Sicurezza

Goal	**Threat**
Confidenzialità dei dati	Divulgazione
Integrità dei dati	Modifiche accidentali o intenzionali
Disponibilità dei servizi	Denial of service

Riservatezza

Può essere garantita solo tramite crittografia

Crittografia debole: proteggo i miei file dal mio fratellino (forse)

Crittografia forte: proteggo i miei file dalla CIA

Gli algoritmi crittografici forti sono pochi

Non fidatevi della crittografia fatta in casa!

Gli intrusi: vari gradi di pericolosità

- "Innocenti" ficcanaso
- Script kiddies
- Crackers per diletto
- Assalti a scopo di lucro
- Spionaggio industriale
- Spionaggio militare

Difendere la propria email dal fratellino piccolo *non è la stessa cosa* che difenderla da un intruso competente e determinato

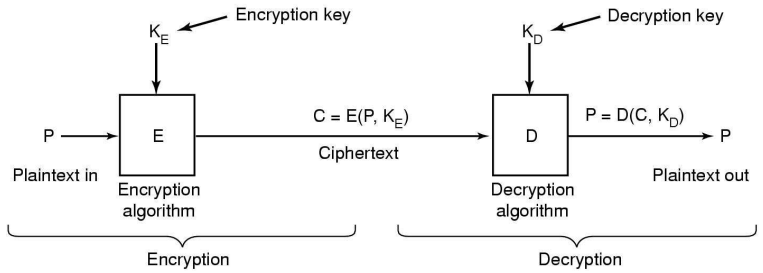
(per un fratellino di pericolosità media...)

Crittografia: che cos'è

Convertire un messaggio *in chiaro* (plaintext) in un messaggio *in cifra* (ciphertext)

Si basa su funzioni matematiche note

Il segreto sta nei parametri degli algoritmi, detti *chiavi*



Un semplice algoritmo di cifratura

Monoalphabetic substitution cipher

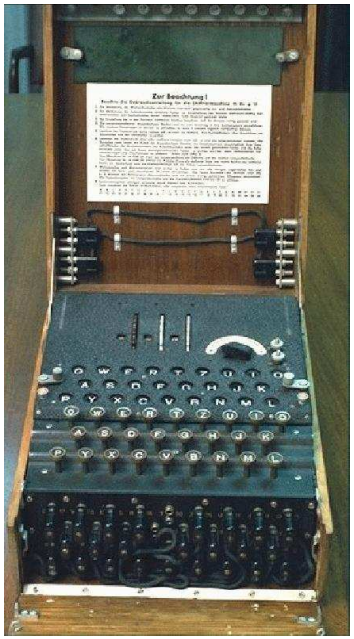
Ad ogni lettera faccio corrispondere una lettera

```
Q W E R T Y U I O P A S D F G H J K L Z X C V B N M
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

"Ciao mamma" → "EOQGDQDDQ"

Si risolve facilmente con l'analisi della frequenza delle lettere

(Vedi "Lo scarabeo d'oro" di Edgar Allan Poe)



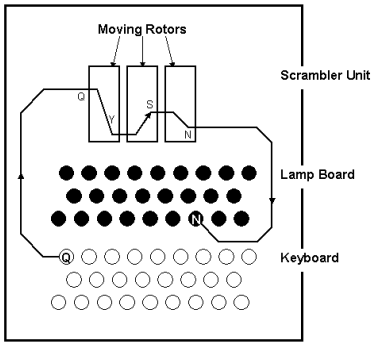
La macchina da cifra "Enigma"

Inventata per il commercio nel 1923

Input: tastiera

Output: display con 26 lampadine

La funzione di sostituzione cambia ad ogni pressione di tasto



Funzioni difficili da invertire

Esempio: "se mi dici quanto fa 1654176541*1987612761 guadagni 100 euro"

un bambino con una calcolatrice può fare questo conto in meno di un ora

Funzioni difficili da invertire

Esempio: "se mi dici quali sono i fattori di 3287862401838439701 guadagni 100 euro"

la maggior parte degli adulti non sono in grado di risolvere questo problema, non importa quanto tempo gli lasciamo

Funzioni botola

Una funzione f è detta botola se

- dato x, calcolare $f(x)$ è facile
- dato y, trovare x' t.c. $y = f(x')$ è difficile
- piccole variazioni nell'input sono amplificate

Digest

Un digest è una funzione botola il cui output è un numero di n bit

Es SHA1:

```
$ echo 'Ciao mamma' | sha1sum
5fbed240447cad74b92a6c9eff1be4a08a784cc3
$ echo 'Ciao mammb' | sha1sum
06995066c72bed7e1674be6bed19e0d284d3cf69
```

$40 * 4 = 160$ bit

Authentication

to prove the identity of a user

- segreti: password, certificati digitali
- oggetti fisici: smart card, chiavi, ...
- biometria: impronte digitali, iride, ...

Password = problemi

- brute force attacks
- dictionary attacks
- audit
- fattori umani

Authorization

to determine what the user can do

Protection domain: l'insieme dei diritti posseduti da un processo

Un *diritto* è una coppia (oggetto, azioni permesse)
es. (/home/matteo/foo.txt: read,write), (/etc/passwd: read)

Ci deve essere un meccanismo per *cambiare* domain
(es. permettere a un utente di cambiarsi la password)

Protection domains in Unix

Ci sono due protection domains: *kernel* e *user*

In pratica questi due domain sono insufficienti, per cui si definiscono domini di protezione *associati agli utenti*

Protection matrix

Concettualmente, esiste una *matrice di protezione*: una relazione fra *utenti* e *oggetti*

	Object							
	File1	File2	File3	File4	File5	File6	Printer1	Plotter2
Domain 1	Read	Read Write						
2			Read	Read Write Execute	Read Write		Write	
3						Read Write Execute	Write	Write

Domain switching

Per rappresentare la possibilità di cambiare dominio, trattiamo i "domini" come "oggetti"

	Object										
	File1	File2	File3	File4	File5	File6	Printer1	Plotter2	Domain1	Domain2	Domain3
main 1	Read	Read Write								Enter	
2			Read	Read Write Execute	Read Write		Write				
3						Read Write Execute	Write	Write			

Audit

Quando un tentativo di accesso fallisce, occorre registrare questo fatto?

E quando un tentativo di accesso ha successo?

Occorre predisporre una *audit policy*

Trasferimento dei diritti

Può un utente trasferire un diritto a un altro utente? ⇒ aggiungiamo il permesso "copy right"

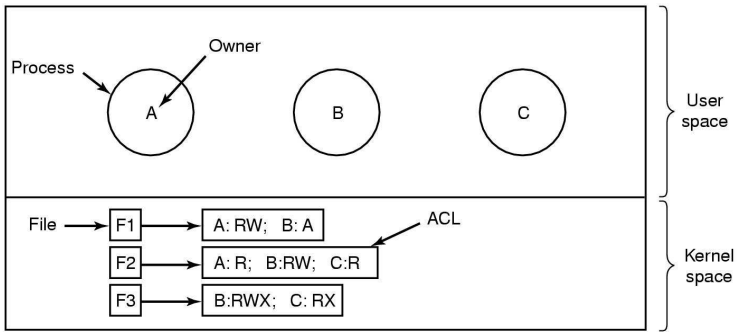
L'utente beneficiato può a sua volta trasferire il diritto? ⇒ distinguiamo "copy" e "copy limited"

Una volta trasferito il diritto, l'utente originale lo perde? ⇒ in questo caso si chiama "transfer"

Dobbiamo marcare nella matrice di protezione quali diritti hanno i permessi copy, copy limited, transfer

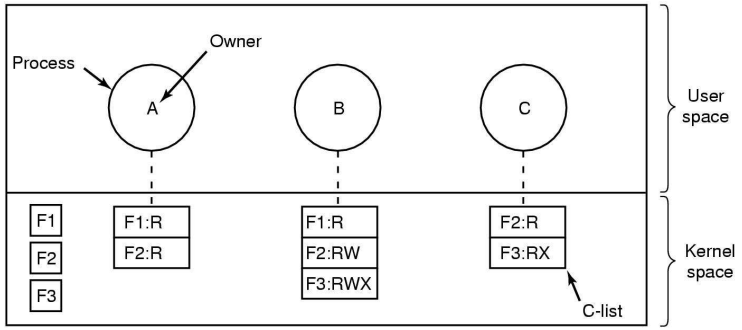
Implementazione della matrice di protezione

Per righe: *Access Control Lists* (ACL)



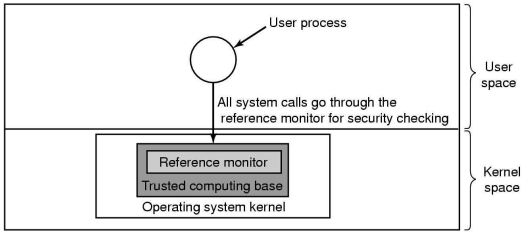
Implementazione della matrice di protezione

Per colonne: *Capability Lists*



Trusted computing base

TCB: *all hardware and software necessary to enforce the security rules*



Multilevel security

Nelle applicazioni militari, ci sono diversi *livelli* di autorizzazione (*clearance*):

- top secret
- secret
- confidential
- unclassified

Un sistema è *multilevel* se deve poter essere usato da persone di diverso livello di clearance

Il modello Bell-La Padula

- simple security property (no read up)
- star property (no write down)

Orange book security

Livello D: no security (MS-DOS, Win 95/98/Me)

Livello C1: multiuser
- authentication
- discretionary access control (DACL)
- protected mode OS

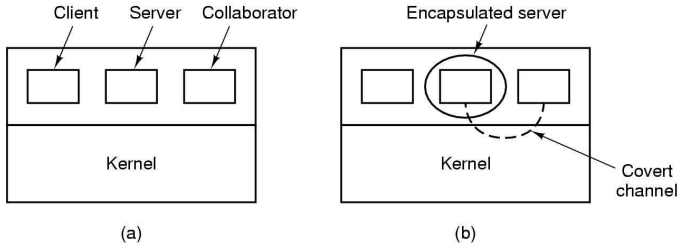
Livello C2: DACL a livello di singolo utente ⇒ Unix raggiunge C1 ma non C2 (perché non ha le ACL)

Livello B: multilevel
- ogni utente e oggetto ha un livello di *clearance*

Livello A: formally verified

Covert channels

La "star property" non è realizzabile in pratica



Autenticazione in Unix: /etc/passwd

Database delle password: /etc/passwd

Password crittate *one-way*

Salt

/etc/shadow

Autorizzazione in Unix

Users are identified by UID (an integer)

Groups are identified by GID (an integer)

Every user belongs to one or more groups

/etc/passwd

```
root:x:0:0::/root:/bin/bash
ftp:x:14:50::/home/ftp:
matteo:x:500:100:Matteo Vaccari:/home/matteo:/bin/bash
```

/etc/groups

```
root::0:root
ftp::50:
users::100:
project-X::1234:matteo,marco,guido
```

Every file has a UID and a GID, and RWX permissions for the UID, the GID and everyone else

```
-rw-r-r-    1 matteo   users          5 May 30 11:21 pippo.txt
```

Inizialmente UID e GID sono quelli dell'utente che ha creato il file

Posso cambiare UID (e GID) con il comando chown(1)

```
chown <newuser> <file> <file> <file> ...
chgrp <newgroup> <file> <file> <file> ...
```

Per esempio:

```
chown guest pippo.txt pluto.txt topolino.c
chgrp other pippo.txt pluto.txt topolino.c
```

Oppure, in un colpo solo:

```
chown guest.other pippo.txt pluto.txt topolino.c
```

Permission bits for gurus

la rappresentazione `rw-r--r--` è per niubbi

```
rw-  r--  r--  niubbo
110  100  100  binario
 6    4    4   ottale
```

Allora diciamo che un file ha il **modo** 644

(Nota: in C la costante ottale si indica con il prefisso "0", es. 0644)

I gruppi

Sono il meccanismo principale per gestire la collaborazione in Unix

newgrp(1) is used to change the current group ID during a login session.

```
$ touch aaa
$ ls -l aaa
-rw-r-r-    1 matteo   users          0 Jun  9 10:49 aaa
$ newgroup project-X
$ touch bbb
$ ls -l bbb
-rw-r-r-    1 matteo   project-X      0 Jun  9 10:49 bbb
$
```

chgrp(1) e newgrp(1) mi permettono di usare solo i gruppi a cui appartengo

(chown(1) è riservato a root)

Problema

dati questi permessi:

```
-rw---- 1 root root 605 Dec 2 2002 /etc/shadow
```

come può un utente cambiare la sua password?

Il bit SETUID

```
S_ISUID   04000 set user ID on execution
```

Un normale eseguibile: modo 711

```
-rwx-x-x    1 root     bin    46700 May 28  2002 /bin/ls
```

Un eseguibile setuid: modo 4711

```
-rws-x-x    1 root     bin    36800 Jun 10  2002 /usr/bin/passwd
```

Quando un processo fa una exec(2) di un eseguibile setuid, il suo UID *effettivo* diventa quello del file eseguibile

$\Rightarrow$ eseguo /usr/bin/passwd con i permessi di root

Pericolo! Passwd deve essere scritto con *molta* attenzione

... infatti si rifiuta di cambiare la password di altri utenti

Chiamate di sistema relative alle autorizzazioni in Unix

int getuid(void);
    returns the **real** user ID of the current process

int geteuid(void);
    returns the **effective** user ID

int getgid(void)
    returns the **real** group ID of the current process

int getegid(void);
    returns the **effective** group ID

int setuid(int uid);
    sets the effective user ID of the current process

Ancora su setuid

Ciascun processo ha un

- *real* UID
  - usato per sapere chi è il proprietario originale
- *effective* UID
  - usato per decidere che cosa può fare

Normalmente:
effective UID == real UID

Quando eseguo un programma setuid:
effective UID != real UID

/usr/bin/passwd

- usa il real UID per sapere di chi bisogna cambiare la password
- ha bisogno di effective UID == 0 per potere cambiare /etc/shadow

Il problema fondamentale della crittografia tradizionale

La chiave per crittare è la stessa usata per decrittare

Per potere comunicare in cifra le due parti devono prima scambiarsi la chiave

Il costo della sicurezza delle chiavi è preponderante

Il problema dello scambio delle chiavi rende impossibile l'uso della crittografia fra sconosciuti $\Rightarrow$ la crittografia non è di alcuna utilità in Internet

... o no?

Crittografia a chiave pubblica

Una chiave serve a crittare, l'altra per decrittare

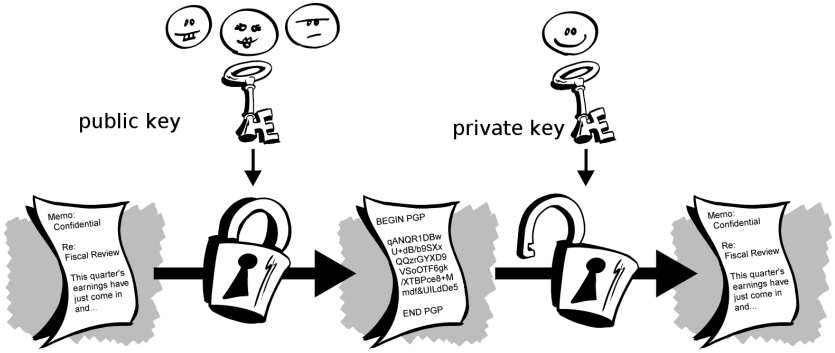
Non è possibile ricavare l'una dall'altra

Whitfield Diffie e Martin Hellman, 1976

Rivest, Shamir, Adelman 1979

Martin Hellman

Whitfield diffie



Crittografia a chiave pubblica

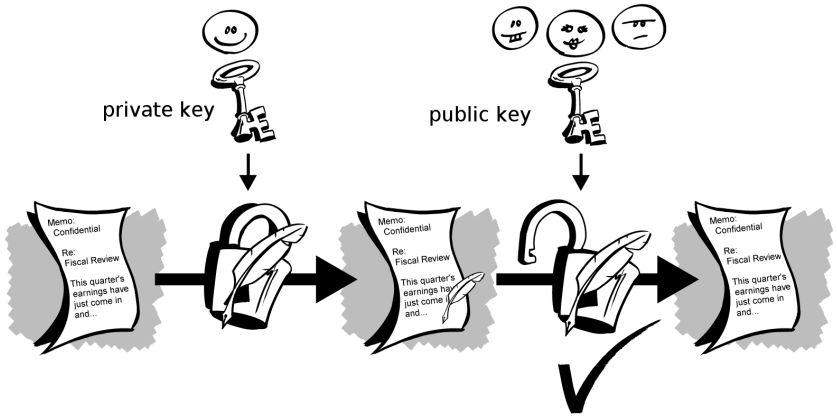
Bob manda la sua chiave pubblica ad Alice (in chiaro)

Alice critta un messaggio per Bob con la chiave pubblica di Bob

Bob riceve il messaggio crittato e lo decritta con la sua chiave privata

L'attaccante intercetta il messaggio, ma non può decrittarlo perché ha solo la chiave pubblica!

Ma: l'attaccante può intercettare il messaggio, e fare arrivare a Bob un messaggio diverso (man-in-the-middle attack)



Firme digitali

Bob vuole essere certo che i messaggi di Alice siano tali

Alice critta il messaggio M con la sua chiave privata: ottiene $Apr(M)$

Alice manda M e $Apr(M)$ a Bob

Bob decritta $Apr(M)$ con la chiave pubblica di Alice: ottiene $Apu(Apr(M))$

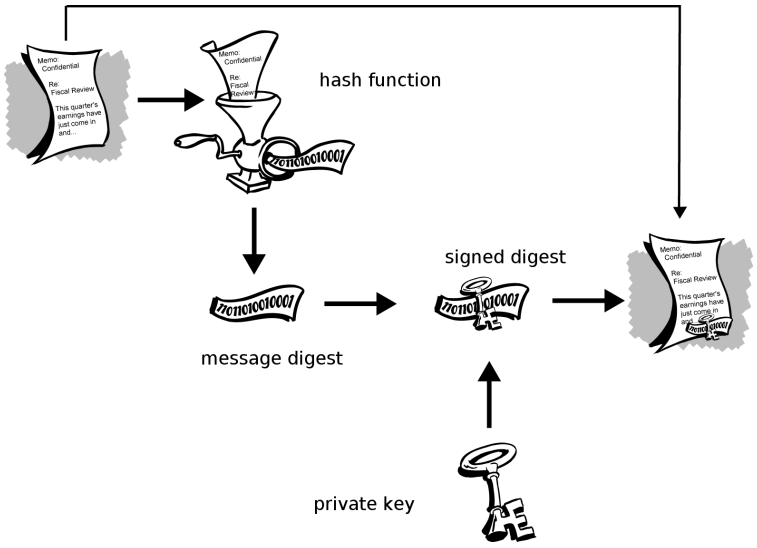
Se $M = Apu(Apr(M))$ allora il messaggio proviene proprio da Alice, e non è stato manipolato!

Un messaggio firmato con PGP

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Un esempio di narrativa cypherpunk è "Cryptonomicon" di Neil Stephenson
-----BEGIN PGP SIGNATURE-----
Version: PGPfreeware 6.0.2i

iQA/AwUBOPKqkRLxzDGSvFPxEQI9sQCgi0+MSTQnDNPkQ6ZgFdSn2sJflb4AoINa
ndz5LJmqG3wgT/oATh1pejPR
=M9ra
-----END PGP SIGNATURE-----
```



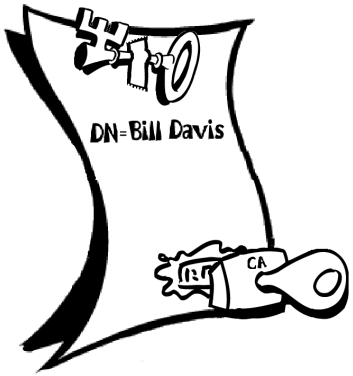
Ma...

...siamo certi che la chiave pubblica di Alice sia autentica?

man-in-the-middle attack

Certificato digitale

- chiave pubblica
- identità del proprietario
- una o più firme digitali



Certificati in formato X.509

- X.509 version number
- chiave pubblica
- numero di serie del certificato
- distinguished name del proprietario
  CN=Bob Allen, O=Network Associates, Inc., C=US
- data di inizio e fine validità
- nome unico dell'ente emittente
- firma digitale dell'emittente

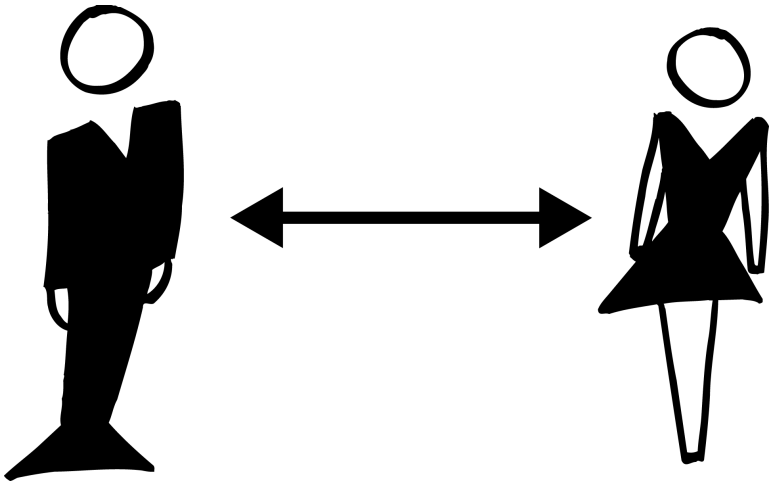
Validità e fiducia

Un certificato può essere valido

Una persona può essere fidata

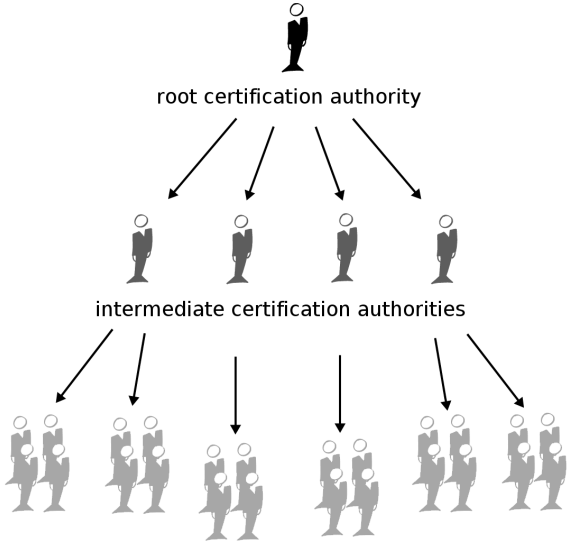
Modelli di fiducia (i)

Direct trust



Modelli di fiducia (ii)

Hierarchical trust



Modelli di fiducia (iii)

Web of trust

Alice vuole certificare che la sua chiave pubblica è veramente sua

Alice chiede a Dave di firmare la chiave pubblica di Alice

Dave usa la chiave privata di Dave per firmare la chiave pubblica di Alice; quindi *Dave certifica che la chiave è autentica*

Bob ha una copia della chiave pubblica di Dave di cui è certo

Bob ha una copia della chiave pubblica di Alice di cui è meno certo

Bob verifica che la firma di Dave sulla sua copia della chiave di Alice è autentica,

Ora Bob si fida un po' di più della sua copia della chiave di Alice

SSH

sshd, ssh, scp, sftp

```
matteo@orata $ ssh matteo@pisolo
Last login: Sat Jun  3 07:02:53 2006 from 213-140-17-98.ip.fast
matteo@pisolo $
```

```
$ scp file0 file1 file2 matteo@pisolo:/home/matteo/files
```

The Internet security model

- The hosts are secure
- The attacker owns the channel

telnet, rlogin, and ftp send your user ID and password as unencrypted plain text