

NOTES ON NUMBER THEORY AND CRYPTOGRAPHY

KARL PETERSEN

1. MODULAR ARITHMETIC

Many interesting and useful properties of the set of integers \mathbb{Z} (the whole numbers $\dots, -3, -2, -1, 0, 1, 2, 3, 4, 5, \dots$) can be studied by thinking in terms of divisibility by a particular base or *modulus*, a positive integer m . For example, it is frequently important to know whether a given number k is even or odd, that is to say, whether or not k is (evenly) divisible by 2. (An even number of young swimmers can be paired off in a buddy system, but if there are an odd number, a problem arises.) The entire set \mathbb{Z} of integers breaks into two disjoint subsets, the evens and the odds. Note that two numbers x and y are in the same class if and only if their difference is *even*, that is, 2 divides $x - y$. Further, the sum of any two even numbers is even, the sum of any two odd numbers is even, and the sum of an even number and an odd number is odd. This allows us to do arithmetic with the two classes of even and odd numbers. Similar statements hold for divisibility by any positive integer modulus m , and we now give those statements formally.

Definition 1. Let m be a positive integer. We say that two integers x and y are *congruent modulo m* , and write $x \equiv y \pmod{m}$, alternatively $x \equiv_m y$, in case m divides $x - y$, which is written as $m|(x - y)$.

Thus $x \equiv y \pmod{m}$ if and only if there is $k \in \mathbb{Z}$ such that $x - y = km$.

The relation $x \equiv_m y$ divides the set of integers \mathbb{Z} into m disjoint *congruence classes*:

$[0]_m$ = all those $x \in \mathbb{Z}$ which are congruent to $0 \pmod{m}$, i.e., all those x which are divisible by m ;

$[1]_m$ = all those $x \in \mathbb{Z}$ which are congruent to $1 \pmod{m}$, i.e., all those x which leave a remainder of 1 when divided by m ;

.

$[m-1]_m =$ all those $x \in \mathbb{Z}$ which are congruent to $m-1 \pmod m$, i.e., all those x which leave a remainder of $m-1$ when divided by m .

This set of congruence classes is denoted by \mathbb{Z}_m . It has exactly m elements. We will see in a moment that it is possible to do some arithmetic with these elements.

Think of the integers as being written on a long ribbon, which is then wrapped around a circle of circumference m . The numbers that fall on top of (or underneath) 0 constitute the congruence class $[0]_m$, and similarly for the congruence classes of $1, 2, \dots, m-1$. Each of the m congruence classes contains infinitely many elements. Any two elements in the same congruence class differ by a multiple of m ; and elements of different congruence classes always differ by an amount that is not a multiple of m . The congruence classes are also called *residue classes*, since they concern the remainders, or residues, left after division by m .

To work with \mathbb{Z}_m , we can deal either with entire congruence classes or else just pick convenient representatives from each congruence class. A set of integers that contains exactly one member of each congruence class in \mathbb{Z}_m is called a *complete set of representatives modulo m* . A particularly natural way to do this is to pick the smallest nonnegative member of each class; this gives the complete set of representatives $\{0, 1, \dots, m-1\}$. If x is any integer, its representative in this set is denoted by $x \pmod m$:

$$x \pmod m = \text{that } j \in \{0, 1, \dots, m-1\} \text{ such that } x \equiv_m j.$$

This is not always the most convenient set of representatives. For example, when $m = 3$ the complete set of representatives $\{-1, 0, 1\}$ is handy because multiplying is really easy.

This brings us up to the topic of arithmetic modulo m .

Proposition 1. *If $a \equiv_m c$ and $b \equiv_m d$, then $a + b \equiv_m c + d$, $a - b \equiv_m c - d$, and $ab \equiv_m cd$.*

Exercise 1. Prove this Proposition by using the definition of congruence modulo m .

This Proposition allows us to do addition, subtraction, and multiplication with the elements of \mathbb{Z}_m . For example, to compute

$$j + k \pmod m,$$

choose any $a \in [j]_m$ (that is, any integer a that is congruent to j modulo m) and any $b \in [k]_m$, and compute $a + b$. The answer is the congruence class modulo m of this result, namely $[a + b]_m$. If we had made different choices of a and b (say we had chosen $c \equiv_m j$ and $d \equiv_m k$), we might have gotten a different result for $j + k$, but the *congruence class* $[j + k]_m$ of the result would have been the same.

These arithmetic operations in \mathbb{Z}_m obey the familiar laws of arithmetic. Addition and multiplication are associative ($x(yz) = (xy)z$) and commutative ($xy = yx$). There is an identity element, $[0]_m$ for addition modulo m . Every element of \mathbb{Z}_m has an additive inverse: $[a]_m + [-a]_m = [0]_m$. And multiplication distributes over addition: $x(y + z) = xy + yz$. Any set with operations satisfying these conditions is called a *ring*.

\mathbb{Z}_m also has an identity element, $[1]_m$, for multiplication. What about division? Can we find multiplicative inverses of nonzero elements?

There is a bit of a problem here. Let's consider the case $m = 6$. If $a = [5]_6$, then we can indeed find a multiplicative inverse for a : $a \cdot [5]_6 = [5]_6[5]_6 = [5 \cdot 5]_6 = [25]_6 = [1]_6$, so that a is its own multiplicative inverse! But looking at $b = [2]_6$, when we multiply in turn by all the possible candidates, namely the classes of $0, 1, 2, 3, 4, 5$, we get $[0]_6, [2]_6, [4]_6, [0]_6, [2]_6, [4]_6$, and none of these equals $[1]_6$!

We can tell that there is trouble because $[2]_6[3]_6 = [0]_6$, so that neither $[2]_6$ nor $[3]_6$ can have a multiplicative inverse modulo 6. If $a, b \in \mathbb{Z}_m$ satisfy $ab = [0]_m$, and if, say, a has a multiplicative inverse x modulo m , then $[0]_m = x[0]_m = x(ab) = (xa)b = 1 \cdot b = b$. Thus if two nonzero elements of \mathbb{Z}_m have product equal to $[0]_m$, then neither of them can have a multiplicative inverse in \mathbb{Z}_m .

2. COMMON DIVISORS AND THE EUCLIDEAN ALGORITHM

The previous discussion showed that it can be important to know whether or not two positive integers have a common divisor. The *greatest common divisor* of two positive integers m and n is defined already by its name; it is denoted by $\gcd(m, n)$ or occasionally and more briefly, when no confusion is likely because of the context, simply by (m, n) . We say that m and n are *relatively prime* in case $\gcd(m, n) = 1$. Two integers have no common prime divisors if and only if they are relatively prime. Isn't it clear that if n_1 and m are relatively prime, and $n_1 \equiv_m n_2$, then n_2 and m are relatively prime? Thus we can talk about

congruence classes relatively prime to m . We will see below that the set of congruence classes relatively prime to a fixed modulus m forms an interesting and important set; already the number of elements that it contains deserves attention and study.

Definition 2. For each positive integer $m > 1$, the number of integers $n \in \{0, 1, \dots, m-1\}$ such that $\gcd(n, m) = 1$ is denoted by $\phi(m)$. We also define $\phi(1) = 1$. The function ϕ is called the *Euler ϕ - (or totient) function*.

Exercise 2. Show that if p is prime, then $\phi(p) = p - 1$.

Exercise 3. Show that if p and q are distinct primes, then $\phi(pq) = (p - 1)(q - 1)$.

Recall that every positive integer m has a unique factorization

$$(1) \quad m = p_1^{e_1} \cdots p_r^{e_r},$$

where $r \geq 1$, p_1, \dots, p_r are prime numbers with $p_i < p_j$ whenever $1 \leq i < j \leq r$, and e_1, \dots, e_r are positive integers. In fact, to prove this statement one usually uses the Euclidean Algorithm, which we are about to discuss (so our presentation is not in strict logical order).

The greatest common divisor of m and n can be read off from looking at their prime factorizations: for each prime that appears in both factorizations, take the lowest of the two powers to which it appears, then multiply together all these prime powers to arrive at $d = \gcd(m, n)$. But usually we do not know the prime factorizations of numbers that we come across, and it is a hard problem to determine the prime factorizations of numbers. Indeed, it can be very hard to tell whether a number is prime. (The apparent difficulty of primality testing and prime factorization is at the base of modern asymmetric, public-key cryptographic systems.) Therefore it is important to have a straightforward method for calculating greatest common divisors, and this is what the Euclidean Algorithm provides. We will see that the information accumulated while running the algorithm is also useful for calculating multiplicative inverses in \mathbb{Z}_m .

The Euclidean Algorithm for finding $d = \gcd(m, n)$ just consists of repeated division, keeping track of the remainders. The last nonzero remainder is the sought-for d . Let us suppose that $m < n$. We put $r_0 = n$ and $r_1 = m$. Now divide r_0 by r_1 , getting a remainder r_2 with $0 \leq r_2 < r_1$:

$$(2) \quad r_0 = k_1 r_1 + r_2.$$

Now divide r_1 by r_2 , getting a remainder r_3 with $0 \leq r_3 < r_2$, and continue this process until arriving at a remainder of 0:

$$\begin{aligned}
 r_0 &= k_1 r_1 + r_2 \\
 r_1 &= k_2 r_2 + r_3 \\
 (3) \quad & \vdots \\
 r_{i-1} &= k_i r_i + r_{i+1} \\
 r_i &= k_{i+1} r_{i+1} + 0.
 \end{aligned}$$

Because the remainders are all nonnegative and they are decreasing, each smaller than the next, eventually one has to equal 0. The one just before this, the last nonzero remainder, is $r_{i+1} = d = \gcd(m, n)$.

Example 1. Let us perform this algorithm to find the greatest common divisor of $m = 24$ and $n = 128$:

$$\begin{aligned}
 (4) \quad & 128 = 5 \cdot 24 + 8 \\
 & 24 = 3 \cdot 8 + 0 \\
 & \gcd(24, 128) = 8.
 \end{aligned}$$

Check by looking at the prime factorizations: $24 = 2^3 \cdot 3$, $128 = 2^7$.

How do we see that the Euclidean Algorithm does, as claimed, produce the greatest common divisor of r_0 and r_1 ? The key is to notice that any integer k that divides both r_0 and r_1 also divides r_2 : this is because $k|a$ and $k|b$ implies $k|(a - b)$. Then by the same reasoning any such k must also divide r_3 , and so on, down to $d = r_{i+1}$. So any common divisor of r_0 and r_1 also divides r_{i+1} . Similarly, the equations show that r_{i+1} divides r_i , hence also r_{i-1} , etc., until we see that r_{i+1} divides both r_0 and r_1 . This shows that indeed $r_{i+1} = \gcd(m, n)$.

The following fact may seem strange at first, but it turns out to be very useful for computing multiplicative inverses modulo m . It says that if m and n are positive integers, and we lay off all of their integer multiples on the number line, then it is possible to find two of them at distance $\gcd(m, n)$ from one another.

Proposition 2. *If m and n are positive integers and $d = \gcd(m, n)$, then there are integers x and y such that*

$$(5) \quad d = xm - yn.$$

Remark 1. Replacing y by $-y$, this is the same as being able to find integers x and y such that

$$(6) \quad d = xm + yn.$$

Proof. This is accomplished by reading backwards the sequence of steps in the Euclidean Algorithm for finding $\gcd(m, n) = r_{i+1}$. We have

$$(7) \quad d = r_{i+1} = r_{i-1} - k_i r_i.$$

Plugging in

$$(8) \quad r_i = r_{i-2} - k_{i-1} r_{i-1},$$

we find

$$(9) \quad \begin{aligned} r_{i+1} &= r_{i-1} - k_i(r_{i-2} - k_{i-1}r_{i-1}) \\ &= r_{i-1}(1 + k_i k_{i-1} - k_i r_{i-2}). \end{aligned}$$

Continuing in this way, we arrive eventually at

$$(10) \quad r_{i+1} = xr_0 + yr_1,$$

as required. \square

Example 2. Looking at the preceding example with $m = 24$ and $n = 128$,

$$(11) \quad 8 = -5 \cdot 24 + 128.$$

Of course this was a bit easy, involving only one step.

Exercise 4. (a) Use the Euclidean Algorithm to find $\gcd(792, 2145)$.
 (b) Find the prime factorizations of 792 and 2145 and use this information to check your answer to part (a).
 (c) Use the reverse of the Euclidean Algorithm to find integers x and y such that $\gcd(792, 2145) = 792x + 2145y$.

3. FINDING MODULAR MULTIPLICATIVE INVERSES

Continue to denote by m a fixed positive integer. First let us note that if n is a positive integer which is not relatively prime to m , so that $d = \gcd(m, n) > 1$, then it is *not* possible to find a multiplicative inverse for n modulo m . For if we can find some integer x with

$$(12) \quad xn \equiv_m 1,$$

this means that there is an integer k such that

$$(13) \quad xn - 1 = km,$$

so that

$$(14) \quad xn - km = 1.$$

Now d divides both terms on the left side of this equation, so d divides 1, hence $d = 1$.

On the other hand, if n is relatively prime to m , then n *does* have a multiplicative inverse modulo m . Use the Euclidean Algorithm reversed to find x and y such that

$$(15) \quad xm + yn = 1.$$

Then

$$(16) \quad yn - 1 = -xm,$$

so that $yn \equiv_m 1$. Thus $[y]_m$ is the multiplicative inverse of $[n]_m$ in \mathbb{Z}_m .

Here is another way to find modular multiplicative inverses—the *power method*. Let us assume that $\gcd(n, m) = 1$, so that we know in advance that n has a multiplicative inverse modulo m , and all we have to do is identify which congruence class of \mathbb{Z}_m it is. Let us look at the powers of n modulo m , namely

$$(17) \quad [n]_m, [n^2]_m, \dots, [n^j]_m, \dots$$

Since there are only m congruence classes in \mathbb{Z}_m and this list is infinite, there have to be repeats. In fact, I claim that before there is a repeat there is a first power $i \geq 1$ such that $n^i \equiv_m 1$. For consider the first repeat in this list, say $[n^{j+i}]_m = [n^j]_m$ with $i, j \geq 1$. Since n has a multiplicative inverse $[y]_m$ modulo m , we can multiply by it j times and obtain $n^i \equiv_m 1$. If $i = 2$, then $n^2 \equiv_m 1$, and n is its own multiplicative inverse modulo m . This can happen! For example, 5 is its own multiplicative inverse modulo 4. If $i > 1$, then $[n^{i-1}]_m$ is the multiplicative inverse of n modulo m . If $i = 1$, then $n \equiv_m 1$ and n is its own multiplicative inverse modulo m .

Exercise 5. Use both the reverse Euclidean Algorithm and the power method to find the modular inverse of 9 modulo 38.

You can use a spreadsheet (or a calculator or other software, such as Matlab or Mathematica) to perform the Euclidean Algorithm and to calculate even very long lists of powers modulo m .

Exercise 6. Set up a spreadsheet for the Euclidean Algorithm as follows. (a) in the first row, in cells C1 and D1, place the labels r_0 and r_1 . In cell E1, place the label $\text{int}(r_0/r_1)$. This will be the integer part of r_0/r_1 , that is, the number of times that r_0 “goes evenly” into r_1 . In cell F1, place the label $\text{Rem} = c - de$. Here we will put $r_0 - r_1 \text{int}(r_0/r_1)$, which is the remainder after dividing r_0 by r_1 .

(b) This was just setting up the labels. Now in C3 put 38, in D3 put 9, in E3 put $=\text{int}(C3/D3)$, and in F3 put $=C3-D3*E3$:

C	D	E	F
r0	r1	int(r0/r1)	Rem=c-de
38	9	4	2

(c) Now we want to get this to repeat. In C4 put =D3, and in D4 put =F3. This is the important recursive step in which we replace r_0, r_1 by r_1, r_2 . Put the mouse pointer on the small black square in the lower right corner of cell C4 and drag it down several rows. Do the same with each of D4–F4. The last nonzero (and non-error) entry in column F should be $\gcd(r_0, r_1)$.

(d) Change the choice of r_0 and r_1 and see whether you still get the right gcd.

Exercise 7. Set up a spreadsheet for powers modulo m as follows. In cells C1–F1 put the labels $n, m, n^j, \text{Mod}(n^j, m)$, respectively. In C3 put 9, in D3 put 38, in E3 put =C3, In C4 put =C3, in D4 put =D3, in E4 put =C3*E3, and in F3 put =Mod(E3,D3). Think about what this means! Then drag the black square in the lower right of C4 downwards a ways, similarly for D4, E4, and F3. The entry in column F above 1 should be the inverse of n modulo m . (You can keep track of the power involved if you like by putting label j in G1, entering 1 in G3, = 1+G3 in G4, and then dragging down the black square in G4.)

You may experiment around with the spreadsheet in the preceding example (and the one before). Notice that if the numbers involved get too big, the capacity of the software is exceeded and the computer refuses to give a numerical answer. Mathematica and Matlab do much better at handling large numbers than does the average spreadsheet, but eventually the capacity of the computer or our patience will be exceeded. However, there is a very nice aspect of arithmetic modulo m : *we never have to deal with numbers larger than $m - 1$* : just keep reducing modulo m at every stage. The size of numbers involved can be reduced even more by using a complete set of representatives modulo m that includes negative numbers.

Exercise 8. Modify the spreadsheet in the preceding exercise so that all calculations are done modulo m . Then use it to find the multiplicative inverse of 33 modulo 23.

Later on we will explore a bit more the algebraic nature of \mathbb{Z}_m . While thinking about the set of powers of n modulo m , in the case that $\gcd(n, m) = 1$, we may note the following properties of this set:

Closure: The product of two elements of the set is also in the set;

Identity: The (multiplicative) identity element $[1]_m$ is in the set;

Inverses: each element of the set has a multiplicative inverse in the set.

Together with the fact that multiplication in \mathbb{Z}_m is associative ($a(bc) = (ab)c$ for all $a, b, c \in \mathbb{Z}_m$), these properties say that if $\gcd(n, m) = 1$, then the set of powers of n modulo m form a *group* with the operation of multiplication modulo m . Since multiplication is also commutative ($ab = ba$ for all $a, b \in \mathbb{Z}_m$), we say that they form a *commutative* or *abelian* group.

Exercise 9. Use the modular powers spreadsheet developed previously to study the set of powers of n modulo m in several cases where n and m are not relatively prime. What structure, if any, do these sets have? What structure, if any, does \mathbb{Z}_m have in terms of these sets?

4. THE RSA PUBLIC-KEY CRYPTOSYSTEM

This brilliant and important cryptographic system was invented in 1977 by Ronald Rivest, Adi Shamir, and Leonard Adelman and discovered independently in 1973, but not announced, by Clifford Cocks and James Ellis of the British Government Communications Headquarters, the successor of Bletchley Park [2, p. 279 ff.]. The system has a strange but extremely useful asymmetry property: there are *two* keys involved: each recipient of messages has both a *public key* and *private key*. The recipient announces his public key to everyone but keeps the private key secret. Anyone can use the public key to encode messages for a particular recipient, but, curiously, only someone with knowledge of the private key can decode them. How is this possible? Why does the system work? How is it used in practice? What other applications and implications are there of the existence of such asymmetric systems?

Asymmetric cryptographic systems are based on *one-way* functions—functions whose outputs can be computed in a reasonable amount of time but whose inverses are inordinately difficult and time-consuming to compute (given the output, or result of applying the function, it is essentially impossible to determine what the input was). It is believed that prime multiplication and modular exponentiation— $f(x) = b^x \bmod m$ —have this property: factorization into primes and modular logarithms are difficult. A *trapdoor function* is a one-way function whose inverse *can* be feasibly computed if one is given an extra piece

of information. Cryptographic systems based on trapdoor functions are now used in real-world communication by governments, businesses, and individuals, notably for secure transactions over the internet.

Even the application of the RSA system takes a lot of computing power and time, because in order for the one-way property to take real effect, very large numbers must be used as keys. Thus this system is typically used to agree on a key for a standard, symmetric, single-key system, such as the Advanced Encryption Standard, which replaced the Data Encryption Standard.

Here is the algorithm for RSA encoding and decoding. We will follow up with examples, plus an explanation of why the system works.

1. The recipient's *public key* consists of the product n of two distinct large primes p and q and an *encoding exponent* e which is *relatively prime* to $m = \phi(n) = (p - 1)(q - 1)$.

2. The recipient's *private key* consists of the two primes p and q and of the *decoding exponent* d , which is the multiplicative inverse of e modulo $m = \phi(n)$: $de \equiv_m 1$. Someone who knows n and e but not m will have a great deal of trouble finding d . To know m , one should know the factors p and q of n . The factorization of n is the extra piece of information that provides the secret "trapdoor" entrance to the inverse of the presumed one-way function that accomplishes the encoding.

3. Each message to be sent is an integer $M \in \{0, 1, \dots, n - 1\}$. In practice, any text to be sent is first converted to a string of numbers, for example by assigning the numerical ASCII codes that correspond to ordinary keyboard characters. These are then written in binary (base 2) notation, so that the text becomes a string of 0's and 1's. Then a preliminary encoding may be applied, followed by encoding for error correction, for example by adding check digits. Then the text can be cut into blocks of length no more than $\log_2 n$, so that each 0, 1-block represents a unique integer between 0 and $n - 1$.

4. The message is converted to $R = M^e \pmod n$. This is the message transmitted to the recipient. Note that all senders use the same encoding method for a particular recipient. They can see each other's encoded messages, but no one can decode them except the intended recipient.

5. The recipient receives R and applies to it his secret decoding exponent d , computing, as we will later see,

$$(18) \quad R^d \equiv_n (M^e)^d = M^{ed} \equiv_n M.$$

A *modification of the RSA algorithm* allows one to replace $m = \phi(n) = (p-1)(q-1)$ in the preceding steps by $m = \text{lcm}(p-1, q-1)$ (the *least common multiple* of $p-1$ and $q-1$). We will verify this later.

It is believed that, knowing the public key n and e , it would take too much computing power and time to find the decoding exponent d , without knowledge of $m = \phi(n) = (p-1)(q-1)$, for which knowledge of the factorization $n = pq$ would be necessary. It is on this belief that the one-way property of the coding algorithm is based.

Example 3. Let's see how this works with two small primes, $p = 5$ and $q = 11$. We have $n = pq = 55$ and $m = \phi(n) = (p-1)(q-1) = 4 \cdot 10 = 40$. We need an encoding exponent e relatively prime to $40 = 2^3 \cdot 5$; let's say we choose $e = 7$. Now for our message $M \in \{0, 1, \dots, n-1\}$, say $M = 13$.

To encode, we compute $R = M^e \pmod n = (13)^7 \pmod{55}$. This can be done by spreadsheet, by calculator, or even pencil and paper—remembering that we need only deal with numbers in $\{0, 1, \dots, 54\}$ and that the power 7 can be approached rapidly by repeated squaring:

$$(19) \quad \begin{aligned} (13)^2 &= 169 \equiv_{55} 4 \\ (13)^4 &\equiv_{55} 4^2 = 16 \\ (13)^6 &\equiv_{55} 4 \cdot 16 = 64 \equiv_{55} 9 \\ (13)^7 &\equiv_{55} 9 \cdot 13 = 117 \equiv_{55} 7. \end{aligned}$$

So the encoded message is $R = 7$.

The recipient needs the decoding exponent d , which is the multiplicative inverse modulo $m = 40$ of $e = 7$. This can be found by the power method, the Euclidean algorithm reversed, or guessing. Taking powers of 7 modulo 40, perhaps with the help of the spreadsheet in Exercise 8, we find successively

$$(20) \quad 7^1 = 7, 7^2 = 49 \equiv_{40} 9, 7^3 \equiv_{40} 63 \equiv_{40} 23, 7^4 \equiv_{40} 7 \cdot 23 \equiv_{40} 161 \equiv_{40} 1,$$

so that $d \equiv_{40} 7^3 \equiv_{40} 23$.

The recipient decodes by calculating $R^d = 7^{23}$. Again the spreadsheet in Exercise 8 will do this very quickly and easily. With paper

and pencil we might calculate this way:

(21)

$$\begin{aligned}
 7^2 &= 49 \equiv_{55} -6 \\
 7^4 &\equiv_{55} 36 \\
 7^8 &\equiv_{55} (36)^2 = 1296 \equiv_{55} -24 \\
 7^{16} &\equiv_{55} (-24)^2 = 576 \equiv_{55} 26 \\
 7^{23} &\equiv_{55} 7^{16+4+2+1} \equiv_{55} (26)(36)(-6)(7) \equiv_{55} 1 \cdot (-42) \equiv_{55} 13 = M.
 \end{aligned}$$

Example 4. What if the message M happens not to be relatively prime to the modulus n ? Let's try it in the same system as above, but with, for example, $M = 15$ instead of 13.

We find $R = (15)^7 \pmod{55} = 5$, and $R^d = 5^{23} \equiv_{55} 15$ (by any of the methods suggested above), so the method seems to work in this case too. We will investigate all of this below.

Exercise 10. Set up a spreadsheet to perform RSA encoding and decoding as follows. The first column shows what to type in column C, the second what to type in column E, starting with row 2:

*p	5
*q	11
n=pq	=E2*E3
phi(n)=(p-1)(q-1)	=(E2-1)*(E3-1)
*e rel prime to phi(n)	7
*d with de=1 mod phi(n)	23
*message M in Z_n	13
$M^e \pmod n = R$	=mod(E9^E6,E4)
$R^d \pmod n$	=mod(E10^E7,E4)

The entries marked by * indicate where the user will have to supply input in those rows in column E; the spreadsheet is supposed to calculate the rest. The computer may fizzle when numbers get too large, like perhaps 7^{23} . The decoding exponent d can be found by means of the spreadsheet in Exercise 8.

Exercise 11. Let's extend the spreadsheet of the preceding exercise so that it performs the modified RSA algorithm, using $m = \text{lcm}(p-1, q-1)$ in place of $m = \phi(n) = (p-1)(q-1)$. Starting in row 14, enter

*gcd(p-1,q-1)	2
m=lcm(p-1,q-1)	=E5/E14
*f rel prime to m	7
*g with gf=1 mod m	3
message M	
M^f mod n = R	13 =mod(E19^E16,E4)
R^g mod n	=mod(E20^E17,E4)

Note that g is easier to find than d was (because m is now smaller), and the computer also has an easier time accomplishing the coding.

Exercise 12. Explain why $\text{lcm}(a, b) = (ab) / \text{gcd}(a, b)$. (This fact is used in the preceding spreadsheet.)

Exercise 13. Try to modify the preceding spreadsheet so that the computer will not be stymied by excessively large numbers. One idea could be to do the arithmetic modulo n when computing powers. For this purpose, see the spreadsheet in Exercise 8, and consider using the spreadsheet functions *index* or *lookup*, applied to vectors or arrays.

5. THE MATHEMATICS BEHIND THE RSA CRYPTOGRAPHIC SYSTEM

The key to the functioning of the RSA algorithm is the important but easy-to-understand Euler-Fermat Theorem. We proceed to develop the small amount of number theory needed to prove this theorem.

Fix a positive integer m . We want to see that the set of congruence classes relatively prime to m forms a *commutative group* G_m with respect to multiplication. Recall that if $\text{gcd}(n_1, m) = 1$, and $n_2 \equiv_m n_1$, then also $\text{gcd}(n_2, m) = 1$. Thus we are justified in using the terminology “congruence classes relatively prime to m ”. We need to check that products of elements of G_m are again in G_m , that G_m contains an identity element for multiplication, that every element of G_m has a multiplicative inverse which is in G_m , and that multiplication is commutative (taking the product of two elements in either order gives the same result).

First, recall that we know that the identity element $[1]_m$ is in G_m and that each element of $x \in G_m$ has a multiplicative inverse $y \in \mathbb{Z}_m$. We must have $y \in G_m$, because we know that y has a multiplicative inverse (x) , and we have seen above that an element of \mathbb{Z}_m has a multiplicative

inverse if and only if it is relative prime to m , i.e., if and only if it is in G_m .

We have not yet verified that G_m is *closed under multiplication*: $x, y \in G_m$ implies $xy \in G_m$. This can be seen with the help of the following very basic and important fact about prime numbers and divisibility.

Proposition 3. *If p is prime and m, n are positive integers such that $p|(mn)$, then either $p|m$ or $p|n$.*

Proof. This well-known fact seems to be clearly true, and it is an immediate consequence of the prime factorization property of integers—but the most logical among us would wonder how the prime factorization property is proved. Maybe with the help of this Proposition? The Proposition can be proved directly with the help of the very handy application of the Euclidean Algorithm, Proposition 2.

So suppose that $p|(mn)$ and p does not divide m . Since p has no divisors besides itself and 1, we have $\gcd(p, m) = 1$. Now using Proposition 2, find x and y such that $xp + ym = 1$. Multiplying through by n gives $n = nxp + ymn$. Now p divides both terms on the right side of the equation, so $p|n$. \square

Corollary 1. *If $m|(n_1n_2)$ and $\gcd(n_1, m) = 1$, then $m|n_2$.*

Proof. We just pick up the last part of the preceding proof. Since $\gcd(n_1, m) = 1$, we can find $x, y \in \mathbb{Z}$ with $xn_1 + ym = 1$. Then $n_2 = xn_1n_2 + ymn_2$ is seen to be divisible by m . \square

Corollary 2. *If $\gcd(n_1, m) = \gcd(n_2, m) = 1$, then $\gcd(n_1n_2, m) = 1$. The converse is also true ($\gcd(n_1n_2, m) = 1$ implies $\gcd(n_1, m) = \gcd(n_2, m) = 1$).*

Proof. We can argue that if $\gcd(n_1n_2, m) \neq 1$, then n_1n_2 and m have in common an integer divisor $d > 1$ and hence have in common a prime divisor. This requires the extra observation that every positive integer greater than 1 is either prime or else has a prime divisor. How to prove this? We could consider the *smallest* integer $d > 1$ which is not prime and has no prime divisor. But then d must have some divisor d_0 with $1 < d_0 < d$, and d_0 can't have any prime divisor, because if it did, so would d .

Okay, if we don't like that argument, we can fall back on Proposition 2. Find x_1 and y_1 with $1 = x_1n_1 + y_1m$ and x_2 and y_2 with $1 =$

$x_2n_2 + y_2m$. Multiply out and regroup, to see that any d that divides both m and n_1n_2 also divides 1 and hence must equal 1.

For the converse, note that if, for example, n_1 and m have a common divisor $d > 1$, then n_1n_2 and m share the same common divisor. \square

Recall that the number of elements in G_m is given by Euler's ϕ -function, $\phi(m)$. Recall also that we already know (from preceding exercises) that if p is prime, then $\phi(p) = p - 1$, and if q is a different prime, then $\phi(pq) = (p - 1)(q - 1)$. Because of its importance for the RSA algorithm, and indeed in the theory of numbers in general, we want to note the following two facts, which will allow us to calculate $\phi(m)$ for all positive integers m .

Proposition 4. *If p is prime and r is a positive integer, then $\phi(p^r) = p^r - p^{r-1}$.*

Proof. Among the complete set of representatives $\{1, 2, \dots, p^r\}$ modulo p^r , the only numbers which are not relatively prime to p are the ones that are divisible by p (because p is prime). We can list these as

$$(22) \quad p, 2p, 3p, \dots, p^{r-1}p,$$

which shows that there are exactly p^{r-1} of them. \square

Theorem 1. *The Euler ϕ -function is multiplicative in the following sense: if m and n are positive integers such that $\gcd(m, n) = 1$, then $\phi(mn) = \phi(m)\phi(n)$.*

Proof. We give two proofs, or, perhaps, one proof with a visualization. Consider all congruence classes $[xm + yn]_{mn}$ for $x \in G_n$ and $y \in G_m$. There are $\phi(m)\phi(n)$ choices of the pair (x, y) . We want to show that they all give different congruence classes in \mathbb{Z}_{mn} , and that every congruence class relatively prime to mn comes up in this way.

First we check that all such $xm + yn$ are in fact relatively prime to mn . If p is a prime that divides mn , then p must also divide m or n . Let's suppose that $p|m$. If also $p|(xm + yn)$, then (taking the difference) $p|(yn)$. Now p cannot divide n , since m and n are relatively prime, and p is known already to divide m . Therefore $p|y$. But this is also impossible since y and m are relatively prime. A similar argument applies, of course, in case it is n and not m that p divides.

For the second point, suppose that $x_1m + y_1n \equiv_{mn} x_2m + y_2n$, with $1 = \gcd(x_1, n) = \gcd(x_2, n) = \gcd(y_1, m) = \gcd(y_2, m)$. Then

$$(23) \quad \begin{aligned} & x_1m + y_1n \equiv x_2m + y_2n \pmod{mn} \text{ implies that} \\ & (x_1 - x_2)m + (y_1 - y_2)n \equiv 0 \pmod{mn}, \text{ and hence} \\ & (x_1 - x_2)m = -(y_1 - y_2)n + kmn \text{ for some } k \in \mathbb{Z}. \end{aligned}$$

Thus $n|(x_1 - x_2)m$, and, since $\gcd(m, n) = 1$, by Corollary 1 we have $n|(x_1 - x_2)$. By symmetry of the notation and situation, $m|(y_1 - y_2)$.

These observations imply that $\phi(mn) \geq \phi(m)\phi(n)$. To prove the reverse inequality, we show now that every residue class \pmod{mn} that is relatively prime to mn arises as the class of some $xm + yn$ with x relatively prime to n and y relatively prime to m . For suppose that $k \in \mathbb{Z}$ and $\gcd(k, mn) = 1$. Since $\gcd(m, n) = 1$, by the Euclidean Algorithm (read backwards) there are integers r and s such that $1 = rm + sn$, and hence there are integers x and y such that

$$k = xm + yn.$$

Now y must be relatively prime to m , since otherwise this equation shows that k and m have a common prime factor, contradicting the fact that k and mn are relatively prime. Similarly, x is relatively prime to n . This concludes the first proof.

We are using here Corollary 2, according to which a number is relatively prime to a product mn if and only if it is relatively prime to each of the factors, m and n . This is also the basis of the following more visual proof of the fact that $\gcd(m, n) = 1$ implies $\phi(mn) = \phi(m)\phi(n)$.

Let us lay out the standard complete set of representatives modulo mn in n rows and m columns as follows:

$$\begin{array}{cccccc} 0 & 1 & 2 & \dots & m-1 \\ m & m+1 & m+2 & \dots & 2m-1 \\ 2m & 2m+1 & 2m+2 & \dots & 3m-1 \\ \vdots & & & & \\ (n-1)m & (n-1)m+1 & (n-1)m+2 & \dots & nm-1 \end{array}$$

Because of the foregoing observation, we are interested in the number of entries i in this array that are relatively prime to both m and n .

In each column, all the entries are congruent to one another modulo m . The columns correspond to the the elements of \mathbb{Z}_m , the congruence classes modulo m . There are $\phi(m)$ columns whose entries are all

relatively prime to m . We now restrict our attention to those $\phi(m)$ columns.

In any one of these columns relatively prime to m , how many entries are there that are relatively prime to n ? Let us note that each column contains n entries, no two of which are congruent modulo n : If $0 \leq j, k \leq n-1$ and $jm+r \equiv_n km+r$ for some r , then $n|(j-k)m$, and, since $\gcd(n, m) = 1$, $n|(j-k)$, so that $j = k$. Thus each column consists of a complete set of representatives modulo n and hence contains $\phi(n)$ elements relatively prime to n .

We conclude that in the array there are exactly $\phi(n)\phi(m)$ elements relatively prime to both m and n , and hence to mn . \square

Exercise 14. (a) Find $\phi(68)$.
(b) Find $\phi(7911)$.

We now advance our understanding of the $\phi(m)$ -element abelian group G_m of congruence classes relatively prime to m in preparation for proving the Euler-Fermat Theorem, which is the secret of success of the RSA algorithm.

Proposition 5. *If $\{r_1, r_2, \dots, r_{\phi(m)}\}$ is a complete set of representatives relatively prime to m modulo m , so that*

$$G_m = \{[r_1]_m, [r_2]_m, \dots, [r_{\phi(m)}]_m\},$$

and $\gcd(a, m) = 1$, then $\{ar_1, ar_2, \dots, ar_{\phi(m)}\}$ is also a complete set of representatives relatively prime to m modulo m .

Proof. We know that all the ar_i are relatively prime to m , because we have shown that G_m is closed under multiplication. Moreover, because G_m , being a group, has multiplicative inverses, all of these congruence classes are distinct:

$$(24) \quad ar_i \equiv_m ar_j \text{ implies } r_i \equiv_m r_j$$

(upon multiplying by the multiplicative inverse a^{-1} of a in G_m). Since we have here $\phi(m)$ distinct congruence classes modulo m relatively prime to m , we must be looking at all of G_m , each element represented exactly once. \square

Euler-Fermat Theorem. *If m is a positive integer and a is an integer with $\gcd(a, m) = 1$, then*

$$(25) \quad a^{\phi(m)} \equiv 1 \pmod{m}.$$

Proof. Let $\{r_1, r_2, \dots, r_{\phi(m)}\}$ be a complete set of representatives for the group G_m of congruence classes modulo m relatively prime to m . Because of the preceding Proposition,

$$(26) \quad (ar_1)(ar_2) \cdots (ar_{\phi(m)}) \equiv_m r_1 r_2 \cdots r_{\phi(m)},$$

since both products contain exactly the same factors, and their order does not affect the value of the product, the group G_m being commutative. If we denote the righthand side by R (an element of the group G_m), this equation says

$$(27) \quad a^{\phi(m)} R \equiv_m R,$$

and multiplying by $R^{-1} \in G_m$ gives

$$(28) \quad a^{\phi(m)} \equiv_m 1.$$

□

Fermat's Little Theorem. *If p is prime and a is an integer not divisible by p , then*

$$(29) \quad a^{p-1} \equiv 1 \pmod{p}.$$

Corollary 3. *If p is prime and $a \in \mathbb{Z}$, then*

$$(30) \quad a^p \equiv a \pmod{p}.$$

Proof. If $p|a$, then both sides are 0 modulo p . □

Remark 2. If G is any finite commutative ($ab = ba$ for all $a, b \in G$) group, let I denote the identity element of G and $|G|$ the number of elements in G . Then

$$(31) \quad a^{|G|} = I \quad \text{for all } a \in G.$$

The proof is the same as for Proposition 5 and the Euler-Fermat Theorem.

6. VERIFICATION OF THE FUNCTIONING OF THE RSA ALGORITHM

1. The recipient's public key consists of the product n of two distinct primes p and q (which are kept secret) and an encoding exponent e which is relatively prime to $m = \phi(n) = (p-1)(q-1)$. A message (or piece of a message) is a congruence class $M \in \mathbb{Z}_n$. We deal first with the case when M is relatively prime to n . (This covers $(p-1)(q-1)$ of the pq congruence classes modulo n .) The encoded message is

$$(32) \quad R = M^e \pmod{n}.$$

The recipient knows p and q , and hence also m , and so can find the secret decoding exponent d , which is the multiplicative inverse of e modulo m . There is $k \in \mathbb{Z}$ such that $de = km + 1$, and so, if $\gcd(M, n) = 1$,

$$(33) \quad R^d \equiv_n M^{ed} = M^{km+1} = M(M^m)^k = M(M^{\phi(n)})^k \equiv_n M(1^k) \equiv_n M,$$

by the Euler-Fermat Theorem. The decoding is accomplished correctly!

2. What if $\gcd(M, n) \neq 1$? For this, as well as the improved RSA algorithm, we need an ancient and useful number theory result.

Chinese Remainder Theorem. *Suppose that m_1, m_2, \dots, m_r are pairwise relatively prime integers, all at least 2.*

(1) *Given $d_1, \dots, d_r \in \mathbb{Z}$, the system of congruences*

$$(34) \quad x \equiv d_i \pmod{m_i}, i = 1, \dots, r$$

has an integer solution x with $0 \leq x < m = m_1 m_2 \cdots m_r$.

(2) *The solution is unique modulo $m = m_1 m_2 \cdots m_r$: if $x \equiv y \pmod{m_i}$ for each $i = 1, \dots, r$. Then $x \equiv y \pmod{m = m_1 m_2 \cdots m_r}$.*

Proof. The proof of part (1) is Exercise 15. Statement (2) is credible in view of prime factorization, but we can indicate also a proof based on tools we have been using before. We have

$$(35) \quad x - y = k_1 m_1 \text{ for some } k_1 \in \mathbb{Z}.$$

Since $m_2 | (x - y)$ and $\gcd(m_2, m_1) = 1$, necessarily $m_2 | k_1$, and so we may write

$$(36) \quad x - y = k_2 m_2 m_1 \text{ for some } k_2 \in \mathbb{Z}.$$

Now m_3 divides $x - y$ but is relatively prime to $m_2 m_1$ (recall Corollary 2, which was key to proving that ϕ is multiplicative), so $m_3 | k_2$, and so

$$(37) \quad x - y = k_3 m_3 m_2 m_1 \text{ for some } k_3 \in \mathbb{Z}.$$

Continuing in this way (or, in an explicitly more rigorous proof, applying the Axiom of Mathematical Induction), we arrive at the asserted conclusion. \square

Exercise 15. Prove part (1) of the Chinese Remainder Theorem. (*Hint:* For each $i = 1, \dots, r$ let $u_i = m/m_i$ and let v_i be the multiplicative inverse of u_i modulo m_i (explain why this exists). Then try $x = d_1 u_1 v_1 + \cdots + d_r u_r v_r$.)

Now let's see that RSA decoding works even if $\gcd(M, n) \neq 1$. First, if $\gcd(M, p) = 1$, then $M^{p-1} \equiv_p 1$, by Fermat's Little Theorem, so that

$$(38) \quad \begin{aligned} M^m &\equiv_p (M^{p-1})^{q-1} \equiv_p 1^{q-1} = 1, \text{ and hence} \\ M^{ed} = M^{km+1} &= (M^m)^k M \equiv_p 1^k M \equiv_p M. \end{aligned}$$

But this latter congruence holds even if $\gcd(M, p) \neq 1$, since then both sides are 0 modulo p . Similarly,

$$(39) \quad M^{ed} \equiv_q M.$$

Since p and q are distinct primes, the Chinese Remainder Theorem applies, and we conclude that

$$(40) \quad M^{ed} \equiv_n M.$$

Exercise 16. In Example 4, the message $M = 15 \equiv 0 \pmod{p}$, since $p = 5$. Yet the decoding gives us back 15, not 0. Sort through the steps of this calculation in light of the Chinese Remainder Theorem to explain exactly how the decoding works.

3. Finally, we will check that decoding works in the modified RSA algorithm, in which $m = \phi(n) = (p-1)(q-1)$ is replaced by $m = \text{lcm}(p-1, q-1)$. The decoding exponent d is still the multiplicative inverse of the encoding exponent e modulo this new (and probably greatly reduced) m , so we still have $ed = km + 1$ for some $k \in \mathbb{Z}$. Because m is a multiple of $p-1$, there is $j_1 \in \mathbb{Z}$ such that $m = j_1(p-1)$. If $\gcd(M, p) = 1$, then

$$(41) \quad M^{ed} = M^{km+1} = M(M^{p-1})^{j_1 k} \equiv_p M(1^{j_1 k}) \equiv_p M,$$

by Fermat's Little Theorem. As before, the congruence

$$(42) \quad M^{ed} \equiv_p M$$

holds true whether or not $\gcd(M, p) = 1$. Similarly,

$$(43) \quad M^{ed} \equiv_q M,$$

and then

$$(44) \quad M^{ed} \equiv_n M$$

follows from the Chinese Remainder Theorem.

7. A FEW APPLICATIONS

7.1. Digital signature. In electronic communication there is inherent anonymity: we cannot see or directly hear the people with whom we are exchanging information. When sensitive information is involved, or commands are sent, or money is moved around, it is important to know that the person on the other end of the communication line has the necessary authority. This is the problem of *authentication*. For 2002 tax returns submitted electronically, the IRS used an electronic signature consisting of birthdate and a self-selected 5-digit identification number. An asymmetric cryptographic system, such as the RSA system, allows for a clever and much surer form of authentication.

The sender who wishes to adduce proof of his identity prepares a message S consisting of his name, plus the date and time of transmission. (This will prevent illicit reuse of the signature by anyone who might intercept it.) He then encodes this message using his personal secret *decoding exponent*, d :

$$R = S^d \pmod{n}.$$

(The RSA encoding setup involving $p, q, n = pq, d$, and e is as before.) This signature R can be sent as an addendum to any other message, M . The combined message and signature can be encoded using a recipient's public key and then sent to that recipient.

When the recipient applies his private decoding key to what he has received, he obtains MR —a legible message M followed by a random-looking stream of bits, R . Now the receiver applies the sender's *public encoding key*, the exponent e , to R , obtaining

$$R^e \equiv_n S^{de} \equiv_n S^{k\phi(m)+1} \equiv_n S,$$

the original, legible, signature S .

The recipient is sure that only someone with knowledge of the sender's secret decoding exponent d could have encoded S to R so as to produce this result and is therefore convinced that the sender is indeed who he says he is.

7.2. Diffie-Hellman key exchange. The problem of key exchange has long been one of the basic issues in cryptology. How can two parties wishing to communicate agree securely and conveniently on a key that will be used to encode (and decode) messages between them? In practice, keys must be changed often in order to foil cryptanalysis.

Weakness in key selection played a major role in the Polish and British attacks on the German Enigma cipher in World War II.

For our purposes, let us say that the problem is for two people A and B to agree on an element k of \mathbb{Z}_m . A clever way to use one-way functions to accomplish this task, even while communicating over an insecure channel, was discovered by Whitfield Diffie and Martin Hellman, predating the discovery of the RSA system. It is based on the presumed one-way function, for a fixed base $r \in \mathbb{Z}_m$,

$$(45) \quad f(x) = r^x \pmod{m}.$$

The two partners A and B each pick elements of \mathbb{Z}_m , say A picks a and B picks b . Then

A sends $f(a) = \alpha$ to B, and

B sends $f(b) = \beta$ to A.

The partners do not care if these communications are intercepted. Upon receipt,

A calculates $k = \beta^a \pmod{m} = r^{ba} \pmod{m}$, and

B calculates $k = \alpha^b \pmod{m} = r^{ab} \pmod{m}$.

Because $ab = ba$, A and B arrive at the same key, k . Any interceptor of the communications would have trouble determining a or b and hence, it is presumed, k , from $\alpha = f(a)$ and $\beta = f(b)$.

7.3. Digital coin flipping. How can two people on opposite coasts accomplish a fair coin flip over the telephone? A classic example involves a divorcing couple trying to decide who gets the car. The spouse in California flips a coin, the spouse in New York calls “Tails!”, and the spouse in California says, “Sorry, you lose!”. The New York spouse might not be convinced that the process was fair.

Maybe each of the two people, A and B, could choose either 0 or 1, with the understanding that A wins if their choices agree, while B wins if the choices disagree. The choices could be transmitted to a third party, who would then announce the result. But the involvement of a third party would lead to complication and delay and would also raise the problem of trustworthiness of that arbiter, who could perhaps be bribed or otherwise influenced.

Maybe A and B could simultaneously send their choices to each other. But achieving true simultaneity and preventing cheating present their own problems.

A long-distance fair coin flip can be accomplished by using a one-way function f (say from \mathbb{Z} to \mathbb{Z} or from \mathbb{Z}_m to \mathbb{Z}_m for some m) which has the property that given $f(x)$ it is not only essentially impossible to determine x but also even the *parity* of x , that is, $x \bmod 2$. Particular such functions can be constructed using number theory—see [1, p. 52]. Here is one procedure to accomplish a fair long-distance coin flip:

A chooses $a \in \{0, 1, \dots, m-1\}$, B chooses $b \in \{0, 1, \dots, m-1\}$.

A sends $f(a)$ to B, B sends b to A.

Having received b , A can compute $a + b \bmod 2$ and determine the winner.

A sends a to B. Now B can also compute $a + b \bmod 2$ and determine the winner.

B can plug the received a into f and verify that it does indeed produce the value $f(a)$ previously transmitted by A, showing that A did not change the value of a after receiving b , and thus the process was fair.

REFERENCES

1. Gilles Brassard, *Modern Cryptology: A Tutorial*, Lecture Notes in Computer Science, vol. 325, Springer-Verlag, 1988.
2. Simon Singh, *The Codebook*, Anchor Books, 1999.

DEPARTMENT OF MATHEMATICS, CB 3250, PHILLIPS HALL, CB 3250, PHILLIPS HALL, UNIVERSITY OF NORTH CAROLINA, CHAPEL HILL, NC 27599 USA

E-mail address: `petersen@math.unc.edu`