Week 5 - Wednesday

COMP 4290

Last time

- What did we talk about last time?
- Key management

Questions?

Assignment 2

Project 2

Samuel Costa Presents

Hash Function Motivation

Where do passwords go?

- What magic happens when you type your password into…
  - Windows or Unix to log on?
  - Amazon.com to make a purchase?
  - A *Cobra Kai* fan site so that you can post on the forums?
- A genie from the 8th dimension travels back in time and checks to see what password you originally created

In reality…

- The password is checked against a file on a computer
- But, how safe is the whole process?
  - *Cobra Kai* fan site may not be safe at all
  - Amazon.com is complicated, much depends on the implementation of public key cryptography
  - What about your Windows or Unix computer?

Catch-22

- Your computer needs to be able read the password file to check passwords
- But even an administrator shouldn't be able to read everyone's passwords
- Hash functions to the rescue!

Hash Functions Defined

Definition

- A **cryptographic** (or one-way) hash function (called a cryptographic checksum in the book) takes a variable sized message ***M*** and produces a fixed-size hash code **H**(***M***)
- **Not the same as hash functions from data structures**
- The hash code produced is also called a **digest**
- It can be used to provide authentication of both the integrity and the sender of a message
- It allows us to store some information about a message that an attacker cannot use to recover the message

Crucial properties

Preimage Resistance

- Given a digest, should be hard to find a message that would produce it
- One-way property

Second Preimage Resistance

- Given a message m, it should be hard to find a different message that has the same digest

Collision Resistance

- Should be hard to find any two messages that hash to the same digest (collision)

Additional properties

Avalanching	• A small change in input should correspond to a large change in output
Applicability	• Hash function should work on a block of data of any size
Uniformity	• Output should be a fixed length
Speed	• It should be fast to compute a digest in software and hardware • No longer than retrieval from secondary storage

Common Hash Functions

MD5

- Message Digest Algorithm 5
- Very popular hashing algorithm
- Designed by Ron Rivest (of RSA fame)
- **Digest size:** 128 bits
- **Security**
  - Completely broken
  - Reasonable size attacks (2^{32}) exist to create two messages with the same hash value
- MD5 hashes are still commonly used to check to see if a download finished without error

SHA family

- **Secure Hash Algorithm**
- Created by NIST
- SHA-0 was published in 1993, but it was replaced in 1995 by SHA-1
- The difference between the two is only a single bitwise rotation, but the NSA said it was important
- Digest size: 160 bits
- Security
  - Broken if you have the resources
  - Theoretical attacks running in 2^{51} - 2^{57} time exist
  - Google generated two PDF files with the same hash in just over 2^{63} hashes in 2017
- SHA-2 is a successor family of hash functions
  - 224, 256, 384, 512 bit digests
  - Much better security
  - Designed by the NSA

The future of hash functions

- NIST had the contest for SHA-3 a few years ago
- It got down to five finalists:
  - BLAKE
  - Grøstl
  - JH
  - Keccak
  - Skein
- Keccak was announced as the winner in 2012
  - As with AES, Keccak beat out its competitors partly because it's so fast
  - Joan Daemen (of Rijndael fame) was also one of its designers

Keccak (SHA-3)

- Keccak uses a completely different form of hashing than SHA-0, SHA-1, and SHA-2
- Although the attacks on SHA-1 are expensive and no real attacks exist on SHA-2, the attacks on SHA-0 made people nervous about hash functions following the same design
- Keccak also allows for variable size digests, for added security
  - 224, 256, 384, and 512 are standard for SHA-3, but it is possible to go arbitrarily high in Keccak

Birthday Paradox

Activity

- Everyone stand up
- Sort yourselves using merge sort by birthday

Two people must share a birthday

- How many people do we need in the room so that two **must** share a birthday?
- 366 (well, 367, counting leap years)
- Pigeonhole principle
- This is the only way we can guarantee with 100% probability that there is a collision

Probability that two people share a birthday

- What if we only want it to be really likely that two people share a birthday?
- How many people do we need to have a 50/50 chance?
- Only 23!
  - That's an excited 23, not 23 factorial

Birthday paradox: the math

- The number of ways you can have no duplicate birthdays in a group of k people:

$$365 \cdot 364 \cdot 363 \ldots (365 - k + 1) = \frac{365!}{(365 - k)!}$$

- To find the probability that there are no duplicate birthdays in a group of k people, divide by all possible ways of assigning birthdays:

$$\frac{365!}{(365 - k)!} \cdot \frac{1}{365^k} = \frac{365!}{(365 - k)!\, 365^k}$$

- The probability that there is at least one duplicate is simply one minus this quantity

Probabilities for groups of various sizes

People (k)	Probability of Collision
10	12%
20	41%
23	50.7%
30	70%
40	89%
50	97%
100	99.99996%

General case

- If we care about a group of k items which can have a value between 1 and n, the probability that two are the same is:

$$P(n,k) = 1 - \frac{n!}{(n-k)!\,n^k}$$

- Because this form is a little unwieldy, we have an approximation that is easier to punch into a calculator:

$$P(n,k) > 1 - e^{\frac{-k(k-1)}{2n}}$$

Count it up

- If we want to find the number of items needed before there is greater than a $\frac{1}{2}$ probability of collision we get:

$$\frac{1}{2} = 1 - e^{\frac{-k(k-1)}{2n}}$$
$$-\frac{1}{2} = -e^{\frac{-k(k-1)}{2n}}$$
$$2 = e^{\frac{k(k-1)}{2n}}$$
$$\ln(2) = \frac{k(k-1)}{2n}$$

- For large k, $k(k-1) \approx k^2$, giving:

$$k \approx \sqrt{2\ln(2)\, n} \approx 1.18\sqrt{n}$$

Upcoming

Next time…

- Attacks on hash functions
- Digital signatures
- Review for Exam 1
- Jennifer Perez presents

Reminders

- Review Chapters 1, 2, and 12
- **Finish Assignment 2**
  - **Due Friday**
- Start on Project 2