TOPIC: 6.857 - Computer & Network Security

DATE: 2/19/14

FILE UNDER:

PAGE: L5.1

Admin:

Pset #1 due, & pset #2 out, on Monday 2/24.

Lectures by TA's next week (secret sharing & bitcoin)

Submit passwords (not real ones) to TA's. for pset #2. today

Project idea:

"Format-Transforming Encryption"
Shrimpton 2014 Real-Word crypto talk
slides in our "top-secret" folder
also see https://fteproxy.org

Today:

Crypto hash fns: applications & constructions

Applications:
- signatures
- commitments
- Merkle trees
- Payword
- Hash-cash

Construction:
- Merkle-Damgard
- Sponge function

③ Digital signatures ("hash & sign")

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

SK_A = Alice's secret key (for signing)

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

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

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 <u>CR</u> (Else Alice gets Bob to sign x, where $h(x) = h(x')$, then claims Bob really signed x', not x.)

- Don't need <u>OW</u> (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.)

TOPIC

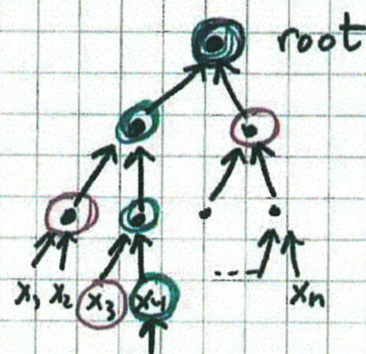
DATE

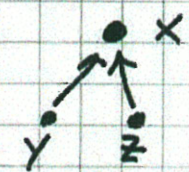
FILE UNDER

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 for 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> <u>CR</u>

TOPIC:

DATE: 2/18/14

FILE UNDER:

PAGE: L5.6

Hash-cash (by Adam Back)

- "Proof of work" by email sender
- Intent: reduce spam by making email "expensive" (computational)
- Sender must solve puzzle:

  find r s.t.

  h(sender, recipient, date, time, r)

  ends in 20 zeros

- include r in header as "proof of work/payment"
- each for recipient to verify
- takes about 2^{20} trials to solve for r
- doesn't work against bot-nets

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

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

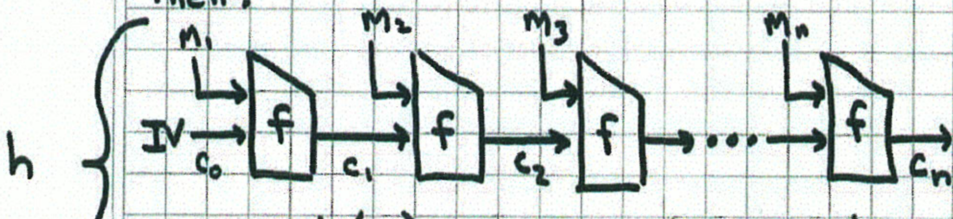
  [f should be OW, CR, PR, NM, TCR, ...]
- Merkle-Damgard is essentially a "mode of operation" allowing for variable-length inputs:
  * Choose a c-bit initialization vector IV, c_0
    [Note that c_0 is fixed & public.]
  * [Padding] Given message, append
    - 10^* bits
    - fixed-length representation of length of input

  so result is a multiple of b bits in length:

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

m	1000...0	|m|

Then:



$h(m) = c_n$ 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.

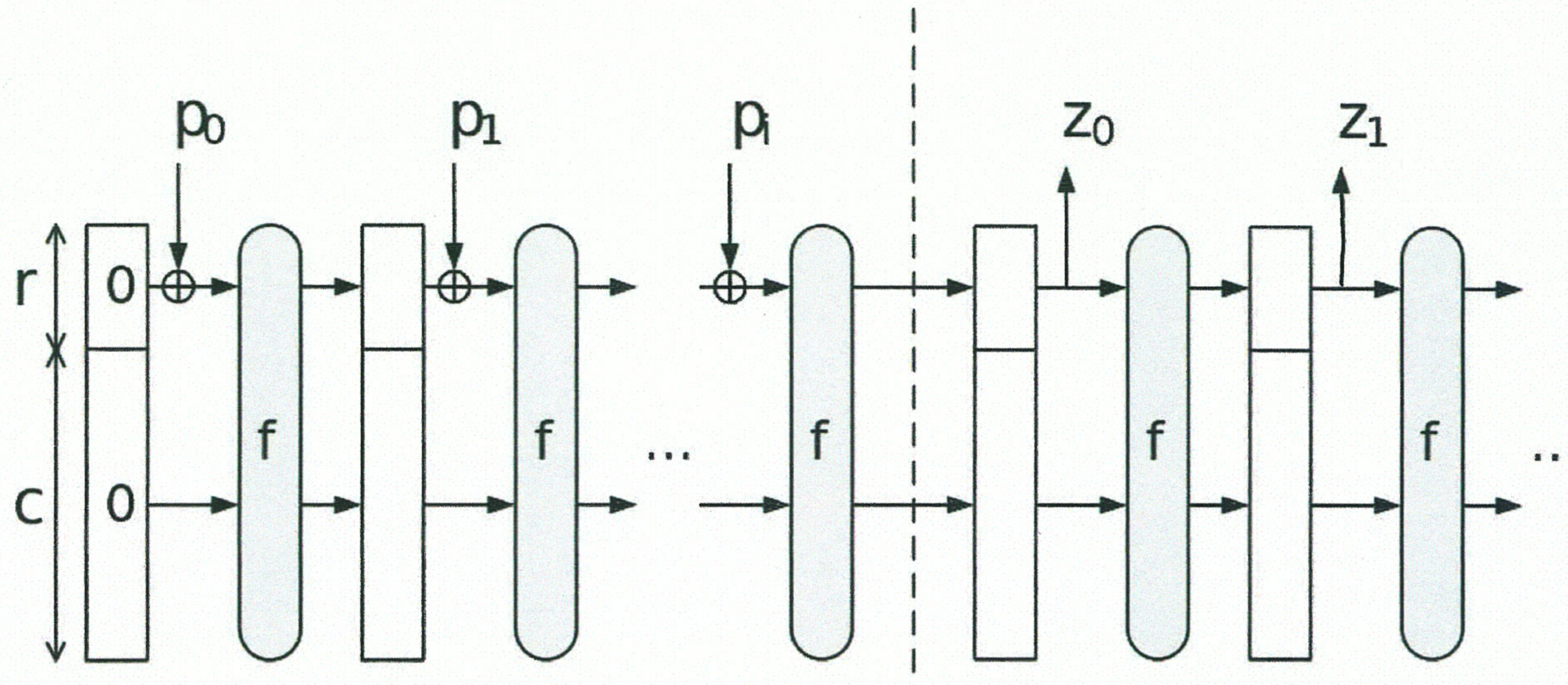
<u>**Common design pattern for f:**</u>

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

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

(Davies-Meyer construction)

Keccak



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