Admin:

Pset #2 is out, due Wed 3/24

Projects! (Post ideas on Piazza...)

Today:

Constructing Hash Fns:

1. Merkle-Damgard (MD5, SHA2)
2. "Sponge" construction of SHA-3

Readings:

(for this part)

Wikipedia: Merkle-Damgard
SHA-3
Merkle-Damgard Construction

- Choose output size $d$ (e.g., $d = 256$ bits)
- Choose "chaining variable" size $c$ (e.g., $c = 512$ bits)
  (Make $c \geq 2d$ for good security)
- 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, TCR, NM, ... ]
- Merkle-Damgard is essentially a "mode of operation"
  allowing for variable-length inputs:
  - Choose a $c$-bit initialization vector $IV$, $c_0$
    ($c_0$ is fixed & public)
  - [Padding] Given message, append
    - at least one "1" bit, then
    - enough "0" bits so result is a multiple of $b$ bits
      after concatenation of length of message too.

\[ M = M_1 \cdot M_2 \cdots \cdot M_n \] (n b-bit blocks)

Then

\[ \text{Then } \]

\[ IV \rightarrow f \rightarrow c_0 \rightarrow f \rightarrow c_1 \rightarrow f \rightarrow c_2 \rightarrow f \rightarrow \cdots \rightarrow f \rightarrow c_n \rightarrow f \rightarrow c_n \text{ truncated to } d \text{ bits} \]

Thm: If $f$ is CR, then so is $h$.
PF: Given collision for $h$, can find one for $f$
      by working backwards through chain

Thm: Same for OW.
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.

(See Wikipedia for more details on MD5.)

SHA-3 (Keccak)

Different construction: “sponge” construction

- Can add entropy (message block) at any time
- Can extract output (squeeze sponge) at any time
Sponge construction (SM4-3)

State = 5x5x64 = 1600 bits

"Encryption function" (keyless)

new state

"absorb"

"squeeze"

1088

input bits

output bits

state
Keccak Sponge Construction

d = output length in bits, e.g. 224, 256, 384, 512
C = 2d bits
state size = 25w where w = word size (e.g. w=64)
c + r = 25w
r >= d (so hash can be fast d bits of z_0)
Input padded with \(10^l\) until length is a multiple of r
f has 24 rounds (for w=64), not quite identical (word oriented)
f is public, efficient, invertible function from \(\{0,1\}^{25w}\) to \(\{0,1\}^{25w}\)

Keccak = SHA-3