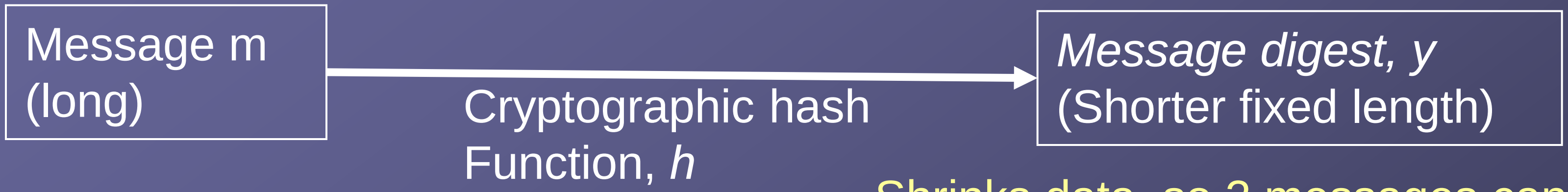
DTTF/NB479: Dszquphsbqiz Day 27

- Announcements:
- Questions?
- This week:
  - Discrete Logs, Diffie-Hellman, ElGamal
  - Hash Functions and SHA-1
  - Birthday attacks

Hash Functions



Message m (long)

Cryptographic hash Function, *h*

Message digest, y

(Shorter fixed length)

Shrinks data, so 2 messages can have the same digest: m_1 != m_2, but $H(m_1) = h(m_2)$

- Goal: to provide a unique “fingerprint” of the message.
- How? Must demonstrate 3 properties:

1. Fast to compute y from m.
2. One-way: given y = h(m), can’t find *any* m’ satisfying h(m’) = y easily.
3. Strongly collision-free: Can’t find any m_1 != m_2 such that $h(m_1)=h(m_2)$ easily
4. (Sometimes we can settle for weakly collision-free: given m, can’t find m’ != m with h(m) = h(m’).

EHA: Easy Hash Algorithm

- Break m into n-bit blocks, append zeros to get a multiple of *n*.
- There are L of them, where L =|m|/n

- Fast! But not very secure.
- Doing a left shift on the rows helps a little:
  - Define $m \mathbin{\leftarrow\!\rfloor} y$ as left-shifting m by y bits
  - Then $m_i' = m_i \mathbin{\leftarrow\!\rfloor} y$

$$m = \begin{bmatrix} m_1 \\ m_2 \\ \ldots \\ m_l \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{l1} & m_{l2} & \cdots & m_{\mathrm{ln}} \end{bmatrix}$$

$$\begin{array}{cccc} \oplus & \oplus & \oplus & \oplus \\ \Downarrow & \Downarrow & \Downarrow & \Downarrow \end{array}$$

$$\begin{bmatrix} c_1 & c_2 & \ldots & c_n \end{bmatrix}$$ =h(m)

$$\begin{bmatrix} m_{11} & m_{12} & \ldots & m_{1n} \\ m_{22} & m_{23} & \ldots & m_{21} \\ \vdots & \vdots & \ddots & \vdots \\ m_{ll} & m_{l,l+1} & \cdots & m_{l,l-1} \end{bmatrix}$$

EHA: Easy Hash Algorithm

3 properties:

1. Fast to compute
2. One-way: given y = h(m), can't find *any* m' satisfying h(m') = y easily.
3. Strongly collision-free: Can't find m_1 != m_2 such that $h(m_1)=h(m_2)$

$$m = \begin{bmatrix} m_1 \\ m_2 \\ \dots \\ m_l \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{l1} & m_{l2} & \cdots & m_{\mathrm{ln}} \end{bmatrix}$$

$$\begin{array}{cccc} \oplus & \oplus & \oplus & \oplus \\ \Downarrow & \Downarrow & \Downarrow & \Downarrow \end{array}$$

$$\begin{bmatrix} c_1 & c_2 & \dots & c_n \end{bmatrix} = h(m)$$

Exercise:

1. Show that the basic (unrotated) version doesn't satisfy properties 2 and 3.
2. Show that the rotated version doesn't satisfy properties 2 and 3 either.

Conclusion: Need nonlinearity!

SHA-1: Secure Hash Algorithm

NSA → NIST

“This standard specifies a Secure Hash Algorithm (SHA), which is necessary to ensure the security of the Digital Signature Algorithm (DSA). When a message of any length < 2^{64} bits is input, the SHA produces a 160-bit output called a message digest. The message digest is then input to the DSA, which computes the signature for the message. Signing the message digest rather than the message often improves the efficiency of the process, because the message digest is usually much smaller than the message. The same message digest should be obtained by the verifier of the signature when the received version of the message is used as input to SHA. The SHA is called secure because it is designed to be computationally infeasible to recover a message corresponding to the message digest. Any change to the message in transit will, with a very high probability, result in a different message digest, and the signature will fail to verify. The SHA is based on principles similar to those used by Professor Ronald L. Rivest of MIT when designing the MD4 message digest algorithm, and is closely modelled after that algorithm.”

(Proposed Federal Information Processing Standard for Secure Hash Standard,” *Federal Register*, v. 57, n. 177, 11 Sep 1992, p. 41727)

…how?

SHA-1: Prepare the message

1. Prepare the message.
   Given m, create mmm…m1000…000xxxxx….x:
   - **Append** a 1 and then enough zeros to make the total congruent to 448 (mod 512) bits (to leave room for the length)
   - **Append** the length of m ($\leq 2^{64}$, so can be written in 64 bits)

   Break into L 512-bit chunks. Each will be used to compress into a 160- bit total message digest.

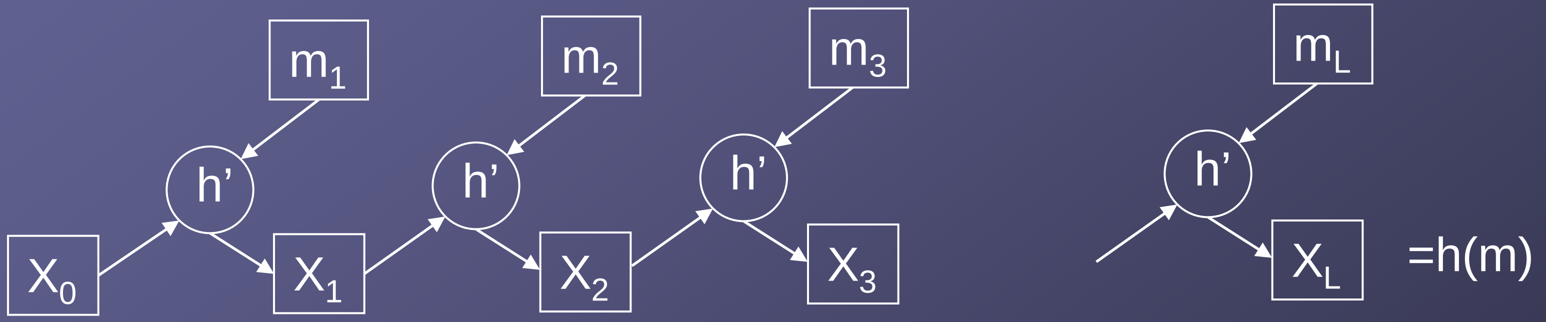
Example: Encode m with length 5000 bits.
What is L?

SHA-1: Notation

$\wedge$	Bitwise AND
$\vee$	Bitwise OR
$\oplus$	Bitwise XOR
$\neg$	Bitwise NOT
↵	Left-shift, with wrap-around
$+$	Addition, mod 2^{32}

SHA-1: Iterative compression

Idea: iterate over all of the L blocks, outputting a value that is a function of the previous output and the current block:



(X_0 is constant)

Now, the function h'…

SHA-1: Compression function: h’

- Input: X_0 (160 bits), m_1 (512 bits): Output: X_1 (160 bits)

Expand m_1 from 512→2560 bits.

$m_1=(W_0..W_{15})$ (32 bits each)

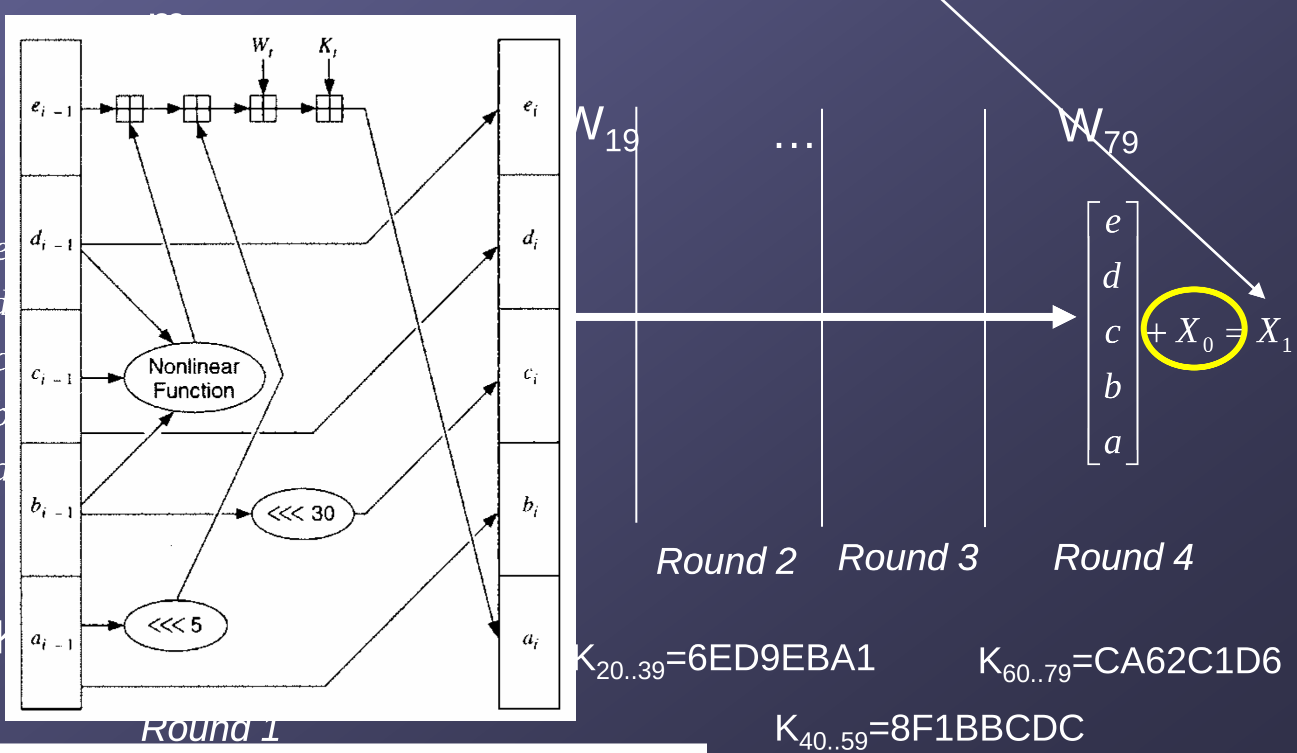
$$W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) \hookleftarrow 1$$

Initialization

$$X_0 = \begin{bmatrix} H_4 \\ H_3 \\ H_2 \\ H_1 \\ H_0 \end{bmatrix} = \begin{bmatrix} e \\ d \\ c \\ b \\ a \end{bmatrix}$$

4 rounds of 20 iterations each:

Each round uses a different K and different nonlinear mixing function f



$K_{20..39}$=6ED9EBA1

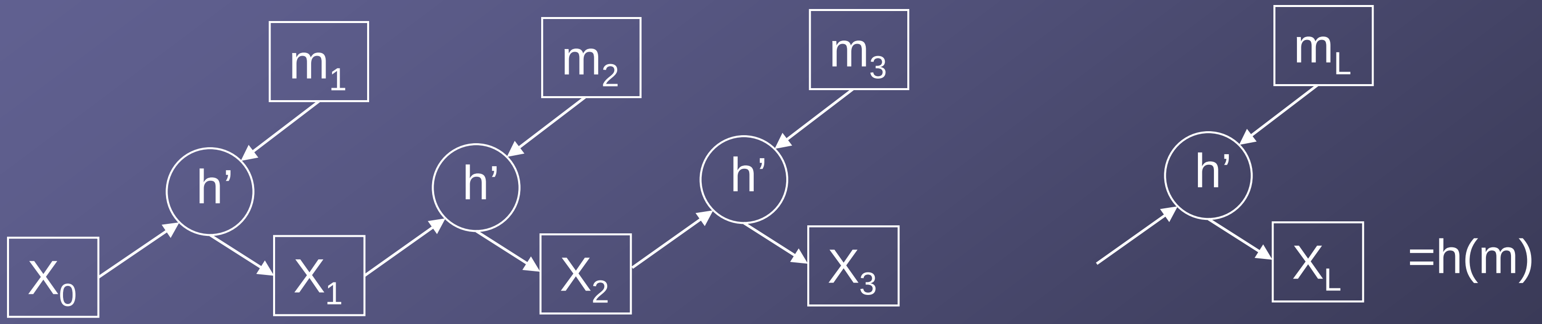
$K_{40..59}$=8F1BBCDC

$K_{60..79}$=CA62C1D6

$$f_t(B,C,D) = \begin{cases} (B \wedge C) \vee ((\neg B) \wedge D) & \text{if } \ 0 \leq t \leq 19 \\ B \oplus C \oplus D & \text{if } 20 \leq t \leq 39 \\ (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) & \text{if } 40 \leq t \leq 59 \\ B \oplus C \oplus D & \text{if } 60 \leq t \leq 79 \end{cases}$$

SHA-1: Iterative compression

Repeat the algorithm on the previous slide L times until you've compressed the whole message into a single 160-bit vector.



Each can be implemented in hardware.

Interesting trivia

- The NSA added the left shift in *w* after the fact. The change “corrects a technical flaw that made the standard less secure than have been thought”.

(Proposed Revision of Federal Information Processing Standard (FIPS) 180, for Secure Hash Standard,” *Federal Register*, v. 59, n. 131, 11 Jul 1994, p. 35317-35318)

Summary

- What’s an attack on SHA-1 look like?
- In other words, how do we find collisions?
- Stay tuned…
    - Next time we’ll learn what birthdays have to do with collisions
- How long before SHA-1 will be broken?