TOPIC: 6.857

DATE: 2/23/15

FILE UNDER: Hash Functions II

PAGE: L06.1

Admin:

Pset #1 due today
Pset #2 out today

Today:

Cryptographic Hash Functions II ("Merkle Day")

- Merkle trees
- Puzzles & brute-force search
- PK crypto based on puzzles (Merkle puzzles)
- Hash function construction methods
  - Merkle-Damgard
  - Keccak

Readings:

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

News:

Lenovo "Superfish"

Citizenfour wins Oscar for best documentary
(HBO tonight at 9pm)

Project idea:

Do security analysis of "OpenWrT" router software

TOPIC:

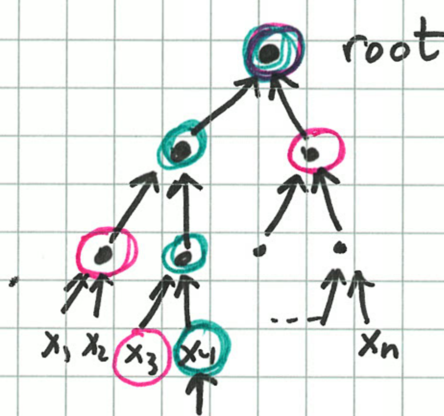
DATE:

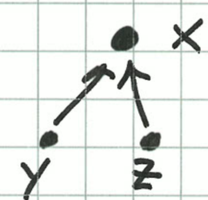
FILE UNDER:

PAGE: LOG.2

(5) <u>To authenticate a collection of n objects</u>:

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





value at x

$= h(\text{value at } y \,//\, \text{value at } z)$

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 his ancestors up to root

Apply to: time-stamping data

authenticating whole file system

<u>Need</u>: <u>CR</u>

Used in bitcoin...

TOPIC: 6.857

DATE: 2/23/15

FILE UNDER:

PAGE: L06.3

Can also have restricted domain

Puzzles & Brute-Force Search

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

If h is well-modeled as a random oracle, inverting h requires 2^d steps on average:

Given $y \in \{0,1\}^d$, adversary can do no better than trying $x_1, x_2, \ldots,$ until he finds x_i s.t. $h(x) = y$. Probability that $h(x_i) = y$ is 2^{-d} (by ROM), so expected # trials needed is 2^d. Brute-force

To make a "puzzle", choose d to be "not too large".

E.g. $h(x) = \text{sha256}(x) \bmod 2^d$ where $d = 40$

Takes 2^{40} steps to solve, on average.

Note: special-purpose chips & boards can do $\approx 2^{40}$ hashes/second, so this is maybe a "one-second puzzle" for such a device.

Puzzle difficulty is controllable (by choosing d)

Easy to create many puzzles: $h_k(x) = h(k \| x)$ so one puzzle for each parameter k.

Puzzle spec = (k, d, y) want x s.t. $h_k(x) = y$

Puzzle creator knows solution (computes y, given x)

TOPIC: 6.857 | DATE: 2/23/15

FILE UNDER: | PAGE: L06.4

Hash cash (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 $\approx 2^{20}$ trials to solve
- doesn't work well against botnets 

TOPIC: 6.857

DATE: 2/23/15

FILE UNDER:

PAGE: L06.5

Merkle Puzzles (1974)

- First "public key" system. (Really: key agreement)

How can Alice & Bob agree on a key K over channel, while Eve is eavesdropping?

Parameters: n = # of puzzles
$D = 2^d$ = puzzle difficulty

① Bob makes n puzzles of difficulty D

$P_i \equiv (y_i, E_{x_i}(K_i))$
where
$h(i \| x_i) = y_i \in \{0,1\}^d$
K_i is session key $\in \{0,1\}^{160}$
$x_i \in \{0,1\}^d$

$$P_1, P_2, \ldots, P_n$$

& sends them all to Alice (& Eve)

② Alice picks random i $(1 \le i \le n)$ & solves P_i (work D for Alice)

③ Alice lets Bob know (but not Eve) which one she has solved, e.g. by sending e.g. $h(K_i)$

④ Further communications protected with session key K_i.

Time for good guys $= \underbrace{O(n)}_{\text{Bob}} + \underbrace{O(D)}_{\text{Alice}}$

Time for Eve $\geq O(n \cdot D)$

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 $c \geq d$; better if $c \geq 2 \cdot d$ ...]
- Choose "message block size" b (e.g. $b = 512$ bits)
- Design "compression function" f

  $$f : \{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c$$

  [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

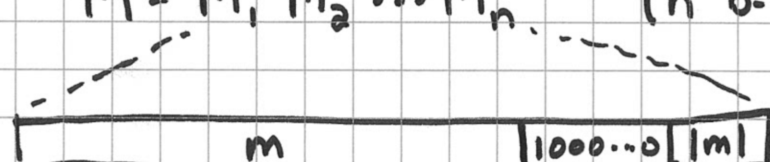
  [Note that c_0 is fixed & public.]

  * [Padding] Given message, append

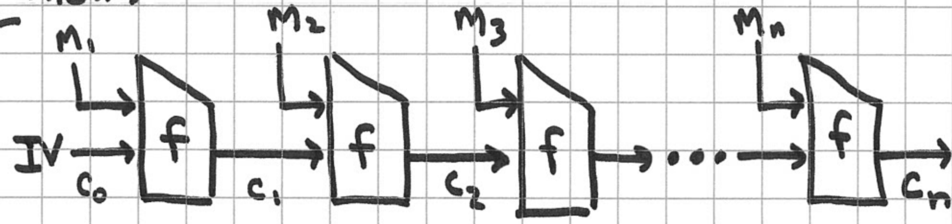
  - 10^* bits
  - fixed-length representation of length of input

  so result is a multiple of b bits in length:

  $$M = M_1 M_2 \cdots M_n \quad (n \text{ b-bit blocks})$$



Then:



h

$h(m) = c_n$ truncated to d bits

<u>Theorem:</u> If f is CR, then so is h.

<u>Proof:</u> Given collision for h, can find one for f by working backwards through chain. ■

<u>Thm:</u> Similarly for OW.

<u>Common design pattern for f:</u>

$$f(c_{i-1}, M_i) = c_{i-1} \oplus E(M_i, c_{i-1})$$

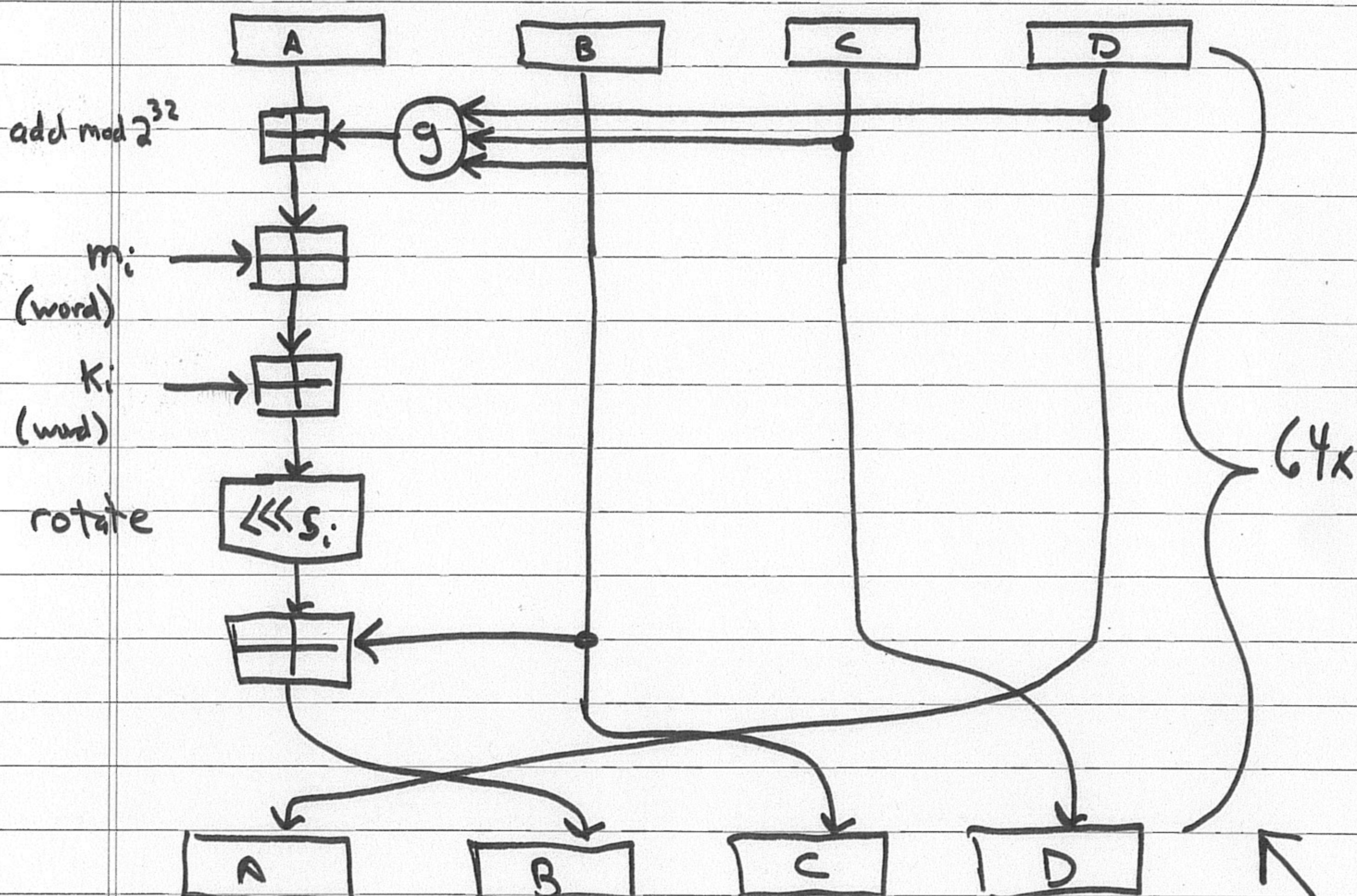
where $E(K, M)$ is an encryption function (block cipher) with b-bit key and c-bit input/output blocks.

(Davies-Meyer construction)

6.857 Rivest

Typical compression function (MD5):

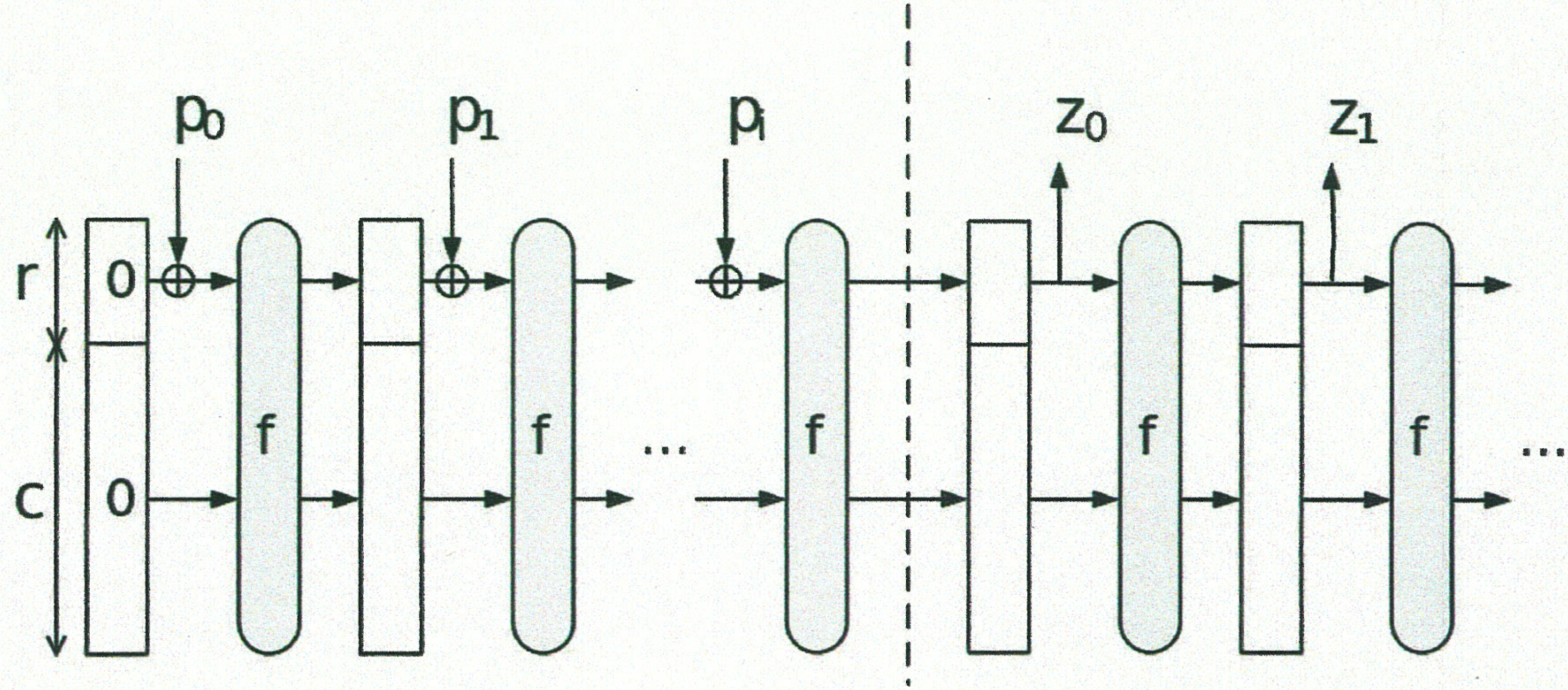
- chaining variable & output are 128 bits = 4 x 32
- IV = fixed value
- 64 rounds; each modifies state (in reversible way) based on selected message ~~block~~ word
- message block b = 512 bits considered as 16 32-bit words
- uses end-around XOR too around entire compression fn (as above)



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

$$g(x,y,z) = \begin{cases} xy \vee \bar{x}z \\ xz \vee y\bar{z} \\ x \oplus y \oplus z \\ y \oplus x\bar{z} \end{cases}$$ depending on round

Keccak = SHA-3



Keccak Sponge Construction

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

e.g. $d = 256$
$c = 512$
$r = 1088$
$w = 64$