

# Attacchi Generici alle Funzioni Hash

Paolo D'Arco  
pdarco@unisa.it

Università di Salerno

Elementi di Crittografia

## 1 Attacchi generici a funzioni hash

# Attacchi generici a funzioni hash

Qual é il livello massimo di sicurezza che possiamo sperare di ottenere con una funzione hash?

Due attacchi sempre possibili

$$H : \{0, 1\}^* \rightarrow \{0, 1\}^\ell$$

- 1 Valuta  $H$  su  $2^\ell + 1$  input distinti  $\Rightarrow$  due output devono essere uguali
- 2 Scegli  $q$  input *distinti*  $x_1, \dots, x_q$ , calcola  $y_i = H(x_i)$  per  $i = 1, \dots, q$  e controlla che due  $y_i$  siano uguali

Se  $q > 2^\ell$ , con prob. 1 una collisione viene trovata

Quando  $q$  é piú piccolo, qual é la prob. di ottenere una collisione?

L'analisi può essere complicata

# Attacchi generici a funzioni hash

Approccio: **idealizziamo**  $H$  dicendo che é una funzione casuale, i.e., gli output sono uniformemente distribuiti in  $\{0, 1\}^\ell$



Il problema cosí diventa: "qual é la prob. che  $\exists i, j$  tali che  $y_i = y_j$ , se scegliamo  $y_1, \dots, y_q \in \{0, 1\}^\ell$  indipendentemente ed uniformemente a caso?"



É il problema del compleanno:  $q$  persone in stanza,  $y_i$  data compleanno

Sappiamo che se  $q = \Theta(N^{\frac{1}{2}})$  e  $y_1, \dots, y_q \in \{1, \dots, N\}$ ,  $\Rightarrow$  prob.  $\approx \frac{1}{2}$

Nel nostro caso, la funzione hash ha un codominio di  $2^\ell$  stringhe

$\Rightarrow$  prendendo  $q = \Theta(2^{\frac{\ell}{2}})$ , la prob. di ottenere una collisione é  $\approx \frac{1}{2}$ .

Assumendo che una valutazione di  $H$  richieda un'unità di tempo, per resistere ad attacchi di tempo al più  $T$ , la lunghezza dell'output deve essere almeno  $2 \log T$  bit. Infatti:

$$2^{\frac{2 \log T}{2}} = 2^{\log T} = T.$$

Come possiamo trovare collisioni "significative"?

Alice, licenziata al lavoro, vuole trovare due lettere  $x$  ed  $x'$  tali che  $H(x) = H(x')$ .

La lettera  $x$  motiva il licenziamento. La lettera  $x'$  é una lettera di raccomandazione.

Se l'approccio "Hash-and-Mac" viene usato per autenticare le lettere, Alice può usare il tag della prima per autenticare la seconda!

L'attacco descritto prima richiede che  $x_1, \dots, x_q$  siano soltanto *distinti* (non casuali).

# Attacchi generici a funzioni hash

Pertanto, Alice può:

- produrre  $q = \Theta(2^{\frac{\ell}{2}})$  messaggi del primo tipo ( $x$ )
- produrre  $q = \Theta(2^{\frac{\ell}{2}})$  messaggi del secondo tipo ( $x'$ )
- cercare collisioni **tra** i messaggi del primo gruppo e quelli del secondo

È un attacco simile a quello del compleanno.

Un'analisi simile mostra che l'attacco dá una collisione con prob. circa  $1/2$ .

Nota che é facile scrivere messaggi equivalenti dei due tipi. Per esempio:

*È duro/difficile/impegnativo/impossibile  
immaginare/credere di trovare/individuare/assumere  
un'altra persona/segretaria che possieda simili  
abilità/conoscenze/caratteristiche di Alice.*



# Attacchi generici a funzioni hash

Ogni combinazione di un termine per ogni attributo fornisce una lettera equivalente.

Nel nostro caso:  $4 \cdot 2 \cdot 3 \cdot 2 \cdot 3 = 144$

Pertanto é facile generare  $2^{64}$  versioni dello stesso messaggio: basta disporre di 64 parole, ognuna con un sinonimo!

Osservazione: l'attacco del compleanno richiede una grossa quantità di memoria

$\Rightarrow \Theta(2^{\frac{\ell}{2}})$  valori da memorizzare.

Possiamo far meglio?

Tradeoff: piú computazione, meno memoria.

# Attacchi generici a funzioni hash

Idea dell'attacco (basata sull'alg. di Floyd per trovare cicli).

Si consideri la sequenza  $x_i$  definita da

$$x_i = H(x_{i-1})$$

con  $x_0$  valore iniziale.

Se é ciclica, esistono interi  $j > 0$  e  $k > 0$  tali che  $x_{j+k} = x_j$  e, quindi, per tutti gli interi  $i \geq j$  ed  $n \geq 1$  risulta

$$x_{i+nk} = x_i.$$

L'ultima vale naturalmente anche per  $i = nk \Rightarrow x_{nk+nk} = x_{2nk} = x_{nk}$   
 $\Rightarrow \exists$  un intero  $i$  tale che  $x_i = x_{2i}$

D'altra parte, l'esistenza di un tale intero implica che la sequenza é ciclica.

# Attacchi generici a funzioni hash

Pertanto, per individuare cicli in  $\{x_i\}$  é sufficiente controllare per ogni intero positivo  $i$  se  $x_i = x_{2i}$ .

Come procediamo? Scegliamo  $x_0$  a caso e calcoliamo

$$\begin{array}{ll} x_1 = H(x_0) & x_2 = H(H(x_0)) \\ x_2 = H(x_1) & x_4 = H(H(x_2)) \\ \dots & \dots \\ x_i = H(x_{i-1}) & x_{2i} = H(H(x_{2(i-1)})) \end{array}$$

Quando  $x_i = x_{2i}$  sappiamo che  $\exists$  un minimo  $j$  tra 0 ed  $i$  tale che  $x_j = x_{j+i}$  e, quindi,  $(x_{j-1}, x_{j+i-1})$  é una collisione per  $H$ .

Come lo troviamo? Ripetiamo al piú  $i$  volte un calcolo.

# Attacchi generici a funzioni hash

Poniamo all'inizio  $y = x_0$ ;  $z = x_i$ .

Calcoliamo  $H(y)$  e  $H(z)$ .

Se  $H(y) = H(z)$ , allora  $(y, z)$  é una collisione;

altrimenti, aggiorniamo  $y = H(y)$  e  $z = H(z)$  e ripetiamo il calcolo.

Quanto costa? Consideriamo la sequenza  $x_0, x_1, x_2, \dots$

Se **modelliamo**  $H$  come una funzione casuale, risulta per ogni  $y$ ,

$$H(x) = y \text{ con probabilit  } 1/2^\ell.$$

(i valori sono cio  distribuiti indipendentemente ed uniformemente fino a quando la ripetizione non si presenta)

Pertanto ci aspettiamo che una ripetizione occorra con probabilità maggiore di  $1/2$  durante la generazione dei primi  $q = \Theta(2^{\frac{\ell}{2}})$  termini della sequenza.

Se esiste una ripetizione nei primi  $q$  termini, l'algoritmo impiega al più  $q$  passi per trovarla.

# Attacchi generici a funzioni hash

Sono stati messi a punto diversi attacchi che esibiscono un tradeoff tempo/spazio per **invertire** funzioni hash.

Si faccia riferimento al libro di testo per descrizione ed analisi.

# Attacchi generici a funzioni hash

E come trovare collisioni per messaggi significativi?

Con questo attacco Adv non ha piú controllo sugli  $x_i$ . Procediamo come segue:

- Alice scrive ciascun messaggio in modo tale che ci siano  $\ell - 1$  parole intercambiabili
- definisce una funzione uno-a-uno  $g : \{0, 1\}^\ell \rightarrow \{0, 1\}^*$  dove l' $\ell$ -esimo bit seleziona il tipo (0 o 1) del messaggio e l' $i$ -esimo bit la parola intercambiabile.

Per esempio:

0: *Bob é un buon/volenteroso e onesto/fidato lavoratore/dipendente*

1: *Bob é un fastidioso/problematico e saccente/irritante  
lavoratore/dipendente*

# Attacchi generici a funzioni hash

$g(0000) = \text{Bob é un buon e onesto lavoratore}$

$g(0001) = \text{Bob é un buon e onesto dipendente}$

$g(1010) = \text{Bob é un fastidioso e irritante lavoratore}$

...

Definiamo ora  $f : \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell$  come

$$f(x) = H(g(x)).$$

Alice può trovare collisioni in  $f$  usando l'attacco del compleanno con poco spazio precedentemente descritto.

Risulta:  $x, x'$  collisione per  $f \Rightarrow g(x)$  e  $g(x')$  collidono rispetto ad  $H$

Se  $x$  e  $x'$  sono una collisione causale, con prob.  $1/2$  i loro bit "di tipo" sono diversi e l'attacco riesce. Altrimenti, si riprova ripetendo l'attacco dall'inizio.