

Corso di Sicurezza

A.A. 2006-2007

Funzioni hash crittografiche e paradosso del compleanno

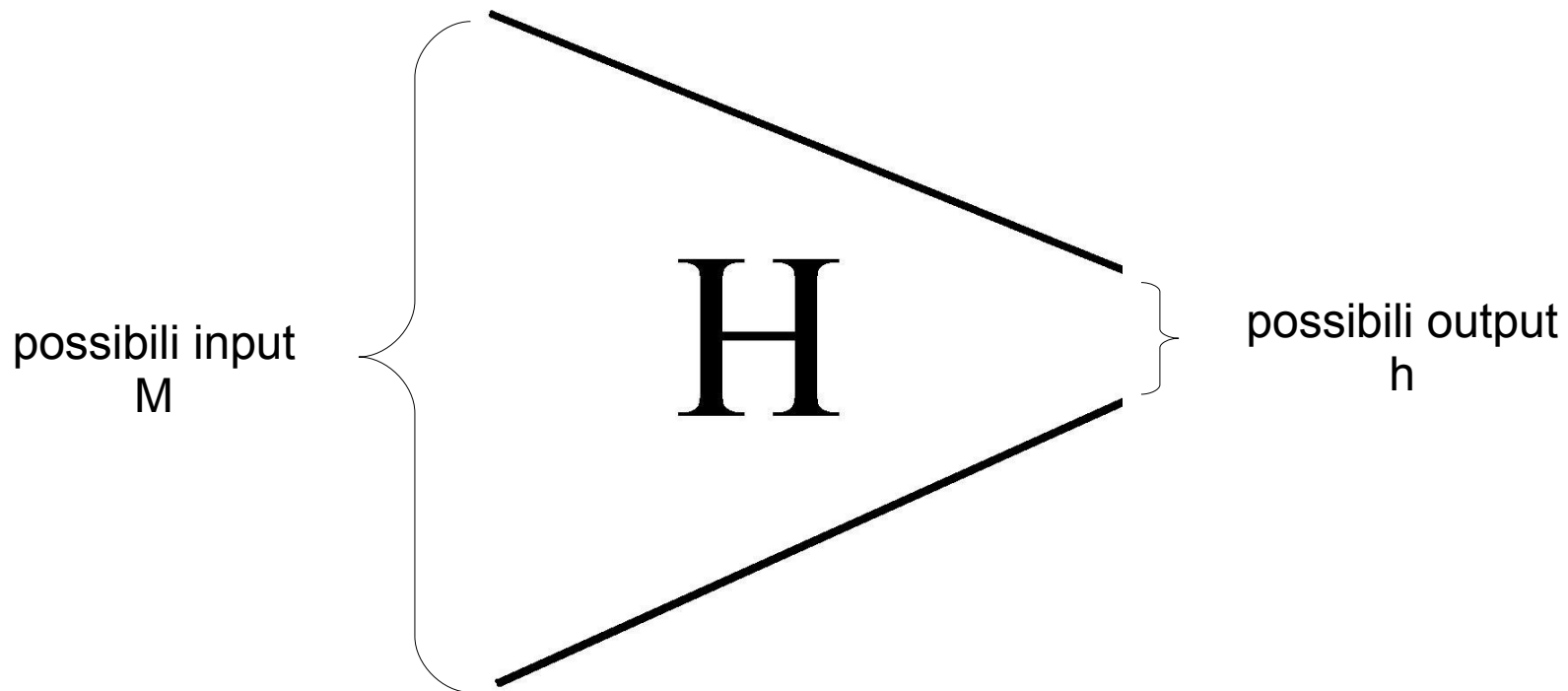
22/05/2007

Paolo BIONDI
0000269207

Cos'è una funzione hash

- Qualunque funzione che trasformi i dati in input in un output di lunghezza costante

- $H: M \rightarrow h$



Proprietà delle funzioni hash 1

- M può essere di qualsiasi dimensione
- h è sempre della stessa dimensione
- $H(M)$ è facile da calcolare
- ogni valore di h ha la stessa probabilità di essere restituito

Proprietà delle funzioni hash 2

Per essere usate in ambito crittografico le funzioni devono avere altre proprietà:

1 - Unidirezionalità

noto h deve essere computazionalmente impossibile trovare M tale che $H(M) = h$

2 - Resistenza debole alle collisioni

conoscendo M deve essere computazionalmente impossibile trovare m' tale che $H(M) = H(M')$

3 - Resistenza forte alle collisioni

deve essere computazionalmente impossibile trovare una coppia (M, M') tale che $H(M) = H(M')$

4 - Effetto valanga

una piccola modifica di M deve alterare tutto h

Controllo di ridondanza longitudinale (CRL)

Usa un solo operatore logico, l'OR esclusivo (XOR).

L'input viene suddiviso in m blocchi di n bit e la funzione H è così definita:

$$h_i = b_{1,i} \oplus b_{2,i} \oplus \dots \oplus b_{m,i}$$

	bit 1	bit 2	...	bit n
blocco 1	$b_{1,1}$	$b_{1,2}$...	$b_{1,n}$
blocco 2	$b_{2,1}$	$b_{2,2}$...	$b_{2,n}$
...
blocco m	$b_{m,1}$	$b_{m,2}$...	$b_{m,n}$
Codice hash	h_1	h_2	...	h_n

Problemi del CRL

La funzione CRL è ragionevole per dati soggetti ad errori casuali ma non idonea come funzione hash crittografica a causa di molte sue debolezze.

- 1 - L'ordine dei blocchi in ingresso non influisce su h
un estraneo può alterare la sequenza del flusso dati senza alterare $H(M)$
- 2 - È possibile creare m che restituisca l'hash desiderato
dati M (messaggio originale) e M' (messaggio contraffatto) ho $H(M) \neq H(M')$
aggiungendo ad M' il blocco $H(M) \oplus H(M')$ ottengo M'' e ho $H(M) = H(M'')$
- 3 - Non ha alcun effetto valanga, variando un bit di M varia solo un bit di h

Alternativa:

Come primitiva di cifratura uso lo XOR ruotato (LXOR), elimino i problemi 1 e 3 ma non ho ancora raggiunto il livello di sicurezza richiesto.

Concatenamento a blocchi

Tecnica di hash proposta da Rabin negli anni '70:
si suddivide M in n blocchi di uguale lunghezza e si utilizza un algoritmo di cifratura simmetrico (es. DES) per calcolare $H(M)$.

$$H_0 = \text{Valore iniziale}$$

$$H_i = E_{M_i}[H_{i-1}]$$

$$G = H_n$$

Questa tecnica è sicura nella misura in cui è sicuro l'algoritmo di cifratura utilizzato.

Il messaggio funge da chiave per DES.

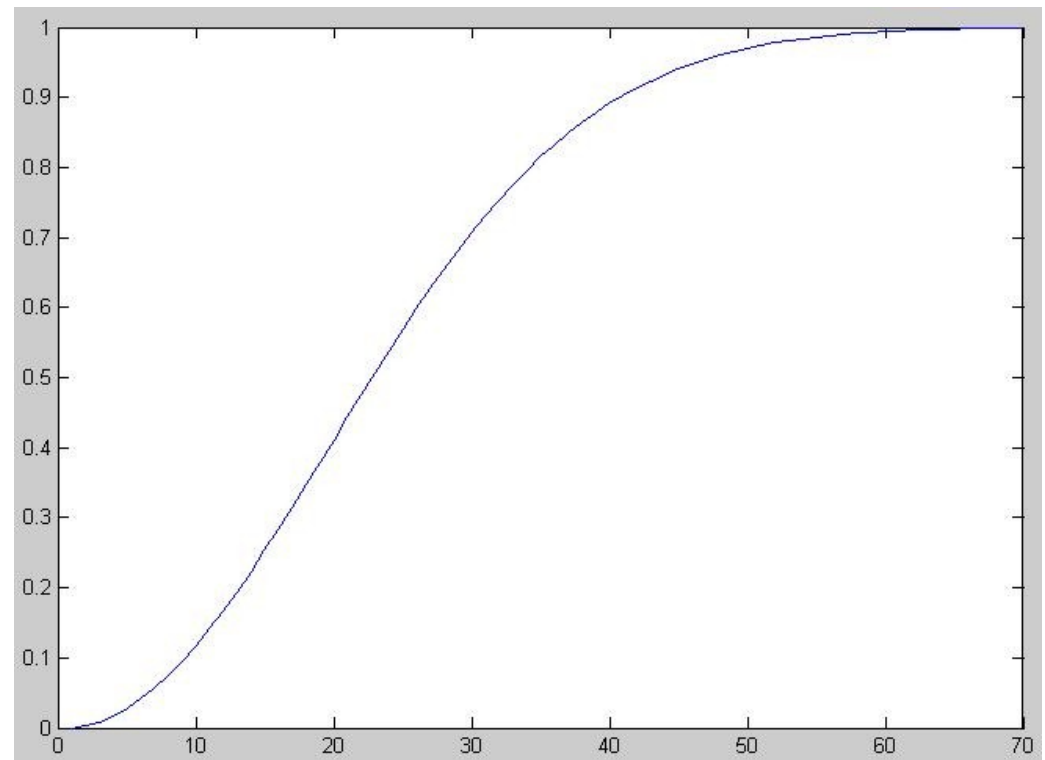
É vulnerabile inoltre all'attacco Meet-in-the-middle.

Paradosso del compleanno 1

La probabilità che in un gruppo di persone almeno due siano nate nello stesso giorno è molto più alta di quanto si possa immaginare.

Questa probabilità è:

$$P(365, k) = 1 - \frac{365!}{(365 - k)! 365^k}$$



Si ottiene una probabilità del 50% con sole 23 persone.

Paradosso del compleanno 2

Si suppongono gli n valori equamente probabili.

La formula generale è:

$$P(n, k) = 1 - \frac{n!}{(n-k)!n^k}$$

Con le dovute approssimazioni ottengo

$$P(n, k) > 1 - e^{-\frac{k(k-1)}{2n}}$$

Eguagliando questa quantità a 0,5 scopro per che valore di k ho una probabilità del 50% di avere due elementi uguali

$$k = \sqrt{\ln(2) * 2n} = 1,18 \sqrt{n} \approx \sqrt{n} = \sqrt{2^m} = 2^{\frac{m}{2}}$$

Per verificare l'esattezza della formula la proviamo con il risultato che già conosciamo:

$$k = 1,18 \sqrt{365} = 22,54$$

Paradosso del compleanno 3

Dati due insiemi X e Y di k elementi che possono assumere un valore tra 1 e n, qual'è la probabilità che i due insiemi non siano disgiunti?

Supponendo gli elementi tutti distinti all'interno del proprio insieme otteniamo:

$$P(X \cap Y) = 1 - \left(1 - \frac{1}{n}\right)^{k^2}$$

Con le dovute approssimazioni ottengo

$$P(X \cap Y) = 1 - e^{-\frac{k^2}{n}}$$

Imponendo questa quantità uguale a 0,5 trovo:

$$k = \sqrt{\ln(2) * n} = 0,83 \sqrt{n} \approx \sqrt{n} = \sqrt{2^m} = 2^{\frac{m}{2}}$$

Tipi di attacchi

Gli attacchi possono essere divisi in due categorie:

1 - forza bruta

Si effettua una ricerca esaustiva dei possibili valori fino a trovare quello desiderato, l'attacco a compleanno rientra in questa categoria

2 - analisi crittografica

Si studia il funzionamento della funzione crittografica, dopodiché si individuano i punti deboli e si cerca di generare delle collisioni

Attacco a compleanno 1

Questo è un esempio di attacco basato sul paradosso del compleanno.

- 1 - Il mittente A si prepara a “firmare” un messaggio, m , aggiungendo il relativo codice hash (lungo n bit) crittografato con la propria chiave privata
- 2 - L'estraneo genera $2^{\frac{n}{2}}$ varianti di m , tutte con lo stesso significato. Prepara inoltre un uguale numero di varianti del messaggio fraudolento da sostituire al vero messaggio.
- 3 - Per il paradosso del compleanno ha una probabilità maggiore di 0,5 di trovare una coppia di messaggi (vero, fraudolento) con lo stesso codice hash. In caso di insuccesso è sufficiente generare altri due insiemi di messaggi.
- 4 - L'estraneo offre la variante valida ad A per la firma. Poiché le due varianti hanno lo stesso codice hash, produrranno la stessa firma; ciò permetterà all'estraneo di sostituire il messaggio firmato da A con uno fraudolento senza dover conoscere la chiave segreta di A.

Attacco a compleanno 2

Ecco come ottenere 2^{36} varianti di un messaggio senza alterarne il significato:

{ This letter is } to introduce { you to } { Mr. } Alfred { P. }
{ I am writing } { to you } { -- }
Barton, the { new } { chief } jewellery buyer for { our }
{ newly appointed } { senior } { the }
Northern { European } { area } . He { will take } over { the }
{ Europe } { division } { has taken } { -- }
responsibility for { all } our interests in { watches and jewellery }
{ the whole of } { jewellery and watches }
in the { area } . Please { afford } him { every } help he { may need }
{ region } { give } { all the } { needs }
to { seek out } the most { modern } lines for the { top } end of the
{ find } { up to date } { high }
market. He is { empowered } to receive on our behalf { samples } of the
{ authorized } { specimens }
{ latest } { watch and jewellery } products, { up } to a { limit }
{ newest } { jewellery and watch } { subject } { maximum }
of ten thousand dollars. He will { carry } a signed copy of this { letter }
{ hold } { document }
as proof of identity. An order with his signature, which is { appended }
{ attached }
{ authorizes } you to charge the cost to this company at the { above }
{ allows } { head office }
address. We { fully } expect that our { level } of orders will increase in
{ -- } { volume }
the { following } year and { trust } that the new appointment will { be }
{ next } { hope } { prove }

Attacco Meet-in-the-middle

Si suppone che l'estraneo abbia intercettato un messaggio in chiaro con relativo hash crittografato.

Sono noti M e $E_k[H(M)]$

1 - Usando l'algoritmo visto in precedenza si calcola $G = H(M)$

2 - Si costruisce il messaggio contraffatto Q_1, Q_2, \dots, Q_{n-2}

3 - Calcolo $H_i = E_{Q_i}(H_{i-1})$ per $1 \leq i \leq n-2$

4 - Genero $2^{m/2}$ blocchi casuali, per ogni blocco X calcolo $E_X(H_{n-2})$. Genero ulteriori

$2^{m/2}$ blocchi casuali, per ogni blocco Y calcolo $D_Y(G)$, D è la funzione inversa di E

5 - Per il paradosso del compleanno ho una probabilità del 50% di trovare un X e un Y

per cui $E_X(H_{n-2}) = D_Y(G)$

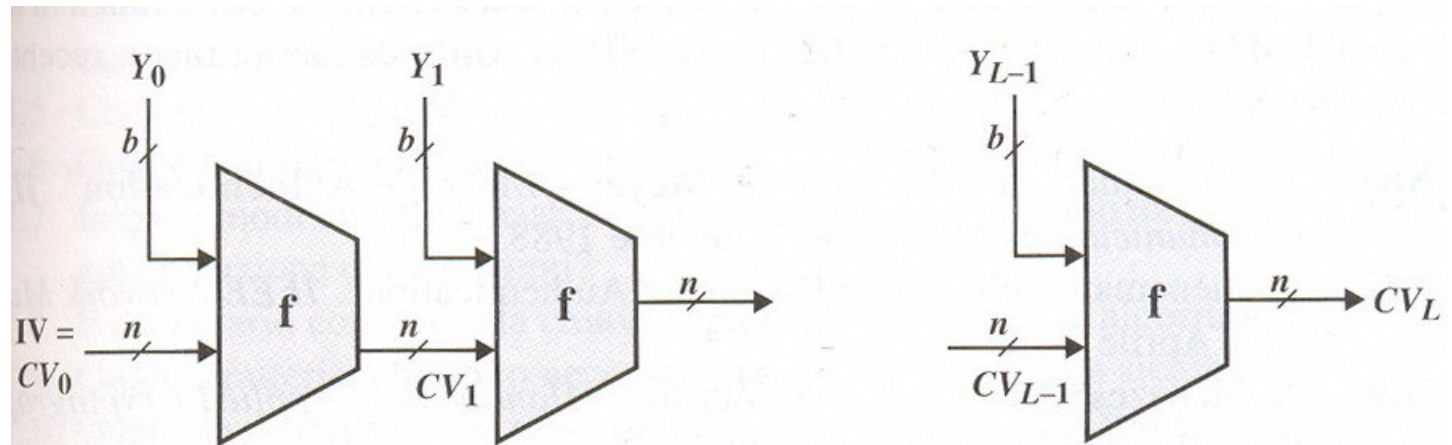
6 - Costruisco infine $M' = Q_1, Q_2, \dots, Q_{n-2}, X, Y$ e ottengo $H(M') = H(M) = G$

Tutto ciò è necessario in quanto l'estraneo non può modificare il codice hash del messaggio perché cifrato, l'unica soluzione possibile è la creazione di un nuovo messaggio che abbia lo stesso hash dell'originale.

Funzioni hash sicure 1

La maggior parte delle funzioni hash crittografiche adottano la struttura iterata proposta da Merkle e Damgard all'inizio degli anni '80.

$$CV_0 = IV$$
$$CV_i = f(CV_{i-1}, Y_{i-1})$$
$$H(M) = CV_L$$



- IV = valore iniziale
- CV = variabile di concatenamento
- Y_i = i -esimo blocco di input
- f = algoritmo di compressione
- L = numero di blocchi di input
- n = uguale lunghezza del codice hash
- b = lunghezza del blocco di input

Funzioni hash sicure 2

Merkle e Damgard hanno dimostrato che se la funzione di compressione f è resistente alle collisioni allora lo è anche H (non vale il contrario).

Il problema della progettazione di funzioni hash sicure si è quindi ridotto alla progettazione di funzioni di compressione resistenti alle collisioni che operano su un input di dimensioni fisse.

L'analisi crittografica studia il funzionamento della funzione f per cercare di ottenere collisioni con un'unica esecuzione di f .

La tecnica più utilizzata è l'analisi differenziale che cerca di individuare come le modifiche sul messaggio influiscano sul codice hash.

Funzioni hash sicure 3

Le funzioni più note e diffuse sono:

Algoritmo	Lunghezza output	Lunghezza buffer	Lunghezza blocco	Lunghezza word	Sicura
MD4	128	128	512	32	NO
MD5	128	128	512	32	NO
RIPEMD	128	128	512	32	NO
RIPEMD 160	160	160	512	32	SI
SHA-0	160	160	512	32	NO
SHA-1	160	160	512	32	NO
SHA-2	224 - 512	256 - 512	512 - 1024	32 - 64	SI

Tutte queste funzioni si basano su MD4 perché utilizza un'architettura collaudata che è stata oggetto di molti studi.

MD5

La funzione MD5 venne progettata nel 1991 da Ronald Rivest per sostituire MD4.

Nel 1993 fu trovata una pseudo collisione, ottenibile utilizzando due diversi vettori di inizializzazione.

Nel 1996 fu trovata una collisione che rese insicuro MD5 per gli ambiti ad alta sicurezza.

Nel 2006 è stato pubblicato un algoritmo che riesce a trovare collisioni in meno di un minuto.

```
The first block collision took : 4.800000 sec 19.05.2007 14:25:19.937
```

```
Check: The same MD5 hash
```

```
The second block collision took : 4.900000 sec 19.05.2007 14:25:24.937
```

```
The first and the second blocks together took : 9.700000 sec
```

```
AVERAGE time for the 1st block = 4.800000 sec
```

```
AVERAGE time for the 2nd block = 4.900000 sec
```

```
AVERAGE time for the complete collision = 9.700000 sec
```

```
No. of collisions = 1
```

SHA-1

Secure Hash Algorithm è stato inizialmente pubblicato nel 1993.

Il progetto passò sotto la supervisione dell'NSA che nel 1996 rilasciò una versione corretta, SHA-1.

É molto simile a MD5 ma ha una maggiore resistenza agli attacchi a forza bruta grazie agli ulteriori 32 bit di codice hash.

Nonostante MD5 si sia rivelato vulnerabile all'analisi crittografica, SHA-1 non sembra vulnerabile a questo tipo di attacchi. Ciò è dovuto anche al fatto che NSA non abbia mai rilasciato informazioni sulle scelte progettuali.

La dimensione massima del messaggio in input è 2^{64} bit.

Le funzioni catalogate come SHA-2 sono coperte da brevetto.

É stato progettato per architetture Big-Endian.

RIPEMD-160

RIPEMD fu rilasciato nel 1992 nell'ambito del progetto europeo RIPE (RACE Integrity Primitives Evaluation) dalle stesse persone che avevano condotto attacchi efficaci contro MD4 e MD5.

Sostituito prima da RIPEMD-128, poi da RIPEMD-160 grazie ad Hans Dobbertin che individuò vulnerabilità caratteristiche di MD5 nella prima versione della funzione.

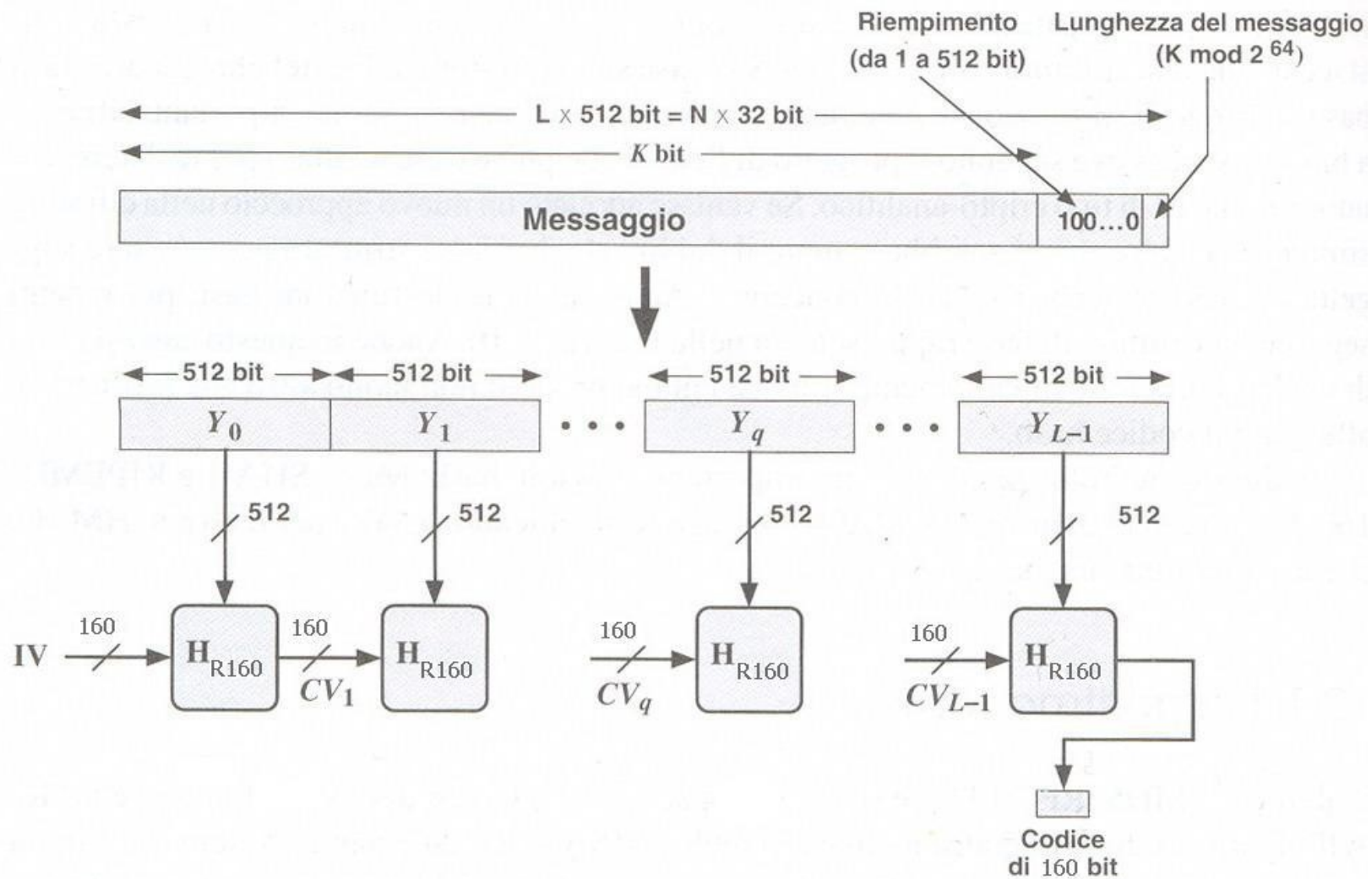
La dimensione massima del messaggio è 2^{64} bit.

É progettato per architetture Little-Endian ed offre prestazioni simili ad SHA-1.

É meno diffuso di SHA-1 / SHA-2 ma non è coperto da alcun brevetto.

FUNZIONAMENTO RIPEMD-160 1

Il messaggio con riempimento e lunghezza viene suddiviso in L blocchi ognuno dei quali è elaborato dalla funzione di compressione H_{R160} .



FUNZIONAMENTO RIPEMD-160 2

IV viene visto come cinque registri di 32 bit A, B, C, D, E che vengono inizializzati con i seguenti valori esadecimali:

A=67452301

B=EFCDAB89

C=98BADCFE

D=10325476

E=C3D2E1F0

Sono gli stessi valori utilizzati da SHA-1 e MD5 (eccetto l'ultimo).

La funzione di compressione utilizza cinque differenti funzioni logiche primitive ed è composta da due file parallele di cinque fasi. Ogni fase è composta da sedici passi.

Dopo che l'input è stato elaborato dalle dieci fasi viene sommato al vecchio valore di concatenamento per generare il nuovo

FUNZIONAMENTO RIPEMD-160 3

In figura si possono vedere gli 80 passi compiuti su entrambi i rami necessari a calcolare il nuovo CV.

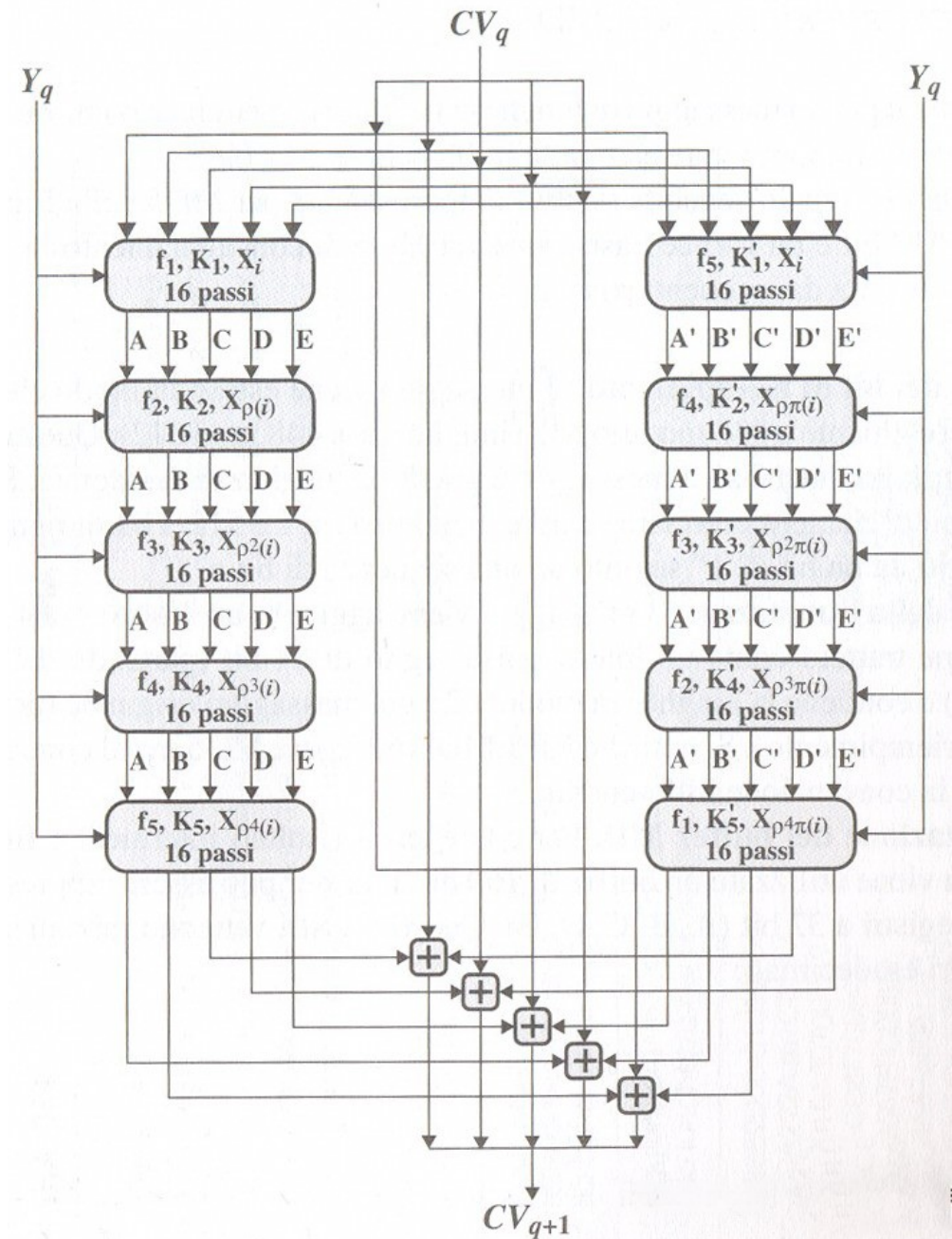
f_i = i-esima primitiva logica

K_i = K-esima costante additiva

K'_i = K'-esima costante additiva

$X_{r(i)}$ = parola di 32 bit prelevata dall'input
secondo la permutazione $r(j)$

+ = somma modulo 2^{32}



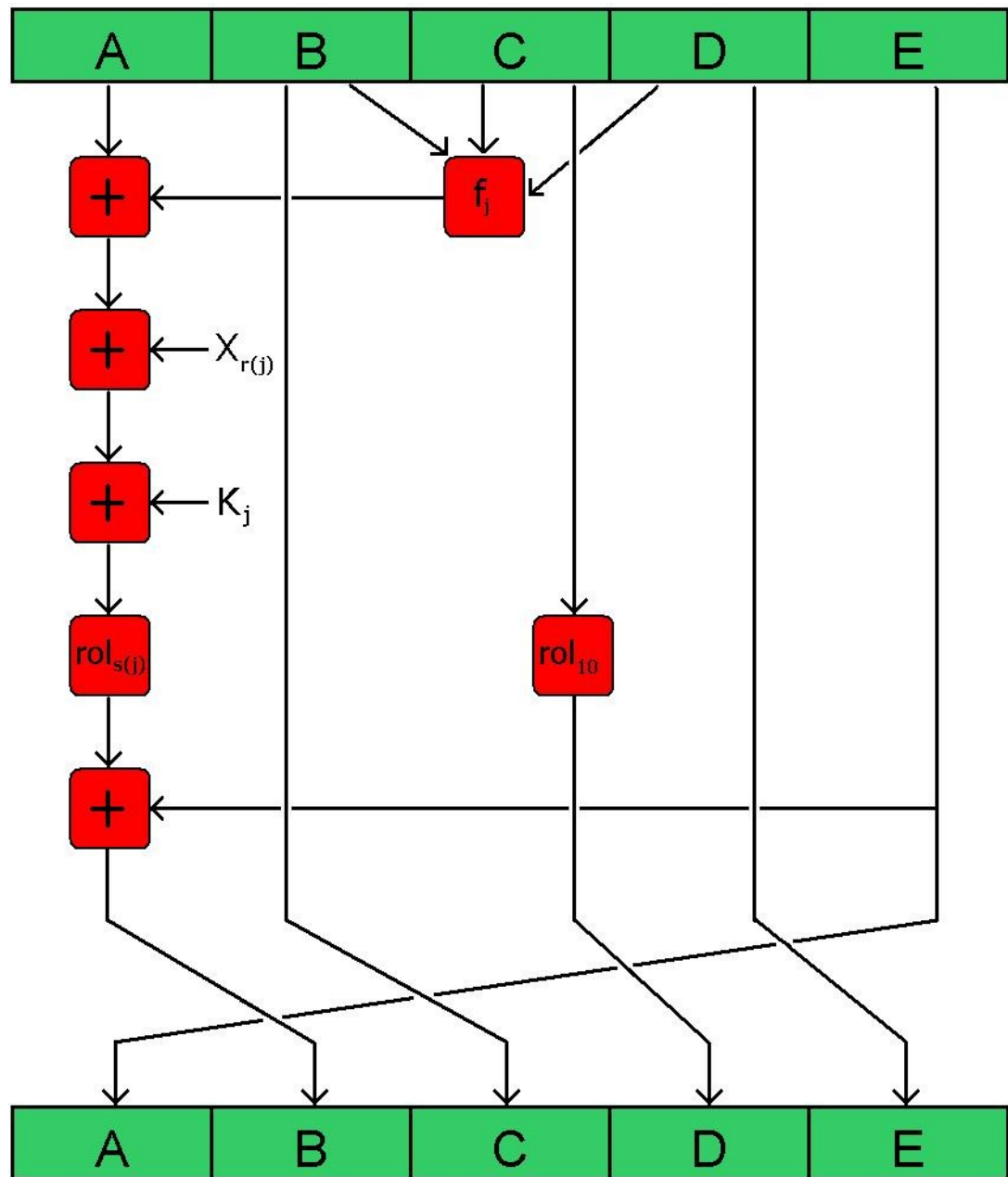
FUNZIONAMENTO RIPEMD-160 4

Numero passo	Metà sinistra		Metà destra	
	Esadecimale	Parte intera di	Esadecimale	Parte intera di
$0 \leq j \leq 15$	$K_1 = K(j) = 00000000$	0	$K'_1 = K'(j) = 50A28BE6$	$2^{30} \times \sqrt{2}$
$16 \leq j \leq 31$	$K_2 = K(j) = 5A827999$	$2^{30} \times \sqrt{2}$	$K'_2 = K'(j) = 5C4DD124$	$2^{30} \times \sqrt{3}$
$32 \leq j \leq 47$	$K_3 = K(j) = 6ED9EBA1$	$2^{30} \times \sqrt{3}$	$K'_3 = K'(j) = 6D703EF3$	$2^{30} \times \sqrt{5}$
$48 \leq j \leq 63$	$K_4 = K(j) = 8F1BBCDC$	$2^{30} \times \sqrt{5}$	$K'_4 = K'(j) = 7A6D76E9$	$2^{30} \times \sqrt{7}$
$64 \leq j \leq 79$	$K_5 = K(j) = A953FD4E$	$2^{30} \times \sqrt{7}$	$K'_5 = K'(j) = 00000000$	0

Passo	Nome funzione	Valore funzione
$0 \leq j \leq 15$	$f_1 = f(j, B, C, D)$	$B \oplus C \oplus D$
$16 \leq j \leq 31$	$f_2 = f(j, B, C, D)$	$(B \wedge C) \vee (\bar{B} \wedge D)$
$32 \leq j \leq 47$	$f_3 = f(j, B, C, D)$	$(B \vee \bar{C}) \oplus D$
$48 \leq j \leq 63$	$f_4 = f(j, B, C, D)$	$(B \wedge D) \vee (C \wedge \bar{D})$
$64 \leq j \leq 79$	$f_5 = f(j, B, C, D)$	$B \oplus (C \vee \bar{D})$

FUNZIONAMENTO RIPEMD-160 5

Questa è la struttura del passo j-esimo.



FUNZIONAMENTO RIPEMD-160 6

Le funzioni di permutazione dell'indice di X sono:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\rho(i)$	7	4	13	1	10	6	15	3	12	0	9	5	2	14	11	8
$\pi(i)$	5	14	7	0	9	2	11	4	13	6	15	8	1	10	3	12

Applicate nel seguente ordine e calcolate mod 16:

Ramo	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5
SX	Id	ρ	ρ^2	ρ^3	ρ^4
DX	π	$\rho\pi$	$\rho^2\pi$	$\rho^3\pi$	$\rho^4\pi$

Gli scorrimenti circolari a sinistra utilizzati nelle varie fasi sono:

Fase	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
1	11	14	15	12	5	8	7	9	11	13	14	15	6	7	9	8
2	12	13	11	15	6	9	9	7	12	15	11	13	7	8	7	7
3	13	15	14	11	7	7	6	8	13	14	13	12	5	5	6	9
4	14	11	12	14	8	6	5	5	14	12	15	14	9	9	8	6
5	15	12	13	13	9	5	8	6	15	11	12	11	8	6	5	5

SCELTE PROGETTUALI RIPEMD-160

- 1 - Sono stati utilizzati due rami separati per l'implementazione delle varie fasi, ciò rende più complessa la ricerca delle collisioni
- 2 - Le due linee utilizzano la stessa logica ma con il maggior numero di differenze possibili. Ciò dovrebbe garantire una maggiore resistenza all'analisi crittografica
- 3 - Lo scorrimento di 10 bit effettuato in ogni passo neutralizza un attacco per MD5
- 4 - Le permutazioni ρ e π hanno lo scopo di utilizzare i dati in ingresso in maniera differente tra le varie fasi
- 5 - Gli scorrimenti circolari sono stati scelti secondo i seguenti criteri
 - scorrimenti inferiori a 5 bit sono considerati troppo deboli
 - ogni parola X_i deve ruotare di valori differenti nelle cinque fasi
 - gli scorrimenti applicati a ciascuna parola non devono essere multipli di 32
 - non troppi scorrimenti devono essere divisibili per 4

RIFERIMENTI BIBLIOGRAFICI

- Crittografia e sicurezza delle reti - William Stallings - 2004
- Introduction to computer security - Matt Bishop - 2005
- http://en.wikipedia.org/wiki/Cryptographic_hash_function
- <http://en.wikipedia.org/wiki/MD5>
- http://en.wikipedia.org/wiki/SHA_hash_functions
- <http://en.wikipedia.org/wiki/RIPEMD>