Definition: A cryptographic hash function

\[ h : \{0,1\}^* \rightarrow \{0,1\}^d \]
maps bit-strings of arbitrary length
to bit-strings of length \(d\)
in an efficient, deterministic, public,
"random" manner.

- Notes: "message digest", "fingerprint"
no secret key

Examples:
- MD5 \( d = 128 \)
- SHA-256 \( d = 256 \)
- SHA3-256 \( d = 256 \)
- SHAKE \( d \) variable

(Ideal) Random Oracle Model (ROM)

\[ h(x) : \]
\[ \begin{align*}
&\text{if } x \text{ already seen} \\
&\quad \text{give same output as before} \\
&\text{else} \\
&\quad \text{flip coin } d \text{ times & give result as output}
\end{align*} \]

(All parties have access to same random oracle.)

Typically: show scheme works in ROM
replace RO by SHA-256 & hope it works!

"pseudorandomness" - PR
Properties

Definitional style:
1. Add "salt" or key, to make hash function family
   \[ H = \{ h_s : s \in \{0,1\}^k \} \]
2. Let \( h \) be a single hash function, assume no violations of property occur in practice.

Collision-Resistance (CR)

"Infeasible" to find a "collision" (an \( x, x' \) s.t. \( x \neq x' \) and \( h(x) = h(x') \))

\[ \{0,1\}^k \rightarrow \{0,1\}^d \]

Collisions exist, you just can't find one.

Can find one in time \( \Theta(2^{d/2}) \) - "birthday paradox" (ROM)
[& using \( \Theta(1) \) storage - Floyd's "two-finger" algorithm]

\( \therefore \) can use \( h(x) \) as unique representative or proxy for \( x \)

Target Collision-Resistance (TCR)

"Infeasible", given \( x \), to find \( x' \neq x \) s.t. \( h(x') = h(x) \).

Time is \( \Theta(2^d) \) in ROM.

Thm: \( CR \Rightarrow TCR \)
Properties (cont.)

One-wayness (OW)

"Infeasible", given $y$ (image under $h$ of random $x$), to find any $x'$ s.t. $h(x') = y$ (pre-image of $y$)

$\exists 0,1^*$

\[
\begin{array}{ccc}
\exists 0,1^d \\
x \\
\rightarrow \\
y
\end{array}
\]

Brute-force in ROM takes $O(2^d)$ trials.

(DW different than CR, !)
Applications:

1. Password storage (for login)
   - Store \( h(PW) \), not PW, on computer
   - At login, check hash of PW against stored \( h(PW) \)
   - Disclosure of \( h(PW) \) should not reveal PW (or anything usable to log in)
   - Need: OW

2. File modification detector
   - For each file \( F \), store \( h(F) \) securely
   - Before using \( F \), recompute \( h(F) \) & check against stored value.
   - Need: TCR
   - Equivalent problems: hashes of downloadable software
   - But: can't have machine "check itself"!
Applications (cont.)

3. Digital signatures ("hash & sign")

\[ \text{PK}_A = \text{Alice's public key (for signature verification)} \]

\[ \text{SK}_A = \text{Alice's secret key (for signing)} \]

\[ \text{Sign}(\cdot) = \text{Sign}(\text{SK}_A, M) \quad [\text{Alice's sig on } M] \]

\[ \text{Verify}(M, \sigma; \text{PK}_A) \in \{\text{True}, \text{False}\} \]

Adversary wants to forge a signature that verifies.

- For large messages \( M \), easier to sign \( h(M) \):

\[ \sigma = \text{Sign}(\text{SK}_A, h(M)) \quad [\text{"hash \& sign"}] \]

Verifier recomputes \( h(M) \) from \( M \), then verifies \( \sigma \).

\( h(M) \) is a "proxy" for \( M \).

- Need: \( \text{CR} \) [Else Alice finds collision \( x, x' \)]

gets Bob to sign \( x \) (where \( h(x) = h(x') \))

then claims Bob really signed \( x' \), not \( x \).

- Don't need OW: (e.g. using \( h = \text{identity fn} \) is OK.)
MAC (Message Authentication Code)

MAC(K, M) needs shared key K

How to insert secret key K into hash function?

\[ h(K || M) \]  - does the work?

\[ \text{concatenation} \]

For some h, doesn't work (extension attacks!)

Fixes:

1. \[ \text{HMAC}(K, M) = h(k \oplus \text{pad}_2, h(k \oplus \text{pad}_2, M)) \]

2. Use SHA3 for h (sponge construction!)
Constructions of hash functions

1. Merkle-Damgård construction (e.g. SHA-256) (e.g. MD5)

- $f$ has 64 rounds on 8 32-bit words of state $OW, CR$
- $m_t$ includes padding $10^*L$
- 64-bit length $M$

- Vulnerable to length-extension attacks if used as a MAC
- Great as a hash function
Construction of hash functions

2) "sponge" construction (e.g., Keccak, SHA3, SHAKE)

- water = randomness (entropy)
- sponge can absorb (input keys or message)
- """" be squeezed (output PR bit string)

Can be used as MAC:
- absorb key
- absorb message
- squeeze out MAC

Can be used as PRF:
- absorb key
- squeeze out PR stream (as long as wanted)

Can be used as hash fn:
- absorb message
- squeeze out hash value

stateful (e.g., for SHA-3)

\[ S = \]

\[
\rightarrow \quad c \quad \leftarrow \quad r
\]

state = 25 words of 64 bits (1600 bit total)

- 8 word "capacity" (hidden state; adversary can't see or touch)
  \( c = 512 \)

- 17 word "rate" (visible state; adversary can see and/or modify)
  \( r = 1088 \)

Thm: Security (for CR & OW) is \( c/2 \) bits (e.g., 256-bits for SHA-3)
(continued) sponge construction

\( f \) = fixed, public function that maps state \( S \) 1-to-1 to a new value (permutation of 80,131,600)

\( f \) is keyless

1600 bits of state so every bit of output depends in a complicated way on every bit of input

AND, OR, NOT, rotate instructions

24 "rounds" - slightly different constant

absorb

\[ \text{capacity } c \]

rate \( r \)

\[ \text{m}_i \] (last on padded 100)
Admin:

- Psets (#1 due, #2 out)
- Guest speaker Wednesday - Andy Sells

Today:

(Cryptographic) Hash Functions

- Definition & Random Oracle Model
- Properties: Collision-resistant, one-wayness
- Applications
- Constructions

Readings:

Katz / Lindell (2nd ed) - Chapter 5
Paar / Pelzl - Chapter 11
Ferguson / Schneier / Kohno - Chapter 5