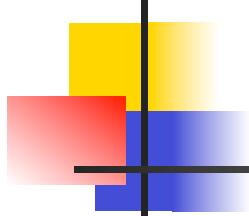


Introduzione

- Una funzione hash è (generalmente) una funzione che comprime.
- Riceve un input di lunghezza arbitraria e restituisce un output di lunghezza fissata (es. 128,160,256 bit).
- Vi sono molti tipi (diversi) di funzioni hash utili alla crittografia.
- Oggi studieremo tali funzioni.

LE PROPRIETÀ DESIDERATE SONO:

- RESISTENZA ALLE COLLISIONI;
- UNIVERSALITÀ;
- UNIDIREZIONALITÀ.

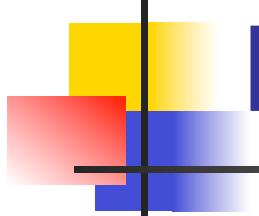


Famiglie di funzioni

- $H: \{0,1\}^k \times D \rightarrow \{0,1\}^n$ famiglia di funzioni
 - Ogni chiave K definisce una mappa

$$h(x)=H_k(x)$$

- K è resa pubblica
- Parleremo di resistenza alle collisioni per **famiglie** di funzioni

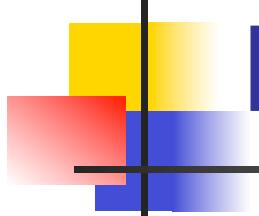


Resistenza alle Collisioni

Def: Una collisione per $H:D \rightarrow R$ è una coppia x_1, x_2 tale che $H(x_1)=H(x_2)$ ma $x_1 \neq x_2$

Se $|D| > |R|$ le collisioni sono inevitabili.

H è resistente alle collisioni se è computazionalmente impossibile trovare collisioni.



Resistenza alle collisioni

- E' sorprendente che tale proprietà sia possibile per funzioni non limitiamo la dim dell'input.
- Anche se ci limitassimo a input di 256 bit ci avremmo un numero ENORME di input che collidono.
- **Principio della Piccionaia:** Se 2^{256} piccioni cercano posto in una piccionaia con 2^{160} posti, almeno due piccioni dovranno condividere il posto.
- In realtà la nostra piccionaia è estremamente sovraffollata.

$$\frac{2^{256}}{2^{160}} = 2^{96} \text{ COLLISIONI MEDIE}$$

Definizione (funzioni Universali o quasi Universali)

- $H: K \times D \rightarrow R$ funzione hash

$Esp_H^{cr0}(A)$

$(x_1, x_2) \leftarrow_R A(); k \leftarrow_R K;$

If $((H_k(x_1) = H_k(x_2)) \text{ and } (x_1 \neq x_2) \text{ and } (x_1, x_2 \in D))$

Return 1

else Return 0

$$Adv^{cr0}(A) = Pr[Esp_H^{cr0}(A) = 1]$$

In tal caso il tempo di calcolo dell'avversario non e' rilevante, quanto in altre definizioni.

Definizione (funz. Universali Unidirezionali)

- $H: K \times D \rightarrow R$ funzione hash

$Esp_H^{cr1-kk}(A)$

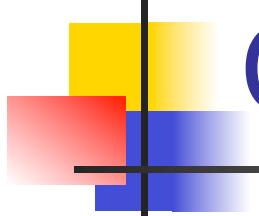
$(x_1, \text{stato}) \leftarrow_R A(); k \leftarrow_R K; x_2 \leftarrow_R A(k, \text{stato});$

If $((H_k(x_1) = H_k(x_2)) \text{ and } (x_1 \neq x_2) \text{ and } (x_1, x_2 \in D))$

Return 1

else Return 0

$$\text{Adv}^{cr1-kk}(A) = \Pr[Esp_H^{cr1-kk}(A) = 1]$$



Osservazioni

- Chiaramente una funzione Universale Unidirezionale e' anche una funzione universale.

$$\text{Adv}^{\text{cr0}}(B) \leq \text{Adv}^{\text{cr1-kk}}(A)$$

- La dimostrazione di questo fatto e' molto semplice.

H_p: SIA H: K × D → R UNA FUNZIONE UNIDIREZIONALE
UNIVERSALE

T_S: H È UNA FUNZIONE UNIVERSALE

PER ASSURDO H NON È UNA FUNZIONE UNIVERSALE
ALLORA ∃ B CON $\text{Ad}_v^{\text{C}^{\infty}}(B) \gg 0$.

A():

$(x_1, x_2) \leftarrow B()$

stato $\leftarrow x_2$

output (x_1, stato)

A(x_1, stato):

$x_2 \leftarrow \text{stato}$

output x_2

$$\text{Ad}_v^{\text{C}^1 - \text{KK}}(A) = \text{Ad}_v^{\text{C}^{\infty}}(B)$$

ASSURDO

Definizione (Funzioni Resistenti alle Collisioni)

- $H: K \times D \rightarrow R$ funzione hash

$Esp_H^{cr2-kk}(A)$

$k \leftarrow_R K; (x_1, x_2) \leftarrow_R A(k);$

If $((H_k(x_1) = H_k(x_2)) \text{ and } (x_1 \neq x_2) \text{ and } (x_1, x_2 \in D))$

Return 1

else Return 0

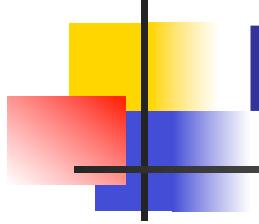
$$\text{Adv}^{cr2-kk}(A) = \Pr[Esp_H^{cr2-kk}(A) = 1]$$

Una funz. res. alle coll. e' anche una funz. universale
unidirezionale. (GUARDARE CASO PRECEDENTE) [SIMILE]

OSSERVAZIONI SULLE TRE DEFINIZIONI

NORMALMENTE LE FUNZIONI HASH SONO
KEYLESS

LE DEFINIZIONI PERDONO DI SIGNIFICATIVITÀ A
LIVELLO PRATICO



Esempio

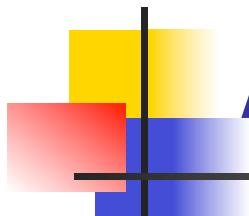
- $H: \{0,1\}^k \times \{0,1\}^{256} \rightarrow \{0,1\}^{128}$

$$H_k(x) = \text{AES}_k(x[1]) \oplus \text{AES}_k(x[2])$$

- Resistente alle collisioni? No!

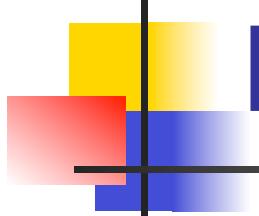
$$x_1 = X \parallel Y \quad x_2 = Y \parallel X$$

$$H_k(x_1) = \text{AES}_k(X) \oplus \text{AES}_k(Y) = H_k(x_2)$$



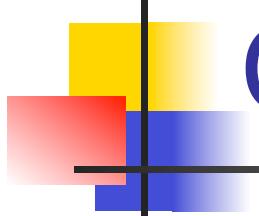
Applicazioni di funzioni hash

- Ingrediente fondamentale di molte primitive crittografiche
- Utile in molte applicazioni di security
- Utile anche in applicazioni che non hanno a che fare con sicurezza/crittografia



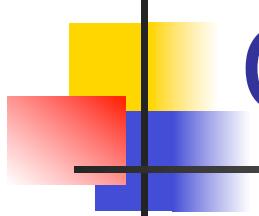
Password Verification

- Client A ha una password pw condivisa col server B
- A può farsi “riconoscere” inviando pw a B attraverso un canale sicuro
 - Es. SSL
- Problema: se il server viene attaccato pw è compromessa



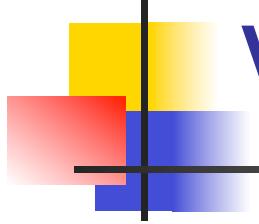
Compare by hash

- A hanno due enormi files FA e FB
- A e B vogliono scoprire se i due files coincidono
- La soluzione ovvia può essere molto dispendiosa.



Compare by hash

- A hanno due enormi files FA e FB
- A e B vogliono scoprire se i due files coincidono
- A calcola $h(FA)$ e lo invia a B.
- B verifica che $h(FA)=h(FB)$.



Virus Protection

- Un eseguibile X è disponibile in diversi siti S_1, S_2, \dots, S_N .
- Quale di essi è affidabile?

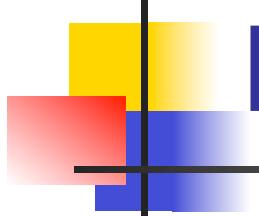
Soluzione:

- Una volta procuratosi $h = H(X)$ (in modo sicuro) si può scaricare X da qualunque sito!

Funzioni hash in pratica: SHA1 (Secure Hash Algorithm 1)

$$\text{SHA1} : \{0, 1\}^{<2^{64}} \rightarrow \{0, 1\}^{160}$$

- Prende in input valori di taglia (praticamente) illimitata e produce output di 160 bit.
- Fu proposta nel 1995
- E' (non più) molto usata in pratica.



La funzione SHA1 – Storia

- SHA1 deriva dalla funzione MD4, proposta da Ron Rivest nel 1990.
- Un'altra funzione (molto usata in pratica) e' MD5
- Gli algoritmi SHA1, MD4 e MD5 sono tutti piuttosto simili.
- Idea: concatenare una funzione di compressione che va da $512 + \ell$ bit a ℓ bit.
- Per MD4 e MD5 $\ell = 128$
- Per SHA1 $\ell = 160$.

La funzione SHA1 – Idee di base

- Si compone di diverse procedure:
 - Inizia col modificare l'input M attraverso la procedura SHAPAD
 - Quindi viene iterata la funzione di compressione $shf1$
- Non descriveremo molto in dettaglio SHA1.
- Ignoreremo del tutto MD4 e MD5.
- Fra qualche lucido vedremo perche'.

SHA1(M) // $|M| < 2^{64}$

$V \leftarrow \text{SHF1}(5A827999||6ED9EBA1||8F1BBCDC||CA62C1D6, M)$

return V

SHF1(K, M) // $|K| = 128$ e $|M| < 2^{64}$

$y \leftarrow \text{shapad}(M)$

Sia $y = M_1||M_2||\dots||M_n$ ($|M_i| = 512$)

$V \leftarrow 67452301||\text{EFCDAB89}||\text{98BADC}F||\text{10325476}||\text{C3D2E1F0}$

for $i = 1, \dots, n$

$V \leftarrow \text{shf1}(K, M_i||V)$

return V

shapad(M) // $|M| < 2^{64}$

$d \leftarrow (447 - |M|) \bmod 512$

Sia ℓ la rappr. binaria (64 bit) di $|M|$

$y \leftarrow M||1||0^d||\ell$ // $|y|$ multiplo di 512

return y

Shapad

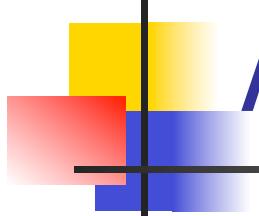
$$d = (447 - |M|) \bmod 512$$

$$d + |M| = 512 \cdot t + 447$$

$$|y| = |M| + 1 + d + l$$

$$|y| = 512 \cdot t + 447 + 1 + 64$$

$$|y| = 512 \cdot t + 512$$



Attacchi alle funzioni hash

- Ci concentriamo sulla proprieta' di resistenza alle collisioni (CR2).
- Questa e' la proprieta' piu' utile in pratica.
- In particolare mostreremo solo attacchi generici.
- Accenneremo soltanto ad attacchi piu' specifici.

Definizione (Funzioni Resistenti alle Collisioni)

- $H: K \times D \rightarrow R$ funzione hash

$Esp_H^{cr2-kk}(A)$

$k \leftarrow_R K; (x_1, x_2) \leftarrow_R A(k);$

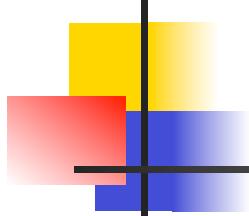
If $((H_k(x_1) = H_k(x_2)) \text{ and } (x_1 \neq x_2) \text{ and } (x_1, x_2 \in D))$

Return 1

else Return 0

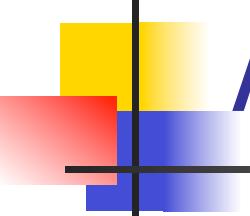
$$\text{Adv}^{cr2-kk}(A) = \Pr[Esp_H^{cr2-kk}(A) = 1]$$

Una funz. res. alle coll. e' anche una funz. universale unidirezionale.



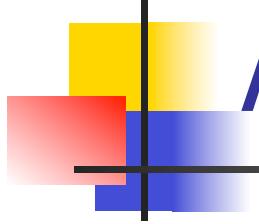
Forza Bruta

- Scegliamo un input a caso x_1 e calcoliamo $H(x_2)$
- Proviamo tutti i possibili input x_2 fino a quando $H(x_1) = H(x_2)$ ma $x_1 \neq x_2$
- Nel caso di SHA1 tale attacco potrebbe costare 2^{160} passi! SE SI HA LA PROBABILITÁ 1/IRI DI TROVARE UNA COLLISIONE ALTRIMENTI DOVREMMO FARE $q = IDI$ ITERAZIONI
- Possiamo far meglio?



Attacco del compleanno

- Idea: Prendiamo q input a caso e calcoliamo i corrispondenti valori della funzione.
 - Se troviamo 2 punti distinti che collidono, abbiamo vinto
 - Il paradosso del compleanno sembrerebbe garantire che per trovare una collisione bastano
- $$q \approx O\left(\sqrt{|R|}\right)$$
- tentativi.



Analisi

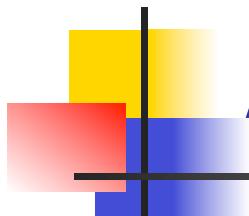
Sia $H: \{0,1\}^k \times D \rightarrow \{0,1\}^n$

$A(K)$

for $i=1, \dots, q$ **do**

$x_i \leftarrow_R D; y_i \leftarrow H_k(x_i);$

if $\exists i, j (i \neq j \wedge y_i = y_j \wedge x_i \neq x_j)$ **return** x_i, x_j
else return FAIL



A vs Attacco del Compleanno

Avversario A

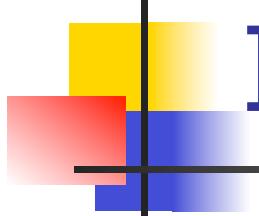
```
for i=1,...,q do
     $x_i \leftarrow_R D; y_i \leftarrow H_k(x_i);$ 
if  $\exists i,j (i \neq j \wedge y_i = y_j \wedge x_i \neq x_j)$ 
COLL  $\leftarrow$  True
```

$\Pr[\text{COLL}] = ?$

Attacco del Compleanno

```
for i=1,...,q do
     $y_i \leftarrow_R \{0,1\}^n;$ 
if  $\exists i,j (i \neq j \wedge y_i = y_j)$ 
COLL  $\leftarrow$  True
```

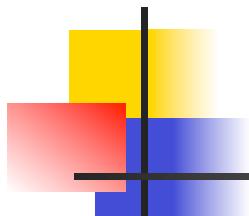
$\Pr[\text{COLL}] = q^2 / 2^{n+1}$



In particolare

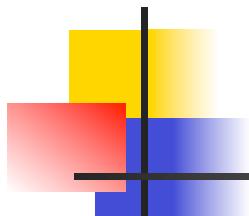
- L'analisi dell'attacco del compleanno, assume che ogni x abbia la stessa probabilità di "produrre" ogni $H_k(y)$
- Questo puo' non essere vero nel nostro caso.
- Sia $P(y)$ la probabilità che $H_k(x)=y$

$$P(y) = \frac{|H_k^{-1}(y)|}{|D|}$$



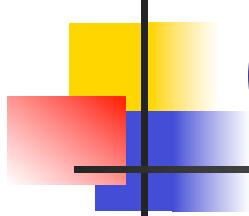
Funzioni Regolari

- Una funzione con la proprietà che
$$\forall y_1, y_2 \ |H_k^{-1}(y_1)| = |H_k^{-1}(y_2)|$$
e' detta regolare.
- Per tali funzioni hash possiamo applicare l'analisi dell'attacco del compleanno.
- Cosa possiamo dire per le funzioni non regolari?



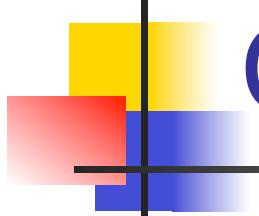
Funzioni non regolari

- Se H non è regolare, l'attacco può solo diventare più efficace.
 - Troviamo collisioni prima
- Per questa ragione una buona funzione hash crittografica dovrebbe essere quanto più regolare possibile.



Conseguenze dell'attacco (caso di SHA1)

- SHA1 utilizza come insieme di arrivo $\{0,1\}^{160}$.
- Assumendo che l'attacco del compleanno sia il migliore possibile, bisognerebbe calcolare SHA1 2^{80} volte per trovare una collisione.
- Tale valore, renderebbe l'operazione non fattibile in pratica.
- Quanto e' ragionevole tale ipotesi?



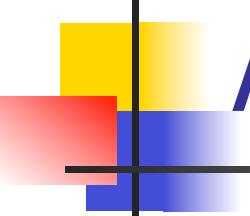
Cosa succede in pratica

- Abbiamo soltanto accennato alla funzione SHA1.
- Non descriveremo ne' MD4 e nemmeno MD5.
- Esistono ragioni piuttosto forti alla base di tali scelte.

When	Against	Time	Who
1993,1996	md5	2^{16}	[dBBo,Do]
2005	RIPEMD	2^{18}	
2004	SHA0	2^{51}	[JoCaLeJa]
2005	SHA0	2^{40}	[WaFeLaYu]
2005	SHA1	$2^{69}, 2^{63}$	[WaYiYu,WaYaYa]
2009	SHA1	2^{52}	[MHP]
2005,2006	MD5	1 minute	[WaFeLaYu,LeWadW,KI]

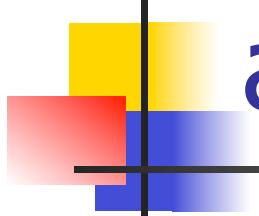
md5 is the compression function of MD5

SHA0 is an earlier, weaker version of SHA1



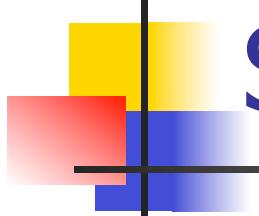
Attacchi a MD5

- Attualmente possiamo effettuare i seguenti attacchi
 - Trovare due msg apparentemente casuali che differiscono in 3 bit (**poco interessante**)
 - Trovare due documenti PDF che “collidono” (**più interessante**)
 - Trovare due eseguibili Win32 che collidono (**molto interessante**)
 - Rompere protocolli crittografici in uso (**molto interessante**)



Come funzionano questi attacchi?

- Trovare collisioni per 2 msg che differiscono in soli 3 bit
 - Pochi minuti su un portatile!
- Trovare due eseguibili Win32 che collidono
 - <http://www.win.tue.nl/hashclash/SoftIntCodeSign/>
 - 2 giorni su una Playstation 3 (2009)



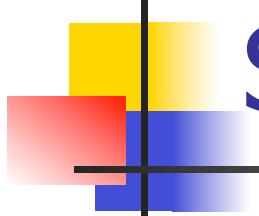
SHA3

- National Institute for Standards and Technology (NIST) ha da poco concluso la competizione per determinare il nuovo standard

<http://csrc.nist.gov/groups/ST/hash/index.html>

SHA3 - Caratteristiche richieste

- **Design**
 - Family of functions with 224, 256, 384, 512 bit output sizes
- **Compatibilità**
 - existing cryptographic standards
- **Sicurezza**
 - CR, one-wayness, near-collision resistance,...
- **Efficienza:**
 - as fast as (or faster than) SHA-256



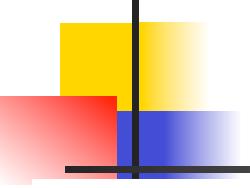
SHA3 Competition

- Submissions: 64
- Round 1: 51 Round 2: 14
- The round 2 functions: BLAKE, Blue Midnight Wish, CubeHash, ECHO, Fugue, Grostl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, Skein.
- Winner: Keccak

http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo

La trasformazione Merkle-Damgard

- L'idea base di SHF1 (subroutine di SHA1) e' di iterare l'applicazione della funzione di compressione shf1
- Il metodo e' scelto in modo accurato: se shf1 e' resistente alle collisioni, lo e' anche SHF1.
- Il paradigma mostra come trasformare una funzione di compressione in una funzione hash, in modo che se la prima e' CR, lo e' pure la seconda.
- Adesso guarderemo questo paradigma.



$h : \mathcal{K} \times \{0, 1\}^{b+v} \rightarrow \{0, 1\}^v$ funz. di compr.
resistente alle collisioni

- b lunghezza del blocco, v parametro di concatenazione
- B insieme di stringhe w ($|w|$ multiplo di b), D sottoinsieme di $\{0, 1\}^{< 2^b}$

Definizione

$\text{pad} : D \rightarrow B$ e' MD compatibile se $\forall M, M_1, M_2 \in D$

- (1) M e' prefisso di $\text{pad}(M)$
 - (2) Se $|M_1| = |M_2|$, $|\text{pad}(M_1)| = |\text{pad}(M_2)|$
 - (3) Se $|M_1| \neq |M_2|$, UB di $\text{pad}(M_1) \neq \text{UB}$ di $\text{pad}(M_2)$
- UB: Ultimo blocco

SHA1(M) // $|M| < 2^{64}$

$V \leftarrow \text{SHF1}(5A827999||6ED9EBA1||8F1BBCDC||CA62C1D6, M)$

return V

SHF1(K, M) // $|K| = 128$ e $|M| < 2^{64}$

$y \leftarrow \text{shapad}(M)$

Sia $y = M_1||M_2||\dots||M_n$ ($|M_i| = 512$)

$V \leftarrow 67452301||\text{EFCDAB89}||10325476||\text{C3D2E1F0}$

for $i = 1, \dots, n$

$V \leftarrow \text{shf1}(K, M_i||V)$

return V

shapad(M) // $|M| < 2^{64}$

$d \leftarrow (447 - |M|) \bmod 512$

Sia ℓ la rappr. binaria (64 bit) di $|M|$

$y \leftarrow M||1||0^d||\ell$ // $|y|$ multiplo di 512

return y

Funzioni hash da funzioni di compressione

$\mathsf{H}(K, M)$

$y \leftarrow \text{pad}(M)$

Sia $y = M_1 || M_2 || \dots || M_n$
 // ($|M_i| = b$)

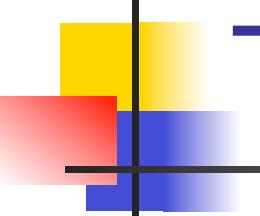
$V \leftarrow \text{IV}$

 // IV vettore iniziale di v -bit

for $i = 1, \dots, n$

$V \leftarrow h(K, M_i || V)$

Return V



Teorema

- $h: K \times \{0,1\}^{b+v} \rightarrow \{0,1\}^v$ fam. di funz. DI COMPRESSIONE
- $H: K \times D \rightarrow \{0,1\}^v$ costruita come descritto.
- Sia A_H avversario che trova collisioni in H
- Possiamo costruire A_h che trova collisioni per h e

$$\text{Adv}_H^{\text{cr2-kk}}(A_H) \leq \text{Adv}_h^{\text{cr2-kk}}(A_h)$$

- Il teorema ci dice che se h e' resistente alle collisioni, altrettanto deve valere per H