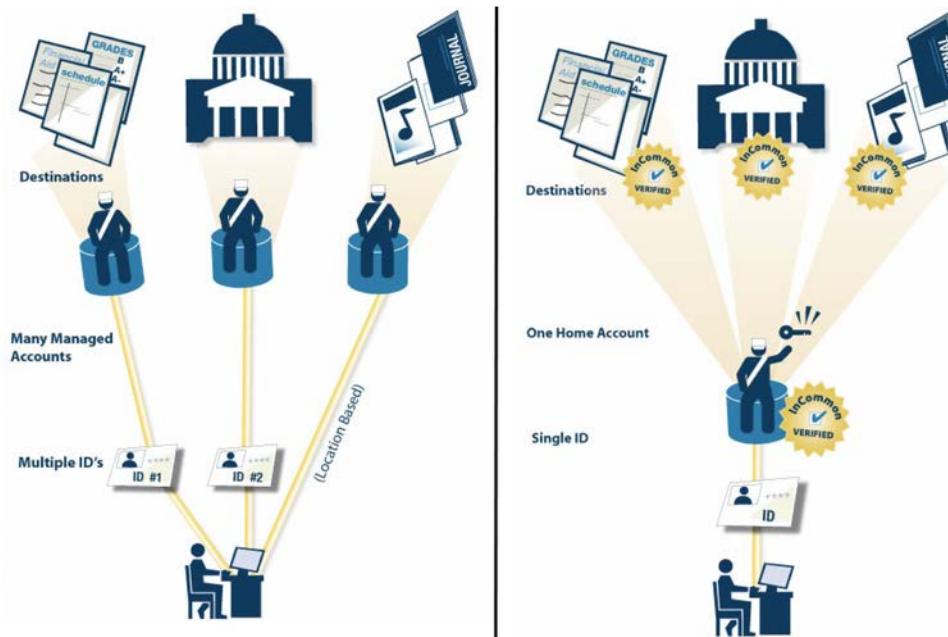


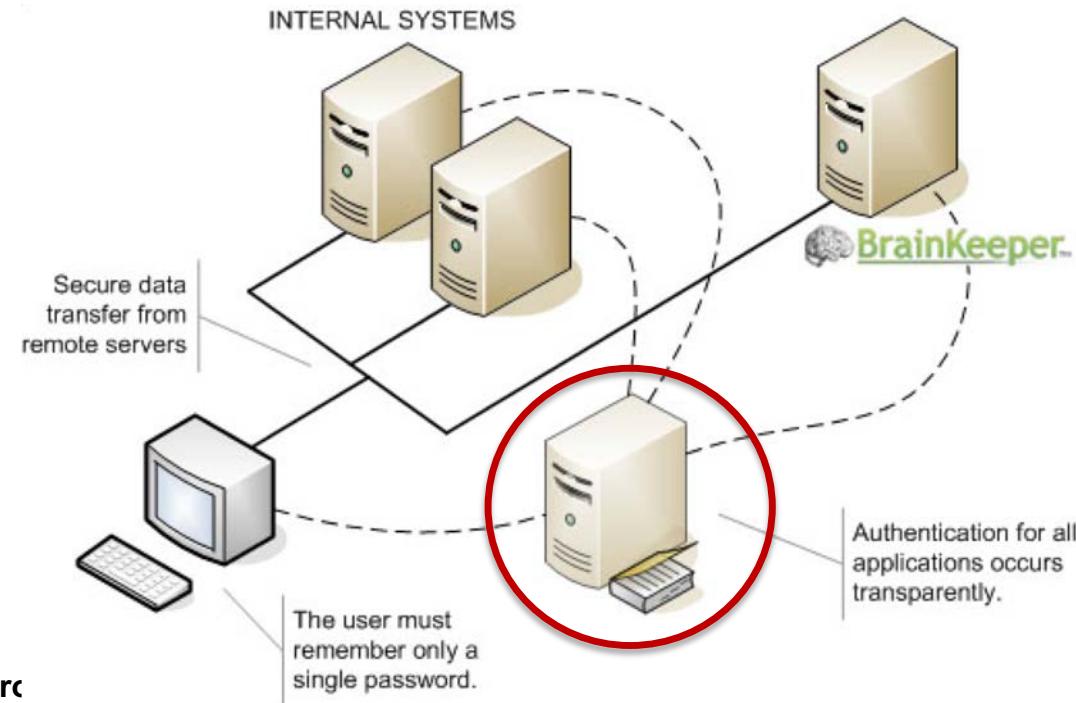
Mecanismo de Single Sign-On para Autenticación

Single Sign-On (SSO)

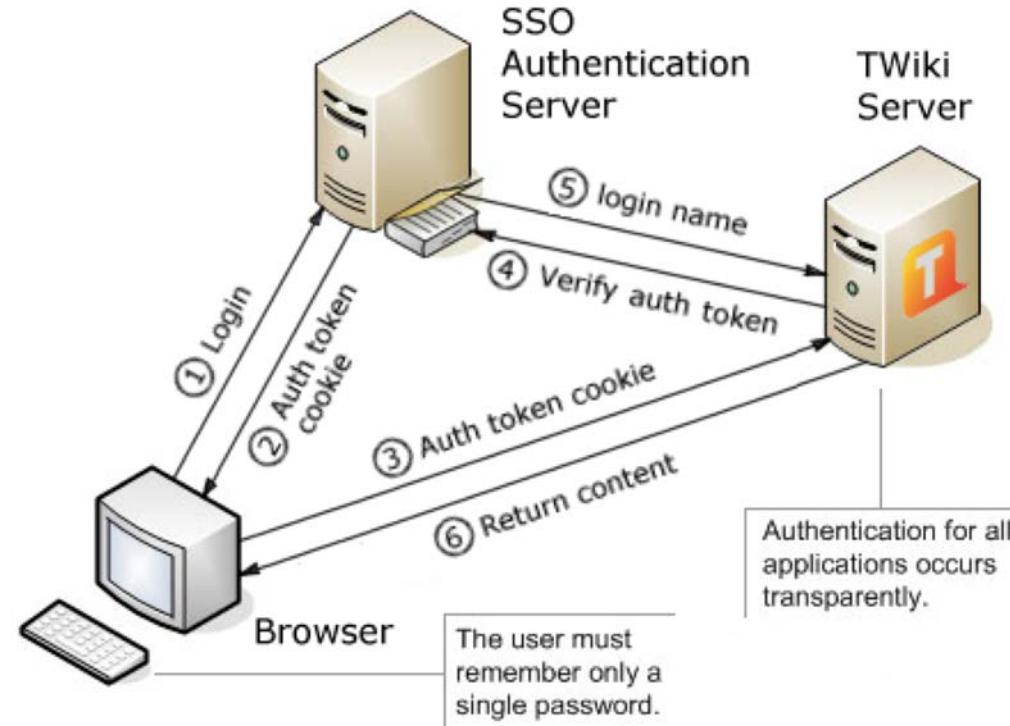
- El Single Sign-On es un mecanismo que permite a un usuario **autenticarse una sola vez** para acceder a todos los sistemas, independientes pero relacionados, a los que tiene acceso
- **Una vez autenticado, el usuario puede ir cambiando de un sistema a otro sin necesidad de autenticarse de nuevo**



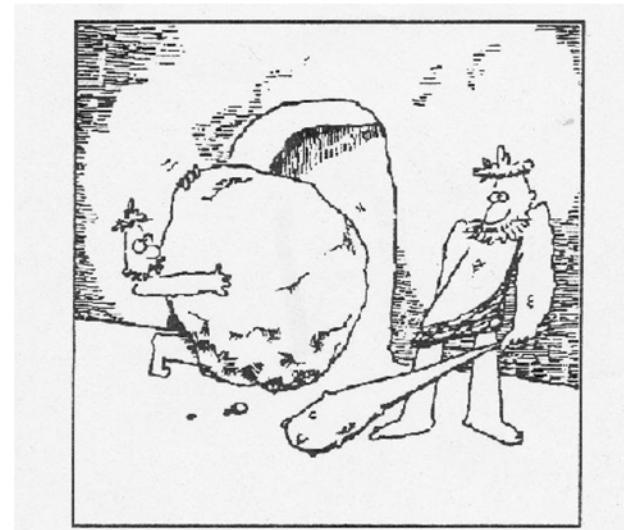
- Existen diferentes ventajas para el SSO:
 - **Usabilidad:** el usuario sólo ha de recordar un password, o usar un solo token, o un solo certificado, etc.
 - Reduce, por lo tanto, la probabilidad del error humano
 - **Seguridad:** reduce el riesgo de los ataques de intercepción
 - **Productividad:** reduce el tiempo de autenticación



- No obstante, también tiene la **desventaja** de que hay un único punto de ataques, el servidor SSO
 - Además, el intruso podrá entrar en todos los sistemas si su ataque tiene éxito aunque sea una vez



MECANISMOS DE CONTROL DE ACCESO



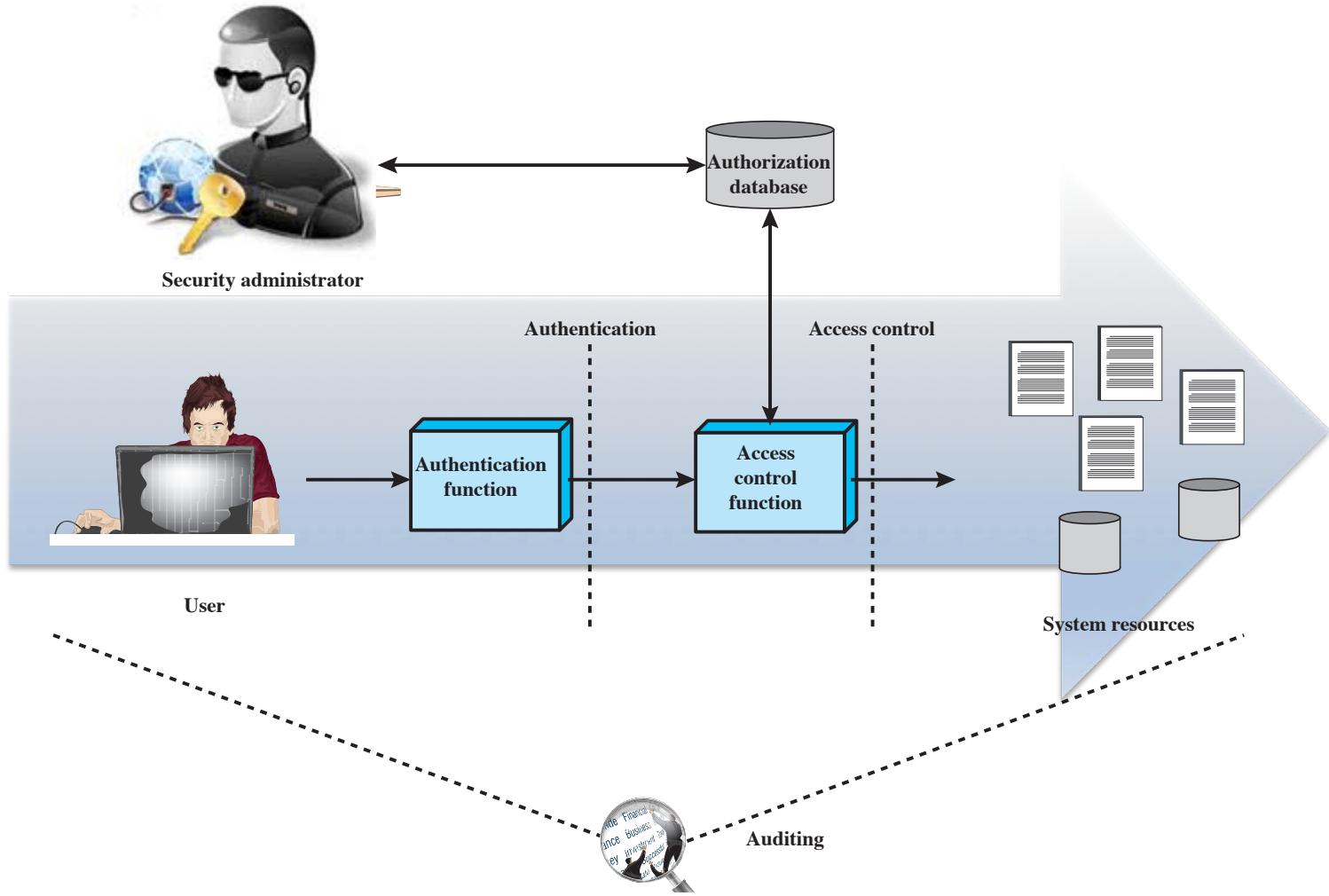
32,217 BC
FIRST ACCESS CONTROL SYSTEM

- El **control de acceso** es un elemento central – uno de los servicios esenciales – de la **Seguridad en Ordenadores**
- El RFC-2828 define la **Seguridad en Ordenadores** como:
“measures that implement and assure security services in a computer system, particularly those that assure access control service”
- Los objetivos principales del control de acceso son:
 - prevenir los accesos a los recursos por parte de usuarios no autorizados
 - prevenir que los usuarios legítimos accedan a los recursos de forma no autorizada
 - permitir a los usuarios legítimos acceder a los recursos de una forma autorizada



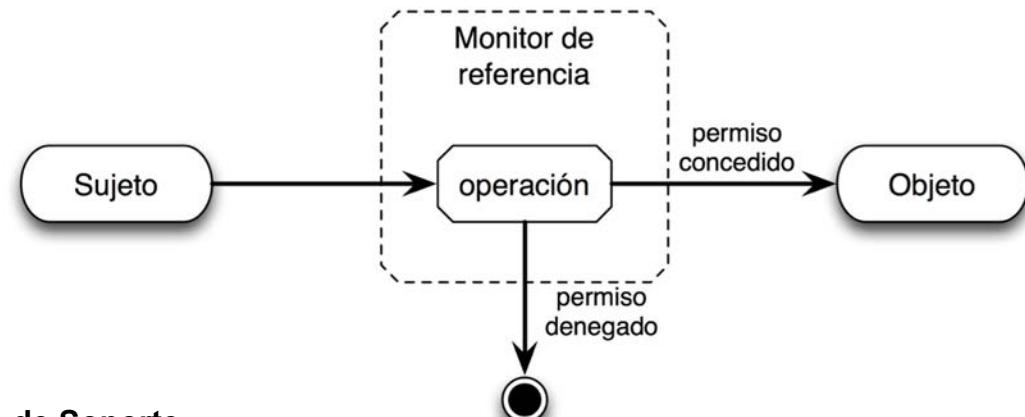
- Por lo tanto, el control de acceso implementa una **política de control de acceso** que especifica:
 - quién o qué puede tener acceso a cada recurso del sistema
 - el tipo de acceso que se permite (cuándo, cómo, etc.)
- Existe una relación clara entre el control de acceso y otros servicios de seguridad, concretamente con los servicios de **autenticación, autorización y auditoría**
 - **Autorización:** concesión de un derecho o un permiso a una entidad para acceder a un recurso
 - **Auditoría:** revisión de los registros y actividades del sistema para:
 - garantizar el cumplimiento de la política establecida y los procedimientos operacionales
 - recomendar cambios en la política y en los procedimientos
 - comprobar la adecuación de los sistemas de control
 - detectar problemas de seguridad





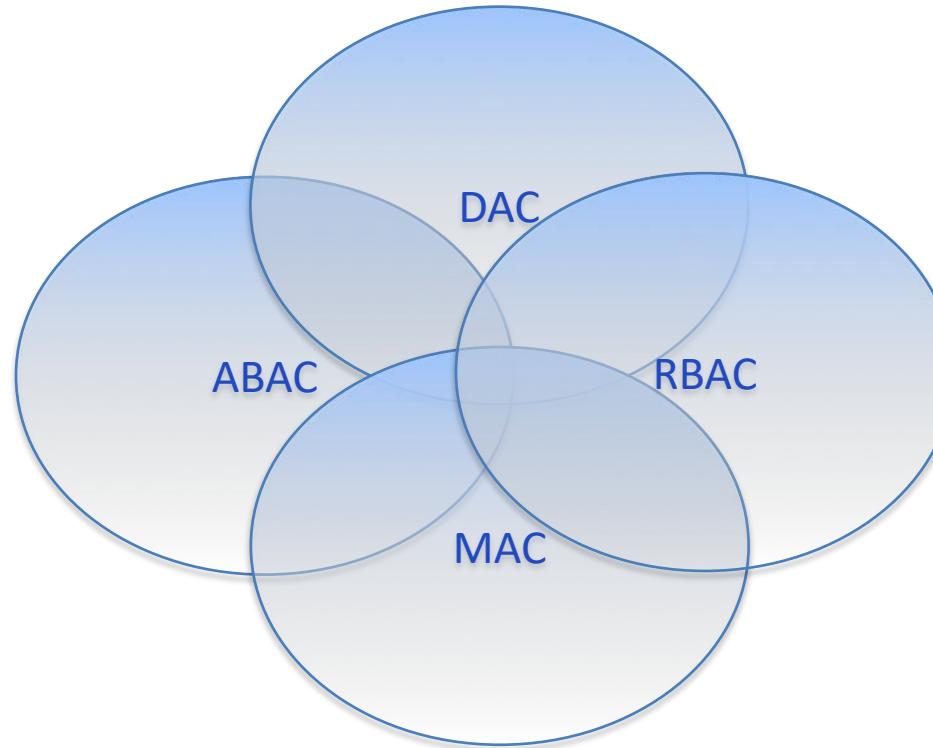
- Como se puede observar en la figura anterior, el mecanismo de control de acceso hace de **mediador entre un usuario** (o un proceso) y **los recursos del sistema**:
 - Aplicaciones
 - Sistemas operativos
 - Firewalls
 - Routers
 - Ficheros
 - Bases de datos
 - Dispositivos concretos: servidores, sensores, dispositivos móviles,
- La figura muestra un modelo simple de control de acceso, pero en la práctica puede haber **muchos componentes** que, de forma **cooperativa**, comparten la función de control de acceso

- Los elementos básicos de un control de acceso son:
 - **Objeto:** recurso al cual se controla el acceso
 - Ejemplos: registros, páginas, segmentos, ficheros, directorios, mailboxes, programas, procesadores, puertos de comunicación y nodos de red.
 - **Sujeto:** entidad que potencialmente accede a los objetos
 - generalmente el concepto de sujeto se asimila al concepto de **proceso**
 - de hecho, cualquier usuario o aplicación consigue el acceso a un objeto a través de un proceso que lo representa
 - **Derecho de acceso:** Describe la forma en que el sujeto podría acceder al objeto
 - read, write, execute, delete, create, ...



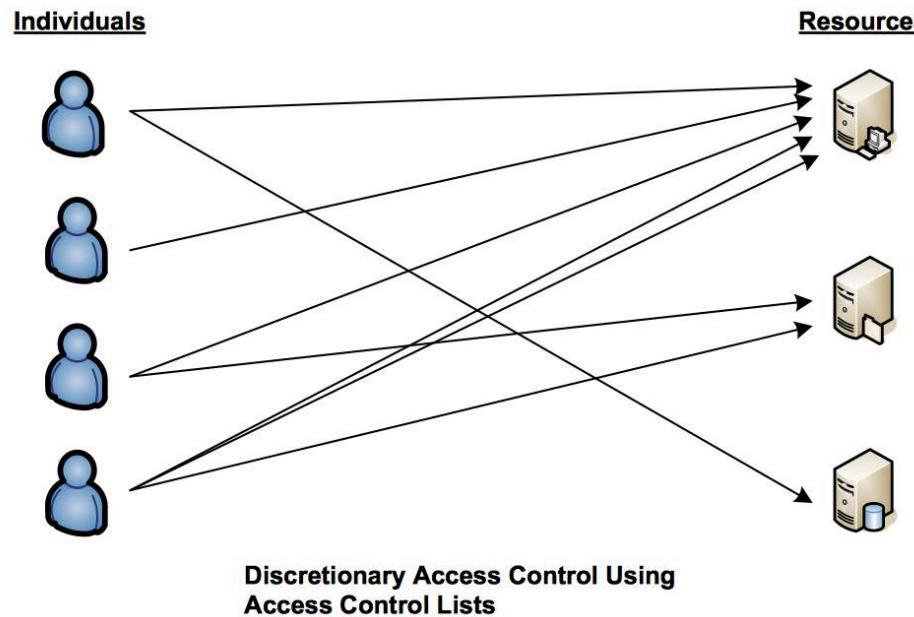
- Las esquemas de control de acceso se dividen principalmente en varias categorías:
 - **DAC (Discretionary Access Control)**: se basa en
 - **identidad del solicitante** y
 - **reglas de acceso** (que indican qué solicitantes están o no autorizados a hacer algo)
 - **MAC (Mandatory Access Control)**: se basa en comparar
 - **etiquetas de seguridad** (que indican la criticidad de los recursos) con
 - **autorizaciones de seguridad** (que indican las entidades que pueden acceder a ciertos recursos)
 - **RBAC (Role-based Access Control)**: se basa en
 - **rol** que tienen cada usuario dentro del sistema, y
 - **reglas** que indican qué accesos están permitidos a quien poseen un determinado rol
 - **ABAC (Attribute-Based Access Control)**: se base en
 - **Atributos** asociados con el usuario y que dependiendo del atributo se permite o no el acceso a un sistema
 - ...

- Estas políticas no son mutuamente exclusivas
- De hecho, un mecanismo de control de acceso puede usar dos, tres o incluso todos los mecanismos para cubrir diferentes tipos de recursos del sistema



DAC (Discretionary Access Control)

- Como se ha comentado, DAC se basa en la identidad del solicitante y en las reglas de acceso (autorizaciones) que indican qué solicitantes están o no autorizados a hacer algo



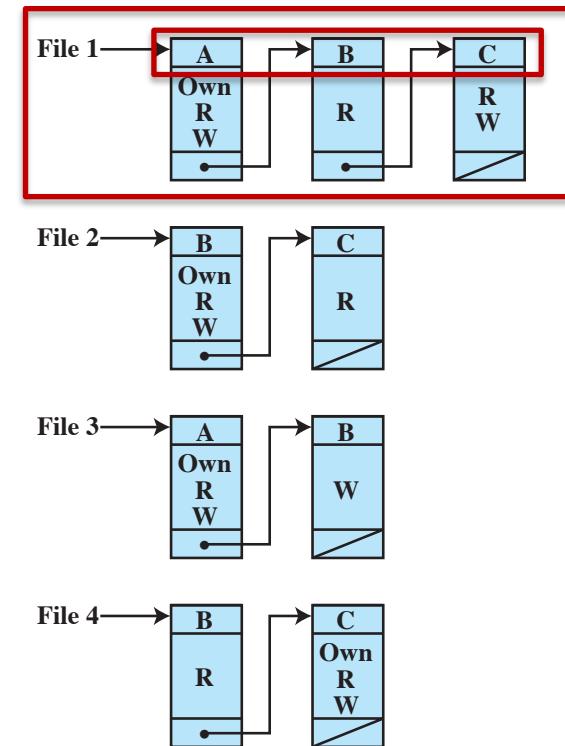
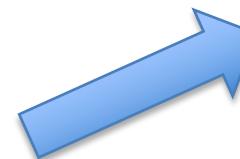
- La **matriz de acceso** es una solución general para DAC, tal y como ocurre en los S.O. y en los sistemas de administración de B.D.
- Una dimensión de esa matriz está formada por los sujetos: usuarios individuales, grupos de usuarios, equipos de red, hosts, aplicaciones, etc.
que potencialmente acceden a los recursos
- La otra dimensión de la matriz está formada por los objetos: campos individuales de datos, registros, ficheros o una base de datos, etc.
que se podrían acceder
- Cada entrada en la matriz indica los derechos de acceso del sujeto para ese objeto

SUBJECTS

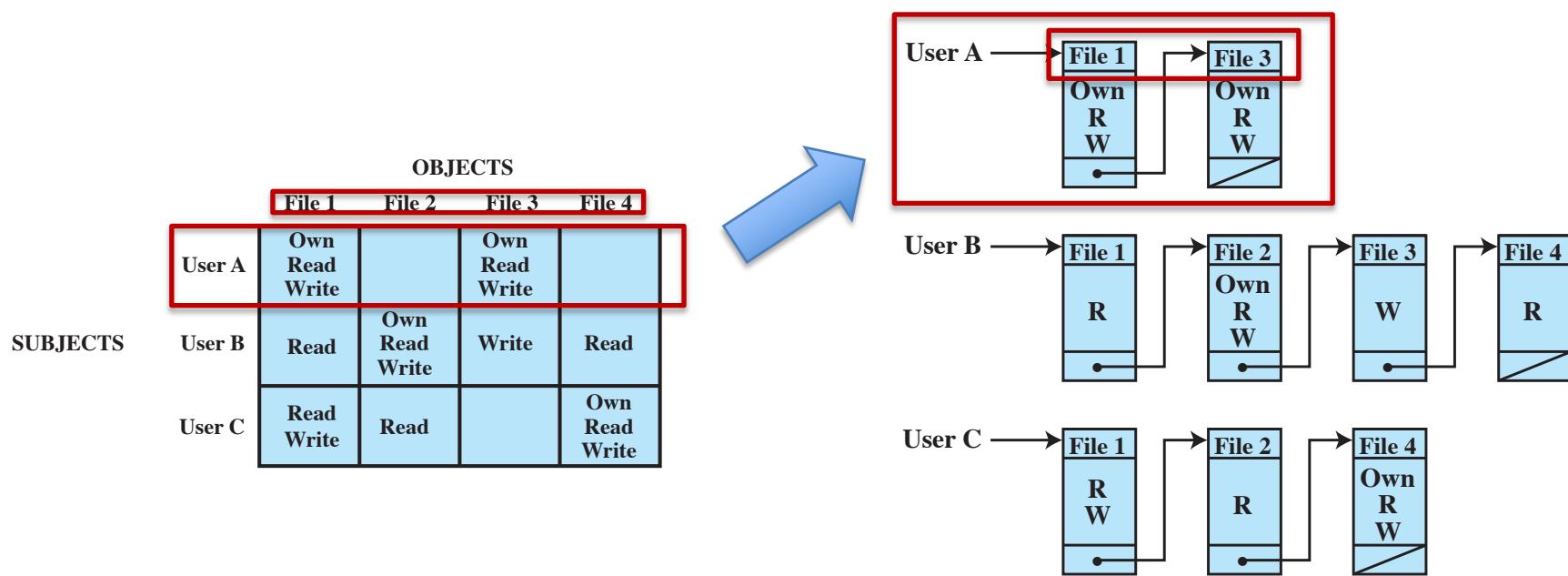
| | OBJECTS | | | |
|--------|----------------------|----------------------|----------------------|----------------------|
| | File 1 | File 2 | File 3 | File 4 |
| User A | Own Read Write | | Own Read Write | |
| User B | Read | Own Read Write | Write | Read |
| User C | Read Write | Read | | Own Read Write |

- En la práctica, una matriz de acceso se descompone en dos partes:
 - **Access Control List (ACL)**: es el resultado de la descomposición por columnas
 - por cada objeto, una ACL lista los usuarios y sus correspondientes derechos de acceso

| | | OBJECTS | | | |
|----------|--------|----------------------|----------------------|----------------------|----------------------|
| | | File 1 | File 2 | File 3 | File 4 |
| SUBJECTS | User A | Own Read Write | | Own Read Write | |
| | User B | Read | Own Read Write | Write | Read |
| | User C | Read Write | Read | | Own Read Write |



- **Ticket de capacidades (o perfil de acceso):** descomposición por filas; especifica los objetos autorizados y las operaciones para cada usuario



| Dominio | Objeto | Derechos de Acceso |
|----------------|---------------|---------------------------|
| D1 | A1 | lec, eje |
| D2 | A1 | lec, esc, borr |
| | UC2 | lec, esc, reb |
| D3 | A2 | lec, esc, eje |
| | A4 | esc, lec, eje |
| | Imp1 | imp |
| | UC2 | lec, reb |
| | A3 | esc |
| D4 | Imp1 | imp |
| | UC1 | lec, esc |
| | A3 | lec, eje |
| D5 | A2 | lec, eje |
| | A3 | lec, eje |
| | Imp2 | imp |



- Ejemplo de lista de ACL:

$$L_{bar.txt} = \{ (pepe, \{r\}), (paco, -), (luis, \{r, d\}) \}$$

$$L_{foo.exe} = \{ (pepe, -), (paco, \{x, d\}), (luis, \{x\}) \}$$



- Ejemplo de lista de ACL:

$$L_{bar.txt} = \{ (pepe, \{r\}), (paco, -), (luis, \{r, d\}) \}$$

$$L_{foo.exe} = \{ (pepe, -), (paco, \{x, d\}), (luis, \{x\}) \}$$

- Ventajas:

- Es fácil ver los permisos de acceso de un determinado objeto
- Es fácil revocar todos los permisos sobre un objeto, poniendo $L_{ob} = \{ \}$
- Es fácil eliminar los permisos asociados a un objeto que ya no existe, por simplemente eliminar L_{ob}

- Desventajas:

- Comprobar permisos de acceso de un determinado sujeto, usabilidad

- Uso:

- Se suelen implementar en sistemas orientados a la gestión de recursos, como los S.O.

- Ejemplo de lista de capacidades / perfil de acceso:

$$L_{pepe} = \{ (bar.txt, \{r\}), (foo.exe, -) \}$$

$$L_{paco} = \{ (bar.txt, -), (foo.exe, \{x, d\}) \}$$

$$L_{luis} = \{ (bar.txt, \{r, d\}), (foo.exe, \{x\}) \}$$

- Ventajas:

- Es fácil comprobar todos los permisos de un sujeto
 - Es fácil revocar todos los permisos de un sujeto, poniendo $L_{sj} = \{ \}$
 - Es fácil eliminar los permisos asociados a un sujeto que ya no existe, eliminando L_{sj}

- Desventajas:

- Comprobar los permisos de acceso sobre un determinado objeto, usabilidad

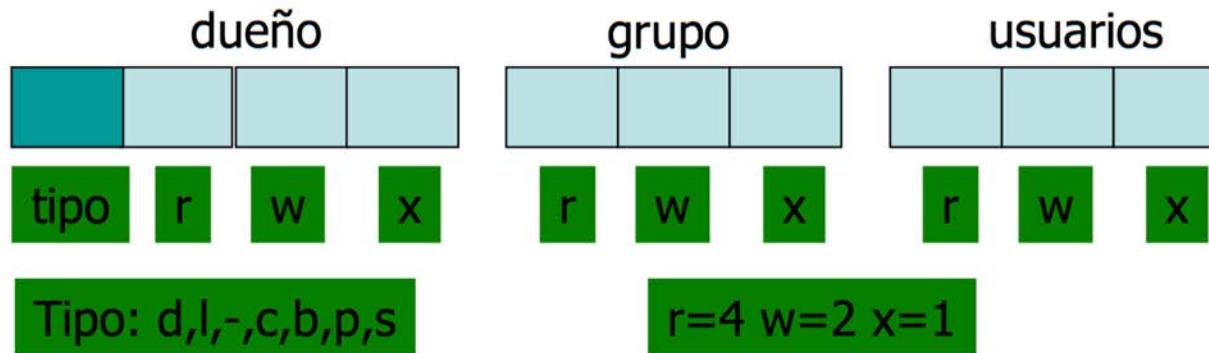
- Uso:

- Se suelen implementar en sistemas orientados al usuario, como bases de datos o sistemas distribuidos

- **Tabla de Autorización:** Es una alternativa a la matriz de acceso. Contiene una fila por cada derecho de acceso de un sujeto a un recurso
 - es de uso más ágil en comparación con las ACLs o los tickets de capacidades

| Subject | Access Mode | Object |
|---------|-------------|--------|
| A | Own | File 1 |
| A | Read | File 1 |
| A | Write | File 1 |
| A | Own | File 3 |
| A | Read | File 3 |
| A | Write | File 3 |
| B | Read | File 1 |
| B | Own | File 2 |
| B | Read | File 2 |
| B | Write | File 2 |
| B | Write | File 3 |
| B | Read | File 4 |
| C | Read | File 1 |
| C | Write | File 1 |
| C | Read | File 2 |
| C | Own | File 4 |
| C | Read | File 4 |
| C | Write | File 4 |

- Ejemplo de permisos básicos en UNIX, usando DAC



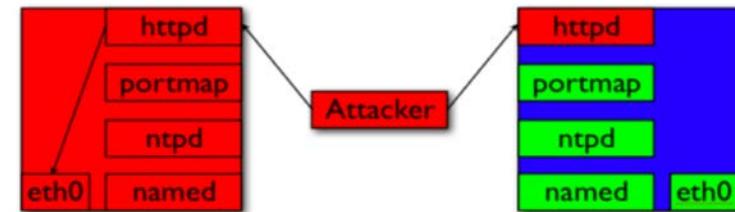
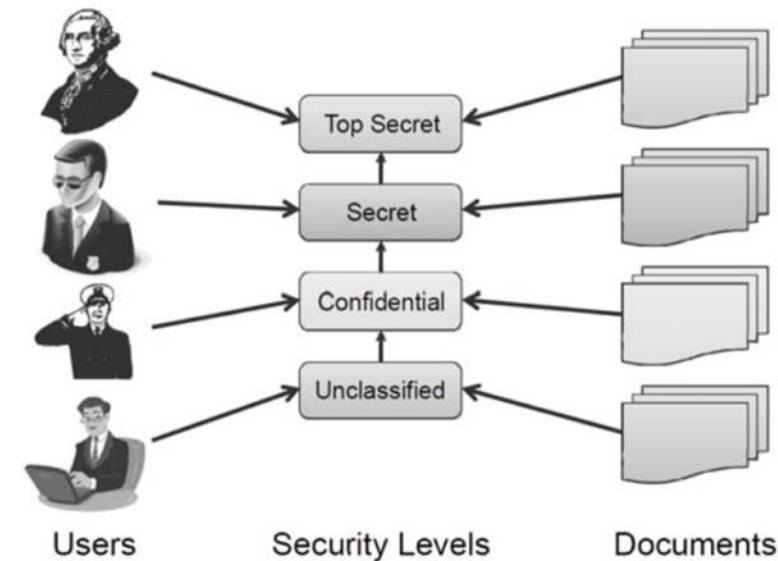
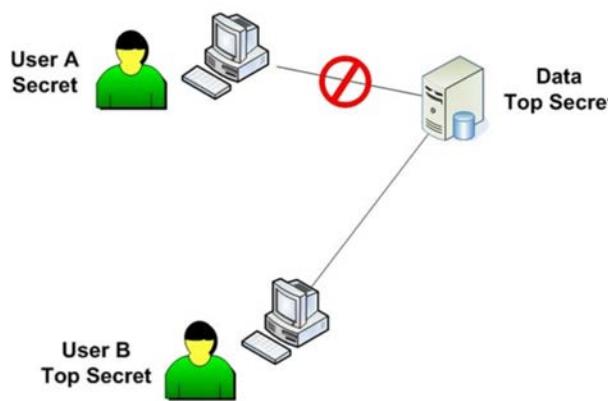
```

192.168.234.139 - PuTTY
-rw-r----- 1 root shadow 928 Feb 14 08:23 shadow
-rw----- 1 root root 928 Feb 14 08:23 shadow-
-rw-r--r-- 1 root root 165 Feb 13 22:05 shells
drwxr-xr-x 2 root root 4096 Feb 13 22:05 skel
drwxr-xr-x 2 root root 4096 Mar 1 15:30 snmp
drwxr-xr-x 3 root root 4096 Feb 14 08:14 snort
drwxr-xr-x 2 root root 4096 Feb 13 23:06 ssh
drwxr-xr-x 4 root root 4096 Feb 15 14:44 ssl
-rw-r--r-- 1 root root 2082 Feb 24 2010 sysctl.conf
drwxr-xr-x 2 root root 4096 Feb 13 22:06 sysctl.d
drwxr-xr-x 2 root root 4096 Feb 13 22:05 terminfo
drwxr-xr-x 3 root root 4096 Feb 13 23:05 texmf
-rw-r--r-- 1 root root 21 Feb 13 22:06 timezone
-rw-r--r-- 1 root root 1260 May 30 2008 ucf.conf
drwxr-xr-x 4 root root 4096 Feb 13 22:06 udev
drwxr-xr-x 3 root root 4096 Feb 13 23:05 ufw
-rw-r--r-- 1 root root 274 Nov 4 2009 updatedb.conf
drwxr-xr-x 2 root root 4096 Feb 13 22:06 vim
drwxr-xr-x 2 root root 4096 Feb 13 23:06 w3m
drwxr-xr-x 2 root root 4096 Feb 24 11:07 webalizer
-rw-r--r-- 1 root root 4496 Sep 5 2010 wgetrc
drwxr-xr-x 3 root root 4096 Feb 13 22:06 X11
drwxr-xr-x 2 root root 4096 Feb 13 23:06 xml
root@debian6:/etc#

```

MAC (Mandatory Access Control)

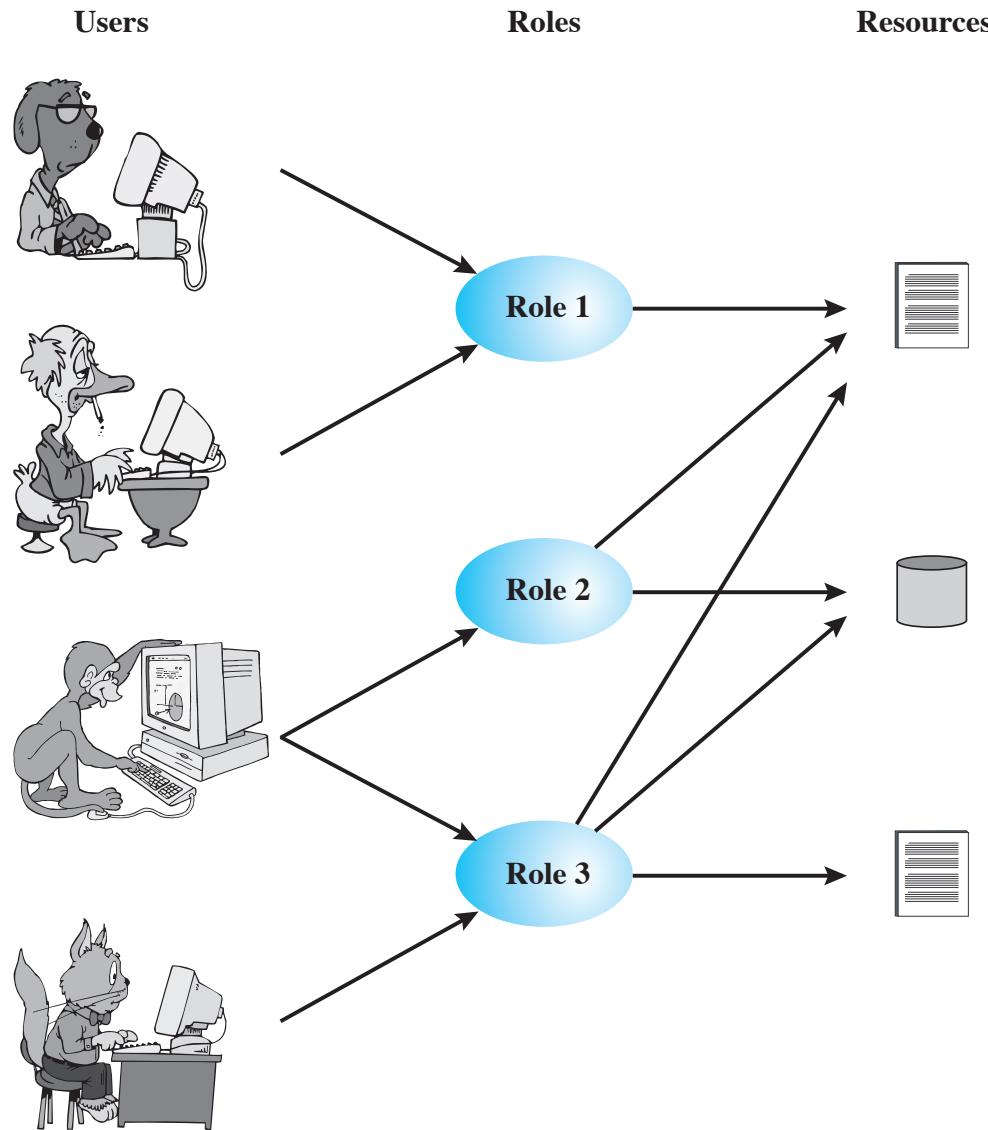
- Se basa en comparar etiquetas de seguridad (que indican la criticidad de los recursos) con las autorizaciones de seguridad (que indican las entidades que pueden acceder a ciertos recursos)
- Por lo tanto, a cada recurso se le asigna una etiqueta de seguridad, que de alguna forma lo clasifica
 - top secret – secret – confidential – restricted – unmarked – unclassified



RBAC (Role-based Access Control)

- En el modelo RBAC, se considera que un rol es una función o tarea que se lleva a cabo dentro de una empresa u organización
- No se basa en la identidad de los usuarios, sino en los roles que asumen tales usuarios. Es decir,
 - los derechos de acceso se asignan a los roles
 - y luego se asignan usuarios a esos roles
- Ese funcionamiento se fundamenta en que:
 - el conjunto de roles de un sistema, aún pudiendo ser complejo, es relativamente **estático** en la mayoría de los escenarios
 - el conjunto de recursos y los derechos de acceso específicos asociados a un rol particular tampoco cambian con frecuencia
- RBAC tiene un uso comercial bastante amplio hoy en día, y por ello ha sido estandarizado por el NIST en el documento:
 - *Security requirements for cryptographic modules (FIPS PUB 140-2, 2001)*

RBAC (Role-based Access Control)



RBAC (Role-based Access Control)

- RBAC es utilizado en:

- Oracle DBMS
- PostgreSQL
- SAP R/3
- ISIS Papyrus
- FusionForge
- Wikipedia
- Microsoft Lync
- Microsoft Active Directory
- Microsoft SQL Server
- **SELinux**

RBAC (Role-based Access Control)

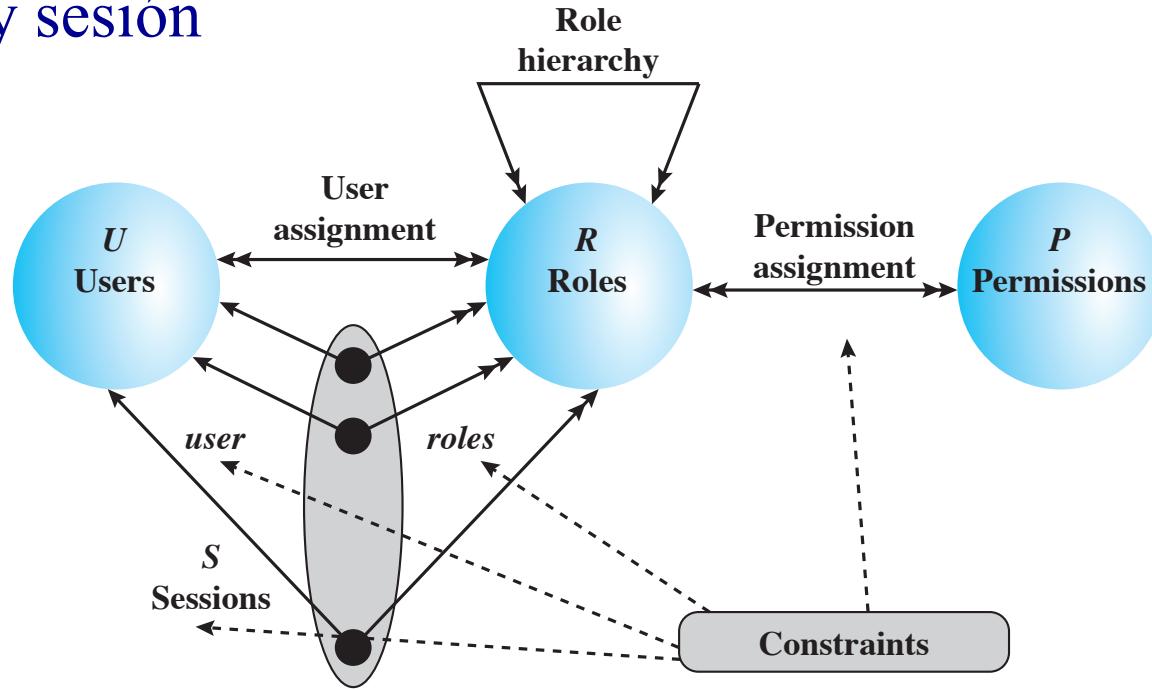
- Podemos utilizar la representación de matriz de acceso para una representación simple de los elementos clave de un sistema RBAC

| | R ₁ | R ₂ | • • • | R _n |
|----------------|----------------|----------------|-------|----------------|
| U ₁ | X | | | |
| U ₂ | X | | | |
| U ₃ | | X | | X |
| U ₄ | | | | X |
| U ₅ | | | | X |
| U ₆ | | | | X |
| • | | | | |
| • | | | | |
| • | | | | |
| U _m | X | | | |

| | OBJECTS | | | | | | | | |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| | R ₁ | R ₂ | R _n | F ₁ | F ₁ | P ₁ | P ₂ | D ₁ | D ₂ |
| R ₁ | control | owner | owner control | read * | read owner | wakeup | wakeup | seek | owner |
| R ₂ | | control | | write * | execute | | | owner | seek * |
| • | | | | | | | | | |
| • | | | | | | | | | |
| • | | | | | | | | | |
| R _n | | | control | | write | stop | | | |

RBAC (Role-based Access Control)

- El Modelo **RBAC** consta de 4 tipos de entidades: usuario, rol, permiso y sesión



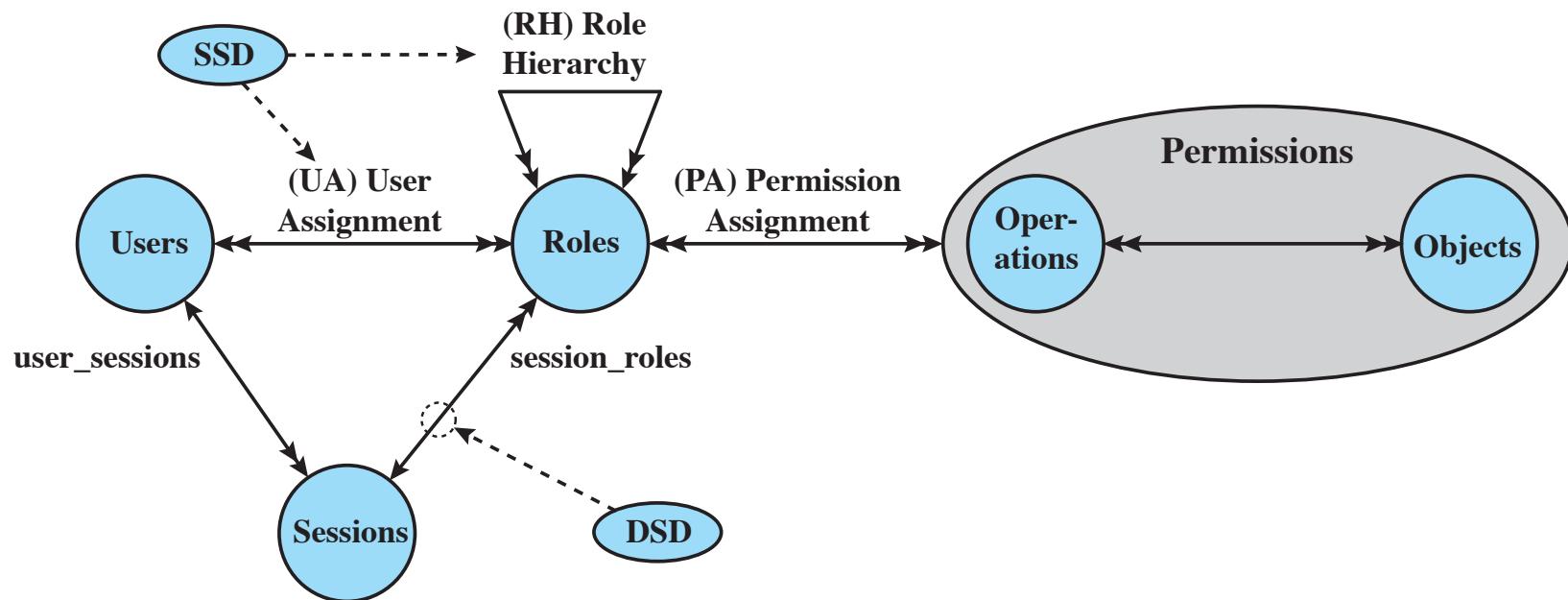
- Las relaciones muchos-a-muchos entre usuarios y roles, y entre roles y permisos proporcionan una **flexibilidad** y **granularidad** de asignación no proporcionada por DAC

RBAC (Role-based Access Control)

- RBAC permite definir:
 - **Roles mutuamente exclusivos**: es una restricción de tal forma que un usuario sólo se puede asignar a uno de los roles del conjunto
 - esta limitación puede ser estática o dinámica en una sesión
 - **Cardinalidad**: se refiere al establecimiento de un número máximo con respecto a los roles
 - número de usuarios que se pueden asignar a un rol
 - número de roles asignados a un usuario
 - número de roles que un usuario puede tener en una sesión
 - número de roles que se puede conceder a un permiso particular
 - **Prerrequisitos**: por ejemplo a un usuario sólo se puede asignar a un rol si ya está asignado a otro rol especificado

RBAC (Role-based Access Control)

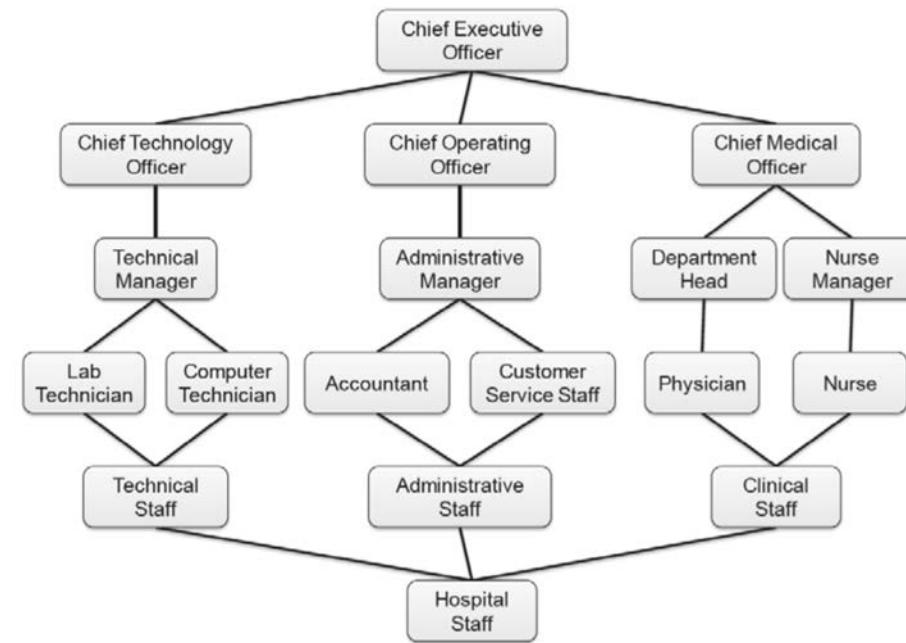
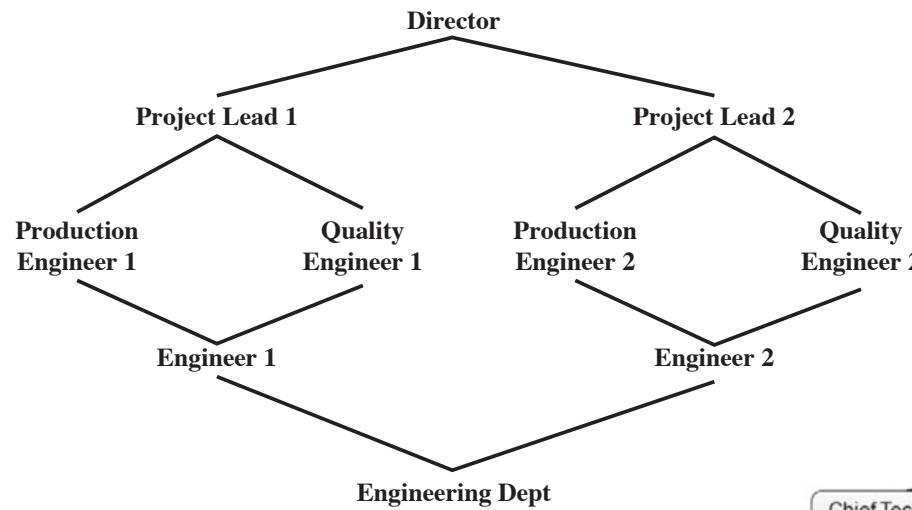
- El modelo **RBAC estándar de NIST** introduce algunas extensiones al RBAC tradicional, como:
 - **Static Separation of Duties (SSD)**: para definir roles mutuamente excluyentes
 - **Dynamic Separation of Duties (DSD)**: para definir restricciones sobre los roles que un usuario puede activar en una sesión



SSD = static separation of duty

DSD = dynamic separation of duty

- Ejemplos de jerarquía de roles:



ABAC (Attribute-Based Access Control)

- El acceso no está basado en los permisos del usuario, sino en los **atributos del usuario**, por ejemplo, tener más de 18 años, morenos
 - Cualquier usuario con más de 18 años, moreno tiene acceso al sistema
 - » lo que permite, incluso, el **acceso anónimo** si la identificación y autenticación no es estrictamente necesaria
 - Por tanto, el atributo del sujeto es lo que autoriza al usuario a acceder a un recurso

CapBAC (Capability-Based Access Control)

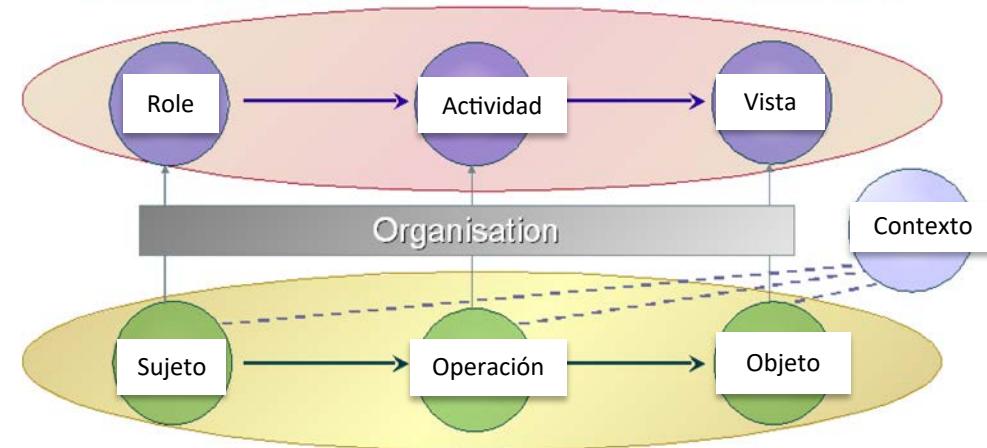
- CapBAC basa su modelo en un conjunto de roles y atributos, en donde el acceso sólo es posible si el usuario recibe del proveedor del recurso un **token** (un “certificado de autorización”) que demuestra su “capability” para realizar determinadas acciones sobre dicho recurso demandado
 - Por tanto, si un usuario necesita acceder a un recurso, éste sólo debe mostrar su certificado de autorización al proveedor antes de solicitar una operación
- La principal desventaja es que se requiere “mantener” todos los certificados de autorización

Risk-Based Access Control model

- Fue diseñado para escenarios heterogéneos compuestos de múltiples tipos de organizaciones funcionando con múltiples y diversas políticas de seguridad, y en donde no es posible predecir el número real de usuarios y recursos de acceso
 - Para gestionar este nivel heterogeneidad y los comportamientos dinámicos de su contexto, el modelo Risk-Based Access Control debe requerir de gestores y/o algoritmos funcionando en tiempo real
- El acceso es dependiendo del riesgo que se evalúa:
 - **Risk = V x P**, donde:
 - V es el valor de información (la criticidad o susceptibilidad del recurso demandado) que se puede computar de acuerdo al grado de disponibilidad al recurso demandado, el nivel de confidencialidad e integridad
 - P representa la probabilidad del acceso cuyo valor puede ser determinado por estudiar previamente un conjunto de escenarios amenazantes, las políticas de seguridad y los niveles de seguridad

OrBAC (Organizational-Based Access Control)

- Extiende y mejora el uso de RBAC, y ambos relacionan de la siguiente forma:
 - Sujetos: entidades con roles predefinidos y conteniendo específicos permisos de seguridad
 - Actividades: un conjunto de acciones a realizar en una organización bajo una misma política de seguridad (permisos asociados)
 - Vistas: relacionados con los objetos y su acceso, y todos ellos funcionando sobre políticas de seguridad preestablecidas por la organización
- Sin embargo, este modelo depende de (1) las políticas de seguridad implantadas en cada organización, y del (2) nivel de confianza de cada entidad implicada

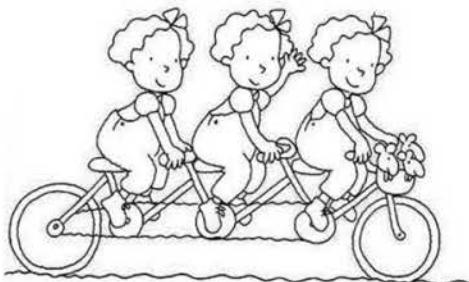


Protocolos criptográficos avanzados



- Un **protocolo criptográfico** es un algoritmo distribuido definido para alcanzar un objetivo específico de seguridad
 - consta de una secuencia de pasos que especifican, de forma precisa, las acciones a llevar a cabo por parte dos o más entidades
- Algoritmos de cifrado, firmas digitales, funciones hash, funciones MAC, generadores pseudo-aleatorios, etc. son las primitivas criptográficas que sustentan el diseño de protocolos criptográficos
- Existen multitud de protocolos criptográficos. Entre ellos protocolos como los ya vistos de administración/intercambio de claves o de autenticación de entidades
- Otros más avanzados son:
 - división y compartición de secretos, timestamping, bit-commitment, lanzamiento de monedas, poker mental, zero-knowledge, etc...

Compartición de Secretos



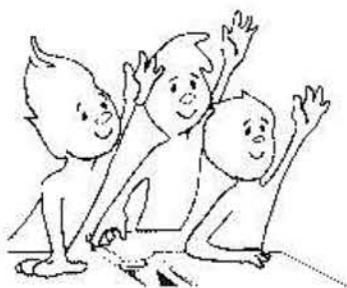
Canal Subliminal



Lanzamiento de monedas



E
-
c
o
-
c
i
-
o
n
e
s



Firma de Contratos



Protocolos
Criptográficos



Transferencia
Inconsciente



Demostraciones de
Conocimiento Nulo

Criptografía-ULL

Poker Mental

Protocolo de división de secretos

- Se usa en escenarios donde hay que dividir un mensaje M en n trozos
- Cada trozo por sí mismo no tiene valor, pero cuando se ponen todos juntos se recupera el mensaje original M
- El esquema de división más simple fracciona un mensaje entre sólo dos personas
- Es necesario utilizar una Tercera Parte Confiable, o TTP



- La división la realiza *Trent*, que crea dos trozos, uno para *Alice* otro para *Bob*:
 - ① *Trent* genera una cadena aleatoria de bits, R , de la misma longitud que el mensaje M
 - ② *Trent* hace la operación XOR de M y R para generar S

$$M \oplus R = S$$
 - ③ *Trent* distribuye R a *Alice* y S a *Bob*
- Para reconstruir el mensaje:
 - ④ *Alice* y *Bob* hacen XOR de sus trozos para reconstruir el mensaje

$$R \oplus S = M$$
- En esencia, ***Trent* está cifrando el mensaje con un one-time pad, y le da el texto cifrado a una persona y el pad a otra**



- Es fácil extender este esquema a más personas. El ejemplo siguiente lo muestra para el caso de 4 personas:
 - ① *Trent* genera tres cadenas aleatorias de bits, R , S y T de la misma longitud que el mensaje M
 - ② *Trent* realiza el XOR de M con las tres cadenas para generar U

$$M \oplus R \oplus S \oplus T = U$$
 - ③ *Trent* le da R a *Alice*, S a *Bob*, T a *Carol* y U a *Dave*
- Para reconstruir el mensaje:
 - ④ *Alice*, *Bob*, *Carol* y *Dave* computan

$$R \oplus S \oplus T \oplus U = M$$
- **El problema del protocolo de división de secretos es que si una de las partes se pierde, entonces el mensaje no se puede recuperar**

Protocolo de compartición de secretos

- Este protocolo se basa en el concepto de **esquema umbral (k-n)**
 - consiste en que se divide un mensaje M en n trozos, llamados **sombra**s, de forma que, con k de ellos, se puede reconstruir el mensaje original
 - los **esquemas umbrales** son incluso más versátiles
- El esquema umbral (k-n) se basa en una interpolación lineal
 - Dados n puntos hay uno y sólo un polinomio $q(x)$ de grado $k-1$ tal que:
$$q(x_i) = y_i, \quad \forall i$$
- Funcionamiento:
 - se divide el dato D (mensaje M codificado como un número) en n trozos de forma que D es fácilmente reconstruible a partir de k trozos cualesquiera
 - incluso el conocimiento de $k-1$ trozos no revela ninguna información sobre D



- Más concretamente, se eligen aleatoriamente los coeficientes de un polinomio de grado $k-1$:

$$q(x) = a_0 + a_1x^1 + a_2x^2 \dots + a_{k-1}x^{k-1}$$

donde $a_0 = D_0 = D$ y evaluamos:

$$D_1 = q(1), \dots, D_i = q(i), \dots, D_n = q(n)$$

- A partir de k de esos valores D_i , se pueden encontrar por interpolación los coeficientes de $q(x)$, y entonces evaluar $D = q(0)$
 - Como se ha mencionado, el conocimiento de $k-1$ de esos valores no es suficiente para calcular D

- Ejemplo:
 - Supongamos un esquema umbral (3,5) → tupla: (sombras, númer usuarios) y un mensaje secreto D , donde $D = 11$
 - o sea, “11” es el mensaje que queremos compartir entre los 5 usuarios
 - Como $k = 3$, el polinomio que necesitamos sería del tipo:
$$q(x) = a_0 + a_1x + a_2x^2$$
 - Entonces, dado que a_0 ha de ser necesariamente 11 (el mensaje que queremos compartir), sólo nos queda asignar de forma aleatoria valores a a_1 y a_2
por ejemplo: $a_1 = 2, a_2 = 1$
 - Así, el polinomio a usar resultante es:
$$q(x) = 11 + 2x + x^2$$

- Queda ahora asignar a cada uno de los cinco usuarios su trozo (sombra) del secreto. Para ello, sustituimos en el polinomio anterior la variable x por los valores 1...5, y se le entrega a cada usuario el resultado que le corresponde:

$$q(1) = 14 \rightarrow Alice$$

$$q(2) = 19 \rightarrow Bob$$

$$q(3) = 26 \rightarrow Carol$$

$$q(4) = 35 \rightarrow Dave$$

$$q(5) = 46 \rightarrow Eve$$

- Ningún usuario posee el mensaje original (11), pero cualesquiera tres de ellos pueden cooperar para recuperar ese mensaje

- Supongamos que *Alice*, *Bob* y *Dave* cooperan. Cada uno ha de aportar su sombra, más el valor de la variable x para su caso (o sea, el orden en el que recibieron la sombra)

$$Alice: q(1) = 14 = a_0 + a_1 \cdot 1 + a_2 \cdot 1$$

$$Bob: q(2) = 19 = a_0 + a_1 \cdot 2 + a_2 \cdot 2^2$$

$$Dave: q(4) = 35 = a_0 + a_1 \cdot 4 + a_2 \cdot 4^2$$

- Queda entonces un sistema de tres ecuaciones con tres incógnitas, a partir del cual se puede calcular a_0 , es decir, el mensaje D

- Dado el siguiente sistema, descubrir el valor real del mensaje oculto :

$$\left\{ \begin{array}{l} a_0 + x a_1 + x^2 a_2 = q(x) \\ a_0 + x a_1 + x^2 a_2 = q(x) \\ a_0 + x a_1 + x^2 a_2 = q(x) \end{array} \right.$$



Si además, el computo de (3,6) y su interpolación en el polinomio original resulta en:

- n = 1 → 1494
- n = 3 → 2578
- n = 4 → 3402
- n = 6 → 5614
- n = 8 → 8578
- n = 11 → 14434

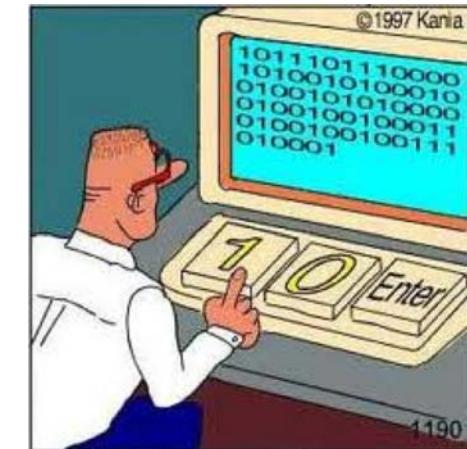
Concretamente, estudiar la reconstrucción del mensaje para n=3, n=8 y n=11

Resultado: D=1234

Protocolos de bit-commitment

- El problema general que este tipo de protocolos pretende resolver es el siguiente:

- *Alice* quiere realizar una predicción (apuesta), pero no revelarla hasta después de producirse el hecho
 - *Bob*, por otro lado, quiere asegurarse de que *Alice* no puede cambiar su predicción después de que el hecho se haya producido



- Se puede solucionar usando tanto criptografía simétrica como funciones hash

- Usando criptografía simétrica:
 - ① Bob genera aleatoriamente una cadena R de bits y se la envía a Alice
 $Bob \rightarrow Alice: R$
 - ② Alice crea un mensaje con el bit b a dejar en compromiso a Bob, y la cadena aleatoria de Bob. Lo cifra con una clave aleatoria K , y envía el resultado a Bob
 $Alice \rightarrow Bob: E_K(R, b)$
 - Alice ha realizado el compromiso (apuesta), y Bob no puede descifrar el mensaje así que no puede conocer el bit.
 - Cuando le llega a Alice la hora de revelar el bit (apuesta) el protocolo continúa
 - ③ Alice envía a Bob la clave K
 $Alice \rightarrow Bob: K$
 - ④ Bob descifra el mensaje para obtener el bit, y chequea su cadena aleatoria para verificar la validez del bit
 $Bob: D_K(E_K(R, b))$

- Usando funciones hash:

- ① *Alice* genera aleatoriamente dos cadenas de bits R_1 y R_2 y crea un mensaje que consta de las dos cadenas aleatorias y el bit de compromiso

Alice: (R_1, R_2, b)

- ② A continuación, computa la función hash de ese mensaje y envía el resultado a *Bob* junto a una de las cadenas aleatorias

Alice→*Bob*: $H(R_1, R_2, b), R_1$

- Esta transmisión es la evidencia del compromiso. La función hash en el paso 3 previene que *Bob* pueda invertir la función y determinar el bit
- Cuando *Alice* tiene que revelar el bit, el protocolo continúa

- ③ *Alice* envía a *Bob* el mensaje original

Alice→*Bob*: R_1, R_2, b

- ④ *Bob* computa la función hash del mensaje y compara ésta y R_1 con el valor y la cadena aleatoria recibida en el paso 3. Si coinciden el bit es válido

Bob: $H(R_1, R_2, b)$

Protocolos de lanzamiento de moneda

- Supongamos que *Alice* y *Bob* desean hacer un “cara o cruz” sin encontrarse presencialmente
- Uno de ellos hace el lanzamiento de la moneda, pero el otro no lo ve
- En general, necesitamos un protocolo con las siguientes propiedades:
 - *Alice* debe lanzar la moneda antes de que *Bob* se pronuncie
 - *Alice* no debe ser capaz de re-lanzar la moneda después del pronunciamiento de *Bob*
 - *Bob* no debe saber de qué lado cayó la moneda antes de pronunciarse
- Podemos solucionar este problema utilizando un protocolo de compromiso de bit
- También se puede solucionar utilizando criptografía de clave pública, como se ve a continuación



- Utilización de criptografía de clave pública:
 - Ha de cumplirse el requisito: $D_{K1}(E_{K2}(E_{K1}(M))) = E_{K2}(M)$
- ① *Alice* y *Bob* generan, cada uno, un par <clave pública, clave privada>
- Alice*: K_{Apu}, K_{Apr}
Bob: K_{Bpu}, K_{Bpr}
- ② *Alice* genera dos mensajes, uno indicando “cara” y otro “cruz”.
 Estos mensajes contienen además alguna cadena aleatoria, para verificar posteriormente su autenticidad.

Alice cifra ambos mensajes con su clave pública y los envía a *Bob* en un orden aleatorio

Alice → *Bob*: $E_{Apu}(M_1), E_{Apu}(M_2)$

- ③ *Bob*, que no puede leer los mensajes, elige uno, lo cifra con su clave pública y se lo devuelve a *Alice*

Bob → *Alice*: $E_{Bpu}(E_{Apu}(M))$ donde M es o bien M_1 o bien M_2

- ④ *Alice*, que no puede leer el mensaje que *Bob* le ha devuelto, lo descifra con su clave privada y se lo devuelve a *Bob*

Alice: $D_{Apr}(E_{Bpu}(E_{Apu}(M))) \xleftarrow{\text{ }} D_{K1}(E_{K2}(E_{K1}(M))) = E_{K2}(M)$

Alice → *Bob*: $E_{Bpu}(M)$

- ⑤ *Bob* descifra el mensaje con su clave privada para averiguar el lanzamiento de la moneda, y envía el mensaje descifrado a *Alice*

Bob: $D_{Bpr}(E_{Bpu}(M))$

Bob → *Alice*: M

- ⑥ *Alice* lee el resultado del lanzamiento y verifica que la cadena aleatoria es correcta
- Cualquiera de las partes puede detectar inmediatamente si el otro miente, y no hace falta una tercera parte para completar el protocolo

Protocolo de póker mental

- Similar al protocolo de lanzamiento de monedas en el que se utiliza criptografía de clave pública

- ① *Alice* y *Bob* generan, cada uno, un par clave pública/clave privada

Alice: K_{Apu}, K_{Apr}

Bob: K_{Bpu}, K_{Bpr}

- ② *Alice* genera 52 mensajes, uno para cada carta de la baraja.

Estos mensajes contienen además alguna cadena aleatoria, para verificar posteriormente su autenticidad.

Alice cifra todos los mensajes con su clave pública y los envía a *Bob* en un orden aleatorio

$Alice \rightarrow Bob: E_{Apu}(M_i)$ donde $i = 1..52$



- ③ *Bob*, que no puede leer ninguno de los mensajes, elige aleatoriamente cinco de ellos; los cifra con su clave pública y se los devuelve a *Alice*

$$Bob: \quad E_{Bpu}(E_{Apu}(M_j^B))$$

donde M_j^B ($j = 1..5$) son las cinco cartas que *Bob* ha elegido

- ④ *Alice*, que no puede leer los mensajes que *Bob* le ha devuelto, los descifra con su clave privada y se los devuelve a *Bob*

$$Alice: \quad D_{Apr}(E_{Bpu}(E_{Apu}(M_j^B)))$$

$$Alice \rightarrow Bob: \quad E_{Bpu}(M_j^B)$$

- ⑤ *Bob* descifra los mensajes con su clave privada para averiguar su mano

$$Bob: \quad D_{Bpr}(E_{Bpu}(M_j^B)) = M_j^B$$

- ⑥ De los 47 mensajes $E_{Apu}(M_i)$ que restan de los 52 que recibió en el paso 2, *Bob* elige aleatoriamente cinco de ellos, $E_{Apu}(M_k^A)$, y se los envía a *Alice*

Bob → *Alice*: M_k^A ($k = 1..5$)

donde M_k^A ($j = 1..5$) son las cinco cartas de la mano de *Alice*

- ⑦ *Alice* descifra esos cinco mensajes y ve cuáles son las cartas que le han tocado

Alice: $D_{Apu}(E_{Apu}(M_k^A)) = M_k^A$

Referencias bibliográficas

Bibliografía básica

- “Applied Cryptography: Protocols, Algorithms, and Source Code in C”
Bruce Schneier
Wiley, 1996 (2^a edición)
- RFC 5280, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", 2008
- ITU-T, “X.509 : Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks”, 2012
- “*Cryptography and Network Security: Principles and Practice*”
William Stallings
Prentice Hall, 2010 (5^a edición)
- “*Computer Security: Principles and Practice*”
William Stallings and Lawrie Brown
Prentice-Hall, 2011 (2^a edición)

Referencias

- Guía de Referencia del DNIe con NFC, 2015
- RFC 6818, “Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile”, 2013
- RFC 6960, “X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP”, 2013
- RFC 2828, “Internet Security Glossary”, 2000

Apéndice - Ejemplos de software de PKI



PKI Certificate Authority Configuration Mode Commands

- ☐ Cisco Wide Area Application Services Command Reference (Software Version 4.3.1)
- ☐ Command Summary by Mode
- ☐ Preface
- ☐ Using the WAAS Command-Line Interface
- ☐ Cisco WAAS Software Command Summary
- ☐ CLI Commands
- ☐ EXEC Mode Commands
- ☐ Global Configuration Mode Commands
- ☐ Interface Configuration Mode Commands
- ☐ Standard ACL Configuration Mode Commands
- ☐ Extended ACL Configuration Mode Commands
- ☐ Preposition Configuration Mode Commands
- ☐ Virtual Blade Configuration Mode Commands
- ☐ **PKI Certificate Authority Configuration Mode Commands**
- ☐ PKI Global Settings Configuration Mode Commands
- ☐ SSL Accelerated Service Configuration Mode Commands
- ☐ SSL Cipher List Configuration Mode Commands
- ☐ SSL Global Service Configuration Mode Commands
- ☐ SSL Host Peering Service Configuration Mode Commands
- ☐ SSL Management Service Configuration Mode Commands
- ☐ Acronyms and Abbreviations

Downloads: [This chapter](#) (PDF - 64.0 KB) [The complete book](#) (PDF - 7.96 MB) | [Feedback](#)

Table Of Contents

[PKI Certificate Authority Configuration Mode Commands](#)

- [\(config-ca\) ca-certificate](#)
- [\(config-ca\) description](#)
- [\(config-ca\) revocation-check](#)

PKI Certificate Authority Configuration Mode Commands

To configure public key infrastructure (PKI) encryption certificate authorities on a WAAS device, use the **crypto pki ca** global configuration command. To delete a PKI encryption certificate authority, use the **no** form of the command.

crypto pki ca certificate_authority_name

no crypto pki ca certificate_authority_name

Syntax Description

| | |
|-----------------------------------|--|
| certificate_authority_name | The name of the certificate authority (CA). The CA name may contain up to 64 characters. |
|-----------------------------------|--|

Defaults

No default behavior or values.

Command Modes

global configuration

Device Modes

application-accelerator

central-manager

Usage Guidelines

Use the command to add and configure a certificate authority. This command initiates the certificate authority configuration mode, indicated by the **(config-ca)** prompt.

Within certificate authority configuration mode, you can use the various commands (**ca-certificate**, **description**, **revocation check**, and so on) to define an encryption certificate authority. To return to global configuration mode, enter **exit** at the certificate authority configuration mode prompt.

Examples

The following example shows how to create or edit a certificate authority named **mycertauth**. If the certificate authority is already established on the WAAS device, the **crypto pki ca** command edits it. If the certificate authority does not exist, the **crypto pki ca** command creates it.

```
WAE(config)# crypto pki ca mycertauth
```

Entrust

[>Products & Services](#)[>SSL](#)[>Solutions](#)[>Partners](#)[>Resources](#)[>Support](#)[>About](#)[>Contact](#)

Entrust Authority PKI

An Integrated Security Infrastructure for Encryption, Digital Signatures & Certificate Authentication

A PKI establishes and maintains a trustworthy networking environment by providing key and certificate management services that enable encryption and digital signature capabilities across applications — all in a manner that is transparent and easy to use.

Entrust's first [public key infrastructure](#) — the world's first commercially available PKI — was released in 1994. Now in its eighth edition, the Entrust Authority public key infrastructure product portfolio is the industry's most relied-upon PKI solution.

By managing the full lifecycles of digital certificate-based identities, Entrust Authority PKI enables encryption, digital signature and certificate authentication capabilities to be consistently and transparently applied across a broad range of applications and platforms.

[PDF](#)[Data Sheet](#)[> Download](#)

Watch Our Webcast

How can a certificate improve security?

[> View Now](#)

Is Malware Already Inside Your Organization?

[> View](#)

Try Entrust Managed Service
PKI FREE for 60 days



Resources

[> Data Sheets](#)



PKI segura

Infraestructura PKI segura con cifrado y autenticación

Las vulnerabilidades de infraestructura representan un riesgo significativo para las organizaciones que dependen solamente de public key infrastructure (PKI) para salvaguardar las aplicaciones digitales. La clave está en que las organizaciones construyan una base criptográfica segura a fin de aprovechar por completo los beneficios y las oportunidades que permiten las aplicaciones digitales, al mismo tiempo que salvaguardan sistemáticamente la infraestructura y confianza en los entornos de PKI.

[CONTACT US →](#)

Protección constante

- Infraestructura de medición avanzada
- Firmas digitales
- Seguridad del sistema de nombres de dominio (DNS)
- Administración de la identidad

Protección en cada punto

- Hardware security modules (HSM)
- Criptografía como un servicio
- Autenticación USB en PKI
- Tarjetas inteligentes para PKI
- Autenticador virtual eToken

Recursos

- Resumen de la solución: Hardware security modules para Smart Grid
- Resumen de la solución: HSM para DNSSEC
- Resumen de la solución: eBoarding
- Resumen de la solución: HSM para facturación electrónica
- Caso de éxito: SURFNet selecciona HSM para asegurar material DNSSEC

Protección total para infraestructuras de clave pública crítica

Con las soluciones de autenticación y cifrado de SafeNet, las organizaciones pueden aprovechar por completo los beneficios y las oportunidades que permiten las aplicaciones digitales, al mismo tiempo que salvaguardan sistemáticamente la infraestructura y confianza en los entornos de PKI. Al implementar las soluciones de SafeNet, las organizaciones pueden disfrutar de los siguientes beneficios:

PKI Needs a Layered Security Approach

Mark Yakabuski,
VP of Product Management

[PLAY PODCAST →](#)

Key Management and Payments Security Blog - Thales e-Security

What is a PKI and why is it important? Part 1

By Richard Moulds

Businesses are becoming ever more dependent on digital information and electronic transactions, and as a result face stringent data privacy compliance challenges and data security regulations. With the enterprise increasingly under threat of cyber attacks and malicious insiders, business applications and networks are now dependent on the use of digital credentials to control how users and entities access sensitive data and critical system resources.

Public key infrastructures (PKIs) are necessary to help ascertain the identity of different people, devices, and services. In a nutshell, PKIs go way beyond the use of user IDs and passwords, employing cryptographic technologies such as [digital signatures](#) and digital certificates to create unique credentials that can be validated beyond reasonable doubt and on a mass scale.

PKI technology is already used more widely than you might think. It is a cornerstone of how data is encrypted as it is passed over the internet using SSL/TLS – without it, e-commerce wouldn't be practical. PKI is used to digitally sign documents transactions,

Search Blog

SEARCH

Archive

2015

2014

2013

2012

2011

2010

2009

Syndication

[Thales e-Security Blogs](#)

Hacer el cambio a

DigiCert

La Autoridad de Certificación de Mayor Crecimiento

» Los certificados suelen emitirse en menos de una hora

» Herramientas de punta para una fácil gestión de sus certificados

[COMPARAR CERTIFICADOS](#)

Certificados SSL

Certificados SSL de 2048-Bit con Cifrado Fuerte y Autenticación

[COMPRAR](#) [APRENDER](#)

SSL Plus™

As low as
\$139/yr[COMPRAR](#) [APRENDER](#)

SSL Plus con EV

[COMPRAR](#) [APRENDER](#)

Certificado UC

[COMPRAR](#) [APRENDER](#)

Multi-Dominio EV

[COMPRAR](#) [APRENDER](#)

Wildcard Plus™

Renovar

Renovar sus Certificados Existentes

[RENOVAR](#)

Firma de Código

Servicios de Autenticación Digital

[COMPRAR](#)

¿Ayuda Elijiendo?

Comparar Certificados y Precios

[AYÚDAME ELEJIR](#)

Managed PKI

Gestión de Certificados a Nivel Empresarial

[APRENDER MÁS](#)

Recent Posts

- [OpenXPKI Workshop - Slides](#)
- [OpenXPKI Release Event + Workshop](#)
- [New website goes online](#)

Links

- [Quickstart](#)
- [Documentation](#)
- [Demo Site](#)
- [Users Mailinglist](#)

The OpenXPKI Project

The OpenXPKI project has the vision to publish a software stack that provides all necessary components to manage keys and certificates primarily based on the X509v3 cryptography standard.

Project Status

As of Summer 2015 the core and new WebUI are quite stable and we are doing some final polishing. We plan to release the one dot zero on the 10th anniversary of the project on October 20th. We will have a free workshop and release party, [see this post for details](#).

There are already several installations used in production and we encourage everybody to give it a try and provide feedback.

Core Features

- WebUI compatible with all major browsers
- Ready-to-run example config included
- Support for SCEP (Simple Certificate Enrollment Protocol)
- Easy adjustment of workflows to personal needs
- Run multiple separate CAs with a single installation, automated rollover of CA generations
- Can use Hardware Security Modules (e. g. Thales HSMs) for crypto operations
- Issue certificates with public trusted CAs (e. g. SwissSign, Comodo, VeriSign)
- Based on OpenSSL and Perl, runs on most *nix platforms
- 100% Open Source, commercial support available

Resources

- Development and issue tracking using [github](#)
- Mailing lists for [users](#) and [developer](#)
- Documentation (some) at [ReadTheDocs](#)
- Package repository at <http://packages.openxpki.org/> (Debian Wheezy only, yet. SLES and Ubuntu coming soon).

Home**CA**

- Política CP/CPS
- Certificado CA
- Certificados válidos
- CRLs
- Información contacto
- Coordinación

RAs

- Elección RA
- Información RAs
 - Mapa RAs
 - Alta RA
- Política RA
- Auditoría datos

Documentación**Software****Ayuda****Noticias** **Dirección envío CUSOs**

pkIRISGrid - RedIRIS
Edificio CICA
Avda. Reina Mercedes
s/n
41012, Sevilla

**Acreditación
EUGridPMA**[◀ pkIRISGrid](#)

pkIRISGrid

PKI para la e-Ciencia española



Accediendo con IPv4: 150.214.58.137

pkIRISGrid CA, es la infraestructura de clave pública usada en la iniciativa nacional de Grid [IRISGrid](#).

Fue acreditada, el 25 de enero de 2006, por la [EUGridPMA](#), organización internacional dedicada a la coordinación de la red de confianza entre las PKIs que dan servicio a la e-Ciencia europea.

La EUGridPMA establece unos requisitos mínimos exigibles a los proveedores de identidad, para temas relacionados con Grid, que permite crear una red común de confianza aplicable a la autenticación de las entidades finales para el acceso inter-organizacional a los recursos distribuidos en el Grid. La EUGridPMA asegura que los certificados emitidos por las PKIs acreditadas cumplen con los requisitos exigidos para el establecimiento de esta red de confianza.

Desde nuestra acreditación por la EUGridPMA los certificados emitidos por la pkIRISGrid son aceptados por todas las organizaciones afiliadas a la EUGridPMA y a las de otras PMAs afiliadas al [IGTF](#).

Actualidad

- 20150907 - Reunión: [35th EUGridPMA meeting, Amsterdam, The Netherlands](#)
- 20150511 - Reunión: [34th EUGridPMA meeting, Copenhagen, Denmark](#)
- 20150507 - Exportación automática de CSRs tras ser validadas
- 20150428 - Baja RA 17 - CICA
- 20150413 - Baja RA 48 - IC3
- 20150310 - Procedimiento de [baja de una RA](#)
- 20150303 - [Entra en producción el nuevo certificado raíz de pkIRISGrid](#)
- 20150204 - [Emisión de certificados con 365 días de validez](#)
- 20150112 - Reunión: [33rd EUGridPMA meeting, Berlin, Germany](#)
- 20141201 - [Renovación certificado raíz \(hasta 2025\)](#)
- 20141114 - Nueva RA 50 - UBU

Elección de la Autoridad de Registro

Tenemos disponible un [listado de las autoridades de registro](#) junto con los espacios de nombre que gestionan. También puede verlas distribuidas en el [mapa de RAs de pkIRISGrid CA de España](#)

Si no encuentra una autoridad de registro con la que desee realizar las gestiones para la obtención de certificados puede solicitar la [creación de su propia RA](#).