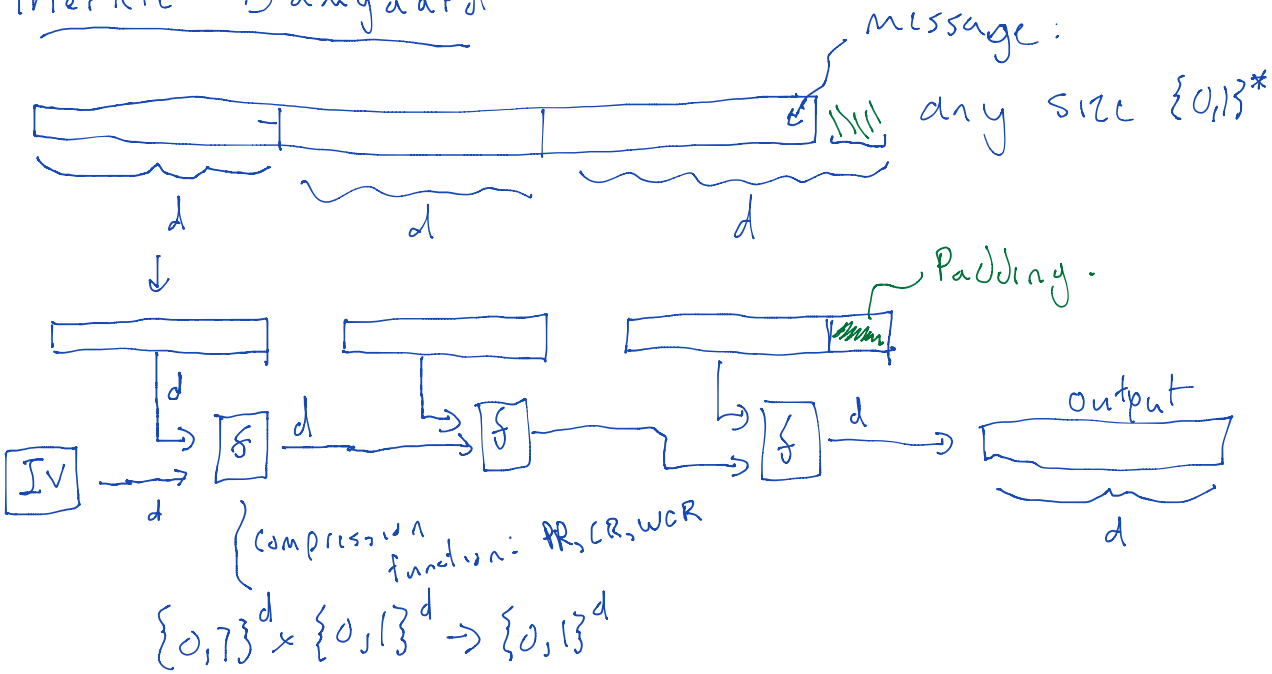
Merkle - Damgaard



IV a number chosen by NIST (always the same)

Padding

* Be careful not to break CR through padding

* Consider: fill end with all zero's (zero padding)

$m = 1000000c0$ ← same block

$m' \neq m$ $\quad m' = 1000000\ldots0$

↳ $h(m) = h(m')$

* Padding:

Last block: $mmmmm'1000\ldots0$ len(m)

↑ fixed size

Example: $M \,\|\, 10000\ldots0$ (len(m))

m

d

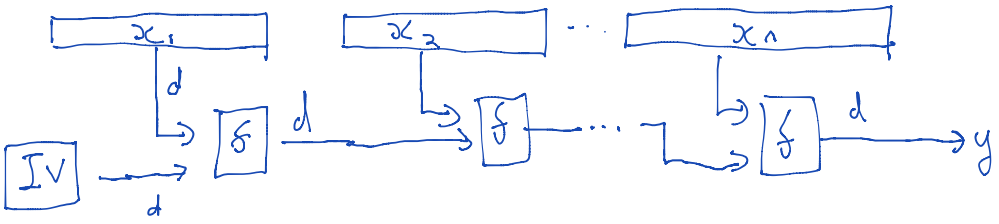
"Issue" with Merkle-Damgard

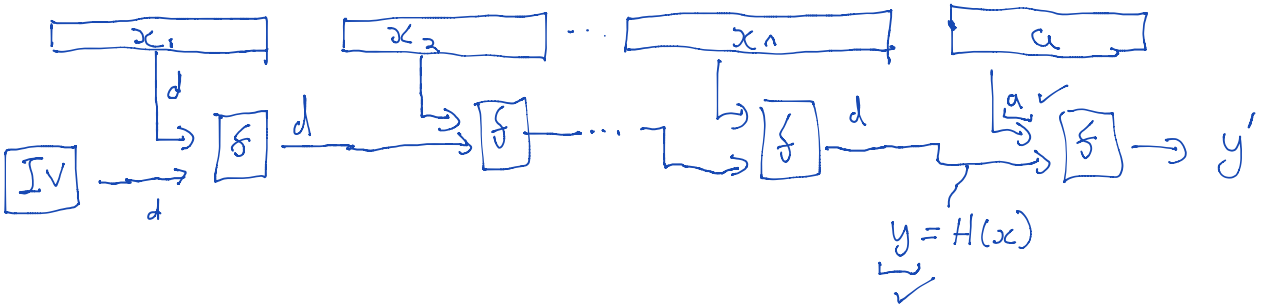
↳ length extension attack (LEA)

↳ Given $y = H(x)$ but not x.

choose any value a and

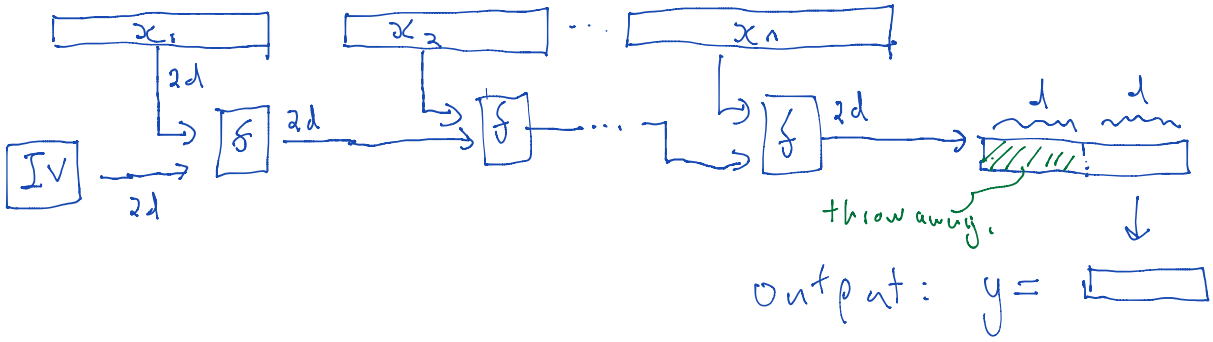
compute $y' = H(x \| a)$





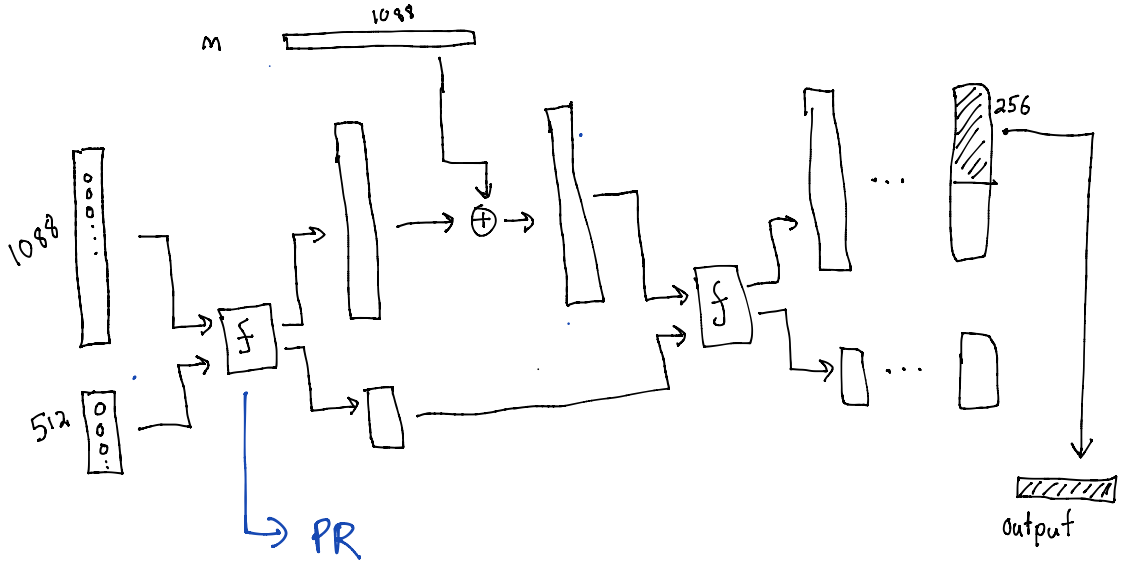
Solutions

① Double (or more) the internal state.



② Different Data Structure

↳ SHA-3: Sponge.



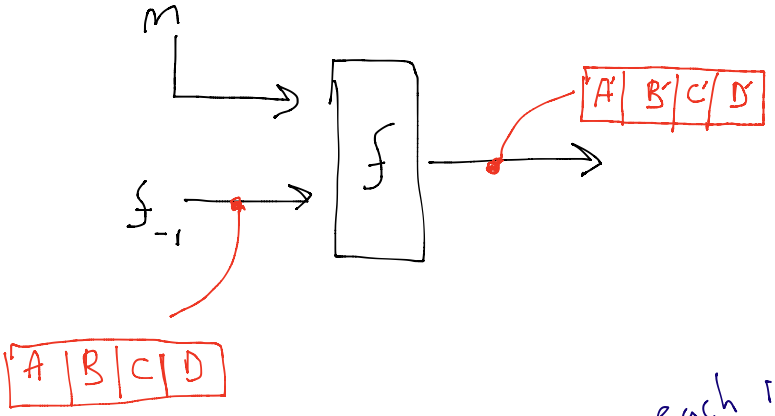
MD5
SHA1
SHA2
} → Merkle Damgaard → LEA

SHA3 → Sponge → No LEA

Example: Compression function of MD4 → very old, insecure, for educational purposes only.

MD4 Compression function

↳ old and insecure → Don't use!



changes each round

changes each time

$$t = A + g_i(B, C, D) + M + \text{const.}$$

$$(A', B', C', D') = (D, t \lll \text{const.}, B, C)$$

Repeat 15 times

Repeat 3 Rounds

bit rotation

changes each time

Each round:

bitwise AND

OR

not

$$g_1 = (B \wedge C) \vee (\neg B \wedge D)$$

$$g_2 = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$$

$$g_3 = B \oplus C \oplus D$$

Xor

Randomness

What is randomness?

↳ uncertainity about a future outcome

What is information?

↳ resolves randomness / reduces uncertainity

↳ more information means for reduction in randomness.

(1) Uncertainity

↳ Probability. (p)

↳ $[0, 1]$

impossible ↑ ↑ certain

↳ Set of outcomes (one will happen)

↳ Probability distribution

$p_1, p_2, p_3 \cdots p_n$

↳ $\Sigma p_i = 1$

② Randomness

↳ $H_2(p_1, p_2, \dots, p_n)$

↳ standard H use H

↳ not a hash

↳ Entropy.

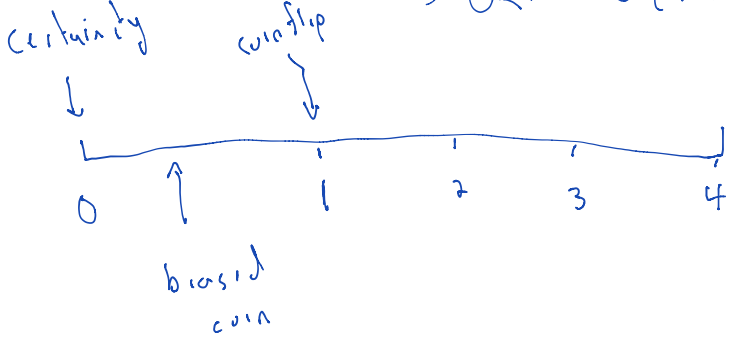
↳ Why entropy looks the way it looks.

1) $H(1) = 0$

2) $H(1/2, 1/2) = 1$ ← one unit of randomness (bits)

↳ coin flip

3) $0 \leq H(p_1, p_2, \dots p_n) \leq \infty$



4) If repeat uncertain event N times, the entropy: ← trials.

$H(\text{event}) \cdot N$

Shannon Entropy:

$$H_2(p_1, p_2, \ldots, p_n)$$

$$= -p_1(\log_2 p_1) - p_2(\log_2 p_2) \cdots - p_n(\log_2 p_n)$$

$\log_2 p_i$: info(p_i)

$$= -\sum_i p_i (\log_2 p_i)$$

$\log_2(57) = \frac{\log_{10}(57)}{\log_{10}(2)}$ ← change of base

Examples: Coin Flip.

$z = \log_2(0.5) = \log_2(1/2)$

$2^z = 2^{-1} \rightarrow z = -1$

$$H(1/2, 1/2) = -(0.5)\log_2(0.5) - 0.5\log_2(0.5)$$

$$= -(0.5)(-1) - 0.5(-1)$$

$$= 1 \text{ bits.}$$

Die: $H(1/6, 1/6 \ldots 1/6) = -\sum_6 (1/6)(\log_2 1/6)$

$$= -6(1/6)(\log_2 1/6)$$

$$= 2.58 \text{ bits}$$

Intuition: 1 coin = 2 outcomes = 2-sided die

6 sided die = 2.58 coins.

2 coins = 4 outcomes = 4-sided die

3 coins = 8 outcomes = 8-sided die

Example: Biased coin

$P_{Heads} = 1/3$

$P_{Tails} = 2/3$

$$H(1/3, 2/3) = -1/3 \log_2(1/3) - 2/3 \log_2(2/3)$$
$$= 0.52 + 0.39$$
$$= 0.9$$

Example: Roll 10 dice

$$H(1 \text{ die roll}) \cdot 10$$
$$= 10(2.58) = 25.8$$

Extractors

$y = Ext(x)$ — input $\{0,1\}^*$, output $\{0,1\}^d$

An extractor converts a large semi-random input of at least $2d$ bits of entropy into an output with d bits of (near) uniform randomness.

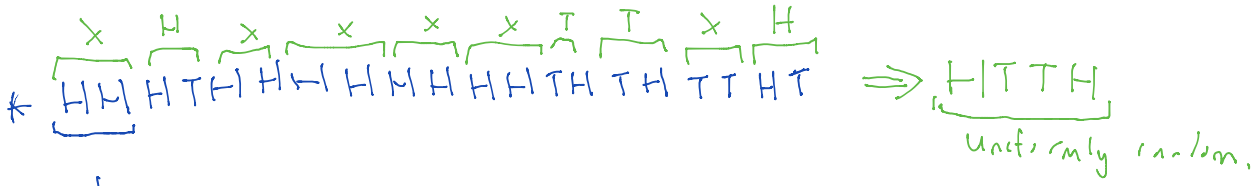
* Note: a d-bit binary string can have at most d bits of entropy
  - ↳ when uniformly random
  - ↳ like d coin flips

* Caveat: $2d$ → min entropy not Shannon entropy.

* Example: Von Neumann extractor
  * doesn't work in general but good for educational purposes.

  * You're playing a game requiring coin tosses but only have a biased coin

coin = { Heads 60%, Tails 40% }



HH	(0.6)(0.6) = 36%	→ toss
HT	(0.6)(0.4) = 24%	→ heads
TH	(0.4)(0.6) = 24%	→ fails.
TT	(0.4)(0.4) = 16%	→ toss

In general, typically use a hash function as extractor

↳ in theory, you should HMAC or CBCMAC

↳ see later

Use cases

① HTTP over SSL/TLS (HTTPS)

② Computer does this.

seed = Ext(X)

seed: statistically-near uniform random
↳ 112 bits

X: harvests entropy from uncertain events (user inputs) over time

Next class: output = PRG(seed)

any size pseudo-randomness