

Departamento de Lenguajes y Ciencias de la  
Computación Universidad de Málaga

# Sistemas Operativos

Curso 2001/2002

## TEMA 2: Seguridad y Protección



E.T.S.I. Informática

Profesor:  
Francisco López Valverde

<http://polaris.lcc.uma.es/~valverde>

## Índice de contenidos

### Introducción Seguridad

- Introducción
- Amenazas a la seguridad
- Ataques genéricos
- Principios de diseño para la seguridad
- Control de acceso de usuario
- Amenazas de origen software

### Protección

- Dominios de protección
- Listas de control
- Capacidades

### AP1: Protección en WinNT/2000

### AP2: Criptografía

- Criptografía básica
- Firma de documentos
- Certificados digitales
- PGP
- Kerberos

### Referencias



- La **seguridad** comprende toda la problemática relacionada con hacer que los ficheros no puedan ser leídos o escritos por ninguna persona no autorizada, e incluye aspectos técnicos, administrativos, legales y políticos.
- La **protección** se podría definir como el conjunto de mecanismos específicos del sistema operativo que tienen como finalidad el proporcionar seguridad.



- La seguridad se aplica tanto a **pérdida de datos** como a **intrusos** que intentan hacer lo que no deben.
- Algunas causas comunes de pérdida de información:
  - **Desastres** tales como incendios, terremotos, riadas, etc.
  - **Errores hardware o software.**
  - **Errores humanos.**
- La mayoría de estos problemas se pueden solucionar realizando copias de seguridad y almacenando las cintas en lugares físicamente lejanos a los datos originales.





## Seguridad

- Existen dos tipos de intrusos: intrusos pasivos que únicamente quieren leer información no autorizada e intrusos activos que quieren realizar modificaciones no permitidas. Los intrusos se pueden clasificar en las siguientes categorías:
  - Usuarios no técnicos motivados por la curiosidad.
  - Personal cualificado (estudiantes, programadores, técnicos, ...), que consideran un reto personal violar la seguridad de un sistema.
  - Intento de obtener beneficios económicos.
  - Espionaje comercial o militar.
- El esfuerzo dedicado a la seguridad depende del tipo de intrusión que se pretenda prevenir.



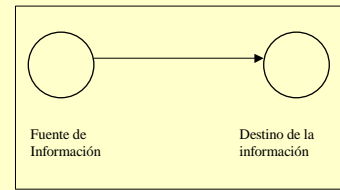
## Seguridad

- Podemos definir tres requisitos necesarios para la seguridad dentro de un sistema informático:
  - Privacidad: la información debe ser accesible para lectura únicamente por las partes autorizadas. Este tipo de acceso incluye la impresión, tanto por pantalla como impresora, e incluso la revelación de la existencia de un determinado objeto o suceso.
  - Integridad: los elementos relacionados con la seguridad únicamente pueden ser modificados por las partes autorizadas. Modificación incluye escritura, creación, cambio de estado y borrado.
  - Disponibilidad: los elementos relacionados con la seguridad deben ser estar disponibles únicamente a las partes autorizadas.



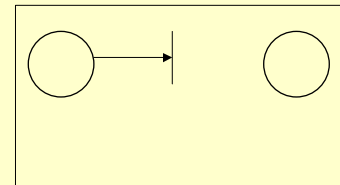
## Tipos de amenazas

En general, hay un flujo de información de una fuente a un destino:

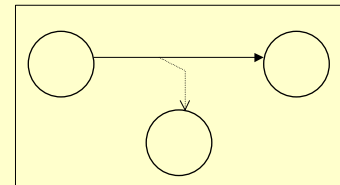


Podemos señalar cuatro categorías de amenazas:

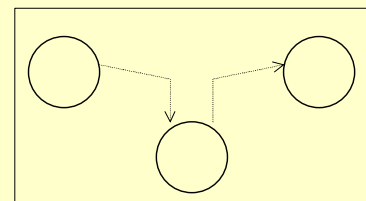
① **Interrupción**: Un elemento del sistema es destruido o se hace inservible. Es una amenaza a la **disponibilidad**.



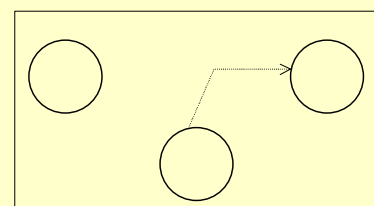
② **Intercepción**: Una parte no autorizada obtiene acceso a un elemento relacionado con la seguridad. Es una amenaza a la **privacidad**.



③ **Modificación**: Parte no autorizada no sólo obtiene acceso, sino que puede modificar un elemento relacionado con la seguridad. Es una amenaza a la **integridad**.



④ **Fabricación**: Una parte no autorizada inserta nuevos elementos en el sistema. Es una amenaza a la **integridad**.



## Ataques genéricos a la seguridad

- Solicitar páginas de memoria o bloques de disco y leer su contenido. Si el sistema no borra la información que contienen antes de asignarlos, pueden contener información relevante.
- Ejecutar llamadas al sistema ilegales o llamadas legales con parámetros ilegales.
- Iniciar el programa de conexión al sistema y pulsar teclas de borrado, escape, etc.
- Intentar modificar las estructuras del sistema que estén en el espacio de usuario.
- Engañar al usuario ejecutando un programa que tenga la apariencia del indicador de conexión.
- Leer los manuales y cuando se advierta que no se realice una acción determinada, intentar probar tantas variaciones de la misma como sea posible.
- Si todo esto falla, intentar sobornar al personal que pueda tener información interesante.



## Principios de diseño para la seguridad

Algunos principios de diseño relacionados con la seguridad :

- Diseño abierto: La seguridad de un sistema no debe de depender de la suposición de que su diseño es secreto. Asumir que un intruso no lo conoce es engañar a los diseñadores.
- Negación en caso de duda: Por defecto, se deben de negar los accesos no autorizados.
- Verificación completa: Toda operación debe de ser contrastada con la información de control de acceso.
- Principio del menor privilegio: Todos los procesos deberían de tener los mínimos privilegios necesarios para realizar sus tareas.
- Economía: El mecanismo de protección debe de ser simple, uniforme e integrado hasta las capas más bajas del sistema.
- Aceptabilidad: El sistema elegido debe de ser psicológicamente aceptable por los usuarios, ya que en caso contrario éstos no lo usarán.



## Control de acceso de usuarios

Muchos esquemas de protección se basan en que el sistema conoce la identidad de todos los usuarios.

El problema de identificar a los usuarios cuando éstos se conectan se denomina **autenticación de usuarios**.

Hoy en día se conocen tres únicos métodos para identificar personas:

- ⇒ Por las características físicas: **biométricos**.
- ⇒ Por un secreto compartido: contraseñas (**Passwords**).
- ⇒ Por la posesión de un objeto (software o hardware):  
**Tokens** o **certificados digitales**



## Control de acceso de usuarios

Los más utilizados son los de control por contraseña

Los sistemas más nuevos son los biométricos. Problemas de precio de captación, costumbre de uso, etc.

Los sistemas de posesión de un objeto físico son los más antiguos en control de accesos físicos (llaves, sellos, salvoconductos). En el mundo digital se han usado muy poco (certificados digitales)

Todos los sistemas se pueden combinar para aumentar la seguridad





# Control de acceso de usuarios

## Control de acceso por contraseñas

- Las contraseñas son un punto débil de los sistemas de seguridad, pero son el sistema más sencillo, popular y probado.
- Normalmente, un sistema debe mantener un archivo que asocia contraseñas con usuarios. Este archivo se puede proteger de dos formas:
  - **Cifrado unidireccional.**
  - **Control de acceso.**
- El objetivo de un **intruso** es obtener acceso a un sistema o aumentar el conjunto de privilegios que posee en un sistema.



# Control de acceso de usuarios

## Técnicas de obtención de contraseñas

- ✓ Probar las contraseñas empleadas en las cuentas estándares que se suministran junto al computador.
- ✓ Probar exhaustivamente todas las contraseñas cortas.
- ✓ Probar palabras del diccionario o de una lista de contraseñas probables.
- ✓ Reunir información sobre los usuarios y utilizarlo como contraseña.
- ✓ Probar los números de teléfono de los usuarios, sus números de DNI y números de habitación.
- ✓ Emplear un caballo de Troya para saltarse las restricciones de acceso.
- ✓ Intervenir la línea situada entre un usuario remoto y el sistema anfitrión.

# Control de acceso de usuarios

## Estrategias de elección de contraseñas

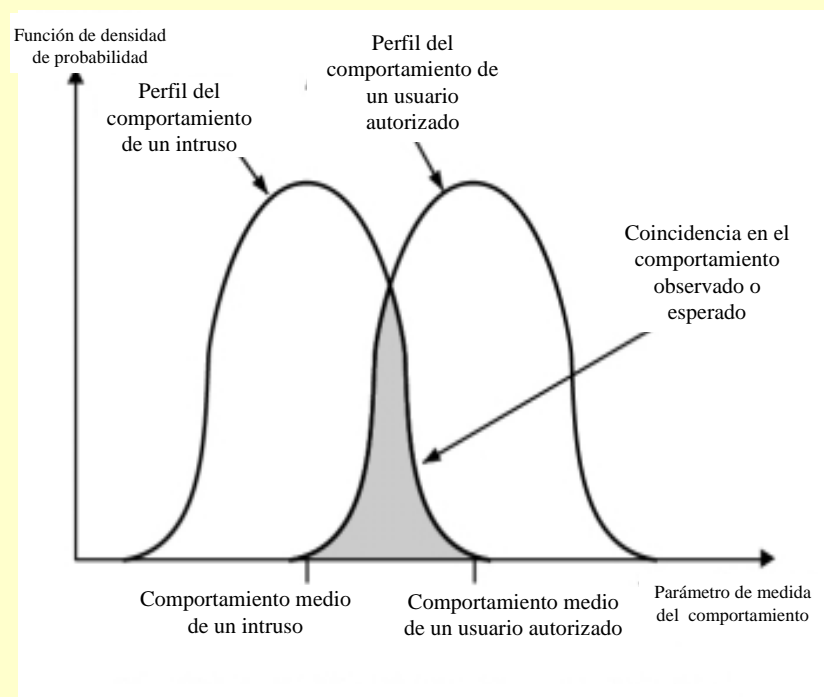
- ✓ Contraseñas generadas por computador:
  - Los usuarios pueden encontrar dificultades a la hora de recordarlas.
  - Necesidad de escribirlas.
  - Tienen antecedentes de escasa aceptación.
- ✓ Inspección reactiva de contraseñas:
  - El sistema ejecuta periódicamente su propio averiguador de contraseñas para encontrar contraseñas adivinables.
  - El sistema cancela todas las contraseñas que se adivinen y se lo notifica al usuario.
  - Se consumen recursos para realizarlo.
  - Un pirata informático puede utilizarla en su propia CPU con una copia del archivo de las contraseñas.
- ✓ Inspección proactiva de contraseñas:
  - En el momento de la selección, el sistema comprueba si la contraseña es permisible.
  - Con las directrices del sistema, los usuarios pueden elegir contraseñas recordables que son difíciles de adivinar.



# Control de acceso de usuarios

## Detección de intrusiones

Supone que el comportamiento del intruso se diferencia del comportamiento del usuario legítimo.





# Control de acceso de usuarios

## Detección de intrusiones

### Técnicas:

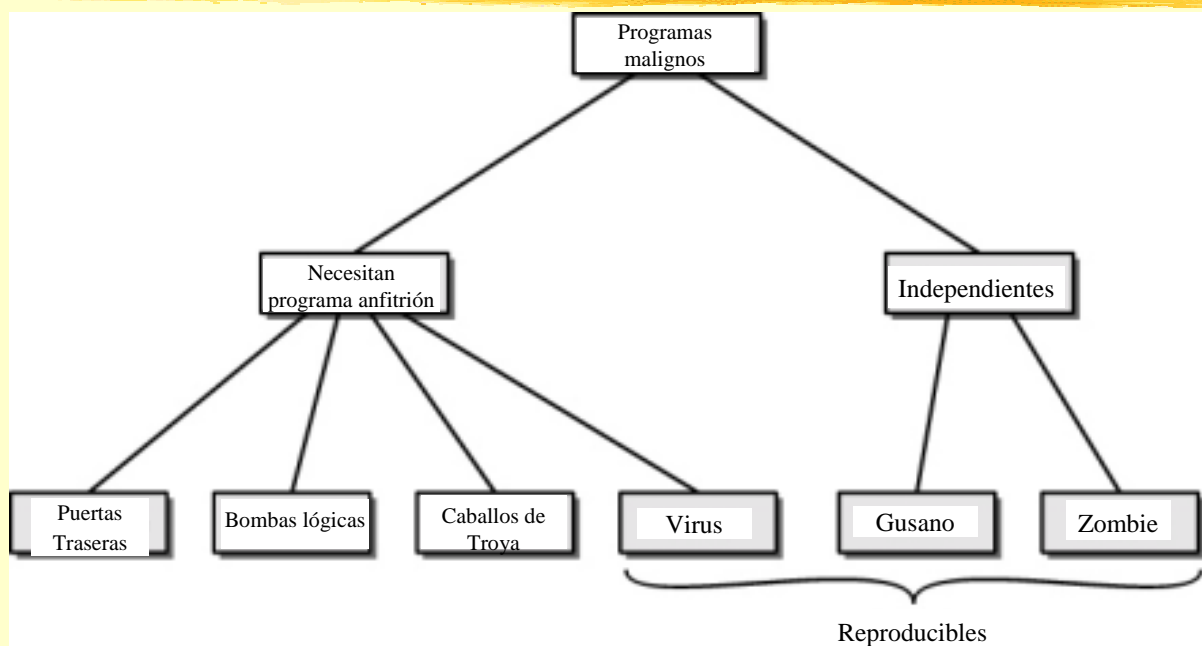
- ✓ Detección de anomalías estadísticas:
  - Recolección de datos del comportamiento de los usuarios legítimos durante un periodo de tiempo.
  - Las pruebas estadísticas determinan si el comportamiento no es legítimo.
- ✓ Detección basada en reglas:
  - Se construyen reglas para detectar desviaciones con respecto a pautas de uso anteriores.
  - Un sistema experto persigue comportamientos sospechosos.

### Herramientas:

- ✓ Registros de auditoría:
  - Registros de auditoría nativos:
  - Registros de auditoría específicos para la detección:



# Amenazas de origen software



# Amenazas de origen software

## Puerta trasera

- Punto de entrada a un programa que permite a alguien que la conoce conseguir el acceso.
- Utilizadas por los programadores para depurar y probar los programas:
  - Evita toda la configuración y autenticación necesarias.
  - Hay un método para activar el programa en el caso de que algo vaya mal en el procedimiento de autenticación.

## Bomba lógica

- Código incrustado en un programa legítimo que “explota” cuando se cumplen ciertas condiciones:
  - Presencia o ausencia de ciertos archivos.
  - Día concreto de la semana.
  - Un usuario en particular ejecuta la aplicación.



# Amenazas de origen software

## Caballo de Troya

- Programa útil que contiene un código oculto que, cuando se invoca, lleva a cabo alguna función dañina o no deseada.
- Se pueden utilizar para efectuar funciones indirectamente que un usuario no autorizado no podría efectuar directamente.
  - El usuario puede conceder permiso de acceso a sus archivos para que puedan ser leídos por cualquier usuario.

## Virus

- Programa que puede “infectar” a otros programas, alterándolos.
  - La alteración incluye una copia del programa de virus.
  - El programa infectado puede seguir infectando a otros programas.



# Amenazas de origen software

## Gusano

- Emplean conexiones de la red para extenderse de un sistema a otro.
- Servicio de correo electrónico:
  - El gusano divulga una copia de sí mismo a otros sistemas.
- Capacidad de ejecución remota:
  - El gusano ejecuta una copia de sí mismo en otro sistema.
- Capacidad de conexión remota:
  - El gusano se conecta a un sistema remoto como un usuario y después emplea órdenes para copiarse a sí mismo de un sistema a otro.

## Zombie

- Programa que secretamente toma posesión de otro computador conectado a Internet.
- Utiliza ese computador para lanzar ataques que hacen difícil detectar a su creador.



# Amenazas de origen software

## Etapas en la vida de un virus

- 1. Fase latente:**
  - El virus está inactivo.
- 2. Fase de propagación:**
  - El virus hace copias idénticas a él en otros programas o en ciertas áreas del sistema del disco.
- 3. Fase de activación:**
  - El virus se activa para realizar su trabajo. Puede producirse por diversas causas: tiempo, veces que se ha copiado, fecha, etc.
- 4. Fase de ejecución:**
  - Se lleva a cabo el trabajo, que puede ser dañina o no.





## Protección

Aparición de la multiprogramación



Posibilidad de compartir recursos entre varios usuarios.



**Necesidad de protección.**

Existen varios **motivos** por lo que la protección es necesaria.

- Prevenir intentos de violación de restricciones de acceso por parte de un usuario.
- Asegurar que cada proceso use los recursos del sistema de forma consistente de acuerdo con las políticas establecidas para el uso de esos recursos.

Fundamental para que un sistema sea fiable.



## Protección

El papel de la protección en un sistema informático es:

proporcionar un conjunto de mecanismos que permita llevar a cabo las políticas que gobiernan el uso de los recursos.

Estas políticas pueden ser



- Fijas
- Establecidas en el diseño del SO
- Formuladas por el administrador del sistema
- Establecidas por los usuarios.

## En el ámbito de la protección

Los **mecanismos** determinan **cómo** hacer cierta cosa, mientras que las **políticas** deciden **qué** es lo que hay que hacer.

La política define qué datos han de ser protegidos y el mecanismo la forma de llevar a cabo la política.

- Esta separación favorece la flexibilidad.
- Debido a que las políticas pueden cambiar con el tiempo o con el lugar, es deseable que el sistema operativo ofrezca un conjunto general de mecanismos por medio de los cuales se puedan establecer políticas diferentes.



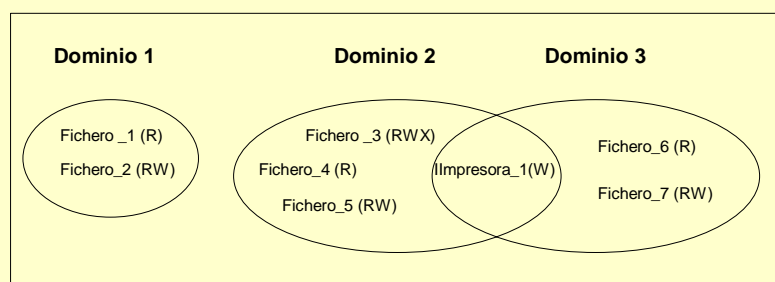
## Dominios de protección

Un computador contiene un conjunto de objetos que han de ser protegidos. Estos objetos pueden ser tanto hardware como software.

Cada objeto tiene un identificador único en el sistema y un conjunto de operaciones que le pueden ser aplicadas (dependen del objeto).

El mecanismo de protección ha de prohibir a los procesos el acceso a aquellos objetos sobre los que no tengan autorización.

Un **dominio de protección** es un conjunto de pares **objeto-derechos**, de forma que cada par especifica un objeto y algún subconjunto de operaciones que se pueden realizar sobre él. Los dominios no tienen por qué ser disjuntos.



## Dominios de protección

- En cada instante de tiempo, cada proceso se ejecuta dentro de un dominio concreto. Este dominio puede cambiar durante la vida del proceso.
- Un dominio puede ser varias cosas:

- Cada **usuario**

- Los objetos que pueden ser accedidos dependen de la identidad del usuario.
- Los cambios de dominio ocurren cuando se cambia de usuario.

- Cada **proceso**

- Los objetos que puede ser accedidos dependen de la identidad del proceso.
- Los cambios de dominio se producen cuando un proceso envía un mensaje a otro y espera la respuesta.

- Cada **procedimiento**

- Los objetos que pueden ser accedidos se corresponden con las variables locales.
- Los cambios de dominio se producen cuando se realiza una llamada a procedimiento.



## Dominios de protección


- El dominio de un proceso lo determina su **uid** y su **gid**.

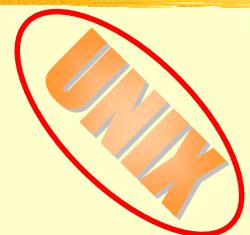
Dado un par de ellos, es posible obtener una lista de todos los objetos que puede utilizar, y el modo de acceso.

- Cada proceso tiene dos partes, la parte de usuario y la parte de núcleo.
- Cuando un proceso invoca una llamada al sistema pasa a ejecutarse en modo núcleo.



Este cambio de modo de ejecución da lugar a un **cambio de dominio**.

- Cuando un proceso realiza una llamada **exec()** sobre un fichero que tiene uno de los bits **SETUID** o **SETGID** a uno, adquiere un nuevo **uid** o **gid** efectivo.  Esto también provoca un cambio de dominio.



## Dominios de protección

- El sistema operativo debe de conocer en todo momento los objetos que pertenecen a cada dominio.
- Una forma es tener una **matriz de acceso**, en la que las filas son los dominios y las columnas los objetos.
- Cada elemento de la matriz muestra los derechos de un dominio sobre un objeto.

Objeto Dominio	File 1	File 2	File 3	File 4	File 5	File 6	Printer 1	Printer 2
D <sub>1</sub>	R	R W						
D <sub>2</sub>			R	R W X	R W		W	
D <sub>3</sub>						R W E	W	W



## Dominios de protección

- Los cambios de dominio se pueden incluir fácilmente en la matriz de protección considerando que cada dominio es a su vez un objeto sobre el que existe una operación que permite entrar en él.
- En la práctica no es viable almacenar la matriz de protección porque es grande y tiene muchos huecos. La mayoría de los dominios no tiene acceso a la mayoría de los objetos.

- Dos soluciones prácticas:

Almacenar la matriz por **filas**, pero únicamente los elementos no vacíos.



**capacidades**

Almacenar la matriz por **columnas**, pero únicamente los elementos no vacíos.



**listas de control de acceso**



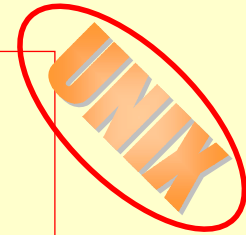
## Listas de Control de Acceso

Cuando la matriz se almacena por columnas, a cada objeto se le asocia una lista ordenada que contiene todos los dominios que pueden acceder al objeto y la forma de hacerlo.

Esta lista se denomina **lista de control de acceso** (*control access list* o **ACL**).

Esta lista está compuesta por tres grupos de tres bits,

- cada grupo los permisos para el propietario, grupo y resto de usuarios
- los tres bits indican si existe permiso de lectura, escritura y/o ejecución.



## Listas de Control de Acceso

Sin embargo, un esquema basado en listas de control de acceso puede ser más general:

```
Fichero 0: (sod4, *, RWX)
Fichero 1: (antonio, profesores, RWX)
Fichero 2: (sod2,*,RW-), (*, practicas, R--),
           (root,system, RWX)
Fichero 3: (*, practicas, R--)
```

El propietario de un objeto puede modificar la lista de control de acceso del mismo.

Los cambios pueden no afectar a los usuarios que ya estén usando el objeto.





Almacenar la matriz de protección por filas.



cada proceso tiene una lista de los **objetos** a los que puede acceder así como una indicación de las **operaciones** que están permitidas sobre cada uno de ellos.

Esta lista se denomina **lista de capacidades** (listas\_C o *C\_lists*)

Cada elemento de la lista se llama **capacidad**.

Cada objeto está representado por una capacidad.

Cuando un proceso quiere ejecutar una operación sobre un objeto, ha de especificar la capacidad del objeto como parámetro.

La simple posesión de la capacidad permite que se conceda el acceso **(siempre que éste esté permitido sobre el objeto)**.



Cada capacidad tiene un campo que indica el tipo del objeto, un campo de derechos de acceso, y un campo que identifica al objeto.

	Tipo	Permisos	Objeto
0	Fichero	R--	Puntero al fichero_3
1	Fichero	RWX	Puntero al fichero_4
2	Fichero	RW-	Puntero al fichero_5
3	Impresora	-W-	Puntero a la impresora_1

Las listas de capacidades se asocian a un dominio, pero los procesos que se ejecutan en ese dominio no deben de acceder a la lista de forma directa.

Cada capacidad es un objeto protegido que debe ser gestionado por el sistema operativo.

Generalmente, suelen ser referenciadas por su posición dentro de la lista de capacidades.



## Capacidades

Las listas de capacidades son también objetos, por lo que pueden ser apuntadas desde otras listas, permitiendo el compartir subdominios.

Para proporcionar protección, las capacidades se distinguen de otros tipos de objetos. Esta distinción se puede hacer de varias formas:

- Cada objeto tiene asociado **una etiqueta** que indica si se trata de un objeto o de una capacidad. Las etiquetas no deben ser accesibles directamente por los programas de aplicación.
- **Dividir el espacio de direcciones** de un programa en dos zonas. La primera es accesible al programa y contiene las instrucciones y los datos. La segunda contiene la lista de capacidades y es accesible únicamente por el SO.
- Mantener las capacidades en el espacio de usuario, pero **encriptadas** con una clave secreta desconocida para el usuario.



## Capacidades

Además de los permisos relativos a cada objeto, las capacidades tienen **permisos genéricos** que son aplicables a todos los objetos. Algunos son:


- **Copiar capacidad:** crear una nueva capacidad para el mismo objeto.
- **Copiar objeto:** Crear un objeto duplicado con una nueva capacidad.
- **Eliminar capacidad:** borrar una entrada de la lista\_C, pero dejando inalterado al objeto.
- **Eliminar objeto:** eliminar un objeto y su capacidad.

Los sistemas basados en capacidades se suelen organizar como una colección de módulos, siendo cada **módulo** un gestor para cada tipo de objeto. Como ejemplo: **gestor de ficheros** y **gestor de memoria**

Cada operación va acompañada de la capacidad correspondiente.



# Capacidades

Un problema que tiene este esquema basado en módulos gestores  son programas ordinarios

Ejemplo: gestor de ficheros

Las capacidades asociadas a ficheros permiten a los programas las operaciones típicas sobre éstos. Sin embargo, el gestor de ficheros ha de ser capaz de acceder a la representación interna de los ficheros.

Es esencial que los módulos gestores tengan más atribuciones de las que las capacidades permiten a los procesos ordinarios.

Solución del sistema  
**Hydra**



técnica llamada **amplificación de derechos**



Se les asigna a los módulos gestores una plantilla de permisos que les permiten obtener más derechos sobre los objetos de los que les permiten las capacidades.



# Capacidades

## Problema

En los sistemas basados en capacidades es **difícil revocar los accesos a un objeto**, ya que ello supone localizar todas las capacidades que hacen referencia al objeto, y éstas se encuentran en las listas de capacidades, que se encuentran repartidas por todo el disco.

## Solución

- ✓ Consiste en que una capacidad no apunte al objeto en sí, sino a un objeto indirecto que apunta al objeto.
- ✓ Asociar a cada objeto un número aleatorio, que también está presente en la capacidad. Cuando se presenta una capacidad al solicitar un acceso, los dos números son comparados y el acceso se permite si la comparación ha tenido éxito.



La forma de revocar los accesos consiste en que el propietario solicite que el número del objeto sea cambiado, lo que automáticamente invalida el resto de las capacidades.



## Caso de estudio: Win NT/2000

Esquema uniforme para el control de accesos



se aplica tanto a los procesos como hebras, ficheros, semáforos, ventanas y otros objetos del sistema.

El control de acceso se lleva a cabo a través de dos entidades:

1. **Token de acceso**, que está asociado a cada proceso
2. **Descriptor de seguridad**, que está asociado a todo objeto.

Cuando un usuario se conecta a un sistema Windows NT, usa un esquema nombre/contraseña para autenticar al usuario.

Si la conexión es aceptada, se crea un proceso para ese usuario y se le asocia un token de acceso.



## Caso de estudio: Win NT/2000

El token de acceso presenta la siguiente estructura:

- **Identificador de seguridad** (*Security ID* o *SID*): identifica a un usuario. Se corresponde, normalmente, con el nombre que el usuario introduce al conectarse.
- **Identificadores de grupo** (*Group SIDs*): lista de grupos a los que pertenece el usuario. El acceso a un objeto se puede definir en función del *SID* de grupo, del *SID* individual, o una combinación de ambos.
- **Privilegios**: lista de servicios del sistema que el usuario puede solicitar. La mayoría de los usuarios no tienen privilegios.
- **Propietario por defecto**: normalmente, el propietario de un proceso es el del usuario lo crea.
- **Lista de control de accesos por defecto**: lista que indica las protecciones aplicadas inicialmente a los objetos que crea el proceso.





## Caso de estudio: Win NT/2000

El token de acceso tiene dos funciones:

- Mantener toda la información de seguridad unida para permitir que la validación de los accesos se realice con rapidez. El subsistema de seguridad puede usar el token asociado con un proceso para determinar los privilegios de acceso del usuario.
- Permite que cada proceso pueda modificar su características de seguridad hasta cierto punto, sin que ello afecte a otros procesos del usuario.



## Caso de estudio: Win NT/2000

Cada objeto que puede ser accedido por un proceso tiene asociado un **descriptor de seguridad**, que presenta la siguiente estructura:

- ❑ **Lista de control de acceso del sistema** (*System Access Control List* o *SACL*):
  - o Especifica las operaciones sobre el objeto que deberían de generar mensajes de auditoría.
  - o Las aplicaciones no deben tener privilegio para leer o escribir la lista de control de acceso del sistema.
- ❑ **Propietario**: El propietario del objeto puede realizar, generalmente, cualquier acción sobre el descriptor de seguridad. Puede ser un individuo o un grupo.
- ❑ **Lista de control de acceso discrecional** (*Discretionary Access Control List*):
  - o Indica qué usuarios y grupos pueden acceder al objeto y con qué operaciones.
  - o Consiste en una lista de entradas de control de acceso (ACE).
- ❑ **Indicadores**: Define el tipo y contenidos del descriptor de seguridad.

## Caso de estudio: Win NT/2000

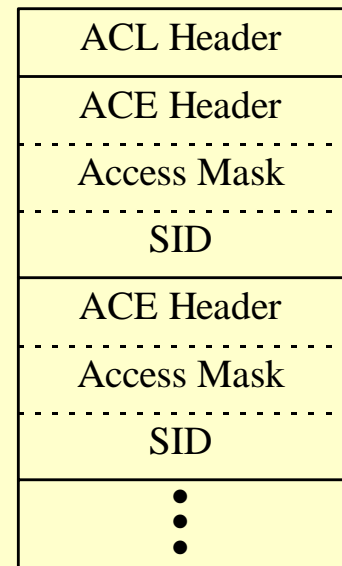
Cada **lista de control de acceso** consiste en una cabecera y en un número variable de ACEs.

Cada entrada especifica un SID individual o de grupo y una máscara de acceso que define los derechos que son permitidos a este SID.

### Funcionamiento:

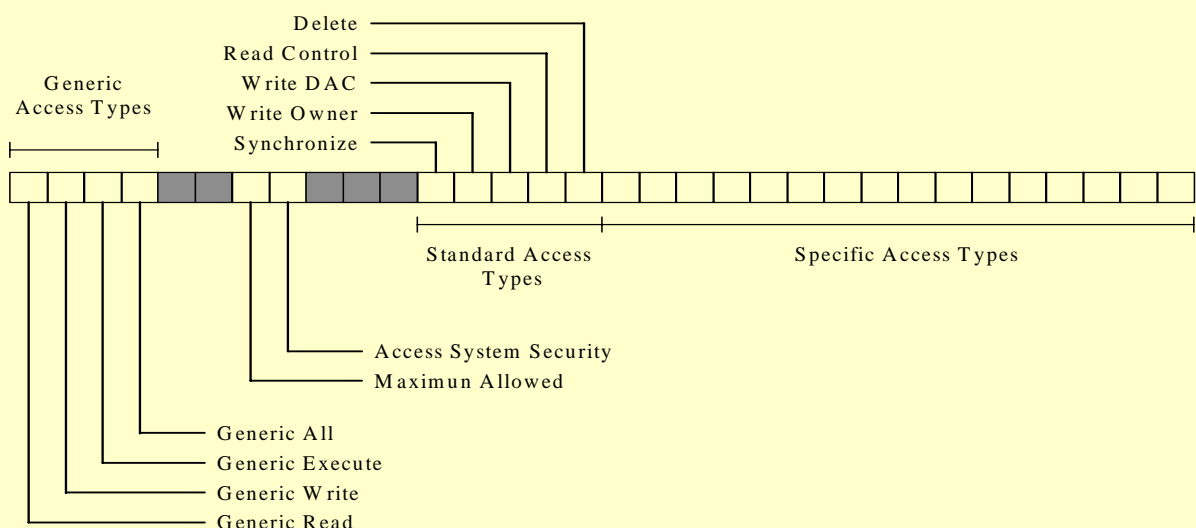
Cuando un proceso intenta acceder a un objeto, el gestor de objetos de Windows NT lee los SID individual y de grupo del token de acceso y realiza una búsqueda en la DACL del objeto.

Si se encuentra una ACE cuyo SID coincida con uno de los del token de acceso, entonces el proceso tiene los derechos de acceso especificados en la máscara de acceso de la ACE.



## Caso de estudio: Win NT/2000

La **máscara de acceso** tiene 32 bits. Los 16 bits menos significativos especifican derechos de acceso que se aplican a un tipo particular de objeto.



## Criptografía – Conceptos Básicos

- La Criptografía es la herramienta más poderosa para proporcionar servicios de seguridad en los sistemas informáticos
- El proceso de disfrazar un *texto en claro* para convertirlo en ininteligible se llama **cifrado**. La conversión de un texto cifrado a un texto en claro se llama **descifrado**.
- **Criptografía**: ciencia que estudia cómo mantener la seguridad en los mensajes ( $M$ )
- **Criptoanálisis**: ciencia que estudia cómo romper los textos cifrados
- **Criptología**: Criptografía + Criptoanálisis



## Criptografía – Conceptos Básicos

- La función de cifrado se denota por  **$E$**  ; opera sobre el mensaje  **$M$**  para producir el texto cifrado (o criptograma)  **$C$** :

$$E(M) = C$$

- La función de descifrado se denota por  **$D$** ; opera sobre  **$C$**  para producir el mensaje  **$M$** :

$$D(C) = M$$

- El objetivo del cifrado y del posterior descifrado de un mensaje es obtener el texto en claro original; así :

$$D(E(M)) = M$$



# Criptografía – Algoritmos y claves

**Algoritmo criptográfico** o **cifrado**: función utilizada para cifrar y descifrar

Tanto las operaciones de cifrado como descifrado utilizan una clave

$$E_K(M) = C \quad D_K(C) = M \quad D_K(E_K(M)) = M$$

Algunos algoritmos utilizan para cifrar una clave diferente que para descifrar ( $K_1$  y  $K_2$ ):

$$E_{K_1}(M) = C \quad D_{K_2}(C) = M \quad D_{K_2}(E_{K_1}(M)) = M$$

La seguridad de estos algoritmos reside en mantener secreta la/s clave/s



el algoritmo puede ser publicado y analizado

**Criptosistema**: algoritmo + textos en claro + textos cifrados + claves



# Criptografía – Algoritmos simétricos

También llamados algoritmos de **clave secreta**

La clave para cifrar se puede calcular a partir de la clave para descifrar y viceversa (coinciden en la mayoría de los casos)

Requieren que el emisor y el receptor acuerden, a priori, una clave

Dos categorías:

1. **Cifrados en flujo**: operan, cada vez, sobre un único bit (o byte) de texto en claro
2. **Cifrados en bloque**: operan, cada vez, sobre un bloque de bits de texto en claro





## Criptografía – Algoritmos asimétricos

También llamados algoritmos de **clave pública**

La clave utilizada para cifrar es distinta de la utilizada para descifrar, y no puede ser calculada a partir de la anterior

La clave para cifrar se puede hacer pública (**clave pública**)

- cualquiera puede cifrar un mensaje
- sólo quien tenga la correspondiente clave de descifrado (**clave privada**) es capaz de descifrar

Se puede utilizar la clave privada para cifrar y la pública para descifrar; esto se llama **firma digital**



## Criptografía – Cifrados clásicos

Antes de la existencia de ordenadores, la criptografía consistía en algoritmos basados en caracteres

Los algoritmos criptográficos o bien sustituían caracteres o bien los transponían

Cifrado por **sustitución**: cifrado en el que cada carácter del texto en claro se sustituye por otro carácter en el texto cifrado.

Cifrado por **transposición**: consiste en realizar una permutación de las posiciones que ocupan los símbolos en el mensaje en claro.

Cifrados **producto**: combinan sustitución y transposición. Se pueden considerar como la aplicación sucesiva de varios cifrados



## Criptografía – Criptosistemas simétricos

Data Encryption Standard (DES)

DES

DES triple

GDES

NEWDES

FEAL (Fast Encryption Algorithm)

IDEA (International Data Encryption Standard)



## Criptografía – Criptosistemas asimétricos

El concepto de criptografía de clave pública fue inventado por Diffie y Hellman, e independientemente por Merkle.

Cada usuario  $X$  posee dos claves:

- una **secreta**, conocida sólo  $X$ , su dueño, y utilizada por éste para descifrar todo lo que los demás usuarios hayan cifrado con la clave pública
- otra **pública**, conocida por los demás usuarios y que éstos utilizan para cifrar la información que desean enviar a  $X$

La clave pública debe estar disponible, para lo cual puede crearse una especie de directorio

Ambas claves se caracterizan por tener una gran longitud



# Criptografía – Criptosistemas asimétricos

## Características:

- es computacionalmente imposible deducir la clave privada a partir de la clave pública
- cualquiera con la clave pública puede cifrar un mensaje, pero no descifrarlo
- sólo la persona con la clave privada puede descifrar el mensaje
- cifrar el mensaje con la clave pública es como poner el correo en un buzón (todo el mundo puede hacerlo)
- descifrar el mensaje con la clave privada es como coger el correo del buzón (sólo el que tiene la llave del buzón puede hacerlo)



# Criptografía – Criptosistemas asimétricos

## Ventajas sobre los de clave secreta (simétrica):

- Cada usuario establece sus claves
  - Sin embargo, un usuario de un sistema de clave secreta necesita una comunicación previa con su interlocutor
  - De lo anterior se deduce una mayor seguridad de los sistemas de clave pública sobre los de clave secreta
- Menor número de ataques criptoanalíticos posibles. Muchos de los ataques válidos para los criptosistemas de clave secreta no tienen aquí ningún sentido
- Suponiendo ambos sistemas con  $n$  usuarios, en los de clave pública sólo se deben manejar  $2n$  claves [ $n(n-1)/2$  en los simétricos]

Las principales desventajas de los sistemas de clave pública son sus mayores complejidades en el tiempo y/o en el espacio:

- son bastante más lentos que cualquier sistema de clave secreta y la longitud de los textos cifrados con los que trabajan también es bastante mayor



## Criptografía – Criptosistemas asimétricos

Estas características hacen que los sistemas de clave pública no puedan reemplazar a los de clave secreta (que usan técnicas rápidas como las permutaciones o las sustituciones)

Hay un par de aplicaciones donde es preferible el uso de los sistemas de clave pública:

- *La distribución de claves secretas*
- *Las firmas digitales*

La base técnica es muy reducida

La gran mayoría de todos los sistemas de clave pública se fundamentan en la exponenciación y/o el producto de números primos



## Criptografía – Criptosistemas asimétricos

Parece preocupante el escaso número de sistemas de clave pública

En los primeros años de existencia, se inventaron tres sistemas:

- uno fue roto (Algoritmo de la Mochila de Merkle-Hellman)
- otro es considerado imposible de llevar a la práctica (Esquema de McEliece)
- y el tercero (el RSA) es el punto de partida para nuevos sistemas



# Criptografía – Criptosistemas asimétricos

## RSA

Es el criptosistema de clave pública más usado, con diferencia

Su nombre procede de sus inventores, Rivest, Shamir y Adleman

Se basaron para su descubrimiento en 1.977 en una idea bastante simple:

*“es muy fácil multiplicar dos números enteros primos grandes, pero extremadamente difícil hallar la factorización del producto”*

Puede darse a conocer dicho producto sin que nadie descubra esos números de procedencia, a no ser que conozca al menos uno de ellos



# Criptografía – Criptosistemas asimétricos

## RSA

### Valores necesarios:

- encontrar dos números primos grandes **p** y **q**, y calcular  $n = p * q$
- se calcula  $\phi(n)$ , de forma que  $\phi(n) = (p - 1) * (q - 1)$
- se elige aleatoriamente un número grande **d** tal que

$$\text{MCD}(d, \phi(n)) = 1$$

(o sea,  $d$  y  $\phi(n)$  son primos relativos)

- se calcula otro número, **e**:  $e * d \equiv 1 \pmod{\phi(n)}$



# Criptografía – Criptosistemas asimétricos

## RSA

$n$ ,  $d$  y  $e$  (y por supuesto,  $p$  y  $q$ ) son la base del sistema:

$n$	módulo
$d$	clave secreta
$e$	clave pública

Uso { La clave secreta para descifrar y firmar mensajes  
La clave pública para cifrar y verificar la firma

Para *cifrar*:  $C = M^e \pmod{n}$  Para *descifrar*:  $M = C^d \pmod{n}$


Algoritmo exponencial



# Criptografía – Criptosistemas asimétricos

## RSA

El texto original,  $M$  (y también el cifrado,  $C$ ) debe tomarse como un código decimal. Así, si se trabaja con el código ASCII:

$M = \text{"Hola"}$   valor decimal 72,111,108,097

Como se está trabajando en aritmética modular, todos los valores usados deben ser menores que el módulo,  $n$ .

Si  $M > n$ , entonces  $M$  debe ser troceado en otros menores que  $n$ .

Suponiendo  $n = 100,000$ , bloques:  $m_0 = 72,111$ ,  $m_1 = 10,809$  y  $m_2 = 7$



# Criptografía – Criptosistemas asimétricos

## RSA

Para seleccionar  $p$  ó  $q$  debe testearse si es primo o no.

También hay que tener cuidado al elegir el tamaño

- cuanto mayor sea, más seguro pero también más lento será el sistema en sus cálculos
- a pesar de usar los mejores algoritmos de primalidad y técnicas de exponenciación, las operaciones en RSA son unas 1,000 veces más lentas que en DES
- la fuerza del RSA reside en la dificultad de factorizar su módulo,  $n$ . Si es pequeño, se puede calcular fácilmente dicha factorización (con búsqueda exhaustiva)



# Criptografía

## Comunicación mediante criptografía de clave privada

1. Alice y Bob acuerdan un criptosistema
2. Alice y Bob acuerdan una clave
3. Alice toma su mensaje en claro y lo cifra utilizando el algoritmo de cifrado y la clave. Esto crea el mensaje cifrado
4. Alice envía el mensaje cifrado a Bob
5. Bob descifra el mensaje cifrado con el mismo algoritmo y la misma clave que utilizó Alice, y lee el mensaje

Un buen criptosistema es aquel en el que toda la seguridad reside en el conocimiento de la clave y no en el conocimiento del algoritmo

Con un algoritmo simétrico, Alice y Bob pueden realizar el paso 1 en público, pero el paso 2 ha de realizarse en secreto



### Comunicación mediante criptografía de clave privada

La clave debe permanecer secreta antes, durante y después del protocolo, tanto tiempo como el mensaje deba permanecer secreto.

Los criptosistemas simétricos tienen los siguientes problemas:

- las claves se deben distribuir en secreto; ardua tarea para sistemas de cifrado de alcance mundial
- si la clave queda comprometida (robada, adivinada, obtenida por la fuerza, ...), se puede no sólo descifrar mensajes sino producirlos
- asumiendo una clave por cada dos usuarios, una red de  $n$  usuarios requiere  $n(n-1)/2$  claves



### Comunicación mediante criptografía de clave pública

1. Alice y Bob acuerdan un criptosistema de clave pública
2. Bob envía a Alice su clave pública
3. Alice cifra el mensaje utilizando la clave pública de Bob y se lo envía a Bob
4. Bob descifra el mensaje de Alice utilizando su propia clave privada

Gran ventaja: *la criptografía de clave pública resuelve el problema de la administración de claves de los criptosistemas simétricos*

Normalmente no se utilizan para cifrar mensajes:

- los algoritmos de clave pública son lentos
- los algoritmos de clave pública permiten que el criptoanalista cifre los posibles textos en claro y los compare con  $C$  (mediante sucesivas comparaciones puede descubrir  $M$ )





## Comunicación mediante criptosistemas híbridos

1. Alice obtiene de la base de datos la clave pública de Bob
2. Alice genera una *clave de sesión* aleatoria,  $K_{AB}$ , y la cifra utilizando la clave pública de Bob  

$$\text{Alice: } E_{Bpu}(K_{AB})$$
3. Bob, con su clave privada, descifra el mensaje de Alice para recuperar la clave de sesión  

$$\text{Bob: } D_{Bpr}(E_{Bpu}(K_{AB})) = K_{AB}$$
4. Ambos cifran sus comunicaciones utilizando la clave de sesión

La criptografía de clave pública se suele utilizar para asegurar y distribuir las claves de sesión; que posteriormente utilizan los algoritmos simétricos.



Esto se llama **criptosistema híbrido**



## Criptografía - Firma de documentos

La **firma digital** es el procedimiento de seguridad que permite al autor de un mensaje (binario) firmarlo con las mismas propiedades que tiene la firma de un documento sobre papel

La utilidad de los protocolos de firma digital (protocolos de autenticación) es que el receptor de un mensaje se convenza de la identidad del emisor y de la integridad del mensaje

1. Alice cifra el mensaje con su clave privada; por lo tanto, firma el documento  

$$\text{Alice: } E_{Apr}(M)$$
2. Alice envía el mensaje firmado a Bob  

$$\text{Alice} \rightarrow \text{Bob: } E_{Apr}(M)$$
3. Bob descifra el mensaje con la clave pública de Alice; verifica con ello la firma  

$$\text{Bob: } D_{Apu}(E_{Apr}(M))$$



# Criptografía - Firma de documentos

Este protocolo satisface las características que se buscan:

1. La firma es auténtica; cuando Bob verifica el mensaje con la clave pública de Alice, sabe que ella lo firmó
2. La firma es infalsificable; sólo Alice conoce su clave privada
3. La firma no es reutilizable; la firma es una función del documento y no se puede transferir a otro documento
4. El documento firmado es inalterable
5. La firma no puede ser repudiada

Nos podemos encontrar con dos problemas:

- o Las firmas no llevan ni fecha ni hora
- o El proceso de firma es muy lento si el documento  $M$  es muy largo, debido a la lentitud implícita del algoritmo de clave pública



# Criptografía - Firma de documentos

1. Las firmas no llevan ni fecha ni hora
  - ✓ puede reutilizar el documento y la firma juntas (hay problemas si ha firmado, por ejemplo, un cheque digital)
  - ✓ Las firmas digitales incluyen, a menudo, *timestamps* (sellos temporales). La fecha y la hora de la firma se añaden al mensaje
2. El proceso de firma es muy lento si el documento  $M$  es muy largo
  - ✓ Para ahorrar tiempo, los protocolos de firma digital se suelen utilizar junto a las **funciones hash unidireccionales**



en vez de firmar un documento, se firma el hash del documento



# Criptografía - Firma de documentos

## Uso de Funciones Hash unidireccionales

1. Alice produce un hash unidireccional del mensaje. *Alice:*  $H(M)$
2. Alice cifra el hash con su clave privada; por lo tanto, firma el mensaje *Alice:*  $E_{Apr}(H(M))$
3. Alice envía el documento y el hash firmado a Bob *Alice → Bob:*  $M, E_{Apr}(H(M))$
4. Bob produce un hash unidireccional del documento que Alice envió *Bob:*  $H(M)$
5. Bob, utilizando el algoritmo de firma digital y con la clave pública de Alice, descifra el hash firmado. Si los hashes coinciden, entonces la firma es válida *Bob:*  $D_{Apr}(E_{Apr}(H(M)))$

### Beneficios:

- ✓ la firma se puede mantener separada del documento
- ✓ los requisitos de almacenamiento y firma son mucho menores



# Criptografía - Certificados Digitales

- Los certificados digitales permiten navegar identificados por la red Internet, dando identidad a una clave pública y se comportan como una persona en el espacio cibernético.
- Nace para resolver el problema de administrar las claves públicas y que la identidad del dueño no pueda ser falsificada.
- La idea es que una tercera entidad intervenga en la administración de las claves públicas y asegure que las claves públicas tengan asociado un usuario claramente identificado.
- Las tres partes más importantes de un certificado digital son:
  - ✓ Una clave pública
  - ✓ La identidad del implicado: nombre y datos generales,
  - ✓ La firma privada de una tercera entidad llamada autoridad certificadora que todos reconocen como tal y que válida la asociación de la clave pública en cuestión con el tipo que dice ser.
- En la actualidad casi todas las aplicaciones de comercio electrónico y transacciones seguras requieren un certificado digital.



# Criptografía - Certificados Digitales

- Los certificados digitales permiten navegar identificados por la red Internet, dando identidad a una clave pública y se comportan como una persona en el espacio cibernético.
- Nace para resolver el problema de administrar las claves públicas y que la identidad del dueño no pueda ser falsificada.
- La idea es que una tercera entidad intervenga en la administración de las claves públicas y asegure que las claves públicas tengan asociado un usuario claramente identificado.
- Las tres partes más importantes de un certificado digital son:
  - ✓ Una clave pública
  - ✓ La identidad del implicado: nombre y datos generales,
  - ✓ La firma privada de una tercera entidad (**autoridad certificadora**) que valida la asociación de la clave pública con el tipo que dice ser.
- En la actualidad casi todas las aplicaciones de comercio electrónico y transacciones seguras requieren un certificado digital.



# Criptografía - PGP

PGP = pretty good privacy

Proyecto iniciado por **Phill Zimmerman** a principios de los 90

**Motivación:** Total ausencia de herramientas sencillas, potentes y baratas, que acerquen la criptografía **seria** al usuario

Se ha convertido en uno de los mecanismos más populares para mantener seguridad y privacidad en las comunicaciones. Especialmente para correo electrónico.

Hasta principios del 2001

{  
Gratuito para usos no comerciales.  
Código fuente publicado

Actualmente

{  
Estándar internacional  
Múltiples productos PGP (PGP<sub>Disk</sub>, PGP<sub>Net</sub>)



- Problemática legal: Leyes de exportación de material criptográfico
- **Fundamentos:**
  - Criptografía asimétrica
  - Se debe poseer una clave privada propia + múltiples claves públicas



Anillo de claves o llavero

- Fácil gestión de llaves



### Características

- Codificación de Mensajes
- Firma Digital
- Armaduras ASCII
- Gestión de Claves
  - o Dos llaveros, uno para privadas y otro para públicas
  - o Huella digital (***fingerprint***)
- Distribución de Claves y Redes de Confianza
  - o Susceptible de ataques de intermediario
  - o Posibilidad de firmar claves
  - o Certificados de revocación

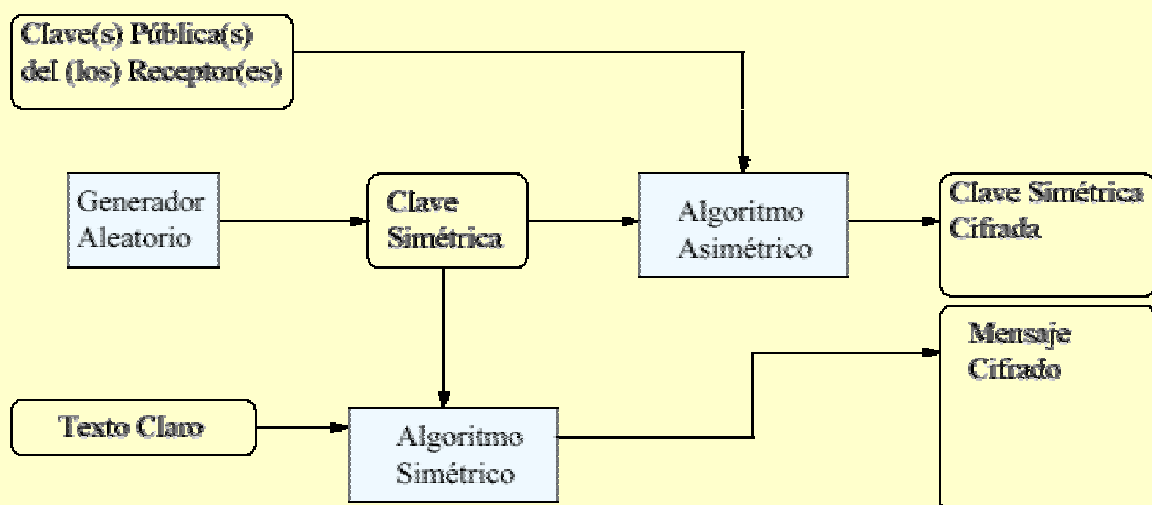


## Buenas Costumbres

- Escoger contraseñas adecuadas
- Proteger adecuadamente los archivos sensibles
- Emitir revocaciones de nuestras claves al generarlas y guardarlas en lugar seguro
- Firmar solo las claves de cuya autenticidad estemos seguros



## Codificación de un mensaje PGP



**Cerberos:** Es el perro mitológico que guarda el Hades, mundo de los muertos. Impedía la entrada de los vivos al Hades, pero sobre todo impedía la salida de los muertos. Tenía tres cabezas, aunque en algunas versiones llega a tener hasta cincuenta. Tenía el lomo erizado de serpientes, la cola de dragón y sus fauces destilaban veneno. Su sitio era la otra orilla de la laguna Estigia, por donde el barquero [Caronte](#) llevaba las almas desde la tierra al Hades.

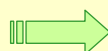
- Sistema de control de accesos y autenticación completo inventado por el MIT.
- Objetivos
  - Exigir autenticación a los usuarios para la utilización del sistema y en particular para cada servicio ofrecido
  - Exigir autenticación a los servicios



### Características

- Permite centralizar la gestión de accesos.
- Utiliza clave simétrica.
- Las claves nunca viajan por la red
- Control de acceso individualizado para cada servicio, pero solo se introduce la password una vez por sesión
- Basa el control de accesos en un sistema llamado **Servidor de Autenticación AS**.
- En la v.4 usa **DES** y en la v.5 permite cualquier algoritmo y cualquier longitud de clave.

- Intervienen:



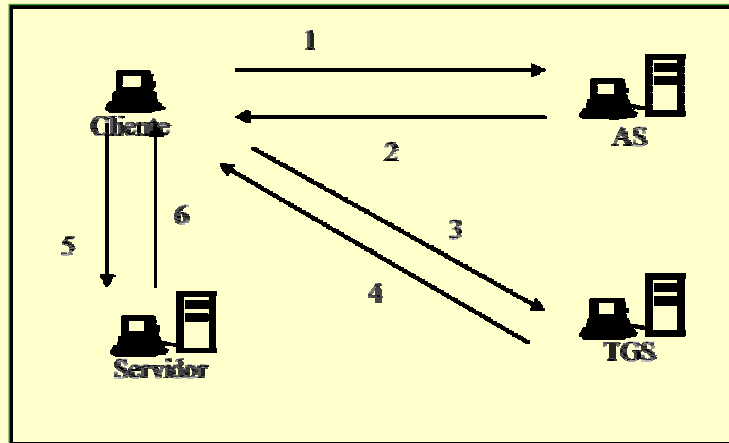
- Usuario
- Servidor de servicios
- Servidor de autenticación (**SA**)
- Servidor de concesión de tickets (**TGS**)



Funcionamiento

- Uso de tickets para validar el acceso a los servicios
- El usuario debe tener el ticket y enviarlo al servidor para obtener el acceso
- El ticket es información encriptada con fecha de caducidad, que no se puede almacenar

El proceso de autenticación se divide en **2 fases y 6 mensajes**

MENSAJES

1. Solicitud de un ticket de acceso
2. Ticket<sub>TGS</sub> + clave de sesión
3. Solicitud de un ticket de acceso al servicio
4. Ticket<sub>SERVICIO X</sub> + clave de sesión con el servicio
5. Solicitud de servicio
6. Autenticación por parte del servidor





1. Un usuario desde una workstation requiere un servicio. El password de usuario genera (con un proceso matemático) una clave para encriptar el mensaje 1.
2. AS verifica el correcto acceso del usuario a la Base de Datos, crea un ticket y una clave de sesión. Los resultados son encriptados usando la clave derivada de la password del usuario.
3. La workstation utiliza la password del usuario para desencriptar el mensaje, luego envía al TGS el ticket y el autenticador que contienen el nombre de usuario, la dirección de red y el tiempo de vida del ticket.
4. El TGS desencripta el ticket y el autenticador, verifica la solicitud y crea un ticket para ser enviado al servidor.
5. La workstation envía el ticket y el autenticador al servidor.
6. El servidor verifica que el ticket y el autenticador coincidan, luego permite el acceso al servicio.



- ❖ “Modern Operating Systems”. A.S. Tanenbaum. Prentice-Hall. 1992.
- ❖ "Operating Systems Concepts". Silberschatz, Peterson. Addison-Wesley, 1994.
- ❖ “Operating Systems, Second Edition”. Stallings, W. Prentice-Hall. 1995.



- ❖ “Criptografía y Seguridad en Computadores”. M.J. Lucena López. 2001. <http://www.kriptopolis.com>
- ❖ “Control de Accesos”. M. Pons Martorell. <http://www.kriptopolis.com>
- ❖ Kerberos:  
<http://web.mit.edu/kerberos/www/>

