CRYPTOGRAPHY HANDOUT 19

PIGEON BIRTHDAY PARTIES

1. Pigeons

1. There are 11 children in a family. Show that at least 2 of the children were born on the same day of the week.

2. Assuming the 11 children live with their 2 parents, show that 2 family members were born in the same month.

3. There are 430 students taking Calculus I. Show at least 2 of them were born on the same day of the year.

The Pigeonhole Principle (aka Dirichlet's Box Principle) If there are k pigeonholes, n pigeons, and $n > k$, then at least one pigeonhole contains at least 2 pigeons.

Apply the Pigeonhole Principle to the following problem:
If there are n people who can shake hands with one another, show that there is always a pair of people who will shake hands with the same number of people. [Hint: what are the pigeons and what are the pigeonholes?]

2. The Birthday Problem

We'll look at a related problem now in this section, known as the Birthday Problem, which is stated as follows:

If there are n people, what is the probability that some pair of them will have the same birthday (assuming 366 possible birthdays and that each day of the year is equally probably for a birthday)?

1. Come up with some initial ideas here.

2. It's easier to compute the probability that no two people have the same birthday. Start with $n = 2$. Say that the first person's birthday is on a certain day. Then the second person has probability $(1 - \frac{1}{366}) \approx .9973$ of having a different birthday. Make sure this makes sense to you before you move on.

3. What is the probability that $n = 2$ people have the same birthday?

4. If $n = 3$, the probability that no two people have the same birthday is

$$(1 - \frac{1}{366})(1 - \frac{2}{366}) \approx .9918.$$

Why?

5. What is the probability that two people out of $n = 3$ have the same birthday?

6. What do you think is the general probability that out of n people, two have the same birthday?

Facts:

- When $n = 367$, the probability that two people have the same birthday is 1 (Why?).
- When $n = 70$, the probability is .999.
- When $n = 23$, the probability is .5.

Conclusion: it's actually probable pretty quickly (for small n) to have two people with the same birthday. We'll use this idea in the next section.

3. Pigeon Birthday Parties

A **cryptographic hash function** h takes an input or message m of arbitrary length and produces an output $h(m)$ of a fixed length. These output values are **hash values** or **hashes**.

Properties.

1. $h(m)$ is quick to compute.
2. h is a *one-way function*, meaning it is computationally infeasible to find a preimage. i.e. given a hash y, it's hard to find m such that $h(m) = y$.
3. h is *strongly collision-free*, meaning it is computationally infeasible to find messages m_1 and m_2 such that $h(m_1) = h(m_2)$.

Make sure the definition and properties make sense to you before you move on.

Example. The discrete log hash function is defined as follows:

1. Choose a large prime p such that $q = \frac{p-1}{2}$ is also prime.
2. Choose 2 primitive roots α and β for p. Since α is a primitive root, there exists some number x such that $\alpha^x \equiv \beta \bmod p$. Finding x is hard (this is the discrete log problem).
3. Write the message m as $m = x_0 + x_1 q$ for $0 \leq x_0, x_1 \leq q - 1$.
4. The hash function is $h(m) = \alpha^{x_0} \beta^{x_1} \bmod p$.

A specific example with numbers chosen is hard to find, but make sure the example above makes sense to you before you move on.

Recall: Digital signatures are used to verify that messages m are authentic. Last time, we talked about RSA Signatures and the ElGamal Signature Scheme.

Since the process can be long to sign an entire message, sometimes a signature scheme is applied to as hash rather than the actual message. i.e. Alice has a message m. She computes the hash $h(m)$, which is much shorter, and she applies the signature scheme to $h(m)$, producing $\text{sig}(h(m))$. The pair $(m, \text{sig}(h(m)))$ conveys the same knowledge as $(m, \text{sig}(m))$ but is shorter.

Make sure this makes sense to you before you move on.

It turns out that digital signatures are susceptible to **birthday attacks**, which can be explained in the scenario below:

Suppose Eve (the pigeon) wants to trick Alice (also a pigeon) into signing a fraudulent contract online. She does the following:

- Eve has a fair contract m. She creates several copies with minor edits of m that basically are still the same (e.g. adding commas, removing commas, using synonyms, and other minor edits).
- Eve also creates a fraudulent contract m' and also makes several copies with minor edits of m' that are basically the same.
- She applies a hash function to all variations until $h(m) = h(m')$.
  Why is this likely to happen?
- Eve presents a fair contract to Alice to sign. Alice signs this. Eve takes Alice's signature and appends it to the fraudulent contract with the same hash. Because they have the same hash, the fraudulent contract seems valid.

Last task: discuss ways Alice might try to prevent a birthday attack.