Introduction
Hash functions

Luis J. Dominguez Perez

CRYPTO-CO, Junio 29/Julio 1 de 2016

Agenda I

One-way functions

Hash functions versus MAC functions

Hash functions

Construction

SHA

One-way functions

Definition

A function $f()$ is a one-way function if:

- $y = f(x)$ is computationally easy, and,
- $x = f^{-1}(y)$ is computationally impractical

$$x \xrightarrow{\text{easy}} f(x)$$

$$x \xleftarrow{\text{hard}} f(x)$$

One-way functions: Examples

- Discrete Logarithm
  - Given x, a, y n, it is easy to compute $y = x^a \bmod n$; however, given y, x, y n, finding a is very hard

- Factorisation
  - Given x, and y, it's easy to compute $n = xy$; however, given n, finding the x, and y factors is very hard

- Discrete square root
  - Given x, and n, it's easy to compute $a = x^2 \bmod n$; however, given a, and n, finding x is very hard.

Agenda 2

One-way functions

Hash functions versus MAC functions

Hash functions

Construction

SHA

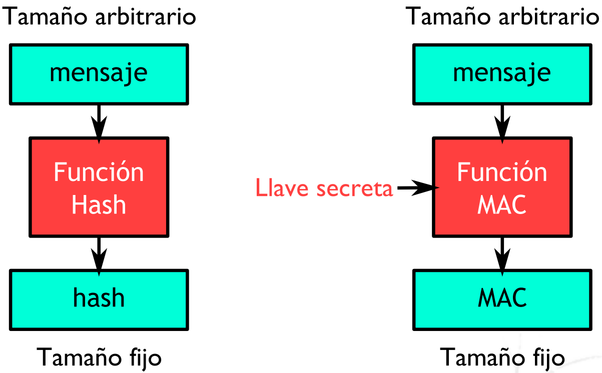
Hash functions

- Hash function
  - Generate fixed size fingerprint from an arbitrary size message
  - There are no involved keys
  - It most be one-way to be useful
- Applications
  - Hash with keys: MAC generation
  - Hash without keys: digital signatures, password files, key stram/ pseudorandom numbers, etc.
- Constructions
  - Iterative hash functions (family of functions MD-4): MD5, SHA-1, SHA-2, HAVAL, HAS160, etc.
  - Block cipher-based hash functions: MDC (Manipulation Detection Code)
  - Sponge hash functions, HAIFA, Merkel-Damgård, Conc-permute, UBI, etc.

Message Authentication Code

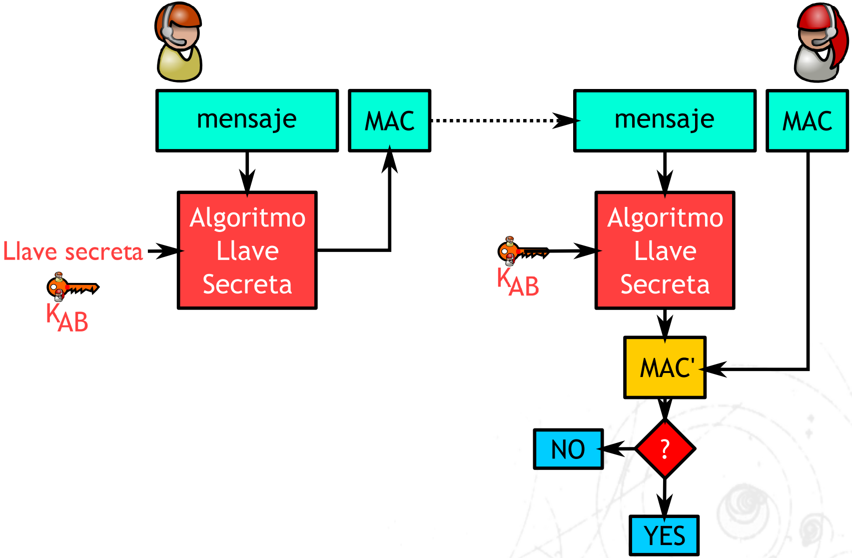
- MAC
    - Generate a MAC of a fixed size given a random size message
    - It's a hash function with key
    - Message origin is authenticated
    - Message integrity is verified
    - The entity is authenticated
- Constructions
    - Hash with key: HMAC
    - Based on encryptors: CBC-MAC
    - Parallelizables: P-MAC

Comparison between hash, and MAC functions

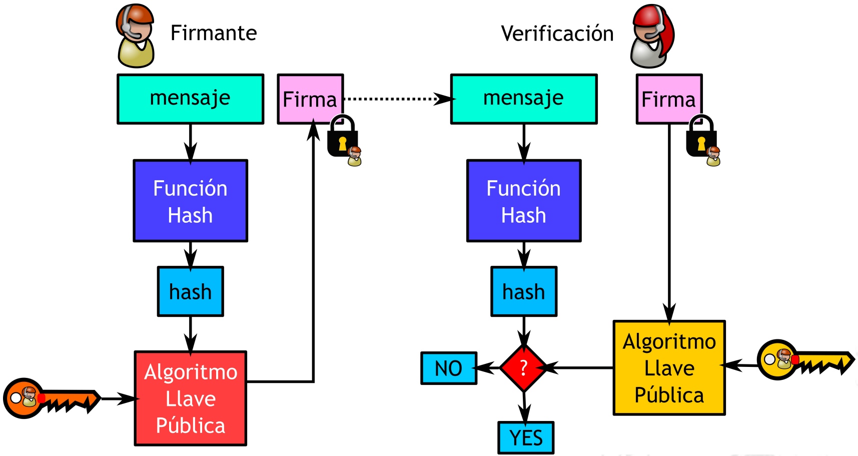


- Easy to compute
- Compression: random size to fixed size
- Function without key versus with a key.

Message authentication using MAC



Digital signature with a hash function



MAC, and digital signature

- MAC
  - Generated, and verified by a private key algorithm
  - Message origin authentication, and message integrity
  - Schemes:
    - Based on keys: HMAC
    - Based on encryptors: CBC-MAC
- Digital Signature
  - Generated, and verifies by a private key algorithm
  - Message origin authentication, and message integrity
  - No-repudiation
  - Schemes:
    - Hash + Digital signature Algorithm
    - RSA, DSA, etc. Signatures

Agenda 3

One-way functions

Hash functions versus MAC functions

Hash functions

Construction

SHA

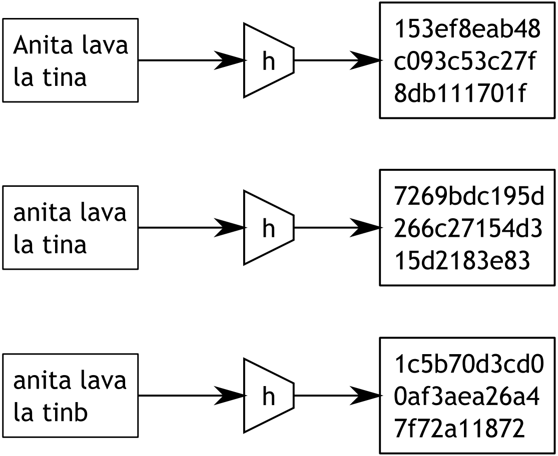
Digest function

- Produce digital fingerprints of fixed length from random length documents
- A small variation in the original text provides a entirely different digital fingerprint
  - Convert passwords to fixed length messages
  - They are used to generate random numbers
  - Provide basic authentication through the use of MACs (Message autentication code)
  - Basic blocks for digital signatures.

Digest function - black box

The hash function, or digital fingerprint recives a text of undefined length (normally, the message is completed with zeros until matching a size that can be broken in blocks of specific size), and the output is of fixed length.

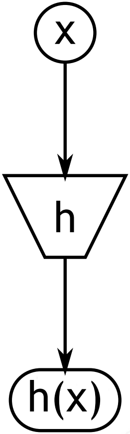
Digest function - black box 2



A cryptographically useful digest function provides totally different digests despite minimum changes.

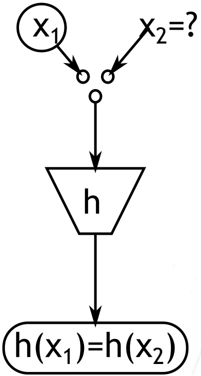
Security requirements of the hash functions - 1

- Pre-imagen resistance (one wayness)
  Given y, it's computationally unpractical to find any x value, such that $y = h(x)$



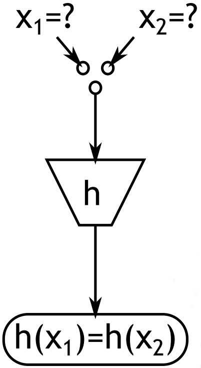
Security requirements of the hash functions - 2

- Second Pre-imagen resistance (weak resistance to collisions)
  Given x, it's computationally unpractical to fond another
  $x' \neq x$, such that $h(x) = h(x')$



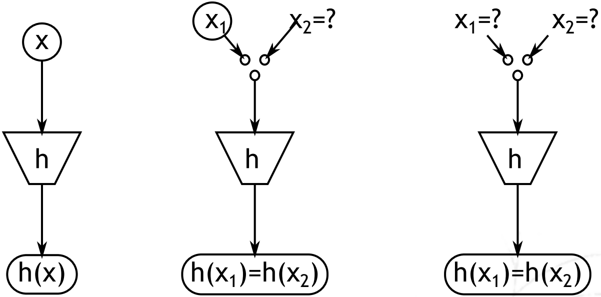
Security requirements of the hash functions - 3

- Collision resistance (strong resistance to collisions)
  It's computationally unpractical to find two values x, and x such that $h(x) = h(x')$



Besides being efficiently computable.

Security requirements of the hash functions - 4



Agenda 4

One-way functions

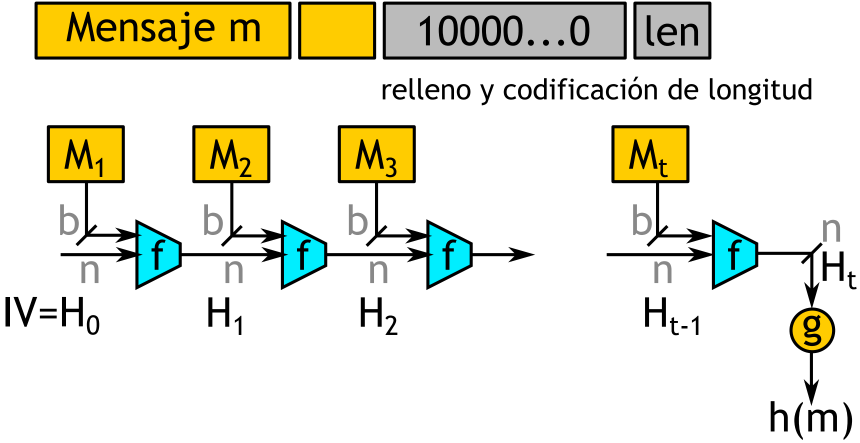
Hash functions versus MAC functions

Hash functions

Construction

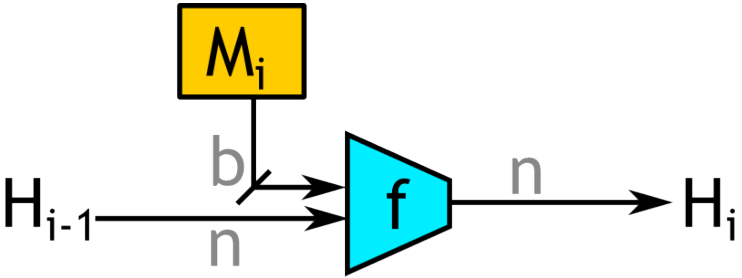
SHA

Function construction



- IV - Initial Vector/Value
- H_i - Chaining variable
- M_i - Input Block
- f - Compression function
- g - optional process
- t - Input Blocks
- b - Bits from Block

Function construction - 2

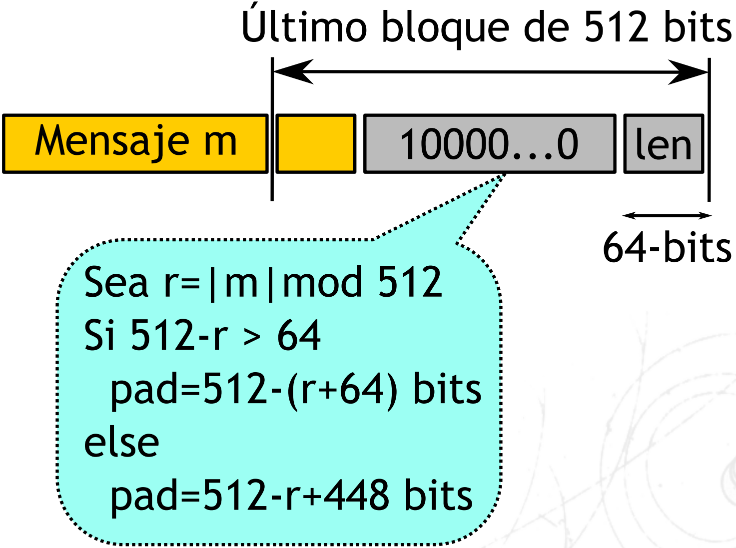


Complete Hash function:

- $H_0 = IV$
- $H_i = f(H_{i-1}, M_i) \, \forall 1 \leq i \leq t$
- $H(M) = h(H_t)$

Filling

Assuming a fixed length block of 512-bits:



Agenda 5

One-way functions

Hash functions versus MAC functions

Hash functions

Construction

SHA

Secure Hash Algorithm

- SHA ws designed by the NIST (National Institute of Standards and Technology), and the NSA (National Security Agency)
- This is the USA's standard for the DSA (Digital Signature Algorithm)
- The algorithm name is SHA, whereas SHS is the standard's name
- It's based in MD4, and MD5
  - SHA-0, FIPS PUB 180, 1993
  - SHA-1, FIPS PUB 180-1, 1995
    - message bit rotation
    - used by SSL/TLS, PGP, SSH, S/MIME, and IPSec.
  - SHA-2, FIPS PUB 180-2, 2001
    - SHA-224, SHA-256, SHA-384, SHA-512
    - At the beginning, it wasn't as popular as SHA-1
    - (took a while to be used widely)

SHA-1 Algorithm

Break the D-document in 512 bits blocks, filling with zeros if necessary
Set the initial $h_0 \ldots h_4$ values
loop
    Break one 512 bits block in sixteen 32-bits words.
    Generate eighty 32-bits words from this block
    for $i \leftarrow 0 \ldots 79$ **do**
        $a \leftarrow h_0$, $b \leftarrow h_1$, $c \leftarrow h_2$, $d \leftarrow h_3$, $e \leftarrow h_4$
        Compute f using $\oplus$, $\wedge$ sobre a, b, c, d, e.
        Mix a, b, c, d, e rotating some bits, by permutation, and add f, and w_i to a
    end for$h_0 \leftarrow h_0 + a$, $h_1 \leftarrow h_1 + b$, $\ldots$, $h_4 \leftarrow h_4 + e$
end loop
return $h_0||h_1||h_2||h_3||h_4$

Initial SHA-1 values

- Initial values
  - $A = 67452301$
  - $B = EFCDAB89$
  - $C = 98BADCFE$
  - $D = 10325476$
  - $E = C3D2E1F0$
- Constants K_t
  - $t = 0 \ldots 19 \quad K_t = 5A827999$
  - $t = 20 \ldots 39 \; K_t = 6ED9EBA1$
  - $t = 40 \ldots 59 \; K_t = 8F1BBCDC$
  - $t = 60 \ldots 79 \; K_t = CA62C1D6$
- Boolean function f_t
  - $t = 0 \ldots 19 \quad f_t(B, C, D) = B \cdot C + \bar{B} \cdot D$
  - $t = 20 \ldots 39 \; f_t(B, C, D) = B \oplus C \oplus D$
  - $t = 40 \ldots 59 \; f_t(B, C, D) = B \cdot C + B \cdot D + C \cdot D$
  - $t = 60 \ldots 79 \; f_t(B, C, D) = B \oplus C \oplus D$

SHA-1 rounds