In Lectures 2 and 3 we proved various facts about numbers such as: A number is even if and only if its square is even, the square of an odd number has the form $8k+1$, n^3-n is divisible by 6, and so on. There is a type of algebra for numbers which is very helpful for this type of reasoning called modular arithmetic—the arithmetic of remainders.

Let's fix an integer n, which we will call the *modulus*. Division with remainder (Theorem 5 in the last lecture) says that for every integer a there exists a unique integer between 0 and $p-1$ such that $a = qn + r$ for some quotient q. We call r the *remainder* of a *modulo* n—in short "a modulo n"—and we write

$$r = a \bmod q.$$

We can add, subtract, and multiply the numbers 0, 1, up to $q-1$ "modulo q". The arithmetic remains valid as long as we take the remainders of all expressions modulo q. For example, 3 is the remainder of dividing 9 by 6 and 2 is the remainder of dividing 14 by 6. What is the remainder of dividing $9+14$ by 6? As you may expect this remainder is $2+3=5$:

$$(9+15) \bmod 6 = (9 \bmod 6) + (14 \bmod 6) = 3 + 2 = 5.$$

How about the remainder of dividing $9+16$ by 6? As $9 \bmod 6 = 3$ and $16 \bmod 6 = 4$, the remainder should be determined by $3+4=7$. But 7 is bigger than 6 so to obtain a legitimate remainder we have to further take the remainder of dividing 7 by 6 which is 1:

$$(9+17) \bmod 6 = ((9 \bmod 6) + (16 \bmod 6)) \bmod 6) = (3+4) \bmod 6 = 7 \bmod 6 = 1.$$

The rule for adding remainders modulo q is:

$$(a+b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n. \tag{1}$$

This strategy for calculating remainders of sums works even if the numbers are negative. For example:

$$(-11+27) \bmod 5 = ((-11 \bmod 5) + (27 \bmod 5)) \bmod 5 = (4+2) \bmod 5 = 6 \bmod 5 = 1.$$

Why does this work? The reason is that adding or subtracting multiples of n from a does not change the value of $a \bmod n$. Conceptually it helps to think of $9, 3, -3$ etc. as "equivalent" to one another modulo 6, with 3 playing a special role as the unique representative of those numbers whose value falls between 0 and 5. There is a special notation for this:

$$\boxed{\text{We say that } a \text{ and } a' \text{ are } \textit{congruent} \text{ modulo } n \text{ if } n \text{ divides } a-a'. \text{ We write } a \equiv b \pmod{n}.}$$

An equivalent way of saying "n divides $a-a'$" is "$a \bmod n = a' \bmod n$". You can prove this as an exercise.

Thus $7 \equiv 22 \pmod 5$ and $9 \equiv -3 \pmod 6$. We use $a \bmod n$ to represent the *operation* of calculating the remainder when a is divided by n. In contrast $a \equiv a' \bmod n$ stands for the *predicate* "a and a' have the same remainder when divided by n."

Congruences are useful because they are friendly with the usual arithmetic operations:

$$\text{If} \quad a \equiv a' \pmod{n} \quad \text{and} \quad b \equiv b' \pmod{n}, \qquad \text{then} \quad a+b \equiv a'+b' \pmod{n} \tag{2}$$

$$\text{and} \quad a \cdot b \equiv a' \cdot b' \pmod{n}. \tag{3}$$

Proof of (2) *and* (3). By assumption, n divides both $a-a'$ and $b-b'$. Thus n must divide any combination of them. $(a+b)-(a'+b')$ is one such combination, proving (2). $ab-a'b'$ is also a combination of them because $ab-a'b' = a(b-b') + b'(a-a')$. This proves (3). □

There equations are powerful. Let us reprove some familiar theorems in this notation.

Theorem 1. *n is even if and only if n^2 is even.*

In modular notation, "n is even" translates to "$n \equiv 0 \bmod 2$", and n is odd translates to "$n \equiv 1 \bmod 2$".

Proof. Any number n is congruent to 0 or 1 modulo 2. If $n \equiv 0 \pmod 2$ then $n^2 \equiv 0^2 = 0 \pmod 2$ so n^2 is also even. If $n \equiv 1$ then $n^2 \equiv 1^2 = 1 \pmod 2$ so n^2 is also odd. □

Theorem 2. *The square of an odd number has the form $8k + 1$ for some k.*

Proof. If n is odd then n must be congruent to 1, 3, 5, or 7 modulo 8. As $5 \equiv -3$ and $7 \equiv -1$, n must be congruent to ± 1 or ± 3. Therefore n^2 is congruent to $(\pm 1)^2 = 1$ or $(\pm 3)^2 = 9 \equiv 1$. In all cases $n^2 \equiv 1 \pmod 8$. □

1 Modular multiplication and modular exponentiation

Rule (1) tells us how to add numbers modulo n. It also tells us how to subtract: Even if b happens to be negative, $b \bmod n$ is a remainder therefore positive. For example

$$13 - 27 \bmod 11 = (13 \bmod 11 + (-27) \bmod 11) \bmod 11 = 2 + 6 \bmod 11 = 8.$$

Multiplication is no different: As any number is congruent to its remainder modulo n (in symbols, $a \equiv (a \bmod n) \pmod n$), from (3) we get that the remainder of the product ab modulo n is the product of the remainders:

$$(ab \bmod n) \equiv (a \bmod n)(b \bmod n) \pmod n$$

so we can calculate ab modulo n by first taking the remainders of a and b modulo n, then multiplying them, and taking the remainder modulo n again:

$$(ab \bmod n) = ((a \bmod n) \cdot (b \bmod n)) \bmod n. \tag{4}$$

For example,

$$(13 \cdot 24) \bmod 7 = ((13 \bmod 7) \cdot (24 \bmod 7)) \bmod 7 = 6 \cdot 3 \bmod 7 = 18 \bmod 7 = 4.$$

Of course we could have also calculated $13 \cdot 24$ modulo 7 by first multiplying them out to obtain 312 and then taking the remainder of division by 7 which is 4 ($312 = 44 \cdot 7 + 4$). The advantage of using (4) is that it is faster. This is especially valuable when we have to multiply many times, namely for *modular exponentiation*.

Suppose we want to calculate $19^{11} \bmod 21$. We could first calculate 19^{11}, namely multiply 19 by itself 11 times to obtain 19, 361, 6859, 130321, 2476099, 47045881, 893871739, 16983563041, 322687697779, 6131066257801, and finally $19^{11} = 116490258898219$. We would then divide 19^{11} by 21 to obtain quotient 5547155185629 and remainder 10.

If we use (4) instead the intermediate values are more manageable–they are all between 0 and 20. The sequence is now

$$\begin{aligned} 19^2 \bmod 21 &= 361 \bmod 21 = 4 \\ 19^3 \bmod 21 &= 4 \cdot 19 \bmod 21 = 76 \bmod 21 = 13, \end{aligned}$$

then 16, 10, 1, 19, 4, 13, 16, and finally

$$19^{11} \bmod 21 = 13 \cdot 19 \bmod 21 = 247 \bmod 21 = 10.$$

The numbers never get too large. As we are only ever multiplying remainders modulo 21, the product of two such numbers cannot exceed 400. Generally, in the course of calculating a^e modulo n using this method we never have to deal with numbers larger than $(n-1)^2$.

There is an even faster way to do modular powering. Suppose we want to calculate $7^{64} \bmod 87$. With the method we just describe we still need to multiply 7 by itself 64 times (even if we get to simplify our intermediate results). It turns out that we can get away with only six multiplications because $64 = 2^6$:

$$\begin{aligned}
7^{64} \bmod 87 &= (((((7^2)^2)^2)^2)^2)^2 \bmod 87 \\
&= ((((49^2)^2)^2)^2)^2 \bmod 87 \\
&= (((52^2)^2)^2)^2 \bmod 87 && \text{because } 49^2 \bmod 87 = 52 \\
&= ((7^2)^2)^2 \bmod 87 && \text{because } 52^2 \bmod 87 = 7 \\
&= (49^2)^2 \bmod 87 \\
&= 52^2 \bmod 87 \\
&= 7.
\end{aligned}$$

You may object that we got lucky in this example because the exponent is a power of two. It turns out that it can be modified to work for any exponent e:

Fast modular exponentiation algorithm $Power(a, e, n)$:

```
    If $e = 1$ return $a \bmod n$
    Otherwise,
        If $a$ is even return $Power(a^2 \bmod n, e/2, n)$
        If $a$ is odd return $a \cdot Power(a, e-1, n) \bmod n$.
```

Theorem 3. *$Power(a, e, n)$ always terminates and outputs $a^e \bmod n$.*

Let's use this algorithm to calculate $8^{13} \bmod 17$:

$$\begin{aligned}
Power(8, 13, 17) &= 8 \cdot Power(8, 12, 17) \bmod 17 && \text{because 13 is odd} \\
&= 8 \cdot Power(13, 6, 17) \bmod 17 && \text{because } 8^2 \bmod 17 = 13 \\
&= 8 \cdot Power(16, 3, 17) \bmod 17 && \text{because } 13^2 \bmod 17 = 16 \\
&= 8 \cdot (16 \cdot Power(16, 2, 17) \bmod 17) \bmod 17 && \text{because 3 is odd} \\
&= 8 \cdot (16 \cdot Power(1, 1, 17) \bmod 17) \bmod 17 && \text{because } 16^2 \bmod 17 = 1 \\
&= 8 \cdot (16 \cdot 1 \bmod 17) \bmod 17 \\
&= 128 \bmod 17 \\
&= 9.
\end{aligned}$$

The proof of Theorem 3 is a good exercise.

2 Division modulo a prime

It may sometimes happen that two nonzero numbers multiply to zero in modular arithmetic, for example $6 \cdot 10 \bmod 15$ equals zero. This is a problem for division, because we cannot take an equation like $6 \cdot 10 \equiv 0 \cdot 10 \pmod{15}$, divide both sides by 10 and get a meaningful answer.

Fortunately this does not happen when the modulus n is a prime number p. First, let's show how to calculate *multiplicative inverses*, namely ratios of the form $1/a$, more commonly written as a^{-1}. As usual, division by zero is forbidden. If a is nonzero, division is *well-defined*:

Lemma 4. *Assume p is prime. For every a between 1 and $p-1$ there exists a unique b between 1 and $p-1$ such that $ab \equiv 1 \pmod{p}$.*

Proof of existence. First we show existence of b. Take any a between 1 and $p-1$. Since p is prime, $\gcd(x,p)=1$. Therefore 1 can be written as a combination of a and p:

$$s \cdot a + t \cdot p = 1.$$

Taking both sides modulo p, we get $s \cdot a \equiv 1 \pmod{p}$. Take $b = s \bmod p$. Then $a \cdot b \equiv s \cdot a \equiv 1 \pmod{p}$. □

Proof of uniqueness. Now we show uniqueness. If $ab \equiv 1 \pmod{p}$ and $ab' \equiv 1 \pmod{p}$ then

$$b' \equiv b' \cdot 1 \equiv b' \cdot (ab) \equiv b \cdot (ab') \equiv b \cdot 1 \equiv b \pmod{p},$$

so p must divide $b'-b$. Since $b'-b$ is between $-(p-2)$ and $p-2$, this is impossible unless $b'=b$. □

The proof of existence tells us how to calculate $b = a^{-1}$: First, use the Extended Euclid's algorithm to find s and t such that $s \cdot a + t \cdot p = 1$. Then output $s \bmod p$. For example, to get the multiplicative inverse of 11 modulo 17, I run $X(11,17)$ to get $(-3,2)$, namely $(-3)\cdot 11 + 2 \cdot 17 = 1$. Therefore

$$11^{-1} \bmod 17 = -3 \bmod 17 = 14.$$

To divide two numbers, we first take the multiplicative inverse of the denominator, then multiply by the numerator. For example, to divide 3 by 11 modulo 17, we calculate

$$3 \cdot 11^{-1} \bmod 17 = 3 \cdot 14 \bmod 17 = 42 \bmod 17 = 8.$$

Solving modular equations Using the rules for the basic operations we can solve systems of equations in modular arithmetic just like in ordinary arithmetic. Suppose we want to find x and y that satisfy

$$\begin{aligned} 2x + 3y &\equiv 1 \pmod{7} \\ 4x - 5y &\equiv -3 \pmod{7}. \end{aligned}$$

We start by eliminating one of the variables. To eliminate x, we multiply the first equation by 2 and subtract it from the second one to get

$$-5y - 2 \cdot 3y \equiv -3 - 2 \cdot 1 \pmod{7}$$

which simplifies to $-11y \equiv -5 \pmod{7}$, or $3y \equiv 2 \pmod{7}$. Therefore $y \equiv 2 \cdot 3^{-1} \pmod{7}$. To find $3^{-1} \pmod{7}$ we can use Euclid's extended algorithm. Here the numbers are small enough that we can figure out 3^{-1} by hand. Since $3 \cdot (-2) \equiv -6 \equiv 1 \pmod{7}$ we know that the inverse of 3 must be $-2 \equiv 5$, so $y \equiv 2 \cdot 5 \equiv 3 \pmod{7}$. We can now go back to the first equation to solve for x. $2x \equiv 1 - 3y = 1 - 3 \cdot 3 \equiv 6 \pmod{7}$, so $x \equiv 2^{-1} \cdot 6 \equiv 4 \cdot 6 \equiv 3 \pmod{7}$. You should now check that $x = 3, y = 3$ is indeed a valid solution.

3 Encryption

Encryption is a type of mechanism that allows Alice and Bob to exchange private messages in a public environment, like a cellular network, or the internet.

Here is a simple protocol for doing this called the *one-time pad.*[1] Say Alice wants to send Bob the message "`Charlie is a spy`". This message is represented as a large number, say by replacing each

[1] The reason for this name is that the secret key k cannot be reused.

character with some standard numerical representation like its ASCII encoding in which `C` $= 067$, `h` $= 104$, etc:

$$m = \texttt{067104097114108105101032105115032097032115112121}$$

Alice and Bob have agreed ahead of time on a number n, which is at least as large as the message m to be encoded. In this example, $n = 10^{50}$ would suffice as m has fewer than 50 digits. Moreover, we assume that they had met ahead of time and exchanged in secret a random number k between 0 and $p - 1$ called the *secret key*.

If Alice wants to send Bob the secret message m, she *encrypts* m as $c = m + k \bmod n$ and posts the *ciphertext* c in public. To recover Alice's message, Bob calculates $c - k \bmod n$.

I will now argue that the content of Alice's message remains secret with respect to any observer Eve who sees the ciphertext but does not know the secret key. This requires a bit of elementary probability. Let's assume that all the numbers in the range $0, 1, \dots, n - 1$ were equally likely to be chosen as the secret key k. Then, regardless of what the message m is, the ciphertext $m + k \bmod n$ observed by Eve is also equally likely to take any of the values $0, 1, \dots, n - 1$. For example, if $m = 1$ and $n = 4$, then k is equally likely to take any of the values 0, 1, 2, and 3, so $m + k \bmod n$ is equally likely to evaluate to 1, 2, 3, and 0. Therefore the value observed by Eve is completely independent of the message sent by Alice, and Eve obtains no information about the message.

This type of encryption works as long as Alice and Bob have previously met to exchange the secret key k (and kept it secure in the meantime). Such an assumption is unreasonable if, say, Alice is a customer in Bob's online store and she wants to send him her credit card number in secret. Alice and Bob could be living on opposite sides of the world and might have never talked to one another before. How can they communicate secretly in such conditions?

Public-key encryption

Rivest, Shamir, and Adleman came up with an ingenious proposal known as RSA in 1977. In the one-time pad the same secret key k is held by both Bob and Alice: Encryption and decryption are *symmetric* operations. In contrast, the RSA scheme has two keys: A *private key* that is only known to the receiver—in our case, Bob—and a *public key* that is known to the whole world. The scheme is designed so that anyone can encrypt messages for Bob using his public key, but only Bob can decrypt them; decryption requires knowledge of the private key.

We will first describe a simplified variant of the RSA scheme that is *not* secure but contains almost all of their ideas. Let n be a large prime number, typically a few hundred digits. The message m to be sent by Alice is represented as a number between 0 and $n - 1$. Bob's public key e and secret key d are two numbers between 2 and $n - 1$ with a mysterious property:

$$x^{ed} \equiv x \pmod{n} \qquad \text{for every } x. \tag{5}$$

To encrypt message m, Alice calculated the *ciphertext* $c = m^e \bmod n$ and sends it to Bob. Notice that this requires knowing Bob's public key e but not his private key. To decrypt, Bob calculates $c^d \bmod n$. By 5, it must be that $c^d = (m^e)^d = m^{ed} \equiv m \pmod{n}$ so Bob does recover the message m that Alice intended for him.

There is a theorem that looks much like equation (5) called Fermat's little theorem:

Theorem 5. *Assume n is prime. Then $x^n \equiv x \pmod{n}$ for every x.*

Therefore equation (5) will hold as long as $ed = n$. But n is a prime number so this doesn't provide any choices for e and d. Fortunately, the requirement $ed = n$ turns out to be too strong. It is sufficient that ed be congruent to 1 modulo $n - 1$. In the special case when $ed = n$ this is certainly true, but there are now many more possibilities for choosing the keys.

Lemma 6. *If n is prime and $ed \equiv 1 \pmod{n-1}$, then $x^{ed} \equiv x \pmod{n}$.*

Proof. If $x \equiv 0 \bmod n$ then both sides of the congruence are zero. Now assume $x \not\equiv 0$. By assumption, $ed = k(n-1)+1$ for some integer k. Then

$$x^{ed} = x^{k(n-1)+1} = (x^n)^k \cdot x^{-k+1} \equiv x^k \cdot x^{-k+1} \equiv x. \qquad \square$$

Although this scheme looks very nice it turns out it is not secure. The reason is that Bob's "secret key d" is not secret at all! If Eve knows his public key e she can *find* a number e for which $ed \equiv 1 \pmod{n-1}$. She can accomplish this by running extended Euclid's algorithm on inputs $n-1$ and d. If d and $n-1$ happen to be coprime (which will happen for "typically" chosen d and n), that is $\gcd(d, n-1) = 1$, the algorithm will output e and k for which $ed + k(n-1) = 1$, thereby compromising Bob's secret key.

Proof of Theorem 5 (optional). Since multiplication is invertible modulo n, multiplying all numbers in the sequence $s = (1, 2, \dots, n-1)$ by x modulo n produces a *permutation* of this sequence. Namely, each number between 1 and $n-1$ is represented exactly once in the sequence $t = (1x, 2x, \dots, (n-1)x)$ (everything modulo n). So taking the product of all the elements of s and all the elements of t should produce the same number modulo n. The product of the elements of s is $(n-1)!$, while the product of the elements of t is $(n-1)! \cdot x^{n-1}$. Therefore

$$(n-1)! \cdot x^{n-1} \equiv (n-1)! \pmod{n}.$$

Dividing both sides by $(n-1)!$ we obtain that $x^{n-1} \equiv 1 \pmod{n}$, so $x^n \equiv x \pmod{n}$. $\square$

4 The RSA encryption scheme

To turn this into a secure proposal we need to, at the very least, make it difficult for eve to solve the equation $xd \equiv 1 \pmod{n-1}$. Rivest, Shamir, and Adleman accomplish this by making the modulus n a *composite* number. The simplest type of composite number is a product $n = pq$ of two primes p and q. This will suffice for the RSA cryptosystem.

RSA encryption.
Bob (the receiver):
 chooses two large primes p and q and sets $n = pq$.
 chooses his public key e and private key d so that $ed \equiv 1 \mod (p-1)(q-1)$.
 sends his public key d together with the modulus n to everyone,
 but keeps his private key close to his heart.
Alice (the sender): To send message m:
 calculates the ciphertext $c = m^e$ and announces it to everyone, including Bob.
Bob (the receiver): After hearing that Alice broadcast ciphertext c intended for him,
 calculates c^d which will equal Alice's message m.

For Bob's decryption to be correct it better be the case that $c^d = (m^e)^d$ equals m. This is a consequence of another famous theorem, this one due to Euler. Fermat's little theorem is a special case of Euler's theorem. For the correctness of RSA decryption we will need another special case:

Theorem 7. *If n is a product of two distinct primes p and q and $\gcd(x, n) = 1$ then $x^{(p-1)(q-1)} \equiv 1 \pmod{n}$.*

Armed with Theorem 7 we can reason like this. Since Bob's chosen keys satisfy $ed \equiv 1 \mod (p-1)(q-1)$ it must be that $ed = k(p-1)(q-1)+1$ for some k. Then[2]

$$(m^e)^d \equiv m^{ed} \equiv m^{k(p-1)(q-1)+1} = m^{(p-1)(q-1)} \cdot m \equiv 1 \cdot m = 1 \pmod{n}.$$

[2]This works only when m is coprime to n. $m^{ed} \equiv m \bmod n$ even for m that are not coprime to n but we won't prove it.

Let's work out an example. Suppose Bob picks the prime numbers

$$p = \texttt{90519402977290531385367802684 1}$$
$$q = \texttt{77264704393305770992496152962 1}$$

Their product is the number

$$n = pq = \texttt{6993954912898875207256857389135555764742010771526884545572 61}$$

Bob then sets his public key e to `65537`. To find his private key, he runs extended Euclid's algorithm on inputs e and

$$(p-1)(q-1) = \texttt{6993954912898875207256857389118777354004951141289098150008 00}$$

to obtain

$$d = \texttt{5882917413498686157060004584167137958967864513400107185518 73}$$

You can verify on the computer that indeed $ed \equiv 1 \mod (p-1)(q-1)$.

Alice now wants to encrypt (the ASCII encoding of) "`Charlie is a spy`". This message falls into the range between 0 and $n-1$ so she broadcasts the ciphertext

$$c = m^e \bmod n = \texttt{4828702969024855483580801448399810063661147451778527390153 59}.$$

After receiving c, to decrypt it Bob calculates

$$c^d \bmod n = \texttt{67104097114108105101032105115032097032115112121}.$$

This is the same message that Alice sent.

Why is RSA secure while the "baby RSA" we described in Section 3 wasn't? Eve can still attempt to learn Bob's public key by solving the equation $xd \equiv 1 \mod (p-1)(q-1)$ for x. To solve this equation she will likely want to run extended Euclid's algorithm on inputs $(p-1)(q-1)$ and d. The difference is that she doesn't know the value of $(p-1)(q-1)$.[3] Essentially the only way she can calculate $(p-1)(q-1)$ from $n = pq$ is by first factoring n, subtracting one from each factor, and multiplying those numbers.[4] But factoring products of hundred-digit prime numbers is hard!

In our description of RSA we left several things underspecified. How should Bob exactly go about picking the prime numbers p and q and the keys e and d? These choices are important for the security of RSA but they are beyond the scope of this course. Even the encoding of messages by numbers we described is not quite right. For RSA to be really secure it is important that the message encoding also contains some random numbers. If you are interested in learning more take a look at the reference below, or even better—take a course in cryptography!

As a final point, nobody in fact knows how to *prove* that RSA is secure even though it has withstood numerous attempts at cracking it. Even if it is broken one day, there are alternative public-key encryption schemes that could replace it in e-commerce and virtually all other uses.

5 Discrete logarithms and key exchange*

We describe another route to secure communication which was proposed by Whitfield Diffie and Martin Hellman in 1976, the year before RSA was invented. To enable secure communication, it is enough for Alice

[3] Notice how in our example $(p-1)(q-1)$ has the same leading digits as $n = pq$ but the two numbers start to diverge around the middle. Even though knowing n provides some hint about the value of $(p-1)(q-1)$, calculating it exactly is a different matter.

[4] It can be proved that if you can calculate $(p-1)(q-1)$ from $n = pq$ in *any way*, then you can factor n with just a little extra work. Do you see why?

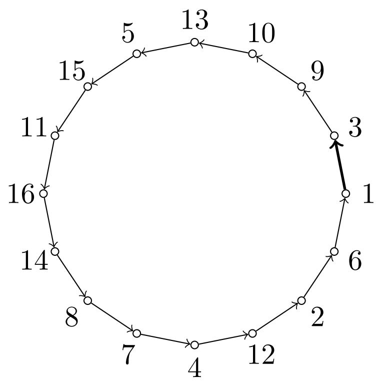
and Bob to exchange a *secret key* that they can then use to instantiate the one-time pad. The challenge is that this secret key must be established over a public communication channel where Eve can observe all messages exchanged between them.

When we work with real numbers, the operation inverse to powering is taking logarithms. By analogy, we call an integer x a *discrete logarithm* of y in base b modulo p if $b^x \bmod p = y$. Unlike ordinary logarithms, discrete logarithms do not behave well at all: Some numbers may have multiple logarithms, and some may have none at all. In these notes we will focus on the situation when p is a prime number, which is already complicated enough.

Let's take the modulus $p = 17$ as an example. Then 10 has no discrete logarithm in base 9. On the other hand, 13 has (at least) two base 9 logarithms as both 9^2 and 9^{10} equal 13 modulo 17. The key to understanding the discrete logarithm is the following theorem which we won't prove:

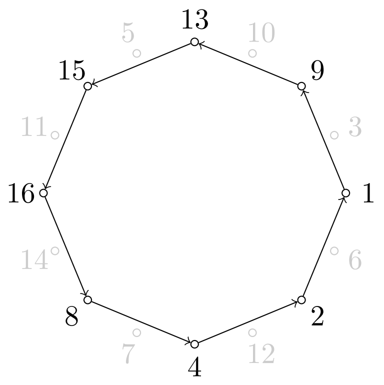
Theorem 8. *For every prime number p there exists a number g between 1 and $p-1$ such that the sequence $g^0, g^1, \ldots, g^{p-2}$ covers every nonzero number modulo p exactly once.*

Such a number g is called a *primitive root* modulo p. For example, $g = 3$ is a primitive root modulo 17. In the following diagram, arrows indicate multiplication by 3. You can see that the sequence $1, 3, 3^2, \ldots, 3^{15}$ covers every nonzero number modulo 17 exactly once.



Notice also that $3^{16} \equiv 1 \pmod{17}$, so calculating $3^x \bmod 17$ for an arbitrary x reduces to calculating $3^{x \bmod 16} \pmod{17}$ for every x. This is a special case of Fermat's little Theorem 5. In particular, the theorem tells us that a discrete logarithm can always be represented as a nonzero number modulo $p-1$. Theorem 8 then says that if the base is a primitive root g then every nonzero number modulo $p-1$ has a unique discrete logarithm in base g. In the above diagram, the discrete logarithm of x base 3 (modulo 17) is the number of steps it takes to reach x starting at 1.

If the base b is not a primitive root, then the powers of b will not cover all the nonzero numbers modulo p, so not every number will have a discrete logarithm in base b. For example, if $b = 9$ then the numbers that have discrete logarithms modulo 17 are the ones reachable from 1 in the following diagram (where the arrows indicate multiplication by 9):



Notice also that $9^8 \bmod 17 = 1$, so the value of the base 9 discrete logarithm (if it exists) reduces to a unique value between 0 and 7, and not between 0 and 15 as was the case in base 3. This is no accident as $9 = 3^2$ so the powers of 9 are the even powers of 3 modulo 17.

Calculating discrete logarithms Let us now assume that the base is a primitive root g modulo p and see how we could go about calculating the discrete logarithm of some input y. One way to do so is to look for the first occurrence of y in the sequence $1, g, g^2, \dots, g^{p-2}$ modulo p and output the index of this occurrence. This works fine when p is of moderate size, but would be unimaginably slow when p is, say, on the order of 10^{50}. Is there a better way?

One possible strategy would be to begin by looking for some partial information about the discrete logarithm of y — for example, is it even or is it odd? The following theorem tells us that this information can be extracted by calculating a suitable power of g:

Theorem 9. *Assume $p > 2$ is a prime number, g is a primitive root modulo p, and y is a nonzero number modulo p. Then the discrete logarithm of y in base g modulo p is even if and only if $y^{(p-1)/2} \equiv 1 \bmod p$.*

Proof. We prove the theorem by cases depending on the parity of the discrete logarithm of y. If it is even, then $y \equiv g^{2n} \pmod{p}$ for some k, and $y^{(p-1)/2} \equiv (g^{p-1})^n \equiv 1 \pmod{p}$ by Theorem 5. If it is odd, then $y \equiv g^{2n+1} \pmod{p}$ for some n, and $y^{(p-1)/2} \equiv (g^{p-1})^n \cdot g^{(p-1)/2} \equiv g^{(p-1)/2} \pmod{p}$ by Theorem 5. By Theorem 8, $g^{(p-1)/2}$ cannot be equal to 1 modulo p. $\square$

Let's illustrate this on our example $p = 17, g = 3$. Theorem 9 says that the discrete logarithm of y in base g is even if and only if $y^8 \equiv 1$. The numbers that satisfy this condition are exactly those that are at an even distance from 1 in the diagram: Moving forward by the same even amount 8 times always brings us back to 1, but if the amount is odd we must end up in a different place.

Once we know if the discrete logarithm of y in base g is even, one natural idea is to try recursion and find out the parity of the higher-order bits of the discrete logarithm n. To develop some intuition about how this strategy might work, let's fix $p = 17, g = 3$, and look for the discrete logarithm of 14. We first calculate $14^8 \bmod 17 = 16$, so the discrete logarithm of 14 must be even. In other words, we now know that $n = 2n' + 1$ and so

$$14 = 3^{2n'+1} \quad \text{for some } n' \text{ between 0 and 7.}$$

We can rewrite this equation as

$$14 \cdot 3^{-1} = (3^2)^{n'} \quad \text{for some } n' \text{ between 0 and 7.}$$

We have now reduced our problem of finding the discrete logarithm of 14 in base 3 to the problem of finding the discrete logarithm of $14 \cdot 3^{-1} \equiv 16$ in base $3^2 = 9$! The number 9 is, unfortunately, no longer a primitive root modulo 17 so we cannot apply Theorem 9 again. We can, however, apply the following *generalization*, which you should be able to prove on your own by applying the same reasoning as in the proof of Theorem 9.

Theorem 10. *Assume $p > 2$ is a prime number, b is a nonzero number modulo p, and k is the smallest positive integer such that $b^k \equiv 1 \pmod{p}$. Assume that k is even and y has a discrete logarithm in base b modulo p. The discrete logarithm is even if and only if $y^{k/2} \equiv 1 \pmod{p}$.*

Theorem 10 tells us that the base 9 discrete logarithm of 16 is even if and only if $16^4 = 1$ modulo 17. This is indeed the case, so we can conclude that n' is even and so we can write $n' = 2n''$, which tells us that

$$16 = 9^{2n''} \quad \text{for some } n'' \text{ between 0 and 3.}$$

So we are left with finding the discrete logarithm n'' of 16 in base $9^2 = 13 \bmod 16$. Continuing in the same manner, we find that the parity of n'' is determined by the value $16^2 \bmod 17$, which again equals 1.

Therefore n'' itself is even, and so we must have $16 = 13^{2n'''}$ for $n''' = n''/2$, which is either zero or one. Now $13^2 \pmod 17 = 16$, so it follows by inspection that $n''' = 1$, from where $n'' = 2$, $n' = 4$, and $n = 9$.

In general, this strategy for finding discrete logarithm is much faster than brute-force search through the list of powers as the range of possible discrete logarithms is halved at each iteration. But were we lucky with our example here, or can it be used in a general context? One seemingly innocuous assumption that we made in Theorem 10 is that the number k, which represents the *period* of the sequence $1, b, b^2, \ldots$ must be even. When the modulus p is of the form $2^t + 1$ (as in our example) then the period of the base will remain even in all iterations because it has initial value 2^t and it is halved in every iteration. Prime numbers of this form are called *Fermat primes*, and indeed discrete logarithms modulo a Fermat prime can be calculating reasonably efficiently using the algorithm that we informally described.

Efficient calculation of discrete logarithms can, more generally, be extended to work modulo any prime p provided that the number $p - 1$ does not contain any "large" prime factors.[5] Many prime numbers p have these property, but there are also many that don't. How large can a prime factor of $p - 1$ be when p is prime? Assuming p is greater than 2, $p - 1$ must be an even number, so its largest prime factor cannot exceed $(p - 1)/2$. A prime number p for which $(p - 1)/2$ is also a prime number are called a *safe prime*. It is not known if the number of safe primes is infinite, but very large examples are known. Wikipedia says that "Finding a 500-digit safe prime, like $2 \cdot (1,416,461,893 + 10^{500}) + 1$, is now quite practical."

Unlike in the case of Fermat primes, it is not known how to calculate discrete logarithms modulo a safe prime in an efficient manner. For a number like the Wikipedia safe prime, the best known algorithms may require 10^{160} or so steps.

Key exchange We are now ready to describe a famous proposal by which Alice and Bob can exchange a key that looks secret to any observer using only public communication channels like the internet. First, Alice and Bob agree (in public) on a safe prime p and choose a primitive roof g modulo p. For example, they could take p to equal the Wikipedia safe prime and g to equal 2 (I checked that 2 is a primitive root on my computer). They then exchange the following public messages:

A: Choose a random nonzero number x modulo p (in private). Send the value $a = g^x$ to Bob.

B: Choose a random nonzero number y modulo p (in private). Send the value $b = g^y$ to Alice.

A: Upon receiving b from Bob, calculate b^x. This is Alice's secret key.

B: Upon receiving a from Alice, calculate a^y. This is Bob's secret key.

At the end of the protocol, Alice and Bob, they will both agree on the same secret key g^{xy}.

Now consider a potential observer Eve that has seen Alice's and Bob's messages. This Eve has seen the values g^x and g^y. Can she use these to gain any knowledge about the secret key g^{xy}? One way for Eve to do this is to calculate the discrete logarithms x and y, in which case she can easily compute the secret key. But calculating discrete logarithms modulo safe primes is very time-consuming.

It is, in fact, not known how to calculate g^{xy} efficiently given only knowledge of the values g^x and g^y. This gives Alice and Bob some confidence that their secret key is indeed secret, even though all their messages were exchanged in public.

The actual protocols used for key exchange and secure communication used on the internet are a bit more complex than the one I described here. One of the aspects in which the protocol I described may be deficient is that even though Eve can conceivably not completely recover Alice's and Bob's secret key, she might still be able to recover some partial information about it that may allow her to intercept future messages exchanged between them.

[5]More precisely, the running time of the discrete logarithm procedure has a linear dependence on the size of the largest prime factor of $p - 1$.

References

This lecture is based on Chapter 8 of the text *Mathematics for Computer Science* by E. Lehman, T. Leighton, and A. Meyer. If you want to know more about public-key encryption and key exchange, you can look at the book *Introduction to Modern Cryptography* by Jonathan Katz and Yehuda Lindell.