Admin:

Pset #1 due tonight
Pset #2 out today, due Mon 3/12
Projects! (also see talk announcement on Piazza)

Today:

Cryptographic Hash Fns II (aka “Merkle Day”)

- Merkle Trees
- Merkle Puzzles
- PK crypto based on puzzles (“Merkle puzzles”)
  - Constructions:
    - Merkle–Damgård
    - Keccak (SHA-3)

Readings:

Katz & Lindell: Ch. 5
Paar & Pelzl: Chapter 11
Ferguson: Chapter 5
To authenticate a collection of $n$ objects:

Build a tree with $n$ leaves $x_1, x_2, \ldots, x_n$ & compute authenticator node as fn of values at children... This is a "Merkle tree":

```
\[
\text{Root} = h(\text{Value at } y \text{ || Value of } z)
\]
```

Root is authenticator for all $n$ values $x_1, x_2, \ldots, x_n$
To authenticate $x_i$, give sibling of $x_i$ & sibling of all its ancestors up to root

Apply to: time-stamping data
authentication whole file system

**Need:** CR

Used in Bitcoin...
Puzzles & Brute-Force Search

Want to create puzzle with solution known to creator that requires (on average) a fixed amount of work to solve.

Let \( h : \{0,1\}^* \rightarrow \{0,1\}^d \) be a crypto hash fn (e.g. SHA-256 with \( d = 256 \)).

The "puzzle" will be to invert \( h \), i.e. solve \( h(x) = y \) for \( x \) given \( y \).

To make this a puzzle, we restrict \( x \) to be in a known set \( S \) of possible solutions. E.g. \( S = \{0,1\}^s \) for \( s = 40 \).

To create a puzzle, pick \( x \in S \) at random, compute \( y = h(x) \).

Difficulty of solving is \( |S|/2 \) by brute-force search.

If \( s < d \), there will be no "false solutions"—no collisions.

Can create multiple (keyed) puzzles \((k, y)\) means solving \( h(k, x) = y \) for \( x \in S \).

Puzzle spec is \((h, k, s, y)\).

Puzzle creator knows solution.

Can also have puzzles where creator doesn't know solution with truncated hashes

\( h : \{0,1\}^* \rightarrow \{0,1\}^s \).

Try \( x \) at random until \( h(x) = y \).
Hash cash (Adam Back, 1997)

- Anti-spam measure
- Requires sender to provide "proof of work" ("stomp")
- Email without POW or from sender on whitelisted is discarded.
- POW:
  solve puzzle $h(k, r) = 0000000000000000$ where $k =$ sender||receiver||date||time
  $r =$ variable to be solved for
- Include $r$ in header as POW
- easy for receiver to verify payment (POW)
- takes $x 2^{20}$ trials to solve
- doesn't work well against botnets 😞
Merkle puzzles

- First "public key" system (really: key agreement)

\[ Alice \rightarrow Eve \rightarrow Bob \]

Eve is passive eavesdropper. How can Alice & Bob agree on a key?

Use puzzles (with restricted domain, so have unique solns)

\[ n = \# \text{puzzles of difficulty } 2^{500} = D \]

1. Bob chooses \( n \) values \( x_1, x_2, \ldots, x_n \) from \( S = \{0, 1\}^5 \)

Bob computes \( y_i = h(i \| x_i) \)

Bob sends \( (y_i, E_{x_i}(K_i)) \) to Alice for \( 1 \leq i \leq n \), where \( K_i \in \{0, 1\}^{256} \)

2. Alice picks random \( i \) from \( \{n/2, n/2 + 1, \ldots, n\} \)

Alice solves \( P_i \) for \( x_i \)

"decrypts to obtain \( K_i \)

"sends \( h(K_i) \) to Bob

3. Bob & Alice use \( K_i \) to communicate securely from then on.

Time for good guys = \( \frac{O(n)}{Bob} + \frac{O(D)}{Alice} \)

Time for Eve = \( O(n \cdot D) \)

For \( n = D = 10^9 \), "almost practical".
Hash function construction ("Merkle-Damgard" style)

- Choose output size \(d\) (e.g. \(d = 256\) bits)
- Choose "chaining variable" size \(c\) (e.g. \(c = 512\) bits)
  
  [Must have \(c > d\); better if \(c > 2d\)...]

- Choose "message block size" \(b\) (e.g. \(b = 512\) bits)
- Design "compression function" \(f\)

\[ f : \mathbb{F}_2 \times \mathbb{F}_2^b \rightarrow \mathbb{F}_2^c \]

[F should be OW, CR, PR, NM, TCR, ...]

- Merkle-Damgard is essentially a "mode of operation" allowing for variable-length inputs:
  
* Choose a \(c\)-bit initialization vector \(IV, c_0\)

[Note that \(c_0\) is fixed & public.]

* [Padding] Given message, append

  - \(10^b\) bits
  
  - fixed-length representation of length of input

  so result is a multiple of \(b\) bits in length:

\[ M = M_1, M_2, \ldots, M_n \] (\(n\) \(b\)-bit blocks)
h \{ 
\begin{align*}
\text{Then: } &\quad h(m) = c_n \text{ truncated to } d \text{ bits} \\
\text{Theorem: } &\quad \text{IF } f \text{ is CR, then so is } h. \\
\text{Proof: } &\quad \text{Given collision for } h, \text{ can find one for } f \text{ by working backwards through chain. } \\
\text{Thm: } &\quad \text{Similarly for OW.} \\
\end{align*}
\]

Common design pattern for \( f \):

\[
f(c_{i-1}, M_i) = c_{i-1} \oplus E(M_i, c_{i-1})
\]

where \( E(K, M) \) is an encryption function (block cipher) with \( b \)-bit key and \( c \)-bit input/output blocks.

(Davies-Meyer construction)
Typical compression function (MD5):

- Chaining variable & output are 128 bits = 4 x 32
- IV = fixed value
- 64 rounds; each modifies state (in reversible way) based on selected message word
- Message block b = 512 bits considered as 16 32-bit words
- Uses end-around XOR too around entire compression fn (as above)

![Diagram of MD5 compression function]

Xiaoyun Wang discovered how to make collision for MD4, MD5...
("Differential Cryptanalysis")

\[ g(x, y) = \begin{cases} 
xy \oplus x \oplus y \oplus \overline{e} \\
xy \oplus y \oplus \overline{x} \\
xy \oplus y \oplus \overline{x} \\
y \oplus x 
\end{cases} \]

depending on round
Keccak = SHA-3

Keccak Sponge Construction

- Output size: $d = 256$ bits
- State size: $r = 1088$ bits
- Round count: $c = 512$

Input:
- Padding with $w = 64$ bits
- Nonce padding
- Finalization

$w_{final} = 384$, $l = 1600$, $r = 60$, $c = 343$, $d = 256$