Chương 9

An ninh hệ thống

9.1 Môi trường an ninh hệ thống
9.2 Cơ sở về mật mã hóa
9.3 Xác nhận người dùng
9.4 Tấn công từ nội bộ
9.5 Tấn công từ bên ngoài
9.6 Các cơ chế bảo vệ
9.7 Các Hệ thống tin cậy

Môi trường an ninh dữ liệu
Các mối đe dọa

Mục tiêu	Mối đe dọa
Bảo mật dữ liệu	Phô bày dữ liệu
Toàn vẹn dữ liệu	Sửa bậy, làm lộn xộn
Sẵn sàng	Từ chối phục vụ

Các mục tiêu an ninh dữ liệu và các mối đe dọa

Các tác nhân bất hợp pháp

Các loại tác nhân bất hợp pháp thông thường :

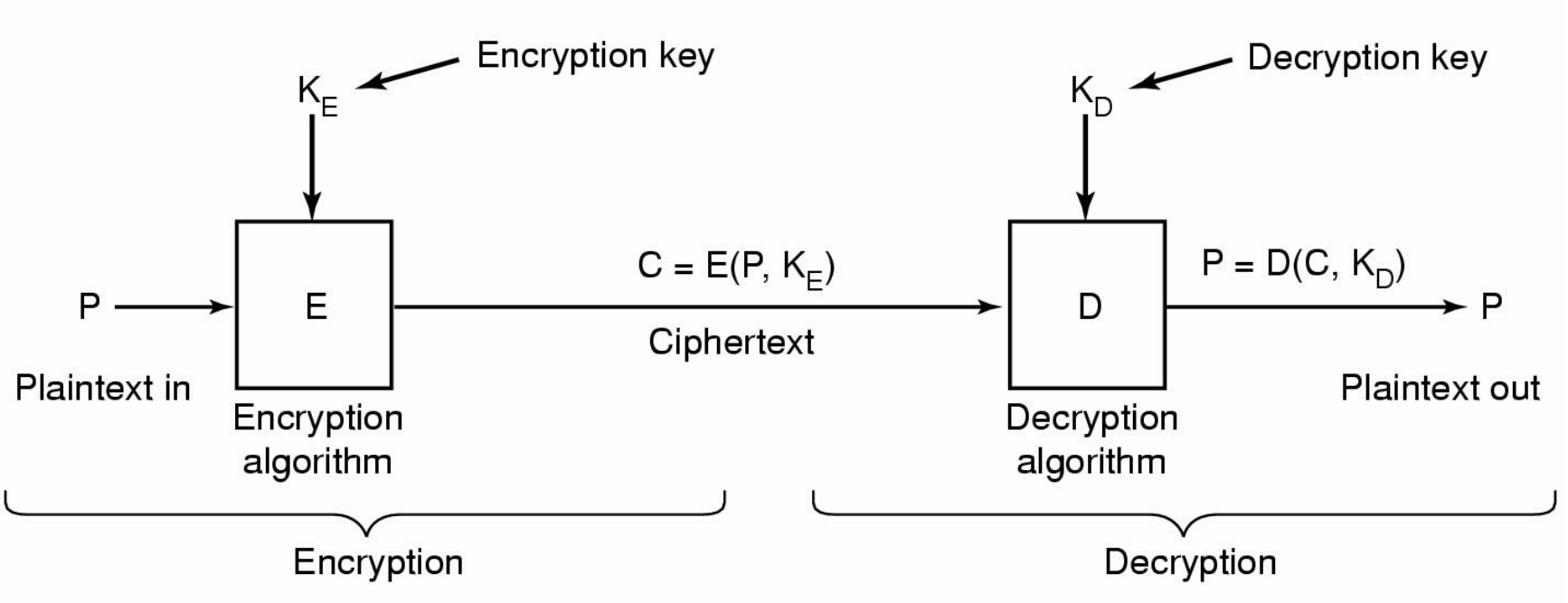
1. Tò mò vô tình của người không chuyên.
2. Sự rình mò của người bên trong hệ thống
3. Cố gắng tống tiền
4. Tình báo thương mại hay quân sự

Sự mất dữ liệu bất ngờ

Các nguyên nhân thông thường :

1. Hoạt động của trời đất
   - hỏa hoạn, lụt lội, chiến tranh
2. Lỗi phần cứng hay phần mềm
   - CPU hư, disk hư, chương trình có lỗi
3. Lỗi con người
   - nhập sai, gắn băng/đĩa sai

Căn bản về mật mã hóa (Cryptography)



Mối quan hệ giữa dữ liệu rõ (plaintext) và dữ liệu mật (ciphertext)

Mật mã hóa dùng khóa bí mật

- Thay thế từng ký tự
  - mỗi ký tự được thay thế bằng ký tự khác nhờ bảng ánh xạ.
- Với 1 khóa mật biến trước,
  - dễ dàng tìm được khóa giải mật.
- Còn được gọi là mật mã hóa đối xứng.

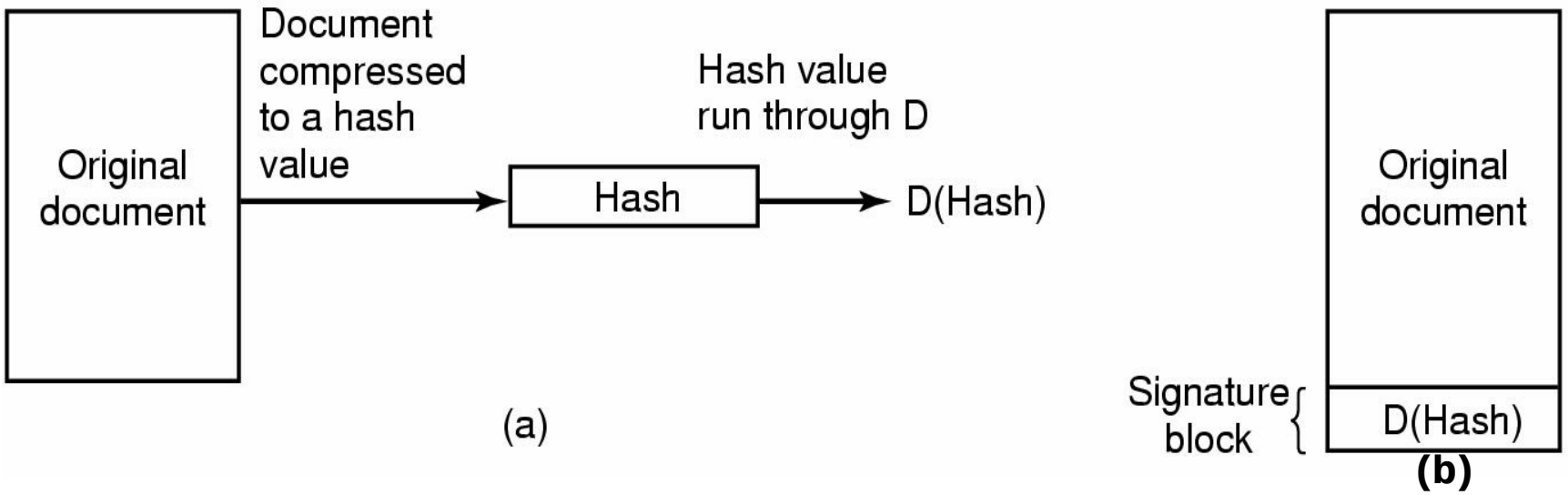
Mật mã hóa dùng khóa bí mật (Public-Key)

- Mỗi user chọn cặp khóa : khóa công khai / khóa bí mật.
  - phân phối khóa công khai cho mội người biết.
  - giữ kín khóa bí mật
- Khóa công khai là khóa để mật mã hóa
  - khóa bí mật là khóa để giải mật.

Các hàm 1 chiều

- Hàm được đặc tả bởi công thức phụ thuộc x, sao cho với x xác định :
  - dễ dàng tính được y = f(x)
- Nhưng nếu biết y
  - thì thực tế, không thể tính được x.

Chữ ký điện tử



- chữ ký điện tử của 1 user trên 1 document là chuỗi byte có được từ qui trình thực hiện a.
- User gởi document + chữ ký của mình trên document đó để chứng thực.

Xác nhận người dùng (User Authentication)

Về nguyên tắc, xác nhận người dùng phải nhận dạng được :

1. cái gì đó mà user cần xác nhận biết.
2. cái gì đó mà user cần xác nhận có.
3. cái gì đó mà user cần xác nhận là cái đó.

Xác nhận user là công việc cần phải làm trước khi cho phép người dùng truy cập hệ thống.

Xác nhận dùng Passwords

LOGIN: ken
PASSWORD: FooBar
SUCCESSFUL LOGIN

(a)

LOGIN: carol
INVALID LOGIN NAME
LOGIN:

(b)

LOGIN: carol
PASSWORD: Idunno
INVALID LOGIN
LOGIN:

(c)

(a) Login thành công

(b) Login bị từ chối ngay sau khi nhập tên user

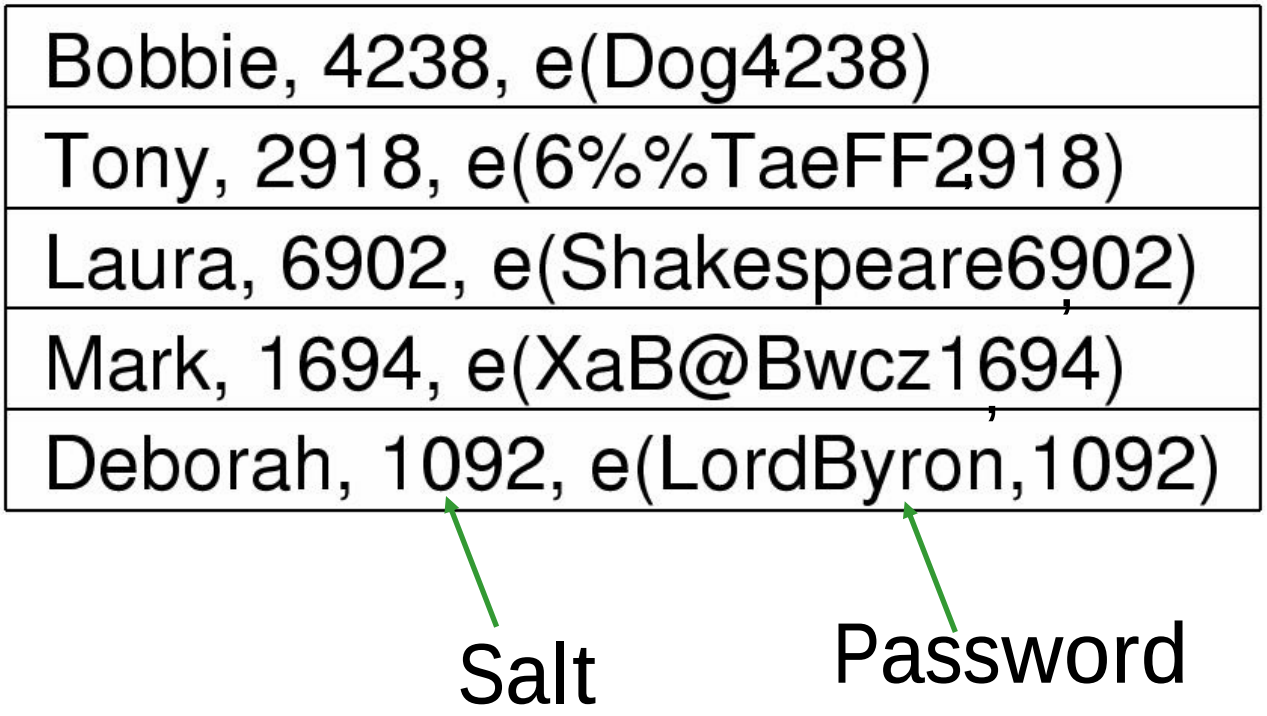
(c) Login bị từ chối sau khi nhập password

Xác nhận dùng Passwords

```
LBL> telnet elxsi
ELXSI AT LBL
LOGIN: root
PASSWORD: root
INCORRECT PASSWORD, TRY AGAIN
LOGIN: guest
PASSWORD: guest
INCORRECT PASSWORD, TRY AGAIN
LOGIN: uucp
PASSWORD: uucp
WELCOME TO THE ELXSI COMPUTER AT LBL
```

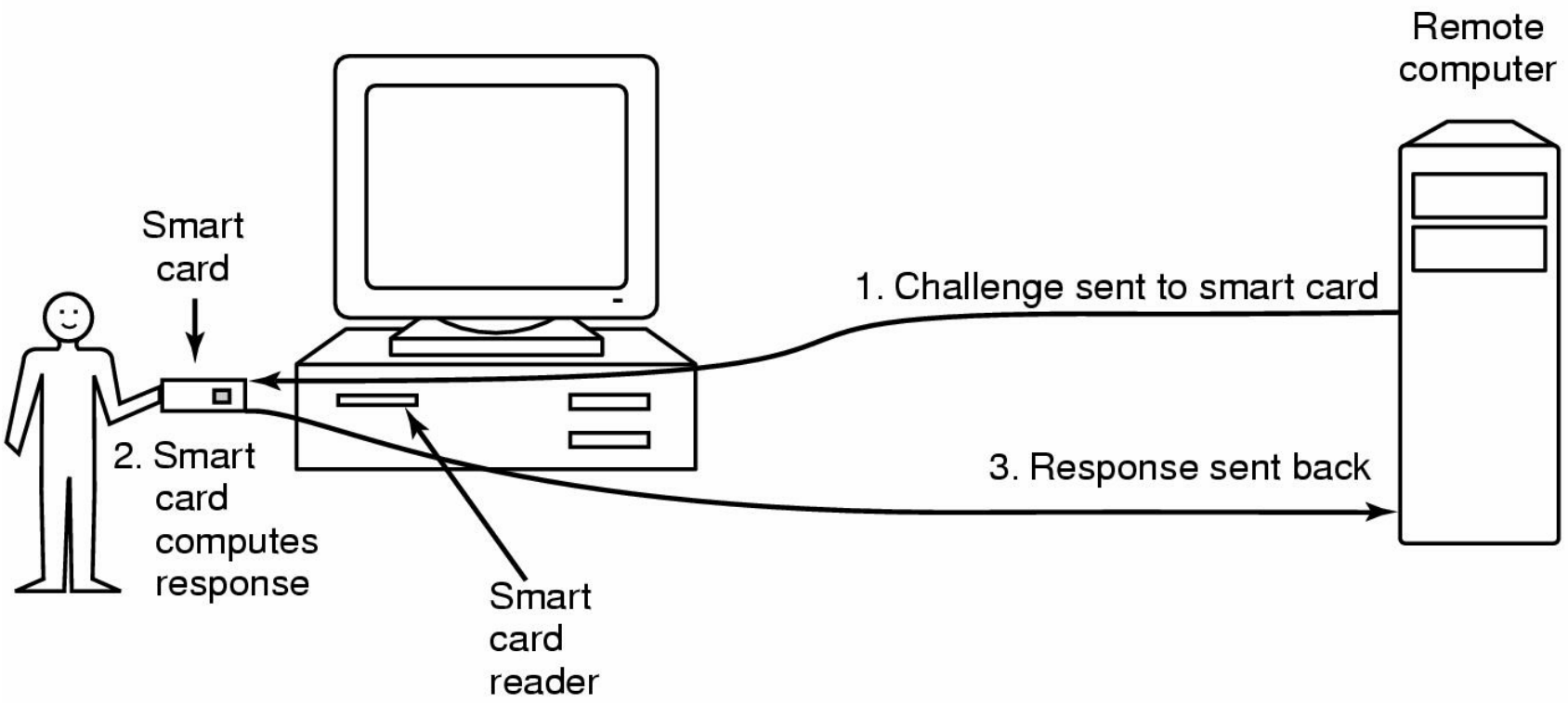
- 1 tình huống mà cracker login thành công do username+ password quá dễ

Xác nhận dùng Passwords



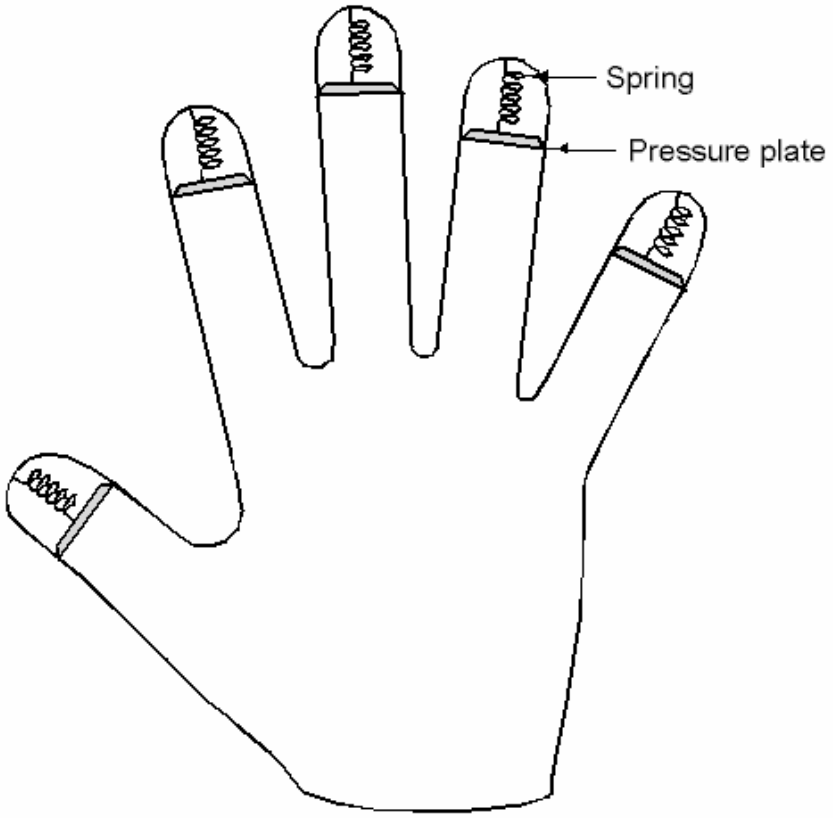
The use of salt to defeat precomputation of encrypted passwords

Xác nhận dùng đối tượng vật lý



- Card từ
  - magnetic stripe cards
  - chip cards: stored value cards, smart cards

Xác nhận dùng sinh trắc học



A device for measuring finger length.

Các biện pháp đối phó với cracker

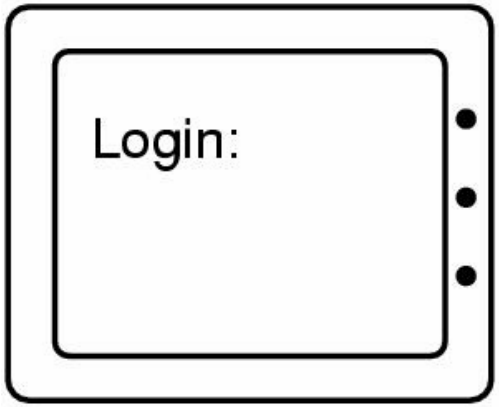
- Hạn chế thời gian user thực hiện login.
- Gọi lại tự động tới số vừa đang ký.
- Hạn chế số lần cố gắng login (3).
- Có database lưu mọi login
- Xem việc nhập name=password như 1 cái bẩy
  - chương trình login cảnh báo cho nhân viên an ninh biết.

An ninh hệ điều hành
Trojan Horses

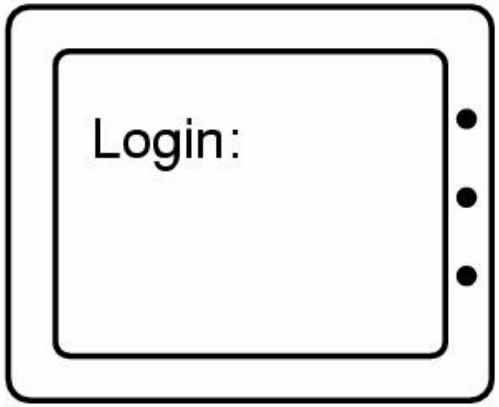
- là chương trình free và để cho người dùng cả tin dễ dàng tiếp cận.
  - nhưng có chứa code thực hiện điều tai hại nào đó.

- đặt version khác của tiện ích phổ dụng trên máy tính của nạn nhân.
  - lừa người dùng chạy chương trình này.

Nhạy theo trình login



(a)



(b)

(a) Màn hình login thật

(b) Màn hình login giả

Bom luận lý

- Người của công ty viết ứng dụng :
  - tiềm tàng chứa code gây hại.
  - chạy OK miễn sao họ nhập password đặt biệt hàng ngày.
  - nếu người lập trình này rời công ty, không có password đặc biệt cho nó, ứng dụng sẽ chạy code gây hại.

Cửa sổ bẩy

```
while (TRUE) {
        printf("login: ");
        get_string(name);
        disable_echoing( );
        printf("password: ");
        get_string(password);
        enable_echoing( );
        v = check_validity(name, password);
        if (v) break;
}
execute_shell(name);
```

(a)

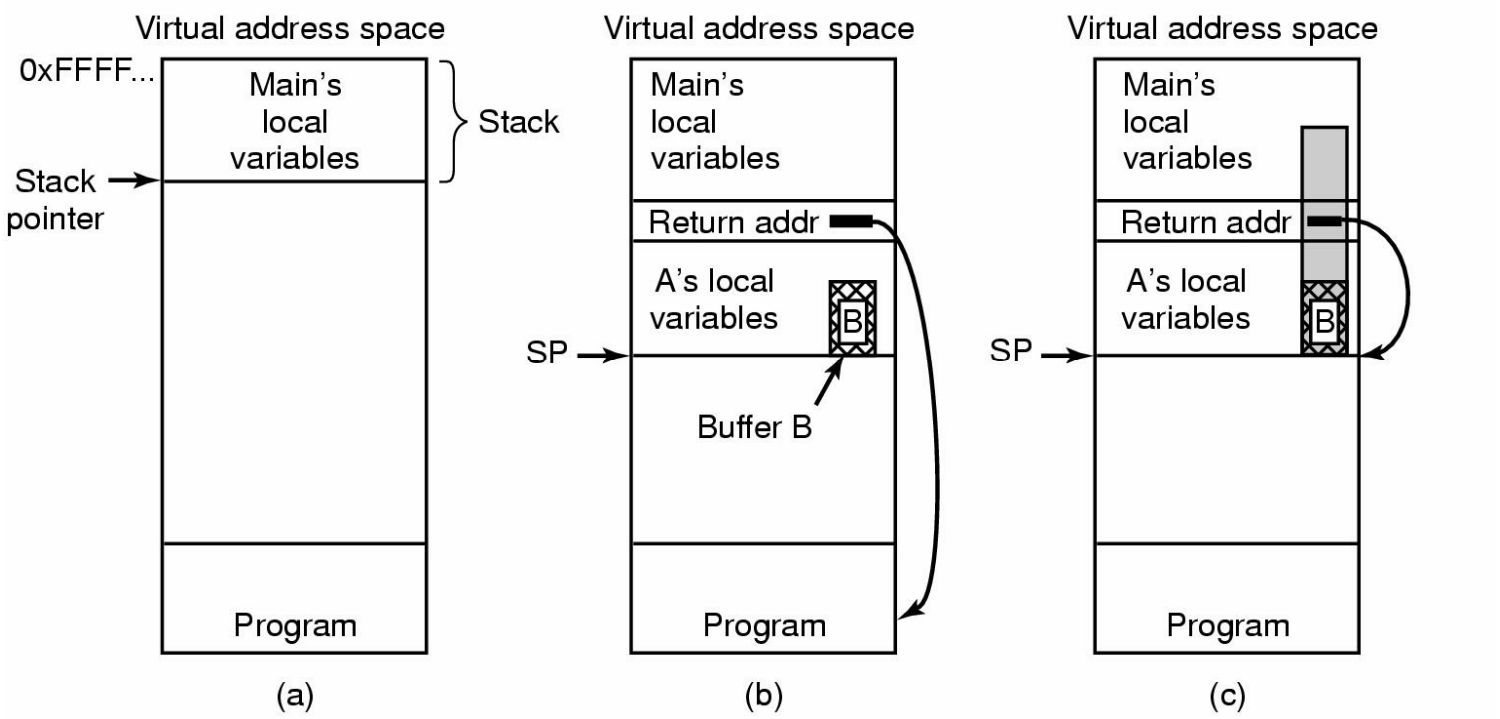
```
while (TRUE) {
        printf("login: ");
        get_string(name);
        disable_echoing( );
        printf("password: ");
        get_string(password);
        enable_echoing( );
        v = check_validity(name, password);
        if (v || strcmp(name, "zzzzz") == 0) break;
}
execute_shell(name);
```

(b)

(a) Code bình thường.

(b) Code có 1 cửa sổ bẩy.

Làm tràn/cạn buffer



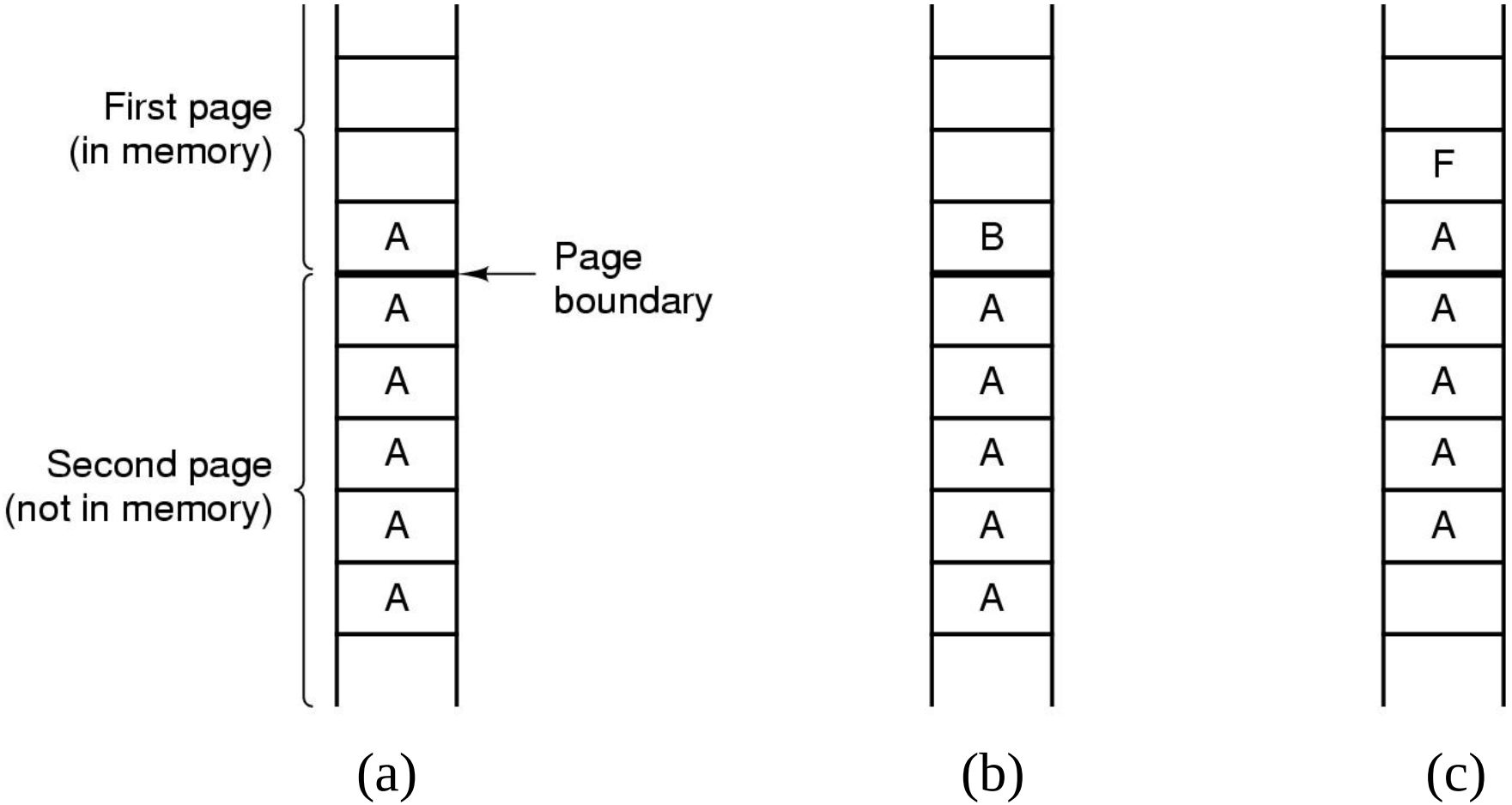
- (a) Situation when main program is running
- (b) After program *A* called
- (c) Buffer overflow shown in gray

Các tấn công vào hệ thống

Các tấn công điển hình :

- Xin bộ nhớ, không gian đĩa, băng rồi chỉ đọc.
- Thử gọi các điểm nhập không có.
- Chạy login rồi ấn nút DEL, RUBOUT, hay BREAK
- Thử hiệu chỉnh các cấu trúc phức tạp của hệ thống.
- Thử làm những việc được yêu cầu không nên làm.
- Thuyết phục người lập trình thêm cửa sổ bẩy vào hệ thống.
- Giả quên password và nhờ admin giúp tìm password.

Các sai lầm về an ninh nổi tiếng



The TENEX – password problem

Các nguyên tắc thiết kế về an ninh

1. Nên để thiết kế hệ thống là việc công công.
2. Mặc định nên chỉ cho tối đa n lần truy xuất (3)
3. Kiểm tra thểm quyền hiện hành
4. Gán process quyền ưu tiên thấp nhất có thể
5. Cơ chế bảo vệ nên :
   - đơn giản
   - đồng nhất
   - trong cấp thấp nhất của hệ thống
6. Sơ đồ an ninh nên chấp nhận được về mặt tâm lý

An ninh mạng

- Mối đe dọa từ ngoài
  - code được gởi tới máy mục tiêu.
  - code được thi hành ở đó, thực hiện điều tai hại.
- Cá mục tiêu của người viết virus
  - lây lan nhanh
  - khó phát hiện
  - khó giết
- Virus = đoạn chương trình có thể tự nhân bản
  - ghép code của nó vào chương trình khác.
  - và thường làm điều tai hại.

Các kịch bản tai hại của virus

- Thư đen
- Từ chối dịch vụ khi virus chạy
- Làm hại phần cứng thường xuyên
- Mục tiêu nhắm vào máy đối thủ :
  - làm hại
  - tình báo
- Các gian trá hèn hạ trong nội bộ công ty :
  - phá hoại các file của nhân viên khác.

Cách virus hoạt động (1)

- Virus thường được viết bằng assembly.
- rồi được chèn vào ứng dụng khác.
  - dùng công cụ được gọi là “dropper”
- Virus ngủ cho đến khi ứng dụng chạy
  - nó sẽ tiêm nhiễm qua các ứng dụng khác.
  - và có thể thi hành đoạn code phá hoại của nó.

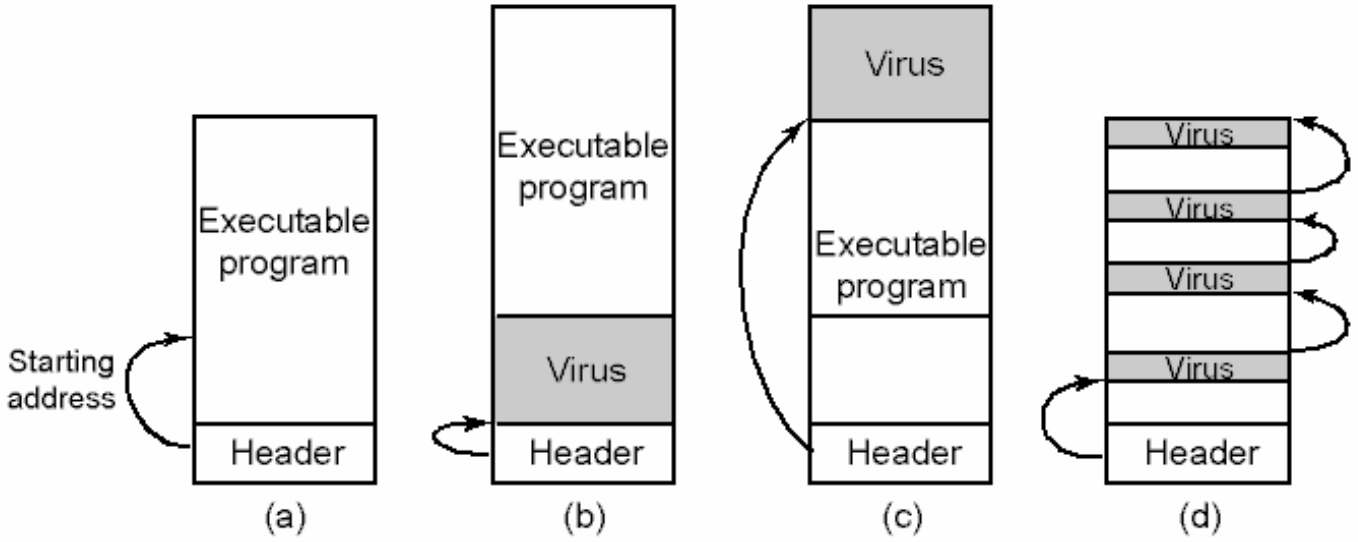
Cách virus hoạt động (2)

Hàm search() cho phép duyệt tìm đệ quy các file khả thi trên Linux.

Virus dùng hàm search() và có thể tiềm nhiễm vào bất kỳ file khả thi nào trên máy.

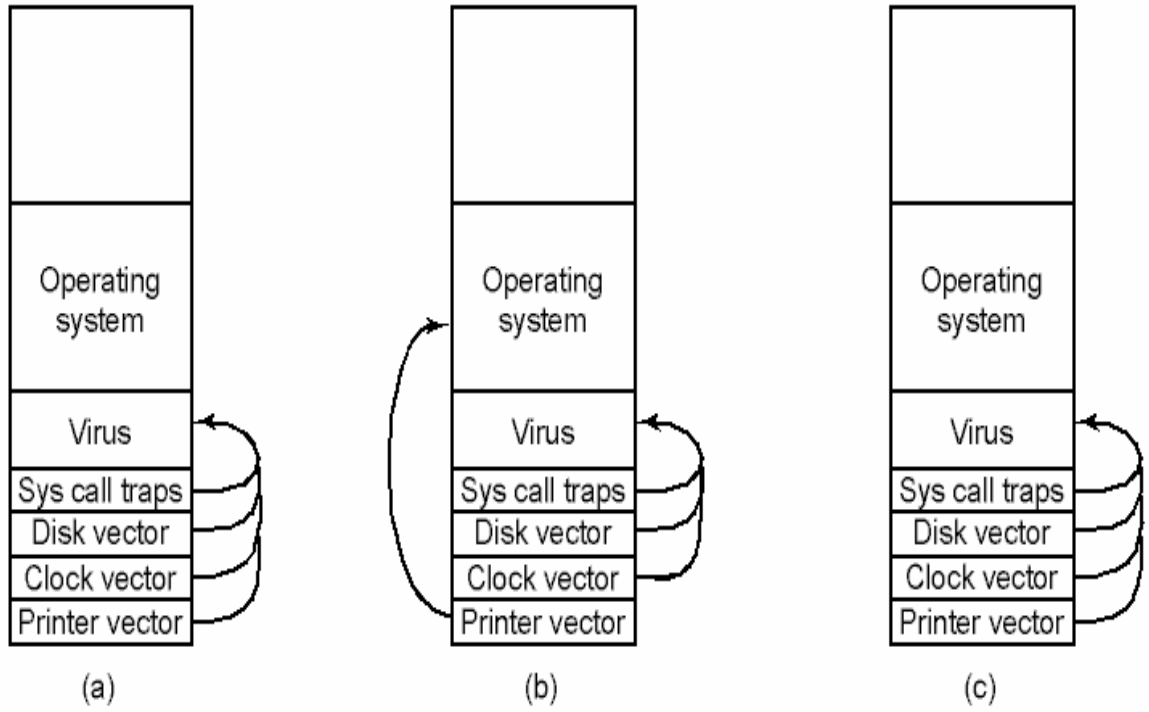
```
#include <sys/types.h>                  /* standard POSIX headers */
#include <sys/stat.h>
#include <dirent.h>
#include <fcntl.h>
#include <unistd.h>
struct stat sbuf;                       /* for lstat call to see if file is sym link */

search(char *dir_name)
{                                       /* recursively search for executables */
    DIR *dirp;                          /* pointer to an open directory stream */
    struct dirent *dp;                  /* pointer to a directory entry */

    dirp = opendir(dir_name);           /* open this directory */
    if (dirp == NULL) return;           /* dir could not be opened; forget it */
    while (TRUE) {
        dp = readdir(dirp);             /* read next directory entry */
        if (dp == NULL) {               /* NULL means we are done */
        chdir ("..");                   /* go back to parent directory */
        break;                          /* exit loop */
    }
    if (dp->d_name[0] == '.') continue; /* skip the . and .. directories */
    lstat(dp->d_name, &sbuf);           /* is entry a symbolic link? */
    if (S_ISLNK(sbuf.st_mode)) continue; /* skip symbolic links */
    if (chdir(dp->d_name) == 0) {       /* if chdir succeeds, it must be a dir */
        search(".");                    /* yes, enter and search it */
    } else {                                /* no (file), infect it */
        if (access(dp->d_name,X_OK) == 0) /* if executable, infect it */
            infect(dp->d_name);
    }
    closedir(dirp);                     /* dir processed; close and return */
}
```

Cách virus hoạt động (3)



a. chương trình khả thi gốc.
b. với virus chèn vào ở đầu file.
c. với virus chèn vào cuối file.
d. với virus chèn vào các chỗ trống trong chương trình.

Cách virus hoạt động (4)



a. sau khi virus đã chiếm ngắt và vector bẩy của hệ thống.

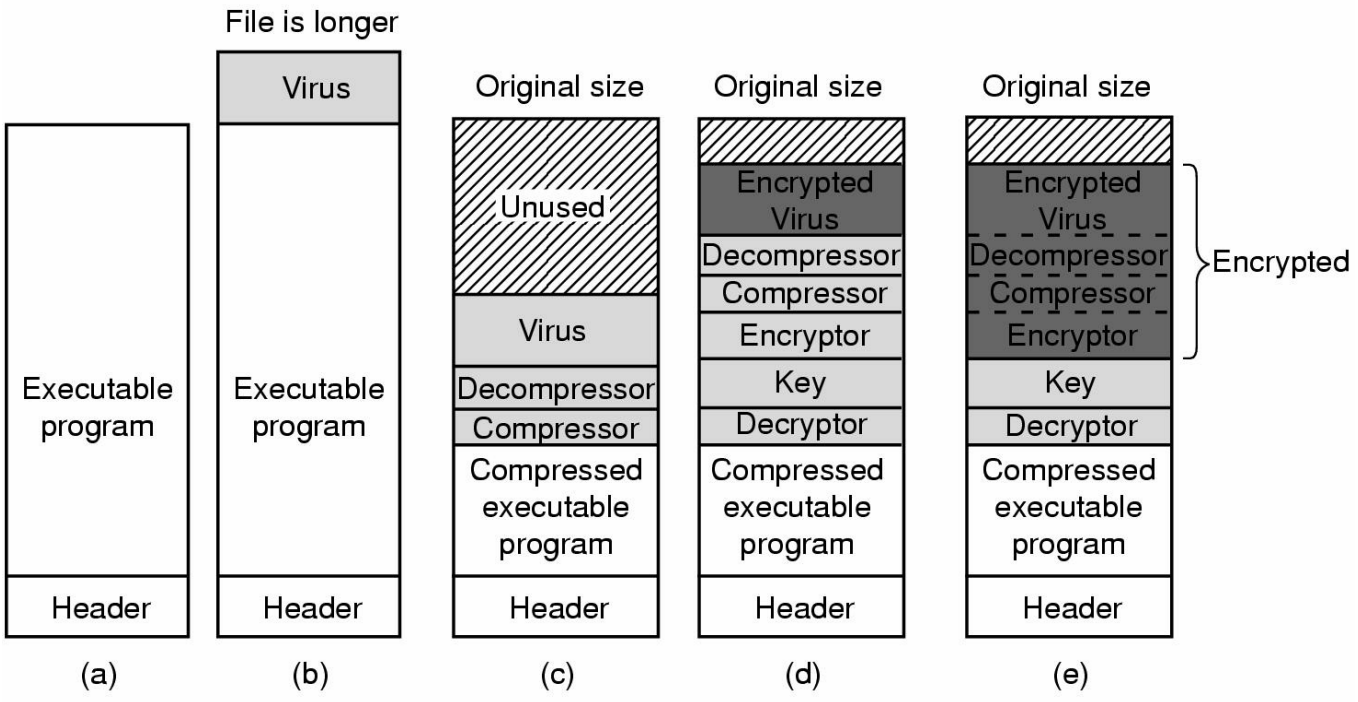
b. sau khi OS chiếm lại được ngắt máy in.

c. sau khi virus phát hiện mất điều khiển trên ngắt máy in và chiếm lại.

Cách virus lây lan

- Virus được đặt ở chỗ dễ được copy nhất.
- Khi được copy, virus :
  - tiêm nhiễm vào các file trên các thiết bị chứa tin
  - và cố gắng lây lan tiếp sang đường mạng.
- Ghép vào e-mail sạch nào đó.
  - khi được xem, virsu dùng mailing list để nhân bản.

Các kỹ thuật Antivirus & Anti-Antivirus



(a) Chương trình gốc
(b) Chương trình bị nhiễm
(c) Chương trình bị nhiễm và bị nén
(d) Virus đã mật mã hóa
(e) virus được nén với code nén bị mật mã hóa.

Các kỹ thuật Antivirus & Anti-Antivirus

```
MOV A,R1
ADD B,R1
ADD C,R1
SUB #4,R1
MOV R1,X
```
(a)

```
MOV A,R1
NOP
ADD B,R1
NOP
ADD C,R1
NOP
SUB #4,R1
NOP
MOV R1,X
```
(b)

```
MOV A,R1
ADD #0,R1
ADD B,R1
OR R1,R1
ADD C,R1
SHL #0,R1
SUB #4,R1
JMP .+1
MOV R1,X
```
(c)

```
MOV A,R1
OR R1,R1
ADD B,R1
MOV R1,R5
ADD C,R1
SHL R1,0
SUB #4,R1
ADD R5,R5
MOV R1,X
MOV R5,Y
```
(d)

```
MOV A,R1
TST R1
ADD C,R1
MOV R1,R5
ADD B,R1
CMP R2,R5
SUB #4,R1
JMP .+1
MOV R1,X
MOV R5,Y
```
(e)

Những thí dụ về virus đa hình (polymorphism) các đoạn code trên đều miêu tả cùng 1 virus.

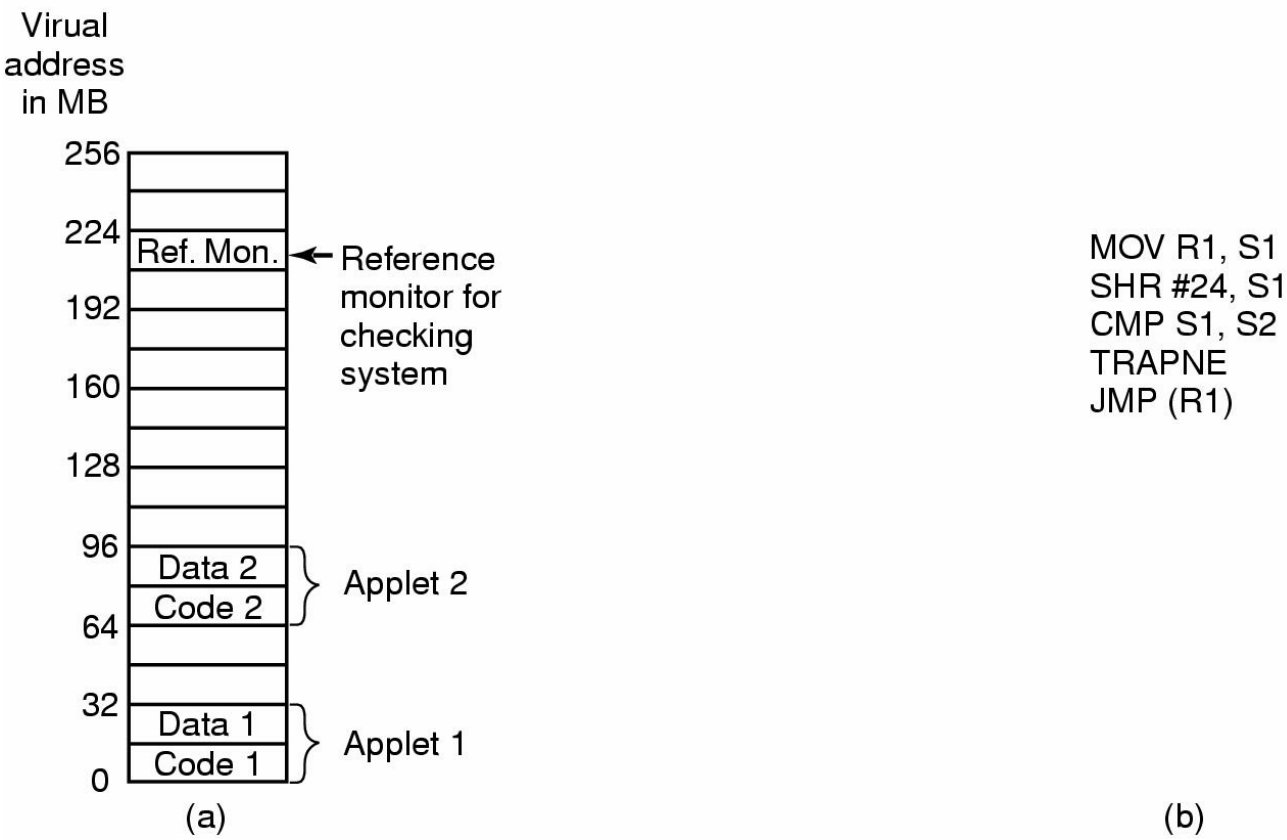
Các kỹ thuật Antivirus & Anti-Antivirus

- Kiểm tra tính toàn vẹn.
- Kiểm tra hành vi.
- Tránh virus
  - dùng HĐH sạch
  - chỉ cài ứng dụng "shrink-wrapped"
  - dùng ứng dụng diệt virus.
  - không click chuột trên các attach của e-mail.
  - backup thông tin thường xuyên
- Phục hồi khi bị virus tấn công
  - dừng máy, boot lại từ đĩa sạch, chạy trình diệt virus.

Sâu Internet

- Gồm 2 ứng dụng :
  - bootstrap để tải sâu lên mạng
  - bản thân sâu
- Đầu tiên Sâu ẩn mình
- rồi tự nhân bản lên các máy mới.

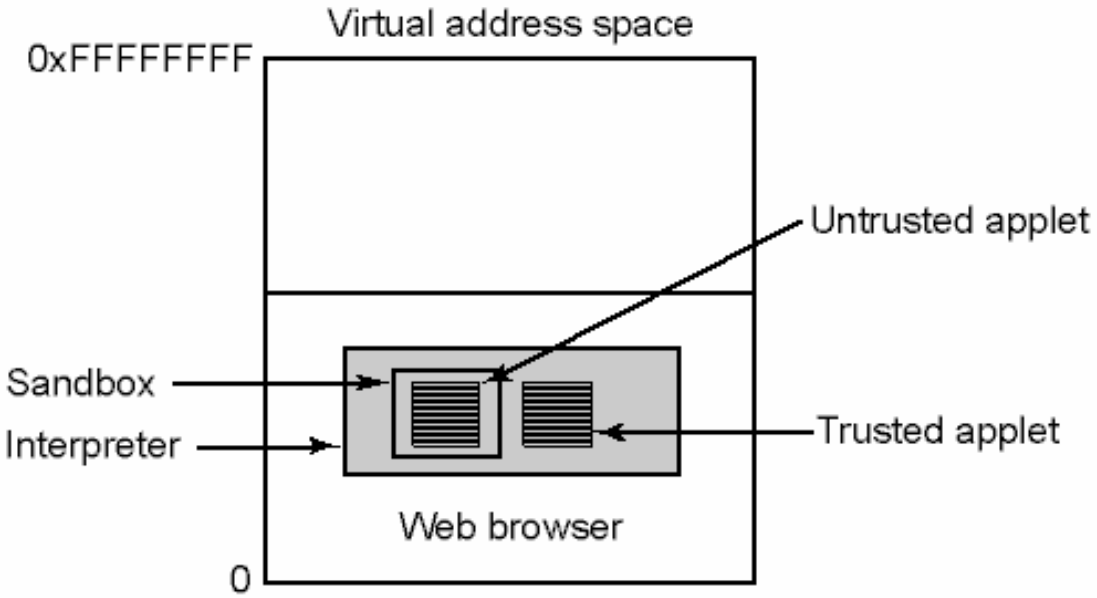
Mobile Code (1) Sandboxing



(a) Memory divided into 1-MB sandboxes

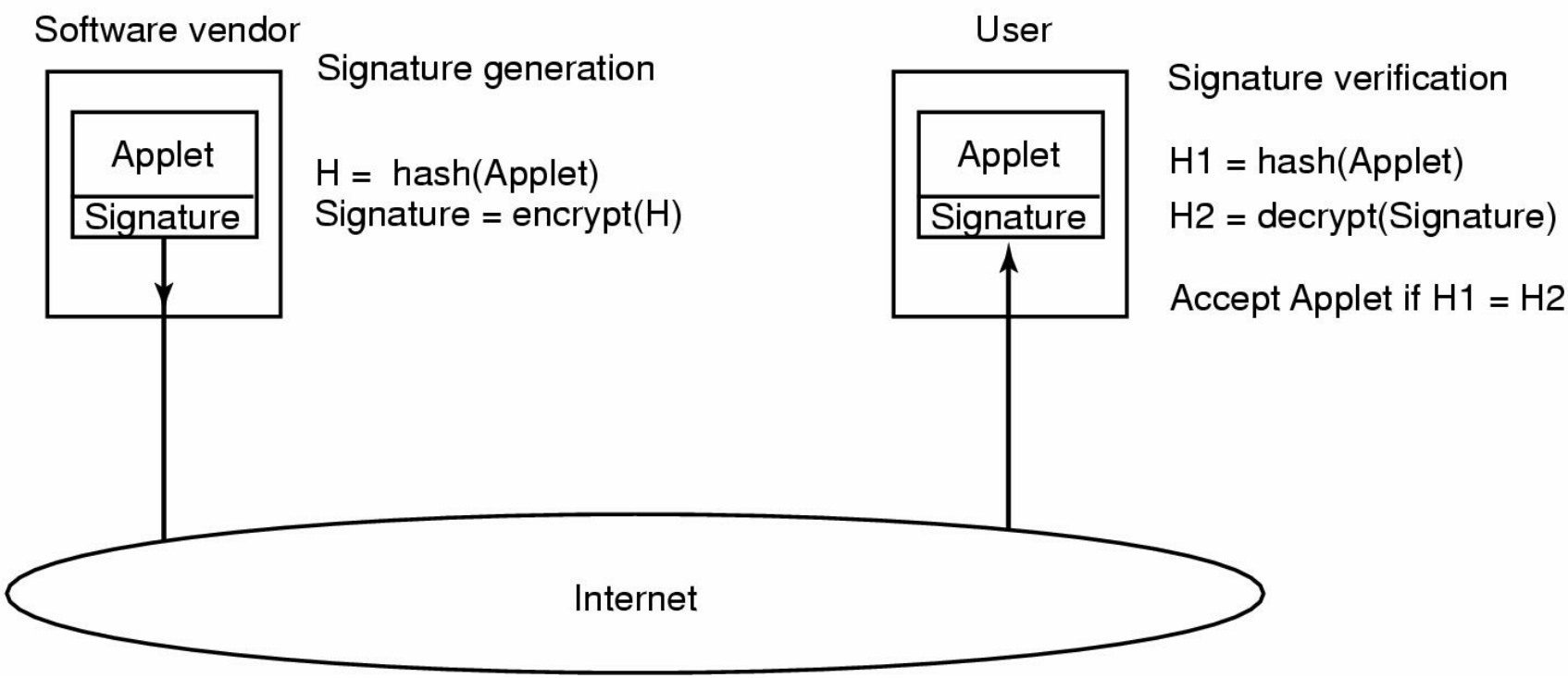
(b) One way of checking an instruction for validity

Mobile Code (2)



Applets can be interpreted by a Web browser

Mobile Code (3)



How code signing works

Java Security (1)

- A type safe language
  - compiler rejects attempts to misuse variable

- Checks include …
  1. Attempts to forge pointers
  2. Violation of access restrictions on private class members
  3. Misuse of variables by type
  4. Generation of stack over/underflows
  5. Illegal conversion of variables to another type

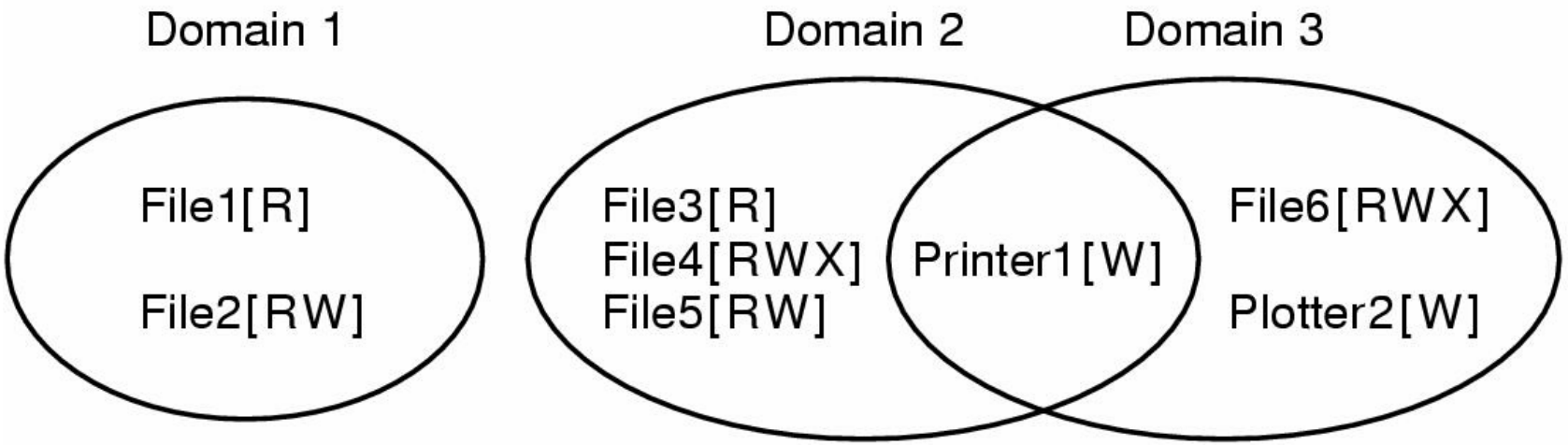
Java Security (2)

URL	Signer	Object	Action
www.taxprep.com	TaxPrep	/usr/susan/1040.xls	Read
*		/usr/tmp/*	Read, Write
www.microsoft.com	Microsoft	/usr/susan/Office/–	Read, Write, Delete

Examples of specified protection with JDK 1.2

Protection Mechanisms

Protection Domains (1)



Examples of three protection domains

Protection Domains (2)

	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

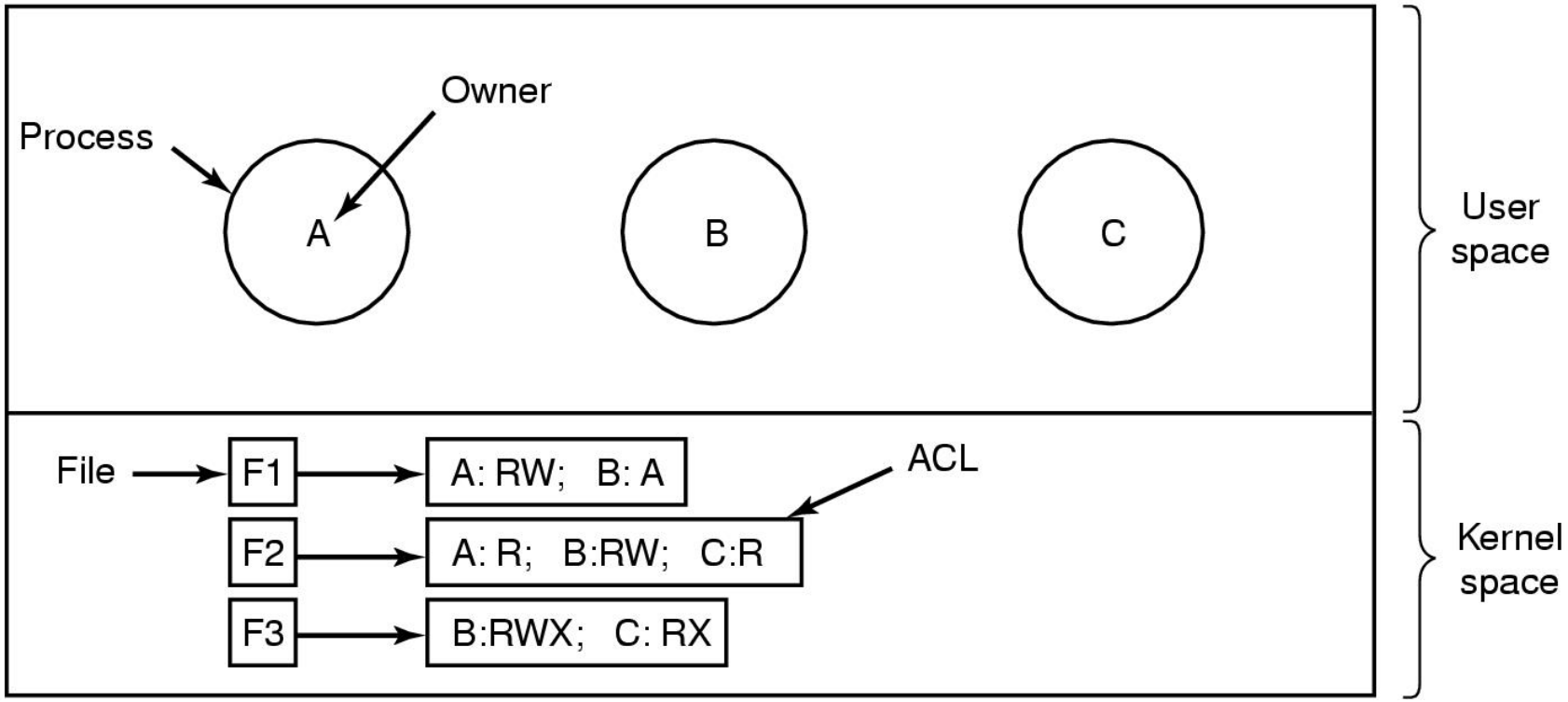
A protection matrix

Protection Domains (3)

	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			

A protection matrix with domains as objects

Access Control Lists (1)



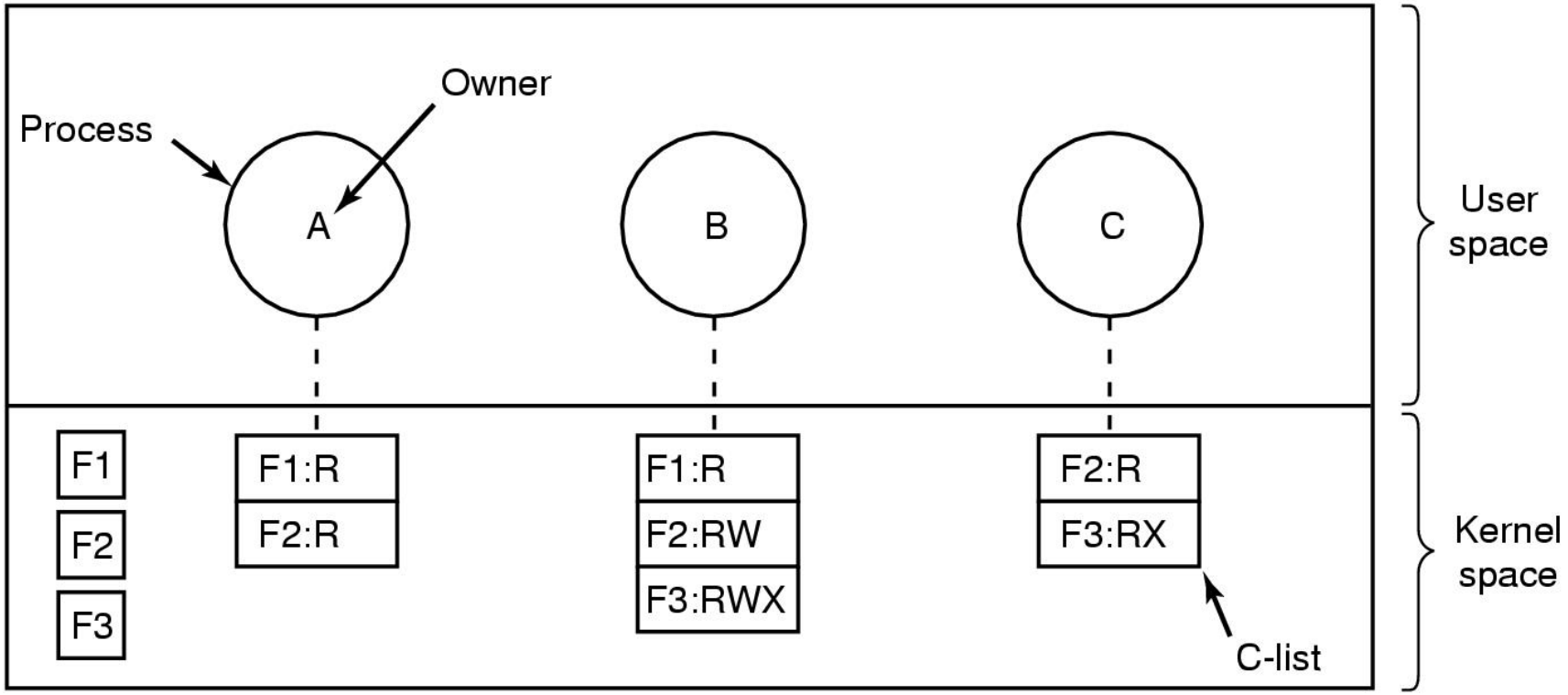
Use of access control lists of manage file access

Access Control Lists (2)

File	Access control list
Password	tana, sysadm: RW
Pigeon_data	bill, pigfan: RW; tana, pigfan: RW; ...

Two access control lists

Capabilities (1)



Each process has a capability list

Capabilities (2)

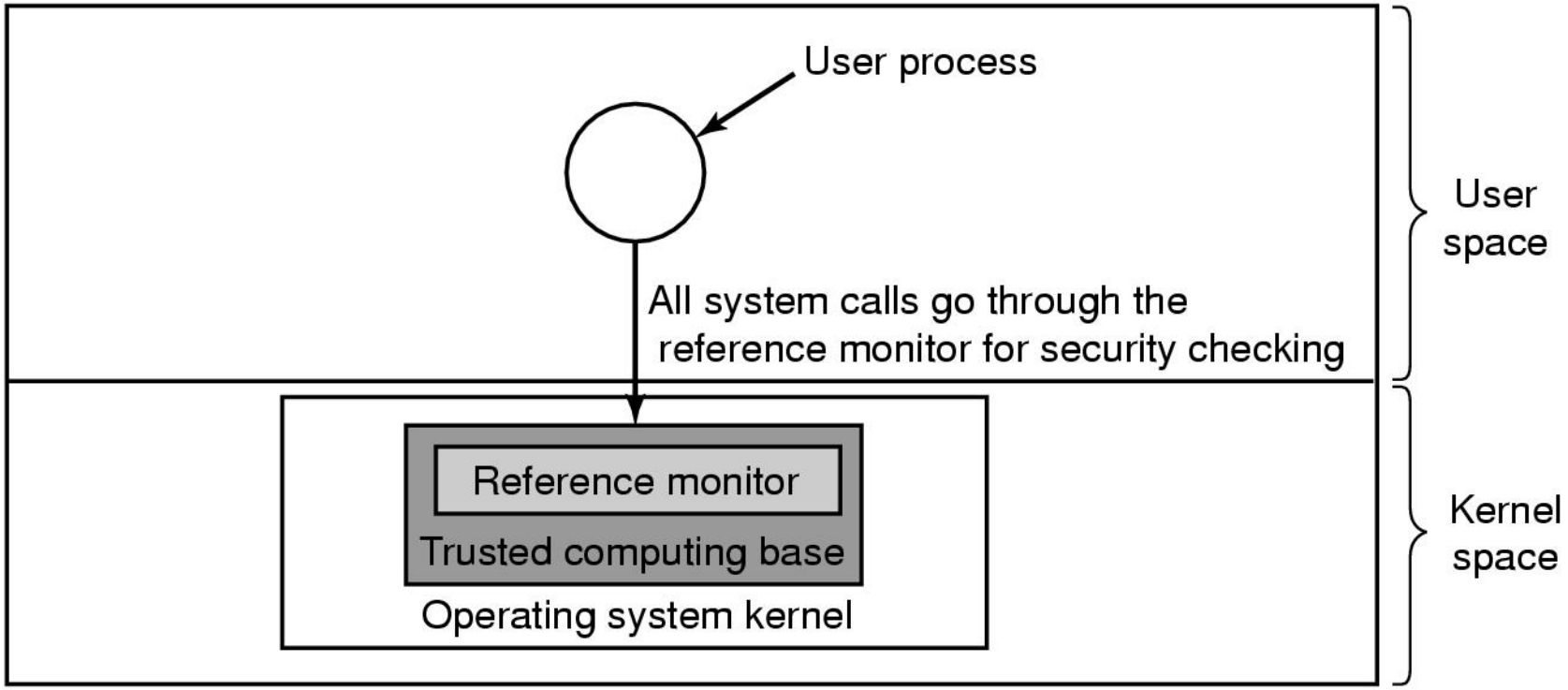
- Cryptographically-protected capability

Server	Object	Rights	f(Objects, Rights, Check)

- Generic Rights
  1. Copy capability
  2. Copy object
  3. Remove capability
  4. Destroy object

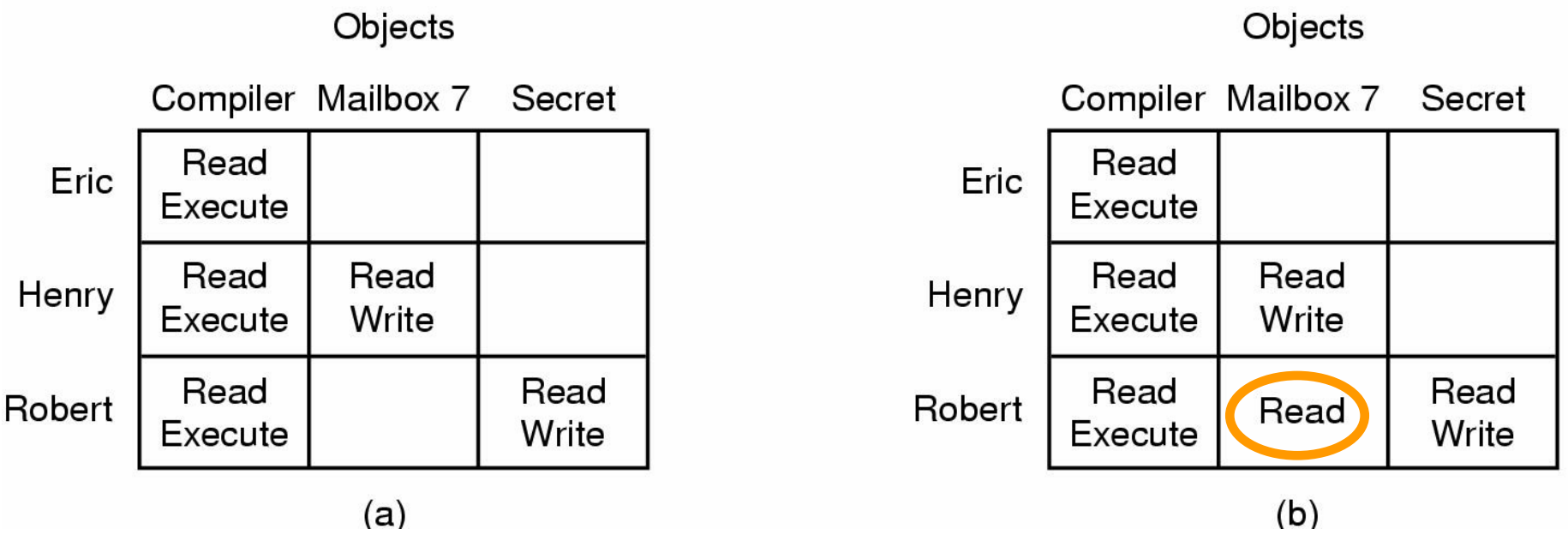
Trusted Systems

Trusted Computing Base



A reference monitor

Formal Models of Secure Systems



Objects

	Compiler	Mailbox 7	Secret
Eric	Read Execute		
Henry	Read Execute	Read Write	
Robert	Read Execute		Read Write

(a)

Objects

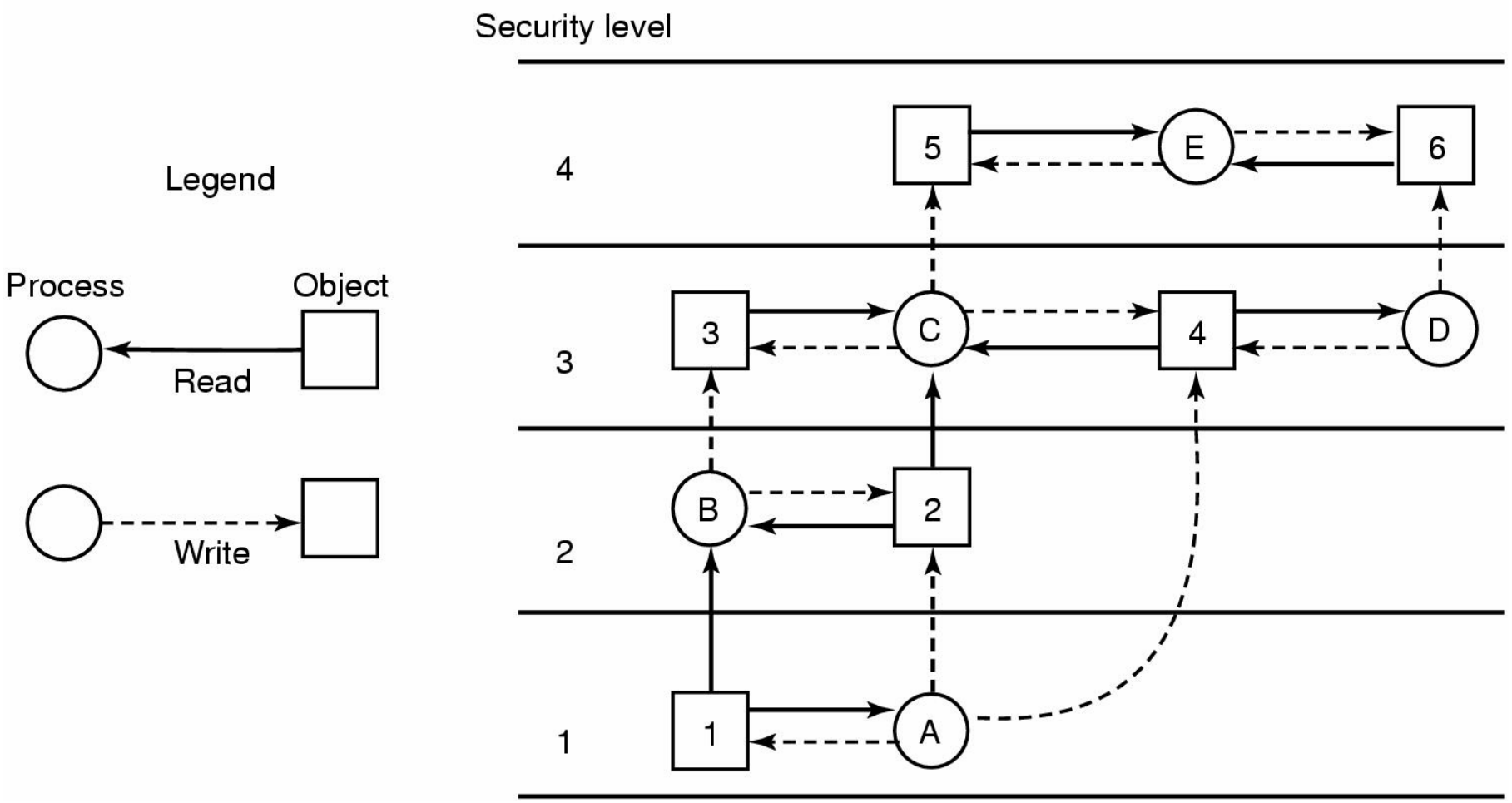
	Compiler	Mailbox 7	Secret
Eric	Read Execute		
Henry	Read Execute	Read Write	
Robert	Read Execute	Read	Read Write

(b)

(a) An authorized state

(b) An unauthorized state

Multilevel Security (1)



The Bell-La Padula multilevel security model

Multilevel Security (2)

The Biba Model

- Principles to guarantee integrity of data

1. Simple integrity principle
   - process can write only objects at its security level or lower
2. The integrity * property
   - process can read only objects at its security level or higher

Orange Book Security (1)

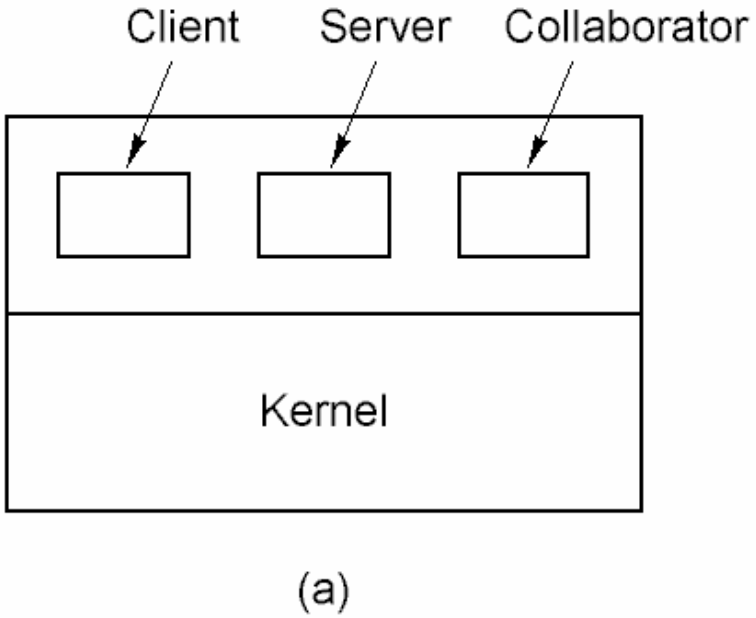
Criterion	D	C1	C2	B1	B2	B3	A1
Security policy							
Discretionary access control		X	X	→	→	X	→
Object reuse			X	→	→	→	→
Labels				X	X	→	→
Label integrity				X	→	→	→
Exportation of labeled information				X	→	→	→
Labeling human readable output				X	→	→	→
Mandatory access control				X	X	→	→
Subject sensitivity labels					X	→	→
Device labels					X	→	→
Accountability							
Identification and authentication		X	X	X	→	→	→
Audit			X	X	X	X	→
Trusted path					X	X	→

- Symbol X means new requirements
- Symbol -> requirements from next lower category apply here also

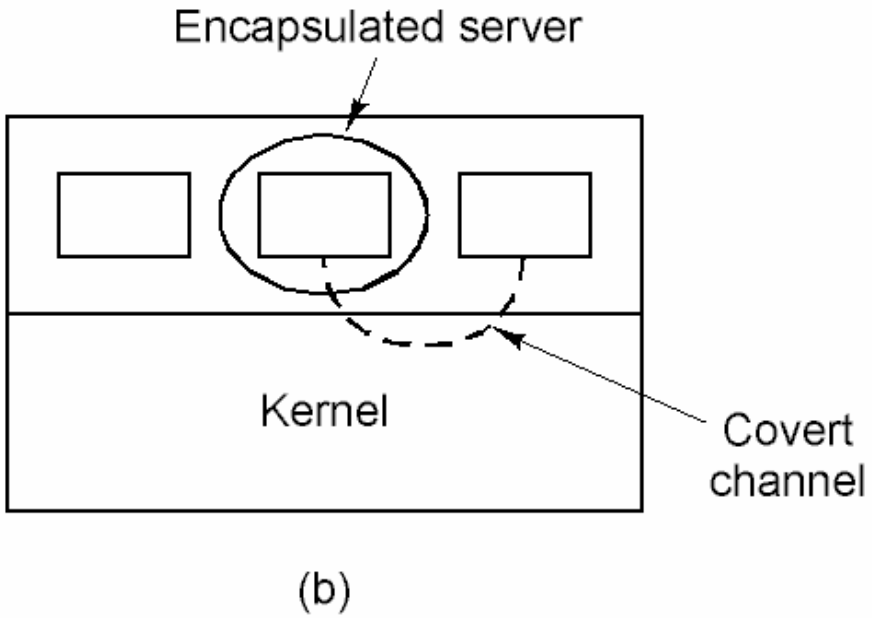
Orange Book Security (2)

Assurance							
System architecture		X	X	X	X	X	→
System integrity		X	→	→	→	→	→
Security testing		X	X	X	X	X	X
Design specification and verification				X	X	X	X
Covert channel analysis					X	X	X
Trusted facility management					X	X	→
Configuration management					X	→	X
Trusted recovery						X	→
Trusted distribution							X
Documentation							
Security features user's guide		X	→	→	→	→	→
Trusted facility manual		X	X	X	X	X	→
Test documentation		X	→	→	X	→	X
Design documentation		X	→	X	X	X	X

Covert Channels (1)

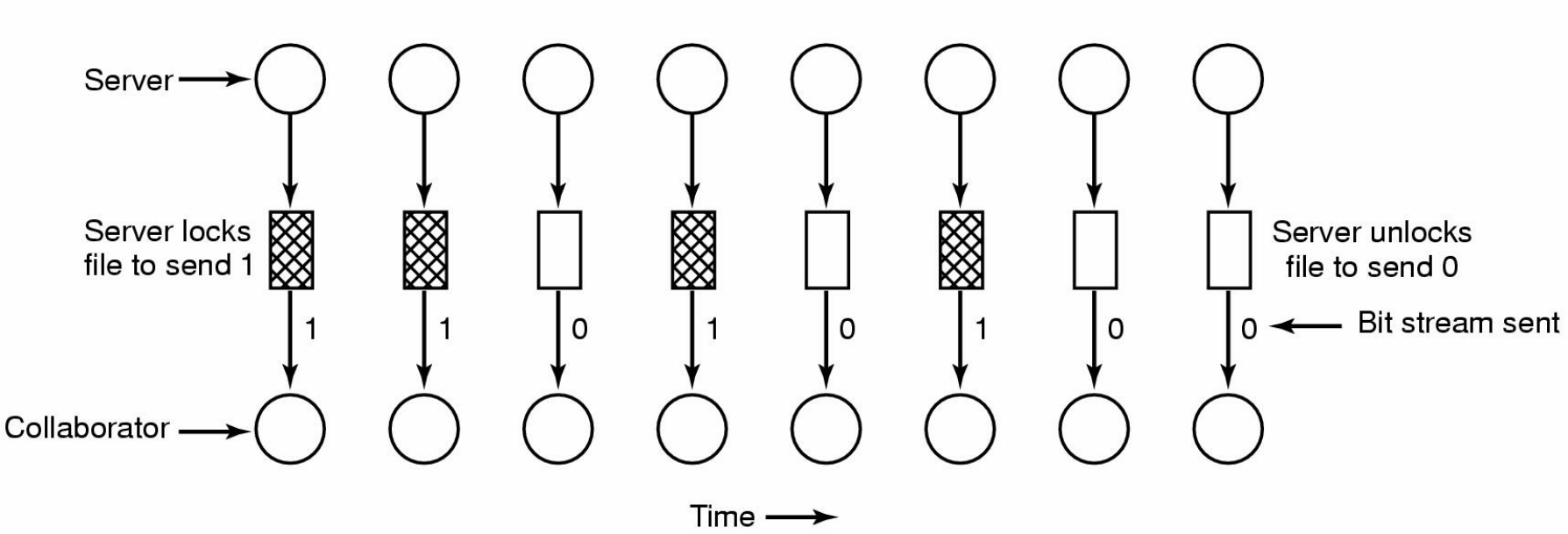


Client, server and collaborator processes



Encapsulated server can still leak to collaborator via covert channels

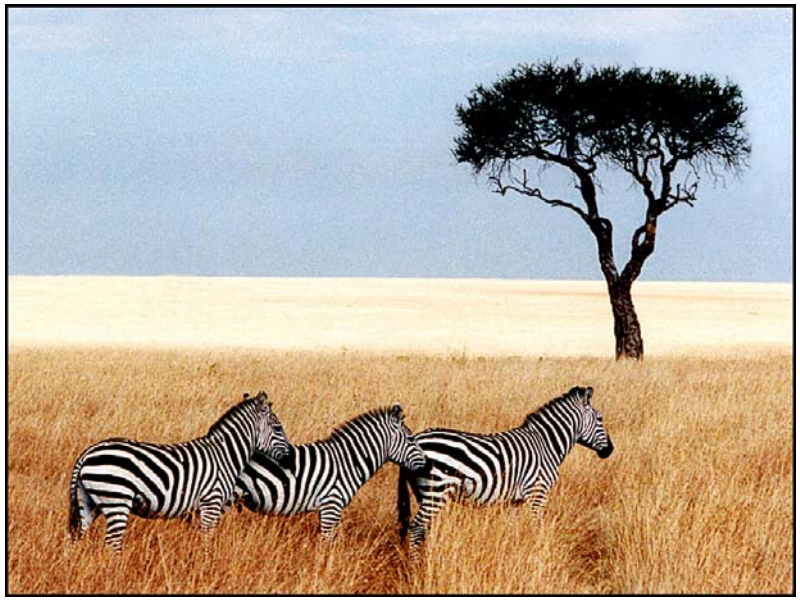
Covert Channels (2)



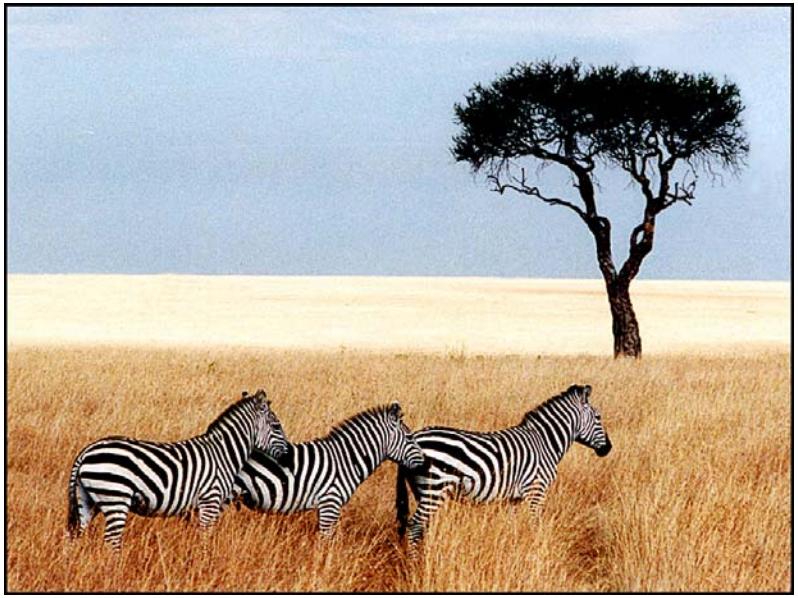
A covert channel using file locking

Covert Channels (3)

- Pictures appear the same
- Picture on right has text of 5 Shakespeare plays
  - encrypted, inserted into low order bits of color values



Zebras



Hamlet, Macbeth, Julius Caesar
Merchant of Venice, King Lear