SE 4472a

Information Security

Week 5:

Hash Functions

Fall 2015
Prof. Aleksander Essex

Western Engineering

Fingerprints: An example

This file:

- ubuntu-14.04-desktop-amd64.iso

Has this SHA-1 hash:

- d4d44272ee5f5bf887a9c85ad09ae957bc55f89d

We've taken a 1GB file, and created a 20 byte fingerprint

http://old-releases.ubuntu.com/releases/trusty/SHA1SUMS

Hash Functions

- What does it do?
  - Hash is a breakfast of chopped up meat, potatoes and vegetables
  - A cryptographic hash does something analogous: chops up and rearranges a string into a new string

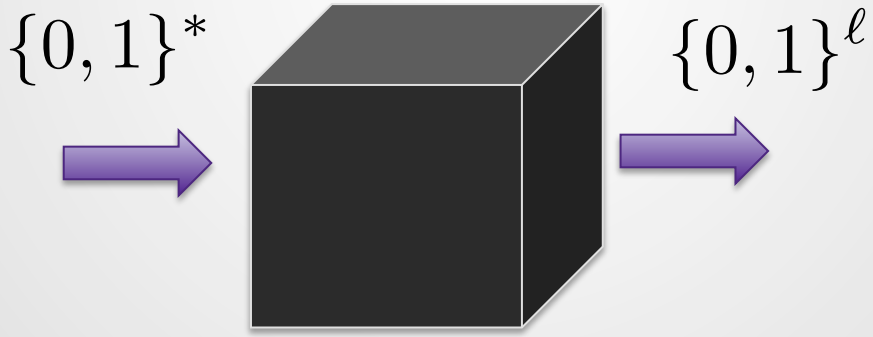
- Why do I need it?
  - Whenever it would be beneficial to create a short fixed-length string as kind of fingerprint, or digest of a long, arbitrary-length string.

Applications Hash Function

- Hash functions in TLS
  - An efficiency aid for digital signatures
  - Message authentication codes (MACs)
  - Key derivation function (to turn one key into a bunch of keys)
- Other uses
  - Fingerprinting e.g., for viruses/malware, intrusion detection, chain-of-custody
  - Integrity checks, e.g. in open source code/binary distribution
  - Secure password storage

Ideal Cryptographic Hashes

- A pseudo-random function that accepts an arbitrary-length string, and produces a fixed-length string



- Some additional properties required

Random Oracle Model

A random oracle is a way to conceptualize an ideal hash function

- For every arbitrary input, a random oracle outputs a random fixed-length string
- Each unique query is independent of the others
- If you repeat a query to the random oracle, it gives you the same answer

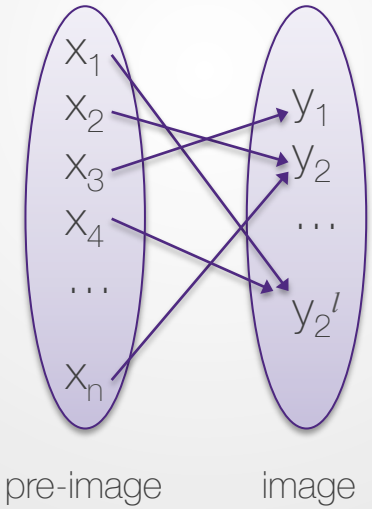
Random Oracles Don't Exist!

- Since there are infinitely many possible input strings, a random oracle would require infinite memory to maintain input-output pairs

- You could tweak the def'n and restrict the permissible input length (some real hash functions do this). This would still require exponential memory (relative to the security parameter). And that's not practical.

Hash Functions in Practice

- In practice we implement a hash function as (an efficient) pseudo-random-function



- But wait, we need more!

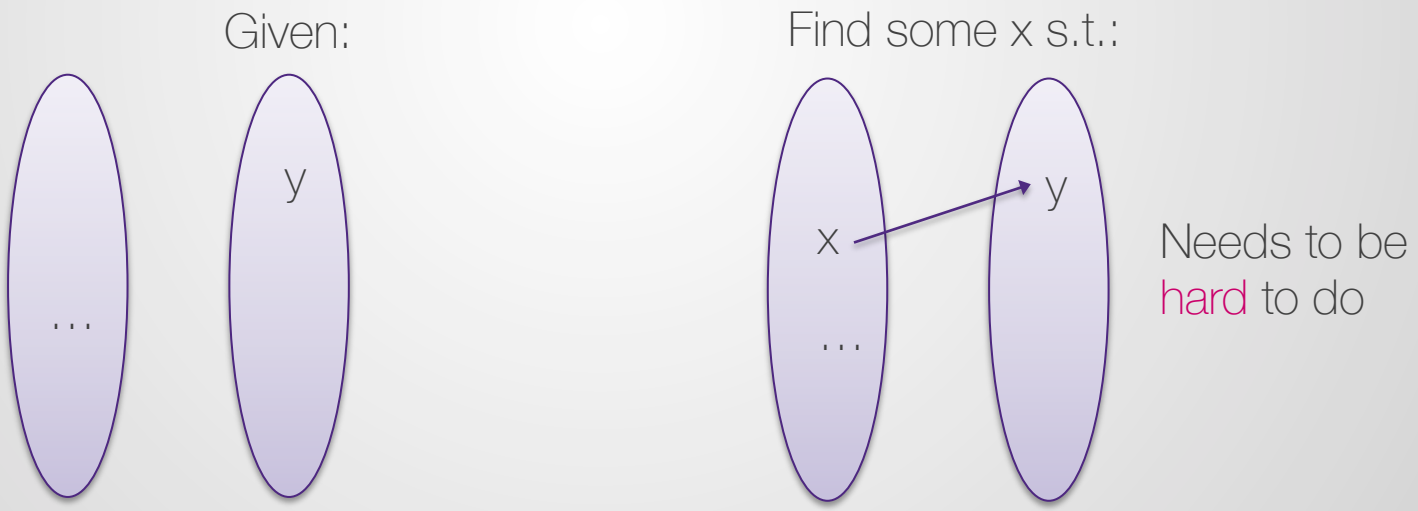
Properties of Cryptographic Hashes

There are three important security properties we need from a hash function:

- Pre-image resistance
  - Given a hash, it should be hard to find a message that produces that hash
- Second pre-image resistance
  - Given a message, it should be hard to find *another* message that produces the same hash
- Collision resistance
  - It should be hard to find any two messages that give the same hash (a so-call collision)

Pre-image Resistance

- Given $y \in \{0,1\}^{\ell}$ (i.e., the output of a hash) it should be computationally infeasible to find an $x \in \{0,1\}^{*}$ such that $y = h(x)$

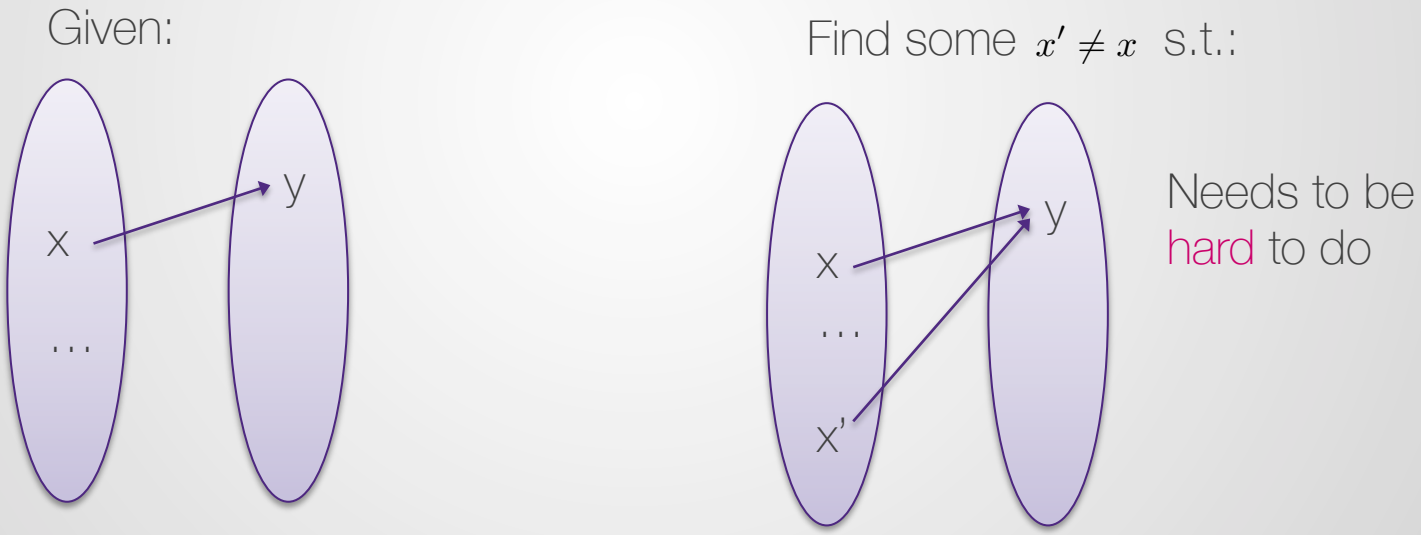


Pre-image Resistance

- If h( ) is a RO, what is the probability of finding a pre-image?
  - Balls in bins analogy: pick an x (i.e., a ball) and throw it into random bin
  - 2^l bins means $1/2^l$ chance of hitting target bin
  - Throw n balls, probability of success grows
  - You can expect to hit the target bin after 2^{l-1} tosses

Second Pre-Image Resistance

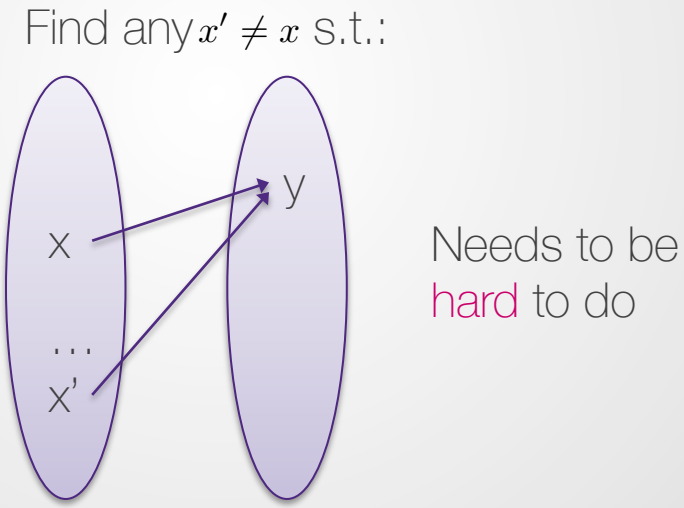
- Given a pre-image x, find a second *distinct* pre-image x' such that h(x)=h(x')



- What's the probability here?

Collision Resistance

- It should be computationally infeasible to find *any distinct* pair x, x' such that h(x)=h(x')



Needs to be hard to do

- What's the probability of this?

Birthday Paradox

- A related question: what's the probability there are two people in this room who share the same birthday?
  - If n=23, the probability is ~50%
  - Number is lower that you might expect, hence "paradox"
- How to compute?
  - Throw two balls. What's the probability they land in same bin (i.e., collide)? P(single collision)
  - How many collisions would you expect?
    - (n choose 2) * P(single collision)
  - Intuition: individual collision probability may be low, but number of possible pairings is high
  - How do these two values grow wrt to each other?

Collision Resistance

- You can expect to find a collision after $2^{\ell/2}$ attempts
- Implication for hash length: if 2^f is considered an infeasible number of operations, then we need:

$$2^f \leq 2^{\frac{\ell}{2}}$$

$$f \leq \frac{\ell}{2}$$

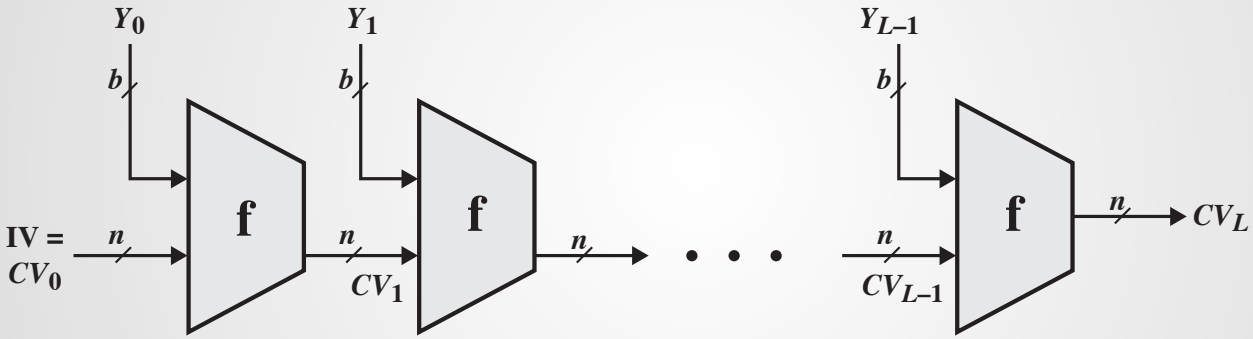
$$2f \leq \ell$$

- So if 2^{128} is considered infeasible, what should the hash length be to resist collisions?

SHA Hash Family

Merkle-Damgård Construction

- F is called a “compression function”



IV = Initial value
CV_i = chaining variable
Y_i = ith input block
f = compression algorithm

L = number of input blocks
n = length of hash code
b = length of input block

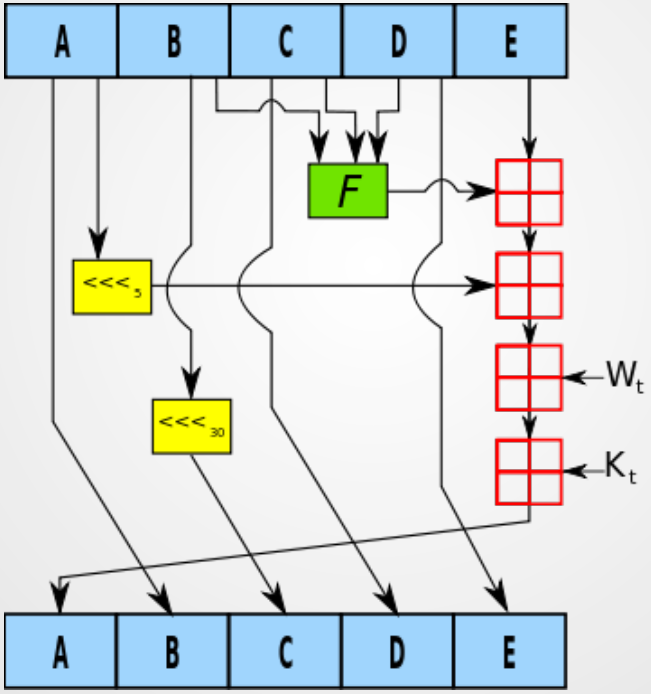
Figure 11.8 General Structure of Secure Hash Code

See course text (Stallings)

SHA Family

- SHA: Secure Hash Algorithm
- NIST standardized in 1993/93
- SHA-1 produces 160-bit output
- 2002 added 256- 384- and 512- bit variants

SHA-1 Compression Function



32-bit message word

http://en.wikipedia.org/wiki/SHA-1

Repeated 80 times

Other hashes

- MD5
  - Length = 128-bits
  - Collisions in 2^64 work (naïve, but computationally feasible in 2014)
  - Flaws discovered that can generate collisions much sooner
    - Might make a good assignment question
  - Pre-image attack in 2^{123} work
  - People still us it for some reason. It's creator Ron Rivest encourages you don't anymore.
- Keccack (SHA-3)
  - Sponge construction (very neat alternative to Merkle-Damgård)
  - Upcoming NIST standard

Other issues

- Length extension attacks