Sistemas Operativos II

Clase 5

Mecanismos de Protección

Universidad Arturo Jauretche
Ingeniería Informática

Docentes:

Coordinador: Ing. Eduardo Kunysz

Profesores: Ing. Jose Vera

Ing. Daniel Alonso

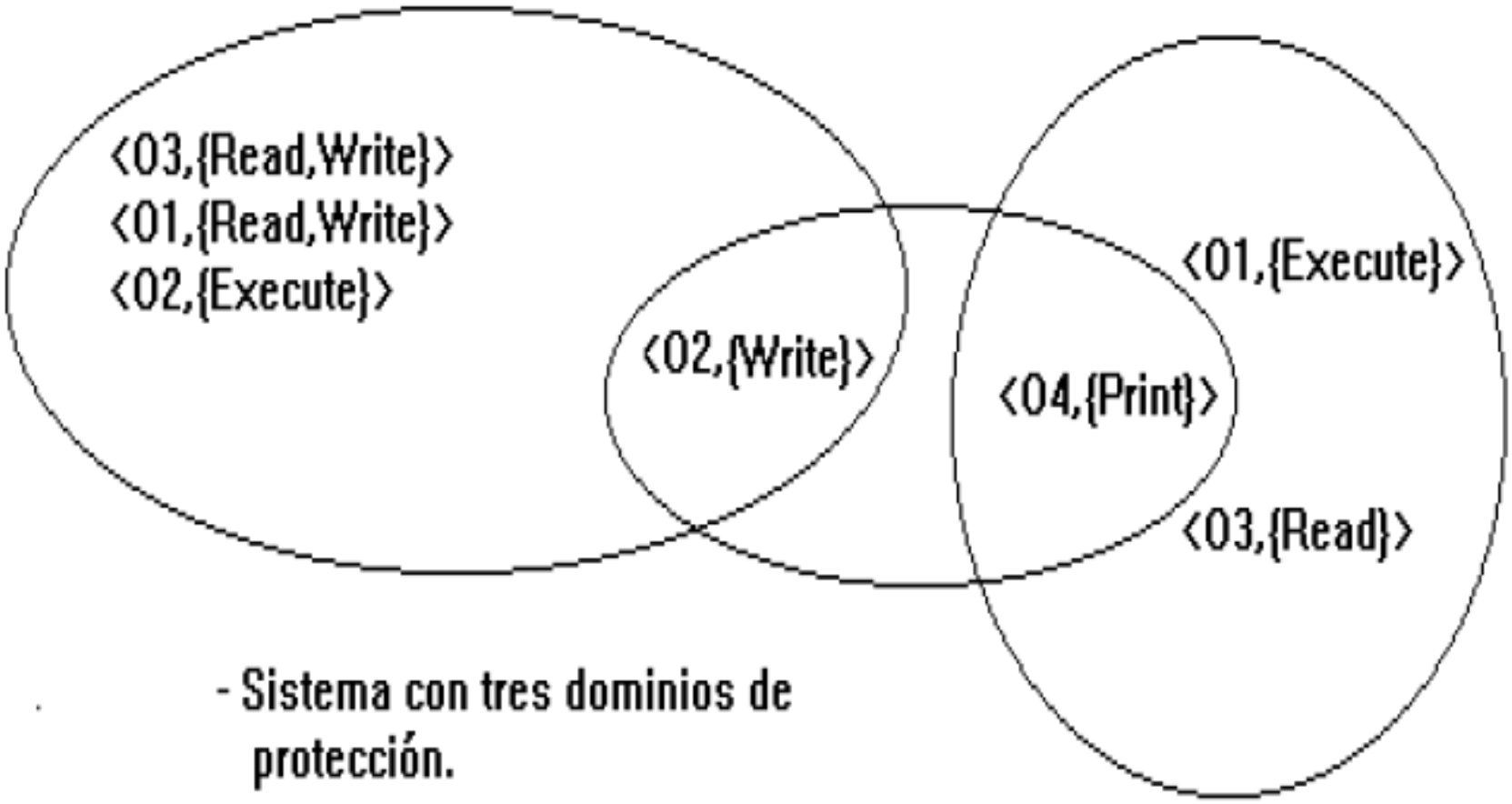
Definición

- **Protección:**
- Se refiere al mecanismo para controlar el acceso de programas, procesos o usuarios a los recursos definidos del sistema.

Elementos de un esquema de protección

- **DOMINIO:** Entidad activa que realiza acciones (un proceso, un usuario, una CPU, etc.)
- **OBJETO:** Entidad pasiva que recibe las acciones realizadas por los dominios (memoria, un periférico, un usuario, etc.)
- **DERECHO DE ACCESO:** Calificación respecto del **tipo de acción que puede realizarse** (por ej.: lectura, escritura, etc.)

Sistema con tres Dominios



- Sistema con tres dominios de protección.

Implementación de esquemas de protección

- MATRIZ DE ACCESO
- TABLA GLOBAL
- LISTAS DE ACCESO
- LISTAS DE CAPACIDADES
- LOCK/KEY

Matriz de Accesos

- Matriz con los objetos en las columnas y los dominios en las filas.
- En la intersección de una fila/columna se encuentra el derecho de acceso.
- Si el derecho de acceso está en blanco implica que no se puede acceder.

objetos / Dominios	Compiladores	Archivos
Programadores	Ejecutar	Leer
System Programmer	Leer/grabar/ejecutar	
Usuarios	Ejecutar	Leer/grabar

Tabla Global

- Un **conjunto de uplas (<Dominio, Objeto, Derecho de acceso>)** con todos los elementos NO nulos de la Matriz de acceso.

- <Programadores, Compilador, Ejecutar>
- <Programadores, Archivos, Lectura>
- <System Programmer, Compilador, Lectura/ ejecución/ Grabación >
- <Usuarios, Compilador, Ejecutar>
- <Usuarios, Archivos, Lectura/grabación>

objetos / Dominios	Compiladores	Archivos
Programadores	Ejecutar	Leer
System Programmer	Leer/grabar/ejecutar	
Usuarios	Ejecutar	Leer/grabar

Listas de Acceso

- Se construyen por **objeto (**las LCA - **listas de control** de acceso- son un subconjunto de éstas)

- **Compilador** < Programadores, Ejecutar>
  < System Programmer, Lect./Grab./Ej>
  < Usuarios, Ejecutar>
- **Archivos** < Programadores, Lectura>
  < Usuarios, Lectura/Grabación>

objetos / Dominios	Compiladores	Archivos
Programadores	Ejecutar	Leer
System Programmer	Leer/grabar/ejecutar	
Usuarios	Ejecutar	Leer/grabar

Listas de Capacidad

- Se construyen por **dominio (las LCU - listas de control de usuarios-** son un subconjunto de éstas)
- **Programador** < Compilador, Ejecutar>
  < Archivos, Leer>
- **Usuarios** < Compilador, Ejecutar>
- < Archivos, Leer/Grabar>
- **Syst. Programmer** < Compilador, Lect/Ej/Grab>

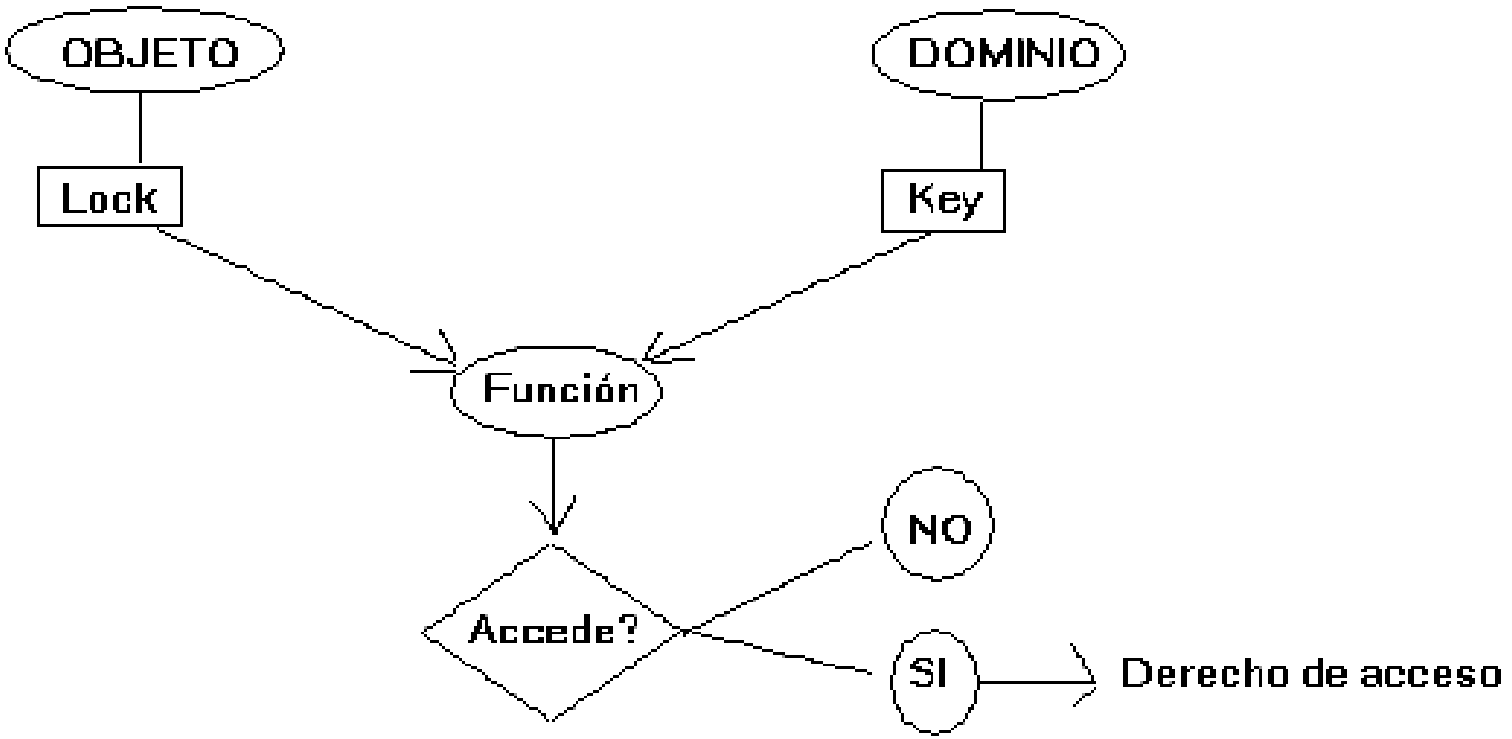
objetos / Dominios	Compiladores	Archivos
Programadores	Ejecutar	Leer
System Programmer	Leer/grabar/ejecutar	
Usuarios	Ejecutar	Leer/grabar

Listas de Capacidad

- La **capacidad de llegar al objeto** la posee el **dominio** ya que debe conocer la ubicación del objeto para poder accederlo.
- En las **LCU** se debe recordar que **cada usuario debe poseer un puntero al archivo** que no le pertenece **para poder encontrarlo**.

Mecanismo Lock-Key

- Cada Dominio provee su key y cada objeto provee su lock (usualmente son unos pocos bits).
- Ambos se combinan en una función (usualmente de tipo booleano) y de acuerdo al resultado se otorga o no el acceso.



Mecanismo Lock-Key (ejemplo)

- 3 dominios, 2 objetos y **la función es la operación AND**. Se accede si el resultado es Falso.
- Se arma solo el acceso al objeto Archivos
- Programadores <10> key
- Syst. Prog. <01> key
- Usuarios <10> key
- Archivos <01> lock

objetos / Dominios	Compiladores	Archivos
Programadores	Ejecutar	Leer
System Programmer	Leer/grabar/ ejecutar	
Usuarios	Ejecutar	Leer/grabar

- Aquí **solo se construye el permiso de acceso sin clasificarlo**, es decir no se sabe aún si accede en modalidad de lectura o grabación -> para esto se deben usar más bits.

Estructuras Dinámicas

- Hasta aquí las estructuras vistas son estáticas. **Si queremos que la estructura pueda cambiar** en tiempo de ejecución **deben agregarse nuevos tipos de derechos de acceso**.
- Veremos algunos, a saber:
  - **COPY**
  - **OWNER**
  - **SWITCH**
  - **CONTROL**

COPY

- Permite que **un dominio otorgue el permiso** que posee sobre un objeto **a otro dominio**.
- Se denota con un *.
- Ejemplo: Permitir que los Programadores puedan grabar archivos
- OJO! No se puede otorgar lo que no se posee!!

objetos / Dominios	Compiladores	Archivos
Programadores	Ejecutar	Leer
System Programmer	Leer/grabar/ejecutar	
Usuarios	Ejecutar	Leer/grabar *

OWNER

- **- Otorga** la **propiedad de un Objeto a un dominio en particular**.
- - El dominio **puede otorgar permisos** sobre ese objeto a otros dominios **y además eliminarlos**.
- Ejemplo: Quitar la autorización dada a los programadores para grabar archivos.

objetos / Dominios	Compiladores	Archivos
Programadores	Ejecutar	Leer
System Programmer	Leer/grabar/ejecutar	
Usuarios	Ejecutar	Leer/grabar OWNER

CONTROL

- Permite que **un dominio controle a otro dominio**. Se necesita que los dominios ahora sean pasibles de recibir acciones.
- **Asumiremos** que el CONTROL **solo permite eliminar derechos**.
- Ejemplo: El System Programmer controla la operatoria total de los otros dominios

objetos / Dominios	Compila-dores	Archivos	Programa-dores	usuarios	System programmer
Programadores	Ejecutar	Leer			
System Programmer	Leer/grabar/ ejecutar		control	control	
Usuarios	Ejecutar	Leer / grabar			

SWITCH

- Permite que **un dominio pase a actuar como otro dominio perdiendo sus derechos** y **asumiendo los derechos del dominio** al que se cambió.

- Ejemplo: Se desea que los System Programmer puedan probar nuevas aplicaciones desarrolladas para los usuarios para ver si efectivamente funcionan bien.

objetos / Dominios	Compila-dores	Archivos	Programa-dores	usuarios	System programmer
Programadores	Ejecutar	Leer			
System Programmer	Leer/grabar/ ejecutar			switch	
Usuarios	Ejecutar	Leer / grabar			

Sistemas confiables

- La cuestión del por qué no se están construyendo sistemas seguros se resume en dos razones fundamentales.

1. En **primer** lugar, los sistemas actuales no son seguros pero **los usuarios no desean descartarlos**.
2. La única forma conocida de construir un sistema seguro es **mantenerlo simple**. Las **características son enemigas de la seguridad**. Los usuarios quieren más características. Esto significa más complejidad, más código, más errores y más errores de seguridad.

Sistemas confiables

- Cuando el **servicio Web** consistía en páginas de HTML pasivas, no imponía un problema grave de seguridad. Ahora que muchas páginas Web contienen programas (applets) que el usuario tiene que ejecutar para ver el contenido, aparece una fuga de seguridad tras otra. Tan pronto como se corrige una, otra toma su lugar.

Sistemas confiables

- Para construir un sistema seguro, hay que tener un modelo de seguridad en el núcleo del **sistema operativo que sea lo bastante simple** como para que los diseñadores puedan comprenderlo de verdad, y que resistan toda la presión de desviarse de este modelo para agregar nuevas características.

Base de cómputo confiable

- En el mundo de la seguridad, todos hablan con frecuencia sobre los **sistemas confiables en vez de** sistemas seguros. Éstos son sistemas que han declarado formalmente requerimientos de seguridad y cumplen con ellos.
- En el centro de todo sistema confiable hay una **TCB (*Trusted Computing Base,*** Base de cómputo confiable) mínima que consiste en el hardware y software necesarios para cumplir con todas las reglas de seguridad.
- Si la **base de cómputo confiable funciona** según las especificaciones, **no se puede comprometer la seguridad** del sistema, sin importar qué otra cosa esté mal.

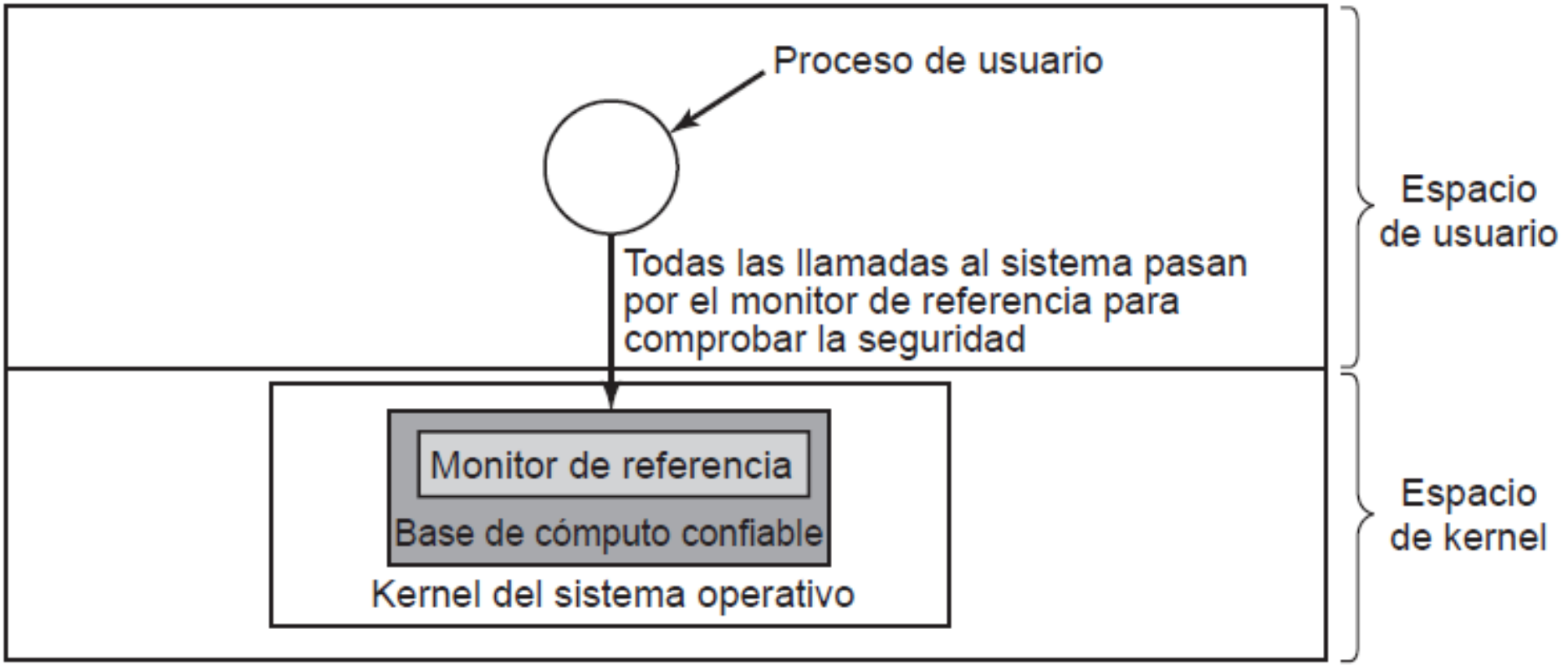
Base de cómputo confiable

- La **TCB consiste en la mayoría del hardware** (excepto los dispositivos de E/S que no afectan a la seguridad), una porción del kernel del sistema operativo, y la mayoría o todos los **programas de usuario que tienen poder de superusuario** (por ejemplo, los programas raíz SETUID en UNIX).
- **Las funciones del sistema operativo que deben formar parte de la TCB son**: creación de procesos, cambio de procesos, administración del mapa de memoria y parte de la administración de archivos y de la E/S.
- En un diseño seguro es frecuente que la **TCB esté muy separada del resto del sistema operativo** para poder minimizar su tamaño y verificar que sea correcto.

Base de cómputo confiable

- **El monitor de referencia** es una parte importante de la TCB, como se muestra en la figura siguiente.
- **El monitor de referencia acepta** todas las **llamadas al sistema** relacionadas con la **seguridad** (como abrir archivos), y decide si se deben procesar o no.
- **El monitor de referencia** permite que **todas** las decisiones de seguridad se coloquen en **un solo lugar**, sin posibilidad de pasarlo por alto. La mayoría de los sistemas operativos están diseñados de esta forma, lo cual es una de las **razones por las que son tan inseguros**.

Base de cómputo confiable



Base de Cómputo confiable

- Uno de los objetivos actuales referentes a la búsqueda de seguridad consisten en **reducir la base de cómputo confiable**, de millones de líneas de código a unas cuantas decenas de miles.
- Con MINIX 3, sólo se ejecutan aproximadamente **4000 líneas de código en el kernel**. Todo lo demás se ejecuta como un conjunto de procesos de usuario.
- Algunos de estos **procesos**, como **el sistema de archivos y el administrador de procesos**, **forman parte de la base de cómputo confiable** debido a que pueden comprometer con facilidad la seguridad del sistema.
- Otras partes como el driver de impresora **no forman** parte de la base de cómputo confiable, por lo tanto **pueden ser afectados** por agentes externos, pero **no comprometen al sistema**.

Modelos formales de los sistemas seguros

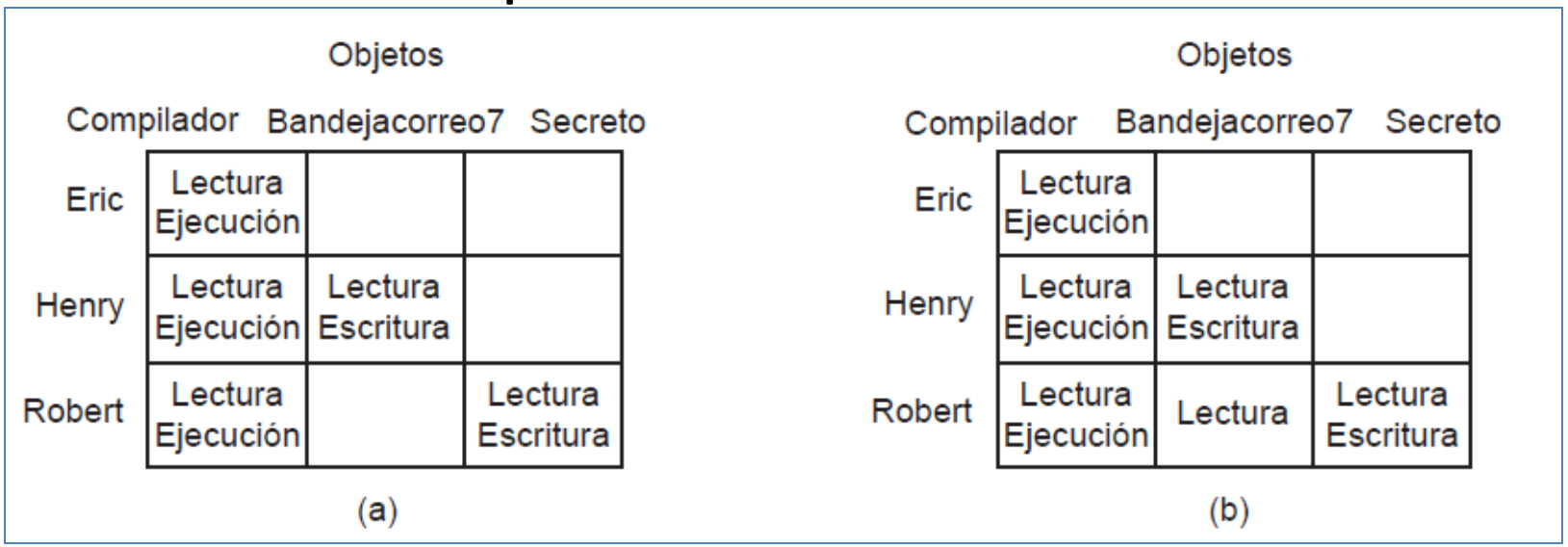
- Las **matrices de protección no son estáticas**. Cambian con frecuencia a medida que **se crean objetos**, los objetos antiguos se destruyen y **los propietarios deciden aumentar o restringir el conjunto de usuarios** para sus objetos.
- Hace unas décadas, se identificaron **seis operaciones primitivas importantes** en la matriz de protección:
  - create object,
  - delete object,
  - create domain,
  - delete domain,
  - insert right,
  - remove right

Modelos formales de los sistemas seguros

- Las últimas dos primitivas insertan y eliminan permisos de elementos específicos de la matriz
- Estas seis primitivas se pueden combinar en **comandos de protección.**
- **Los programas de usuario** pueden ejecutar estos comandos de protección para cambiar la matriz pero no pueden ejecutar las primitivas en forma directa.
- Por ejemplo, el sistema podría tener **un comando para crear un nuevo archivo** (verificar si existe y en caso contrario crear y dar permisos).
- También podría haber un **comando para permitir al usuario otorgar permisos** para leer el archivo a todos los usuarios en el sistema (permiso “lectura” en todos los dominios)

Modelos formales de los sistemas seguros

- En cualquier instante, la matriz determina qué es lo que puede hacer un proceso en un dominio.
- La matriz es lo que el sistema hace cumplir; **la autorización está relacionada con la directiva de administración**.
- Como ejemplo de esta distinción, en la figura los dominios corresponden a los usuarios.



Objetos

	Compilador	Bandejacorreo7	Secreto
Eric	Lectura Ejecución		
Henry	Lectura Ejecución	Lectura Escritura	
Robert	Lectura Ejecución		Lectura Escritura

(a)

Objetos

	Compilador	Bandejacorreo7	Secreto
Eric	Lectura Ejecución		
Henry	Lectura Ejecución	Lectura Escritura	
Robert	Lectura Ejecución	Lectura	Lectura Escritura

(b)

Modelos formales de los sistemas seguros

- En la figura anterior (b) *Robert pudo emitir comandos para cambiar* la matriz. Ahora el tiene acceso a *bandejacorreo7, algo* que no está autorizado para tener.
- Si trata de leer este archivo, **el sistema operativo** llevará a cabo su petición debido a que **no sabe que el estado** de la figura (b) **es no autorizado**.
- Ahora se debe tener claro que el conjunto de todas las matrices posibles se puede particionar en dos conjuntos desunidos:
  - el conjunto de todos los estados autorizados
  - el conjunto de todos los estados no autorizados.

Modelos formales de los sistemas seguros

- Estamos preguntándonos si el mecanismo disponible (los comandos de protección) **es adecuado para implementar una directiva de protección**. Dada esta directiva, cierto estado inicial de la matriz y el conjunto de comandos para modificarla, lo conveniente sería una forma de probar que el sistema es seguro.
- Dicha prueba resulta ser muy difícil de adquirir: muchos sistemas de propósito general no son seguros en teoría.
- Sin embargo, **para un sistema específico** es posible demostrar si el sistema puede **pasar** en algún momento dado de **un estado autorizado a un estado no autorizado**.
-

Seguridad multinivel

- La mayoría de los sistemas operativos los usuarios individuales determinan quién puede leer y escribir sus archivos.
- A esta directiva se le conoce como **control de acceso discrecional. Hay entornos donde se** requiere una seguridad más estricta, como en la computación militar, etc.
- En estos entornos, la organización ha establecido reglas sobre qué es lo que puede ver cada quién, y los usuarios comunes no pueden editar las reglas.
- Estos entornos necesitan **controles de acceso obligatorio para asegurar que el sistema implemente las directivas de seguridad** establecidas, además de los controles de acceso discrecional estándar.

Seguridad multinivel

- El modelo de seguridad multinivel más utilizado es el **modelo Bell-La Padula:**
  - En el mundo **militar**, los documentos (objetos) pueden tener un nivel de seguridad, como **no clasificado, confidencial, secreto y de alta confidencialidad**.
  - A las personas también se les asignan estos niveles, dependiendo de los documentos que puedan ver.
  - **Un general** podría tener permiso para ver todos los documentos, mientras que a un **teniente** se le podría restringir a todos los documentos clasificados como confidenciales o con un nivel menor de seguridad.
  - Un proceso que se ejecute a beneficio de un usuario adquiere el nivel de seguridad del usuario. Como hay varios niveles de seguridad, a este esquema se le conoce como **sistema de seguridad multinivel.**

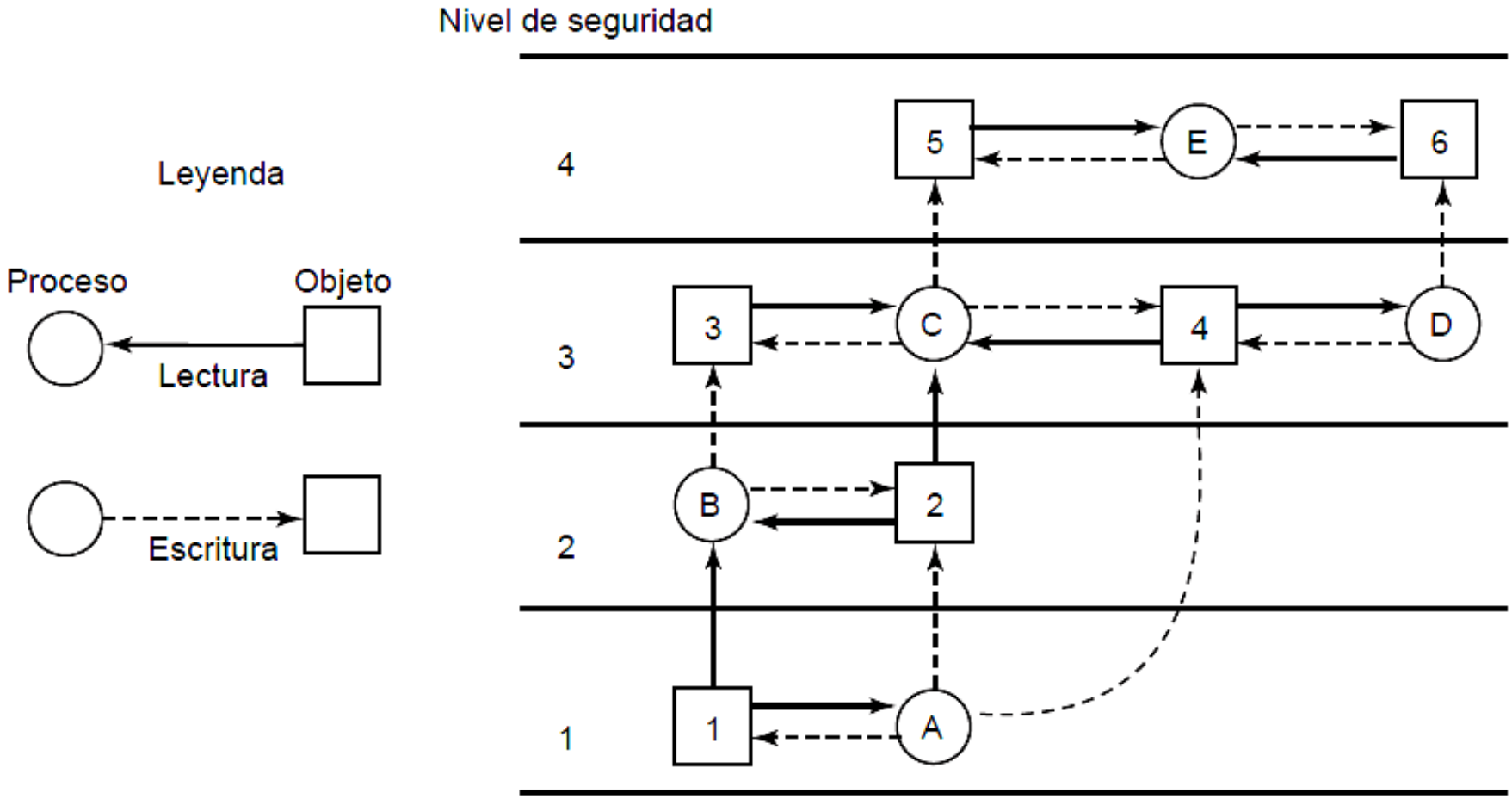
Seguridad multinivel

- Reglas sobre la forma en que debe fluir la información:
- 1. **La propiedad de seguridad simple, donde un proceso que se ejecuta en el nivel de seguridad *k*** sólo puede leer objetos que estén en su nivel o en uno menor.
- 2. **La propiedad *, donde un proceso que se ejecuta en el nivel de seguridad *k sólo puede escribir objetos*** en su nivel o en uno mayor.

- En este modelo, **los procesos leen y escriben objetos**, pero **no se comunican entre sí** de una manera directa. El modelo Bell-La Padula se ilustra en modo gráfico en la figura siguiente.

* No sabían que nombre ponerle y povisoriamente le pusieron "la propiedad … nunca mas se cambió

Seguridad Multinivel



Seguridad Multinivel

- El modelo Bell-La Padula hace referencia a la estructura organizacional, pero en última instancia el sistema operativo es quien tiene que implementarlo.
- Una manera de hacerlo es asignar **a cada usuario un nivel de seguridad**, que se debe almacenar junto con otros datos específicos del usuario, como el UID y el GID.
- Al momento de iniciar sesión, el shell del usuario adquiere su nivel de seguridad, y todos sus hijos lo heredan.
- Si un proceso que se ejecuta en el **nivel de seguridad *k*** ***trata de abrir un archivo u otro objeto cuyo nivel de seguridad sea mayor que k****, el sistema operativo* debe **rechazar el intento** de apertura.
- Todos **los intentos** similares **de abrir cualquier objeto de un nivel de seguridad menor que *k para escribir en él deben fallar***.

Seguridad Multinivel

- **El modelo Biba**
- **Bell-La Padula** se ideó para **guardar secretos**, **no** para garantizar la **integridad de los datos**. Para esto último necesitamos precisamente las propiedades inversas (Biba, 1977):
- 1. **El principio de integridad simple Un proceso que se ejecuta en el nivel de seguridad *k sólo*** puede escribir objetos en su nivel o en uno inferior (no hay escrituras hacia arriba).
- 2. **La propiedad * de integridad Un proceso que se ejecute en el nivel de seguridad *k puede*** leer sólo los objetos en su nivel o en uno superior (no hay lecturas hacia abajo).

Seguridad Multinivel

- Desde luego que ciertas organizaciones desean tanto las propiedades de Bell-La Padula como las de Biba, pero como están en conflicto directo, es difícil de obtener las propiedades de ambos modelos a la vez.

Canales encubiertos

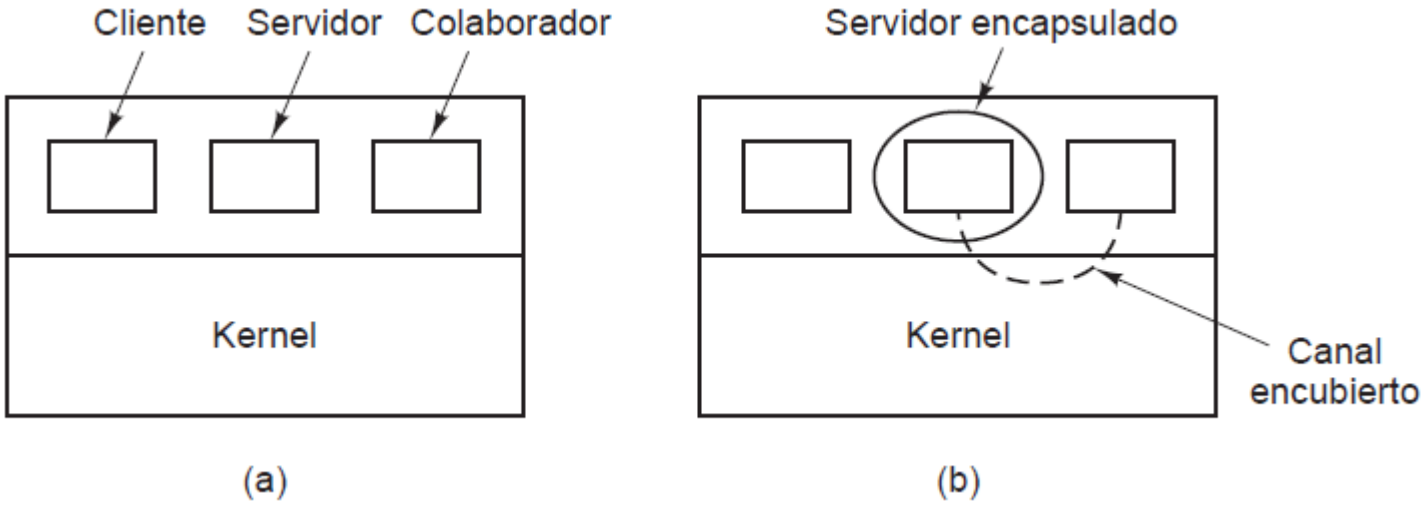
- Todas **estas ideas sobre modelos formales** y sistemas definitivamente seguros suenan bien, pero ¿**en realidad funcionan? En una palabra: no**.
- **Aun en un sistema** que tenga un modelo de seguridad subyacente, **que haya demostrado ser seguro** y que esté implementado en forma correcta, **puede haber fugas de seguridad**.
- A continuación analizaremos **cómo puede haber fuga de información** incluso cuando se haya demostrado con rigor que dicha fuga es matemáticamente imposible.
- El **modelo de Lampson** se formuló originalmente en términos de un solo sistema de tiempo compartido, pero se pueden adaptar las mismas ideas a las LANs y otros entornos multiusuario.

Modelo de Lampson

- En su forma más pura, **implica tres procesos** en cierta máquina protegida. El primer proceso (el cliente) desea que el segundo (el servidor) realice cierto trabajo.
- El **cliente y el servidor no confían completamente uno en el otro**. Por ejemplo, el trabajo del servidor es ayudar a que los clientes llenen sus formularios fiscales. A los clientes les preocupa que el servidor registre en secreto sus datos financieros; por ejemplo, para mantener una lista secreta de cuánto gana cada quién, y después vender la lista. Al servidor le preocupa que los clientes traten de robar el valioso programa fiscal.
- El **tercer proceso es el colaborador**, que está conspirando con el servidor para robar los datos confidenciales de los clientes.

Modelo de Lampson

- El objeto de este ejercicio es diseñar un sistema en el que sea imposible que **el proceso servidor filtre al proceso colaborador la información** que ha recibido de manera legítima del proceso cliente.
- A este problema, Lampson lo llamó **problema del confinamiento.**



Modelo de Lampson

- Desde el punto de vista del diseñador, el objetivo es **encapsular o confinar el servidor** de tal forma que no pueda pasar información al colaborador.
- Mediante el **uso de un esquema de matriz de protección**, podemos garantizar que el servidor **no se podrá comunicar** con el colaborador **mediante la escritura de un archivo al** que el colaborador tiene acceso de lectura.
- También podremos asegurar que el servidor no se podrá comunicar con el colaborador **mediante el uso del mecanismo de comunicación entre procesos** del sistema.

Modelo de Lampson

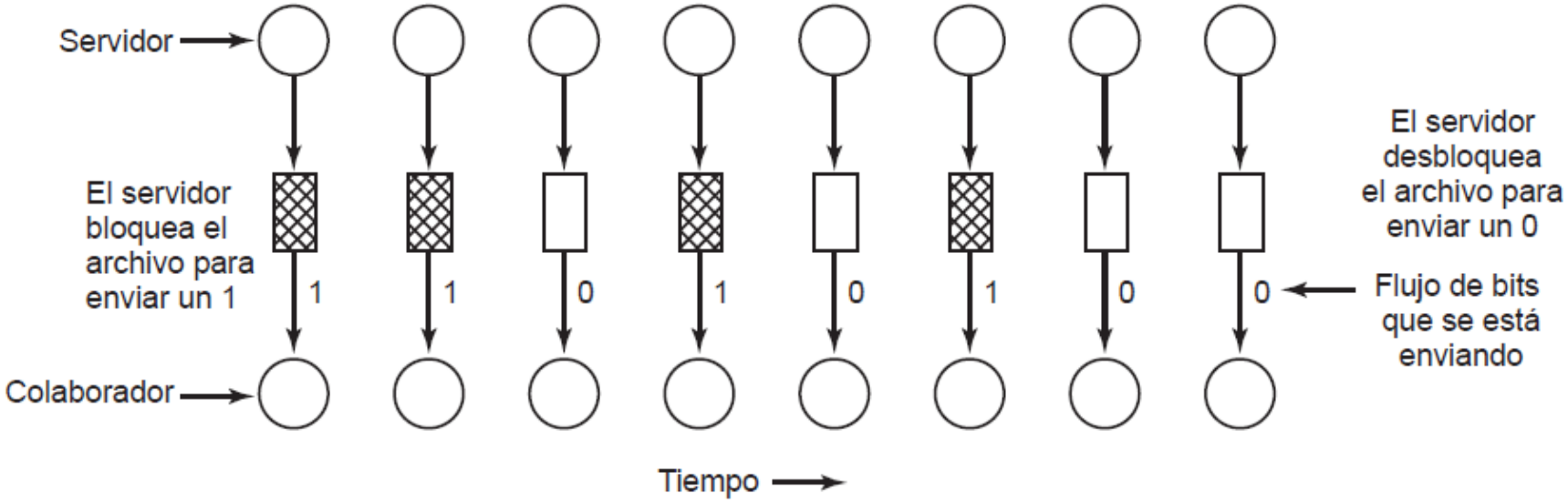
- Por desgracia pueden existir canales de comunicación más sutiles. Por ejemplo, el servidor puede tratar de comunicar un flujo de bits binario de la siguiente manera:
  - Para enviar un bit 1, realiza todos los cálculos que pueda durante un intervalo fijo.
  - Para enviar un bit 0, permanece inactivo durante la misma cantidad de tiempo.
- El colaborador puede tratar de detectar el flujo de bits al monitorear cuidadosamente su tiempo de respuesta.
- Este canal de comunicación se conoce como **canal encubierto, y se ilustra en la** figura siguiente.
- Es bastante obvio que **ningún modelo** de protección basado en una matriz de objetos y dominios **podrá evitar este tipo de fuga.**

Modelo de Lampson

- La **modulación** del uso de la CPU no es el único canal encubierto.
- La velocidad de paginación también se puede modular (muchos fallos de página para un 1, ningún fallo de página para un 0).
- Si el sistema proporciona una forma de **bloquear los archivos**, entonces el servidor puede bloquear cierto archivo para indicar un 1, y desbloquearlo para indicar un 0. Este canal encubierto se muestra en la figura siguiente, en donde el archivo se bloquea o se desbloquea durante cierto intervalo fijo, conocido para el servidor y para el colaborador.

Modelo de Lampson

En este ejemplo se está transmitiendo el flujo de bits secreto 11010100, mediante el uso de bloqueo y desbloqueo



Modelo de Lampson

- Bloquear y desbloquear un archivo *S preparado con anticipación* requiere de una sincronización bastante precisa.
- La confiabilidad se pueden incrementar si se usa un protocolo de reconocimiento. Este protocolo utiliza otros **dos archivos (*F1 y F2)*** *bloqueados por el* servidor y el colaborador, respectivamente, para mantener los dos procesos sincronizados.
- Después de que el servidor bloquea o desbloquea *S, cambia el estado de bloqueo de F1 para indicar que se* ha enviado un bit.
- Después de que el colaborador lee el bit, cambia el estado de bloqueo de *F2 para* indicar al servidor que está listo para recibir otro bit.
- *Este protocolo es muy confiable ya que* no requiere sincronización.

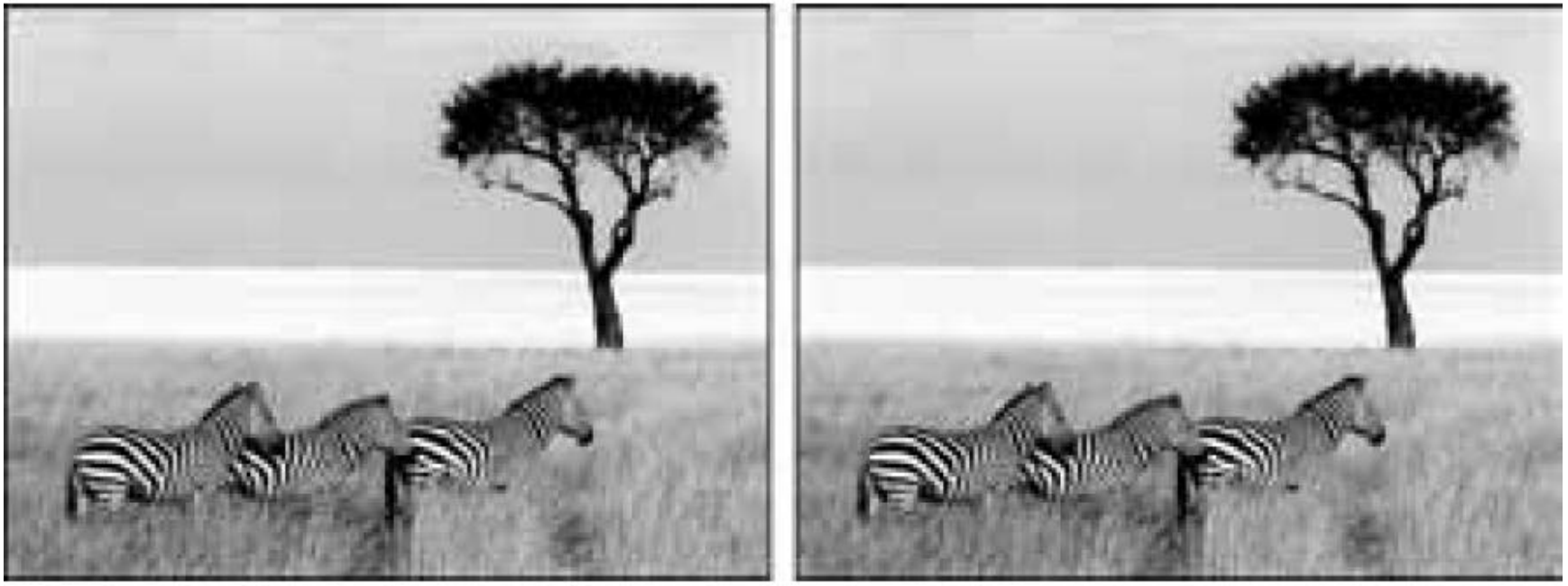
Modelo de Lampson

- Lampson también mencionó una **forma de filtrar información al propietario (humano)** del proceso servidor.
- Se supone que el proceso **servidor tiene el derecho de indicar** a su propietario **cuanto trabajo realizó** por el cliente, para poder cobrarle.
- Si la factura por el cálculo realizado es de $100 por ejemplo, y el ingreso del cliente es de $53.000, el servidor podría reportar la factura como de **$100,53** a su propietario.
- Si es muy difícil tan sólo encontrar todos los canales encubiertos, es mucho más difícil bloquearlos.

Esteganografía

- Considere la figura (a) siguiente como un buen ejemplo. Esta fotografía que tomó el autor en Kenya
- contiene tres cebras que contemplan un árbol.
- La figura (b) parece tener las mismas tres cebras y el mismo árbol acacia, pero se le agregó una atracción adicional. Contiene el texto íntegro de cinco obras de Shakespeare: *Hamlet, El Rey Lear, Macbeth, El mercader de Venecia y Julio César.*
- En conjunto, estas obras contienen más de 700 KB de texto.
- Se utiliza el bit menos significativo de cada canal RGB en cada pixel. Entonces si tenemos una imagen de 1024x768 pixeles. El total de bytes secretos será de 294912. ver www.cs.vu.nl/~ast

Esteganografía



(a) (b)

bibliografía

- Cap 9, Tanembaum “Sistemas Operativos modernos “