FUNZIONI HASH
ONE-WAY ITERATE

SNEFRU SHA-1

di
Davide Gallo
per
Elementi di Crittografia 2004/2005
Prof.sa Rosaria Rota

Funzioni Hash : contesto

La crittografia asimmetrica è alla base della moderna crittografia ed è particolarmente utilizzata nei sistemi di protezione delle comunicazioni digitali via Internet.

Questo nuovo tipo di approccio consente di eliminare il problema dell'interscambio della chiave segreta, usata nei cifrari simmetrici sia per cifrare che per decifrare i messaggi.

Il problema dello scambio della chiave segreta viene eliminato introducendo una seconda chiave. In pratica si utilizza una chiave per le operazioni di cifratura ed un altra per le operazioni di decifratura, la chiave utilizzata per cifrare un messaggio viene resa pubblica (da qui il nome public key) e quindi inviata liberamente a chiunque mentre quella per decifrare rimane privata (da qui il nome private key) e quindi "segreta" di esclusiva proprietà dell'utente. In questo modo se due persone devono scambiarsi un messaggio non sono costrette a scambiarsi un'informazione segreta, così come avviene per i cifrari simmetrici, basterà inviare le rispettive chiavi pubbliche e cifrare il messaggio utilizzando la chiave del destinatario, non c'è quindi bisogno di tirare in ballo il "canale sicuro di comunicazione" che nella realtà non esiste. Due utenti A e B vogliono comunicare attraverso un canale insicuro di comunicazione, ad esempio Internet. L'utente A

desidera inviare un messaggio protetto all'utente B, per fare ciò utilizza la chiave pubblica dell'utente B (KpuB) cifrando il messaggio con un algoritmo di tipo asimmetrico.
Il messaggio viene spedito sul canale insicuro di comunicazione e quando arriverà all'utente B solo lui potrà decifrarlo utilizzando la propria chiave privata (KprB). La chiave pubblica di ogni utente può circolare tranquillamente sul canale insicuro di comunicazione un po' come avviene con i numeri di telefono delle persone, esistono infatti dei siti Internet, denominati key server, contenenti l'elenco delle chiavi pubbliche di moltissimi utenti.
Cerchiamo di approfondire il legame tra chiave pubblica e chiave privata con una domanda provocatoria: dal momento che la chiave privata è legata, in qualche modo, alla chiave pubblica di una persona, altrimenti non sarebbe possibile decifrare il msg. cifrato con la chiave pubblica corrispondente, è possibile ricavare la chiave privata da quella pubblica?
Per rispondere a questa domanda entrano in gioco la matematica e gli algoritmi utilizzati per la costruzione del cifrario asimmetrico. La stabilità di un algoritmo asimmetrico dipende proprio da quest'ultimo punto. L'impossibilità o meglio la complessità computazionale derivante dal calcolo della chiave privata da quella pubblica è strettamente collegata alla risoluzione di un problema matematico complesso come ad esempio la fattorizzazione di

numeri “grandi” (il concetto di numero “grande” ovviamente è relativo, attualmente per numero grande, in ambito crittografico, si intende un numero dell’ordine di 200 o più cifre). Attraverso l’utilizzo dei cifrari asimmetrici sono nati i moderni sistemi PKI (Public Key Infrastructure) ossia dei sistemi per la protezione delle comunicazioni nei quali vengono utilizzati uno o più cifrari asimmetrici.

Oltre alla Segretezza i moderni sistemi di crittografia a chiave asimmetrica vogliono garantire :

1. **Integrità** del messaggio inviato (si vuole evitare il rischio che, per errori di trasmissione o per intrusione, un messaggio pervenga al destinatario corrotto rispetto all’originale)

2. **Autenticazione** dell’invio (il mittente deve essere riconoscibile come autore del messaggio)

3. **Impossibilità di ripudio** (il mittente non deve poter negare la produzione del messaggio che è avvenuta per sua volontà in un certo istante)

4. **Identificazione** del destinatario (il mittente deve essere sicuro dell’associazione chiave pubblica/destinatario)

Funzioni Hash : finalità

I primi due punti (**Integrità e Autenticazione**) sono risolti con la nota tecnica della **Firma Digitale**, che nella sua forma più semplice, non è altro che il messaggio cifrato con la chiave privata del mittente.

Uno degli aspetti più conosciuti della crittografia a chiave pubblica soprattutto per le sue implicazioni di carattere legale è la firma digitale o firma elettronica (vedi il DPR 10 novembre 1997, n. 513 ed il recente decreto legislativo n. 10 del 2002, pubblicato in Gazzetta ufficiale n.39 il 15 febbraio 2002, che recepisce la Direttiva del Parlamento europeo e del Consiglio del 13 dicembre 1999 n.93).

La firma digitale consiste in un procedimento matematico che consente di legare un documento elettronico al suo legittimo proprietario attraverso l'utilizzo di una sequenza univoca di numeri binari, denominata appunto firma digitale.

La firma digitale nasce dall'utilizzo della crittografia asimmetrica poiché per consentire l'autenticità del documento firmato viene utilizzato un algoritmo a chiave pubblica tramite la chiave privata dell'utente firmatario. A differenza quindi del procedimento di cifratura di un documento nel quale l'utente utilizza la chiave pubblica del destinatario del messaggio per la protezione dello

stesso, nel caso del processo di firma di un documento l'utente utilizzerà la propria chiave privata per firmare il messaggio. Si tenga presente che in quest'ultimo caso non si ha la cifratura del messaggio, quindi esso non risulta protetto da occhi indiscreti, il processo di firma serve solo per garantire l'integrità e l'autenticità del messaggio, o meglio l'autenticità dell'utilizzo di una particolare coppia di chiavi pubbliche e private.

Dal punto di vista teorico basterà applicare tale procedimento sull'intero documento con la propria chiave privata per generarne una firma elettronica, infatti chiunque attraverso l'utilizzo della chiave pubblica dell'utente firmatario potrà verificare l'autenticità dello stesso attraverso un procedimento matematico inverso rispetto al precedente.

Dal punto di vista pratico la firma digitale viene applicata soltanto ad un "sunto" (**residuo**) del documento, questo per evitare dei tempi di elaborazione troppo lunghi nel rilascio della firma, dipendenti dalla complessità computazionale degli algoritmi asimmetrici, e soprattutto per evitare che la firma digitale occupi più spazio, in termini di byte, del documento stesso.

Per questo motivo si utilizzano le **Funzioni Hash** che consentono di concentrare l'unicità di un documento in poche centinaia di byte o caratteri : l'**Impronta del messaggio** (in inglese **Fingerprint** o anche **Message Digest**).

In pratica queste funzioni matematiche consentono di generare una sorta di targa univoca di un documento digitale.
L'output di una funzione hash, rappresentato da una sequenza di byte di lunghezza fissa, è strettamente legato, in maniera biunivoca, con il documento digitale in input.
Ciò vuol dire che il risultato di una stessa funzione hash applicata su due documenti diversi deve essere differente anche se i due documenti differiscono di un solo byte.
Ritornando al discorso della firma digitale possiamo sfruttare il risultato di una funzione hash come "sunto" del documento ed applicare il procedimento di firma, descritto all'inizio del paragrafo, solo su tale sequenza di byte rappresentativa del documento. E proprio questo il procedimento che viene utilizzato dalla maggioranza dei software in circolazione per il rilascio della firma elettronica.
In pratica utilizzando questi procedimenti di firma su di un documento digitale otterrò una sequenza di byte di lunghezza fissa strettamente legata al documento stesso. Tale firma potrà essere inviata in chiaro, ad esempio, in coda al documento come se fosse stato firmato in maniera tradizionale utilizzando carta e penna. Tale procedimento matematico consente di garantire l'autenticità, l'integrità ed il non ripudio del documento firmato elettronicamente.

Per la legge italiana, una firma digitale per risultare valida dovrà essere convalidata attraverso una Certification Authority tramite il rilascio di un certificato digitale che attesti la corrispondenza biunivoca tra la chiave pubblica e le persona fisica.

Vediamo nel dettaglio come avviene il rilascio del certificato digitale con l'utilizzo di una CA.

Il primo passo da realizzare è il rilascio del certificato digitale della CA. Questa operazione viene eseguita dopo aver inviato i dati personali dell'utente, comprensivi della chiave pubblica, alla CA. La **Certification Authority** dopo aver verificato la correttezza dei dati relativi all'utente inserirà questi valori nel proprio archivio assegnando all'utente un codice identificativo (ID). Il rilascio del certificato digitale avviene dopo aver apposto la firma, utilizzando la chiave privata della CA, sui dati relativi all'utente comprensivi di ID e chiave pubblica.

La verifica del certificato digitale da parte del destinatario avverrà semplicemente utilizzando la chiave pubblica della CA, in questo modo si verificherà l'autenticità della chiave pubblica dell'utente (compresa nel certificato) che potrà essere utilizzata per verificare a sua volta la firma digitale del messaggio originale del mittente.

Funzioni Hash : caratteristiche

1. Una Funzione di Hash (Message Digest) è una funzione H che, dato **un input M di dimensione qualsiasi**, produce un **output Y (Hash) di dimensione fissa** (in genere 128 bit) : Y = H(M)

2. L'idea alla base è che il valore hash H(M) sia una rappresentazione **non ambigua e non falsificabile** del messaggio M

3. Dato un input X deve essere altamente improbabile che un altro input Z generi lo stesso hash (**Collisioni**) cioè : **H(Z) = H(X)**

4. Inoltre la funzione deve essere **computazionalmente semplice** da eseguire, cioè calcolare Y come H(M)

5. **One-Way** (irreversibili) : cioè deve essere computazionalmente impossibile, noto H(M), ricalcolare M

Le Funz. Hash si classificano in base alla resistenza alle collisioni:

- Resistenza Debole alle Collisioni :
  "Dato M, è computazionalmente impraticabile trovare un altro M´ tale che H(M) = H(M´)"

- Resistenza Forte alle Collisioni :
  "Computazionalmente impraticabile trovare una coppia (M, M'), con M'≠ M, tale che H(M) = H(M')

Dalla Resistenza alle Collisioni dipende la Sicurezza di una Funzione Hash !!

Funzioni Hash : attacchi

Le due proprietà sull'assenza di collisioni (debole e forte), corrispondono due diversi tipi di attacchi :

- **Attacco a forza bruta (Brute Force) :**

Trovare un messaggio che produca un dato hash (cioè un hash uguale a quello di un messaggio dato)

Caso di una **funzione hash debole** contro le collisioni con **output di 32 bit**:

1. Mefisto prepara 2 versioni di un contratto M ed M* tali che :
   - **M** è favorevole ad Annarella
   - **M*** è sfavorevole ad Annarella

2. Mefisto **modifica M* a caso** facendo piccoli cambiamenti come aggiunta di spazi, punteggiatura, sinonimi (32 possibilità sono **232 messaggi** !) **finchè H(M) = H(M*)**

3. Annarella firma M con la sua chiave => **Firma(h(M))**

4. Mefisto ha quindi ottenuto anche la firma di M* => **Firma(h(M*)) !!**

Un esempio di possibile messaggio modificabile (non fraudolento)

Cara Annarella,

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

……..

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

……..

- **Attacco del compleanno (Birthday Attack) :**

Trovare due messaggi che producano lo stesso hash, indipendentemente dal valore di questo hash.

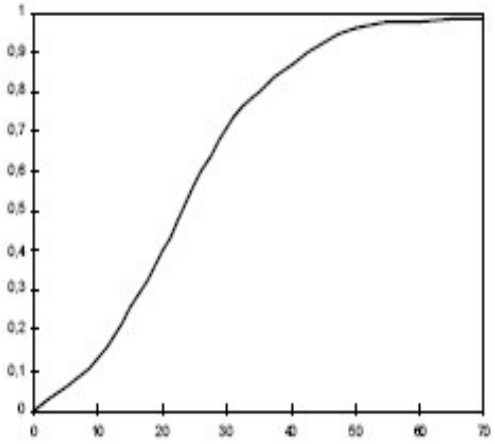
Domanda : "Quante persone scegliere a caso affinché, con probabilità ≈ 50%,ce ne siano almeno due con lo stesso compleanno?"

Risposta : "Ne bastano 23 !"

N = 365 numero di giorni possibili ; T = numero di persone prese a caso

ε = Prob. di trovarne due uguali e 1- ε = Prob. che siano tutti diversi

$$\varepsilon = \frac{365 \cdot 364 \cdot \ldots \cdot (N - t + 1)}{365^t}$$



Quindi con t = 23 à ε = 0,5073
mentre con t = 100 à ε = 0,99997

Applicando all'inverso questo ragionamento otteniamo il principio di base :

1. **Scelgo a caso diversi messaggi**
2. **Verifico se ottengo almeno due valori hash uguali (collisione)**

Problema da risolvere :

Dato un insieme di cardinalità **n** (output dell'hash) , quanti elementi **t** (messaggi) scegliere se si vuole che la probabilità che ci siano almeno due elementi uguali sia ε ?

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

Cioè $\mathbf{t \approx \sqrt{n}}$ con $\varepsilon = 1 - e^{-t(t-1)/n}$

es. con $\mathbf{n = 2^{128}}$ servono $\mathbf{t = 2^{64}}$

Confrontando :

La **complessità** dei due attacchi è molto diversa : se l'hash è lungo m bit, la complessità del primo è **2m**, quella del secondo è **2m/2** !!

Sicurezza con Hash 128 bit :

Costo di un attacco per computare collisioni 264 ≈ 2*1019 valutazioni della funzione :

Attacco < 1mese con costo di $10.000.000

P. van Orschot e M. Wiener [1994]

I due autori ipotizzano che il costo si dimezza ogni 18 mesi.

Oggi quindi costerebbe solo $75.000 dollari, costo che qualcuno potrebbe ritenere economicamente accettabile per falsificare un contratto !

Funzioni Hash Iterate

Per difendersi da questi attacchi, si è reso quindi necessario uno studio per creare delle funzioni hash, che rendessero la probabilità di trovare delle collisioni più bassa possibile. Una soluzione, divenuta la base per molti algoritmi, è quella che si basa sul concetto di iterazione (ovvero ripetizione dell'algoritmo, con delle varianti ad ogni passaggio). Questa famiglia di tecniche sono note come **Funzioni Hash Iterate**.

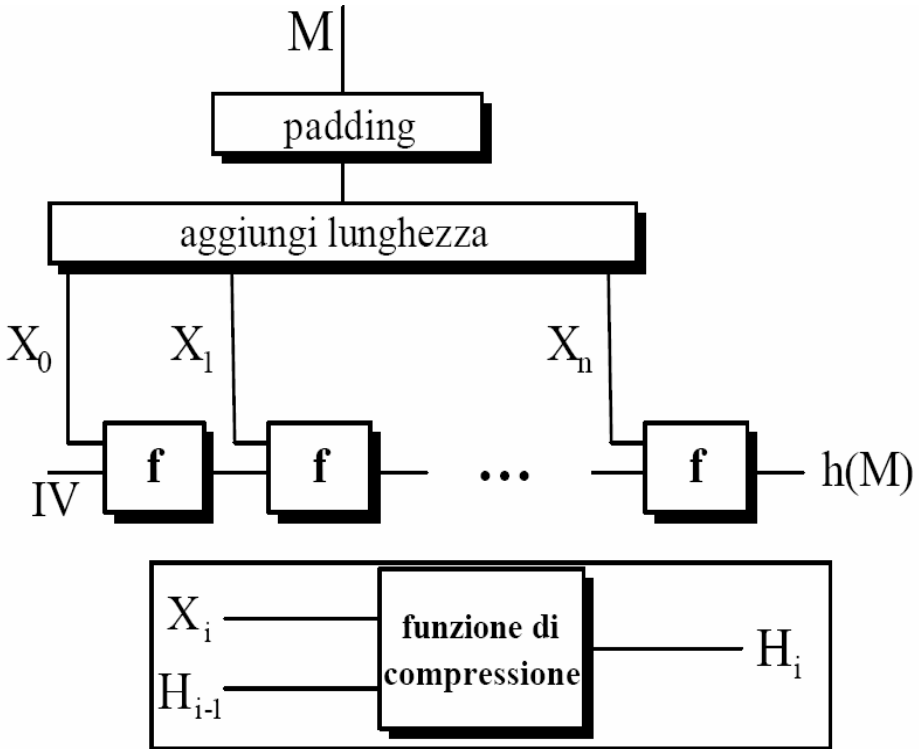
Caratteristiche :

- **Veloci :** implementabili via software in modo efficiente
- **Sicure :** difficile trovare due messaggi diversi che vadano in collisione

Principi :

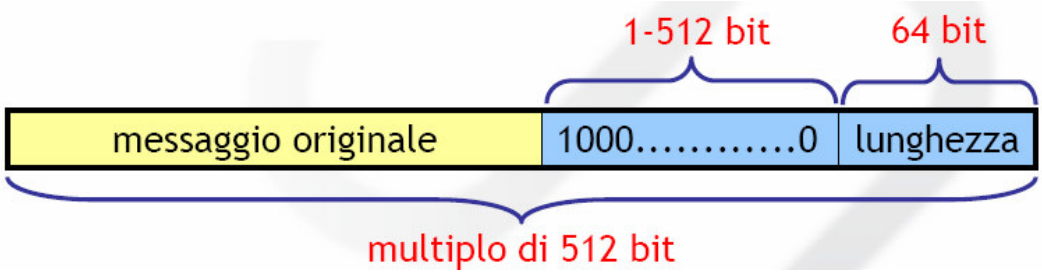
- **Input di lunghezza fissa :** il messaggio viene suddiviso in blocchi di dimensione prefissata pari all'input (in genere 512 bit),quindi la lunghezza totale deve essere un multiplo di quella dei blocchi, ciò si ottiene aggiungendo un **PADDING**
- **Parametrizzazione :** la funzione viene iterata in modo che al passo n prende in input un blocco del messaggio e l'hash calcolato al passo n-1, inoltre viene inizializzata con un valore Random detto **IV** (initializing value)

Modello di funzionamento



Padding :

- Va aggiunto al messaggio un certo numero di bit in modo che la lunghezza totale risulti un **multiplo di 512** bit.
- Si aggiunge **un bit a 1** e **poi tanti bit a 0** quanto basta perché la lunghezza risulti di 64 bit minore rispetto ad un multiplo di 512 bit (se la lunghezza originale è già corretta si aggiungono comunque 512 bit).
- Si aggiungono **64 bit** contenenti la **lunghezza originale del messaggio (modulo 2^{64})**.



Funzioni di Compressione

Il cuore di ogni algoritmo, risiede sempre nella funzione di compressione, il cui ruolo è quello base delle funzioni hash, cioè trasformare l'input in un residuo.

Caratteristiche generali :

- Funzione HASH applicata ad ogni iterazione su uno dei blocchi del messaggio insieme al risultato dell'iterazione precedente.
- E' costituita dalla composizione di **operazioni bit-wise : SHIFT, AND, OR, XOR, ROTATION.**
- E' suddivisa in passi di elaborazione detti **ROUND.**
- Per la **proprietà della composizione** :
  se H(M)=F(M1)°….° F(Mn) ha collisioni tutte le F(Mi) hanno collisioni.

Ogni famiglia di algoritmi hash, si defferenzia dalle altre proprio per questo elemento, infatti è da esso che dipendono in larga parte la robustezza, l'efficienza e il costo computazionale di tutto l'algoritmo.

Ne presenteremo due molto famose : SNEFRU e SHA-1, accennando anche agli altri principali algoritmi (RIPEMD, MD5).

SNEFRU

- Funzione One-Way Hash inventata da **Ralph Merkle**, nel 1990
- Capostipite di una famiglia di algoritmi di crittografia che include i successori **Khufu** e **Khafre** (leggendari Faraoni Egizi)
- Khufu è veloce e adatta a grandi quantità di dati perché si avvale di cifrari precalcolati
- Khafre è lenta e adatta a piccole quantità di dati (non usa cifrari precalc.)
- Elabora **blocchi di 512 bit**
- Fornisce attualmente in **outbut digest di 128 oppure 256 bit**
- La funzione prende in input l'elemento calcolato all'iterazione precedente e lo cripta usando **un cifrario con una chiave fissa**, dopodichè applica al risultato uno **XOR** con il blocco corrente :

$$F(B_n) = Encript(key, F(B_n\text{-}1)) \text{ XOR } B_n$$

N.B. La versione 2-round a 64 bit era stata violata nel 1998 da Eli Biham.

SHA-1

- SHA-1 (**Secure Hash Algorithm 1**) è stato progettato dal **NIST**

- Il messaggio (che deve essere di lunghezza inferiore a 2^{64} bit) viene elaborato a blocchi di 512 bit.

- Fornisce in **outbut digest di 160 bit**.

- Ad ogni iterazione si calcola una funzione che dipende dal blocco corrente del messaggio e dal valore dell'hash all'iterazione precedente: il risultato viene sommato (per singola parola) al valore dell'iterazione precedente.

- Le **5 parole di 32 bit** del digest (A, B, C, D, E) vengono inizializzate con:

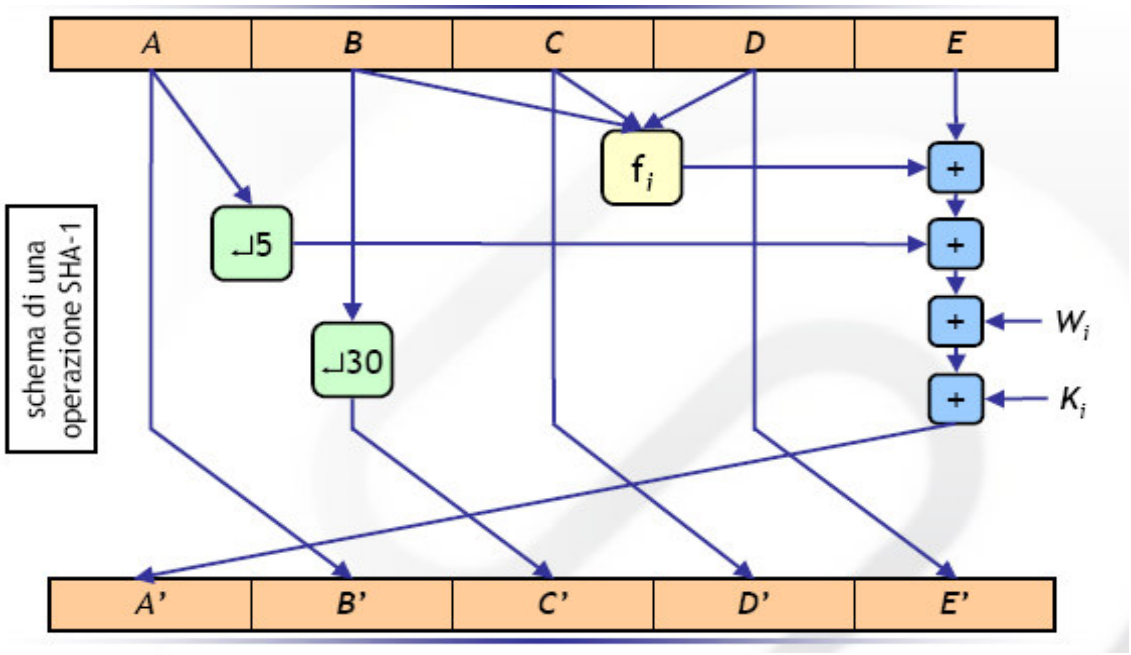
A = 0x67452301, B = 0xEFCDAB89, C = 0x98BADCFE, D = 0x10325476,
E = 0xC3D2E1F0.

- Ogni iterazione calcola dei valori di A, B, C, D, E che vengono sommati ai precedenti prima di passare all'iterazione successiva.

- Alla fine, **la concatenazione di A, B, C, D, E costituisce il digest** del messaggio.

SHA-1 : ITERAZIONI

- Ogni iterazione è composta da **80 operazioni.**
- Il blocco corrente di messaggio (512 bit) viene utilizzato per costruire **una stringa di 5 x 512 bit (80 parole di 32 bit)**.
- Si scorre la stringa di 80 parole una alla volta (80 operazioni), calcolando i nuovi valori A', B', C', D', E' (con operazioni bit-wise specifiche).
- Alla fine, i nuovi valori verranno aggiunti agli A, B, C, D, E precedenti.
- Per calcolare la stringa vengono usati lo **XOR** e la **Rotazione di 1 bit** (aggiunta di recente)



Altre funzioni : RIPEMD, MD5

❑ RIPEMD-160 :

- ❑ **RACE Integrity Primitives Evaluation**, progetto della Comunità Europea (1988-1992)
- ❑ **Tabella "left line" e "right line"**
- ❑ Input blocchi di 512 bit (**16 parole da 32 bit**)
- ❑ Output **digest di 160, 250, 320 bit**
- ❑ E' risultata **debole e facilmente attaccabile**

❑ MD5 :

- ❑ Message Digest 5 è stato progettato **da Ron Rivest**, ed è definito in **RFC 1321** (1992)
- ❑ Input blocchi di 512 bit
- ❑ Output **digest di 128 bit**
- ❑ Ogni iterazione prevede **4 passi** per elaborare altrettante parole, e **ogni passo 16 operazioni**
- ❑ **Non è considerato molto sicuro**
- ❑ E' molto efficiente e viene utilizzato per grandi quantità di dati

Implementazioni e prestazioni

Per concludere possiamo affermare che dal punto di vista della sicurezza, Snefru offre sicuramente maggiori garanzie in virtù dell'utilizzo di un cifrario, ma questo ha un prezzo da pagare in termini di prestazioni. Dall'altra parte troviamo l'altissima efficienza di MD5, cha però non offre reali garanzie di sicurezza, e viene per questo utilizzato in contesti non critici. Nel mezzo ci sono in fine SHA-1, Tiger e RIPEMD, che in virtù del buon rapporto tra robustezza e prestazioni, sono gli algoritmi ad oggi più diffusi , se è richiesto un certo grado di sicurezza.

	Size (byte)	Cicli	Mbit/sec	Mbyte/sec
MD2		2709	4.25	0.53
MD4	1190	241	191.2	23.90
MD5	1713	337	136.7	17.09
RIPEMD	2291	480	96.0	12.00
RIPEMD-128	2929	592	77.8	9.73
SHA-1	4323	837	55.1	6.88
RIPEMD-160	4808	1013	45.5	5.69
Snefru-128		5730	6.03	0.75
Snefru-256		5738	4.02	0.50
Tiger		1320	34.9	4.36

Rif. Architetture 80x86 (Pentium 90Mhz) , 1996