In-Class Problems Week 15, Wed.

Problem 1.
Let's try out RSA! There is a complete description of the algorithm in the text box. You'll probably need extra paper. **Check your work carefully!**

(a) Go through the **beforehand** steps.

- Choose primes p and q to be relatively small, say in the range 10–40. In practice, p and q might contain hundreds of digits, but small numbers are easier to handle with pencil and paper.
- Try $e = 3, 5, 7, \ldots$ until you find something that works. Use Euclid's algorithm to compute the gcd.
- Find d (using the Pulverizer).

When you're done, put your public key on the board prominentally labelled "Public Key." This lets another team send you a message.

(b) Now send an encrypted message to another team using their public key. Select your message m from the codebook below:

- 2 = Greetings and salutations!
- 3 = Yo, wassup?
- 4 = You guys are slow!
- 5 = All your base are belong to us.
- 6 = Someone on *our* team thinks someone on *your* team is kinda cute.
- 7 = You *are* the weakest link. Goodbye.

(c) Decrypt the message sent to you and verify that you received what the other team sent!

Problem 2. (a) Just as RSA would be trivial to crack knowing the factorization into two primes of n in the public key, explain why RSA would also be trivial to crack knowing $\phi(n)$.

(b) Show that if you knew n, $\phi(n)$, and that n was the product of two primes, then you could easily factor n.

Problem 3.
A critical fact about RSA is, of course, that decrypting an encrypted message always gives back the original message m. Namely, if $n = pq$ where p and q are distinct primes, $m \in [0..pq)$, and

$$d \cdot e \equiv 1 \pmod{(p-1)(q-1)},$$

then

$$\widehat{m}^d ::= \left(m^e\right)^d = m \ (\mathbb{Z}_n). \tag{1}$$

We'll now prove this.

(a) Explain why (1) follows very simply from Euler's theorem when m is *relatively prime to* n.

All the rest of this problem is about removing the restriction that m be relatively prime to n. That is, we aim to prove that equation (1) holds for *all* $m \in [0..n)$.

It is important to realize that there is no practical reason to worry about—or to bother to check for—this relative primality condition before sending a message m using RSA. That's because the whole RSA enterprise is predicated on the difficulty of factoring. If an m ever came up that wasn't relatively prime to n, then we could factor n by computing $\gcd(m, n)$. So believing in the security of RSA implies believing that the liklihood of a message m turning up that was not relatively prime to n is negligible.

But let's be pure, impractical mathematicians and get rid of this technically unnecessary relative primality side condition, even if it is harmless. One gain for doing this is that statements about RSA will be simpler without the side condition. More important, the proof below illustrates a useful general method of proving things about a number n by proving them separately for the prime factors of n.

(b) Prove that if p is prime and $a \equiv 1 \pmod{p-1}$, then

$$m^a = m \ (\mathbb{Z}_p). \tag{2}$$

(c) Give an elementary proof[1] that if $a \equiv b \pmod{p_i}$ for distinct primes p_i, then $a \equiv b$ modulo the product of these primes.

(d) Note that (1) is a special case of

Claim. *If n is a product of distinct primes and $a \equiv 1 \pmod{\phi(n)}$, then*

$$m^a = m \ (\mathbb{Z}_n).$$

Use the previous parts to prove the Claim.

[1] There is no need to appeal to the Chinese Remainder Theorem.

The RSA Cryptosystem

A **Receiver** who wants to be able to receive secret numerical messages creates a *private key*, which they keep secret, and a *public key*, which they make publicly available. Anyone with the public key can then be a **Sender** who can publicly send secret messages to the **Receiver**—even if they have never communicated or shared any information besides the public key.

Here is how they do it:

Beforehand The **Receiver** creates a public key and a private key as follows.

1. Generate two distinct primes, p and q. These are used to generate the private key, and they must be kept hidden. (In current practice, p and q are chosen to be hundreds of digits long.)
2. Let $n ::= pq$.
3. Select an integer $e \in [1, n)$ such that $\gcd(e, (p-1)(q-1)) = 1$.

   The *public key* is the pair (e, n). This should be distributed widely.
4. Compute $d \in [1, n)$ such that $de \equiv 1 \pmod{(p-1)(q-1)}$. This can be done using the Pulverizer.

   The *private key* is the pair (d, n). This should be kept hidden!

Encoding To transmit a message $m \in [0, n)$ to **Receiver**, a **Sender** uses the public key to encrypt m into a numerical message

$$\widehat{m} ::= \text{rem}(m^e, \ n).$$

The **Sender** can then publicly transmit $\widehat{m}$ to the **Receiver**.

Decoding The **Receiver** decrypts message $\widehat{m}$ back to message m using the private key:

$$m = \text{rem}(\widehat{m}^d, \ n).$$