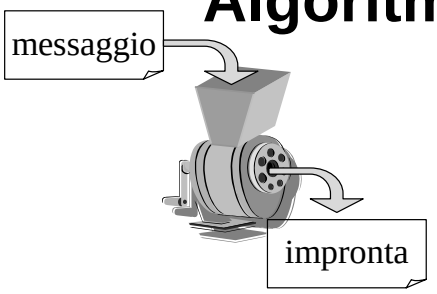
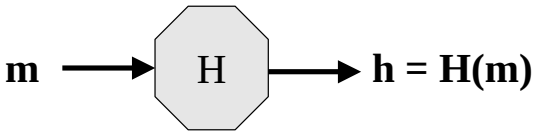
La funzione hash crittografica

- Pseudo Random Number Generator
- Controllo d'integrità
- Controllo d'autenticità
- Riservatezza di una password
- Protocollo a sfida/risposta
- ….

Algoritmi di hash: proprietà





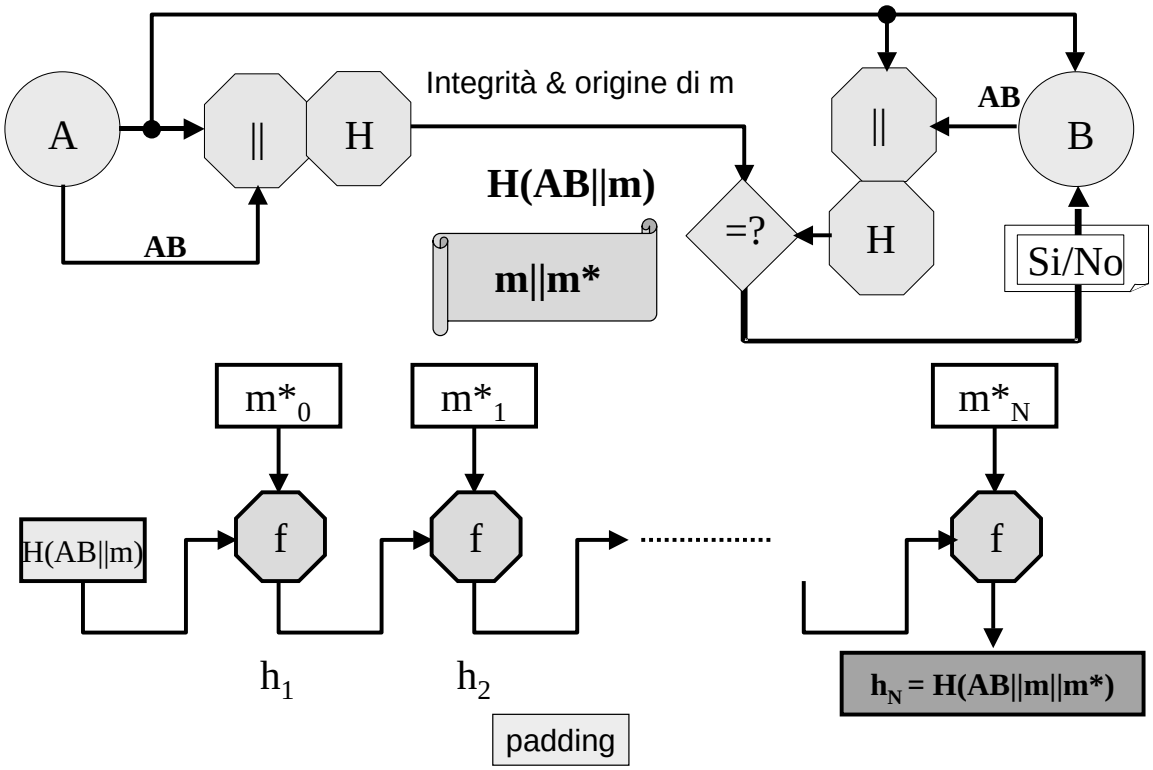
❑R18(efficienza: "*il calcolo di H(x) deve essere facile per ogni x*"

❑R19 (robustezza debole alle collisioni): "*per ogni x deve essere infattibile trovare un y ≠x tale che H(y) = H(x)*"

❑R20 (robustezza forte alle collisioni): "*deve essere infattibile trovare una qualsiasi coppia y, x tale che H(y) = H(x)*"

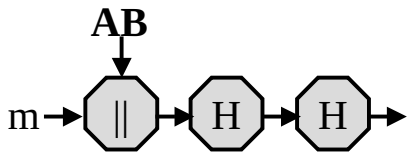
❑R21: "*per ogni h deve essere infattibile trovare un x tale che H(x) = h*"

Attacco con lenght extension

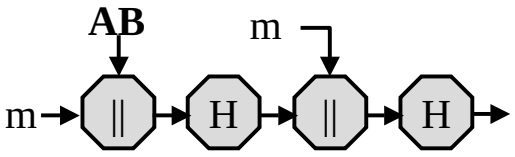


Contromisura: impronta di un'impronta

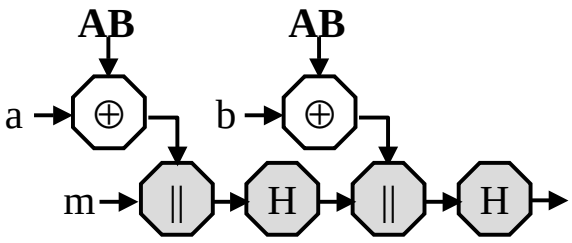
- H(H(AB||m))



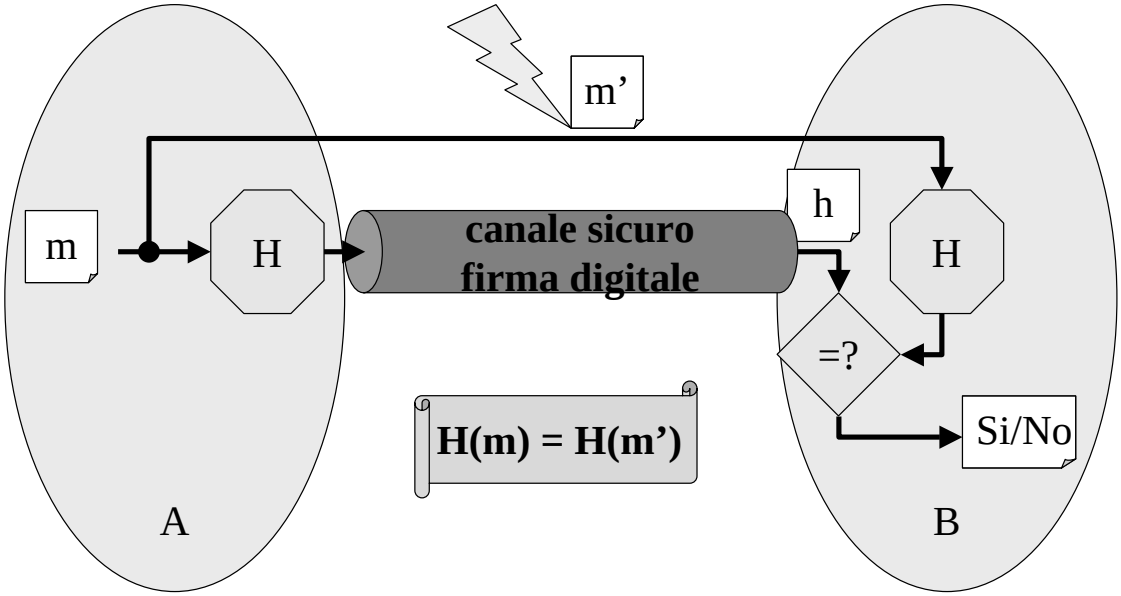
- H(H(AB||m)||m)



- **HMAC(AB, m)**

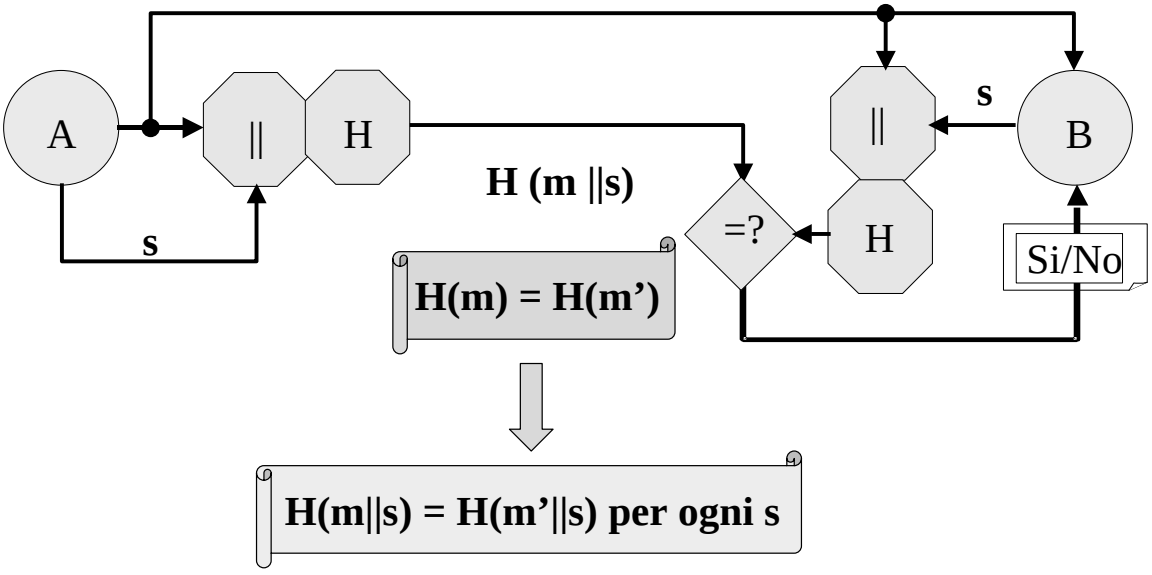


Attacco all'integrità con una collisione



R19 - L'intruso non deve poter forgiare un m' in collisione con m

Attacco al segreto con una collisione



☹ H(m||s) autentica m' !!!

Complessità del calcolo di una collisione

IPOTESI: una funzione hash sottoposta ad ingressi scelti a caso restituisce, con eguale probabilità, uno dei suoi 2^n valori d'uscita.

Problema: individuare un ingresso che fornisca un'impronta assegnata

un tentativo: probabilità di successo $\mathbf{P_1(2^n, 1) = 2^{-n}}$,
probabilità di insuccesso $\mathbf{1 - 2^{-n}}$.
k tentativi: probabilità di successo $\mathbf{P_1(2^n, k) = 1 – (1 - 2^{-n})^k}$

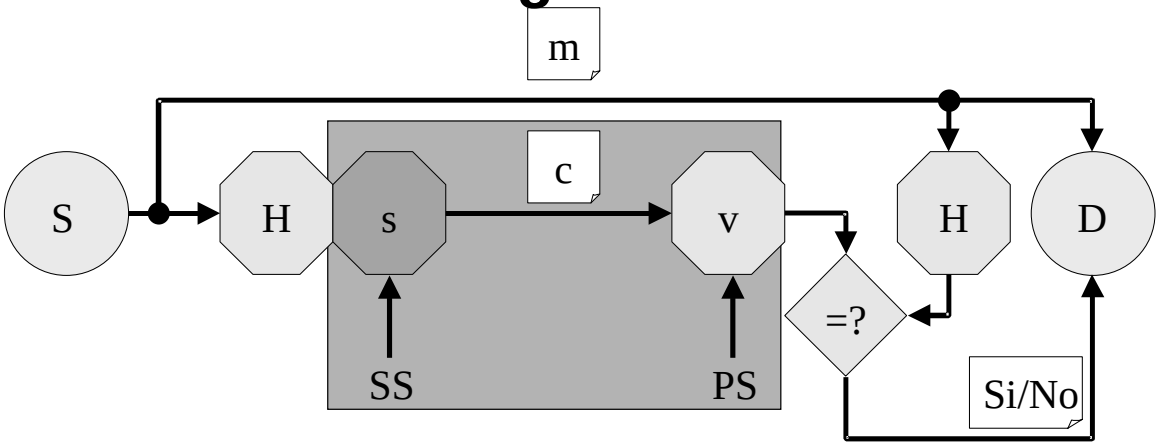
Teorema binomiale: $\mathbf{(1-a)^k = 1- ka + k(k-1)a^2/2! – k(k-1)(k-2)\, a^3/3! +..}$

$\mathbf{P_1(2^n, k)} = \mathbf{k.\, 2^{-n} – k.(k-1).\, 2^{-2n}/2 + k.(k-1).(k-2).\, 2^{-3n}/6 - ..}$ ecc.
$= \mathbf{k.\, 2^{-n}}$ quando $\mathbf{2^{-n}}$ è molto piccolo

S: probabilità di successo desiderata

$$\mathbf{S = P_1(2^n, k) \rightarrow k = S.2^n}$$
$$\mathbf{O(exp(n))}$$
$$\mathbf{n \geq 80}$$

Resistenza alle collisioni e firma digitale



Se è impossibile calcolare collisioni, l'impronta **H(m) identifica m**

Con la firma di H(m) si ottiene:
- efficienza
- individuazione di modifiche a **m** e/o a **c** apportate dall'intruso
- S non può sostenere di aver inviato **m*** e non **m**
- D non può sostenere di aver ricevuto da S un **m*** da lui inventato

Il paradosso del giorno del compleanno

Birthday paradox

Nell'ipotesi che le date di nascita siano equiprobabili, è sufficiente scegliere a caso **253** persone per avere una probabilità > 0,5 che una di queste compia gli anni in un dato giorno.

Sono invece sufficienti **23** persone scelte a caso per avere una probabilità > 0,5 che due o più compiano gli anni nello stesso giorno.

Calcolo di due input in collisioni

$P_2(2^n, k)$ probabilità di due uscite identiche con **$k \leq 2^n$** ingressi scelti a caso
- sequenze d'uscita possibili: **$(2^n)^k$** differenti
- sequenze con valori tutti diversi: **$2^n! / (2^n - k)!$**

$P_2(2^n, k) = 1 - 2^n! / (2^n)^k (2^n - k)! = 1 - 2^n \times (2^n-1) \times (2^n-2) \times \ldots \times (2^n-k+1)/2^{n\,k}$
$= 1 - (1-1/2^n)(1-2/2^n) \ldots (1-(k-1)/2^n)$

N.B. *$(1-x) \leq e^{-x}$, valida per $x \geq 0$, è una buona approssimazione per $x < 0,3$*

$P_2(2^n, k) \cong 1-\exp[-2^{-n}(1+2+..+(k-1)]$
$= 1 - \exp[-2^{-n}(k(k-1)/2)]$ e per k grande
$\cong 1 - \exp[-2^{-n}(k^2/2)]$

IPOTESI**: $P_2 = ½$**

$1 - ½ = \exp[-2^{-n}(k^2/2)]$
$\ln 2 = 2^{-n} \times (k^2/2)$
$k = \text{sqrt}[2 \times (\ln 2) \times 2^n] = 1,18 \times 2^{n/2}$
$O(\exp(n/2))$

Paradosso del compleanno: **k = sqrt(2.ln(2)).sqrt(365) = 22,54**.

Birthday attack alla firma digitale

L'intruso che cerca 2 messaggi con la stessa impronta …

L'attaccante che vuole due messaggi con la stessa impronta
1. genera $2^{n/2}$ piccole varianti del primo messaggio
2. calcola e memorizza gli hash
3. modifica lievemente il secondo messaggio, calcola l'hash e controlla se è in memoria; in caso contrario ripete

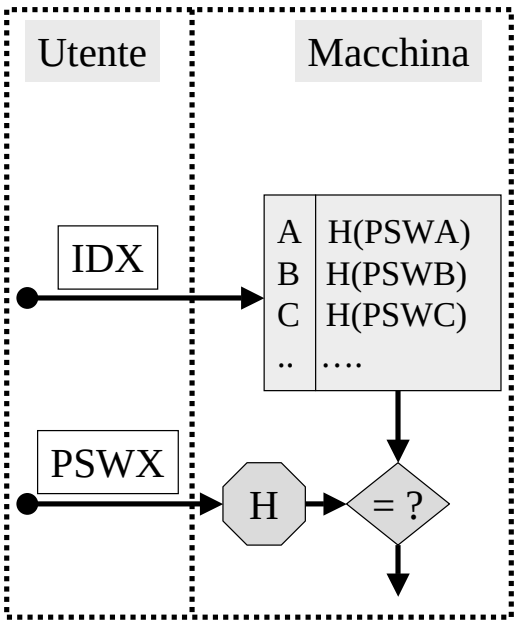
O(exp(n/2))
n ≥ 160

Programma di prova in Java per hash di 40 bit

R20 - L'intruso non deve poter individuare una coppia m,m' in collisione

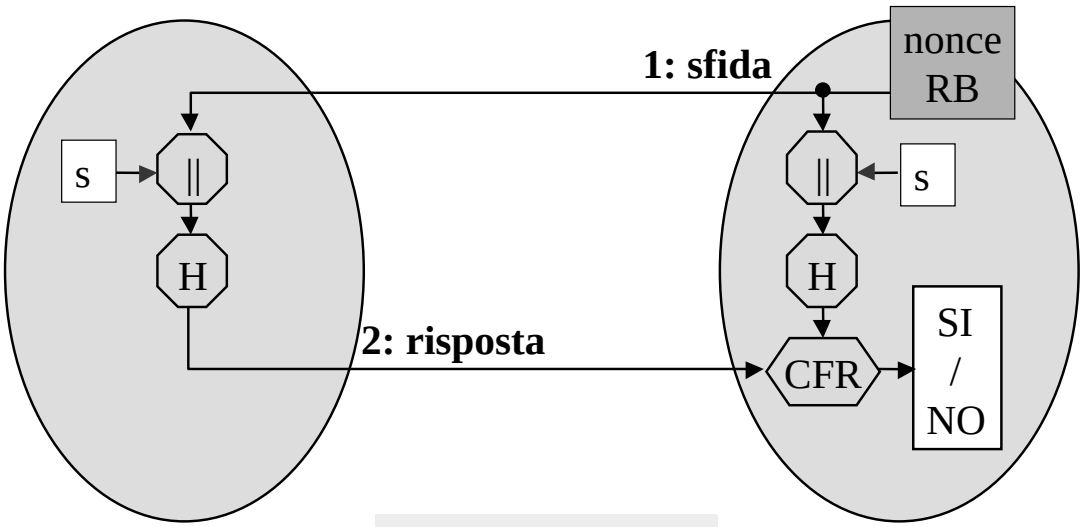
hash delle password

Hash: cifratura senza decifrazione!



R21: hash one-way

Sfida/Risposta (Hash)



R21: hash one-way

Attacco con forza bruta: O(exp(n))

Secure HASH functions

MD5 (1991)
128 bit

SHA-1 (1994)
160 bit

RIPEMD (1996)
160 bit

Tiger (1996)
192 bit

NIST 2002:
SHA-256
SHA-384
SHA-512

Whirlpool (2002)
512 bit