Cryptographic Hash Functions

by Paolo Bernardi

Cosa sono?

$$y = h(x)$$

N → numero di possibili **input**

M → numero di possibili **output**

N >> M (di solito almeno N > 2M)

Input e output sono di <u>lunghezza fissa</u>

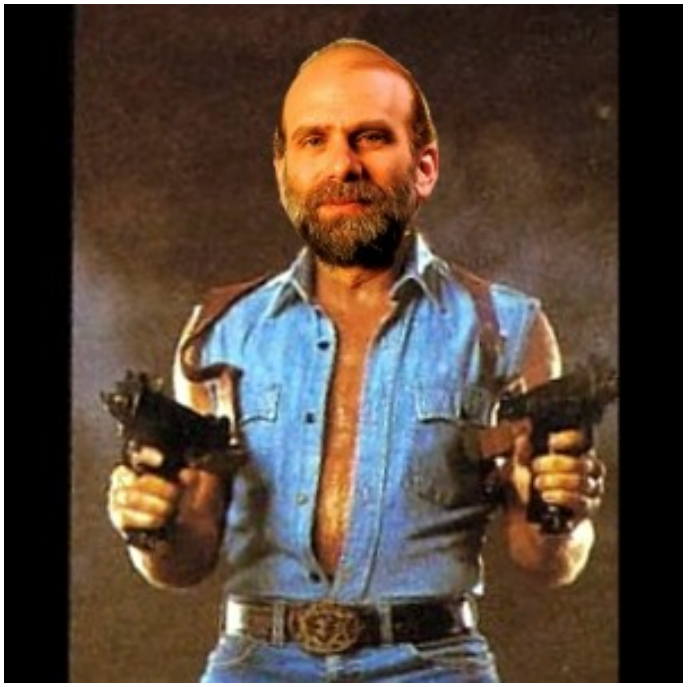
Perché "crittografiche"?

- Le funzioni hash sono usate anche per implementare strutture dati associative (HashMap, dizionari, etc...)

- Quando sono usate per scopi crittografici devono soddisfare requisiti più stringenti

Perché perderci tempo?

"*Much more than encryption algorithms, one-way hash functions are the workhorses of modern cryptography*"



Perché perderci tempo?

- Tante applicazioni:
  - Integrità dei dati
  - Firma digitale
  - Gestione delle password
  - CSPRNG
  - ...
- Poco studiate rispetto alle altre funzioni crittografiche

Perché "crittografiche"?

- Difficile trovare una collisione

$$h(x) = h(x'),\ x \neq x'$$

- Difficile trovare la preimmagine

$$h(x) \rightarrow x$$

Perché "crittografiche"?

- Difficile, ma non impossibile, trovare una collisione: principio dei cassetti
- Difficile, ma non impossibile trovare la preimmagine: forza bruta

Il modello Random Oracle

- Hash crittografico ideale (ma non fattibile)
- Solo l'oracolo conosce la funzione hash
- Non si può ricostruire dai risultati

Random Oracle: collisioni

- Per trovare una collisione devo fare all'oracolo un numero di domande proporzionale alla radice quadrata di M

- Paradosso del compleanno: con 23 persone c'è il 51% di probabilità di averne due con lo stesso compleanno

Random Oracle: preimmagine

- Per trovare la preimmagine di un hash dato devo fare all'oracolo un numero di domande proporzionale ad M

- Resistenza alle collisioni implica resistenza alla preimmagine

Back to the real world

- N >> M, ma non è infinito!

- Data una funzione hash con dominio finito, come si può estenderla ad un dominio infinito?

Funzioni hash iterate

Sia compress una funzione hash

$$\text{compress} : \{0,\ 1\}^{m\ +\ t} \rightarrow \{0,\ 1\}^{m}$$

compress viene estesa in modo da accettare input di qualsiasi lunghezza.

Funzioni hash iterate

Pre-processing: padding

L'input viene allungato in modo da essere divisibile per t

RICORDA: compress : $\{0, 1\}^{m+t} \rightarrow \{0, 1\}^{m}$

Funzioni hash iterate

Processing: the dirty work

1. l'input x è suddiviso in blocchi lunghi t
2. Vettore di inizializzazione (IV) lungo m

RICORDA: compress : $\{0, 1\}^{m + t} \rightarrow \{0, 1\}^{m}$

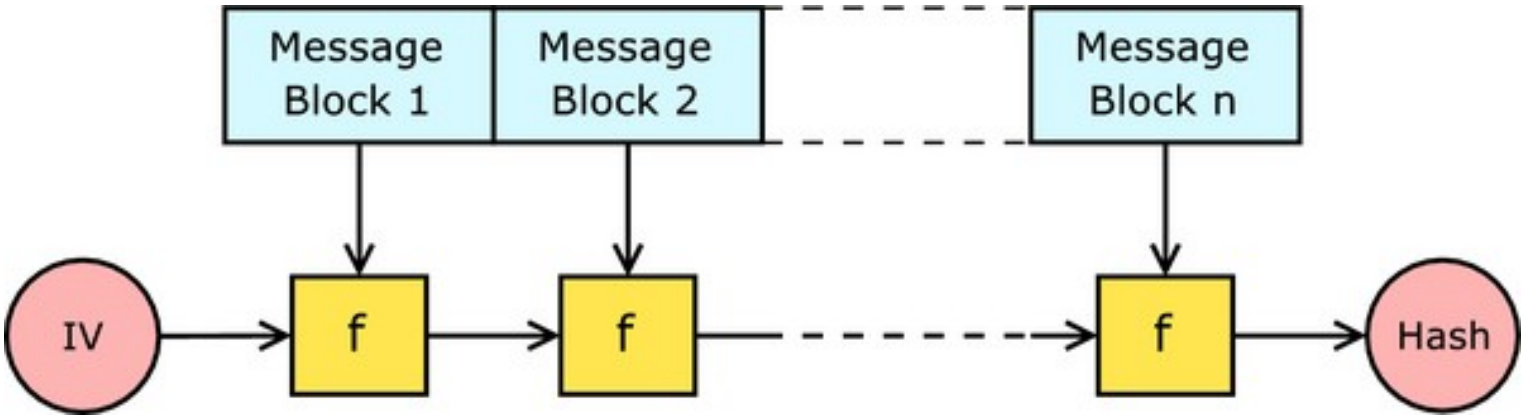
Funzioni hash iterate

$z_0 = IV$

$z_1 = compress(z_0 \mid x_1)$

$z_2 = compress(z_1 \mid x_2)$

(...)



Merkle-Damgård

Funzione hash iterata che gode della seguente proprietà:

compress sicura implica hash sicuro

Lo stato dell'arte: MD5

- Hash di 128 bit
- Correzione di MD4, del quale si può trovare una collisione con 2^8 valutazioni (secondo RO dovrebbero essere 2^{64}!!!)
- Ron Rivest

Attacchi: MD5

- Prima è stata violata la compress
- Attualmente, collisioni in 2^{33} passi (secondo RO dovrebbe essere 2^{64})
- Collisioni usate per creare certificati di CA fasulle riconosciuti come validi dai browser

Lo stato dell'arte: SHA-1

- Hash di 160 bit
- Correzione di SHA-0, subito ritirata per una vulnerabilità teorica notata dopo la pubblicazione
- NSA

Attacchi: SHA-1

- Esiste un metodo per trovare collisioni in 2^{69} passi (secondo RO dovrebbe essere 2^{80})
- Esiste un rimpiazzo: la famiglia SHA-2 (SHA-256, SHA-512...)

Lo stato dell'arte: RIPEMD-160

- Hash di 160 bit
- Correzione di RIPEMD, del quale è stata trovata una collisione
- Creato da un gruppo di ricercatori europei
- Risposta della comunità scientifica a SHA
- Non ha vulnerabilità note

Lo stato dell'arte

- La situazione è tutt'altro che rosea (RICORDA: resistenza alle collisioni implica resistenza alla preimmagine)
- Algoritmi basati su operatori a 32bit e inerentemente sequenziali

Al fuoco!

"*It's time to walk, but not run, to the fire exits. You don't see smoke, but the fire alarms have gone off.*"

Jon Callas, PGP CTO

Back to the future: SHA-3

- Competizione organizzata dal NIST
- Riprende il modello usato per AES
- 2007 / 2012 (revisione di FIPS 180-2)
- Attualmente siamo al 2° round

Back to the future: SHA-3

- Focus su funzioni più sicure
- Focus su hardware moderno (64bit, multicore)
- es. Skein (Schneier, Ferguson et al.), 512bit, basata su Threefish
- Molti partecipanti basati su AES

Back to the future: SHA-3

- Ritiro “eccellente”: MD6 di Ron Rivest
- Problemi di velocità
- Impossibilità di fornire una prova formale di sicurezza

That's all, folks!