

SEGURIDAD DE LA INFORMACIÓN

TEMA 3 - PARTE 1

**ESQUEMAS, PROTOCOLOS Y MECANISMOS
DE SOPORTE
(A LA SEGURIDAD DE APLICACIONES Y DE REDES)**

Índice del tema

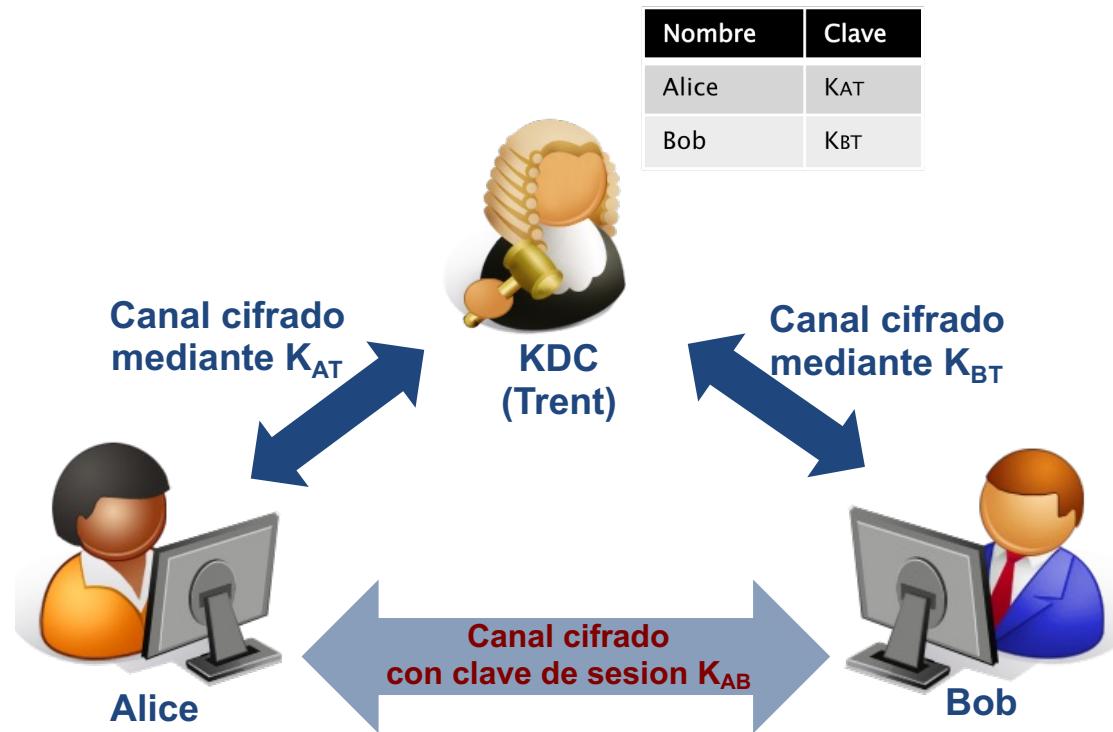
- **Gestión de las “claves”**
 - Protocolos de distribución de claves simétricas
 - Mecanismos e infraestructuras de administración de claves públicas
 - El caso del DNI-e
 - Mecanismo de Single Sign-On para Autenticación
- **Mecanismos de Control de Acceso**
 - DAC, MAC, RBAC, ABAC
 - Otros
- **Protocolos criptográficos avanzados**
 - Protocolos de división y compartición de secretos
 - Protocolos de compromiso de bit (bit-commitment)
 - Protocolos de lanzamiento de moneda
 - Protocolo de póker mental
- **Referencias bibliográficas**

Gestión de las Claves

Protocolos de distribución de claves simétricas

- Hay escenarios donde la utilización de la criptografía de clave pública (o asimétrica) para el intercambio de una clave de sesión K_{AB} puede ser NO conveniente
 - P. ej. redes LAN grandes sin conexión a la red WAN. En este caso, *Alice* y *Bob* van a seguir necesitando de alguna solución que les permitan, aún estando geográficamente (moderadamente) lejanos, decidir esa clave de sesión K_{AB}
- En estos casos, la solución pasa por algún protocolo de **distribución centralizada de claves** para los usuarios del sistema
 - consiste en hacer uso de una tercera parte confiable (TTP – Trusted Third Party), que en este caso se denomina **Centro de Distribución de Claves** (o **KDC** – *Key Distribution Center*)
- Existen diferentes protocolos que proporcionan una solución para ese escenario:
 - Yahalom, Needam-Schroeder, Otway-Rees, Kerberos, ...

- En el modus operandi general de este tipo de protocolos, cada usuario del sistema comparte, de inicio, una **clave secreta** con el KDC
 - mediante algún proceso de registro o inscripción del usuario ante el KDC



KDC: modelos y protocolos

- El uso de un KDC se basa en el uso de claves jerárquicas, de manera que se requieren al menos dos niveles de claves
- La mayoría de las técnicas de distribución de claves se adaptan a situaciones, escenarios y aplicaciones específicas
 - de manera que son diversos los esquemas que se integran a entornos locales donde todos los usuarios tienen acceso a un **servidor común de confianza**
- Hay muchos modelos de distribución de claves:
 - Simples
 - Genéricos, y dentro de los genéricos nos podemos encontrar:
 - Los modelos **PULL** o modelos **PUSH**, o sus combinaciones

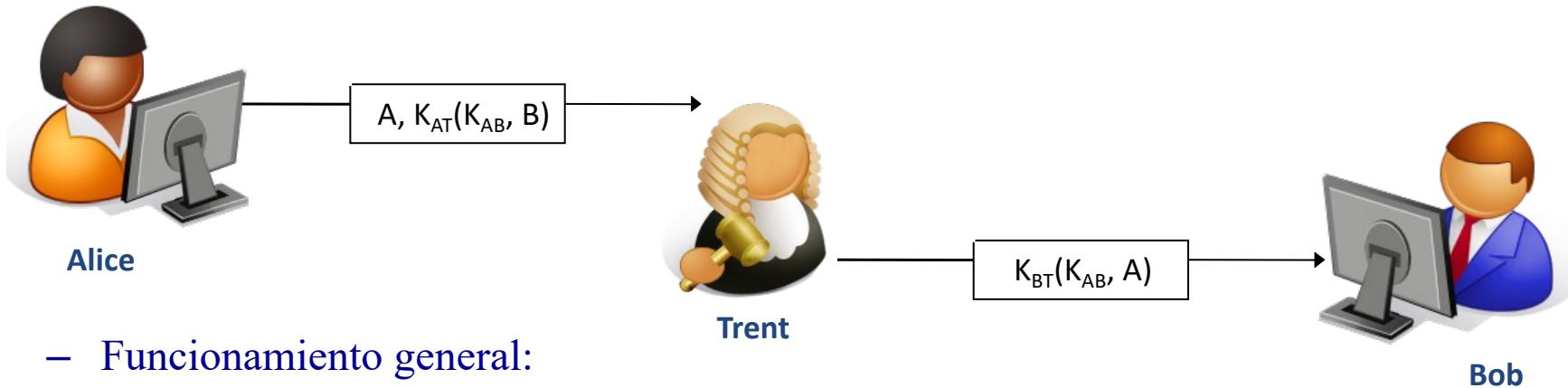
Canal cifrado
mediante K_{AT}, K_{BT}



Canal cifrado
mediante K_{AB}

KDC: modelos y protocolos - Modelo Simple

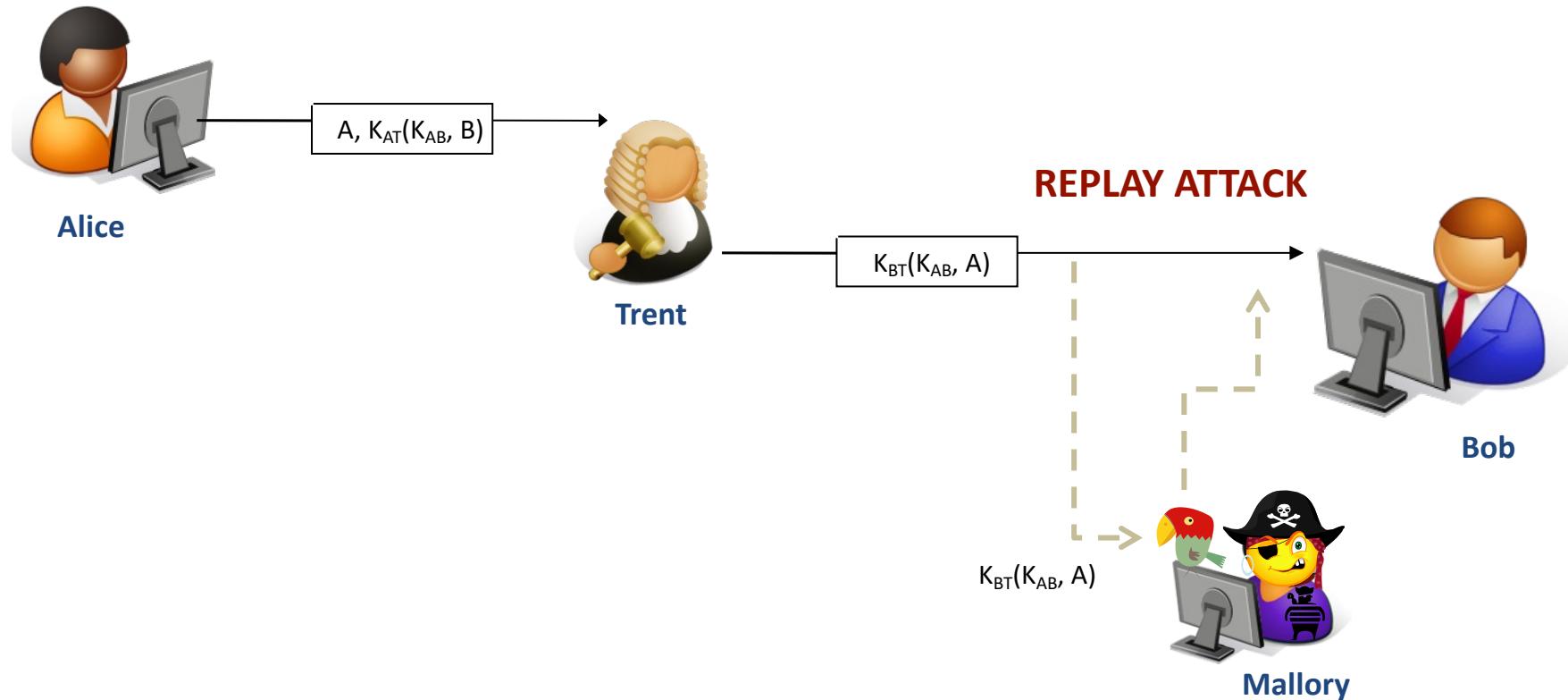
- El protocolo “**La Rana de la Boca Grande**” es un ejemplo de modelo simple para la distribución de claves:



- Funcionamiento general:
 - **Paso 1:** A genera una clave de sesión K_{AB} y se la envía al KDC
 - el mensaje incluye la identidad de A, la identidad de B y la clave de sesión cifrada con el K_{AT}
 - **Paso 2:** el KDC verifica la identidad de A y reenvía la K_{AB} a B cifrado con K_{BT}
 - **Paso 3:** B verifica la identidad de KDC por la K_{BT} y obtiene la clave de sesión
- Como se puede observar existe **validación de identidad**:
 - Las claves con el KDC son secretas, por lo que nadie más habría sido capaz de cifrar la clave secreta K_{AB} , además existe autenticación de cada parte involucrada

KDC: modelos y protocolos - Modelo Simple

- Sin embargo, existe un **fallo de seguridad**:



Si Mallory intercepta el canal y captura todos los mensajes de KDC a B, entonces es posible que Mallory cause un **ataque de repetición (ataque replay)**, y, por consiguiente, un ataque de Denegación de Servicio (DoS) sin necesidad de que éste derive K_{AB} o K_{BT}

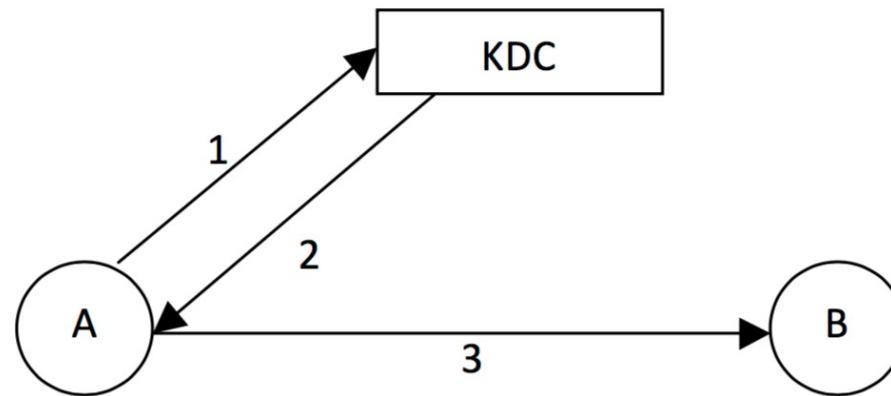
KDC: modelos y protocolos - Modelo Simple

- Para resolver el problema anterior, se pueden hacer uso de alguno de los mecanismos existentes:
 - **Marca de tiempo:** incluir en cada mensaje una marca de tiempo (un sello de tiempo) de forma que pueda descartar mensajes obsoletos
 - Problema: los relojes nunca están perfectamente sincronizados en toda una red
 - **Nonce / único:** incluir un número aleatorio único para cada mensaje enviado, de forma que cada parte de la comunicación debe siempre recordar todos los únicos enviados o recibidos, y rechazar cualquier mensaje que contenga un único previamente usado
 - Problema: si una de las partes pierde la lista de nonce / únicos, es susceptible a ataques replays
 - **Combinación de ambas** estrategias para limitar el tiempo que pueden recordarse los únicos, pero el protocolo se volverá más complicado

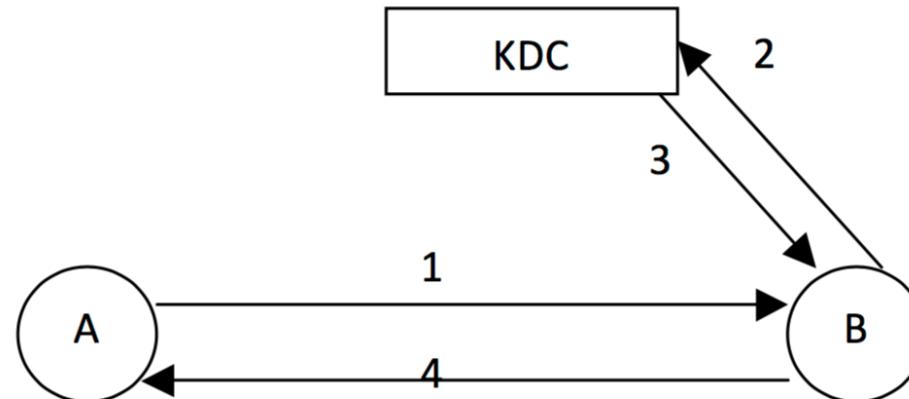
Freshnesses

KDC: modelos y protocolos - Modelos Genéricos

- Modelo **PULL** para la distribución de claves:

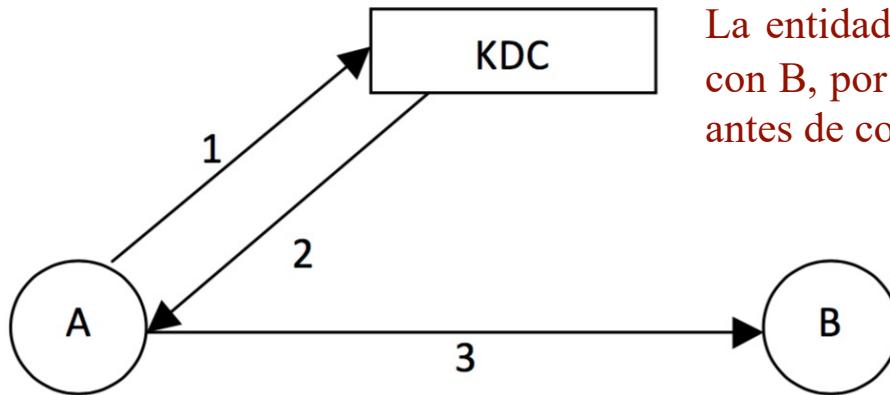


- Modelo **PUSH** para la distribución de claves:



KDC: modelos y protocolos

- Modelo **PULL** para la distribución de claves:



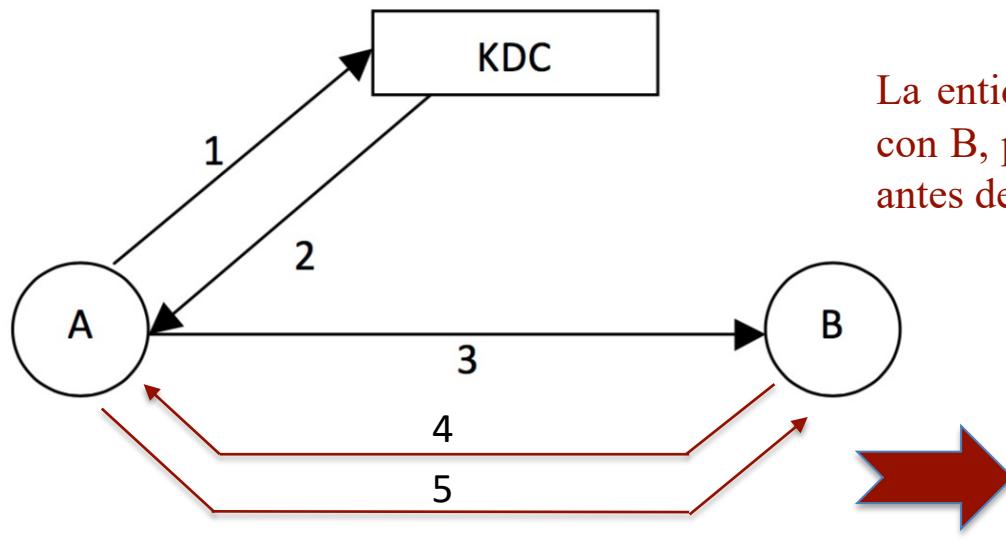
La entidad A desea tener comunicación segura con B, por lo que contacta con el KDC primero antes de comunicarse con B

- Funcionamiento general:

- **Paso 1:** A solicita una clave de sesión K_{AB} al KDC
 - el mensaje incluye la identidad de A, la identidad de B y un valor **N1 (sello de tiempo, valor aleatorio)**
- **Paso 2:** El KDC le contesta a A con un mensaje cifrado mediante la clave maestra K_{AT} , de manera que solamente A puede leer dicho mensaje y con ello, sabe, además, que el KDC es el único que pudo haberlo generado
 - el mensaje contiene la clave K_{AB} , N1, y un mensaje cifrado para B con el K_{AB}
- **Paso 3:** A obtiene la información recibida y reenvía el mensaje a B para que pueda obtener el K_{AB} también

KDC: modelos y protocolos

- Modelo **PULL** para la distribución de claves:



La entidad A desea tener comunicación segura con B, por lo que contacta con el KDC primero antes de comunicarse con B

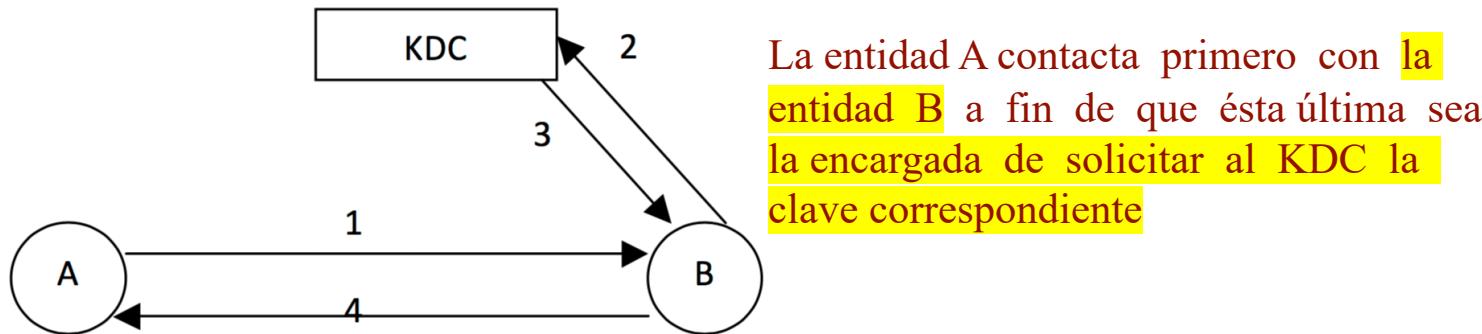
**desafío-respuesta
“challenge-response”**

- Funcionamiento general:

- **Paso 4:** B utiliza la K_{AB} para cifrar un valor único $N2$ y se lo envía a A
- **Paso 5:** A recibe el valor $N2$, le aplica una transformación $f(N2)$, lo cifra con K_{AB} y lo transmite a B

KDC: modelos y protocolos

- Modelo **PUSH** para la distribución de claves:

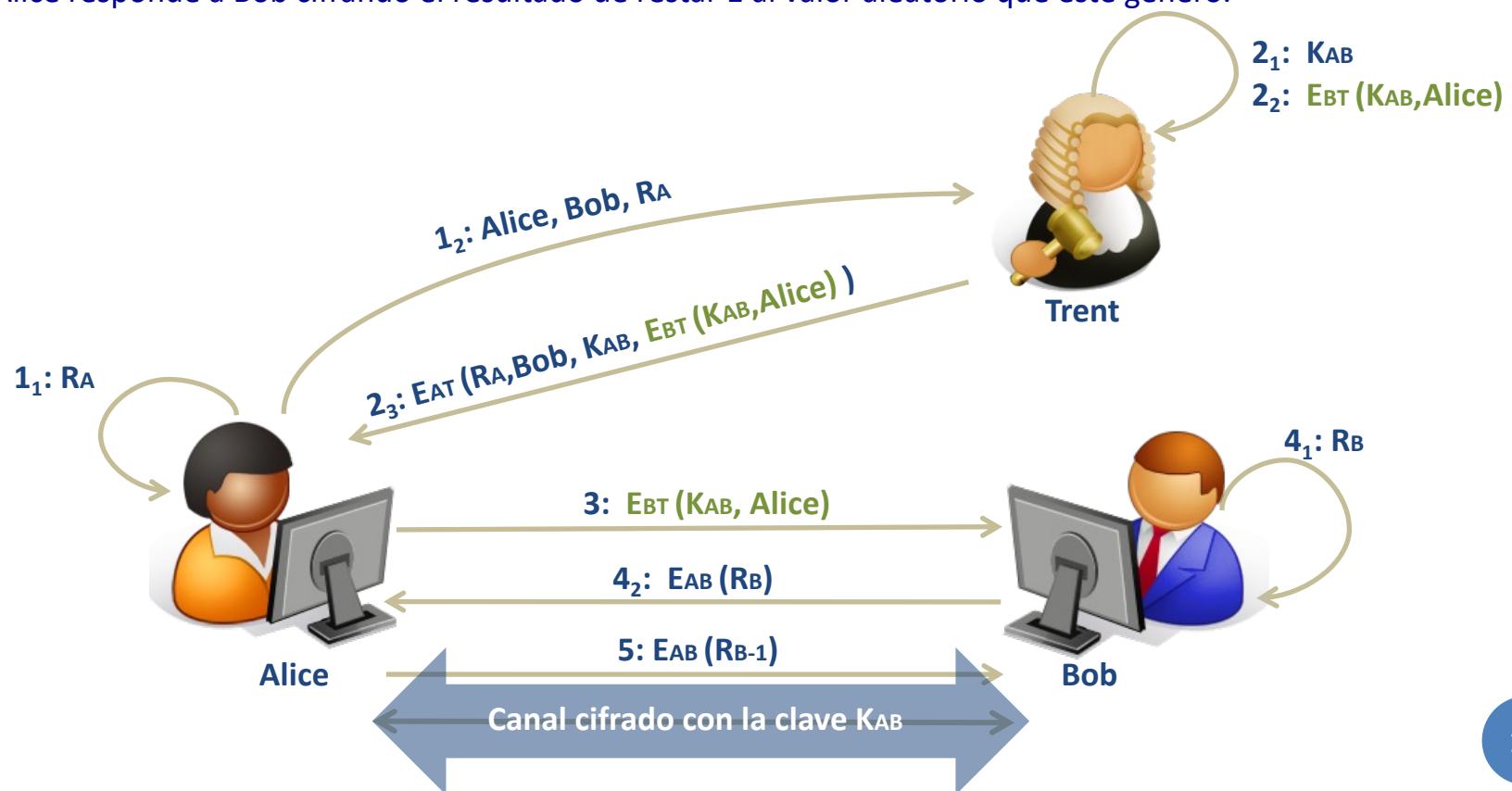


- Funcionamiento general:

- **Paso 1:** A solicita conexión segura con B a B
 - le manda, como mínimo, su identidad, la identidad de B y un nonce
- **Paso 2:** B reenvía dicha solicitud a KDC para que éste genere la K_{AB}
- **Paso 3:** KDC verifica las identidades y el *freshnesses* de los mensajes, genera la K_{AB} , y dicha información se reenvía a B cifrada con la correspondiente K_{xT}
- **Paso 4:** B reenvía dicha solicitud a A para que obtenga K_{AB}
- **Paso 5 (opcional):** A establece un desafío y respuesta

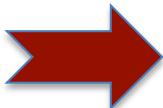
• Protocolo de Needham-Schroeder

- 1: Alice genera el valor aleatorio (único) $R_A <1_1>$, y se lo envía a Trent $<1_2>$.
- 2: Trent genera la clave de sesión $K_{AB} <2_1>$, y se la envía a Alice, junto a un mensaje para Bob $<2_3>$.
- 3: Alice envía a Bob el mensaje que ella ha recibido de Trent.
- 4: Bob genera valor aleatorio R_B y se lo envía a Alice usando la clave K_{AB} (proceso de “challenge-response”).
- 5: Alice responde a Bob cifrando el resultado de restar 1 al valor aleatorio que éste generó.



Needham-Schroeder

- Diseño formalizado:

- $A \rightarrow S : A, B, N_A$  **freshness** - [aunque sin cifrar ☺]
 - $S \rightarrow A : \{N_A, B, K_{A,B}, \{K_{A,B}, A\}_{K_{B,S}}\}_{K_{A,S}}$
 - $A \rightarrow B : \{K_{A,B}, A\}_{K_{B,S}}$
 - $B \rightarrow A : \{N_B\}_{K_{A,B}}$
 - $A \rightarrow B : \{N_B - 1\}_{K_{A,B}}$
- 
- desafío-respuesta**
-
- “challenge-response”**

- [aunque con una función
muy obvia $f(n) = n - 1$ ☺]

N_A : nonce/valor aleatorio

S: KDC

A: Alice

B: Bob

$K_{A,B}$: clave secreta compartida

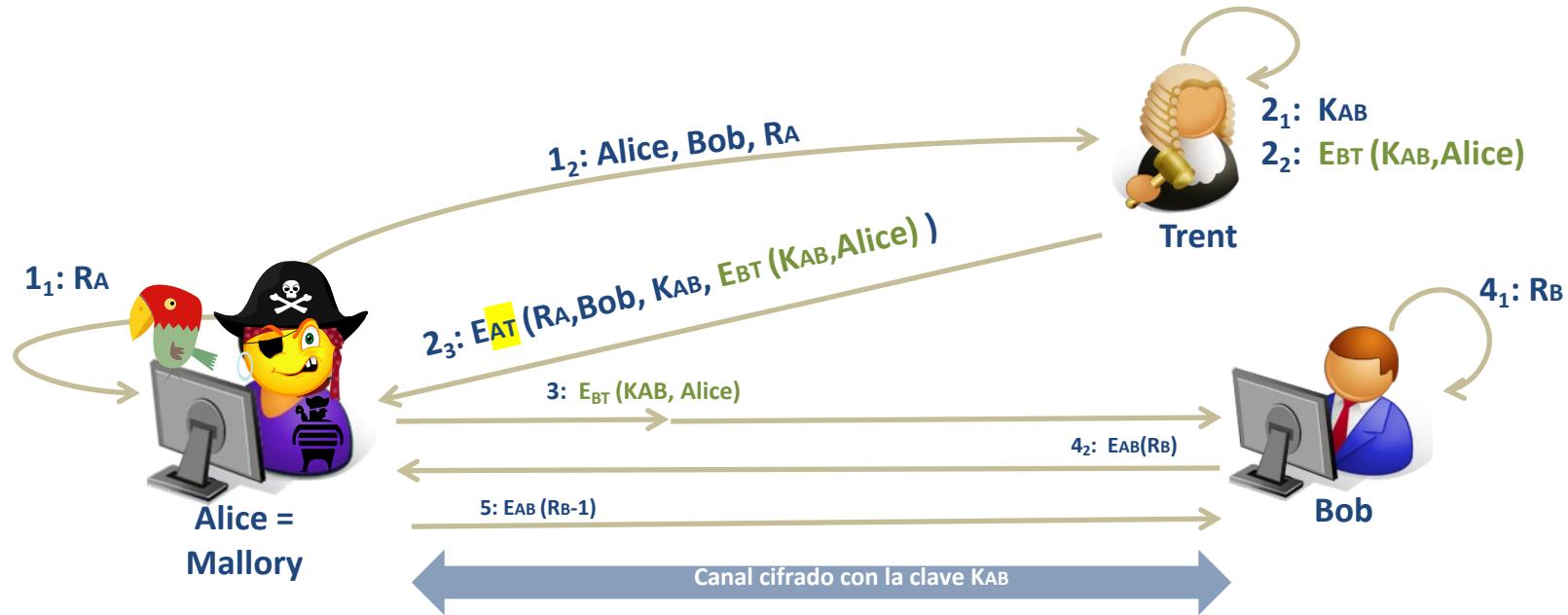
- Este protocolo tiene más **fallos de seguridad**, que fueron descubiertos años después de su funcionamiento



10 minutos – tic, tac, tic, tac ...

- Este protocolo tiene más **fallos de seguridad**, que fueron descubiertos años después de su funcionamiento
 - **Ataque 1:** Mallory, el atacante, puede suplantar la identidad de Alice si éste consigue derivar la clave K_{AT}
... obvio
 - **Ataque 2:** Mallory puede suplantar la identidad de Bob si éste consigue derivar la clave K_{BT}
... obvio
 - **Ataque 3:** Mallory puede producir un ataque de DoS debido a un **ataque de repetición**, especialmente en las últimas fases del protocolo
... ¿ dónde ?

• Ataque 1

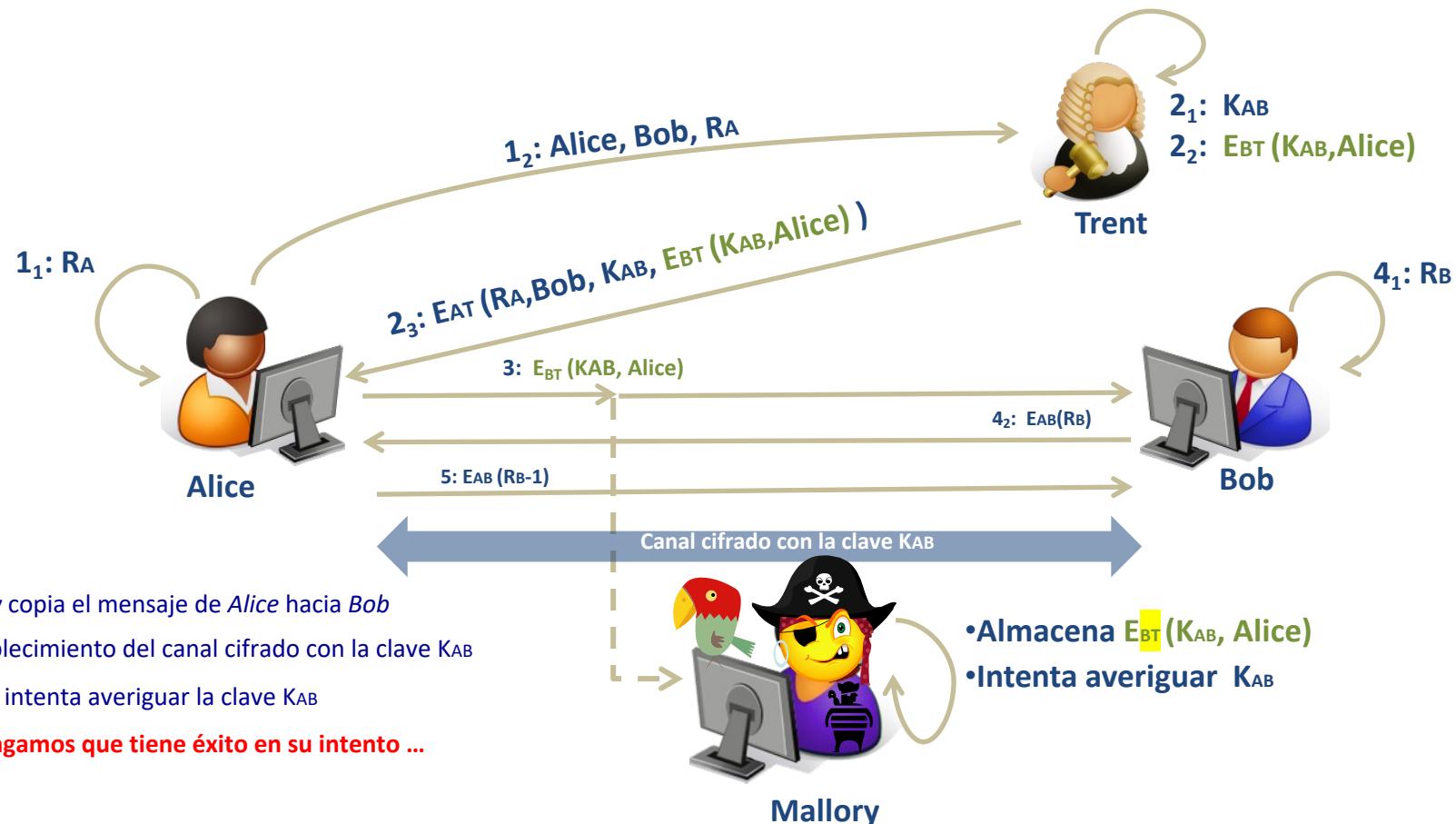


0: Mallory copia cualquier mensaje de Alice hacia Trent en el pasado y deriva la clave K_{AT} . A partir de aquí, todos los mensajes quedan comprometidos

supongamos que tiene éxito en su intento ...

- Almacena $E_{BT}(K_{AB}, Alice)$
- Intenta averiguar K_{AB}

- **Ataque 2**



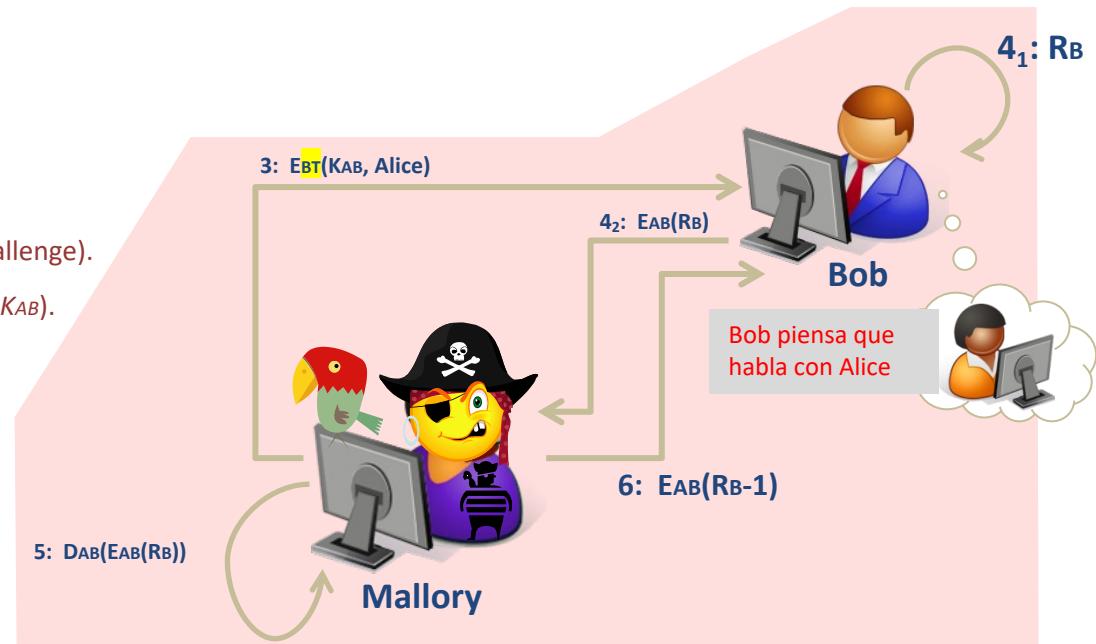
- continuación...

3: *Mallory* envía el mensaje $E_B(T(K_{AB}, \text{Alice}))$ a *Bob*.

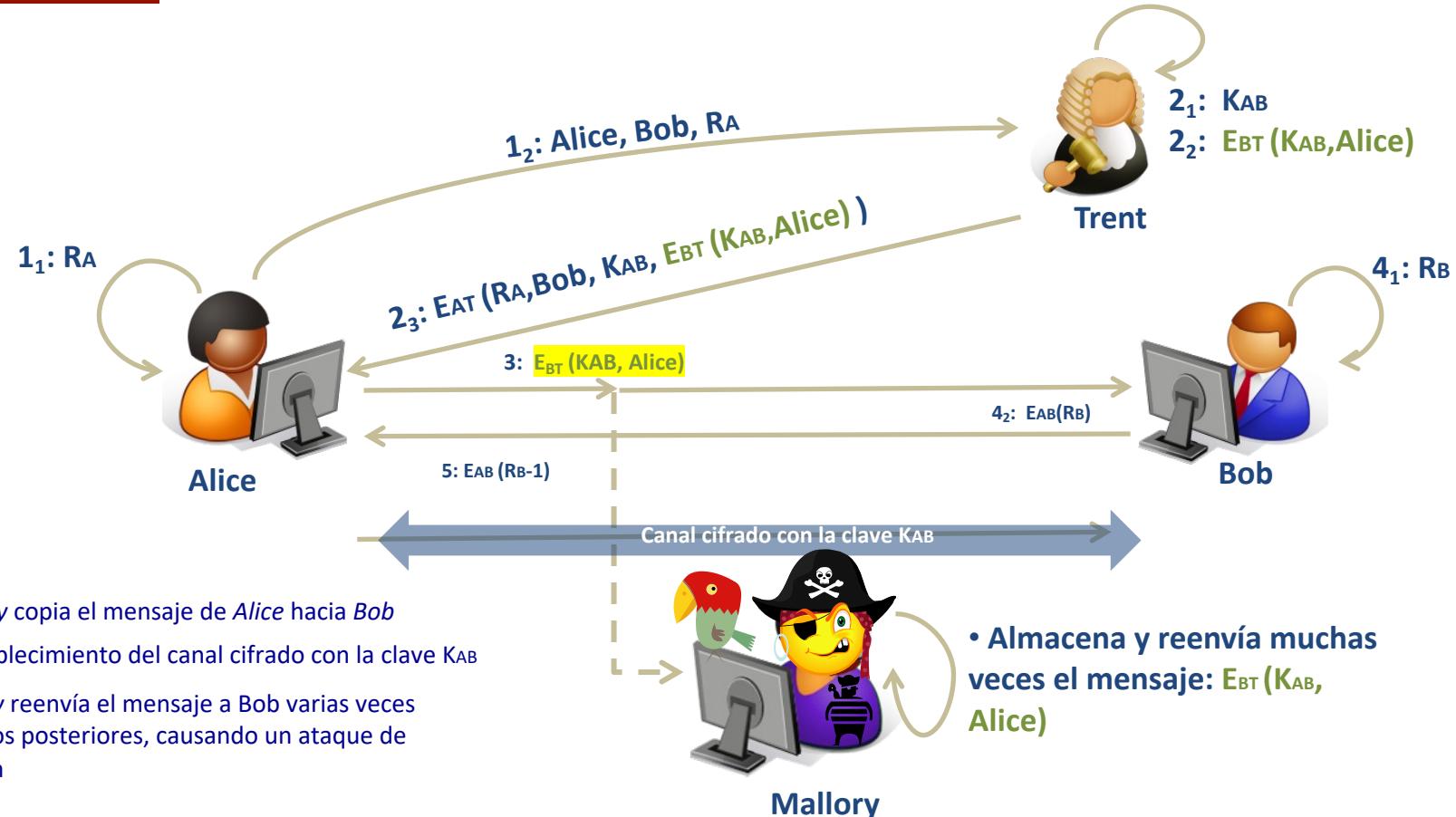
4: *Bob* responde a *Mallory* con un valor aleatorio (challenge).

5: *Mallory* descifra el valor aleatorio (porque conoce K_{AB}).

6: *Mallory* responde al challenge de *Bob*, y *Bob* piensa que habla con *Alice*.



• Ataque 3



- Metiendo en el mensaje 3 un nonce:

1. $A \rightarrow S : A, B, N_A$
2. $S \rightarrow A : \{N_A, B, K_{A,B}, \{K_{A,B}, A\}_{K_{B,S}}\}_{K_{A,S}}$

Problema: la frescura de los mensajes solo se encuentra en los mensajes 1 y 2, pero no en el resto de mensajes

3. $A \rightarrow B : \{K_{A,B}, A\}_{K_{B,S}}$
4. $B \rightarrow A : \{N_B\}_{K_{A,B}}$
5. $A \rightarrow B : \{N_B - 1\}_{K_{A,B}}$

- Solución:

1. $A \rightarrow S : A, B, N_A$
2. $S \rightarrow A : \{N_A, B, K_{A,B}, \{K_{A,B}, A\}_{K_{B,S}}\}_{K_{A,S}}$
3. $A \rightarrow B : \{K_{A,B}, A\}_{K_{B,S}}$
4. $B \rightarrow A : \{N_B\}_{K_{A,B}}$
5. $A \rightarrow B : \{N_B - 1\}_{K_{A,B}}$

Solución: extender el uso del nonce en el resto de transacciones

• Protocolo Amended Needham Schroeder protocol

- soluciona el fallo del anterior Needham-Schroeder (en relación a los ataques de repetición)

$A \rightarrow B : A$

$B \rightarrow A : E_{KBT}\{A, N_{b0}\}$

$A \rightarrow T : A, B, N_a, E_{KBT}\{A, N_{b0}\}$

$T \rightarrow A : E_{KAT}\{A, N_a, K_{AB}, E_{KBT}\{A, K_{AB}, N_{b0}\}\}$

$A \rightarrow B : E_{KBT}\{A, K_{AB}, N_{b0}\}$

$B \rightarrow A : E_{KAB}\{N_b\}$

$A \rightarrow B : E_{KAB}\{N_{b-1}\}$

$3_1 : RA$

$3_2 : Alice, Bob, RA, E_{BT}(A, R_{B0})$

$4_3 : E_{AT}(RA, Bob, K_{AB}, E_{BT}(K_{AB}, Alice, R_{B0}))$

$1 : A$

$2_2 : E_{BT}(A, R_{B0})$

$5 : E_{BT}(K_{AB}, Alice, R_{B0})$

$6_2 : E_{AB}(R_B)$

$5 : E_{AB}(R_{B-1})$

Alice

$4_1 : K_{AB}$

$4_2 : E_{BT}(K_{AB}, Alice, R_{B0})$



Trent

$2_1 : R_{B0}$

$6_1 : R_B$



Bob

• Protocolo Otway-Rees

- soluciona también el fallo del **Needham-Schroeder**, aunque con un diseño diferente

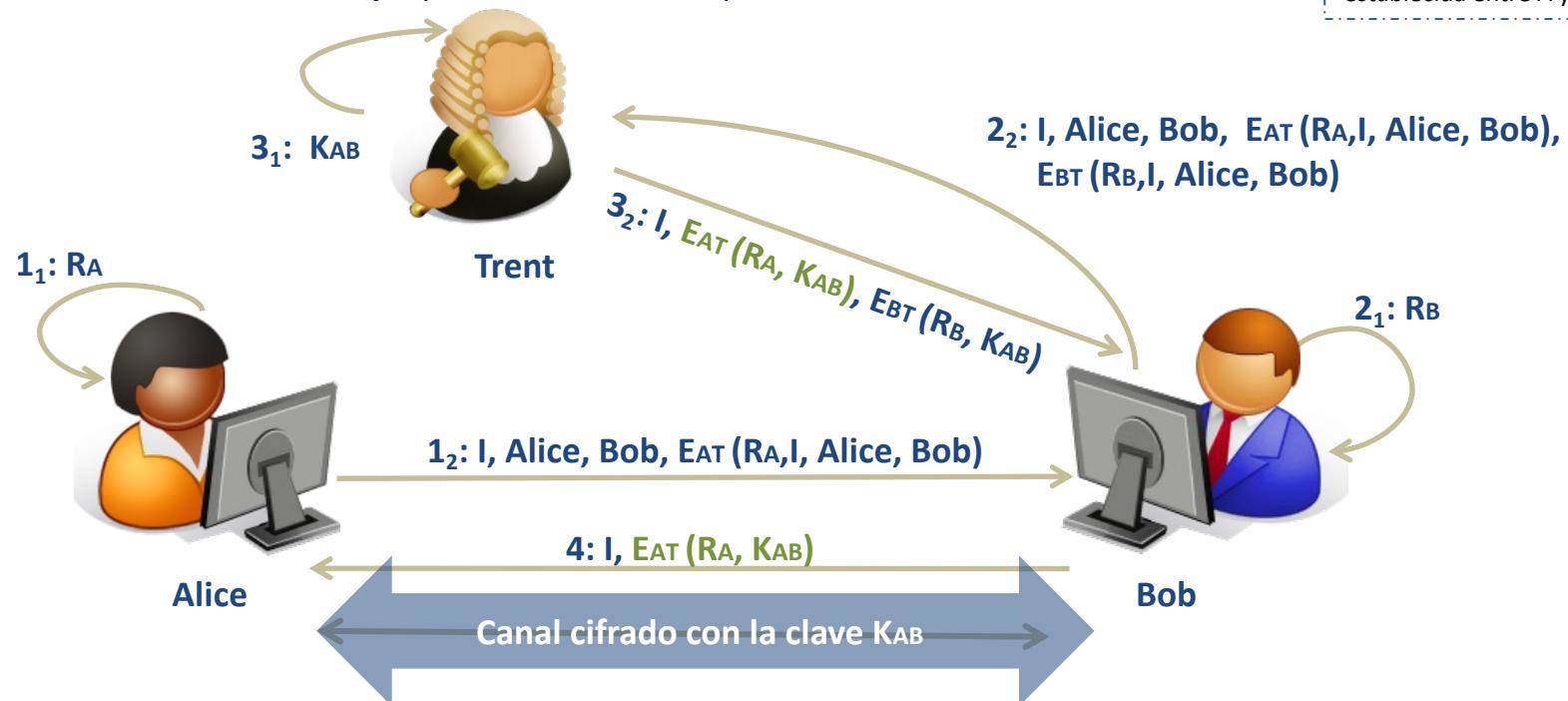
1: Alice genera el valor aleatorio R_A $\langle 1_1 \rangle$ y se lo envía hacia Bob, dentro de un mensaje cifrado con la clave que comparte con Trent $\langle 1_2 \rangle$.

2: Bob genera un valor aleatorio R_B y se lo envía a Trent usando la clave que comparte con éste $\langle 2_2 \rangle$. También le envía el mensaje que recibió de Alice.

3: Trent descifra el mensaje cifrado con la clave que comparte con Alice, genera la clave de sesión K_{AB} $\langle 3_1 \rangle$ y se la envía a Bob cifrada, junto a un mensaje para Alice $\langle 3_2 \rangle$.

4: Bob envía a Alice el mensaje que recibió de Trent para ella.

I (índice): I-ésima sesión establecida entre A y B



Otway-Rees

- Diseño formalizado:

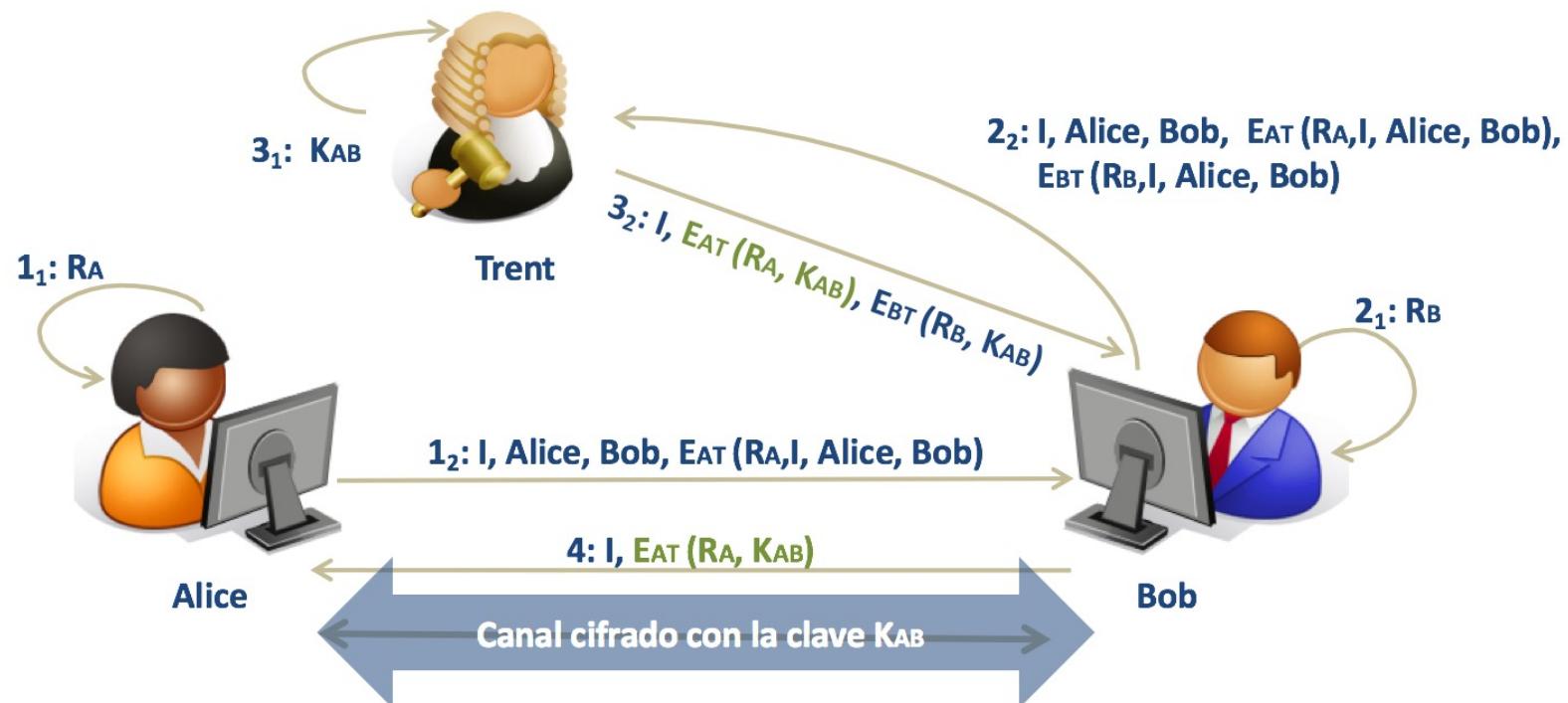
$A \rightarrow B : I, A, B, E_{KAT}\{N_a, I, A, B\}$

$B \rightarrow T : I, A, B, E_{KAT}\{N_a, I, A, B\}, E_{KBT}\{N_b, I, A, B\}$

$T \rightarrow B : I, E_{KAT}\{K_{AB}, N_a\}, E_{KBT}\{K_{AB}, N_b\}$

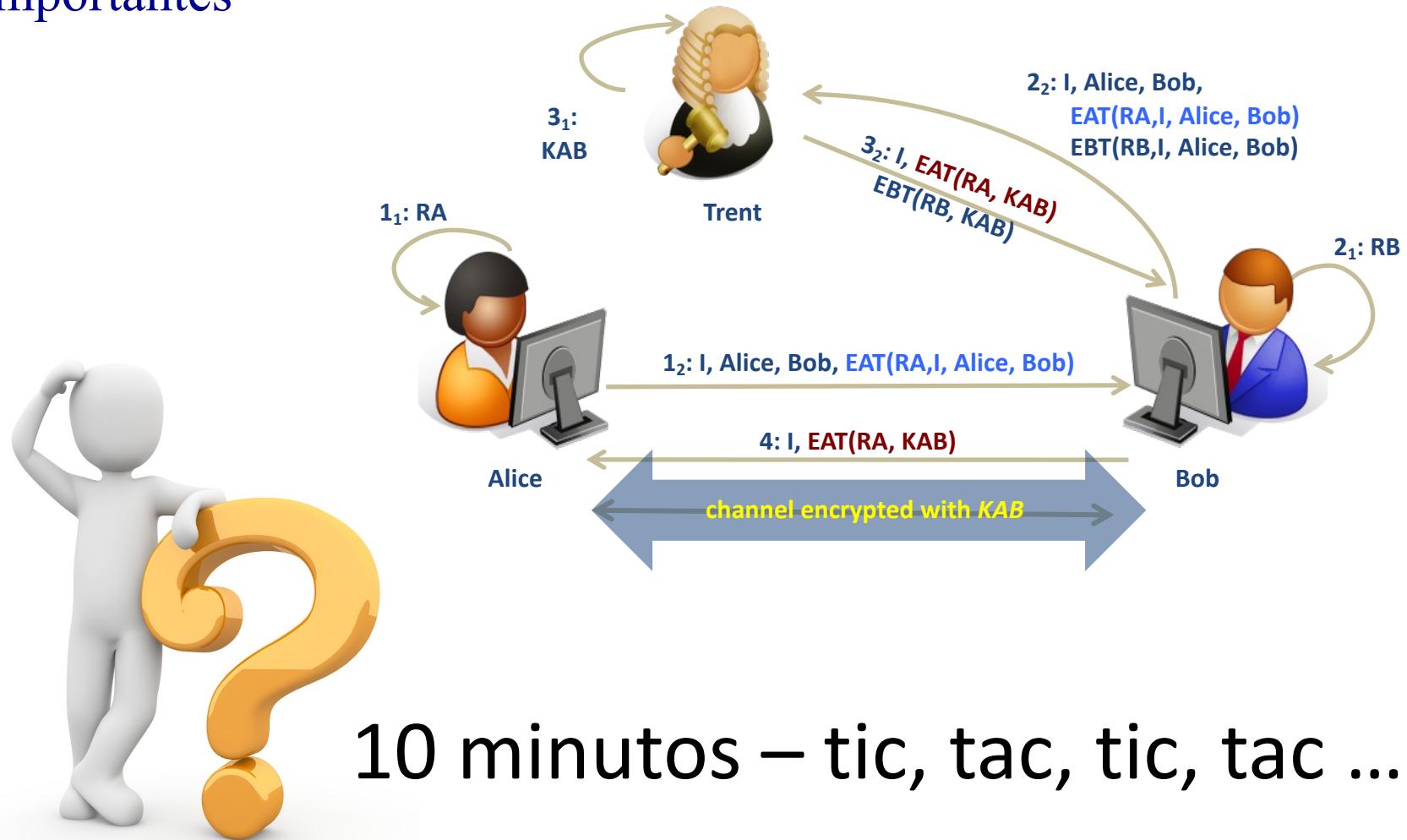
$B \rightarrow A : I, E_{KAT}\{K_{AB}, N_a\}$

Como se puede ver en el protocolo, Otway-Rees también intenta solucionar el problema del freshness en los mensajes

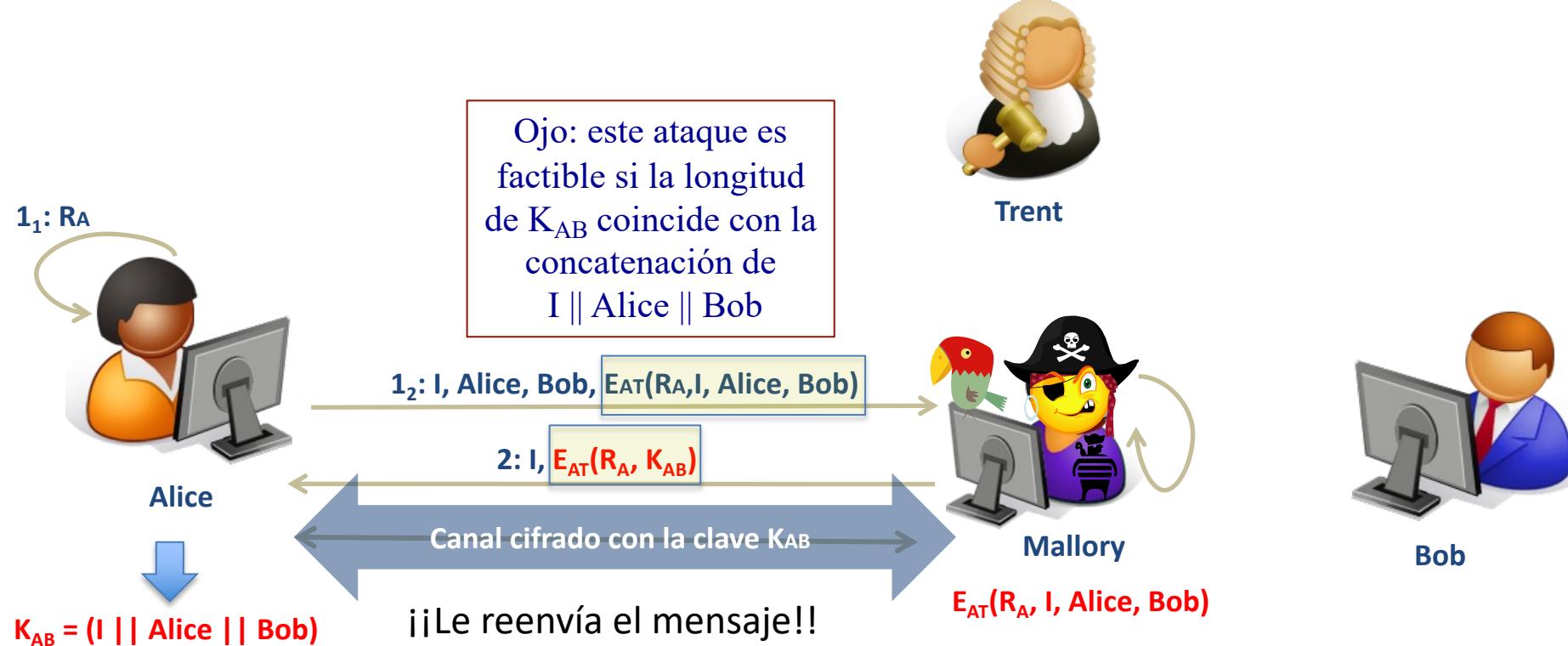


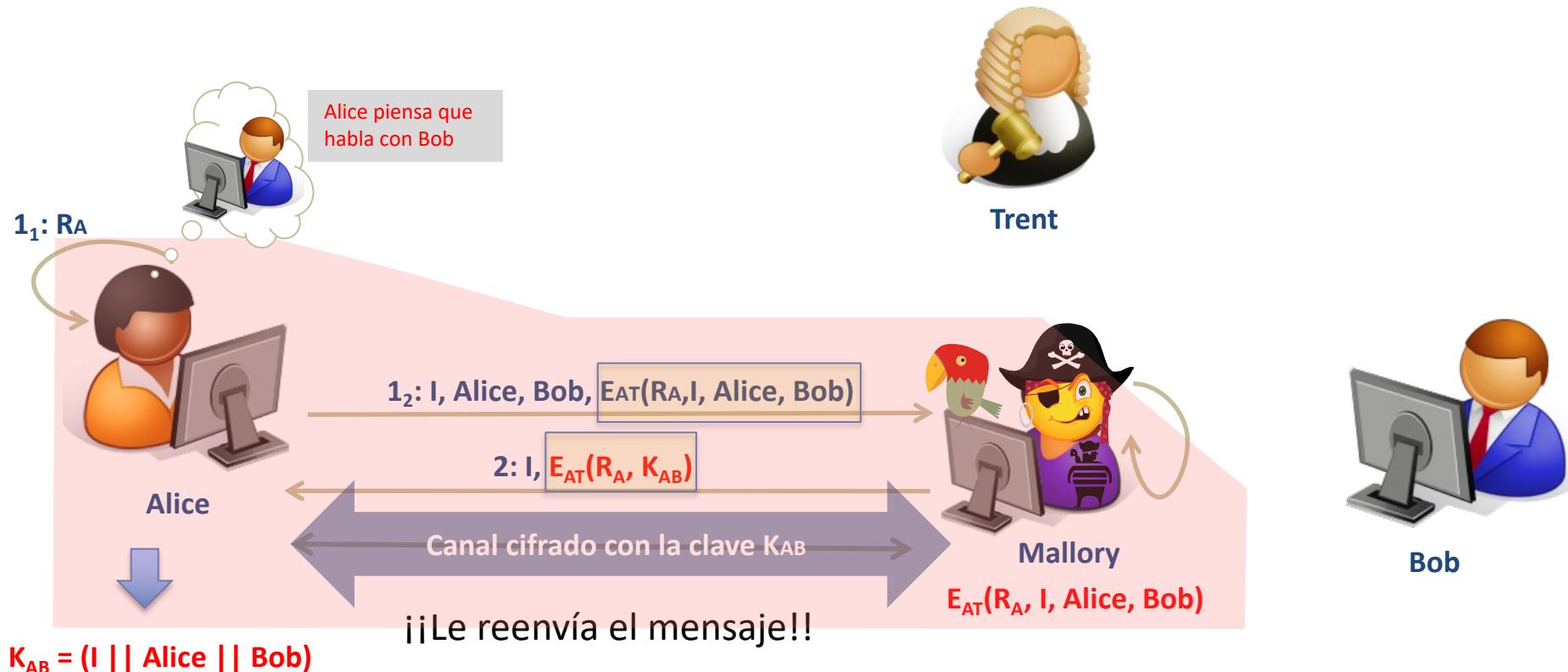
Otway-Rees

- Sin embargo, el protocolo presenta dos vulnerabilidades importantes



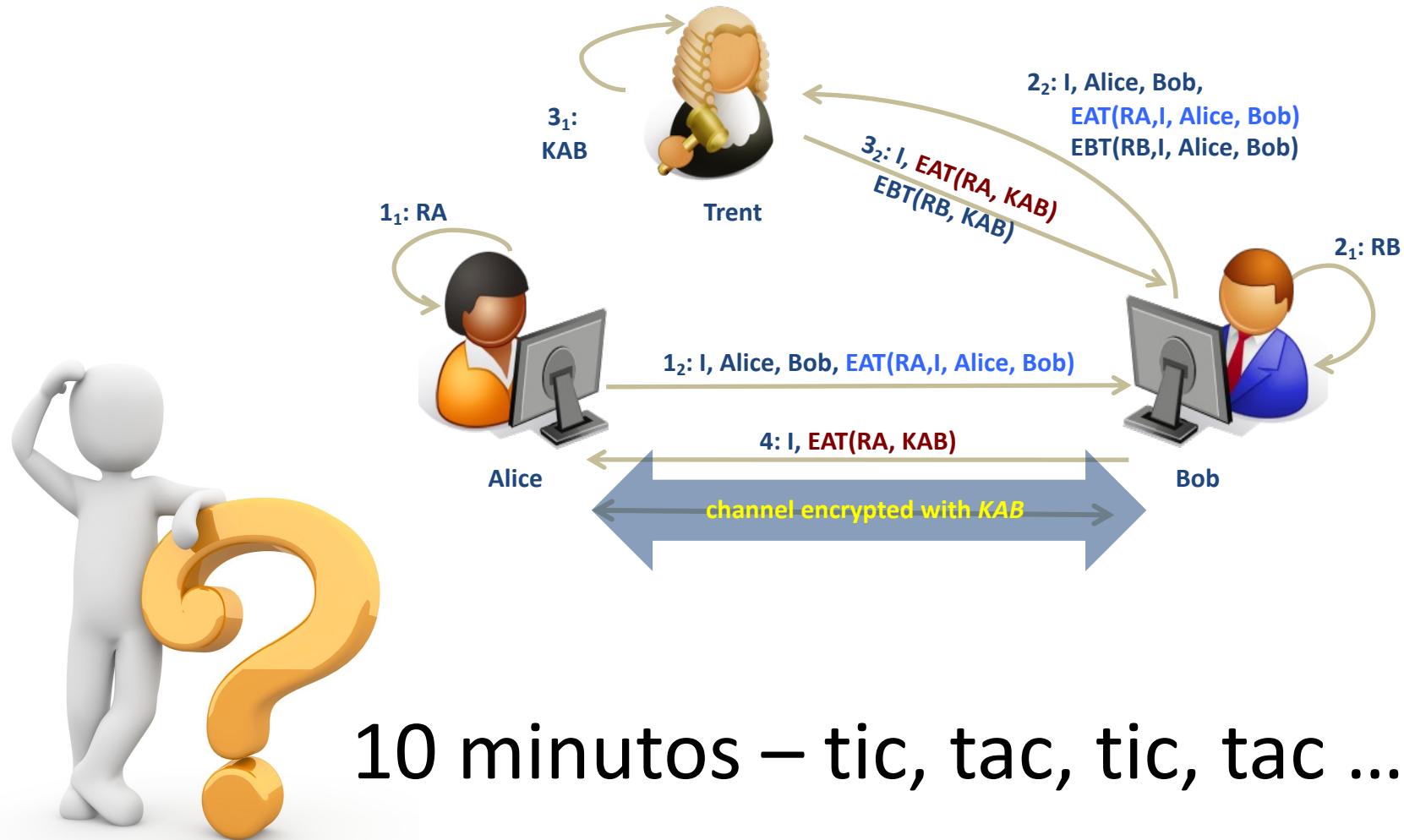
– Primer agujero de seguridad:





Otway-Rees

- ¿ Y el segundo agujero ?

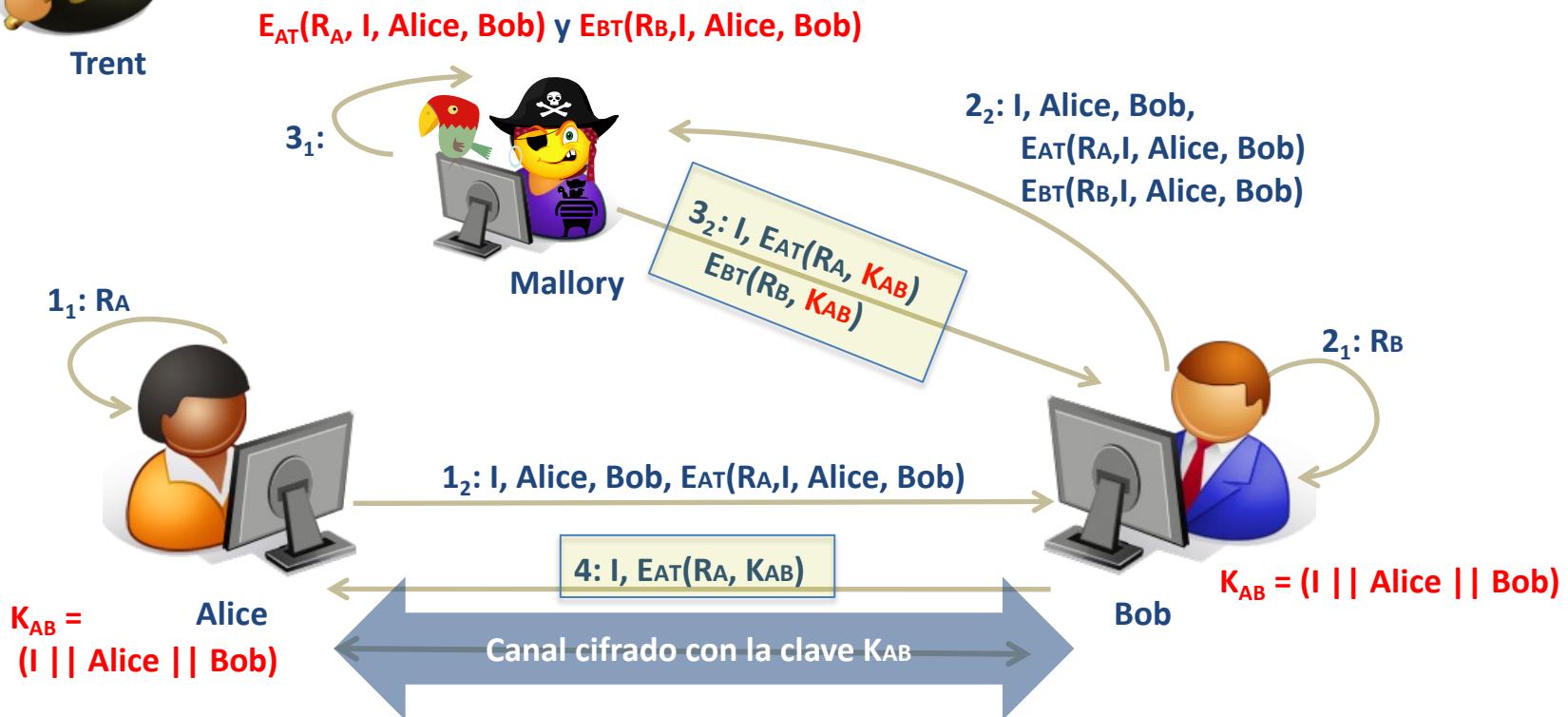


– Segundo agujero de seguridad:



Trent

¡¡Le reenvía los mismos mensajes!!

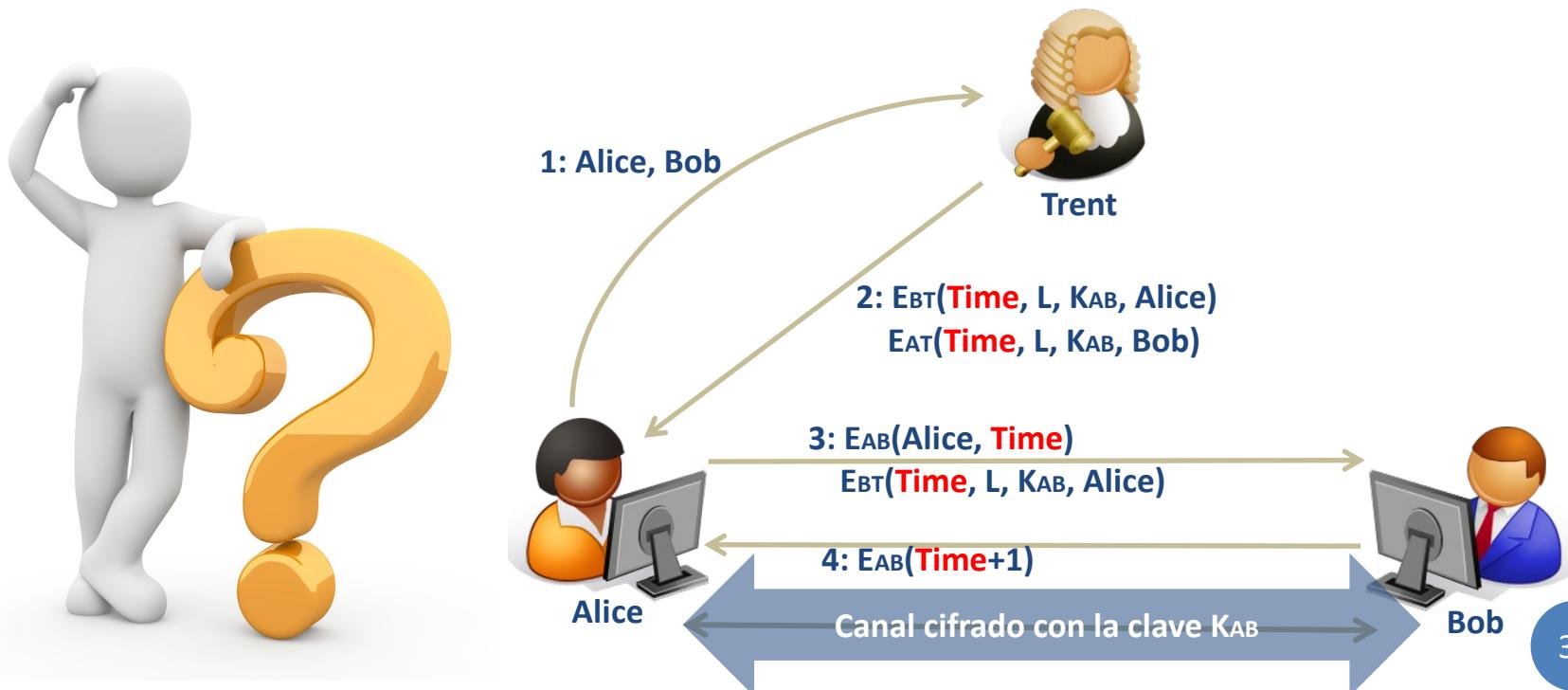


– Segundo agujero de seguridad:



• Protocolo Kerberos

- 1: Alice envía un mensaje a Trent con su identidad y la identidad de Bob.
- 2: Trent genera un mensaje con un *timestamp* (*Time*), un *tiempo de vida* (*L*), una clave de sesión aleatoria, y la identidad de Alice. Lo cifra con la clave compartida con Bob. Prepara un mensaje similar para Alice. Envía ambos mensajes cifrados a Alice.
- 3: Alice obtiene K_{AB} , genera un mensaje con su identidad y el timestamp, y lo cifra con K_{AB} para enviárselo a Bob. Alice también envía a Bob el mensaje cifrado que recibió de Trent.
- 4: Bob genera un mensaje que consta del timestamp más uno, lo cifra con K_{AB} y se lo envía a Alice.

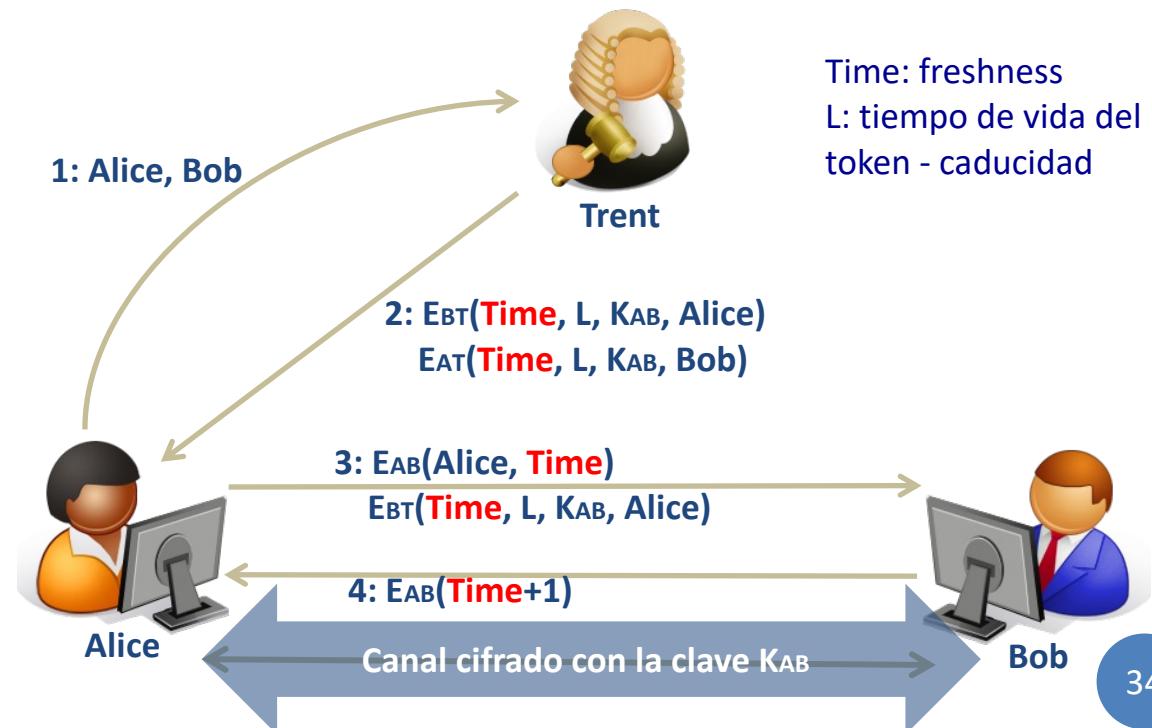


• Protocolo Kerberos

- 1: Alice envía un mensaje a Trent con su identidad y la identidad de Bob.
- 2: Trent genera un mensaje con un *timestamp* (*Time*), un *tiempo de vida* (*L*), una clave de sesión aleatoria, y la identidad de Alice. Lo cifra con la clave compartida con Bob. Prepara un mensaje similar para Alice. Envía ambos mensajes cifrados a Alice.
- 3: Alice obtiene K_{AB} , genera un mensaje con su identidad y el timestamp, y lo cifra con K_{AB} para enviárselo a Bob. Alice también envía a Bob el mensaje cifrado que recibió de Trent.
- 4: Bob genera un mensaje que consta del timestamp más uno, lo cifra con K_{AB} y se lo envía a Alice.

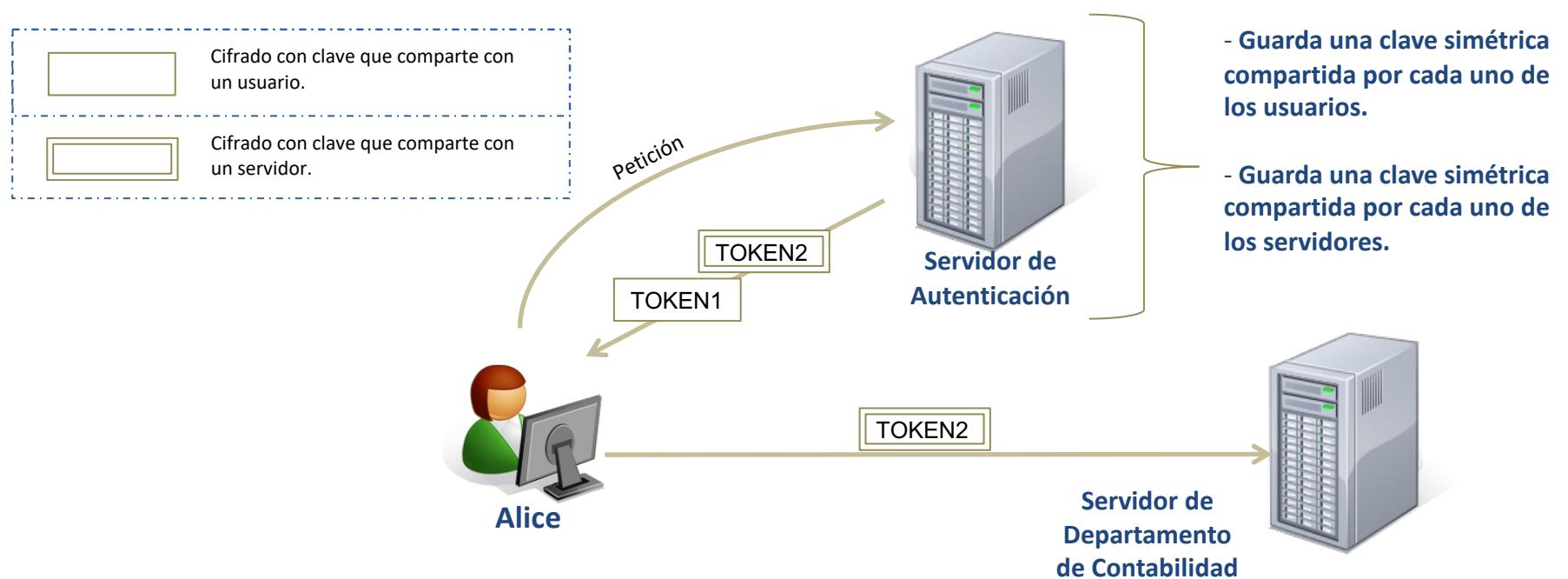
- Asume que los relojes de todos los sistemas están sincronizados
- En la práctica se sincronizan en el rango de unos pocos minutos

- Por fallos del sistema o por sabotaje, los relojes pueden desincronizarse (→ ataque de denegación de servicio)

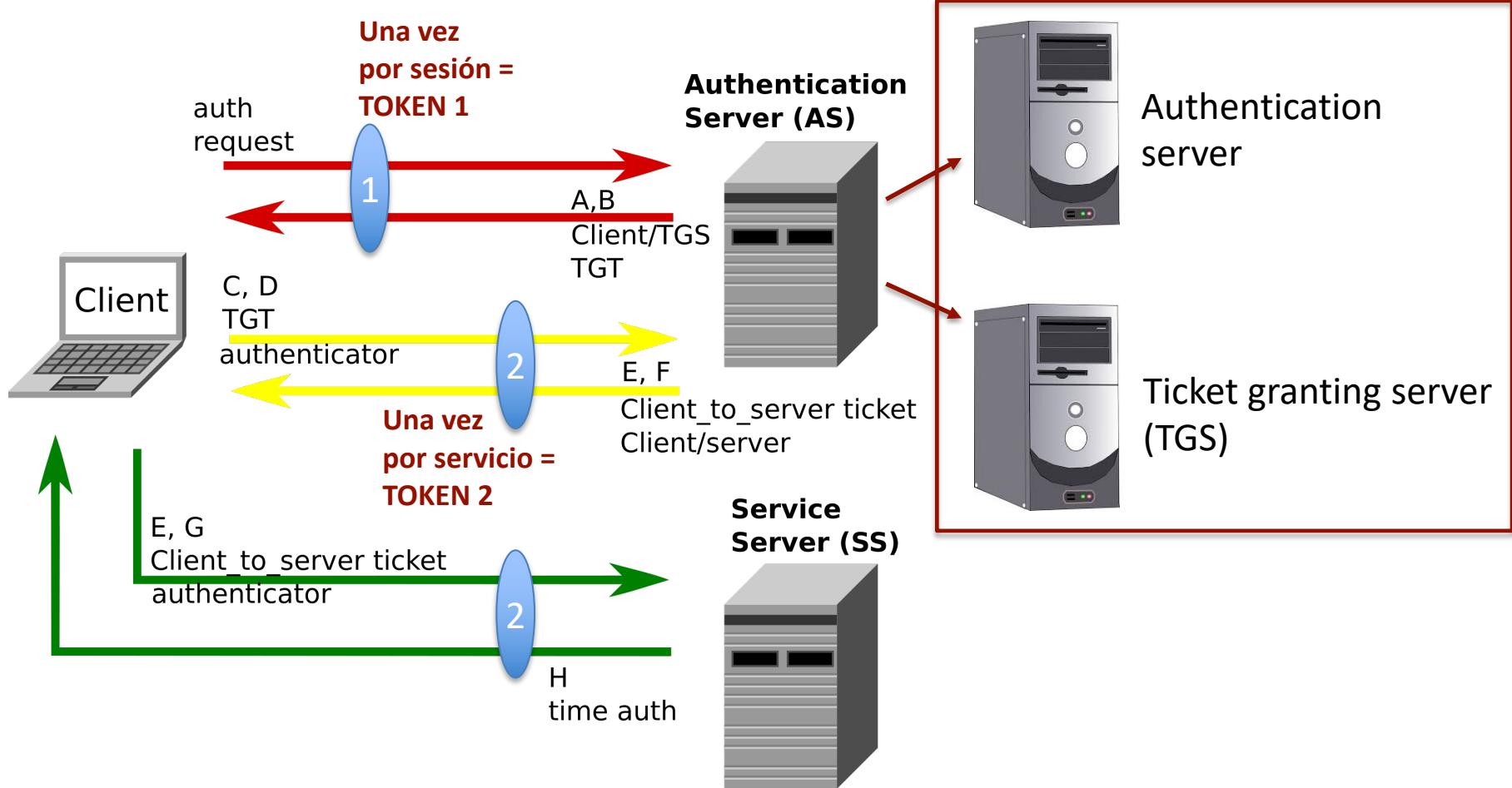


Kerberos

- Ejemplo de escenario de uso de Kerberos
 - Campus universitario, empresa, ...
 - Se usa Kerberos para evitar que cada usuario tenga una cuenta en cada servidor con el que va a contactar, y un tiempo límite de uso
 - en el escenario de abajo *Alice* accede al Servidor del Departamento de Contabilidad sin tener una cuenta en ese servidor

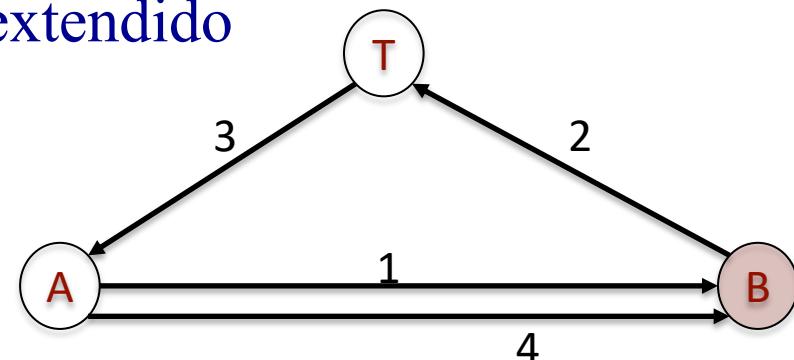


Kerberos

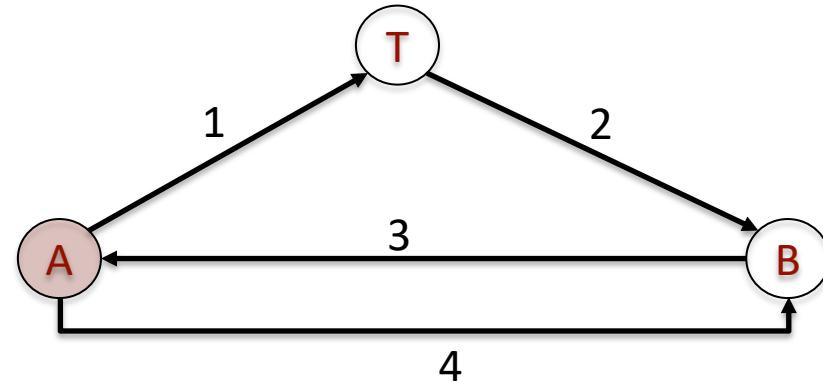


- Aparte de los protocolos anteriores, hay otros que combinan múltiples tipos de estrategias (a nivel de modelos como de mecanismos de seguridad):

– PUSH con PULL → PUSH extendido



– PULL con PUSH → PULL extendido



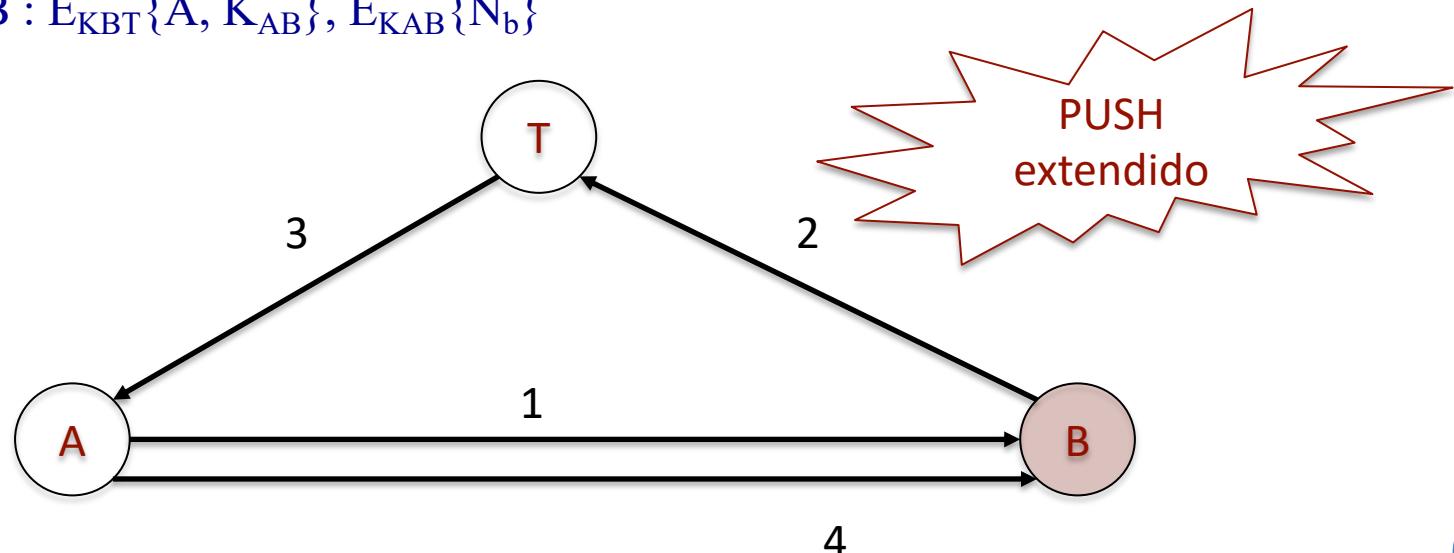
- **Yahalom**
 - Objetivo: permitir a Trent generar la clave de sesión K_{AB} y enviar dicha clave a Alice (de manera directa) y a B (de manera indirecta)

$A \rightarrow B : A, N_a$

$B \rightarrow T : B, E_{KBT}\{A, N_a, N_b\}$

$T \rightarrow A : E_{KAT}\{B, K_{AB}, N_a, N_b\}, E_{KBT}\{A, K_{AB}\}$

$A \rightarrow B : E_{KBT}\{A, K_{AB}\}, E_{KAB}\{N_b\}$



- Dado el protocolo anterior:

$A \rightarrow B : A, N_a$

$B \rightarrow T : B, E_{KBT}\{A, N_a, N_b\}$

$T \rightarrow A : E_{KAT}\{B, K_{AB}, N_a, N_b\}, E_{KBT}\{A, K_{AB}\}$

$A \rightarrow B : E_{KBT}\{A, K_{AB}\}, E_{KAB}\{N_b\}$

- Analizar los posibles problemas de seguridad
- ¿Hay desafío y respuesta?



- **Neuman Stubblebine**

- Objetivo: combinar formas para verificar la frescura de las transacciones – *time-stamps, nonces*

$A \rightarrow B : A, N_a$

$B \rightarrow T : B, EK_{BT}\{A, N_a, \text{time-stamp}_b\}, N_b$

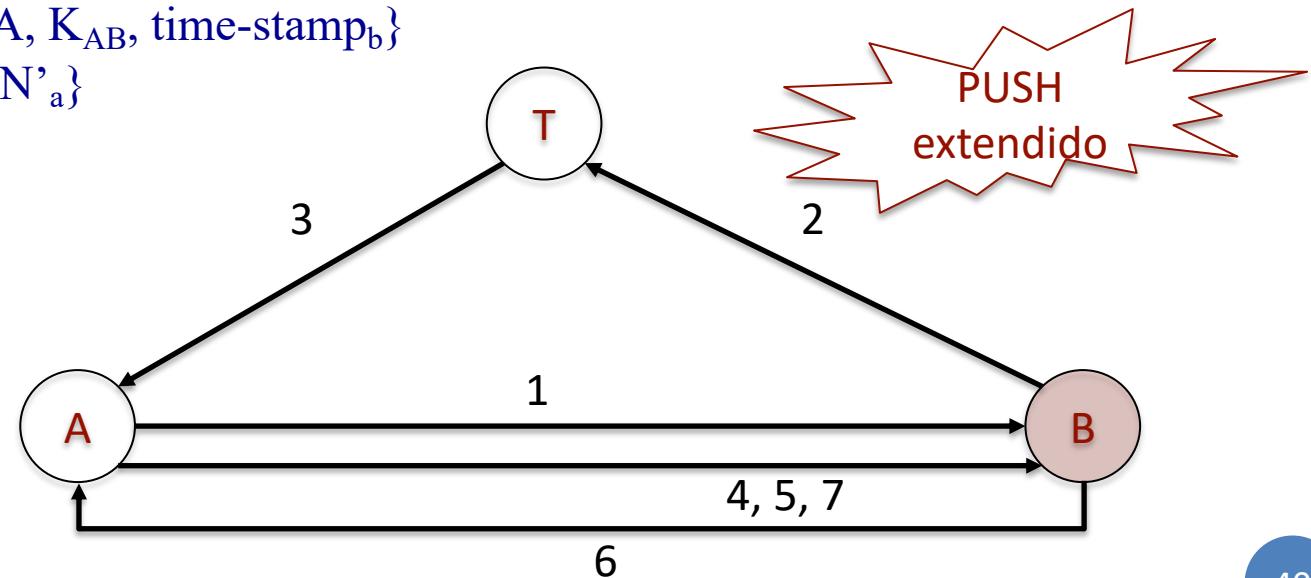
$T \rightarrow A : EK_{AT}\{B, K_{AB}, N_a, \text{time-stamp}_b\}, EK_{BT}\{A, K_{AB}, \text{time-stamp}_b\}, N_b$

$A \rightarrow B : EK_{BT}\{A, K_{AB}, \text{time-stamp}_b\}, EK_{AB}\{N_b\}$

$A \rightarrow B : N'_a, EK_{BT}\{A, K_{AB}, \text{time-stamp}_b\}$

$B \rightarrow A : N'_b, EK_{AB}\{N'_a\}$

$A \rightarrow B : EK_{AB}\{N'_b\}$



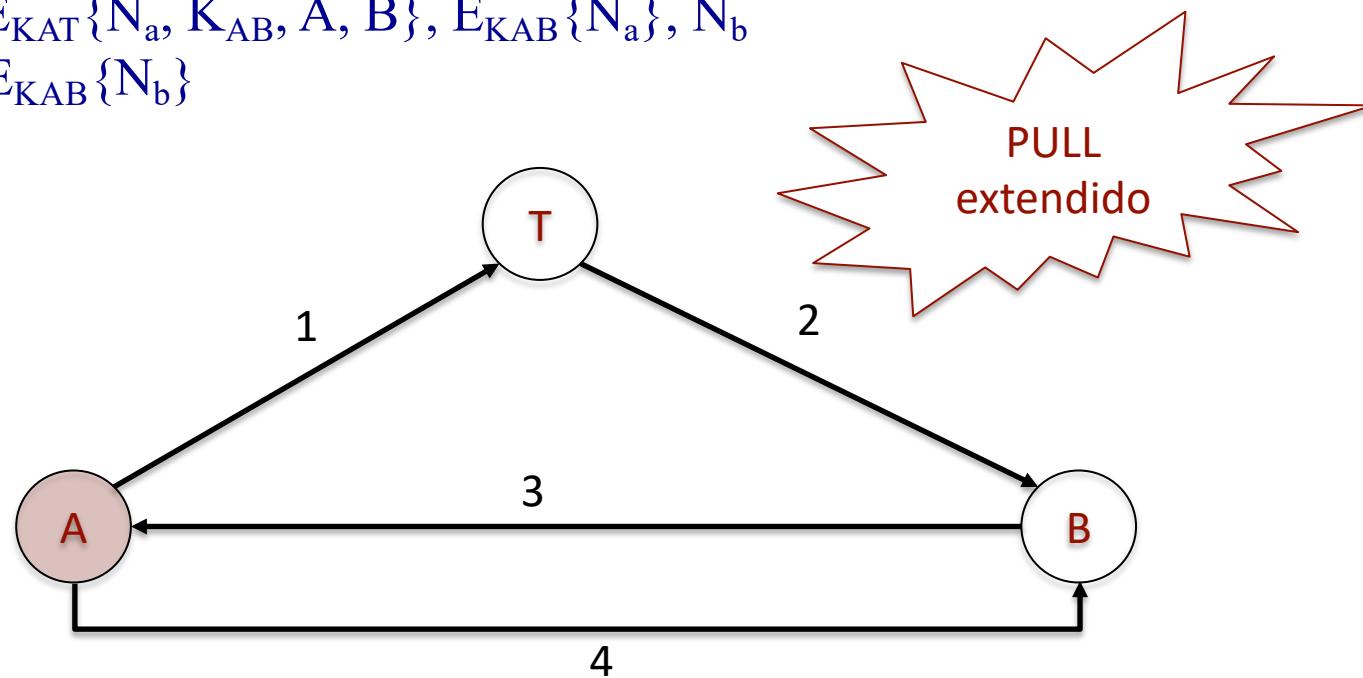
- **Kao Chow**

$A \rightarrow T : A, B, N_a$

$T \rightarrow B : E_{KAT}\{N_a, K_{AB}, A, B\}, E_{KBT}\{N_a, K_{AB}, A, B\}$

$B \rightarrow A : E_{KAT}\{N_a, K_{AB}, A, B\}, E_{KAB}\{N_a\}, N_b$

$A \rightarrow B : E_{KAB}\{N_b\}$

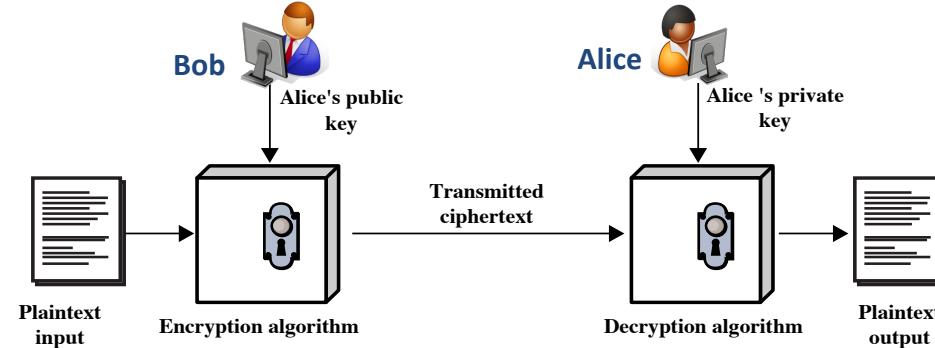


- Hemos visto que existen diversos protocolos para solucionar el problema de la administración/intercambio de claves
 - El protocolo a elegir depende del escenario y de lo que se quiere proteger
 - ¿Se quiere aprovechar los canales de comunicación?
 - ¿Quién ha de contactar primero con el KDC: Alice o Bob?
 - ¿Debe el KDC contactar directamente con ambos o es suficiente que lo haga con sólo uno de ellos?
 - ¿Se quiere proteger las comunicaciones de posibles ataques replay?
 - ¿Se quiere desafío y respuesta?
 -

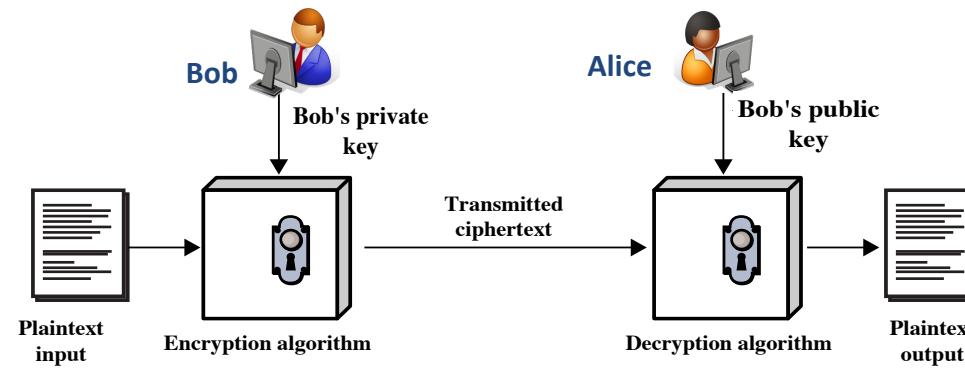
- Usar un KDC no está exento de potenciales problemas:
 1. el KDC posee suficiente **información para suplantar a cualquier usuario**
 - y si un intruso llega hasta él todos los documentos cifrados que circulan por la red se hacen vulnerables
 2. el KDC representa un **único punto de fallos** (o ataques)
 - si queda inutilizado, nadie puede establecer comunicaciones seguras dentro de la red
 3. el **rendimiento** de todo el sistema **puede bajar** cuando el KDC se convierte en un cuello de botella
 - lo cual no es difícil ya que todos los usuarios necesitan comunicar con él de forma frecuente con objeto de obtener las claves

Mecanismos e Infraestructuras de administración de claves públicas.

- ¿Cómo sabe *Bob* en este escenario si la clave pública de *Alice* es genuina?

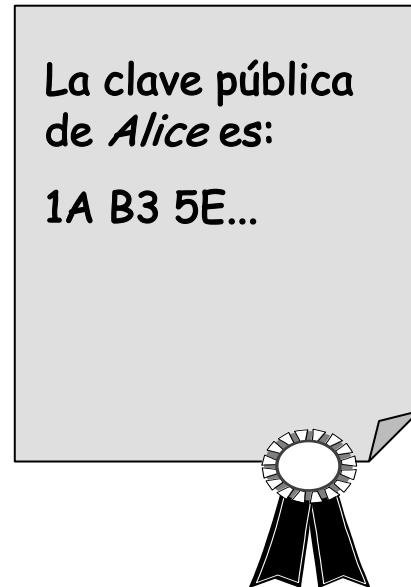


- ¿Cómo sabe *Alice* en este escenario si la clave pública de *Bob* es genuina?



Certificados digitales

- Las preguntas anteriores equivalen a plantearse ¿cómo garantizar que **las claves públicas** de *Alice* y *Bob* **son auténticas**?
- Veamos, como ejemplo, el caso en el que *Bob* necesita la clave pública de *Alice*
 - *Bob* necesitaría algún documento digital con algún “**sello de garantía**”, o sea, algo equivalente a lo que en papel sería:



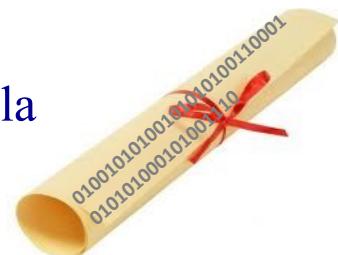
Certificados digitales

- En realidad, ¿qué información relevante habría de contener ese documento digital para optimizar su utilidad?
 - La **identidad del usuario** al respecto del cual se ofrece información (*Alice* en la figura anterior)
 - El valor de la **clave pública** de Alice (o sea, 1A B3 5E ...)
 - Algo que identifique unívocamente a ese documento entre otros muchos (por ejemplo, un **número de serie**)
 - ¿sirve la identidad del usuario para este menester?
 - No, porque un usuario puede tener más de un par <clave pública, clave privada>
 - Algo que indique “desde” cuándo y “hasta” cuándo es válido el documento digital (por ejemplo, una fecha de **emisión** y una de **expiración**)
 - La identidad de **quien emite** el documento
 - La **firma digital** de quien emite el documento



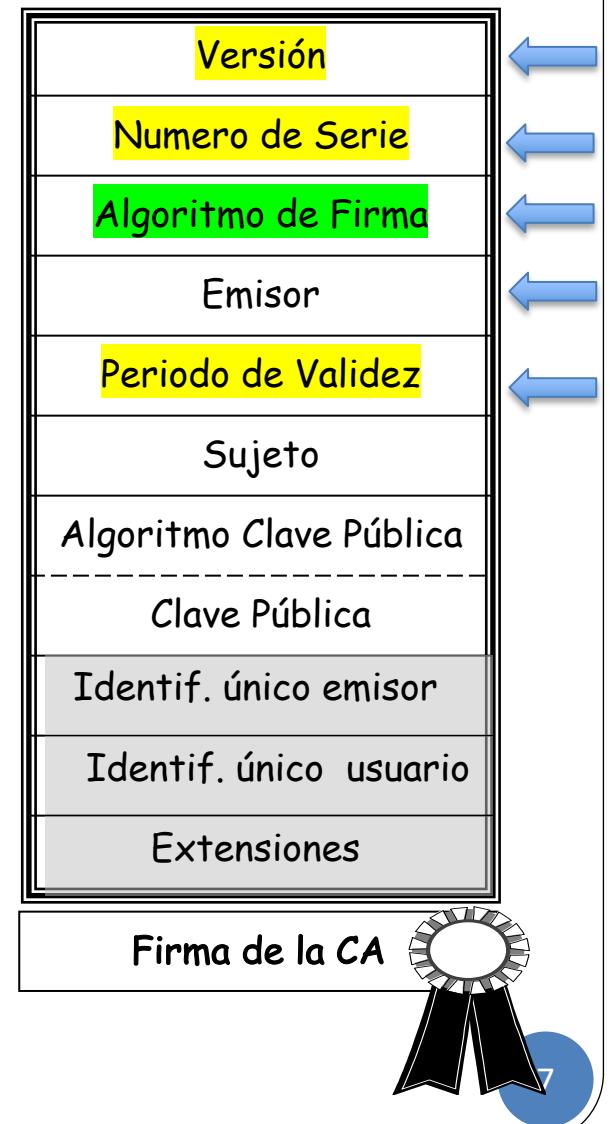
Certificados digitales

- El documento digital con esa información se denomina **certificado digital o certificado de clave pública**
 - Es la firma digital de un documento (que contiene la información antes mencionada) la que garantiza que cierta clave pública pertenece a un determinado usuario
 - Dicho de otro modo, un certificado digital corresponde a un documento digital que permite validar técnica y legalmente la identidad de una persona, y asociar la clave pública a dicha persona
 - Luego, se garantiza al menos autenticación
 - Con el certificado digital se puede realizar varias acciones
 - firmar digitalmente documentos (ej. doc en PDF) y garantizar el no-repudio
 - conectar máquinas – cliente-servidor
- Se denomina **Autoridad de Certificación** a la *tercera parte confiable* (TTP) que emite y administra los certificados digitales de los usuarios de un sistema
 - Garantiza que una clave pública pertenece a cierto usuario inequívocamente identificado

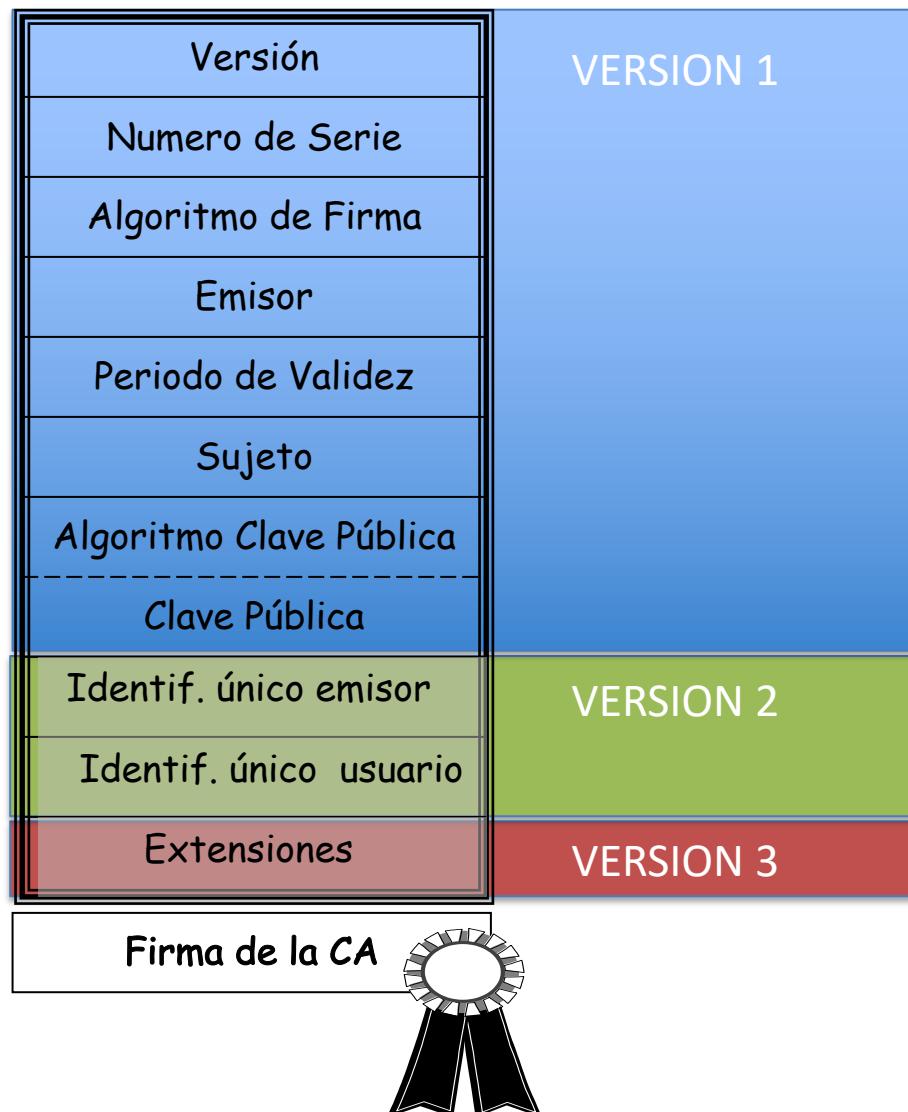


Certificados digitales

- La ITU-T ha definido una estructura estándar de certificado digital que ha sido adoptada internacionalmente: **certificado X.509**
 - *Versión:*
 - indica el número de versión de X.509 (o sea, 1, 2 ó 3)

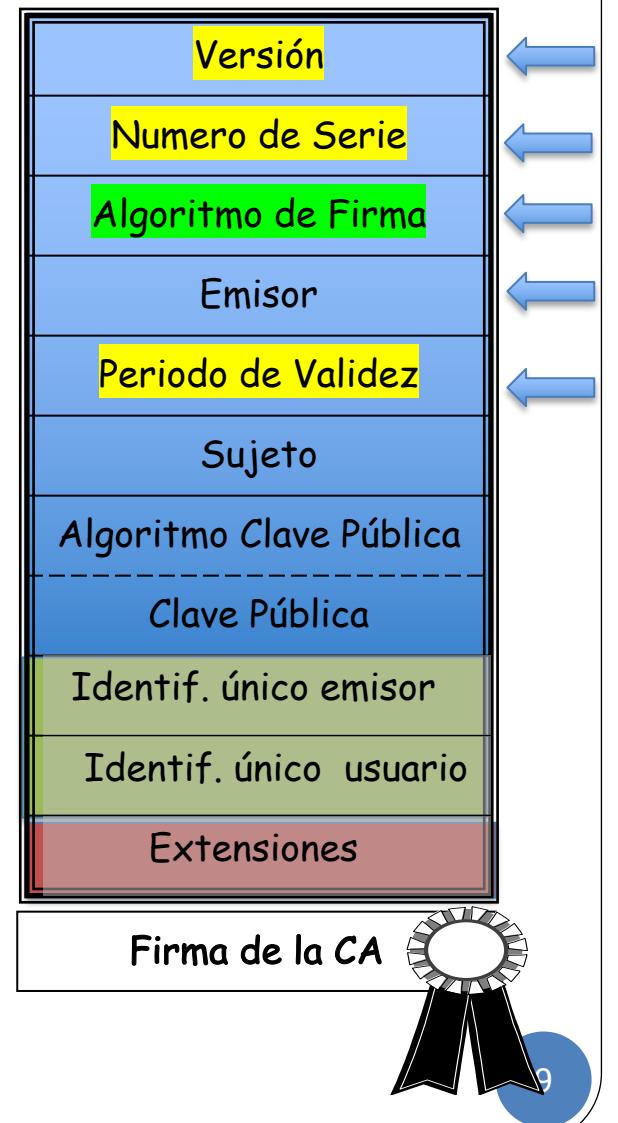


Certificados digitales



Certificados digitales

- La ITU-T ha definido una estructura estándar de certificado digital que ha sido adoptada internacionalmente: **certificado X.509**
 - **Versión:**
 - indica el número de versión de X.509 (o sea, 1, 2 ó 3)
 - **Número de Serie:**
 - número de identificación único para este certificado digital, asignado por la CA
 - **Algoritmo de Firma:**
 - identificador del algoritmo de firma digital usado por la CA para firmar el certificado
 - ¿Para qué?: para que la persona quien reciba el certificado conozca cómo verificar la firma del certificado
 - **Emisor:**
 - Nombre X.500 de la **CA emisora**
 - **Periodo de Validez:**
 - Fecha desde el que el certificado comienza a ser válido, y día y hora de expiración
 - Es decir:
 - Fecha de firma: XX-XX-XXXX
 - Fecha de expiración: XX-XX-XXXX



Certificados digitales

– *Sujeto:*

- nombre en formato X.500 del usuario cuya clave pública se está certificando

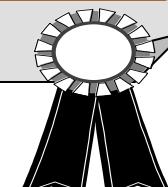
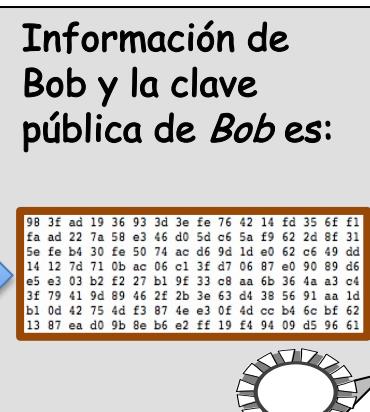
– *Algoritmo Clave Pública:*

- identificador del algoritmo de clave pública con el que se ha de utilizar la clave pública
- ¿Para qué?: para que la persona quien reciba el certificado conozca cómo usar la clave pública del sujeto, dueño del certificado

– *Clave Pública:*

- Valor de la clave pública del usuario
- En definitiva, este valor contiene la clave pública del sujeto, dueño del certificado

```
98 3f ad 19 36 93 3d 3e fe 76 42 14 fd 35 6f f1  
fa ad 22 7a 58 e3 46 d0 5d c6 5a f9 62 24 8f 31  
5e fe b4 30 fe 50 74 ac d6 9d 1d e0 62 c6 49 dd  
14 12 7d 71 0b ac 06 c1 3f d7 06 87 e0 90 89 d6  
e5 e3 03 b2 f2 27 b1 9f 33 c8 aa 6b 36 4a a3 c4  
3f 79 41 9d 89 46 2f 2b 3e 63 d4 38 56 91 aa 1d  
b1 0d 42 75 4d f3 87 4e e3 0f 4d cc b4 6c bf 62  
13 87 ea d0 9b 8e b6 e2 ff 19 f4 94 09 d5 96 61
```



Certificados digitales

– *Identificador único de emisor:*

- Cadena **opcional** para que el nombre de la **CA** no sea ambiguo, en caso de que esto pudiera ocurrir

Ejemplo:

- Dirección de email, dirección postal

– *Identificador único de usuario:*

- Cadena **opcional** para que el nombre del **usuario** no sea ambiguo, en caso de que pudiera ocurrir

Ejemplo:

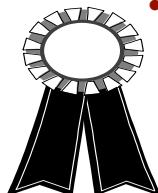
- Dirección de email, dirección postal

– *Extensiones:*

- Campo **opcional** para almacenar información de distinto tipo (versión 3)

– *Firma:*

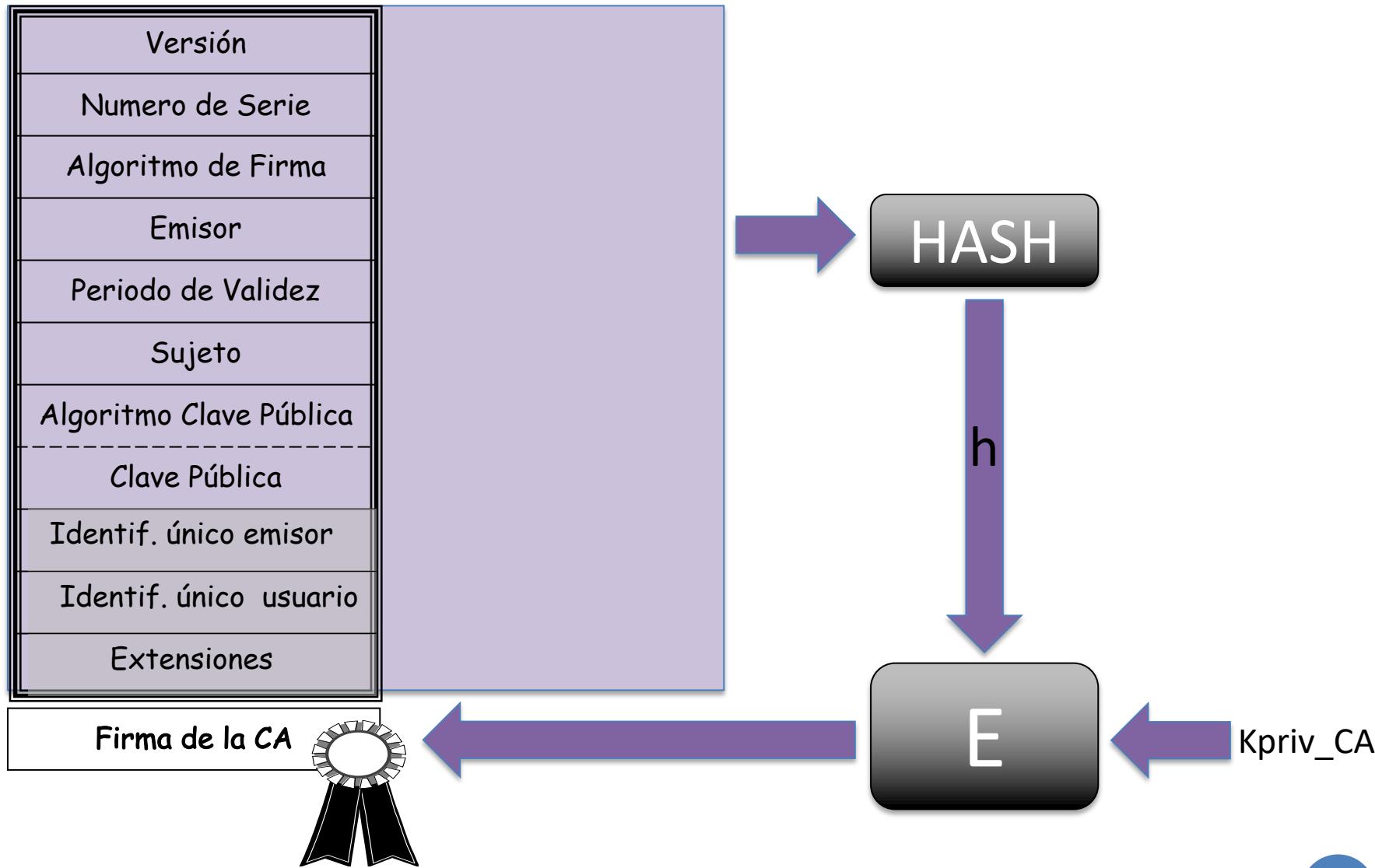
- **Firma digital de la CA sobre el valor hash del conjunto de los demás campos del certificado**



SELLO DIGITAL: $E_{K_{privCA}}(H(\text{documento}))$



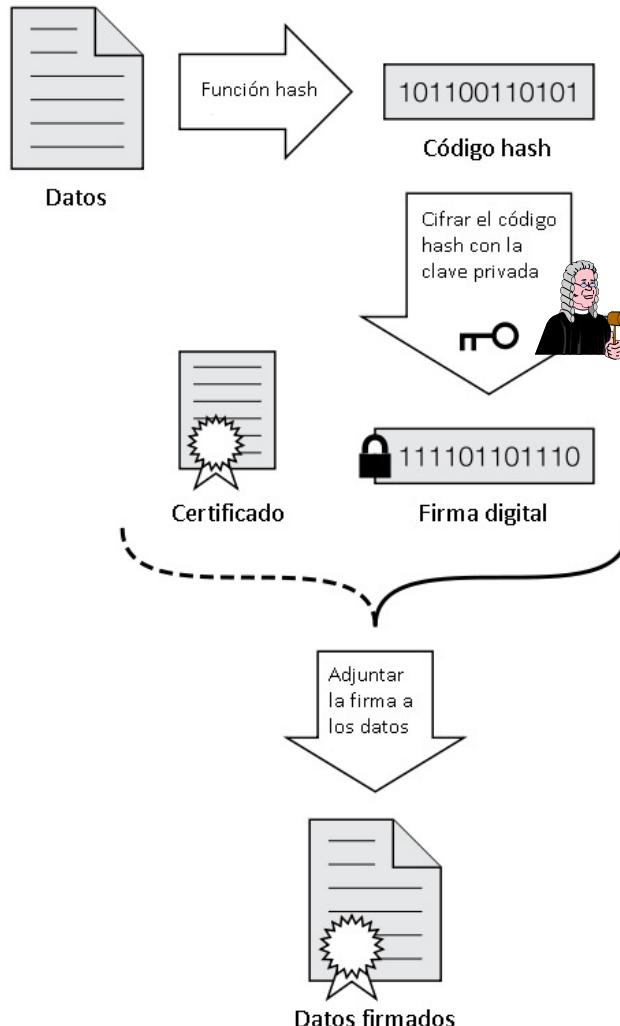
Certificados digitales



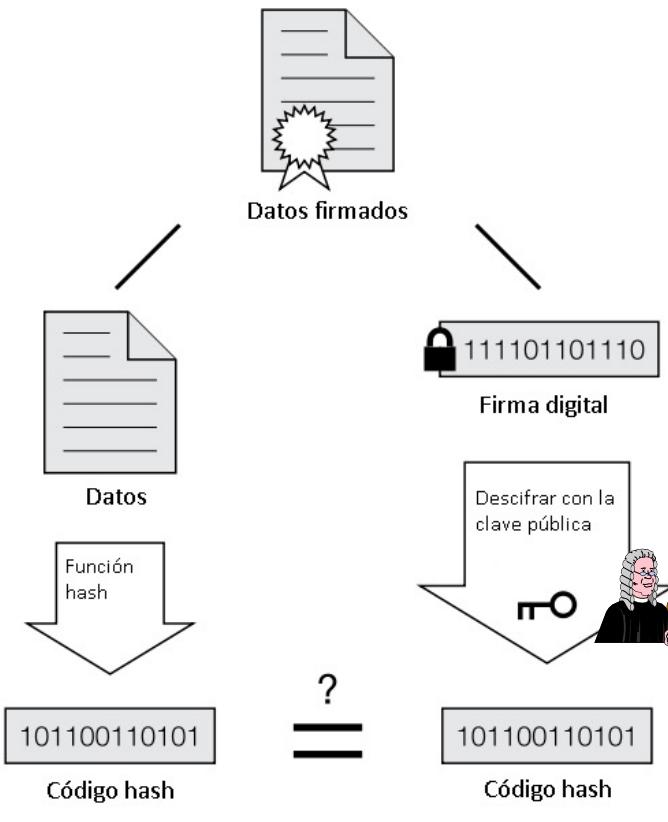
Certificados digitales



Firma Digital



Comprobación de una Firma



Si los códigos hash coinciden, la firma es válida

Ejemplo de un certificado digital - en MAC



Cristina

Entidad de certificación raíz

Caduca: jueves, 19 de marzo de 2026, 20:26:58 (hora estándar de Europa central)

✗ Este certificado raíz no es fiable

► Confiar

▼ Detalles

Nombre del sujeto

País SP

Estado/Provincia Malaga

Localidad Malaga

Empresa UMA

Unidad organizativa UMA

Nombre común Cristina

Dirección de correo ab@lcc.uma.es

Nombre del emisor

País SP

Estado/Provincia Malaga

Localidad Malaga

Empresa UMA

Unidad organizativa UMA

Nombre común Cristina

Dirección de correo ab@lcc.uma.es

Número de serie 00 D9 AE F5 9B 24 2D 04 FE

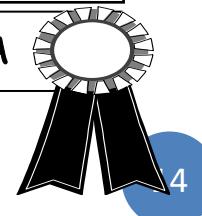
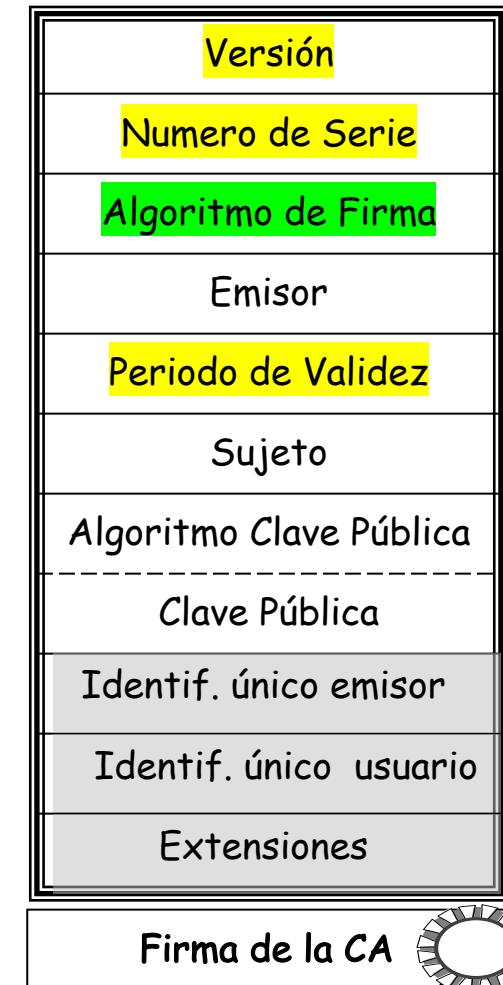
Versión 3

Algoritmo de firma SHA-1 con encriptación RSA (1.2.840.113549.1.1.5)

Parámetros ninguno/a

No válido antes de lunes, 21 de marzo de 2016, 20:26:58 (hora estándar de Europa central)

No válido después de jueves, 19 de marzo de 2026, 20:26:58 (hora estándar de Europa central)



Ejemplo de un certificado digital - en MAC



Cristina

Entidad de certificación raíz

Caduca: jueves, 19 de marzo de 2026, 20:26:58 (hora estándar de Europa central)

✗ Este certificado raíz no es fiable

► Confiar

▼ Detalles

Nombre del sujeto

País SP

Estado/Provincia Malaga

Localidad Malaga

Empresa UMA

Unidad organizativa UMA

Nombre común Cristina

Dirección de correo ab@lcc.uma.es

Nombre del emisor

País SP

Estado/Provincia Malaga

Localidad Malaga

Empresa UMA

Unidad organizativa UMA

Nombre común Cristina

Dirección de correo ab@lcc.uma.es

Número de serie 00 D9 AE F5 9B 24 2D 04 FE

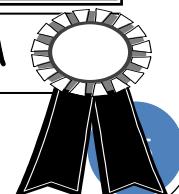
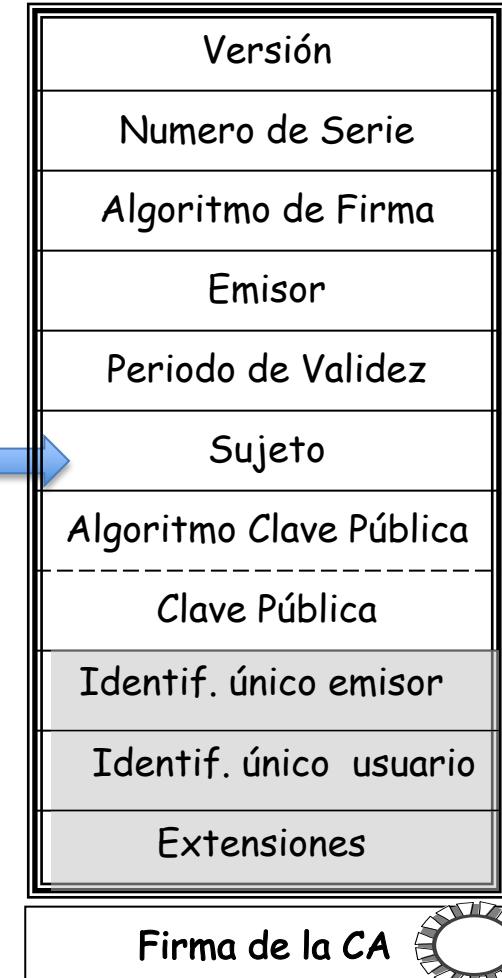
Versión 3

Algoritmo de firma SHA-1 con encriptación RSA (1.2.840.113549.1.1.5)

Parámetros ninguno/a

No válido antes de lunes, 21 de marzo de 2016, 20:26:58 (hora estándar de Europa central)

No válido después de jueves, 19 de marzo de 2026, 20:26:58 (hora estándar de Europa central)



Ejemplo de un certificado digital - en MAC

Información de la clave pública

Algoritmo Encriptación RSA (1.2.840.113549.1.1.1)
Parámetros ninguno/a
Clave pública 128 bytes: BF F9 49 27 D5 E6 29 D5 ...
Exponente 65537
Tamaño de la clave 1024 bits
Uso de la clave Cualquiera
Firma 128 bytes: 98 43 60 F8 B4 C4 D7 E1 ...

Extensión Restricciones básicas (2.5.29.19)
Crítico NO
Entidad de certificación Sí

Extensión Identificador de clave del sujeto (2.5.29.14)
Crítico NO
Nombre de la clave 24 85 28 DC 50 C1 BD 39 5D D0 D5 72 A5 09 1B F4 73 9D AF 7F

Extensión Identificador de clave de entidad emisora (2.5.29.35)
Crítico NO
Nombre de la clave 24 85 28 DC 50 C1 BD 39 5D D0 D5 72 A5 09 1B F4 73 9D AF 7F
Nombre de directorio
País SP
Estado/Provincia Malaga
Localidad Malaga
Empresa UMA
Unidad organizativa UMA
Nombre común Cristina
Dirección de correo ab@lcc.uma.es
Número de serie 00 D9 AE F5 9B 24 2D 04 FE

Huellas digitales
SHA1 8B 5F 16 E7 64 66 15 9C 89 F3 C1 13 44 94 44 A0 69 75 8F 90
MD5 D6 DB ED 1D 4B DC B3 42 17 31 78 D7 70 8E 0A 96

Versión

Número de Serie

Algoritmo de Firma

Emisor

Periodo de Validez

Sujeto

Algoritmo Clave Pública

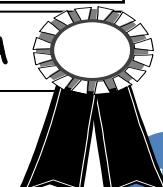
Clave Pública

Identif. único emisor

Identif. único usuario

Extensiones

Firma de la CA



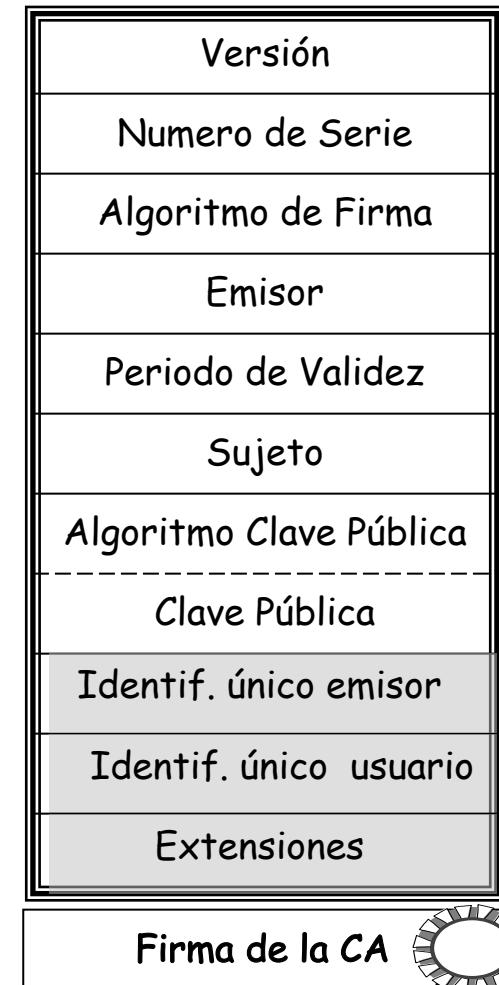
Ejemplo de un certificado digital - en MAC

Información de la clave pública

Algoritmo	Encriptación RSA (1.2.840.113549.1.1.1)
Parámetros	ninguno/a
Clave pública	128 bytes: BF F9 49 27 D5 E6 29 D5 ...
Exponente	65537
Tamaño de la clave	1024 bits
Uso de la clave	Cualquiera
Firma	128 bytes: 98 43 60 F8 B4 C4 D7 E1 ...

Extensión	Restricciones básicas (2.5.29.19)
Crítico	NO
Entidad de certificación	
Extensión	Identificador de clave del sujeto (2.5.29.14)
Crítico	NO
Nombre de la clave	24 85 28 DC 50 C1 BD 39 5D D0 D5 72 A5 09 1B F4 73 9D AF 7F
Extensión	Identificador de clave de entidad emisora (2.5.29.35)
Crítico	NO
Nombre de la clave	24 85 28 DC 50 C1 BD 39 5D D0 D5 72 A5 09 1B F4 73 9D AF 7F
Nombre de directorio	
País	SP
Estado/Provincia	Malaga
Localidad	Malaga
Empresa	UMA
Unidad organizativa	UMA
Nombre común	Cristina
Dirección de correo	ab@lcc.uma.es
Número de serie	00 D9 AE F5 9B 24 2D 04 FE

Huellas digitales	
SHA1	8B 5F 16 E7 64 66 15 9C 89 F3 C1 13 44 94 44 A0 69 75 8F 90
MD5	D6 DB ED 1D 4B DC B3 42 17 31 78 D7 70 8E 0A 96



Ejemplo de un certificado digital - en MAC

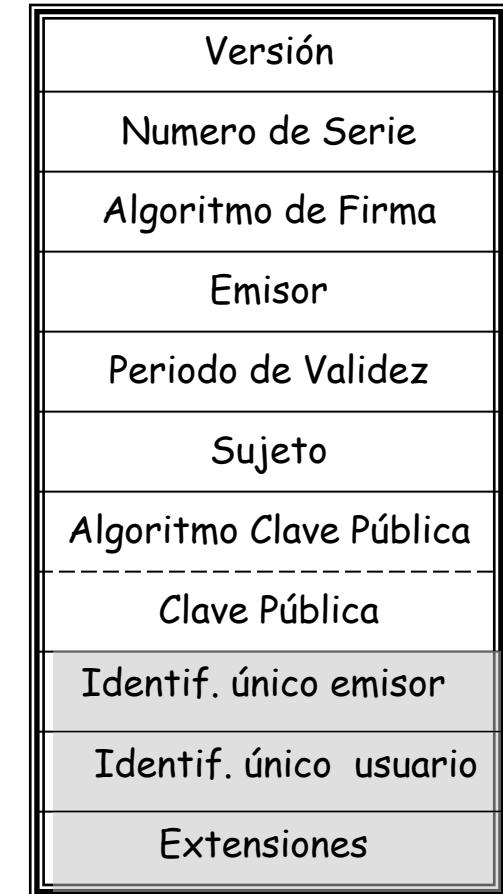
Información de la clave pública

Algoritmo	Encriptación RSA (1.2.840.113549.1.1.1)
Parámetros	ninguno/a
Clave pública	128 bytes: BF F9 49 27 D5 E6 29 D5 ...
Exponente	65537
Tamaño de la clave	1024 bits
Uso de la clave	Cualquiera
Firma	128 bytes: 98 43 60 F8 B4 C4 D7 E1 ...

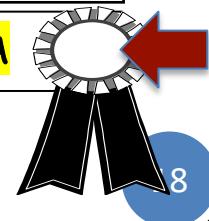
Extensión	Restricciones básicas (2.5.29.19)
Crítico	NO
Entidad de certificación	
Extensión	Identificador de clave del sujeto (2.5.29.14)
Crítico	NO
Nombre de la clave	24 85 28 DC 50 C1 BD 39 5D D0 D5 72 A5 09 1B F4 73 9D AF 7F
Extensión	Identificador de clave de entidad emisora (2.5.29.35)
Crítico	NO
Nombre de la clave	24 85 28 DC 50 C1 BD 39 5D D0 D5 72 A5 09 1B F4 73 9D AF 7F
Nombre de directorio	
País	SP
Estado/Provincia	Malaga
Localidad	Malaga
Empresa	UMA
Unidad organizativa	UMA
Nombre común	Cristina
Dirección de correo	ab@lcc.uma.es
Número de serie	00 D9 AE F5 9B 24 2D 04 FE

Huellas digitales

SHA1 8B 5F 16 E7 64 66 15 9C 89 F3 C1 13 44 94 44 A0 69 75 8F 90
MD5 D6 DB ED 1D 4B DC B3 42 17 31 78 D7 70 8E 0A 96

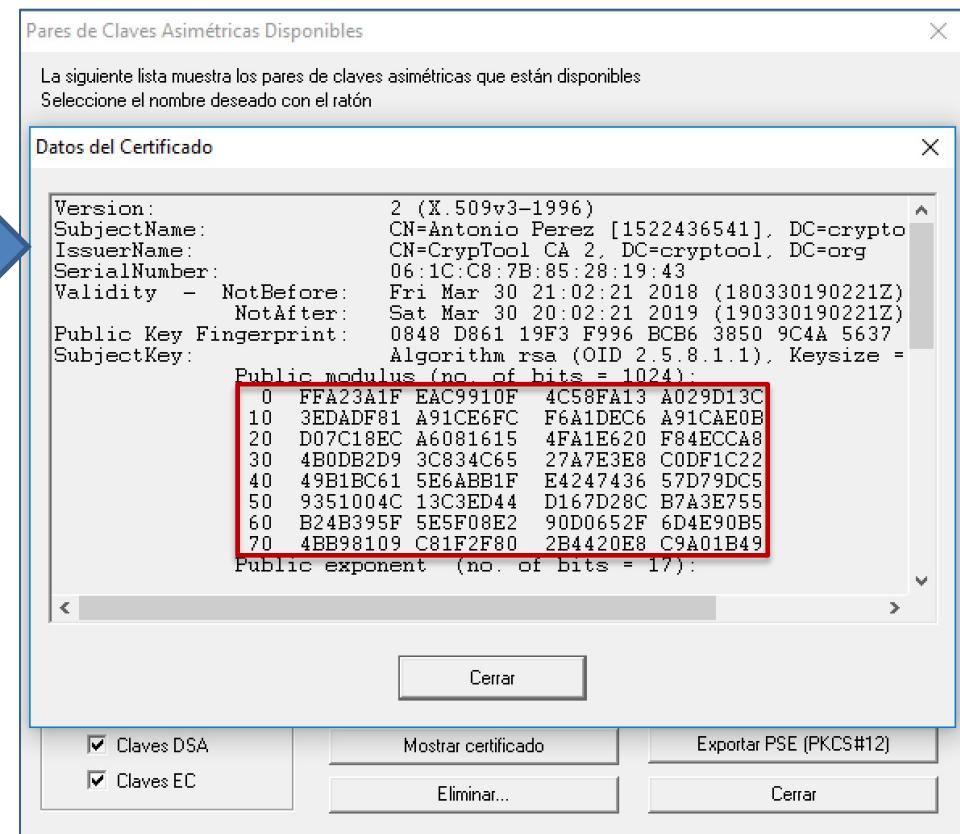
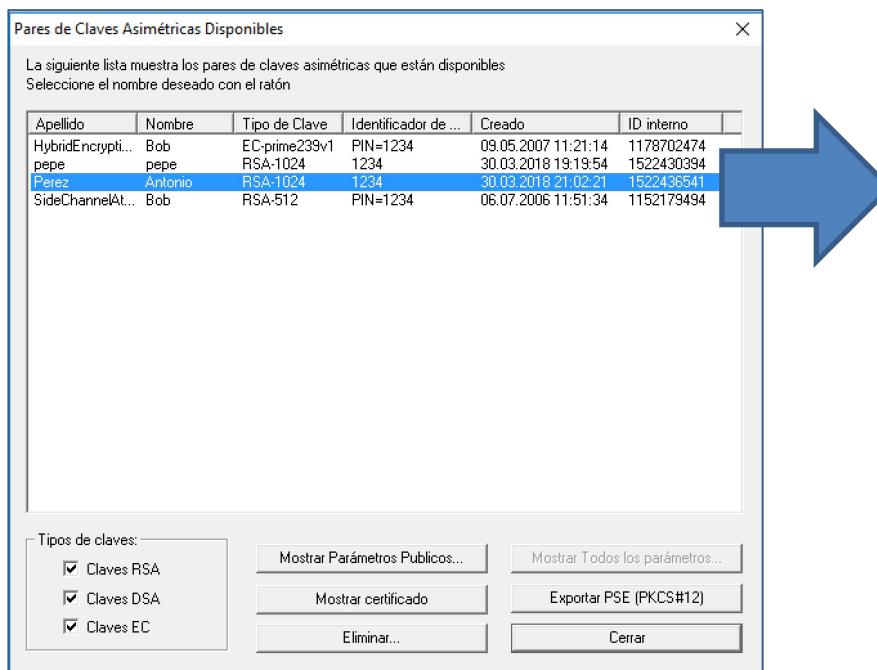


Firma de la CA



Uso de claves en los certificados digitales

- Cuando generamos certificados debemos exportarlos en formato PKCS#12:
 - incluye la pareja de claves para otras aplicaciones, como el correo electrónico
 - Recordar que el PKCS#12 (RFC7292) suele contener un certificado y su correspondiente **clave privada** (en formato DER), la cual puede estar protegida con contraseña

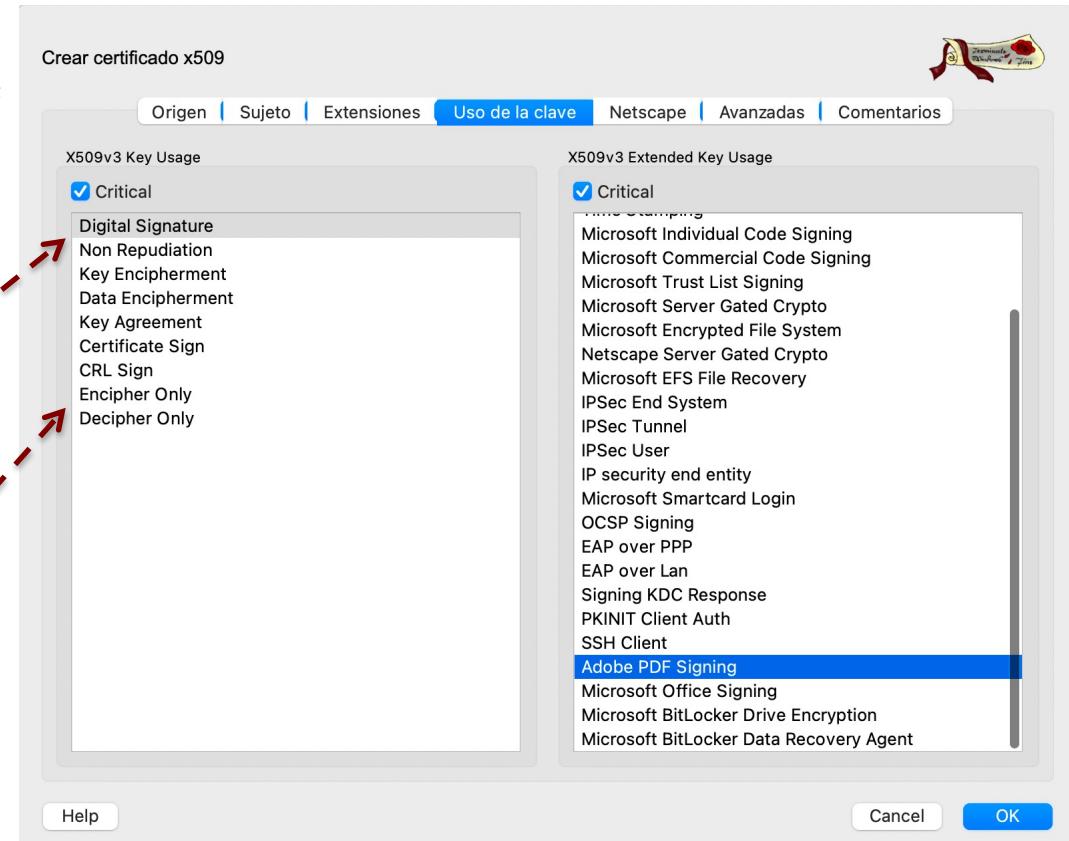


Uso de claves en los certificados digitales

- Cuando una CA tiene que crear un certificado, esta debe indicar expresamente para qué se utilizan las claves asociadas a dicho certificado

Necesario para firmar

Necesario para cifrar

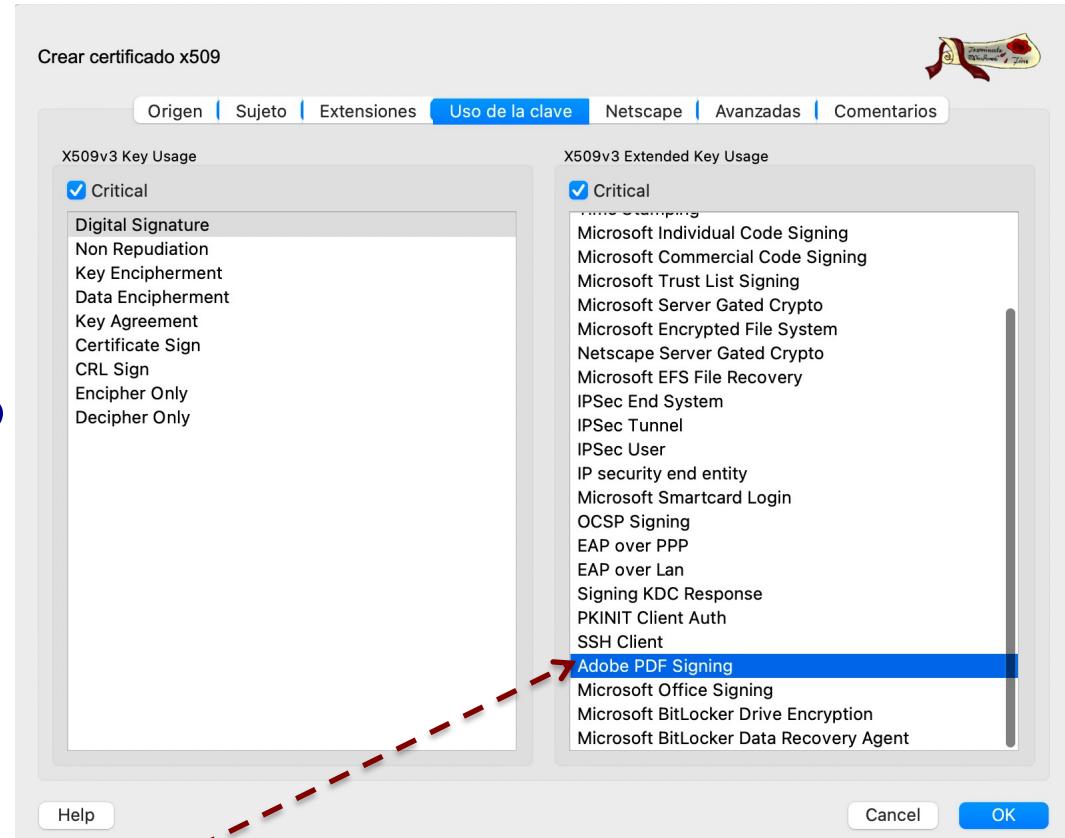


Con critical lo que hacemos es forzar el proceso

Uso de claves en los certificados digitales

- Cuando una CA tiene que crear un certificado, esta debe indicar expresamente para qué se utilizan las claves asociadas a dicho certificado
- Pero también para qué tipo de aplicaciones se va a aplicar dichas claves (o su uso)
 - Ojo que esto es una extensión

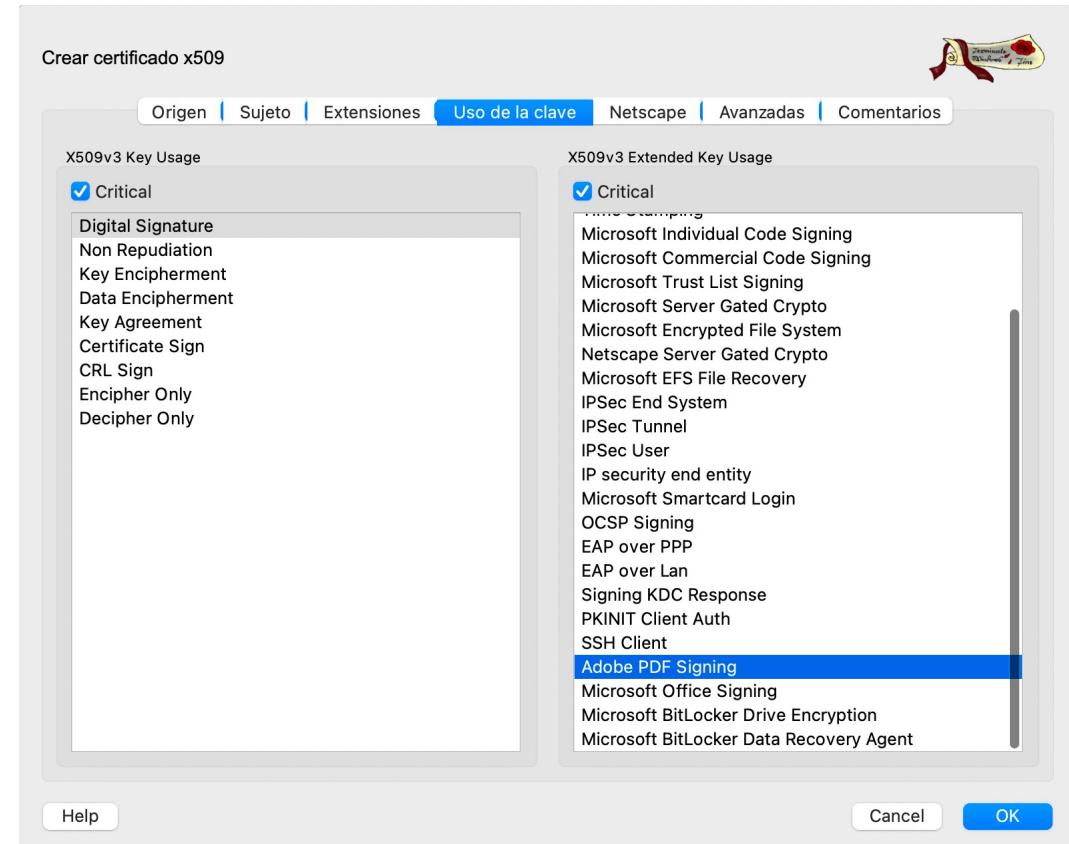
Por ejemplo, para firmar documentos PDF



Con critical lo que hacemos es forzar el proceso

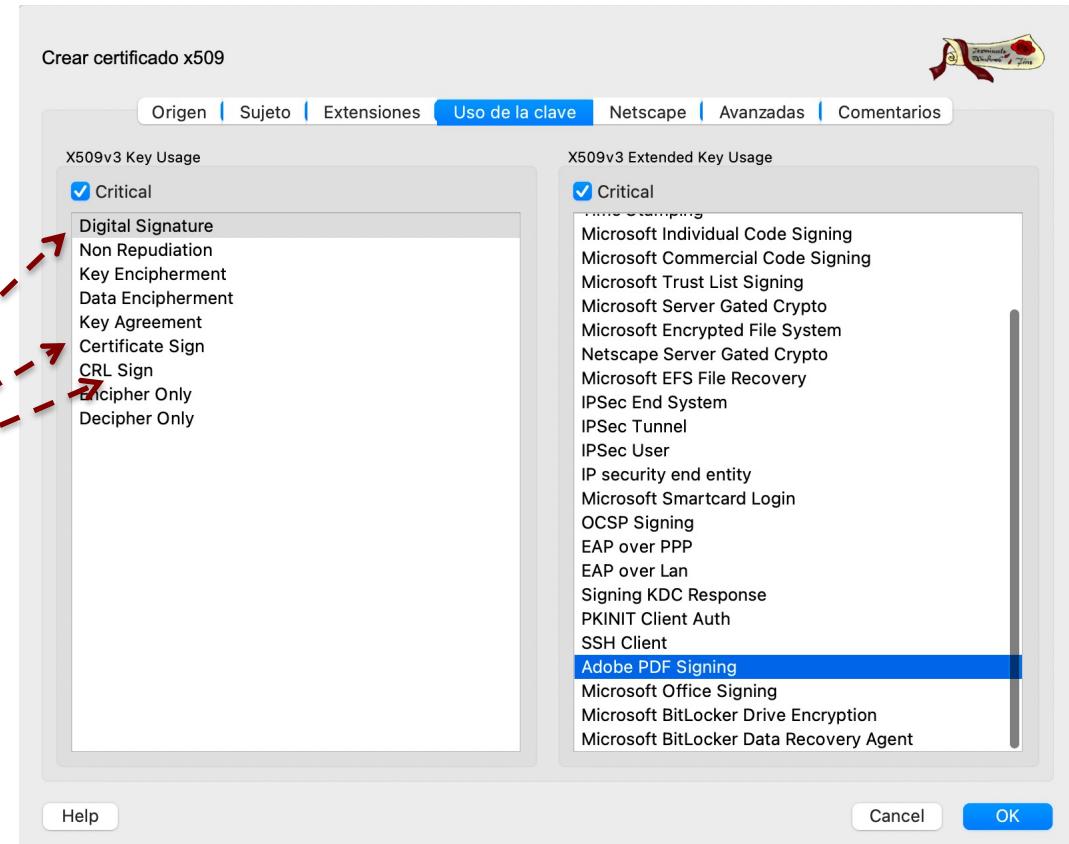
Uso de claves en los certificados digitales

- Supongamos que somos una entidad CA certificando a otra CA
 - ¿Qué tipo de uso de clave se le debería asignar a la nueva CA por defecto?



Uso de claves en los certificados digitales

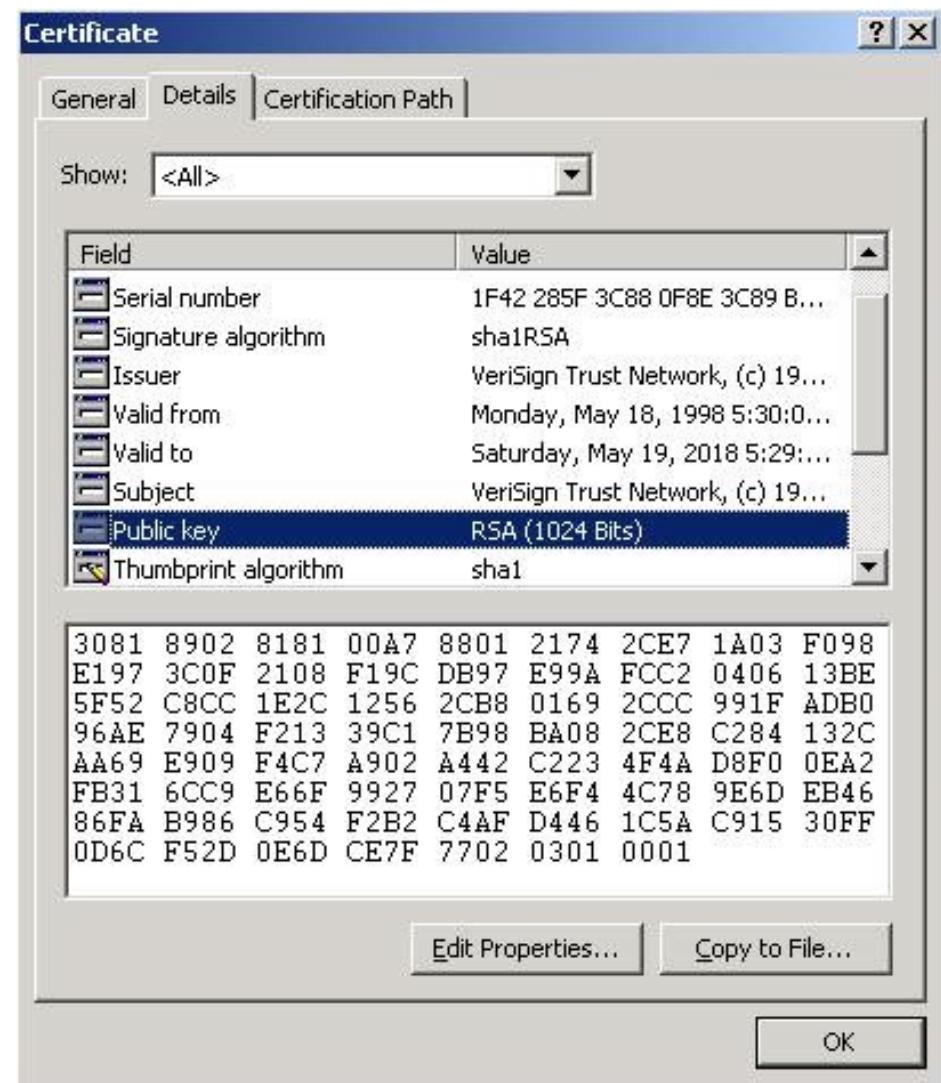
- Supongamos que somos una entidad CA certificando a otra CA
 - ¿Qué tipo de uso de clave se le debería asignar a la nueva CA por defecto?



LUEGO TODA ESTA INFORMACIÓN TAMBIÉN ESTÁ EN EL CERTIFICADO

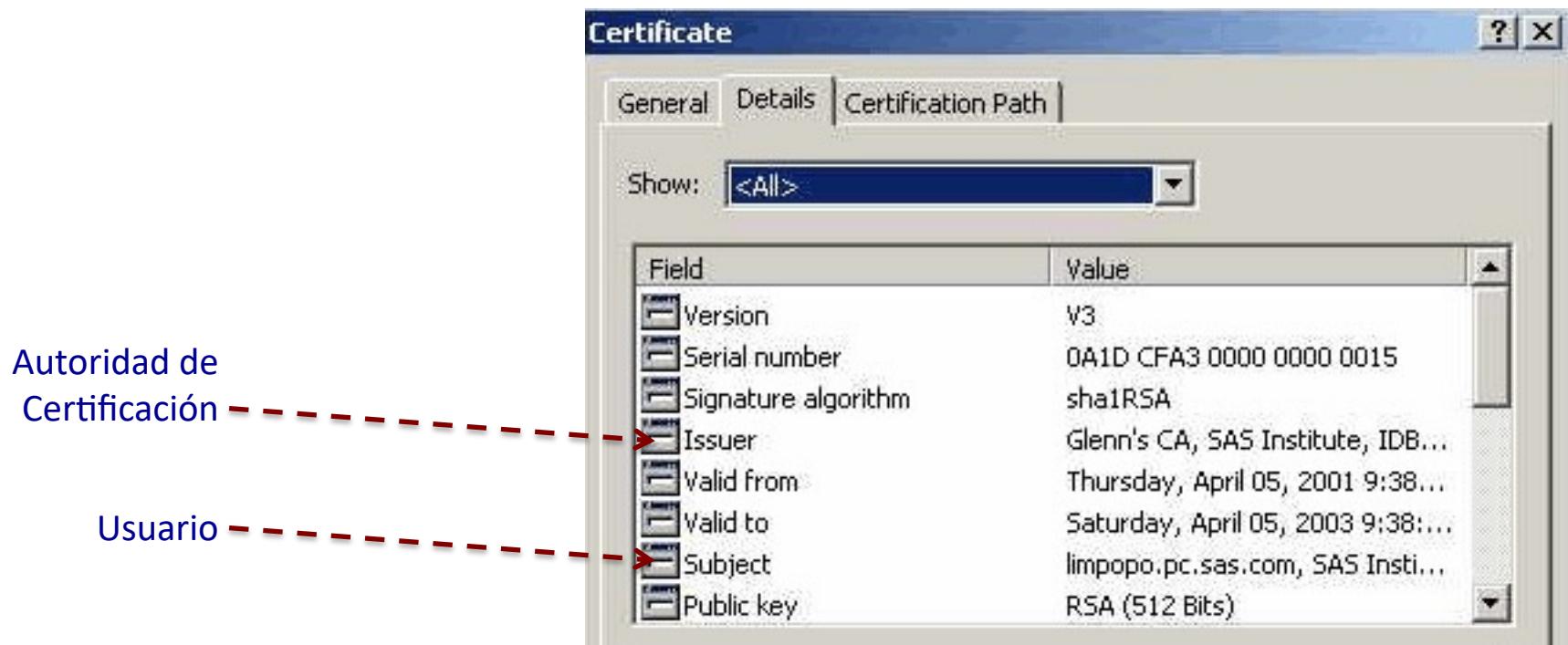
Certificados digitales en navegadores

- Ejemplo de certificado X.509 instalado en un navegador



Certificados digitales en navegadores

- Ejemplo de certificado X.509 instalado en un navegador:



Certificados digitales en navegadores

- Ejemplo de certificado X.509 de una página HTTPS:

Información de la página - https://www.amazon.es/

General Medios Permisos Seguridad

Identidad del sitio web

Sitio web: www.amazon.es
Propietario: Este sitio web no proporciona información sobre su dueño.
Verificado por: DigiCert Inc
Expira el: viernes, 29 de marzo de 2019

Ver certificado

Privacidad e historial

¿Se ha visitado este sitio web anteriormente? Sí, 11 veces
¿Este sitio web almacena información en mi ordenador? Sí, cookies Limpiar cookies y datos del sitio
¿Se han guardado contraseñas de este sitio web? No Ver contraseñas guardadas

Detalles técnicos

Conexión cifrada (TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, claves de 128 bits, TLS 1.2)
La página que está viendo fue cifrada antes de transmitirse por Internet.
El cifrado dificulta que personas no autorizadas vean la información que viaja entre sistemas. Es, por tanto, improbable que nadie lea esta página mientras viajó por la red.

Ayuda

Visor de certificados: "www.amazon.es"

General Detalles

Este certificado ha sido verificado para los siguientes usos:
Certificado del cliente SSL
Certificado del servidor SSL

Emisor para
Nombre común (CN) www.amazon.es
Organización (O) Amazon.com, Inc.
Unidad organizativa (OU) <No es parte de un certificado>
Número de serie 06:43:8B:6B:C1:E3:7E:02:FA:FC:4A:1E:25:8F:BD:0D

Emisor por
Nombre común (CN) DigiCert Global CA G2
Organización (O) DigiCert Inc
Unidad organizativa (OU) <No es parte de un certificado>

Periodo de validez
Comienza el miércoles, 28 de marzo de 2018
Caduca el viernes, 29 de marzo de 2019

Huellas digitales
Huella digital SHA-256 C1:91:78:29:7A:45:76:82:AF:2E:CD:A3:A2:DA:9C:B4:
ED:98:B9:1C:65:87:25:6F:1A:67:FA:AD:59:7C:BB:15
Huella digital SHA1 19:14:24:90:97:C2:77:F1:F1:F4:8B:D2:27:F0:8B:64:3C:FC:3C:3E

Cerrar

Usuario

Autoridad de Certificación



Certificados digitales



<http://www.cacert.org>

¿Nuevo en CACert?

CACert.org es una autoridad certificadora dirigida por la comunidad que emite certificados gratuitos al público.

El objetivo de CACert es promover el conocimiento y la educación sobre la seguridad informática a través del cifrado, ofreciendo específicamente certificados criptográficos. Estos certificados se pueden utilizar para firmar digitalmente y cifrar mensajes de correo electrónico, autenticar y autorizar usuarios que se conectan a sitios web y asegurar la transmisión de datos a través de Internet. Cualquier aplicación que tenga soporte del protocolo Secure Socket Layer (SSL o TLS) puede hacer uso de los certificados firmados por CACert, así como cualquier aplicación que utilice certificados X.509, por ejemplo para el cifrado o firmado de código y las firmas digitales en documentos.

Si desea obtener certificados gratuito emitidos en su nombre, [úñase a la Comunidad CACert](#).

Si desea utilizar los certificados emitidos por CACert, lea la CACert [Root Distribution License](#). Esta licencia se aplica cuando se utilizan [claves raíz](#) de CACert.

ÚLTIMAS NOTICIAS

Accréditation à / Assurance in Paris

Le prochain rendez-vous mensuel à Paris à lieu le mardi 21 mars 2017 entre 19:00 heures et 20:00 heures. Nous vous proposons une rencontre pour toutes personnes intéressées par CACert. Validation, certification, accréditation de vos identités et informations sur CACert. Bar de l'Hôtel Novotel Les Halles 8, place Marguerite de Navarre Paris 1er, Mo Châtelet. Pour [...]

CACert 2017

February brought the start of the exhibition season for CACert with our presence at FOSDEM – one of the biggest Europe-wide developer conferences in Brussels, Belgium. Of course we performed our well-known assurances, which is very popular at such events, with which CACert safeguards its certificates by checking users' ID documents. This allows us to [...]

CACert and secure-u e.V. present at FOSDEM 2017

CACert and secure-u e.V. will be present at FOSDEM 2017, the Free and Open Source Software Developers' European Meeting in Brussels, on February 4th and February 5th. Booth (Sat + Sun) Keysigning Party If you want to help at our booth, register yourself on our events wiki page for FOSDEM 2017 planning. CU at FOSDEM [...]

[[MYÉs Noticias](#)]

Dirigido a los miembros de la comunidad CACert

¿Ha superado ya la [Prueba de Notario](#) de CACert?

¿Ha leído ya el [Acuerdo de Comunidad](#) de CACert?

Para encontrar la documentación general y ayuda visite el sitio de [Documentación Wiki](#) de CACert. Para leer acerca de directrices específica, lea la [página de Directrices Aprobadas](#) de CACert.

[Alta en CACert.org](#)
[Darse de alta](#)
[Acuerdo de la comunidad](#)
[Certificado raíz](#)

Mi cuenta
[Iniciar sesión con contraseña](#)
[Contraseña olvidada](#)
[Iniciar sesión con Net Cafe](#)
[Iniciar sesión con certificado](#)

+ Acerca de CACert.org

+ Traducciones

Publicidad

Certificados digitales



<https://letsencrypt.org/>

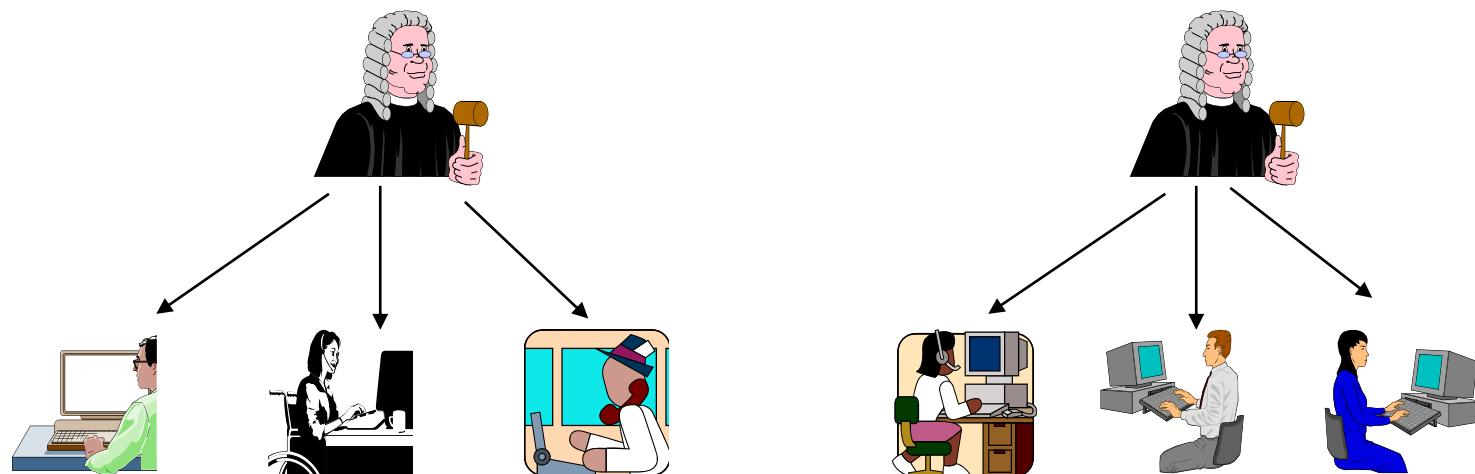
The screenshot shows the official website for Let's Encrypt. At the top left is the Let's Encrypt logo, which consists of a stylized orange lock icon above the text "Let's Encrypt". To the right of the logo is a navigation bar with links: Documentation, Get Help, Donate, About Us, and Languages. Above the navigation bar, there is a small text "LINUX FOUNDATION COLLABORATIVE PROJECTS" next to a square icon. The main content area features a large, semi-transparent white box containing text about Let's Encrypt's mission. Below this box are two blue rectangular buttons with white text: "Get Started" and "Sponsor". The background of the page has a geometric pattern of overlapping triangles in shades of blue, green, and orange.

Let's Encrypt is a **free**, **automated**, and **open** Certificate Authority.

[Get Started](#) [Sponsor](#)

Cadena de confianza

- La situación ideal sería que una única CA pudiera certificar a todos los usuarios de Internet
- Sin embargo, la situación real es bien distinta, dado que existe una gran multiplicidad de grupos de usuarios en Internet, y distribuidos geográficamente, lo que implica la necesidad de **múltiples CAs**



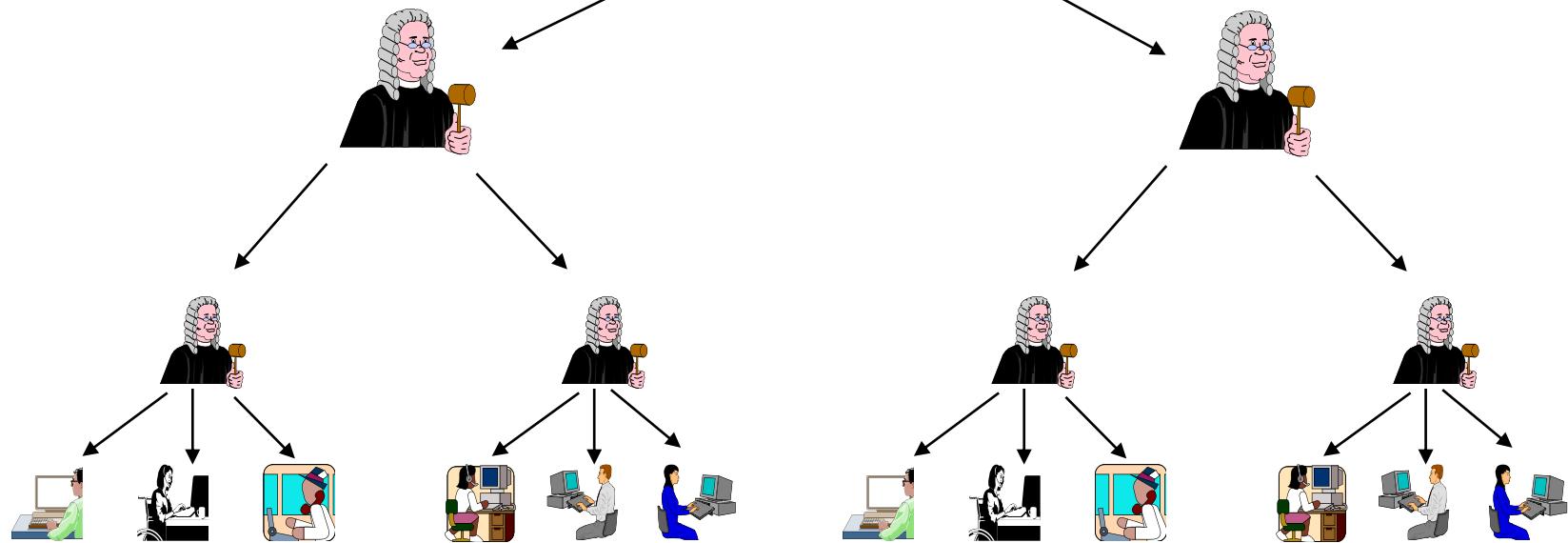
Cadena de confianza



VeriSign®

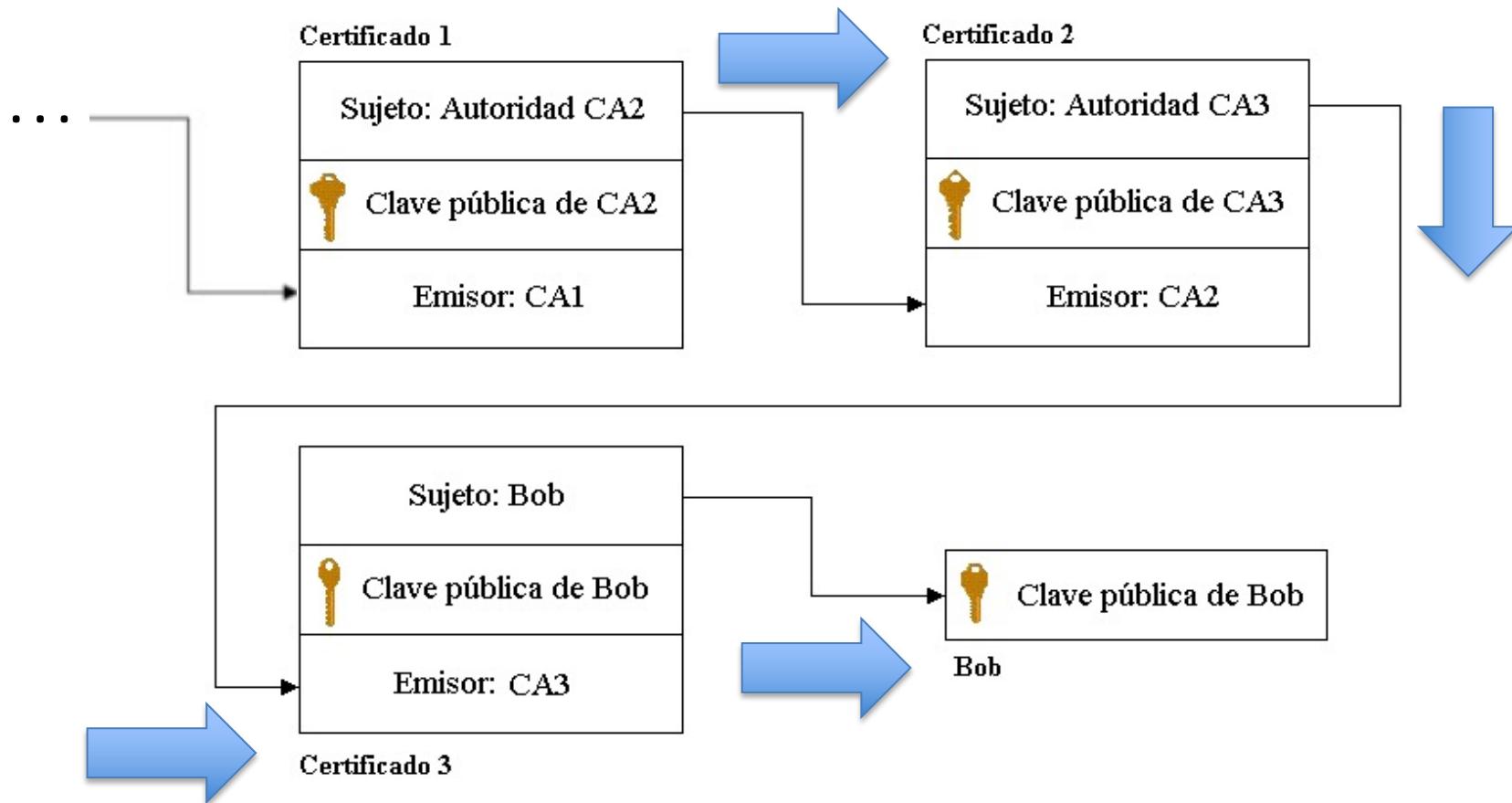


cAcert

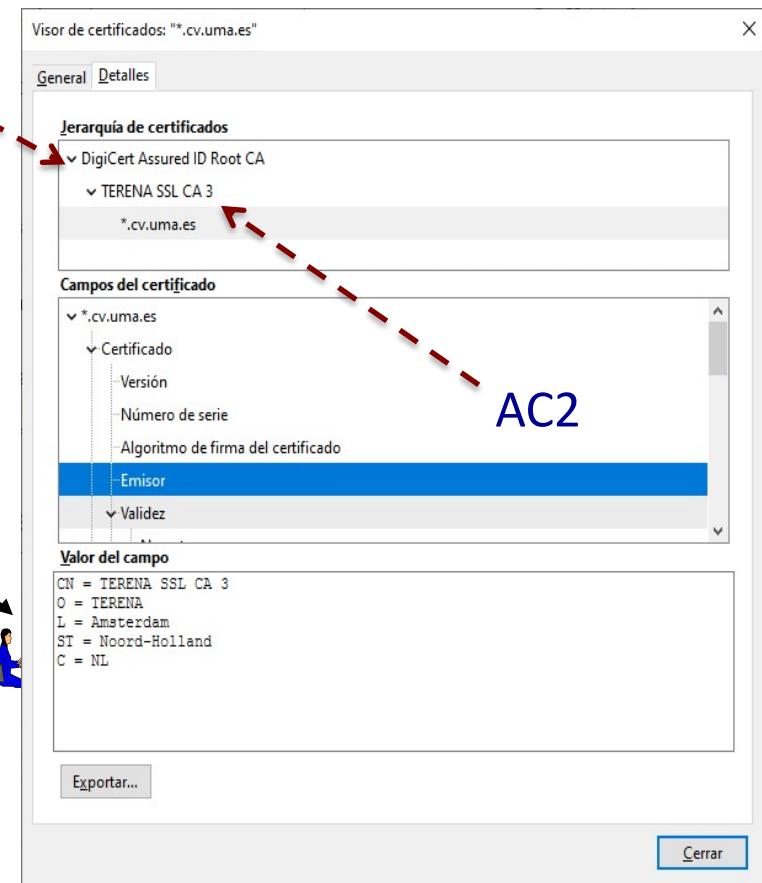
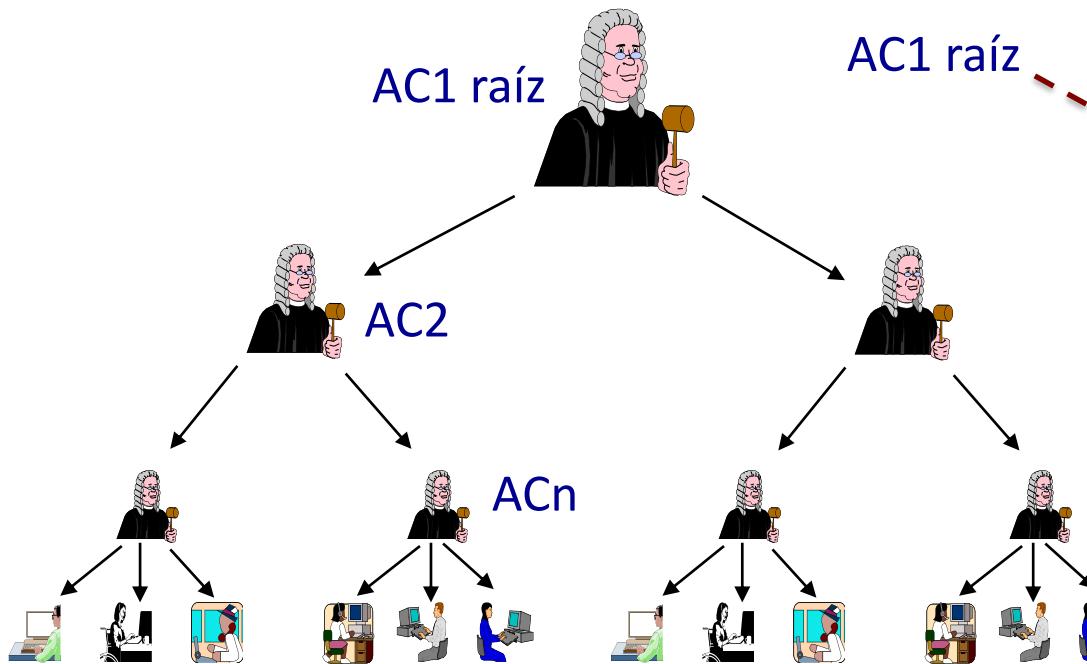


Cadena de confianza

- Entre las CAs se utiliza de forma recursiva el esquema de certificación, creándose las **cadenas de confianza** (o **caminos de certificación**)

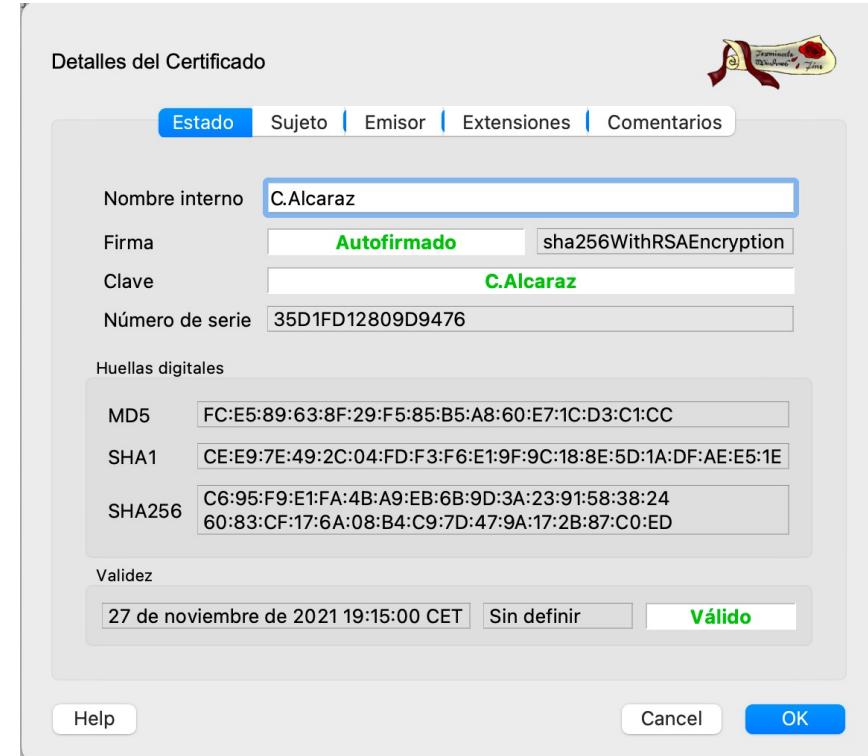


Creando en este caso una **cadena de confianza**, donde una autoridad firma el certificado digital firmado por otra autoridad



Tipos de certificados

- La jerarquía que establece entonces la cadena de confianza da lugar a dos tipos de certificados:
 - Certificado firmado por una CA
 - Certificado autofirmado



Infraestructura de Clave Pública

- Esas cadenas de confianza se forman gracias a la infraestructura de CAs, denominada **Infraestructura de Clave Pública (PKI – Public Key Infrastructure)**
 - Una PKI proporciona el marco subyacente que permite la implantación de la tecnología de clave pública
 - Servicios ofrecidos por una PKI:
 1. Emisión de Certificados
 2. Distribución de Certificados
 3. Obtención de Certificados
 4. Certificación Cruzada
 5. Generación de Claves
 6. Actualización de Claves
 7. Salvaguarda y Recuperación de Claves
 8. Revocación y Suspensión de Certificados



Revocación de certificados

- Revocación de certificados
 - Puede ser recomendable **invalidar** (revocar) un certificado antes de la fecha de expiración cuando:
 - la clave pública deja de ser válida
 - el usuario identificado en el certificado no se considera por más tiempo un usuario con potestad sobre la clave privada correspondiente
 - varía la información dentro del certificado
 - La **CA** se encarga de realizar la revocación, bajo petición del usuario
 - ha de **publicar** esa información acerca del estado del certificado para que el resto de usuarios puedan realizar la comprobación antes de usarlo
 - La comunidad Internet y la ITU-T han desarrollado el concepto de ***Lista de Revocación de Certificados, CRL***, como mecanismo de revocación
 - Una CRL es una lista (con timestamping) de certificados revocados, **firmada por la autoridad que emitió los certificados**

Revocación de certificados

- Escenario típico de uso:
 - Para que *Bob* verifique la firma de *Alice* sobre un documento digital, no sólo ha de verificar el certificado de *Alice* y su validez, además, ha de comprobar que ese certificado no está en la CRL
 - o sea, ha de adquirir la versión más reciente de la CRL y confirmar que el número de serie del certificado de *Alice* no está en tal CRL
- Una CA emite CRLs regularmente (cada hora, día, semana,...) con independencia de que se hayan producido nuevas revocaciones
- El intervalo de emisión de CRLs depende de la política de certificación de la CA
- El certificado **se borra de la CRL cuando alcanza la fecha de expiración** (o sea, cuando se hubiese producido su caducidad natural)

Revocación de certificados

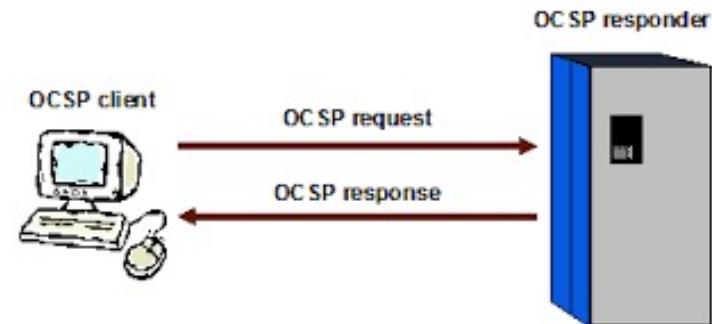
- Estructura de una CRL, según el estándar X.509:



Revocación de certificados

- El protocolo ***Online Certificate Status Protocol (OCSP)*** es otra solución de revocación (RFC 6960)

- Define un formato estándar para mensajes de **peticiones y respuestas**



- Su funcionamiento se basa en que un **usuario puede confirmar on-line el estado de un certificado** al servidor (OCSP Responder) asociado a la CA
 - La CA debe poner a disposición de todos los usuarios potenciales un **servicio online de alta disponibilidad**, y además, el servicio ha de proporcionarse dentro de un entorno seguro
 - El servidor OCSP puede ser o bien la misma CA o alguna entidad autorizada por ella

Tarjetas inteligentes

- Una **tarjeta inteligente** o *smartcard* es una tarjeta que incluye un chip cuya función puede ser variada:
 - desde **simplemente almacenar** cierta información en su memoria interna (con o sin medidas de protección) ...
 - ... hasta realizar **complejos cálculos criptográficos** y encargarse de proteger el acceso a las claves que almacena
- Su uso se extiende hoy a muchos sectores:
 - tarjetas de fidelización
 - tarjetas bancarias
 - tarjetas de parking
 - documentos de identificación (DNI electrónico o pasaporte electrónico)
 - etc.



...¿Cómo clasificamos estas tarjetas?

- Si atendemos al **método de comunicación o interfaz** con el circuito integrado, las smartcards se clasifican en:
 - **Tarjetas de contacto**
 - **Tarjetas sin contacto**
- Si atendemos a las **capacidades del chip**, se clasifican en:
 - **Tarjetas de memoria:** sólo contienen datos y no albergan aplicaciones
 - Uso: identificación y control de acceso sin altos requisitos de seguridad
 - **Tarjetas microprocesadas:** albergan datos y aplicaciones
 - Uso: pago con monederos electrónicos
 - **Tarjetas criptográficas:** tarjetas microprocesadas avanzadas que incluyen módulos hardware para la ejecución de cifrados y firmas digitales
 - Uso: puede almacenar de forma segura un certificado digital (y su clave privada), así como firmar documentos o autenticarse
 - El procesador de la tarjeta es el que realiza la firma



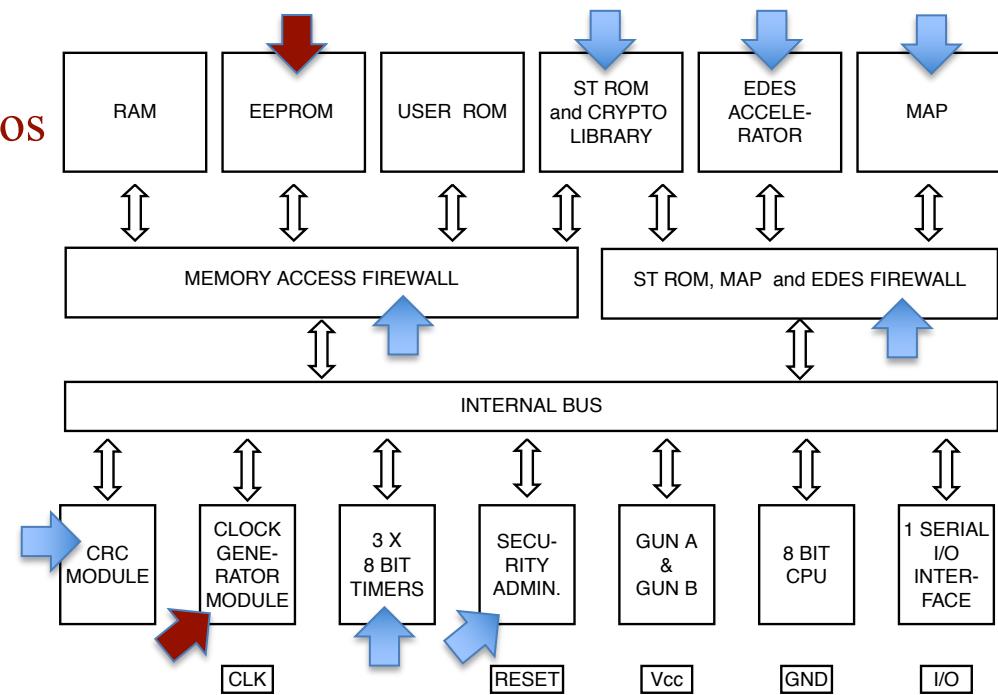
DNI Electrónico (DNI-e)

- El DNI electrónico, a través de las capacidades criptográficas que aporta, permite:
 - **identificación en medios telemáticos**
 - **firmar electrónicamente**
- El DNI-e está dotado con el chip ST19WL34 (STMicroelectronics), compuesto por:
 - microprocesador securizado de 8 bits
 - 6 Kb de memoria RAM
 - 224 KB de memoria ROM para el almacenamiento del sistema operativo y código de programas
 - 34 KB de memoria EEPROM para el almacenamiento de datos personales con tecnología de almacenamiento fiable y código de corrección de errores
- El chip ofrece una retención de datos de al menos 10 años, y una resistencia de 500.000 ciclos de borrado y escritura



DNI Electrónico (DNI-e)

- Este chip se caracteriza por incorporar también:
 - procesador aritmético modular (MAP) de 1088 bits para **criptografía de clave pública**
 - motor de aceleración por hardware de los **algoritmos DES y triple-DES**
 - módulo para el cálculo de **funciones CRC**
 - interfaz de entrada/salida serie
 - generador de números aleatorios
 - bus de interconexión interno
 - 3 timers de 8 bits
 - reloj interno



DNI Electrónico (DNI-e)

- La tabla muestra algunas medidas de tiempo de la ejecución de operaciones criptográficas

1. Typical values, independent from external clock frequency and supply voltage.
2. CRT: Chinese Remainder Theorem.

Function	Speed ⁽¹⁾
RSA 1024 bits signature with CRT ⁽²⁾	85 ms
RSA 1024 bits signature without CRT ⁽²⁾	282 ms
RSA 1024 bits verification ($e='\$10001'$)	5.5 ms
RSA 1024 bits key generation	2.5 s
RSA 2048 bits signature with CRT ⁽²⁾	570 ms
RSA 2048 bits verification ($e='\$10001'$)	91 ms
Triple DES (with enhanced security)	58.0 μ s
Single DES (with enhanced security)	43.0 μ s



- El sistema operativo que gestiona el chip se denomina **DNIe v3.0**, desarrollado por la FNMT a partir de las especificaciones funcionales de la Dirección General de Policía
 - este sistema operativo ha sido sometido con posterioridad a los perfiles de protección de la certificación *Common Criteria*



DNI Electrónico (DNI-e)

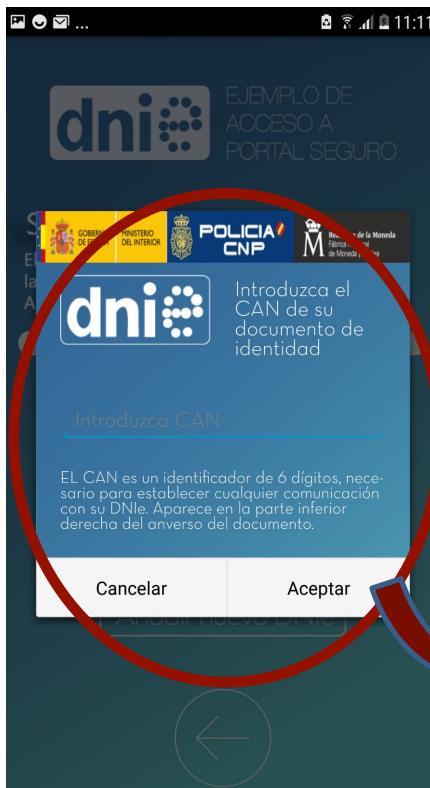
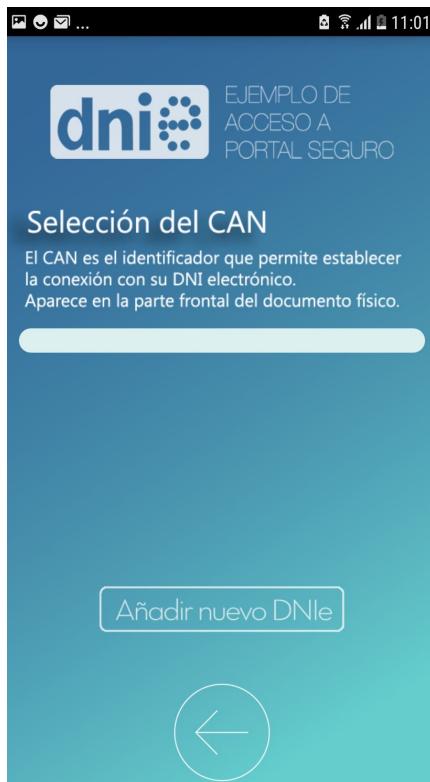
- Dos opciones para acceder al chip



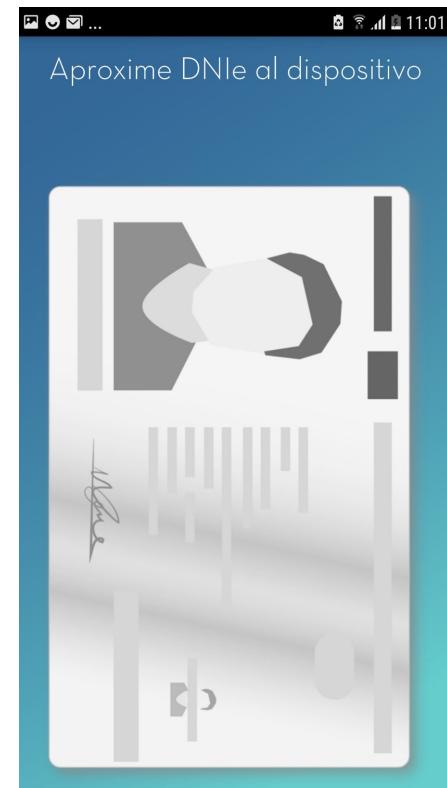
Big buck bunny image: <http://www.bigbuckbunny.org/index.php/about/>
Creative Commons CC by 3.0



Leer DNI-e 3.0 en Android - con NFC (Near-Field Comm.)



CAN (Card Access Number) es un número que aparece en la parte inferior del DNI 3.0, y corresponde con el número de la tarjeta como medida de seguridad



Componentes del DNI-e

- El DNI-e contiene dos certificados digitales asociados al titular:
 - **certificado de autenticación**: asegura que la comunicación electrónica se realiza con el titular del DNI, pero no demuestra voluntad de firma
 - restringido a operaciones para confirmar la identidad y acceso seguro a sistemas remotos
 - **certificado de firma digital**: para la firma de documentos, garantizando la integridad del documento y el no repudio de origen
- Cuenta también con un **certificado de componente**, emitido para autenticar al propio chip y cifrar la comunicación con él
 - de forma similar a como se utiliza un certificado TLS en un servidor Web
- El generador interno de números aleatorios origina el par de claves de cada certificado, en presencia del ciudadano:
 - se garantiza que sólo existirá una copia de cada clave privada, y que ésta residirá siempre en el interior del chip

Componentes del DNI-e

- La información de la **memoria EEPROM** del chip está distribuida en tres zonas, con diferentes niveles y condiciones de acceso
- Las tres son sólo accesibles para realizar **operaciones de lectura**, no siendo posible para el ciudadano escribir o grabar datos
 - **zona pública**: accesible sin restricciones
 - certificado CA emisora
 - claves Diffie-Hellman
 - certificado X.509 de componente
 - **zona privada**: accesible por el ciudadano mediante la utilización de su PIN
 - certificado de autenticación (identificación)
 - certificado de firma
 - **zona de seguridad**: accesible por el ciudadano de forma exclusiva en los puntos de actualización del DNI-e (en las comisarías)
 - datos de filiación del ciudadano
 - fotografía del titular
 - imagen de la firma manuscrita

Policía Nacional

CUERPO NACIONAL DE POLICÍA

GOBIERNO DE ESPAÑA MINISTERIO DEL INTERIOR DIRECCIÓN GENERAL DE LA POLICÍA

DNI y Pasaporte

Cuerpo Nacional de

DNI electrónico

- Obtención del DNI
- Cómo utilizar el DNI
- Guía de referencia básica
- Certificados Electrónicos**
 - » Qué son los certificados electrónicos
 - » Renovación de Certificados
 - » Aceptación de los Certificados
 - » Autoridades de validación
 - » Política de certificación
- Marco legal
- Glosario
- Atención al Ciudadano
- Preguntas más frecuentes
- Recursos

PASAPORTE

POLICIA NACIONAL

sede electrónica Cuerpo Nacional de Policía

[Inicio](#) / Certificados Electrónicos / Qué son los Certificados Electrónicos

Renovación Certificados

Renovación de claves sin renovación del soporte físico (tarjeta):

La renovación de las claves es voluntaria, gratuita y por iniciativa del ciudadano.

En fechas próximas a la caducidad de sus certificados, recibirá, en la cuenta de correo electrónico momento de la expedición de su DNI, un aviso procedente de la dirección oficial notificaciones@policiacanaria.es informando de la fecha de caducidad de sus certificados electrónicos.

El titular puede proceder a renovar los certificados, si el estado de los mismos es uno de los siguientes:

- Si fueron revocados a petición del ciudadano (solo podrá revocarse el certificado de firma digital)
- Por caducidad. Los certificados caducan pasados 60 meses desde la emisión de los mismos o si la fecha de caducidad es inferior a esos 60 meses, limitación a la fecha de caducidad del mismo (mejora de notoriedad impidiendo que se limitaba a 30 meses y solo se podían renovar una vez caducados o dentro de los 30 días de la fecha de caducidad).

Para proceder a la renovación deberá mediar la presencia física del titular en una Oficina de expedición. El ciudadano, haciendo uso de los Puntos de Actualización del DNIe 3.0 habilitados en dichas oficinas y previa autenticación mediante la tarjeta y las plantillas biométricas (impresiones dactilares) capturadas durante la expedición de la Tarjeta, podrá desencadenar de forma desatendida el proceso de renovación de sus certificados.

EL PROCESO DE RENOVACIÓN DE CERTIFICADOS EN EL PUNTO DE ACTUALIZACIÓN DEL DNI 3.0 ES EL SIGUIENTE:

- El titular tras introducir correctamente el PIN, accede a la pantalla de "información sobre el contenido de su DNI 3.0", en la parte inferior puede visualizar el estado de sus certificados. En su caso, en la parte izquierda aparece una casilla "renovar certificados". Si se selecciona "renovar certificados" solicita nuevamente el PIN y posteriormente la presentación de la huella dactilar. Si el resultado es positivo se procede a la renovación de los certificados; este proceso dura aproximadamente 3 minutos. Es importante, no retirar el documento del lector de tarjetas hasta la finalización del proceso, porque el DNIe 3.0 podría quedar inservible. Si no fuere posible obtener la impresión dactilar de alguno de los dedos, el ciudadano deberá solicitar la renovación en un puesto de expedición atendido por un funcionario.



CUERPO NACIONAL DE POLICÍA

GOBIERNO DE ESPAÑA MINISTERIO DEL INTERIOR DIRECCIÓN GENERAL DE LA POLICÍA

DNI y Pasaporte Cuerpo Nacional de Policía

Ciudadanos Empresas Administraciones Oficina Técnica

[Inicio](#) / Certificados Electrónicos / Qué son los Certificados Electrónicos

Renovación Certificados

Renovación de claves sin renovación del soporte físico (tarjeta):

La renovación de las claves es voluntaria, gratuita y por iniciativa del ciudadano.

En fechas próximas a la caducidad de sus certificados, recibirá, en la cuenta de correo electrónico que usted haya proporcionado en el momento de la expedición de su DNI, un aviso procedente de la dirección oficial notificaciones@policia.es, en el que le advierten de la próxima caducidad de sus certificados electrónicos.

El titular puede proceder a renovar los certificados, si el estado de los mismos es uno de los siguientes:

- Si fueron revocados a petición del ciudadano (solo podrá revocarse el certificado de firma digital).
- Por caducidad. Los certificados caducan pasados 60 meses desde la emisión de los mismos o si la fecha de caducidad del documento es inferior a esos 60 meses, limitación a la fecha de caducidad del mismo (mejora de notoria importancia, puesto que la anterior regulación los limitaba a 30 meses y solo se podían renovar una vez caducados o dentro de los 30 días de la fecha de caducidad).
- Para proceder a la renovación deberá mediar la presencia física del titular en una Oficina de expedición. El ciudadano, haciendo uso de los Puntos de Actualización del DNIE 3.0 habilitados en dichas oficinas y previa autenticación mediante la tarjeta y las plantillas biométricas (impresiones dactilares) capturadas durante la expedición de la Tarjeta, podrá desencadenar de forma desatendida el proceso de renovación de sus certificados.

EL PROCESO DE RENOVACIÓN DE CERTIFICADOS EN EL PUNTO DE ACTUALIZACIÓN DEL DNI 3.0 ES EL SIGUIENTE:

- El titular tras introducir correctamente el PIN, accede a la pantalla de "información sobre el contenido de su DNI 3.0", en la parte inferior puede visualizar el estado de sus certificados. En su caso, en la parte izquierda aparece una casilla "renovar certificados". Si se selecciona "renovar certificados" solicita nuevamente el PIN y posteriormente la presentación de la huella dactilar. Si el resultado es positivo se procede a la renovación de los certificados; este proceso dura aproximadamente 3 minutos. Es importante, no retirar el documento del lector de tarjetas hasta la finalización del proceso, porque el DNIE 3.0 podría quedar inservible. Si no fuere posible obtener la impresión dactilar de alguno de los dedos, el ciudadano deberá solicitar la renovación en un puesto de expedición atendido por un funcionario.

Claves criptográficas en el DNI-e

- Cualquier operación criptográfica que requiera el uso de una de las claves privadas debe ser ejecutada en el interior del chip
- Las claves públicas se envían, tras su generación en el acto de expedición del DNI-e, a la CA para su inclusión en los correspondientes certificados digitales
 - una vez emitidos los certificados, estos se incorporan a la tarjeta para ser empleados en operaciones posteriores
 - los certificados digitales pueden ser leídos para su proceso de forma externa al chip



Revocación - DNI-e

- En el ámbito del DNI-e se usa **OCSP** para las revocaciones
 - cuando una aplicación requiere el estado actual de un certificado, envía una petición OCSP (mediante HTTP) a la URL del servicio de validación
 - una vez recibida la petición, el *OCSP Responder* accede a las CRLs, y averigua si dicho certificado se encuentra ahí incluido
- En la PKI adoptada para el DNI-e se ha optado por asignar las funciones de Autoridad de Validación a entidades diferentes de la **Autoridad de Certificación**
 - con el fin de aislar la comprobación de la vigencia de un certificado
 - existen tres **Autoridades de Validación**:
 - FNMT
 - Ministerio de Administraciones Públicas
 - Ministerio de Industria



Normativas y leyes del DNI-e

- El marco legal básico del DNI-e es el siguiente:
 - Directiva 1999/93/CE del Parlamento Europeo y del Consejo, de 13 de diciembre, por la que se establece un marco comunitario para la firma electrónica
 - Ley 59/2003, de 19 de diciembre, de Firma Electrónica
 - Ley Orgánica 15/1999, de 13 de diciembre, de Protección de los Datos
 - Real Decreto 1553/2005, de 23 de diciembre, por el que se regula documento nacional de identidad y sus certificados de firma electrónica
 - Real Decreto 1720/2007, de 21 de diciembre, relacionado con la protección de datos de carácter personal
 - Real Decreto 1586/2009, de 16 de octubre, Real Decreto 869/2013, de 8 de noviembre, y Real Decreto 414/2015, de 29 de mayo, por los que se modifica el Real Decreto 1553/2005

MECANISMOS DE AUTENTICACIÓN

Tipos de autenticación de usuarios

- En general, existen tres formas en los que clasificar los mecanismos de autenticación:

- Algo que sólo yo CONOZCO
 - P.ej. Contraseñas



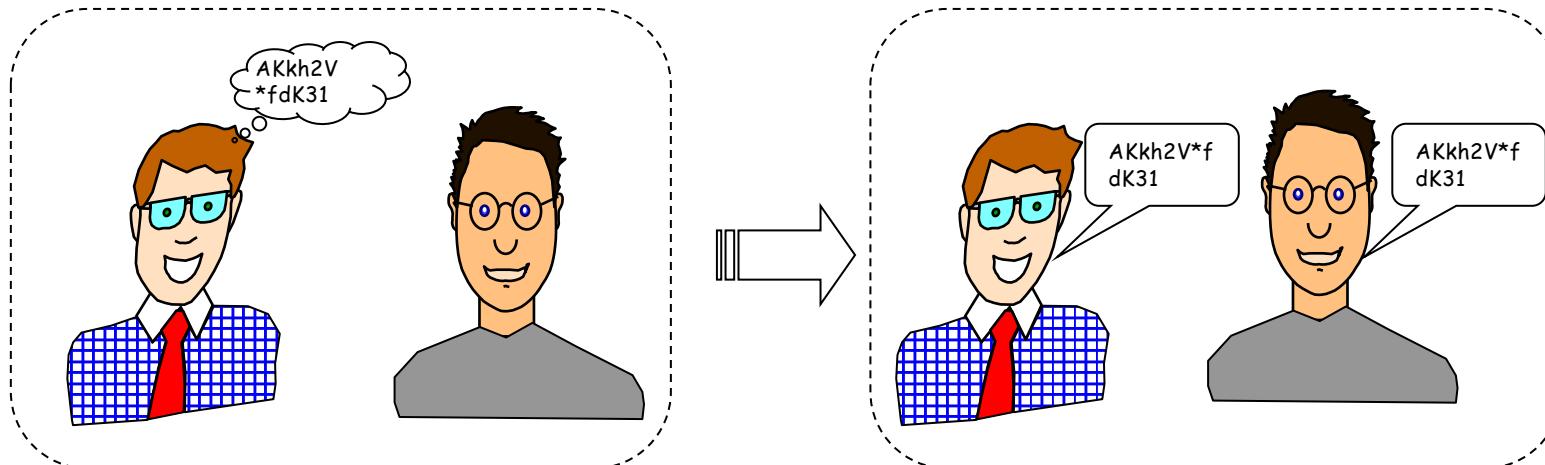
- Algo que sólo yo TENGO
 - P.ej. Claves públicas, Tokens, certificados

- Algo que yo SOY
 - P.ej. Autenticación biométrica



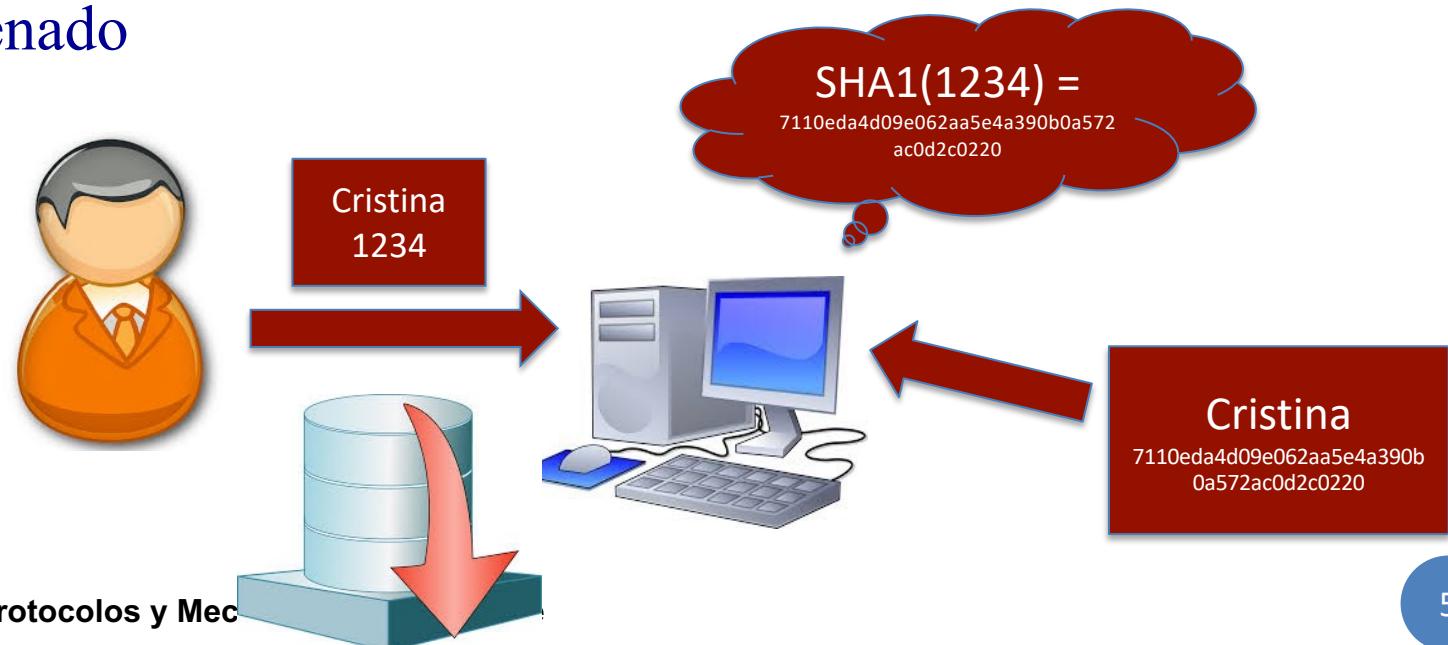
Tipos de autenticación de usuarios - CONOZCO

- En los sistemas basados en contraseñas o “*passwords*”, el usuario conoce cierta información que nadie más conoce
 - Acceso sistema operativo, acceso a servicios web...
- Características principales:
 - Se puede pasar la contraseña de un usuario a otro
 - Más de un usuario puede usarla a la vez



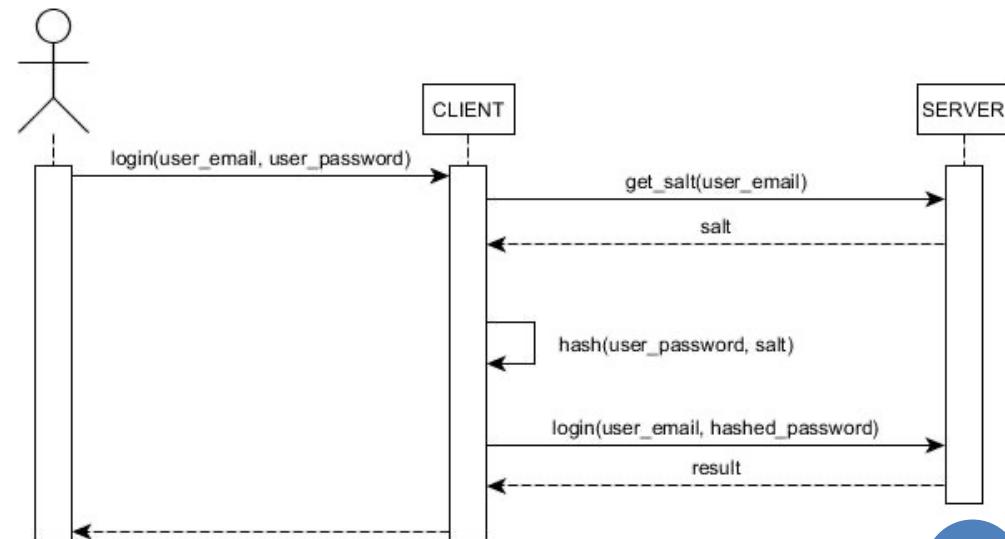
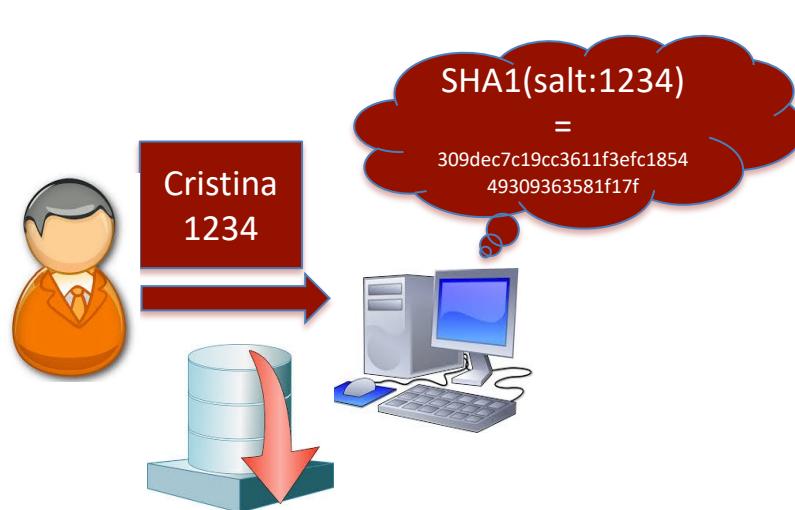
Salt: Contraseñas con Salt

- Las cuentas almacenadas en un disco duro de un sistema y protegidas con contraseñas, suelen tener asociado un HASH a dichas contraseñas.
 - ¡Nunca se debe almacenar las contraseñas en claro!
 - Cuando el usuario quiere entrar al equipo se le pide la contraseña, se hace el hash y se compara con el hash almacenado



Salt: Contraseñas con Salt

- Si alguien intenta hacer un **ataque de diccionario** y preparar un fichero con todas las posibles combinaciones posibles de HASH podría derivar la contraseña inicial
 - A este tipo de ataque se les conoce como “**Rainbow Table**”, y es la forma más eficaz de “romper” contraseñas en bases de datos desprotegidas
- Para dificultar los ataques de diccionario se usa una “**Sal**” → valores aleatorios que se asocia al HASH



Salt: Contraseñas con Salt

- ¿Qué puede hacer ese atacante para tratar de averiguar nuestra contraseña?
 - **CASE 1:** si el atacante lo que tiene es *un listado de hashes de contraseña, no puede hacer nada*
 - No puede comparar $H(1234)$ con $H(\text{Sal} \parallel 1234)$
 - Porque de aquí tiene que intentar extraer la contraseña
 - » pero, ¿con la Sal? → 😞
 - **CASE 2:** si el atacante lo que tiene es *un listado de contraseñas robadas de otros sitios*, puede calcular (para cada usuario) $H(\text{Sal} \parallel \text{Contraseña Robada})$
 - *Sin embargo, no puede saber si dos usuarios tienen la misma contraseña* (la sal hará que los hashes sean distintos)

Salt: Contraseñas con Salt

- Imaginemos que somos un atacante, y que hemos robado una base de datos de usuarios con su sal asociada (en el lado del servidor)
... 😞

Cristina + Sal + BCrypt(Sal || 1234)

Bcrypt es una función hash basada en el algoritmo Blowfish

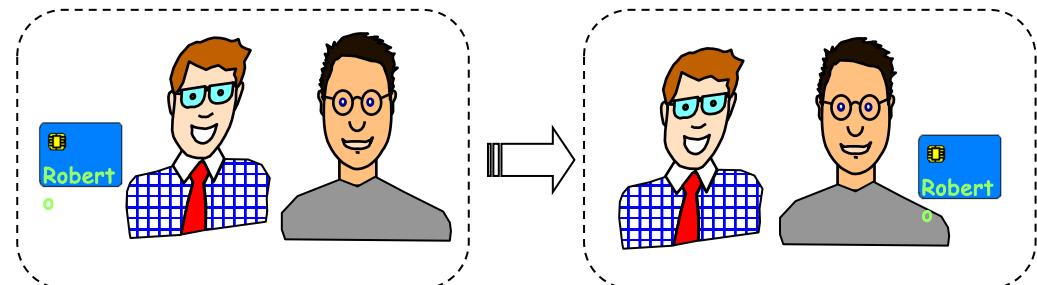
- **bcrypt** es una función hash que combina contraseñas + Salt, y está basada en el cifrado Blowfish
 - Resistente a ataques de fuerza bruta por ser una función hash adaptativa
 - Por implicar el factor tiempo el cual es dependiente de un número de iteraciones que se quieran hacer
 - Cuanto más interacciones ➔ más lento y más resistente

Salt: Contraseñas con Salt

- **LOS PRINCIPIOS DE LA SAL:**
 - NUNCA reutilizar la sal
 - Crear una sal aleatoria por cada contraseña, usando un generador aleatorio criptográfico (p.ej. Os.urandom())
 - Usar una sal **SUFICIENTEMENTE LARGA**
 - /etc/shadow: 10 caracteres [a-z | A-Z | 0-9] ≈ 57 bits
 - Utilizar **FUNCIONES HASH LENTAS**
 - Argon2id, PBKDF2

Tipos de autenticación de usuarios - TENGO

- Existe un secreto guardado en un **token físico**
 - El usuario posee un objeto físico por el que prueba su identidad
 - Los más comunes son las tarjetas (o smartcards)
- Características principales (Tokens físicos):
 - Principalmente basado en **criptografía asimétrica**
 - “Tengo una clave secreta (fichero) en el token físico”, ej. certificados
 - Se podría pasar el token de un usuario a otro, pero depende de cada token
 - Una tarjeta que liste varios usuarios
 - Si es físico, sólo un usuario puede usarlo a la vez



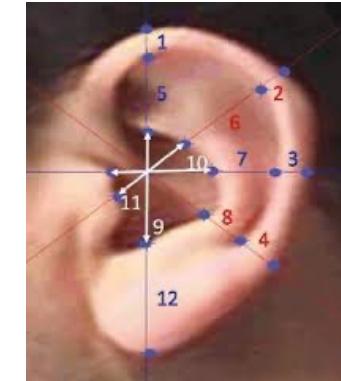
Ejemplo: DNI Electrónico (DNI-e)

- El DNI electrónico, a través de las capacidades criptográficas que aporta, permite:
 - **identificación en medios telemáticos**
 - **firmar electrónicamente**
- El DNI-e contiene dos certificados digitales asociados al titular:
 - **certificado de autenticación**: asegura que la comunicación electrónica se realiza con el titular del DNI, pero no demuestra voluntad de firma
 - restringido a operaciones para confirmar la identidad y acceso seguro a sistemas remotos
 - **certificado de firma**: para la firma de documentos, garantizando la integridad del documento y el no repudio de origen
- El generador interno de números aleatorios origina el par de claves de cada certificado, en presencia del ciudadano:
 - se garantiza que sólo existirá una copia de cada clave privada, y que ésta residirá siempre en el interior del chip



Tipos de autenticación de usuarios - SOY

- En los sistemas basados en datos biométricos se extraen información de las características biológicas del usuario:
 - huella,
 - imagen del iris,
 - tamaño y forma de la oreja,
 - etc.
- Características principales:
 - No se pueden traspasar los datos biométricos de un usuario a otro
 - Sólo el usuario en cuestión puede usarlos en un momento determinado



Tipos de autenticación de usuarios - Inconvenientes

- Inconvenientes del uso de *contraseñas*:
 - Es estrictamente necesario utilizar contraseñas robustas, y no repetirlas
 - Resulta complicado recordar todas las que se usan → ¡demasiadas contraseñas!
 - No siempre se cumple las políticas de seguridad:
 - Tamaño mínimo y alfa-numéricas
 - Actualizar con bastante frecuencia - *rekeying*
- Inconvenientes del uso de *tokens físicos*:
 - Dependiendo del token, no siempre prueba realmente la identidad de los usuarios
 - Cualquiera en posesión del token es autenticado de forma positiva
 - En caso de pérdida o daño, el usuario legítimo se queda sin posibilidad de ser autenticado
 - En ocasiones se pueden falsificar o clonar
- Inconvenientes del uso de *datos biométricos*:
 - El perfil del usuario debe ser almacenado en el ordenador antes de proceder a la autenticación
 - Requieren medidas de protección especiales
 - Estos sistemas son más caros que los otros vistos anteriormente
 - Si un dato biométrico se pierde, se pierde por vida – ej. una huella por quemadura

Autenticación de doble factor

- Mecanismo de autenticación que combina dos de los mecanismos anteriores
 - Ej: contraseña + token
 - Ej: dato biométrico + token
- Ejemplo de implantación legal: **Directiva Europea PSD2**
 - Segunda Directiva de Servicios de Pago por Internet
 - Obliga al uso de autenticación de doble factor (código de la tarjeta + aplicación móvil / mensaje SMS)
- El uso de SMS para la doble autenticación es peligroso
 - Duplicado del SIM del móvil



Autenticación basado en códigos QR

- Los códigos de QR (Quick Response) son otras de las formas para autenticar y autorizar el acceso a usuarios haciendo uso de dispositivos móviles



- Inconvenientes:
 - La información del usuario debe estar en el servidor remoto
 - El servidor requiere la instalación de una aplicación que genere códigos QR
 - Los dispositivos móviles (los clientes) necesitan también instalar alguna aplicación para escanear el código QR proporcionado por el servidor

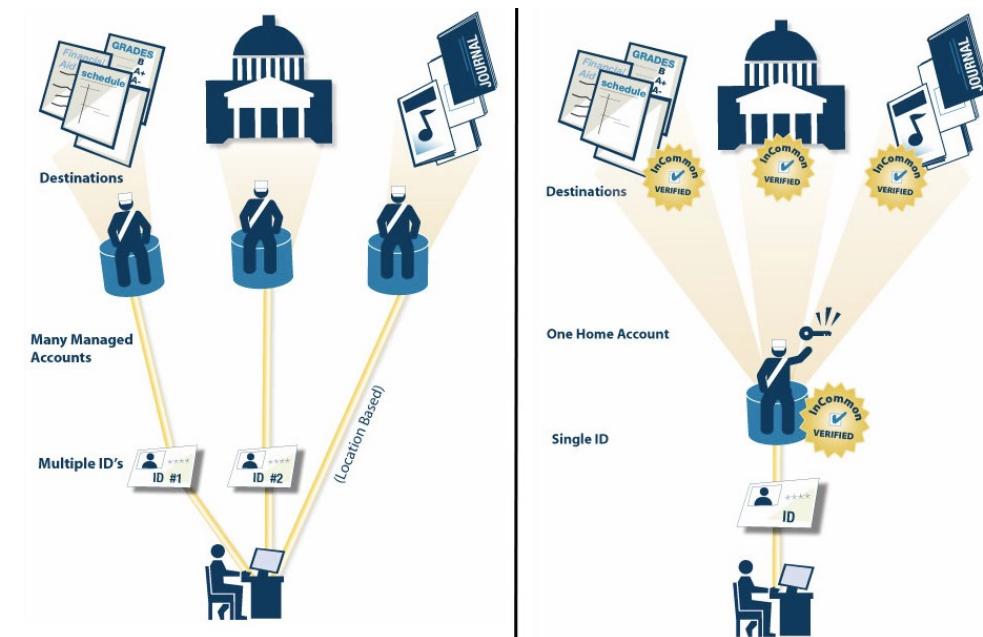
Autenticación basado en códigos QR

- Se recomienda el uso combinado de códigos QR con la técnica de “One Time Password” (OTP)
 - QR + OTP (recuerda de un solo uso)
- Funcionamiento:
 - El servidor genera un código QR junto con una contraseña (OTP) codificada en el propio QR
 - El usuario escanea el código QR para proceder con su autenticación en el servidor



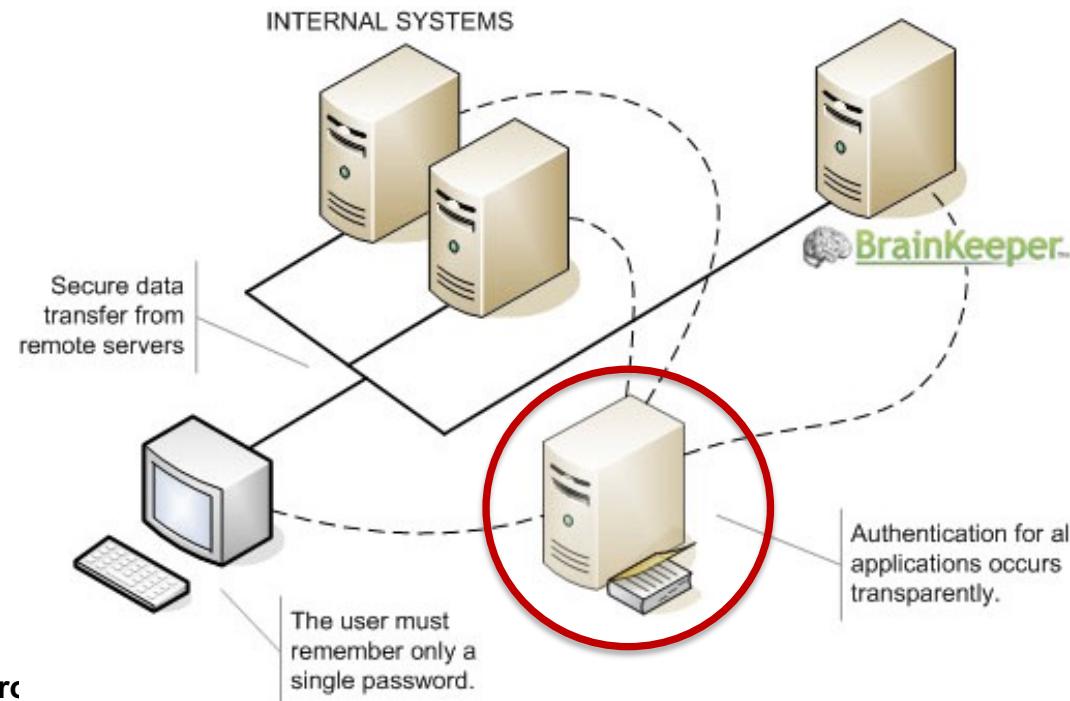
Single Sign-On (SSO)

- 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
 - Con esto se evita a los usuarios tener que gestionar y recordar muchos tipos de contraseñas



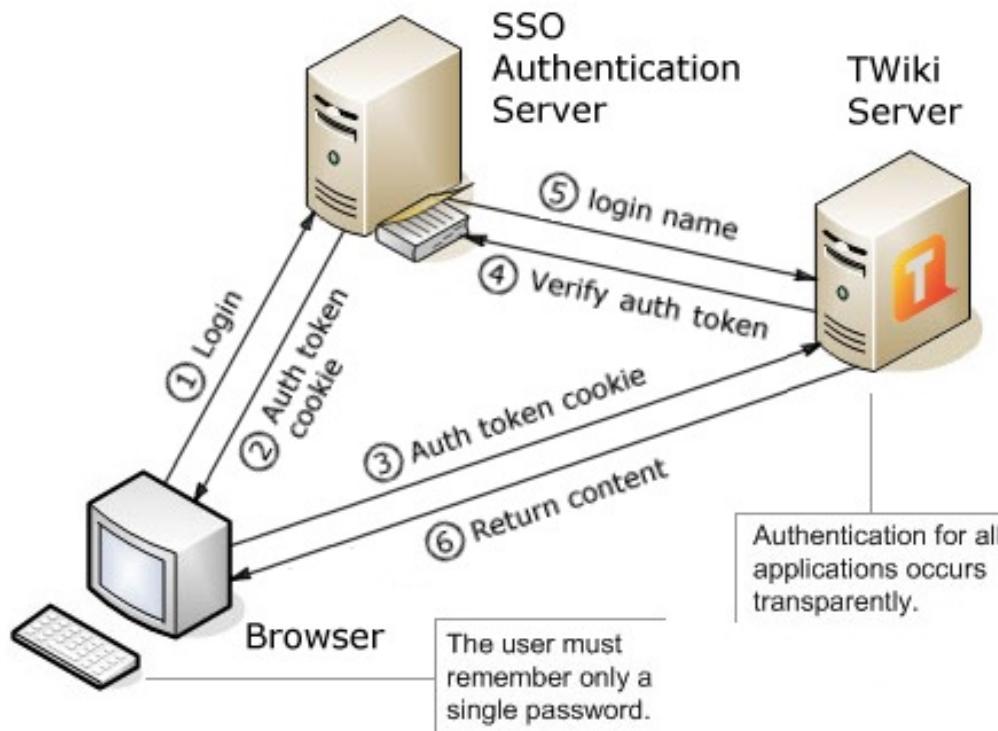
Single Sign-On (SSO)

- 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

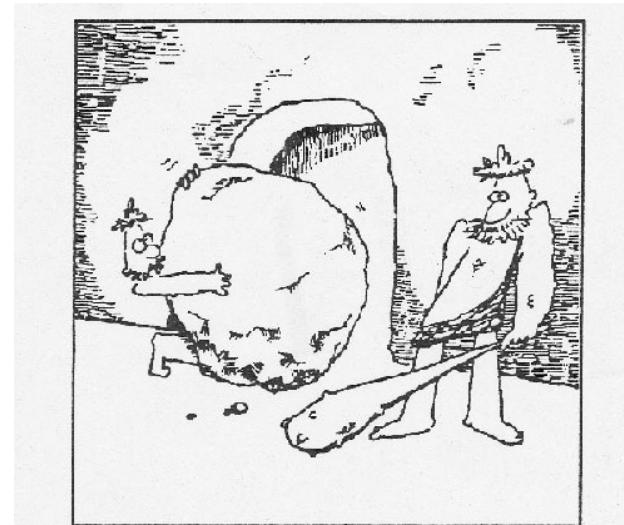


Single Sign-On (SSO)

- 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

Control de acceso

- 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



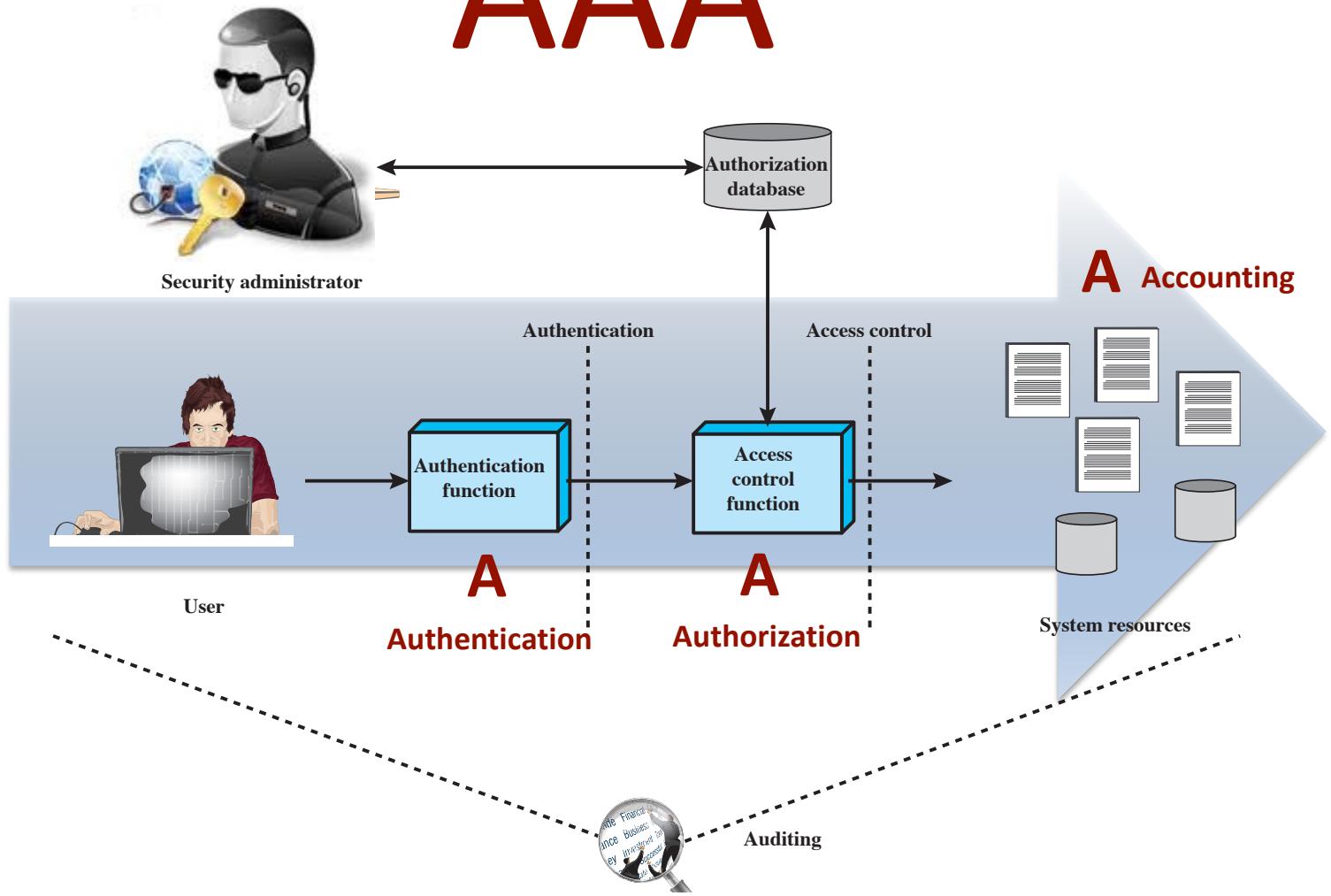
Control de acceso

- 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 accounting (AAA)**
 - **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



*Accounting – registro para permitir la auditoría y determinar el nivel de responsabilidad

AAA



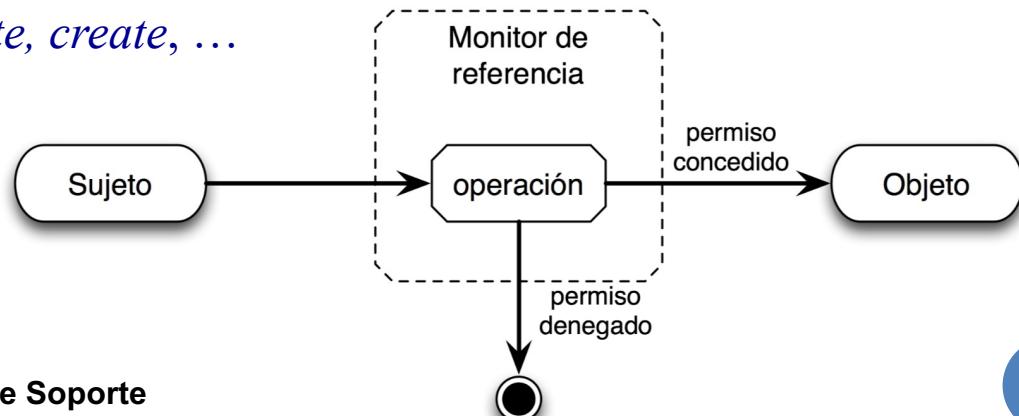
Control de acceso

- 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 anterior 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

Un mediador también es conocido como **monitor de referencia**

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, programas, puertos de comunicación, etc.
 - **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, ...*

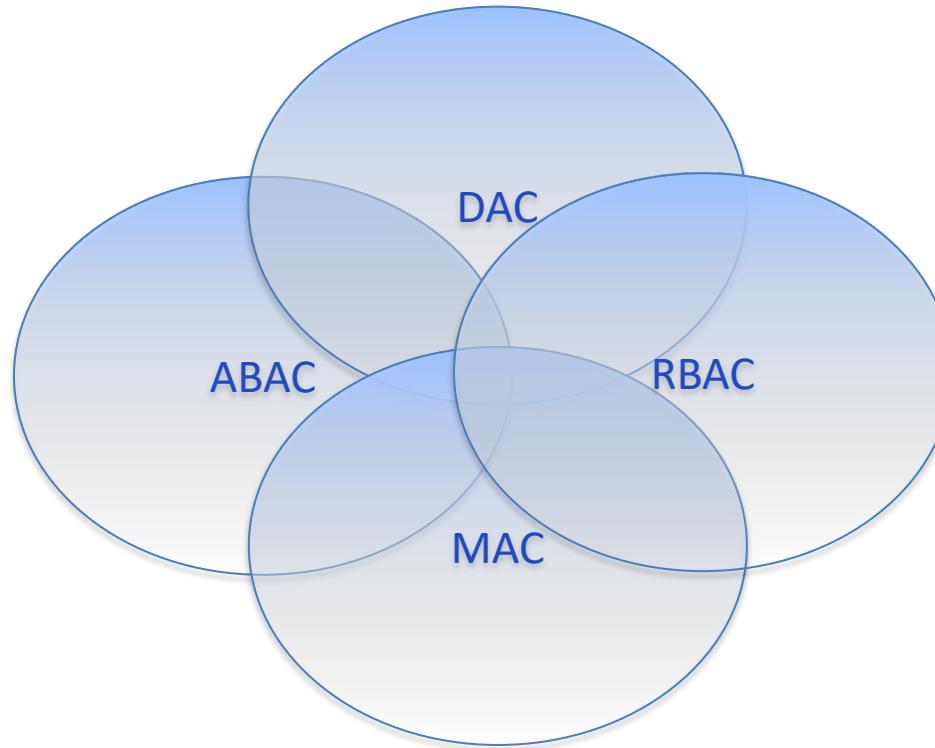


Mecanismos de control de acceso

- Las esquemas de control de acceso se dividen principalmente en varias categorías:
 - **DAC (Discretionary Access Control)**: se basa en
 - identidad del solicitante y
 - reglas/condiciones de acceso
 - **MAC (Mandatory Access Control)**: se basa en comparar
 - etiquetas de seguridad (que indican la criticidad de los recursos) con
 - identidades (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/condiciones de acceso que indican qué accesos están permitidos a quien poseen un determinado rol
 - **ABAC (Attribute-Based Access Control)**: se basa en
 - atributos asociados con el usuario y que dependiendo del atributo se permite o no el acceso a un sistema
 - características del usuario o sujeto
 - ...

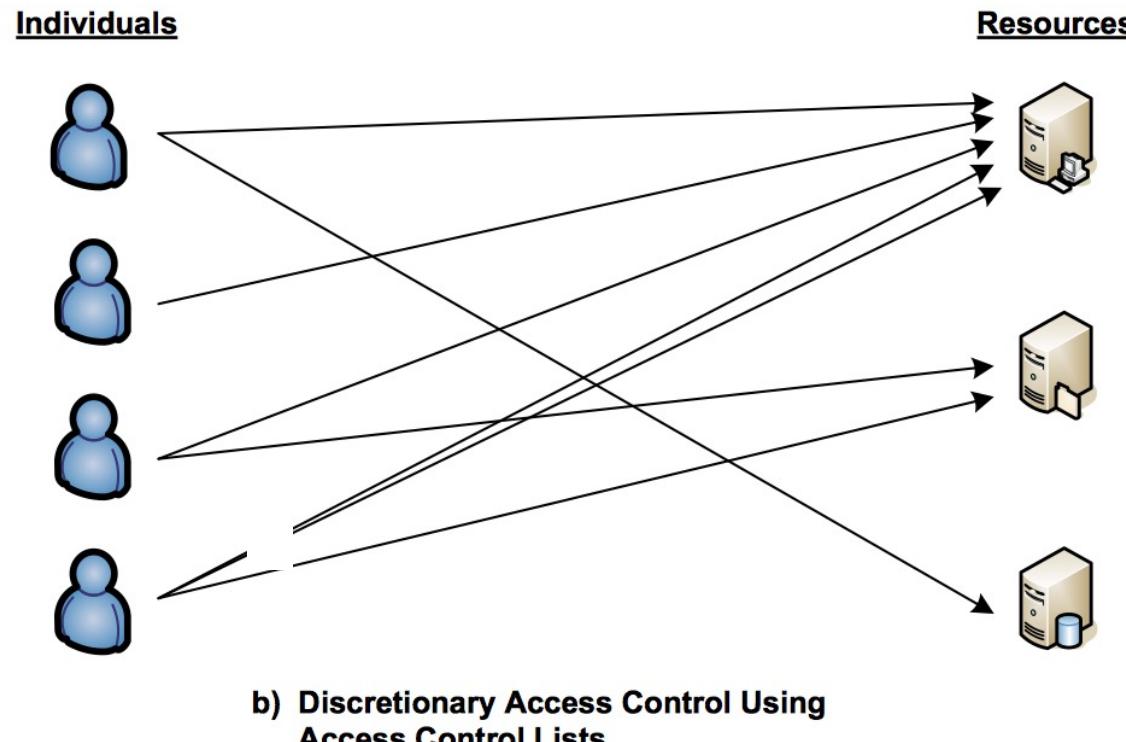
Mecanismos de control de acceso

- 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



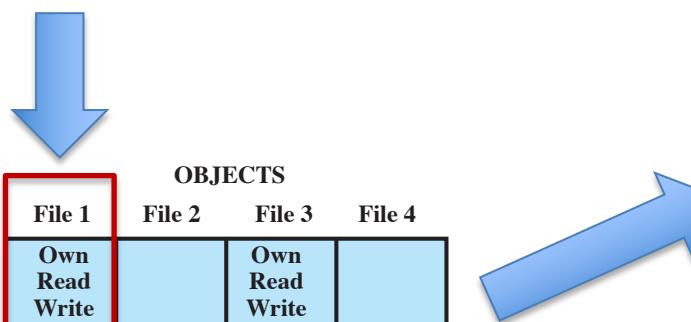
DAC (Discretionary Access Control)

- 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

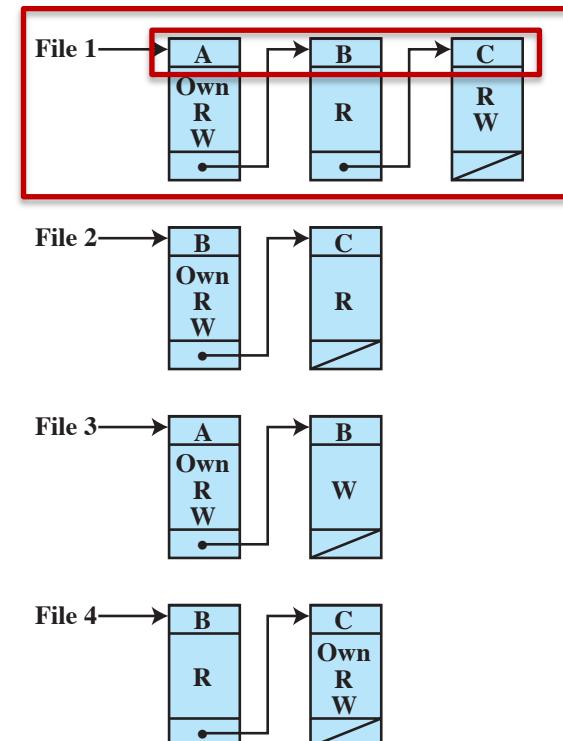
		OBJECTS			
		File 1	File 2	File 3	File 4
SUBJECTS	User A	Own Read Write		Own Read Write	
	User B	Read	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write

DAC (Discretionary Access Control)

- 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

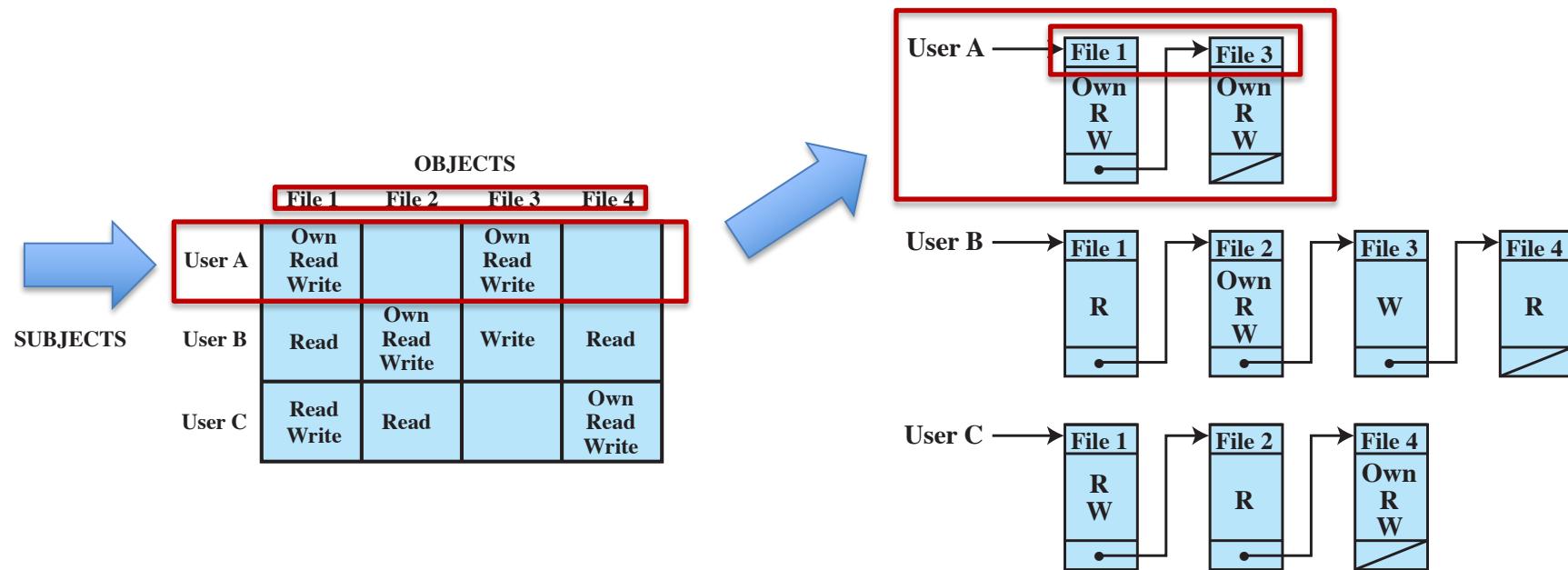


		OBJECTS				
		File 1	File 2	File 3	File 4	
SUBJECTS		User A	Own Read Write		Own Read Write	
		User B	Read	Own Read Write	Write	Read
		User C	Read Write	Read		Own Read Write



DAC (Discretionary Access Control)

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



DAC (Discretionary Access Control)

- Ejemplo de lista de ACL:

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

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



¿Ventajas? ¿Inconvenientes?

DAC (Discretionary Access Control)

- 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.

DAC (Discretionary Access Control)

- 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

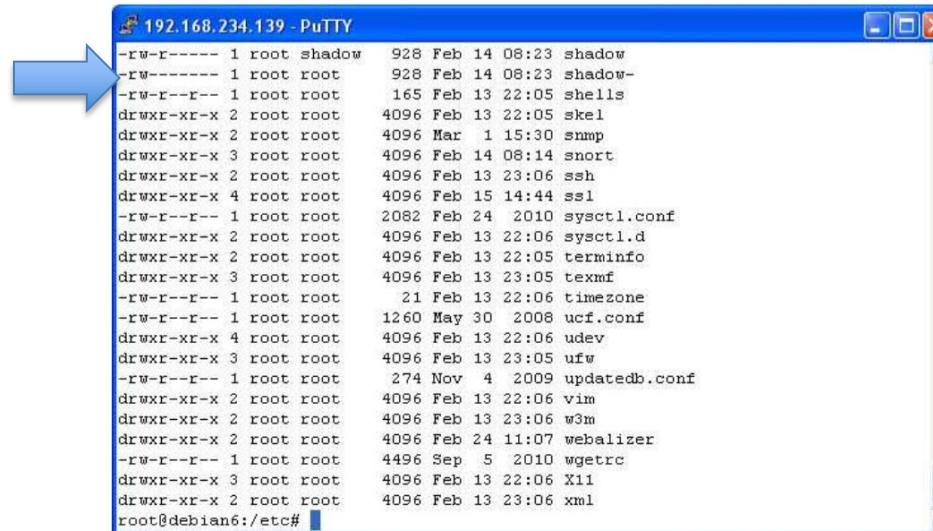
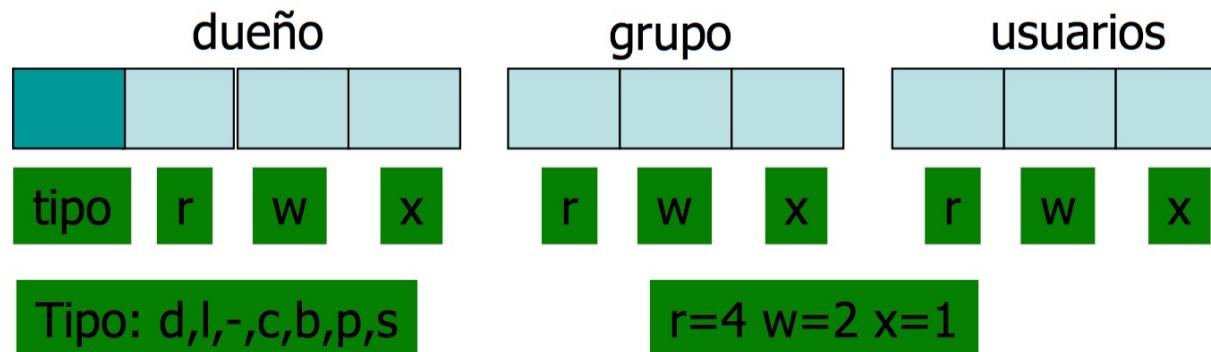
DAC (Discretionary Access Control)

- **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

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

DAC (Discretionary Access Control)

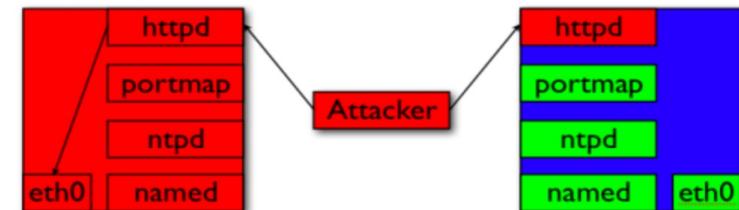
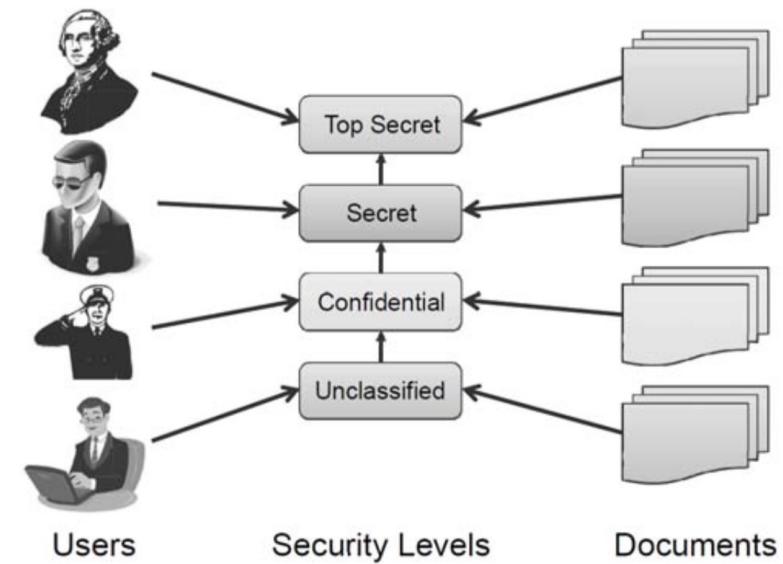
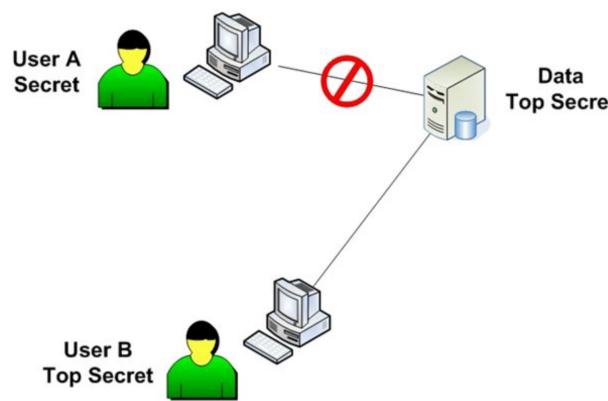
- Ejemplo de permisos básicos en UNIX, usando DAC



```
192.168.234.139 - PuTTY
-rw-r----- 1 root shadow 928 Feb 14 08:23 shadow
-rw-r----- 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)

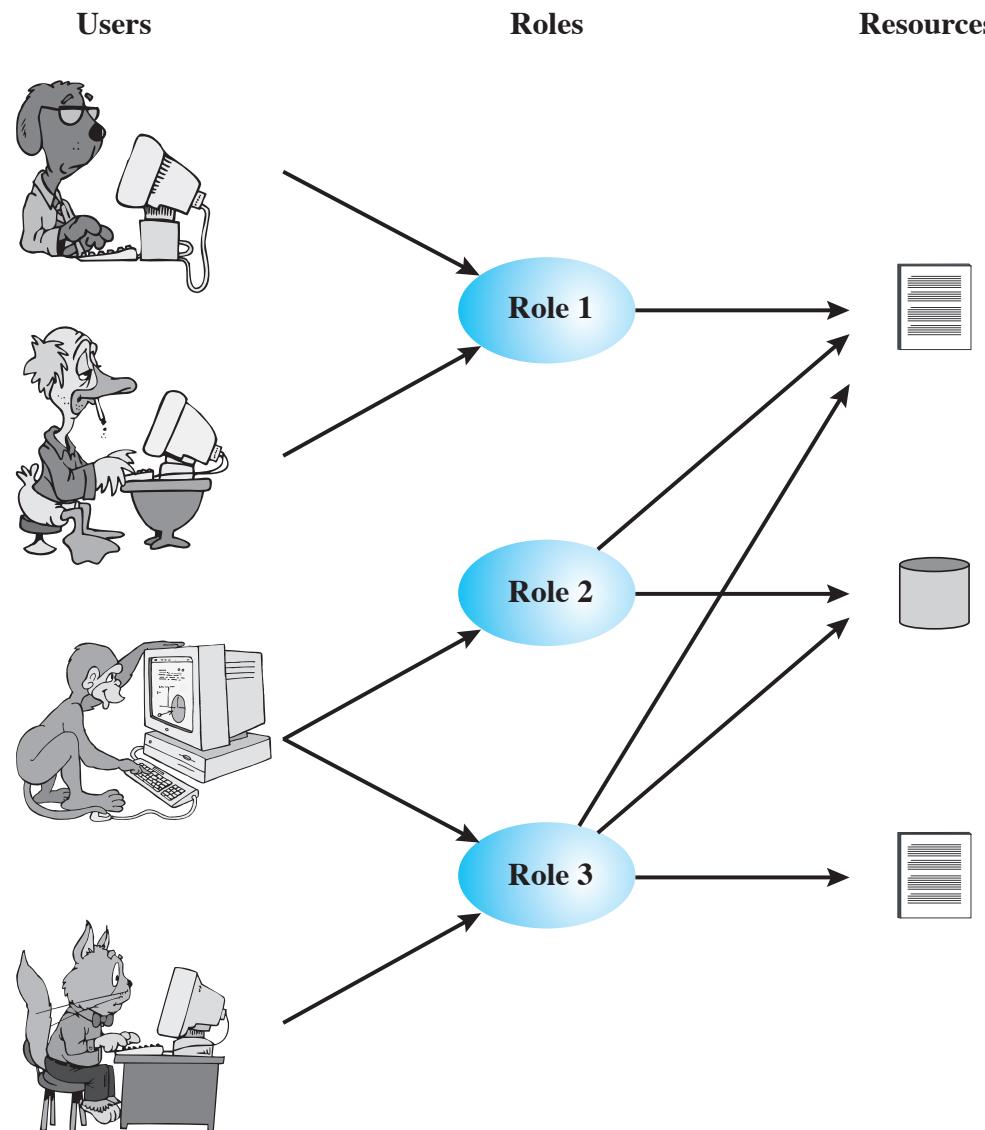
- Se basa en comparar etiquetas de seguridad (que indican la criticidad de los recursos) con las autorizaciones de seguridad (que indican las entidades que pueden acceder a ciertos recursos)
- Por lo tanto, a cada recurso se le asigna una etiqueta de seguridad, que de alguna forma lo clasifica
 - top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



RBAC (Role-based Access Control)

- No se basa en la identidad de los usuarios, sino en los **roles** que asumen tales usuarios
 - Un rol es una función/tarea a realizar dentro de una empresa u organización
- El modelo RBAC asigna:
 - los derechos de acceso a los roles
 - y luego asigna los 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 muy utilizado, y se puede aplicar:
 - 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

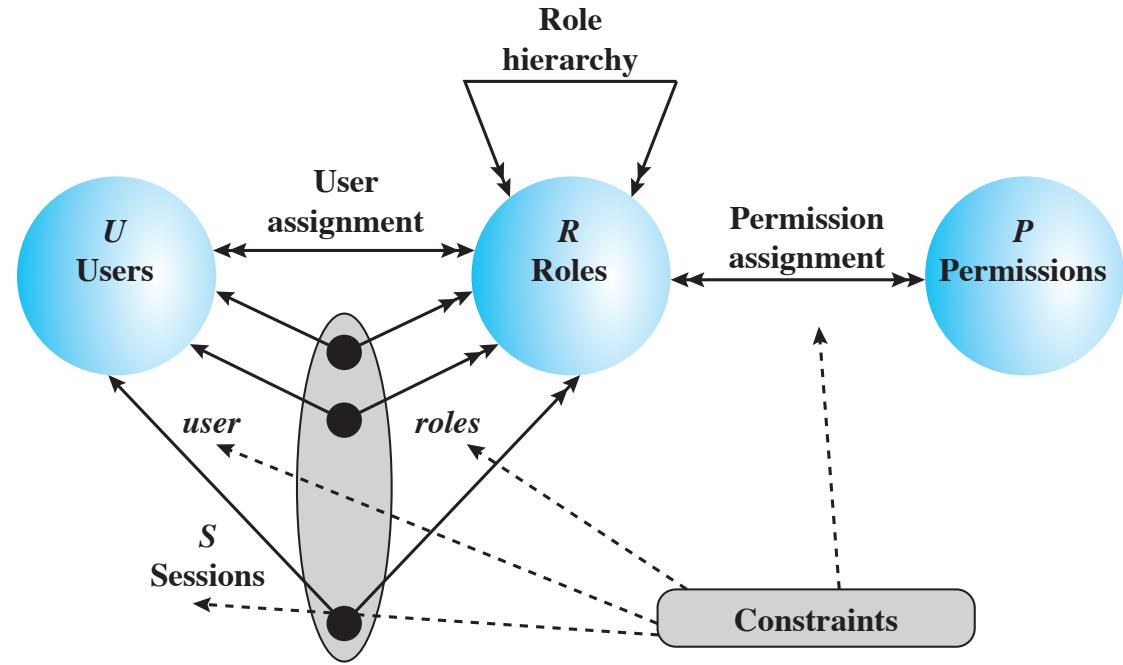
	R ₁	R ₂	• • •	R _n
U ₁	X			
U ₂	X			
U ₃		X		X
U ₄				X
U ₅				X
U ₆				X
•				
U _m	X			

	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



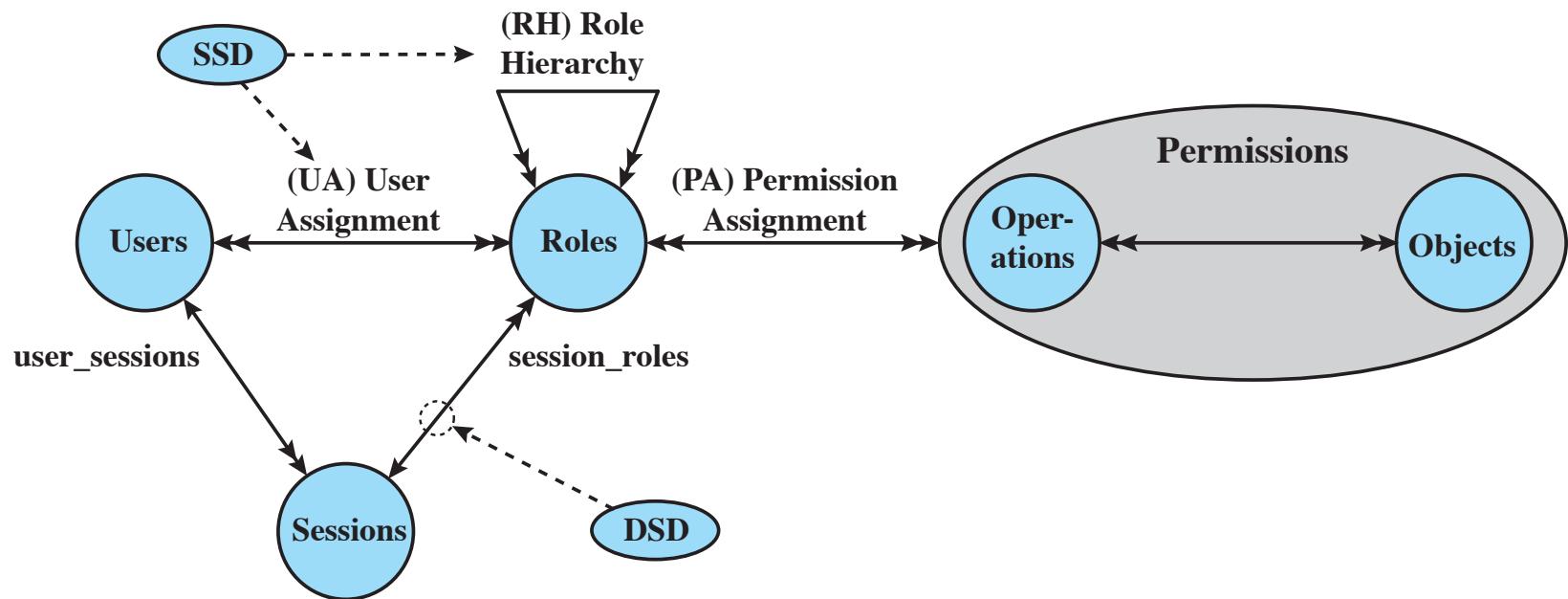
- Las relaciones muchos-a-muchos entre usuarios y roles, y entre roles y permisos, proporcionan **flexibilidad y granularidad** a la hora de gestionar las condiciones de acceso a un usuario
 - Tal una gestión que no ocurre con DAC

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
 - Ej. escalar en funciones según la jerarquía de una organización

RBAC (Role-based Access Control)

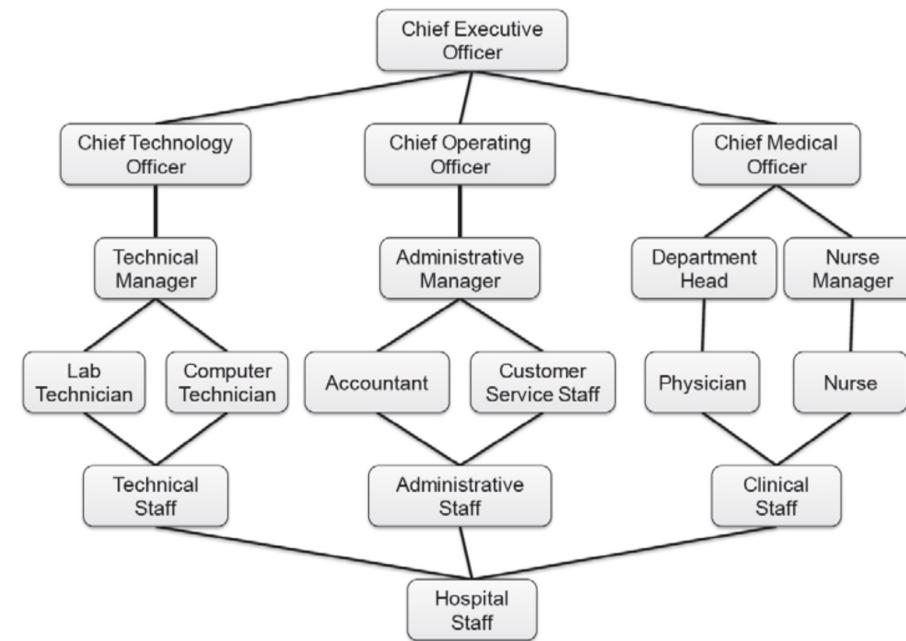
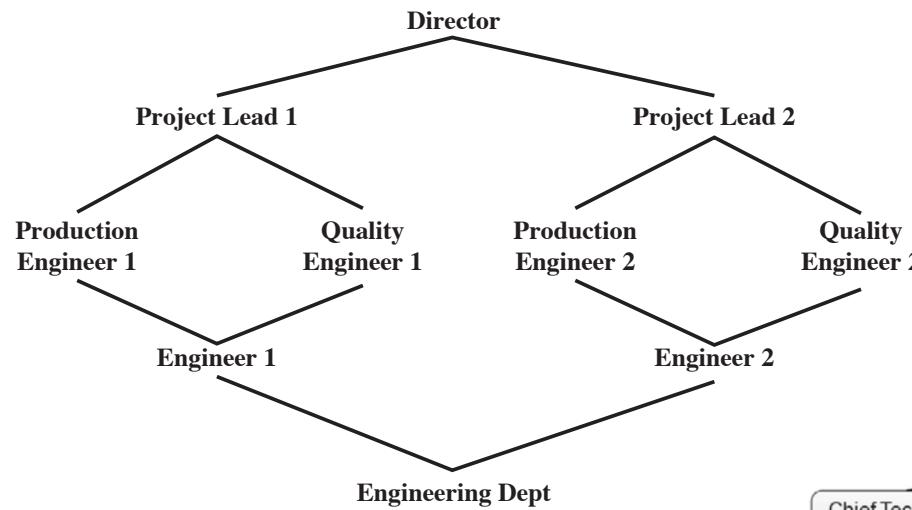
- El modelo **RBAC estándar de NIST** introduce algunas extensiones al RBAC tradicional, como:
 - **Static Separation of Duties (SSD)**: para definir roles mutuamente excluyentes
 - **Dynamic Separation of Duties (DSD)**: para definir restricciones sobre los roles que un usuario puede activar en una 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, moren@s/rubi@s/canos@s, baj@s/alt@s,...
 - Cualquier usuario con más de 18 años, moren@ 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
 - » **Atributo == condición de acceso**

CapBAC (Capability-Based Access Control)

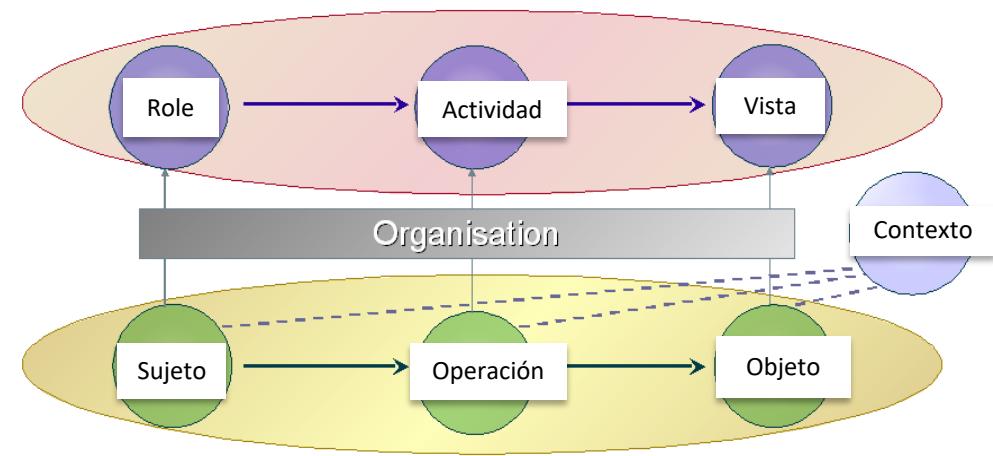
- CapBAC basa su modelo en un conjunto de **roles** y **atributos**,
 - Atributos → capacidades
 - Roles → funciones
- El acceso sólo es posible si el usuario recibe del proveedor del recurso un **token** (un “certificado de autorización”) que demuestra su “capability” para realizar determinadas acciones 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 y complejos**, 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

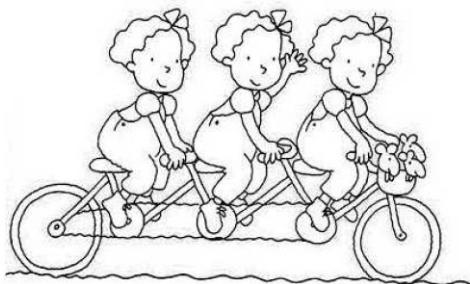


Protocolos criptográficos avanzados



- 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



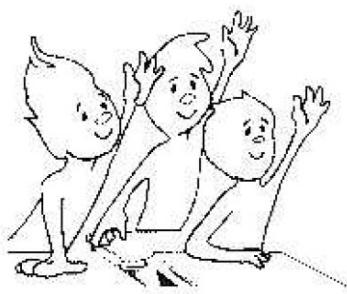
Canal Subliminal



Lanzamiento de monedas



Efecto Mente



Protocolos Criptográficos



Firma de Contratos



Transferencia Inconsciente



Demostraciones de Conocimiento Nulo

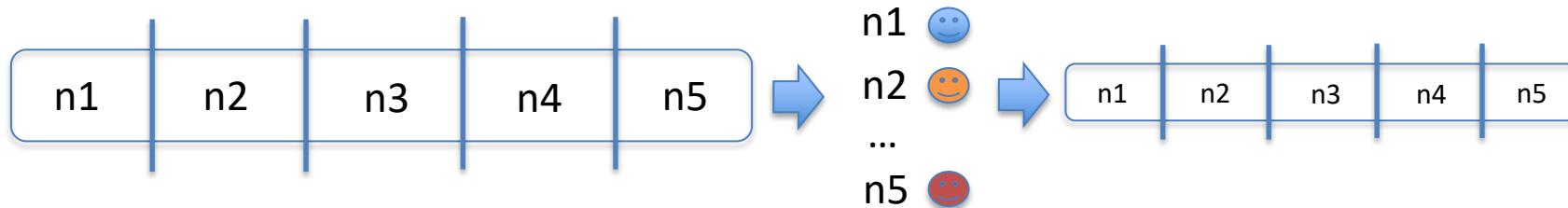
Criptografía-ULL

Poker Mental

Protocolo de división de secretos

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
- El protocolo requiere la intervención de una Tercera Parte Confiable, o TTP

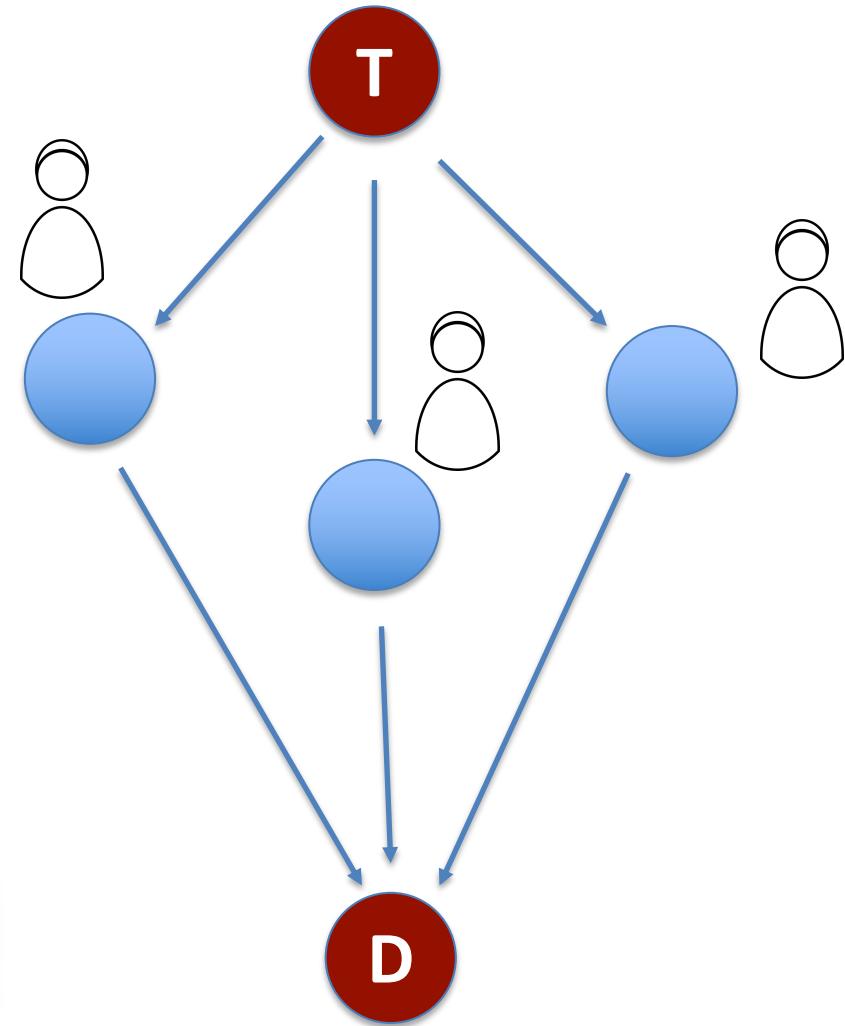


Protocolo de división de secretos - ejemplo 1

- 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 $S \rightarrow M \oplus R = S$
 - 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 $\rightarrow R \oplus S = M$
- 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



- ¿Y para 3 o más personas?



Protocolo de división de secretos - ejemplo 2

- 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

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 sombras, 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



Protocolo de compartición de secretos

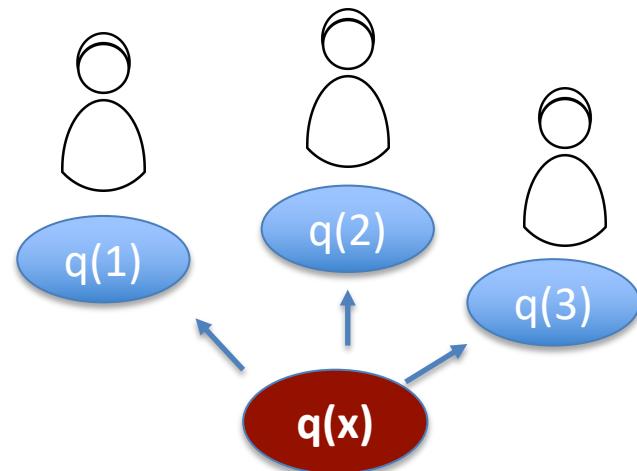
- Más concretamente, se eligen aleatoriamente los coeficientes de un polinomio de grado $k-1$:

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

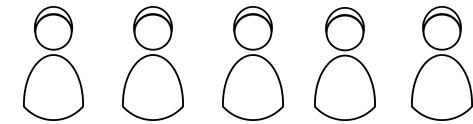
donde $\mathbf{a}_0 = \mathbf{D}_0 = \mathbf{D}$ y evaluamos:

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

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



Protocolo de compartición de secretos - ejemplo 1



- 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:

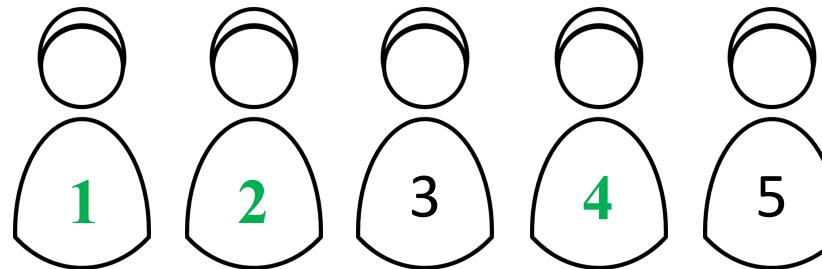
$$q(1) = 14 \rightarrow Alice$$

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

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

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

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



- 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*: $q(1) = 14 = a_0 + a_1 \cdot 1 + a_2 \cdot 1^2$
- 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**

Protocolo de compartición de secretos - ejemplo 2

- 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

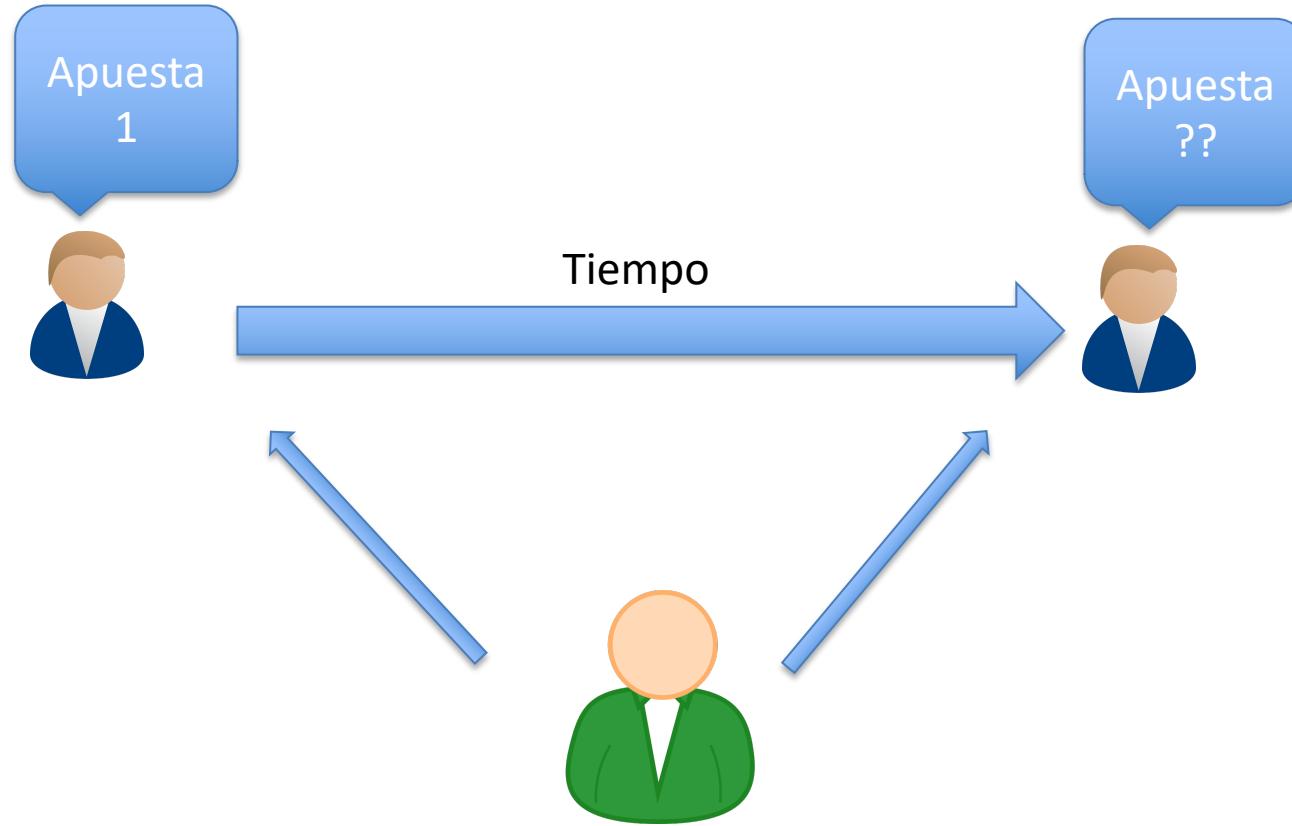
Protocolos de bit-commitment

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



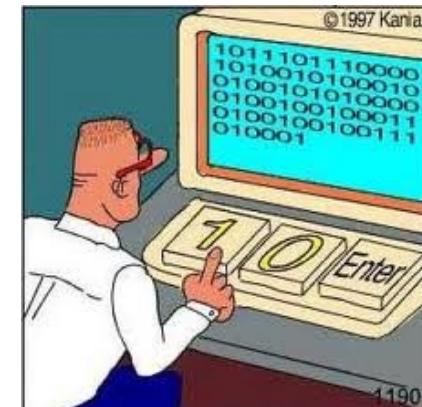
Protocolos de bit-commitment



Bob debe asegurar que Alice no hace trampa, pero
no debería conocer la apuesta de Alice

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



Protocolos de bit-commitment - criptografía simétrica

- 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

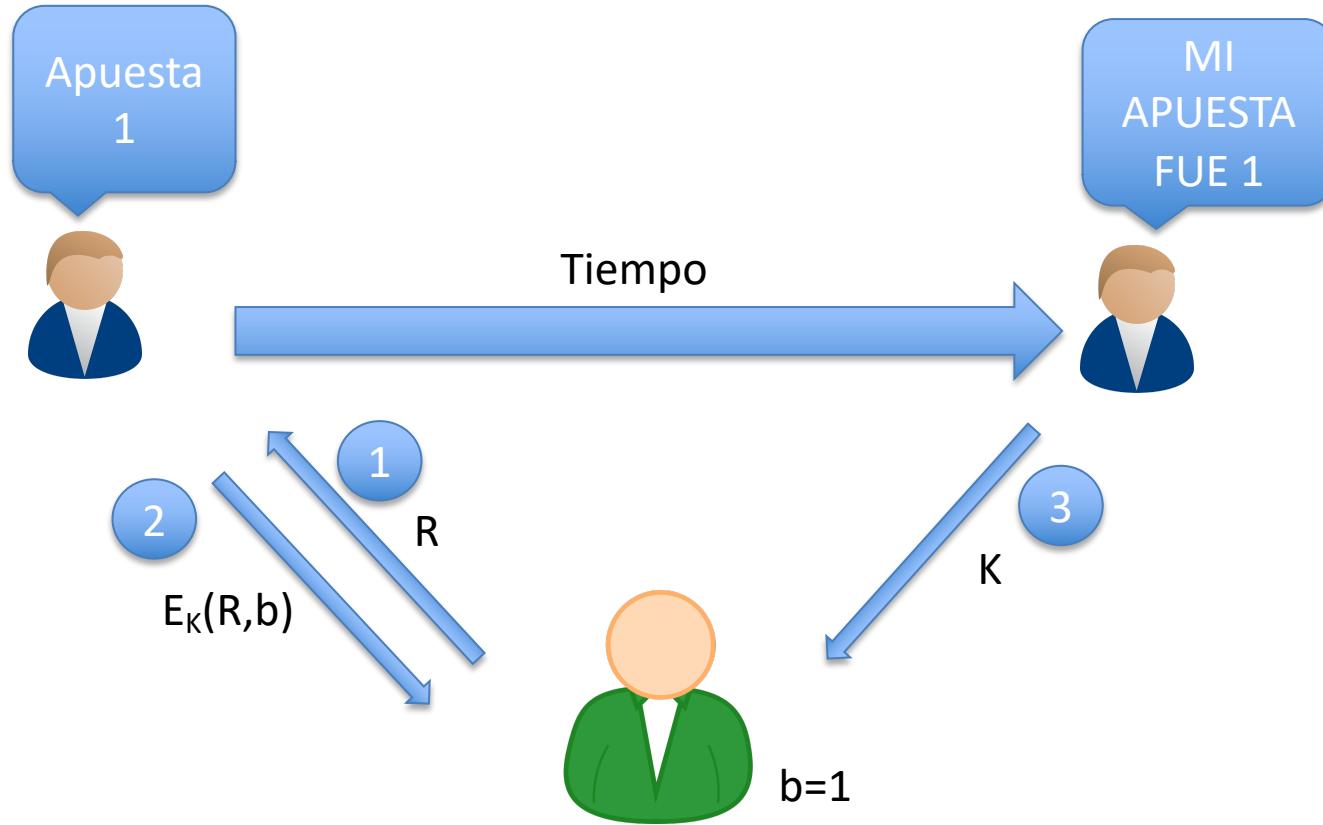
③ Alice envía a Bob la clave K

$$Alice \rightarrow Bob: K$$

④ Bob descifra el mensaje para obtener el bit, y chequea su cadena aleatoria para verificar la validez del bit

$$Bob: D_K(E_K(R, b))$$

Protocolos de bit-commitment - criptografía simétrica



Protocolos de bit-commitment - funciones hash

- 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

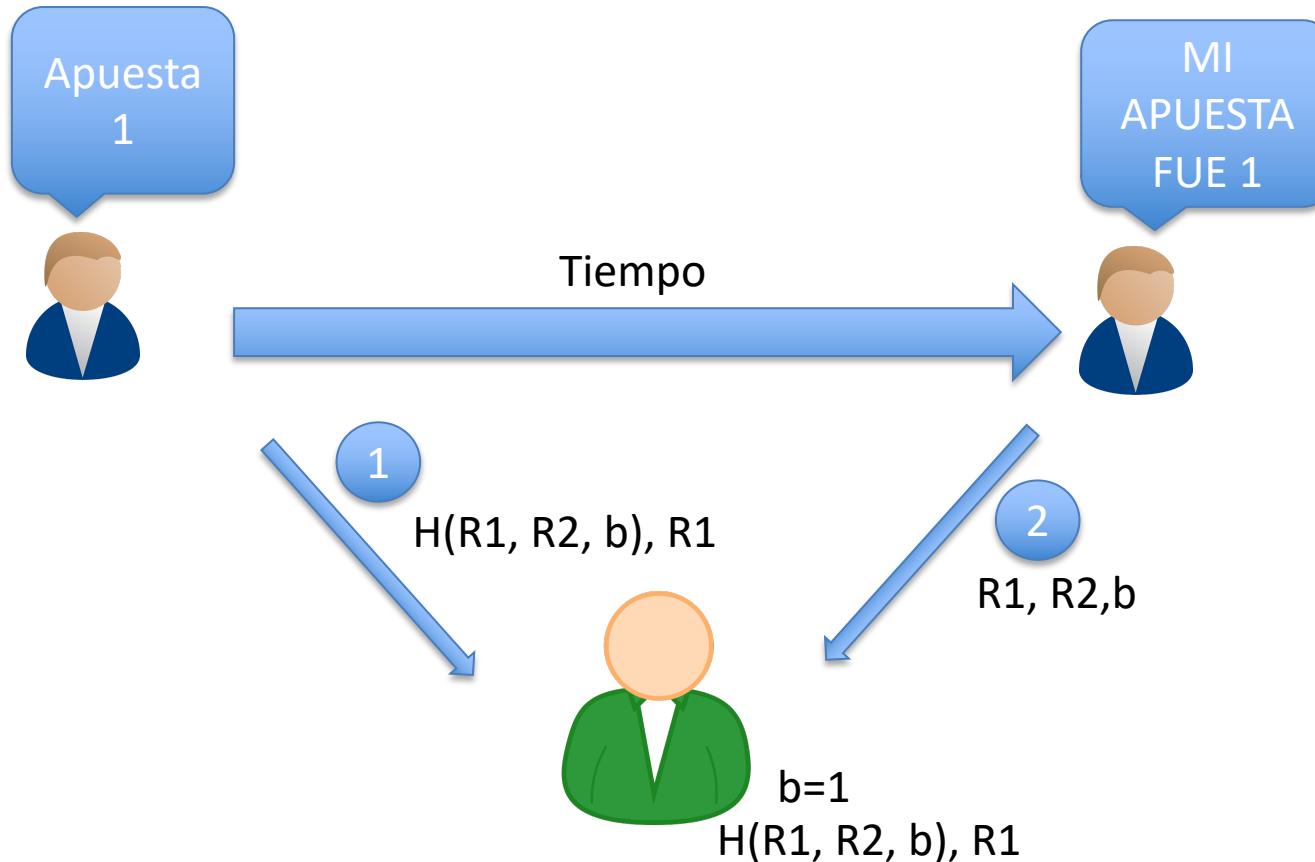
③ *Alice* envía a *Bob* el mensaje original

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 bit-commitment - funciones hash



Bob debe asegurar que Alice no hace trampa

Protocolos de lanzamiento de moneda

Protocolos de lanzamiento de moneda

- Supongamos que *Alice* desea hacer al lanzamiento de la moneda (“cara o cruz”)
 - *Alice* hace el lanzamiento de la moneda con ayuda de otro, *Bob* – *sin encontrarse físicamente los dos*
- 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 que lado cayó la moneda antes de pronunciarse
- Podemos solucionar este problema utilizando:
 - el protocolo de compromiso de bit o
 - criptografía de clave pública, como se ve a continuación



Protocolos de lanzamiento de moneda - crip. asimétrica

- 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}

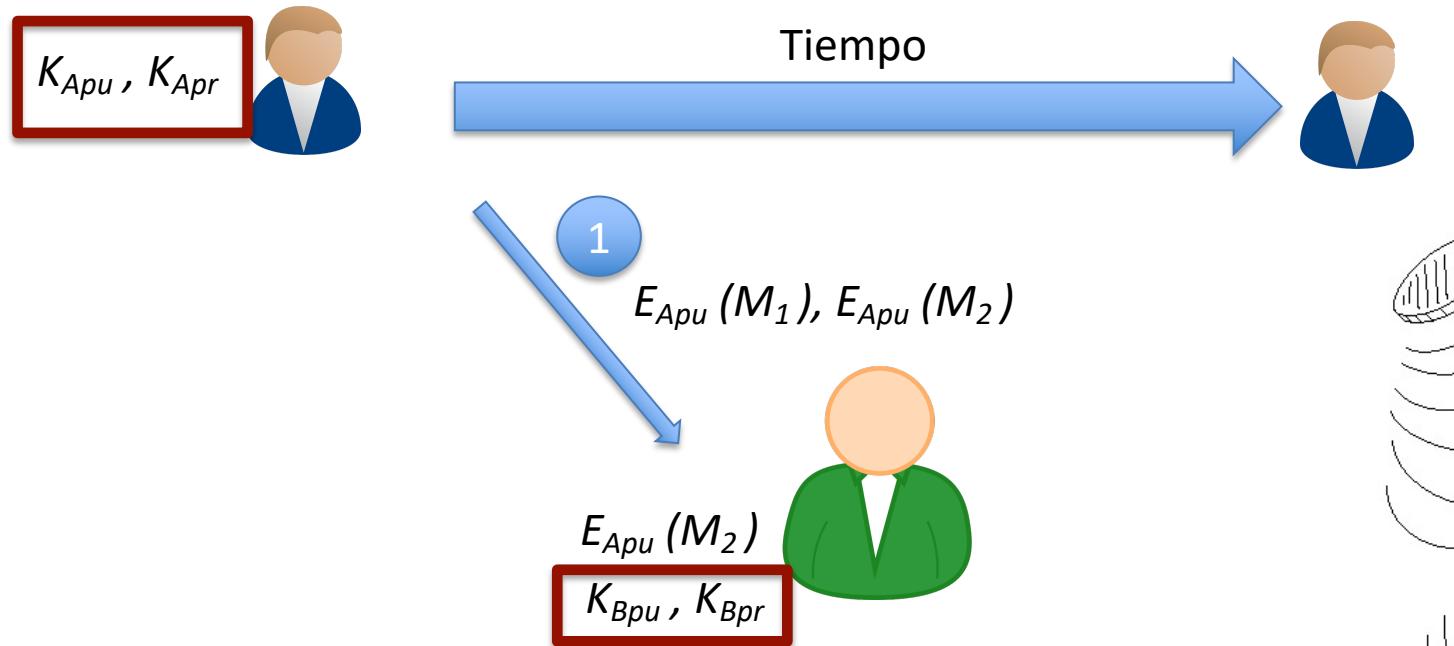
Bob: K_{Bpu}, K_{Bpr}

- 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*: $E_{Apu}(M_1), E_{Apu}(M_2)$

Protocolos de lanzamiento de moneda - crip. asimétrica



Bob debe asegurar que Alice no hace trampa

Protocolos de lanzamiento de moneda - crip. asimétrica

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

$Bob \rightarrow Alice: E_{Bpu}(E_{Apu}(M))$ 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*

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

$Alice \rightarrow Bob: E_{Bpu}(M)$

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

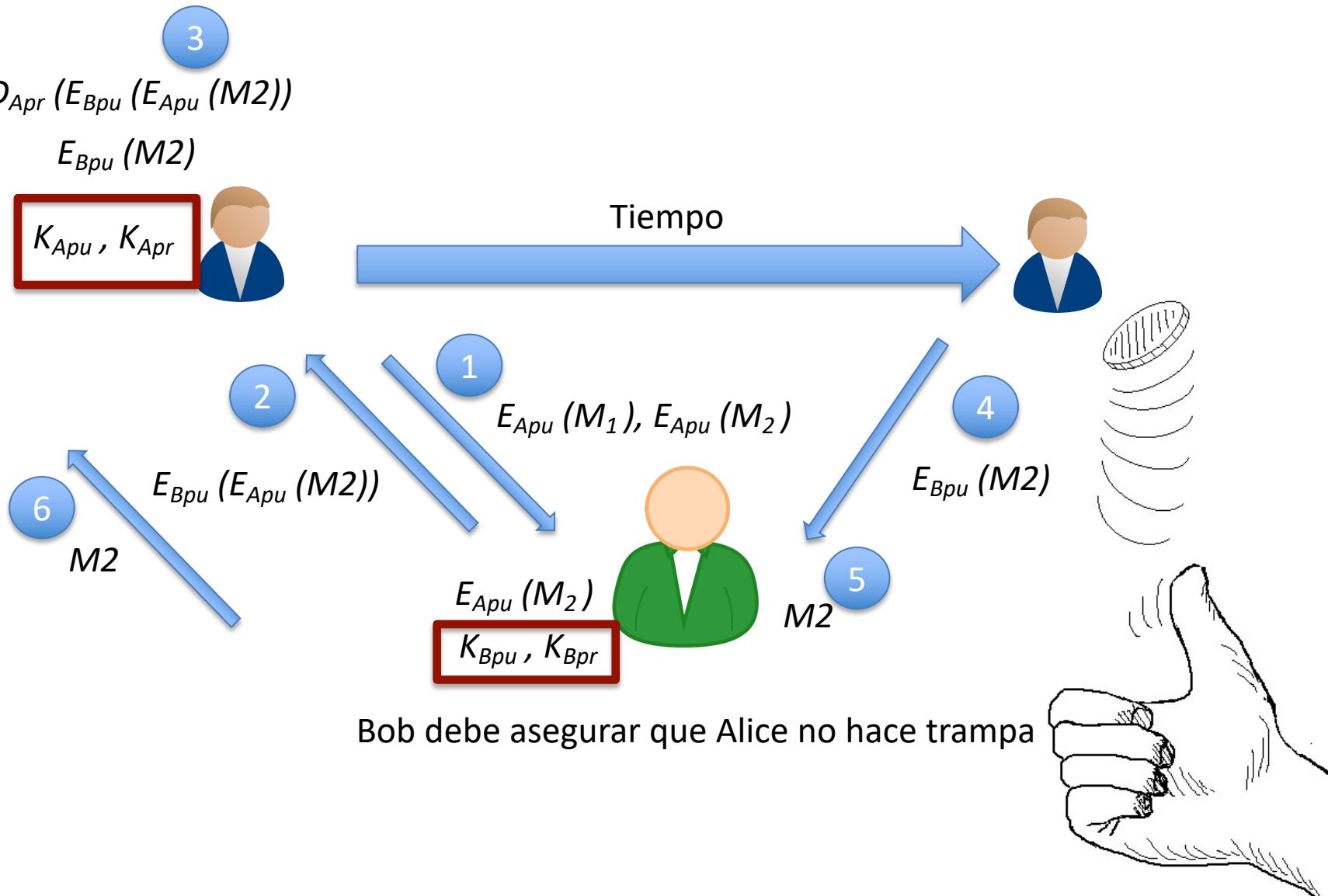
$Bob: D_{Bpr}(E_{Bpu}(M))$

$Bob \rightarrow Alice: M$

- ⑥ *Alice* lee el resultado del lanzamiento y verifica que la cadena aleatoria es correcta

- No hace falta una tercera parte para completar el protocolo

Protocolos de lanzamiento de moneda - crip. asimétrica



Protocolo de póker mental

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 $i = 1 .. 52$



Protocolo de póker mental

- 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*

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

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*

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

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

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

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

Protocolo de póker mental

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