Computing exponents modulo a number: Repeated squaring

- How do you compute $(1415)^{13} \bmod 2537 = 2182$ using just a calculator? Or how do you check that $2^{340} \bmod 341 = 1$? You can do this using the method of repeated squaring; we'll illustrate by calculating $(1415)^{13} \bmod 2537$.
- Start by looking at the exponent (13) and represent it as a sum of powers of 2. To do this, find the largest power of 2 less than your exponent; in this case it's $2^3 = 8$. Subtract 8 from 13, getting 5. Then $5 = 4 + 1 = 2^2 + 2^0$. So $13 = 8 + 4 + 1$.
- Now
$$(1415)^{13} \bmod 2537 = (1415)^{(1+4+8)} \bmod 2537.$$
So all I have to do is to calculate the numbers
$$(1415)^1 \bmod 2537, (1415)^4 \bmod 2537, \text{ and } (1415)^8 \bmod 2537$$
and multiply them together, then take the product mod 2537. I already have the first number: 1415. I could raise it to the 4th power and take the result mod 2537, but to be systematic I'll square it, take the result mod 2537, and then repeat that same process.
- $1415^2 \bmod 2537 = 2002225 \bmod 2537 = 532$. So to get $1415^4 \bmod 2537$ I square $532 = 283024$ and take this mod $2537 = 1417$.
- To get $(1415)^8 \bmod 2537$ I square $(1415)^4 \bmod 2537$ and take the result mod 2537. So I square 1417, getting 200789 which mod 2537 is 1122.
- Multiply $1415 \cdot 1417 \cdot 1122$ and take the result mod 2537. The result is 2182.

Correctness of RSA

- Now we are ready to prove that the RSA encryption and decryption functions are inverses of each other. We'll check that $d(e(M)) = M$, where

$$e(M) = M^e \bmod pq, \text{ and } d(C) = C^s \bmod pq,$$

  where s is an inverse of e modulo $(p-1)(q-1)$.

- To do this, we'll need Fermat's Little Theorem, together with a simple lemma:

- **Lemma** *If p and q are distinct primes, $N \equiv M \pmod{p}$, and $N \equiv M \pmod{q}$, then $N \equiv M \pmod{pq}$.*

  Proof: By hypothesis $N - M = jp$ for some j, and $N - M = kq$ for some k. Thus $p \mid kq$. But p is a prime, and cannot divide q, so $p \mid k$. Thus $k = lp$ and so $N - M = lpq$, which is what we need. □

  (The easy way to see this, of course, is to use the uniqueness part of the Chinese Remainder Theorem.)

- We now want to show $d(e(M)) = M$, and writing this out:

$$C^s \bmod pq = M, \text{ where } C = M^e \bmod pq.$$

  To do this we'll show $C^s \equiv M \pmod{p}$ and $C^s \equiv M \pmod{q}$. The conclusion then follows from the lemma.

RSA Correctness, continued

- Our goal is now to show $C^s \equiv M \pmod{p}$, where s is an inverse of e modulo $(p-1)(q-1)$. We also need $C^s \equiv M \pmod{q}$, but the proof will be exactly the same.
- Now
$$C^s = (M^e \bmod pq)^s \equiv M^{es} \pmod{pq}.$$
We claim
$$C^s \equiv M^{es} \bmod p;$$
this is because $C^s - M^{es} = jpq = (jq)p$ for some j.
- What is $M^{es} \bmod p$?
- Since s is an inverse of e modulo $(p-1)(q-1)$, we have $es = 1 + k(p-1)(q-1)$ for some k. Therefore
$$M^{es} = M^{1+k(p-1)(q-1)} = M^1 \cdot (M^{(p-1)})^{k(q-1)}.$$

RSA Correctness, continued

- From last slide

$$M^{es} = M^{1+k(p-1)(q-1)} = M^1 \cdot (M^{(p-1)})^{k(q-1)}.$$

  We are trying to prove $C^s \equiv M \pmod{p}$. On the last slide we showed $C^s \equiv M^{es} \pmod{p}$.

- We take both sides of the top equation mod p, because we want $M^{es} \bmod p$. We get

$$M^{es} \bmod p = ((M^1 \cdot (M^{(p-1)})^{k(q-1)}) \bmod p = (M \cdot (M^{(p-1)} \bmod p)^{k(q-1)}) \bmod p.$$

- There are now two cases. It's possible that $p|M$ so that $M \bmod p = 0$. But then $M^{es} \bmod p = 0$, so that $C^s = M^{es} \equiv M \pmod{p}$. This is our desired conclusion, so we consider the case that p does not divide M.

- In this case, by Fermat's little theorem, $M^{(p-1)} \bmod p = 1$. Thus

$$M^{es} \bmod p = (M \cdot 1^{k(q-1)}) \bmod p = M \bmod p.$$

  Thus in this case, $C^s \equiv M \pmod{p}$ as well, finishing the proof.

Summing up

- Let's look at what math we needed to do this RSA encryption.
- Correctness depended on Fermat's little theorem, whose proof depended on the idea of a one-to-one and onto function.
- None of the ideas made any sense without the notion of an equivalence relation (congruence mod n in our examples).
- Feasibility depended on the gcd algorithm extended to find multiplicative inverses. The extended algorithm was proved correct by strong induction (actually proving the $sm + tn$ theorem), and the inductive proof gave the formulas we needed.
- In that inductive proof, we gave a formal statement of a proposition $P(n)$, using logic and quantifiers.
- Analyzing the efficiency involved another theorem (Lame) whose proof was by strong induction.
- GCDs exist using the lattice of positive integers under the divisibility ordering.
- So, almost all of the discrete math ideas we introduced were exemplified in understanding the application.

What's next?

- Encryption functions are a special case of what are called computable functions from strings to strings.
- In EECS 376 you learn about what kinds of functions can be computed at all. Most functions, in a precise sense, cannot be computed – there is no program even conceivable that could calculate them.
- You also learn about useful things like finite-state automata. These have applications everywhere in computer science.
- Specifying different kinds of (theoretical models of) computation requires all of the discrete math ideas we have learned.
- We have not done things like counting and combinatorics. You do learn these things in a probability course. The discrete math ideas help understand this stuff too. Don't forget about it!