Funzioni di Verifica

Gregorio D'Agostino

20 Aprile 2021

Controllo ciclico di ridondanza (Cyclic redundancy Check)

Collisioni di compleanni

Un esempio di funzione di verifica sicura

Controllo di integrità ciclico

- Supponiamo di aver inviato un blocco di dati cioè una sequenza di bit contenente la sequenza informativa e altri bit che dipendono dal protocollo.

Controllo di integrità ciclico

- Supponiamo di aver inviato un blocco di dati cioè una sequenza di bit contenente la sequenza informativa e altri bit che dipendono dal protocollo.
- La verifica ciclica di integrità consiste nell'allocare una parte di lunghezza prefissata del blocco (collocata tipicamente alla fine) al digest ottenuto tramite una predefinita hash function. La hash function è assolutamente nota a tutti, lo scopo è solo evitare gli errori, non difendersi dagli origliamenti (eavesdropping) o manipolazioni (tampering).

Controllo di integrità ciclico

- Supponiamo di aver inviato un blocco di dati cioè una sequenza di bit contenente la sequenza informativa e altri bit che dipendono dal protocollo.
- La verifica ciclica di integrità consiste nell'allocare una parte di lunghezza prefissata del blocco (collocata tipicamente alla fine) al digest ottenuto tramite una predefinita hash function. La hash function è assolutamente nota a tutti, lo scopo è solo evitare gli errori, non difendersi dagli origliamenti (eavesdropping) o manipolazioni (tampering).
- Supponiamo che la lunghezza del digest sia fissata in n bit collocati alla fine della sequenza trasmessa di lunghezza totale $L + n$. Dal punto di vista matematico la lunghezza della sequenza infomativa è L, mentre n è la componente di ridondanza. Sia m il numero binario corrispondente alla sequenza informativa composto da L bit $(b_1, b_2, \ldots, b_L)$:

$$m \stackrel{def}{=} \sum_{i=1,L} b_i \cdot 2^{L-i}.$$

Controllo di integrità ciclico -cont

- Inserire in coda alla sequenza codificante un digest di controllo equivale a moltiplicare per 2^n (traslare di n passi a sinistra la sequenza condificante) ed aggiungere un intero d (il digest) a n bit ($0 \leq d < 2^n$):

$$m' \stackrel{def}{=} m \cdot 2^n + d.$$

Controllo di integrità ciclico -cont

- Inserire in coda alla sequenza codificante un digest di controllo equivale a moltiplicare per 2^n (traslare di n passi a sinistra la sequenza condificante) ed aggiungere un intero d (il digest) a n bit ($0 \leq d < 2^n$):

$$m' \stackrel{def}{=} m \cdot 2^n + d.$$

- Per definire il digest bisogna specificare la funzione di hash.

$$d \stackrel{def}{=} f(m);$$

per costruzione d è minore di 2^n.

Esempio di controllo di parità a blocchi

- un pacchetto di 8 blocchi da 4 bit in cui l'ultimo blocco è di controllo: $A3179CFX$:

A	3	1	7	9	C	F	x
1010	0011	0001	0111	1001	1100	1111	x

$$
\begin{array}{lcc}
A & \oplus & \\
3 & \oplus & \\
1 & \oplus & \\
7 & \oplus & \text{cioè} \\
9 & \oplus & \\
C & \oplus & \\
F & = & \\
x & &
\end{array}
\qquad
\begin{array}{lc}
1010 & \oplus \\
0011 & \oplus \\
0001 & \oplus \\
0111 & \oplus \\
1001 & \oplus \\
1100 & \oplus \\
1111 & = \\
0101 &
\end{array}
$$

Esempio di controllo di parità a blocchi

- un pacchetto di 8 blocchi da 4 bit in cui l'ultimo blocco è di controllo: $A3179CFX$:

A	3	1	7	9	C	F	x
1010	0011	0001	0111	1001	1100	1111	x

$$
\begin{array}{lcc}
A & \oplus & \\
3 & \oplus & \\
1 & \oplus & \\
7 & \oplus & \text{cioè} \\
9 & \oplus & \\
C & \oplus & \\
F & = & \\
x & &
\end{array}
\qquad
\begin{array}{lc}
1010 & \oplus \\
0011 & \oplus \\
0001 & \oplus \\
0111 & \oplus \\
1001 & \oplus \\
1100 & \oplus \\
1111 & = \\
0101 &
\end{array}
$$

- X=5, quindi la sequenza completa è: $A3179CF5$

Esempio di controllo di parità ciclizzata

- un pacchetto di 8 blocchi da 4 bit in cui l'ultimo blocco è di controllo: $A3179CFX$
  [t trasforma una sequenza ciclicamente: t(wxyz)=zwxy]:

A	3	1	7	9	C	F	x
1010	0011	0001	0111	1001	1100	1111	x

$$
\begin{array}{rcc}
A & \oplus & \\
t(3) & \oplus & \\
t(t(1)) & \oplus & \\
t(t(t(7))) & \oplus & \\
t(t(t(t(9)))) & \oplus & \\
t(t(t(t(t(C))))) & \oplus & \\
t(t(t(t(t(t(F)))))) & = & \\
X & &
\end{array}
\quad \text{cioè} \quad
\begin{array}{rcc}
1010 & \oplus & \\
1 \to\ 0011 & \oplus & \\
2 \to\ 0001 & \oplus & \\
3 \to\ 0111 & \oplus & \\
4 \to\ 1001 & \oplus & \\
5 \to\ 1100 & \oplus & \\
6 \to\ 1111 & = & \\
X & &
\end{array}
\quad \text{cioè} \quad
\begin{array}{rcc}
1010 & \oplus & \\
1001 & \oplus & \\
0100 & \oplus & \\
1110 & \oplus & \\
1001 & \oplus & \\
0110 & \oplus & \\
1111 & = & \\
1001 & &
\end{array}
$$

Esempio di controllo di parità ciclizzata

- un pacchetto di 8 blocchi da 4 bit in cui l'ultimo blocco è di controllo: $A3179CFX$
  [t trasforma una sequenza ciclicamente: t(wxyz)=zwxy]:

A	3	1	7	9	C	F	x
1010	0011	0001	0111	1001	1100	1111	x

$$
\begin{array}{rcc}
A & \oplus & \\
t(3) & \oplus & \\
t(t(1)) & \oplus & \\
t(t(t(7))) & \oplus & \text{cioè} \\
t(t(t(t(9)))) & \oplus & \\
t(t(t(t(t(C))))) & \oplus & \\
t(t(t(t(t(t(F)))))) & = & \\
X & &
\end{array}
\qquad
\begin{array}{rcc}
1010 & \oplus & \\
1 \rightarrow 0011 & \oplus & \\
2 \rightarrow 0001 & \oplus & \\
3 \rightarrow 0111 & \oplus & \text{cioè} \\
4 \rightarrow 1001 & \oplus & \\
5 \rightarrow 1100 & \oplus & \\
6 \rightarrow 1111 & = & \\
X & &
\end{array}
\qquad
\begin{array}{rcc}
1010 & \oplus & \\
1001 & \oplus & \\
0100 & \oplus & \\
1110 & \oplus & \\
1001 & \oplus & \\
0110 & \oplus & \\
1111 & = & \\
1001 & &
\end{array}
$$

- X=9, quindi la sequenza completa è: $A3179CF9$

Anelli di polinomi

- Dato un campo $\mathbb{K}$ (ad esempio $\mathbb{Z}_2, \mathbb{Z}_p, \mathbb{Z}etc$) Si costruisce un anello di polinomi imponendo oltre agli elementi di $\mathbb{K}$ allo spazio appartenga un elemento indicato formalmente con x e che lo spazio sia chiuso rispetto alla somma ed al prodotto e siano rispettate le proprietà associativa e distributiva.

Anelli di polinomi

- Dato un campo $\mathbb{K}$ (ad esempio $\mathbb{Z}_2, \mathbb{Z}_p, \mathbb{Z}etc$) Si costruisce un anello di polinomi imponendo oltre agli elementi di $\mathbb{K}$ allo spazio appartenga un elemento indicato formalmente con x e che lo spazio sia chiuso rispetto alla somma ed al prodotto e siano rispettate le proprietà associativa e distributiva.
  - Esiste x;

Anelli di polinomi

- Dato un campo $\mathbb{K}$ (ad esempio $\mathbb{Z}_2, \mathbb{Z}_p, \mathbb{Z}etc$) Si costruisce un anello di polinomi imponendo oltre agli elementi di $\mathbb{K}$ allo spazio appartenga un elemento indicato formalmente con x e che lo spazio sia chiuso rispetto alla somma ed al prodotto e siano rispettate le proprietà associativa e distributiva.
  - Esiste x;
  - Il prodotto è chiuso quindi esistono $a \cdot x$ ed $a + x$ con $a \in \mathbb{K}$;

Anelli di polinomi

- Dato un campo $\mathbb{K}$ (ad esempio $\mathbb{Z}_2, \mathbb{Z}_p, \mathbb{Z}etc$) Si costruisce un anello di polinomi imponendo oltre agli elementi di $\mathbb{K}$ allo spazio appartenga un elemento indicato formalmente con x e che lo spazio sia chiuso rispetto alla somma ed al prodotto e siano rispettate le proprietà associativa e distributiva.
  - Esiste x;
  - Il prodotto è chiuso quindi esistono $a \cdot x$ ed $a + x$ con $a \in \mathbb{K}$;
  - Il prodotto è chiuso, quindi esiste $x \cdot x \stackrel{def}{=} x^2$; da questo si ricavano (per induzione) tutte le potenze:

$$x^k \cdot x \stackrel{def}{=} x^{k+1} :$$

Anelli di polinomi

- Dato un campo $\mathbb{K}$ (ad esempio $\mathbb{Z}_2, \mathbb{Z}_p, \mathbb{Z}etc$) Si costruisce un anello di polinomi imponendo oltre agli elementi di $\mathbb{K}$ allo spazio appartenga un elemento indicato formalmente con x e che lo spazio sia chiuso rispetto alla somma ed al prodotto e siano rispettate le proprietà associativa e distributiva.
  - Esiste x;
  - Il prodotto è chiuso quindi esistono $a \cdot x$ ed $a + x$ con $a \in \mathbb{K}$;
  - Il prodotto è chiuso, quindi esiste $x \cdot x \stackrel{def}{=} x^2$; da questo si ricavano (per induzione) tutte le potenze:

$$x^k \cdot x \stackrel{def}{=} x^{k+1} :$$

  - Il prodotto è associativo quindi:

$$c \cdot a \cdot x \stackrel{def}{=} (c \cdot a) \cdot x;$$

Anelli di polinomi

- In tal modo si generano tutti le espressioni formali (gli elementi dell'anello) detti polinomi del tipo

$$a_0 + a_1 \cdot x + a_2 \cdot x^2 \ldots;$$

in cui $a_0, a_1, \ldots \in \mathbb{K}$.

Anelli di polinomi

- In tal modo si generano tutti le espressioni formali (gli elementi dell'anello) detti polinomi del tipo

$$a_0 + a_1 \cdot x + a_2 \cdot x^2 \ldots;$$

in cui $a_0, a_1, \ldots \in \mathbb{K}$.

- Un monomio è una espressione formale del tipo ax^k in cui $a \in \mathbb{K}$ e $k \in \mathbb{Z}$.

Anelli di polinomi

- In tal modo si generano tutti le espressioni formali (gli elementi dell'anello) detti polinomi del tipo

$$a_0 + a_1 \cdot x + a_2 \cdot x^2 \ldots;$$

in cui $a_0, a_1, \ldots \in \mathbb{K}$.

- Un monomio è una espressione formale del tipo ax^k in cui $a \in \mathbb{K}$ e $k \in \mathbb{Z}$.
- Il grado di un polinomio è la potenza massima di x.

Anelli di polinomi

- In tal modo si generano tutti le espressioni formali (gli elementi dell'anello) detti polinomi del tipo

$$a_0 + a_1 \cdot x + a_2 \cdot x^2 \ldots;$$

in cui $a_0, a_1, \ldots \in \mathbb{K}$.

- Un monomio è una espressione formale del tipo ax^k in cui $a \in \mathbb{K}$ e $k \in \mathbb{Z}$.
- Il grado di un polinomio è la potenza massima di x.
- Un binomio di primo grado è quindi: $ax + b$ in cui $a, b \in \mathbb{K}$.

Anelli di polinomi

- In tal modo si generano tutti le espressioni formali (gli elementi dell'anello) detti polinomi del tipo

$$a_0 + a_1 \cdot x + a_2 \cdot x^2 \ldots;$$

in cui $a_0, a_1, \ldots \in \mathbb{K}$.

- Un monomio è una espressione formale del tipo ax^k in cui $a \in \mathbb{K}$ e $k \in \mathbb{Z}$.
- Il grado di un polinomio è la potenza massima di x.
- Un binomio di primo grado è quindi: $ax + b$ in cui $a, b \in \mathbb{K}$.
- Un polinomio si dice monico quando il coefficiente del suo grado più alto è unitario.
- Un polinomio definisce una funzione sul campo $\mathbb{K}$ ottenuta sostituendo ad x un elemento di $\mathbb{K}$. La funzione é il polinomio calcolato sul dominio $f : \mathbb{K} \to \mathbb{K}$:

$$f : x \to p(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 \ldots.$$

Anelli di polinomi -cont

- Verifichiamo le proprietà che caratterizzano gli anelli:

Anelli di polinomi -cont

- Verifichiamo le proprietà che caratterizzano gli anelli:
- Associativa somma:

$$\mathcal{A} + (\mathcal{B} + \mathcal{C}) = (\mathcal{A} + \mathcal{B}) + \mathcal{C};$$

Anelli di polinomi -cont

- Verifichiamo le proprietà che caratterizzano gli anelli:
- Associativa somma:

$$\mathcal{A} + (\mathcal{B} + \mathcal{C}) = (\mathcal{A} + \mathcal{B}) + \mathcal{C};$$

- Elemento neutro somma:

$$\mathcal{A} + 0 = 0 + \mathcal{A} = \mathcal{A};$$

Anelli di polinomi -cont

- Verifichiamo le proprietà che caratterizzano gli anelli:
- Associativa somma:

$$\mathcal{A} + (\mathcal{B} + \mathcal{C}) = (\mathcal{A} + \mathcal{B}) + \mathcal{C};$$

- Elemento neutro somma:

$$\mathcal{A} + 0 = 0 + \mathcal{A} = \mathcal{A};$$

- Elemento opposto somma:

$$\forall \mathcal{A} = a_0 a_1 \cdot x + a_2 \cdot x^2 \ldots + : \exists - \mathcal{A} \stackrel{def}{=} -a_0 - a_1 \cdot x - a_2 \cdot x^2 \ldots;$$

Anelli di polinomi -cont

- Verifichiamo le proprietà che caratterizzano gli anelli:
- Associativa somma:

$$\mathcal{A} + (\mathcal{B} + \mathcal{C}) = (\mathcal{A} + \mathcal{B}) + \mathcal{C};$$

- Elemento neutro somma:

$$\mathcal{A} + 0 = 0 + \mathcal{A} = \mathcal{A};$$

- Elemento opposto somma:

$$\forall \mathcal{A} = a_0 a_1 \cdot x + a_2 \cdot x^2 \ldots + : \exists - \mathcal{A} \stackrel{def}{=} -a_0 - a_1 \cdot x - a_2 \cdot x^2 \ldots;$$

- Associativa prodotto

$$\mathcal{A}(\mathcal{B}\mathcal{C}) = (\mathcal{A}\mathcal{B})\mathcal{C};$$

Anelli di polinomi -cont

- Verifichiamo le proprietà. Distribuitiva:

$$\mathcal{A}(\mathcal{B}+\mathcal{C}) = \mathcal{A}\mathcal{B}+\mathcal{A}\mathcal{C};$$

Anelli di polinomi -cont

- Verifichiamo le proprietà. Distribuitiva:

$$\mathcal{A}(\mathcal{B}+\mathcal{C}) = \mathcal{A}\mathcal{B}+\mathcal{A}\mathcal{C};$$

- La proprietà distribuitiva ci fornisce una regola per sommare i monomi pari grado:

$$ax^k + bx^k = (a+b)x^k;$$

- Analogamente per i binomi:

$$ax + b + (cx + d) = (ax + cx) + (b + d) = (a + c)x + (b + d);$$

$a + c$ e $b + d$ si eseguono in $\mathbb{K}$.

Anelli di polinomi -cont

- Verifichiamo le proprietà. Distribuitiva:

$$\mathcal{A}(\mathcal{B}+\mathcal{C}) = \mathcal{A}\mathcal{B}+\mathcal{A}\mathcal{C};$$

- La proprietà distribuitiva ci fornisce una regola per sommare i monomi pari grado:

$$ax^k + bx^k = (a+b)x^k;$$

- Analogamente per i binomi:

$$ax+b+(cx+d) = (ax+cx)+(b+d) = (a+c)x+(b+d);$$

$a+c$ e $b+d$ si eseguono in $\mathbb{K}$.

- In generale per i polinomi si sommano i termini di stesso grado (esponente) in x.

Anelli di polinomi -cont

- Some detto il grado di un polinomio è la massima potenza a cui appare la x nella sua espressione formale.

Anelli di polinomi -cont

- Some detto il grado di un polinomio è la massima potenza a cui appare la x nella sua espressione formale.
- Analogamente all'anello dei numeri naturali, possiamo definire una divisione. Dato un polinomio $\mathcal{A}$ dividendo ed un polinomio $\mathcal{B}$ divisore eseguire la divisione significa trovare due polinomi $\mathcal{Q}$ ed $\mathcal{R}$ tali che:

$$\mathcal{A} = \mathcal{Q} \cdot \mathcal{B} + \mathcal{R};$$

in cui il grado di $\mathcal{R}$ sia minore di quello di $\mathcal{B}$.

Anelli di polinomi -cont

- Some detto il grado di un polinomio è la massima potenza a cui appare la x nella sua espressione formale.
- Analogamente all'anello dei numeri naturali, possiamo definire una divisione. Dato un polinomio $\mathcal{A}$ dividendo ed un polinomio $\mathcal{B}$ divisore eseguire la divisione significa trovare due polinomi $\mathcal{Q}$ ed $\mathcal{R}$ tali che:

$$\mathcal{A} = \mathcal{Q} \cdot \mathcal{B} + \mathcal{R};$$

in cui il grado di $\mathcal{R}$ sia minore di quello di $\mathcal{B}$.

Anelli di polinomi -cont

- Some detto il grado di un polinomio è la massima potenza a cui appare la x nella sua espressione formale.
- Analogamente all'anello dei numeri naturali, possiamo definire una divisione. Dato un polinomio $\mathcal{A}$ dividendo ed un polinomio $\mathcal{B}$ divisore eseguire la divisione significa trovare due polinomi $\mathcal{Q}$ ed $\mathcal{R}$ tali che:

$$\mathcal{A} = \mathcal{Q} \cdot \mathcal{B} + \mathcal{R};$$

  in cui il grado di $\mathcal{R}$ sia minore di quello di $\mathcal{B}$.
- Il polinomio $\mathcal{Q}$ si dice polinomio quoziente (o quoziente). Il polinomio $\mathcal{R}$ si dice polinomio resto (o resto).

Anelli di polinomi -cont

- Some detto il grado di un polinomio è la massima potenza a cui appare la x nella sua espressione formale.
- Analogamente all'anello dei numeri naturali, possiamo definire una divisione. Dato un polinomio $\mathcal{A}$ dividendo ed un polinomio $\mathcal{B}$ divisore eseguire la divisione significa trovare due polinomi $\mathcal{Q}$ ed $\mathcal{R}$ tali che:

$$\mathcal{A} = \mathcal{Q} \cdot \mathcal{B} + \mathcal{R};$$

  in cui il grado di $\mathcal{R}$ sia minore di quello di $\mathcal{B}$.
- Il polinomio $\mathcal{Q}$ si dice polinomio quoziente (o quoziente). Il polinomio $\mathcal{R}$ si dice polinomio resto (o resto).
- Nel caso dei polinomi la divisione si può ottenere algoritmicamente in modo analogo a quella aritmetica.

Anelli di polinomi -cont

- Some detto il grado di un polinomio è la massima potenza a cui appare la x nella sua espressione formale.
- Analogamente all'anello dei numeri naturali, possiamo definire una divisione. Dato un polinomio $\mathcal{A}$ dividendo ed un polinomio $\mathcal{B}$ divisore eseguire la divisione significa trovare due polinomi $\mathcal{Q}$ ed $\mathcal{R}$ tali che:

$$\mathcal{A} = \mathcal{Q} \cdot \mathcal{B} + \mathcal{R};$$

  in cui il grado di $\mathcal{R}$ sia minore di quello di $\mathcal{B}$.
- Il polinomio $\mathcal{Q}$ si dice polinomio quoziente (o quoziente). Il polinomio $\mathcal{R}$ si dice polinomio resto (o resto).
- Nel caso dei polinomi la divisione si può ottenere algoritmicamente in modo analogo a quella aritmetica.
- Quando il resto è nullo il polinomio $\mathcal{B}$ e anche per consenguenza $\mathcal{Q}$ si dicono "divisori" di $\mathcal{A}$.

Algoritmo per la divisione tra polinomi

- Se il grado n di $\mathcal{A}$ è minore di quello m di $\mathcal{B}$ allora $\mathcal{Q} = 0$ e $\mathcal{R} = \mathcal{A}$; infatti: $0 \cdot \mathcal{B} + \mathcal{R} = \mathcal{A}$.

Algoritmo per la divisione tra polinomi

- Se il grado n di $\mathcal{A}$ è minore di quello m di $\mathcal{B}$ allora $\mathcal{Q} = 0$ e $\mathcal{R} = \mathcal{A}$; infatti: $0 \cdot \mathcal{B} + \mathcal{R} = \mathcal{A}$.
- Se il grado n di $\mathcal{A}$ è maggiore di quello m di $\mathcal{B}$ il polinomio quoziente $\mathcal{Q}$ si costruisce iterativamente partendo dal grado massimo:

Algoritmo per la divisione tra polinomi

- Se il grado n di $\mathcal{A}$ è minore di quello m di $\mathcal{B}$ allora $\mathcal{Q} = 0$ e $\mathcal{R} = \mathcal{A}$; infatti: $0 \cdot \mathcal{B} + \mathcal{R} = \mathcal{A}$.
- Se il grado n di $\mathcal{A}$ è maggiore di quello m di $\mathcal{B}$ il polinomio quoziente $\mathcal{Q}$ si costruisce iterativamente partendo dal grado massimo:
- Se $\mathcal{A} = a_0 + a_1 x + a_2 x^2 \cdots a_n x^n$ e $\mathcal{B} = b_0 + b_1 x + b_2 x^2 \cdots b_m x^m$ Il termine di grado massimo della divisione sarà :

$$\mathcal{Q} = x^{n-m} a_n / b_m + \mathcal{Q}^1;$$

in cui $\mathcal{Q}^1$ è di grado inferiore a $\mathcal{Q}$ cioè $n - m$.

Algoritmo per la divisione tra polinomi

- Se il grado n di $\mathcal{A}$ è minore di quello m di $\mathcal{B}$ allora $\mathcal{Q} = 0$ e $\mathcal{R} = \mathcal{A}$; infatti: $0 \cdot \mathcal{B} + \mathcal{R} = \mathcal{A}$.
- Se il grado n di $\mathcal{A}$ è maggiore di quello m di $\mathcal{B}$ il polinomio quoziente $\mathcal{Q}$ si costruisce iterativamente partendo dal grado massimo:
- Se $\mathcal{A} = a_0 + a_1 x + a_2 x^2 \cdots a_n x^n$ e $\mathcal{B} = b_0 + b_1 x + b_2 x^2 \cdots b_m x^m$ Il termine di grado massimo della divisione sarà :

$$\mathcal{Q} = x^{n-m} a_n / b_m + \mathcal{Q}^1;$$

  in cui $\mathcal{Q}^1$ è di grado inferiore a $\mathcal{Q}$ cioè $n - m$.
- I termini successivi (i termini di grado massimo di $\mathcal{Q}^k$) si trovano sottraendo $(x^{n-m} a_n / b_m)\mathcal{B}$ da $\mathcal{A}$

$$\mathcal{A}^1 = \mathcal{A} - (x^{n-m} a_n / b_m)\mathcal{B}.$$

Algoritmo per la divisione tra polinomi

- Se il grado n di $\mathcal{A}$ è minore di quello m di $\mathcal{B}$ allora $\mathcal{Q} = 0$ e $\mathcal{R} = \mathcal{A}$; infatti: $0 \cdot \mathcal{B} + \mathcal{R} = \mathcal{A}$.
- Se il grado n di $\mathcal{A}$ è maggiore di quello m di $\mathcal{B}$ il polinomio quoziente $\mathcal{Q}$ si costruisce iterativamente partendo dal grado massimo:
- Se $\mathcal{A} = a_0 + a_1 x + a_2 x^2 \cdots a_n x^n$ e $\mathcal{B} = b_0 + b_1 x + b_2 x^2 \cdots b_m x^m$ Il termine di grado massimo della divisione sarà :

$$\mathcal{Q} = x^{n-m} a_n / b_m + \mathcal{Q}^1;$$

  in cui $\mathcal{Q}^1$ è di grado inferiore a $\mathcal{Q}$ cioè $n - m$.
- I termini successivi (i termini di grado massimo di $\mathcal{Q}^k$) si trovano sottraendo $(x^{n-m} a_n / b_m)\mathcal{B}$ da $\mathcal{A}$

$$\mathcal{A}^1 = \mathcal{A} - (x^{n-m} a_n / b_m)\mathcal{B}.$$

- La procedura si arresta dopo $n - m$ passi perché si ottiene un polinomio di grado inferiore ad m.

La funzione di hash del controllo ciclico di integrità

- La funzione f si costruisce associando un polinomio su $\mathbb{Z}_2$ alla sequenza da trasmettere di lunghezza L:

$$m \rightarrow \mathcal{M} \stackrel{def}{=} b_0 x^{L-1} + b_1 x^{L-2} + b_2 x^{L-3} + \cdot b_{L-1}.$$

La funzione di hash del controllo ciclico di integrità

- La funzione f si costruisce associando un polinomio su $\mathbb{Z}_2$ alla sequenza da trasmettere di lunghezza L:

$$m \to \mathcal{M} \stackrel{def}{=} b_0 x^{L-1} + b_1 x^{L-2} + b_2 x^{L-3} + \cdot b_{L-1}.$$

- Si definisce un "polinomio generatore" (nulla a che vedere con i generatori dei gruppi) di grado n:

$$g \stackrel{def}{=} g_0 + g_1 x + g_2 x^2 + \cdots + g_{n-1} x^{n-1} + x^n;$$

si noti che g è monico cioè il coefficiente del suo grado massimo è unitario.

La funzione di hash del controllo ciclico di integrità

- La funzione f si costruisce associando un polinomio su $\mathbb{Z}_2$ alla sequenza da trasmettere di lunghezza L:

$$m \rightarrow \mathcal{M} \stackrel{def}{=} b_0 x^{L-1} + b_1 x^{L-2} + b_2 x^{L-3} + \cdot b_{L-1}.$$

- Si definisce un "polinomio generatore" (nulla a che vedere con i generatori dei gruppi) di grado n:

$$g \stackrel{def}{=} g_0 + g_1 x + g_2 x^2 + \cdots + g_{n-1} x^{n-1} + x^n;$$

  si noti che g è monico cioè il coefficiente del suo grado massimo è unitario.
- Si calcola il resto della divisione di $\mathcal{M} x^n$ per g

$$\mathcal{R} \stackrel{def}{=} mod(\mathcal{M} x^n, g).$$

La funzione di hash del controllo ciclico di integrità

- La funzione f si costruisce associando un polinomio su $\mathbb{Z}_2$ alla sequenza da trasmettere di lunghezza L:

$$m \rightarrow \mathcal{M} \stackrel{def}{=} b_0 x^{L-1} + b_1 x^{L-2} + b_2 x^{L-3} + \cdot b_{L-1}.$$

- Si definisce un "polinomio generatore" (nulla a che vedere con i generatori dei gruppi) di grado n:

$$g \stackrel{def}{=} g_0 + g_1 x + g_2 x^2 + \cdots + g_{n-1} x^{n-1} + x^n;$$

si noti che g è monico cioè il coefficiente del suo grado massimo è unitario.

- Si calcola il resto della divisione di $\mathcal{M} x^n$ per g

$$\mathcal{R} \stackrel{def}{=} mod(\mathcal{M} x^n, g).$$

- Si converte $\mathcal{R}$ nella sequenza binaria corrispondente. Si noti che essendo il resto della divisione è un polinomio di grado $n-1$, ad esso corrisponde una sequenza di lunghezza n.

$$\mathcal{R} = r_o + r_1 x + r_2 x^2 + \cdots + r_{n_1} x^{n-1} \rightarrow r_{n-1} r_{n-2} \ldots r_2 r_1 r_0.$$

La funzione di hash del controllo ciclico di integrità

- La funzione descritta è una buona hash function perché restituisce sempre una sequenza binaria di n bit.

La funzione di hash del controllo ciclico di integrità

- La funzione descritta è una buona hash function perché restituisce sempre una sequenza binaria di n bit.
- E' interessante notare che sul campo $\mathbb{Z}_2$ ogni polinomio è uguale al suo opposto (in particolare $\mathcal{R} + \mathcal{R} = 0$) e quindi:

$$\mathcal{A}x^n + \mathcal{R} = (\mathcal{Q}g + \mathcal{R}) + \mathcal{R} = \mathcal{Q}g.$$

La funzione di hash del controllo ciclico di integrità

- La funzione descritta è una buona hash function perché restituisce sempre una sequenza binaria di n bit.
- E' interessante notare che sul campo $\mathbb{Z}_2$ ogni polinomio è uguale al suo opposto (in particolare $\mathcal{R} + \mathcal{R} = 0$) e quindi:

$$\mathcal{A}x^n + \mathcal{R} = (\mathcal{Q}g + \mathcal{R}) + \mathcal{R} = \mathcal{Q}g.$$

- Il polinomio generatore g divide esattamente il polinomio associato alla sequenza completa m' da spedire.

$$m' \to \mathcal{M} = \mathcal{A}x^n + \mathcal{R} = \mathcal{Q}g.$$

Controllo di integrità ciclico

- La sequenza così ottenuta ($m' = m \cdot 2^n + d$) possiede proprietà molto utili:

Controllo di integrità ciclico

- La sequenza così ottenuta ($m' = m \cdot 2^n + d$) possiede proprietà molto utili:
- La sua lunghezza è esattamente $L + n$, quindi la ridondanza richiesta è $\frac{n}{l+n}$. Se facciamo pacchetti da 1kB e controlli da $n = 64$ bit la ridondanza è meno dell'un percento: $\frac{64}{8192+64} \sim 0.77\%$.

Controllo di integrità ciclico

- La sequenza così ottenuta ($m' = m \cdot 2^n + d$) possiede proprietà molto utili:
- La sua lunghezza è esattamente $L + n$, quindi la ridondanza richiesta è $\frac{n}{l+n}$. Se facciamo pacchetti da 1kB e controlli da $n = 64$ bit la ridondanza è meno dell'un percento: $\frac{64}{8192+64} \sim 0.77\%$.
- La condizione che consente di "verificare" l'integrità è il calcolo del resto rispetto al polinomio g sull'intera sequenza m'.

Controllo di integrità ciclico

- La sequenza così ottenuta ($m' = m \cdot 2^n + d$) possiede proprietà molto utili:
- La sua lunghezza è esattamente $L + n$, quindi la ridondanza richiesta è $\frac{n}{l+n}$. Se facciamo pacchetti da 1kB e controlli da $n = 64$ bit la ridondanza è meno dell'un percento: $\frac{64}{8192+64} \sim 0.77\%$.
- La condizione che consente di "verificare" l'integrità è il calcolo del resto rispetto al polinomio g sull'intera sequenza m'.
- Il processo è iterativo e richiede solo la memoria di interi da n bit: si inizia con la sequenza dei primi n bit di m' e si sottrae (o somma che è lo stesso) la sequenza che definisce il polinomio generatore $g_{n-1}g_{n-2}\ldots g_0$ bit a bit in $\mathbb{Z}_2$. Il risultato si scrive in memoria e si prosegue aggiungendo il bit che si ottiene traslando (shift di uno a dx) la sequenza m'. Al termine della procedura otterremo una sequenza di n zeri.

Controllo di integrità ciclico

- La sequenza così ottenuta ($m' = m \cdot 2^n + d$) possiede proprietà molto utili:
- La sua lunghezza è esattamente $L + n$, quindi la ridondanza richiesta è $\frac{n}{l+n}$. Se facciamo pacchetti da 1kB e controlli da $n = 64$ bit la ridondanza è meno dell'un percento: $\frac{64}{8192+64} \sim 0.77\%$.
- La condizione che consente di "verificare" l'integrità è il calcolo del resto rispetto al polinomio g sull'intera sequenza m'.
- Il processo è iterativo e richiede solo la memoria di interi da n bit: si inizia con la sequenza dei primi n bit di m' e si sottrae (o somma che è lo stesso) la sequenza che definisce il polinomio generatore $g_{n-1}g_{n-2}\ldots g_0$ bit a bit in $\mathbb{Z}_2$. Il risultato si scrive in memoria e si prosegue aggiungendo il bit che si ottiene traslando (shift di uno a dx) la sequenza m'. Al termine della procedura otterremo una sequenza di n zeri.
- Se la verifica è positiva si prendono i primi L bit (informativi) e si eliminano gli ultimi n bit (ridondanti) dalla sequenza

Sicurezza del Controllo ciclico di integrità

- La ridondanza ciclica strutturata non difende da attacchi deliberati, perché aggiungendo al messaggio un multiplo del polinomio generatore g ($g\mathcal{B}$), la sequenza risultante passa il vaglio del controllo:

$$\mathcal{A}'x^n + \mathcal{R} = (\mathcal{A} + g\mathcal{B})x^n + \mathcal{R} = (\mathcal{Q} + \mathcal{B}x^n)g + \mathcal{R} + \mathcal{R} = \mathcal{Q}'g.$$

quindi il metodo non è resistente rispetto alla ricerca di una seconda immagine. Abbiamo visto che si dice anche che non soddisfa i requisiti per la sicurezza debole.

Sicurezza del Controllo ciclico di integrità

- La ridondanza ciclica strutturata non difende da attacchi deliberati, perché aggiungendo al messaggio un multiplo del polinomio generatore g ($g\mathcal{B}$), la sequenza risultante passa il vaglio del controllo:

$$\mathcal{A}'x^n + \mathcal{R} = (\mathcal{A} + g\mathcal{B})x^n + \mathcal{R} = (\mathcal{Q} + \mathcal{B}x^n)g + \mathcal{R} + \mathcal{R} = \mathcal{Q}'g.$$

quindi il metodo non è resistente rispetto alla ricerca di una seconda immagine. Abbiamo visto che si dice anche che non soddisfa i requisiti per la sicurezza debole.

Sicurezza del Controllo ciclico di integrità

- La ridondanza ciclica strutturata non difende da attacchi deliberati, perché aggiungendo al messaggio un multiplo del polinomio generatore g ($g\mathcal{B}$), la sequenza risultante passa il vaglio del controllo:

  $$\mathcal{A}'x^n + \mathcal{R} = (\mathcal{A} + g\mathcal{B})x^n + \mathcal{R} = (\mathcal{Q} + \mathcal{B}x^n)g + \mathcal{R} + \mathcal{R} = \mathcal{Q}'g.$$

  quindi il metodo non è resistente rispetto alla ricerca di una seconda immagine. Abbiamo visto che si dice anche che non soddisfa i requisiti per la sicurezza debole.
- Ovviamente non può nemmeno soddisfare i requisiti per la sicurezza forte.

Sicurezza del Controllo ciclico di integrità

- La ridondanza ciclica strutturata non difende da attacchi deliberati, perché aggiungendo al messaggio un multiplo del polinomio generatore g ($g\mathcal{B}$), la sequenza risultante passa il vaglio del controllo:

  $$\mathcal{A}'x^n + \mathcal{R} = (\mathcal{A} + g\mathcal{B})x^n + \mathcal{R} = (\mathcal{Q} + \mathcal{B}x^n)g + \mathcal{R} + \mathcal{R} = \mathcal{Q}'g.$$

  quindi il metodo non è resistente rispetto alla ricerca di una seconda immagine. Abbiamo visto che si dice anche che non soddisfa i requisiti per la sicurezza debole.
- Ovviamente non può nemmeno soddisfare i requisiti per la sicurezza forte.
- Il CRC controllo ciclico di ridondanza non è robusto; infatti è facile trovare una contro-immagine di qualsiasi digest $\mathcal{R}$; scegliendo un qualsiasi $\mathcal{B}$:

  $$\mathcal{A}' = g\mathcal{B} + \mathcal{R}.$$

Scommesse sui compleanni

- In una class di 23 studenti, qual 'è la probabilità che due compiano gli anni lo stesso giorno?

Scommesse sui compleanni

- In una class di 23 studenti, qual 'è la probabilità che due compiano gli anni lo stesso giorno?
- Questo problema si chiama impropriamente paradosso del compleanno e fu introdotto da Richard von Mises nel 1939. La scommessa è favorevole 51% contro 49 %. Vediamo perché. Se p ù la probabilità di non collisione ovvero la probabilità che gli studenti siano anti tutti in date diverse ed n il numero degli studenti:

$$p = 1 \cdot \frac{364}{365} \cdot \frac{363}{365} \cdots \frac{365-(n-1)}{365} = \frac{365!}{(365-n+1)! \cdot 365^{n-1}}.$$

Stima delle probabilità di non collisione

- In generale dovremo scegliere n elementi da un gruppo di N senza collisioni. Sviluppiamo la formula generale rispetto ad $1/N$.

$$p = 1 \cdot \left(1 - \frac{1}{N}\right) \cdot \left(1 - \frac{2}{N}\right) \cdots \left(1 - \frac{n-1}{N}\right) \approx 1 - \frac{1}{N} \left(\sum_{k=0}^{n-1} k\right)$$

cioè:

$$p \approx 1 - \frac{1}{N} \binom{n}{2} + o(N^2).$$

Stima delle probabilità di non collisione

- In generale dovremo scegliere n elementi da un gruppo di N senza collisioni. Sviluppiamo la formula generale rispetto ad $1/N$.

$$p = 1 \cdot \left(1 - \frac{1}{N}\right) \cdot \left(1 - \frac{2}{N}\right) \cdots \left(1 - \frac{n-1}{N}\right) \approx 1 - \frac{1}{N} \left(\sum_{k=0}^{n-1} k\right)$$

cioè:

$$p \approx 1 - \frac{1}{N} \binom{n}{2} + o(N^2).$$

- La scommessa diviene conveniente per $p < 1/2$

$$\frac{1}{N} \binom{n}{2} < 1/2$$

cioè:

$$n^2 \approx N$$

Stima delle probabilità di non collisione

- In generale dovremo scegliere n elementi da un gruppo di N senza collisioni. Sviluppiamo la formula generale rispetto ad $1/N$.

$$p = 1 \cdot \left(1 - \frac{1}{N}\right) \cdot \left(1 - \frac{2}{N}\right) \cdots \left(1 - \frac{n-1}{N}\right) \approx 1 - \frac{1}{N} \left(\sum_{k=0}^{n-1} k\right)$$

cioè:

$$p \approx 1 - \frac{1}{N} \binom{n}{2} + o(N^2).$$

- La scommessa diviene conveniente per $p < 1/2$

$$\frac{1}{N} \binom{n}{2} < 1/2$$

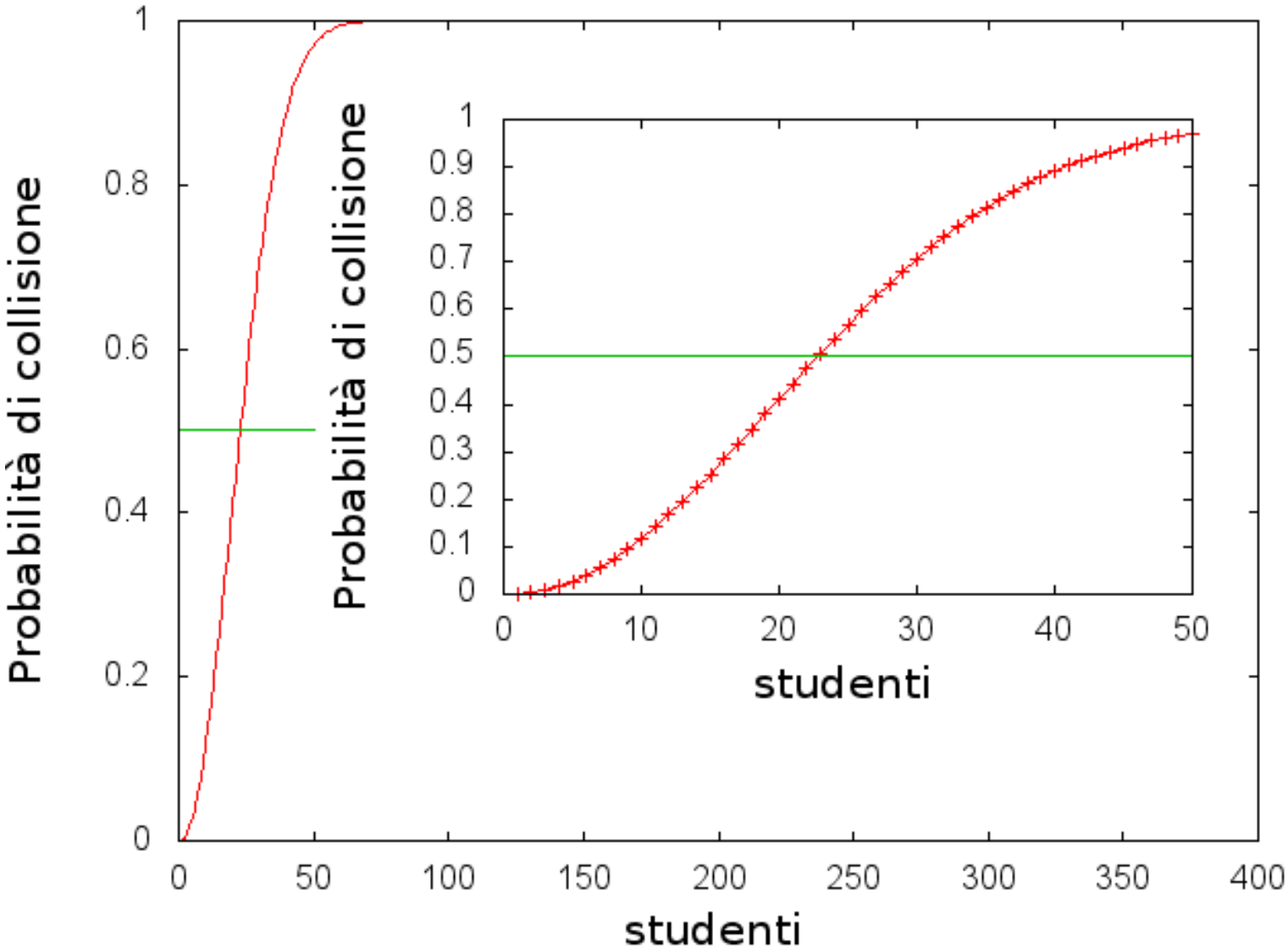
cioè:

$$n^2 \approx N$$

- Tenendo conto delle correzioni quadratiche il valore di n per cui inizia la convenienza è 23.

Probabilità di collisione

- Ecco la curva della probabilità di collisione dei compleanni in funzione del numero di studenti



Altre storielle

- L'anno dopo 1940 gli studenti erano esattamente 23, ma il professore non fece la scommessa; perché?

Altre storielle

- L'anno dopo 1940 gli studenti erano esattamente 23, ma il professore non fece la scommessa; perché?
- Nel 1962 le classi dei cadetti di accademia erano da 20 allievi, il professore fece la stessa scommessa e la vinse; perché?

Altre storielle

- L'anno dopo 1940 gli studenti erano esattamente 23, ma il professore non fece la scommessa; perché?
- Nel 1962 le classi dei cadetti di accademia erano da 20 allievi, il professore fece la stessa scommessa e la vinse; perché?
- Negli anni 70 le classi dei cadetti erano da 15 studenti, ma il professore vinse ugualmente la scommessa perché?

Altre storielle

- L'anno dopo 1940 gli studenti erano esattamente 23, ma il professore non fece la scommessa; perché?
- Nel 1962 le classi dei cadetti di accademia erano da 20 allievi, il professore fece la stessa scommessa e la vinse; perché?
- Negli anni 70 le classi dei cadetti erano da 15 studenti, ma il professore vinse ugualmente la scommessa perché?
- I cadetti hanno tipicamente 16 anni. Il 1924 (1940 -16) era bisestile con 366 giorni. Nell'estate del 1946 ci fu un enorme aumento di nascite causato dalla fine della guerra. Negli anni 70 i cadetti erano divisi per semestri di nascita per aumentare l'uniformità.

Altre storielle

- L'anno dopo 1940 gli studenti erano esattamente 23, ma il professore non fece la scommessa; perché?
- Nel 1962 le classi dei cadetti di accademia erano da 20 allievi, il professore fece la stessa scommessa e la vinse; perché?
- Negli anni 70 le classi dei cadetti erano da 15 studenti, ma il professore vinse ugualmente la scommessa perché?
- I cadetti hanno tipicamente 16 anni. Il 1924 (1940 -16) era bisestile con 366 giorni. Nell'estate del 1946 ci fu un enorme aumento di nascite causato dalla fine della guerra. Negli anni 70 i cadetti erano divisi per semestri di nascita per aumentare l'uniformità.
- In generale la collisione minima si ha quando la distribuzione delle nascite è uniforme. La scommessa del 1939 è vera il resto è fantasia.

Robustezza delle Funzioni di verifica "hash function"

- In molti casi oltre a proteggerci contro gli errori accidentali è necessario difendersi da modifiche deliberate del flusso di dati (tampering). Le funzioni che lo consentono so dette sicure. Rivediamo le diverse forme di robustezza:

Robustezza delle Funzioni di verifica "hash function"

- In molti casi oltre a proteggerci contro gli errori accidentali è necessario difendersi da modifiche deliberate del flusso di dati (tampering). Le funzioni che lo consentono so dette sicure. Rivediamo le diverse forme di robustezza:
- "Resistenza alla ricerca di una preimmagine". Deve essere computazionalmente difficile trovare una sequenza s che abbia una hash function prefissata.

Robustezza delle Funzioni di verifica "hash function"

- In molti casi oltre a proteggerci contro gli errori accidentali è necessario difendersi da modifiche deliberate del flusso di dati (tampering). Le funzioni che lo consentono so dette sicure. Rivediamo le diverse forme di robustezza:
- "Resistenza alla ricerca di una preimmagine". Deve essere computazionalmente difficile trovare una sequenza s che abbia una hash function prefissata.
- "Sicurezza Debole": Data una sequenza, deve essere computazionalmente difficile trovare un'altra sequenza s' che abbia la stessa hash function.

Robustezza delle Funzioni di verifica "hash function"

- In molti casi oltre a proteggerci contro gli errori accidentali è necessario difendersi da modifiche deliberate del flusso di dati (tampering). Le funzioni che lo consentono so dette sicure. Rivediamo le diverse forme di robustezza:
- "Resistenza alla ricerca di una preimmagine". Deve essere computazionalmente difficile trovare una sequenza s che abbia una hash function prefissata.
- "Sicurezza Debole": Data una sequenza, deve essere computazionalmente difficile trovare un'altra sequenza s' che abbia la stessa hash function.
- "Sicurezza Forte" o "Resistenza alle collisioni". In generale, deve essere computazionalmente difficile trovare due sequenze con la stesso hash, indipendentemente dal suo valore.

MD5

- Nel 1992 Ron Rivest pubblicò per lo IEFT l'RFC (Request Forr Comment) un metodo per calcolare una funzione di verifica (Hash function) sicura. Essendo una versione di una serie simile fu chiamata MD5

MD5

- Nel 1992 Ron Rivest pubblicò per lo IEFT l'RFC (Request Forr Comment) un metodo per calcolare una funzione di verifica (Hash function) sicura. Essendo una versione di una serie simile fu chiamata MD5
- MD5 è una funzione a 128 bit usata ancora oggi per la "verifica" dell"integrità del software.

MD5

- Nel 1992 Ron Rivest pubblicò per lo IEFT l'RFC (Request Forr Comment) un metodo per calcolare una funzione di verifica (Hash function) sicura. Essendo una versione di una serie simile fu chiamata MD5
- MD5 è una funzione a 128 bit usata ancora oggi per la "verifica" dell"integrità del software.
- Prima di installare un software o un aggiornamento si verifica una sua hash function (fornita e firmata dal produttore) per essere più certi della provenienza e della integrità, In questo caso bisogna difendersi contro eventuali alterazioni malevole deliberate del software e della sua hash function.

MD5

- Nel 1992 Ron Rivest pubblicò per lo IEFT l'RFC (Request Forr Comment) un metodo per calcolare una funzione di verifica (Hash function) sicura. Essendo una versione di una serie simile fu chiamata MD5
- MD5 è una funzione a 128 bit usata ancora oggi per la "verifica" dell"integrità del software.
- Prima di installare un software o un aggiornamento si verifica una sua hash function (fornita e firmata dal produttore) per essere più certi della provenienza e della integrità, In questo caso bisogna difendersi contro eventuali alterazioni malevole deliberate del software e della sua hash function.
- Una hash function utile alla verifica del software deve godere della sicurezza debole, cioè essere resistente rispetto alla ricerca della seconda controimmagine. In altre parole non deve essere semplice creare un nuovo software con la stessa contro-immagine (Tampering).

MD5 - software per il calcolo

- Sulle macchine linux esiste il comando md5 (apple) o md5sum (e quindi il sorgente in c) che fornisce l'hash md5 di un qualsiasi file in input. Si noti che essendo linux "open source" chiunque può verificare che il comando md5sum esegua esattamente l'algoritmo di Rivest.

MD5 - software per il calcolo

- Sulle macchine linux esiste il comando md5 (apple) o md5sum (e quindi il sorgente in c) che fornisce l'hash md5 di un qualsiasi file in input. Si noti che essendo linux "open source" chiunque può verificare che il comando md5sum esegua esattamente l'algoritmo di Rivest.
- In windows il software non è nativo, ma si può scaricare un software anch'esso "open source" chiamato FCIV che è un eseguibile equivalente.

MD5 - software per il calcolo

- Sulle macchine linux esiste il comando md5 (apple) o md5sum (e quindi il sorgente in c) che fornisce l'hash md5 di un qualsiasi file in input. Si noti che essendo linux "open source" chiunque può verificare che il comando md5sum esegua esattamente l'algoritmo di Rivest.
- In windows il software non è nativo, ma si può scaricare un software anch'esso "open source" chiamato FCIV che è un eseguibile equivalente.
- Quando installiamo o aggiorniamo il software dai siti certificati del produttore (vedremo in seguito cosa significa), il sistema operativo esegue automaticamente la verifica delle "check sum": hash function come md5 o altre.

MD5 - Algoritmo

- Vediamo l'algoritmo,
  Supponiamo di avere una sequenza di bit (ad esempio un messaggio o un software da installare) di lunghezza b:

$$M = \{m_0, m_1, \ldots, m_{b-1}\}$$

MD5 - Algoritmo

- Vediamo l'algoritmo,
  Supponiamo di avere una sequenza di bit (ad esempio un messaggio o un software da installare) di lunghezza b:

$$M = \{m_0, m_1, \ldots, m_{b-1}\}$$

- "Pudding" si aggiungono bit finali in modo da avere N blocchi da 512 bit ed una sequenza residua di 448. Il pudding è formato da un 1 seguito da tutti zeri.

MD5 - Algoritmo

- Vediamo l'algoritmo,
  Supponiamo di avere una sequenza di bit (ad esempio un messaggio o un software da installare) di lunghezza b:

$$M = \{m_0, m_1, \ldots, m_{b-1}\}$$

- "Pudding" si aggiungono bit finali in modo da avere N blocchi da 512 bit ed una sequenza residua di 448. Il pudding è formato da un 1 seguito da tutti zeri.
- Gli ultimi 64 bit conterranno il numero b.

MD5 - Algoritmo

- Vediamo l'algoritmo,
  Supponiamo di avere una sequenza di bit (ad esempio un messaggio o un software da installare) di lunghezza b:

$$M = \{m_0, m_1, \ldots, m_{b-1}\}$$

- "Pudding" si aggiungono bit finali in modo da avere N blocchi da 512 bit ed una sequenza residua di 448. Il pudding è formato da un 1 seguito da tutti zeri.
- Gli ultimi 64 bit conterranno il numero b.
- Si otterranno quindi N blocchi da 512 bit.

MD5 - Algoritmo

- Vediamo l'algoritmo,
  Supponiamo di avere una sequenza di bit (ad esempio un messaggio o un software da installare) di lunghezza b:

$$M = \{m_0, m_1, \ldots, m_{b-1}\}$$

- "Pudding" si aggiungono bit finali in modo da avere N blocchi da 512 bit ed una sequenza residua di 448. Il pudding è formato da un 1 seguito da tutti zeri.
- Gli ultimi 64 bit conterranno il numero b.
- Si otterranno quindi N blocchi da 512 bit.
- Si alloca un "buffer" (un area di memoria) di 128 bit per il digest che viene divisa in 4 parole ognuna da 32bit=8caratteri esadecimali chiamate A, B, C, D.

MD5 - Algoritmo

- Per ogni blocco, l buffer si inizializza sempre cosí: (usando notazione esadecimale):

$$A : 01234567 \; B : 89abcdef \; C : fedcba98 \; D : 76543210$$

MD5 - Algoritmo

- Per ogni blocco, I buffer si inizializza sempre cosí: (usando notazione esadecimale):

$$A : 01234567 \; B : 89abcdef \; C : fedcba98 \; D : 76543210$$

- Le operazioni che si eseguono sulle "parole" da 32 bit sono basate su 4 funzioni logiche elementari F, G, H, I:

$$\begin{aligned} F(X,Y,Z) &= (X.AND.Y).OR.(\bar{X}.AND.Z) \\ G(X,Y,Z) &= (X.AND.Z).OR.(Y.AND.\bar{Z}) \\ H(X,Y,Z) &= X.XOR.Y.XOR.Z = X \oplus Y \oplus Z \\ I(X,Y,Z) &= Y.XOR.(X.OR.\bar{Z}) \end{aligned}$$

Queste funzioni si eseguono sulle parole in modalità "vettoriale" cioè un bit alla volta. Essendo funzioni logiche F,G,H,I devono manipolare singoli bit.

MD5 - Algoritmo

- Per ogni blocco, I buffer si inizializza sempre cosí: (usando notazione esadecimale):

$$A : 01234567 \; B : 89abcdef \; C : fedcba98 \; D : 76543210$$

- Le operazioni che si eseguono sulle "parole" da 32 bit sono basate su 4 funzioni logiche elementari F, G, H, I:

$$\begin{aligned} F(X,Y,Z) &= (X.AND.Y).OR.(\bar{X}.AND.Z) \\ G(X,Y,Z) &= (X.AND.Z).OR.(Y.AND.\bar{Z}) \\ H(X,Y,Z) &= X.XOR.Y.XOR.Z = X \oplus Y \oplus Z \\ I(X,Y,Z) &= Y.XOR.(X.OR.\bar{Z}) \end{aligned}$$

  Queste funzioni si eseguono sulle parole in modalità "vettoriale" cioè un bit alla volta. Essendo funzioni logiche F,G,H,I devono manipolare singoli bit.

- Oltre a queste funzioni logiche si usa una funzione trigonometrica $T[i]$:

$$T[i] \stackrel{def}{=} Int[4294967296 \cdot |\sin(i)|] = Int[2^{32} \cdot |\sin(i)|].$$

MD5 - Algoritmo

- Per ogni blocco si eseguono 4 trasformazioni dette round di questo tipo:

  $$[abcd\ k\ s\ i]: \quad a \rightarrow b + ((a + F(b,c,d) + X[k] + T[i]) <<< s)$$

  in cui a,b,c,d sono "parole" del digest, $X[k]$ è la k-esima parola (da 32 bit) della sequenza in ingresso (da 512 bit), s è lo shift a sx e F è una delle funzioni logiche definite in precedenza.

MD5 - Algoritmo

- Per ogni blocco si eseguono 4 trasformazioni dette round di questo tipo:

  $[abcd\ k\ s\ i]: \quad a \rightarrow b + ((a + F(b,c,d) + X[k] + T[i]) <<< s)$

  in cui a,b,c,d sono "parole" del digest, $X[k]$ è la k-esima parola (da 32 bit) della sequenza in ingresso (da 512 bit), s è lo shift a sx e F è una delle funzioni logiche definite in precedenza.
- Ecco il primo round:
  [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]

MD5 - Algoritmo

- Ecco il secondo round:
  [ABCD 1 5 17] [DABC 6 9 18] [CDAB 11 14 19] [BCDA 0 20 20]
  [ABCD 5 5 21] [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 31] [BCDA 12 20 32]

MD5 - Algoritmo

- Ecco il secondo round:
  [ABCD 1 5 17] [DABC 6 9 18] [CDAB 11 14 19] [BCDA 0 20 20]
  [ABCD 5 5 21] [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 31] [BCDA 12 20 32]
- Ecco il terzo round:
  [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 47] [BCDA 2 23 48]

MD5 - Algoritmo

- Ecco il secondo round:
  [ABCD 1 5 17] [DABC 6 9 18] [CDAB 11 14 19] [BCDA 0 20 20]
  [ABCD 5 5 21] [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 31] [BCDA 12 20 32]
- Ecco il terzo round:
  [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 47] [BCDA 2 23 48]
- E il quarto round:
  [ABCD 0 6 49] [DABC 7 10 50] [CDAB 14 15 51] [BCDA 5 21 52]
  [ABCD 12 6 53] [DABC 3 10 54] [CDAB 10 15 55] [BCDA 1 21 56]
  [ABCD 8 6 57] [DABC 15 10 58] [CDAB 6 15 59] [BCDA 13 21 60]
  [ABCD 4 6 61] [DABC 11 10 62] [CDAB 2 15 63] [BCDA 9 21 64]

MD5 - Algoritmo

- Alla fine si somma a ogni parola del buffer risultante il valore ottenuto dall'elaborazione dei blocchi precedenti:

$$A \rightarrow A + A_{prec} \quad B \rightarrow B + B_{prec}$$
$$C \rightarrow C + C_{prec} \quad D \rightarrow D + D_{prec}$$

MD5 - Algoritmo

- Alla fine si somma a ogni parola del buffer risultante il valore ottenuto dall'elaborazione dei blocchi precedenti:

  $A \rightarrow A + A_{prec} \quad B \rightarrow B + B_{prec}$
  $C \rightarrow C + C_{prec} \quad D \rightarrow D + D_{prec}$

- Il risultato quindi dipende da tutti i blocchi e basta cambiare un bit della sequenza in ingresso per ottenere un digest diverso.

MD5 - Algoritmo

- Alla fine si somma a ogni parola del buffer risultante il valore ottenuto dall'elaborazione dei blocchi precedenti:

  $A \rightarrow A + A_{prec} \quad B \rightarrow B + B_{prec}$
  $C \rightarrow C + C_{prec} \quad D \rightarrow D + D_{prec}$

- Il risultato quindi dipende da tutti i blocchi e basta cambiare un bit della sequenza in ingresso per ottenere un digest diverso.
- Malgrado la procedura sia deterministica e nota a tutti, MD5 gode della sicurezza debole. Effettivamente questa funzione di hash non sembra essere stata "craccata". Non si è trovato un metodo per trovare una seconda controimmagine.

MD5 - Algoritmo

- Alla fine si somma a ogni parola del buffer risultante il valore ottenuto dall'elaborazione dei blocchi precedenti:

  $A \rightarrow A + A_{prec} \quad B \rightarrow B + B_{prec}$
  $C \rightarrow C + C_{prec} \quad D \rightarrow D + D_{prec}$

- Il risultato quindi dipende da tutti i blocchi e basta cambiare un bit della sequenza in ingresso per ottenere un digest diverso.
- Malgrado la procedura sia deterministica e nota a tutti, MD5 gode della sicurezza debole. Effettivamente questa funzione di hash non sembra essere stata "craccata". Non si è trovato un metodo per trovare una seconda controimmagine.
- Sulle riviste specialistiche sono stati riportati casi di "collisioni" (coppie di sequenze con lo stesso digest MD5) quindi MD5 non gode della sicurezza forte.

MD5 - Algoritmo

- Alla fine si somma a ogni parola del buffer risultante il valore ottenuto dall'elaborazione dei blocchi precedenti:

  $A \rightarrow A + A_{prec} \quad B \rightarrow B + B_{prec}$
  $C \rightarrow C + C_{prec} \quad D \rightarrow D + D_{prec}$

- Il risultato quindi dipende da tutti i blocchi e basta cambiare un bit della sequenza in ingresso per ottenere un digest diverso.
- Malgrado la procedura sia deterministica e nota a tutti, MD5 gode della sicurezza debole. Effettivamente questa funzione di hash non sembra essere stata "craccata". Non si è trovato un metodo per trovare una seconda controimmagine.
- Sulle riviste specialistiche sono stati riportati casi di "collisioni" (coppie di sequenze con lo stesso digest MD5) quindi MD5 non gode della sicurezza forte.
- Sono stati definiti degli altri standard denominati SHA (Secure Hash Algorithm), che vedremo in seguito, dei quali non si riportano collissioni.

MD5 - Esempio noto di collisione

- Queste due stringhe (in formato esadecimale):

d131dd02c5e6eec4693d9a0698aff95c 2fcab58712467eab4004583eb8fb7f89

55ad340609f4b30283e4888325**7**1415a 085125e8f7cdc99fd91dbdf280373c5b

d8823e3156348f5bae6dacd436c919c6 dd53e2**b**487da03fd02396306d248cda0

e99f33420f577ee8ce54b67080a80d1e c69821bcb6a8839396f9652b6ff72a70

e

d131dd02c5e6eec4693d9a0698aff95c 2fcab50712467eab4004583eb8fb7f89

55ad340609f4b30283e4888325**f**1415a 085125e8f7cdc99fd91dbd7280373c5b

d8823e3156348f5bae6dacd436c919c6 dd53e2**3**487da03fd02396306d248cda0

e99f33420f577ee8ce54b67080280d1e c69821bcb6a8839396f965ab6ff72a70

Hanno in comune la hash MD5:
79054025255fb1a26e4bc422aef54eb4

MD5 - Esempio noto di collisione

- Queste due stringhe (in formato esadecimale):

  d131dd02c5e6eec4693d9a0698aff95c 2fcab58712467eab4004583eb8fb7f89

  55ad340609f4b30283e4888325**7**1415a 085125e8f7cdc99fd91dbdf280373c5b

  d8823e3156348f5bae6dacd436c919c6 dd53e2**b**487da03fd02396306d248cda0

  e99f33420f577ee8ce54b67080a80d1e c69821bcb6a8839396f9652b6ff72a70

  e

  d131dd02c5e6eec4693d9a0698aff95c 2fcab50712467eab4004583eb8fb7f89

  55ad340609f4b30283e4888325**f**1415a 085125e8f7cdc99fd91dbd7280373c5b

  d8823e3156348f5bae6dacd436c919c6 dd53e2**3**487da03fd02396306d248cda0

  e99f33420f577ee8ce54b67080280d1e c69821bcb6a8839396f965ab6ff72a70

  Hanno in comune la hash MD5:
  79054025255fb1a26e4bc422aef54eb4

- Dal sito di gordion si possono scaricare due piccoli file con lo stesso hash md5.

MD5 - Esempio noto di collisione

- Queste due stringhe (in formato esadecimale):

  d131dd02c5e6eec4693d9a0698aff95c 2fcab58712467eab4004583eb8fb7f89

  55ad340609f4b30283e4888325**7**1415a 085125e8f7cdc99fd91dbdf280373c5b

  d8823e3156348f5bae6dacd436c919c6 dd53e2**b**487da03fd02396306d248cda0

  e99f33420f577ee8ce54b67080a80d1e c69821bcb6a8839396f9652b6ff72a70

  e

  d131dd02c5e6eec4693d9a0698aff95c 2fcab50712467eab4004583eb8fb7f89

  55ad340609f4b30283e4888325**f**1415a 085125e8f7cdc99fd91dbd7280373c5b

  d8823e3156348f5bae6dacd436c919c6 dd53e2**3**487da03fd02396306d248cda0

  e99f33420f577ee8ce54b67080280d1e c69821bcb6a8839396f965ab6ff72a70

  Hanno in comune la hash MD5:
  79054025255fb1a26e4bc422aef54eb4
- Dal sito di gordion si possono scaricare due piccoli file con lo stesso hash md5.
- Ma non si conoscono casi di stringhe con lo stesso hash MD5 di una sequenza assegnata (ad esempio un software). Quindi MD5 finora gode della sicurezza debole, ma non di quella forte.

Le funzioni di verifica sicure "secure hash functions" SHA

- In questo caso non si cerca di proteggerci solo dagli errori casuali, ma

Le funzioni di verifica sicure "secure hash functions" SHA

- In questo caso non si cerca di proteggerci solo dagli errori casuali, ma
- Nel 1993 il National Institute of Standards and Technology (NIST) ha pubblicato lo standard (FIPS 180) in 1993 che definisce i Secure Hash Algorithm SHA (algoritmo sicuro di hash). Nel 1995 si scoprì una debolezza dell'algoritmo e fu definito il (FIPS 180-1) a cui corrisponde l'algoritmo SHA-1. Infine nel 2002 furono definiti SHA-256, SHA-384 e SHA-512 (in cui il nome viene dalla lunghezza della hash generata). SHA-256, SHA-384 e SHA-512 vengono chiamati SHA-2.

SHA-512

- L'algoritmo trasforma una sequenza di al più 2^{312} bit in una sequenza di 512 bit.

SHA-512

- L'algoritmo trasforma una sequenza di al più 2^{312} bit in una sequenza di 512 bit.
- L'input viene suddiviso in blocchi da 1024 bit. Quindi si deve applicare il pudding

SHA-512

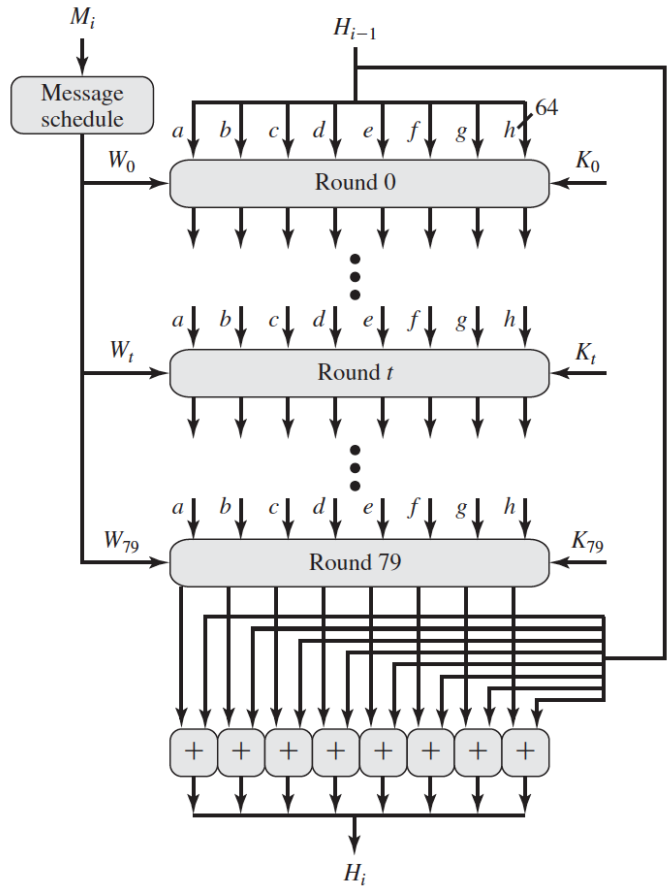
- L'algoritmo trasforma una sequenza di al più 2^{312} bit in una sequenza di 512 bit.
- L'input viene suddiviso in blocchi da 1024 bit. Quindi si deve applicare il pudding
- Si aggiunge un blocco di 128 bit che contiene la lunghezza del messaggio originale (da cui il limite).

SHA-512

- L'algoritmo trasforma una sequenza di al più 2^{312} bit in una sequenza di 512 bit.
- L'input viene suddiviso in blocchi da 1024 bit. Quindi si deve applicare il pudding
- Si aggiunge un blocco di 128 bit che contiene la lunghezza del messaggio originale (da cui il limite).
- L'output (512 bit) viene scomposto in 8 "registri" da 64 bit e viene inizializzato ai valori delle parti decimali delle radici quadrate dei primi 8 numeri primi, corrispondenti ai seguenti valori in formato esadecimale:
  a: 6A09E667F3BCC908 b: BB67AE8584CAA73B
  c : 3C6EF372FE94F82B d: A54FF53A5F1D36F1
  e: 510E527FADE682D1 f: 9B05688C2B3E6C1F
  g: 1F83D9ABFB41BD6B h: 5BE0CD19137E2179
  In realtà le sequenze vengono scritte in ordine inverso (bit più significativo a dx) detto formato "big endian".

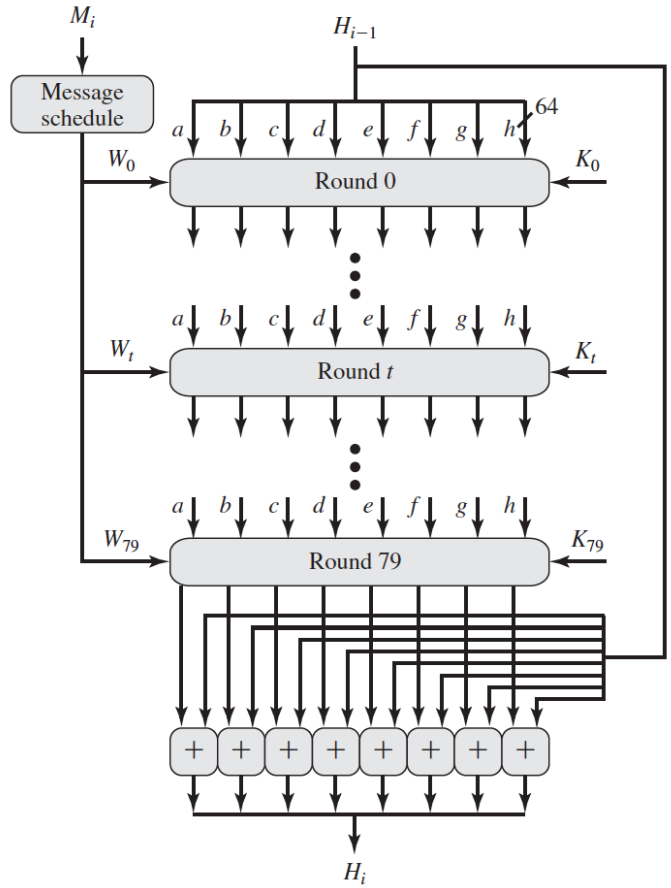
SHA-512: Il modulo F centrale

- Ogni blocco da 1024 bit viene elaborato tramite una funzione F che è il punto chiave della crittazione. F è composto da 80 round, ognuno dei quali ha uno shift ciclico ed alcune operazioni logiche binarie.



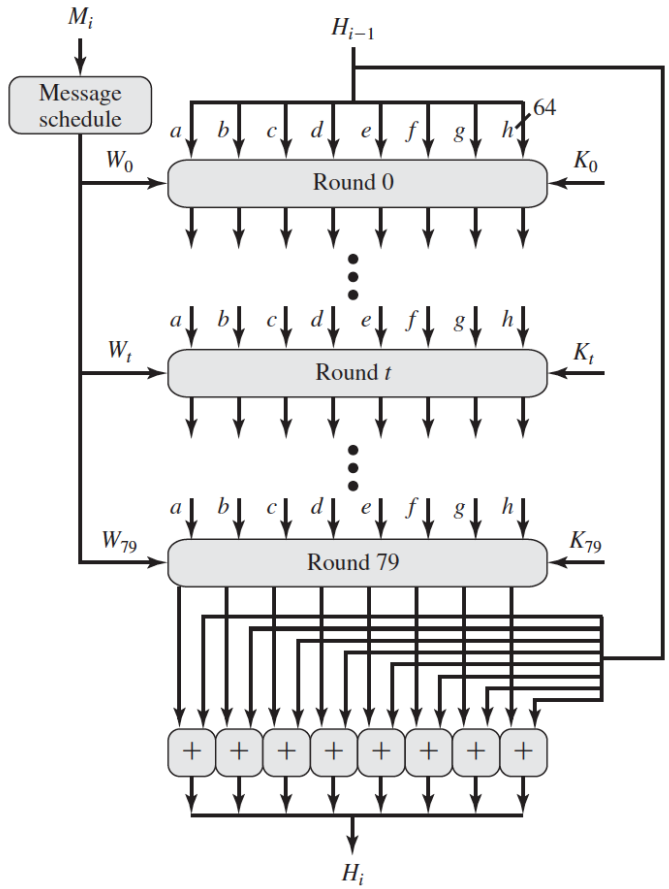
SHA-512: Il modulo F centrale

- Ogni blocco da 1024 bit viene elaborato tramite una funzione F che è il punto chiave della crittazione. F è composto da 80 round, ognuno dei quali ha uno shift ciclico ed alcune operazioni logiche binarie.
- K_i sono i primi 64 bit delle radici cubiche dei primi 80 primi;



SHA-512: Il modulo F centrale

- Ogni blocco da 1024 bit viene elaborato tramite una funzione F che è il punto chiave della crittazione. F è composto da 80 round, ognuno dei quali ha uno shift ciclico ed alcune operazioni logiche binarie.
- K_i sono i primi 64 bit delle radici cubiche dei primi 80 primi;
- W_i sono 80 blocchi diversi da 64 bit ottenuto dal messaggio di 1024 bit.
- Operazioni modulo 2^{64}.



SHA-512 Output

- Il risultato di ogni blocco da 1024 bit è sommato al successivo registro per registro, cioè 64 bit alla volta modulo 2^{64}

SHA-512 Output

- Il risultato di ogni blocco da 1024 bit è sommato al successivo registro per registro, cioè 64 bit alla volta modulo 2^{64}
- Eccola la tabella riassuntiva (presa da Stalling e Brown):

Table 21.1 Comparison of SHA Parameters

	SHA-1	SHA-256	SHA-384	SHA-512
Message digest size	160	256	384	512
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	1024	1024
Word size	32	32	64	64
Number of steps	80	64	80	80
Security	80	128	192	256

SHA-512 Output

- Il risultato di ogni blocco da 1024 bit è sommato al successivo registro per registro, cioè 64 bit alla volta modulo 2^{64}
- Eccola la tabella riassuntiva (presa da Stalling e Brown):

Table 21.1 Comparison of SHA Parameters

	SHA-1	SHA-256	SHA-384	SHA-512
Message digest size	160	256	384	512
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	1024	1024
Word size	32	32	64	64
Number of steps	80	64	80	80
Security	80	128	192	256

- Si sta già lavorando sullo standard SHA-3 che avrà delle hash più grandi.

Uso dell' SHA

- L'applicazione più semplice è i cosiddetto controllo di versione. Come detto diverse volte, ogni pacchetto software vienne fornito con una hash function (digest) che ne "garantisce" l'integrità. Le SHA rendono difficile (computazionalmente troppo oneroso) modificare il software e la sua hash per inserire al suo interno del malware.

Uso dell' SHA

- L'applicazione più semplice è i cosiddetto controllo di versione. Come detto diverse volte, ogni pacchetto software vienne fornito con una hash function (digest) che ne "garantisce" l'integrità. Le SHA rendono difficile (computazionalmente troppo oneroso) modificare il software e la sua hash per inserire al suo interno del malware.
- In generale in tutte le firme digitali, l'adozione di SHA le rende robuste rispetto agli attacchi.

Uso dell' SHA

- L'applicazione più semplice è i cosiddetto controllo di versione. Come detto diverse volte, ogni pacchetto software vienne fornito con una hash function (digest) che ne "garantisce" l'integrità. Le SHA rendono difficile (computazionalmente troppo oneroso) modificare il software e la sua hash per inserire al suo interno del malware.
- In generale in tutte le firme digitali, l'adozione di SHA le rende robuste rispetto agli attacchi.
- Le SHA sono anche usate per la verifica di integrità dei pacchetti provenienti dai server nel protocollo TSL alla base di https. Nello scambio delle chiavi si potrebbero commettere errori o sostituire i pacchetti, ma se si usa una SHA è più difficile. [Lo vedremo in seguito]

Uso dell' SHA

- L'applicazione più semplice è i cosiddetto controllo di versione. Come detto diverse volte, ogni pacchetto software vienne fornito con una hash function (digest) che ne "garantisce" l'integrità. Le SHA rendono difficile (computazionalmente troppo oneroso) modificare il software e la sua hash per inserire al suo interno del malware.
- In generale in tutte le firme digitali, l'adozione di SHA le rende robuste rispetto agli attacchi.
- Le SHA sono anche usate per la verifica di integrità dei pacchetti provenienti dai server nel protocollo TSL alla base di https. Nello scambio delle chiavi si potrebbero commettere errori o sostituire i pacchetti, ma se si usa una SHA è più difficile. [Lo vedremo in seguito]
- SHA si usa anche per la memorizzazione delle password e molte altre applicazioni (bitcoin) che vedremo in seguito.

Una buona pratica

- Quando si installa un software è opportuno verificarne l'origine e l'integrità.

Una buona pratica

- Quando si installa un software è opportuno verificarne l'origine e l'integrità.
- L'integrità si verifica tramite una hash function.

Una buona pratica

- Quando si installa un software è opportuno verificarne l'origine e l'integrità.
- L'integrità si verifica tramite una hash function.
- L'origine si verifica tramite un certificato digitale, cioè con la firma digitale del digest.

Una buona pratica

- Quando si installa un software è opportuno verificarne l'origine e l'integrità.
- L'integrità si verifica tramite una hash function.
- L'origine si verifica tramite un certificato digitale, cioè con la firma digitale del digest.
- Utilizzando gli aggiornamenti dei sistemi operativi, il meccanismo di verifica è automatico per il software dello stesso produttore, ma la verifica va fatta manualmente per il software delle "terze parti".

Esempi in aula

- La divisione in colonna senza resto: or esclusivo componente per componente. Attenzione è diversa dalla divisione intera.

Esempi in aula

- La divisione in colonna senza resto: or esclusivo componente per componente. Attenzione è diversa dalla divisione intera.
- Esempio con il polinomio $g(x) = x^5 + x^3 + x^2 + 1$.

Esempi in aula

- La divisione in colonna senza resto: or esclusivo componente per componente. Attenzione è diversa dalla divisione intera.
- Esempio con il polinomio $g(x) = x^5 + x^3 + x^2 + 1$.
- Sequenza equivalente $(1, 0, 1, 1, 0, 1)$.

Esempi in aula

- La divisione in colonna senza resto: or esclusivo componente per componente. Attenzione è diversa dalla divisione intera.
- Esempio con il polinomio $g(x) = x^5 + x^3 + x^2 + 1$.
- Sequenza equivalente $(1, 0, 1, 1, 0, 1)$.
- Esercizio per casa: aggiungere il digest (con la hash function del polinomio g) as una sequenza casuale di 100 bit.

Esempi in aula

- La divisione in colonna senza resto: or esclusivo componente per componente. Attenzione è diversa dalla divisione intera.
- Esempio con il polinomio $g(x) = x^5 + x^3 + x^2 + 1$.
- Sequenza equivalente $(1, 0, 1, 1, 0, 1)$.
- Esercizio per casa: aggiungere il digest (con la hash function del polinomio g) as una sequenza casuale di 100 bit.
- Verificare la sequenza.

Esempi in aula

- La divisione in colonna senza resto: or esclusivo componente per componente. Attenzione è diversa dalla divisione intera.
- Esempio con il polinomio $g(x) = x^5 + x^3 + x^2 + 1$.
- Sequenza equivalente $(1, 0, 1, 1, 0, 1)$.
- Esercizio per casa: aggiungere il digest (con la hash function del polinomio g) as una sequenza casuale di 100 bit.
- Verificare la sequenza.
- Verificare che il resto (diversamente dalla divisione intera) è sempre minore di 2^n.

Esempi in aula

- La divisione in colonna senza resto: or esclusivo componente per componente. Attenzione è diversa dalla divisione intera.
- Esempio con il polinomio $g(x) = x^5 + x^3 + x^2 + 1$.
- Sequenza equivalente $(1, 0, 1, 1, 0, 1)$.
- Esercizio per casa: aggiungere il digest (con la hash function del polinomio g) as una sequenza casuale di 100 bit.
- Verificare la sequenza.
- Verificare che il resto (diversamente dalla divisione intera) è sempre minore di 2^n.
- Verifichiamo la collisione di MD5 su file1 e file2 con il comando FCIV -md5 -sha1 filex

Messaggio

- Nelle trasmissioni su una rete ethernet si esegue una ridondanza strutturata detta "ciclica" che consente la "verifica" della corretta trasmissione delle sequenze. Il processo di verifica è continuo e comporta una modesta allocazione di tempo di elaborazione suppletivo.

Messaggio

- Nelle trasmissioni su una rete ethernet si esegue una ridondanza strutturata detta "ciclica" che consente la "verifica" della corretta trasmissione delle sequenze. Il processo di verifica è continuo e comporta una modesta allocazione di tempo di elaborazione suppletivo.
- Il CRC controllo ciclico di ridondanza protegge dagli errori casuali, ma NON difende contro alterazioni deliberate delle sequenze. Non presenta alcuna caratteristica di sicurezza per la difesa contro gli attacchi.

Messaggio

- Nelle trasmissioni su una rete ethernet si esegue una ridondanza strutturata detta "ciclica" che consente la "verifica" della corretta trasmissione delle sequenze. Il processo di verifica è continuo e comporta una modesta allocazione di tempo di elaborazione suppletivo.
- Il CRC controllo ciclico di ridondanza protegge dagli errori casuali, ma NON difende contro alterazioni deliberate delle sequenze. Non presenta alcuna caratteristica di sicurezza per la difesa contro gli attacchi.
- Il fenomeno della collisione è a volte meno raro di quanto non appaia con una analisi superficiale. La sicurezza forte richiede l'assenza di collisioni nelle funzioni di hash.

Messaggio

- Nelle trasmissioni su una rete ethernet si esegue una ridondanza strutturata detta "ciclica" che consente la "verifica" della corretta trasmissione delle sequenze. Il processo di verifica è continuo e comporta una modesta allocazione di tempo di elaborazione suppletivo.
- Il CRC controllo ciclico di ridondanza protegge dagli errori casuali, ma NON difende contro alterazioni deliberate delle sequenze. Non presenta alcuna caratteristica di sicurezza per la difesa contro gli attacchi.
- Il fenomeno della collisione è a volte meno raro di quanto non appaia con una analisi superficiale. La sicurezza forte richiede l'assenza di collisioni nelle funzioni di hash.
- L'algoritmo di hashing SHA gode di sicurezza forte. Abbiamo visto anche MD5 che è un esempio di funzione di verifica a sicurezza debole, ma non forte.

Messaggio

- Nelle trasmissioni su una rete ethernet si esegue una ridondanza strutturata detta "ciclica" che consente la "verifica" della corretta trasmissione delle sequenze. Il processo di verifica è continuo e comporta una modesta allocazione di tempo di elaborazione suppletivo.
- Il CRC controllo ciclico di ridondanza protegge dagli errori casuali, ma NON difende contro alterazioni deliberate delle sequenze. Non presenta alcuna caratteristica di sicurezza per la difesa contro gli attacchi.
- Il fenomeno della collisione è a volte meno raro di quanto non appaia con una analisi superficiale. La sicurezza forte richiede l'assenza di collisioni nelle funzioni di hash.
- L'algoritmo di hashing SHA gode di sicurezza forte. Abbiamo visto anche MD5 che è un esempio di funzione di verifica a sicurezza debole, ma non forte.
- Una applicazione importante delle hash function è la verifica del software detta controllo di versione .