

# Sicurezza II

Prof. Dario Catalano

---

Breve introduzione alla  
Crittografia



# Introduzione

---

- La crittografia nasce come mezzo di protezione della comunicazione.
- Due utenti Alice e Bob, vogliono comunicare attraverso un canale.
- Se tale canale fosse: dedicato, impenetrabile e non spiabile, saremmo a posto!
- Purtroppo però un canale del genere non esiste.



# Obiettivi

---

- Isolare alcune delle caratteristiche del suddetto canale e cercare di realizzarle in pratica.

**Goal no.1 *Privacy*.** Il contenuto della comunicazione deve rimanere "sconosciuto" all'avversario.

**Goal no.2 *Autenticita' e Integrita'*.** Bob deve essere sicuro che cio' che sembra arrivare da Alice arrivi effettivamente da Alice.

La crittografia si occupa di fornire ad Alice e Bob tutta una serie di algoritmi che permettano loro di realizzare tali obiettivi, in presenza di un *avversario*.



# Osservazioni

---

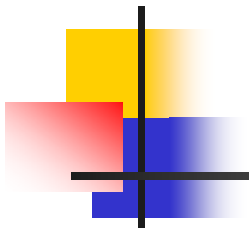
- Se l'avversario ha le stesse capacità / conoscenze / possibilità di accesso di Alice e Bob nessuna forma di sicurezza è possibile.
- Alice e Bob devono avere qualche informazione sconosciuta all'avversario.
- Due modelli fondamentali: *crittografia simmetrica* (o a chiave privata) e *crittografia asimmetrica* (o a chiave pubblica)



# Chiavi e Algoritmi

---

- I sistemi crittografici coinvolgono tipicamente un algoritmo e una chiave.
- In nessun caso si mantiene segreto l'algoritmo
  - Nascondere algoritmi sembra piuttosto complicato.
  - Sembra piu' difficile scambiare un algoritmo che scambiare una chiave.

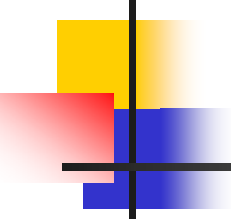
- 
- 
- Si potrebbe pensare che mantenere segreti anche gli algoritmi "aumenta" la sicurezza.
  - In realta' basare la sicurezza di un sistema nascondendo il metodo usato non e' una buona pratica
    - Il metodo potrebbe essere a conoscenza dell'avversario
    - Lasciare che i crittografi tentino di romperlo e' un buon modo per testarne le qualita'.



# Crittografia Simmetrica

---

- Alice e Bob si incontrano per stabilire una chiave  $k$
- Nel caso simmetrico  $k$  e' sovente una stringa casuale di bit
- Assumiamo che tale stringa sia tenuta al sicuro da chi la utilizza.

- 
- 
- L'approccio crittografico e' quello di procedere per piccoli passi
    - Si individuano un certo numero di problemi (che sembrano) difficili da risolvere
    - Si costruiscono sistemi sicuri sull'ipotesi che tali problemi siano davvero difficili

**Filosofia di base:** Se tante persone brillanti non sono riuscite a risolvere un problema, allora tale problema non verra' risolto (presto).



# Cifrari Simmetrici

---

Definiti da tre algoritmi

- L'algoritmo di generazione della chiave.
- L'algoritmo di cifratura.
- L'algoritmo di decifratura.

**Nota:** Non viene mai specificato il comportamento di chi attacca il sistema (avversario)



# Il ruolo dell'avversario

---

- Gli schemi non ci dicono cosa fa chi li attacca.
- Per analizzare la sicurezza di un sistema dobbiamo capire quali sono gli obiettivi e gli attacchi da fronteggiare.
- Dobbiamo anche capire quante risorse ha a disposizione l'avversario.

# Il ruolo dell'avversario (cont.)



---

- L'avversario e' il vero protagonista della crittografia.
- In crittografia si immagina spesso di fronteggiare avversari quanto piu' potenti possibile.
- L'approccio pessimistico e' utile.
- Se nemmeno un avversario bravo riesce a rompere un sistema, possiamo dormire sonni tranquilli.



# Obiettivi

---

- Nel caso dei cifrari l'obiettivo dell'avversario e' ricavare il messaggio (interamente o in parte) corrispondente a un dato crittotesto.
- Il modo in cui questi realizza tale obiettivo e' abbastanza irrilevante.
- E' facile stabilire condizioni necessarie alla sicurezza.
- E' piu' difficile definire condizioni sufficienti



# Che limitazioni porre all'avversario?

---

- Non possiamo stabilire come verrebbe attaccato il nostro sistema crittografico.
- Possiamo limitare le risorse disponibili all'avversario.
  - Tempo di calcolo
  - Coppie messaggi-crittotesti.
  - ...



# Che limitazioni porre all'avversario? (cont)

---

- A seconda dei limiti imposti all'avversario (e dei suoi obiettivi) possono distinguersi diverse nozioni di sicurezza.
- Quanto piu' deboli sono gli obiettivi e le limitazioni tanto piu' forte e' la nozione di sicurezza.



# Cifrari a blocchi

---

- La componente di base dei cifrari simmetrici sono i cifrari a blocchi.
- Un cifrario a blocchi e' qualcosa che prende in input una chiave (di lunghezza fissata), un messaggio (di lunghezza fissata) e produce un crittotesto (di lunghezza fissata).

Esempi: AES, DES



# Cifrari a blocchi (cont.)

---

- I CB sono la componente essenziale dei cifrari simmetrici.
- Non possono essere utilizzati direttamente per cifrare.

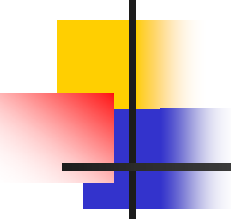


# Cifrari simmetrici da cifrari a blocchi.

---

- I cifrari simmetrici sono per lo piu' degli algoritmi che utilizzano cifrari a blocchi per realizzare determinate nozioni di sicurezza.
- Tali algoritmi vengono detti modi d'operazione.

Idea: Partendo dall'ipotesi che il CB abbia determinate proprieta' (semplici) si costruisce un cifrario che ha proprieta' di sicurezza piu' complesse.



# Cifrari simmetrici da cifrari a blocchi (cont.)

---

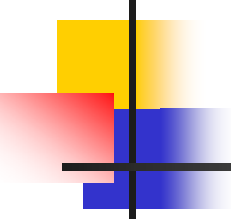
- Questo approccio e' molto utile:
  - Consente un approccio modulare al problema
  - Permette di limitare le fonti di errore.
  - Permette di costruire strumenti complessi da primitive piu' semplici.



# Modi d'operazione

---

- Esistono svariati modi d'operazione.
- Alcuni dei piu' noti sono i seguenti
  - Electronic Code Book (ECB)
  - Cipher Block Chaining (CBC)
  - Counter Mode (CTR)
- Non entreremo nei dettagli.



# Message Authentication

---

Problema: Bob riceve un messaggio che sembra provenire da Alice.

Goal: Bob vuole assicurarsi che esso provenga effettivamente da Alice.

- Problema intrinsecamente diverso da quello di proteggere i contenuti.



# Autenticita' vs Privacy

---

- L'obbiettivo storico della crittografia e' garantire privacy.
- Garantire autenticita' e' forse ancora piu' importante in pratica.
- Le primitive per realizzare autenticita' sono simili, ma sono utilizzate in modo molto diverso.



# Message Authentication Codes (MAC)

---

- Nel caso simmetrico, il problema dell'autenticita' puo' essere risolto utilizzando i cosiddetti MAC.

Tre algoritmi,

- **KeyGen**: algoritmo di gen. della chiave
- **Tag**: prende in input un msg e una chiave e produce un'autentica **tag**.
- **Ver**: prende in input un msg e un tag e restituisce 1 o 0.



# Funzioni Hash

---

- Una funzione hash e' qualcosa che prende un messaggio di lunghezza arbitraria e lo "riduce" ad un output di lunghezza arbitraria.
- Proprieta' richieste:
  - resistenza alle collisioni
  - Non invertibilita'
  - elevata efficienza.



# Funzioni hash (cont.)

---

- Le funzioni hash CR sono utilissime in pratica.

Esempi (entrambi non piu' molto sicuri):

- Secure Hash Algorithm (SHA1)
- Message Digest Algorithm (MD5)



# Unix Password Hash

---

- UNIX utilizza funzioni hash per gestire le password.

Idea: piuttosto che memorizzare la pwd si memorizza la sua hash.

Quando un utente fa il login, Unix fa l'hash della password digitata e lo confronta con il valore in memoria.



# Unix Password Hash (cont.)

---

- Converta pwd in una chiave segreta  $k$ .
  - Compatta i 7 bit ASCII di ognuno degli 8 caratteri della pwd in una chiave di 56 bit.
- Usa una variante DES per cifrare 0 con la chiave  $k$ .
  - Un seme di 12 bit è conservato insieme al valore hash (fra qualche lezione vedremo il perché).
  - Tale seme è utilizzato per "modificare" il comportamento del DES.



# Crittografia Asimmetrica

---

- Il limite della crittografia simmetrica e' che i due utenti devono condividere una chiave.
- Devono quindi incontrarsi o trovare il modo di scambiare una chiave in modo sicuro.
- Questa e' una limitazione abbastanza seria.



# Crittografia Asimmetrica

---

**Idea:** Abbiamo due chiavi una (pubblica) per cifrare e una (privata) per decifrare.

**Proprieta' richiesta:** dalla chiave pubblica non deve essere possibile ricavare la chiave privata in maniera efficiente.



# Crittografia Asimmetrica

---

Due vantaggi immediati:

- Non e' piu' necessario incontrarsi per scambiare chiavi.
- La stessa chiave (pubblica) puo' essere usata da piu' utenti.



# L'approccio "asimmetrico"

---

- Anche in questo caso si tende a costruire qualcosa di complesso a partire da primitive (piu') semplici.
- La crittografia costruisce le sue primitive di base dalla teoria dei numeri.



# L'approccio "asimmetrico" (cont)

---

- La teoria dei numeri ci fornisce problemi (apparentemente) intrattabili sui quali basare la difficoltà di rompere i sistemi asimmetrici.
- Alcuni esempi:
  - Logaritmo discreto su alcuni gruppi finiti.
  - Fattorizzazione.



# RSA

---

- Uno dei primissimi e piu' famosi sistemi asimmetrici.
- Sia  $N$  il prodotto di due primi  $p, q$  ed  $e$  un esponente pubblico.
- La funz. RSA e' definita come
$$RSA[N, e](x) = x^e \bmod N$$
- Dati  $e, N$  e  $y = x^e \bmod N$ , la conoscenza di  $p$  e  $q$  permette di ritrovare  $x$



# Standards

---

- La crittografia asimmetrica si basa molto sulla teoria dei numeri.
- La teoria dei numeri e' piuttosto complessa.
- Questo rende delicati l'utilizzo e l'implementazione di primitive asimmetriche.
- E' utile avere standard che ci dicano come utilizzare correttamente tali primitive.
  - Diverse implementazioni possono interagire senza problemi.
  - Molti errori possono essere evitati.



# Arsenale Asimmetrico

---

Prima di proseguire e' opportuno stabilire quali sono le primitive asimmetriche che ci interessano.

- Cifrari
  - Firme digitali
- 
- I cifrari sappiamo gia' cosa sono.



# Firme Digitali

---

- Una firma digitale e' l'equivalente informatico di una firma convenzionale.
- Molto simile a MA, solo che qui abbiamo una struttura asimmetrica.
  - La chiave segreta e' utilizzata per creare firme
  - La chiave pubblica per verificarle



# Differenza fondamentale

---

- Una firma digitale e' (in generale) non ripudiabile.
- La natura asimmetrica di tale primitiva fa si che solo un utente (colui che conosce la chiave segreta) possa firmare.
- Nei MAC non vi e' sostanziale differenza tra colui che autentica e colui che verifica.



# Piu' precisamente

---

- Uno schema di firma digitale consiste nei seguenti algoritmi
  - Un algoritmo di generazione della chiave KeyGen
  - Un algoritmo di generazione della firma
  - Un algoritmo di verifica



# Firme RSA di base

---

Idea: sfruttare la funzione RSA in modo inverso a quanto fatto per cifrare.

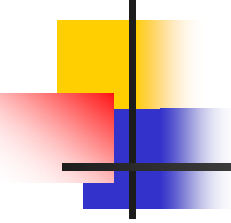
- L'algoritmo di cifratura diventa l'algoritmo di verifica.
- L'algoritmo di decifratura diventa l'algoritmo di firma



# Public-Key Cryptography Standard (PKCS)

---

- Stabilisce come utilizzare RSA (o un'altra primitiva) per evitare problemi quando (ad es.)
  - Si cifrano messaggi piccoli o indovinabili.
  - Si firmano documenti con RSA.
  - Si effettuano operazioni che renderebbero l'uso di RSA di base totalmente insicuro.



# PKCS#1 Encryption

---

- Stabilisce un formato per cifrare con RSA

0||2||almeno 8 bytes (random)||0||m

- PKCS#1 e' stato modificato a seguito di un attacco.



# Diffie Hellman

---

- Idea molto semplice per permettere a due utenti, Alice e Bob, di scambiare una chiave
  - Senza incontrarsi.
  - Nessuno oltre Alice e Bob conoscerà la chiave.
- Qui descriveremo l'idea di base (che ovviamente ha qualche problemino)



# Diffie-Hellman (cont.)

---

- $G$  insieme dove il problema del logaritmo discreto e' difficile.
- $g$  in  $G$  pubblico.
- Alice sceglie  $x$  (a caso), calcola  $A=g^x$  e manda  $A$  a Bob.
- Bob sceglie  $y$  (a caso), calcola  $B=g^y$  e manda  $B$  ad Alice.
- LA chiave comune e'  $C=g^{xy}$ .



# Problemi

---

- Per quanto Alice e Bob possano scambiare un segreto, non e' garantita autenticita'.
- Alice potrebbe scambiare la chiave con un perfetto sconosciuto che si spaccia per Bob.
- Se la chiave e' utilizzata per cifrare messaggi che solo Alice e Bob dovrebbero poter conoscere, questo potrebbe avere conseguenze sgradevoli.



# Una prima soluzione

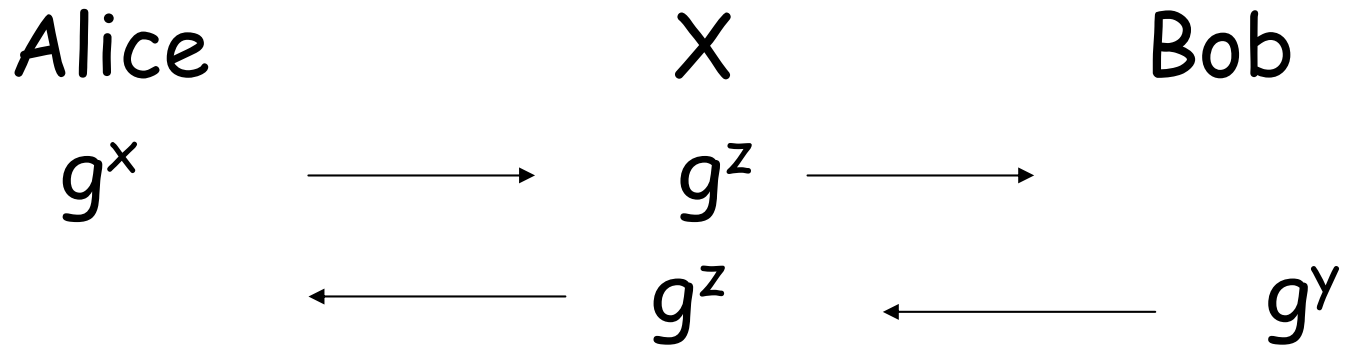
---

- Una volta stabilita la chiave Alice ed il suo interlocutore (X) scambiano, in forma cifrata, una password (precedentemente stabilita con Bob).
- Se Alice riceve la password corretta, allora X deve essere Bob...



# Man in the Middle Attack

---



Chiave in comune tra Alice e X:  $g^{xz}$

Chiave in comune tra X e Bob:  $g^{yz}$ .



# Potenziali Soluzioni

---

- Potremmo pensare di risolvere il problema chiedendo ad Alice e Bob di trasmettere la chiave che "credono" di condividere.
- Se la chiave e' la stessa allora i due condividono effettivamente la stessa chiave.
- Funziona?



## Potenziali Soluzioni (cont.)

---

- Alice e Bob trasmettono la propria identità, alla fine del protocollo.
- Anche questo approccio presenta lo stesso tipo di limitazioni.
- Guardiamo adesso (brevemente) qualche metodo più efficace.



# Published DH numbers

---

- Assumiamo che ogni utilizzatore abbia un numero  $g^x$  fisso (e certificato).
- Tutti gli utilizzatori devono essere d'accordo nell'utilizzare lo stesso  $g$ .
- I numeri sono pubblicati in un qualche modo affidabile (per es. tramite PKI)



# Published DH numbers (cont)

---

- Se X non puo' modificare tali numeri l'attacco visto non e' piu' fattibile.
- Vantaggio ulteriore: lo schema diventa non interattivo.
- Il tutto diventa un po' come un elenco telefonico...



# Diffie Hellman Autenticato

---

- Se Alice e Bob condividono un qualcosa che permette loro di riconoscersi allora il problema puo' essere risolto.
  - Es: Alice puo' verificare firme generate da Bob e viceversa.



# Diffie-Hellman Aut. (cont.)

---

- Alice sceglie  $x$  (a caso), calcola  $A=g^x$ , firma  $A$ , e manda a Bob  $A$  e la firma  $S_{\text{Alice}}$ .
- Bob sceglie  $y$  (a caso), calcola  $B=g^y$ , firma  $B$ , e manda ad Alice  $B$  e la firma  $S_{\text{Bob}}$ .
- Se entrambe le firme sono valide, la chiave comune è  $C=g^{xy}$ .



# Generatori Pseudo-Casuali

---

- In crittografia abbiamo spesso bisogno di fare scelte casuali.
- Un generatore pseudo casuale è un algoritmo che prende in input un seme casuale e lo espande in una stringa più lunga che "sembra" casuale.
- In crittografia abbiamo bisogno di generatori particolarmente "forti".