

Number theory (Chapter 4)

Review

- Compute $6^{11} \bmod 13$ in an efficient way
- What is the prime factorization of 100? 138?

- What is $\gcd(100, 138)$?
- What is $\text{lcm}(100, 138)$?

- What does it mean for two integers to be “relatively prime”?
- Find the $\gcd(100, 138)$ using the Euclidian Algorithm.

- What is the advantage of using the Euclidian Algorithm instead of the prime factorization to find the \gcd of two numbers?

- We then wanted to get to four results. We did first, though we started on the last one.
 - **Bezout’s theorem** which states that $\forall a, b \exists s, t \text{ } sa + bt = \gcd(a, b)$
 - **The definition of an inverse of a modulo m and a proof that it exists if a and m are relatively prime** That is, that $\forall a \forall (m > 1) [\gcd(a, m) = 1 \rightarrow \exists x (ax \equiv 1 \pmod{m})]$
 - **Fermat’s Little Theorem** which states if p is prime and a is not divisible by p then $a^p \equiv 1 \pmod{p}$
 - **Chinese remainder theorem** which states that if you’ve a group of relatively prime positive integers greater than 1 then you can count to the product of those primes in a unique way just using those primes (this one is actually easy, just hard to state succinctly).

BÉZOUT'S THEOREM If a and b are positive integers, then there exist integers s and t such that $\gcd(a, b) = sa + tb$.

At first glance, this seems quite reasonable, after all, for any a, b , can't we find integers s and t that are equal to *any* number? And the answer is no. Consider $a=2$ and $b=4$. You can find integers that get to any even number, but not any odd. And for those values of a and b , that's exactly what the theory says.

1. Find an s and t for $a=9$ and $b=6$ so that $9s+6t=\gcd(9,6)$.
2. Find an s and t for $a=12$ and $b=28$

The general proof for this is by construction. The construction is done by using the "Extended Euclidean Algorithm"ⁱ. The following example is from the book (page 270) and was done on the board on Tuesday.

Express $\gcd(252, 198) = 18$ as a linear combination of 252 and 198.

Solution: To show that $\gcd(252, 198) = 18$, the Euclidean algorithm uses these divisions:

$$\begin{aligned} 252 &= 1 \cdot 198 + 54 \\ 198 &= 3 \cdot 54 + 36 \\ 54 &= 1 \cdot 36 + 18 \\ 36 &= 2 \cdot 18. \end{aligned}$$

Using the next-to-last division (the third division), we can express $\gcd(252, 198) = 18$ as a linear combination of 54 and 36. We find that

$$18 = 54 - 1 \cdot 36.$$

The second division tells us that

$$36 = 198 - 3 \cdot 54.$$

Substituting this expression for 36 into the previous equation, we can express 18 as a linear combination of 54 and 198. We have


$$18 = 54 - 1 \cdot 36 = 54 - 1 \cdot (198 - 3 \cdot 54) = 4 \cdot 54 - 1 \cdot 198.$$

The first division tells us that

$$54 = 252 - 1 \cdot 198.$$

Substituting this expression for 54 into the previous equation, we can express 18 as a linear combination of 252 and 198. We conclude that

$$18 = 4 \cdot (252 - 1 \cdot 198) - 1 \cdot 198 = 4 \cdot 252 - 5 \cdot 198,$$

completing the solution. 

ⁱ Example and some text from https://en.wikipedia.org/wiki/Extended_Euclidean_algorithm (also a proof).

Fermat's Little Theorem (4.4)

FERMAT'S LITTLE THEOREM If p is prime and a is an integer not divisible by p , then

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

Furthermore, for every integer a we have

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

We'll use this one without proof as all known proofs are fairly ugly. But see https://en.wikipedia.org/wiki/Proofs_of_Fermat%27s_little_theorem for some proofs if you are interested.

1. Use Fermat's Little Theorem to find $7^{222} \pmod{11}$.

Chinese Remainder Theorem (4.4)

THE CHINESE REMAINDER THEOREM Let m_1, m_2, \dots, m_n be pairwise relatively prime positive integers greater than one and a_1, a_2, \dots, a_n arbitrary integers. Then the system

$$x \equiv a_1 \pmod{m_1},$$

$$x \equiv a_2 \pmod{m_2},$$

.

.

.

$$x \equiv a_n \pmod{m_n}$$

has a unique solution modulo $m = m_1 m_2 \cdots m_n$. (That is, there is a solution x with $0 \leq x < m$, and all other solutions are congruent modulo m to this solution.)

These looks complex, but really it isn't. Tuesday we did a group exercise and had one group be "mod 2" one group be "mod 3" and one group be "mod 5". This theorem says that if we count up to 30 ($2 \cdot 3 \cdot 5$) and each group counts by their mod (so mod 2 counts as 0,1,0,1, etc.) then we can count from 0 to 29 before there is a repeat.

We'll prove this a bit differently than the text does. We are trying to show that there is a unique solution. First let's define this scheme as a function f that maps from a domain m to a co-domain of $m_1 \times m_2 \times m_3 \dots \times m_n$ using the notation found in the above definition of the problem. Notice that the cardinality of the domain and co-domain are identical (m). Now let's assume there are two values of m , a and b , that generate the same values in the co-domain. In that case, $a-b$ must each be divisible by all values of m_i . And as such, since each of the m_i 's are relatively prime, it must be divisible by their product. But that's impossible as $|a-b| < m$ as a and b are both between 0 and $m-1$. Thus there are no two that have the same mapping (the function is one-to-one). And because they have the same cardinality, the function is also onto. It is thus a bijection and every instance in the domain maps to a unique instance in the co-domain. Done.

1. If we are using the values 2, 3, and 5 for m_1, m_2, m_3 , what values of "a" would $x=6$ generate?
2. If we are using those same values, what is x if $a_1=1, a_2=2$, and $a_3=1$?

Cryptography (4.6)

(Notice, we have skipped 4.5) (Also the next couple of paragraphs are from Rosen)

One of the earliest known uses of cryptography was by Julius Caesar. He made messages secret by shifting each letter three letters forward in the alphabet (sending the last three letters of the alphabet to the first three). For instance, using this scheme the letter B is sent to E and the letter X is sent to A. This is an example of **encryption**, that is, the process of making a message secret.

To express Caesar's encryption process mathematically, first replace each letter by an element of \mathbf{Z}_{26} , that is, an integer from 0 to 25 equal to one less than its position in the alphabet. For example, replace A by 0, K by 10, and Z by 25. Caesar's encryption method can be represented by the function f that assigns to the nonnegative integer p , $p \leq 25$, the integer $f(p)$ in the set $\{0, 1, 2, \dots, 25\}$ with

$$f(p) = (p + 3) \bmod 26.$$

In the encrypted version of the message, the letter represented by p is replaced with the letter represented by $(p + 3) \bmod 26$.

1. Encode "MACY" using the above cypher.

Obviously, this type of cypher isn't very hard to break. Even if you don't know the exact shift being used (3 in this case), it's not hard to walk all the options. And even if you just use an arbitrary mapping from one code to the next, this type of "substitution" cypher can be broken by attacking it by looking at letter frequency (some letters occur at a greater rate than others).

So what is often used are block ciphers. These work on larger blocks of letters. One possible scheme would be to take the letters, treat them as numbers (A=0, B=1, etc.), write those numbers in binary and concatenate them and then XOR that with a rather large number. That number becomes the "key" and if the number is large enough it can be very difficult to break the cipher without having the key.

Let's work a simple example. Let's encode letters in the same way, but then break the message into groups of two letters and multiply each of those pairs to create a single number. Then convert to base 2 and do a bitwise XOR with the value 1001100111 (why did I chose a 10-bit number?). Let's do this with MACY again.

It isn't obvious how to decrypt this without knowing the key. For real encryption we use much larger values for our key so that searching all possibilities isn't viable.

Public Key Cryptography

One really big issue is how do you share a key with someone? If you have a secure way of communicating available, you may not need any cryptography at all. And if you don't have a secure communication channel, you may have problems sharing a key with the intended recipient in a private way.

So... A "holy grail" of cryptography would be if two people (generally Alice and Bob) could communicate so no eavesdropper, generally "Eve" or "the attacker", can read their communication even if Eve can see all the bits sent. How could we do that?

Imagine we had a system where you used a different key for encryption and decryption. Alice generates a key for encrypting, which she shares with the world (called a public key). She keeps the decryption key private. Now Bob can encode a message to her using the public key and only Alice can read it. Someone trying to eavesdrop on the message could not do so.

Of course, all this relies on the public key not giving away enough information that it's easy to figure out the private key. That is, that knowing the *encryption* key doesn't let you easily figure out how to *decrypt* the message. That seems weird—I mean if you know how to generate a message, shouldn't you know how to read it?

RSA

In RSA the public key is found by creating an integer "n" which is the product of two large primes (often 200 digits or so) called p and q. We then pick fairly small number, "e", which is relatively prime to $(p-1)(q-1)$. "n" is the public key and is freely shared, as is "e" (in fact many encryption schemes use a fixed e, $2^{16}+1$ is popular for a number of reasons and if it happens that $(p-1)(q-1)$ isn't relatively prime to e, we just find a new p and/or q).

Messages are broken into blocks (in our example above, we used blocks of two letters) and are encoded as $C=M^e \bmod n$, where "M" is a given block of the message.

Notice that we will want to have a fast scheme for computing that value. And thankfully we have developed a scheme for fast modular exponentiation (recall we did an example at the start of class!).

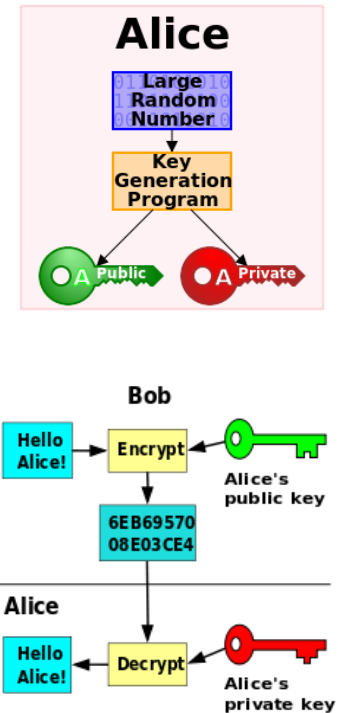


Figure 1: Public and Private Keys.
From Wikipedia

The private key “d” is the inverse of “e” mod $(p-1)(q-1)$. We’ve shown such an inverse exists if e is relatively prime to $(p-1)(q-1)$. And, using the EEA, we know how to find it fairly efficiently.

Why does that help?

Note that if $de \equiv 1 \pmod{(p-1)(q-1)}$ then there exists a k such that $de = 1 + k(p-1)(q-1)$. Let the encoded message be “C” and the plaintext (original) message be M.

$$C^d \equiv (M^e)^d = M^{de} = M^{1+k(p-1)(q-1)} \pmod{n}$$

From Fermat’s Little Theorem it follows that $M^{p-1} \equiv 1 \pmod{p}$ and $M^{q-1} \equiv 1 \pmod{q}$.ⁱⁱ Thus:

$$C^d \equiv M \cdot (M^{p-1})^{k(q-1)} \equiv M \cdot 1 = M \pmod{p}$$

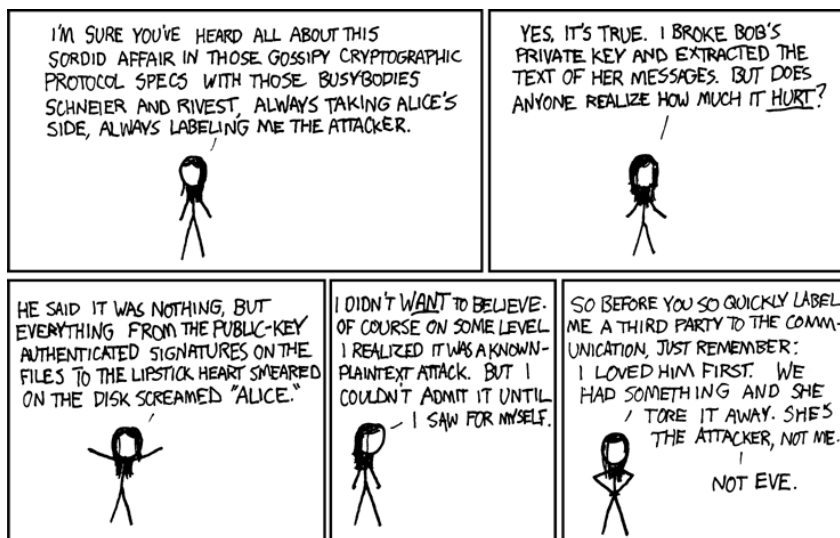
$$C^d \equiv M \cdot (M^{q-1})^{k(p-1)} \equiv M \cdot 1 = M \pmod{q}$$

The Chinese Remainder Theorem (or a bit of common sense) then leads us to

$$C^d \equiv M \pmod{pq}.$$

Comment

The “trick” here is that if we know p and q we can find d. Because d is the inverse of e mod $(p-1)(q-1)$. But if we don’t know q and q, we can’t find d. So if someone wants to find d, it would seem that they’d need to factor n. But factoring a number that is composed of two 200-digit primes is, in the general case, extremely difficult (billions of years as of 2010 estimates and algorithms). So the scheme is relying on the intractability of factoring large numbers!



ⁱⁱ This relies on M being relatively prime to p and q. If it's not, things get a bit ugly, but solvable. Problem 28 in 4.6 addresses the issue if you want to look into it.