In the terminology we introduced in Week 2 (see Definition (2.17)), this says that 3 is the order of 2 modulo 7: the number 3 is the least positive integer m such that $2^m \equiv 1 \pmod 7$. From the table above we can determine the order of 2 modulo various other primes: $\text{ord}_5(2) = 4$, $\text{ord}_{31}(2) = 5$, $\text{ord}_{127}(2) = 7$, $\text{ord}_{17}(2) = 8$, $\text{ord}_{11}(2) = 10$, $\text{ord}_{13}(2) = 12$, $\text{ord}_{8191}(2) = 13$. A quick inspection shows that for every prime value of m in our little table, the number $2^m - 1$ is also prime. But a more careful inspection reveals that actually there is an exception to this: the number $2^{11} - 1 = 2047$ is not prime, but equals 23×89. We should have added $\text{ord}_{23}(2) = \text{ord}_{89}(2) = 11$ to our list.

Notice that Fermat's Little Theorem guarantees that $a^{p-1} \equiv 1 \pmod p$ whenever $p \nmid a$. So $\text{ord}_p(a)$, which is the least positive integer m such that p occurs as a factor of $a^m - 1$, is bound to be a divisor of $p - 1$. Fermat guarantees that p occurs as a factor in $a^{p-1} - 1$, and since the values of m for which p is a factor of $a^m - 1$ are precisely the multiples of the order, we can be sure that $p - 1$ is a multiple of the order.

In particular, $\text{ord}_{23}(2)$ had to be a divisor of 22, and, sure enough, 11 is a divisor of 22. Likewise $11 = \text{ord}_{89}(2)$ is a divisor of 88. Similarly, if m is any prime then any prime divisor of $2^m - 1$ will have to be of the form $km + 1$ for some integer k.

(5.11) Definition: Numbers of the form $2^m - 1$, where m is prime, are called *Mersenne numbers*. A prime Mersenne number is called a *Mersenne prime*.

Despite the fact that our little table above shows five Mersenne primes, there are actually not many Mersenne primes known. In fact, there are just 43 of them (including, incidentally, the largest prime known). But of course the numbers $2^m - 1$ get very big very quickly, as m increases; so it is perhaps not surprising that only 43 Mersenne primes have been found so far. In fact it is widely believed that infinitely many Mersenne primes exist.

PART 5A: *Statistical attacks on classical cipher systems*

We consider a message $M = c_1c_2c_3 \ldots c_N$, regarded as a sequence of letters c_n from an alphabet $\mathscr{A}$, which we identify with residues mod α. That is, $\mathscr{A} = \{0, 1, 2, \ldots, \alpha - 1\}$. For the present, in fact, we consider only the case $\alpha = 26$, identifying 0 with A, 1 with B, and so on, and we may pass back and forth from numbers to letters without comment. Usually M will be the plaintext, and the corresponding ciphertext will be denoted by $M' = c'_1c'_2c'_3 \ldots c'_N$. For this week at least, M' will have the same length as M.

For each $i \in \mathscr{A}$ the number of occurrences of i in the message M is the number of elements in the set $\{n \mid 1 \le n \le N \text{ and } c_n = i\}$. We use the notation $\#S$ for the number of elements of a set S; so the number of occurrences of i in M is $\#\{n \mid 1 \le n \le N \text{ and } c_n = i\}$. The *relative frequency* p_i of the letter i is the number of occurrences of i divided by the total length of M; that is,

$$p_i = \frac{\#\{\, n \mid 1 \le n \le N \text{ and } c_n = i \,\}}{N}.$$

We see that $0 \le p_i \le 1$ for all i, and $\sum_{i=1}^{l} p_i = 1$. A family of numbers satisfying these two conditions is commonly called a *probability distribution*. If we imagine choosing at random one of the letters c_n that make up the message $M = c_1c_2c_3 \ldots c_N$, with each c_n having probability $1/N$ of being chosen, then the probability that the chosen letter c_n is an i is equal to p_i, the relative frequency of i in M.

Whenever we speak of "choosing a random letter from M", we mean it the sense just described: all of the letters $c_1, c_2, \ldots, c_N$ have probability $1/N$ of being chosen. When we speak of choosing two letters at random, we shall always assume that the two choices are made independently of each other. Furthermore, we shall always assume that the first letter is replaced before the second one is chosen (so that in principle the same one could be chosen twice).

§26 Substitution ciphers

For simple substitution ciphers, the ciphertext $M' = c'_1c'_2c'_3 \ldots c'_N$ is obtained from the plaintext $M = c_1c_2c_3 \ldots c_N$ by the rule that $c'_n = \pi(c_n)$ where $\pi\colon \mathscr{A} \to \mathscr{A}$ is some permutation. This permutation π is called the substitution key. If p'_j denotes the relative frequency of j in M' then clearly $p'_{\pi(i)} = p_i$, since the number of occurrences of $\pi(i)$ in M' is exactly the number of occurrences of i in M. One statistic that is not changed by a substitution cipher is the probability that two randomly chosen letters in the message are the same. This quantity, which is known as the *coincidence index*, equals $\sum_{i\in\mathscr{A}} p_i^2$, since for each $i \in \mathscr{A}$ the probability that two randomly chosen letters are both i is p_i^2. Observe that if π is a permutation of $\mathscr{A}$ then as i runs through all elements of $\mathscr{A}$, so too does $j = \pi(i)$ (in a different order). Thus

$$\sum_{j\in\mathscr{A}} (p'_j)^2 = \sum_{i\in\mathscr{A}} (p'_{\pi(i)})^2 = \sum_{i\in\mathscr{A}} p_i^2,$$

showing that the ciphertext and plaintext have the same coincidence index. For typical unencrypted English text the coincidence index is usually about 0.065.

Of course if one computes the coincidence index for some ciphertext and finds that it is roughly 0.065, it does not prove that a substitution cipher has been used, but it is nevertheless a reasonable indication that a substitution cipher has perhaps been used, and one should investigate the possibility further. The natural next step for the cryptanalyst would be to sort the numbers p'_j into decreasing order, and see if the resulting sequence is roughly the same as that which typical unencrypted text would yield. Specifically, the largest p'_j should be about 0.13 or 0.12, matching the relative frequency of E in typical text, the next two should be round about 0.095 to

0.085, matching T and A, followed by another six in the range 0.075 to 0.06, and so on. If one then simply replaces the most frequently occurring letter by E, the next by T, then A, O, S, I, N, H, R, L, D, M, U, C, G, Y, F, W, P, B, V, K, J, X, Q, Z, one might hope that the resulting text M'' will be recognizably close to the plaintext.

In practice this does not work all that well, because the relative frequencies do vary from one piece of text to another. Sometimes A is more frequent than T, sometimes O is more frequent than A, and so on. We can improve things by taking into account the fact that some digraphs (pairs of adjacent letters) are much more frequent than others. For example, the most common English digraphs are TH, with a relative frequency of about 0.047, and HE, with a relative frequency of about 0.038, while some other digraphs almost never occur. After obtaining M'' in the manner described above, one should attempt to find a permutation that maps each letter to a letter with similar frequency and at the same time makes the digraph frequency distribution of the resulting text more closely match that of typical English text. The total number of permutations of 26 letters is too enormous to handle, but the number that only take letters to letters with similar frequencies is quite manageable.

For each digraph ij define p''_{ij} to be the relative frequency of ij in M'' and p_{ij} the relative frequency of ij in the plaintext M. Then $p''_{\sigma(i)\sigma(j)} = p_{ij}$, where σ is the permutation of $\mathscr{A}$ that transforms M into M''. Choose a sample of typical English text for comparison, and for each ij let s_{ij} be the relative frequency of ij in the sample. Then $|p_{ij} - s_{ij}|$ will be small for all i and j, and so $\sum_{i,j} |p_{ij} - s_{ij}|^2$ will also be small. So one should search for a permutation σ such that $\sum_{i,j} |p''_{\sigma(i)\sigma(j)} - s_{ij}|^2$ is small, searching through the not-too-large list of permutations that only take letters to letters with similar frequencies. When one has found the σ that minimizes the above sum and used it to again modify the ciphertext, the result will probably be almost readable, and any remaining permutation of $\mathscr{A}$ will be easy to find.

§27 Friedman's attack on the Vigenère cipher

As above we assume that our plaintext message M is a string $c_1c_2c_3\dots c_N$ of letters from the alphabet $\mathscr{A} = \{0, 1, \dots, \alpha-1\}$. The Vigenère system involves choosing a key $K = k_1k_2\dots k_m$, where $k_i \in \mathscr{A}$ for each i. The integer m is called the *period*.Extend K to an infinite sequence by defining $k_{m+1} = k_1$, $k_{m+2} = k_2$, and so on. (In general, if i is a positive integer and j is the residue of i modulo m, then k_i is defined to be equal to k_j.) The enciphered message is $M' = c'_1c'_2\dots c'_N$, defined by the rule that $c'_n \equiv c_n + k_n \pmod{\alpha}$.

The *decimation of M with period r and index e* is defined to be the sequence

$$\mathrm{Dec}(M, e, r) = c_e c_{e+r} c_{e+2r} c_{e+3r} \dots$$

obtained by taking every rth letter of M, starting at the eth. Usually these decimations of M give letter frequency distributions that are similar to that of M

itself. In particular, the coincidence index for a decimation of M will usually be 0.06 or more.

Now consider decimations of the ciphertext. A decimation whose period r is equal to m, the length of the keyword, is simply a translation, or alphabetic shift, of the corresponding decimation of M, since the encryption process uses the same letter of the keyword for all the terms in this decimation. That is, since $k_e = k_{e+m} = k_{e+2m} = \cdots$, it follows that for all q,

$$c'_{e+qm} \equiv c_{e+qm} + k_{e+qm} = c_{e+qm} + k_e \pmod{\alpha}.$$

The number of occurrences of any the letter i in $\mathrm{Dec}(M, e, m)$ is therefore the same as the number of occurrences of the letter $j = i + k_e$ in $\mathrm{Dec}(M', e, m)$. (To be more precise, j is the residue of $i + k_e$ mod α.) Hence the letter frequency distribution for $\mathrm{Dec}(M', e, m)$ should be typical of the letter frequency distribution for ordinary text, except that each letter has been replaced the letter that is k_e places further on in the alphabet. And, in particular, the coincidence index for $\mathrm{Dec}(M', e, m)$ should be typical of ordinary text, taking a value round about 0.065. On the other hand, a decimation of period r that is not a multiple of n will have a more even distribution of the letters, and a lower value for the coincidence index.

So Friedman's method of decrypting Vigenère ciphertext is to compute the coincidence index for decimations of periods 2, 3, 4, . . . , until we find one that is greater than 0.06, or thereabouts. Then the frequency distributions are computed for all the decimations of this period. They should all turn out to be shifted versions of something typical of ordinary text. The amount of each shift then determines the corresponding letter of the keyword. This is very easy to determine, since usually it just involves matching the most frequent letter in the decimation with the most frequently used letter of ordinary text (namely E in English).

The weakness of the Vigenère system is that the periodicity is statistically detectable, at least if the cryptanalyst is able to obtain decimations of the ciphertext that are long enough to give reliable frequency distributions for the various letters. The longer the keyword the harder it becomes for the cryptanalyst. In the extreme case that the keyword is longer than the message itself there is no statistical information to be had. Indeed, there is no information of any kind to be had: the only way to decrypt such a message is to know the key (or be able to guess it).

§28 One time pads

The one time pad system is simply a Vigenère system with a key that is used only once and only for a message that is shorter than the key itself. The key should be generated by some random process, and of course should only be known by the person who intends to send the message and intended recipient. Any statistical properties of the plaintext will be completely destroyed by the randomness of the key, making the ciphertext appear just as random as the key. There is no way to decrypt such a message without access to the key.

Of course, the problems with the one time pad are that the key has to be long enough to cope with any message that might be sent, and may have to be generated and kept secret for a long time before it is used. The problem of securely transmitting a key to a remote person is just as hard as the problem of securely sending them a message.*

§29 Block transposition ciphers

In block transposition cipher systems the plaintext $M = c_1c_2c_3 \ldots c_N$ is split into blocks of some fixed length m (so that the first block is $c_1c_2 \ldots c_m$, the second block is $c_{m+1}c_{m+2} \ldots c_{2m}$, and so on), and then the letters in each block are rearranged using the same permutation of $\{1, 2, \ldots, m\}$. The key is the permutation used. For example, if the block length m is 5 and the key is $[3, 5, 2, 1, 4]$ then the ciphertext will be

$$M' = c_3c_5c_2c_1c_4c_8c_{10}c_7c_6c_9c_{13}c_{15}c_{12}c_{11}c_{14}c_{18}c_{20}c_{17}c_{16}c_{19} \ldots .$$

In other words, writing $M' = c'_1c'_2c'_3 \ldots c'_N$ as usual, this says that

$$\begin{aligned} c'_1 &= c_3, \\ c'_2 &= c_5, \\ c'_3 &= c_2, \\ c'_4 &= c_1, \\ c'_5 &= c_4, \end{aligned}$$

and so on. In general, the key must have the form $[k_1, k_2, \ldots, k_m]$, where k_1, k_2, …, k_m are the numbers 1, 2, …, m in some order, and enciphering is done by reordering the letters in each block so that k_nth letter in the original (plaintext) block becomes the nth letter in the new (ciphertext) block, for all values of n from 1 to m. Thus the first block of ciphertext is

$$c'_1c'_2 \ldots c'_m = c_{k_1}c_{k_2} \ldots c_{k_m},$$

the second block is

$$c'_{m+1}c'_{m+2} \ldots c'_{2m} = c_{m+k_1}c_{m+k_2} \ldots c_{m+k_n},$$

and so on. Indeed, if $1 \leq n \leq m$ and q is any natural number, then $c'_{qm+n} = c_{qm+k_n}$.

The message recipient recovers the plaintext from the ciphertext by exactly the same process, using the inverse key $[h_1, h_2, \ldots, h_m]$ in place of the key $[k_1, k_2, \ldots, k_m]$. The key and inverse key are related as follows: $h_n = r$ if and only if $k_r = n$ (for all

* This is not really true. Since the key is going to be a random sequence of letters, it could be encrypted by (say) a simple substitution cipher and be immune to the ciphertext-only attacks that rely on statistical properties of the message.

n, $r \in \{1, 2, \dots, m\}$). The inverse key for the example given above is $[4, 3, 1, 5, 2]$, and

$$M = c'_4c'_3c'_1c'_5c'_2c'_9c'_8c'_6c'_{10}c'_7c'_{14}c'_{13}c'_{11}c'_{15}c'_{12}c'_{19}c'_{18}c'_{16}c'_{20}c_{17}\dots\ .$$

expresses the plaintext in terms of the ciphertext.

As is the case with all transposition ciphers, this system does not alter the frequency distribution of the letters. The whole ciphertext M' is just some reordering of the plaintext M, and so each letter still occurs the same number of times in M' as it did in M. (In fact it is reasonable to presume that a ciphertext in which the relative frequencies of the various letters match their typical frequencies in ordinary text was created with a transposition cipher.) But the digraph frequency distribution does change. Common plaintext digraphs such as TH will be split up by a transposition cipher, and digraphs such as QQ that never occur in the plaintext could easily occur in the ciphertext. A sensible strategy for the cryptanalyst would be to try to put the correct digraphs back together, by finding a reordering the ciphertext that has a digraph frequency distribution matching that of typical unencrypted text. But this is easier said than done!

As with the Vigenère system, the periodicity of a block transposition system is a weakness that can be exploited by statistical analyses. The longer the blocks, the harder this becomes.

When confronted with a ciphertext M' that we believe was encrypted by a block transposition system, our first task is to find the block length. We do this by successively trying all the possibilities—2, then 3, then 4, and so on—by applying various statistical tests, until we find a plausible block length. To test if m is a plausible block length we examine the *$[k, l]$-decimation of M' of period m* for various k and l from the set $\{1, 2, \dots, m\}$. This decimation is the following sequence of pairs:

$$\mathrm{Dec}(M', [k, l], m) = [c'_kc'_l,\ c'_{m+k}c'_{m+l},\ c'_{2m+k}c'_{2m+l},\ c'_{3m+k}c'_{3m+l},\ \dots]. \qquad (10)$$

The idea is that if the decimation period m equals the block length, and if the numbers k and l are consecutive terms of the inverse key—say $k = h_n$ and $l = h_{n+1}$—then $c'_k = c'_{h_n} = c_n$ and $c'_l = c'_{h_{n+1}} = c_{n+1}$ were adjacent to each other in the plaintext, and so were $c'_{qm+k} = c_{qm+n}$ and $c'_{qm+l} = c_{qm+n+1}$ for all values of q. Thus for the example we considered above, in which the block length is 5 and the inverse key is $[4, 3, 1, 5, 2]$, the $[1, 5]$-decimation of M' of period 5 is

$$\begin{aligned}\mathrm{Dec}(M', [1, 5], 5) &= [c'_1c'_5,\ c'_6c'_{10},\ c'_{11}c'_{15},\ c'_{16}c'_{20},\ c'_{21}c'_{25},\ c'_{26}c'_{30},\ \dots] \\ &= [c_3c_4,\ c_8c_9,\ c_{13}c_{14},\ c_{18}c_{19},\ c_{23}c_{24},\ c_{28}c_{29},\ \dots]\end{aligned}$$

which is a sequence of digraphs from the original plaintext. So, in principle, it should include a lot of the digraphs that are common in unencrypted text, and not many of the uncommon ones.

In general, if m is the block length and $[h_1, h_2, \dots, h_m]$ is the inverse key, then for all n such that $1 \le n < m$,

$$\begin{aligned}\mathrm{Dec}(M', [h_n, h_{n+1}], m) &= \mathrm{Dec}(M, [n, n+1], m) \\ &= [c_nc_{n+1}, c_{m+n}c_{m+n+1}, c_{m+n}c_{m+n+1}, \dots\ ]\end{aligned}$$

is a sequence of plaintext digraphs. As we shall see, there are various statistics that we can compute to help us decide whether or not a given sequence of pairs of letters is likely to be a sequence of digraphs from a piece of unencrypted text. If we happen upon a decimation of the ciphertext M' that statistically seems likely to be a sequence of plaintext digraphs, then there is a good chance that the period of the decimation is the block length of the cipher, and the pairs of letters that make up the decimation were adjacent pairs of letters in the plaintext.

Given any text composed of letters from the alphabet $\mathscr{A}$, the *digraph coincidence index* of the text is defined to be $\sum p_{ij}^2$, where the sum is over all i and j in $\mathscr{A}$, and p_{ij} is the relative frequency of ij as a digraph in the text. It is the probability that two randomly chosen digraphs are equal. If the text is $s_1s_2s_3 \dots s_L$, where L is the length of the text, then there are $L-1$ digraphs altogether, namely s_1s_2, s_2s_3, $\dots$, $s_{L-1}s_L$. If ij occurs F_{ij} times in this list then by definition $p_{ij} = F_{ij}/(L-1)$. So if we imagine randomly choosing a digraph s_ns_{n+1} from the above list, all terms having equal probability of being chosen, then p_{ij} is the probability that the chosen one equals ij. If we do this twice then the probability that a particular ij is chosen twice is p_{ij}^2 (assuming that the two choices are made independently), and summing this over all possibilities for ij gives the probability that the two choices yield digraphs that are the same.

Note that the digraph coincidence index of the text $s_1s_2s_3 \dots s_L$ is the same thing as the coincidence index of the sequence $[s_1s_2, s_2s_3, s_3s_4, \dots, s_{L-1}s_L]$: it is the probability that two randomly chosen terms of this sequence are equal.

For typical English text, the digraph coincidence index is usually about 0.007, or more.* Of course different texts do give different values, and it is always possible that some particular text may give an unusually low value for the digraph coincidence index. In general, the longer the given text is, the more likely it is to yield reliable statistical information.

The minimum possible value for the digraph coincidence index of a text composed of letters from an alphabet of 26 letters is $(1/26)^2 \approx 0.00148$, occurring when all 26^2 digraphs have the same frequency in the text. Applying a transposition cipher to typical English text will reduce the digraph coincidence index, but still some digraphs will be significantly more frequent than others, just because some letters are more frequent than others. Digraphs containing an E will be far more common than digraphs containing an X. So the digraph coincidence of the ciphertext will never be very close to 0.00148. It seems that 0.0045 is fairly typical.

If we now consider any sequence of randomly selected digraphs from our given text, then we can expect that the relative frequencies of the various digraphs in the sequence will be similar to their relative frequencies in the text overall. So, in particular, if we consider the $[n, n+1]$ decimation of M of period m, and for each pair of letters ij we let p_{ij} be the relative frequency of ij in the decimation,

* Edgar Allen Poe's short story "The Black Cat" gave the value 0.0072409. Conan-Doyle's "The Red-Headed League" gave 0.0072464.

then (assuming that the decimation is long enough) p_{ij} is likely to be a good approximation to the relative frequency of ij in the whole of M, and the coincidence index $\sum_{ij} p_{ij}^2$ is likely to be reasonably close to the digraph coincidence index of M.

So, given only the ciphertext M', but suspecting that it comes from a block transposition cipher, we can test whether a positive integer $m \geq 2$ is a plausible block length by computing the coincidence indices of the decimations $\mathrm{Dec}(M', [1,l], m)$ for all values of l from 2 to m. If any one of these has a coincidence index of about 0.007, or more, then we should strongly suspect that m is the block length, and that 1 and l were adjacent terms of the inverse key.

There is a minor point that has been glossed over in the above discussion. Suppose that for each pair of letters ij we define q_{ij} to be the relative frequency of ij as a reversed digraph in our given text, $s_1 s_2 s_3 \ldots s_L$. In other words, $q_{ij} = G_{ij}/(L-1)$, where G_{ij} is the number of times that i follows j in the text. Of course $q_{ij} = p_{ji}$, the relative frequency of the digraph ji, and so the *reversed digraph coincidence index* $\sum_{ij} q_{ij}^2$ is exactly the same as the digraph coincidence index $\sum_{ij} p_{ij}^2$. This means that the digraph coincidence index test cannot distinguish between digraphs and reversed digraphs. If we take a sequence of digraphs from our given text, and reverse them all, then the coincidence index of the sequence will not change, but rather than many TH's, HE's and ND's, there will be many HT's, EH's and DN's. So a sequence of pairs having coincidence index value of about 0.007 is just as likely to be a sequence of reversed digraphs as a sequence of digraphs.

Thus, returning again to the example we considered above, with block length 5 and inverse key $[4,3,1,5,2]$, we find that

$$\begin{aligned}\mathrm{Dec}(M', [1,3], 5) &= [c'_1c'_3,\ c'_6c'_8,\ c'_{11}c'_{13},\ c'_{16}c'_{17},\ c'_{21}c'_{23},\ c'_{26}c'_{28},\ \ldots] \\ &= [c_3c_2,\ c_8c_7,\ c_{13}c_{12},\ c_{18}c_{17},\ c_{23}c_{22},\ c_{28}c_{27},\ \ldots] \\ &= \mathrm{Dec}(M, [3,2], 5),\end{aligned}$$

a sequence of reversed digraphs of M rather than a sequence of digraphs of M. Its coincidence index should approximate the digraph coincidence index of M.

In general, if $\mathrm{Dec}(M', [k,l], m)$ has a coincidence index that is about 0.007, it is reasonable to conclude that m is the block length and that k and l are adjacent in the inverse key, but whether k precedes l in the inverse key (as in the case $[k,l] = [1,5]$ with the inverse key $[4,3,1,5,2]$) or l precedes k (as in the case $[k,l] = [1,3]$ with the inverse key $[4,3,1,5,2]$), we cannot say. But this is not really a serious drawback. If we can find all the pairs of letters that are adjacent in the inverse key, without knowing in which order they occur, then there will be exactly two possibilities for the inverse key, and it will be easy to try them both and see which one works. For example, if $m = 5$ and we have found that pairs of adjacent letters in the inverse key are $\{1,3\}$, $\{1,5\}$, $\{2,5\}$ and $\{3,4\}$, then it follows that the inverse key must be either $[4,3,1,5,2]$ or its reverse, $[2,5,1,3,4]$. We can apply both of these to the ciphertext and see which one produces a readable result.

Another statistic that may sometimes be useful is the *coincidence discriminant* of a text. For each $i \in \mathscr{A}$ define $p_{i\,\_}$ to be $\sum_{h \in \mathscr{A}} p_{ih}$, the relative frequency in the text of digraphs that begin with i. Similarly, for each $j \in \mathscr{A}$ define $p_{\_j}$ to be $\sum_{h\in\mathscr{A}} p_{hj}$, the relative frequency of digraphs that end with j. The coincidence discriminant of the text (or CD for short) is defined to be $\sum_{ij}(p_{ij} - p_{i\_}p_{\_j})^2$.

The rationale for this definition is as follows. If the event that a randomly chosen digraph has i as its first letter and the event that a randomly chosen digraph has j as its second letter are independent of each other, then the event that a randomly chosen digraph has i as its first letter and j as its second letter is $p_{i\_}p_{\_j}$. In other words, $p_{ij} = p_{i\_}p_{\_j}$. Of course it is unlikely in practice that p_{ij} will be exactly equal to $p_{i\_}p_{\_j}$, but one might expect that p_{ij} and $p_{i\_}p_{\_j}$ will always be close to each other, making $(p_{ij} - p_{i\_}p_{\_j})^2$ close to zero in all cases, and so making $\sum_{ij}(p_{ij} - p_{i\_}p_{\_j})^2$ rather small. But in fact ordinary text is not like this. If, for given letters i and j, it is the case that $p_{i\_}p_{\_j}$ approximates p_{ij}, it means that the number of times ij occurs as a digraph depends on nothing other than the number of times i occurs in the text and the number of times j occurs in the text, and chance. But the truth is that some digraphs are much more common than the relative frequencies of the separate letters would suggest, and others are much less common. For example, since A is an extremely common letter, $p_{\mathsf{A}\_}$ and $p_{\_\mathsf{A}}$ are usually both relatively large (close to 0.1), but p_{AA} is always much smaller than $p_{\mathsf{A}\_}p_{\_\mathsf{A}}$. Indeed, AA almost never occurs as a digraph in ordinary English text. On the other hand, TH occurs much more often than relative frequencies of T and H would lead you to expect, with p_{TH} typically being about 0.047 and $p_{\mathsf{T}\_}p_{\_\mathsf{H}}$ typically being about 0.005. The effect is that the CD for normal unencrypted text typically takes a much larger value than frequency distribution of the letters would lead one to expect. Application of a transposition cipher can be expected to bring p_{ij} closer to $p_{i\_}p_{\_j}$, and dramatically reduce the CD.

For the story "The Black Cat" the CD was found to be about 0.00343, while for "The Red-Headed League" it was about 0.00346. After applying a transposition cipher with block length 16 and a randomly chosen key, the CD was reduced to 0.000127 for "The Black Cat", and to 0.0000905 for "The Red-Headed League".

We can make use of the CD of a given sequence of pairs of letters in exactly the same way as we used the CI (coincidence index), to help us decide whether the sequence is likely to be a sequence of digraphs from unencrypted text. If some decimation $\mathrm{Dec}(M', [k,l], m)$ of the ciphertext yields a much higher value for the CD than is obtained from other such decimations of M' then it is an indication that m may be the block length of the transposition cipher, and k and l may be adjacent in the inverse key. (However, it is again true that reversing all the pairs in the given sequence does not alter its CD, so that this test, like the CI test, does not help us distinguish the inverse key from its reverse.)

For the purpose of finding the adjacencies in the inverse key, it is unclear whether the CD test is likely to be any better or any worse than the CI test. For either test the time consuming part is counting up the number of times the various digraphs occur in the text, and after doing that one may as well use both tests.

So a strategy for decrypting a block transposition cipher is as follows. Starting at $m = 2$, compute the CD and CI of $\mathrm{Dec}(M', [1, l], m)$ for all values of l from 2 to m. Reject m if we do not find any l such that the CI exceeds 0.007 and the CD exceeds 0.003. The first m that we find for which there is some l that gives suitably large values for the CI and CD should be accepted as the likely block length. For this value of m, compute the CI and CD of $\mathrm{Dec}(M', [k, l], m)$ for all k, l in $\{1, 2, \ldots, m\}$ with $k < l$. You should find that there are $m - 1$ pairs (k, l) that give values for the CI and CD that are noticeably larger than those obtained from the other pairs (k, l), and then be able to arrange the numbers from 1 to m in a sequence in such a way that the pairs of numbers that are adjacent in the sequence are exactly the pairs that gave high values for the CI and CD. There will be two ways of doing this, one being the reverse of the other. One will be the inverse key, and trying them both will tell you which it is.

Another (more obvious) test is simply to choose some sample of ordinary text and compute the relative frequencies of the various digraphs. Let s_{ij} be the relative frequence of ij in the sample. Compare this with p_{ij}, the relative frequency of ij in $\mathrm{Dec}(M', [k, l], m)$. If m is the period and if k immediately precedes l in the inverse key then p_{ij} should be reasonably close to s_{ij} for most pairs ij. Hence $\sum_{ij}(p_{ij} - s_{ij})^2$ should be small. If m is not the period, or if k does not immediately precede l in the inverse key, a much larger value for $\sum_{ij}(p_{ij} - s_{ij})^2$ should be obtained. Compared with the CI and CD tests, this test has the small advantage that when i and j are adjacent it tells you their order.

Week 6

§30 Residue arithmetic

If n is an arbitrary positive integer, we define $\mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}$, the set of residues modulo n. We shall frequently meet situations in which we are interested only in the mod n residue of some integer rather than the integer itself, and in such cases we can frequently simplify arithmetical calculations by replacing the results of any additions or multiplications we need to perform by their residues. The key fact that we make use of here is Theorem (2.4), which tells us that if a and b are any (possibly very large) integers then we can find the mod n residue of the sum or product of a and b by replacing a and b by their mod n residues before performing the addition or multiplication. The saving is that addition or multiplication of large numbers is replaced by addition or multiplication of small numbers. In a long calculation with many additions or multiplications, or if the input numbers a and b are really enormous, the savings can be considerable.

So we define new addition and multiplication operations on the set $\mathbb{Z}_n$ as follows: if $x, y \in \mathbb{Z}_n$ then the "sum mod n" and "product mod n" of x and y are defined to be the mod n residues of their ordinary sum and product. This gives us a new arithmetic, which we call *residue arithmetic mod n*, in which the only numbers are the elements of $\mathbb{Z}_n$: if we are given x and y in Z_n, then by definition $x + y$ and xy are also in $\mathbb{Z}_n$, if residue arithmetic is used. We shall not distinguish notationally between ordinary addition and multiplication and their counterparts in residue arithmetic, but we shall say explicitly when residue arithmetic is being used.

(6.1) ***Example:***

1) If we use residue arithmetic mod 17 then $10 \times 12 = 1$ and $10 + 12 = 5$.
2) If we define matrix multiplication in the usual way, but use residue arithmetic mod 27 in place of ordinary arithmetic, then

$$\begin{pmatrix} 13 & 20 \\ 6 & 16 \end{pmatrix} \begin{pmatrix} 7 & 11 \\ 13 & 21 \end{pmatrix} = \begin{pmatrix} 0 & 23 \\ 7 & 24 \end{pmatrix}.$$

3) If $a \in \mathbb{Z}_n$ and k is any natural number, then $a^k = 1$ in residue arithmetic mod n if and only if k is a multiple of $\operatorname{ord}_n(a)$. (See Definition (2.17). Note, however, that $\operatorname{ord}_n(a)$ exists only if $\gcd(a, n) = 1$.)

The special case of residue arithmetic modulo a prime number p is of particular importance. If residue arithmetic mod p is used then $\mathbb{Z}_p = \{0, 1, \dots, p-1\}$ is an example of an algebraic system known as a *field*. What this means is that the following rules of ordinary arithmetic are still satisfied in residue arithmetic mod p:

F1) $a + b = b + a$ for all a, b;
F2) $(a + b) + c = a + (b + c)$ for all a, b, c;
F3) $a + 0 = a$ for all a;
F4) For every a there is a b such that $a + b = 0$;
F5) $ab = ba$ for all a, b;
F6) $(ab)c = a(bc)$ for all a, b, c;
F7) $a1 = a$ for all a;
F8) For every $a \neq 0$ there is a b such that $ab = 1$;
F9) $(a + b)c = ac + bc$ for all a, b, c.

These nine rules are the *field axioms*. If F is any set and it is possible to define addition and multiplication for elements of F so that $a + b \in F$ and $ab \in F$ whenever $a, b \in F$, and the above nine axioms are satisfied, then we say that F is a field with respect to the addition and multiplication in question.

The element b appearing in Axiom F4 is called the *negative* of the element a, and it is usually written as $-a$. In the case of residue arithmetic modulo p, we see that $-a = p - a$. Similarly, the element b appearing in Axiom F8 is called the *inverse* of a, and it is usually written as a^{-1}, although we shall occasionally use the notation $\frac{1}{a}$ instead. Thus, for example, if we write 5^{-1} or $\frac{1}{5}$ in a context where we are using residue arithmetic mod 17, then we mean 7, since $5 \times 7 \equiv 1 \pmod{17}$.

There are two key theoretical facts that ensure that residue arithmetic mod p satisfies these nine axioms. The first is Theorem (2.4), which ensures, for example, that when calculating the residue of $(a + b) + c$, the same answer is obtained whether reduction mod p is applied after each addition, or both additions are performed first, with reduction mod p deferred until the end. Since the same applies to $a + (b + c)$, it follows that $(a + b) + c$ and $a + (b + c)$ in residue arithmetic both yield the residue of $a + b + c$ as calculated with ordinary arithmetic. Similar remarks apply for Axioms F6 and F9. The other key fact is that if a is a nonzero element of $\mathbb{Z}_p$ then $\gcd(a, p) = 1$, so that Proposition (2.14) ensures the existence of a b such that $ab = 1$ in residue arithmetic mod p. Hence Axiom F8 is satisfied. All the other axioms are now trivial to check.

The set of all real numbers forms a field with respect to ordinary addition and multiplication, and so does the set of all rational numbers. The set of all complex numbers forms a field with respect to addition and multiplication of complex numbers, defined in the usual way. And the residues modulo a prime form a field with respect to residue arithmetic. There are other fields, but it is important to realize that fields are very special things. We should count ourselves lucky that the residues modulo p form a field. We should also note that if n is not prime then the residues modulo n do not form a field with respect to residue arithmetic modulo n, the reason being that $a \not\equiv 0 \pmod{n}$ does not imply that $\gcd(a, n) = 1$ if n is not prime, and consequently Axiom F8 is not satisfied. Nevertheless residue aithmetic modulo n is useful even when n is not prime, and Axiom F8 is the only field axiom that fails.

§31 The Data Encryption Standard

In 1972/73 the US Federal Department of Commerce called for proposals for a cryptographic standard for processing and distributing information on computer systems. They wanted it to be easy to implement, cheap and available to all, with comprehensive and transparent specifications and a high level of security. Security was not to rely on secrecy of the algorithm.

No proposals were received until 1974, when IBM put forward a (variant of a) system called Lucifer that they had been developing. The U.S. National Security Agency took it over and modified it, and the resulting system became the “Data Encryption Standard”.

The N.S.A.’s changes were simplifications: they reduced the key size from 128 bits to 64,* and they simplified the so called S-boxes (to be explained below). This was done so that the whole mechanism could be implemented on a single chip. Suspicious people believed (possibly correctly!) that the N.S.A. had built in some secret mechanism that would enable them to decrypt things without having the key. But despite people searching very hard for this secret, no one has ever found it (or at least not disclosed it). The DES is now obsolete since the key space is not large enough given the power of present day computers. On an appropriate machine, an exhaustive key search will take less than a day of computing time.

Although the DES uses 64 bit keys—that is, sequences of 64 terms that are each either 0 or 1—each 8th bit is a parity check, meaning that it is the sum (in residue arithmetic mod 2) of the previous 7 bits. This is simply to provide a means of detecting errors in recording the key. In effect, the key length is really only 56 bits.

The same key is used for both enciphering and deciphering. Security of the key is crucial to the security of the system.

After choosing a key the user creates 16 “subkeys”, by a formula that we now describe. First, a permutation is applied, reordering the 56 bits that comprise the key. In fact, the permutation used is as follows.

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

* The term “bit” is a contraction of “binary digit”. In our terminology a bit is an element of the set $\mathbb{Z}_2 = \{0, 1\}$; that is, a bit is a residue mod 2.

This table is meant to be represent a string of 56 numbers: concatenate the rows, top row first, then the next row, then the next, and so on through to the bottom row. What is meant is that the 57th bit of the original 64 bit key becomes the 1st bit of the new key, the 49th bit of the original becomes the 2nd bit of the new, and so on. Notice that the parity check bits (corresponding to the multiples of 8) are not used.

Next, the new 56 bit key is split into two halves, which we shall refer to as C_0 and D_0. Here C_0 consists of the first 28 bits of the key and D_0 the last 28 bits of the key. (So the first three bits of C_0 are the 57th, 49th and 41st bits of the original key, and the first three bits of D_0 are the 63rd, 55th and 47th bits of the original key.) Now each of C_0 and D_0 are left-shifted one place, with wrap-around, to produce C_1 and D_1. That is to say, the 1st bit of C_0 is moved to the other end, becoming the 28th bit of C_1, the 2nd bit of C_0 becomes the 1st bit of C_1, the 3rd bit of C_0 becomes 2nd bit of C_1, and so on. And D_1 is created from D_0 in the same way. After this C_1 and D_1 are tacked together, to once more form a 56 bit sequence, and a sequence of length 48 is extracted from it according to the following table.

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

That is, the 14th bit of C_1D_1 is the 1st bit of this new sequence, the 17th bit of C_1D_1 is the 2nd bit of the new sequence, and so on. This new sequence is K_1, the first of the 16 subkeys we are creating.

To get the next subkey we split C_1D_1 back into its two halves C_1 and D_1, left shift them both one place to produce C_2 and D_2, tack these together and extract the 48 bit sequence K_2 using the same table (above) that we used to get K_1.

And this same process is repeated to get the other subkeys, except that you do not always left shift C_i and D_i one place; in fact, you usually left shift them two places. (Left shifting two places means that the 3rd bit becomes the 1st, the 4th becomes the 2nd, and so on, with the 1st and 2nd becoming the 27th and 28th respectively.) The rules are that to get C_{i+1} and D_{i+1} from C_i and D_i you left shift them 1 place if i is 0, 1, 8 or 15, and you left shift them two places otherwise. There are $4 \times 1 + 12 \times 2 = 28$ left shifts performed altogether; so in fact $C_{16}D_{16} = C_0D_0$.

Note that these rules are completely specific: one can say exactly which bits of the original 64 bit key make up the 48 bits of each of the 16 subkeys. To be exact, there are 16 one-to-one functions $\phi_i\colon \{1, 2, \ldots, 48\} \to \{1, 2, \ldots, 64\}$ such that the ith subkey is $\phi_i(1)\phi_i(2)\ldots\phi_i(48)$.

The plaintext to be enciphered should be a 64 bit string. Presumably in practice one will want to encipher longer things than this; so you have to do the enciphering

64 bits at a time. There are several different “modes of operation” possible; you may, for example, choose to somehow use the first 64 bits of ciphertext to modify the second 64 bits of plaintext before enciphering them. We shall not concern ourselves with the modes of operation, and look only at the procedure for enciphering 64 bits of data using the DES.

First, the 64 input bits are permuted, using the following permutation, known as *IP* (for “initial permutation”).

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

As before, this means that the 58th bit becomes the new 1st bit, the 50th the new 2nd, and so on. After this reordering, the 64 bits are divided into two 32 bit blocks; we call the first 32 bits L_0 and the last 32 bits R_0. Then an iterated block transposition cipher, known as a *Feistel cipher*, is applied. In this case there are 16 iterations, or *rounds*, which successively use the 16 subkeys K_i defined above.

The first round takes L_0, R_0 and K_1 as input; the output consists of L_1 and R_1, defined by

$$\begin{aligned} L_1 &= R_0 \\ R_1 &= L_0 + f(K_1, R_0) \pmod 2, \end{aligned}$$

where f is a certain function that takes a 48 bit sequence K and a 32 bit sequence R as input and returns a 32 bit sequence as output. We shall describe this function f below. The $+$ in the second equation is to be interpreted in the sense of residue arithmetic mod 2, defined by $0 + 0 = 1 + 1 = 0$ and $0 + 1 = 1 + 0 = 1$.* Adding two 32 bit sequences means treating them as row vectors, and adding them component by component.

The second round takes L_1, R_1 and K_2 as input, and produces L_2 and R_2 as output. The formulas are exactly the same as those used in the first round:

$$\begin{aligned} L_2 &= R_1 \\ R_2 &= L_1 + f(K_2, R_1) \pmod 2. \end{aligned}$$

Notice that exactly the same function f is used. We continue like this, doing it 16 times altogether, so that each of the 16 subkeys K_i is used once. Thus the 16th and last round is

$$\begin{aligned} L_{16} &= R_{15} \\ R_{16} &= L_{15} + f(K_{16}, R_{15}) \pmod 2. \end{aligned} \tag{11}$$

* Computer scientists often call mod 2 addition “xor”, for “exclusive or”.

After this the 32 bit sequences L_{16} and R_{16} are joined together to make a 64 bit sequence. However, they are joined in the order $R_{16}L_{16}$ rather than $L_{16}R_{16}$; this is done to make the decryption process more similar to the encryption process than it would be otherwise. Finally, the ciphertext is obtained by applying the inverse of *IP* to the sequence $R_{16}L_{16}$. For what it's worth, the inverse of *IP* is as follows.

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

It remains to describe the function f, but before we do so we should consider what the the decryption process will entail. In fact it is just the same as the encryption process, except that the subkeys are used in the reverse order. Thus, taking the ciphertext one first applies the permutation *IP*, obtaining a sequence which in fact is exactly the sequence $R_{16}L_{16}$ above. This is split into two halves L'_0 and R'_0 and the Feistel cipher is applied, just as was done when encrypting. The only difference is that we use K_{16} rather than K_1 in the first round, and K_{15} rather than K_2 in the second round, and so on. Note also that the first half is R_{16} and the last half is L_{16}, so that L's and R's are now round the other way compared with before. So the first round of decryption is

$$\begin{aligned} L'_1 &= R'_0, \\ R'_1 &= L'_0 + f(K_{16}, R'_0) \pmod 2. \end{aligned}$$

But $L'_0 = R_{16}$ and $R'_0 = L_{16}$, and $L_{16} = R_{15}$ in view of the equations (11) above. So $f(K_{16}, R'_0) = f(K_{16}, L_{16}) = f(K_{16}, R_{15})$. Moreover, the equations (11) also give $R_{16} = L_{15} + f(K_{16}, R_{15})$, and so

$$R'_1 = L'_0 + f(K_{16}, R'_0) = R_{16} + f(K_{16}, R_{15}) = (L_{15} + f(K_{16}, R_{15})) + f(K_{16}, R_{15}) = L_{15}$$

since adding the same thing twice gets you back to where you started when using residue arithmetic modulo 2. (Effectively, minus is the same as plus in this situation. This is as it should be, because $-1 \equiv 1 \pmod 2$.) And we also have that $L'_1 = R'_0 = L_{16} = R_{15}$. So just as $L'_0R'_0 = R_{16}L_{16}$, so we also have that $L'_1R'_1 = R_{15}L_{15}$. And, in the same way, applying second round of decryption to $L'_1 = R_{15}$ and $R'_1 = L_{15}$ with the key K_{15} will produce $L'_2 = R_{14}$ and $R'_2 = L_{14}$. Continuing on through all the keys we eventually get $L'_{16} = R_0$ and $R'_{16} = L_0$, and we stick these back together in the order $R'_{16}L'_{16}$ to obtain L_0R_0. Finally, the inverse of *IP* is applied to L_0R_0 and we get back our original plaintext.

Now we describe the function f; this is where the S-boxes we mentioned above come into the story. The function f requires a 32 bit sequence R and a 48 bit sequence K as input, and must produce a 32 bit sequence as output. The first step is to expand R to 48 bits by re-using some of the bits of R. The rule for doing this is as follows.

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Our notation is as before. Notice that bits 4, 5, 8, 9, 12, 13, 16, 17, 20, 21, 24, 25, 28, 29, 32 and 1 get used twice, giving the extra 16 needed to expand from 32 to 48. The resulting expanded sequence is added to K – row vector addition modulo 2, as above – to give another 48 bit sequence. This is split into 8 pieces of 6 bits each. There are eight S-boxes, each of which describes a function from 6 bit sequences to 4 bit sequences; the 1st of these functions is applied to the 1st 6 bit piece, the 2nd to the 2nd, and so on, and the eight outputs are concatenated (in order) to form the 32 bit output of f. The S-boxes are as follows.

S_1	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1110	0100	1101	0001	0010	1111	1011	1000	0011	1010	0110	1100	0101	1001	0000	0111
01	0000	1111	0111	0100	1110	0010	1101	0001	1010	0110	1100	1011	1001	0101	0011	1000
10	0100	0001	1110	1000	1101	0110	0010	1011	1111	1100	1001	0111	0011	1010	0101	0000
11	1111	1100	1000	0010	0100	1001	0001	0111	0101	1011	0011	1110	1010	0000	0110	1101

S_2	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1111	0001	1000	1110	0110	1011	0011	0100	1001	0111	0010	1101	1100	0000	0101	1010
01	0011	1101	0100	0111	1111	0010	1000	1110	1100	0000	0001	1010	0110	1001	1011	0101
10	0000	1110	0111	1011	1010	0100	1101	0001	0101	1000	1100	0110	1001	0011	0010	1111
11	1101	1000	1010	0001	0011	1111	0100	0010	1011	0110	0111	1100	0000	0101	1110	1001

S_3	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1010	0000	0101	1110	0110	0011	1111	0101	0001	1101	1100	0111	1011	0100	0010	1000
01	1101	0111	0000	1001	0011	0100	0110	1010	0010	1000	0101	1110	1100	1011	1111	0001
10	1101	0110	1000	1001	1000	1111	0011	0000	1011	0001	0010	1100	0101	1010	1110	0111
11	0001	1010	1101	0000	0110	1001	1000	0111	0100	1111	1110	0011	1011	0101	0010	1100

S_4	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	0111	1101	1110	0011	0000	0110	1001	1010	0001	0010	1000	0101	1011	1100	0100	1111
01	1101	1000	1011	0101	0110	1111	0000	0011	0100	0111	0010	1100	0001	1010	1110	1001
10	1010	0110	1001	0000	1100	1011	0111	1101	1111	0001	0011	1110	0101	0010	1000	0100
11	0011	1111	0000	0110	1010	0001	1101	1000	1001	0100	0101	1011	1100	0111	0010	1110

S_5	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

S_6	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1100	0001	1010	1111	1001	0010	0110	1000	0000	1101	0011	0100	1110	0111	0101	1011
01	1010	1111	0100	0010	0111	1100	1001	0101	0110	0001	1101	1110	0000	1011	0011	1000
10	1001	1110	1111	0101	0010	1000	1100	0011	0111	0000	0100	1010	0001	1101	1011	0110
11	0100	0011	0010	1100	1001	0101	1111	1010	1011	1110	0001	0111	0110	0000	1000	1101

S_7	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	0100	1011	0010	1110	1111	0000	1000	1101	0011	1100	1001	0111	0101	1010	0110	0001
01	1101	0000	1011	0111	0100	1001	0001	1010	1110	0011	0101	1100	0010	1111	1000	0110
10	0001	0100	1011	1101	1100	0011	0111	1110	1010	1111	0110	1000	0000	0101	1001	0010
11	0110	1011	1101	1000	0001	0100	1010	0111	1001	0101	0000	1111	1110	0010	0011	1100

S_8	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1101	0010	1000	0100	0110	1111	1011	0001	1010	1001	0011	1110	0101	0000	1100	0111
01	0001	1111	1101	1000	1010	0011	0111	0100	1100	0101	0110	1011	0000	1110	1001	0010
10	0111	1011	0100	0001	1001	1100	1110	0010	0000	0110	1010	1101	1111	0011	0101	1000
11	0010	0001	1110	0111	0100	1010	1000	1101	1111	1100	1001	0000	0011	0101	0110	1011

Given a 6 bit sequence as input to the S-box function, the output is obtained as follows. The first and last bit of the input are stuck together to make a 2 bit number, and this specifies a row of the S-box. The middle four bits specify the column. Thus, for example, the input 011001 to S_5 will return the entry in row 01 and column 1100 of the table above, namely 0011.

Observe that each row of each S-box is a permutation of the numbers 0 to 15 expressed in binary notation.* (That is to say, all 16 four-bit sequences appear in each row of each S-box.) So, in effect, the first and last bits of the input specify a permutation which is then applied to the number made from the middle 4 bits.

This completes our description of the DES. In the year 2002 it was replaced as the USA standard by the so-called "advanced encryption standard", or AES. The AES uses components that are somewhat similar to those used by the DES, but is more complicated, and, in particular, uses longer keys. Full information about the AES can be obtained from http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

As mentioned above, the DES is no longer secure against a sufficiently determined attack, because the key space is too small. But your can still use "triple DES": encrypt your message using one key, then encrypt the encrypted message using another key, and finally encrypt the result of this using a third key. Given

* Binary (or base 2) notation will be discussed in §35. In binary notation the integers from 0 to 15 are 0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1100, 1101, 1111.

that the DES succumbs to an exhaustive key search in a day of computing time, triple DES would require $(2^{56})^2$ days of computing time, and even if your enemy's computer would last that long, the solar system certainly won't. Triple DES is one of the three (USA) "Federal Information Processing Standard" approved algorithms, the others being AES and Skipjack. One can find out about these things from http://csrc.nist.gov/CryptoToolkit/tkencryption.html.

The above discussion of the DES is not examinable; it is just included for interest and to give an example of a modern cipher system. Students are welcome to try to find the secrets that the NSA allegedly built into it!

§32 The Rivest–Shamir–Adleman cryptosystem

RSA is a public key cryptosystem. This means that the encryption key is made public, the decryption key being kept secret. Of course it is essential that knowing the encryption key (public key) does not enable one to work out the decryption key (private key)!

Public key cryptography was invented in 1976. Perhaps before then no one had realized that such systems were possible, but surely the most important step was to realize that they would be useful. RSA was invented in 1978.

Suppose that persons A and B wish to communicate securely. Following convention, let us call these two "Alice" and "Bob". In this case, we assume that in fact Bob does not necessarily know that Alice wishes to send him a message, he only knows that someone may wish to. Perhaps Bob runs a large firm with many clients, any of whom may need to send secure messages from time to time. Alice may not even be an existing client, merely a potential client.

Bob chooses two Very Large Primes p and q, and a number e less than $(p-1)(q-1)$ and such that $\gcd(e,(p-1)(q-1)) = 1$. He computes $n = pq$, and makes the pair (n,e) public. (For example, he could post the necessary instructions on his web site. However, there is a danger that evil Mallory may set up a bogus web site and fool people into sending him (Mallory) messages meant for Bob.)

Bob privately computes the inverse of e modulo $(p-1)(q-1)$: this is a number d such that $ed \equiv 1 \pmod{(p-1)(q-1)}$. Recall that such a d exists because e is coprime to $(p-1)(q-1)$, and that d is computed via the extended Euclidean Algorithm (which is extremely fast despite the fact that the numbers are huge).

Bob's RSA public key is (n,e), his private key is (n,d).It is worth noting that there is complete symmetry between d and e; so it would have been just as good to make (n,d) the public key and (n,e) the private key. All that matters is that one of d and e is public, one is secret, and their product is congruent to 1 modulo $(p-1)(q-1)$. Of course, it is also crucial that the value $(p-1)(q-1)$ is kept secret, since if one knows $(p-1)(q-1)$ as well as e then one can easily compute d. Since $n = pq$ is public, the security of the system relies on the fact that knowing pq does not enable one to work out $(p-1)(q-1)$. If one could factorize pq, thereby finding p and q, then of course one could find $(p-1)(q-1)$.

But the number pq has been chosen to be so large that no known factorization algorithm is likely to succeed before the end of the world.

To send Bob a message, Alice first of all transforms the message into a sequence $[m_1, m_2, m_3, \ldots, m_l]$, where each m_i is a number less than n. (That is, m_i is a residue mod n.) There are many conceivable ways to do this, and Bob must tell everyone the system he wants them to use. In practice, Bob just has to say which software package he uses, and then his clients should use the same one. This process of converting the message into a sequence of numbers is called "encoding", and is not to be confused with encrypting. There is nothing secret about the encoding process or the reverse decoding process; it will usually be something completely standard and widely used.

One very naive encoding method that would work (at least for messages in English) is as follows. Each letter, numeral and punctuation mark used in English text has an ASCII code, which is a positive integer less than 128.(Here a space is counted as a punctuation mark, and so it also has an ASCII code.) So that each symbol is associated with a three digit number, let us start by adding 100 to all the ASCII codes. Then simply concatenate all the numbers associated with the symbols appearing in the message to make a string of digits of length three times the number of symbols in the message. Call this string M. Now suppose, for example, that the number n has 1000 digits. Then we break M into pieces that each have 999 digits or fewer. That is, if M has more than 999 digits, we define m_1 be the number formed by the first 999 digits of M, then we delete the first 999 digits of M and repeat the process to get m_2, and so on. Only the final term m_l may have fewer than 999 digits. (Note that since 999 is a multiple of 3, and symbols in the message are all associated with numbers between 100 and 228, no m_i will have 0 as its initial digit: the initial digit will actually always be 1 or 2.)

In practise RSA is only used for relatively short messages. Consequently, since the number n is enormous, one residue mod n will be sufficient to encode the entire message.

Having encoded her message as a sequence $[m_1, m_2, m_3, \ldots, m_l]$ of residues mod n, Alice computes the number n_i such that $0 \le n_i < n$ and $n_i \equiv m_i^e \pmod{n}$. That is, n_i is the mod n residue of m_i^e. Alice sends Bob the sequence $[n_1, n_2, n_3, \ldots, n_l]$.

On receiving this from Alice, Bob computes, for each i, the mod n residue of n_i^d (using his secret d). It turns out that $n_i^d \equiv m_i \pmod{n}$; so in fact m_i is the residue of n_i^d, and Bob has reconstructed Alice's message $[m_1, m_2, m_3, \ldots, m_l]$.

Why does this work? The crucial piece of theory is the Euler–Fermat Theorem, which we proved in Week 4. Since the only prime factors of n are p and q, applying the formula for the values of the Euler Phi function gives

$$\phi(n) = n\frac{(p-1)}{p}\frac{(q-1)}{q} = \frac{pq(p-1)(q-1)}{pq} = (p-1)(q-1).$$

The Euler–Fermat Theorem says that $m^{\phi(n)} \equiv 1 \pmod{n}$ whenever $\gcd(m,n) = 1$.

It follows that for all positive integers k,

$$m^{k\phi(n)+1} = (m^{\phi(n)})^k m \equiv 1^k m \equiv m \pmod{n}.$$

Since $de \equiv 1 \pmod{(p-1)(q-1)}$, and $(p-1)(q-1) = \phi(n)$, it follows that $ed = 1 + k\phi(n)$ for some positive integer k; so that $m^{ed} = m^{1+k\phi(n)} \equiv 1 \pmod{n}$. So, at least if $\gcd(m_i, n) = 1$, it follows that

$$n_i^d \equiv (m_i^e)^d \equiv m_i^{ed} \equiv m_i \pmod{n},$$

as claimed above.

The restriction to integers m satisfying $\gcd(m,n) = 1$ is actually unnecessary, since we showed at the end of §20 that if k is any positive integer then the congruence

$$m^{k\phi(n)+1} \equiv m \pmod{n}$$

is satisfied for all m, not just for those that are coprime to n. In proving this we made use of the fact that n is the product of two distinct primes. (In fact, it works whenever n is square-free.) It is perhaps worth observing that the probability that m_i is not coprime to n is so minuscule that it would scarcely have mattered if the decryption process failed in that case.*

We claimed above that one needs to be able to find the factors p, q of n in order to be able to compute $(p-1)(q-1)$ (after which it would be easy to discover the secret number d). But $(p-1)(q-1) = \phi(n)$ is just the number of numbers less than n and coprime to n. So what is wrong with just counting them up? The answer is that to do this would require going through all the numbers i from 1 to n, and working out $\gcd(i,n)$ each time. It is faster to factorize n by going through all the numbers i from 1 to $\sqrt{n}$ and checking whether i is a divisor of n. Hence, as far as anyone knows, the only way to crack RSA is to find a good factorizing algorithm.

§33 More on multiplicative functions

To illustrate the next result before stating it in full generality, we consider the set of natural numbers k that are less than 24, and split this set into subsets according to the value of $\gcd(k, 24)$. It is easy to construct a table of values.

k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
$\gcd(k,24)$	24	1	2	3	4	1	6	1	8	3	2	1	12	1	2	3	8	1	6	1	4	3	2	1

* The number of numbers less than n that are not coprime to n is $n - \phi(n)$; so the probability that a random number less than n is not coprime to n is $\frac{1}{n}(n - \phi(n)) = 1 - \frac{1}{n}\phi(n)$. If $n = pq$, with p, q prime, then $\frac{1}{n}\phi(n) = \frac{(p-1)}{p}\frac{(q-1)}{q} = 1 - \frac{1}{p} - \frac{1}{q} + \frac{1}{pq}$. So the probability in question is $\frac{1}{p} + \frac{1}{q} - \frac{1}{pq}$, which, given the size of p and q, is so small as to be negligible.

So if we define N_d to be the set of those k such that $\gcd(k, 24) = d$, we find that

$$\begin{aligned}
N_1 &= \{1, 5, 7, 11, 13, 17, 19, 23\},\\
N_2 &= \{2, 10, 14, 22\},\\
N_3 &= \{3, 9, 15, 21\},\\
N_4 &= \{4, 20\},\\
N_6 &= \{6, 18\},\\
N_8 &= \{8, 16\},\\
N_{12} &= \{12\},\\
N_{24} &= \{0\}.
\end{aligned}$$

Since every number from 1 to 24 occurs in exactly one of these sets, we conclude that

$$24 = \#N_1 + \#N_2 + \#N_3 + \#N_4 + \#N_6 + \#N_8 + \#N_{12} + \#N_{24} = \sum_{d|24} \#N_d$$

where we use the notation $\#S$ to denote the number of elements of the set S. What we need next is some formula for the number of elements in the various sets N_d. Observe that by the definition of the Euler phi function, $\#N_1 = \phi(24)$ (the number of natural numbers less than 24 and coprime to 24).

Consider another one of the sets N_d above. The first thing to note is that all of the numbers in the set N_d are multiples of d. For example, the elements of the set N_3 are those k for which $\gcd(k, 24) = 3$; they are bound to be multiples of 3. It is natural to divide them all by 3, and look at the numbers that we get. In fact, $N_3 = \{\, 3m \mid m \in S\,\}$, where

$$S = \{1, 3, 5, 7\}.$$

The numbers in S are obviously less than $8 = 24/3$, since the numbers in N_3 are less than 24. It is intuitively reasonable that if the greatest common divisor of k and 24 is 3, then after we divide both k and 24 by 3 there should be no common divisor left. So the four numbers in the set S above should all be coprime to 8. We see at once that this is indeed the case. Furthermore, S contains all the natural numbers h less than 8 and coprime to 8, since $\gcd(h, 8) = 1$ implies that $\gcd(3h, 24) = 3$, and hence that $3h \in N_3$. What we are using here is the general fact that $\gcd(ka, km) = k \gcd(a, m)$, which we will prove below.

Returning to the discussion of our example, we have shown that the elements of the set N_3 are in one to one correspondence with the elements of the set S, which consists of all numbers less than 8 and coprime to 8, and which therefore has $\phi(8)$ elements. Similarly, dividing all the numbers in the set N_2 by 2 will give all the numbers less than 12 and coprime to 12; so N_2 has $\phi(12)$ elements. The same works for all d: dividing the elements of N_d by d yields all the numbers less than $24/d$ that are coprime to $24/d$. So $\#N_d = \phi(24/d)$, and the equation $\sum_{d|24} \#N_d = 24$ proved above becomes $\sum_{d|24} \phi(24/d) = 24$.

In general, we have the following result.

(6.2) Proposition: *Let n be any positive integer. Then $\sum_{d|n} \phi(n/d) = n$.*

The proof is just like the example above. In particular, it relies on the fact about greatest common divisors we mentioned; so let us prove that first.

(6.3) Lemma: *Let a, $m \in \mathbb{Z}$ and let $k \in \mathbb{N}$. Then* $\gcd(ka, km) = k \gcd(a, m)$.

Proof. We remark first that when $k = 0$ the result holds, since both sides of the equation are zero. (Some authors do not define $\gcd(0, 0)$, but in fact the definition $\gcd(0, 0) = 0$ is consistent with all the properties required of gcd's.) So we now assume that $k \neq 0$.

Let $D = \gcd(ka, km)$. By the definition of gcd, this means that

(i) $D|ka$ and $D|km$,

(ii) for all c, if $c|ka$ and $c|km$ then $c|D$.

In particular, since $k|ka$ and $k|km$, it follows from (ii) that $k|D$. Hence we may write $D = kd$ for some $d \in N$, and our task is to prove that $d = \gcd(a, m)$. By the definition, this means we must prove that

(i') $d|a$ and $d|m$,

(ii') for all c, if $c|a$ and $c|m$ then $c|d$.

First of all, condition (i) tells us that $kd|ka$ and $kd|km$; that is, $ka = (kd)r$ and $km = (kd)s$ for some integers r and s. Cancelling k gives $a = dr$ and $m = ds$, so that $d|a$ and $d|m$. Thus we have shown that (i') is satisfied, and it remains to prove (ii').

Suppose then that c is an integer satisfying $c|a$ and $c|m$. Multiplying through by k, we see that $kc|ka$ and $kc|km$. By condition (ii) above, it follows that $kc|D$; that is, $kc|kd$. As before, we can cancel k and conclude that $c|d$. So we have shown that any c that satisfies $c|a$ and $c|m$ must also satisfy $c|d$; that is, condition (ii') is satisfied, as required. □

Proof of Proposition (6.2). For each divisor d of n let

$$N_d = \{k \in N \mid k < n \text{ and } \gcd(k, n) = d\}.$$

Every natural number less than n lies in exactly one of the sets N_d; thus

$$\{0, 1, 2, \ldots, n-1\} = \bigcup_{d|n} N_d,$$

and this is a *disjoint union*, meaning that no two of the sets N_d have any elements in common. So the number of elements in $\bigcup_{d|n} N_d$ is the sum $\sum_{d|n} \#N_d$, whence we deduce that

$$\sum_{d|n} \#N_d = n. \tag{12}$$

Now fix a divisor d of n, and let $m = n/d$. If $k \in N_d$ then $d|k$ (since $d = \gcd(k, n)$), and we may write $k = dh$, where $h \in \mathbb{N}$ and $h < n/d$. Since $\gcd(dh, dm) = \gcd(k, n) = d$, it follows from the lemma that $\gcd(h, m) = 1$. Conversely, if $h < m$ and $\gcd(h, m) = 1$ then $\gcd(dh, dm) = d$, so that $dh \in N_d$. So there is a one to one correspondence between the elements of N_d and the elements of the set $S = \{h \in \mathbb{N} \mid h < m \text{ and } \gcd(h, m) = 1\}$, where $k \in N_d$ corresponds to $h = k/d \in S$. Thus $\#N_d = \#S$. However, by the definition of the Euler phi function, $\#S = \phi(m) = \phi(n/d)$. And since our choice of d was arbitrary, this holds for all divisors d of n, so that the equation (2) above says

$$\sum_{d|n} \phi(n/d) = n,$$

as required. □

Let n be a positive integer. Observe that if d is a divisor of n then $n = de$ for some e, and e is uniquely determined by d. Conversely, e also determines d uniquely. We can call e the divisor of n that is complementary to the divisor d, and d the divisor that is complementary to e. Explicitly, if d is a divisor of n then the complementary divisor is n/d. Now as d runs through all the divisors of n, the divisor complementary to d will also run through all the divisors of n. So the quantity $\sum_{d|n} \phi(n/d)$ in the proposition above can equally well be written as $\sum_{e|n} \phi(e)$, by putting $e = n/d$. So we can restate the proposition in the following simpler form, which now deserves the more respectful title of "Theorem".

(6.4) Theorem: *Let n be any positive integer. Then $\sum_{d|n} \phi(d) = n$.*

We now turn to a discussion of another famous multiplicative function of number theory. It is known as the Möbius function, and its importance stems from the remarkable Möbius Inversion Formula, which basically describes how to find the inverse of a certain matrix. We shall describe Möbius inversion after we have defined the Möbius function and proved its other basic properties.

(6.5) Definition: Let n be a positive integer, and define $\mu(n)$ as follows:

$$\mu(n) = \begin{cases} 0 & \text{if } n \text{ is divisible by the square of any prime,} \\ (-1)^r & \text{if } n = p_1p_2\cdots p_r \text{ for distinct primes } p_1, p_2, \ldots, p_r. \end{cases}$$

The function μ is called the *Möbius function*.

A positive integer n is said to be *square-free* if it is not divisible by the square of any prime. It is clear that in the prime factorization of a square-free integer, each prime factor appears to the power one only. In other words, a positive integer is square-free if and only if it is a product of distinct primes. Thus the two alternatives in the definition of μ above cover all possibilities for the positive integer n. You could reasonably complain that we have not defined $\mu(1)$, on the grounds that 1 is certainly not divisible by the square of any prime, and also cannot be expressed as a product $p_1p_2\cdots p_r$ of distinct primes. But this is really a matter of convention, and in my view the best convention to adopt is that 1 is the product of an empty set of primes. In other words, the equation $1 = p_1 p_2 \cdots p_r$ holds with $r = 0$ (so that $\{p_1, p_2, \ldots, p_r\} = \emptyset$). And therefore $\mu(1) = (-1)^0 = 1$.

If you do not like this, you can just regard $\mu(1) = 1$ as a separate definition. In any case, it is true that $\mu(1) = 1$ by definition.

Recall that a function defined on the positive integers is said to be multiplicative if $f(mn) = f(m)f(n)$ whenever $\gcd(m,n) = 1$. It is not hard to see that the Möbius function has this property.

(6.6) Theorem: *If m and n are coprime positive integers then $\mu(mn) = \mu(m)\mu(n)$.*

Proof. Suppose that $m, n \in \mathbb{Z}^+$ and $\gcd(m,n) = 1$. If either m or n is not square-free then either $\mu(m) = 0$ or $\mu(n) = 0$, and in either case it follows that $\mu(m)\mu(n) = 0$. Moreover, since the divisors of mn include all the divisors of m and all the divisors of n, if either m or n is divisible by the square of a prime then mn is also divisible by the square of a prime. In other words, if either m or n is not square-free then mn is not square-free, and consequently $\mu(mn) = 0$. Hence the equation $\mu(mn) = \mu(m)\mu(n)$ holds if either m or n is not square-free, since both sides of the equation are 0.

It remains to consider the case that m and n are both square-free. In this case we have

$$m = p_1 p_2 \cdots p_r$$

and

$$n = q_1 q_2 \cdots q_s$$

where $p_1, p_2, \ldots, p_r$ are distinct primes and $q_1, q_2, \ldots, q_s$ are distinct primes. This gives $\mu(m) = (-1)^r$ and $\mu(n) = (-1)^s$. Furthermore, since $\gcd(m,n) = 1$, no p_i is equal to any q_j, and so

$$mn = (p_1 p_2 \cdots p_r)(q_1 q_2 \cdots q_s)$$

is a product of $r + s$ distinct primes. Hence

$$\mu(mn) = (-1)^{r+s} = (-1)^r(-1)^s = \mu(m)\mu(n),$$

as required. □

We now prove a general result which we could have—and perhaps should have—proved earlier, when we were discussing the functions σ and τ, since the proof is virtually identical to the proof that σ is multiplicative (see §24).

(6.7) Theorem: *Suppose that f is a multiplicative function defined on $\mathbb{Z}^+$, and define another function F by the rule that $F(n) = \sum_{d|n} f(d)$ for all $n \in \mathbb{Z}^+$. Then F is multiplicative.*

Proof. Suppose that $\gcd(m,n) = 1$. As we saw in the proof that σ is multiplicative, there is a one to one correspondence between the sets $\{\, k \in \mathbb{Z}^+ \bigm| k|mn \,\}$ and $\{\, (a,b) \bigm| a|m \text{ and } b|n \,\}$, given by the following rule:

$$k \leftrightarrow (a,b) \text{ if and only if } k = ab.$$

Thus it follows that

$$F(mn) = \sum_{k|mn} f(k) = \sum_{a|m}\sum_{b|n} f(ab) = \sum_{a|m}\sum_{b|n} f(a)f(b)$$

since f is multiplicative, and the fact the $\gcd(m,n) = 1$ implies that $\gcd(a,b) = 1$ whenever $a|m$ and $b|n$. (The results we have just used were proved in Week 5: see Proposition (5.3) and Proposition (5.6).) Now

$$\begin{aligned}\sum_{a|m}\Bigl(\sum_{b|n} f(a)f(b)\Bigr) = \sum_{a|m} f(a)\Bigl(\sum_{b|n} f(b)\Bigr) &= \sum_{a|m} f(a)F(n)\\ &= \Bigl(\sum_{a|m} f(a)\Bigr)F(n) = F(m)F(n),\end{aligned}$$

whence $F(mn) = F(m)F(n)$, as required. □

(6.8) Corollary: *For each positive integer n define $F(n) = \sum_{d|n} \mu(d)$. Then*

$$F(n) = \begin{cases} 1 & \text{if } n = 1,\\ 0 & \text{if } n > 1.\end{cases}$$

Proof. In the case $n = 1$ we have

$$F(n) = F(1) = \sum_{d|1} \mu(d) = \mu(1) = 1,$$

as claimed. If $n > 1$ then we can write n as a product of powers of primes,

$$n = p_1^{k_1}p_2^{k_2}\cdots p_r^{k_r}$$

where $r \geq 1$ and $k_i \geq 1$ for all $i \in \{1, 2, \ldots, r\}$. Now F is multiplicative, since μ is, and so

$$F(n) = F(p_1^{k_1})F(p_2^{k_2})\cdots F(p_r^{k_r}).$$

We now show that all the factors $F(p_i^{k_i})$ on the right hand side of this equation are zero. Since $r \geq 1$ there is at least one such factor, and so it will follow that $F(n) = 0$, which is what we have to prove.

So our task is reduced to proving that if p is prime and $k \geq 1$ then $F(p^k) = 0$. Now the divisors of p^k are precisely the numbers p^i such that $0 \leq i \leq k$; so

$$F(p^k) = \sum_{d|p^k} \mu(d) = \sum_{i=0}^{k} \mu(p^i) = \mu(1) + \mu(p) + \sum_{i=2}^{k} \mu(p^i).$$

The terms $\mu(1)$ and $\mu(p)$ on the right hand side certainly exist since $k \geq 1$. The sum $\sum_{i=2}^{k} \mu(p^i)$ may or may not be empty, since k may or may not be strictly greater than 1. However, since $\mu(p^i) = 0$ whenever $i > 2$ (since p^i is divisible by p^2 when $i > 2$), it follows that $\sum_{i=2}^{k} \mu(p^i) = 0$. Hence

$$F(p^k) = \mu(1) + \mu(p) = 1 + (-1) = 0,$$

as required. □

Week 7

§34 The Möbius inversion formula

The Möbius Inversion Formula can be regarded as a formula for the inverses of certain matrices. We illustrate it first in a particular case.

Suppose that we are given numbers a_d, one for each divisor d of 12, and want to find numbers x_e satisfying the simultaneous equations

$$\sum_{e|d} x_e = a_d$$

where d runs through all the divisors of 12. Writing this out more explicitly, the aim is to solve the linear system

$$\begin{aligned}
x_{12} + x_6 + x_4 + x_3 + x_2 + x_1 &= a_{12}\\
x_6 \phantom{{}+x_4} + x_3 + x_2 + x_1 &= a_6\\
x_4 \phantom{{}+x_3} + x_2 + x_1 &= a_4\\
x_3 \phantom{{}+x_2} + x_1 &= a_3\\
x_2 + x_1 &= a_2\\
x_1 &= a_1.
\end{aligned}$$

Written like this the system is in echelon form, and hence trivial to solve by back-substitution: the last equation tells us x_1, we can put this value into the previous two equations to find x_2 and x_3, then put the values of x_1 and x_2 that we have found into the equation before that to find x_4, and so on. But the Möbius Inversion Formula tells us the answer without having to do this work. It tells us that the solution is

$$x_e = \sum_{d|e} \mu(\tfrac{e}{d})a_d.$$

Writing this out explicitly,

$$\begin{aligned}
x_{12} &= \mu(\tfrac{12}{12})a_{12} + \mu(\tfrac{12}{6})a_6 + \mu(\tfrac{12}{4})a_4 + \mu(\tfrac{12}{3})a_3 + \mu(\tfrac{12}{2})a_2 + \mu(\tfrac{12}{1})a_1\\
x_6 &= \mu(\tfrac{6}{6})a_6 + \mu(\tfrac{6}{3})a_3 + \mu(\tfrac{6}{2})a_2 + \mu(\tfrac{6}{1})a_1\\
x_4 &= \mu(\tfrac{4}{4})a_4 + \mu(\tfrac{4}{2})a_2 + \mu(\tfrac{4}{1})a_1\\
x_3 &= \mu(\tfrac{3}{3})a_3 + \mu(\tfrac{3}{1})a_1\\
x_2 &= \mu(\tfrac{2}{2})a_2 + \mu(\tfrac{2}{1})a_1\\
x_1 &= \mu(\tfrac{1}{1})a_1.
\end{aligned}$$

Now $\mu(12) = \mu(4) = 0$ since 12 and 4 are not square-free, while $\mu(1) = \mu(6) = 1$ and $\mu(2) = \mu(3) = -1$. So the solution (which the reader should check) is

$$\begin{aligned}
x_{12} &= a_{12} - a_6 - a_4 + a_2 \\
x_6 &= a_6 - a_3 - a_2 + a_1 \\
x_4 &= a_4 - a_2 \\
x_3 &= a_3 - a_1 \\
x_2 &= a_2 - a_1 \\
x_1 &= a_1.
\end{aligned}$$

Expressed in terms of matrices, what this says is that if

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{12} \\ x_6 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \end{bmatrix} = \begin{bmatrix} a_{12} \\ a_6 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \end{bmatrix}$$

then

$$\begin{bmatrix} x_{12} \\ x_6 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{12} \\ a_6 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \end{bmatrix},$$

or, equivalently, that

$$\begin{bmatrix} 1 & -1 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^{-1}.$$

The general statement of the Möbius Inversion Formula is as follows.

(7.1) Theorem: *Let n be a positive integer, and suppose that for each divisor d of n we are given a number a_d. Then the system of equations*

$$\sum_{e|d} x_e = a_d$$

(where there is one equation for each divisor d of n) has the unique solution

$$x_e = \sum_{h|e} \mu(\tfrac{e}{h}) a_h$$

for all divisors e of n.

The proof of the theorem relies on a property of the Möbius function that we have already proved, namely that for each positive integer n,

$$\sum_{d|n} \mu(d) = \begin{cases} 1 & \text{if } n = 1, \\ 0 & \text{if } n > 1. \end{cases}$$

We proved this in Week 6, Corollary (6.8).

Proof of the Möbius Inversion Formula. Our first task is to check that the stated formula for the x_e does indeed give a solution to the equations, by substituting it in and checking that the equations are satisfied. In other words we must show that

$$\sum_{e|d}\Bigl(\sum_{h|e} \mu(\tfrac{e}{h})a_h\Bigr) = a_d.$$

This is an exercise in interchanging the order of summation. Each a_h potentially occurs several times on the left hand side; our task is to collect like terms—find the total coefficient of each a_h—and check that the coefficient of a_h is 1 when $h = d$ and zero otherwise.

On the left hand side, e runs through all the divisors of d, and then h runs through all the divisors of e. The resulting numbers h are certainly divisors of d, since $h|e$ and $e|d$. Moreover, each h will occur once for each e that it divides. In other words, the rule for interchanging the order of summation here is

$$\sum_{e|d}\sum_{h|e}(\ldots\ldots) = \sum_{h|d}\ \sum_{\{e \text{ such that } h|e \text{ and } e|d\}} (\ldots\ldots).$$

Viewed in either way, we are summing over all pairs (e, h) such that e and h are divisors of d, and h is a divisor of e. Thus

$$\begin{aligned}\sum_{e|d}\Bigl(\sum_{h|e} \mu(\tfrac{e}{h})a_h\Bigr) &= \sum_{h|d}\ \sum_{\{e \mid h|e \text{ and } e|d\}} \mu(\tfrac{e}{h})a_h \\ &= \sum_{h|d} a_h\Bigl(\sum_{e} \mu(\tfrac{e}{h})\Bigr)\end{aligned}$$

since a_h is a common factor in the inner sum. Now since each e in the inner sum has to be divisible by h, we can write $e = kh$ for some integer k, and the condition that $e|d$ becomes $kh|d$, which is in turn equivalent to $k|(d/h)$. So

$$\sum_{e|d}\Bigl(\sum_{h|e} \mu(\tfrac{e}{h})a_h\Bigr) = \sum_{h|d} a_h \sum_{k|(d/h)} \mu(k).$$

But Corollary (6.8) said that summing $\mu(k)$ over those k that are divisors of some number m will give 1 if $m = 1$ and 0 otherwise. So the inner sum above is 0 if $d/h \neq 1$, and 1 if $d/h = 1$. That is,

$$\sum_{e|d}\Bigl(\sum_{h|e} \mu(\tfrac{e}{h})a_h\Bigr) = \sum_{h|d} a_h\left(\begin{cases} 1 & \text{if } h = d \\ 0 & \text{if } h \neq d \end{cases}\right) = a_d,$$

as required.

This completes the proof of half of Theorem (7.1): we have shown that the formula given in the statement of the theorem does indeed produce a solution to the given system of equations. The remaining half is the assertion that the solution is unique. We need to show that there is no other solution to the given system of equations.

As our example—the case $n = 12$—illustrated, what we are basically doing is solving a system of equations for which the coefficient matrix is square and upper triangular,* with 1's on the leading diagonal. Thus it is an echelon system with no free variables, and the uniqueness of the solution is a standard fact from linear algebra. But we can also prove uniqueness directly by an argument that is much the same as the one given above; so let us do so (a little less verbosely this time).

Our task is to show that if $\sum_{k|d} x_k = a_d$ for all divisors d of n then necessarily $x_e = \sum_{h|e} \mu(\frac{e}{h})a_h$. That is, we must show that

$$\sum_{h|e} \mu(\tfrac{e}{h}) \sum_{k|h} x_k = x_e.$$

for all divisors e of n. Interchanging the order of summation gives

$$\begin{aligned}\sum_{h|e}\sum_{k|h} \mu(\tfrac{e}{h})x_k &= \sum_{k|e} \sum_{\{h \mid k|h \text{ and } h|e\}} \mu(\tfrac{e}{h})x_k \\ &= \sum_{k|e} x_k \sum_{\{m \mid mk|e\}} \mu(\tfrac{e}{mk}) \\ &= \sum_{k|e} x_k \sum_{m|(e/k)} \mu(\tfrac{e/k}{m}).\end{aligned}$$

But if m is a divisor of e/k then $\frac{(e/k)}{m}$ is simply the complementary divisor, and as m runs through all the divisors of e/k the complementary divisor does too. So by Corollary (6.8)

$$\sum_{m|(e/k)} \mu\Big(\frac{e/k}{m}\Big) = \sum_{m'|(e/k)} \mu(m') = \begin{cases} 1 & \text{if } e/k = 1 \\ 0 & \text{if } e/k \neq 1, \end{cases}$$

and therefore

$$\sum_{h|e}\sum_{k|h} \mu(\tfrac{e}{h})x_k = \sum_{k|e} x_k \left(\begin{cases} 1 & \text{if } k = e \\ 0 & \text{if } k \neq e \end{cases}\right) = x_e,$$

as required. □

* This formulation assumes that the divisors of n are taken in order from largest to smallest. (We did in the $n = 12$ case, choosing the order 12, 6, 4, 3, 2, 1.)

§35 Arithmetic and computation

The RSA and other public key cryptosystems rely on the fact that some calculations with large numbers can be performed much faster than others. For example, if a, b and m are large positive integers then, with appropriate programming, a machine may compute the residue of a^b modulo m extremely quickly. But there may be no computer in the world capable of finding the prime divisors of m, if it does not have many divisors and they are large.

We aim to get some understanding of the reasons for such differences. To do so we must first examine the algorithms that people and machines use for performing the basic operations of arithmetic: addition and multiplication.

The standard everyday notation used for natural numbers is a positional base 10 (or decimal) system, the foundation of which is the fact that every natural number can be uniquely expressed in the form $\sum_{i=0}^{k} 10^i d_i$, where k is a natural number and each d_i is a natural number less than 10. So, with 10 separate symbols—namely, the *digits* 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9—to represent the natural numbers less than 10, an arbitrary natural number can be conveniently represented by a sequence of these symbols: $d_k d_{k-1} \ldots d_1 d_0$ represents $\sum_{i=0}^{k} 10^i d_i$. (Thus, for example, 7461 means $7 \times 10^3 + 4 \times 10^2 + 6 \times 10^1 + 1 \times 10^0$.) The notation is positional in the sense that the position of each digit in the sequence determines the power of 10 that it is associated with. The positions are indexed by the natural numbers, starting with 0 as the index of the rightmost position, and then the coefficient of 10^i goes in position i.

The introduction of a decimal point makes it possible to also have positions indexed by negative integers, and use $d_k d_{k-1} \ldots d_1 d_0 . d_{-1} d_{-2} \ldots d_{-h}$ to represent the real number $\sum_{i=-h}^{k-1} d_i 10^i$ (the d_i being digits).

Without doubt the widespread acceptance of this notation is largely due to the fact that it facilitates straightforward procedures for addition and multiplication. Once the sums and products of all single digit numbers have been found and tabulated, arbitrary sums and products can be computed efficiently. The algorithms involved are taught at primary school.

For example, to compute $m + n$ you work from right to left, adding the digit of n in position i to the digit of m in position i, incrementing the answer by 1 if there was a carrying figure from the preceding position. If this gives a single digit number, that number becomes the position i digit of $m + n$, and there is no carrying figure; otherwise it gives a number in the range 10 to 19 (inclusive), the second digit of which becomes the position i digit of $m + n$, and there is a carrying figure for the next position. For example:

$$\begin{array}{r} 57.632 \\ +\ 54_16_13_1.7 \\ \hline 5521.332 \end{array}$$

Note that the addition tables are really only consulted when m and n both have digits in position i; if only one of them has a digit in position i then that digit is just copied down into $m + n$ (although it may need to be incremented if there was a carrying figure). In any event, if the numbers each have at most k digits then the addition tables are used at most k times.

A similar process applies for multiplying a k digit number m by a single digit number n (excluding the trivial cases $n = 0$ and $n = 1$). The multiplication tables must be consulted k times, and there will usually be a carrying figure—which now can be anything from 1 to 8—to add to the answer. For example:

$$\begin{array}{r} 7_{8}9_{5}6_{1}.2 \\ \times\, 9 \\ \hline 7165.8 \end{array}$$

The process of adding or multiplying two single digit numbers and adding a carrying figure to the answer can be called a *digit operation*. The amount of work involved in a digit operation is exceedingly slight, since it just involves consulting a table. Someone who knows the tables can perform each digit operation extremely quickly. For an ideal person, who never gets tired or makes a mistake, the time it would take to perform a complicated arithmetical calculation would be directly proportional to the number of digit operations involved. It would take ten times as long to add two hundred digit numbers as to add two ten digit numbers, and multiplying a hundred digit number by 7 would take ten times as long as multiplying a ten digit number by 7 (and about the same time as adding two hundred digit numbers).

To multiply a k digit number m by an l digit number n one multiplies m by each digit of n (excluding any 0's), left shifting the result i places when using the position i digit of n. The answers are then added. This involves at most $2kl - k$ digit operations: kl multiplications and $k(l - 1)$ additions. (Note that each of the l intermediate answers has at most $k + 1$ digits, and the shifting means that when these are successively added to a running total at most k digit additions are used.)

The most important thing to realize here is that the amount of work involved in adding or multiplying two enormous integers is not so enormous. For example, if they are less than 10^{100} then addition requires no more than 100 digit operations, multiplication fewer than twenty thousand. Doing it by hand would be quite feasible. By contrast, an algorithm that required a person to perform 10^{100} operations, no matter how trivial, would be totally infeasible.

A viable notation for numbers can be obtained using any integer $b > 1$ as base (instead of 10). A positional base b notation uses b symbols—the *base b digits*—for the natural numbers less than b.* If $d_{-h}, d_{-(h-1)}, \dots, d_{-1}, d_0, d_1, \dots, d_{k-1}, d_k$ are base b digits then

$$(d_k d_{k-1} \dots d_1 d_0 . d_{-1} d_{-2} \dots d_{-h})_b$$

is base b notation for the number $\sum_{i=-h}^{k} d_i b^i$. Thus, for example, $(235.4)_7$ denotes the number $2 \times 7^2 + 3 \times 7 + 5 + 4 \times 7^{-1}$ (which is $124.\overline{571428}$ in base 10).

* For those natural numbers that are less than 10 as well as less than b, the usual digits will serve also as base b digits.

If it is clear from the context which base is being used, we may omit the parentheses and subscript, writing (for example) 235.4 rather than $(235.4)_7$.

After the case $b = 10$ (decimal notation) the most important case is $b = 2$ (binary notation). This is because computers, in their internal workings, essentially use binary notation to store numbers. The next most important cases are *octal* (base 8) and *hexadecimal* (base 16); their importance comes from the fact that conversion between binary and octal or hexadecimal is trivial, but a k digit octal number is a $3k$ digit binary number, and a k digit hexadecimal number is a $4k$ digit binary number. People prefer to use extra symbols (up to a point!) for the sake of shorter expressions.

It is customary to use A, B, C, D, E and F as hexadecimal digits representing the integers that in base 10 are represented as 10, 11, 12, 13, 14 and 15. Thus, for example, FF is hexadecimal notation for $15 \times 16 + 15 = 255$.

The standard algorithms for addition and multiplication, described above, work just as well with any base. The only difference is that the digit operations require different tables.

Here are the digit addition and multiplication tables that are needed for arithmetic in base 8:

+	1	2	3	4	5	6	7
1	2	3	4	5	6	7	10
2	3	4	5	6	7	10	11
3	4	5	6	7	10	11	12
4	5	6	7	10	11	12	13
5	6	7	10	11	12	13	14
6	7	10	11	12	13	14	15
7	10	11	12	13	14	15	16

×	2	3	4	5	6	7
2	4	6	10	12	14	16
3	6	11	14	17	22	25
4	10	14	20	24	30	34
5	12	17	24	31	36	43
6	14	22	30	36	44	52
7	16	25	34	43	52	61

A table is not needed for multiplication by 0, since the answer is always 0. Since adding 0 and multiplying by 1 are "do nothing" operations, they also do not need to be tabulated.

Using the above tables and the standard "long multiplication" algorithm, the calculation of $(23.1)_8 \times (73)_8$ goes as follows.

$$\begin{array}{r} 23.1 \\ 73 \\ \hline 71.3 \\ 2057. \\ \hline 2150.3 \end{array}$$

The carrying figures have been omitted, to avoid clutter. The reader is invited to check the calculation.

The above analysis of addition and multiplication using digit operations could be extended to include subtraction (which has complexity similar to that of addition) and division (which has complexity similar to that of multiplication).

Here is an example of long division in octal. The aim is to find the octal representation of the reciprocal of $(12)_8$ (which is 10 in decimal notation). In advance, we can expect that it will turn out to be a “repeating octal”, since in base 10 the reciprocals of integers are always repeating decimals.

$$
\begin{array}{r|l}
\multicolumn{1}{r}{} & 0.06314\ldots \\
\cline{2-2}
12 & 1.000000\ldots \\
\multicolumn{1}{r}{} & \underline{74} \\
\multicolumn{1}{r}{} & 40 \\
\multicolumn{1}{r}{} & \underline{36} \\
\multicolumn{1}{r}{} & 20 \\
\multicolumn{1}{r}{} & \underline{12} \\
\multicolumn{1}{r}{} & 60 \\
\multicolumn{1}{r}{} & \underline{50} \\
\multicolumn{1}{r}{} & 10
\end{array}
$$

We have stopped at the place where repetition starts. The answer is $(0.\overline{06314})_8$.

It is necessary at this point to temporarily change the topic, and remind the reader of some basic facts concerning logarithms and exponentiation. Firstly, the exponential function $x \mapsto e^x$ and the natural logarithm function $x \mapsto \ln x$ are inverses of each other. That is, $t = \ln s$ if and only if $s = e^t$. Note that the domain of the exponential function is the set of all real numbers, but the domain of the natural logarithm function is the set of positive real numbers: if $s < 0$ then $\ln s$ is not defined.

If a is a real number and b a positive real number, then $b^a = e^{a \ln b}$. Thus $\ln(b^a) = a \ln b$. The $\log_b$ function is defined by the requirement that $c = b^a$ if and only if $a = \log_b c$. Consequently, $\log_b c = a$ if and only $\ln c = a \ln b$. In other words, we have the formula

$$\log_b c = \frac{\ln c}{\ln b}. \tag{13}$$

It is an easy consequence of this that

$$\log_b c = \frac{\log_d c}{\log_d b} \tag{14}$$

holds for all positive numbers b, c and d.

Regarding b and d as fixed and c as variable, Eq. (14) tells us that the $\log_b$ function and the $\log_d$ function are proportional to one another. To convert any base d logarithm to a base b logarithm, divide by the constant $\log_d b$. Equally, to convert a base b logarithm to a base d logarithm, divide by $\log_b d$. Of course this means that $log_b d = 1/\log_d b$ (which also follows from Eq. (14) by putting $c = d$).

It is well known that ln is an increasing function, and since $\ln 1 = 0$ it follows that $\ln b > 0$ whenever $b > 1$. Using Eq. (13), we conclude that $\log_b$ is an increasing function whenever $b > 1$.

Observe that if an integer n has k digits in base b notation then $b^{k-1} \leq n < b^k$, and so $k - 1 \leq \log_b n < k$. (For example, $\log_{10} 100 = 2$, and $\log_{10} 999 < \log_{10} 1000 = 3$.) So the number of digits of n in base b is $[\log_b n] + 1$, where $[\ldots]$ is the *integer part* function: $[x]$ is the largest integer less than or equal to x.* In view of our digit operation comments above, we can say (for example) that if m and n are positive integers, then the number of base b digit operations needed to compute the product mn is less than $2(\log_b m)(\log_b n)$.

We have wandered a little away from the shortest path to our goal. Since we live in the age of the electronic computer, we should focus our attention on computer calculations rather than hand calculations. We have already noted that (in essence) computers use binary notation; we concentrate on this case henceforth.

Recall that in binary notation the only digits are 0 and 1, and recall also that binary digits are called “bits”. So we use the term *bit operation*, instead of digit operation, in the binary case.

Although we have already described the standard algorithms for addition and multiplication in base b notation, it will do no harm to describe them again in the case $b = 2$, and perhaps clarify the bit operation concept at the same time. We start by closely examining the procedure for adding two integers expressed in binary notation. We assume that they both have k bits (using leading zeros if necessary).

$$\begin{array}{r} 101100110 \\ {}_1 0 {}_1 1 {}_1 1 {}_1 1 1 0 {}_1 1 0 0 \\ \hline 1001011010 \end{array}$$

Starting with the rightmost column, we apply the following algorithm k times (once for each column) working from right to left. In the first step (the rightmost column) there is no carrying figure.

- If the bit in the first row and the bit in the second row are both 0 and there is no carrying figure, write 0 in the answer row and move on to the next column.
- If the bit in the first row and the bit in the second row are both 0 and there is a carrying figure, write 1 in the answer row and move on to the next column.
- If the bit in the first row is 1 and the bit in the second row is 0, or vice versa, and there is no carrying figure, write 1 in the answer row and move on.
- If the bit in the first row is 1 and the bit in the second row is 0, or vice versa, and there is a carrying figure, write 0 in the answer row, put a carrying figure into the next column, and move on.
- If the bit in the first row and the bit in the second row are both 1, and there is no carrying figure, write 0 in the answer row, put a carrying figure into the next column, and move on.
- If the bit in the first row and the bit in the second row are both 1, and there is a carrying figure, write 1 in the answer row, put a carrying figure into the next column, and move on.

We call each performance of the above algorithm one bit operation.

* In MAGMA, `Floor(x)` returns the integer part of `x`.

The electronics that enables computers to perform bit operations need not concern us; all that matters is that adding two k bit integers involves k bit operations.* More complicated calculations that a computer performs are basically made up of bit operations and things that we shall loosely refer to as “book-keeping” operations, including accessing memory and copying bits from one place to another. Of course, the time it takes a computer to perform one bit operation is unbelievably short, but the number of bit operations involved in complex tasks is unbelievably enormous. In practice, estimating the number of bit operations required for a complicated calculation provides a reasonable way of estimating the time it will take a computer to perform the calculation in question. Presumably this is because the bit operations take most of the time. But the amount of book-keeping is probably also proportional to the number of bit operations.

Since the only binary digits are 0 and 1, the standard algorithm for multiplication in base b is simpler when $b = 2$ than it is when $b > 2$. Multiplication by 0 always gives 0, and multiplication by 1 does nothing. Now consider what is involved in multiplying a k bit integer m by an l bit integer n, using long multiplication.

$$\begin{array}{rr}
m: & 11101 \\
n: & 1101 \\ \hline
 & 11101 \\
 & 11101 \\
 & 11101 \\ \hline
 & 101111001
\end{array}
\qquad\qquad
\begin{array}{rr}
 & 11101 \\
+ & {}_1 1_1 1_1 1 0_1 1 \\ \hline
 & 10010001 \\
+ & {}_1 11101 \\ \hline
 & 101111001
\end{array}$$

The final answer is the sum of several terms, each of which is a left-shifted copy of m. There is one such term for each nonzero bit of n; so there are at most l terms altogether. Adding each successive term to the sum of all the previous terms requires k bit operations, because when the term to be added is a copy of m left-shifted r places, the rightmost r bits of the new sum are obtained by just copying down the rightmost r bits of the previous sum. The bit operations really only start in column $r + 1$.

$$\begin{array}{rl}
\text{previous sum:} & \text{xxxxxxxxxxxxxxxxxxxxxxxxxxxx x} \\
\text{shifted row:} & \text{xxxxxxxxxxxxxx} \\
 & \underbrace{\phantom{\text{xxxxxxxxxxxxxx}}}\underbrace{\phantom{\text{xxxxxxxxxxxxxx}}} \\ \hline
\text{new sum:} & \quad\text{bit operations} \qquad\qquad \text{copy}
\end{array}$$

So the upshot is that we do at most $k(l - 1)$ bit operations (plus book-keeping).

Since $k = [\log_2 m] + 1$ and $l = [\log_2 n] + 1$ we conclude that

$$\left(\left[\frac{\ln m}{\ln 2}\right] + 1\right)\left[\frac{\ln n}{\ln 2}\right]$$

is an upper bound for the number of bit operations required. If there are fewer than l terms to add then the number of bit operations is less than this.

* Subtraction uses a slightly different algorithm that also counts as a bit operation.

In fact we do not need a very precise estimate. For our purposes, it is sufficient to observe that the long multiplication algorithm for computing the product of a k-bit integer and an l-bit integer uses fewer than kl bit operations.

(7.2) ***Example:*** Assuming our estimate above for the number of bit operations involved in multiplication, let us now estimate the number of bit operations required for an exact computation of $n!$, if n is a k-bit integer.

We have to multiply n integers, all of which have at most k bits. The product is certainly less than $(2^k)^n = 2^{kn}$, since the largest k-bit integer is $2^k - 1$. So $n!$ has at most nk bits. Now computing $n!$ exactly requires multiplying $(j-1)!$ by j, for $j = 2, 3, \ldots, n$. Each step involves, at worst, multiplying an nk-bit integer by a k-bit integer, and hence at most nk^2 bit operations. And there are $n-2$ steps; so in all at most $(n-2)nk^2$ bit operations are used. Assuming that n is large, we may as well say that our upper bound is n^2k^2 (since $(n-2)nk^2$ is not much smaller). Remembering that $k \approx \log_2 n$, our estimate is $n^2(\log_2 n)^2$.

We have certainly overestimated, but not by as much as you may think. It can be shown that $n(\ln n - 1)$ is a reasonable approximation to $\ln(n!)$ for large values of n.* Dividing through by $\ln 2$, this tells us that $n(\log_2 n - 1.44) = n(k - 1.44)$ is a reasonable approximation to $\log_2(n!)$, the approximate number of bits in $n!$. Putting the $(k-1)$-bit number $n/2$ in place of the k-bit number n gives $(n/2)(k - 2.44)$ as an estimate for the number of bits in $(n/2)!$. So just in going from $(n/2)!$ to $n!$ one has to do $n/2$ multiplications in which one factor has at least $k-1$ bits and the other at least $(n/2)(k-2.44)$ bits. So we obtain $(n/2)^2(k-2.44)(k-1)$ as a lower bound for the number of bit operations needed.

---

§36 Big O Notation

Let f and g be positive valued functions. We say that "$f(n)$ is $O(g(n))$ as $n \to \infty$" if $f(n)$ is less than a constant times $g(n)$ for all n large enough.

For example, $2n^4 + 20n^3 + 300n^2 + 7500$ is $O(n^4)$ as $n \to \infty$, since for all $n > 20$ we have $20n^3 < n^4$ and $300n^2 < n^4$, and also $7500 < n^4$. So $2n^4 + 20n^3 + 300n^2 + 7500 < 5n^4$ for all $n > 20$.

The O stands for "order", which in this context means "size". In our typical usage of the big O notation, $f(n)$ will be the number of steps involved in an algorithm to perform some task that depends on the integer n (such as factorizing n, or determining whether n is prime). The aim is usually to understand how quickly $f(n)$ grows as n increases, by comparing $f(n)$ with some quantity $g(n)$ whose growth rate is familiar to us. From the point of view of practical computation it is best if the growth rate of $f(n)$ is small; the smaller it is, the more chance there will

* For example, MAGMA tells me that $\ln(10000!)$ is approximately 82108.9, while $10000(\ln 10000 - 1)$ is approximately 82103.4.

be of applying the algorithm for large values of n. Roughly speaking, if $f(n)$ is $O(g(n))$ then $f(n)$ is no worse than $g(n)$, as far as growth rate is concerned.

You could reasonably complain here that if $f(n) < 100g(n)$ for all large enough values of n then $f(n)$ is $O(g(n))$, but $f(n)$ could be almost 100 times greater than $g(n)$, and hence 100 times worse when it comes to calculation. This is true, but in some sense 100 times worse is not all that bad. It may be necessary to run the computer for a hundred days instead of just one, or use a hundred computers instead of just one. The calculation may still be feasible if one is desperate. On the other hand, if $f(n)/g(n) \to \infty$ as $n \to \infty$ then $f(n)$ is seriously worse than $g(n)$ for large values of n, and in this situation $f(n)$ is not $O(g(n))$.

Let us state the definition again, a little more precisely.

(7.3) Definition: Let f and g be positive valued functions defined on $\mathbb{Z}^+$, the set of positive integers. We say that *$f(n)$ is $O(g(n))$ as $n \to \infty$* if there exist real numbers N and K such that $f(n) < Kg(n)$ for all integers $n > N$.

For simplicity we shall usually just say "$f(n)$ is $O(g(n))$", the "as $n \to \infty$" part being understood. In another context a mathematicians might say, for example, "$f(x)$ is $O((x-2)^3)$ as $x \to 2$", but we are not concerned with such contexts.

The three parts of the following proposition are the three main tools we need when using the big O notation.

(7.4) Proposition: (*i*) *If $f_1(n)$ and $f_2(n)$ are both $O(g(n))$ then so is $f_1(n) + f_2(n)$.*
(*ii*) *If $f_1(n)$ is $O(g_1(n))$ and $f_2(n)$ is $O(g_2(n))$ then $f_1(n)f_2(n)$ is $O(g_1(n)g_2(n))$.*
(*iii*) *If $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$ then $f(n)$ is $O(h(n))$.*

Proof. (*i*) There exist constants N_1, N_2, K_1 and K_2 such that

$$f_1(n) < K_1 g(n) \quad \text{for all } n > N_1,$$
$$f_2(n) < K_2 g(n) \quad \text{for all } n > N_2.$$

Put N equal to the larger of N_1 and N_2. Then both of the above inequalities hold when $n > N$, and therefore

$$f_1(n) + f_2(n) < (K_1 + K_2)g(n)$$

for all $n > N$. This proves that $f_1(n) + f_2(n)$ is $O(g(n))$.

(*ii*) There exist N_1, N_2, K_1 and K_2 such that

$$0 \le f_1(n) < K_1 g_1(n) \quad \text{for all } n > N_1,$$
$$0 \le f_2(n) < K_2 g_2(n) \quad \text{for all } n > N_2.$$

So $f_1(n)f_2(n) < K_1K_2g_1(n)g_2(n)$ for all $n > \max(N_1, N_2)$.

(*iii*) We have $f(n) < Kg(n)$ and $g(n) < Lh(n)$ for all n large enough, and these inequalities imply that $f(n) < KLh(n)$ for all n large enough (since $K > 0$). □

Week 8

§37 Time estimates for arithmetic

(8.1) Definition: An algorithm to perform an operation involving integers $n_1, n_2, \ldots, n_r$ of $k_1, k_2, \ldots, k_r$ bits (respectively) is said to be a *polynomial time* algorithm if there exist integers $d_1, d_2, \ldots, d_r$ such that the number of bit operations required is $O(k_1^{d_1}k_2^{d_2}\cdots k_r^{d_r})$.

The standard algorithms for addition and multiplication are polynomial time algorithms, since, as we saw last week, addition of two k-bit integers involves $O(k)$ bit operations, and multiplication of a k-bit integer and an l-bit integer involves $O(kl)$ bit operations.

Polynomial time should be contrasted with *exponential time*, where the bound on the number of bit operations involves powers of 2^{k_i}, or powers of n_i, rather than powers of k_i. (Remember that the size of a k-bit integer is $O(2^k)$.) From a computational point of view, exponential time algorithms are far worse than polynomial time algorithms, since the growth rate of 2^k (as k increases) far exceeds that of k, or k^2, or k^3, or any power of k. Computation with enormous numbers is possible if polynomial time algorithms can be used, but not if exponential time algorithms are needed.

A recursive algorithm that reduces the computation of $f(n)$ to the computation of $f(n-1)$ is bound to be exponential time, since all of $f(1), f(2), \ldots, f(n-1)$ will have to be computed before $f(n)$ is obtained, and this means that the number of steps is $O(2^k)$ for a k-bit integer n. A typical example is computation of $n!$, which, as we have seen, takes $O(n^2(\log_2 n)^2) = O(2^{2k}k^2)$ bit operations. If, on the other hand, a recursive procedure for computing $f(n)$ reduce the number of bits by one at each stage, rather than reducing n by one at each stage, then the number of steps will be proportional to k rather than n, and the end result will be polynomial time (provided that the number of bit operations in each individual step is bounded by a power of k).

As an illustration we estimate the number of bit operations involved in computing the gcd of two k-bit integers. You should recall from the computer tutorials that in practice the Euclidean Algorithm does its job very quickly indeed; so we should find that it is a polynomial time algorithm.

Suppose that a and b are k-bit integers. Thus k is $O(\log_2 a)$. Performing the Euclidean Algorithm, we start by putting $a = r_0$ and $b = r_1$, and then recursively compute r_j for $j \geq 2$ in accordance with

$$r_{j-2} = q_j r_{j-1} + r_j \qquad 0 \leq r_j < r_{j-1}$$

continuing on until $r_j = 0$.

Since the r_j's form a decreasing sequence of natural numbers the process must terminate. The last nonzero r_j is $\gcd(a, b)$.

However, the mere fact that the r_j's decrease does not explain why the Euclidean Algorithm is fast. In principle, a decreasing sequence of natural numbers starting at a very large number can take a very long time to reach zero. It might just go down by 1 each time! There must be some reason why the sequence of r_j's generated by the Euclidean Algorithm actually decreases fast.

(8.2) Lemma: *The sequence of remainders r_j generated by the Euclidean Algorithm has the property that $r_{j+2} < \frac{1}{2}r_j$ for all j.*

Proof. If $r_{j+1} \leq \frac{1}{2}r_j$ then certainly $r_{j+2} < \frac{1}{2}r_j$, since $r_{j+2} < r_{j+1}$. So we may assume that $r_{j+1} > \frac{1}{2}r_j$. But then when we divide r_{j+1} into r_j the quotient is just 1, and so the remainder, which is r_{j+2}, is equal to $r_j - r_{j+1}$. Hence

$$r_{j+2} = r_j - r_{j+1} < r_j - \tfrac{1}{2}r_j = \tfrac{1}{2}r_j,$$

as claimed. □

Dividing a number by 2 decreases the number of bits by 1. The lemma shows that the sequence r_j, far from decreasing by 1 at each step, decreases by one bit every two steps. So the number of steps required to reach 0 is less than twice the number of bits in the integers we started with. But you get big numbers without using many bits: ten million is only a 24 bit integer, and the Euclidean Algorithm for 24 bit integers will take no more than 48 steps, which is surely nothing for a computer. Indeed, a million steps will be next to nothing for a fast computer: so we should be able to rapidly compute gcd's of numbers of a million bits. Huge numbers!

We are getting ahead of ourselves. We have yet to work out how many bit operations are involved in each step. We hope that this too will be dependent on the number of bits rather than the size of the numbers. And so it is. Each step in the Euclidean Algorithm consists of dividing one number by another, and the numbers involved have no more than k bits (since we started with k-bit numbers). Now, like multiplication, division of k-bit numbers can be done in $O(k^2)$ bit operations. We have shown that the number of steps is at most $2k$, which is $O(k)$. So the total number of bit operations is $O(k^3)$.

So for the calculation of the gcd of two million-bit integers, the number of bit operations should be of the order of a million cubed. On an aging laptop it took MAGMA just 3.5 seconds to find the gcd of two million-bit integers.

Of course, a general purpose package like MAGMA is much slower at this sort of thing than a single purpose program would be. A program to just calculate gcd's could be designed to do very little more than the bit operations required for the Euclidean Algorithm; MAGMA has to do a lot of house-keeping.

With a more detailed examination of the Euclidean Algorithm, one can show that the number of bit operations required is actually $O(k^2)$ rather than $O(k^3)$. But the most important point here is that the answer involves powers of k rather than powers of a or b. Remember that k is $O(\log_2 a)$, and this means that powers of k much smaller than powers of a. Or you can say that a is $O(2^k)$, much bigger than k.

§38 Reducing the number of bit operations in multiplication

Suppose that we want to multiply the two $2k$-bit numbers

$$\begin{aligned} a &= (a_{2k-1}a_{2k-2}\dots a_1a_0)_2, \\ b &= (b_{2k-1}b_{2k-2}\dots b_1b_0)_2. \end{aligned}$$

The algorithm we considered in Week 7 needs roughly $4k^2$ bit operations. Now let us write $a = 2^kA_1 + A_0$ and $b = 2^kB_1 + B_0$, where

$$\begin{aligned} A_1 &= (a_{2k-1}a_{2k-2}\dots a_k)_2 \\ A_0 &= (a_{k-1}a_{k-2}\dots a_0)_2 \\ B_1 &= (b_{2k-1}b_{2k-2}\dots b_k)_2 \\ B_0 &= (b_{k-1}b_{k-2}\dots b_0)_2, \end{aligned}$$

observing that this gives

$$ab = 2^{2k}A_1B_1 + 2^kA_1B_0 + 2^kA_0B_1 + A_0B_0. \tag{15}$$

Multiplying by a power of 2 is a shift; no bit operations are involved. But it seems that to calculate ab using Eq. (15) requires doing 4 multiplications of k bit numbers. According to what we did yesterday, these will involve k^2 bit operations each, and we come to $4k^2$ bit operations for ab, the same as we got above. But we can be more cunning, and note that

$$ab = 2^{2k}A_1B_1 + 2^kA_1B_1 + 2^k(A_1 - A_0)(B_0 - B_1) + 2^kA_0B_0 + A_0B_0. \tag{16}$$

If you expand out the middle term in this expression, you will find that several terms cancel, and you get back the same expression as in Eq. (1) above. Now to work out the right hand side of Eq. (16) we have to do several additions and several shifts, and copy several things, but we only have to do three multiplications of k-bit integers. Namely, we have to find A_1B_1, and A_0B_0, and $(A_1 - A_0)(B_0 - B_1)$. When k is large it is the number of multiplications that is the big contributor to the total number of bit operations; the extra additions are insignificant.

The upshot of this is that if we define $M(k)$ to be the number of bit operations required to multiply two k-bit integers, then $M(2k) \approx 3M(k)$ for k large. To be more exact, to compute the right hand side of Eq. (16) we need to do (at most) k bit operations to compute $A_1 - A_0$ and another k to compute $B_1 - B_0$, then $M(k)$ bit operations to compute each of A_1B_1, A_0B_0 and $(A_1 - A_0)(B_0 - B_1)$, and then a further four additions of $2k$-bit integers, making $8k$ bit operations, to get the answer. So we compute the product of two $2k$ bit integers in no more than $3M(k) + 10k$ bit operations. Hence we conclude that

$$M(2k) \le 3M(k) + 10k. \tag{17}$$

We now use induction on l to deduce from Eq. (17) that for all $l \in \mathbb{Z}^+$,

$$M(2^l) \le 10(3^l - 2^l). \tag{18}$$

Observe that in case $l = 1$ this says that $M(2) \le 10$. But as we have already seen, the product of a k-bit integer and an l-bit integer can be computed using no more than kl bit operations; so in fact $M(2) \le 4$. Since $4 < 10$, this starts our induction.

Turning to the inductive step, we assume that l is a positive integer such that

$$M(2^l) \le 10(3^l - 2^l)$$

holds, and set about deducing that $M(2^{l+1}) \le 10(3^{l+1} - 2^{l+1})$. Putting $k = 2^l$ in Eq. (17) we get

$$\begin{aligned} M(2^{l+1}) &\le 3M(2^l) + 10 \times 2^l \\ &\le 30(3^l - 2^l) + 10 \times 2^l, \end{aligned}$$

the second inequality following from the inductive hypothesis. But

$$30(3^l - 2^l) + 10 \times 2^l = 30 \times 3^l - 20 \times 2^l = 10 \times 3^{l+1} - 10 \times 2^{l+1}$$

and so we have shown that

$$M(2^{l+1}) \le 10(3^{l+1} - 2^{l+1}),$$

which completes the induction.

In particular, the above shows that $M(2^l) < 10 \times 3^l$, so that $M(2^l)$ is $O(3^l)$. Now recall that $3 = 2^{\log_2 3}$, and therefore

$$3^l = (2^{\log_2 3})^l = (2^l)^{\log_2 3}.$$

So what we have shown is that $M(2^l)$ is $O((2^l)^{\log_2 3})$, and, putting k in place of 2^l, this tells us that $M(k)$ is $O(k^{\log_2 3})$. Since $\log_2 3 \approx 1.582$, this is a distinct improvement over our previous estimate of $O(k^2)$.

We remark that in fact there is an algorithm that multiplies two k-bit numbers using $O(k(\log_2 k)(\log_2(\log_2 k)))$ bit operations. Naturally, this algorithm is more complicated than the one we have described, and the numbers to be multiplied have to be large before it becomes worthwhile to use such an algorithm.

(8.3) ***Example:*** *Let n be a positive integer. Show that the residue of 2^n modulo n can be computed in polynomial time.*

Solution. Suppose that n has k bits, and let $(d_{k-1}d_{k-2}\dots d_1d_0)_2$ be the binary representation of n. This means that

$$n = d_0 + d_1 2 + d_2 2^2 + \cdots + d_{k-1}2^{k-1} = \sum_{i=0}^{k-1} d_i 2^i$$

where each d_i is either 0 or 1. Erasing the terms with $d_i = 0$ leaves

$$n = T_1 + T_2 + \cdots + T_l$$

where $l \le k$ and the T_i are powers of 2, the largest of which is 2^{k-1}. (For example, if $n = 89$ in base 10 then $k = 7$ and $n = (1011001)_2$, giving $n = 2^6 + 2^4 + 2^3 + 2^0$.) Hence

$$2^n = 2^{T_1}2^{T_2}\cdots 2^{T_l} \equiv R_1R_2\cdots R_l \pmod{n}$$

where R_i is the residue of 2^{T_i} modulo n.

If we start with the number 2, and repeatedly square and find the mod n residue of the answer, then we obtain successively the mod n residues of 2, 2^2, 2^4, 2^8, and so on. Each term is less than n, and so has at most k bits. Squaring a k-bit integer certainly requires no more than k^2 bit operations, and produces a $2k$-bit answer. Dividing this answer by the k-bit integer n to find the residue takes no more than $(2k)k = 2k^2$ bit operations. So each successive term is found using at most $3k^2$ bit operations, and in at most $k - 1$ steps we will have found the mod n residues of all the numbers 2^T such that T is a power of 2 less than 2^k.

So with fewer than $3k^2(k-1)$ bit operations, all the numbers R_i above can be found. To multiply them all together and reduce mod n at each stage requires $k - 1$ repetitions of a process that needs at most $3k^2$ bit operations. So with a further $3k^2(k-1)$ bit operations, making $6k^2(k-1)$ in all, we obtain the mod n residue of 2^n, as desired.

Our algorithm has used $O(k^3)$ bit operations; so it is certainly polynomial time.

---

§39 Factorization using the Pollard Rho method

So far we have discussed only two techniques for finding a nontrivial factor of a given integer n, namely trial division and Fermat's method. Either of these methods could succeed very quickly: we might be lucky and find a factor straight away.

But in the worst case both take exponential time. Trial division could require $\sqrt{n}$ iterations, Fermat's method could require as many as $(n/2) - \sqrt{n}$.

Proving that a number n is composite turns out to be a much simpler task than finding a factor of n. Indeed, Example (8.3) shows that the residue of 2^n modulo n can be found in polynomial time, and if the answer is not 2 then Fermat's Little Theorem tells us that n is not prime. Most composite numbers can be proved composite in this way. The problem of finding a nontrivial factor of a number that is known to be composite is very important, and many people have spent much time trying to find good algorithms to use. At present there is no known algorithm that is guaranteed to succeed in polynomial time. The composite numbers that are most difficult to factorize are those that have very few prime factors, especially those that are of the form pq, where p and q are distinct primes.

The Pollard Rho algorithm, which we shall describe in this section, is a *Monte Carlo algorithm*, which means that it is not guaranteed to be successful. Sometimes the algorithm will terminate without finding a factor of n, even though n is composite. But this is a rare occurrence, and the Pollard Rho algorithm is in fact one of the best factorization algorithms known. The `Factorization` function in MAGMA uses a combination of trial division, Pollard Rho and the quadratic sieve.

Let n be a large composite integer whose factorization is not known, and let p be its smallest prime factor. Suppose that we have some procedure for generating a sequence of natural numbers $(t_0, t_1, t_2, \ldots)$, as many terms as we want, with $t_i < n$ for all i. By the pigeonhole principle it is not possible for all of t_0, t_1, …, t_p to be distinct from each other modulo p, and so there must be some i, j with $0 \le i < j \le p$ and $t_i \equiv t_j \pmod{p}$. If we are unlucky, then t_i will actually be equal to t_j, but if they are not equal then $0 < |t_j - t_i| < n$, and since $|t_j - t_i|$ and n are both multiples of p it follows that $\gcd(|t_j - t_i|, n)$ is a nontrivial proper divisor of n.

In the absence of any reason to the contrary, we can perhaps expect that all numbers in the set $\{0, 1, \ldots, n-1\}$ will occur with roughly the same frequency in the sequence $(t_0, t_1, t_2, \ldots)$. If this is the case, then the probability that $t_1 \not\equiv t_0 \pmod{p}$ is $(p-1)/p$, since modulo p there are just p different possible values for t_1, and only one of them is the same as t_0. Assuming that $t_1 \not\equiv t_0 \pmod{p}$, the probability that $t_2 \not\equiv t_1 \pmod{p}$ and $t_2 \not\equiv t_0 \pmod{p}$ is $(p-2)/p$. And if t_0, t_1 and t_2 are all distinct mod p, then the probability that t_3 is not congruent to any of t_0, t_1, t_2 is $(p-3)/p$. And we can go on like this.

Of course, the sequence of t_i's is not going to be truly random; so the above calculations are not guaranteed to have any validity. We are really just hoping. And, being even more optimistic, we can hope that the events that t_1 is not congruent to $t_0 \pmod{p}$, and that t_2 is not congruent to either t_1 or $t_0 \pmod{p}$, and that t_3 is not congruent to t_2, t_1 or $t_0 \pmod{p}$, and so on, are all independent of each other. Then the probability that t_0, t_1, …, t_j are all distinct modulo p is $(p-1)(p-2)\cdots(p-j)/p^j$. By the time that j is about the square root of p this quantity will be about 0.6; by the time that j is about twice the square root of p it will be about 0.135. So it will probably be the case that in some number of steps that is of the order of the square root of p, some t_j will turn out to be congruent modulo p to some earlier t_i. In any event it will take at most p steps for this to happen. If we are unlucky we shall actually have $t_j = t_i$. Now if $n = pd$ then there are d numbers less than n that are congruent to t_i modulo p; so the probability that $t_j = t_i$ should be only about $1/d$. Since we are dealing with very large numbers, $1/d$ is very close to 0. So the chances are good that $\gcd(t_j - t_i, n)$ will give us a divisor of n that is greater than 1 and less than n.

This is all very well, but to find this divisor it seems that we are going to have to compute $\gcd(t_j - t_i, n)$ for all pairs i and j. We know that computing gcd's is very fast, but computing this many gcd's is not acceptable. If j is roughly $\sqrt{p}$, then the number of gcd's to compute will be roughly $(\sqrt{p})^2 = p$. We may as well go through all the numbers m from 1 to p, computing $\gcd(m, n)$ in each case, because it will be just as fast. We need some way of avoiding computing all these gcd's and yet still finding the factor.

Here is the trick. We choose some polynomial function $f(x)$—the usual choice is $f(x) = x^2 + 1$, since it is easy to compute, but other polynomials of degree greater than 1 may work just as well—and some initial value t_0. Then our "random" sequence is defined by letting t_i be the residue of $f(t_{i-1})$ modulo n, for all $i \geq 1$. This is really not at all random, but in practice it seems to behave as if it is random, as far as the probabilistic arguments above are concerned. Now the point is that since f is a polynomial function, if $t \equiv s \pmod{p}$ then $f(t) \equiv f(s) \pmod{p}$ (by Theorem (2.4)). Furthermore, since $p|n$ (by the choice of p), if two numbers are congruent modulo n then they are also congruent modulo p. In particular, $t_i \equiv f(t_{i-1}) \pmod{p}$ for all positive integers i. So it follows that if $t_j \equiv t_i \pmod{p}$ then

$$t_{j+1} \equiv f(t_j) \equiv f(t_i) \equiv t_{i+1} \pmod{p}$$

By another application of the same idea, we deduce also that $t_{j+2} \equiv t_{i+2} \pmod{p}$, and then $t_{j+3} \equiv t_{i+3} \pmod{p}$, and so on. Consequently, if $\gcd(t_j - t_i, n) > 1$ then we shall have $\gcd(t_{j+k} - t_{i+k}, n) > 1$ also, for all $k \in \mathbb{Z}^+$. This means that we do not have to calculate $\gcd(t_j - t_i, n)$, so long as we calculate $\gcd(t_{j+k} - t_{i+k}, n)$ for some k. Better still, we have the following result.

(8.4) Proposition: *Suppose as above that* $(t_0, t_1, t_2, \ldots)$ *is a sequence of natural numbers satisfying* $t_i \equiv f(t_{i-1}) \pmod{n}$ *for all* $i \geq 1$*, where* f *is a polynomial function. Suppose also that* $i < j$ *and* $t_j \equiv t_i \pmod{p}$ *for some* p *that is a divisor of* n*. Then there exists a number* l *such that* $i \leq l < j$ *and* $t_{2l} \equiv t_l \pmod{p}$.

Proof. Let $j - i = m$. Any set of m consecutive integers has to include some multiple of m; so one of the numbers $i, i + 1, i + 2, \ldots, i + m - 1 = j - 1$ must be a multiple of m. Let l be this number, so that $i \leq l < j$ and $l = qm$ for some q.

Since $t_j \equiv t_i \pmod{p}$, it follows, as explained above, that $t_{j+k} \equiv t_{i+k} \pmod{p}$ for all natural numbers k. Since $j = m + i$, this can be written as $t_{m+i+k} \equiv t_{i+k} \pmod{p}$ for all $k \in \mathbb{N}$; that is, $t_{m+h} \equiv t_h \pmod{p}$ for all integers $h \geq i$. In particular, the following all hold:

$$\begin{aligned} t_{m+l} &\equiv t_l \pmod p, \\ t_{m+(m+l)} &\equiv t_{m+l} \pmod p, \\ t_{m+(2m+l)} &\equiv t_{2m+l} \pmod p, \\ t_{m+(3m+l)} &\equiv t_{3m+l} \pmod p, \\ &\vdots \\ t_{m+((q-1)m+l)} &\equiv t_{(q-1)m+l} \pmod p. \end{aligned}$$

Hence it follows that

$$t_{2l} = t_{qm+l} \equiv t_{(q-1)m+l} \equiv \cdots \equiv t_{2m+l} \equiv t_{m+l} \equiv t_l \pmod p,$$

as required. □

This means that we only have to compute $\gcd(t_{2l} - t_l, n)$ for $l = 1, 2, 3, \ldots,$ until we find one that is greater than 1. It may turn out to be n; if so, we are just unlucky. And it may be that the task is too large, and we have to give up before we find a factor. But sometimes it works, and in practice it seems to work often enough to make this method worth trying before turning to other, more complicated, strategies.

This method is called the Pollard Rho method, for a rather silly reason. Draw a dot on a piece of paper to represent the residue of x_0 mod p, then another dot to represent the residue of x_1 and so on, joining up the dots. When you get $x_j \equiv x_i \pmod p$, the dot for x_j is meant to go on top of the dot for x_i. So the line joining x_0 to x_1 to x_2, and so on, joins back onto itself at x_i. Someone apparently decided that this makes the Greek letter rho (ρ). It seems to me that it could equally well have been sigma (σ).

How much work to we expect to have to undertake to find a factor by Pollard Rho? The prime factor p that we are looking for is certainly less than $\sqrt{n}$, and according to our (unreliable) probability arguments above, we need to compute $\gcd(t_{2l} - t_l, n)$ for all numbers l from 1 to (perhaps) $\sqrt{p}$. So maybe $\sqrt[4]{n}$ steps, each step involving some number of bit operations that will be a smallish power of $\log_2 n$. It is certainly an improvement over dividing n by all the integers up to the square root of n. Except that it is not guaranteed to work.

§40 Polynomial congruences

Polynomial congruences are congruences of the form

$$a_0 + a_1x + a_2x^2 + \cdots + a_dx^d \equiv 0 \pmod m \tag{19}$$

where the a_i are integers, and $m \in \mathbb{Z}^+$.

Note that replacing the a_i by their residues mod m will not change the set of solutions of (19). Also, anything congruent (mod m) to a solution will also be a solution.

Here is an alternative terminology. Let $f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_dx^d$ be a polynomial with integer coefficients. An integer c is called a *mod m root of $f(x)$* if $f(c) \equiv 0 \pmod{m}$. (That is, if $x = c$ is a solution of (19) above.)

(8.5) ***Example:*** *Find all the mod 7 roots of $x^3 + 3x^2 + 4$.*

Solution.

x	0	1	2	3	4	5	6
x^2	0	1	4	2	2	4	1
x^3	0	1	1	6	1	6	6
$x^3 + 3x^2 + 4$	4	1	3	2	4	1	6

There are no solutions of $x^3 + 3x^2 + 4 \equiv 0 \pmod{7}$.

We can answer questions of the kind in Example (8.5) since we know that a number c is a mod m root of $f(x)$ if and only if its residue mod m is also a mod m root. Since there are only a finite number of residues modulo m, there are only a finite number of values we have to test in order to find all the mod m roots. If m is large, proceeding this way will take a Very Long Time. But it is not clear whether there is anything better that we can do.

(8.6) ***Example:*** *Find all solutions of $x^2 + 1 \equiv 0$ (mod 29).*

Solution. $1^2 = 1$, $2^2 = 4$, $3^2 = 9$, $4^2 = 16$, $5^2 = 25$. Note that this is congruent to -2^2 modulo 29. Now $2 \times 15 \equiv 1 \pmod{29}$; so $2^2 \times 15^2 \equiv 1 \pmod{29}$ also. Hence $5^2 \times 15^2 \equiv -2^2 \times 15^2 \equiv -1 \pmod{29}$. So $75^2 \equiv -1 \pmod{29}$, and since $75 \equiv 17 \equiv -12$ we conclude that $12^2 \equiv 17^2 \equiv -1 \pmod{29}$.

But we should make certain that there are no other roots of $x^2 + 1$ mod 29. We got up to 5 above. $6^2 \equiv 7$, $7^2 \equiv 20$, $8^2 \equiv 6$, $9^2 \equiv 23$, $10^2 \equiv 13$, $11^2 \equiv 5$, $12^2 \equiv -1$, $13^2 \equiv 24$, $14^2 \equiv 22$. After this we shall just get the same numbers again, since $15^2 \equiv (-14)^2 \equiv 14^2$, $16^2 \equiv 13^2$, and so on.

(8.7) Theorem: *Let p be a prime and $f(x) = a_dx^d + a_{d-1}x^{d-1} + \cdots + a_1x + a_0$ a polynomial with integer coefficients. Assume that $a_d \not\equiv 0 \pmod{p}$, so that modulo p the degree of $f(x)$ is d. Then the number of residues modulo p that are mod p roots of $f(x)$ is at most d.*

Proof. We use induction on d. If $d = 0$ then $f(x) = a_0$, with $a_0 \not\equiv 0 \pmod p$. Thus $f(x)$ is constant, and no value of x makes $f(x) \equiv 0 \pmod p$. So when $d = 0$ there are no mod p roots of $f(x)$, as required to start our induction.

Now let $d > 0$, and assume (inductively) that polynomial congruences of degree $d - 1$ can have at most $d - 1$ mod p roots. Let $c_1, c_2, \ldots, c_n$ be all the residues that are mod p roots of $f(x)$; our aim is to prove that $n \le d$. (Note that we are assuming that the c_i are all different from one another.) If $n = 0$ there is nothing to prove, and so we can assume that $n \ge 1$. Put $c = c_n$.

We find that

$$\begin{aligned}
f(x) - f(c) &= (a_d x^d + a_{d-1}x^{d-1} + \cdots + a_1 x + a_0) - (a_d c^d + a_{d-1}c^{d-1} + \cdots + a_1 c + a_0)\\
&= a_d(x^d - c^d) + a_{d-1}(x^{d-1} - c^{d-1}) + \cdots + a_2(x^2 - c^2) + a_1(x - c)\\
&= a_d(x-c)(x^{d-1} + x^{d-2}c + \cdots + xc^{d-2} + c^{d-1})\\
&\qquad + a_{d-1}(x-c)(x^{d-2} + x^{d-3}c + \cdots + xc^{d-3} + c^{d-2})\\
&\qquad + \cdots\\
&\qquad + a_2(x-c)(x+c)\\
&\qquad + a_1(x-c)\\
&= (x-c)(a_d x^{d-1} + (a_d c + a_{d-1})x^{d-2} + \cdots)\\
&= (x-c)g(x)
\end{aligned}$$

where all we need to know about the polynomial $g(x)$ is that it has degree $d - 1$, and the coefficient of x^{d-1} is a_d. In particular, the coefficient of x^{d-1} is not congruent to 0 modulo p, and so the inductive hypothesis tells us that there are at most $d - 1$ residues that are mod p roots of $g(x)$.

Since $f(c_i) \equiv 0 \equiv f(c) \pmod p$ for each $i \in \{1, 2, \ldots, n-1\}$, putting $x = c_i$ in the calculation above gives

$$0 \equiv f(c_i) - f(c) = (c_i - c)g(c_i)$$

for each i. That is, p is a divisor of $(c_i - c)g(c_i)$. But $p \nmid (c_i - c)$, since we assumed that the residues c_i and $c_n = c$ are distinct. Since p is prime, it follows that $p \mid g(c_i)$ (by the rule that if a prime p divides a product ab it must either divide a or divide b). So $g(c_i) \equiv 0 \pmod p$ for all $i \in \{1, 2, \ldots, n-1\}$. But since our inductive assumption tells us that there can be at most $d - 1$ residues that are mod p roots of $g(x)$, it follows that $n - 1 \le d - 1$. Hence $n \le d$. That is, the number of mod p roots of the polynomial $f(x)$ is at most d, as required. □

(8.8) Remark: The statement of Theorem (8.7) is simplified if it is formulated in terms of residue arithmetic. It says that in residue arithmetic modulo a prime p, polynomial of degree d has at most d roots.

Week 9

§41 Powers of residues modulo a prime.

In our discussion of the Chinese Remainder Theorem we made use of the "pigeonhole principle", which says that if you have the same number of pigeons as pigeonholes, and each hole can only accommodate one pigeon, then to put every pigeon in a hole you must give every hole a pigeon. Here is a generalized version of it.

Egg Box Principle. If you have N eggs and can put at most e eggs in each box then you need at least N/e boxes.

I trust that this principle is obvious.

(9.1) Definition: An integer b is said to be an *e-th power residue modulo p* if $0 \le b < p$ and there is an integer a such that $a^e \equiv b \pmod{p}$.

In other words, an e-th power residue is a residue that is an e-th power in residue arithmetic modulo p.

We can use the egg box principle to prove the following proposition.

(9.2) Proposition: *Let p be a prime. For each positive integer e there are at least $(p-1)/e$ nonzero e-th power residues mod p.*

Proof. Let B be the set of nonzero e-th power residues mod p. That is,

$$B = \{\, b \in \{1, 2, \ldots, p-1\} \mid b \equiv a^e \pmod{p} \text{ for some } a \,\}.$$

We must show that B has at least $(p-1)/e$ elements.

We think of the elements of B as labels for boxes, and think of the integers from 1 to $p-1$ as eggs. So we have $p-1$ eggs. Since p is prime, if $a \not\equiv 0 \pmod{p}$ then also $a^e \not\equiv 0 \pmod{p}$ (by Corollary (2.7)). So for each egg a, the residue of a^e (mod p) is in the set B, and is thus a label for a box. We put a in this box. In this way, every egg goes in a box. Now consider a fixed $b \in B$. An egg a will go in the box labelled b if and only if $a^e - b \equiv 0 \pmod{p}$. Equivalently, egg a goes into box b if and only if a is a mod p root of the polynomial $x^e - b$. But we have proved (see Theorem (8.7)) that if $f(x)$ has degree e then $f(x) \equiv 0 \pmod{p}$ has at most e solutions in $\{0, 1, \ldots, p-1\}$. So there are at most e residues mod p that are solutions of $x^e - b \equiv 0 \pmod{p}$; so at most e eggs go into box b. By the egg box principle, there are at least $(p-1)/e$ boxes, as required. □

We can use Proposition (9.2) and Fermat's Little Theorem to show that if d is a divisor of $p-1$ then there are exactly d residues mod p that are d-th roots of 1 in residue arithmetic mod p.

(9.3) Theorem: *Let p be a prime. For each divisor d of $p-1$ there are at exactly d nonzero residues mod p satisfying the congruence $x^d \equiv 1 \pmod{p}$.*

Proof. Throughout this proof we shall employ residue arithmetic modulo p, so that the congruence $x^d \equiv 1 \pmod{p}$ becomes the equation $x^d = 1$. Note also that Fermat's Little Theorem, which says that $a^{p-1} \equiv 1 \pmod{p}$ whenever a is an integr such that $p \nmid a$, tells us that the residue arithmetic equation $a^{p-1} = 1$ holds whenever a is a nonzero residue mod p.

Write $p-1 = de$, and suppose that there are exactly h nonzero e-th power residues mod p. By Proposition (9.2) the number h is at least as large as $(p-1)/e = d$.

Let $b_1, b_2, \ldots, b_h$ be the nonzero e-th power residues mod p. For each of these b_i we can find a residue a_i such that $b_i = a_i^e$ in residue arithmetic mod p. Then clearly $a_i \neq 0$, since $a_i^e = b_i \neq 0$. But now raising both sides of the equation $b_i = a_i^e$ to the power d gives

$$b_i^d = (a_i^e)^d = a_i^{de} = a_i^{p-1} = 1,$$

by Fermat's Little Theorem (since $a_i \neq 0$).

We thus have h pairwise distinct residues mod p satisfying $x^d = 1 \pmod{p}$ in residue arithmetic. But Remark (8.8) told us that a polynomial equation of degree d can have at most d solutions in residue arithmetic mod p; so we conclude that $h \leq d$. Since we showed above that $h \geq d$, we conclude that $h = d$ and that $x^d = 1$ has exactly d solutions in residue arithmetic mod p, as claimed. □

Incidentally, the above proof shows also that if $e \mid (p-1)$ then there are exactly $(p-1)/e$ nonzero e-th power residues mod p. And then, going back to the egg box argument of the previous proposition, since there are $p-1$ eggs and $(p-1)/e$ boxes that can each contain at most e eggs, each box must be filled. Recall that the eggs that go in box b are the roots of $x^e = b$. We conclude that when $e \mid (p-1)$, for any given e-th power residue b the number of e-th roots of b is exactly e.

§42 Primitive roots and discrete logarithms

Recall once again the following terminology (which we met in Week 2 and Week 5). If $a, m \in \mathbb{Z}$ and $\gcd(a,m) = 1$, then $\mathrm{ord}_m(a)$ is the least positive integer k such that $a^k \equiv 1 \pmod{m}$. The Euler–Fermat Theorem ensures that $a^{\phi(m)} \equiv 1 \pmod{m}$, but this does not mean that $\mathrm{ord}_m(a) = \phi(m)$, since there could be some positive integer $k < \phi(m)$ satisfying $a^k \equiv 1 \pmod{m}$.

In general, a number b is called a *primitive root* modulo m if $\gcd(b,m) = 1$ and $\mathrm{ord}_m(b) = \phi(m)$. The case that $m = p$, a prime, is the most important. In this case $\phi(p) = p-1$, and the Euler–Fermat Theorem reduces to Fermat's Little Theorem: $a^{p-1} \equiv 1 \pmod{p}$ whenever $p \nmid a$. And a primitive root mod p is a number b whose order modulo p is $p-1$.

Our immediate objective is to show that primitive roots exist for every prime. We remark that most moduli do not have primitive roots; in fact, m has a primitive root if and only if m is 2, or 4, or a power of an odd prime, or twice a power of an odd prime. But we only care about the case that m is prime.

We have already observed in Week 2 (see Example (2.16), Definition (2.17) and Example (2.18)) and also in Week 6 (see Example (6.1)) that if a and m are coprime then the sequence of powers of a, when computed in residue arithmetic mod m, is periodic with period $\mathrm{ord}_m(a)$. Nevertheless, it will do no harm to prove this fact again.

(9.4) Lemma: *Let a and m be coprime positive integers, and let $\mathrm{ord}_m(a) = k$. Then for all natural numbers n,*

$$a^n \equiv 1 \pmod{m} \textit{ if and only if } k|n.$$

Proof. Suppose that $k|n$. Then $n = kl$ for some integer l. Now $\mathrm{ord}_m(a) = k$ tells us, in particular, that $a^k \equiv 1 \pmod{m}$, and raising this to the power l gives

$$a^n = a^{kl} = (a^k)^l \equiv 1^l \equiv 1 \pmod{m}.$$

This proves the "if" half of the statement.

Conversely, suppose that $a^n \equiv 1 \pmod{m}$, and write $n = kq + r$ with $0 \le r < k$. That is, r is the residue of n mod k. Since $a^k \equiv 1 \pmod{m}$ we see that

$$1 \equiv a^n = a^{kq+r} = a^{kq}a^r = (a^k)^q a^r \equiv 1^q a^r \equiv a^r \pmod{m}.$$

Note that since k is by definition the least positive integer such that the corresponding power of a is congruent to 1 modulo m, if $0 < r < k$ then $a^r \not\equiv 1 \pmod{m}$. Since $0 \le r < k$ and we have shown that $a^r \equiv 1 \pmod{m}$, we are forced to conclude that $r = 0$. Since r is the residue of n mod k, this means that $k|n$, as required. □

For later reference we state a mild generalization of the above lemma:

(9.5) Corollary: *Let a and m be coprime positive integers, and let $\mathrm{ord}_m(a) = k$. Then for all natural numbers n_1 and n_2,*

$$a^{n_1} \equiv a^{n_2} \pmod{m} \textit{ if and only if } n_1 \equiv n_2 \pmod{k}.$$

Proof. Without loss of generality we may suppose that $n_1 \ge n_2$. Since $\gcd(a, m) = 1$ it follows that $\gcd(a^{n_2}, m) = 1$, and hence by coprime cancellation we see that $a^{n_1} \equiv a^{n_2} \pmod{m}$ implies $a^{n_1-n_2} \equiv 1 \pmod{m}$. Conversely, if $a^{n_1-n_2} \equiv 1 \pmod{m}$ then multiplying through by a^{n_2} gives $a^{n_1} \equiv a^{n_2} \pmod{m}$. So $a^{n_1} \equiv a^{n_2} \pmod{m}$ if and only if $a^{n_1-n_2} \equiv 1 \pmod{m}$. Now by the Lemma (9.4), $a^{n_1-n_2} \equiv 1 \pmod{m}$ if and only if $k|(n_1 - n_2)$, and by definition $n_1 \equiv n_2 \pmod{k}$ if and only if $k|(n_1 - n_2)$. Hence $a^{n_1} \equiv a^{n_2} \pmod{m}$ if and only if $n_1 \equiv n_2 \pmod{k}$, as required. □

We now need to recall two things, from Week 6 and Week 7. The first is the "summatory property of the Euler phi function" (Theorem (6.4)): for all $n \in \mathbb{Z}^+$,

$$\sum_{d|n} \phi(d) = n.$$

The second is the Möbius Inversion Formula, which says that if

$$\sum_{e|d} a_e = b_d \quad \text{for all divisors } d \text{ of } n,$$

then

$$a_d = \sum_{e|d} \mu(d/e) b_e \quad \text{for all divisors } d \text{ of } n,$$

where the μ appearing here is the Möbius function.

We can apply Möbius Inversion to the equations given by the summatory property of ϕ. If n is any positive integer then $\sum e|d\phi(e) = d$ holds for all $d|n$; so by Möbius Inversion,

$$\phi(d) = \sum_{e|d} \mu(d/e)e \tag{20}$$

is also true for all $d|n$. Since n could be any integer, is true for all positive integers d.

This does not give us an easier way to evaluate ϕ than we have already, but it is an interesting formula. Let us check it for $d = 20$. Since 2 and 5 are the only prime divisors of 20, we have that $\phi(20) = 20(1 - \frac{1}{2})(1 - \frac{1}{5}) = 8$. According to the formula (2),

$$\phi(20) = \mu(1)20 + \mu(2)10 + \mu(4)5 + \mu(5)4 + \mu(10)2 + \mu(20)1.$$

Since 4 and 20 are not square-free, $\mu(4) = \mu(20) = 0$. The other four factors are square-free, 10 and 1 each having an even number of prime divisors, 2 and 5 each having an odd number of prime divisors. So $\mu(2) = \mu(5) = -1$, and $\mu(10) = \mu(1) = 1$. So the equation above becomes

$$\phi(20) = 20 - 10 - 4 + 2 = 8,$$

which is correct.

We are now able to prove a theorem that tells us, amongst other things, that every prime p has a primitive root. In fact, it tells us that the number of residues mod p that are primitive roots is exactly $\phi(p - 1)$.

(9.6) Theorem: *Let p be a prime, and for each divisor d of $p - 1$ let $F(d)$ be the number of integers a such that $1 \le a < p$ and $\mathrm{ord}_p(a) = d$. Then $F(d) = \phi(d)$.*

Before starting the proof we should note that $\operatorname{ord}_p(a)$ is defined if and only if $\gcd(a, p) = 1$, or, equivalently (since p is prime), $a \not\equiv 0 \pmod{p}$. So the mod p residues a such that $\operatorname{ord}_p(a) = d$ are necessarily nonzero residues.

Proof of Theorem (9.6). Lemma (9.4) tells us that for all d and a,

$$a^d \equiv 1 \pmod{p} \quad \text{if and only if } \operatorname{ord}_p(a) \text{ is a divisor of } d.$$

In particular, for every divisor d of $p - 1$, the number of residues mod p such that $a^d \equiv 1 \pmod{p}$ is the sum over all divisors e of d of the number of residues a such that $\operatorname{ord}_p(a) = e$. This last quantity is $\sum_{e|d} F(e)$, given the definition of F in the statement of the theorem. But Theorem (9.3) says that $a^d \equiv 1 \pmod{p}$ is satisfied by exactly d residues (given that $d \mid (p - 1)$). So we have established that

$$\sum_{e|d} F(e) = d \tag{21}$$

for all $d \mid (p - 1)$.

Compare this with the summatory property of the Euler phi function! We know that

$$\sum_{e|d} \phi(e) = d \tag{22}$$

for all $d \mid (p - 1)$. The quantities $F(e)$ and $\phi(e)$ satisfy exactly the same equations. Just as we can apply Möbius inversion to Eq. (22) and deduce that

$$\phi(d) = \sum_{e|d} \mu(d/e)e$$

for all $d \mid (p - 1)$, so we can also apply Möbius inversion to Eq. (21) and deduce that

$$F(d) = \sum_{e|d} \mu(d/e)e$$

for all $d \mid (p - 1)$. So $F(d) = \phi(d)$ for all $d \mid (p - 1)$, as claimed. □

(9.7) ***Example:*** *Show that 3 is a primitive root mod 17.*

Solution. We must show that $\operatorname{ord}_{17}(3) = 16$. Writing $k = \operatorname{ord}_{17}(3)$, we know that $k \mid 16$, and so if $k \neq 16$ then we must have that $k \mid 8$. (The fact that 16 is a power of a prime has made things easy for us!) If $k \mid 8$ then $3^8 \equiv 1 \pmod{17}$. So we only need to show that $3^8 \not\equiv 1 \pmod{17}$ and it will follow that 3 is indeed a primitive root. In fact, we find that $3^2 = 9$; so $3^4 = 81 \equiv -4 \pmod{17}$, and $3^8 \equiv 16 \equiv -1 \pmod{17}$. Thus 16 is the least positive k with $3^k \equiv 1 \pmod{17}$, as claimed.

It turns out that the smallest primitive root mod 73—the smallest a such that $\mathrm{ord}_{73}(a) = 72$—is $a = 5$. If one were to write out all the mod 73 residues of all the powers of 5, from 5^1 to 5^{72}, one would find that every number from 1 to 72 occurs in the list. This follows from the pigeonhole principle. Since there are 72 powers of 5 to consider, and 72 possible values for the residue, we either get all 72 possible values occurring once, or else some value occurs more than once. But if $5^i \equiv 5^j \pmod{73}$ for some $i, j \in \{1, 2, \ldots, 72\}$ with $i < j$, then by coprime cancellation we deduce that $1 \equiv 5^{j-i} \pmod{73}$, where $0 < j - i < 72$, contradicting the fact that $\mathrm{ord}_{73}(5) = 72$.

Of course there is nothing special about 73 and 5 in all of this: the same is true for any r that is a primitive root mod p.

(9.8) Proposition: *Let p be a prime and b a primitive root mod p. Then as i goes from 0 to $p - 2$, the mod p residue of b^i takes every value from 1 to $p - 1$ exactly once.*

Proof. Let t_i be the residue of $b^i \pmod{p}$. The $p - 1$ numbers $t_0, t_1, \ldots, t_{p-2}$ all lie in the set $\{1, 2, \ldots, p - 1\}$, and so either every value of this set occurs, or else there exist i, j with $0 \le i < j \le p - 2$ and $t_i = t_j$. But $t_i = t_j$ says that $b^i \equiv b^j \pmod{p}$, and then coprime cancellation gives $b^{j-i} \equiv 1 \pmod{p}$. But since $0 < j - i < p - 1$, this contradicts the fact that b is a primitive root: $b^k \equiv 1 \pmod{p}$ for $k = p - 1$, and not for any smaller positive integer k. □

(9.9) Definition: Let p be a prime and b a primitive root mod p. For each nonzero residue a mod p, the *discrete logarithm of a to the base b* is the integer $i \in \{0, 1, 2, \ldots, p - 2\}$ such that $b^i \equiv a \pmod{p}$.

This is in fact exactly what you should expect the logarithm to the base b to mean: $\log_b(a) = i$ if and only if $a = b^i$. The only minor alteration we have to make is that $a = b^i$ is replaced by $a \equiv b^i \pmod{p}$. Or, to put it another way, ordinary arithmetic is replaced by residue arithmetic mod p.

To avoid possible confusion with the more familiar (non-discrete) logarithms, and to emphasize the fact that everything depends on the prime p, we shall denote the discrete logarithm of a to the base b by $\log_{b,p}(a)$.

As with familiar logarithms, the logarithm of zero is not defined. The discrete logarithm function $\log_{b,p}$ is a one-to-one and onto function from the set of nonzero residues modulo p to the set of all residues modulo $p - 1$. The inverse of $\log_{b,p}$ is the function from the set of residues modulo $p - 1$ to the set of nonzero residues modulo p such that each i maps to b^i evaluated using residue arithmetic mod p.

Some students are puzzled by the fact that the discrete log of a residue mod p is a residue mod $p - 1$ rather than a residue mod p. But in fact this is perfectly natural, since $b^i \equiv b^j \pmod{p}$ if and only if $i \equiv j \pmod{p - 1}$, as follows from Corollary (9.5) and the fact that $\mathrm{ord}_p(b) = p - 1$ (since b is a primitive root).

In the literature, the term *index* is also used as a synonym for discrete logarithm.

(9.10) ***Example:*** Taking $p = 73$ and using the primitive root 5, we see that $\log_{5,73}(41) = 4$, since $5^4 = 625 = 584 + 41 = 8 \times 73 + 41 \equiv 41 \pmod{73}$. And $5^5 \equiv 5 \times 41 \equiv 59 \pmod{73}$; so $\log_{5,73}(59) = 5$.

If b is a nonzero residue mod p, but not a primitive root, then $k = \mathrm{ord}_p(b)$ is a divisor of $p - 1$ but not equal to $p - 1$. In this case the powers of b, when computed in residue arithmetic mod p, do not yield all the nonzero residues. Indeed, there are exactly k residues that are powers of b. If a is such a residue, then there is exactly one i in the set $\{0, 1, 2, \ldots, k - 1\}$ such that $a = b^i$ in residue arithmetic mod p, and we write $i = \log_{b,p}(a)$. If $a \in \mathbb{Z}_p$ is not a power of b then $\log_{b,p}(a)$ is undefined.

The problem of computing discrete logarithms is believed to be hard. If a is the residue of $b^i \pmod{p}$, then it is easy and fast to compute the value of a given b and i. But if the task is to find an i such that a is the residue of $b^i \pmod{p}$, where b and a are given, then there is no algorithm known that will always succeed in polynomial time. The situation is analogous to the situation with multiplying and factorizing. Multiplication is fast, factorization is not. And just as the difficulty of factorization is exploited in cryptography, in the RSA cryptosystem, so too is the difficulty of computing discrete logarithms, in the Elgamal cryptosystem.

§43 The Elgamal cryptosystem

This is the second most prevalent public key cryptosystem in use. It works like this. Bob publicly releases a triple (p, b, k), consisting of

- a prime p,
- a number b such that the order of b modulo p has a large prime factor,
- a number k that Bob has computed to be the residue of b^m modulo p.

The number m in this third item Bob keeps secret. It is his private key, which he needs to decrypt messages that people send him. The triple (p, b, k) is his *public key*; anyone who wants to send Bob a secret message can do so by using the public key to encrypt the message.

Encryption of a numerically encoded message is performed as follows. The plaintext must be a positive integer M less than p (or a sequence $(M_1, M_2, \ldots, M_l)$ comprised of such integers). Alice, the message sender, must first randomly choose a positive integer i less than $p - 1$. She then computes the residue of $b^i \pmod{p}$ and the residue of $k^i M \pmod{p}$, where M is the plaintext. These two residues are the two parts of the ciphertext. That is, the ciphertext is a pair (c, N), consisting of two residues mod p, defined by

$$c \equiv b^i \pmod{p}$$

and

$$N \equiv k^i M \pmod{p},$$

where p, b and k are as given in the public key, and i was chosen randomly.

If the plaintext is a sequence $(M_1, M_2, \ldots, M_l)$ rather than a single number M, then the second component of the ciphertext, rather than being a single number N, should be the sequence $(N_1, N_2, \ldots, N_l)$, where N_j is the mod p residue of $k^i M_j$.

Note that the number of possible values for c, as i varies, is equal to the order of b modulo p. The choices of b and p guarantee that this is very large. This is important, since (as we shall see) anyone who intercepts the message and can find a number j such that $c \equiv b^j \pmod{p}$ will be able to decrypt the message. If $\operatorname{ord}_p(b)$ were too small then a suitable j could be found by trying all the natural numbers less than $\operatorname{ord}_p(b)$, one by one.

Everyone knows b and p, and so if there were a good algorithm for computing discrete logarithms then anyone who intercepted the ciphertext could use this algorithm to compute $j = \log_{b,p} c$, thus solving $c \equiv b^j \pmod{p}$. But there is not; so they cannot. And if anyone were able to solve $c \equiv b^j \pmod{p}$ then they would be able to find the mod p residue of k^i, since

$$k^j \equiv (b^m)^j \equiv (b^j)^m \equiv c^m \equiv (b^i)^m \equiv (b^m)^i \equiv k^i,$$

after which they would be able to recover plaintext M by the formula.

$$M \equiv N(k^i)^{-1} \pmod{p}.$$

The difficulty of solving $c \equiv b^j \pmod{p}$ ensures that no eavesdropper can do this. But Bob knows the number m such that $k \equiv b^m$, and since $k^i \equiv c^m \pmod{p}$, and Bob also knows c, he can compute k^i and hence read the message.

Security of the Elgamal system relies on the difficulty of solving $c \equiv b^j \pmod{p}$. The choice of p is important, since there are primes for which this problem is not difficult, even when $\operatorname{ord}_p(b)$ is large. We shall see in due course that if all the prime factors of $\operatorname{ord}_p(b)$ are small then $c \equiv b^j \pmod{p}$ can be solved efficiently. Since $\operatorname{ord}_p(b)$ is a divisor of $p-1$ it is therefore important that $p-1$ should have a large prime factor.

Obviously p must be an odd number, and so $2 \mid (p-1)$. So it will certainly not be true that all the prime factors of $p-1$ are large! But it is possible to choose p so that $\frac{1}{2}(p-1)$ is prime, and the primes p with this property are the best ones for the Elgamal cryptosystem.

(9.11) Definition: A *safe prime* is prime p such that $\frac{1}{2}(p-1)$ is also prime.

If p is a safe prime, so that $p = 2q+1$ with q prime, and if b is a nonzero residue mod p, then $\operatorname{ord}_p(b)$ must be a divisor of $p-1 = 2q$. As long as b is chosen to be anything other than 1 (which would give $\operatorname{ord}_p(b) = 1$) or $p-1$ (which would give $\operatorname{ord}_p(b) = 2$) then $\operatorname{ord}_p(b)$ will necessarily be either q or $2q$. Provided that p is large enough, p and b will then be suitable for use with the Elgamal cryptosystem.

§44 The Diffie–Hellman key agreement protocol

Alice and Bob wish to have a private conversation, but Eve the evil eavesdropper is listening! What can they do?

Alice and Bob agree on a Very Large Prime p and a nonzero residue b mod p. Ideally, p will be a safe prime, in which case b can be chosen to be any integer in the range $1 < b < p - 1$. What is most important is that the order of b modulo p should be divisible by a large prime. For example, if MAGMA (or some other powerful factorizing program) cannot factorize $p - 1$, then you can be sure that $p - 1$ has large prime factors, and then if MAGMA cannot compute the order of b modulo p then surely b is suitable.

Of course, Eve hears Alice and Bob discussing p and b, and so she knows the values they select.

Next Alice thinks of a secret number x; it should be large but less than $\mathrm{ord}_p(b)$. Perhaps it is best if she makes sure that x is coprime to $\mathrm{ord}_p(b)$, as this will guarantee that b^x has the same order modulo p as b. She then computes the residue of b^x modulo p, and sends this to Bob. Now Bob knows p, b and the residue of b^x, but not x. Eve knows everything that Bob knows.

Now Bob thinks of a secret number y (large, less than $\mathrm{ord}_p(b)$, preferably coprime to $\mathrm{ord}_p(b)$), computes the residue of b^y modulo p, and sends this to Alice. Now Alice knows p, b, the residue of b^x, the residue of b^y, and also x; Bob knows p, b, the residue of b^x, the residue of b^y, and also y; Eve knows only p, b, the residue of b^x and the residue of b^y.

Alice and Bob can both easily compute the mod p residue of $b^{xy} = (b^x)^y = (b^y)^x$, but Eve is stymied, since (as far as we know!) there is no computationally feasible way to compute the residue of b^{xy} from the information Eve has.

Now, for example, Alice and Bob can express the residue of b^{xy} in binary notation, use the first 56 bits as a key for DES encryption, the next 56 bits as another DES key, the third 56 bits as another DES key, and then converse happily and securely using triple DES encrypted messages.

Meanwhile, Eve is computing b^2, b^3, b^4, and so in, and in 5000 billion years or so she will probably stumble across b^x or b^y, and, knowing x or y, will finally be able to compute b^{xy}, discover the keys and decrypt the conversation.

The problem that Eve wishes she could solve is called the *Diffie–Hellman Problem*.

Diffie–Hellman Problem: *Given p, b and the mod p residues of b^x and b^y, compute the mod p residue of b^{xy}.*

This problem is probably no easier than the *Discrete Logarithm Problem*.

Discrete Logarithm Problem: *Given p, b and the mod p residue of b^x, find x (or some number z such that $b^z \equiv b^x \pmod{p}$).*

It is universally believed that these problems are hard, in the sense that there is no polynomial time algorithm to solve either of them. However, it must be said that it has not been proved that there is no such algorithm, and we can rest assured that if someone had proved the nonexistence of such an algorithm then everyone would know about it. On the other hand, if someone (such as Eve or the CIA) does discover a polynomial time algorithm to solve either problem, they may well keep the fact secret, and start decrypting other people's messages.

Many good people, and probably many evil people too, are working hard trying to find fast algorithms for the Diffie–Hellman and Discrete Log Problems: the evil people hope to benefit by discovering one, the good people need to feel confident that no evil person has found one.

Week 10

Recall that for each positive integer n we have defined $\mathbb{Z}_n$ to be the set of residues modulo n; that is, $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$. Residue arithmetic modulo n deals only with elements of $\mathbb{Z}_n$. So for $a + b$ and ab to be defined in mod n residue arithmetic, a and b must be elements of $\mathbb{Z}_n$; moreover, if a and b are in $\mathbb{Z}_n$ then $a + b$ and ab, as computed in residue arithmetic mod n, are also elements of $\mathbb{Z}_n$. In effect, when we are using residue arithmetic mod n, the elements of $\mathbb{Z}_n$ are the only numbers in existence.

However, in contexts in which residue arithmetic is being used, we shall find it convenient to identify integers that are not in $\mathbb{Z}_n$ with their residues mod n. Thus, for example, when we are using residue arithmetic mod 47 we regard 125 as an alternative name for 31, and 155 as an alternative name for 14. Accordingly, we say things like "155 is 31 in residue arithmetic mod 47", or "$155 = 31$ in $\mathbb{Z}_{47}$", instead of saying "31 is the residue of 155 mod 47". With this convention we are often able to replace cumbersome statements by simpler ones.

§45 Elgamal examples

Recall the description of the Elgamal cryptosystem given in Week 9. Bill, a user of the system, chooses a prime p, a number b and a secret number m. He computes k, the residue of b^m (mod p), and publicly releases the triple (p, b, k). To send Bill a secret message, Amy numerically encodes the message as M, a residue mod p, then chooses a random number i and computes the numbers c and N that are (respectively) the mod p residues of b^i and k^iM, after which she sends the pair (c, N) to Bill. To decrypt Amy's message Bill multiplies N by the inverse mod p of c^m. The residue of this will be the original message M.

It is obviously important to choose the number b so that its order modulo p is astronomically large. For example, an attacker might hope to find the private key, m, by simply computing the residues of b, b^2, b^3, and so on, until he finds the one that is equal to k. It would be ridiculous to allow this approach any hope of success! The sequence that this attacker computes (the mod p residues of b, b^2, b^3, and so on) has $\operatorname{ord}_p(b)$ distinct terms altogether, and so long as $\operatorname{ord}_p(b)$ is astronomically large the attacker is surely wasting his or her time.

To be more precise, p and b should be chosen so that it is computationally difficult to find discrete logarithms to the base b of residues mod p. Next week we shall investigate strategies for computing discrete logarithms. It turns out that the problem is most difficult when $\mathrm{ord}_p(b)$ has a very large prime factor. The largest such a prime factor can possibly be is $\frac{1}{2}(p-1)$ (since we know that $\mathrm{ord}_p(b)$ is a divisor of $p-1$ and that $p-1$ is even). Primes p such that $\frac{1}{2}(p-1)$ is also prime are known as safe primes because they are safest ones to use with Elgamal. Also, of course, p should be very large: three hundred decimal digits would be suitable.

However, so that we can do an example by hand, let us suppose that Bill chooses $p = 47$ and $b = 5$, and takes $m = 4$ as his private key. Using residue arithmetic mod 47, this gives

$$k = b^4 = 5^4 = 5 \times 125 = 5 \times 31 = 155 = 14.$$

So Bill's public key is $(p, b, k) = (47, 5, 14)$.*

Now suppose that Amy wishes to send Bill the secret message 21. She chooses a random number i less than $p-1$: let us suppose that she chooses $i = 7$. She will scramble her message by multiplying it by k^i (using residue arithmetic mod p). So that Bill will be able to unscramble the message, she must also compute b^i and send that to Bill along with the scrambled message.

We find that

$$b^i = 5^7 = 5^3 \times 5^4 = 31 \times 14 = 62 \times 7 = 15 \times 7 = 105 = 11$$

(in residue arithmetic mod 47, of course). So 11 is the first component of the ciphertext. The scrambling factor is $k^i = 14^7$, and since $14^2 = 196 = 8$ we see that $14^6 = 8^3 = 512 = 42 = -5$, and $14^7 = -70 = 24$. So the scrambled message is $21 \times 24 = 10 \times 48 + 24 = 34$, and the complete ciphertext is the pair $(11, 34)$.

When he receives this Bill computes c^m, where c is the first component of the ciphertext and m is his private key. So he computes 11^4 (in residue arithmetic mod 47). Now $11^2 = 121 = -20$, and so $11^4 = 400 = 24$. Sure enough, this was the scrambling factor. Bill needs its inverse mod p. Since $11^{46} = 1$ in $\mathbb{Z}_{47}$ (by Fermat's Little Theorem), in fact the inverse of 11^4 is 11^{42}. In practice, Bill will find the inverse directly by computing c^{p-1-m}, and not bother about computing c^m. However, since we have found that $c^m = 24$, let us continue on by finding the inverse of 24 modulo 47.

In the present example, the calculation happens to be particularly trivial: since $2 \times 24 = 48 = 1$, Bill sees at once that the inverse of 24 is 2. So he multiplies the second component of the ciphertext by 2. Since

$$2 \times 34 = 68 \equiv 21$$

Bill concludes (correctly) that the original plaintext was 21.

* Because the numbers in this example are so small, it is a trivial task to determine the private key from the public key, by finding $\log_{5,47}(14)$. This is despite the fact that 47 is a safe prime. One simply computes powers of 5 until a solution of $5^m = 14$ is found.

Let us do another example, this time taking $p = 59$ and $b = 2$. Since $\operatorname{ord}_{59}(2)$ must be a divisor of 58, which is twice a prime number, it is immediately clear that $\operatorname{ord}_{59}(2)$ is either 29 or 58 (since it is obviously not 1 or 2, which are the only other divisors of 58).

It is a consequence of the next theorem* that 2 is a primitive root mod 59; in other words, $\operatorname{ord}_{59}(2) = 58$. We mention this only for interest's sake, since from the point of view of the example it would not matter if the order of 2 were 29.

(10.1) Theorem: *Let p be a prime such that $p \equiv 3 \pmod 8$, and suppose that the number $q = \frac{1}{2}(p-1)$ is also prime. Then 2 is a primitive root mod p.*

Proof. By the definition—see §42—saying that a is a primitive root mod p is the same as saying that $\operatorname{ord}_p(a) = p - 1$. Observe that the hypotheses of the theorem are not satisfied with $p = 3$, since $\frac{1}{2}(3-1) = 1$ is not prime; so the assumption that $p \equiv 3 \pmod 8$ implies that $p \geq 11$. (Nevertheless, it is in fact true that 2 is a primitive root mod 3.)

Fermat's Little Theorem tells us that $\operatorname{ord}_p(2)$ is a divisor of $p - 1 = 2q$, and it is clear that $\operatorname{ord}_p(2)$ is not either 1 or 2 (since $2^1 \not\equiv 1 \pmod p$ and $2^2 \not\equiv 1 \pmod p$). It thus follows that $\operatorname{ord}_p(2)$ must be either $2q$ or q, since the fact that q is prime ensures that, apart from 1 and 2, these are the only divisors of $p - 1$. So to prove the theorem it will be sufficient to show that $2^q \not\equiv 1 \pmod p$. We shall show that in fact $2^q \equiv -1 \pmod p$.

Since $p \equiv 3 \pmod 8$ we have $p = 8l + 3$ for some integer l. It is convenient to write $k = 2l$, so that k is even, $p = 4k + 3$ and $q = \frac{1}{2}(p-1) = 2k + 1$. Now observe that $2^q q!$ is the product of all the even numbers from 2 to $2q$, since

$$\begin{aligned} 2 \times 4 \times 6 \times \cdots \times 2q &= (2\times 1) \times (2 \times 2) \times (2 \times 3) \times \cdots \times (2 \times q) \\ &= (2 \times 2 \times 2 \times \cdots \times 2) \times (1 \times 2 \times 3 \times \cdots \times q) \\ &= 2^q q!. \end{aligned}$$

Since $2q = 4k + 2$, the product of the even numbers from 2 to $2q$ can also be written as M_1M_2, where M_1 is the product of the even numbers from 2 to $2k$ and M_2 is the product of all the even numbers from $2k + 2$ to $4k + 2$. Noting that $2k + 2 = p - 2k - 1$ and $4k + 2 = p - 1$, it follows that

$$\begin{aligned} 2^q q! = M_1M_2 &= M_1\big((p-2k-1)(p-2k+1)(p-2k+3)\cdots(p-3)(p-1)\big) \\ &\equiv M_1(-2k-1)(-2k+1)(-2k+3)\cdots(-3)(-1) \pmod p, \end{aligned}$$

since $p - n \equiv -n \pmod p$ for all values of n. Observe that the final right hand side above is the product of all the even numbers from 2 to $2k$ and all the negative odd numbers from -1 to $-(2k+1)$. Since there are exactly $k+1$ odd numbers between -1 and $-(2k+1)$ inclusive, their product equals $(-1)^{k+1}$ times the product of the

* This theorem is not usually regarded as part of the syllabus of MATH2088/2988.

positive odd numbers from 1 to $2k + 1$. Moreover, $(-1)^{k+1} = -1$, since k is even. So we conclude that

$$2^q q! \equiv (-1)\big(2 \times 4 \times \cdots \times (2k)\big)\big(1 \times 3 \times \cdots \times (2k+1)\big) \pmod{p}.$$

Since the right hand side here is $(-1)(2k+1)! = (-1)q!$, our calculations have shown that $2^q q! \equiv (-1)q! \pmod{p}$. But all the factors in $q!$ are positive integers that are less than the prime p, and are therefore coprime to p. So by the coprime cancellation rule (Corollary (2.15)) they can all be cancelled away, leaving $2^q \equiv -1 \pmod{p}$, which is precisely what we were aiming to prove. □

The theorem immediately tells us that 2 is a primitive root modulo 59. Of course it is also easy to prove this directly, by checking that $2^{29} \equiv -1 \pmod{59}$.

Returning now to our promised example, assume that Bill chooses $m = 16$ as his private key, having already chosen $p = 59$ and $b = 2$. To determine the final component of the public key Bill must compute 2^{16} in residue arithmetic mod 59. Now $2^8 = 256 = 20$, and so $2^{16} = 400 = 46$. The public key is therefore $(p, b, k) = (59, 2, 46)$; the private key is $m = 16$.

This time Amy has a longer message to send to Bill, one that is too long to encode as a single residue mod 59. So instead she must split the message into blocks that are each small enough to encode as a single residue, so that the encoded plaintext is a sequence of residues mod 59. Let us suppose that the encoded plaintext is $(8, 20, 30)$.

To construct the ciphertext, Amy must first choose a random number i and compute b^i and k^i in residue arithmetic modulo p. The ciphertext then has two parts; the first part is $c = b^i$, and the second is the sequence obtained by multiplying each term in the plaintext by the "scrambling factor" k^i. Bill will be able to find the scrambling factor by using the formula

$$k^i = c^m.$$

Recall from §43 that Amy's random number i is supposed to be less than $p - 1$. It is clear that choosing $i = p - 1$ would be unsatisfactory, since this would give $c = b^{p-1} = 1$, by Fermat's Little Theorem. Then if Eve were to intercept ciphertext, on seeing that $c = 1$ she would immediately deduce that the scrambling factor k^i is also 1, since $k^i = c^m = 1^m = 1$, irrespective of the value of m. Furthermore, this makes the second component of the ciphertext exactly equal to the plaintext. Hence Eve would discover the plaintext.

It is also unacceptable for c to be $p - 1$, which is the same as -1 in $\mathbb{Z}_p$, since if $c = -1$ then $k^i = c^m = (-1)^m = \pm 1$. In possession of the ciphertext, Eve thus narrows the number of possible values for the scrambling factor to 2, and can test them both to see which gives a meaningful value for the plaintext.

This means that $\frac{1}{2}(p-1)$ is not acceptable as a value for Amy's random number i, since $2i = p - 1$ would give $c^2 = (b^i)^2 = b^{p-1} = 1$, forcing $c = \pm 1$, and neither of these alternatives is acceptable. In the present example, if Amy chose $i = 29$ then she would find that $c = 2^{29} = -1$ and $k^i = 46^{29} = 400^{29} = 20^{58} = 1$, so that the ciphertext would be $(-1, (8, 20, 30))$. She should reject this and try another random number.

The restriction that $c = b^i$ is not allowed to be ± 1 is of zero practical concern. The number of possible values for b^i is $\operatorname{ord}_p(b)$, which in practical applications is always astronomically large. They should all have roughly the same chance of becoming the value of c. So the probability that Amy's randomly chosen i makes b^i equal to 1 should be about $1/\operatorname{ord}_p(b)$, which is so exceedingly close to zero that in practice $b^i = 1$ will never occur. The same remarks apply with -1 in place of 1, the only difference being that if $\operatorname{ord}_p(b)$ is odd then $b^i = -1$ is not even a theoretical possibility.

For the present example, let us suppose that Amy's randomly chosen i turns out to be 19. We find that $c = b^i = 2^{19} = 2^{16}2^3 = 46 \cdot 8 = (-13) \cdot 8 = -104 = 14$. This is the first component of Amy's ciphertext.

Next, using the same i, Amy must compute k^i (still using residue arithmetic modulo 59, of course). This will be the scrambling factor. Since $k = 46 = -13$ we find that

$$\begin{aligned} k^2 &= 169 = 177 - 8 = -8, \\ k^4 &= 64 = 5, \\ k^{16} &= 625 = 590 + 35 = 35, \end{aligned}$$

and then

$$\begin{aligned} k^{19} = k^{16}k^2k = 35 \times (-8) \times (-13) &= (-280) \times (-13) \\ &= 15 \times (-13) = -195 = -18 = 41. \end{aligned}$$

Recalling that the plaintext is $(8, 20, 30)$, we now compute

$$\begin{aligned} 41 \times 8 &= 328 = 295 + 33 = 33 \\ 41 \times 20 &= 820 = 13 \times 59 + 53 = 53 \\ 41 \times 30 &= 1230 = 20 \times 59 + 50 = 50, \end{aligned}$$

so that the scrambled sequence is $(33, 53, 50)$. This is the second component of the ciphertext. So the ciphertext is $(14, (33, 53, 50))$.

To decipher this Bill uses the first component of the ciphertext ($c = 14$) and his private key ($a = 16$) to discover the scrambling factor. He computes

$$14^{16} = (((14^2)^2)^2)^2 = ((196^2)^2)^2 = ((19^2)^2)^2 = (361^2)^2 = (7^2)^2 = (-10)^2 = 41.$$

To undo the scrambling he must multiply each term of the second component of the ciphertext by the inverse of the scrambling multiplier. So he applies the extended

Euclidean Algorithm to 59 and 41:

$$\begin{array}{cccccccc}
59 & 41 & 18 & 5 & 3 & 2 & 1 & 0 \\
 & & 1 & 2 & 3 & 1 & 1 & 2 \\
0 & 1 & 1^- & 3 & 10^- & 13 & 23^- & \\
1 & 0 & 1 & 2^- & 7 & 9^- & 16 &
\end{array}$$

This shows that $41 \times (-23) = 1$ in arithmetic modulo 59, and so $41^{-1} = -23 = 36$. To complete the decryption process Bill now computes

$$\begin{aligned}
36 \times 33 &= 1188 = 1180 + 8 = 8 \\
36 \times 53 &= (-23) \times (-6) = 138 = 118 + 20 = 20 \\
36 \times 50 &= 1800 = 1770 + 30 = 30,
\end{aligned}$$

and recovers the plaintext $(8, 20, 30)$.

§46 Sharing secrets

Imagine the following situation. The combination that opens a bank vault is too important and valuable to be entrusted to a single individual. Instead, it is decided that several trustworthy people should haved to cooperate to open the vault.

One could perhaps give the first few digits of the combination to one person, the next few to another, the next few to a third, and so on. A disadvantage of this is that all the people would have to be present to open the vault. Another disadvantage is that if one person died the combination would be lost; so one would really have to give each part of the combination to at least two people. And a third disadvantage is that if $n-1$ of the n people involved got together and pooled their information, then they might be able to do an exhaustive search to find the remaining digits, and illegally open the vault without involving person n.

There is a better scheme, one that overcomes all these disadvantages. There will be n people sharing the secret. Any k of them will be able to recover the combination by pooling their information. So even if $n-k$ of the people die, the combination will not be lost. And if $k-1$ of the people decide to be collectively dishonest, they will find that pooling their information is no advantage. The exhaustive search that they would still have to do is just as large as the one that a person who knows nothing would have to do.

The basic idea is this. A polynomial function $f(x)$ of degree less than or equal to $k-1$ will be chosen, with the secret number (the combination of the lock) being the constant term, $f(0)$. The n sharers of the secret will each be assigned one of the n numbers $f(1)$, $f(2)$, …, $f(n)$. Because the degree of $f(x)$ is at most $k-1$, it is uniquely determined by any k of its values. But knowing only $k-1$ of the values tells you precisely nothing about the value at some other point, since there exists a (unique) polynomial of degree at most $k-1$ taking the prescribed values at the given $k-1$ points, and taking any arbitrarily chosen value at the k-th point.

The calculations will all be done in residue arithmetic modulo some prime p, which must be bigger than the secret number (so that the secret number is a residue mod p). Thus $f(1)$, $f(2)$, …, $f(n)$ will all be residues mod p. If we were to use ordinary arithmetic, these numbers might turn out to be unreasonably large.

To illustrate the procedure, let us do an example in which the secret number is artificially small. (In practice, of course, security would require that the secret number be so large that an exhaustive search would not be feasible.) So let us suppose that the secret number is 19. There are to be seven people involved, any four of them should be able to determine the secret by cooperating. The calculations are to be done in residue arithmetic modulo 97.

Here is a magma session that randomly chooses a suitable polynomial and calculates the numbers to be given to the seven chosen secret sharers.

```
>  F:=FiniteField(97);
>  P<x>:=PolynomialRing(F);
>  a:=Random(97);
>  b:=Random(97);
>  c:=Random(97);
>  f:=19+a*x+b*x^2+c*x^3;
>  f;
90*x^3 + 29*x + 19
>  Evaluate(f,1);  // (This will be the first person's number)
41
>  Evaluate(f,2);  // (This will be the second person's number)
21
>  Evaluate(f,3);  // (This will be the third person's number)
14
>  Evaluate(f,4);  // (This will be the fourth person's number)
75
>  Evaluate(f,5);  // (This will be the fifth person's number)
65
>  Evaluate(f,6);  // (This will be the sixth person's number)
39
>  Evaluate(f,7);  // (This will be the seventh person's number)
52
>  delete f;  // (Erasing the secret from the computer's memory.)
```

The command `F:=FiniteField(97)` in effect defines `F` to be the set of residues modulo 97, and ensures that MAGMA uses residue arithmetic for calculations with elements of `F`. The command `P<x>:=PolynomialRing(F)` defines `P` to be the set of polynomials over `F` in the variable `x`. Now any calculations that MAGMA does with polynomials in `x` will always involve reduction modulo 97. That is why, for example, evaluating `f` at 2 gives 21 rather than $90 \times 8 + 29 \times 2 + 19 = 797$. The command `a:=Random(97)` tells MAGMA to randomly choose a number from the set $\{0, 1, 2, \ldots, 96\}$ and call it `a`. Similarly, `b` and `c` are chosen randomly. It can be seen from the value of `f` that in this instance MAGMA chose 29 for `a`, 0 for `b` and 90 for `c`; of course, running the same commands again would produce different values.

Now suppose that the time comes when Persons 2, 4, 5 and 7 need to get together and discover the secret number. They know that there is a polynomial $f(x)$ of degree 3 such that $f(2) \equiv 21 \pmod{97}$, $f(4) \equiv 75 \pmod{97}$, $f(5) \equiv 65 \pmod{97}$ and $f(7) \equiv 52 \pmod{97}$. Being well grounded in basic mathematics, they know how to do this using the Lagrange Interpolation Formula. First of all they calculate the following four polynomials:

$$\begin{aligned} g_1(x) &= (x-4)(x-5)(x-7) \\ g_2(x) &= (x-2)(x-5)(x-7) \\ g_3(x) &= (x-2)(x-4)(x-7) \\ g_4(x) &= (x-2)(x-4)(x-5) \end{aligned}$$

reducing the coefficients modulo 97. Observe that g_1 vanishes at $x = 4$, 5 and 7, while g_2 vanishes at $x = 2$, 5 and 7, and g_3 vanishes at $x = 2$, 4 and 7, and g_4 vanishes at $x = 2$, 4 and 5. Now $g_1(2) = (2-4)(2-5)(2-7) \equiv -30 \pmod{97}$, and the inverse of -30 modulo 97 is 42. (This can be found by use of the extended Euclidean Algorithm. In fact, $-30 \cdot 42 = -1260 = -13 \cdot 97 + 1$.) So the polynomial

$$h_1(x) = 42g_1(x) = 42(x-4)(x-5)(x-7)$$

has the property that $h_1(2) \equiv 1 \pmod{97}$, and $h_1(4) \equiv h_1(5) \equiv h_1(7) \equiv 0 \pmod{97}$. Similarly, the inverse of $g_2(4)$ modulo 97 is 81, the inverse of $g_3(5)$ modulo 97 is 16, and the inverse of $g_4(7)$ mod 97 is 55. So we define

$$\begin{aligned} h_2(x) &= 81(x-2)(x-5)(x-7) \\ h_3(x) &= 16(x-2)(x-4)(x-7) \\ h_4(x) &= 55(x-2)(x-4)(x-5), \end{aligned}$$

again reducing all the coefficients modulo 97, and note that the following properties must hold:

$$\begin{array}{llll} h_1(2) \equiv 1, & h_1(4) \equiv 0, & h_1(5) \equiv 0, & h_1(7) \equiv 0; \\ h_2(2) \equiv 0, & h_2(4) \equiv 1, & h_2(5) \equiv 0, & h_2(7) \equiv 0; \\ h_3(2) \equiv 0, & h_3(4) \equiv 0, & h_3(5) \equiv 1, & h_3(7) \equiv 0; \\ h_4(2) \equiv 0, & h_4(4) \equiv 0, & h_4(5) \equiv 0, & h_4(7) \equiv 1. \end{array}$$

Now Person 2's number is 21, Person 4's number is 75, Person 5's number is 65 and Person 7's number is 52; so we form the polynomial

$$h(x) = 21h_1(x) + 75h_2(x) + 65h_3(x) + 52h_4(x),$$

and observe that $h(2) \equiv 21$, $h(4) \equiv 75$, $h(5) \equiv 65$ and $h(7) \equiv 52$ (all modulo 97).

So $h(x)$ takes the same values as $f(x)$ (modulo 97) at $x = 2$, 4, 5 and 7. Hence the polynomial $h(x) - f(x)$ takes the value 0 (mod 97) at these four points. Furthermore, $h(x) - f(x)$ does not involve powers of x higher than x^3. It now follows from a theorem we proved in Week 9 that, modulo 97, $h(x) - f(x)$ must be the zero polynomial. This will be proved fully below, but first let us do the actual calculations (using MAGMA again).

```
> F:=FiniteField(97);
> P<x>:=PolynomialRing(F);
> g1:=(x-4)*(x-5)*(x-7);
> g2:=(x-2)*(x-5)*(x-7);
> g3:=(x-2)*(x-4)*(x-7);
> g4:=(x-2)*(x-4)*(x-5);
> h1:=Evaluate(g1,2)^(-1)*g1;
> h2:=Evaluate(g2,4)^(-1)*g2;
> h3:=Evaluate(g3,5)^(-1)*g3;
> h4:=Evaluate(g4,7)^(-1)*g4;
> h1,h2,h3,h4;
42*x^3 + 7*x^2 + 91*x + 37
81*x^3 + 30*x^2 + 26*x + 53
16*x^3 + 83*x^2 + 24*x + 74
55*x^3 + 74*x^2 + 53*x + 31
> 21*h1+75*h2+65*h3+52*h4;
90*x^3 + 29*x + 19
```

We observe that our four people have indeed been able to calculate the polynomial $f(x)$ and hence discover the secret number 19.

As we remarked above, the formula at work here is called the "Lagrange Interpolation Formula". This result can be stated as follows.

Theorem. *Suppose that $x_1, x_2, \ldots, x_k$ are distinct numbers. Then for any numbers $y_1, y_2, \ldots, y_k$ there is a unique polynomial $f(x)$ of degree less than k such that $f(x_i) = y_i$ for all $i \in \{1, 2, \ldots, k\}$. Moreover, $f(x)$ is given by the following formula:*

$$f(x) = \sum_{i=1}^{k} y_i \prod_{\substack{j=1 \\ j\neq i}}^{k} \frac{x - x_j}{x_i - x_j}.$$

Without using sigma and pi notation, in the case $k = 4$ the formula for $f(x)$ is

$$\begin{aligned} f(x) = y_1 &\frac{(x-x_2)(x-x_3)(x-x_4)}{(x_1-x_2)(x_1-x_3)(x_1-x_4)} + y_2 \frac{(x-x_1)(x-x_3)(x-x_4)}{(x_2-x_1)(x_2-x_3)(x_2-x_4)} \\ &+ y_3 \frac{(x-x_1)(x-x_2)(x-x_4)}{(x_3-x_1)(x_3-x_2)(x_3-x_4)} + y_4 \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_4-x_1)(x_4-x_2)(x_4-x_3)}. \end{aligned}$$

Unfortunately, the above statement does not exactly apply to our situation. Basically, we need to replace ordinary arithmetic with residue arithmetic modulo a prime. Recall that if p is a prime number and $\mathbb{Z}_p = \{0, 1, 2, \ldots, p-1\}$ the set of residues mod p, then $\mathbb{Z}_p$ is a field with respect to residue addition and residue multiplication. In other words, the following rules of ordinary arithmetic are satisfied in residue arithmetic mod p:

F1) $a + b = b + a$ for all a, b;
F2) $(a + b) + c = a + (b + c)$ for all a, b, c;
F3) $a + 0 = a$ for all a;
F4) For every a there is a b such that $a + b = 0$;
F5) $ab = ba$ for all a, b;
F6) $(ab)c = a(bc)$ for all a, b, c;
F7) $a1 = a$ for all a;
F8) For every $a \neq 0$ there is a b such that $ab = 1$;
F9) $(a + b)c = ac + bc$ for all a, b, c.

If p were not prime then Axiom F8 would fail, although the other axioms would still hold. Axiom F8 is crucial for our present application since we need t^{-1} (or $1/t$) to exist whenever t is a nonzero element of $\mathbb{Z}_p$.

Given that $\mathbb{Z}_p$ is a field with respect to residue arithmetic, our next step is to consider polynomials over $\mathbb{Z}_p$. If a_0, a_1, …, $a_d \in \mathbb{Z}_p$ then the polynomial $f(x) = a_d x^d + a_{d-1}x^{d-1} + \cdots + a_1 x + a_0$ determines a function from $\mathbb{Z}_p$ to $\mathbb{Z}_p$. Given any $t \in \mathbb{Z}_p$ we define $f(t)$ to be $a_d t^d + a_{d-1}t^{d-1} + \cdots + a_1 t + a_0$ as calculated using residue arithmetic. We can also add and multiply polynomials using residue arithmetic. For example, if $p = 17$ and $f(x) = 2x^2 + 8x + 11$ and $g(x) = 5x + 13$ then

$$\begin{aligned} f(x)g(x) &= (2x^2 + 8x + 11)(5x + 13) \\ &= 10x^3 + 26x^2 + 40x^2 + 104x + 55x + 143 \\ &= 10x^3 + 15x^2 + 6x + 7. \end{aligned}$$

In calculations such as this, reduction modulo p can be applied whenever it is convenient; the final answer will not be affected. Note also that if $h(x) = f(x)g(x)$ then $h(t) = f(t)g(t)$ for all $t \in F$, where of course all sums and products are calculated using residue arithmetic. Thus, for example, if $f(x)$ and $g(x)$ are as above, so that $h(x) = f(x)g(x) = 10x^3 + 15x^2 + 6x + 14$, then $f(3) = 18 + 24 + 11 = 2$ and $g(3) = 28 = 11$, giving $f(3)g(3) = 22 = 5$. And, sure enough, we find that $h(3) = 270 + 135 + 18 + 7 = 15 + 16 + 1 + 7 = 5$. Of course, the same holds for addition of polynomials: if $h(x) = f(x) + g(x)$ then $h(t) = f(t) + g(t)$ for all $t \in F$.

The point of this digression is that the Lagrange Interpolation Formula is valid for polynomials over the field of residues modulo p. (In fact, it is valid over any field.) To prove this, we first restate Theorem (8.7) in the language of residue arithmetic.

(10.2) Theorem: *Let p be a prime and $f(x) = a_d x^d + a_{d-1}x^{d-1} + \cdots + a_1 x + a_0$, where the a_i are residues modulo p. For each residue t let $f(t) = a_d t^d + a_{d-1}t^{d-1} + \cdots + a_1 t + a_0$ as calculated using residue arithmetic. Then, if $a_d \neq 0$, the number of residues t such that $f(t) = 0$ cannot exceed d.*

Notice that if $a_d = 0$ but $a_{d-1} \neq 0$ then the number of t such that $f(t) = 0$ cannot exceed $d - 1$. Since $d - 1 < d$, the statement that the number of roots cannot exceed d is still valid. And if $a_d = a_{d-1} = \cdots = a_{e+1} = 0$ and $a_e \neq 0$ then the number of roots cannot exceed e, which certainly means that it cannot exceed d. Thus the only way that $f(x) = a_dx^d + a_{d-1}x^{d-1} + \cdots + a_1x + a_0$ can have more than d roots is if the coefficients are all zero (making $f(x)$ the zero polynomial).

We can now prove the validity of the Lagrange Interpolation Formula in the context of residues modulo p. Suppose that $x_1, x_2, \ldots, x_k$ are distinct residues and $y_1, y_2, \ldots, y_k$ are arbitrary residues. For each $i \in \{1, 2, \ldots, k\}$ define

$$e_i(x) = \prod_{\substack{j=1 \\ j\neq i}}^{k} \frac{x - x_j}{x_i - x_j} = \frac{(x - x_1)(x - x_2)\cdots(x - x_{i-1})\ (x - x_{i+1})\cdots(x - x_k)}{(x_i - x_1)(x_i - x_2)\cdots(x_i - x_{i-1})\ (x_i - x_{i+1})\cdots(x_i - x_k)},$$

noting that all of the inverses $\frac{1}{x_i - x_j}$ exist—as is necessary for this formula to make sense—since $x_i - x_j \neq 0$ in each case. Then it is obvious that $e_i(x_j) = 0$ whenever $j \neq i$, since $x - x_j$ is a factor in the numerator of the expression for $e_i(x)$. On the other hand, $e_i(x_i) = 1$, since replacing x by x_i makes the numerator equal to the denominator. So

$$e_i(x_j) = \begin{cases} 1 & \text{if } j = i, \\ 0 & \text{if } j \neq i. \end{cases}$$

Hence the polynomial $f(x)$ defined by

$$f(x) = \sum_{i=1}^{k} y_i \prod_{\substack{j=1 \\ j\neq i}}^{k} \frac{x - x_j}{x_i - x_j} = y_1e_1(x) + y_2e_2(x) + \cdots + y_ke_k(x)$$

has the property that for each i,

$$f(x_i) = y_1e_1(x_i) + y_2e_2(x_i) + \cdots + y_ie_i(x_i) + \cdots + y_ke_k(x) = y_i$$

since $y_ie_i(x_i) = y_i$ and the other terms are all 0. Furthermore, since each of the polynomials $e_i(x)$ has degree $k - 1$ (since there are $k - 1$ factors $x - x_j$ in the product that defines $e_i(x)$), it follows that the degree of $f(x)$ is less than k. So all that remains is to prove that $f(x)$ is the only polynomial of degree less than k such that $f(x_i) = y_i$ for each i. But this follows from the theorem above: if $h(x)$ also has degree less than k and satisfies $h(x_i) = y_i$ for each i then the polynomial $f(x) - h(x)$ has degree less than k and satisfies $f(x_i) - h(x_i) = 0$ for all $i \in \{1, 2, \ldots, k\}$. But a nonzero polynomial of degree less than k cannot have k roots. So $f(x) - h(x)$ is the zero polynomial. That is, $h(x) = f(x)$, as required.

Finally, let us return to the secret sharing situation. There is a certain polynomial $f(x)$ defined over F, the field of residues modulo the prime p. The secret number is $f(0)$, the constant term of $f(x)$, and the degree of $f(x)$ is less than or equal to k.

There are n people in the scheme, and the m-th person knows the value $y_m = f(m)$. Now if $\{m_1, m_2, \ldots, m_k\}$ is any k element subset of $\{1, 2, \ldots, n\}$ then, by the Lagrange Interpolation Formula,

$$f(x) = \sum_{i=1}^{k} y_{m_i} \prod_{\substack{j=1 \\ j \neq i}}^{k} \frac{x - m_j}{m_i - m_j}.$$

Hence any k of the people can cooperate to determine $f(x)$, and hence discover $f(0)$.

One final observation should be made. Our k people do not have to compute $f(x)$, they only need to compute $f(0)$, and for this they just have to put $x = 0$ in the above formula. Specifically, the secret number is given by the formula

$$\sum_{i=1}^{k} y_{m_i} \prod_{\substack{j=1 \\ j \neq i}}^{k} \frac{-m_j}{m_i - m_j} = \sum_{i=1}^{k} y_{m_i} \prod_{\substack{j=1 \\ j \neq i}}^{k} \frac{m_j}{m_j - m_i}.$$

(10.3) ***Example:*** As above, suppose that $p = 97$ and the secret number is the constant term of a polynomial of degree at most 3. There are seven people in the scheme, their numbers being (respectively) 41, 21, 14, 75, 65, 39 and 52. If Persons 1, 2, 3 and 4 want to recover the secret, the calculation they perform is

$$41\times\frac{2\times3\times4}{1\times2\times3}+21\times\frac{1\times3\times4}{(-1)\times1\times2}+14\times\frac{1\times2\times4}{(-2)\times(-1)\times1}+75\times\frac{1\times2\times3}{(-3)\times(-2)\times(-1)},$$

giving the answer $67 + 68 + 56 + 22 = 19$ in mod 97 arithmetic. If Persons 2, 3, 4 and 6 needed to recover the secret, they would compute

$$21\times\frac{3\times4\times6}{1\times2\times4}+14\times\frac{2\times4\times6}{(-1)\times1\times3}+75\times\frac{2\times3\times6}{(-2)\times(-1)\times2}+39\times\frac{2\times3\times4}{(-4)\times(-3)\times(-2)},$$

correctly obtaining the answer $92 + 67 + 93 + 58 = 19$.

Week 11

§47 Computing discrete logarithms

Suppose that p is a prime and that b is a fixed nonzero residue modulo p. We are principally interested in the case that b is a primitive root modulo p, or at least has very large order modulo p. (Recall that a primitive root mod p is a residue that has order $p-1$, which is the largest possible order mod p.) Let a be another nonzero residue mod p, and suppose that it is known that $a \equiv b^i \pmod{p}$ for some positive integer i. We investigate strategies for finding i, given a, p and b.

Recall that if $a \equiv b^i \pmod{p}$ then i is called the discrete logarithm to the base b of the residue a mod p. We write $i = \log_{b,p}(a)$. Note that if $N = \operatorname{ord}_p(b)$ then $b^i \equiv b^j \pmod{p}$ if and only if $i \equiv j \pmod{N}$. So discrete logarithms to the base b are not uniquely determined: they are determined only up to congruence modulo N, the order of b. So the i we are trying to compute can be assumed to be a residue mod N.

Since there are only N possible values for i altogether, one algorithm for finding i is to compute the residues (mod p) of b, b^2, b^3, and so on, until we find the one that equals a. In the cases we are interested in, N is so extremely large that an exhaustive search of this kind is not feasible. We need to find a better method.

There is a method that works very well if all of the prime factors of N are small.* Let $N = \prod_{l=1}^{s} q_l^{r_l}$, where the q_l are small primes. (What "small" means here depends on how powerful your computer is.) Since b has order N mod p, it follows that b^{N/q_1} has order q_1 mod p, and since q_1 is small we can compute and store all the powers of b^{N/q_1}, reduced mod p. That is, we compute the residues of b^{jN/q_1}, for all $j \in \{0, 1, 2, \ldots, q_1 - 1\}$. With the aid of this list we shall be able to compute the residue of i modulo $q_1^{r_1}$. After doing this, we repeat the process with q_1 replaced by q_2, then q_3, and so on. Once we have found the residues of i modulo $q_l^{r_l}$ for all values of $l \in \{1, 2, \ldots, s\}$, we shall be able to use the Chinese Remainder Theorem to find i modulo $q_1^{r_1} q_2^{r_2} \cdots q_s^{r_s} = N$, as required.

Rather than attempting to explain the procedure abstractly, we illustrate it with an example. Let $p = 54001$ and $b = 11$. Then the order of b mod p is $54000 = 2^4 \cdot 3^3 \cdot 5^3$. We set ourselves the task of finding $i \in \{0, 1, \ldots, 53999\}$ such that $11^i \equiv 25428 \pmod{54001}$. That is, we wish to compute $\log_{11,54001}(25428)$.

We consider the three prime divisors of 54000 separately. Let us focus first on the prime $q = 5$. Since $\operatorname{ord}_p(11) = 54000$ we know that $11^j \not\equiv 1 \pmod{p}$

* This method is known as the *Pohlig–Hellman algorithm*.

for $1 \leq j < 54000$, and hence $(11^{10800})^j \not\equiv 1 \pmod{p}$ for $1 \leq j < 5$. Thus $\mathrm{ord}_p(11^{10800}) = 5$, and 11^{10800} has exactly five distinct powers modulo p. Because five is small, and computing powers in residue arithmetic is computationally easy, there is no difficulty listing and storing all the powers of 11^{10800} modulo p. It takes MAGMA a fraction of a second.

```
> t:=Cputime();
> for j:=0 to 4 do
for> Modexp(11,j*10800,54001);
for> end for;
1
18177
25211
7861
2751
> Cputime(t);
0.01
```

Thus we have the following table of values.

j	0	1	2	3	4
$(11^{10800})^j$	1	18177	25211	7861	2751

Since we are trying to solve $11^i \equiv 25428 \pmod{p}$, we must have that

$$(11^{10800})^i = (11^i)^{10800} \equiv 25428^{10800} \pmod{p}.$$

Again, MAGMA can do the necessary computation very quickly.

```
> Modexp(25428,10800,54001);
7861
```

Thus $(11^{10800})^i \equiv 7861 \equiv (11^{10800})^3 \pmod{p}$, and therefore $i \equiv 3 \pmod{5}$.

We have taken the first step along the road to determining i modulo 54000: we have determined i modulo 5. The next step is to determine i modulo 5^2.

Since $i \equiv 3 \pmod{5}$ we have $i - 3 = 5j$ for some $j \in \mathbb{Z}$. So $11^i \equiv 25428 \pmod{p}$ gives

$$11^{5j} \equiv 11^{i-3} \equiv 11^i \times (11^{-1})^3 \equiv 25428 \times (11^{-1})^3 \pmod{p},$$

and raising both sides of this to the power 2160 gives

$$(11^{10800})^j \equiv (11^{5j})^{2160} \equiv 25428^{2160} \times (11^{-1})^{6480} \pmod{p}.$$

This last quantity turns out to be congruent to 25211 (mod p).

```
> InverseMod(11,54001);
> 24546
> Modexp(25428,2160,54001)*Modexp(24546,6480,54001) mod 54001;
25211
```

Thus $(11^{10800})^j \equiv 25211 \equiv (11^{10800})^2 \pmod{p}$, and therefore $j \equiv 2 \pmod{5}$. It follows that $j = 5l + 2$ for some integer l, and

$$i - 3 = 5j = 5(5l + 2) = 25l + 10.$$

Since this shows that $i \equiv 13 \pmod{25}$, we have completed the second step.

The third step on our road is to determine i modulo 5^3. Since 5^3 is the highest power of 5 dividing 54000, once we have determined i modulo 5^3 we shall turn our attention to the other prime divisors of 54000.

Since $i - 13 = 25l$ we see that

$$\begin{aligned}(11^{10800})^l &\equiv (11^{25l})^{432} \equiv (11^{i-13})^{432} \\ &\equiv (11^i)^{432} \times ((11^{-1})^{13})^{432} \equiv 25428^{432} \times ((11^{-1})^{13})^{432}.\end{aligned}$$

This last quantity turns out to be congruent to 25211 (mod p). Recall that the inverse of 11 modulo 54001 is 24546 (as MAGMA found above).

```
> Modexp(25428,432,54001)*Modexp(24546,13*432,54001) mod 54001;
25211
```

Thus $(11^{10800})^l \equiv 25211 \equiv (11^{10800})^2 \pmod{p}$, and therefore $l \equiv 2 \pmod{5}$. Writing $l = 5l' + 2$, we find that

$$i - 13 = 25(5l' + 2) = 125l' + 50,$$

and therefore $i \equiv 63 \pmod{5^3}$.

We now turn our attention to the prime 3, and set about finding the residue of i modulo 3^3. The process is exactly the same as the one we used above for the prime 5: we first find the residue of i modulo 3, then use this to find the residue of i modulo 3^2, and finally use this to find the residue modulo 3^3.

Since 11 had order 54000 mod p and $54000/3 = 18000$, it follows that 11^{18000} has order 3. We compute the three distinct powers of 11^{18000} modulo p, and remember them.

```
> for j:=0 to 2 do
for> Modexp(11,18000*j,54001);
for> end for;
1
39940
14060
```

The three distinct powers of 11^{18000} modulo p are given by the following table.

j	0	1	2
$(11^{18000})^j$	1	39940	14060

Hence $11^i \equiv 25428 \pmod{p}$ gives

$$(11^{18000})^i \equiv (11^i)^{18000} \equiv 25428^{18000},$$

and MAGMA tells us that this is congruent mod p to 14060 (the same as $(11^{18000})^2$).

```
> Modexp(25428,18000,54001);
14060
```

Since $(11^{18000})^i \equiv (11^{18000})^2 \pmod{p}$ it follows that $i \equiv 2 \pmod{3}$.

We can now say that $i - 2 = 3m$ for some $m \in \mathbb{Z}$, and use $11^i \equiv 25428 \pmod{p}$ to deduce that

$$11^{3m} \equiv 11^{i-2} \equiv 11^i \times (11^{-1})^2 \equiv 25428 \times (11^{-1})^2 \mod p.$$

Raising both sides to the power 6000 gives

$$(11^{18000})^m \equiv (11^{3m})^{6000} \equiv 25428^{6000} \times (11^{-1})^{12000} \equiv 25428^{6000} \times 24546^{12000}$$

(since 24546 is the inverse of 11 mod p, as we found above). We turn to MAGMA once more.

```
> Modexp(25428,6000,54001)*Modexp(24546,12000,54001) mod 54001;
39940
```

So $(11^{18000})^m \equiv 39940 \equiv (11^{18000})^1 \pmod{p}$, and $m \equiv 1 \pmod{3}$. This means that we can write $m = 3n + 1$ for some integer n, and deduce that $i - 2 = 3(3n + 1)$. So $i - 5 = 9n$, and we have determined the residue of i modulo 9.

Using the same procedure again, $11^i \equiv 25428$ gives

$$11^{9n} \equiv 11^{i-5} \equiv 11^i \times (11^{-1})^5 \equiv 25428 \times (11^{-1})^5,$$

which gives

$$(11^{18000})^n \equiv (11^{9n})^{2000} \equiv 25428^{2000} \times (11^{-1})^{10000},$$

and we use MAGMA to calculate this.

```
> Modexp(25428,2000,54001)*Modexp(24546,10000,54001) mod 54001;
14060
```

So $(11^{18000})^n \equiv 14060 \equiv (11^{18000})^2 \pmod{p}$, and $n \equiv 2 \pmod{3}$. So $n = 3n' + 2$ for some n', whence $i - 5 = 9(3n' + 2)$, and $i \equiv 23 \pmod{27}$.

Finally we consider the prime 2. The procedure is exactly the same as for 5 and 3. We observe to begin with that 11^{27000} has order 2 modulo p, and the two powers of 11^{27000} modulo p are $(11^{27000})^0 = 1$ and $(11^{27000})^1 = 54000$. The MAGMA calculations that support the steps of the process are given below.

From $11^i \equiv 25428 \pmod{p}$ we obtain

$$(11^{27000})^i \equiv (11^i)^{27000} \equiv 25428^{27000} \equiv 54000 \equiv (11^{27000})^1,$$

and it follows that $i \equiv 1 \pmod 2$. Writing $i-1 = 2r$, we find that

$$11^{2r} \equiv 11^{i-1} \equiv 11^{i} \cdot 11^{-1} \equiv 25428 \times 11^{-1}$$

and thus

$$(11^{27000})^{r} \equiv (11^{2r})^{13500} \equiv 25428^{13500} \times (11^{-1})^{13500} \equiv 54000 \equiv 11^{27000}.$$

We deduce that $r = 2s+1$ for some $s \in \mathbb{Z}$, and so $i-1 = 2(2s+1)$, or $i-3 = 4s$.
Next,

$$11^{4s} \equiv 11^{i-3} \equiv 11^{i} \times 11^{-3} \equiv 25428 \times (11^{-1})^{3},$$

and thus

$$(11^{27000})^{s} \equiv (11^{4s})^{6750} \equiv 25428^{6750} \times ((11^{-1})^{3})^{6750} \equiv 54000 \equiv 11^{27000}.$$

So $s = 2t+1$ for some $t \in \mathbb{Z}$, and therefore $i-3 = 4(2t+1)$, or $i-7 = 8t$.
Finally,

$$11^{8t} \equiv 11^{i-7} \equiv 11^{i} \times 11^{-7} \equiv 25428 \times (11^{-1})^{7},$$

giving

$$(11^{27000})^{t} \equiv (11^{8t})^{3375} \equiv 25428^{3375} \times ((11^{-1})^{7})^{3375} \equiv 54000 \equiv 11^{27000}.$$

So $t \equiv 1 \bmod 2$, whence $t = 2u+1$ for some integer u, giving $i-7 = 8(2u+1)$, and $i \equiv 15 \pmod{16}$.

Here is the MAGMA code for the calculations used above.

```
> Modexp(11,27000,54001);
54000
> Modexp(25428,27000,54001);
54000
> Modexp(25428,13500,54001)*Modexp(24546,13500,54001) mod 54001;
54000
> Modexp(25428,6750,54001)*Modexp(24546,3*6750,54001) mod 54001;
54000
> Modexp(25428,3375,54001)*Modexp(24546,7*3375,54001) mod 54001;
54000
```

We have now shown that k satisfies the following three congruences:

$$\begin{aligned} i &\equiv 63 \quad \pmod{125}, \\ i &\equiv 23 \quad \pmod{27}, \\ i &\equiv 15 \quad \pmod{16}. \end{aligned}$$

We should now solve these. The solution will give us the residue class of k modulo 54000.

We have $i = 125h + 63$ for some h, and substituting this into the second congruence gives $125h + 63 \equiv 23 \pmod{27}$. This simplifies to $17h \equiv 14 \pmod{27}$, giving $h \equiv 4 \pmod{27}$. Now putting $h = 27v + 4$ gives $i = 125(27v+4)+63 = 3375v+563$, and substituting into the third congruence gives $3375v + 563 \equiv 15 \pmod{16}$, or $15v \equiv 12 \pmod{16}$. Thus $v \equiv 4 \pmod{16}$, and $i = 3375(16w + 4) + 563$, for some w. Hence $k \equiv 14063 \pmod{54000}$.

The final answer, therefore, is that $\log_{11,54001}(25428) = 14063$.

```
> Modexp(11,14063,54001);
25428
```

§48 Baby Step Giant Step

What if $\mathrm{ord}_p(b)$ has one or more large prime factors? How should we go about solving $b^i \equiv a \pmod{p}$ in this case?

The security of the Elgamal cryptosystem hangs on the fact that it is computationally difficult to do this. So the algorithms that are known are not going to be practical when the numbers get too large.

Let p be a prime. We suppose that we are given a and b, and wish to find i such that $b^i \equiv a \pmod{p}$. We can take it as given that there is a solution; the task is to find it. We can also assume that $N = \mathrm{ord}_p(b)$ is known. The i that we wish to find is a residue modulo N.

The algorithm presented in the previous section should be applied whenever one is able to factorize N, but one aspect of the algorithm needs to be modified when dealing with a large prime factor (or a factor that we are unable to further factorize). If q is the factor in question and b' is the residue of $b^{N/q}$ modulo p, then b' has order q modulo p, and so b' has q distinct powers modulo p. According to the algorithm described in the previous section, the first step is to compute all of these powers of b' (reduced modulo p of course). But when q is too large it is not a good idea to try to do this: it is likely to involve too much computation and require too much storage space.

The reason for wanting a list of the powers of b' is so that when we subsequently compute something that we know has to be a power of b', we shall be able to just look through the list to find which power of b' it is. This is, in fact, just another discrete logarithm problem. We have computed some a', and we know that $(b')^{i'} \equiv a' \pmod{p}$ for some i', and we want to find i'. We want some way of tackling this problem when we cannot factorize $q = \mathrm{ord}_p(b')$, and q is to large for us to be able to list all the powers of b'.

The Baby Step Giant Step method shows that instead of making a list of all q powers of b', it is actually sufficient to make a much shorter list. Rather than computing and storing $(b')^i$ (reduced mod p) for all i from 1 to q, we should restrict ourselves to values of i from 1 up to $\sqrt{q}$.

While this is definitely an improvement, the truth is that in practice $\sqrt{q}$ is likely to be too large as well. If it is, then there may be nothing we can do about it. Computing discrete logarithms is, when the numbers are large, just too hard.

Again we shall not attempt to describe the algorithm abstractly, but will simply illustrate it with an example. We use small numbers this time: $p = 179$, $b = 2$ and $a = 46$. We shall solve $2^i \equiv 46 \pmod{179}$. Because $\frac{1}{2}(p-1) = 89$ is prime and $p \equiv 3 \pmod 4$, a theorem from last week tells us that 2 is a primitive root mod p. So $N = \operatorname{ord}_p(2) = 178$. We factorize this as far as we can. Of course 2 is a factor, since p is odd. But $\frac{1}{2}(p-1)$ is prime: so the factorization $N = 2 \times 89$ is the best we can do.

Applying the algorithm of the previous section, we should try to find the residue of i mod 2 and the residue of i mod 89, then use the Chinese Remainder Theorem to combine these, and thereby obtain the residue of i mod 178.

The first part of this is easy. From $2^i \equiv 46 \pmod{179}$ we deduce that

$$(2^{89})^i \equiv (2^i)^{89} \equiv 46^{89} \equiv 1 \pmod{179},$$

and it follows that i is even.

Turning to the second part, we note that $4 = 2^2$ has order 89 mod 179, and that $2^i \equiv 46 \pmod{179}$ gives

$$4^i \equiv 2^{2i} \equiv 46^2 \equiv 147 \pmod{179}.$$

Our task is to use this to find the residue of i mod 89 (knowing that $\operatorname{ord}_{179}(4)$ is 89). We pretend that listing all 89 powers of 4 modulo 179 would be too expensive in terms of computing resources, but that listing 9 or 10 powers of 4 would be OK.

The first step is to find the smallest integer M such that $M^2 \geq \operatorname{ord}_{179}(4) = 89$. So $M = 10$. Since the number that we are trying to find is less than M^2, it can be written in the form $xM + y$, with $x, y \in \{0, 1, \dots, M-1\}$. We can therefore reexpress the problem as follows: find integers $x, y \in \{0, 1, \dots, M-1\}$ such that $4^{xM+y} \equiv 147 \pmod{179}$. Rearranging this, our aim is to find x and y satisfying $4^y \equiv 147 \times (4^{-M})^x \pmod{179}$.

The strategy is to first form a list of all M possible values for the residue of $4^y \pmod{179}$, and then successively compute the residues of $147 \times (4^{-M})^x \pmod{179}$ for $x = 0, 1, 2, \dots,$ until we find one that is on the list.

The first term that goes on the list is $4^0 = 1$, and the subsequent terms 4^1, 4^2, and so on, can then be computed recursively, since each term is 4 times the previous one. The people who invented the name "Baby Step Giant Step" apparently considered the step from 4^y to 4^{y+1} a baby step, since the exponent changes by 1 only. In the next part of the algorithm one multiplies by 4^{-M} each time, and this is presumably regarded as a giant step, since the exponent changes by M.

Starting with 1 and multiplying by 4 each time (and reducing modulo 179), gives 1, 4, 16, 64, 77, 129, 158, 95 and 88. We build up the following table of values.

4^y	1	4	16	22	64	77	88	95	129	158
y	0	1	2	8	3	4	9	7	5	6

Notice that we have sorted the numbers in the top row of this table into increasing order. It is algorithmically straightforward to to this as the list is generated, since if the previously computed terms have already been arranged in increasing order, then it will be easy to insert each new term into its correct place. It is useful to have the list sorted into increasing order since this makes it easier to subsequently determine whether a given number appears in the list.

We now do the giant steps: we compute the residues mod 179 of $147 \times (4^{-10})^x$ for $x = 0$, 1, 2, and so on, checking each time to see if the number appears in the top row of the table above. The inverse of 4 modulo 179 is 45, and the 10th power of this turns out to be 149 (modulo 179). So each giant step consists of multiplying by 149 and reducing modulo 179. The sequence we obtain is 147 (for $x = 0$), 65 (for $x = 1$), 19 (for $x = 2$), 146 (for $x = 3$), 95 (for $x = 4$). Here we find that the number appears in the top row of our table: $147 \times (4^{-10})^4 \equiv 4^7 \pmod{179}$. Hence $147 \equiv 4^{47} \bmod 89$. So $\log_{4,179}(47) = 47$.

Returning to the original problem, which was to find $i \in \{0, 1, \ldots, 178\}$ such that $2^i \equiv 46 \pmod{179}$, recall that we had shown that $i \equiv 0 \pmod 2$, and then needed to compute the residue of i modulo 89. Since $2^i \equiv 46$ gives $4^i \equiv 46^2 \equiv 147$, the Baby Step Giant Step calculations above now tell us that $i \equiv 47 \pmod{89}$. So $i = 47 + 89m$ for some m, and now the requirement that $i \equiv 0 \pmod 2$ yields that $m \equiv 1 \pmod 2$. Thus $i = 47 + 89(2n + 1)$ for some n, giving $i \equiv 136 \pmod{178}$. So we have found that $\log_{2,179}(46) = 136$.

§49 Pollard Rho for discrete logarithms

The Baby Step Giant Step method is bound to work, but has the disadvantage that it involves storing a long list of values. The Pollard Rho attack on the discrete logarithm problem is a probabilistic method for which the expected number of steps to completion—when it works—is similar to the expected number of steps that the Baby Step Giant Step method will require. It has the advantage that it does not involve any storing of values or seaching through lists, and consequently each step is fast. It has the disadvantage that it is not quite guaranteed to work.

Suppose that a and b are nonzero residues modulo the prime p, with $b \neq 1$, and we wish to find i such that $b^i \equiv a \pmod p$. We assume that $b^q \equiv 1 \pmod p$ and that we are unable to factorize q (quite possibly because it is prime). We assume also that $a^q \equiv 1 \pmod p$ (which is clearly necessary if $b \equiv a^i \pmod p$ is to have a solution). The idea is to generate a sequence of triples $\langle x_j, n_j, m_j \rangle$ with the following properties (for all values of j):

1) x_j is a residue mod p and n_j, m_j are residues mod q, and $x_j \equiv b^{n_j} a^{m_j} \pmod p$;
2) x_{j+1} is uniquely determined by x_j.

If we can then find j and k such that $x_j = x_k$ but $\langle n_j, m_j \rangle \neq \langle n_k, m_k \rangle$ then we shall be able to deduce that

$$b^{n_j} a^{m_j} \equiv b^{n_k} a^{m_k} \pmod p$$

and therefore

$$a^{(m_j-m_k)r} \equiv b^{(n_k-n_j)r} \pmod p$$

for all r. Provided that $m_j - m_k$ is coprime to q we shall be able to choose r so that $(m_j - m_k)r \equiv 1 \pmod q$, and then we shall have $a \equiv b^{(n_k-n_j)r} \pmod p$. Thus $i = (n_k - n_j)r$ will be the solution we seek.

Note that if $m_k - m_j$ is not coprime to q then by computing $\gcd(m_k - m_j, q)$ (or $\gcd(n_j - n_k, q)$ if $m_j = m_k$) we shall be able to find a nontrivial factorization $q = q_1q_2$. Then, using the method of §47 above, we shall be able to replace the original problem by two simpler problems of the same kind.

Since the x_j's are residues mod p, there are only finitely many values they can take, and in at most p steps there will be a repetition. In other words, it is guaranteed that there exist j, k with $j \neq k$ and $x_j = x_k$. The problem is that it may happen that $\langle n_j, m_j\rangle = \langle n_k, m_k\rangle$ as well, in which case the method fails. But it is perhaps reasonable to expect the method to succeed more often than it fails. If $x_j = x_k$, what is the probability that $\langle n_j, m_j\rangle = \langle n_k, m_k\rangle$? We must have that $b^{m_j-m_k} \equiv a^{n_k-n_j}$, but $m_j - m_k = 0 = n_k - n_j$ is only one of q solutions to this. The other solutions are given by $\langle n_k - n_j, m_j - m_k\rangle = \langle n, in\rangle$ for $n \in \{1, 2, \ldots, q-1\}$. Now if—admittedly, this is a big "if"—all solutions are equally likely, then the probability of the method succeeding is $(q-1)/q$. Given that q is enormous, this only differs from 1 by a minuscule amount.

How long do we expect it to take before it happens that some x_k equals some earlier x_j? The well-known "birthday paradox" says that in a group of 23 people the probability that there are two people with the same birthday is better than one half. With 30 people the probability is about 0.7. In general, if x_1, x_2, …, x_j are residues modulo p, and $j > 1.18\sqrt{p}$, then the probability that two of the x_j's are equal is more than 0.5.* So we expect to get a repetition in about $1.18\sqrt{p}$ steps, on average. (Note that in the Baby Step Giant Step method, the expected running time is also $O(\sqrt{p})$.)

But given that we do not wish to store anything, how can we possibly detect whether or not a repetition has occurred? This is the clever part. Since x_j determines x_{j+1} uniquely, once it happens that $x_j = x_k$ for some j and k then it will follow that $x_{j+s} = x_{k+s}$ for all positive integers s. By a proposition we proved in Week 8 while discusssing the Pollard Rho method for factorization, it follows that there is an integer l in the range $j \le l < k$ such that $x_l = x_{2l}$. Now to avoid storing anything at all we can simultaneously compute the two sequences

$$\langle x_1, n_1, m_1\rangle,\ \langle x_2, n_2, m_2\rangle,\ \langle x_3, n_3, m_3\rangle,\ \langle x_4, n_4, m_4\rangle,\ \ldots$$

and

$$\langle x_2, n_2, m_2\rangle,\ \langle x_4, n_4, m_4\rangle,\ \langle x_6, n_6, m_6\rangle,\ \langle x_8, n_8, m_8\rangle,\ \ldots,$$

keeping only the two current terms $\langle x_l, n_l, m_l\rangle$ and $\langle x_{2l}, n_{2l}, m_{2l}\rangle$ in memory, and stop only when $x_l = x_{2l}$.

* For the birthday situation, note that $1.18 \times \sqrt{365} \approx 22.5$.

We have yet to say how to generate the sequence of triples $\langle x_j, n_j, m_j\rangle$. Define $\langle x_1, n_1, m_1\rangle = \langle 1, 0, 0\rangle$, noting that these values satisfy the requirement that $x_1 \equiv b^{n_1}a^{m_1} \pmod{p}$. Proceeding recursively, suppose now that $j > 1$. If $1 \le x_{j-1} < p/3$ we define $\langle x_j, n_j, m_j\rangle$ by requiring that

$$\begin{aligned} x_j &\equiv bx_{j-1} \pmod{p},\\ n_j &\equiv n_{j-1}+1 \pmod{q},\\ m_j &= m_{j-1}. \end{aligned}$$

If $p/3 \le x_{j-1} < 2p/3$ we define $\langle x_j, n_j, m_j\rangle$ by requiring that

$$\begin{aligned} x_j &\equiv x_{j-1}^2 \pmod{p},\\ n_j &\equiv 2n_{j-1} \pmod{q},\\ m_j &\equiv 2m_{j-1} \pmod{q}. \end{aligned}$$

If $2p/3 \le x_{j-1} \le p-1$ we define $\langle x_j, n_j, m_j\rangle$ by requiring that

$$\begin{aligned} x_j &\equiv ax_{j-1} \pmod{p},\\ n_j &= n_{j-1},\\ m_j &\equiv m_{j-1}+1 \pmod{q}. \end{aligned}$$

In the first case we then have that

$$x_j \equiv bx_{j-1} \equiv bb^{n_{j-1}}a^{m_{j-1}} \equiv b^{n_{j-1}+1}a^{m_{j-1}} \equiv b^{n_j}a^{m_j},$$

as we need. Similarly, in the second case we find that

$$x_j \equiv x_{j-1}^2 \equiv (b^{n_{j-1}}a^{m_{j-1}})^2 \equiv b^{2n_{j-1}}a^{2m_{j-1}} \equiv b^{n_j}a^{m_j},$$

again as needed. And it also works in the final case:

$$x_j \equiv ax_{j-1} \equiv b^{n_{j-1}}aa^{m_{j-1}} \equiv b^{n_{j-1}}a^{m_{j-1}+1} \equiv b^{n_j}a^{m_j}.$$

To illustrate the method, we apply it in the case $p = 179$, $b = 4$ and $a = 147$. Recall that $q = \operatorname{ord}_{179}(4) = 89$. We find that the sequence of triples $\langle x_j, n_j, m_j\rangle$ goes as follows: $\langle 4,1,0\rangle$, $\langle 16,2,0\rangle$, $\langle 64,3,0\rangle$, $\langle 158,6,0\rangle$, $\langle 155,6,2\rangle$, $\langle 52,6,3\rangle$, $\langle 29,7,3\rangle$, $\langle 116,8,3\rangle$, $\langle 31,16,6\rangle$, $\langle 124,17,6\rangle$, $\langle 149,17,7\rangle$, $\langle 65,17,8\rangle$, $\langle 108,34,16\rangle$, $\langle 29,68,32\rangle$. Doing hand calculations, recording the terms of the sequence, we notice at this point that we have a repetition. We conclude that $4^7 \times 147^3 \equiv 4^{68} \times 147^{32} \pmod{179}$, and this gives

$$147^{29} \equiv 4^{-61} \equiv 4^{28} \pmod{179}.$$

The inverse of 29 modulo 89 is 43, and so

$$147 \equiv (147^{29})^{43} \equiv (4^{28})^{43} \equiv 4^{47} \pmod{179},$$

since $28 \times 43 \equiv 47 \pmod{89}$.

Doing the calculation by machine, testing only if $x_l = x_{2l}$, we would have to generate a few more terms before we would detect the repetition.

Here is a MAGMA function to perform the Pollard Rho algorithm as described above.

```
prho:=function(a,b,p,q)
  p1:= p div 3; p2:=2*p div 3;
  x:=<1,0,0>; y:=<a,1,0>;
  while x[1] ne y[1] do
    if x[1] le  p1 then
      x:=<a*x[1] mod p, (x[2]+1) mod q, x[3]>;
    elif x[1] le p2 then
      x:=<x[1]^2 mod p, 2*x[2] mod q, 2*x[3] mod q>;
    else
      x:=<b*x[1] mod p, x[2], (x[3]+1) mod q>;
    end if;
    if y[1] le  p1 then
      y:=<a*y[1] mod p, (y[2]+1) mod q, y[3]>;
    elif y[1] le p2 then
      y:=<y[1]^2 mod p, 2*y[2] mod q, 2*y[3] mod q>;
    else
      y:=<b*y[1] mod p, y[2], (y[3]+1) mod q>;
    end if;
    if y[1] le  p1 then
      y:=<a*y[1] mod p, (y[2]+1) mod q, y[3]>;
    elif y[1] le p2 then
      y:=<y[1]^2 mod p, 2*y[2] mod q, 2*y[3] mod q>;
    else
      y:=<b*y[1] mod p, y[2], (y[3]+1) mod q>;
    end if;
  x,y;
  end while;
  if x eq y then
    return "failure";
  elif GCD(x[3]-y[3],q) ne 1 then
    return <q "has factors:",GCD(x[3]-y[3],q),GCD(x[2]-y[2],q)>;
  else
    return (y[2]-x[2])*InverseMod(x[3]-y[3],q) mod q;
  end if;
end function;
```

After each execution of the "while" loop the values of the variables x and y are printed. This is done in this example so that we can see the individual steps of the calculation; in a serious program we would not print out values in each loop, since doing so slows the program down very dramatically.

The successive values taken by the variable x are triples $\langle x_l, n_l, m_l\rangle$ in our description of the algorithm, and the successive values taken by the variable y are the even numbered terms of the same sequence (the triples $\langle x_{2l}, n_{2l}, m_{2l}\rangle$).

Here is some sample output.

```
> prho(4,147,179,89);
<4, 1, 0> <64, 3, 0>
<16, 2, 0> <135, 6, 1>
<64, 3, 0> <52, 6, 3>
<158, 6, 0> <116, 8, 3>
<135, 6, 1> <124, 17, 6>
<155, 6, 2> <65, 17, 8>
<52, 6, 3> <29, 68, 32>
<29, 7, 3> <31, 49, 64>
<116, 8, 3> <149, 50, 65>
<31, 16, 6> <108, 11, 43>
<124, 17, 6> <116, 23, 86>
<149, 17, 7> <124, 47, 83>
<65, 17, 8> <65, 47, 85>
47
> Modexp(4,47,179) eq 147;
true
>
> Modorder(15,179);
89
> prho(4,15,179,89);
<4, 1, 0> <64, 3, 0>
<16, 2, 0> <43, 6, 1>
<64, 3, 0> <74, 7, 2>
<158, 6, 0> <138, 28, 8>
<43, 6, 1> <177, 56, 18>
<172, 7, 1> <87, 56, 20>
<74, 7, 2> <25, 24, 40>
<106, 14, 4> <155, 50, 80>
<138, 28, 8> <149, 50, 82>
<101, 28, 9> <51, 11, 77>
<177, 56, 18> <100, 13, 77>
<149, 56, 19> <177, 26, 66>
<87, 56, 20> <87, 26, 68>
34
> Modexp(4,34,179) eq 15;
true
```

Week 12

§50 Square roots modulo a prime

We consider the following problem: *given a prime $p > 2$ and a positive integer a less than p, solve the congruence $x^2 \equiv a \pmod{p}$.*

As we shall see later, this problem has connections with cryptography, via a public key cryptosystem called *Rabin's cryptosystem*. In fact the system we shall present for solving $x^2 \equiv a$ can be readily generalized to give a system for solving $x^k \equiv a \pmod{p}$ for any k, but we shall not bother with this since we only need the case $k = 2$ for the application to cryptography.

It turns out that the problem is just about trivial in the case $p \equiv 3 \pmod{4}$, and this means that Rabin's cryptosystem becomes particularly easy to implement if we deal only with primes satisfying this condition. Nevertheless, we shall also consider the harder case $p \equiv 1 \pmod{4}$.

As with all equations in residue arithmetic, solving $x^2 \equiv a \pmod{p}$ is a finite problem. We can just try $x \equiv 1$, then $x \equiv 2$, then $x \equiv 3$, and so on, until we find a solution. We want a faster method than this.

However, it is worth noting that if we did try the exhaustive search method then we would only have to test x values from 1 to $\frac{1}{2}(p-1)$, since after that we get repetitions: if $x > p/2$ then $(p - x) < p/2$, and $(p - x)^2 \equiv x^2 \pmod{p}$.

One possible method for solving $x^2 \equiv a \pmod{p}$ is to use discrete logs. If b is a primitive root mod p then

$$x^2 \equiv a \pmod{p}$$

is equivalent to

$$2\log_{b,p}(x) \equiv \log_{b,p}(a) \pmod{p-1}.$$

Since this is simply a linear congruence for $\log_{b,p}(x)$, we can easily solve it to find all the possible values for $\log_{b,p}(x)$, and hence all the possible values for x. The drawbacks of this idea are that we would have to first find a primitive root—which involves some work—and would then have to compute $\log_{b,p}(a)$, which might well be difficult. Nevertheless, it turns out that part of our strategy for solving $x^2 \equiv a$ does involve using discrete logs, though only in a case where the log we need to find is particularly easy to compute via the Pohlig–Hellman algorithm.

Since the function from $\{1, 2, \ldots, p-1\}$ to $\{1, 2, \ldots, p-1\}$ taking each x to the residue of $x^2 \pmod{p}$ is a two-to-one function, only half of the elements a in this set have square roots mod p, and those that do have square roots have exactly two of them. If a does have a square root, $a \equiv t^2$ say, then

$$a^{\frac{1}{2}(p-1)} \equiv (t^2)^{\frac{1}{2}(p-1)} \equiv t^{p-1} \equiv 1 \pmod p$$

by Fermat. Since the polynomial $x^{\frac{1}{2}(p-1)} - 1$ can have at most $\frac{1}{2}(p-1)$ roots mod p we conclude the residues with square roots comprise all the mod p roots of $x^{\frac{1}{2}(p-1)} - 1$. But $a^{p-1} \equiv 1 \pmod p$ implies that $a^{\frac{1}{2}(p-1)} \equiv \pm 1 \pmod p$, and so the $\frac{1}{2}(p-1)$ residues a that do not have square roots must satisfy $a^{\frac{1}{2}(p-1)} \equiv -1 \pmod p$.

Solving $x^2 \equiv a \pmod p$ is trivial if it happens that $\operatorname{ord}_p(a)$ is odd.

(12.1) ***Example:*** *Solve, if possible,* $x^2 \equiv 13$ *(mod 43).*

Solution. We have $p = 43$; so $\frac{1}{2}(p-1) = 21$. By the discussion above, $x^2 \equiv 13 \pmod{43}$ will have a solution if and only if $13^{21} \equiv 1 \pmod{43}$. Now it is easy to check that this condition is indeed satisfied:

$$\begin{aligned}
13^2 &\equiv 169 \equiv -3,\\
13^4 &\equiv (-3)^2 \equiv 9,\\
13^8 &\equiv 9^2 \equiv 81 \equiv -5,\\
13^{16} &\equiv (-5)^2 \equiv 25,\\
13^{21} &\equiv (13)^{16} \times 13^4 \times 13 \equiv 25 \times 9 \times 13\\
&\equiv 225 \times 13 \equiv 10 \times 13\\
&\equiv 130 \equiv 1.
\end{aligned}$$

So there is a solution.

But this also shows us that 13 raised to an odd power is 1 modulo 43, and multiplying through by 13 we deduce that 13 is an even power of itself! Specifically,

$$13 \equiv 13^{22} \equiv (\pm 13^{11})^2 \pmod{43}.$$

Now $13^{11} \equiv 13^8 \times 13^2 \times 13 \equiv (-5) \times (-3) \times 13 \equiv 195 \equiv 23$. So the two solutions to $x^2 \equiv 13 \pmod{43}$ are $x \equiv 20$ and $x \equiv 23 \pmod{43}$.

The method used in the above example will work whenever $p \equiv 3 \pmod 4$, since this means that $\frac{1}{2}(p-1)$ is odd. For example, if $p = 103$ then $\frac{1}{2}(p-1) = 51$, and hence $x^2 \equiv a \pmod{103}$ has a solution if and only if $a^{51} \equiv 1 \pmod{103}$. And if this condition is satisfied then $a^{52} \equiv a \pmod{103}$, so that $x \equiv \pm a^{26} \pmod{103}$ are the two solutions. More generally, whenever $p \equiv 3 \pmod 4$ and $x^2 \equiv a \pmod p$ has a solution, the solutions are $x \equiv \pm a^{\frac{1}{4}(p+1)} \pmod p$.

What can be done if $\frac{1}{2}(p-1)$ is even? The trick for solving $x^2 \equiv a \pmod{p}$ in this case is to express a as a product a_1a_2, where a_1 is a raised to some even power and $\operatorname{ord}_p(a_2)$ is a power of 2, and then solve $x_1^2 \equiv a_1$ and $x_2^2 \equiv a_2$ separately. Given that $a_1 \equiv a^{2l}$ for some l, solving $x_1^2 \equiv a_1$ is trivial: the solutions are $x \equiv \pm a^l$. The fact that $\operatorname{ord}_p(a_2)$ is a power of 2 makes it straightforward to solve $x_2^2 \equiv a_2$ by first solving an easy discrete log problem. Having found x_1 and x_2, the solution to the original congruence is $x \equiv x_1x_2 \pmod{p}$.

Let 2^k be the largest power of 2 that divides $p-1$, so that $p-1 = 2^k m$ with m odd, and suppose that we wish to solve $x^2 \equiv a \pmod{p}$. Of course we must first check that there is a solution by checking that $a^{\frac{1}{2}(p-1)} \equiv 1 \pmod{p}$. Assuming that this condition is indeed satisfied, we see that

$$(a^m)^{2^{k-1}} = a^{\frac{1}{2}(p-1)} \equiv 1 \pmod{p},$$

and it follows that the order of a^m mod p is a divisor of 2^{k-1}. Noting also that $1-m$ is even, we define

$$\begin{aligned} a_1 &\equiv a^{1-m} \pmod{p} \\ a_2 &\equiv a^m \pmod{p}, \end{aligned}$$

giving $a_1a_2 \equiv a^{1-m}a^m \equiv a$, so that a_1 and a_2 satisfy the requirements stated above. Since $1-m$ is negative, of course $a_1 \equiv a^{1-m}$ must be interpreted as saying that a_1 is the mod p inverse of a^{m-1}. The solutions of $x_1^2 \equiv a_1$ are $x_1 \equiv \pm a^{\frac{1}{2}(1-m)}$; that is, $x_1 \equiv \pm a'$ where a' is the mod p inverse of $a^{\frac{1}{2}(m-1)}$.

The remaining problem is to solve $x_2^2 \equiv a_2 \pmod{p}$, given that $p = 2^k m$ and $\operatorname{ord}_p(a_2)$ divides 2^{k-1}. The first step is to find some b with the property that $\operatorname{ord}_p(b)$ is 2^k. It is then guaranteed that it will be possible to find an integer i such that $b^i \equiv a_2 \pmod{p}$, and, moreover, it is also guaranteed that i will be even. And if $i = 2j$ then the solutions of $x_2^2 \equiv a_2$ will be $x_2 \equiv \pm b^j \pmod{p}$.

If r is any non-square mod p and we put $b \equiv r^m$ then it will be the case that $\operatorname{ord}_p(b) = 2^k$. Since half the nonzero residues mod p are non-squares, and it is quick to test whether something is a non-square, it is easy to find a suitable r, and hence a suitable b.

Before continuing, let us prove the various assertions that we have made.

(12.2) Proposition: *Let $p > 2$ be a prime, and let $p-1 = 2^k m$ with m odd. If r is a non-square mod p and $b \equiv r^m \pmod{p}$ then* $\operatorname{ord}_p(b) = 2^k$. *Moreover, if $a^{2^{k-1}} \equiv 1 \pmod{p}$ then $a \equiv b^i \pmod{p}$ for some even integer i.*

Proof. Since r is a non-square, $r^{\frac{1}{2}(p-1)} \equiv -1 \pmod{p}$. So

$$b^{2^{k-1}} \equiv (r^m)^{2^{k-1}} \equiv r^{\frac{1}{2}(p-1)} \equiv -1,$$

and it follows that the order of b mod p is not a divisor of 2^{k-1}. But

$$b^{2^k} \equiv (b^{2^{k-1}})^2 \equiv (-1)^2 \equiv 1,$$

and so the order of b mod p is a divisor of 2^k. Since the only divisor of 2^k that is not also a divisor of 2^{k-1} is 2^k itself, we conclude that $\mathrm{ord}_p(b) = 2^k$.

Since $\mathrm{ord}_p(b) = 2^k$ it follows that $b^{i_1} \equiv b^{i_2} \pmod{p}$ if and only if $i_1 \equiv i_2 \pmod{2^k}$. In particular, $b^0, b^1, b^2, \ldots, b^{2^k-1}$ are pairwise inequivalent mod p. Furthermore, we see that all of these powers of b are solutions of $x^{2^k} \equiv 1 \pmod{p}$, since $(b^i)^{2^k} \equiv (b^{2^k})^i \equiv 1^i \equiv 1$. But by Theorem (9.3) we know that this congruence has exactly 2^k solutions mod p. So the mod p residues of $b^0, b^1, b^2, \ldots, b^{2^k-1}$ are all the solutions of $x^{2^k} \equiv 1 \pmod{p}$. (Recall that a polynomial of degree d can have at most d roots mod p; so, having found 2^k distinct residues that are mod p roots of $x^{2^k} - 1$, we know that there can be no others.) But if $a^{2^{k-1}} \equiv 1 \pmod{p}$ then $a^{2^k} \equiv 1^2 \equiv 1 \pmod{p}$; so a is a solution of $x^{2^k} \equiv 1 \pmod{p}$, and it follows that $a \equiv b^i$ for some natural number i less than 2^k.

It remains to prove that i is even. But since $b^{2^{k-1}} \equiv -1$ we have that

$$(-1)^i \equiv (b^{2^{k-1}})^i \equiv (b^i)^{2^{k-1}} \equiv a^{2^{k-1}} \equiv 1,$$

which immediately implies that i is even. □

We are left with the task of finding this integer i. Since i is a residue mod 2^k it is enough to find out what i is congruent to mod 2^k. The process is exactly as we described last week in our discussion of the Pohlig–Hellman algorithm, and involves first finding the residue of i mod 2, then mod 2^2, then mod 2^3, and so on. We illustrate it with an example.

(12.3) ***Example:*** *Solve, if possible,* $x^2 \equiv 11$ *(mod 97).*

Solution. There is a solution if and only if $11^{48} \equiv 1 \pmod{97}$. It is easy to check this condition, even without a calculator. Successive squaring gives $11^2 = 121 \equiv 24$, then $11^4 \equiv 576 \equiv -6$, then $11^8 \equiv 36$, then $11^{16} \equiv 1296 \equiv 35$. Since we want 11^{48}, let us find 11^{24} and then square that. We have

$$11^{24} \equiv 11^8 \times 11^{16} \equiv 36 \times 35 \equiv 1260 \equiv -1,$$

so that it is indeed the case that $11^{48} \equiv 1$, and 11 is a square mod 97.

If our prime p were congruent to 3 modulo 4 then $\frac{1}{2}(p-1)$ would be odd, and we would be able to find a square root of 11 in just one step. Each extra power of 2 appearing as a factor of $p-1$ makes the task one step harder. In this example $p-1$ is divisible by 2^5; so in fact we shall have to proceed through a relatively large number of steps.

We have $p - 1 = 96 = 3 \times 32$; so the largest odd factor of $p-1$ is $m = 3$. Following the general method explained above, we write $11 = a_1a_2$, where $a_1 \equiv 11^{1-m} \equiv 11^{-2}$ and $a_2 \equiv 11^m \equiv 11^3$. Solving $x_1^2 \equiv a_1$ is trivial since a_1 is an even power of 11: the solutions are $x_1 \equiv \pm 11^{-1}$. It is easily checked that the inverse of 11 mod 97 is 53; so $x_1 \equiv 53$ or 44. Solving $x_2^2 \equiv 11^3$ is the more difficult part, and involves using the Pohlig–Hellman algorithm.

The first step is to find a residue b having order 32 mod 97; this is done by finding any non-square r and computing the residue of r^3. It can be checked that 2^{48} and 3^{48} are both congruent to 1 mod 97, but $5^{48} \equiv -1 \pmod{97}$. So 5 is a non-square, and so we choose $b \equiv 5^3 \equiv 125 \equiv 28$. The general theory described above guarantees that there is an even integer $i \in \{0, 1, \ldots, 31\}$ such that $28^i \equiv a_2$; that is,

$$28^i \equiv 11^3 \pmod{97}.$$

Our task is to find this i.

Write $i = 2j$, so that the problem is to solve $28^{2j} \equiv 11^3$. From calculations we have already done we find that

$$(11^3)^8 \equiv 11^{24} \equiv -1 \pmod{97}$$

and

$$28^{16} \equiv (5^3)^{16} \equiv 5^{48} \equiv -1 \pmod{97}.$$

Thus $(-1)^j \equiv (28^{16})^j \equiv (28^{2j})^8 \equiv (11^3)^8 \equiv -1$, and we conclude that j is odd. So we can write $j = 2l + 1$, and this transforms the problem into $28^{4l+2} \equiv 11^3$, or $28^{4l} \equiv 11^3 \times 28^{-2}$.

We need to compute the right hand side of this. We find that $28^2 \equiv 784 \equiv 8$, giving $28^{-2} \equiv -12$, and $11^3 \equiv 11^2 \times 11 \equiv 24 \times 11 \equiv -27$. So $28^{4l} \equiv 12 \times 27 \equiv 33$, and raising this to the 4th power gives

$$(-1)^l \equiv (28^{16})^l \equiv (28^{4l})^4 \equiv 33^4 \equiv 22^2 \equiv -1.$$

So l is odd, and we can write $l = 2t + 1$. This gives $j = 4t + 3$, as well as transforming the problem into $28^{8t+4} \equiv 33$, or $28^{8t} \equiv 33 \times 28^{-4}$. Since $28^{-4} \equiv (-12)^2 \equiv 47$ we obtain

$$28^{8t} \equiv 33 \times 28^{-4} \equiv 33 \times 47 \equiv 3 \times 517 \equiv 3 \times 32 \equiv -1.$$

Hence we deduce that

$$(-1)^t \equiv (28^{16})^t \equiv (28^{8t})^2 \equiv 1,$$

showing that t is even. Thus $t = 2u$ for some u, giving $j = 8u + 3$, and $28^{16u} \equiv -1$. This finally tells us that $(-1)^u \equiv (28^{16})^u \equiv -1$, and hence that u is odd. Now writing $u = 2v + 1$ we have

$$i = 2j = 16u + 6 = 16(2v + 1) + 6 = 32v + 22$$

giving $i = 22$ since $i \in \{0, 1, \ldots, 31\}$.

We can now say that $x_2 = 28^{11}$ is a solution of $x_2^2 \equiv a_2$. Recall that $28^2 \equiv 8$, so that $28^4 \equiv 64 \equiv -33$, and $28^8 \equiv 33^2 \equiv 22$. This gives $28^{11} \equiv 22 \times 8 \times 28 \equiv -19$. Finally, the solutions of $x^2 \equiv 11$ are $x \equiv x_1 x_2 \equiv \pm(53 \times 19) \equiv \pm 37 \pmod{97}$.

(12.4) *Example:* *Solve, if possible,* $x^2 \equiv 2$ *(mod 41).*

Solution. We check that $2^{20} \equiv 1 \pmod{41}$. So a solution exists.

We have $40 = 5 \times 8$; so $m = 5$ and $a_1 = 2^{1-5}$ and $a_2 = 2^5 \equiv -9$. The square roots of a_1 are $\pm 2^{-2} \equiv \pm 4^{-1} \equiv \pm 10$; we now set about finding the square roots of a_2.

We must find a non-square. We have already checked that 2 is a square; so we test 3, and readily calculate that $3^{20} \equiv -1 \pmod{41}$. So 3 will do. We now put $b \equiv 3^5$, since this will be an element of order 8. Observe that $3^4 \equiv 81 \equiv -1$; so $b \equiv -3$.

We must solve $(-3)^i \equiv -9$, and we know that i will be even. So we put $i = 2j$, and the problem becomes $(-3)^{2j} \equiv -9$. Squaring this and using $(-3)^4 \equiv -1$ gives

$$(-1)^j \equiv (-3)^{4j} \equiv 81 \equiv -1$$

so that j is odd. Say $j = 2l + 1$, so that $i = 4l + 2$. Now $(-3)^{4l+2} \equiv -9$, and cancelling $(-3)^2 = 9$ from this gives $(-3)^{4l} \equiv -1$. So $l = 2t + 1$ for some t. This gives $i = 4(2t + 1) + 2 = 8t + 6$, and since i is meant to be a residue mod 8 we conclude that $i = 6$. So we have shown that $(-3)^6 \equiv -9$ (which is indeed obviously true since $(-3)^4 = 81 \equiv -1$). So the square roots of a_2 are $\pm(-3)^3 \equiv \pm 14$, and $x \equiv x_1 x_2 \equiv \pm(10 \times 14) \equiv \pm 17$.

You can easily check that $17^2 = 289 \equiv 2 \pmod{41}$ and $24^2 \equiv 576 \equiv 2 \pmod{41}$.

---

§51 Square roots modulo a product of two primes

Let $p, q > 2$ be distinct primes. By the Chinese Remainder Theorem

$$x^2 \equiv a \pmod{pq} \tag{23}$$

is equivalent to

$$x^2 \equiv a \pmod{p} \qquad \text{and} \qquad x^2 \equiv a \pmod{q}.$$

Assuming that $p \nmid a$ and $q \nmid a$, these both have two solutions or none. So it follows that if $\gcd(a, pq) = 1$ and $x^2 \equiv a \pmod{pq}$ has a solution then there are four possibilities:

$$x \equiv b \pmod{p} \quad \text{and} \quad x \equiv c \pmod{q},$$

or

$$x \equiv b \pmod{p} \quad \text{and} \quad x \equiv -c \pmod{q},$$

or

$$x \equiv -b \pmod{p} \quad \text{and} \quad x \equiv c \pmod{q},$$

or

$$x \equiv -b \pmod{p} \quad \text{and} \quad x \equiv -c \pmod{q}.$$

where b, c satisfy $b^2 \equiv a \pmod{p}$ and $c^2 \equiv a \pmod{q}$.

The Chinese Remainder Theorem tells us that each of these possibilities has exactly one solution mod pq. So Eq. (23) has four solutions mod pq. Thus if we let S be the set of residues modulo pq that are coprime to pq then the function $S \to S$ mapping each x to x^2 (reduced mod pq) is a four to one function, and consequently its image is a quarter of the size of S. So one quarter of the residues coprime to pq are squares mod pq, and each square has four square roots.

For example, if $p = 5$ and $q = 7$ then $m = pq = 35$, and

$$\phi(m) = \phi(pq) = \phi(p)\phi(q) = (p-1)(q-1) = 4 \times 6 = 24,$$

which tells us that there are 24 residues mod 35 that are coprime to 35. Of course it is easy to write them down: 1, 2, 3, 4, 6, 8, 9, 11, 12, 13, 16, 17, 18, 19, 22, 23, 24, 26, 27, 29, 31, 32, 33, 34. When we square these we find that they come together in sets of size four, giving us just $24/4 = 6$ residues coprime to 35 that are squares.

i	1	2	3	4	6	8	9	11	12	13	16	17
i^2	1	4	9	16	1	29	11	16	4	29	11	9

i	18	19	22	23	24	26	27	29	31	32	33	34
i^2	9	11	29	4	16	11	29	1	16	9	4	1

(12.5) ***Example:*** *Solve $x^2 \equiv 60$ (mod 77).*

Solution. If $x^2 \equiv 60$ (mod 77) then $x^2 \equiv 60 \equiv 4$ (mod 7), and so $x \equiv \pm 2$ (mod 7). Of course if we had some enormous prime in place of 7 then we would not be able to just write down the square roots like this; we would have to compute them using the method described in §50. Let us apply this method anyway, just to confirm that it works! Since $p = 7 \equiv 3$ (mod 4), the square roots of a—if a has square roots—are given by $\pm a^{\frac{1}{4}(p+1)} \equiv \pm a^2$. So the square roots of 4 are $\pm 4^2 \equiv \pm 16 \equiv \pm 2$ (mod 7). Beware that if a does not have a square root then $s = a^{\frac{1}{4}(p+1)}$ will not be a square root of a! In fact, it will turn out that $s^2 \equiv -a$ in this case. So you should always be careful to check your answer.

Turning to the other prime factor of 77 and applying the same method we find that $x^2 \equiv 60 \equiv 5$ (mod 11). Since we also have $11 \equiv 3$ (mod 4) we can find square roots by finding $\frac{1}{4}(11+1)$-th powers. Since $\frac{1}{4}(11+1) = 3$ the square roots of 5 (mod 11) should be $\pm 5^3 \equiv \pm(5^2 \times 5) \equiv \pm(3 \times 5) \equiv \pm 14 \equiv \pm 4$, provided that 5 has a square root. Checking we find that indeed $4^2 \equiv 16 \equiv 5$ (mod 11).

The original problem is now reduced to a Chinese Remainder Theorem problem: solve the simultaneous congruences $x \equiv 2\varepsilon_1$ (mod 7) and $x \equiv 4\varepsilon_2$ (mod 11) for all choices of $\varepsilon_1, \varepsilon_2 \in \{\pm 1\}$. The latter condition gives $x = 11k + 4\varepsilon_2$ for some k, after which the former gives $4k + 4\varepsilon_2 \equiv 2\varepsilon_1$ (mod 7), since $11 \equiv 4$ (mod 7). Multiplying through by 2 (the inverse of 4 mod 7) gives $k \equiv 4\varepsilon_1 - \varepsilon_2$ (mod 7), so that $11k \equiv 44\varepsilon_1 - 11\varepsilon_2$ (mod 77), and

$$x = 11k + 4\varepsilon_2 \equiv 44\varepsilon_1 - 7\varepsilon_2 \pmod{77}.$$

Putting in all the possible values for the signs ε_1, ε_2, and noting that $-44 \equiv 33$, we see that the solution of $x^2 \equiv 60 \pmod{77}$ is $x \equiv 26$, 37, 40 or 51 (mod 77).

(12.6) ***Example:*** *Solve* $x^2 \equiv 1123$ *(mod 10919), given that* $10919 = 61 \times 179$.

Solution. We find that $1123 = 18 \times 61 + 25$ and $1123 = 6 \times 179 + 49$. So the problem is to solve $x^2 \equiv 25 \pmod{61}$ and $x^2 \equiv 49 \pmod{179}$ simultaneously. We require $x \equiv 5\varepsilon_1 \pmod{61}$ and $x \equiv 7\varepsilon_2 \pmod{179}$ with $\varepsilon_1, \varepsilon_2 \in \{\pm 1\}$. So for some k we have

$$x = 179k + 7\varepsilon_2 \equiv -4k + 7\varepsilon_2 \pmod{61}$$

giving $-4k + 7\varepsilon_2 \equiv 5\varepsilon_1 \pmod{61}$. The inverse of -4 mod 61 is 15, and so we obtain $k \equiv -(15 \times 7)\varepsilon_2 + (15 \times 5)\varepsilon_1 \equiv 17\varepsilon_2 + 14\varepsilon_1$. Thus

$$179k \equiv (179 \times 17)\varepsilon_2 + (179 \times 14)\varepsilon_1 \equiv 3043\varepsilon_2 + 2506\varepsilon_1 \pmod{10919}$$

and the solution to the original problem is $x \equiv 3050\varepsilon_2 + 2506\varepsilon_1 \pmod{10919}$, the four choices of the signs giving four values for x. Thus the solutions are 5556, 554, 10375 and 5363 (and numbers congruent to these mod 10919).

Our first step in solving $x^2 \equiv a \pmod{m}$ was to factorize m. But we know that factorizing m is potentially difficult. What if factorizing m proves to be too difficult? Is there any other way to find square roots?

The short answer to this is "No". The longer answer is that if there were a polynomial time algorithm for finding square roots then there would be a polynomial time algorithm for factorizing, and conversely. Although we shall not give a formal proof this result, we can give a reasonably convincing argument for it.

If we had an algorithm that would always find all the mod m square roots of any mod m residue that has one, then we could find a nontrivial factor of m as follows: use our square root algorithm to find all the mod m square roots of 1; then, choosing s to be any one of these square roots other than ± 1, apply the Euclidean algorithm to find $\gcd(s-1, m)$ and $\gcd(s+1, m)$. These numbers are (of course) factors of m; so as long as one of them is greater than 1 and less than m then we have found a nontrivial factor of m. Now since $s^2 \equiv 1 \pmod{m}$ we must have that $m \mid (s^2 - 1) = (s-1)(s+1)$; so if p is a prime factor of m then either $s-1$ or $s+1$ must be divisible by p, and this ensures that at least one of $\gcd(s-1, m)$ and $\gcd(s+1, m)$ is greater than 1.* But we have chosen s so that $s \not\equiv \pm 1 \pmod{m}$, and this guarantees that $m \nmid (s-1)$ and $m \nmid (s+1)$. So both $\gcd(s-1, m)$ and $\gcd(s+1, m)$ are less than m. Hence the result follows.

* In fact we may as well assume that m is odd, and then it is rather easy to show that m is the product of $\gcd(s-1, m)$ and $\gcd(s+1, m)$.

Even if our square root algorithm only finds one square root of a rather than all the square roots of a, we still will be able to use it to find a factor of m fairly quickly. Just pick a random residue r and square it, and let a be the residue of the result. Our square root algorithm will give us a residue r' such that $(r')^2 \equiv a \equiv r^2$, but since a has four square roots (assuming that m is the product of two distinct primes p and q and $\gcd(a, m) = 1$) and our original choice of r was random, there is only a two in four chance that $r' \equiv \pm r$. If we are unlucky, and it happens that $r' \equiv \pm r$, we just pick another r and try again. Soon we shall have a situation where $m \mid (r - r')(r + r')$ and $m \nmid (r - r')$ and $m \nmid (r + r')$. Then either $\gcd(r - r', m)$ or $\gcd(r + r', m)$ will be a nontrivial factor of m.

The following theorem summarizes the above discussion.

(12.7) Theorem: *If $m = pq$, where p and q are distinct odd primes, then the problem of finding p and q given m is computationally equivalent to the problem of finding square roots mod m.*

§52 Rabin's public key cryptosystem

Bill chooses two very large primes p and q, both congruent to 3 mod 4. The pair (p, q) is Bill's private key and the product $m = pq$ is his public key. To send a message to Bill, Amy must first encode the message numerically as a sequence of residues mod m. The resulting sequence $[t_1, t_2, \ldots, t_k]$ is the (encoded form of) the plaintext. The ciphertext is $[s_1, s_2, \ldots, s_k]$ where each s_i is a residue mod m and $s_i \equiv t_i^2 \pmod{m}$.

When he receives the ciphertext Bill computes (for each i) the four mod m square roots of s_i. He has the superficial problem of deciding which of the four square roots is t_i, but this is no great problem in practice since only one of the four will decode into anything meaningful.

Like RSA, the security of Rabin's cryptosystem relies on the fact that it is believed to be computationally infeasible to find p and q knowing only m, provided that these numbers are chosen to be large enough. In the case of RSA, it is strongly believed—but not actually proved—that there is no way to crack the system other than by factorizing m. For Rabin's system, however, it is proved that the problem of cracking the system is computationally equivalent to the problem of factorizing m. The theoretical drawback of Rabin's system is that square roots are not unique. Although one may feel that this make's Rabin's system inelegant and unsatisfying, the truth is that it is not a practical drawback at all.

Index