Admin: Pset #1 due Mon 2/22 (gradescope works by tonight)

Today: Cryptographic Hash Functions II ("Merkle Day")

- Merkle trees
- Puzzles
- PK crypto based on puzzles (Merkle puzzles)
- Construction
  - Merkle-Damgard
  - Keccak (SHA-3)

Readings:
- Katz/Lindell Chapter 5
- Paar/Pelzl Chapter 11
- Ferguson Chapter 5

News: Apple ordered to unlock iPhone
To authenticate a collection of $n$ objects:

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

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

authenticate 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 \( h : \{0,1\}^* \to \{0,1\}^d \) 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. Eg. \( S = \{0,1\}^s \) for \( s = 40 \).

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

Difficulty of solving \( x \) is \( |S|^{1/2} \) 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
\[ h : \{0,1\}^* \to \{0,1\}^s \]

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

- Anti-spam measure
- Requires sender to provide "proof of work" ("stamp")
- Email without POW or from sender on whitelist is discarded.
- POW:
  - Solve puzzle $h(k, r)$ ends in 20 zeros
    - 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 $x \times 2^{20}$ trials to solve
- Doesn't work well against botnets 😞
Merkle puzzles

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

  Alice  --  Eve  --  Bob

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

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

\[ n = \# \text{puzzles of difficulty } 2^{n-1} = D \]

1. Bob chooses \( n \) values \( x_1, x_2, \ldots, x_n \) from \( S = \{0,1\}_3^n \)
   - 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 \( K_i \in \{0,1\}_3^{256} \)

2. Alice picks random \( i \) from \( [n] = \{1, 2, \ldots, n\} \)
   - Alice solves \( P_i \) for \( x_i \)
     - "decrypt to obtain \( K_i \)
     - sends \( h(K_i) \) to Bob

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

   Time for good guys = \( O(n) + O(D) \)
   - Bob
   - 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
  \[ f : \{0,1\}^c \times \{0,1\}^b \rightarrow \{0,1\}^c \]
  [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
    - \( 10^k \) bits
    - fixed-length representation of length of input
  so result is a multiple of b bits in length:
  \[ M = M_1, M_2 \ldots M_n \quad (n \text{ b-bit blocks}) \]
Theorem: IF \( f \) is CR, then so is \( h \).

Proof: Given collision for \( h \), can find one for 
\( f \) by working backwards through chain. \( \square \)

Thm: Similarly for DW.

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 \times 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)

Xiayun Wang discovered how to make collision for MD4, MD5
("Differential cryptanalysis")

\[ g_6(y_2) = \begin{cases} 
xy \\ xz \\ yz \\ x \oplus y \\ z \oplus x \\ y \oplus x \\
\end{cases} \] depending on round
\textbf{Keccak} = SHA-3

Keccak Sponge Construction
\begin{align*}
d & = \text{output hash size in bits} \\
c & = 384, 512, 1024 \text{ \ or \ } n \text{ bits} \\
r & = 25 \\
f & \in \{\text{invertible, efficient function}\} \\
\end{align*}

Input padded with \(0^n\) until length \(d + w\) is a multiple of \(r\).
\(f\) has \(24\) rounds (\(w = 64\), not quite\).