TOPIC: 6.857

DATE: 2/12/2020

FILE UNDER: Crypto Hash Fns II

PAGE: L04.1

Admin:

Final projects!

Today:

- Merkle Trees
- Merkle Puzzles
- PK crypto based on Merkle puzzles
- Constructions
  - Merkle-Damgard
  - Keccak (SHA-3)

Readings:

Katz & Lindell: Chapter 5

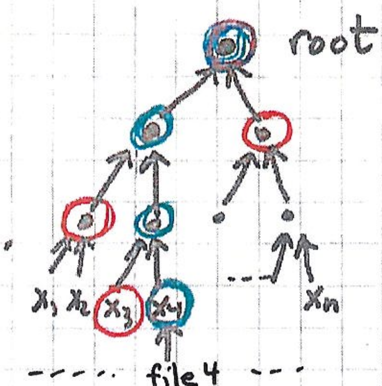
Paar & Pelzl: Chapter 11

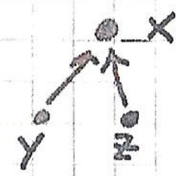
Ferguson: Chapter 5

Wikipedia: SHA-3

(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 "**Merkle tree**":





value at x
$= h$(value at y // 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> **CR**

Used in bitcoin ...

Puzzles & Brute-Force Search

Want to create puzzle with solution nown to creator that requires (on average) a fixed amount of work to solve.

Let $h: \{0,1\}^* \rightarrow \{0,1\}^d$ be a crypto hash fn
(e.g SHA-256 with $d = 256$)

The "puzzle" will be to invert h, i.e. solve $h(x)=y$ for x given y.

restricted domain

To make this a puzzle, we restrict x to be in a known set S of possible solutions. E.g. $S = \{0,1\}^s$ for $s = 40$.

To create a puzzle, pick $x \in S$ at random, compute $y = h(x)$.

Difficulty of solving $\approx |S|/2$ by brute-force search.

If $s \ll d$ there will be no "false solutions" – no collisions.

Can create multiple (keyed) puzzles (k, y) means solving $h(k \| x) = y$ for $x \in S$.
Puzzle spec is (h, k, s, y)

Puzzle creator knows solution

restricted range

= Can also have puzzles where creator doesn't know solution with truncated hashes

$$h: \{0,1\}^* \rightarrow \{0,1\}^s$$

try x at random until $h(x) = y$
Expected work is 2^s.

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

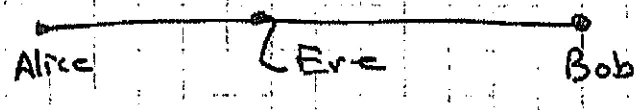
  r = variable to be solved for

$h(k,r) = h(k \| r)$

- 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 ☹

Merkle puzzles

- First "public key" system (really: key agreement)



Eve is passive eavesdropper.
How can Alice & Bob agree on a key?

Use puzzles (with restricted domain, so have unique solns)

$n = \#$ puzzles of difficulty $2^{s-1} = D$ each

(1) Bob choose n values $x_1, x_2, \ldots, x_n$ from $S = \{0,1\}^s$

Bob computes $y_i = h(i \| x_i)$

Bob sends $\underbrace{(y_i, E_{x_i}(K_i))}_{\text{puzzle } P_i}$ to Alice for $1 \le i \le n$, where $K_i \in_R \{0,1\}^{256}$

(2) Alice picks random i from $[n] = \{1, 2, \ldots, n\}$
Alice solves P_i for x_i
" decrypts to obtain K_i
" sends $h(K_i)$ to Bob

(3) Bob & Alice use K_i to communicate secretly from then on.

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

Time for Eve = $\Theta(n \cdot D)$

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

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

  FIL $c+b$
  FOL c

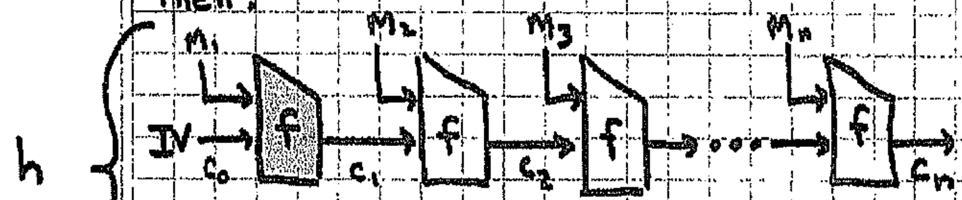
  [f should be OW, CR, PR, NM, TCR, ...]
- Merkle-Damgard is essentially a "mode of operation" allowing for variable-length inputs:
  - * Choose a c-bit initialization vector IV, c_0

    [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:

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

Then:



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

Theorem: If f is CR, then so is h.

Proof: Given collision for h, can find one for f by working backwards through chain. ∎

Thm: Similarly 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.

(Davies-Meyer construction)

E based on "ARX" instructions

A = addition $\quad x \leftarrow x + y$

R = rotation by fixed amt $\quad x \leftarrow x <<< 5$

X = xor $\quad x \leftarrow x \text{ xor } y$

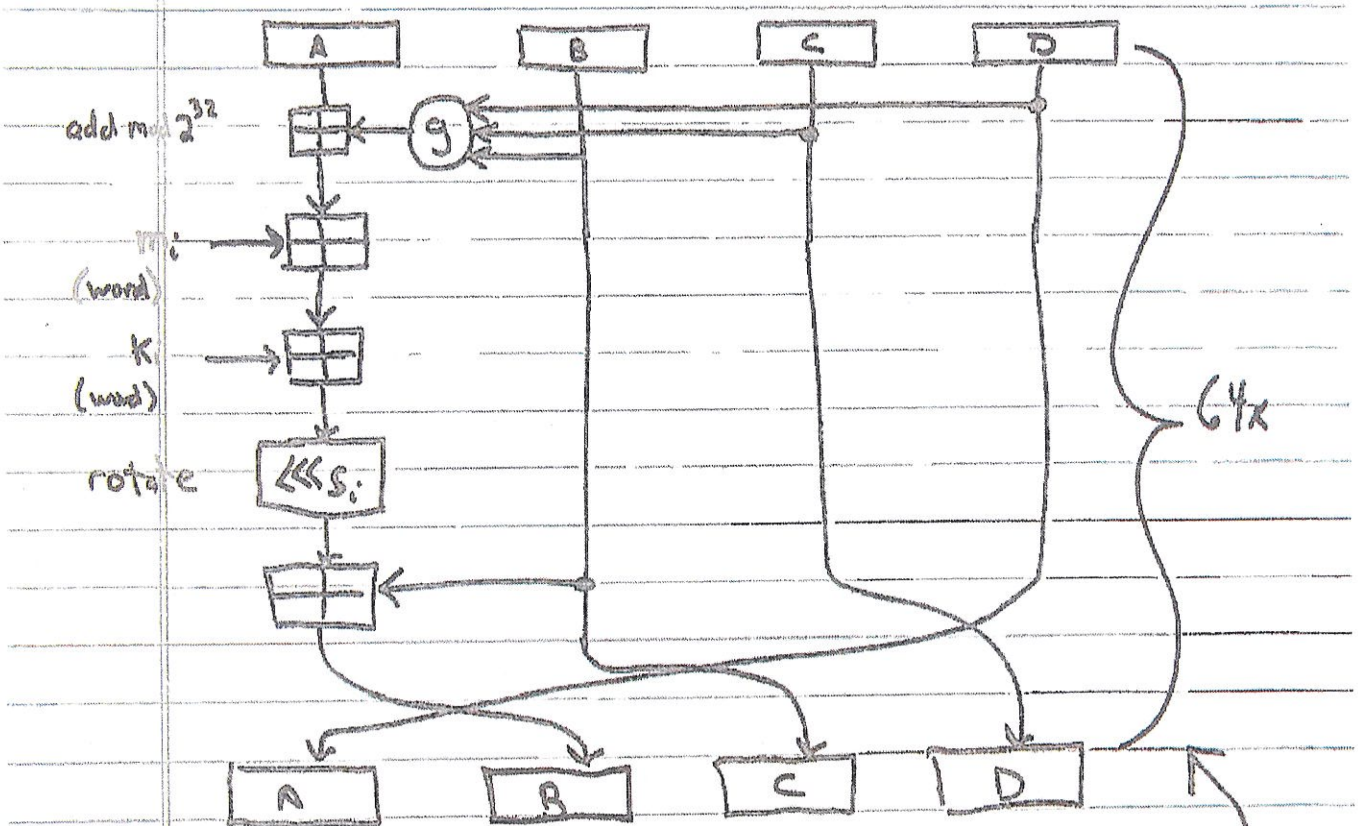
(also "AND" and "OR")

Constant time (no side-channel attacks)

Typical compression function (MD5):

6.857 Rivest
[redacted]

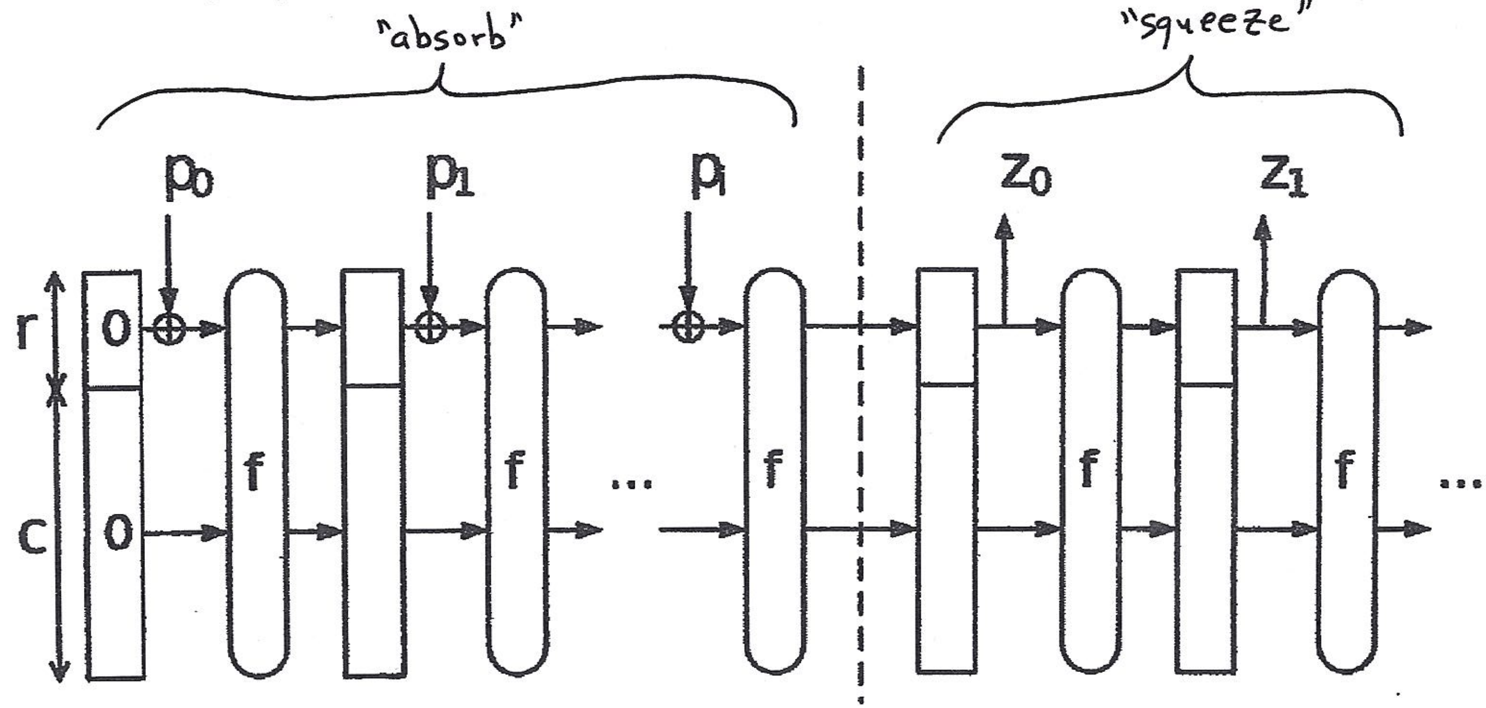
- chaining variable & output are 128 bits = 4 x 32
- IV = fixed value
- 64 rounds; each modifies state (in reversible way) based on selected message ~~block~~ word
- message block b = 512 bits considered as 16 32-bit words
- uses end-around XOR too around entire compression fn (as above)



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

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

SHA-3 "sponge construction"



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. $d = 256$
$c = 512 = 2d$
$r = 1088$
$w = 64$

SHA3-256

Also: SHA3-224
-384
-512

Keccak = SHA-3

TOPIC: 6857

DATE: 2/12/2020

FILE UNDER:

PAGE:

Thm: Can find SHA-3 collision in time $2^{c/2}$

Pf: Find 2 inputs that collide on capacity c bits

(time = $2^{c/2}$ by birthday paradox)

Use ability to arbitrarily tweak 1^{st} r bits to

make collision. $\boxtimes$