TOPIC: 6.857

DATE: 2/18/15

FILE UNDER: Hash functions

PAGE: L05.1

Admin:

Pset #1 due Mon 2/23. [Submit each problem as separate pdf.]

Pset #2 out Mon 2/23.

Charles River CryptoDay: Friday 2/20

(does everyone have access to secret student area?)

Today:

Cryptographic Hash Functions

- definition
- random oracle model (ROM)
- desirable properties (CR, OW, ...)
- applications
- construction

Readings:

Katz/Lindell	Chapter 5
Paar/Pelzl	Chapter 11
Ferguson	Chapter 5
Wikipedia	SHA-3

(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:

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

$\{0,1\}^*$: all strings (of any length ≥ 0)

$\{0,1\}^d$: all strings of length d

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:		d	note
	MD4	128	"broken" wrt CR
	MD5	128	"broken" wrt CR
	SHA-1	160	? CR?
	SHA-256	256	
	SHA-512	512	
	SHA-3 (Keccak) Oct 2012	224, 256, 384, 512	

"Ideal" Hash Function: Random Oracle (RO)

- Theoretical model - not achievable in practice

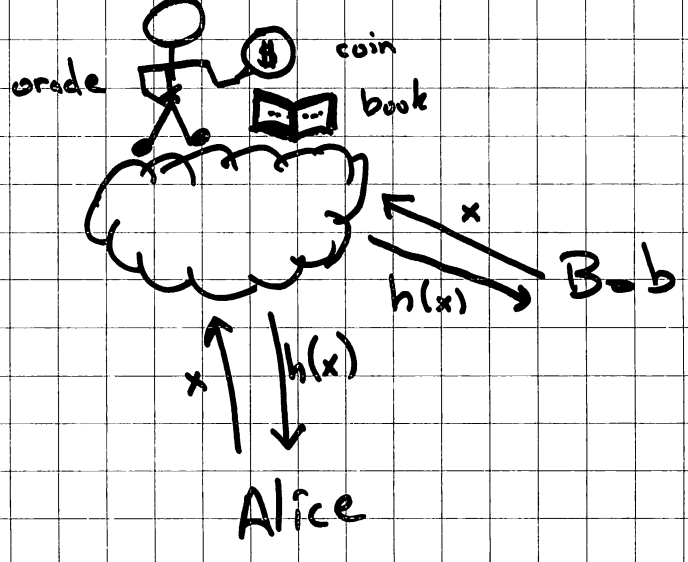
Oracle ("in the sky")

- receives inputs x & returns output $h(x)$, for any $x \in \{0,1\}^*$. $|h(x)| = d$ bits.
- On input $x \in \{0,1\}^*$:

  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 & consistent.



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:

OW

y computed:

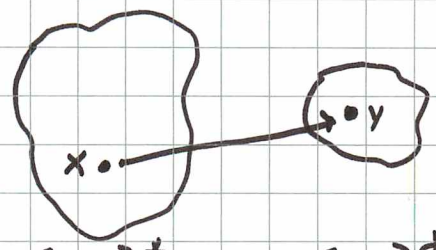
$x \in_R \{0,1\}$

$y = h(x)$

① "One-way" (pre-image resistance)

"Infeasible", given $y \in_R \{0,1\}^d$ to find

any x' s.t. $h(x') = y$ (x' is a "pre-image" of y)



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

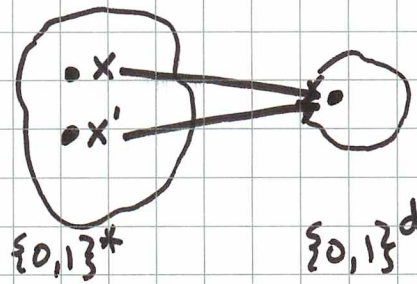
(Note that a "brute-force" approach of trying x's at random requires $\Theta(2^d)$ trials (in ROM).)

CR

② "Collision-resistance" (strong collision resistance)

"Infeasible" to find x, x' s.t. $x \neq x'$ and

$h(x) = h(x')$ (a "collision")



(In ROM, requires trying about $2^{d/2}$ x's $(x_1, x_2, \ldots)$ before a pair x_i, x_j colliding is found. (This is the "birthday paradox".))

Note that collisions are unavoidable since

$$|\{0,1\}^*| = \infty$$

$$|\{0,1\}^d| = 2^d$$

Birthday paradox detail:

If we hash $x_1, x_2, \ldots, x_n$ (distinct strings)

then

$$E(\#\text{collisions}) = \sum_{i \neq j} Pr(h(x_i) = h(x_j))$$

$$= \binom{n}{2} \cdot 2^{-d} \qquad [\text{if } h \text{ "uniform"}]$$

$$\approx \frac{n^2 \cdot 2^{-d}}{2}$$

This is ≥ 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 $x \in \{0,1\}^*$, 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 to find x').)

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 $\not\Longleftrightarrow$ h is CR
(neither 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.

③ Digital signatures ("hash & sign")

PK_A = Alice's public key (for signature verification)

SK_A = Alice's secret key (for signing)

Signing: $\sigma = \text{sign}(SK_A, M)$ [Alice's sig on M]

Verify: $\text{Verify}(M, \sigma, PK_A) \in \{\text{True}, \text{False}\}$

Adversary wants to forge a signature that verifies.

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

  $\sigma = \text{sign}(SK_A, h(M))$ ["hash & sign"]

  Verifier recomputes $h(M)$ from M, then verifies σ.

  In essence, $h(M)$ is a "proxy" for M.

- Need CR (Else Alice gets Bob to sign x, where $h(x) = h(x')$, then claims Bob really signed x', not x.)

- Don't need OW (e.g. h = identity is OK here.)

(4) Commitments

- Alice has value x (e.g. auction bid)
- Alice computes $C(x)$ ("commitment to x") & submits $C(x)$ as her "sealed bid"
- When bidding has closed, Alice should be able to "open" $C(x)$ to reveal x
- Binding property: Alice should not be able to open $C(x)$ in more than one way! (She is committed to just one x.)
- Secrecy (hiding): Auctioneer (or anyone else) seeing $C(x)$ should not learn anything about x.
- Non-malleability: Given $C(x)$, it shouldn't be possible to produce $C(x+1)$, say...
- How:

$$C(x) = h(r \| x) \qquad r \in_R \{0,1\}^{256}$$

To open: reveal r & x

- Note that this method is randomized (as it must be for secrecy.
- Need: OW, CR, NM

(really need more, for secrecy, as $C(x)$ should not reveal partial information about x, even.)