Università degli studi “ROMA TRE”

Facoltà di Ingegneria Informatica

Tesina di
Elementi di Crittografia

La Firma Digitale

De Lucia Giorgio
Saccone Carlo
Mazza Daria

A.A 2004/2005

Introduzione

Con l'avvento di nuove tecnologie si sono moltiplicati gli scambi d'informazioni su reti come Internet. Purtroppo però Internet è sempre più insicuro e corriamo costantemente il rischio che terzi riescano a violare la nostra privacy leggendo informazioni personali o appropriandosi d'informazioni riservate. In altri casi è possibile che malintenzionati agiscano sui nostri dati apportandovi modifiche e alterazioni, con le conseguenze che tutti riusciamo facilmente ad immaginare.
Nasce quindi la necessità di una tecnologia che possa garantire un certo livello di sicurezza per i dati che sono trasmessi, scambiati o solamente archiviati ed evitare così che persone non autorizzate possano ottenere l'accesso alle informazioni in essi contenute proteggendo così i dati diffusi in rete, ma non solo, dai pericoli derivanti da un uso illecito delle informazioni.
E' quindi indispensabile fornire contromisure di sicurezza mirate a garantire:

- Integrità
- Autenticità
- Confidenzialità
- Non ripudiabilità

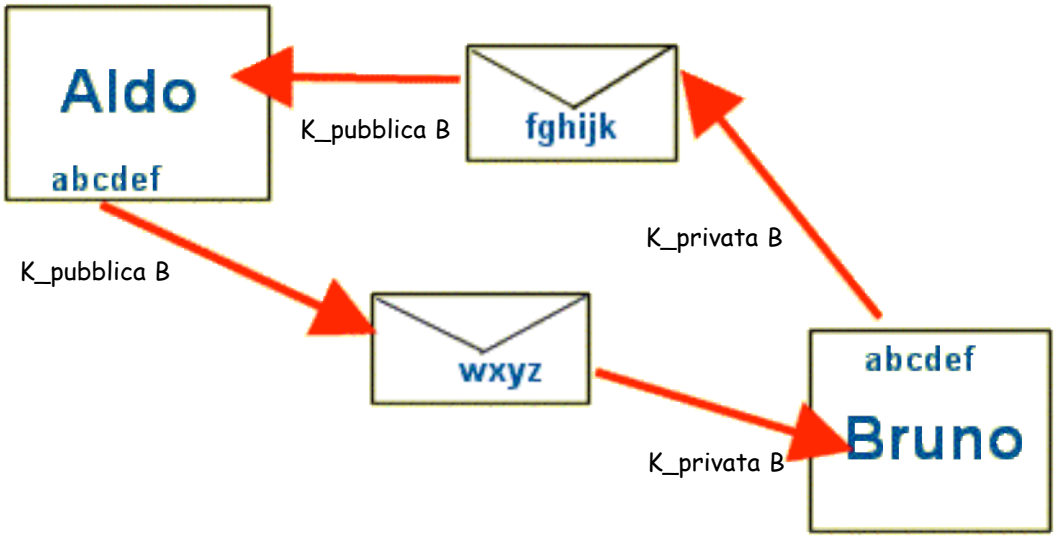
Dobbiamo garantire che un documento non ha subito alterazioni dall'istante in cui è stato creato. Ossia che quanto ricevuto corrisponda a quanto inviato, verificando così l'integrità dei dati.
Bisogna realizzare un sistema di autenticazione forte; in contrapposizione all'autenticazione debole, quella con username e password facilmente attaccabile. In questo modo il ricevente e/o il mittente potranno essere identificati con certezza.
Si dovrà inoltre garantire la privacy, mediante tecniche crittografiche, delle informazioni e dei dati memorizzati preservando così la confidenzialità dei dati.
Infine si deve fornire un meccanismo che garantisca la non ripudiabilità, ossia che un utente che abbia firmato un documento non possa, in seguito, negare di averlo fatto; non riconoscendo quindi la paternità di quanto prodotto.
Riusciamo a raggiungere questi obiettivi grazie all'utilizzo della firma digitale, il risultato di una procedura informatica che ci deve garantire l'autenticità e l'integrità di messaggi e documenti scambiati e archiviati con mezzi informatici. Grazie all'utilizzo di questa tecnologia possiamo effettivamente soddisfare tutti i requisiti richiesti per dare validità legale ad un documento elettronico.
La firma digitale deve svolgere lo stesso ruolo della firma autografa per i documenti tradizionali anche se con qualche differenza. La firma autografa è direttamente riconducibile al soggetto che l'ha generata, mentre la firma digitale è riconducibile solo attraverso il possesso di un segreto, inoltre la prima è facilmente falsificabile, mentre quella digitale richiede una grande mole di lavoro che la rende impossibile da falsificare.
A differenza della firma autografa che è strettamente legata al supporto fisico su cui si trova il documento e su cui questo viene applicata, la firma digitale è legata al contenuto del documento e varia dinamicamente con esso. Punto comune è che entrambe devono essere in qualche modo autenticate per evitare il successivo ripudio da parte di chi l'ha generata.

Crittografia a Chiave Asimmetrica

Il problema della firma digitale è fondamentale su reti come Internet, dove non è poi così difficile inviare messaggi o e-mail sotto falso nome. Quando ci registriamo a questo tipo di servizio, se è vero che ci vengono richiesti i nostri dati personali, è anche vero però che nessuno verifica la reale autenticità di questi. Inoltre per servizi "delicati" quali bonifici bancari o acquisti on-line è assolutamente necessario garantire un'elevata sicurezza che ci assicuri circa l'identità degli utenti e l'accesso non autorizzato ai dati.
Il principio alla base della crittografia a chiave asimmetrica è che la chiave di codifica deve essere diverso dalla chiave di decodifica ed inoltre, deve essere estremamente difficile ricavare una conoscendo l'altra.
E' possibile quindi in questo modo distribuire la propria chiave di decifratura (*chiave pubblica*) e mantenere segreta la chiave di cifratura (*chiave privata*).
Ogni utente del nostro sistema ha così due chiavi, una che deve tenere assolutamente segreta e una che renderà pubblica attraverso registri, associabili a comuni elenchi telefonici. Questa coppia di chiavi può essere dunque utilizzata nell'ambito dei sistemi di validazione o di cifratura di documenti informatici garantendo sicurezza e privacy dei nostri dati. Sono possibili due diversi usi delle chiavi secondo il servizio che si vuole realizzare. Un utente che vuole spedire un messaggio ad un altro ed essere sicuro che un terzo non possa leggerlo, cifrerà il messaggio con la chiave pubblica della persona a cui vuole inviarlo, può essere così sicuro che solo lui possa leggerlo, poiché è il possessore della relativa chiave privata, e quindi l'unico in grado di decifrarlo e leggerlo, offrendo così un servizio definito di confidenzialità.
D'altra parte un mittente può cifrare un messaggio con la propria chiave privata di modo che chi lo riceve, dopo averlo decifrato con la relativa chiave pubblica può sapere con certezza chi gliel'abbia inviato, in quanto esiste solo una persona in possesso della chiave privata con cui è stato firmato il documento. Questo servizio è definito di autenticazione. Con questo procedimento chiunque può leggere il messaggio, la chiave di decifratura è pubblica del resto, ma quello che c'interessa in questo caso è l'identità del mittente.
E' possibile inoltre utilizzare entrambe le chiavi per rendere segreto il messaggio a terzi ma nello stesso tempo rassicurare il reale destinatario dell'identità del mittente.

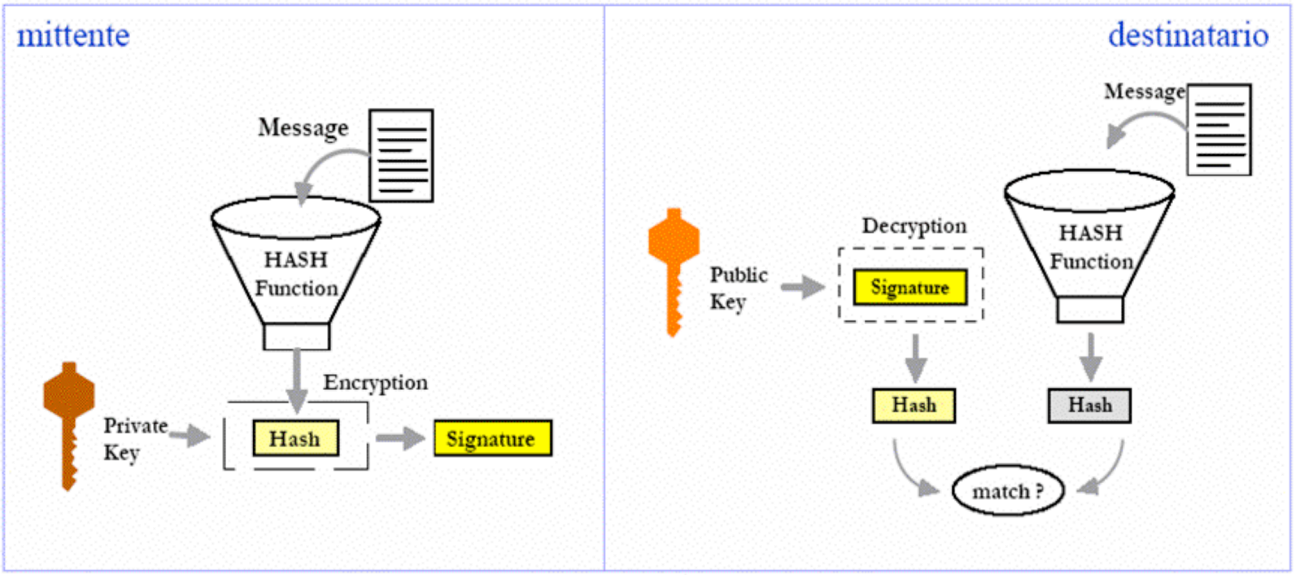
A questo punto si potrebbe pensare di utilizzare le tecniche della crittografia a chiave pubblica per realizzare un sistema di firma digitale sicuro. Si potrebbe cifrare il messaggio da inviare con la propria chiave privata sfruttando il servizio di autenticazione offerto dalla crittografia asimmetrica che abbiamo visto ci garantisce un servizio sicuro. In realtà anche se sicuro questo sistema è molto lento. Rispetto ad un tradizionale sistema di cifratura a chiave simmetrica, tipo il DES, questo impiega un tempo circa 1000 volte maggiore.
Un'idea sarebbe quella di firmare un documento apponendovi qualcosa che svolga lo stesso ruolo della firma tradizionale.
Immaginiamo due utenti, Aldo e Bruno che vogliono dialogare con sicurezza.

Aldo invia una parola qualsiasi, per esempio ***abcdef***, cifrata con la chiave pubblica di Bruno, otterrà qualcosa del tipo ***wxyz***, e la invia. Bruno una volta ricevuta la parola la decifra usando la propria chiave privata, riottenendo così ***abcdef.*** A questo punto Bruno prende la propria chiave privata e la usa per cifrare la parola che aveva ottenuto generando una nuova parola ***fghijk,*** la rispedisce ad Aldo insieme al messaggio che voleva inviare. Aldo decifra la parola ricevuta usando la chiave pubblica di Bruno, se la parola decifrata è identica a quella inviata inizialmente da lui, allora è sicuro che il messaggio provenisse da Bruno.
Solo Bruno infatti avrebbe potuto cifrare la parola inviata. Un eventuale intruso che avesse provato ad inserirsi nel dialogo si sarebbe comunque trovato bloccato,infatti non può decifrare la parola inviata da Aldo perché non conosce la chiave privata di Bruno e anche riuscendo in qualche modo a scoprire quale parola Aldo avesse deciso di inviare non può spacciarsi per Bruno e rispondere ad Aldo, perché continua a non avere la chiave privata di Bruno.



Questo sistema però ancora non va bene, serve qualcosa di più robusto ed efficiente, infatti non ci viene garantito che il messaggio arrivi integro a destinazione o che non venga sostituito, inoltre questa strategia ha lo svantaggio che bisogna utilizzare ogni volta una parola chiave diversa.
Quello che ci servirebbe è qualcosa che sia legato da un lato all'utente che vuole inviare un messaggio e dall'altro all'informazione contenuta in questo. Riusciremo a soddisfare questi requisiti attraverso l'utilizzo, appunto, della firma digitale.

La Firma Digitale



Vediamo a questo punto, brevemente, come funziona la firma digitale.
Il documento che vogliamo inviare è dato in input ad una funzione, detta di hash, che restituisce un breve "riassunto" del documento. Su questo riassunto sarà applicata la chiave privata del mittente,la stringa così ottenuta andrà a costituire la firma digitale per questo messaggio. Dal punto di vista del destinatario, una volta ricevuto il messaggio firmato, applica la chiave pubblica del mittente sulla firma digitale, riottenendo così il valore ricavato dalla funzione di hash. Fatto questo deve utilizzare anche lui la funzione di hash per calcolare il "riassunto" del messaggio che aveva ricevuto in chiaro. Se i due valori così calcolati dal destinarlo corrispondono sarà sicuro dell'effettiva riuscita del processo e quindi dell'identità del mittente e dell'integrità del messaggio che questo gli ha spedito.

Quadro Normativo

Per quanto riguarda le normative che regolano l'utilizzo della firma digitale in Italia, abbiamo che la firma digitale è vista come "*... il risultato della procedura informatica (validazione) basata su un sistema di chiavi asimmetriche a coppia, una pubblica e una privata, che consente al sottoscrittore tramite la chiave privata e al destinatario tramite la chiave pubblica, rispettivamente, di rendere manifesta e di* ***verificare la provenienza e l'integrità di un documento informatico*** *o di un insieme di documenti informatici*".
La firma digitale dal punto di vista legale deve dunque consentire di verificare sempre anche la minima alterazione del documento, riuscendo in questo modo a rendere un qualsiasi documento informatico "valido e rilevante a tutti gli effetti di legge". Per la normativa italiana la firma digitale ha la stessa efficacia della firma autografa, anzi offre un livello di sicurezza maggiore che la firma tradizionale non riesce a dare.

Abbiamo detto che grazie all'impiego della firma digitale un documento elettronico acquisisce sicurezza e riesce a darci la certezza del mittente e del contenuto, questo è molto importante in quanto la legge è chiara sull'uso dei documenti elettronici, stabilisce infatti che: "*Gli atti, i dati e* ***i documenti formati*** *dalla pubblica amministrazione e dai privati con* ***strumenti informatici*** *o telematici, i contratti stipulati nelle medesime forme, nonché la loro archiviazione e trasmissione con strumenti informatici* ***sono validi e rilevanti a tutti gli effetti di legge****; i criteri di applicazione del presente comma sono stabiliti, per la pubblica amministrazione e per i privati, con specifici regolamenti ...*".
Inoltre abbiamo visto che per realizzare la firma digitale vengo utilizzati algoritmi di crittografia asimmetrica per gestire le chiavi pubbliche e private e funzioni di hash per realizzare quello che è stato definito riassunto del messaggio. Anche su questo punto la normativa è molto chiara ed indica cosa è possibile usare.
Leggiamo infatti che:
1. In attesa della pubblicazione degli algoritmi per la generazione e verifica della firma digitale secondo quanto previsto dall'art. 3, i certificatori accreditati … , devono utilizzare l'algoritmo ***RSA*** *(Rivest-Shamir-Adleman) con lunghezza delle chiavi non inferiore a 1024 bit.*
2. In attesa della pubblicazione delle funzioni di hash secondo quanto previsto dall'art. 3, i certificatori accreditati … devono utilizzare uno dei seguenti algoritmi:
*a) dedicated hash-function 3 (****SHA-1****);*
*b) dedicated hash-function 1 (****RIPEMD-160****).*
E' possibile dunque utilizzare esclusivamente l'algoritmo RSA a 1024 bit ,di cui parleremo a breve, per la generazione e verifica delle chiavi, mentre per quanto riguarda le funzioni di hash è invece consentito usare SHA-1 o RIPEMD-160 che saranno analizzate in seguito.

Schema Di Firma

Dal punto di vista informatico la **firma digitale** è una stringa di dati associata ad un messaggio digitale.
Un **algoritmo per la generazione** della firma digitale è un metodo per produrre una firma digitale.
Un **algoritmo per la verifica** della firma digitale è un metodo per verificare l'autenticità di una firma digitale.
Uno **schema** di firma digitale consiste di un algoritmo per la generazione della firma e del relativo algoritmo di verifica.
Vediamo meglio cos'è uno schema di firma digitale;siano dati:
D, un insieme finito di possibili documenti;
F, un insieme finito di possibili firme;
K, un insieme finito di possibili chiavi;

$$se\ k \in K : \begin{cases} sig_k \in algoritmo\ di\ firma \\ ver_k \in algoritmo\ di\ verifica \end{cases}$$

$sig_k : D \rightarrow F$

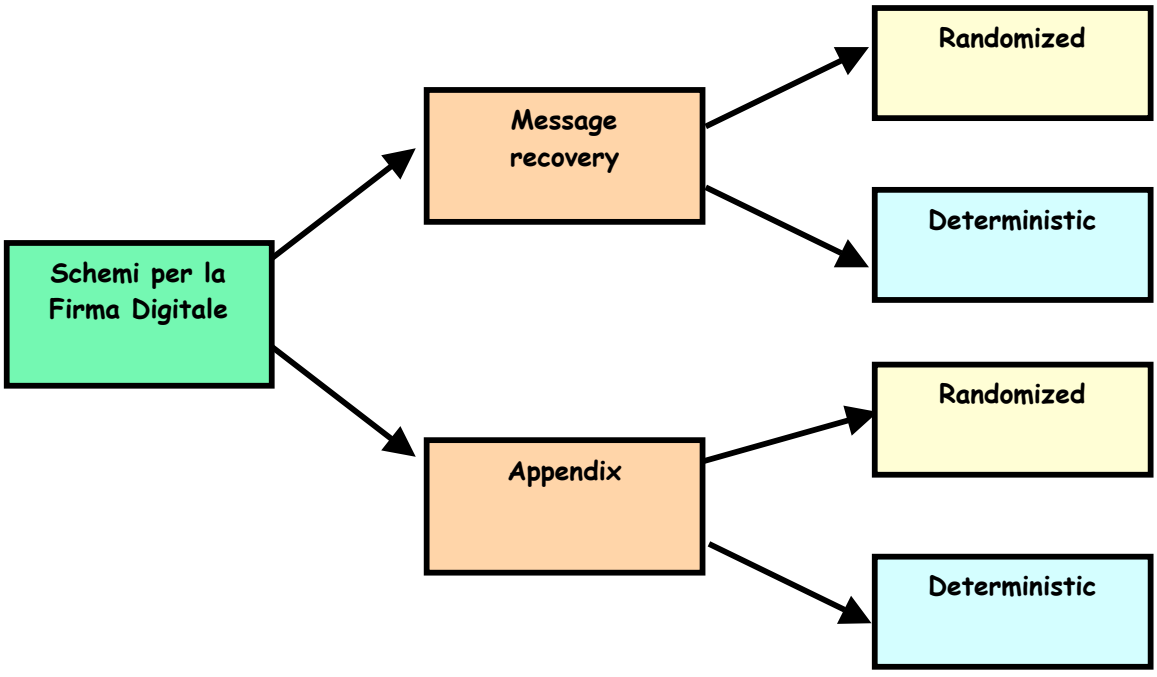
$ver_k : D \times F \rightarrow \{vero, falso\}$ tale che $\forall d \in D, \forall f \in F :$

$ver_k(d, f) = \{vero \text{ se } f = sig_k(d); falso \text{ se } f \neq sig_k(d)\}$

Allora (D, F, K, sig_k, ver_k) costituisce uno **schema di firme**.

DEF: Uno schema di firme è detto **incondizionatamente sicuro** se non esiste un modo per la falsificazione di una firma $f \in F$.

Come è facilmente pensabile non esistono in realtà schemi di firme incondizionatamente sicuri,infatti si potrebbe pensare di testare tutte le possibili firme $y \in F$ di un documento prodotto da un utente fino a quando non si trova quella giusta. Questo tipo di attacco alla sicurezza di uno schema di firme è comunque solo ipotizzabile, infatti poiché la cardinalità dell'insieme delle possibili firme è molto elevata risulta essere computazionalmente troppo oneroso realizzare questo calcolo che diventa così praticamente irrealizzabile.



Questo diagramma riassuntivo ci mostra i vari tipi di schemi di firma digitale. All'inizio abbiamo la divisione in due sottocategorie: Schemi di firma con Message Recovery e schemi con Appendix. Entrambi sono poi divisibili in due sottocategorie, deterministic e randomized. In particolare uno schema di firma si dice di tipo deterministic se effettua una associazione 1 a 1 tra documenti e firme, si definisce randomized in caso contrario.

La caratteristica principale degli schemi di firma che usano message recovery è che le operazioni di verifica di questi schemi recuperano il messaggio dalla firma stessa. In questo modo non è richiesta, dall'algoritmo di verifica, la conoscenza a priori del messaggio. Anche se è utilizzato prevalentemente per messaggi brevi., anche questo tipo di schema fa uso di funzioni di hash. Esempio di algoritmi che

realizzano firma digitale attraverso schemi con message recovery sono RSA, Rabin e Nyberg-Rueppel. Gli schemi con Appendix sono quelli più usati in quanto offrono una maggiore resistenza agli attacchi rispetto ai message recovery. In questo caso l'algoritmo di verifica richiede il messaggio come input, devo quindi trasmettere sia il messaggio originale, in chiaro, che la firma. Si basa anch'esso sull'uso di funzioni di hash. Esempio di algoritmi che realizzano firma digitale attraverso schemi con appendix sono DSA, ElGamal e Schnorr.

Protocollo Diffie-Hellman

Il protocollo Diffie-Hellman, progettato nel 1976 dagli stessi Diffie ed Hellman è un protocollo per lo scambio di una chiave segreta in maniera sicura sopra un canale insicuro, problema questo di grande importanza perché l'intercettazione della chiave comprometterebbe tutta la conversazione successiva. Inizialmente fu progettato per risolvere il problema dell'avvio di un normale sistema di cifratura a chiavi simmetriche,(es. il DES), e quindi per lo scambio iniziale delle chiavi segrete, in realtà ha posto le basi della crittografia a chiavi pubbliche. La sua sicurezza si basa sulla complessità del calcolo del logaritmo discreto che è un problema computazionalmente difficile.
Ricordiamo brevemente alcune definizioni per capire meglio il protocollo di Diffie-Hellman. In un'aritmetica finita di ordine N esistono in genere numeri particolari **g** detti **generatori**: i generatori altro non sono che una base che successivamente elevata a potenza, genera tutti i numeri che sono primi con N. In particolare, se **N** è primo esiste sicuramente per definizione un generatore **g** che in questo caso genererà tutti i numeri compresi tra 1 e N-1, poiché saranno tutti primi con N.
Inoltre abbiamo che la potenza di un numero in un'aritmetica finita è definita come:

$$y = g^{x} \bmod n$$

mentre per quanto riguarda la definizione di logaritmo, questo è *l'esponente che si deve dare alla base g per ottenere il valore y*:

$$x = \log_g y \bmod n$$

ed è proprio questo logaritmo che si dice *logaritmo discreto*, che come ricordavamo prima è estremamente difficile da calcolare.

Vediamo adesso come funziona praticamente il protocollo. Abbiamo due interlocutori, Aldo e Bruno, che vogliono scambiarsi una chiave segreta in modo sicuro su un canale insicuro.

- Inizialmente questi scelgono e quindi rendono pubblico un numero primo **N** molto elevato (per esempio con 1024 bit, sono circa 300 cifre decimali) e un *generatore* **g** per N.
- Aldo genera un numero casuale **a** < N e primo con N e calcola **A** = g^a mod N. Il numero **A** così calcolato viene comunicato pubblicamente a Bruno mentre rimane segreto **a**.
- Allo stesso modo Bruno genera il numero **b**, minore di N e primo con esso, e poi calcola **B** = g^b mod N e invia pubblicamente **B** ad Aldo, e a sua volta mantiene segreto **b**.
- A questo ognuno conosce, oltre ad N e al generatore che avevano scelto insieme,anche il valore calcolato dall'altro.
- Aldo calcola il numero **k** utilizzando il numero inviato da Bruno (B) ed elevandolo al suo numero segreto (a), il tutto modulo N, quindi **k** = $\mathbf{B}^a$ mod **N** allo stesso modo Bruno calcola **k** utilizzando il numero inviato da Aldo (A) e il suo numero segreto (b), quindi **k** = $\mathbf{A}^b$ mod **N**. In questo modo per creare la chiave ogni interlocutore utilizza il proprio numero segreto, ma anche il numero che gli aveva comunicato l'altro, frutto a sua volta di un altro numero segreto.
- Si noti che le due chiavi **k** sono uguali! Infatti entrambe valgono $\mathbf{g}^{ab}$ mod **N**.

In questo modo entrambi possiedono la chiave **k** ma sopra il canale sono stati trasferiti solo **N, g, A** e **B** che non consentono di ricostruirla. In realtà i numeri inviati consentirebbero di ricostruire k, ma questo richiederebbe una mole di tempo enorme, proprio perché richiederebbe il calcolo del logaritmo discreto.
Vediamo un esempio per chiarire come funziona il protocollo. In questo caso l'esempio è stato fatto utilizzando numeri molto piccoli per facilitare i conti, ma ricordiamo che si consiglia l'uso di valori estremamente alti: circa 300 cifre decimali.

- Aldo e Bruno scelgono **N = 17 e g = 3** (3 è generatore di 17)
- Aldo sceglie **a = 6** e quindi **A** = 3^6 = 729 mod 17 = **15**.
- Bruno sceglie **b = 11** e quindi **B** = 3^{11} = 177147 mod 17 = **7**.
- Aldo calcola **k** = $\mathbf{7}^6$ mod 17 = **9**.
- Bruno calcola **k** = $\mathbf{15}^{11}$ mod 17 = **9**.
- La chiave segreta è quindi **9**.

L'algoritmo RSA

Il nome dell'algoritmo deriva dalla prima lettera dei cognomi di coloro che, nell'aprile del 1977, lo inventarono: Ronald L. **R**ivest, Adi **S**hamir e Leonard M. **A**dleman.
Questo algoritmo è sfruttato da moltissime applicazioni fra le quali possiamo ricordare:

- Pretty Good Privacy (PGP)
- Secure Socket Layer (SSL)
- Secure Electronic Transactions (SET)

Anche l'algoritmo RSA si basa su un procedimento che utilizza i numeri primi e funzioni matematiche che è quasi impossibile invertire,tra queste :
Il principio di Eulero ($m^{(p-1)(q-1)} \bmod n = 1$)
Il principio di Fermat ($m^{(p-1)} \bmod p = 1$)

Inoltre sfrutta la complessità computazionale della fattorizzazione in interi di grandi numeri. Dati due numeri primi,è facile calcolarne il prodotto,mentre è difficile determinare, a partire da un determinato numero,quali numeri primi hanno prodotto quel risultato.
Questo ci permette di mantenere valido il principio alla base della crittografia asimmetrica: la difficoltà di derivare la chiave segreta conoscendo la chiave pubblica.
Vediamo allora qual è il procedimento per la generazione della coppia di chiavi.

- Si scelgono due numeri primi **p**, **q** molto grandi;
- Si calcola il loro prodotto **n = p□q**;
- Si calcola la funzione di Eulero di n, che riporta quanti sono i numeri primi con n **_(n) = (p - 1)(q - 1)**. A questo punto i valori p e q non servono più e sono eliminati per evitare che qualcuno possa entrarne in possesso.
- Si sceglie un intero **E** minore di **_(n)** e primo con esso;
- Si calcola,utilizzando la versione estesa dell'algoritmo di Euclide, l'intero **D** così da avere **E□D = 1 mod _(n)**. L'intero D è uguale all'inverso di E modulo _(n). **$D = E^{-1} \bmod \_(n)$**
- Si rendono pubblici i valori **E**,**n** che costituiscono la chiave pubblica.
- Viene mantenuto segreto **D** che utilizzato con **n** rappresenta la chiave privata

Vediamo anche in questo caso un esempio per rendere più chiaro il funzionamento dell'algoritmo. Calcoliamo il valore di n, prodotto di p e q, due numeri primi molto elevati, sono consigliati valori maggiori di $10^{\mathbf{100}}$ ma in questo caso per una maggiore comprensione dell'esempio sono stati utilizzati numeri piccoli.
Prendiamo quindi **p = 7 e q = 5**; il valore di n,loro prodotto sarà: **n** = p□q = 7□5 = **35**.
Calcoliamo la funzione di Eulero di n, **_(n)** =(p-1)(q-1) = **24**. A questo punto p e q non servono più e vanno eliminati.
Dobbiamo scegliere un intero **E** che sia primo rispetto a **_(n)** e minore di esso, decidiamo per **E** = 7.

A questo punto dobbiamo trovare un numero **D** che sia l'inverso di **E** mod _(n)**,** e che quindi E□D mod **_(n)** = 1.
Utilizzando l'algoritmo di Euclide troviamo che **E** = 7 infatti 7□7 = 49 mod 24 = 1.
Come si vede per la cifratura si devono conoscere **E** ed **n** (chiave pubblica), mentre per decifrare è necessario conoscere **D** ed **n** (chiave privata).
E' importante sottolineare che questi passaggi servono per creare la coppia di chiavi di un solo utente a differenza del protocollo Diffie-Hellman in cui entrambi gli interlocutori concorrono alla creazione della medesima chiave segreta, infatti come abbiamo visto questa risultava essere identica. Nell'algoritmo RSA ciò non avviene, infatti ogni utente che vuole utilizzare questo sistema deve compiere da solo questi passi, ottenendo così una coppia di chiavi esclusiva. In Diffie-Hellman inoltre la chiave segreta che si otteneva era utilizzabile esclusivamente da quella coppia di utenti che l'aveva creata. Se uno dei due avesse voluto comunicare con una terza persona avrebbe dovuto generare una nuova chiave.

Dopo aver calcolato le chiavi può avvenire la cifratura dei documenti. I testo in chiaro viene visto come una stringa di bit e viene diviso in blocchi costituiti da k-bit, dove k è il più grande intero che soddisfa la disequazione $2^k < n$.
Per ogni blocco M così ottenuto si ha che:
Cifratura: Il blocco cifrato è uguale al blocco in chiaro elevato alla E modulo n (dove E ed n costituiscono la chiave pubblica) → $\mathbf{M_C = M^E \bmod n}$.
Decifratura: Il blocco in chiaro è uguale al blocco cifrato elevato a D mod n (dove D ed n costituiscono la chiave privata) → $\mathbf{M = M_C^D \bmod n}$.

Funzioni e algoritmi Hash nella firma digitale

Quadro generale

Le funzioni e gli algoritmi di hash sono il punto cruciale del meccanismo di firma digitale, perché dalla loro robustezza dipende la sicurezza dell'intero processo di firma digitale.
Vengono utilizzati per produrre un'impronta del messaggio che si vuole inviare, che prende il nome di **Message Digest**. Questa impronta deve avere una lunghezza prestabilita e soprattutto deve essere necessariamente **non invertibile**.
Infatti, avendo una lunghezza prestabilita, breve rispetto al messaggio, l'impronta *h(m)* che viene generata a partire da un messaggio *m* è meno costosa da trasmettere rispetto allo stesso messaggio ed inoltre è più semplice da manipolare, ad esempio allo scopo di cifrarla.

La non invertibilità è la caratteristica più importante dell'impronta, e deriva dal fatto che le funzioni utilizzate per il suo calcolo sono funzioni **one-way**. Questo ha come prima conseguenza il fatto che non esiste un algoritmo noto che, dato un digest, sia in grado di generare un messaggio che gli corrisponda. Inoltre bisogna anche osservare che è estremamente difficile produrre un messaggio che abbia un digest predeterminato, e quindi è arduo il tentativo di trovare delle collisioni di messaggi diversi sulla stessa impronta.

Per la delicatezza delle applicazioni che ne fanno uso le funzioni hash crittografiche devono soddisfare requisiti di sicurezza particolarmente stringenti. In primo luogo devono essere **Strongly Collision Resistant**, ovvero, come già osservato, devono fornire un meccanismo che crei una corrispondenza one-way tra messaggio e valore hash relativo. Purtroppo, per natura stessa del problema, Si deve mappare un numero infinito di messaggi su un insieme finito di possibili impronte (2^n dove n è il numero di bit del digest), per cui è certo che vi saranno delle collisioni. La validità di una funzione o di un algoritmo di hash è misura di quanto questo è in grado di ridurre ad un valore irrisorio la probabilità che queste si verifichino.

Storicamente gli algoritmi più utilizzati e studiati, nell'ambito delle applicazioni di firma digitale, sono **MD4**, **MD5**, **SHA-1** e **RIPEMD-160**. Attualmente Solo SHA-1 e RIPEMD-160 sono validi per la legge italiana.

Ciononostante è interessante introdurre qualche concetto relativo a MD5, perché dagli studi su questo algoritmo hanno avuto origine i progetti che hanno in seguito portato alla definizione degli altri algoritmi che attualmente vengono utilizzati nella pratica.

Algoritmo MD5

Ideato nel 1991 da uno degli autori di **RSA (Rivest)** è uno dei più noti algoritmi per la produzione di digest. Genera impronte di 128 bit su messaggi di lunghezza arbitraria, articolandosi in cinque fasi.
Recentemente sono stati sollevati dubbi sulla sua robustezza. In particolare il ricercatore **Hans Dobbertin** ha mostrato una tecnica che permetterebbe di trovare due versioni dello stesso messaggio che differiscono per un solo carattere ma che producono lo stesso digest MD5.
La pericolosità di un simile risultato è evidente, e questo giustifica la necessità di studiare algoritmi differenti per la generazione delle firme digitali.

L'algoritmo MD5 utilizza cinque fasi per generare l'impronta di un messaggio:

1. **Aggiunta dei bit di riempimento**
   Il messaggio viene portato tramite **padding** ad una lunghezza congrua a 448 mod 512. Il padding è una sequenza di bit costituita da un 1 seguito da tutti zeri

2. **Aggiunta della lunghezza**
   Viene aggiunta al messaggio una rappresentazione a 64 bit della lunghezza iniziale, prima del padding. Il risultato è un messaggio perfettamente divisibile in blocchi da 448+64=512 bit

3. **Inizializzazione del buffer MD (initial variable/chaining variable)**
   Si tratta di un buffer di 4 words (A,B,C,D) da 32 bit, collegate in una catena, ciascuna con valori predefiniti:
   A=01234567 B=89abcdef C=fedcba98 D=76543210

4. **Elaborazione del messaggio (compression function)**
   La fase centrale dell'algoritmo applica delle trasformazioni sulle 4 variabili di chaining in base a 4 funzioni che vengono applicate alternativamente in base a predicati booleani sui valori dei singoli bit dei blocchi del messaggio in ingresso (quindi non secondo una logica predeterminata)

5. **Output**
   Viene ricavato dopo aver processato tutti i blocchi del messaggio leggendo a partire dal bit meno significativo le 4 parole (A,B,C,D) precedentemente trasformate.

Lo Standard SHS

Dall'esigenza di maggiore sicurezza generata dalle preoccupanti ricerche di Dobbertin sulle debolezza di MD5 nasce nel 2002 il **Secure Hash Signature Standard (SHS)**, ad opera della **Federal Information Processing Standards.**

Il documento prodotto definisce quattro algoritmi sicuri **(SHA-1, SHA-256, SHA-384 e SHA-512)** per la rappresentazione condensata di messaggi.
I quattro algoritmi, che seguono procedimenti relativamente simili, differiscono per dimensioni dei blocchi processati (da 512 a 1024 bit), lunghezza dell'impronta di output (da 160 a 512 bit), e funzioni applicate.

SHA-1

E' il primo e maggiormente diffuso degli algoritmi dello standard SHS, nonché quello attualmente utilizzato nella procedura di firma digitale. Elabora il messaggio in ingresso in due fasi:

Preprocessamento

Il messaggio viene preparato affinché possa essere inviato al motore di compressione in un formato ad esso noto. Questo risultato è perseguito attraverso procedure di **padding** e **parsing**. Come nel caso dell'algoritmo MD5 il messaggio viene incrementato con una sequenza di bit riconoscibile, in modo da portarlo ad avere una dimensione che sia un multiplo di 512 bit, la dimensione del blocco processato dall'algoritmo.
Successivamente Il messaggio viene scomposto in blocchi da 512 bit, ciascuno rappresentato come 16 parole da 32 bit.

Preprocessamento

$H_0^{(0)} = 67452301$
$H_1^{(0)} = efcdab89$
$H_2^{(0)} = 98badcfe$
$H_3^{(0)} = 10325476$
$H_4^{(0)} = c3d2e1f0.$

Una volta completate le operazioni di padding e parsing vengono inizializzate con valori predefiniti 5 parole da 32 bit, che verranno usate come variabili di transizione nei passaggi algoritmici di calcolo del message digest.
Queste parole sono l'equivalente delle variabili di chaining dell'algoritmo MD5, ma vengono qui utilizzate in maniera più sofisticata.

Hash Computation
Con queste premesse si passa alla fase di Hash Computation sui blocchi del messaggio, che si svolge in **4 step**. La complessità del meccanismo è determinante per la sicurezza e per l'affidabilità dell'algoritmo.

Step 1: Message schedule
In primo luogo viene calcolato un **"message schedule"** da applicare per il calcolo del valore hash. In sostanza per ciascun blocco del messaggio ad ogni iterazione vengono prodotti 80 valori dello schedule che concorreranno a determinare le operazioni che saranno svolte in seguito nelle varie iterazioni.
Si può pensare a questi valori come operatori che modificano il **piano strategico** delle operazioni da effettuare, da cui il termine schedule.
Tramite questo meccanismo le trasformazioni subite dalle variabili H non sono fisse, ma vengono influenzate dai blocchi del messaggio, pertanto si ha un sensibile aumento della sicurezza.
Siano $\mathbf{M_i}$ i blocchi del messaggio dopo il preprocessamento. Quando un blocco $\mathbf{M_i}$ attraversa lo **scheduler** alla t-esima iterazione viene calcolato il valore dello schedule $\mathbf{(W_t)}$ da utilizzare nella t-esima iterazione del terzo step, relativo al blocco $\mathbf{M_i}$ secondo questa procedura:

$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ ROTL^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) & 16 \le t \le 79 \end{cases}$$

Calcolo dei valori di Schedule

L'operatore $ROTL^n(x)$ indica uno shift circolare a sinistra di n posizioni.

Step 2: Aggiornamento delle variabili di supporto
Aggiorna 5 variabili di supporto (a,b,c,d,e) con i valori (i-1)esimi delle 5 variabili H, dove i indica il numero del blocco che si sta analizzando.
In pratica vengono memorizzati, per modificarli nel passo successivo, i valori intermedi delle 5 parole di hash restituiti dall'esecuzione dell'algoritmo sul blocco precedente.

$$a = H_0^{(i-1)}$$
$$b = H_1^{(i-1)}$$
$$c = H_2^{(i-1)}$$
$$d = H_3^{(i-1)}$$
$$e = H_4^{(i-1)}$$

aggiornamento delle variabili di supporto

Step 3: Fase di compressione
Iterando da 0 a 79 vengono eseguite delle operazioni sulle variabili di supporto. K_t è la costante per il calcolo dell'hash da utilizzare all'iterazione t.
In questo passo si vede come vengono modificate ad ogni iterazione le variabili di supporto dalle quali verrà calcolato il valore hash parziale.

$$T = ROTL^5(a) + f_t(b,c,d) + e + K_t + W_t$$
$$e = d$$
$$d = c$$
$$c = ROTL^{30}(b)$$
$$b = a$$
$$a = T$$

Fase di compressione

Si noti che intervengono insieme le costanti, il valore corrente dello schedule, i precedenti valori delle parole di hash e soprattutto la funzione f, così definita:

$$f_t(x,y,z) = \begin{cases} Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z) & 0 \le t \le 19 \\ Parity(x,y,z) = x \oplus y \oplus z & 20 \le t \le 39 \\ Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) & 40 \le t \le 59 \\ Parity(x,y,z) = x \oplus y \oplus z & 60 \le t \le 79. \end{cases}$$

Funzione caratteristica

che prende in input 3 parole da 32 bit e restituisce una sola parola da 32 bit utilizzando 4 differenti funzioni di logica booleana sui bit basate sull'operatore XOR.

Step 4: Generazione dell'output
Calcola i cinque valori hash ottenuti in seguito all'elaborazione del blocco corrente, dati dalla somma delle variabili di supporto (a,b,c,d,e) e dei rispettivi valori hash del ciclo precedente

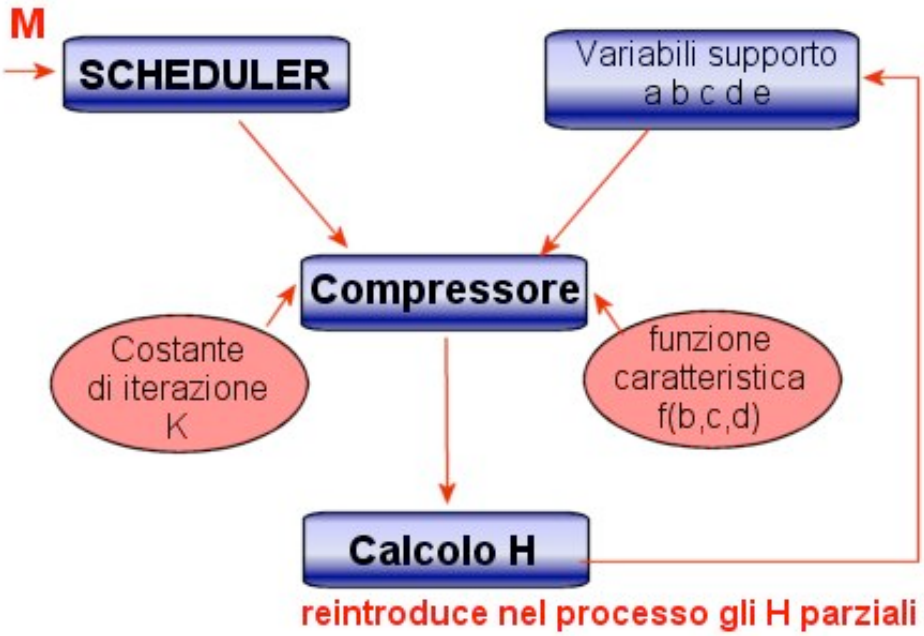
$$H_0^{(i)} = a + H_0^{(i-1)}$$
$$H_1^{(i)} = b + H_1^{(i-1)}$$
$$H_2^{(i)} = c + H_2^{(i-1)}$$
$$H_3^{(i)} = d + H_3^{(i-1)}$$
$$H_4^{(i)} = e + H_4^{(i-1)}$$

calcolo dell'hash parziale

Una volta ripetuti i passi da 1 a 4 per un numero di volte N pari al numero di blocchi del messaggio, si ottiene l'impronta dalla concatenazione dei valori delle 5 variabili hash restituiti dall'ultima iterazione.

$$H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)} .$$

Generazione dell'output



Panoramica

SHA-1 vs MD5

Come già accennato in precedenza per motivazioni legate alla sicurezza si è ritenuto opportuno sostituire nell'ambito delle applicazioni di firma digitale l'algoritmo MD5 con soluzioni più efficaci, in testa SHA-1. A questo proposito è interessante fornire un parallelo tra i due algoritmi:

MD5	SHA-1
• Output di 128 bit • 5 round di elaborazione • Le operazioni sulle variabili di chaining vengono modificate tramite predicati sui bit del blocco	• Output di 160 bit • 80 cicli di elaborazione per ciascun blocco • Le operazioni sulle variabili di chaining vengono determinate con l'ausilio di uno schedule legato al contenuto dei blocchi del messaggio mediante una funzione complessa. Ad ogni iterazione la struttura del blocco influenza in modo differente le modifiche.

Sostanzialmente le differenze sono nella lunghezza dell'output, che di per sé è già una buona difesa, generalmente, contro tutti i tipi di attacchi, e nella mole delle elaborazioni effettuate sulle variabili di chaining. Inoltre SHA-1 sfrutta il meccanismo di schedule per aumentare il numero di possibili varianti della procedura di compressione in una singola iterazione, garantendo una difesa migliore contro gli attacchi parziali.

RIPEMD-160

Da una delle molteplici linee di sviluppo originatesi a seguito degli studi relativi a MD4 e MD5 e ai relativi punti deboli, nasce nel 1996 la versione potenziata dell'algoritmo RIPEMD sviluppato nell'ambito del progetto **RIPE (RACE Integrity Primitives Evaluation)** ad opera di un gruppo di ricercatori, tra cui lo stesso Hans Dobbertin (l'autore del metodo di attacco ad MD5 di cui si è parlato in precedenza). Rispetto all'originale RIPEMD si notano i seguenti cambiamenti:

1. La dimensione del digest viene portata a 160 bit.
2. Vengono aggiunti due round e modificate le operazioni booleane bit a bit
3. La velocità di esecuzione è molto minore rispetto agli antenati MD e persino rispetto a SHA-1

RIPEMD-160 è organizzato in 5 round che operano su 5 parole da 32 bit. In ciascun round vengono applicate le seguenti tipologie di operazioni:

1. **Shift**

Round	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	15	12	15	14	9	9	8	6
5	15	12	13	13	9	5	8	6	14	11	12	11	8	6	5	5

2. **Funzioni booleane bit a bit**

$$f_1(x, y, z) = x \oplus y \oplus z,$$
$$f_2(x, y, z) = (x \wedge y) \vee (\neg x \wedge z),$$
$$f_3(x, y, z) = (x \vee \neg y) \oplus z,$$
$$f_4(x, y, z) = (x \wedge z) \vee (y \wedge \neg z),$$
$$f_5(x, y, z) = x \oplus (y \vee \neg z).$$

queste funzioni vengono applicate secondo questo schema:

Line	Round 1	Round 2	Round 3	Round 4	Round 5
left	f_1	f_2	f_3	f_4	f_5
right	f_5	f_4	f_3	f_2	f_1

3. **Ordinamento delle parole (secondo diverse permutazioni)**
4. **Addizione di costanti (le parti intere dei valori della tabella)**

Line	Round 1	Round 2	Round 3	Round 4	Round 5
left	0	$2^{30} \cdot \sqrt{2}$	$2^{30} \cdot \sqrt{3}$	$2^{30} \cdot \sqrt{5}$	$2^{30} \cdot \sqrt{7}$
right	$2^{30} \cdot \sqrt[3]{2}$	$2^{30} \cdot \sqrt[3]{3}$	$2^{30} \cdot \sqrt[3]{5}$	$2^{30} \cdot \sqrt[3]{7}$	0

Sicurezza degli algoritmi hash

La letteratura è ricca di documenti che descrivono e testimoniano attacchi agli algoritmi menzionati in questo lavoro, in particolare basati sulla possibilità di trovare due messaggi con lo stesso digest. Lo stesso algoritmo MD5 è stato messo in disuso dalla legislazione italiana esattamente per questo motivo, dopo le scoperte di Dobbertin che ne hanno messo in discussione la sicurezza.
Come è risaputo, la resistenza di una particolare funzione hash ad una serie di strategie di attacco "classiche" fornisce una buona misura del suo livello di sicurezza. Pertanto la prima domanda da porsi per stabilire se un algoritmo di hash sia più o meno valido è: come si comporta contro gli attacchi tradizionali?

Esistono diverse tipologie di attacchi che possono essere portati ad un algoritmo e possono essere suddivisi in due macrogruppi:

1. **Indipendenti dall'algoritmo**
   Es. birthday attack
2. **Legati alla natura degli algoritmi**
   Es. chaining attack

Analizzeremo in seguito i due esempi di attacco che sono maggiormente utilizzati in questo settore della crittoanalisi.

Yuval's Birthday attack
Questo attacco si basa sul noto paradosso del compleanno, secondo il quale, seguendo le leggi del calcolo delle probabilità, se si sceglie casualmente un elemento in un insieme di n elementi, con buona probabilità un elemento ripetuto verrà trovato dopo O(sqrt(n)) estrazioni.
Questo risultato è particolarmente significativo se applicato al caso delle funzioni hash one way, perché, come già ampiamente provato è più semplice trovare due collisioni piuttosto che cercare una controimmagine di un dato valore hash. L'attacco in questione è applicabile a tutte le funzioni hash senza chiave in un tempo $O(2^{m/2})$, dove m è la lunghezza dell'impronta.
Concretamente effettuare un attacco di questo tipo contro un meccanismo di firma digitale potrebbe avere la logica seguente:

1. Idea: posso ottenere la firma su un certo documento x1 originale, e voglio applicarla in seguito al documento x2, contraffatto, in modo che risulti firmato in modo autentico
2. Creo un insieme di messaggi ottenuti da piccole variazioni di x1, ne calcolo l'hash e lo memorizzo per una ricerca successiva
3. Allo stesso modo creo un insieme di messaggi varianti di x2
4. Cerco nel secondo insieme un messaggio che produca una delle impronte del primo insieme
5. Se il tentativo riesce posso richiedere una firma autentica per il documento x1 ed applicarla al documento x2

Ad ogni modo per un algoritmo che produce un digest di 128 bit, considerando una serie di 64 piccole modifiche, ad esempio sostituzioni di tabulazioni con spazi ed affini, l'attacco necessita di un tempo $O(2^{64})$ e spazio in memoria per $O(2^{64})$ messaggi, il che è pensabile solo avendo a disposizione numerosi calcolatori che lavorano in parallelo.

In alternativa agli attacchi generici sulle collisioni si possono utilizzare i cosiddetti **"chaining attack",** che sfruttano l'iteratività degli algoritmi di hash, ed in particolare l'uso che questi fanno delle **"chaining variables",** le parole che vengono modificate ad ogni iterazione per produrre il valore del message digest finale.

Questi attacchi si concentrano sulla funzione di compressione piuttosto che su tutto il processo di hash, e di conseguenza sono specifici per gli algoritmi su cui sono costruiti. Anche in questo caso è utile illustrare una possibile logica applicativa per un attacco di questo tipo:

1. Supponiamo di avere un algoritmo hash h con blocchi x_i di 512 bit, chaining variables H_i di 128 bit e una funzione di compressione f che produce un output $H_{i+1} = f(H_i, x_i)$ che dipende dal ciclo precedente
2. Dato un messaggio di 10 blocchi si può supporre di voler sostituire uno di questi blocchi senza modificare il valore di output dell'algoritmo
3. Se f si comporta come una distribuzione casuale il numero di blocchi in cui è possibile trovarne uno che abbia questa proprietà è $2^{512} / 2^{128} = 2^{384}$
4. Un qualsiasi metodo che consenta di trovare un blocco simile in modo efficiente rappresenta un potenziale attacco all'algoritmo

Si può pensare di applicare il metodo per sostituire ciascun blocco, e quindi per modificare l'intero messaggio, ottenendo un risultato simile a quello ottenuto nel caso di attacco del compleanno.

Attacchi efficaci sono anche quelli di analisi statistica delle impronte ottenute da messaggi simili, nei quali, sulla base di osservazioni sulle impronte, si applicano modifiche al messaggio originale, allentando il vincolo delle "piccole variazioni".
Nel febbraio 2005 alcuni ricercatori cinesi hanno individuato, pare utilizzando questa tecnica, una falla in SHA-1 che permetterebbe di ridurre di circa 2000 volte il tempo necessario per la ricerca di una collisione.
Per la natura stessa dell'attacco, di cui tra l'altro non sono stati pubblicati i dettagli, non è opportuno allarmarsi per la sicurezza della firma digitale.
Infatti le collisioni trovate mediante questa tecnica possono presentare modifiche rispetto al messaggio originale difficilmente camuffabili, come ad esempio sequenze di caratteri del tutto prive di senso o vistose alterazioni di formattazione, che possono essere identificate facilmente sul messaggio.

Ad esempio, è alquanto improbabile che una banca accetti il seguente messaggio:

> "**msg. Da Alice** #àà@@]++[ì?^ **a banca:** ç°è*?$)&&£ **versare** !"!"!??$)%(^§]#[**500 dollari sul conto di Bob**"

Questo perché i messaggi importanti solitamente hanno un formato prestabilito, che riduce al minimo le variazioni accettate, per cui il messaggio stesso è la sua prima arma di difesa da un attacco.

Inoltre Il calcolo necessario per la ricerca di una collisione utilizzando una tecnica simile richiederebbe comunque circa 1000 anni con un PC di ultima generazione.

Ad ogni modo, poiché, come è noto, una falla in un algoritmo di hash è una finestra sulla quale si affacciano nuove sperimentazioni di tecniche sempre più raffinate per la sua violazione, si potrebbe decidere di rimpiazzare SHA-1, ad esempio con uno degli altri algoritmi dello standard SHS, per mantenere comunque un livello di sicurezza del tutto adeguato alle garanzie che la firma digitale deve offrire.

Il processo di firma digitale

Il processo di firma digitale richiede che l'utente effettui una serie di azioni preliminari necessarie alla predisposizione delle chiavi utilizzate dal sistema di crittografia su cui il meccanismo di firma si basa; in particolare occorre:

1. la registrazione dell'utente presso una Autorità di Certificazione (CA),
2. la generazione di una coppia di chiavi K_s e K_p,
3. la certificazione della chiave pubblica K_p,
4. la registrazione della chiave pubblica K_p.

Una volta espletate tali operazione l'utente è in grado di firmare elettronicamente un numero qualunque di documenti, sfruttando la sua chiave segreta K_s, durante il periodo di validità della certificazione. Tale periodo può essere interrotto prima del suo naturale termine dalla revoca della certificazione della chiave pubblica, che di norma viene effettuata su richiesta del proprietario nel caso in cui questi ritenga che la segretezza della sua chiave privata sia stata compromessa.

La firma viene apposta mediante una sequenza di tre operazioni:

1. generazione dell'impronta del documento da firmare,
2. generazione della firma mediante cifratura dell'impronta,
3. apposizione della firma al documento.

Azioni Preliminari

Registrazione dell'utente

La registrazione dell'utente presso una autorità di certificazione ha il duplice scopo di rendere questa certa della sua identità ed instaurare con esso un canale di comunicazione sicuro attraverso il quale verranno fatte viaggiare le chiavi pubbliche di cui viene richiesta la certificazione. All'atto della registrazione l'autorità di certificazione attribuisce all'utente un identificatore, di cui viene garantita l'univocità, attraverso il quale sarà possibile a chiunque reperire in modo diretto e sicuro i certificati rilasciati al soggetto corrispondente all'interno dei cataloghi pubblici in cui questi sono registrati.

La registrazione avviene attraverso la seguente procedura:

1. L'utente richiede alla CA la registrazione fornendo la documentazione richiesta da questa per accertare l'identità del richiedente.
2. Verificata la validità della richiesta, la CA attribuisce all'utente un identificatore di cui essa garantisce l'univocità.
3. La CA inserisce l'utente con l'identificatore attribuitogli nei cataloghi di utenti registrati che essa gestisce.
4. La CA fornisce attraverso un canale sicuro la chiave crittografica che l'utente dovrà utilizzare per le richieste di certificazione delle chiavi.

La necessità di utilizzare un canale sicuro asserita al punto 4 nasce dal fatto che, sebbene le richieste di certificazione contengono chiavi pubbliche per le quali non è richiesta una protezione ai fini della riservatezza, la CA deve essere certa che ciascuna richiesta provenga effettivamente dall'utente in essa indicato e non da un altro soggetto che lo sta impersonando. La segretezza della chiave, che di norma è una chiave pubblica e pertanto non dovrebbe essere protetta, viene qui utilizzata come strumento di autenticazione iniziale.

Generazione della coppia di chiavi

L'utente, mediante un programma adatto al sistema crittografico adottato, genera una coppia di chiavi da utilizzare una per la generazione della firma, che verrà mantenuta segreta e corrisponde perciò a K_s, e l'altra, destinata alla verifica, che verrà resa pubblica ed assume perciò il ruolo di K_p.

Certificazione della chiave pubblica

La certificazione della chiave pubblica ha lo scopo di rassicurare chiunque riceva un documento correttamente firmato circa l'identità del soggetto che ha apposto la firma. L'operazione avviene attraverso tre passi:

1. L'utente invia alla CA la richiesta di certificazione per la chiave K_p generata nella fase precedente, autenticandola mediante la chiave ricevuta dalla CA durante il processo di registrazione.
2. La CA genera il certificato e lo sottoscrive per garantirne la provenienza che potrà essere accertata da chiunque utilizzando la chiave pubblica della CA.
3. Il certificato viene inviato al richiedente.

Registrazione della chiave pubblica

Una volta emesso, il certificato può essere reso disponibile in uno o più cataloghi ai quali può accedere chiunque abbia bisogno di accertare la validità di una sottoscrizione digitale. Questa operazione viene di norma effettuata, almeno per i cataloghi di sua competenza, dalla CA contestualmente all’emissione.

Processo di Firma

Nella prima fase di *Generazione dell'impronta digitale*, viene applicata al documento in chiaro una funzione di hash appositamente studiata che produce una stringa binaria di lunghezza costante e piccola, normalmente 128 o 160 bit, chiamata *digest message*, l'impronta del documento. Queste funzioni devono avere due proprietà:

- unidirezionalità, ossia dato x è facile calcolare f(x), ma data f(x) è computazionalmente difficile risalire a x.
- prive di collisioni (collision-free), ossia a due testi diversi deve essere computazionalmente impossibile che corrisponda la medesima impronta, ovvero (se $x \neq y$ allora $f(x) \neq f(y)$)

Poiché la dimensione del digest message è fissa, e molto più piccola di quella del messaggio originale, la generazione della firma risulta estremamente rapida.
L'utilità dell'uso delle funzioni hash consente di evitare che per la generazione della firma sia necessario applicare l'algoritmo di cifratura, che è intrinsecamente inefficiente, in quanto molto lento, all'intero testo che può essere molto lungo.
Quindi la seconda fase, la *generazione della firma*, consiste semplicemente nella cifratura con la propria chiave privata dell'impronta generata il precedenza.
In questo modo la firma risulta legata, da un lato (attraverso la chiave privata usata per la generazione) al soggetto sottoscrittore, e dall'altro (tramite l'impronta) al testo sottoscritto.
Nell'ultima fase, *l'apposizione della firma al documento*, la firma digitale generata precedentemente viene aggiunta in una posizione predefinita, normalmente alla fine del testo del documento.

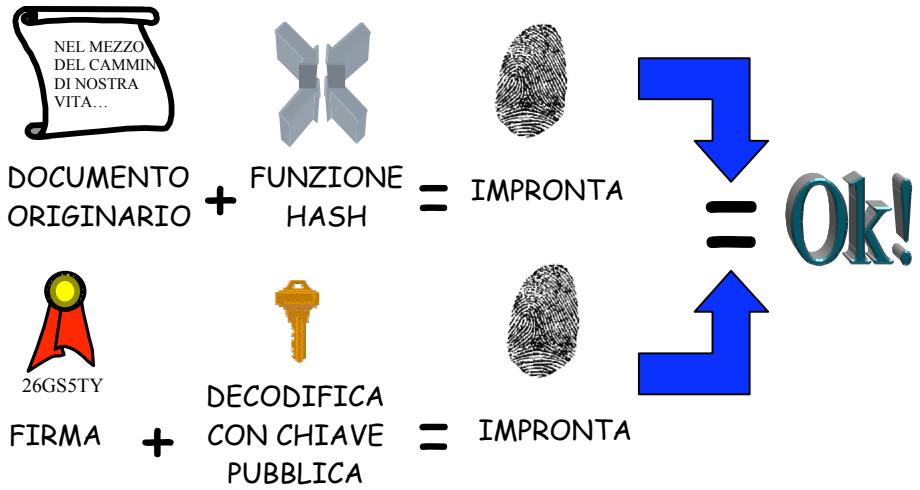
A questo punto possiamo capire come funziona la firma digitale, cioè come si può dare a un testo chiaro la certezza dell'identità del mittente e dell'integrità del contenuto, cioè avere la sicurezza che il documento non è stato modificato da nessuno dopo che il firmatario ha posto la propria firma.
Al documento viene applicata una determinata funzione, denominata "hash function", che produce un riassunto chiamato "impronta" di lunghezza fissa (20 caratteri), indipendentemente dalle dimensioni dell'originale.
L'impronta è unica, nel senso che modificando anche un solo carattere del testo si otterrà un'impronta diversa. L'impronta, e non l'intero documento, viene quindi criptata con la chiave privata del mittente in questo modo si ottiene la generazione della "firma digitale" che verrà apposta al documento originale.

Al destinatario vengono spediti:

- il documento in chiaro
- la "firma digitale" in calce al documento
- il certificato rilasciato dalla competente autorità di certificazione a garanzia della titolarità della chiave pubblica necessaria per decriptare la firma digitale

Chi riceve il messaggio procederà *all'apertura e/o verifica* dello stesso mediante l'operazione di verifica di una firma digitale. Le operazioni da eseguire sono:

1. Per prima cosa chi riceve il messaggio firmato si procura il certificato del mittente (spesso è in allegato al messaggio stesso) e, dopo averne controllato la validità, ne estrae la chiave pubblica che è contenuta.
2. Con questa chiave pubblica il ricevente può decriptare la firma digitale ed estrarre il digest che il mittente aveva calcolato per il messaggio.
3. A questo punto il ricevente calcola un suo digest per lo stesso messaggio, avendo cura di usare lo stesso sistema del mittente, per esempio SHA-1.
4. Il ricevente confronta i due digest, quello che ha appena calcolato e quello estratto dalla firma digitale: se sono uguali significa che il messaggio non è stato in alcun modo alterato durante la spedizione.
5. Il fatto poi che l'operazione di crittografia per estrarre il digest sia riuscita significa che esso era stato criptato, al momento della spedizione, con l'unica chiave privata corrispondente a quella pubblica contenuta nel certificato. Questo garantisce anche l'identità del mittente.



Possibili in casi in cui la verifica non va a buon fine sono:

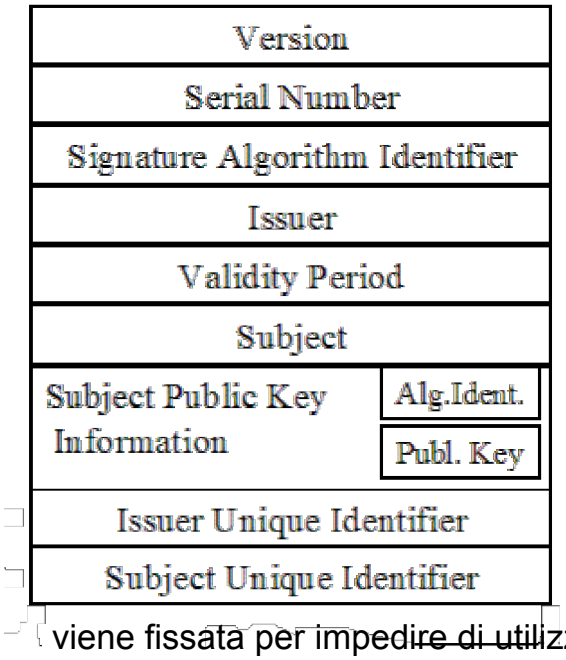
1. Il documento che ci arriva in realtà è stato modificato o addirittura cambiato da terze persone e quindi la seconda impronta che otteniamo non coincide con la prima
2. Il documento che ci arriva è stato inviato e quindi cifrato da terze persone utilizzando un chiave privata a cui non corrisponde la chiave pubblica che noi abbiamo di quel mittente, quindi la prima impronta che otteniamo non coincide con la seconda

Certificati

Nella tecnologia di crittografia a chiave pubblica, sia in fase di cifratura che in fase di verifica di una firma digitale, occorre ritrovare la chiave pubblica o del destinatario di un messaggio o del firmatario del messaggio firmato. In entrambi i casi il valore delle chiavi pubbliche non è confidenziale e la criticità del reperimento delle chiavi sta nel garantire l'autenticità delle chiavi pubbliche. Quindi si deve essere certi che una certa chiave pubblica appartenga effettivamente all'interlocutore per cui si vuole cifrare o di cui si deve verificare la firma. Se, infatti, una terza parte prelevasse la chiave pubblica del destinatario sostituendola con la propria, il contenuto dei messaggi cifrati sarebbe svelato e non si riuscirebbe a verificare la validità di una firma digitale.
La distribuzione delle chiavi pubbliche è il problema cruciale della tecnologia a chiave pubblica. In un dominio con un numero limitato di utenti si potrebbe anche ricorrere ad un meccanismo manuale di distribuzione delle chiavi: due interlocutori che abbiano una relazione di conoscenza già stabilita, potrebbero, ad esempio, scambiarsi reciprocamente le chiavi attraverso floppy disk. Meccanismi di distribuzione manuale diventano, tuttavia, assolutamente inadeguati ed impraticabili in dominio scalabile dove non c'è alcuna diretta conoscenza prestabilita tra gli interlocutori.
Il problema della distribuzione delle chiavi pubbliche è risolto tramite l'impiego dei certificati elettronici. I certificati a chiave pubblica costituiscono, infatti, lo strumento affidabile e sicuro attraverso cui rispondere ad esigenze di scalabilità; attraverso i certificati elettronici, le chiavi pubbliche vengono distribuite e rese note agli utenti finali con garanzia di autenticità ed integrità.
L'utilizzo dei certificati elettronici presuppone l'esistenza di una Autorità di Certificazione (Certification Authority, CA) che li emetta e li gestisca.
Le CA svolgono i seguenti compiti fondamentali:

- verificano ed attestano, emettendo un apposito certificato digitale, l'identità del titolare di una determinata chiave pubblica (e quindi della chiave privata corrispondente, che viene usata per "firmare" digitalmente)
- stabiliscono il termine di scadenza dei certificati, quindi il periodo di validità delle chiavi
- pubblicano il certificato e la chiave pubblica
- ricevono la segnalazione di eventuali smarrimenti, furti, cancellazioni, divulgazioni improprie, ecc. di chiavi private e pubblicano quindi la lista dei certificati revocati o sospesi in conseguenza di tali fatti

Ogni certificato è una struttura dati costituita da diverse parti. Lo standard più utilizzato è X.509 e i suoi campi sono:



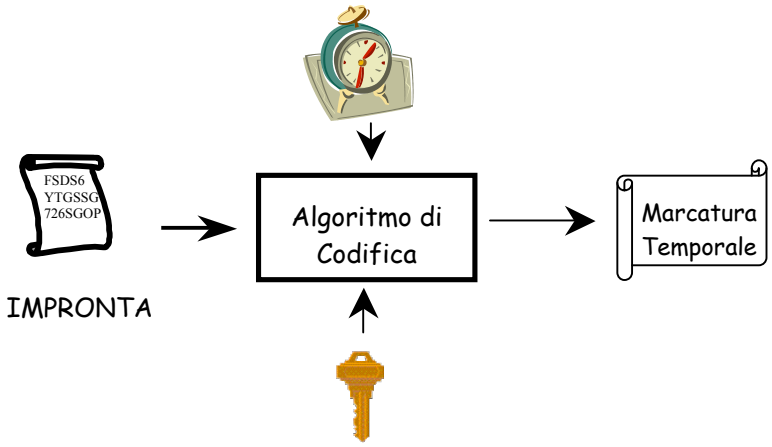
- ***Version**:* Indica la Versione dello standard X.509 applicata al certificato.
- ***Serial Number**:* Numero univoco assegnato dall'autorità di certificazione che lo ha emesso. Tale attributo, unito al nome dell'autorità che lo ha emesso, identifica univocamente un certificato.
- ***Signature Algorithm Identifier**:* Identifica l'algoritmo di firma digitale utilizzato dalla CA per firmare i certificati pubblici.
- ***Issuer**:* Nome X.500, della CA che emette i certificati.
- ***Validity Period**:* Data e ora d'inizio e fine validità del certificato. La data di scadenza si riferisce all'utilizzo della chiave privata e non di quella pubblica. Infatti questa data viene fissata per impedire di utilizzare per troppo tempo una chiave asimmetrica. La chiave pubblica può essere utilizzata anche successivamente alla scadenza per quei file che sono stati immagazzinati e di cui se ne vuole controllare la veridicità in un momento successivo
- ***Subject**:* Nome X.500 del soggetto a cui è rilasciato il certificato pubblico.
- ***Subject public key information**:* identifica sia la chiave pubblica dell'intestatario del certificato che l'identificativo dell'algoritmo di hashing e di crittografia pubblica con cui la chiave può essere utilizzata
- ***Issuer unique identifier**:* Una stringa di bit opzionale per identificare univocamente una CA nel caso che lo stesso nome sia stato assegnato ad un'altra entità.
- ***Subject unique identifier**:* Una stringa di bit opzionale per identificare univocamente il soggetto, Subject, nel caso che lo stesso nome sia stato assegnato ad un'altra entità.

Marcatura Temporale

Qualora sia necessario attribuire ad un documento certezza circa il momento in cui questo è stato redatto ed è divenuto valido, si ricorre alla sua marcatura temporale o Time Stamping. Questa consiste nella generazione da parte di una terza parte fidata, normalmente una CA, di una ulteriore firma digitale per il documento marcato.

L'operazione avviene secondo la seguente procedura:

1. L'impronta del documento viene inviata al servizio di marcatura temporale, l'impronta costituisce un riferimento certo al testo originale ma non ne consente la ricostruzione, pertanto la marcatura può essere effettuata da una terza parte senza compromettere la confidenzialità del testo marcato.
2. Il servizio di marcatura aggiunge all'impronta ricevuta la data e l'ora, ottenendo una "impronta marcata".
3. L'impronta marcata viene cifrata con la chiave segreta del servizio, ottenendo la marca temporale da cui è possibile recuperare, mediante la chiave pubblica del servizio, l'impronta del documento e la data e l'ora della sua generazione.
4. La marca temporale viene inviata al richiedente il quale la allega al documento.



Dimostrazione pratica

E' stata realizzata una piccola dimostrazione pratica dell'utilizzo della firma digitale utilizzando uno specifico dispositivo di firma.

Per la normativa italiana con dispositivo di firma si intende:

> *"un apparato elettronico programmabile solo all'origine, facente parte del sistema di validazione, in grado almeno di conservare in modo protetto la chiave privata e generare al suo interno le firme digitali"*

Uno strumento che è possibile utilizzare come dispositivo di firma è la **smart card crittografica.**
La smart card è simile ad una tradizionale carta di credito ma incorpora un processore in grado di memorizzare dati ed informazioni a cui è possibile accedere tramite un codice di sicurezza (PIN)
E' in grado di eseguire:

- un algoritmo di inizializzazione in grado di generare e memorizzare stabilmente una coppia di chiavi pubblica/privata
- un algoritmo di cifratura asimmetrica in grado di cifrare i dati in ingresso con la chiave privata memorizzata internamente

Utilizzando uno specifico software "Firma e Cifra Rupa" è possibile firmare e verificare la validità di documenti passati come input.

Abbiamo prima di tutto fatto vedere com'è possibile firmare un documento:

- si seleziona il documento da firmare
- ci viene richiesto di inserire un PIN che ci è stato spedito tramite raccomandata e che permette l'accesso alla smart card e alla nostra chiave privata memorizzata su di essa
- ci viene presentata una lista da cui è possibile scegliere il certificato con cui vogliamo firmare il documento
- è possibile visualizzare i dettagli del certificato ed in particolare abbiamo fatto vedere i campi di cui avevamo parlato per lo standard X.509, tra cui: numero di serie, ente che ci ha rilasciato il certificato, richiedente del certificato, periodo di validità di questo e tipo di algoritmo a chiave pubblica utilizzato per cifrare l'impronta (RSA a 1024 bit)
- ci viene richiesto di salvare il documento aggiungendo l'estensione .p7m che è l'estensione tipica dei documenti firmati
- l'operazione è conclusa con la visualizzazione del documento firmato a cui è stato allegato il certificato da noi scelto in precedenza

Come secondo passo abbiamo fatto vedere invece il processo di verifica di un documento firmato:

- si seleziona il documento di cui si vuole verificare la validità
- ci viene richiesto di inserire la password dell'Archivio dei Certificati, che essendo un archivio pubblico, è una password nota a tutti

- viene visualizzato il certificato che è allegato al documento con tutti i dettagli menzionati precedentemente, tra cui il nome del firmatario, la validità e non ripudiabilità del documento
- ci viene richiesto di salvare il documento. A questo punto siamo sicuri della integrità ed autenticità del documento .