Number theory

Harald Hanche-Olsen

http://www.math.ntnu.no/~hanche/

Congruences, or modular arithmetic

Arithmetic modulo 12 or 24 is familiar to anyone using a clock, though not usually under that name. The general notion of congruence replaces 12 or 24 by a positive integer n:

Two integers a and b are called *congruent* modulo n if their difference is a multiple (by which we mean an *integer* multiple) of n – that is, $b-a=kn$ for some integer k. When this is the case, we write

$$a \equiv b \pmod{n},$$

though we commonly drop the part $\pmod{n}$ when the *modulus* n is clear from context.

Congruence modulo n "behaves like equality" in the sense that

1. $a \equiv a$ for all a,
2. if $a \equiv b$ then $b \equiv a$,
3. if $a \equiv b$ and $b \equiv c$ then $a \equiv c$.

Any binary relation satisfying the above requirements is called an *equivalence relation*.

Of fundamental importance is the fact that addition and multiplication "respect" congruence modulo n, in the sense that if $a \equiv b$ and $c \equiv d$ then $a+c \equiv b+d$ and $ac \equiv bd$. Thus, in any arithmetic expression involving integers, if any part is replaced by a different number that is congruent to the original modulo n, then the result is still congruent modulo n.

This lies behind a classic trick for checking the accuracy of multiplication: Calculations modulo 9. Every positive integer is congruent modulo 9 to the sum of its decimal digits, because $10 \equiv 1 \pmod{9}$, from which we get $10^k \equiv 1$ for every positive integer k, and so, for example, $831 = 8\cdot 10^2 + 3\cdot 10 + 1 \equiv 8+3+1 = 12$. Repeating the trick until we're left with just one digit, we conclude $831 \equiv 3 \pmod{9}$. Thus you can quickly check for errors in the calculation $831\cdot 42 = 34902$ by repeating it modulo 9: If the result is correct you should also have $3\cdot 6 \equiv 3+4+9+0+2 \equiv 0$, which is indeed the case. (This does not, of course, *prove* that there is no mistake. On average, this procedure ought to catch eight out of nine mistakes.)

Remainders and canonical representatives. The integer division algorithm for computing a/n as we learned it in school results in a quotient q and a remainder r, so that

$$a = qn + r, \quad 0 \le r < n.$$

You may be used to writing the answer as $a/n = q + r/n$, but for our purposes, it is better left in the above form. We call q the *quotient* and r the *remainder* of the division. More precisely, we say r is the remainder of a modulo n, and write this in the form

$$r = a \bmod n.$$

We have introduced similar notations which should not be confused: $a \bmod n$ is a *number*, and the above equation says r equals this number. On the other hand, $r \equiv a \pmod{n}$ is a *relation* between numbers r and a. (Some texts write that relation without the parentheses, as $r \equiv a \mod n$. Note the subtle difference in spacing, though.) It is certainly true that if $r = a \bmod n$ then $r \equiv a \pmod{n}$, but the converse is not true: Certainly $17 \equiv 22 \pmod{5}$, but $22 \bmod 5 = 2 \neq 17$.

Despite the above warning, what is true is that for every integer a (and every integer $n > 0$) there is a *unique* integer r with $0 \le r < n$ and $r \equiv a \pmod{n}$. We have already seen that $r = a \bmod n$ is one such. Assuming that r' is another, we must have $r \equiv r' \pmod{n}$. Thus $r - r'$ is a multiple of n. But the only multiple of n in $\{0, 1, \ldots, n-1\}$ is 0, so $r = r'$.

Thus, among all integers congruent to a modulo n there is one that stands out from the crowd, namely $a \bmod n$. In a sense this *represents* the set of integers congruent to a.

We introduce the notation

$$\mathbb{Z}_n = \{0, 1, 2, \ldots, n-1\}$$

for all the "canonical" representatives of integers modulo n. Also, we write $\mathbb{Z}$ for the set of all integers (positive, zero, and negative).

A more abstract approach is to introduce *equivalence classes*: The equivalence class (modulo n) of an integer a is the set of all integers congruent with a modulo n. We could write it

$$[a] = \{m \in \mathbb{Z} \colon m \equiv a \pmod{n}\} = \{a + kn \colon k \in \mathbb{Z}\} = \{\ldots, a-2n, a-n, a, a+n, a+2n, \ldots\}$$

and then define addition and multiplication on equivalence classes by $[a]+[b] = [a+b]$ and $[a][b] = [ab]$. The resulting algebraic structure (set of equivalence classes with addition and multiplication) is commonly written $\mathbb{Z}_n$. This conflict of notations should be only mildly confusing, since as we have seen, each equivalence class has exactly one member in $\{0, 1, \ldots, n-1\}$.

Divisibility, divisors, factors

If a and b are integers, we say that *a divides b* and write $a \mid b$ if there is an integer q so that $b = aq$. We also say that a is a *divisor* of b, or that it is a *factor* of b. (Such an important and classical concept, there are lots of names for it.)

The following elementary properties of this relation almost characterize it as a partial order:

1. $a \mid a$ for all a,
2. if $a \mid b$ and $b \mid a$ then $a = \pm b$,
3. if $a \mid b$ and $b \mid c$ then $a \mid c$.

If we restrict our attention to positive numbers, the second property will conclude $a = b$, and we really do have a partial order. But the "divides" relation does not distinguish between a number and its negative, so this fails in general.

The number zero behaves a bit oddly under this relation: $a \mid 0$ for all a, but $0 \nmid b$ for all $b \neq 0$ (the symbol $\nmid$ is used for "does not divide").

On the other hand, the number one is a universal divisor: $1 \mid b$ for all b.

A *common divisor* for two nonzero integers a and b is a number c which divides both: $c \mid a$ and $c \mid b$. A *greatest common divisor* is a common divisor $d > 0$ so that every common divisor divides d. The greatest common divisor is clearly unique, if it exists (and we shall see that it does), and we write it $\gcd(a, b)$.

The following simple fact is frequently useful: If c is a common divisor of a and b then $c \mid (ax + by)$ for any integers x and y. Clearly, this fact is just as true with more summands, or if each summand has more factors.

Lemma 1 (Bézout)**.** *Any two nonzero integers a and b have a greatest common divisor, and there exist integers x and y so that*

$$ax + by = \gcd(a, b).$$

Proof. I give here an *abstract* proof, in the sense that it does not provide a practical way to compute the greatest common divisor or to find x and y. We will return to that.

Define the set of integers

$$I = \{ax + by\colon\ x, y \in \mathbb{Z}\}$$

($\mathbb{Z}$ is the set of all integers). It has these simple properties:

1. If $u, v \in I$ then $u + v \in I$,
2. if $u \in I$ and $v \in \mathbb{Z}$ then $uv \in I$.

Such a set is called an *ideal*, but we are not going into the theory of ideals.
Obviously (select $(x, y) = (1, 0)$ or $(x, y) = (0, 1)$) $a \in I$ and $b \in I$, and every common divisor of a and b is also a divisor of every member of I.

Let d be the smallest positive member of I. I claim that

$$I = \{qd\colon\ q \in \mathbb{Z}\}. \tag{1}$$

That $qd \in I$ whenever $q \in \mathbb{Z}$ follows from the second of the above properties. Conversely, if $u \in I$, divide u by d, getting the quotient q and the remainder r:

$$u = qd + r, \quad 0 \leq r < d.$$

Then the properties of I show that $r = u - qd \in I$, but since d is the smallest positive member of I, r cannot then be positive, so we must have $r = 0$. Thus $u = qd$, and (1) is proved.

Equation (1) states that d divides every member of I. Since $a, b \in I$, it follows that d is a common divisor of a and b. Also, since $d \in I$, every common divisor of a and b divides d. Thus d is the greatest common divisor of a and b, and being a member of I, it does have the form $d = ax + by$ for suitable integers x and y. □

Bézout's lemma has two useful corollaries. (A corollary is an immediate consequence of a previous result.)

Corollary 2. *Let a and $n > 0$ be integers. Then a has an inverse modulo n if and only if* $\gcd(a, n) = 1$.

Proof. First, assume that a has an inverse b modulo n. By definition, this means that $ab \equiv 1 \pmod{n}$, so that $ab + kn = 1$ for some integer k. Therefore any common divisor for a and n is also a divisor of 1, and so $\gcd(a, n) = 1$.

Conversely, assuming $\gcd(a, n) = 1$, there are integers x and y such that $ax + ny = 1$. But then $ax \equiv 1 \pmod{n}$, and we are done. □

The above proof is useful, as it shows how to find an inverse modulo n by solving the equation $ax + ny = 1$. Two integers a and n are called *mutually prime*, or *coprime*, if $\gcd(a, n) = 1$. The same is said of three or more integers if they are pairwise mutually prime.

The second corollary will be useful in proving uniqueness of prime factorization.

Corollary 3. *Let p be a prime number and a, b two nonzero integers. If $p \mid ab$ then $p \mid a$ or $p \mid b$.*

Proof. Assume that $p \mid ab$ and $p \nmid a$. Since p has no divisors other than 1 and p (and their negatives) and $p \nmid a$, we must have $\gcd(a, p) = 1$. Thus we can write $ax + py = 1$ for some integers x and y. Multiplying by b we get $abx + pby = b$. Then p divides each term on the left, since $p \mid ab$, and so $p \mid b$. □

Primes and unique factorization

A *prime number* is defined to be an integer > 1 which is not divisible by any positive integer other than 1 and itself.

Notice that this definition is explicitly written to exclude 1. If we allowed 1 to be prime, then unique factorization would fail, as for example $2 = 1 \cdot 2 = 1 \cdot 1 \cdot 2$ and so forth, resulting in an infinite number of factorizations of the number 2.

Theorem 4 (Unique factorization)**.** *Every positive integer is the product of primes. Moreover, this factorization is unique, in that if two different products of primes produce the same integer then the two products involve the same primes the same number of times.*

To clarify what is meant by this, we consider the two prime factorizations

$$2 \cdot 2 \cdot 3 \cdot 3 \cdot 3 \cdot 5 \text{ and } 2 \cdot 5 \cdot 3 \cdot 2 \cdot 3 \cdot 3$$

to be the same factorization (we prefer to write it as $2^2 3^3 5^1$). Obviously, since the order of factors is irrelevant, they produce the same integer. The theorem states that this sort of permutation is the only reason two products of primes can be equal, so we can immediately tell that

$$2^2 3^3 5^1 \neq 2^3 3^2 7^1$$

without doing any calculation at all (except to verify that 2, 3, 5 and 7 are primes).

We should also add that 1 is not an exception to the theorem: By convention, we consider 1 to be a product of no primes. (And the empty set is of course a set of primes, right?)

Proof. First, if there exists a positive number with no prime factorization, there is a smallest such number. Call it a. Then a is not a prime number, for then it is simply the product of itself (taken once). Thus a has a divisor $b > 1$ different from a, so we can write $a = bc$ for positive integers b and c. But then both b and c are less than a, so each is a product of primes. Thus a is a product of primes, which is a contradiction.

Second, if two products of primes produce the same product, we can cancel out common prime factors on both sides until we end with an equality of the form

$$p_1 \cdots p_m = q_1 \cdots q_n$$

where the p_i and q_j are primes (possibly with repetitions) and $p_i \neq q_j$ for all i, j.

But then $p_1 \mid q_1 \cdots q_n$, and so by Corollary 3 (applied repeatedly if needed) $p_1 \mid q_j$ for some j. But since p_1 is neither 1 nor q_j, and q_j is prime, this is a contradiction. □

Euclid's algorithm

We know that the greatest common divisor exists, but not yet how to compute it. Clearly, computing $ax + by$ for an infinite number of (x, y) and then picking the smallest positive answer is not practical.

The idea behind Euclid's algorithm is simple: Assume we wish to compute $\gcd(a, b)$ where $a > b > 0$. Divide a by b to get a quotien q and remainder r:

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

The crucial observation is that any common divisor of a and b divides r as well, and any common divisor of b and r divides a. Therefore $\gcd(a, b) = \gcd(b, r)$. Since $a > b > r$, replacing the problem of computing $\gcd(a, b)$ by the problem of computing $\gcd(b, r)$ is a simplification, in that we are now dealing with smaller numbers.

The following example shows this idea carried out in a systematic manner to compute $\gcd(2328, 2124)$. The left column shows the result of successive divisions, while the right column shows what we conclude about the gcd's from the results on the left.

$$\begin{aligned}
2328 &= 1 \cdot 2124 + 204 & \gcd(2328, 2124) &= \gcd(2124, 204) \\
2124 &= 10 \cdot 204 + 84 & &= \gcd(204, 84) \\
204 &= 2 \cdot 84 + 36 & &= \gcd(84, 36) \\
84 &= 2 \cdot 36 + 12 & &= \gcd(36, 12) \\
36 &= 4 \cdot 12 & &= 12
\end{aligned}$$

Thus *Euclid's algorithm* for computing $\gcd(a, b)$ where $a > b > 0$ is as follows:

1. Write $a = qb + r$ with $0 \leq r < b$.
2. If $r = 0$, $\gcd(a, b) = b$ and we are done.
3. Otherwise, compute $\gcd(b, r)$ and return the result.
   In other words, replace (a, b) by (b, r) and start over.

Note that since the second number becomes progressively smaller (b gets replaced by r, which is smaller, and so forth) this algorithm cannot continue forever, so it will terminate. In fact, it is quite fast.

Notice how each remainder, after first showing up at the right, moves to the middle and then the left in the left column above. Looking down the left side of the equations, it becomes clear that all lines in the calculation have the same form if we define a sequence $(r_0, r_1, r_2, \ldots)$ with $r_0 = a$, $r_1 = b$, and $r_s, \ldots$ being the suc-

cessive remainders. We can then write the above calculation abstractly in the form

$$\begin{aligned} r_0 &= q_2 r_1 + r_2 & \gcd(r_0, r_1) &= \gcd(r_1, r_2) \\ r_1 &= q_3 r_2 + r_3 & &= \gcd(r_2, r_3) \\ r_2 &= q_4 r_3 + r_4 & &= \gcd(r_3, r_4) \\ &\cdots & &\cdots \end{aligned}$$

We now turn to the problem of computing (x, y) so that $ax + by = \gcd(x, y)$. To accomplish this, compute more generally (x_i, y_i) for $i = 0, 1, 2, \ldots$ so that

$$ax_i + by_i = r_i.$$

Thanks to the special definitions $r_0 = a$, $r_1 = b$ we start out with $(x_0, y_0) = (1, 0)$ and $(x_1, y_1) = (0, 1)$. Then, for $i = 0, 1, \ldots$ we use the relation $r_{i+2} = r_i - q_{i+2} r_{i+1}$ and see that

$$(x_{i+2}, y_{i+2}) = (x_i, y_i) - q_{i+2}(x_{i+1}, y_{i+1})$$

does what we need. This is easy enough to program, but if we wish to carry out the computations by hand, it can be difficult to keep track. Here is one possible solution to this problem, first in the abstract formulation,

$$\begin{aligned} & & r_0 &= 1 \cdot a + 0 \cdot b \\ & & r_1 &= 0 \cdot a + 1 \cdot b \\ r_0 &= q_2 r_1 + r_2 & r_2 &= x_2 a + y_2 b \\ r_1 &= q_3 r_2 + r_3 & r_3 &= x_3 a + y_3 b \\ r_2 &= q_4 r_3 + r_4 & r_4 &= x_4 a + y_4 b \\ &\cdots & &\cdots \end{aligned}$$

where each equation (starting at the third row) in the right column follows from the equation to its left and the two equations immediately above it. For example, in the last row we find

$$r_4 = r_2 - q_4 r_3 = x_2 a + y_2 b - q_4(x_3 a + y_3 b) = (x_2 - q_4 x_3)a + (y_2 - q_4 y_3)b,$$

so we compute $x_4 = x_2 - q_4 x_3$ and $y_4 = y_2 - q_4 y_3$ and fill into the last row.

Again, for our concrete example:

$$\begin{aligned} & & 2328 &= 1a + 0b \\ & & 2124 &= 0a + 1b \\ 2328 &= 1 \cdot 2124 + 204 & 204 &= 1a - 1b \\ 2124 &= 10 \cdot 204 + 84 & 84 &= -10a + 11b \\ 204 &= 2 \cdot 84 + 36 & 36 &= 21a - 23b \\ 84 &= 2 \cdot 36 + 12 & 12 &= -52a + 57b \\ 36 &= 4 \cdot 12 \end{aligned}$$

Modular exponentiation

A much needed operation in several public-key cryptosystems is that of computing $a^x \bmod n$, where the exponent x and the modulus n are both large integers (typically a few hundred digits). Computing a^x as a natural number and then reducing modulo n will be utterly impractical, as a^x will easily end up (much) larger than $10^{10^{100}}$, and as such will not fit inside the computer.

Instead, we might use the algorithm of *repeated squaring* and reduction modulo n. Here, we illustrate the procedure on the toy example $5^{19} \bmod 11$. We start with 5 and square it again and again, producing powers 5^2, 5^4, 5^8, 5^{16} (where we stop, since the next exponent, 32, is greater than 19). All congruences below are modulo 11:

$$\begin{aligned} 5^1 &= 5 \\ 5^2 &= 25 \equiv 3 \\ 5^4 &= (5^2)^2 \equiv 3^2 = 9 \\ 5^8 &= (5^4)^2 \equiv 9^2 = 81 \equiv 4 \\ 5^{16} &= (5^8)^2 \equiv 4^2 = 16 \equiv 5 \end{aligned}$$

from which we finally conclude

$$5^{19} = 5^{16+2+1} = 5^{16} \cdot 5^2 \cdot 5^1 \equiv 5 \cdot 3 \cdot 5 = 15 \cdot 5 \equiv 4 \cdot 5 = 20 \equiv 9 \pmod{11}.$$

You may recognize this as using the binary (base 2) representation

$$19 = 10011_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$$

(where the subscript 2 is used to indicate the base) together with the identity

$$a^{2^{k+1}} = \left(a^{2^k}\right)^2$$

which justifies the repeated squaring. In general, the technique is to write the exponent as

$$x = \sum_{k=0}^{n} b_k 2^k \qquad \text{with each } b_k \in \{0, 1\}$$

and noting that then

$$a^x = \prod_{k=0}^{n} a^{b_k 2^k}.$$

The effect of b_k is that when $b_k = 0$, the corresponding term in the product is 1, so we can skip it, and when $b_k = 1$, we get a^{2^k}, and those terms are computed by repeated squaring, all the time reducing the answer modulo n since we only want the final answer modulo n.

If you use some sort of bignum package with your favourite computer language (or if the language already has built-in support for big integers), it is worth looking for a function that takes all three arguments a, x and n and computes $a^x \bmod n$. If the package is any good, it will use an algorithm much like the above. If you don't find such a function, write it yourself. As I mentioned, computing a^x and then reducing modulo n will simply not work if the numbers are large.

A further optimization could use the Chinese Remainder Theorem (CRT) if a factorization of n is known: If $n = uv$ with u, v mutually prime, compute $a^x \bmod u$ and $a^x \bmod v$ and use the CRT to deduce the value of $a^x \bmod uv$. This trick is often employed in the RSA cryptosystem, where the factorization of the modulus is known to the owner of the private key. (Users of the public key will not possess this information, but they will typically only need to compute powers with smallish exponents, such as $2^{16}+1$.)

Diffie–Hellman key exchange. Diffie–Hellman key exchange, or D–H for short, is a method for creating shared secrets while communicating over open channels. Participants in a D–H scheme need to agree in advance on a large prime p and a *generator* g. One wants the generator to be such that the every member of the set

$$\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$$

is congruent modulo p to some power g^x. To participate, Alice selects a large secret integer a. She computes $A = g^a \bmod p$ and publishes A. Similarly, Bob selects a secret b and publishes $B = g^b \bmod p$. Now Alice and Bob have a shared secret, namely $g^{ab} \bmod p$. Both of them can compute this, since

$$g^{ab} \equiv B^a \equiv A^b \pmod{p}$$

and Alice has all the information to compute B^a while Bob can similarly compute A^b.

Clearly, any outsider who is able to guess either a or b can also perform one of these calculations. Finding x given the value of g^x is called a *discrete logarithm problem* (DLP), and is believed to be infeasible in general.

But note that it is not known that solving the DLP is necessary to crack D–H. Even assuming access to a *Diffie–Hellman oracle*, meaning an entity that will tell you the value of $g^{ab} \bmod p$ given $g^a \bmod p$ and $g^b \bmod p$, this is not known to provide enough information to solve the DLP.

The ElGamal cryptosystem. Alice wants everybody to be able to encrypt (relatively short) secret messages to her. She picks a large prime p, a generator g modulo p, a secret number a, and she publishes p, g, and $A = g^a$.

If Bob wishes to send Alice a secret message, he encodes it as a number $m \in \mathbb{Z}_p^*$. He then picks a random integer r and computes

$$(R, T) = (g^r, A^r m) \bmod p$$

and transmits the pair to Alice.

To decrypt this message, Alice computes

$$m = R^{-a} T \bmod p.$$

It is fairly easy (in fact, this is a good exercise) to show that breaking ElGamal is equivalent to breaking Diffie–Hellman, in the sense that access to a Diffie–Hellman oracle makes it trivial to find the message m given A, R, and T, access to an ElGamal oracle makes it easy to break Diffie–Hellman.

The Chinese Remainder Theorem

This theorem, which we shall refer to as the CRT, can be viewed as the statement that arithmetic modulo uv can be reduced to simultaneous arithmetic mod u and mod v, *provided* that u and v are mutually prime.

First, notice that if we know the remainder of x modulo uv then we know the remainder of x modulo u as well. To be more precise, if $x \equiv y \pmod{uv}$ then $x \equiv y \pmod{u}$, and $x \equiv y \pmod{v}$ as well. For then $x - y$ is a multiple of uv, and therefore a multiple of both u and v.

Thus there is a natural mapping $\mathbb{Z}_{uv} \to \mathbb{Z}_u \times \mathbb{Z}_v$, given by

$$x \mapsto (x \bmod u, x \bmod v).$$

The CRT states that this mapping is a one-to-one mapping from $\mathbb{Z}_{uv}$ to $\mathbb{Z}_u \times \mathbb{Z}_v$, if $\gcd(u, v) = 1$. However, it is not usually stated in such abstract language:

Theorem 5 (Chinese Remainder Theorem)**.** *Assume u and v are positive, mutually prime integers, i.e.,* $\gcd(u, v) = 1$. *Then, for any two integers a and b there is an integer x solving the simultaneous congruences*

$$x \equiv a \pmod{u}, \quad x \equiv b \pmod{v}.$$

Moreover x is unique modulo uv in the sense that if y is another solution then $x \equiv y \pmod{uv}$.

Proof. It is useful to begin with the uniqueness part. Since $\gcd(u, v) = 1$ there exist integers s and t so that

$$us + vt = 1.$$

Now assume that x solves the simultaneous congruences. Multiplying the first by v and the second by u, we find

$$xv \equiv av \pmod{uv}, \quad xu \equiv bu \pmod{uv},$$

and therefore (using $us + vt = 1$)

$$x = xus + xvt \equiv bus + avt \pmod{uv}.$$

So any solution x is congruent to $bus+avt$ modulo uv, which proves the uniqueness part. Incidentally, it also provides an *algorithm* for finding the solution.

Thus we know what to do for the existence part: We simply put $x = bus + avt$. The relation $us + vt = 1$ implies $vt \equiv 1 \pmod{u}$, and so we find

$$x \equiv avt \equiv a \pmod{u}.$$

The congruence $x \equiv b \pmod{v}$ is proved the same way. □

Fermat's Little Theorem and Euler's φ function

We begin by stating Fermat's Little Theorem (FLT) without proof, as it is a special case of Euler's theorem (to be proved below).

Theorem 6 (Fermat's little theorem)**.** *If p is a prime number and a an integer so that $p \nmid a$, then*

$$a^{p-1} \equiv 1 \pmod{p}.$$

As an example, we compute 2^{2010} mod 101. Noting that 101 is a prime, so FLT says that $a^{100} \equiv 1 \pmod{101}$ whenever $101 \nmid a$. In particular, $2^{2010} = 2^{2000} \cdot 2^{10} = (2^{20})^{100} \cdot 2^{10} \equiv 1 \cdot 1024 = 1010 + 14 \equiv 14 \pmod{101}$.

For any positive integer n, recall $\mathbb{Z}_n = \{1, 2, \ldots, n-1\}$ and let us define

$$\mathbb{Z}_n^* = \{x \in \mathbb{Z}_n \colon x \text{ is invertible modulo } n\} = \{x \in \mathbb{Z}_n \colon \gcd(x, n) = 1\}.$$

Euler's φ function simply counts the elements of $\mathbb{Z}_n^*$, so that the set $\mathbb{Z}_n^*$ has $\varphi(n)$ members by definition. In particular, if p is a prime number then

$$\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\} \text{ and therefore } \varphi(p) = p - 1.$$

We can now state

Theorem 7 (Euler's theorem)**.** *If n is a positive integer and* $\gcd(a, n) = 1$ *then*

$$a^{\varphi(n)} \equiv 1 \pmod{n}.$$

It should be clear from the above that Fermat's Little Theorem is in fact a special case of Euler's theorem.

Proof. Write a^{-1} for some inverse of a modulo n. It is clear that if $x \in \mathbb{Z}_n^*$ then ax is invertible modulo n, since an inverse is given by $a^{-1}x^{-1}$. Conversely, if y is invertible modulo n then $y = ax$ for some invertible x, namely $x = a^{-1}y$. We see that x determines y uniquely, and it follows that

$$\{ax \bmod n \colon x \in \mathbb{Z}_n^*\} = \mathbb{Z}_n^*$$

and so

$$\prod_{x \in \mathbb{Z}_n^*} x = \prod_{x \in \mathbb{Z}_n^*} (ax \bmod n)$$

because both products multiply together the same numbers. There are precisely $\varphi(n)$ factors in each product, so we conclude

$$\prod_{x \in \mathbb{Z}_n^*} x \equiv a^{\varphi(n)} \prod_{x \in \mathbb{Z}_n^*} x \pmod{n}$$

from this. Multiplying by an inverse of $\prod_{x \in \mathbb{Z}_n^*} x$ yields the desired result. □

We turn now to the question of computing $\varphi(n)$. Two results solve this problem in general:

Proposition 8. *If p is a prime number and $k \geq 1$ then*

$$\varphi(p^k) = (p-1)p^{k-1}.$$

Proof. Just note that $\gcd(x, p^k) = 1$ if and only if $p \nmid x$. Among the p^k numbers in $\mathbb{Z}_{p^k}$, p divides every p'th number, so a fraction $1/p$ of them is not invertible modulo p^k. The invertible ones will be the remaining fraction $1 - 1/p = (p-1)/p$, for a total of $(p-1)/p \cdot p^k = (p-1)p^{k-1}$ invertible members. □

Proposition 9. *If* $\gcd(u, v) = 1$ *then*

$$\varphi(uv) = \varphi(u)\varphi(v).$$

Proof. We will use the Chinese remainder theorem. I claim that a number c is invertible modulo uv if and only if it is invertible modulo u and modulo v.

Certainly, an inverse of c modulo uv is also an inverse of c modulo u and modulo v both. Conversely, let a be an inverse of c modulo u and let b be an inverse of c modulo v. By the CRT, there is some x with $x \equiv a \pmod{u}$ and $x \equiv b \pmod{v}$. Then $cx \equiv ca \equiv 1 \pmod{u}$ and $cx \equiv cb \equiv 1 \pmod{v}$. By the uniqueness part of CRT, $cx \equiv 1 \pmod{uv}$ follows.

We have shown that the one-to-one mapping between $\mathbb{Z}_{uv}$ and $\mathbb{Z}_u \times \mathbb{Z}_v$ given by the CRT is also a one-to-one mapping between the respective subsets $\mathbb{Z}_{uv}^*$ and $\mathbb{Z}_u^* \times \mathbb{Z}_v^*$. But the latter has precisely $\varphi(u)\varphi(v)$ members, and we're done. □

To compute $\varphi(n)$ for any integer n is now easy, provided we know the prime factorization of n. In fact, if $p_1, \ldots, p_m$ are distinct primes then

$$\varphi\left(p_1^{k_1} \cdots p_m^{k_m}\right) = (p_1 - 1)p_1^{k_1 - 1} \cdots (p_m - 1)p_m^{k_m - 1}.$$

For example, since $2010 = 2 \cdot 3 \cdot 5 \cdot 67$ we find

$$\varphi(2010) = 1 \cdot 2 \cdot 4 \cdot 66 = 528.$$

Application to the RSA cryptosystem

Alice wants everybody to be able to encrypt (relatively short) secret messages to her. To achieve this, she picks two large primes p and q and computes their product $m = pq$. She keeps the primes secret, but publishes her *modulus* m. She also picks an *encryption exponent* e so that $\gcd(e, \varphi(m)) = 1$ and publishes that as well. The pair (m, e) is Alice's *public key*.

In the early days of RSA, $e = 3$ was not an uncommon choice. However, that is insecure if the same message is encrypted for several recipients. These days, $e = 2^{16} + 1 = 65537$ (a prime number) is a common choice.

To encrypt a message to Alice, Bob encodes the message as a number $x \in \mathbb{Z}_m^*$, computes $X = x^e \bmod m$ and transmits to Alice.

To decrypt, Alice computes $x' = X^d \bmod m$ where the d is an inverse of e modulo $\varphi(m)$.

To see why this works, assume that $\gcd(x, m) = 1$. Then by Euler's theorem, $x^{\varphi(m)} \equiv 1 \pmod{m}$. Also, since $ed \equiv 1 \pmod{\varphi(m)}$, we have $ed = 1 + k\varphi(m)$ for some integer k. So we find

$$x' \equiv X^d \equiv (x^e)^d = x^{ed} = x^{1+k\varphi(m)} = x \cdot (x^{\varphi(m)})^k \equiv x \cdot 1^k = x.$$

Finally, we get $x' = x$ since $x, x' \in \mathbb{Z}_m$.

It is a curious fact that RSA works even for messages $x \in \mathbb{Z}_m$ which are not mutually prime to m. (I skip the proof here.) However, this is not so important. First, if $x = 0$ then $X = 0$ as well, but *anybody* can decrypt $X = 0$, so this is not secure. (In fact RSA is not secure for any really small x, since an adversary can easily compute a table of x^e for small x and compare with an observed X.) Second, if $1 < x < m$ and $\gcd(x, m) > 1$ then $\gcd(x, m) \in \{p, q\}$. But in that case, since Bob can easily compute $\gcd(x, m)$, he has in fact stumbled upon the factorization of m, thus having broken the security of Alice's secret key.

To save herself some work, Alice only needs compute the *decryption exponent* d once. She can then use it to decrypt all incoming messages, but she must of course keep it secret.

She can save herself even more work by exploiting the Chinese Remainder Theorem. To decrypt X, she only needs to compute $X^d \bmod p$ and $X^d \bmod q$ and combine the results using the CRT. Better still, those two numbers can be computed as $X^{d \bmod (p-1)} \bmod p$ and $X^{d \bmod (q-1)} \bmod q$, thanks to Fermat's little theorem.

For RSA to be secure, it is clearly necessary that no outsiders can compute d. Since computing modular inverses is easy, it is important that nobody can compute $\varphi(m)$. We know that $\varphi(m) = (p-1)(q-1) = pq - p - q + 1$. Since pq is known to the public, computing $\varphi(m)$ is equivalent to computing $p + q$. But someone who knows $m = pq$ and $n = p + q$ can easily compute p and q, since then

$$(p - q)^2 = n^2 - 4m$$

so the attacker now knows $p + q$ and $p - q$ and can get p and q easily enough.

At the beginning of this section, I stated that RSA with encryption exponent $e = 3$ is insecure if used to encrypt the same message x to several recipients. More

precisely, assume we are given public moduli m_1, m_2, and m_3. Assume further that Bob computes $X_i = x^3 \bmod m_i$ for $i = 1, 2, 3$ and transmits all three values. If you intercept all of them, you may be well assured that the moduli m_1, m_2, m_3 are mutually prime, so you can use the Chinese remainder theorem to compute $X = x^3 \bmod m_1 m_2 m_3$ by solving the congruences $X \equiv X_i \pmod{m_i}$ for $i = 1, 2, 3$. *However*, by the way RSA encodes messages, we must have $1 < x < m_i$ for $i = 1, 2, 3$, so $1 < x^3 < m_1 m_2 m_3$. Thus $X = x^3$ (with *no* reduction modulo $m_1 m_2 m_3$ needed), and extracting the cube root of X to discover the value of x is easy.

Primality testing

Large primes are needed for Diffie–Hellman, ElGamal and RSA. Yet, finding large primes can seem quite difficult. The naïve way to check whether an integer n is prime is by *trial division*: For every a with $1 < a \le \sqrt{n}$ (it is enough to consider prime numbers a), check whether $a \mid n$. Only if the answer is no for every a, can we be assured that n is prime. Even for n of relatively modest size (in this connection), such as $n \approx 2^{160}$, we end up having to do 2^{80} trial divisions. This is an impossible task. So other methods are needed.

It was quite a sensation when a polynomial time deterministic primality test was discovered in 2002. By this is meant a test that provides a definite answer to the question whether a given number n is prime or not within a time that is bounded above by a polynomial in $\log n$ (the current best such estimate is a constant times $(\log n)^6$).

Polynomial-time *probabilistic* primality tests have been known far longer, and are still widely used in practice, as they are both simpler and faster and can provide an answer with any desired degree of confidence.

These tests have two possible outcomes: Either they prove conclusively that n is composite, or they don't, in which case one can conclude that n is likely prime.

The Fermat test. To illustrate these ideas, consider the *Fermat test*. This is based on Fermat's little theorem, that if p is prime then $a^{p-1} \equiv 1 \pmod{p}$ for every a with $1 < a < p$.

The test is simple enough: To see if a number n is prime, pick a random a with $1 < a < n$, compute $a^{n-1} \bmod n$ and see if the answer is 1. If it isn't, n is certainly composite, and a is called a *Fermat witness* to the compositeness of n. Otherwise, repeat the procedure a number of times.

Unfortunately, there are some composite numbers, called *Carmichael numbers*, that are unusually likely to pass the Fermat test. To be precise, a Carmichael number is a composite number n so that $a^{n-1} \equiv 1 \pmod{n}$ whenever $\gcd(a, n) = 1$. Though Carmichael numbers are quite rare, it is known that there exist an infinite number of them.

Alwin Korselt proved in 1899 that n is a Carmichael number if and only if n is square free (if $a > 1$ then $a^2 \nmid n$) and for every prime p, if $p \mid n$ then $(p-1) \mid (n-1)$. The smallest Carmichael number is $561 = 3 \cdot 11 \cdot 17$. It was found by Robert Carmichael in 1910; hence the name.

Despite the possible problem with Carmichael numbers, the Fermat test is often used as a first step (typically with $a = 2$) to rapidly weed out non-primes before subjecting surviving candidates to more rigorous tests.

The Miller–Rabin test. Fortunately, a proper primality test can be based on a slight extension of Fermat's little theorem.

Before we state this theorem, however, we state and prove a simple lemma on square roots modulo a prime.

Lemma 10 (Uniqueness of square roots modulo p)**.** *Let p be a prime number. If a and b are numbers so that*

$$a^2 \equiv b^2 \pmod{p}$$

then $a \equiv \pm b \pmod{p}$.

Proof. Since $a^2 - b^2 = (a-b)(a+b)$ we find $(a-b)(a+b) \equiv 0 \pmod{p}$, which is the same as saying $p \mid (a-b)(a+b)$. Since p is prime, according to Corollary 3 either $p \mid (a-b)$ or $p \mid (a+b)$, that is either $a - b \equiv 0$ or $a + b \equiv 0 \pmod{p}$. □

We may note in passing that the above property does not hold if the prime p is replaced by a composite number n. First, if n is a prime power, that is $n = p^r$ for some prime p and $r > 1$, then picking $a = p^k$ with $k < r$ and $2k \geq r$ and $b = 0$ yields a counterexample. Otherwise, if n is composite we can write $n = uv$ for mutually prime numbers $u, v > 1$ (with $u \neq 2$). By the Chinese remainder theorem there exists a solution a to the congruences $a \equiv -1 \pmod{u}$ and $a \equiv 1 \pmod{v}$. But then we know $a^2 \equiv 1 \pmod{uv}$, thanks to the uniqueness part of the CRT and the fact that this congruence holds modulo u and v both. So we have a counterexample with $b = 1$.

Lemma 11 (Miller–Rabin)**.** *Let p be an odd prime and write $p - 1 = 2^s r$ with r odd. If $1 < a < p$ then either*

- $a^r \equiv 1 \pmod{p}$, *or*
- $a^{2^j r} \equiv -1 \pmod{p}$ *for some $j \in \{0, \ldots, s-1\}$.*

Proof. Let k be the smallest nonnegative integer so that $a^{2^k r} \equiv 1$. There certainly is such an integer, and $k \leq s$, since $a^{2^s r} = a^{p-1} \equiv 1$ by Fermat's little theorem. If $k = 0$ we have the first case. Otherwise, put $j = k - 1$ and let $b = a^{2^j r}$. Then $b^2 = a^{2^k r} \equiv 1 = 1^2$. By Lemma 10, $b \equiv \pm 1$. But $b \not\equiv 1$ by the minimality of k, so $b \equiv -1$. □

We can now describe the *Miller–Rabin primality test* as follows:

We are *given* an odd integer $n > 1$ and wish to decide whether n is prime. First of all, write $n = 2^s r$ with r odd.

Next, perform the following computation:

1. Pick a random number a with $1 < a < n$.
2. Compute $a^r \bmod n$. If the answer is 1, stop; n is possibly prime.
3. Otherwise, compute $a^{2^j r} \bmod n$ for $j = 1, 2, \ldots, s-1$. If the answer is congruent to -1 (i.e., equal to $n-1$), stop; n is possibly prime.
4. Otherwise (i.e., after completing the previous step for $j = 1, 2, \ldots, s-1$ without stopping), n is definitely composite. In this case, we call a a *Miller–Rabin witness* to the compositeness of n.

Notice that after the computation of $a^r \bmod n$ in step 2, the subsequent computations of $a^{2^j r} \bmod n$ for $j = 1, 2, \ldots, s-1$ are easily performed by successively squaring the previous result modulo n. Also, if we find in step 3 that $a^{2^j r} \equiv 0$ or $a^{2^j r} \equiv 1$, then obviously no amount of further squaring will yield the result -1, so one might as well stop right away and declare n composite.

The full Miller–Rabin test consists of repeating the above procedure for different random a until either n has been shown to be composite, or t different a have been tried without such a result, in which case you declare that n is *probably* prime.

It can be shown that if n is composite, then at least three quarters of all the numbers in $\{2, \ldots, n-1\}$ are witnesses to the compositeness of n, and so the Miller–Rabin test will fail to show that n is composite with a probability less than 4^{-t}, which goes quite rapidly to zero as t grows bigger. For example, with $t = 40$ there is at most a probability 2^{-80} of failing to show that a composite number is in fact composite.

It is worth noting that there is no (known) way to find a non-trivial factor of n given a (Fermat or Miller–Rabin) witness to the compositeness of n. It is in general much harder to find non-trivial factors of a number than showing it is composite.

Finding large primes. To find a large prime, say with r bits, a good procedure is to pick a random r-bit odd number n and to test it for primality. If it fails, pick another one and start over.

It is not a good idea to replace n by $n+2$ instead; the reason is that gaps between primes are quite variable in length, and this procedure would favour primes with long gaps in front of them, thus perhaps easing the task of a potential adversary trying to guess your choice of prime number. Also, if you were unlucky enough to find yourself in a large gap, finding a prime could take a long time. Gaps between prime numbers can be arbitrarily large: For any number $m > 1$, all the $m-1$ numbers $m!+2$, $m!+3, \ldots, m!+m$ are composite.

To speed up the testing phase, consider trial division with small primes as a first step. Or better yet, compute $\gcd(n, b)$ where b is a product of small primes such as $b = 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19$. If the answer is not 1, n is composite.

Then apply the Miller–Rabin test to the survivors.

Although the Fermat test is easier to describe (and to program) than the Miller–Rabin test, you may note that the latter does almost no work that is not already needed for the former. And of course, a Fermat witness to compositeness is also a Miller–Rabin witness,

so given that we wish to employ Miller–Rabin anyhow, there is little point in screening with the Fermat test.

Of course, if your prime is extremely important to you and you have the computational resources to do it, by all means go ahead and perform a rigorous (non-probabilistic) test after a suitable number of rounds of Miller–Rabin. Such tests are beyond the scope of this text, however.

Primality testing versus factoring. I stated above that there is no known way to find a non-trivial factor given a witness to compositeness. While this is true in general, there are exceptions.

First, even a Carmichael number n can fail the Fermat test, since $\gcd(a,n) \neq 1$ implies $a^{n-1} \not\equiv 1 \pmod{n}$. (For otherwise, a has the inverse a^{n-2} modulo n.) However, if you have come across some a with $1 < a < n$ and $\gcd(a,n) \neq 1$ then $\gcd(a,n)$ is a non-trivial factor of n, so doing the Fermat test computation is just wasted effort if you are looking for factors.

Second, if you stopped the Miller–Rabin test because $a^{2^j r} \equiv 0$ or $a^{2^j r} \equiv 1$ for some $j > 1$, you can find a non-trivial factor. For in this case, $b = a^{2^{j-1} r}$ is a non-trivial square root of either 0 or 1 modulo n.

In the first case, if $b \not\equiv 0$ but $b^2 \equiv 0 \pmod{n}$ then b is not invertible modulo n, so $\gcd(b,n) > 1$. (In this case, a isn't invertible either, so you could have saved some work by computing $\gcd(a,n)$ directly.)

In the second case, we have $b^2 \equiv 1$ but $b \not\equiv \pm 1 \pmod{n}$. But then $(b-1)(b+1) = b^2 - 1 \equiv 0 \pmod{n}$, so $n \mid (b-1)(b+1)$, and at least one of $b \pm 1$ has a common factor with n.

At first glance, this could seem a good way to find factors, but at least for the RSA case (product of two large primes), this case turns out to happen too rarely to be useful.

Finding safe primes and generators. A *safe prime* is a prime p so that $(p-1)/2$ is also a prime. Equivalently, it is a prime of the form $p = 2q+1$ where q is prime. (We may return to the reason for this terminology at a later point.) Focusing on q instead, a *Sophie Germain prime* is a prime q so that $2q+1$ is prime.

One reason for working with these primes is that it is easy to check whether some a is a generator of $\mathbb{Z}_p^*$ when p is a safe prime. The reason is the following simple lemma:

Lemma 12. *Let n be a positive integer, and $a \in \mathbb{Z}_n^*$. Then there is a smallest positive integer r such that $a^r \equiv 1 \pmod{n}$. This integer, which is called the* order *of a, is a divisor of $\varphi(n)$.*

Proof. Since $\mathbb{Z}_n^*$ is finite, the powers a^x cannot all be different modulo n, so there are integers $0 \leq x < y$ with $a^x \equiv a^y \pmod{n}$. Since a is invertible modulo n, $a^{y-x} \equiv 1 \pmod{n}$ follows, and the existence of r is assured.

Now write $\varphi(n) = qr + s$ with $0 \leq s < r$. We find

$$a^s = a^{\varphi(n) - qr} = a^{\varphi(n)} (a^r)^{-q} \equiv 1 \pmod{n}$$

by Euler's theorem and $a^r \equiv 1$. Since r is the *smallest* positive integer with this property, $s = 0$, and so $r \mid \varphi(n)$. □

Applying this to safe primes, we find

Lemma 13. *Let $p = 2q+1$ be a safe prime, and $a \in \mathbb{Z}_p^*$. Then a is a generator of $\mathbb{Z}_p^*$ if and only if $a^2 \not\equiv 1$ and $a^q \not\equiv 1 \pmod{p}$.*

Proof. Recall that a is called a generator if the powers of a fill up all of $\mathbb{Z}_p^*$. Clearly, this is equivalent to the period of a being $p-1$. If this is so, then $a^2 \not\equiv 1$ and $a^q \not\equiv 1 \pmod{p}$.

Conversely, if the above two non-equivalences hold, knowing that the order r of a satifies $r \mid \varphi(p) = p - 1 = 2q$, the only possibilities are $r = 1$, $r = 2$, $r = q$, and $r = pq$. The assumptions rule out all the possibilities except the final one. □

To find a generator of $\mathbb{Z}_p^*$ where p is a safe prime, then, we only need to pick $a \in \mathbb{Z}_p^*$ at random and test them using the above lemma until we find a generator. There are exactly two solutions (modulo p) to $x^2 \equiv 1$ and q solutions to $x^q \equiv 1$, so the probability that a randomly chosen a will be a generator is about $1/2$, and the search should not need to take very long.

Next, how do we find safe primes, or equivalently, how do we find Sophie Germain primes?

The obvious strategy is to first find primes q using strategies described previously, i.e., picking random q and then applying the Miller–Rabin test. Then, once a (probable) prime q has been found, we can check whether $n = 2q+1$ is prime as well. We *could* use the Miller–Rabin test again, but there is a quicker way, based on the assumption that q is prime.

Here is how: Assume that $n = 2q+1$, where q is prime. First we apply the Fermat test with some a. Assuming it passes, we have $a^{n-1} \equiv 1 \pmod{n}$. Hang on to that a; we will use it again.

Next, assume that n is in fact composite. We shall try to get a contradiction. So there is a prime $p \mid n$ with $p \leq \sqrt{n}$. Clearly $p - 1 < q$, so $\gcd(p-1, q) = 1$ since q is prime. From this we conclude that q is invertible modulo $p-1$, i.e., there is some u with

$$qu \equiv 1 \pmod{p-1} \tag{2}$$

We return to the relation $a^{n-1} \equiv 1 \pmod{n}$, which we now write $a^{2q} \equiv 1 \pmod{n}$. Taking the u-th power of this we have $a^{2qu} \equiv 1 \pmod{n}$ and therefore,

in particular, $a^{2qu} \equiv 1 \pmod{p}$. From Fermat's little theorem and (2) we conclude $a^2 \equiv 1 \pmod{p}$ (exercise: show this). In other words $p \mid (a^2-1)$, and so

$$\gcd(a^2-1, n) \neq 1.$$

Now if, on the contrary, we find

$$\gcd(a^2-1, n) = 1$$

we have the desired contradiction, and it follows that n must be prime.

Getting this contradiction is in fact quite likely if n is prime, so this is a useful test.

The above proof generalizes readily into a proof of the following:

Theorem 14 (Pocklington)**.** *Assume that $n = bq+1$ with q a prime number. Assume further that a is a number, and*

$$q > \sqrt{n} - 1,$$
$$a^{n-1} \equiv 1 \pmod{n},$$
$$\gcd(a^b - 1, n) = 1.$$

Then n is a prime number.

I skip the proof, which is just like the above, with the number 2 replaced by b. But I will make one remark: The requirement $q > \sqrt{n} - 1$ is need for the inequality $p - 1 < q$, which was important in the proof.

Quadratic residues

An integer a is called a *quadratic residue* modulo n if it is congruent modulo n to a square:

$$a \equiv x^2 \pmod{n} \quad \text{for some } x \in \mathbb{Z}.$$

For any given quadratic residue a, any x satisfying the above equation is called a *square root* of a modulo n.

The question of deciding if a given number is a quadratic residue, and if so to compute its square roots, is radically different depending on whether n is a prime or not. We deal with the case of odd primes first (obviously, every number is a quadratic residue modulo 2).

Quadratic residues modulo an odd prime p.

First we note that if x is a square root of a then so is $-x$. Furthermore, x and $-x$ are the *only* square roots of a. To be precise, assume that x and y are two square roots of a. Then

$$(x-y)(x+y) = x^2 - y^2 \equiv 0 \pmod{p}.$$

Thus either $x - y \equiv 0$ or $x + y \equiv 0$, so that $x \equiv \pm y \pmod{p}$. (This apparently obvious statement needs justification: By the above, $p \mid (x-y)(x+y)$ so p divides one of the factors, since p is prime.)

In particular, the only square roots of 1 modulo p are ± 1. But Fermat's little theorem implies that

$$\left(a^{\frac{p-1}{2}}\right)^2 = a^{p-1} \equiv 1 \pmod{p}$$

and therefore[1]

$$a^{\frac{p-1}{2}} \equiv \pm 1 \pmod{p} \tag{3}$$

whenever $\gcd(a, p) = 1$. Moreover, subsituting $a = x^2$ yields $x^{p-1} \equiv 1$, so we obtain the plus sign above when a is a quadratic residue.

A trick similar to the above lets us compute square roots modulo p easily in the case when $p+1$ is divisible by 4 – or equivalently, when $p \equiv 3 \pmod{4}$. For then

$$\left(a^{\frac{p+1}{4}}\right)^2 = a^{\frac{p+1}{2}} = a^{\frac{p-1}{2}+1} = a^{\frac{p-1}{2}} a = \pm a,$$

so that $x = a^{(p+1)/4}$ is a square root of $\pm a$. Thus the sign in equation (3) completely settles the question of whether a is a quadratic residue in the case $p \equiv 3 \pmod{4}$.

Proposition 15 (Euler's criterion)**.** *An integer a is a quadratic residue modulo an odd prime p if and only if either $a \equiv 0 \pmod{p}$ or $a^{(p-1)/2} \equiv +1 \pmod{p}$.*

This is still true, but harder to prove, when $p \equiv 1 \pmod{4}$. The result can be established by a counting argument: Since the mapping $x \mapsto x^2$ is two-to-one from $\mathbb{Z}_p^*$ to itself, precisely one half of the members of $\mathbb{Z}_p^*$ are quadratic residues. They are also zeros of the polynomial $x^q - 1$ where $q = (p-1)/2$. But this polynomial can have at most q zeros.

It is also trickier to find square roots in the case $p \equiv 1 \pmod{4}$. We will not consider this problem further.

Quadratic residues modulo a composite number n. If n is composite but not a prime power, we can factor it as $n = uv$ with $u, v > 1$ and $\gcd(u, v) = 1$. Then in general $x \mid n$ if and only if $x \mid u$ and $x \mid v$. This implies that any congruence holds modulo n if and only if it holds modulo u and v both. In particular, $y \equiv x^2 \pmod{n}$ if and only if $y \equiv x^2 \pmod{u}$ and $y \equiv x^2 \pmod{v}$. Thus if a is a quadratic residue modulo n then it is also quadratic residue modulo u and v.

We can prove the converse using the Chinese remainder theorem: If $a \equiv y^2 \pmod{u}$ and $a \equiv z^2 \pmod{v}$ then there is a solution x to the simultaneous congruences $x \equiv y \pmod{u}$, $x \equiv z \pmod{v}$, and then $y \equiv x^2 \pmod{n}$ since this holds modulo u and v.

[1] $a^{(p-1)/2} \bmod p$ is known as a *Legendre symbol*. There are more efficient algorithms for computing it by using properties of the more general *Jacobi symbol*.

However, we have more freedom of signs now: We can find square roots of a from the simultaneous congruences

$$x \equiv \pm y \pmod{u}, \quad x \equiv \pm z \pmod{v}$$

where all four choices of signs are available to us. Thus we usually get at least four different square roots for a, the exception being when $y = 0$ or u is even and $y = u/2$ (or ditto for z, v).

Thus we see that knowing a non-trivial factorization of n allows us to find non-trivial solutions of $x^2 \equiv y^2$. The converse is also true:

Lemma 16. *If $x^2 \equiv y^2$ but $x \not\equiv \pm y \pmod{n}$ then $\gcd(n, x-y)$ is a nontrivial factor of n.*

Proof. Clearly $d = \gcd(n, x-y)$ is a factor of n. To show it is a *nontrivial* factor, we start with

$$(x-y)(x+y) = x^2 - y^2 \equiv 0 \pmod{n}, \tag{4}$$

in other words $n \mid (x-y)(x+y)$. But $n \nmid (x-y)$ since $x \not\equiv y$, so $d < n$, since $d \mid (x-y)$. Also $d > 1$, for if $d = 1$ then $x - y$ is invertible modulo n, and we could multiply (4) by an inverse of $x - y$ to get $x + y \equiv 0 \pmod{n}$, and this is assumed not to be the case. □

The above lemma lurks at the heart of various factorization algorithms. To find a nontrivial factorization of n, it is enough to find nontrivial solutions to $x^2 \equiv y^2$.

Factoring

By factoring an integer n we mean finding the primes p_i and their corresponding exponents k_i in the factorization $n = p_1^{k_1} \cdots p^{k_m}$. One typically proceeds by merely finding a nontrivial factor x, i.e., an integer satisfying $1 < x < n$ and $x \mid n$. Then we have $n = xy$ with $x, y > 1$, and we can repeat the procedure on each of the factors x and y, continuing until only prime factors are found.

Pollard's ρ algorithm. This algorithm finds moderately sized factors of n quickly. If n has only small prime factors, or more generally just one large prime factor, it can thus be completely factored quickly.

We begin with a simple observation. Assume that $p \mid n$ for a moderately sized prime p. Given two distinct integers that are congruent modulo p, subtraction yields $x \neq 0$ with $x \equiv 0 \pmod{p}$, in other words $p \mid x$. Thus $\gcd(x, n) > 1$, and given just a little bit of luck $\gcd(x, n)$ will be a nontrivial factor of n.

To use this observation, we only need to come up with a systematic way of selecting pairs (x, y) of integers that are more likely than average pairs to be mutually congruent modulo p (keeping in mind that we don't actually know p of course).

To do this, we use a function $f\colon \mathbb{Z}_n \to \mathbb{Z}_n$. This function needs be such that $f(x) \bmod p$ will only depend on $x \bmod p$, so that there is also a function $g\colon \mathbb{Z}_p \to \mathbb{Z}_p$ such that $g(x \bmod p) \equiv f(x) \pmod{p}$ for all $x \in \mathbb{Z}_n$.

A good example of such a function is $f(x) = x^2 + 1 \bmod n$: The corresponding g is given by $g(x) = x^2 + 1 \bmod p$. It satisfies the requirement if $p \mid n$ (but not otherwise).

Now we iterate: Pick some $x_0 \in \mathbb{Z}_n$, then put

$$x_i = f(x_{i-1})$$

for $i = 1, 2, \ldots$. We note that the sequence $(\bar{x}_i)$, where $\bar{x}_i = x_i \bmod p$, is *also* given by an iterative formula:

$$\bar{x}_i = g(\bar{x}_{i-1}).$$

Since $\mathbb{Z}_p$ is finite, the sequence $(\bar{x}_i)$ will sooner or later begin to repeat itself (and hopefully much sooner than (x_i) does), so that $\bar{x}_{i+k} = \bar{x}_i$ for all sufficiently large i and some k called the cycle length.

Our next ingredient is Floyd's cycle finding algorithm: If we put $\bar{y}_i = \bar{x}_{2i}$ then sooner or later (for sufficiently large i) both $\bar{x}_i$ and $\bar{y}_i$ will have entered the cycle, and $\bar{y}_i$ moves two steps around the cycle for each single step of $\bar{x}_i$. So $\bar{y}_i$ gains one step on $\bar{x}_i$ per iteration, and will sooner or later catch up, at which point $\bar{y}_i = \bar{x}_i$.

To make the algorithm practical, we don't keep track of the index i, just use one variable x for x_i and another y for x_{2i}. Here, then, is Pollard's ρ algorithm.

- Put $x \leftarrow 2$, $y \leftarrow 2$.
- Repeat $x \leftarrow f(x)$, $y \leftarrow f\big(f(y)\big)$ until either
  - $x = y$, in which case the algorithm failed, or
  - we just plain give up, or
  - $\gcd(|x - y|, n) > 1$, in which case that is a nontrivial factor of n.

Note that it is the final test that is likely to be triggered when we have arrived at a point where $x \equiv y \pmod{p}$. But we could be unlucky and have the first test triggering instead. If so, retrying with a different function might help.

The Fermat factoring method. The security of RSA depends on the difficulty of factoring $n = pq$ when p and q are large primes. Clearly, Pollard's ρ is of no use here! The existence of the Fermat method, on the other hand, demonstrates the importance of not having p and q very close together. It works as follows:

The goal is to factor n as $n = (x-y)(x+y) = x^2 - y^2$. If the two factors are very close together, that means y is small and $x = \sqrt{n + y^2} \approx \sqrt{n}$.

Notice that $x^2 = n + y^2$ implies $x\,dx = y\,dy$, so if $y \ll x$ then y will vary much faster than x. Because we are looking for integer solutions, it is much more efficient to step x through integer values than to do the same for y:

- Set $x \leftarrow \lceil \sqrt{n} \rceil$ (the smallest integer $\geq \sqrt{n}$).

- While $x^2 - n$ is not a square, set $x \leftarrow x+1$.
- Now $n = (x-y)(x+y)$, where $x^2 - n = y^2$.

Assuming $n = pq$ with primes $p \le q$, this algorithm stops when $p = x - y$ and $q = x + y$. That is, it stops when $x = (p+q)/2$. Thus the number of steps required is

$$N = \frac{p+q}{2} - \lceil \sqrt{n} \rceil \approx \frac{p+q}{2} - \sqrt{n}.$$

Multiply by $\frac{1}{2}(p+q) + \sqrt{n} \approx \frac{3}{2}\sqrt{n}$:

$$\tfrac{3}{2}\sqrt{n}N \approx \left(\frac{p+q}{2}\right)^2 - n = \left(\frac{q-p}{2}\right)^2 \qquad \text{(using } n = pq\text{)}$$

and conclude that

$$N \approx \frac{(q-p)^2}{6\sqrt{n}}$$

is the approximate number of steps required.

For this to be about as hard as trial division, we would like to have N be of the same order of magnitude as $\sqrt{n}$. We achieve this by requiring $q - p$ to be of the same order of magnitude as p and q themselves. For example, if $p < \frac{1}{2}q$ then n is quite resistant to Fermat factoring.

The quadratic sieve. This method and its more advanced variations is the most efficient general factoring algorithm currently available. It works by collecting many numbers $x \in \mathbb{Z}_n^*$ such that $x^2 \bmod n$ is a product of small primes, then multiplying together a selection of these. The product will again be a product of small primes. If we can arrange that all these primes appear an even number of times in the product, then we have arrived at a congruence $a^2 \equiv b^2 \pmod{n}$ that just might factor n.

In more detail, assume we have collected l such numbers, all of whose squares are congruent to a product of the first k primes $p_1, \dots, p_k$:

$$x_j^2 \equiv p_1^{\mu_{1j}} \cdots p_k^{\mu_{kj}} \pmod{n}, \qquad j = 1, \dots, l \tag{5}$$

We multiply together *some* of them. Encode the choice in a vector $(\xi_1, \dots, \xi_k)$ with each $\xi_j \in \{0,1\}$: Then

$$\left(x_1^{\xi_1} \cdots x_l^{\xi_l}\right)^2 \equiv p_1^{\eta_1} \cdots p_k^{\eta_k} \pmod{n} \tag{6}$$

where

$$\eta_i = \sum_{j=1}^{l} \mu_{ij}\xi_j, \qquad i = 1, \dots, k.$$

We need each η_i to be even in order for the product $p_1^{\eta_1} \cdots p_k^{\eta_k}$ to be a square: Thus we need to solve the congruences

$$\sum_{j=1}^{l} \mu_{ij}\xi_j \equiv 0 \pmod{2}, \qquad i = 1, \dots, k.$$

This is a set of k homogeneous linear equations in l unknowns, so can find a nontrivial solution if $l > k$. (The theory of linear equations in $\mathbb{Z}_2$, or more generally in $\mathbb{Z}_p$ for a prime p, is virtually identical to the better known linear algebra with real or complex numbers.)

Finally, having solved the system so that each η_i is even, we see that (6) takes the form $a^2 \equiv b^2 \pmod{n}$, where $a = x_1^{\xi_1} \cdots x_l^{\xi_l}$ and $b = p_1^{\eta_1/2} \cdots p_k^{\eta_k/2}$. However, this does not always lead to a nontrivial factor of n, so it is a good idea to collect enough x_i so one can try different possibilities with $l = k+1$.

The remaining question is how one finds numbers x_i satisfying (5). One very simple idea is to ensure that $x_i^2 \bmod n$ is small. Then it is more likely that the prime factors of this number are small, and in particular not greater than p_k. So we can look for x_i among numbers only slightly larger than $\sqrt{n}$ (or more generally $\sqrt{cn}$ for integers $c \ge 1$). Beyond this simple idea, there are various techniques for speeding up the search that we will not cover here.

The quadratic sieve is quite good for factoring numbers up to around 100 digits or so. For bigger numbers, a more complicated, related method called the *general number field sieve* (GNFS) is currently the best available.

These methods are very well suited for parallel computation, as the first step – the sieving, or collection of the x_i – can be done independently by many computers searching different parts of the space of likely candidates.

In December 2009, a 768 bit (232 decimal digit) RSA number was factored using the GNFS. The effort took at least 2000 CPU years, with several hundred computers participating in the sieving process. It is estimated that factoring a 1024 bit RSA number (a popular key size in current RSA usage) is about 1000 times harder. This could soon be within reach of a serious large scale effort!

The discrete logarithm problem

When p is a prime and $g \in \mathbb{Z}_p^*$ is a generator, the discrete logarithm problem (DLP) is the problem, given $y \in \mathbb{Z}_p^*$ of finding the (unique) $x \in \mathbb{Z}_{p-1}$ so that

$$g^x = y.$$

(If g is not a generator, a solution may not exist, or it will not be unique if it exists. We might still call this a discrete logarithm problem, but it will not interest us.)

The Pohlig–Hellman algorithm. This is an algorithm that works well if all the prime factors of $p-1$ are known and of moderate size. If so, we can find those factors using Pollard's ρ algorithm. Because of this, one should avoid using such primes in cryptographic applications whose security relies on the difficulty of the DLP.

The algorithm has three parts.

- If q is a prime and $q \mid p-1$ then we can find $x \bmod q$.
- More generally, if $q^r \mid p-1$ then we can find $x \bmod q^r$.
- Finally, if $p-1 = q_1^{r_1} \cdots q_m^{r_m}$ with distinct primes q_i and we know $x \bmod q_i^{r_i}$ for each i then we can find $x \bmod p-1$ using the Chinese remainder theorem.

The final point is straightforward, so we concentrate on the first two.

First, when $y = g^x$ then a simple application of Fermat's little theorem yields

$$y^{\frac{p-1}{q}} = g^{\frac{p-1}{q}x} \equiv g^{\frac{p-1}{q}k_0} \pmod{p} \qquad \text{if } x \equiv k_o \pmod{q}.$$

So if we compute all the values (obviously it is important in this step that q not be too large)

$$g^{\frac{p-1}{q}k} \bmod q, \qquad k = 0, 1, \ldots, q-1,$$

then a simple comparison is sufficient to find $k_0 = x \bmod q$.

Second, we take the case $r = 2$, i.e., $q^2 \mid p-1$. Write $x = x_1 q + k_0$, where we found k_0 previously. Define $y_1 \in \mathbb{Z}_p^*$ so that

$$y_1 \equiv g^{x_1 q} = g^{-k_0} y \pmod{p}.$$

So we know y_1, but not yet x_1. We take a suitable integer power of y_1:

$$y_1^{\frac{p-1}{q^2}} \equiv g^{\frac{p-1}{q}x_1} \equiv g^{\frac{p-1}{q}k_1} \pmod{p} \qquad \text{if } x_1 \equiv k_1 \pmod{q}.$$

Again, we can find k_1 by comparison. As before we have $x_1 = x_2 q + k_1$ and therefore $x = x_2 q^2 + k_1 q + k_0$, and $x \equiv k_1 q + k_0 \pmod{q^2}$ is known.

If $q^3 \mid p-1$ we can find $x \bmod q^3$ in the same way starting from x_2, and so on up to the largest power r with $q^r \mid p-1$, thus completing the second part of the algorithm.

The index calculus. This method tries to solve the discrete logarithm by focusing the effort on products of small primes. It is very similar to the quadratic sieve method for factoring.

So let p be an odd prime and g a generator of $\mathbb{Z}_p^*$. Let $p_1, \ldots, p_k$ be the k smallest primes. We look for exponents γ so that $g^\gamma \bmod p$ is a product of powers of these primes. Collecting a large enough number of these, we have

$$g^{\gamma_j} \equiv p_1^{\mu_{1j}} \cdots p_k^{\mu_{kj}} \pmod{n}, \qquad j = 1, \ldots, l$$

Next, we look for an exponent β so that

$$y g^{-\beta} \equiv p_1^{\eta_1} \cdots p_k^{\eta_k} \pmod{p},$$

and finally we try to write $p_1^{\eta_1} \cdots p_k^{\eta_k}$ as a power of g by trying to find ξ_j with

$$p_1^{\eta_1} \cdots p_k^{\eta_k} \equiv g^{\gamma_1 \xi_1 + \cdots + \gamma_l \xi_l} \pmod{p},$$

which will be so if

$$\eta_i \equiv \sum_{j=1}^{l} \mu_{ij} \xi_j \pmod{p-1}, \qquad i = 1, \ldots, k.$$

This is a linear set of k equations in l unknowns. The theory of such equations is not quite so straightforward since $p-1$ is not a prime number, so the existence of a nontrivial solution when $l > k$ is not guaranteed. But we can compensate by picking l somewhat larger, and once we have a nontrivial solution, we are done, with the final answer

$$x = \beta + \gamma_1 \xi_1 + \cdots + \gamma_l \xi_l \bmod p-1 \quad \text{satisfying } y \equiv g^x \pmod{p}.$$

How do we pick the number k of primes to use in the algorithm? This does not seem so easy. If k is too small then the search for γ_j and β will take too long. Making k bigger produces more hits in the search, but then of course we need more hits (bigger l), and the linear system to solve at the end becomes bigger and harder to solve. To select k intelligently, we must estimate the proportion of $\mathbb{Z}_p^*$ that are products of powers of $p_1, \ldots, p_k$ as a function of k and use this to find an optimal value of k.

Applications: Commitment and discrete log hash

The Pedersen commitment scheme and (a variant of) the discrete log hash function, described below, use the same mathematical framework, based on a Sophie Germain prime q and its corresponding safe prime $p = 2q+1$. We write $Q_p \subset \mathbb{Z}_p^*$ for the set of invertible quadratic residues modulo p. Recall that exactly half of all the $p-1 = 2q$ members of $\mathbb{Z}_p^*$ belong to Q_p. (This follows from the fact that, on one hand, x and $-x$ have the same square, and on the other, no number can have more than two square roots modulo a prime, so the map $x \mapsto x^2$ is a two-to-one map $\mathbb{Z}_p^* \to Q_p$.) Thus Q_p has q members.

Now, if γ is a generator of $\mathbb{Z}_p^*$ then $\gamma^k \in Q_p$ if and only if k is even. In particular, with $g = \gamma^2$ we find that the powers $g^0, g^1, \ldots, g^{q-1}$ are all distinct and together fill up Q_p. (Of course, $g^q = \gamma^{2q} = \gamma^{p-1} \equiv 1 \pmod{p}$ by Fermat's little theorem.) We call such a g a *generator* of Q_p.

We shall need two generators g and h of Q_p that are *independent* in the sense that nobody can solve the equation $g^\xi \equiv h \pmod{p}$ for ξ. This should involve whoever chose g and h to begin with, and could be achieved by picking two "nothing up my sleeve" numbers: For example, let g be the smallest generator, and let h be the smallest generator greater than $p/\sqrt{2}$. Without such safeguards, whoever picked the generators could just pick a generator g and some random a, pick $h = g^a$, and remember a for later use. This would defeat both of the schemes below.

The Pedersen commitment scheme. The Pedersen commitment scheme has the following features:

- To commit to a message x, Alice computes a "commitment" c and makes it available to others.
- The computation uses a random value r in such a way that knowledge of c reveals no information about x, even to an entity with infinite computational resources.
- When Alice wishes to reveal x to Bob, she tells him the values of x and r. Bob recomputes c based on x and r and checks that the result matches Alice's original commitment.
- If two different pairs (x, r) and (x', r') yield the same commitment, these values can be used to solve a specific instance of the discrete logarithm problem (DLP). To the extent that Bob considers this an impossible task, he must therefore believe that the message received is the message which Alice originally committed herself to.

Using the above framework consisting of a safe prime $p = 2q+1$ and generators g and h of Q_p, here is how to compute the Pedersen commitment for a message $x \in \mathbb{Z}_q$: Pick a random $r \in \mathbb{Z}_q$, drawn from the uniform distribution on $\mathbb{Z}_q$. Compute

$$c = g^x h^r \bmod p.$$

Why $\mathbb{Z}_q$? Notice that if $y \equiv x \pmod{q}$ then $g^y = g^{x+nq} \equiv g^x \pmod{p}$ because $g^q \equiv 1 \pmod{p}$. That is, $g^x \bmod p$ depends only on $x \bmod q$.

First, knowledge of c reveals no information about x because h^r is a random member of Q_p. In fact, since the map $r \mapsto h^r$ is a one-to-one map from Z_q onto Q_p, h^r is drawn from the uniform distribution on Q_p. But then the same is true of $g^x h^r \bmod p$, since multiplication by g^x modulo p is likewise a one-to-one map of Q_p onto itself.

Second, if we can find two different messages with the same commitment string, then can solve the equation $g^\xi = h$ for ξ: For assume that $c \equiv g^x h^r \equiv g^{x'} h^{r'} \pmod{p}$ with $x \not\equiv x' \pmod{q}$. From this we get

$$g^{x-x'} \equiv h^{r'-r} \pmod{p}.$$

From $x - x' \not\equiv 0 \pmod{q}$ we conclude $g^{x-x'} \not\equiv 1 \pmod{p}$, and therefore $r' - r \not\equiv 0 \pmod{q}$. But then $r' - r$ has an inverse u modulo q. Taking the u-th power of the above equivalence we then get

$$g^{(x-x')u} \equiv h^{(r'-r)u} \equiv h \pmod{p},$$

and so $\xi = (x - x')u$ solves $g^\xi \equiv h$.

The discrete log hash function. The discrete log hash is far too inefficient for practical use, yet it is illustrates some basic properties of general hash functions nicely.

Again, it relies on the safe prime $p = 2q+1$ and the two generators g and h of Q_p.

The hash function H maps $\mathbb{Z}_{q^2}$ onto Q_p. Since $\mathbb{Z}_{q^2}$ has q^2 members and Q_p only q, this means a halving in the number of bits.

The hash of a message $m \in \mathbb{Z}_{q^2}$ is

$$H(m) = g^x h^y \bmod p, \qquad m = x + qy, \quad x, y \in \mathbb{Z}_q.$$

Recall that by our conventions, $0 \le m < q^2$, and x and y are chosen with $0 \le x < q$, $0 \le y < q$.

We notice the immediate relationship with Pedersen commitment: The commitment c for a given message x and random nonce r is $H(x + qr)$.

The proof given above that breaking Pedersen commitment implies the ability to solve $g^\xi = h$ also proves that if we have $H(m) = H(m')$ for different $m, m' \in \mathbb{Z}_{q^2}$ then we can once more solve $g^\xi = h$.

Application: Coin tossing by telephone

The protocol to be described allows two parties to declare a winner among them so that each has a 50% chance of winning if the protocol is followed properly. Furthermore, neither party can increase their chance of winning above 50% by departing from the protocol, though they can decrease it. The protocol is as follows.

Alice	**Bob**
Picks primes p and q (preferably $\equiv 3 \pmod 4$) and sends $n = pq$ to Bob.	
	Checks that n is not a prime. Picks random $x \in \mathbb{Z}_n^*$ and sends $y = x^2 \bmod n$ to Alice.
Computes a square root c of y modulo n and sends it to Bob.	
	If $c \equiv \pm x \pmod n$, admit defeat. Otherwise, declare victory. Use $c^2 \equiv x^2 \pmod n$ to find a nontrivial factor of n and send it to Alice as evidence.

In this protocol, y has four square roots modulo n. For, thanks to the Chinese remainder theorem, $c^2 \equiv y \pmod n$ is equivalent to the two equations

$$c^2 \equiv y \pmod p, \quad c^2 \equiv y \pmod q.$$

Each of *those* has two solutions if considered modulo p and q, respectively, say

$$a^2 \equiv y \pmod p, \quad b^2 \equiv y \pmod q.$$

Moreover, Alice can solve these two equations for a and b, and so the four solutions to $c^2 \equiv y \pmod n$ are found by taking the unique solution $c \in \mathbb{Z}_n$ of the simultaneous congruences

$$c \equiv \pm a \pmod p, \quad c \equiv \pm b \pmod q$$

for all four combinations of the two signs.

Two of the four solutions will be congruent modulo n to $\pm x$, and Alice cannot know which. So she has a 50% chance of picking one of those to send to Bob. If she does, she has revealed no information that Bob did not already possess, so he is no closer to factoring n than he did originally.

On the other hand, if Alice picks one of the other two square roots, Bob now knows two numbers c and x with $c^2 \equiv x^2 \pmod n$, but $c \not\equiv \pm x \pmod n$. Thus (see Lemma 16) Bob can factor n and thus wins the coin toss.

The above analysis shows that the protocol works if both parties follow it faithfully. To be useful, however, we must also show that neither party can increase their chance of winning by not following the protocol, but rather will either be exposed or decrease their chances.

For example, Alice could just pick a prime n and send it to Bob. She will be found out, of course, since Bob is supposed to check that n is not a prime. If he skipped the check, but otherwise followed the protocol, there would now only be

two square roots of y modulo n, namely, $\pm x$. Thus Alice would send one of $\pm x$ to Bob, and she would certainly win.

Another possibility for Alice might be to pick n a product of more than two prime factors. But then y will have even more square roots modulo n, so she has less than an even chance of picking one of $\pm x$ to send to Bob, and so she actually diminishes her chance of winning.

Finally, Alice could return some number c that is not a square root of y modulo n. We did not include a check for that, but Bob will surely find out, as his attempt to use c and x will fail to produce a nontrivial factor of n.

It appears that Bob's only possible deviation from the protocol is to send a quadratic non-residue y to Alice. But again, Alice will discover this as her attempt at finding a square root will fail.

Finally, it might be noted that the protocol seems quite inefficient, since Alice needs to find new primes for each round played. (She cannot reuse the primes, even after a round won, since Bob might have falsely admitted defeat, perhaps in the hopes of being able to use the information in a future coin toss where the stakes might be higher.) Finding primes requires quite a bit of computation, hence this protocol seems wasteful. Much better, then, for Alice to commit to a bit, then for Bob to bet on either a zero or a one, and for Alice to reveal her commitment. Bob wins if he guessed right, otherwise Alice wins.