# 1 Lezione del 19-11-25

## 1.1 Integrità dei messaggi

Veniamo quindi a discutere il secondo grande tema della sicurezza in reta, cioè l'**integrità dei messaggi** trasmessi.

In questo caso non ci è di particolare interesse lo *sniffing* dei messaggi, e quindi nemmeno la loro cifratura (assumiamo di trasmettere dati che tutti possono leggere in sicurezza). Piuttosto, vogliamo impedire che ci siano *manomissioni* del messaggio in transito.

In particolare, vogliamo fornire alle parti comunicanti la possibilità di verificare che i messaggi ricevuti siano autentici, cioè:

- Il sorgente del messaggio sia chi si pensa che sia (il problema dell'*autenticazione*);
- I contenuti del messaggio non siano stati manomessi;
- Il messaggio non sia stato replicato da messaggi precedenti;
- La sequenza di messaggi venga mantenuta.

### 1.1.1 Digest

Per verificare l'integrità dei messaggi ci poniamo di realizzare una sorta di *"impronta digitale" digitale* (heh) di lunghezza fissa e veloce da calcolare.

L'approccio banale è quello di applicare una certa funzione hash $H$ al messaggio $m$, per ottenere $H(m)$, che chiamiamo anche **digest** del messaggio.

Le proprietà della funzione di hash $H$ che cerchiamo dovranno essere le seguenti:

- **Veloce** da calcolare;
- **Irreversibile**: dato $x = H(m)$, deve essere computazionalmente impossibile ricavare $m$;
- Resistente alle **collisioni**, cioè dati $(m, H(m))$ deve essere computazionalmente impossibile trovare $m'$ che abbia $H(m') = H(m)$ (si verifichi una *collisione*);
- L'output deve sembrare apparentemente **casuale**.

Notiamo che con *computazionalmente impossibile* intendiamo dire che può succedere, ma con probabilità estremamente bassa.

Vediamo l'esempio di alcuni funzioni di hash usate nei digest:

- Notiamo che in qualche modo la **checksum Internet** (quella che troviamo nei pacchetti IP) ha alcune proprietà di una funzione di hash: produce digest di 16 bit (calcolati come somme) per messaggio. Il problema è che è una funzione di hash molto scadente: dato un messaggio $m$ è molto facile trovare un secondo messaggio $m'$ con la stessa checksum Internet (basta scambiare parole da 16 bit o aggiungere valori modulo $2^16$ a parole da 16 bit);
- Un'approccio più utilizzato è **MD5** (*Message Digest 5*), che genera digest su 128 bit attraverso un approccio con 4 passaggi;
- Anche **SHA-1** (*Secure Hashing Algorithm 1*) è molto usato, che genera digest su 160 bit (solitamente rappresentati come esadecimali su 40 cifre).

### 1.1.2 MAC

Vediamo quindi i **MAC** (*Message Authentication Code*), da non confondere col MAC del *Medium Access Protocol*.

Questi sono codici associati a messaggi che permettono di autenticare il mittente del messaggio e verificare l'integrità del messaggio. Abbiamo già visto come la crittografia può risolvere questo problema: vediamo però un modo basato sulle funzioni di hash.

Introduciamo l'idea del *segreto condiviso*, che non va confusa ma è in qualche modo parallela alla *chiave condivisa* degli algoritmi di cifratura a chiave condivisa. Questo sarà semplicemente rappresentato da una stringa di bit nota solamente alle 2 parti coinvolte nella comunicazione.

1. Ciò che si fa è quindi appendere il segreto al messaggio da inviare, e calcolare con un algoritmo di hash un codice (appunto il MAC) a partire da questo valore. Tale MAC viene quindi appeso al messaggio stesso, che viene trasmesso sul canale inaffidabile.

2. Il ricevitore può quindi estrarre il messaggio vero e proprio da quanto riceve sul canale inaffidabile, e ricalcolare il MAC esattamente come aveva fatto il trasmettitore. A questo punto basta confrontare il valore ottenuto col codice MAC ricevuto per verificare che il messaggio non sia stato manomesso.

### 1.1.3 HMAC

**HMAC** (*Hash-Based MAC*) è uno standard MAC molto popolare: può usare sia MD5 che SHA-1. Il suo funzionamento è il seguente:

1. Si concatena il segreto all'inizio del messaggio: $[s|m]$;

2. Si calcola l'hash del messaggio concatenato: $H([s|m])$;

3. Si concatena il segreto all'inizio del digest: $[H([s|m])|m]$;

4. Si ricalcola l'hash di questa combinazione: $H([H([s|m])|m])$.

### 1.1.4 Replicazione di messaggi

Quando si è stabilito un metodo di verifica dell'integrità dei messaggi funzionante, resta comunque il problema della **replicazione dei messaggi**: non abbiamo modo di distinguere se un messaggio ottenuto e già ricevuto in precedenza è stato reinviato dalla sorgente che ci aspettiamo, o inserito da un terzo malintenzionato.

- Una soluzione che possiamo immaginare è quella di inserire in ogni messaggio in un timestamp. Il timestamp figurerà nel calcolo del MAC, per cui la sua trasmissione sarà affidabile, e assunto che gli host siano sincronizzati, si potranno scartare messaggi troppo vecchi come replicati.

- Un altro approccio è quello delle **OTP** (*One Time Password*), cioè codici richiesti dal destinatario che il mittente deve inserire nel messaggio che vuole inviare.

  Questo significa che le trasmissioni con OTP devono svolgersi in 3 fasi. Riprendiamo l'esempio di Francesca e Maurizio:

1. Se Francesca vuole inviare un messaggio a Maurizio, dovrà prima chiedere a Maurizio una OTP che Maurizio genererà, memorizzera e provvederà a fornire;
2. Una volta ottenuta l'OTP, Francesca la annetterà al messaggio (che secondo il metodo del MAC già visto verrà trasmesso in maniera sicura);
3. Ricevuto il messaggio con OTP, Maurizio potrà confrontare l'OTP ricevuto con quello generato in precedenza, e quindi validare il messaggio.

Intrusi come Giuseppina non potranno replicare il messaggio, in quanto dopo la prima trasmissione l'OTP verrà invalidata, e messaggi con la stessa OTP non verranno più accettati.

### 1.1.5 Firme digitali

Veniamo adesso ad una tecnica crittografica del tutto analoga alle firme scritte a mano: quella delle **firme digitali**.

Una firma digitale deve avere le seguenti caratteristiche:

1. **Verificabile**: il destinatario deve essere capace di provare che il mittente (noto), e solo il mittente ha firmato il documento;
2. **Non falsificabile**: la firma falsificata da un altro deve essere rilevata;
3. **Non ripudiabile**: il destinatario deve poter provare che il mittente ha firmato il documento $m$, e non un altro documento $m'$;
4. **Integra**: il mittente deve poter provare di aver firmato il documento $m$, e non un altro documento $m'$;

Il corrispettivo più vicino alla firma che abbiamo visto finora è il codice MAC con segreto. Il problema è che per il MAC è impossibile capire chi ha messo la firma fra le 2 parti coinvolte nella comunicazione (entrambi hanno accesso sia al segreto che alla funzione di hash). Effettivamente, l'unica caratteristica dei MAC fra le 4 che cerchiamo è la 4 (integrità), mentre le altre 3 non vengono soddisfatte.

Per risolvere il problema, reintroduciamo i concetti di chiavi pubbliche e private (visti in 23.5). Ciò che potremmo fare di base è criptare il messaggio $m$ con la chiave privata del mittente.

1. Tornando ai nostri esempi, Maurizio firma un documento con la sua chiave privata $K_B^-$, dando vita alla firma $K_B^-(m)$.
2. A questo punto Francesca potrà verificare che la firma è effettivamente di Maurizio usando la chiave pubblica, cioè calcolare:

$$m = K_B^+\left(K_B^-(m)\right)$$

Se l'uguaglianza risulta verificata, la firma è valida.

Abbiamo quindi soddisfatto tutte le caratteristiche, in quanto Francesca può verificare che:

- Maurizio ha firmato $m$;

- Nessun altro ha firmato $m$;
- Maurizio ha firmato $m$ e non $m'$;

che è la caratteristica 1. Inoltre la firma non si può ripudiare (solo Maurizio può generare $K_B^-(m)$, in quanto nessun altro ha $K_B^-$) (caratteristiche 2 e 3). Infine, Maurizio potrà dimostrare di aver firmato solo il documento $m$ e non il documento $m'$, che abbiamo detto è la caratteristica 4 che il MAC già ci assicurava.

Quello che si fa nella pratica, in verità, è calcolare le firme non su $m$ ma su hash di $m$, che ci risulta più funzionale. Dettagliamo quindi un ultima volta il processo:

1. Maurizio firma l'hash di un documento $m$ con la sua chiave privata $K_B^-$, dando vita alla firma $K_B^-(H(m))$.
2. A questo punto Francesca potrà verificare che la firma è effettivamente di Maurizio usando la chiave pubblica, cioè calcolare:

$$H(m) = K_B^+\left(K_B^-(H(m))\right)$$

   Ciò si può fare, in quanto sia Maurizio che Francesca hanno accesso a $m$ e alla funzione di hash $H$. Se l'uguaglianza risulta verificata, la firma è valida.

Notiamo quindi le differenze principali fra MAC e firme digitali:

- Nel MAC, ciò che facciamo e calcolare $m|s$ (concateniamo messaggio e segreto), prendere l'hash $H(m|s)$, e inviare $[m, H(m|s)]$, cioè messaggio e codice MAC. Il ricevitore, noto il segreto, può ricalcolare il MAC verificare;
- Nella firma digitale prendiamo invece l'hash $H(m)$ di $m$, e lo cifriamo con la chiave privata del mittente, ottenendo $K^-(H(m))$. La coppia $[m, K^-(H(m))]$ può quindi essere inviata.

La firma digitale è un metodo più pesante dei codici MAC: richiede **PKI** (*Public Key Infrastructure*), cioè infrastruttura a supporto della distribuzione affidabile di chiavi pubbliche.

### 1.1.6 Autorità di certificazione

Le **CA** (*Certification Authority*, o in italiano *autorità di certificazione*) sono enti che memorizzano e rilasciano **certificati digitali**. Un **certificato digitale** lega un dato ente $E$ alla sua chiave pubblica $K^+$: per avere conferma che una data chiave pubblica $K^+$ appartiene all'ente $E$ ci si può rivolgere al CA.

Ciò che il CA fa nella pratica è dettagliato in seguito:

1. Il CA riceve da chi vuole registrare la sua chiave pubblica una *prova di identità*;
2. Quando l'identità e confermata, il CA firma la chiave pubblica $K^+$ con la sua chiave privata $K_{CA}^-$. La chiave pubblica firmata dal CA rappresenta quindi il *certificato digitale*, che conferma che tale chiave appartiene all'ente che la ha depositata.

Chi vuole confermare un certificato quindi, può semplicemente richiedere la chiave pubblica del CA dal CA stesso (cosa che ci aspettiamo un CA sappia fare), e applica tale chiave al certificato digitale. Se si ottiene una copia esatta della chiave associata al certificato, tale chiave è valida.

## 1.2 Autenticazione

Andiamo quindi a dettagliare il problema dell'**autenticazione**. Ciò che vogliamo stabilire è un *protocollo* che permetta ad un soggetto di *autenticarsi* con un altro soggetto, cioè verificare la sua identità.

- Un approccio banale potrebbe essere quello di usare pacchetti contenenti l'identità del mittente.

  Coi nostri esempi, Francesca invia un messaggio a Maurizio contenente la stringa *"Sono Francesca"*. Questo approccio però non è sicuro, in quanto facilmente replicabile da Giuseppina.

- Un approccio leggermente migliore è quello di inviare un pacchetto contenente la stringa *"Sono Francesca"* e l'IP di Francesca. Anche questo però è facilmente replicabile da Giuseppina secondo la tecnica dell'*IP-spoofing*;

- Introduciamo quindi una *password*. Inviamo l'IP di Alice (per identificazione), la password (che non vorremo inviare in chiaro, ma cifrare secondo un algoritmo noto sia a Maurizio che a Francesca), e la stringa di autenticazione. Nemmeno questo approccio, però, è valido, in quanto a Giuseppina basterà replicare esattamente il messaggio inviato da Francesca per potersi autenticare con Maurizio;

-