Math 7H | Professor: Padraic Bartlett

Lecture 4: Cryptography (Modern)

Week 4 | *UCSB 2015*

We talked about a number of historical methods in which people have encrypted messages in last week's talk. Today, we're going to talk about the methods people use today! Specifically, we're going to talk about the **Diffie-Hellman key exchange system**, and how this gets implemented via the concept of **elliptic curve cryptography**.

1 Key Exchanges: A Puzzle

First, before we start, consider the following puzzle:

Puzzle. The Untrustworthy Ferryman.

- Good news! You've found yourself in the possession of a shoebox full of rubies, diamonds, gold, etc. You also have a padlock and key, which you can use to lock the outside of the box.
- Bad news! You're trapped on a desert island in the middle of nowhere.
- Good news! Every day at noon, a ferryman approaches your island. His boat is just big enough to contain him and a small box. If you could get the box delivered to your friend on the mainland, in a way where he could open the box, they could use a handful of the jewels to rescue you and set you free!
- Bad news! The ferryman will steal and not deliver anything that is not a locked box. Locked boxes he will take and deliver to your friend on the mainland for the price of one jewel. But anything else he will steal!
- Possibly useful news? Your friend also has a box and padlock and key, and the ferryman is willing to take packages back to you from your friendas part of the deal. But again, he will steal anything that is not a locked box.

How can you get him the treasure?

The answer is on the next page; try to solve it before continuing!

Answer. Here's the answer!

- You lock up your box and send it to your friend via the ferryman. Because the box is locked, the ferryman is honest and delivers the box.
- Your friend receives the box, and **locks the box with his padlock as well**! I.e. now the box has two locks on it.
- The doubly-locked box is sent back to you. You remove your lock, and send it back to your friend.
- Your friend receives the box with just his lock on it! He removes his lock, takes the jewels, rescues you, and you live your life in comfort and/or splendor.

This idea, roughly speaking, is how modern cryptographic systems work!

2 Public Key Exchange Systems

In general, a public key exchange system, roughly speaking, is any cryptographic system that works like our puzzle above — i.e. where there are two parties that are communicating, who do so in the following manner:

- Party A takes a message and encrypts it with a secret key, known only to them. Then they send this message to party B.
- Party B takes that encrypted message and encrypts it a second time with their secret key, and sends it back to A
- Party A then takes this doubly-encrypted message and decrypts it with their key. If our encryption and decryption functions **commute** — in other words, if encrypting and then decrypting is the same thing as decrypting and then encrypting — then this "undoes" the first layer of encryption, leaving a message encrypted with only B's key. This is sent back to B.
- Party B then takes that message and decrypts it; it is now readable by party B!

This is the idea for how most modern cryptographic systems work. With the rest of this class, we're going to explore one specific method of this encryption scheme: the **Diffie-Hellman** key exchange system!

We start by reminding the reader about how **modular arithmetic works**:

3 Modular Arithmetic

Definition. The set $\mathbb{Z}/12\mathbb{Z}$, of "clock numbers," is defined along with an addition operation $+$ and multiplication operation $\cdot$ as follows:

- Our set is the numbers $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$.

- Our addition operation is the operation "addition mod 12," or "clock arithmetic," defined as follows: we say that $a+b \equiv c \mod 12$ if the two integers $a+b$ and c differ by a multiple of 12. Another way of thinking of this is as follows: take a clock, and replace the 12 with a 0. To find out what the quantity $a+b$ is, take your clock, set the hour hand so that it points at a, and then advance the clock b hours; the result is what we call $a+b$.

  For example, $3+5 \equiv 8 \mod 12$, and $11+3 \equiv 2 \mod 12$. This operation tells us how to add things in our set.

- Similarly, our multiplication operation is the operation "multiplication mod 12," written $a \cdot b \equiv c \mod 12$, and holds whenever $a+b$ and c differ by a multiple of 12. Again, given any pair of numbers a, b, to find the result of this "clock multiplication," look at the integer $a \cdot b$, and add or take away copies of 12 until you get a number between 0 and 11.

  For example, $2 \cdot 3 \equiv 6 \mod 12$, $4 \cdot 4 \equiv 4 \mod 12$, and $6 \cdot 4 \equiv 0 \mod 12$.

We often will denote this object as $\langle \mathbb{Z}/12\mathbb{Z}, +, \cdot \rangle$, instead of as $\mathcal{C}$.

Simple enough, right? We can generalize this as follows:

Definition. The object $\langle \mathbb{Z}/n\mathbb{Z}, +, \cdot \rangle$,i.e.s defined as follows:

- Your set is the numbers $\{0, 1, 2, \ldots n-1\}$.

- Your addition operation is the operation "addition mod n," defined as follows: we say that $a+b \equiv c \mod n$ if the two integers $a+b$ and c differ by a multiple of n.

  For example, suppose that $n = 3$. Then $1+1 \equiv 2 \mod 3$, and $2+2 \equiv 1 \mod 3$.

- Similarly, our multiplication operation is the operation "multiplication mod n," written $a \cdot b \equiv c \mod n$, and holds whenever $a+b$ and c differ by a multiple of n.

  For example, if $n = 7$, then $2 \cdot 3 \equiv 6 \mod 7$, $4 \cdot 4 \equiv 2 \mod 7$, and $6 \cdot 4 \equiv 3 \mod 7$.

Here's a fun property:

Theorem 1. ***Fermat's Little Theorem.*** *Let p be a prime number. Take any $a \neq 0$ in $\mathbb{Z}/p\mathbb{Z}$. Then*

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

Proof. There are many different proofs of this remarkable theorem! The one we present here is perhaps one of the shortest, and relies on the **binomial theorem**:

Theorem. For any positive integer n, and any x, y, we have

$$(x+y)^n = \sum_{i=0}^{n} \binom{n}{i} x^{n-i} y^i.$$

If you haven't seen this theorem before, its proof is not too hard: do it!

Once you've done that, we prove Fermat's little theorem by induction on a. Our base case is easy: when $a = 1$, we have

$$1^{p-1} = 1 \equiv 1 \mod p.$$

Induction is not too much harder. We start by assuming that

$$k^{p-1} \equiv 1 \mod p,$$

and now want to prove that

$$(k+1)^{p-1} \equiv 1 \mod p.$$

To do this, simply look at the quantity

$$(k+1)^p.$$

By the binomial theorem, we have

$$\begin{aligned}(k+1)^p = \sum_{i=0}^{p} \binom{p}{i} k^{p-i} 1^i &= k^p + \binom{p}{1} k^{p-1} 1^1 + \binom{p}{2} k^{p-2} 1^2 + \ldots + \binom{p}{p-1} k^1 1^{p-1} + \binom{p}{p} 1^p \\ &= k^n + \binom{p}{1} k^{p-1} + \binom{p}{2} k^{p-2} + \ldots + \binom{p}{p-1} k + 1.\end{aligned}$$

Now, notice the following property:

Observation. If p is prime, then for any $1 \leq i \leq p-1$, the quantity

$$\binom{p}{i} = \frac{p!}{i! \cdot (p-i)!} = \frac{(p-i+1)(p-i+2)\ldots(p)}{i!}$$

is an integer multiple of p. This is because

- p is a factor of the numerator, and
- no factors of p are in the denominator, because p is prime and greater than i.

For example,

$$\binom{5}{3} = \frac{5!}{3! \cdot (2)!} = \frac{(3)(4)(5)}{3!} = 10,$$

which is a multiple of 5.

Given this observation, we can now see that

$$\begin{aligned}(k+1)^p &= k^p + \binom{p}{1} k^{p-1} + \binom{p}{2} k^{p-2} + \ldots + \binom{p}{p-1} k + 1 \\ &\equiv k^p + 0 \cdot k^{p-1} + \ldots + 0 \cdot k + 1 \mod p \\ &\equiv k^p + 1 \mod p.\end{aligned}$$

This is because all of the $\binom{p}{i}$ terms are multiples of p, and therefore congruent to 0 mod p.

Now, we use our inductive hypothesis, which tells us that

$$\begin{aligned} k^{p-1} &\equiv 1 \mod p \\ \Rightarrow k^p &\equiv k \mod p; \end{aligned}$$

plugging this in gives us that

$$(k+1)^p \equiv k+1 \mod p.$$

Dividing through by $(k+1)$ yields

$$(k+1)^{p-1} \equiv 1 \mod p,$$

as claimed.

□

Why do we care about this? Because it lets us create one of the first (and one of the most fundamental) modern cryptographical systems!

4 Diffie-Hellman

Algorithm. The Diffie-Hellman key exchange algorithm works as follows: suppose that we have two people, Alice and Bob, that want to communicate over a public network.

Ahead of time, Alice and Bob agree on a public base prime p to work in, as well as a public seed g. Selecting this value p and seed g requires some thought in practice due to some complicated mathematical tricks people can perform. However, if you just pick a big prime p (like 300+ digits) this usually works pretty well. Choices for g are a little harder to pin down when they are "good," but usually most anything works — in practice, g is usually 2,3 or 5.

From there, Alice and Bob each pick a "secret key" a, b from $\mathbb{Z}/p\mathbb{Z}$. Then, they do the following:

1. Alice sends the number g^a publicly to Bob.
2. Bob then takes this received number g^a and uses his key to raise it to the b-th power, which gives him g^{ab}.
3. Bob then sends the number g^b publicly to Alice.
4. Alice then takes this received number g^b and uses her key to raise it to the a-th power, which gives her g^{ab}.
5. Bob and Alice are now both in possession of the same secret key g^{ab}. Any eavesdroppers will only have heard g^a and g^b, and thus have no obvious way to figure out g^{ab}.

6. To communicate a message $m \in \mathbb{Z}/p\mathbb{Z}$, then, either Alice or Bob can send to the other party $m \cdot g^{ab}$. No other parties know the secret key, so they cannot decode this message.

7. Moreover, because both Alice and Bob know the secret key g^{ab}, along with the values g^a, g^b, they can use Fermat's little theorem to calculate its inverse! Specifically: notice that Bob, knowing b, p, and g^a, can calculate

$$(g^a)^{p-1-b} = g^{a(p-1)-ab} = g^{a(p-1)} \cdot g^{-ab} = \left(g^{p-1}\right)^a \cdot g^{-ab}.$$

   By Fermat's little theorem, we know that

$$g^{p-1} \equiv 1 \mod p,$$

   and therefore that

$$\left(g^{p-1}\right)^a \cdot g^{-ab} \equiv g^{-ab} \mod p.$$

   Therefore, Bob can calculate g^{-ab} and multiply it by their received encrypted message to get the original message back. Alice, using a, p, and g^b, can do the same trick as well; this allows both people to read their messages!

Why does this work? This is because the **discrete logarithm problem**, described below, is a "hard" problem to answer (in much the same vein as the NP problems we discussed earlier!)

Problem. The **discrete logarithm problem** is the following task: Suppose you're given a pair of numbers a, b in $\mathbb{Z} \mod p\mathbb{Z}$. The discrete logarithm problem asks the user for a value k such that

$$a^k \equiv b \mod p,$$

if some such value exists.

This problem is surprisingly hard to calculate. One naive algorithm that you might be tempted to try is simply raising a to higher and higher powers, and then reducing mod p each time until you got something that is equal to b. However, this takes a hideously long time to run: if we just take values of k starting at 1 and going up, we're effectively looking through the elements of $\mathbb{Z}/p\mathbb{Z}$ one at a time, in the order $a, a^2, a^3 \ldots$. If we assume that b is chosen more or less at random, then we'd expect to find b after going through about half of $\mathbb{Z}/p\mathbb{Z}$ (and in the worst-case after going through all of $\mathbb{Z}/p\mathbb{Z}$.) In either case, we have runtime that is linear in the size of the group! Therefore, if our group has say 300 digits (like we have for the size of prime requested in the algorithm above,) our algorithm has runtime that's absolutely awful (i.e. exponential in the number of digits in the size of the group.)

There are better algorithms known, but none that are substantially better — every method we have for solving this problem (that we know of) takes about this long to run,

up to small improvements! (In later talks, we'll discuss the concept of P versus NP that will make this idea of "runtime" make more sense!)

As a consequence, Alice and Bob in the discussion above can be reasonably sure that (unless someone comes up with an algorithm that's much faster than the one above) their communications are going by un-eavesdropped, because everything they send needs the discrete log problem to be decoded without keys!