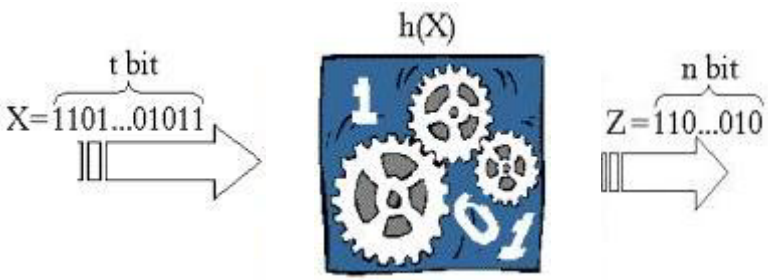
La funzione Hash

Garanzia dell’integrità dei dati e autenticazione dei messaggi

Come funziona l'Hash function

Associa stringhe di bit di lunghezza arbitraria e finita stringhe di bit di lunghezza inferiore.



h: D→C

con dominio e codominio C t.c.

IDI>ICI è del tipo molti-uno

La firma digitale nei messaggi lunghi

Si potrebbe:

- dividere in più blocchi il msg
- firmare ogni blocco singolarmente

Ma ciò comporterebbe:

- Aumento della dimensione della firma poiché proporzionale al numero di blocchi
- Se il messsaggio viene diviso in N blocchi, l'algoritmo di verifica verrà ripetuto N volte, aumentando di molto il tempo di verifica visto il calcolo ripetuto di operazioni complesse
- I blocchi già firmati potrebbero essere captati e selezionati in modo da cambiare il significato originario del messaggio. Essendo la firma originale sarebbe difficile accorgersi dell'avvenuto.

La firma digitale nei messaggi lunghi

La soluzione ottima è l'utilizzo delle Hash function.

Un utente A intende firmare un proprio messaggio x di lunghezza arbitraria e finita:

- Calcola l'Hash function del messaggio x z=h(x)
- Firma z tramite la sua chiave privata
- Trasmette il messaggio x e la z firmata.

ATTENZIONE, LA FUNZIONE NON DEVE PRODURRE COLLISIONI

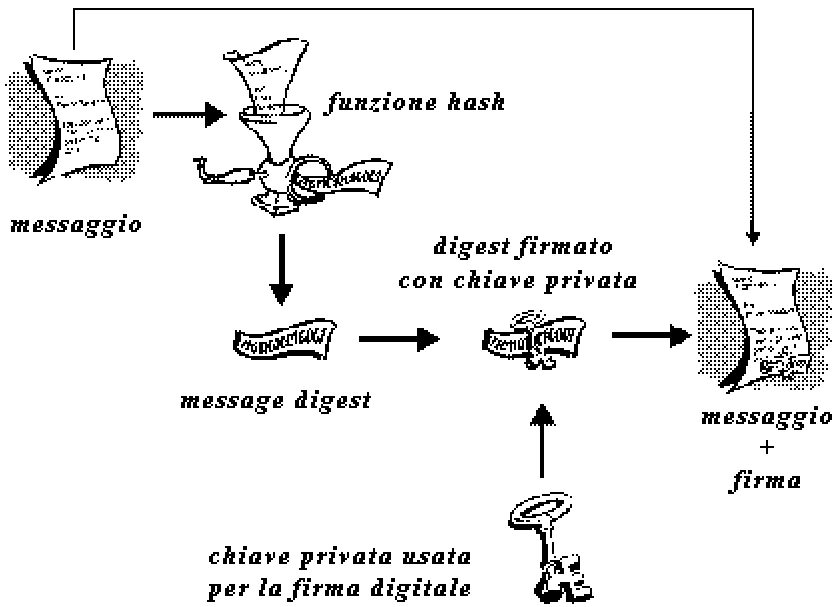
Poiché indebolirebbe lo schema di firma, cosa molto facile per sistemi che permettono la firma di documenti di qualsiasi dimensione

Nasce quindi una classe delle Hash function chiamata MAC che permettono l'autenticazione a chiave simmetrica. Prendono due input in ingresso, messaggio e chiave segreta, e producono un output di dimensione fissata

Uno dei suoi campi d'applicazione: nella sicurezza della posta elettronica

Il mittente genera un messaggio detto di "digest" che ha una lunghezza di solito compresa tra i 128 ed i 160 bits. Qust'ultima non è nient'altro che la "hash function" applicata al messaggio originale. Il digest viene codificato con la chiave privata del mittente (segnatura) ed inoltrato dallo stesso insieme al messaggio originale.

Tramite la chiave pubblica del mittente e riproducendo il digest dal messaggio originale, il ricevente può verificare la fonte del messaggio.



Le proprietà delle Hash function

SICUREZZA DEBOLE: h è debolmente priva di collisioni se noto un messaggio x è computazionalmente inammissibile trovare un messaggio x' t.c.

$$x \neq x' \text{ e } h(x)=h(x')$$

SICUREZZA FORTE: h è fortemente libera da collisioni se è computazionalmente inammissibile trovare due messaggi x e x' t.c.

$$x \neq x' \text{ e } h(x)=h(x')$$

SICUREZZA ONE-WAY: h() è una funzione Hash one-way se dato un digest z è computazionalmente inammissibile trovare un x t.c.

$$h(x)=z$$

Quest'ultima ha la funzione di non permettere ad un utente A di risalire dal digest al messaggio originale.

Le proprietà delle Hash function

Si analizzi il caso in cui la firma sia apposta a h(x) anziché al messaggio originale x

h deve dovrebbe essere debolmente sicura altrimenti un nemico C potrebbe osservare la firma su h(x) e cercare un x' che produce h(x)=h(x').

Se fosse anche fortemente sicura sarebbe anche molto arduo trovare una coppia di messaggi x e x' che facciano collisione.

La necessità della proprietà one-way è più chiara nell'analisi dei sistemi RSA

Sicurezza forte implica sicurezza debole

DIM.

h è fortemente sicura per definizione. È computazionalmente inammissibile trovare fra tutte le coppie una che collide. Stessa cosa se si parte da un messaggio noto e se ne vuole trovare una che crei collisione con essa (definizione di sicurezza debole).

h è fortemente sicura ma non debole. È computazionalmente possibile trovare, noto un messaggio x, un altro con cui collida. Ciò renderebbe possibile che si trovino due coppie che collidano (assurdo poiché è assicurata la sicurezza forte) c.v.d.

Si può inoltre dimostrare che sicurezza forte implica sicurezza one-way

DIM. OMESSA

Attacco del compleanno

Se il codominio della Hash function ha una cardinalità troppo piccola aumenta facilità di individuare nell'insieme delle collisioni. Molto utile per imporre un limite inferiore alla cardinalità di Z si è dimostrato il "Paradosso del compleanno".

Il paradosso: presi 23 individui a caso almeno due hanno lo stesso compleanno con probabilità pari a ½.

Supponiamo che si abbia $h:X \rightarrow Z$ con $|X|=m$ e $|Z|=n$ con n e m finiti si dimostra che il numero di collisioni minimo è paria a n. Si potrebbe pensare di scegliere k valori casuali di X e verificare la presenza di eventuali collisioni.

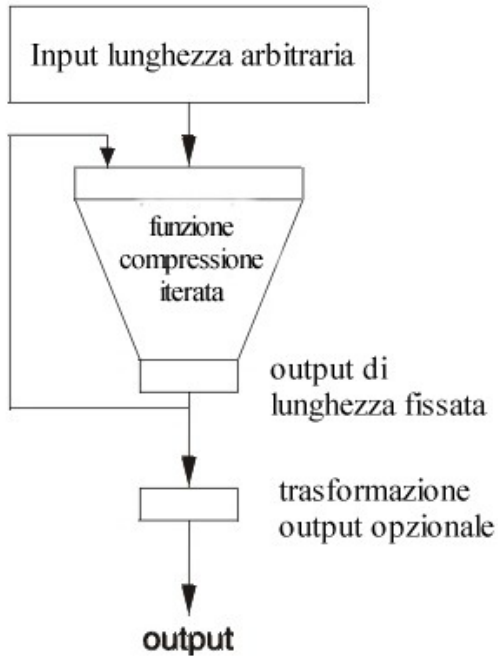
L'analisi del paradosso porta la scelta del numero di bits ad un minimo di 128 poiché con ad esempio 40 bits ancora sarebbe possibile pensare all'utilizzo di un programma di ricerca esaustivo al calcolatore.

Hash function iterate e blocchi

Generalmente le Hash function vengono realizzate con programmi iterativi che accettano in input stringhe di lunghezza finita ma arbitraria ed in uscita producono dei blocchi di lunghezza fissata.

Si rende indispensabile l'aggiunta di alcuni bit di riempimento detti "padding" poichè si deve ottenere una lunghezza della stringa in input che sia multiplo della lunghezza dei singoli blocchi in cui verrà divisa.

Nel padding generalmente è incluso un così detto "blocco" o "blocco parziale" dove è riportata la dimensione originale dell'input prima del padding.



Modello generale di funzione hash iterata