Sistemi Operativi

a.a. 2008/2009

A. Petrosino

Protezione dei sistemi informatici

Cifratura

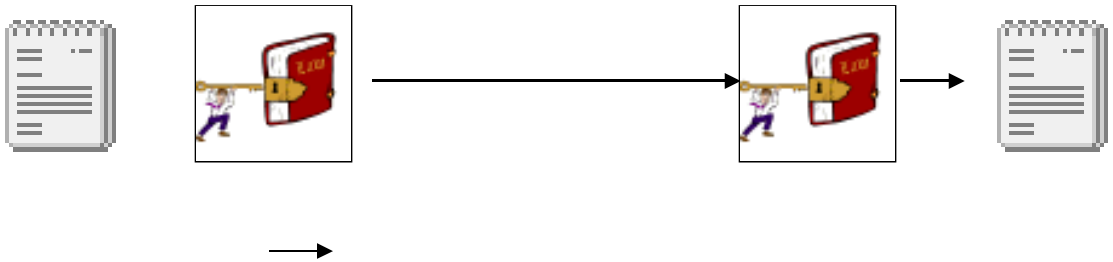
- La cifratura è un metodo comune di protezione delle informazioni trasmesse su collegamenti affidabili.
- L'informazione (testo) viene cifrata (codificata), passando dal testo in chiaro (forma leggibile) ad una forma interna (testo cifrato)
- Il testo cifrato può essere memorizzato in un file leggibile oppure trasmesso su canali non protetti
- Il ricevente deve decifare (decodificare) il testo riconducendolo al testo in chiaro
- Proprietà di una buona tecnica di cifratura:
  - Relativamente semplice per utenti autorizzati a cifrare e decifrare i dati.
  - Schema di cifratura non dipendente dalla segretezza dell'algoritmo, ma da un parametro dell'algoritmo chiamata chiave di cifratura.
  - Estremamente difficile determinare la chiave di cifratura per un intruso.

Cifratura

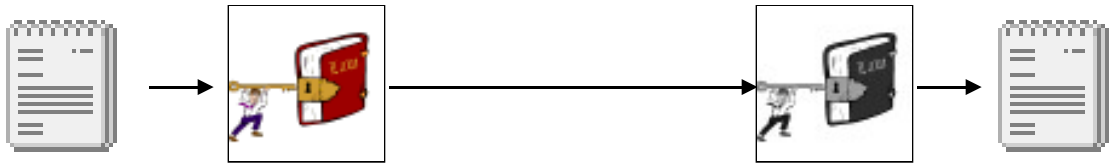
- Un algoritmo è basato sulla coppia D(.), E(.). E' importante distinguere:
- Algoritmi riservati: si basano anche sulla segretezza dell'algoritmo usato
- Algoritmi basati su chiavi: l'algoritmo è noto, la segretezza è nella(e) chave(i):
  - $C = E(K,M) = E_k(M)$
  - $M = D(K,C) = D_k(C)$
  - A loro volta si distinguono:
  - con chiave singola/multipla
  - simmetrici
  - con chiave pubblica
  - con chiave privata, nota solo all'utente che deve decifrare

Cifratura

- Cifratura simmetrica
  - Codifica e decodifica usano la stessa chiave



- Cifratura asimmetrica
  - Codifica e decodifica usano chiavi diverse

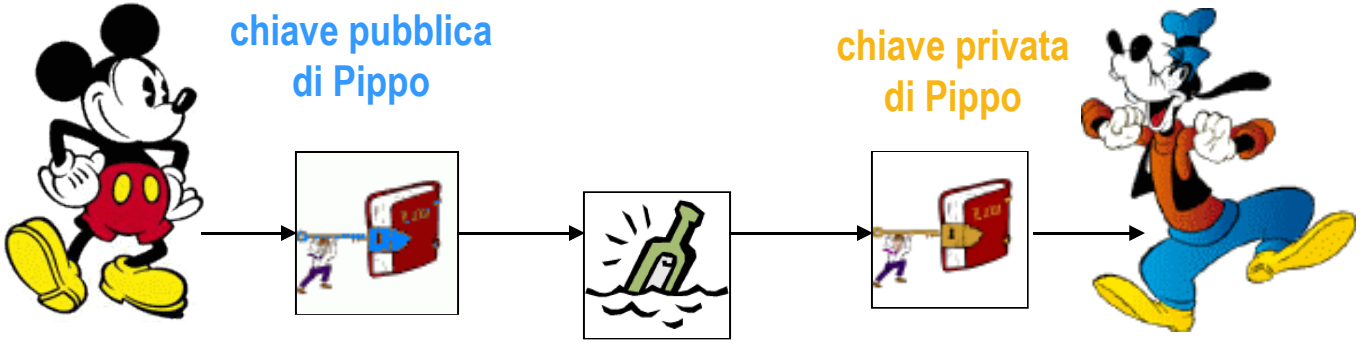


Cifratura

- Cifratura simmetrica: problemi
  - Trasmettere la chiave su un canale sicuro
- Il problema viene risolto dalla cifratura a chiave pubblica
  - Ogni utente ottiene una coppia di chiavi, una delle quali viene resa pubblica

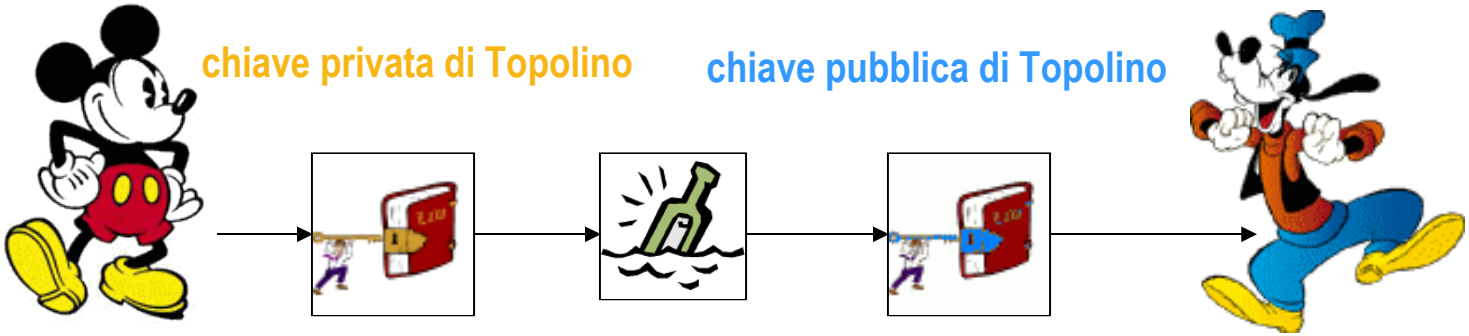
Messaggi cifrati

- Topolino vuole mandare un messaggio segreto a Pippo
  - Topolino usa la chiave pubblica di Pippo per cifrare il messaggio
  - Solo Pippo è in grado di decodificare il messaggio
  - Pippo tuttavia non può essere sicuro dell'identità di Topolino



Messaggi firmati

- Topolino vuole mandare un messaggio firmato a Pippo
  - Topolino usa la sua chiave privata per cifrare il messaggio
  - Pippo è in grado di decodificare il messaggio usando la chiave pubblica di Topolino
  - Solo Topolino poteva inviare quel messaggio
  - Il messaggio però non è segreto, tutti lo possono leggere



Tentativi di decodifica

- Criptoanalisi
- Forza bruta

Dimensione della chiave in bit	Numero di chiavi alternative	Tempo richiesto a 1 cifratura/us	Tempo richiesto a 10^6 cifratura/us
32	4.3×10^9	35.8 minuti	2.15 millisec.
56	7.2×10^{16}	1142 anni	2.15 ore
128	3.4×10^{38}	5.4×10^{24} anni	5.4×10^{18} anni

Rottura di algoritmi di sicurezza

- **Rottura totale:** si trova la(e) chiave(i)
- **Deduzione globale:** senza conoscere k il croptoanalista trova A(.) tale che

$$A(C) = D_k(C) = P$$

- **Rottura locale:** il crittoanalista scopre il testo in chiaro per un certo testo cifrato
- **Deduzione di informazione:** il crittoanalista scopre informazione utile per la rottura successiva

Comunicazioni a chiave pubblica

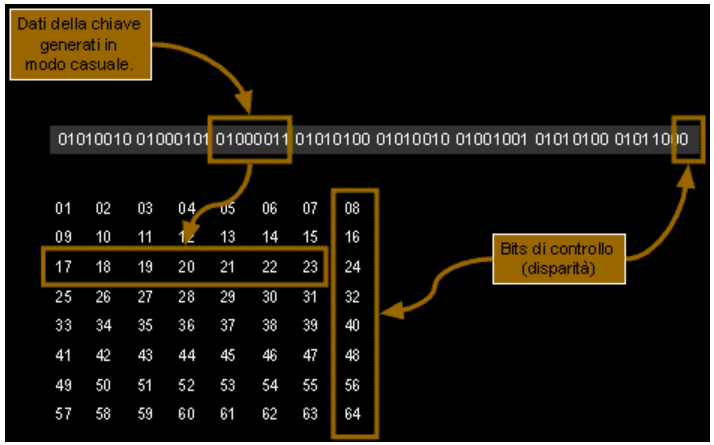
- Topolino e Pluto si accordano su un certo algoritmo crittografico
- Pluto spedisce a Topolino la sua chiave pubblica
- Topolino codifica il suo messaggio usando la chiave pubblica di Pluto e lo spedisce a Pluto
- Pluto decodifica il messaggio di Topolino usando la propria chiave privata

- Si può decidere di ospitare tutte le chiavi pubbliche in uno stesso database comune, di cui proteggere l'integrità

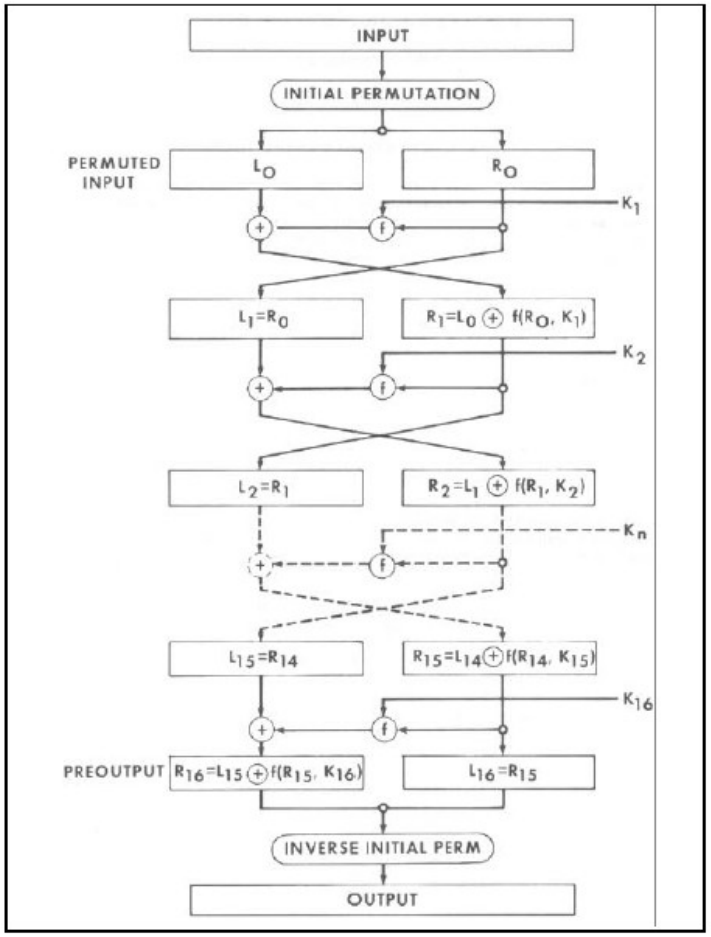
non viene chiesto a Pluto di inviare la sua chiave pubblica per ricevere un messaggio

Data Encryption Standard

- *DES* sostituisce caratteri e riorganizza l'ordine sulla base della chiave di cifratura fornita all'utente autorizzato attraverso un meccanismo sicuro.
- La chiave iniziale è costituita da 64 bit generati in modo casuale
  - l'ottavo bit di ogni riga è un controllo di disparità effettuato sui bit costituenti la riga



DES


INPUT
INITIAL PERMUTATION
PERMUTED INPUT
L_0
R_0
K_1
$L_1=R_0$
$R_1=L_0 \oplus f(R_0, K_1)$
K_2
$L_2=R_1$
$R_2=L_1 \oplus f(R_1, K_2)$
K_n
$L_{15}=R_{14}$
$R_{15}=L_{14} \oplus f(R_{14}, K_{15})$
K_{16}
PREOUTPUT
$R_{16}=L_{15} \oplus f(R_{15}, K_{16})$
$L_{16}=R_{15}$
INVERSE INITIAL PERM
OUTPUT

DES

- I 64 bit della chiave iniziale vengono sottoposti alla permutazione ***IP***

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

DES

- La computazione è dipendente dalla chiave e consta di 16 iterazioni. Ogni iterazione ha come output $L_n R_n$
- Sia K_n, per l'iterazione n-ma per $n \geq 0$, un blocco di 48 bit scelto dalla chiave a 64 bit ***KEY***

$$L_n = R_n$$

$$R_n = L_{n-1} \oplus f(R_{n-1}, K_n)$$

- Dove $\oplus$ indica la somma modulo 2.
- Alla 16-ma iterazione, si ricava il blocco di pre-output $R_{16} L_{16}$.

DES

- Al termine delle 16 iterazioni, il blocco di pre-output è soggetto alla seguente permutazione, inversa di quella iniziale:

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

La funzione *f*

- ***E*** fornisce in output un blocco di 48 bit su input di 32 bit

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

- I primi tre bit di ***E(R)*** sono i bit nelle posizioni 32, 1 e 2 di ***R*** mentre gli ultimi 2 bit di ***E(R)*** sono i bit nelle posizioni 32 e 1.
- Ognuna delle funzioni di selezione $S_1, S_2, \ldots, S_8$, prende in input un blocco a 6 bit e produce un blocco a 4. Es. S_1

Column Number

Row No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

La funzione *f*

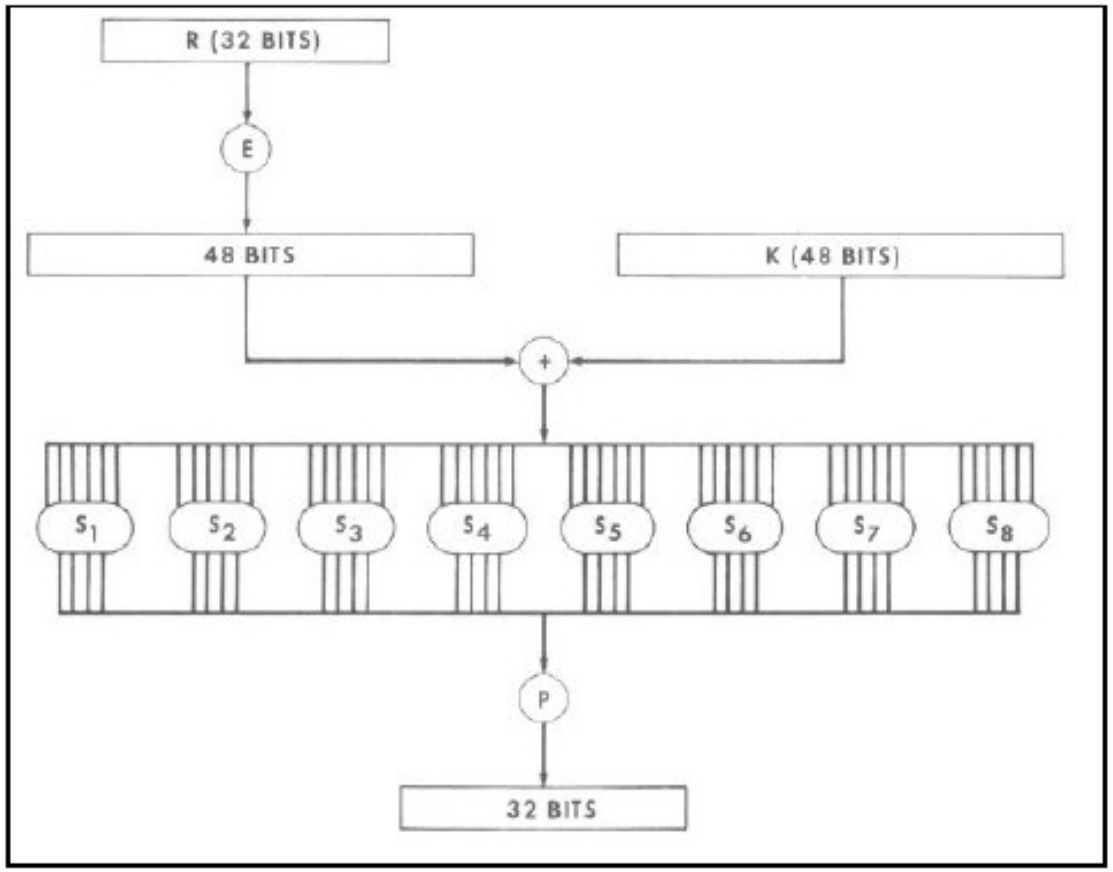
- Se S_1 è la funzione definita nella tabella e ***B*** è un blocco di 6 bit, allora $S_1(B)$ viene determinato come segue:
  - Il primo ed ultimo bit di ***B*** rappresenta un numero in [0,3] in base 2. Sia tale numero i. I 4 bit centrali di ***B*** rappresentano un numero in [0,15] in base 2. Sia tale numero j. S_1 fornisce in output a ***B*** la codifica binaria del numero presente nella posizione (i,j) della tabella.
  - Esempio, per 011011 la riga é 01 (i=1) e la colonna è 1101 (j=13). L'output è 0101.
- La funzione di permutazione ***P*** fornisce un blocco a 48 bit in output da un input a 32 bit secondo la seguente tabella:

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

La funzione f

- Per definire f(R,K), si definiscono dapprima i blocchi B1, …, B8 come

$$B_1 B_2 \ldots B_8 = K \oplus E(R)$$



$$f(R,K) = P(S_1(B_1)S_2(B_2)\ldots S_8(B_8))$$

Decifrazione

- La permutazione IP^{-1} applicata al blocco pre-output è l'inversa della permutazione iniziale ***IP*** applicata all'input.
- Per **decifrare** è necessario applicare lo stesso algoritmo al messaggio cifrato.

$$R_{n-1} = L_n$$
$$L_{n-1} = R_n \oplus f(L_n, K_n)$$

- dove $R_{16}L_{16}$ è il blocco input permutato e L_0R_0 è il blocco pre-output.

RSA (Rivest-Shamir-Adleman)

- La chiave pubblica di cifratura è formata da una coppia (e,n), la chiave privata da una coppia (d,n)
  - e, d, n interi positivi
- Un messaggio è rappresentato da una sequenza di numeri da 0 ad n-1
- Le funzioni E e D sono definite come segue:
  - $E(M) = M^e \bmod n = C$
  - $D(C)\ C^d \bmod n$
- L'algoritmo si basa sul Teorema di Eulero

  Se $\gcd(w,n)=1$ allora $w^{\phi(n)} \equiv 1 \bmod n$
- E' facile fare il prodotto di due grossi interi, è difficile fattorizzare un grosso intero. La fattorizzazione è intrattabile?

RSA (Rivest-Shamir-Adleman)

- L'intero n viene scelto come prodotto di due numeri primi grandi tale che:

$$n = p \times q$$

- L'intero d viene scelto in modo che

$$\text{mcd}\,[d, (p-1)x(q-1)] = 1$$

- L'intero e viene calcolato da p, q e d come inverso moltiplicativo di d

$$e \times d \bmod (p-1) \times (q-1) = 1$$

Algoritmo efficiente per RSA

- Calcolare in modo efficiente

$$w = c^d \bmod n \qquad c = w^e \bmod n$$

- Si divide il messaggio in blocchi di lunghezza *i*: $10^{i-1} < n < 10^i$

- Esempio: Calcola 7^{23} mod 61. In binario $23 =_2 10111$. Dunque:

$$7^{23} \bmod 61 =$$
$$= 7^{(1\cdot 2^4 + 0\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0)} \cdot \bmod 61$$
$$= 7^{2^4} \cdot 7^0 \cdot 7^{2^3} \cdot 7^{2^2} \cdot 7^{2^1} \cdot 7^{2^0} \bmod 61$$
$$(7^{2^4} \bmod 61 \cdot 7^0 \bmod 61 \cdot 7^{2^3} \bmod 61 \cdot 7^{2^2} \bmod 61$$
$$\cdot 7^{2^1} \bmod 61 \cdot 7^{2^0} \bmod 61) \bmod 61$$

- Dunque mi servono termini del tipo 7^{2^j} mod 61

Algoritmo efficiente per RSA

- Semplice soluzione ricorsiva:

$$7^{2^j} \bmod 61 = 7$$

- Passo base: se j = 0 allora
- Induzione

$$7^{2^{j+1}} \bmod 61 = 7^{2^j \cdot 2} \bmod 61 = (7^{2^j})^2 \bmod 61 = (7^{2^j} \bmod 61)^2$$

- dunque, per j=0,1,2,3,4,… si ha rispettivamente
- $7^{2^{j+1}} \bmod 61 = 7^{2^j \cdot 2} \bmod 61 = (7^{2^j})^2 \bmod 61 = (7^{2^j} \bmod 61)^2$

Il Problema dello Knapsack (zaino)

- Dati i pesi $M = \{ M_i, 1 \leq i \leq n \}$
- Dato il sacco S, determinare, se esiste, una disposizione $\{ b_i, 1 \leq i \leq n \}$ dei pesi nel sacco tale che

$$\sum_{i=1}^{n} b_i M_i = S$$

- Il problema è generalmente intrattabile

Superincreasing Knapsack

- Non sempre il knapsack è intrattabile
- E' possibile una soluzione lineare per pesi (M_i^s) superincreasing, tali che

$$M_i^s > \sum_{j=1}^{i-1} M_j^s$$

- Esempio: (52, 27, 13, 6, 3, 2).

- Come si risolve il superincreasing knapsack con sacco *S*?

Superincreasing Knapsack

- L'algoritmo è lineare
- Esempio: (52,27,13,6,3,2)

- Formulazione per semplice caso in cui esiste soluzione
- Superincreasing Knapsack

```
for i=1, …, L do
if M_i ≤ S then
begin
        b_i = 1;
        S = S - M_i;
end
else b_i = 0
```

Codifica con Hard Knapsack

- **Esempio:** Se il testo in chiaro raggruppato per 6 é:

$$P = (\ 111001\ 010110\ 000000\ 011000\ )$$

ed i pesi del knapsack

$$(1,5,6,11,14,20)$$

si ottiene il messaggio in codice

$$(1+5+6+20,\ 5+11+14,\ 0,\ 5+6) \equiv (32,30,0,11\ )$$

- Un attacco basato su solo codice cifrato richiede di risolvere *l'hard knapsack*.

Dal superincreasing all'hard-knapsack

- Si usa il seguente risultato di teoria dei numeri:

  Siano (M_i^s, $1 \leq i \leq L$) *i pesi di un* superincreasing knapsack. *Se*

$$m > \sum_{i=1}^{L} M_j^s \quad n : \gcd(m,n) = 1$$

  allora i pesi

$$M_i^h = n \cdot M_i^s \bmod m$$

  definiscono un hard-knapsack

- Partendo da pesi superincreasing trovo un hard knapsack, che uso per la codifica. I pesi dell'hard knapsack sono la *chiave pubblica*.

Decifrare da superincreasing knapsack

- Risultato:

  Sia S_h un sacco dell'hard knapsack determinato a seguito dell'applicazione del risultato presedente. Allora il corrispondente sacco del superincreasing knapsack è

$$S_s = n^{-1} S_h \bmod m$$

  dove

$$n^{-1} : n \cdot n^{-1} \equiv 1 \bmod m$$

- Appena ho il sacco superincreasing S_s posso usare l'algoritmo lineare con la chiave privata per recuperare il messaggio.

Decifratura - Esempio

- $n = 31$, $m = 105$
  - (62, 93, 81, 88, 102, 37) - chiave pubblica
  - (2, 3, 6, 13, 27, 52) - chiave privata
  - (174, 280, 333) - messaggio in codice (chiave pubblica).

- Da cui $n^{-1} = 61$ ed i sacchi relativi al superincreasing sono

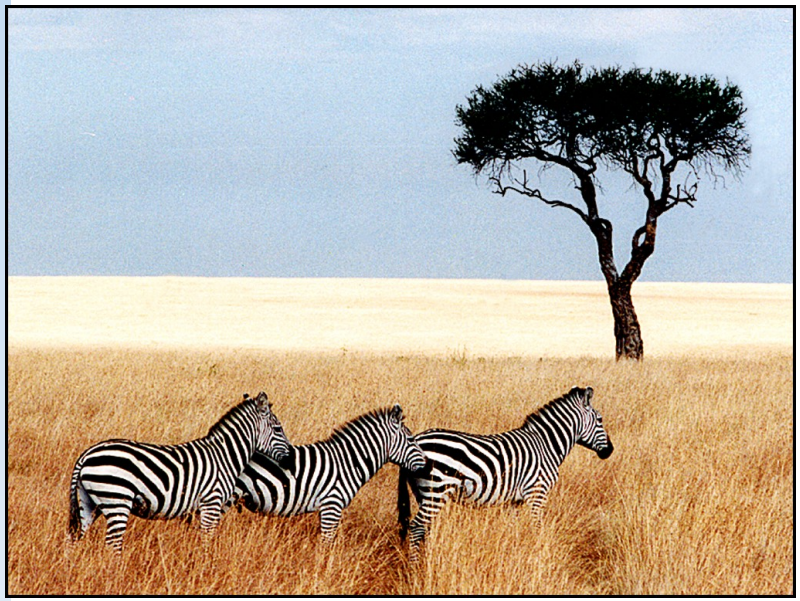
$174*61 \bmod 105 = 9 = 3 + 6 \rightarrow 011000$

$280*61 \bmod 105 = 70 = 2 + 3 + 13 + 52 \rightarrow 110101$

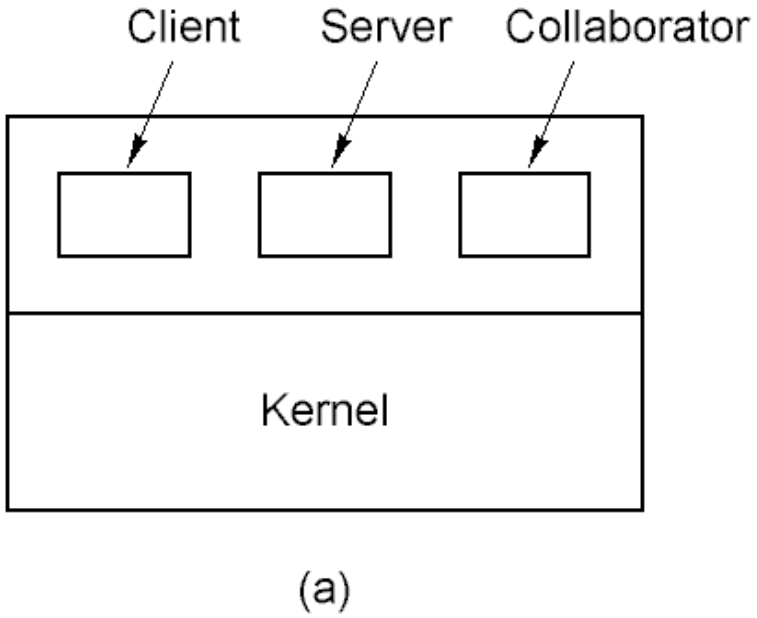
$333*61 \bmod 105 = 48 = 2 + 6 + 13 + 27 \rightarrow 101110$

Canali nascosti

- Le immagini appaiono le stesse
- L'immagine di destra include il testo completo di cinque lavori di Shakespeare
  - Cifrato ed inserito nei primi bit dei valori di colore

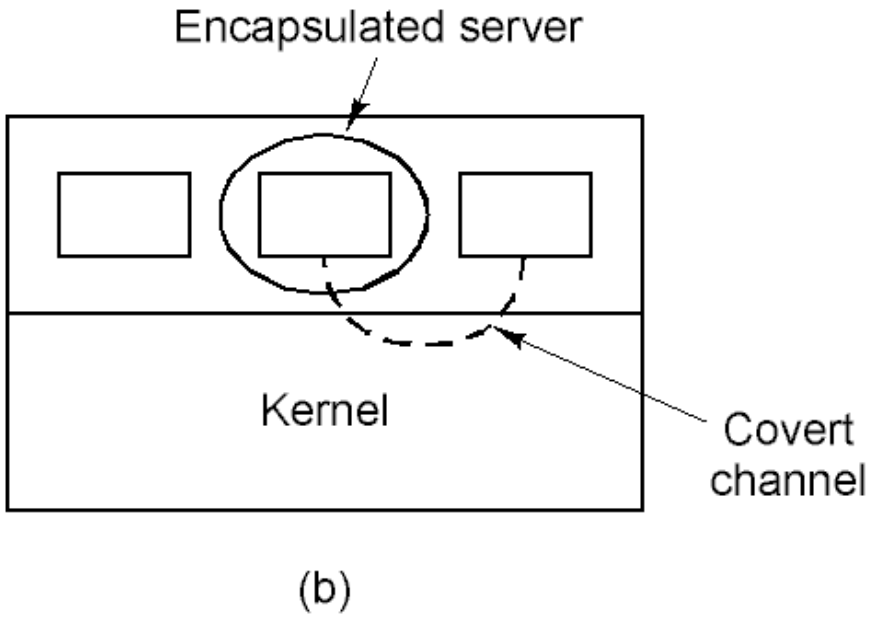


Canali nascosti



(a)

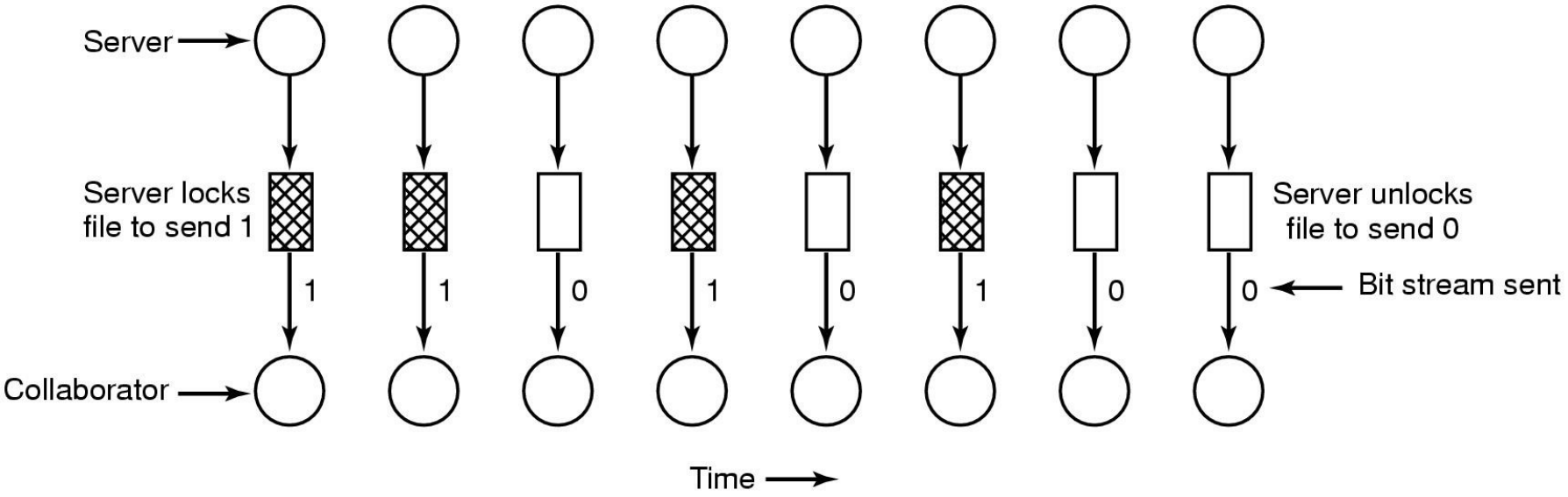
I processi cliente, server e collaboratore



(b)

Il server incapsulato può ancora comunicare col collaboratore tramite canali nascosti

Canali nascosti



Un canale nascosto che usa il bloccaggio di file

Sistemi transazionali

- Un sistema transazionale è un sistema di accesso ad archivi (generalmente database) in cui le operazioni (transazioni) sono raggruppate in modo da lasciare sempre l'insieme dei dati in uno stato coerente.
  - Operazioni logicamente elementari
  - Tempi di esecuzione brevi
  - Numero di operazioni molto elevato
  - Terminali remoti, trasmissione a distanza
  - Gestione dei malfunzionamenti

Sistemi transazionali

- Un sistema transazionale deve presentare quattro proprietà:
  - *Atomicità* . Una transazione termina solo in due modi: con l'esecuzione completa delle operazioni (*commit*), o con il ripristino della situazione di partenza (*abort*).
  - *Persistenza* . Gli effetti di una transazione completata non devono essere persi, e devono essere immediatamente visibili.
  - *Serialità* . L'esecuzione di più transazioni deve poter avvenire logicamente in parallelo, e il suo effetto globale deve essere indipendente dall'ordine reale delle transazioni.
  - *Isolamento* . Le transazioni devono operare indipendentemente, e l' *abort* di una non deve causare l' *abort* di altre.
- Atomicità e persistenza riguardano l'affidabilità (memoria stabile, *recovery manager*), serialità e isolamento riguardano la concorrenza (*concurrency control system*)

Giornale di transazione

- Il giornale (o *log*) di transazione è un file che contiene in sequenza la descrizione di tutte le operazioni effettuate sulla base dei dati Ogni operazione è un record del file. Attraverso il *log* è possibile ricostruire uno stato coerente della base di dati a seguito di una interruzione nel funzionamento o di un guasto.

- I record sul giornale di transazione sono scritti nell'ordine cronologico di esecuzione delle operazioni.
  - Inizialmente viene scritto un record di *begin- transaction* .
  - Successivamente, per ogni operazione effettuata, vengono scritti due record:
    - lo stato iniziale del dato (*before image*)
    - lo stato finale del dato (*after image*)

Giornale di transazione

- Al termine viene scritto un record con l'informazione di *commit* o *abort* , seguita da un record di *end- transaction.*
- I dati sul giornale di transazione sono scritti prima dell'esecuzione delle operazioni sulla base dei dati.
- L'istante in cui la transazione va a buon fine coincide con la scrittura del record *commit* sul giornale di transazione.
- Un guasto dopo tale istante consente di ripristinare lo stato delle transazioni attive al momento del guasto:
  - utilizzando l' *after image* , se sul log è presente il record *commit*
  - utilizzando la *before image* se il record *commit* non è presente.

Meccanismi di ripresa

- Dopo un guasto, il sistema esegue una *procedura di ripresa* , al termine della quale riprende il funzionamento normale. La procedura di ripresa ricostruisce la coerenza della base dei dati.
  - *Ripresa a caldo* . La memoria permanente non è danneggiata, ma alcune transazioni sono rimaste incomplete.
  - *Ripresa a freddo* . La memoria permanente è parzialmente danneggiata.

- La ripresa a caldo utilizza il giornale per rifare o disfare le transazioni incomplete (cioè quelle che non hanno registrato un *end- transaction*).
  - le transazioni da *disfare* sono quelle che:
    - hanno registrato un *begin- transaction* , ma non hanno registrato un *commit*
    - hanno registrato un *abort* , ma non hanno registrato un *end-transaction*

Meccanismi di ripresa

- Sono disfatte copiando nella base dei dati la loro *before image*
  - le transazioni da *rifare* sono quelle che:
    - hanno registrato un *commit* , ma non hanno registrato un *end-transaction*
- Sono rifatte copiando nella base dei dati la loro *after image*
- La ripresa a freddo utilizza una copia storica della base dei dati (*dump, backup*), a cui applica le transazioni complete avvenute dopo la copia, e da cui procede con una ripresa a caldo per le transazioni incomplete.

Protezione

- La protezione consiste nella predisposizione e esecuzione di metodi di controllo dell'uso di risorse da parte di un processo o di un utente. E' motivata da due necessità:
  - prevenire la violazione intenzionale e dannosa di un vincolo di accesso ad una risorsa
  - assicurare che ogni processo utilizzi le risorse in modo coerente con le regole stabilite per i loro utenti.

Protezione

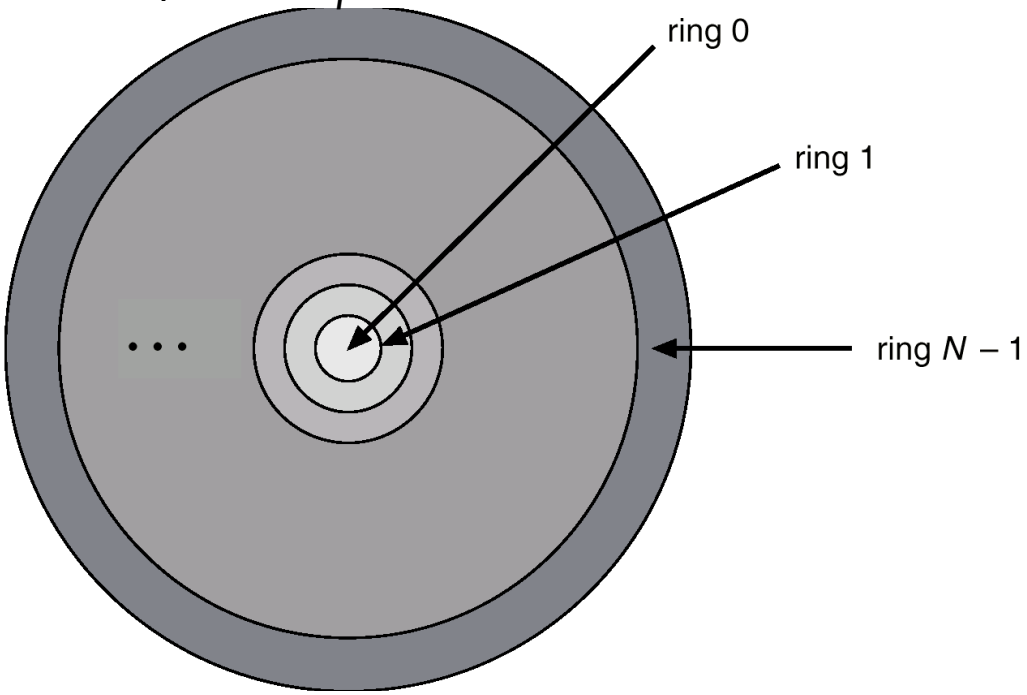
- Un sistema di protezione offre meccanismi per imporre politiche di controllo dell'utilizzo delle risorse:
  - le politiche possono variare in funzione dell'applicazione e del tempo
  - le politiche (cosa deve essere fatto) rimangono separate dai meccanismi (come si fa a farlo)
- Un sistema di protezione si basa sul concetto di oggetto esteso alle risorse di un sistema di elaborazione:
  - ogni risorsa è un oggetto: CPU, memoria, file, processi, mailbox, ...
  - di ogni tipo di risorsa sono definite le operazioni eseguibili: esecuzione, lettura, scrittura, chiamata, invio, ricezione, ...

Dominio di protezione

- Un dominio di protezione definisce un insieme di oggetti e le operazioni che possono essere eseguite su di essi (sottoinsieme delle operazioni proprie di ogni oggetto)
  - La possibilità di eseguire una operazione è detta *diritto di accesso*
  - Un dominio è un insieme di diritti di accesso
  - Un diritto di accesso è una coppia ordinata <oggetto, set di diritti>

Anelli di Multics

- Siano D_i e D_j due anelli di dominio.
- Se $j < I \Rightarrow D_i \subseteq D_j$



Dominio di protezione

- L'associazione tra un processo e un dominio può essere:
  - statica: l'insieme delle risorse disponibili e i diritti di accesso sono costanti per la durata del processo
    - è necessario poter modificare il contenuto di un dominio per evitare di allocare tutte le risorse in blocco, o perché non è possibile conoscere in anticipo tutte le risorse necessarie
    - in alcuni momenti dell'esecuzione un processo può avere più diritti di quelli che gli servono
  - dinamica: i domini sono fissi, ma un processo può passare da un dominio ad un altro
    - è necessario poter creare nuovi domini o modificarne il contenuto per soddisfare esigenze non note al momento della creazione del processo

Cambio di dominio

- Normalmente un dominio corrisponde a un utente, a un processo o a una procedura.
  - *Utente* . I diritti di accesso sono definiti al momento del login.
    - Il cambio di dominio consiste in una coppia logout- login, o nella modifica dell'identità dell'utente (login con un nome differente).
    - Questo schema è implementato nei normali processi sequenziali.
  - *Processo* . I diritti di accesso sono definiti al momento della esecuzione del processo.
    - Il cambio di dominio consiste nella comunicazione con un altro processo avente i diritti necessari.
    - Questo caso generalmente è implementato nelle architetture client- server.
  - *Procedura* . I diritti di accesso sono definiti dalla procedura in esecuzione.
    - Il cambio di dominio avviene chiamando una procedura diversa.
    - Questo schema generalmente è implementato nel meccanismo delle system-call (user- kernel).

Matrice di accesso

- La matrice di accesso M è una struttura dati che descrive i domini e i loro diritti. L'elemento $M_{i,j}$ rappresenta i diritti che un processo in esecuzione nel dominio i ha su una risorsa j .

- La matrice è un meccanismo astratto su cui possono essere realizzate politiche diverse che dipendono da come gli elementi vengono riempiti e modificati, e da come i processi operano nei diversi domini

- I domini stessi possono essere oggetti su cui sono definiti diritti di accesso; In questo modo i processi possono essere abilitati a muoversi tra i domini o a modificarli.

Matrice di accesso

- Sui domini sono definite i diritti *copia* , *possesso* e *controllo*
  - un diritto può essere copiato oppure trasferito ad un altro dominio
  - la copia o il trasferimento possono essere limitati al diritto in sé, oppure anche al diritto di ulteriore copia o trasferimento
  - un dominio può possedere una risorsa, e quindi modificarne i diritti di accesso negli altri domini
  - un dominio può controllare un altro dominio, cioè modificare i diritti di accesso di quest'ultimo

Implementazione della Matrice di Accesso

- Ogni colonna rappresenta la lista dei controlli di accesso per un oggetto
  Definisce chi può compiere quella operazione.

Domain 1 = Read, Write
Domain 2 = Read
Domain 3 = Read

⋮

- Ogni riga rappresenta la lista delle Capability (come una chiave)
  Per ogni dominio, indica quali operazioni sono permesse su quali oggetti.

Object 1 – Read

Object 4 – Read, Write, Execute

Object 5 – Read, Write, Delete, Copy

Uso della Matrice di Accesso

- Se un processo si trova nel dominio D_i tenta di fare un'operazione su un oggetto O_i, allora l'operazione deve trovarsi nella matrice di accesso.
- Può essere estesa per comprendere la protezione dinamica.
  - Operazioni da aggiungere, cancellare i diritti di accesso.
  - Speciali diritti di accesso:
    - *proprietario di* O_i
    - *ccopia l'operazione da* O_i *a* O_j
    - *controllo –* D_i *può modificare i diritti di accesso di* D_j
    - *trasferimento – commuta dal dominio* D_i *a* D_j

Processi e diritti

- Principio del minimo privilegio
  - un processo dovrebbe possedere solamente i diritti sufficienti (e non di più) per eseguire il suo compito
  - il dominio di un processo dovrebbe essere specifico di quel processo (difficile da implementare - i processi sono solitamente raggruppati)

Processi e diritti

- In ogni istante ciascun processo viene eseguito all'interno di uno specifico dominio di protezione

	File1	File2	File3	File4	File5	File6	Print er	Plotte r
1	R	RW						
2			R	RWX	RWX		W	
3						RWX	W	W

Processi e diritti

- i processi possono passare da un dominio all'altro durante l'esecuzione

	File1	File2	File3	File4	File5	File6	Printer	Plotter	Dom 1	Dom 2 lng.	Dom 3
Dom 1	R	RW									
Dom 2			R	RWX	RWX		W				
Dom 3						RWX	W	W			

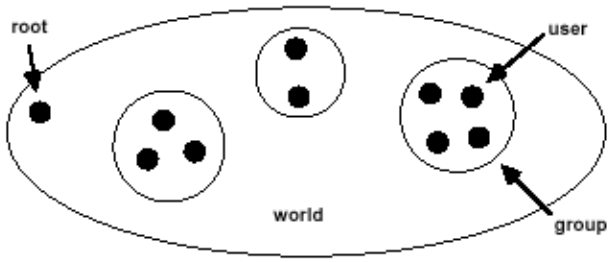
La protezione in Unix

Ogni utente è identificato da due codici numerici, detti *user- id* (*uid*) e *group- id* (*gid*). All'interno di ogni gruppo possono esserci più utenti, e un utente può appartenere ad un solo gruppo.

- Le protezioni di un file sono impostate su tre diritti di accesso: in lettura, in scrittura e in esecuzione.
- Le protezioni sono definite per quattro entità: l'utente proprietario del file (owner), il gruppo a cui l'utente appartiene, tutti gli altri utenti (world), e l'amministratore del sistema (root, o super- user).
- Le protezioni sono definite al momento della creazione del file, e possono essere in seguito

modificate da chi ha diritto di scrittura

sul file e sul directory che lo contiene.



La protezione in Unix

- Il cambio di dominio all'interno di una sessione di login- logout è associato all'esecuzione di un processo (attraverso il corrispondente file di programma), e viene effettuato con un meccanismo chiamato *set-uid* .

- A ogni file eseguibile sono associate due informazioni: l'identificativo del proprietario e un segnalatore di cambio di dominio (*set-uid flag*)

- Se il set-uid flag è *off*, il file è eseguito nel dominio dell'utente che lo esegue. I diritti di accesso sono quindi quelli dell'utente.

- Se il set-uid flag è *on*, il file è eseguito nel dominio del proprietario del file, i cui diritti di accesso sostituiscono, per la durata dell'esecuzione, quelli dell'utente che esegue il programma.

Ssh (Secure Shell)

Permette di:

1 - connettersi ad un altro sistema sulla rete.

2 - eseguire comandi su un sistema remoto.

3 - muovere files da un sistema all'altro sulla rete.

Fornisce un autenticazione severa e una comunicazione sicura.

Ssh deve rimpiazzare completamente i comandi "r" (rlogin, rsh, rcp)

SSHD -Secure Shell Server

Sshd e' il daemon che sta in attesa per controllare le connessioni.

E' preferibile che venga lanciato ed eseguito come ROOT.

Normalmente resta in ascolto su una porta predefinita,

ma e' possibile indirizzarlo su altra porta al momento del lancio.

Comando: sshd [-p port]

Creazione di una chiave sicura

Comando **ssh-keygen:**

Esempio di sessione: (L'input utente e' in bold)

beowulf% **ssh-keygen**

Initializing random number generator...

Generating p: .++ (distance 6)

Generating q: .....++ (distance 110)

Computing the keys...

Testing the keys...

Key generation complete.

(segue.....)

Creazione di una chiave sicura

(….Comando **ssh-keygen**:)

Enter file in which to save the key ($HOME/.ssh/identity):
[RETURN]

Enter passphrase (empty for no passphrase): **litt1e 1amp jumb3d**

Enter same passphrase again: **litt1e 1amp jumb3d**

Nota: (1) La "pass phrase" non resa in echo.

(2) Usare sempre un "pass phrase" non vuota.

(segue…..)

Creazione di una chiave sicura

(….Comando **ssh-keygen**:)

Your identification has been saved in /u/kim/.ssh/identity.

Your public key is:

1024 37 [lots of numbers] kim@beowulf.gw.com

Your public key has been saved in /u/kim/.ssh/identity.pub

Creazione di una chiave sicura

Se volete piu' di una security key create delle chiavi separate.

(probabilmente usate piu' di un account e in diversi ambienti

- Ambiente ufficio
- Proprio sistema in ufficio
- Proprio sistema a casa
- L'account all'univerista'

Potrete usare la chiave appropriata in ogni occasione e controllare le vostre comunicazioni sicure.

Cambiare la vostra "pass phrase"

E' possibile cambiare la “pass phrase”:

beowulf% **ssh-keygen -p**

Enter file in which the key is ($HOME/.ssh/identity): [RETURN]

Enter old passphrase: **litt1e 1amp jumb3d**

Key has comment 'kim@beowulf.gw.com'

Enter new passphrase (empty for no passphrase): **br0wn cow 3ats grass**

Enter same passphrase again: **br0wn cow 3ats grass**

Your identification has been saved with the new passphrase.

Autorizzare gli Accessi

Per permettere ad un utente di accedere al vostro sistema, inserire la sua public key nel file

~/.ssh/authorized_keys.

Tutte le chiavi listate in questo file hanno permesso di accesso.

Inserite in questo file anche la vostra chiave pubblica.

(segue…..)

Autorizzare gli Accessi

(….Per permettere ad un utente di accedere al vostro sistema)

beowulf% **cd ~/.ssh**

beowulf% **cp identity.pub authorized_keys**

Per aggiungere chiavi: usare un qualunque text editor.

Avvertenza: la chiave che inserite deve essere una sola linea.

Permissions delle directory e dei files

Controllare le permission su:

- La home directory
- La directory ~/.ssh
- Il file ~/.ssh/authorized_keys

(segue…..)

Permissions delle directory e dei files

(…..Controllare le permission su)

Esempio di permission corrette:

hrothgar% **cd**

hrothgar% **ls -ld . .ssh .ssh/authorized_keys**

drwxr-xr-x 36 kim kim 4096 Jul 25 02:24 .

drwxr-xr-x 2 kim kim 512 Apr 10 02:30 .ssh

-rw-r--r-- 1 kim kim 1674 Apr 10 02:29 ssh/authorized_keys

(segue…..)

Permissions delle directory e dei files

(…..Controllare le permission su)

Per non permettere l'accesso ai vostri dati, cambiate le permission: SCRIVIBILE SOLO PER VOI.

hrothgar% **cd**

hrothgar% **chmod go-w ssh ssh/authorized_keys**

Connettere sistemi remoti

Normalmente i comandi da usare per connettere sistemi remoti sono: RLOGIN e RSH.

Adesso: SLOGIN e SSH.

(segue…..)

Connettere sistemi remoti

Usando questo meccanismo la password da utilizzare e' la vostra "pass phrase".

beowulf% **slogin hrothgar**

Enter passphrase for RSA key 'kim@beowulf.gw.com':
litt1e 1amp jumb3d

Last login: Wed Oct 16 20:37:00 1996 from idefix

[altre informazioni dal remote host]

hrothgar%

(segue…..)

Connettere sistemi remoti

Si puo' evitare la richiesta della "pass phrase"
mantenendo le chiavi di autenticazione in memoria.

In questo caso, sara' sufficiente:

beowulf% **slogin -l suominen panix.com**

Last login: Sun Oct 13 14:55:17 1996 from idefix.gw.com

[altre informazioni dal remote host]

panix%

Mantenere le autenticazioni in memoria

Questo puo' rendere piu' facile la vita se ne avete continuamente bisogno.

beowulf% **ssh-agent $SHELL**

beowulf%

Se si deve utilizzare XWINDOW

beowulf% ssh-agent startx &

(segue…..)

Mantenere le autenticazioni in memoria

(…..Questo puo' rendere piu' facile la vita)

In questo modo le autenticazioni vengono gestite automaticamente dall'agente.

Alla fine della sessione, tutte le chiavi vengono rimosse.

Lanciare comandi su un sistema remoto

Normalmente i comandi da usare per lanciare comandi su sistemi remoti sono: RSH.

Adesso: SSH.

(segue…..)

Lanciare comandi su un sistema remoto

Il comando ssh puo' essere usato anche per eseguire comandi remoti il cui output e' visualizzato sulla finestra locale.

Al termine dell'esecuzione

il controllo ritorna alla sessione corrente.

beowulf% **ssh hrothgar who**

christos ttyp8 Oct 17 20:42 (milou)

beowulf%

(segue…..)

Lanciare comandi su un sistema remoto

(….. Il comando ssh puo'...)

Se si sta usando un sistema X-Window si puo' usare il seguente comando per far partire un XTERM con una sessione interattiva sul sistema

remoto.

beowulf% ssh -n hrothgar xterm &

[1] 15866

beowulf%

Una nuova finestra dal sistema remoto verra' evidenziata sul vostro display.

Copiare file tra sistemi

Normalmente i comandi da usare per connettere sistemi remoti sono: RCP.

Adesso: SCP.

beowulf% scp -p hrothgar:aliases .

beowulf%

(segue…..)

Cambiare il valore dei default

Si puo' configurare per ogni account il default per i comandi Ssh nel file ~/.ssh/config .

Esiste anche un file di configurazione generale /etc/ssh_config

Ciascuna entry inizia per la parola chiave Host.

(segue…..)

Cambiare il valore dei default

Possono essere usate dei caratteri jolly:

- ? matches any single character
- * matches any sequence zero or more characters

Ogni entry puo' essere configurata agendo su delle opzioni.

(segue…..)

Cambiare il valore dei default

Opzioni:

- Compression yes/no (no)

Controlla se sulla connessione e' usate la compressione.

- CompressionLevel 1-9 (6)

Livello di compressione; 1 e' piu' veloce e 9 e' piu' lento.

La compressione e' da usare su link lenti e con macchine veloci.

(segue…..)

Cambiare il valore dei default

Opzioni:

- FallBackToRsh yes/no (yes)

Se con un sistema remoto non e' possibile una connessione sicura allora puo' provare una connessione non-sicura (con notifica).

- User account (local account)

Specifica il nome dell'utente (account) sul sistema remoto.

Usando questo si puo' omettere l'opzione "-l" con i comandi che lo richiedono

Esempio di file ~/.ssh/config

Host *panix.com

User suominen

Compression no

Host *gw.com

FallBackToRsh no

Host *

Compression yes

CompressionLevel 9

FallBackToRsh yes

KeepAlive no

Nota: Le opzioni vengono accumulate.

Una entry piu' specifica ne sovrascrive una meno specifica (generale).