Lecture 18: Public key cryptography

Last time:

- Claim (Euler's theorem): if \([a]_m\) is a unit, then \([a^{\phi(m)}]_m := [a^k]_m\) is well-defined
- We reduced this to \([a^{\phi(m)}]_m = [1]_m\).

Proof: Choose an arbitrary \(m\) and unit \([a]_m\). We consider what happens to the elements of \(\mathbb{Z}_m\) when we multiply by \([a]\).

Example: Suppose \(m = 9\). Here are some pictures for various \([a]\):

Back to proof: we'll always have the units divided into loops all same length.

\[[a^0] \rightarrow [a] \rightarrow [a^2] \rightarrow [a^3] \rightarrow \ldots\]

Can't go forever without repeating, only finitely many units.

Can't happen.

\(1 = a^k = q^0\)

for some \(k\) \(\leq \phi(m)\).

Starting at any \([a]\), taking \(d\) steps \([a^d] \leftarrow [a^k] \leftarrow [a^{k+1}] \leftarrow \ldots \). If this brings me back \([a^0] \leftarrow [a^0] \leftarrow \ldots \), so all loops are same length.

If there are \(k\) loops, each of length \(l\), there are \(\phi(m)\) total elements (i.e. units), so \(\phi(m) = k \cdot l\).

So \([a^{\phi(m)}] = [a^k]^l = ([a^l]^k)^l = [1]_m\). 

[1] because of defn of \(l\).
Cryptography

Goal: the sender wants to send a message to the recipient without the attacker being able to decipher it.

Example: add 7 to each letter, wrapping around past 26:

- \( a \rightarrow g \)  
- \( b \rightarrow h \)

One-time pad: pick a different \( n \) randomly for each letter, add it to character:

- \( "hi" \) one-time pad
- \( (3, 7) \)  
- \( \rightarrow \)  
- \( "k n" \)

Problem: have to tell the recipient what the pad is, so they can decrypt.
Public key cryptography

Key idea: there's a secret that only the recipient knows. Related "public key" that everyone knows.

Sender

\[ \text{msg} \]

\[ \rightarrow \text{public key} \]

\[ \arrow{\text{msg} \rightarrow \text{Enc}} \rightarrow \text{cypher text} \rightarrow \arrow{\text{Dec}} \rightarrow \text{msg} \]

Recipient

- came up with
- private key related.

RSA cryptosystem.

\[ \text{Sender} \]

\[ \text{msg} \rightarrow \text{Enc} \rightarrow \text{cypher text} \rightarrow \text{Dec} \rightarrow \text{msg} \]

\[ \text{Receiver} \]

\[ \text{msg} \rightarrow \text{Dec} \rightarrow \text{cypher text} \rightarrow \text{Enc} \rightarrow \text{msg} \]

\[ \text{msg} \rightarrow \text{Dec} \rightarrow \text{cypher text} \rightarrow \text{Enc} \rightarrow \text{msg} \]

- there's an efficient way to test to see if \( n \) is prime.
- there's lots of primes: pick a large odd \( n \), check whether prime, if not, try \( n + 2 \) and repeat, there's enough primes that you'll quickly find one.

Conjecture: if \( p \) and \( q \) are large, then it is difficult to factor \( n \) into \( p, q \).

Naive algorithm: To factor \( n \), divide by every prime \( \leq \sqrt[n]{n} \).

This algorithm takes exponential time in \( \sqrt[n]{n} \) digits of \( m \).

Note! Nobody has been able to prove that there's no fast factoring algorithm.