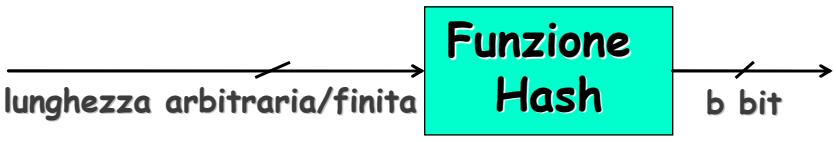
Funzioni hash

Barbara Masucci

Dipartimento di Informatica ed Applicazioni
Università di Salerno

masucci@dia.unisa.it
http://www.dia.unisa.it/professori/masucci

Funzioni Hash



Idea alla base:

il valore hash h(M) è una rappresentazione non ambigua e non falsificabile del messaggio M

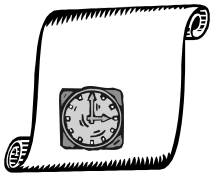
Proprietà: comprime ed è facile da computare

Uso delle funzioni hash

Firme digitali

Integrita' dei dati

Certificazione del tempo



Firme digitali e Funzioni hash

Problema: firma digitale di messaggi lunghi

Soluzione naive: Divisione in blocchi e firma per ogni blocco

problema per la sicurezza: una permutazione/composizione delle firme è una nuova firma

Soluzione di uso corrente:

firmare il valore hash del messaggio

[firma di M] = $F_k(h(M))$

Vantaggi: integrità dei dati ed efficienza degli algoritmi

Integrità dei dati e Funzioni hash

Tipico uso delle funzioni hash

Computo al tempo T il valore hash del file M

Conservo H = h(M) in un luogo sicuro

Per controllare se il file è stato successivamente modificato, calcolo h(M´) e verifico se H = h(M´)

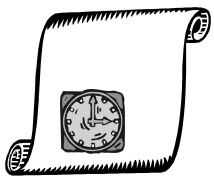
h(M) è l'impronta digitale del file

Assicura se un file è stato modificato!

Certificazione del tempo e Funzioni Hash

Il notaio digitale

Quando è stato creato il documento **D** ?



Funzioni Hash: Proprietà

Facili da calcolare

… poi?

Un possibile attacco

prepara 2 versioni di un contratto M ed M´
- M è favorevole ad Alice
- M´ è sfavorevole ad Alice

modifica M´ a caso (piccoli cambiamenti come aggiunta spazi) finchè h(M) = h(M´)

Alice firma M ⟹ $Firma_{kpriv}(h(M))$

ha quindi la firma di M´ ⟹ $Firma_{kpriv}(h(M'))$

Esempio di lettera fraudolenta

Cara Alice,

ti {scrivo / sto scrivendo} da {un bellissimo / uno splendido} posto {della costiera Amalfitana / vicino Amalfi}

……..

{Colui / La persona} che {ti porterà / è portatore di} questa {lettera / missiva} è di fiducia!

……..

Funzioni hash: sicurezza

- **Sicurezza debole**: dato M è computazionalmente difficile trovare un altro M´ tale che h(M) = h(M´)
- **Sicurezza forte**: computazionalmente difficile trovare 2 diversi messaggi con lo stesso valore hash
- **One-way**: dato y è computazionalmente difficile trovare M tale che y = h(M)

Sicurezza forte ⇒ One-way

- h: X→Z funzione hash, $|X| \geq 2 \cdot |Z|$
  - $\log|X| \geq \log(2 \cdot |Z|) = \log 2 + \log|Z| = 1 + \log|Z|$
- Supponiamo che **ALG** sia un algoritmo di inversione per h
- … allora esiste un algoritmo *Las Vegas* che trova collisioni con probabilità $\geq 1/2$

Sicurezza forte ⇒ One-way

- h: X→Z funzione hash, $|X| \geq 2 \cdot |Z|$
- Supponiamo che **ALG** sia un algoritmo di inversione per h
- … allora esiste un algoritmo *Las Vegas* che trova collisioni con probabilità $\geq 1/2$

1. Scegli a caso x in X
2. $z \leftarrow h(x)$
3. $x' \leftarrow$ **ALG**(z)
4. **If** $x' \neq x$ **then** (x', x) è una collisione
   else fallito

Sicurezza forte ⇒ One-way

Definiamo una relazione di equivalenza ~ su X
- $x \sim x' \iff h(x) = h(x')$
- $[x]=\{ x' \in X: x \sim x' \}$ classe di equivalenza
- $C=\{ [x]: x \in X \}$ insieme delle classi di equivalenza
- $|C| \leq |Z|$ ($|C|=|Z|$ solo se tutti i valori in Z sono invertibili)

Sia $x \in X$ l'elemento scelto nel passo 1 dell'algoritmo

Valutiamo la probabilità di successo (collisione) per x
- Casi possibili: $|[x]|$ valori in $[x]$
- Casi favorevoli: $|[x]|-1$ valori in $[x]$ differenti da x

$$\text{Prob(successo}|x) = \frac{|[x]|-1|}{|[x]|}$$

Sicurezza forte ⇒ One-way

$$\text{Prob(successo)} = \sum_{x \in X} \text{Prob(successo}|x) \cdot \text{Prob(scelgo x)}$$

$$= \frac{1}{|X|} \sum_{x \in X} \frac{|[x]|-1}{|[x]|}$$

$[x]=\{x' \in X: x \sim x'\}$

$$= \frac{1}{|X|} \sum_{c \in C} \sum_{x \in c} \frac{|c|-1}{|c|}$$

$C=\{[x]: x \in X\}$

$$= \frac{1}{|X|} \sum_{c \in C} (|c|-1) = \frac{1}{|X|} \left(\sum_{c \in C} |c| - \sum_{c \in C} 1 \right)$$

$$\geq \frac{|X|-|Z|}{|X|} \geq \frac{|X|-|X|/2}{|X|} = \frac{1}{2}$$

Lunghezza valore hash

- Quanto grande l'hash per la sicurezza forte?
- Se $|Z| = 2^3$ bit…
  - Quanti valori scegliere per essere certi di trovare almeno una collisione?
- $|Z|+1$ diversi valori di M

  $\Rightarrow$ certezza di trovare almeno una collisione

Esempi { 2^{40}  2^{80}  2^{160}

Lunghezza valore hash

- Nuovo attacco per trovare collisioni
  - h: $X \rightarrow Z$ funzione hash, $|X| = m$ e $|Z| = n$
  - Scelgo a caso diversi messaggi
  - Verifico se ottengo almeno due valori hash uguali
- Quanti messaggi per avere una buona probabilità di successo?
  - n numero dei diversi valori hash
  - t numero messaggi da scegliere
  - ε probabilità di successo

Paradosso del compleanno

- Quante persone scegliere a caso affinché, con probabilità ≥ 0.5, ci siano almeno due con lo stesso compleanno?

Risposta: bastano 23 persone!

Paradosso del compleanno

- Scegliamo a caso elementi in un insieme di cardinalità n.
- Quanti elementi scegliere se si vuole che la probabilità che ci siano almeno due elementi uguali sia ε?

$$t \approx \sqrt{n \cdot 2\ln\left(\frac{1}{1-\varepsilon}\right)}$$

Paradosso del compleanno

- Scegliamo a caso 2 elementi z_1, z_2 in un insieme di cardinalità n.
- Probabilità che siano diversi

$$\text{Prob}(z_2 \neq z_1) = 1 - \text{Prob}(z_2 = z_1)$$
$$= 1 - 1/n$$

Paradosso del compleanno

Scegliamo a caso 2 elementi z_1, z_2 in un insieme di cardinalità n.
Probabilità che siano diversi

$$\text{Prob}(z_2 \neq z_1) = 1 - 1/n$$

Scegliamo a caso 3 elementi z_1, z_2, z_3 in un insieme di cardinalità n.
Probabilità che siano diversi

$$\text{Prob}(z_3 \neq z_1 \wedge z_3 \neq z_2 \mid z_2 \neq z_1) \cdot \text{Prob}(z_2 \neq z_1) =$$
$$= [1 - \text{Prob}(z_3 = z_1 \vee z_3 = z_2 \mid z_2 \neq z_1)] \cdot \text{Prob}(z_2 \neq z_1)$$
$$= (1 - 2/n)(1 - 1/n)$$

Paradosso del compleanno

Scegliamo a caso $z_1, z_2, \ldots, z_t$ in un insieme di cardinalità n.
Probabilità che siano diversi
Prob($z_1, z_2, \ldots, z_t$ diversi) = $(1 - (t-1)/n) \cdot \ldots \cdot (1-2/n) \cdot (1 - 1/n)$

Per piccoli x abbiamo $1-x \approx e^{-x}$

$$e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \ldots$$

Paradosso del compleanno

Scegliamo a caso $z_1, z_2, \ldots, z_t$ in un insieme di cardinalità n.
Probabilità che sono diversi

$$\text{Prob(} z_1, z_2, \ldots, z_t \text{ diversi)} = \prod_{i=1}^{t-1} \left(1 - \frac{i}{n}\right)$$

$$\approx \prod_{i=1}^{t-1} \left(e^{-i/n}\right)$$

Per piccoli x abbiamo $1-x \approx e^{-x}$

$$e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \ldots$$

Paradosso del compleanno

Scegliamo a caso $z_1, z_2, \ldots, z_t$ in un insieme di cardinalità n.
Probabilità che sono diversi

$$\text{Prob}(\ z_1, z_2, \ldots, z_t \text{ diversi }) = \prod_{i=1}^{t-1}\left(1-\frac{i}{n}\right)$$

$$\approx \prod_{i=1}^{t-1}\left(e^{-i/n}\right)$$

$$\approx e^{-t(t-1)/(2n)}$$

$\varepsilon \triangleq$ Prob(almeno una collisione tra $z_1, z_2, \ldots, z_t$)

$\approx 1 - e^{-t(t-1)/n}$

Paradosso del compleanno

$\varepsilon \triangleq$ Prob(almeno una collisione tra $z_1, z_2, \ldots, z_t$)

$$1 - \varepsilon \approx e^{-t(t-1)/(2n)}$$

$$\frac{-t(t-1)}{2n} \approx \ln(1-\varepsilon)$$

$$t^2 - t \approx 2n \cdot \ln\frac{1}{1-\varepsilon}$$

$$t \approx \sqrt{n \cdot 2\ln\frac{1}{1-\varepsilon}}$$

Paradosso del compleanno

- Scegliamo a caso elementi in un insieme di cardinalità n.
- Quanti elementi scegliere se si vuole che la probabilità che ci siano almeno due elementi uguali sia ε?

$$t \approx \sqrt{n \cdot 2\ln\left(\frac{1}{1-\varepsilon}\right)}$$

- Se $\varepsilon = 0.5$ allora $t \approx 1.17\sqrt{n}$
- Applicazione: $n = 365$ e $\varepsilon = 0.5$ allora $t = 22.3$

Che relazione c'è con le funzioni hash?

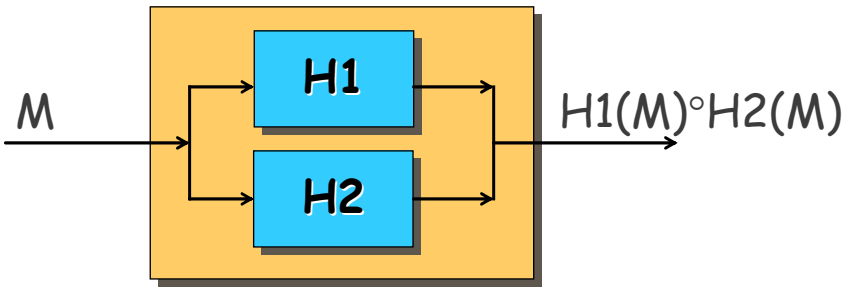
Attacco del compleanno

- Scegliere t elementi a caso e calcolarne i valori hash.
- Quanti elementi scegliere per avere almeno una collisione?
  - Assumiamo che tutte le classi di equivalenza abbiano più o meno la stessa cardinalità, caso migliore per chi sceglie h
- Per una fissata probabilità ε, t è circa $\sqrt{n}$
- Se $n = 2^{80}$ allora $t \approx 2^{40}$ 
- Se $n = 2^{160}$ allora $t \approx 2^{80}$ 

Sicurezza Hash 128 bit

- Costo di un attacco per computare collisioni $2^{64} \approx 2 \cdot 10^{19}$ valutazioni della funzione
- Attacco <1mese e $10.000.000
  - P. van Orschot e M. Wiener [1994]
- Si ipotizza che il costo dimezzi ogni 18 mesi

Composizione funzioni hash



Trovare una collisione per H(M)=H1(M)◦H2(M) significa trovare una collisione sia per H1 che per H2

Modello generale per funzioni hash iterate

Input taglia arbitraria ⇒ taglia fissata

Input M. Padding ed aggiunta della lunghezza di M.

Si ottiene un messaggio con blocchi di taglia uguale $X_1X_2…X_n$

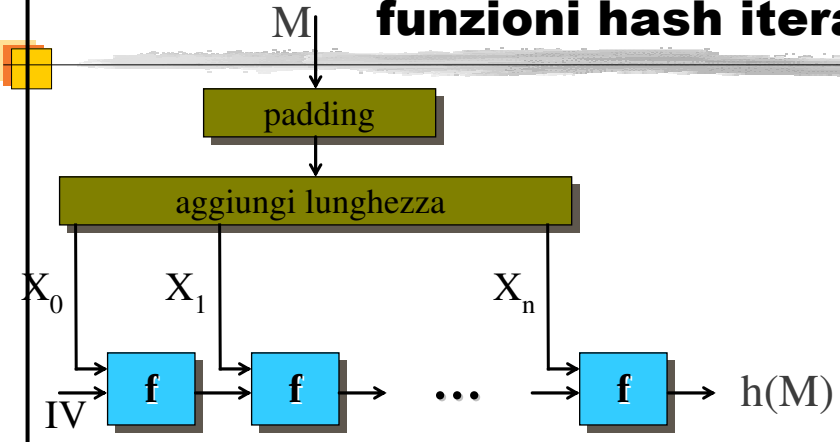
H_0 è una costante iniziale

Computazione di … $H_i = f(X_i,H_{i-1})$ …

Valore hash $H_n = f(X_n,H_{n-1})$

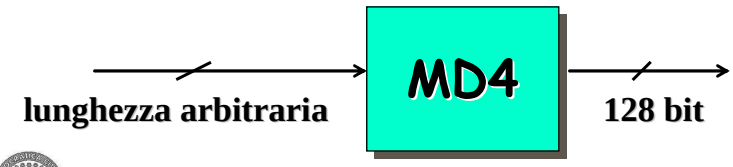
computazione del valore hash

Modello generale funzioni hash iterate



Funzione hash MD4

- Progettata nel 1990 da Ron Rivest
- MD da **M**essage **D**igest
- Operazioni efficienti su architetture 32 bit little-endian
  - $a_1a_2a_3a_4$ rappresenta l'intero $a_4 2^{24}+a_3 2^{16}+a_2 2^8+a_1$



Obiettivi di progettazione per MD4

- **Sicurezza forte**: computazionalmente difficile trovare 2 messaggi con lo stesso valore hash
- **Sicurezza diretta**: sicurezza non basata su problemi teorici difficili computazionalmente
- **Velocità**: algoritmo adatto per implementazioni software molto veloci
- **Semplicità** e **Compattezza**: semplice da descrivere e da implementare, nessun uso di tabelle e di complesse strutture dati

MD4: padding del messaggio

- MD4 processa il messaggio in blocchi di 512 bit
  - Ogni blocco consta di 16 parole di 32 bit
- M messaggio originario di b bit ⇒ padding

$M' = M\ 100....0\ b$

(447-b) mod 512 bit 64 bit

- M' consta di un numero di bit multiplo di 512, ovvero di un numero di parole N multiplo di 16
  - N/16 blocchi di 512 bit

MD4: operazioni

Funzioni definite su parole di 32 bit:

round 1: $f(X,Y,Z) = (X \wedge Y) \vee ((\neg X) \wedge Z)$ (if X then Y else Z)
round 2: $g(X,Y,Z) = (X \wedge Z) \vee (Y \wedge Z) \vee (X \wedge Y)$ (2 su 3)
round 3: $h(X,Y,Z) = X \oplus Y \oplus Z$ (bit di parità)

X	Y	Z	f	g	h
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	1	1	0
1	1	1	1	1	1

Ogni round consiste di 16 operazioni

- Usa 4 parole di 32 bit: A, B, C, D

$X+Y$ somma modulo 2^{32}, $X \ll s$ shift ciclico a sinistra di s bit

MD4

A ← 67452301; B ← efcdab89; C ← 98badcfe; D ← 10325476;
for i=0 **to** N/16-1 **do**
  for j=0 **to** 15 **do** X[j] ← M´[16i+j]
  AA ← A; BB ← B; CC ← C; DD ← D;

round 1	round 2	round 3
A ←(A+f(B,C,D)+X[0])«3	A ←(A+g(B,C,D)+X[0] +p)«3	A ←(A+h(B,C,D)+X[0] +q)«3
D ←(D+f(A,B,C)+X[1])«7	D ←(D+g(A,B,C)+X[4] +p)«5	D ←(D+h(A,B,C)+X[8] +q)«9
C ←(C+f(D,A,B)+X[2])«11	C ←(C+g(D,A,B)+X[8] +p)«9	C ←(C+h(D,A,B)+X[4] +q)«11
B ←(B+f(C,D,A)+X[3])«19	B ←(B+g(C,D,A)+X[12]+p)«13	B ←(B+h(C,D,A)+X[12]+q)«15
A ←(A+f(B,C,D)+X[4])«3	A ←(A+g(B,C,D)+X[1] +p)«3	A ←(A+h(B,C,D)+X[2] +q)«3
D ←(D+f(A,B,C)+X[5])«7	D ←(D+g(A,B,C)+X[5] +p)«5	D ←(D+h(A,B,C)+X[10]+q)«9
C ←(C+f(D,A,B)+X[6])«11	C ←(C+g(D,A,B)+X[9] +p)«9	C ←(C+h(D,A,B)+X[6] +q)«11
B ←(B+f(C,D,A)+X[7])«19	B ←(B+g(C,D,A)+X[13]+p)«13	B ←(B+h(C,D,A)+X[14]+q)«15
A ←(A+f(B,C,D)+X[8])«3	A ←(A+g(B,C,D)+X[2] +p)«3	A ←(A+h(B,C,D)+X[1] +q)«3
D ←(D+f(A,B,C)+X[9])«7	D ←(D+g(A,B,C)+X[6] +p)«5	D ←(D+h(A,B,C)+X[9] +q)«9
C ←(C+f(D,A,B)+X[10])«11	C ←(C+g(D,A,B)+X[10]+p)«9	C ←(C+h(D,A,B)+X[5] +q)«11
B ←(B+f(C,D,A)+X[11])«19	B ←(B+g(C,D,A)+X[14]+p)«13	B ←(B+h(C,D,A)+X[13]+q)«15
A ←(A+f(B,C,D)+X[12])«3	A ←(A+g(B,C,D)+X[3] +p)«3	A ←(A+h(B,C,D)+X[3] +q)«3
D ←(D+f(A,B,C)+X[13])«7	D ←(D+g(A,B,C)+X[7] +p)«5	D ←(D+h(A,B,C)+X[11]+q)«9
C ←(C+f(D,A,B)+X[14])«11	C ←(C+g(D,A,B)+X[11]+p)«9	C ←(C+h(D,A,B)+X[7] +q)«11
B ←(B+f(C,D,A)+X[15])«19	B ←(B+g(C,D,A)+X[15]+p)«13	B ←(B+h(C,D,A)+X[15]+q)«15

A ← A+AA; B ← B+BB; C ← C+CC; D ← D+DD;
output: (A, B, C, D)

costanti
p=5a827999
q=6ed9eba1

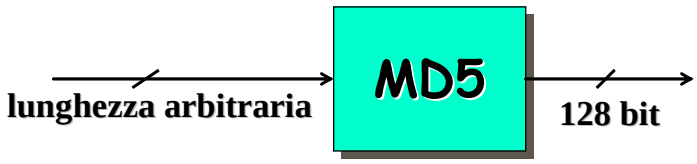
Sicurezza di MD4

MD4 è stato oggetto di molti attacchi

- crittoanalisi dei primi 2 round: Merkle ha provato che è facile trovare collisioni con round 3 omesso
- crittoanalisi degli ultimi 2 round: den Boer e Bosselaers [Crypto '91] hanno trovato collisioni con round 1 omesso
- Settembre 1995: Dobbertin [FSE ´96] ha trovato collisioni per MD4 con un PC in pochi secondi

MD5

- Progettato nel 1991 da Ron Rivest
- MD da **M**essage **D**igest
- Operazioni efficienti su architetture 32 bit little-endian



MD5	MD4
4 round con 4 ·16 operazioni	3 round con 3 ·16 operazioni
4 funzioni logiche	3 funzioni logiche
64 costanti additive	2 costanti additive
ogni passo aggiunge il risultato del passo precedente	non accade

MD5: padding del messaggio

- MD5 processa il messaggio in blocchi di 512 bit
  - Ogni blocco consta di 16 parole di 32 bit
- M messaggio originario di b bit ⇒ padding

$$M' = \boxed{M \; 100\ldots.0 \; b}$$

(447-b) mod 512 bit 64 bit

- M' consta di un numero di bit multiplo di 512, ovvero di un numero di parole N multiplo di 16
  - N/16 blocchi di 512 bit

MD5: operazioni

Funzioni definite su parole di 32 bit:

round 1: $F(X,Y,Z) = (X \wedge Y) \vee ((\neg X) \wedge Z)$ (if X then Y else Z)
round 2: $G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge (\neg Z))$ (if Z then X else Y)
round 3: $H(X,Y,Z) = X \oplus Y \oplus Z$ (bit di parità)
round 4: $I(X,Y,Z) = Y \oplus (X \vee (\neg Z))$ (nuova funzione)

X	Y	Z	F	G	H	I
0	0	0	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	0	1	1
1	0	1	0	1	0	1
1	1	0	1	1	0	0
1	1	1	1	1	1	0

Ogni round consiste di 16 operazioni [ABCD.k.s.i]

$A \leftarrow B + ((A + W(B,C,D) + X[k] + T[i])) \lll s$

- K è l'indice della parola, s indica lo shift ciclico, i è l'indice dell'iterazione, W è la funzione del round

Costanti T[1...64]

T[i] ← primi 32 bit di |sin(i)| = ⌊2^{32} · |sin(i)|⌋ (i in radianti)

1	d76aa478	17	f61e2562	33	fffa3942	49	f4292244
2	e8c7b756	18	c040b340	34	8771f681	50	432aff97
3	242070db	19	265e5a51	35	6d9d6122	51	ab9423a7
4	c1bdceee	20	e9b6c7aa	36	fde5380c	52	fc93a039
5	f57c0faf	21	d62f105d	37	a4beea44	53	655b59c3
6	4787c62a	22	02441453	38	4bdecfa9	54	8f0ccc92
7	a8304613	23	d8a1e681	39	f6bb4b60	55	ffeff47d
8	fd469501	24	e7d3fbc8	40	bebfbc70	56	85845dd1
9	698098d8	25	21e1cde6	41	289b7ec6	57	6fa87e4f
10	8b44f7af	26	c33707d6	42	eaa127fa	58	fe2ce6e0
11	ffff5bb1	27	f4d50d87	43	d4ef3085	59	a3014314
12	895cd7be	28	455a14ed	44	04881d05	60	4e0811a1
13	6b901122	29	a9e3e905	45	d9d4d039	61	f7537e82
14	fd987193	30	fcefa3f8	46	e6db99e5	62	bd3af235
15	a679438e	31	676f02d9	47	1fa27cf8	63	2ad7d2bb
16	49b40821	32	8d2a4c8a	48	c4ac5665	64	eb86d391

MD5

A ← 0123456; B ← 89abcdef; C ← fdecba98; D ← 76543210;
for i = 0 **to** N/16-1 **do**
  for j = 0 **to** 15 **do** X[j] ← M´[16i+j]
  AA ← A; BB ← B; CC ← C; DD ← D;

round 1
[ABCD. 0.7. 1] [DABC. 1.12. 2] [CDAB. 2.17. 3] [BCDA. 3.22. 4]
[ABCD. 4.7. 5] [DABC. 5.12. 6] [CDAB. 6.17. 7] [BCDA. 7.22. 8]
[ABCD. 8.7. 9] [DABC. 9.12.10] [CDAB.10.17.11] [BCDA.11.22.12]
[ABCD.12.7.13] [DABC.13.12.14] [CDAB.14.17.15] [BCDA.15.22.16]

round 2
[ABCD. 1.5.17] [DABC. 6. 9.18] [CDAB.11.14.19] [BCDA. 0.20.20]
[ABCD. 5.5.22] [DABC.10. 9.22] [CDAB.15.14.23] [BCDA. 4.20.24]
[ABCD. 9.5.25] [DABC.14. 9.26] [CDAB. 3.14.27] [BCDA. 8.20.28]
[ABCD.13.5.29] [DABC. 2. 9.30] [CDAB. 7.14.21] [BCDA.12.20.32]

round 3
[ABCD. 5.4.33] [DABC. 8.11.34] [CDAB.11.16.35] [BCDA.14.23.36]
[ABCD. 1.4.37] [DABC. 4.11.38] [CDAB. 7.16.39] [BCDA.10.23.40]
[ABCD.13.4.41] [DABC. 0.11.42] [CDAB. 3.16.43] [BCDA. 6.23.44]
[ABCD. 9.4.45] [DABC.12.11.46] [CDAB.15.16.45] [BCDA. 2.23.48]

round 4
[ABCD. 0.6.49] [DABC. 7.10.50] [CDAB. 5.15.51] [BCDA. 5.21. 5]
[ABCD.12.6.53] [DABC. 3.10.54] [CDAB. 1.15.55] [BCDA. 1.21.56]
[ABCD. 8.6.57] [DABC.15.10.58] [CDAB.13.15.59] [BCDA.13.21.60]
[ABCD. 4.6.61] [DABC.11.10.62] [CDAB. 9.15.63] [BCDA. 9.21.64]

  A←A+AA; B ← B+BB; C ← C+CC; D ← D+DD;
output: (A, B, C, D)

Avalanche effect nell'MD5

In MD5 ogni passo somma il valore precedente

Round 1 in MD5

$A \leftarrow B + ((A + F(B,C,D) + X[0] + T[1])) \lll 7$
$D \leftarrow A + ((D + F(A,B,C) + X[1] + T[2])) \lll 12$
$C \leftarrow D + ((C + F(D,A,B) + X[2] + T[3])) \lll 17$
$B \leftarrow C + ((B + F(C,D,A) + X[3] + T[4])) \lll 22$
...

Round 1 in MD4:

$A \leftarrow (A + f(B,C,D) + X[0]) \lll 3$
$D \leftarrow (D + f(A,B,C) + X[1]) \lll 7$
$C \leftarrow (C + f(D,A,B) + X[2]) \lll 11$
$B \leftarrow (B + f(C,D,A) + X[3]) \lll 19$
...

Little-endian e Big-endian

Come si trasformano sequenze di byte in parole di 32 bit?

Conversione ambigua!

Sequenza byte B1, B2, B3, B4 nella parola W

Architetture **Little-endian** (come processori Intel 80xxx)

byte con indirizzo più basso è quello meno significativo

valore parola $W=2^{24}B4+ 2^{16}B3+ 2^{8}B2+ 2^{0}B1$

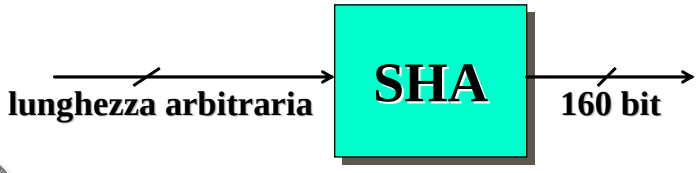
Architetture **Big-endian** (come SUN SPARCstation)

byte con indirizzo più basso è quello più significativo

valore parola $W=2^{24}B1+ 2^{16}B2+ 2^{8}B3+ 2^{0}B4$

SHS

- SHS per **S**ecure **H**ash **S**tandard
- SHA per **S**ecure **H**ash **A**lgorithm
- Standard del Governo americano dal 1993
- Modificato nel luglio 1994, denotato SHA-1
  - (unica differenza: aggiunta di uno shift nell'espansione dei blocchi)
- Operazioni efficienti su architetture 32 bit big-endian
- Stessi principi di MD4 ed MD5, ma più sicuro



SHA: padding del messaggio

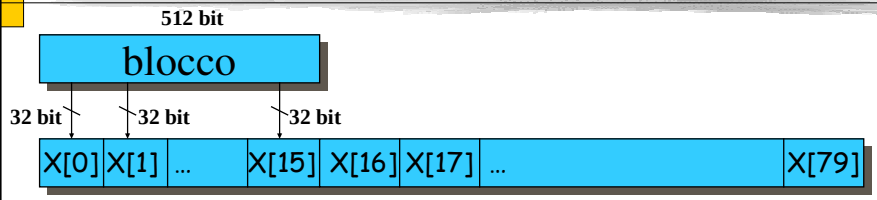
- SHA processa il messaggio in blocchi di 512 bit
  - Ogni blocco consta di 16 parole di 32 bit
- M messaggio originario di b bit ⇒ padding

$$M' = M\ 100....0\ b$$

(447-b) mod 512 bit 64 bit

- M' consta di un numero di bit multiplo di 512, ovvero di un numero di parole N multiplo di 16
  - N/16 blocchi di 512 bit

Espansione blocco ed Iterazioni



$X[t] \leftarrow (X[t-3] \oplus X[t-8] \oplus X[t-14] \oplus X[t-16]) \lll 1$

- 80 iterazioni
- Una parola ed una costante per ogni iterazione

Funzioni logiche di SHA

Funzione F(t,X,Y,Z)

round t= 0,...,19: $F(t,X,Y,Z) = (X \wedge Y) \vee ((\neg X) \wedge Z)$ (if X then Y else Z)
round t=20,...,39: $F(t,X,Y,Z) = X \oplus Y \oplus Z$ (bit di parità)
round t=40,...,59: $F(t,X,Y,Z) = (X \wedge Z) \vee (Y \wedge Z) \vee (X \wedge Y)$ (2 su 3)
round t=60,...,79: $F(t,X,Y,Z) = Y \oplus X \oplus Z$ (bit di parità)

X	Y	Z	F(0,..)	F(20,..)	F(40,..)	F(60,..)
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	0	1	0	1
0	1	1	1	0	1	0
1	0	0	0	1	0	1
1	0	1	0	0	1	0
1	1	0	1	0	1	0
1	1	1	1	1	1	1

Costanti additive di SHA

- Costante additiva K[t]:
  - round t = 0,…,19: 5a827999
  - round t = 20,…,39: 6ed9eba1
  - round t = 40,…,59: 8f1bbcdc
  - round t = 60,…,79: ca62c1d1

A=67452310; B=efcdab89; C=98badcfe; D=10325476; E=c3d2e1f0;
for i = 0 **to** N/16-1 **do**
  for j = 0 **to** 15 **do**
    X[j] ← M´[16i+j]
  for t = 16 **to** 79 **do**
    X[t] ← (X[t-3] ⊕ X[t-8] ⊕ X[t-14] ⊕ X[t-16]) « 1
  AA ← A; BB ← B; CC ← C; DD ← D; EE ← E;
  for t=0 **to** 79 **do**
    TEMP ← (A«5) + F(t,B,C,D) + E + X[t] + K[t]
    E ← D
    D ← C
    C ← (B«30)
    B ← A
    A ← TEMP
  A ← A + AA; B ← B + BB; C ← C + CC; D ← D + DD; E ← E + EE;
output: (A, B, C, D, E)

SHA-1

espansione
da 16 ad 80 parole
"«1" non c'era in SHA

SHA-256, SHA-512, SHA-384

- Hash di SHA-1 è 160 bit
  - Sicurezza contro attacco del compleanno 80 bit
- Lunghezza chiavi AES: 128, 192, 256
- Proposti nuovi SHA (12 ottobre 2000)
  - Lunghezza valore hash: 256, 512, 384 bit
  - Sicurezza attacco del compleanno 128, 256, 192 bit
- Draft di Federal Information Processing Standard (FIPS), gennaio 2001

SHA-256, SHA-512, SHA-384

- Stessi principi di MD4, MD5, SHA-1
- SHA-256
  - Messaggio diviso in blocchi di 512 bit
  - Parole da 32 bit
- SHA-512
  - Messaggio diviso in blocchi di 1024 bit
  - Parole da 64 bit
- SHA-384
  - Valore hash = primi 384 bit di SHA-512, con costanti iniziali cambiate

Altre funzioni Hash

- **Snefru**, Ralph Merkle [1990], 128 oppure 256 bit
- **N-hash**, Nippon Telephone and Telegraph [1990], 128 bit
- **HAVAL**, Zheng-Pieprzyk-Seberry [1992] 128-160-192-224-256 bit
- **FFT-hash I**, C. Schnorr [1991], rotto dopo pochi mesi
- **FFT-hash II**, C. Schnorr [1992], rotto dopo poche settimane
- ...

Funzioni Hash basate su cifrari a blocchi

Se è disponibile una implementazione di un cifrario a blocchi ...
Cifrario a blocchi $E_K(\cdot)$ per input ad n bit
Funzione g che da n bit produce una chiave
$M'_1 \ldots M'_t$ è il messaggio M con eventuale padding
H_0 è una costante predefinita, H_t è il valore hash

$$H_i = E_{g(H_{i-1})}(M'_i) \oplus M'_i$$ [Matyas-Meyer-Oseas]

$$H_i = E_{g(H_{i-1})}(M'_i) \oplus M'_i \oplus H_{i-1}$$ [Miyaguchi-Preneel]

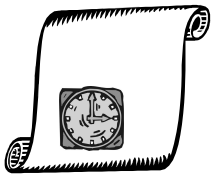
$$H_i = E_{M'_i}(H_{i-1}) \oplus H_{i-1}$$ [Davies-Meyer]

|K| bit

n bit

Marcatura Temporale di Documenti Digitali

- Il notaio digitale



- Quando è stato creato il documento $\mathcal{D}$?

Digital Timestamp

La ***marca temporale*** di un documento è qualcosa aggiunto ad esso che prova che il documento è stato "prodotto" **prima**, **dopo** oppure **ad** un fissato momento

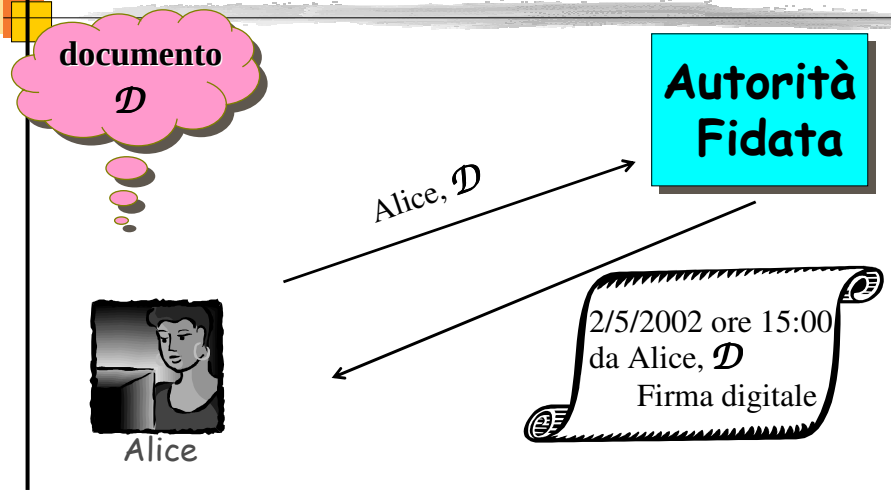
Alcune idee

- Depositare il documento presso un notaio
- Inviare il documento a se stesso, tramite il servizio postale
- Brevetto (se brevettabile…)
- Pubblicare il documento su di un giornale
- Uso di un registro di protocollo
- Foto con un quotidiano (se è un sequestro…)

Facile e Difficile

- È in genere ***facile*** provare che un documento è stato prodotto ***dopo*** una data fissata
- È in genere ***difficile*** provare che un documento è stato prodotto prima di una data fissata

Una soluzione naive



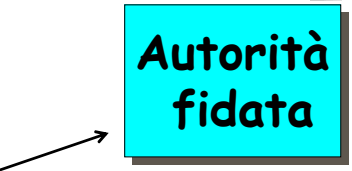
Problemi con la soluzione naive

- Dimensioni del documento $\mathcal{D}$
  - per la comunicazione
  - per la memorizzazione dell'Autorità Fidata
- Privatezza del contenuto di $\mathcal{D}$
- Quanto è fidata l'Autorità Fidata?

Idea: Funzioni Hash

- Dimensioni del documento $\mathcal{D}$
  - per la comunicazione
  - per la memorizzazione dell'Autorità Fidata
- Privatezza del contenuto di $\mathcal{D}$
- Quanto è fidata l'Autorità Fidata?

Soluzione naive migliorata



Alice, $H(\mathcal{D})$

Alice

2/5/2002 ore 15:00
Alice, $H(\mathcal{D})$
Firma digitale

Problema

Sed quis custodiet
ipsos custodes?

Giovenale, *Satire*, VI, 100 A.C.

Possibili Soluzioni

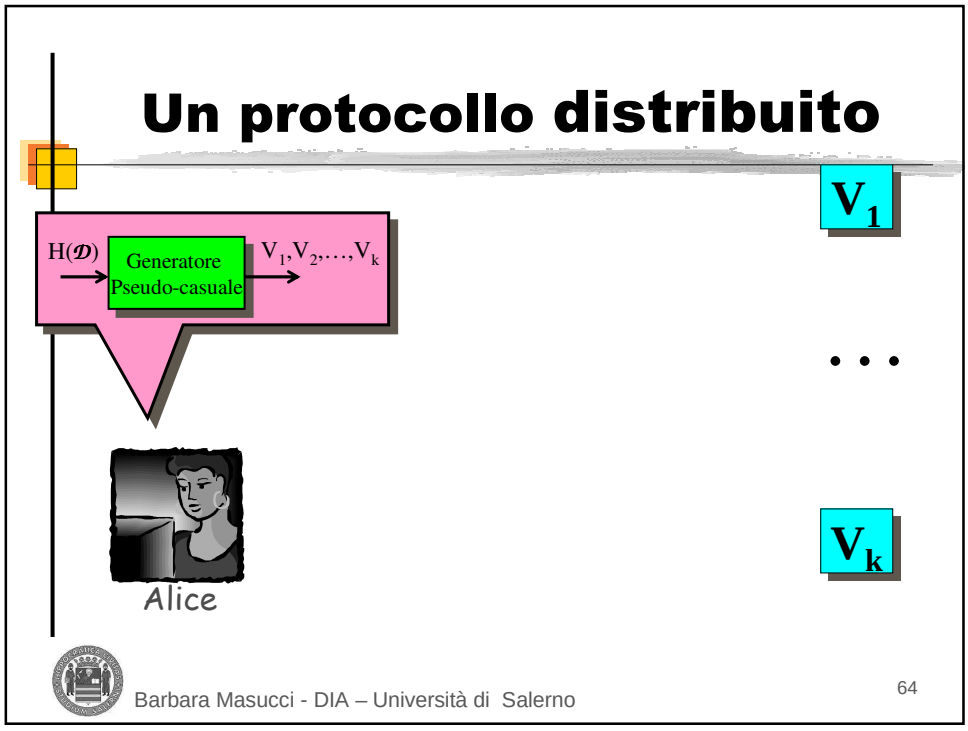
Due famiglie di protocolli

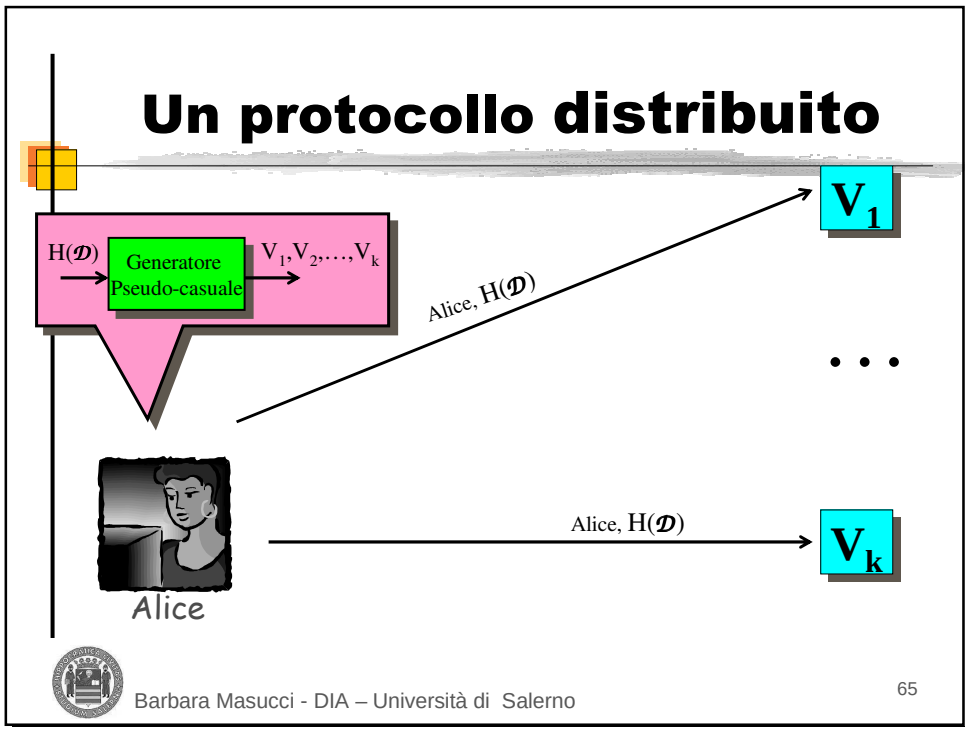
Protocolli distribuiti (senza Autorità Fidata)

Avere più "testimonianze" del tempo

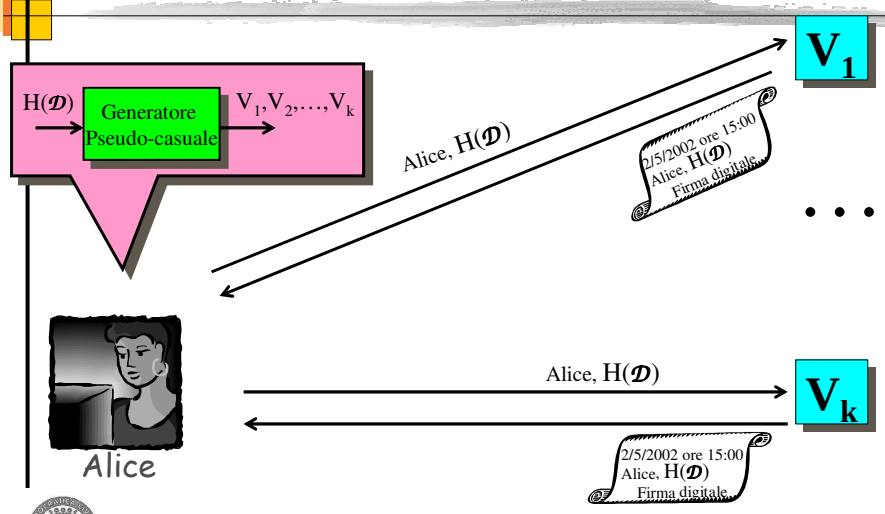
Protocolli con "link" (con Autorità Fidata)

Collegare tra loro le marche
dei documenti


Un protocollo distribuito
H(𝒟)
Generatore Pseudo-casuale
V1,V2,…,Vk
V1
• • •
Vk
Alice
Barbara Masucci - DIA – Università di Salerno
64


Un protocollo distribuito
H(𝒟)
Generatore Pseudo-casuale
V1,V2,…,Vk
V1
Alice, H(𝒟)
• • •
Alice, H(𝒟)
Vk
Alice
Barbara Masucci - DIA – Università di Salerno
65

Un protocollo distribuito



Protocollo Distribuito: Sicurezza

k grande ⇒ difficile per Alice corrompere k persone

La scelta delle persone da contattare è
- casuale
- dipendente dal documento

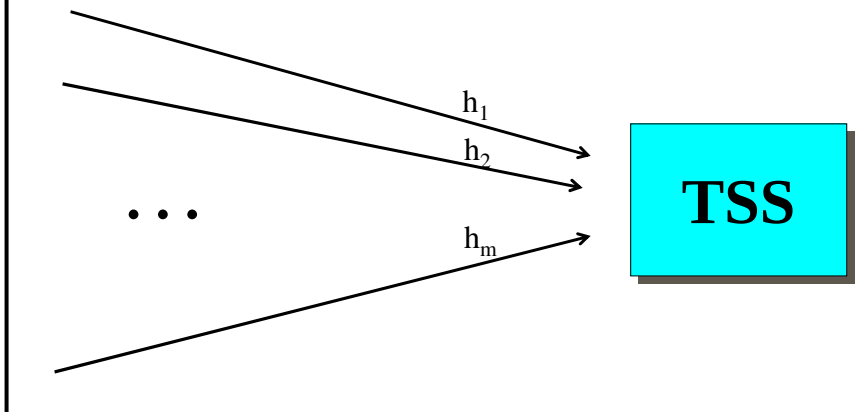
Protocollo Distribuito: Problemi

- Ci vogliono molte persone in grado di rispondere immediatamente ad Alice
- Durata (vita) delle firme digitali:
  - La firma potrebbe non essere più valida al tempo della verifica della marca temporale:
    - La chiave privata è stata compromessa
    - Lo schema di firme è stato rotto

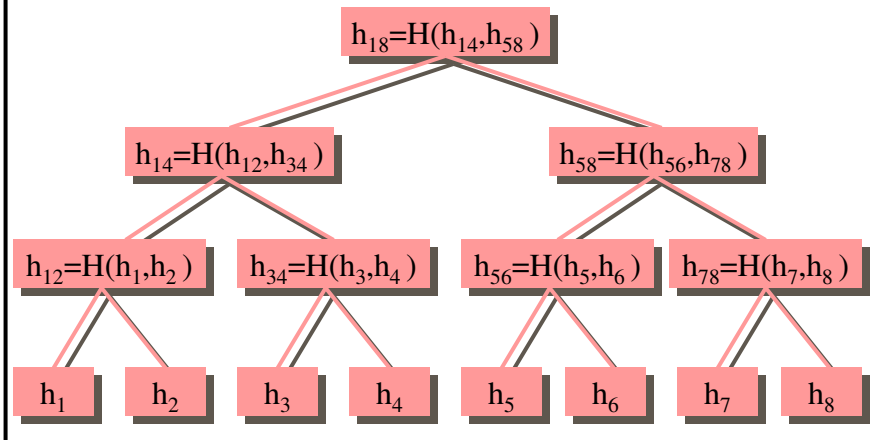
Protocollo con "link"

- Time Stamping Service
  - (Autorità fidata, … ma non troppo)
- Riceve tutte le richieste in intervalli prefissati
- Le collega tra loro
- Invia ad ognuno una marca temporale
- Vincola se stesso a "non poter predatare"

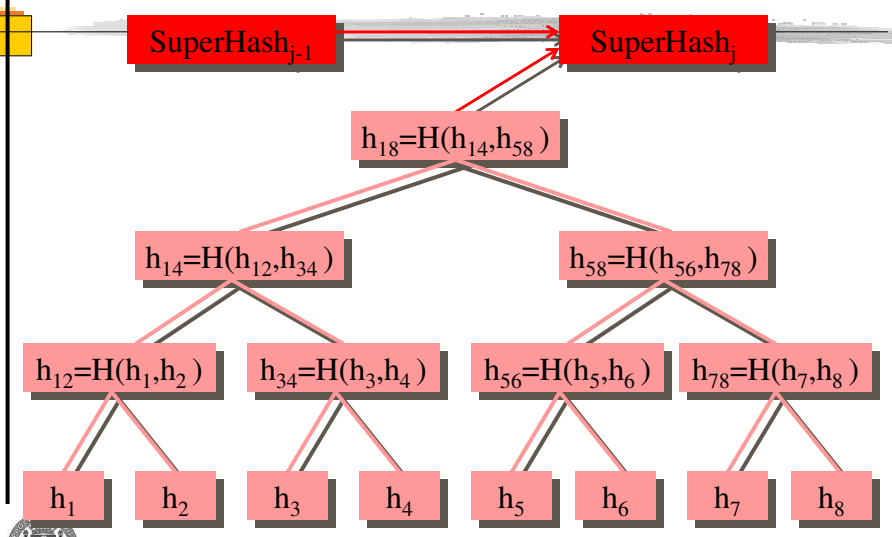
Ricezione richieste
in un prefissato intervallo di tempo



Costruzione albero di hash



Collegamento tra intervalli successivi

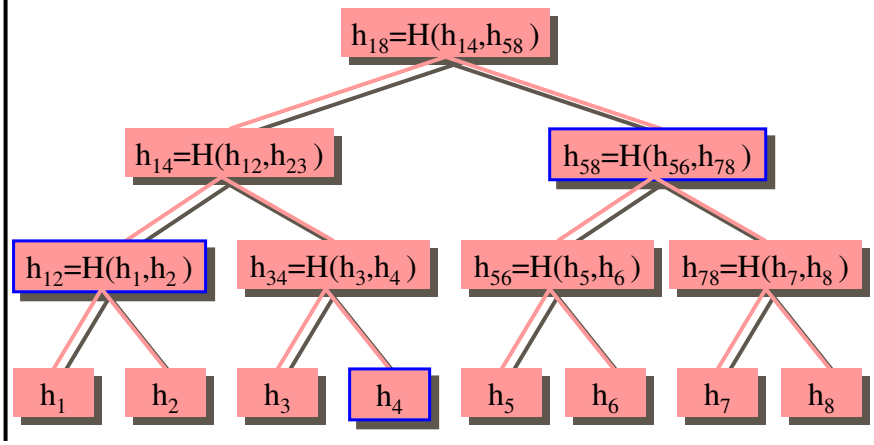


Marca temporale

Inviata per ogni richiesta ricevuta nell'unità di tempo

ID utente della richiesta
h_i
data ed ora
h_{1m} (valore hash della radice dell'albero)
info necessarie per verificare che h_i è stato utilizzato per costruire l'albero con radice h_{1m}
SuperHash$_{j-1}$ e SuperHash$_j$
Firma del TSS

Info per verifica di "h_3 in albero con radice h_{18}"



Sicurezza del Sistema

Fissato il valore hash della radice,
non è possibile

- inserire un nuovo valore nell'albero di hash
- cambiare anche un solo valore nell'albero di hash

…altrimenti si determinerebbe una collisione per la funzione hash

Sicurezza del Sistema

- Si potrebbe rompere lo schema colludendo solo con il TSS e creando una insieme di alberi collegati lunghi "a sufficienza"

- Una possibile soluzione:

  pubblicizzare SuperHash ad intervalli regolari
  - ogni giorno su Internet, su quotidiani,…
  - distribuzione mediante e-mail, CD,…

Digital Notary

http://www.surety.com

- Il cliente usa del software venduto dalla Surety
- Funzione hash con un digest di 288 bit (MD5+SHA)
- Il sistema usa una struttura ad albero
- L'unità di tempo corrisponde ad un secondo
- Un numero seriale è inserito nel documento
- Il SuperHash è pubblicato in posti accessibili via rete, su un CD-ROM, ed ogni settimana sul Sunday New-York Times

PGP Digital Timestamping Service

http://www.itconsult.co.uk/stamper.htm

Da ottobre 1995

- Il TSS firma ogni documento che riceve
- Ogni firma ha un numero seriale
- Il TSS memorizza tutte le firme che genera
- Tutte le marche (Serial Number, Date, Time) emesse possono essere esaminate
- Ogni giorno pubblica
  - Numero seriale dell'ultima firma effettuata
  - Tutte le marche emesse nella giornata

PGP Digital Timestamping Service

Posting settimanale

To:
stamper-weekly@stamper.itconsult.co.uk

Firma id 0076695 giorno 20020428
Firma id 0076762 giorno 20020429
Firma id 0076824 giorno 20020430
Firma id 0076896 giorno 20020501
Firma id 0076975 giorno 20020502
Firma id 0077059 giorno 20020503
Firma id 0077139 giorno 20020504

Stamper id 0077152
Firma PGP

PGP Digital Timestamping Service

Riepilogativi per anno

0069148 2002/01/01 23:53
0069210 2002/01/02 23:53
0069256 2002/01/03 23:53
0069307 2002/01/04 23:53
...
0077606 2002/05/09 23:53
0077687 2002/05/10 23:53
0077767 2002/05/11 23:53
0077841 2002/05/12 23:53

Firma PGP