TOPIC: 6.857

DATE: 2/19/13

FILE UNDER:

PAGE: L4.1

Admin:

Today:

- ☐ Finish OTP & random bit generation
- ☐ Cryptographic hash functions
  - defn
  - random oracle model
  - desirable properties
  - applications

Notes:

Users need to
- generate large secrets
- share them securely
- keep them secret

} usability??

- avoid re-using them (google "Venona")

$$C_1 \oplus C_2 = (M_1 \oplus K) \oplus (M_2 \oplus K)$$
$$= M_1 \oplus M_2$$

from which you can derive M_1, M_2 often.

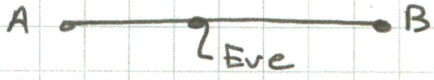
Theorem: OTP is malleable.

(That is, changing ciphertext bits causes corresponding bits of decrypted message to change.)

OTP does not provide any authentication of message contents or protection against modification ("mauling").

How to generate a random pad?

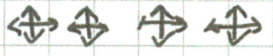
- Coins
- Dice
- Radioactive sources (old memory chips were susceptible to alpha particles)
- Microphone, camera
- Hard disk speed variations
- Intel 82802 chip set
- User typing or mouse movements
- Lavarand (lava lamp ⇒ camera)
- Alpern & Schneider:

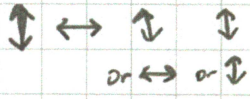


Eve can't tell <u>who</u> transmits.
A & B randomly transmit beeps.
They can derive shared secret.

- Quantum Key Distribution

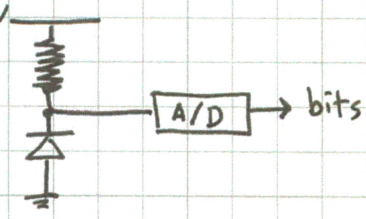
Polarized light: ↕ ↔ ↘ ↙

Filters (2)  (example filter)

result 

A sends single photons, polarized randomly.
B publicly announces filter choices
Then they know which bits they should have in common.

- ref today's lecture on Certifiable Quantum Dice
- "Noise diodes"



(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 (coming!)	224, 256, 384, 512	

"Ideal" Hash Function: Random Oracle (RO)

- Theoretical model – not achievable in practice

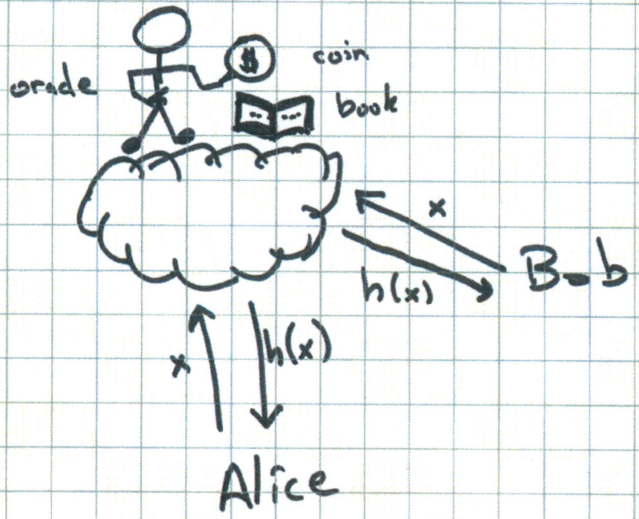
Oracle ("in the sky")

- 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.

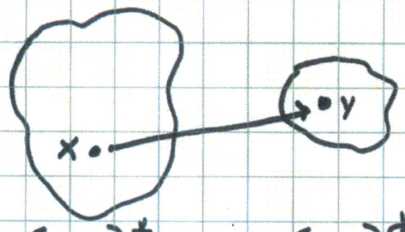


Many cryptographic schemes are proved secure in ROM ("Random Oracle Model"), which assumes existence of RO. Then RO is replaced by conventional hash function (e.g. SHA-256) in practice, which is hopefully "pseudorandom enough" (!?).

Hash function desirable properties:

OW 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)

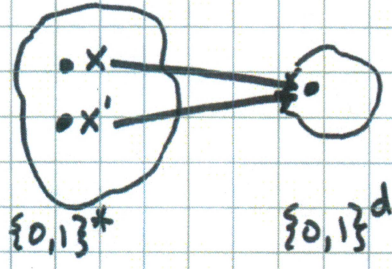


$h: \{0,1\}^* \longrightarrow \{0,1\}^d$

(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.