Admin:
- Problem Set #1 due in Lecture 6.
- Problem Set #2 out Lecture 6. (new groups for Problem Set #2)

Project Idea:
- AEG: Automatic Exploit Generation CACM 2/14 p.74-84

Discuss:
- (The Tech) Tidbit students/letter/MIT legal aid 2/18/14

Today: Cryptographic hash functions
- definitions
- random oracle model
- desirable properties
- applications
- Keccak (SHA-3) overview
(Cryptographic) Hash functions

A cryptographic hash function $h$ maps bit-strings of arbitrary length to a fixed-length output in an efficient, deterministic, public, "random" manner:

$$h : \{0,1\}^* \rightarrow \{0,1\}^d$$

- all strings of length $d$
- all strings (of any length $\geq 0$)

Sometimes called a "message digest" function.

Typical output lengths are $d = 128, 160, 256, 512$ bits.

No secret key. Anyone can compute $h$ from its public description. Computation is efficient (poly-time).

Examples:

- MD4
- MD5
- SHA-1
- SHA-256
- SHA-512
- SHA-3 (coming!)

<table>
<thead>
<tr>
<th>MD4</th>
<th>MD5</th>
<th>SHA-1</th>
<th>SHA-256</th>
<th>SHA-512</th>
<th>SHA-3 (coming!)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$128$</td>
<td>$128$</td>
<td>$160$</td>
<td>$256$</td>
<td>$512$</td>
<td>$224, 256, 384, 512$</td>
</tr>
</tbody>
</table>
**Ideal Hash Function:** Random Oracle (RO)

- Theoretical model - not achievable in practice.

**Oracle ("in the sky")**

- receives inputs $x$ & returns output $h(x)$, for any $x \in \mathbb{Z}_q$ *\text{ and } |h(x)| = d$ bits.

- On input $x \in \mathbb{Z}_q$ *:
  
  if $x$ not in book:
  
  - flip coin $d$ times to determine $h(x)$
  
  - record $(x, h(x))$ in book

  else: return $y$ where $(x, y)$ in book.

- Gives random answer every time, but uses book to record previous answers, so $h$ is deterministic.

![Diagram](https://via.placeholder.com/150)
Many cryptographic schemes are proved secure in ROM ("Random Oracle Model"), which assumes existence of RO. Then RO is replaced by conventional hash function (e.g. SHA-256) in practice, which is hopefully "pseudorandom enough"?!
Hash function desirable properties:

1. "One-way" (pre-image resistance)
   "Infeasible", given $y \in \{0,1\}^d$ to find any $x$ s.t. $h(x) = y$ ($x$ is a "pre-image" of $y$)

   ![Diagram](image)

   $h: \{0,1\}^* \rightarrow \{0,1\}^d$

   (Note that a "brute-force" approach of trying $x$'s at random requires $\Theta(2^d)$ trials (in ROM).

2. "Collision-resistance" (strong collision resistance)
   "Infeasible" to find $x, x'$ s.t. $x \neq x'$ and $h(x) = h(x')$ (a "collision")

   ![Diagram](image)

   $h(x) = h(x')$

   (In ROM, requires trying about $2^{d/2}$ $x$'s $(x, x_2, \ldots)$ before a pair $x_i, x_j$ colliding is found. (This is the "birthday paradox".)

Actually, the correct definition is that is hard for an adversary, given $y = h(x)$ (where $x$ was picked uniformly at random from $\{0,1\}^n$) to find any $x'$ such that $h(x') = y$. 
Note that collisions are unavoidable since
\[ |\mathbb{E}_0,1^{2^d}| = \infty \]
\[ |\mathbb{E}_1,1^{2^d}| = 2^d \]

Birthday paradox detail:
If we hash \( x_1, x_2, \ldots, x_n \) (distinct strings)
then
\[
E(\#\text{collisions}) = \sum_{i \neq j} \Pr(h(x_i) = h(x_j))
\]
\[
= \binom{n}{2} \cdot 2^{-d} \quad [\text{if } h \text{ "uniform"}]
\]
\[
= \frac{n^2 \cdot 2^{-d}}{2}
\]

This is \( \geq 1 \) when \( n \geq 2^{(d+1)/2} \approx 2^{d/2} \)

The birthday paradox is the reason why hash function outputs are generally twice as big as you might naively expect; you only get 80 bits of security (w.r.t. CR) for a 160-bit output.

With some tricks, memory requirements can be dramatically reduced.
TCR 3  "Weak collision resistance" (target collision resistance, 2nd pre-image resistance)

"Infeasible", given $x \in \mathbb{Z}/13^+$, to find $x' \neq x$ s.t. $h(x) = h(x')$.

Like CR, but one pre-image given & fixed.

(In ROM, can find $x'$ in time $\Theta(2^d)$
(as for OW, since knowing $x$ doesn't help in ROM).

PRF 4  Pseudo-randomness

"$h$ is indistinguishable under black-box access
from a random oracle"

To make this notion workable, really need a
family of hash functions, one of which is chosen at random. A single, fixed, public hash function
is easy to identify...

NM 5  Non-malleability

"Infeasible", given $h(x)$, to produce $h(x')$ where $x$ and $x'$ are "related"
(e.g. $x' = x + 1$).

These are informal definitions...
Theorem: If $h$ is CR, then $h$ is TCR. (But converse doesn't hold.)

Theorem: $h$ is OW $\iff$ $h$ is CR
(norther implication holds)
But if $h$ "compresses", then CR $\Rightarrow$ OW.

Hash function applications

1. Password storage (for login)
   - Store $h(PW)$, not PW, on computer
   - When user logs in, check hash of his PW against table.
   - Disclosure of $h(PW)$ should not reveal PW (or any equivalent pre-image)
   - Need OW

2. File modification detector
   - For each file $F$, store $h(F)$ securely (e.g. on off-line DVD)
   - Can check if $F$ has been modified by recomputing $h(F)$
   - Need WCR (aka TCR)
     (Adversary wants to change $F$ but not $h(F)$.)
   - Hashes of downloadable software = equivalent problem.