

SEGURIDAD DE LA INFORMACIÓN

TEMA 1

FUNDAMENTOS DE SEGURIDAD

Índice del tema

- Introducción
 - Seguridad de la Información: Conceptos
 - Ciclo de vida de la Seguridad
 - Modelo de escenario de Seguridad,
 - Ataques principales
- Servicios y mecanismos de seguridad
 - Las cinco categorías fundamentales:
 - C - confidencialidad
 - I - integridad
 - A - autenticación
 - A – (control de acceso) - autorización
 - N – no repudio
 - La jerarquía: Servicios, Protocolos, Mecanismos, Técnicas

INTRODUCCIÓN

- Algunas definiciones de “Seguridad de la Información”

*“Information security is the **protection of information from a wide range of threats** in order to ensure business continuity, minimize business risk, and maximize return on investments and business opportunities”*

ISO/IEC 17799: Code of practice for information security management

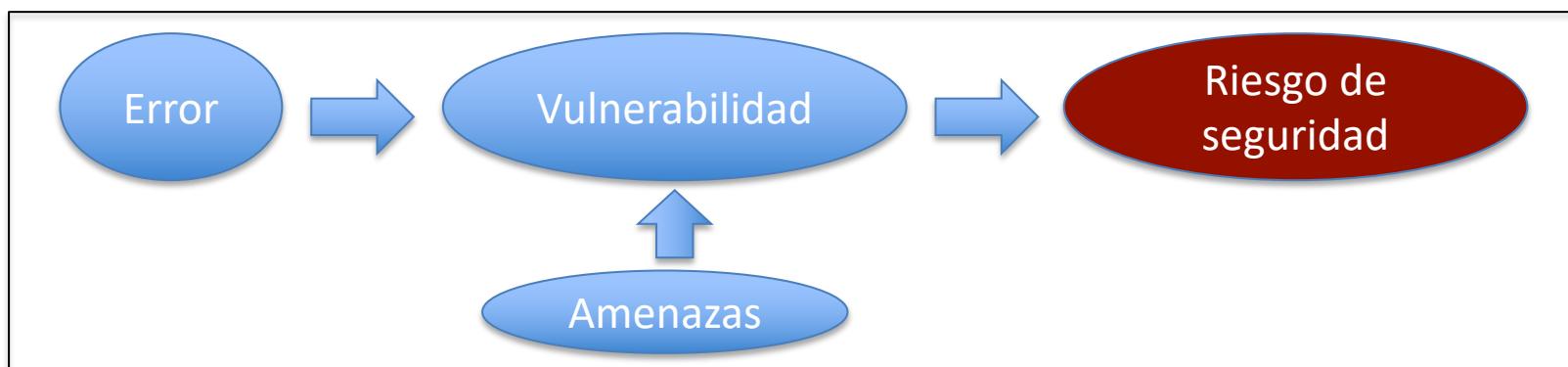
*“The **protection of information assets** through the use of technology, processes, and training”*

Microsoft Security Glossary

*“The ability of a system to **manage, protect, and distribute sensitive information**”*

Software Engineering Institute, Carnegie Mellon University

- Un error en la fase de análisis, diseño, desarrollo o implementación puede producir, a posteriori, un **fallo** de seguridad
 - También llamado **vulnerabilidad**



- Como consecuencia, se viola la **política de seguridad** del sistema, y este queda en peligro
 - En una red como Internet, con las dimensiones, números de hosts y número de usuarios actuales, el efecto devastador es exponencial



Ciclo de vida de la Seguridad

- La política de seguridad es el conjunto de reglas/requisitos que gobiernan el comportamiento del sistema, en lo que a seguridad se refiere
- Ejemplos de requisitos:

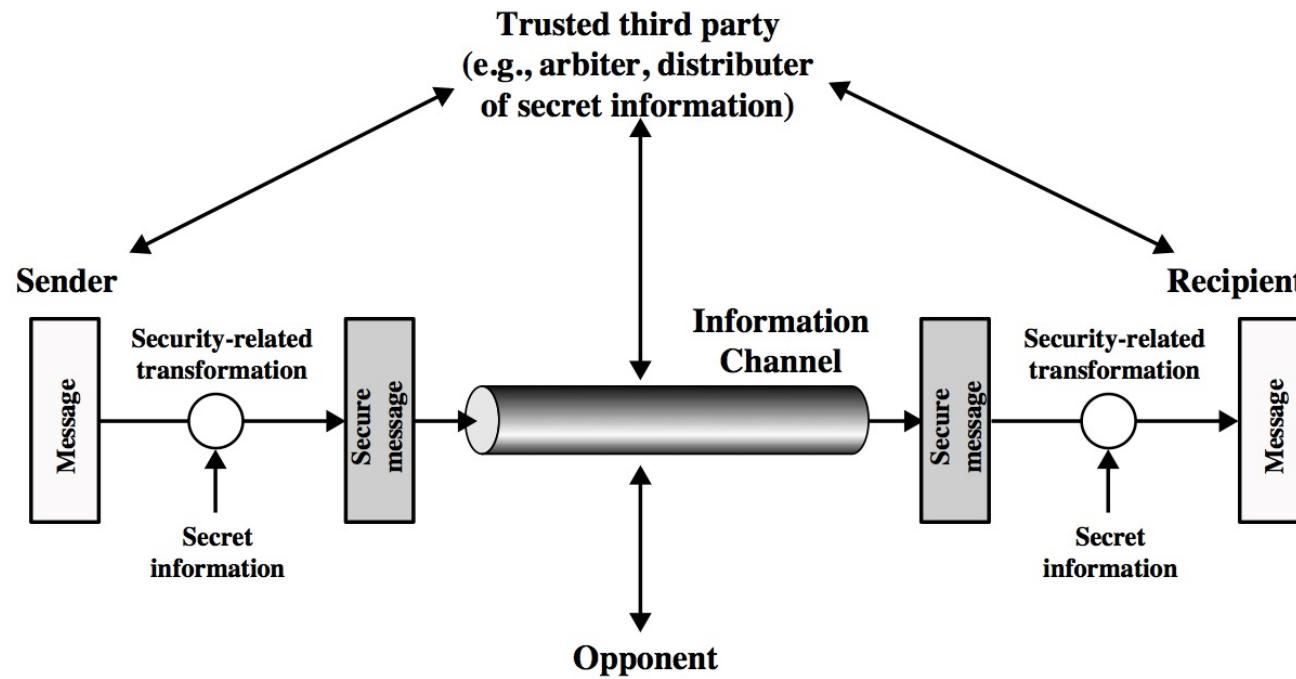
keeping information secret from all but those who are authorized to see it.
ensuring information has not been altered by unauthorized or unknown means.
corroboration of the identity of an entity (e.g., a person, a computer terminal, a credit card, etc.).
corroborating the source of information; also known as data origin authentication.
a means to bind information to an entity.
conveyance, to another entity, of official sanction to do or be something.
a means to provide timeliness of authorization to use or manipulate information or resources.
restricting access to resources to privileged entities.
endorsement of information by a trusted entity.
recording the time of creation or existence of information.
verifying the creation or existence of information by an entity other than the creator.
acknowledgement that information has been received.
acknowledgement that services have been provided.
a means to provide an entity with the legal right to use or transfer a resource to others.
concealing the identity of an entity involved in some process.
preventing the denial of previous commitments or actions.
retraction of certification or authorization.

Fuente: “Handbook of Applied Cryptography”

- La política de seguridad es sólo una de las fases del **ciclo de vida de la seguridad**. El modelo general de ciclo de vida se incluye en el estándar ISO-7498-2, y consta de cinco pasos:
 1. Definición de una **política de seguridad** que contiene una serie de requisitos genéricos de seguridad para el sistema
 2. Análisis de **requisitos de seguridad**, incluyendo el análisis de riesgos, y un análisis de los requisitos legales, gubernamentales y normativos
 - 3. Definición de los **servicios de seguridad** necesarios para satisfacer los requisitos de seguridad
 - 4. Diseño del sistema e implementación, así como la selección de los **mecanismos de seguridad** que van a proporcionarnos los servicios de seguridad definidos en la etapa anterior
 5. Administración y mantenimiento de la seguridad

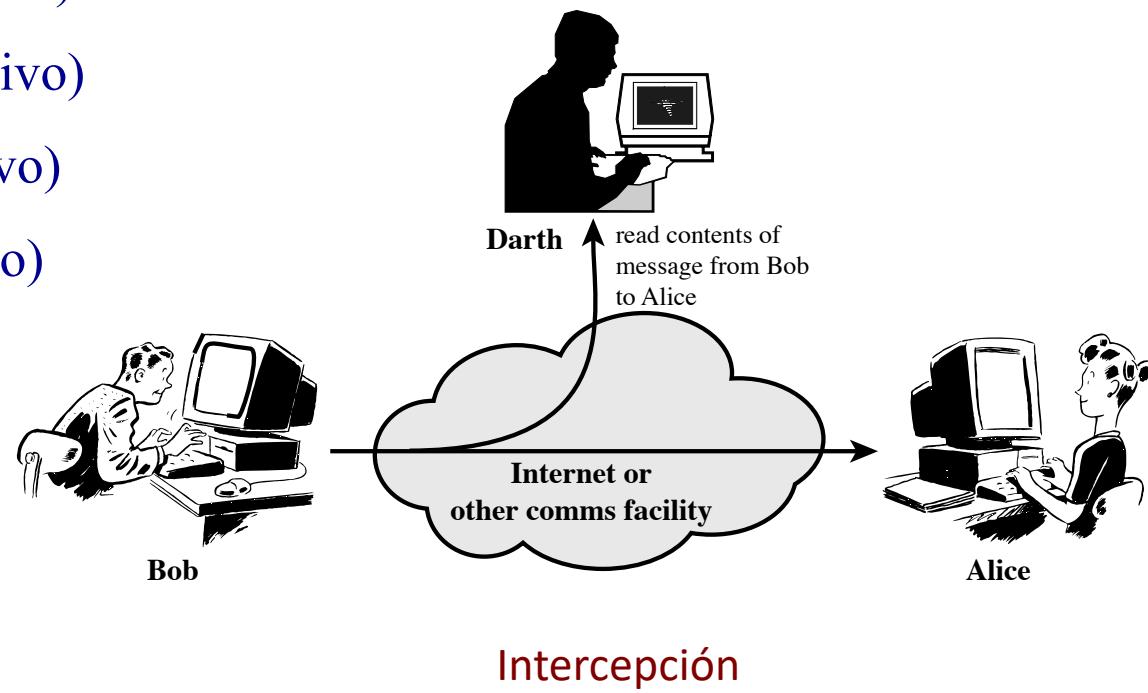
Modelo de escenario de Seguridad

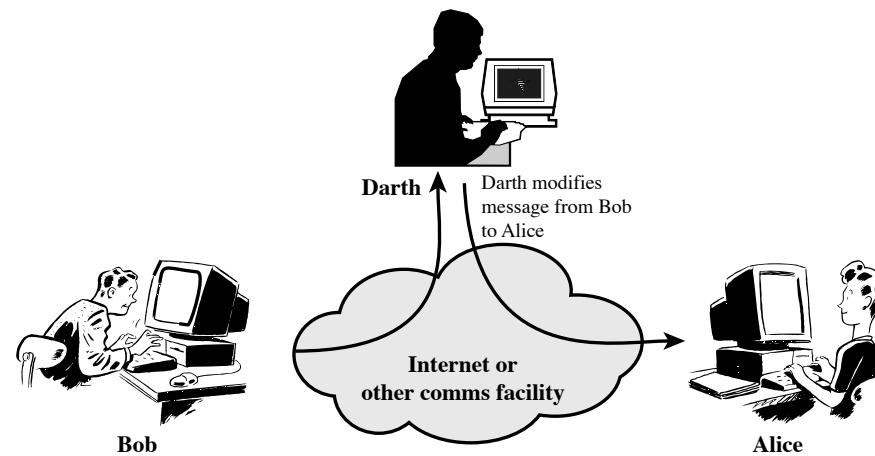
- Es necesario un escenario básico para empezar a razonar sobre:
 - las amenazas que pueden existir y los ataques que se pueden sufrir
 - las soluciones (servicios y mecanismos) de seguridad que podemos utilizar



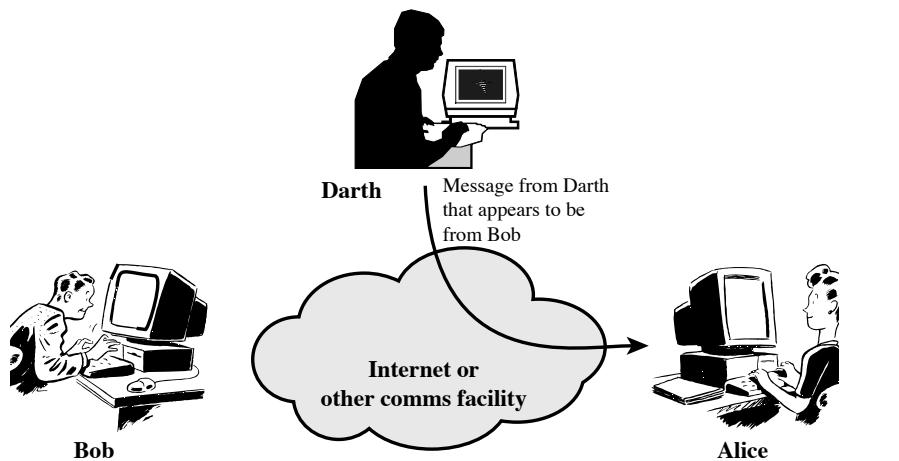
- ¿Quiénes pueden ser el emisor y el receptor en un escenario real?
 - Navegador web y Servidor web para transacciones electrónicas (por ejemplo, compra on-line)
 - Banca on-line (cliente y servidor)
 - Servidores DNS
 - Routers intercambiando tablas de enrutamiento
 - Dos usuarios en un chat, o enviándose e-mails, ...
 - Etc.

- Los ataques se pueden clasificar en **activos** y **pasivos**
- Más concretamente, se pueden considerar los siguientes cuatro tipos:
 - Intercepción (pasivo)
 - Modificación (activo)
 - Interrupción (activo)
 - Generación (activo)

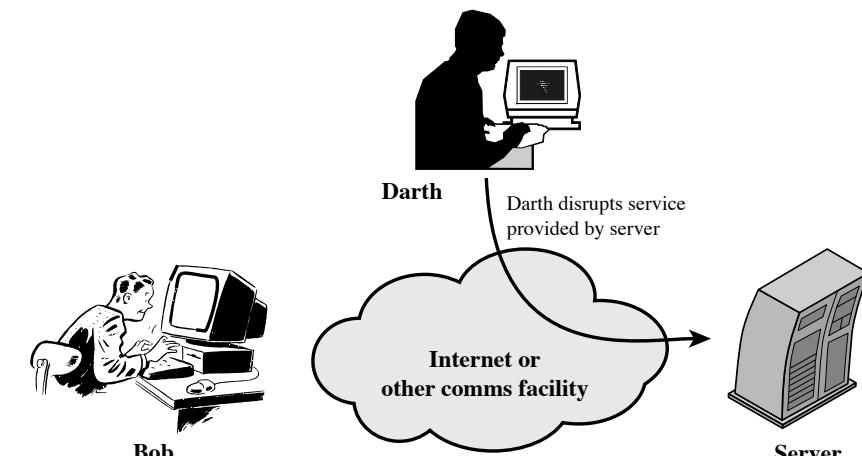




Modificación



Generación



Interrupción

SERVICIOS Y MECANISMOS DE SEGURIDAD

- Los servicios de seguridad ponen en funcionamiento las políticas de seguridad
- Algunas definiciones más precisas para este concepto:

“A processing or communication service that is provided by a system to give a specific kind of protection to system resources”

RFC 2828: Internet Security Glossary

“A service, provided by a layer of communicating open systems, which ensures adequate security of the systems or the data transfers”

ISO 7498-2: Basic Reference Model -- Part 2: Security Architecture

ITU X.800: Security Architecture for Open Systems Interconnection for CCITT Applications

- Los estándares ISO 7498-2 e ITU X.800 dividen los servicios de seguridad en **cinco categorías**, y a partir de ahí distinguen **catorce servicios específicos**
- Las categorías son:
 - Confidencialidad de datos
 - Autenticación
 - Integridad de datos
 - No-repudio
 - Control de acceso (Autorización)

DATA CONFIDENTIALITY

The protection of data from unauthorized disclosure.

Connection Confidentiality

The protection of all user data on a connection.

Connectionless Confidentiality

The protection of all user data in a single data block

Selective-Field Confidentiality

The confidentiality of selected fields within the user data on a connection or in a single data block.

Traffic-flow Confidentiality

The protection of the information that might be derived from observation of traffic flows.

Se usa este servicio porque no deseo que otros usuarios conozcan:

- mails que envío o chats
- mi DNI o número de la Seg. Social
- mi número de tarjeta de crédito
- mis datos médicos
- las webs que visito, lo que compro y dónde viajo
- mi salario
- lo que voto
- mis gustos (música, cine, ...)

- Otros ejemplos:

- Coca-Cola no desea que se conozca su fórmula
- Las empresas quieren proteger sus tecnologías
- Los gobiernos quieren mantener en secreto sus planes

AUTHENTICATION

The assurance that the communicating entity is the one that it claims to be.

Peer Entity Authentication

Used in association with a logical connection to provide confidence in the identity of the entities connected.

Data-origin Authentication

In a connectionless transfer, provides assurance that the source of received data is as claimed.

- Se usa este servicio porque quiero estar seguro de que las entidades con las que interactúo son quienes dicen ser:
 - mi amiga Alice
 - mi médico
 - Amazon
 - ...
- Es decir, quiero tener garantías de que nadie está suplantando la identidad de mi interlocutor

DATA INTEGRITY

The assurance that data received are exactly as sent by an authorized entity (i.e., contain no modification, insertion, deletion, or replay).

Connection Integrity with Recovery

Provides for the integrity of all user data on a connection and detects any modification, insertion, deletion, or replay of any data within an entire data sequence, with recovery attempted.

Connection Integrity without Recovery

As above, but provides only detection without recovery.

Selective-Field Connection Integrity

Provides for the integrity of selected fields within the user data of a data block transferred over a connection and takes the form of determination of whether the selected fields have been modified, inserted, deleted, or replayed.

Connectionless Integrity

Provides for the integrity of a single connectionless data block and may take the form of detection of data modification. Additionally, a limited form of replay detection may be provided.

Selective-Field Connectionless Integrity

Provides for the integrity of selected fields within a single connectionless data block; takes the form of determination of whether the selected fields have been modified.



- Se usa este servicio porque no deseo que:
 - los mails o chats que envío o recibo sean modificados o falsificados
 - alguien borre (conscientemente o no) una parte de mis registros médicos
 - se puedan falsificar las órdenes que envío a mi banco para realizar pagos/cobros
 - alguien pueda modificar mi declaración de Hacienda cuando la relleno/envío por la Web

NONREPUDIATION

Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication.

Nonrepudiation, Origin

Proof that the message was sent by the specified party.

Nonrepudiation, Destination

Proof that the message was received by the specified party.

- Se usa este servicio porque deseo:
 - tener pruebas de que ha ocurrido cierto evento:
 - envío de un ítem específico de información
 - recepción de un ítem específico de información
 - tener pruebas del instante exacto en que ha tenido lugar ese evento
 - tener pruebas de qué entidades han intervenido en el evento
 - conocer cualquier información adicional específicamente asociada al evento

ACCESS CONTROL

The prevention of unauthorized use of a resource (i.e., this service controls who can have access to a resource, under what conditions access can occur, and what those accessing the resource are allowed to do).

- Se usa este servicio porque deseo:
 - Permitir el acceso a mis recursos de usuarios autorizados
 - Denegar acceso a mis recursos de usuarios desconocidos
 - Limitar y monitorizar el uso de cierto recurso
 - Definir reglas de acceso
 - Garantizar el uso de credenciales correctos de acceso
 - en forma
 - en tiempo

- Dentro de una comunicación, estos servicios de seguridad se pueden proporcionar en distintas capas del modelo de referencia OSI, como indica la siguiente tabla:

Service / Layer	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5/6	Layer 7
Entity authentication			Y	Y		Y
Origin authentication			Y	Y		Y
Access control			Y	Y		Y
Connection confidentiality	Y	Y	Y	Y		Y
Connectionless confidentiality		Y	Y	Y		Y
Selective field confidentiality						Y
Traffic flow confidentiality	Y		Y			Y
Connection integrity with recovery				Y		Y
Connection integrity without recovery			Y	Y		Y
Selective field connection integrity						Y
Connectionless integrity			Y	Y		Y
Selective field connectionless integrity						Y
Non-repudiation of origin						Y
Non-repudiation of delivery						Y

- Por otro lado, un **mecanismo de seguridad** proporciona soporte a un servicio de seguridad
 - Definición:

*“A process (or a device incorporating such a process) that can be used in a system to **implement a security service** that is provided by or within the system”*
- RFC 2828: Internet Security Glossary
- Los estándares ISO 7498-2 e ITU X.800 distinguen entre dos tipos de mecanismos de seguridad:
 - **específicos**: están implementados en una capa específica de la pila de protocolos
 - **ubicuos**: no son específicos de ninguna capa en particular

ESPECÍFICOS

Cifrado
Firma digital
Control de acceso
Integridad del dato
Autenticación
Padding
Control de enrutamiento
**Tercera persona de confianza
(Trusted Third Party)**

UBÍCUOS

Controles de seguridad
Etiqueta de seguridad
Certificación, validación, simulación
Detección y prevención de eventos
Auditoría y accountability
Recuperación de la seguridad
Forense
Gestión de la confianza, reputación
...

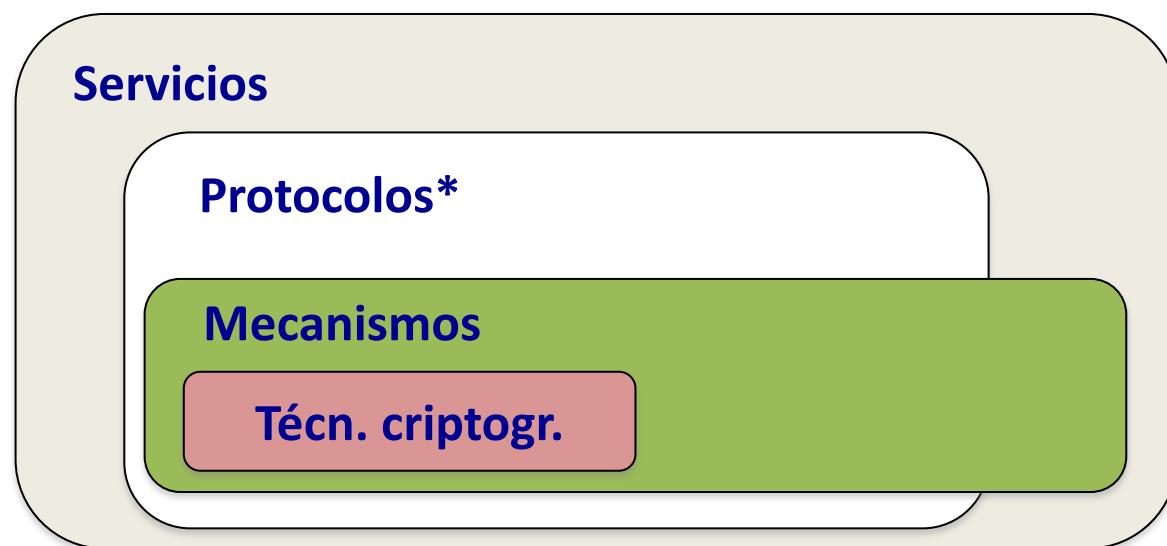
Mechanism Service	Encipherment	Digital signature	Access control	Data integrity	Authentication exchange	Traffic padding	Routing control	Notarization
Peer entity authentication	Y	Y	.	.	Y	.	.	.
Data origin authentication	Y	Y
Access control service	.	.	Y
Connection confidentiality	Y	Y	.
Connectionless confidentiality	Y	Y	.
Selective field confidentiality	Y
Traffic flow confidentiality	Y	Y	Y	.
Connection Integrity with recovery	Y	.	.	Y
Connection integrity without recovery	Y	.	.	Y
Selective field connection integrity	Y	.	.	Y
Connectionless integrity	Y	Y	.	Y
Selective field connectionless integrity	Y	Y	.	Y
Non-repudiation. Origin	.	Y	.	Y	.	.	.	Y
Non-repudiation. Delivery	.	Y	.	Y	.	.	.	Y

· The mechanism is considered not to be appropriate.

Y Yes: the mechanism is considered to be appropriate, either on its own or in combination with other mechanisms.

Note – In some instances, the mechanism provides more than is necessary for the relevant service but could nevertheless be used.

- Resumiendo, un servicio de seguridad está basado de:
 - Un **protocolo de seguridad*** (opcional) es un conjunto de reglas y formatos que determinan la información que se intercambian dos (o más) entidades con objeto de proporcionar un servicio de seguridad
 - Los **mecanismos de seguridad** son las piezas básicas con las que se construyen protocolos de seguridad
 - Los mecanismos de seguridad se apoyan en **técnicas criptográficas**



Referencias bibliográficas

Bibliografía básica

- "User's Guide To Cryptography And Standards"
Alex W. Dent, Chris J. Mitchell
Artech House, 2004
- "Handbook of Applied Cryptography"
Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone,
CRC Press, 1996

Bibliografía complementaria

- *ISO 7498-2*
 - Information processing systems -- Open Systems Interconnection -- Basic Reference Model -- Part 2: Security Architecture, 1989.
- *RFC 2828*
 - RFC2828: Internet Security Glossary, R. Shirey, May 2000.
- *ITU-T X.800*
 - Recommendation X.800: Security Architecture for Open Systems Interconnection for CCITT Applications, ITU, 1991.
- *ITU-T X.509*
 - Recommendation X.509: Information technology – Open systems interconnection – The Directory: Public-key and attribute certificate frameworks, ITU, 2005

SEGURIDAD DE LA INFORMACIÓN

TEMA 2

**TÉCNICAS CRIPTOGRÁFICAS BÁSICAS
(Y SERVICIOS DE SEGURIDAD ASOCIADOS)**

Indice del tema (I)

- Introducción a la criptografía clásica
 - Cifrados por sustitución y transposición. Ejemplos
 - Cifrado producto
 - Cifrado Vernam (one-time pad)
- Algoritmos simétricos
 - Fundamentos
 - Algoritmo DES
 - Algoritmo triple-DES
 - Algoritmo AES
 - Otros algoritmos simétricos
 - Modos de operación para algoritmos simétricos
 - Ventajas y desventajas de los algoritmos simétricos

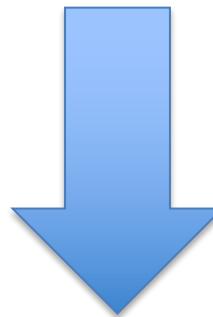
Indice del tema (II)

- Algoritmos asimétricos (o de clave pública)
 - Cifrado/descifrado
 - Firma Digital
 - Intercambio de Claves
 - Algoritmo de Diffie-Hellman
 - Algoritmo RSA
- Otras primitivas criptográficas
 - Funciones hash
 - Códigos de autenticación de mensajes
- Referencias bibliográficas

Introducción a la criptografía

Criptografía, Criptoanálisis, Criptología

- Ya se sabe por el tema anterior que un **algoritmo de cifrado** es un mecanismo fundamental para el desarrollo de servicios de seguridad, como puede ser la confidencialidad



- Criptografía: ciencia que estudia cómo mantener la seguridad en los mensajes (M)
 - usando, entre otros mecanismos, los algoritmos de cifrado
- Criptoanálisis: ciencia que estudia cómo romper los textos cifrados
- Criptología: Criptografía + Criptoanálisis

- El algoritmo de **cifrado** es un mecanismo que transforma un texto en claro en texto ininteligible
 - Su objetivo es dar cobertura al servicio de CONFIDENCIALIDAD
 - El **ALGORITMO DE CIFRADO**, caracterizado por **E** (del inglés “encrypt”), opera sobre el **texto en claro M** (mensaje) para producir el **texto cifrado C** (criptograma)



- La transformación inversa de un texto cifrado a un texto en claro, se denomina ALGORITMO DE DESCIFRADO
 - El algoritmo se denota por la letra D (“decrypt”) y opera sobre C para producir el mensaje M

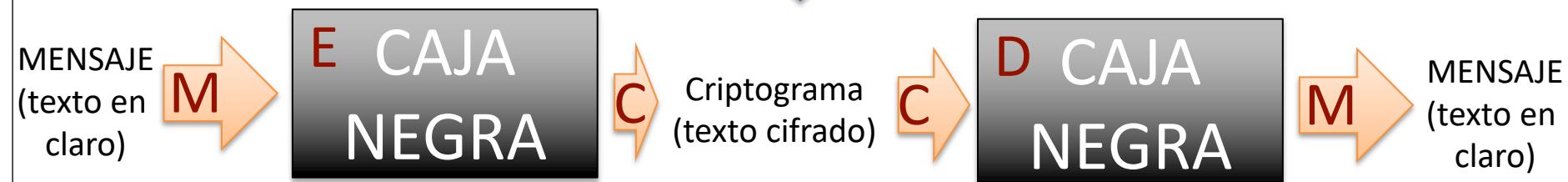


- Se cumple también que:

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

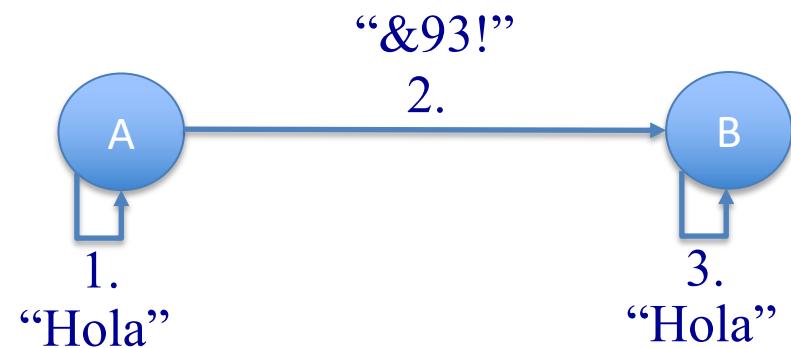
$$D(E(M)) = M$$

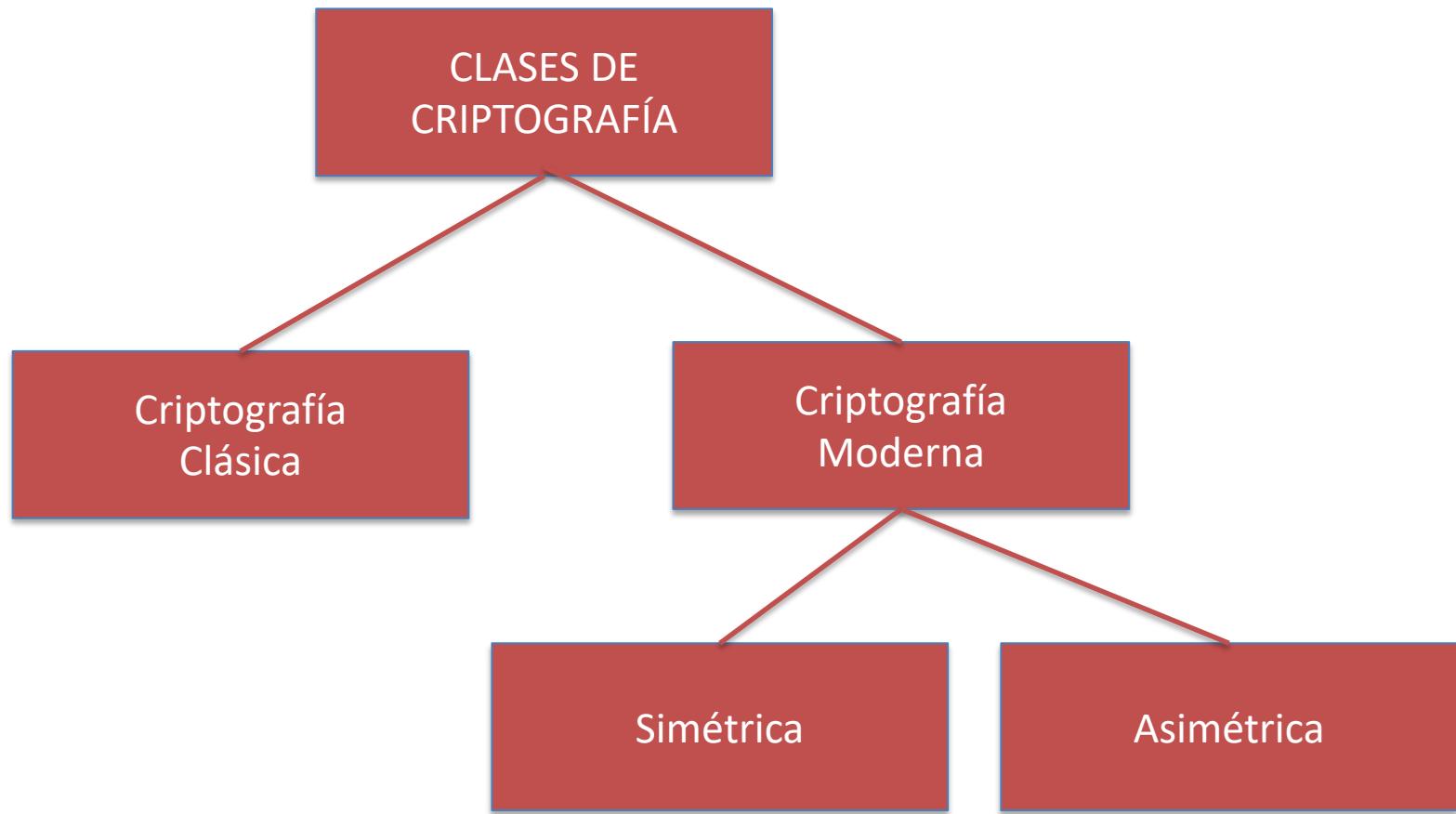
Otra forma
de verlo es



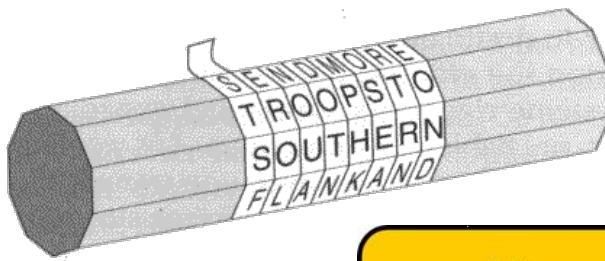
- Ejemplo:

1. A: "Hola"
2. A → B: E("Hola") → "&93!"
3. B: D("&93!") → "Hola"





Criptografía clásica



- Antes de la existencia de ordenadores, la criptografía clásica consistía en algoritmos basados en caracteres
- Estos algoritmos se basaban en dos técnicas principales:
 - **Cifrado por sustitución:**
 - Cada carácter del texto en claro se sustituye por otro carácter en el texto cifrado
 - $A \rightarrow V$
 - $V \rightarrow W$
 - ...
 - **Cifrado por transposición:**
 - Realizar una permutación con respecto a las posiciones que ocupan los símbolos en el mensaje en claro
 - HOLA \rightarrow ALHO

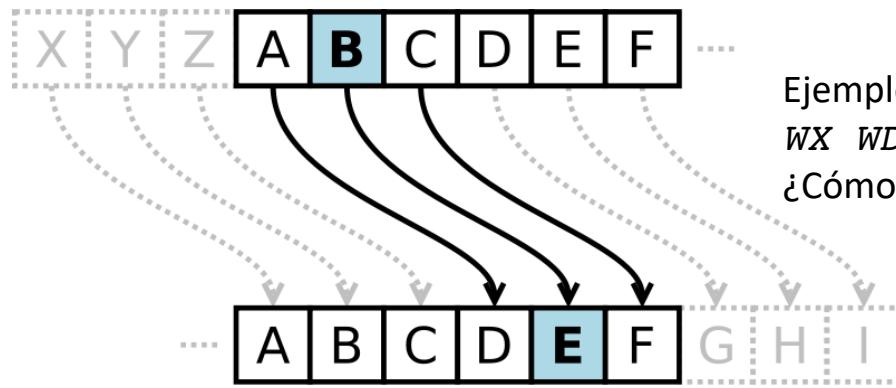
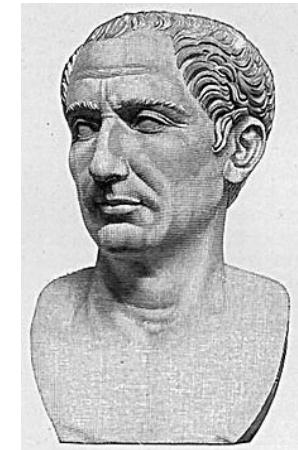
- Antes de la existencia de ordenadores, la criptografía clásica consistía en algoritmos basados en caracteres
- Estos algoritmos se basaban en dos técnicas principales:
 - **Cifrado por sustitución:**
 - Cada carácter del texto en claro se sustituye por otro carácter en el texto cifrado
 - $A \rightarrow V$
 - $V \rightarrow W$
 - ...
 - **Cifrado por transposición:**
 - Realizar una permutación con respecto a las posiciones que ocupan los símbolos en el mensaje en claro
 - HOLA \rightarrow ALHO

VAMOS A VER ALGUNOS EJEMPLOS

Ejemplo 1: cifrado por sustitución César

- Objetivo:
 - Cada carácter de texto en claro se reemplaza por aquel posicionado a tres posiciones a la derecha (módulo 27)

$$C: M \rightarrow M + 3 \text{ (mod 27)}$$



Ejemplo texto cifrado:

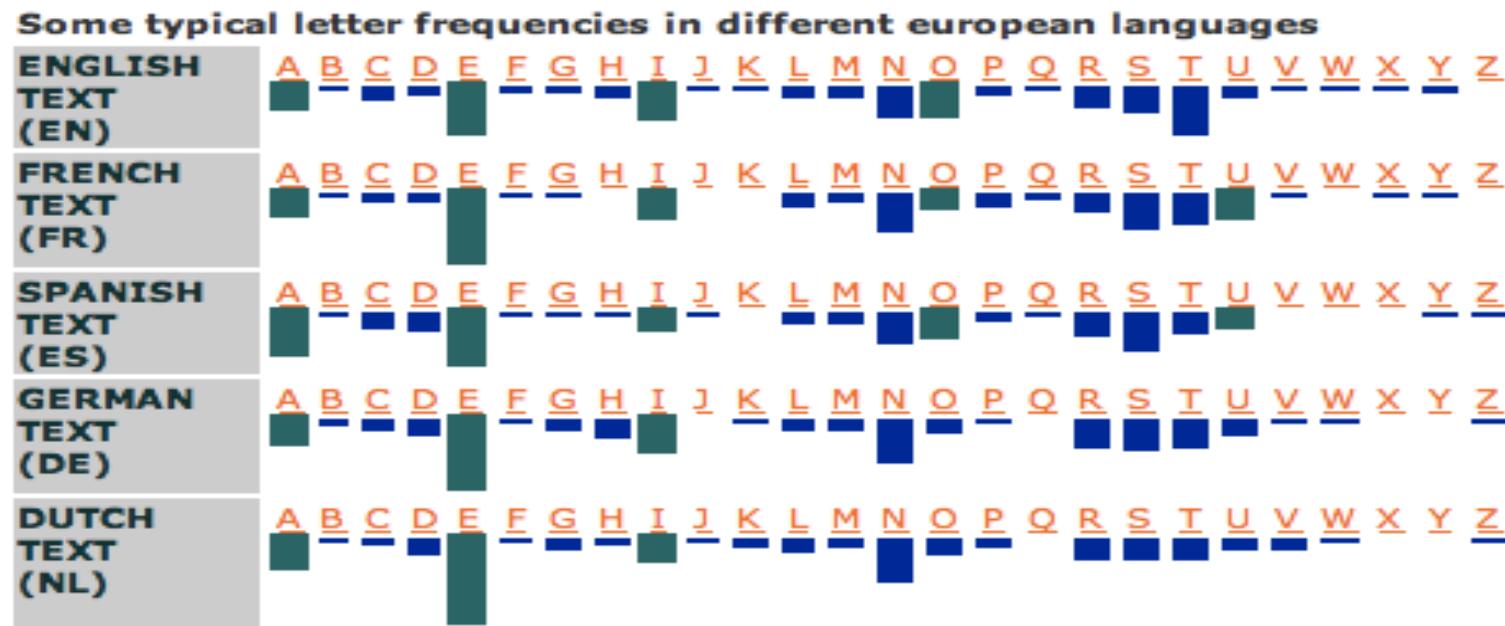
WX WDPHELHQ, EUXWR, KLMR PLR

¿Cómo sería el descifrado de este texto cifrado?

- Se puede generalizar a un sistema de cifrado con 27 posibles combinaciones

$$C: M \rightarrow M + i \text{ (mod 27)} \quad 1 \leq i \leq 27$$

- El algoritmo proporciona ventajas al criptoanalista, porque la frecuencia de aparición de las letras es bien conocida. Así:



English

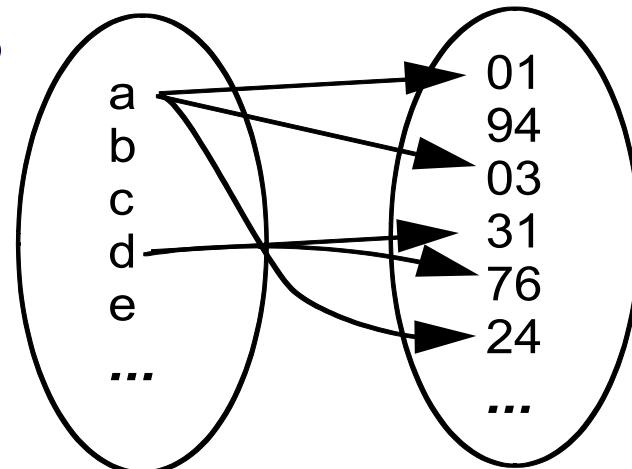
E	12.4%	H	6.5%	U	2.7%	G	2.0%	K	0.7%
T	8.9%	S	6.2%	M	2.5%	Y	2.0%	Q	0.1%
A	8.0%	R	6.1%	W	2.3%	P	1.6%	X	0.1%
O	7.6%	D	4.6%	C	2.2%	B	1.3%	J	0.1%
N	7.0%	L	3.6%	F	2.2%	V	0.8%	Z	0.0%
I	6.7%								

Spanish

E	13.0%	S	6.9%	U	3.6%	V	1.0%	J	0.3%
A	11.1%	T	5.3%	P	3.0%	F	0.8%	Z	0.3%
O	9.7%	C	5.2%	M	2.9%	Y	0.7%	X	0.2%
I	8.2%	D	4.5%	G	1.4%	H	0.6%	W	0.1%
N	8.0%	L	3.6%	B	1.3%	Q	0.6%	K	0.0%
R	7.7%								

Ejemplo 2: cifrado por sustitución homofónico

- Se basa en la idea de asignar a un símbolo del alfabeto fuente varios del alfabeto cifrado, solventando el problema de la frecuencia de letras
 - Correspondencia uno a muchos ⇒ al cifrar un mensaje podemos obtener varios criptogramas
 - Ejemplo:



Letra	% (redondeado)	Símbolos asignados
A	8	10, 11, 23, 45, 76, 79, 87, 98
L	6	02, 15, 21, 25, 56, 60
N	3	44, 63, 71
O	8	04, 16, 28, 29, 37, 52, 69, 90
P	2	30, 88
T	2	24, 77

“PLATON” se cifra como “882110772963”

Ejemplo 3: cifrado por sustitución POLIalfabética

- Alfabeto para posiciones impares:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
V	K	L	s	w	e	M	N	U	f	a	b	Q	r	S	t	o	j	I	P	x	s	Ñ	h	d	Z	W

- Alfabeto para posiciones pares:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
z	g	X	Y	a	b	D	K	L	P	q	s	t	U	O	Ñ	Q	k	e	c	H	W	M	N	f	g	i

- Cifrado del texto: “*HOLA A TODOS*”

H O L A A T O D O S

- #### • Descifrado:

Ejemplo 3: cifrado por sustitución POLIalfabética

- Alfabeto para posiciones impares:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
V	K	L	s	w	e	M	N	U	f	a	b	Q	r	S	t	o	j	I	P	x	s	Ñ	h	d	Z	W

- Alfabeto para posiciones pares:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
z	g	X	Y	a	b	D	K	L	P	q	s	t	U	O	Ñ	Q	k	e	c	H	W	M	N	f	g	i

- Cifrado del texto: “**HOLAA TODOS**”

H	O	L	A	A	T	O	D	O	S
N	Ñ	b	z	V	H	t	Y	t	c

- Descifrado:

N	Ñ	b	z	V	H	t	Y	t	c
H	O	L	A	A	T	O	D	O	S

- Antes de la existencia de ordenadores, la criptografía clásica consistía en algoritmos basados en caracteres
- Estos algoritmos se basaban en dos técnicas principales:
 - **Cifrado por sustitución:**
 - Cada carácter del texto en claro se sustituye por otro carácter en el texto cifrado
 - $A \rightarrow V$
 - $V \rightarrow W$
 - ...
 - **Cifrado por transposición:**
 - Realizar una permutación con respecto a las posiciones que ocupan los símbolos en el mensaje en claro
 - HOLA \rightarrow ALHO

VAMOS A VER ALGUNOS EJEMPLOS

Ejemplo 4: cifrado por transposición

- Objetivo: el texto en claro se escribe como secuencia de filas (con una cierta profundidad) y se lee como secuencia de columnas

Restricción
a nivel de
fila



H	O	L	A	M	U	N
D	O	Y	A	S	Í	C
O	N	T	O	D	O	.

- Ejemplo:
 - “EN ANDALUCIA, EL MULHACEN Y EL VELETA, SON LAS MONTAÑAS MAS ALTAS”



ENANDALUCIAELMULHACENYELVE
LETASONLASMONTAÑASMASALTAS

Hay que quitar los espacios,
los símbolos, etc.

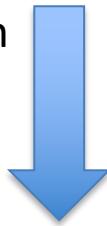
- Mensaje cifrado:

ELNEATNADSAOLNULCAISAMEOLNMTUALÑHAASCMEANSYAELLTVAES

Ejemplo 5: cifrado por transposición con CLAVE

- Se podría complicar el procedimiento anterior estableciendo una restricción en el número de columnas cuyo valor va a depender del tamaño que tenga una **clave**

Restricción
a nivel de
fila



H	O	L	A	M	U	N
D	O	Y	A	S	Í	C
O	N	T	O	D	O	.



Restricción a nivel de columna

- Ejemplo:
 - Texto en claro: “**HOLA A TODOS, QUE TENGÁIS UN BUEN DÍA**”
 - Clave: ”**SECRETO**” con un tamaño de 7

S	E	C	R	E	T	O

Ejemplo 5: cifrado por transposición con CLAVE

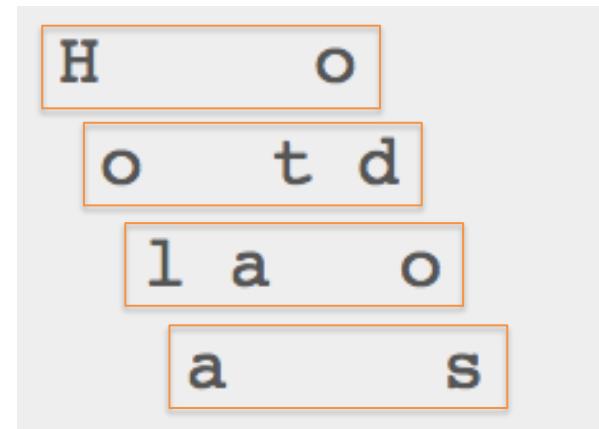
- Ejemplo:
 - Texto en claro: “**HOLA A TODOS, QUE TENGÁIS UN BUEN DÍA**”
 - Clave: ”**SECRETO**” con un tamaño de 7

S	E	C	R	E	T	O
H	O	L	A	A	T	O
D	O	S	Q	U	E	T
E	N	G	A	I	S	U
N	B	U	N	D	I	A

- Podemos fortalecer la seguridad si añadimos más restricciones:
 - Para el cifrado se puede poner la condición siguiente: se va a ir cogiendo las letras de aquellas columnas por orden alfabético del secreto, es decir: **C, E, E, O, R, S, T**, resultando en: “_____”
 - Solución: “**LSGUOONBAUIDOTUAQAQNHDENTESI**”

Ejemplo 6: cifrado por transposición Railfence

- El cifrado consiste
 - en escribir diagonalmente el texto en claro con una profundidad P específica
 - el criptograma se escribe leyendo las filas
- Ejemplo: $M = \text{"Hola a todos"}$, con una profundidad de $P=4$, entonces el criptograma es: **Hoot d laoas**
por simplemente computar:



- Antes de la existencia de ordenadores, la criptografía clásica consistía en algoritmos basados en caracteres
- Estos algoritmos se basaban en dos técnicas principales:
 - **Cifrado por sustitución:**
 - Cada carácter del texto en claro se sustituye por otro carácter en el texto cifrado
 - $A \rightarrow V$
 - $V \rightarrow W$
 - ...
 - **Cifrado por transposición:**
 - Realizar una permutación con respecto a las posiciones que ocupan los símbolos en el mensaje en claro
 - HOLA \rightarrow ALHO

SE PUEDEN COMBINAR

Cifrado Producto

- **Combinación de algoritmos sustitución y transposición**
- Se pueden considerar como la aplicación sucesiva de varios cifrados E_i

$$E = E_1 \cdot E_2 \cdot \cdots \cdot E_r$$

$$E(M) = E_1(E_2(\cdots(E_r(M))))$$

- La composición de funciones de descifrado D_i se realiza en orden inverso

$$D = D_r \cdot D_{r-1} \cdots D_1$$

$$M = D(C) = D_r(D_{r-1}(\dots(D_1(C))))$$

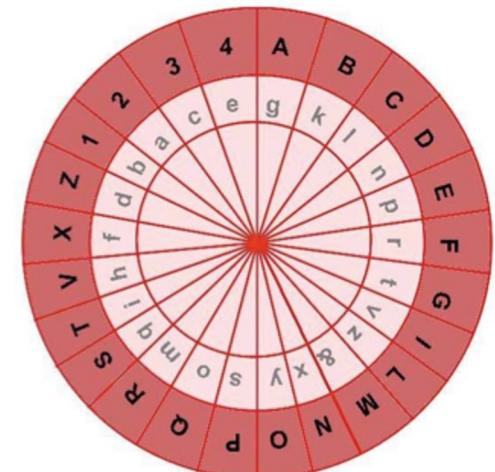
- Es un esquema utilizado para obtener un alto grado de seguridad con sistemas relativamente sencillos aplicados reiterativamente
- Dan lugar a sistemas de cifrado complejos, seguros y difíciles de atacar, así como fácilmente trasladables a un ordenador

Ejemplo 7: métodos polialfabéticos y nomenclátores

- Para complicar el proceso de cifrado, se puede hacer uso del disco de Alberti junto con nomenclátores, los cuales consisten en asociar a determinados palabras códigos específicos

Felipe II	123
Rey	124
Walshingan	122

- Se desea descifrar el siguiente texto: “*baa&hpmiyvsoiyrlxckngkl*”
- Uso del disco y condiciones:
 - Cada diez letras descifradas, se ha de girar el disco externo (de las mayúsculas) dos posiciones en el sentido de las agujas del reloj
 - En el disco de Alberti, la **u** se identifica con la **v** al cifrar
 - Al descifrar, por el sentido de la frase, se puede conocer si se ha de escribir una u otra letra

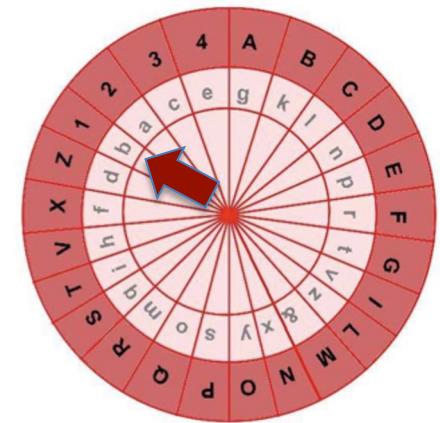


Ejemplo 7: métodos polialfabéticos y nomenclátóres

- Funcionamiento para cifrar:
 - Posicionar los disco en el estado inicial

b	a	a	&	H	p	m	i	Y	V
1	2								

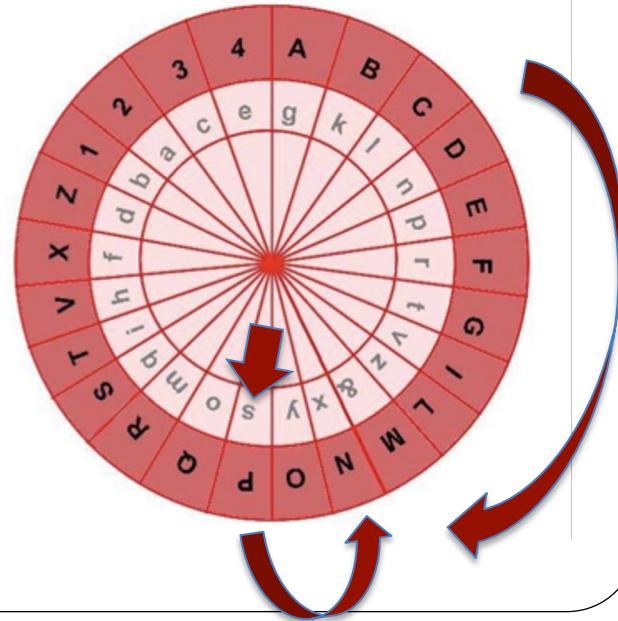
"baa&hpmiyvsvoiylrlxckngkl"



- Con el disco externo girar 2 posiciones en el sentido de las agujas del reloj (sólo en cada diez letras descifrada):

s	v	o	i	Y	l	r	l	x	c
N	F								

"baa&hpmiyvsvoiylrlxckngkl"

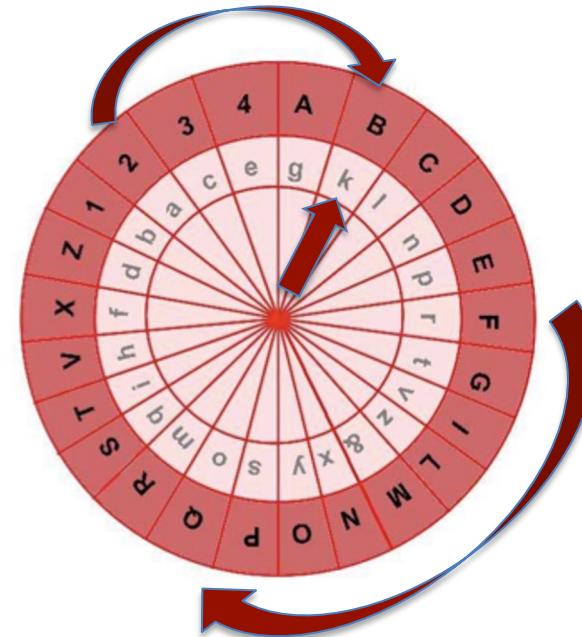


Ejemplo 7: métodos polialfabéticos y nomenclátores

- Con el disco externo volver a girar 2 posiciones en el sentido de las agujas del reloj:

k	n	g	k	L
2	4	1	2	3

"baa&hpmiyvsvoiylrlxckngkl"



- Por consiguiente, el texto en claro es:

b	a	a	&	H	p	m	i	Y	v
1	2	2	M	V	E	R	T	O	I

s	v	o	I	Y	l	r	l	x	c
N	F	O	R	M	A	D	A	L	1

k	n	g	k	L
2	4	1	2	3

"1 2 2 M V E R T O I N F O R M A D A L 1 2 4 1 2 3"

- Si, además, añadimos los nomenclátores + la restricción de la V → U:

Felipe II	123
Rey	124
Walshingan	122

"WALSHINGAN MUERTO INFORMAD AL REY FELIPE II"

Ejemplo 8: Cifrado Vernam

- Aplica el concepto de **one-time pad (OTP)**
- Un one-time pad es un *conjunto infinito y no repetitivo* de letras aleatorias
- Cada letra del pad se usa para cifrar una única letra del texto en claro, en módulo n (longitud del alfabeto)



Texto : T H I S I S S E C R E T
OTP: X V H E U W N O P G C Z

Cifrado : Q C P W C O F S R X H S

One-time pad booklet and microdot reader, concealed in a toy truck and used by an illegal agent that operated in Canada
© Canadian Security Intelligence Service

- Ejemplo:
 - Aquí se observan grupos de tres filas, que se corresponden con texto en claro (en decimal), clave y criptograma

0 3 2 3 4 6 2 7 6 2	0 9 7 6 2	6 3 1 8 3	7 6 4 8 7	0 6 2 6 7	6 7 0 6 7
4 3 8 6 4	6 8 4 3 2	4 6 0 3 7	8 7 7 7 2	3 8 2 7 7	0 3 0 2 3
6 9 7 4 0	1 0 3 7 9	4 9 7 1 3	4 0 5 1 4	4 4 6 7 7	9 0 2 8 0
2 3 7 7 7	6 8 2 7 9	6 5 8 6 7	0 5 7 0 9	5 8 3 9 5	7 6 3 8 8
6 2 7 7 3	4 1 1 6 5	4 2 2 3 7	4 7 4 3 5	6 2 1 3 3	7 1 3 5 0
8 5 6 8 0	0 9 3 3 8	0 7 1 1 4	4 5 1 5 4	1 0 4 2 8	7 7 7 8
6 3 0 9 5	8 7 0 6 9	5 8 6 7 2	7 1 5 7 8	7 2 8 4 3	9 3 7 0 7
4 7 7 7 9	0 7 8 8 1	4 9 1 2 6	6 0 0 9 8	6 2 2 9 2	8 8 6 5 6
5 1 7 7 9	8 4 8 6 9	3 6 9 9 7	2 1 3 1 6	3 4 7 2 2	7 1 3 7 5
3 1 7 2 6	5 0 8 3 3	8 2 0 5 8	2 8 7 2 7	8 8 6 2 6	3 1 8 3 3
4 2 7 7 1	7 9 2 1 3	7 6 4 3 9	7 1 3 3 0	9 2 3 4 0	7 6 2 3 0
4 2 7 6	6 9 2 0 4	5 0 2 9 1	9 4 3 1 7	5 6 9 1 2	7 3 3 7 7
7 7 7 7 3	2 8 3 6 6	5 8 7 7 6	4 6 7 6 0	7 7 6 1 3	0 5 8 6 7
3 5 6 0 1	3 5 6 0 1	7 4 5 0 8	5 2 0 6 0	5 7 7 2 1	5 2 5 0 4
5 3 9 6 2	5 3 9 6 2	4 2 4 7 4	9 8 7 2 0	9 9 4 4 4	5 7 3 6 1
2 4 7 7 3	7 8 2 0 8	7 6 9 2 6	3 9 3 7 6	3 2 6 7 6	0 3 7 4 6
6 7 7 6 1	0 0 6 2 1	0 7 4 6 8	7 8 5 7 9	6 7 2 3 0	6 7 8 0 8
8 0 0 0 1	8 8 2 2 9	7 3 3 2 9	0 3 8 8 1	9 9 8 0 6	2 0 7 4 4
1 7 4 3 9	7 6 8 5 6	7 8 7 6 7	2 6 7 9 6	5 9 3 7 7	7 3 7 6 7
2 3 8 9 2	3 0 5 4 2	3 8 0 9 1	9 0 1 4 7	4 8 4 3 3	4 6 6 2 5
3 1 2 2 1	3 1 2 2 1	2 6 7 5 8	6 1 8 9 3	9 1 7 7 0	3 9 7 0 2
0 5 7 2 8	0 5 7 2 8	7 3 3 3 3	0 0 0 7 7	1 5 8 8 2	6 5 6 7 1
0 6 4 3 2	0 6 4 3 2	2 5 6 6 2	3 2 2 4 7	8 8 0 7 3	8 8 7 2 8
0 5 4 0 2	0 5 4 0 2	9 8 3 3 2	3 2 2 1 4	9 3 3 9 3	3 2 3 7 3

Fuente: <http://www.caslab.cl/che.php>

B	A	R	R	O	Y	C	Á	N	A	B	R	A	V	A
1	0	1	8	1	8	1	5	2	5	2	0	1	4	0
E	D	S	A	S	A	C	E	T	N	I	E	V	E	D
4	3	1	9	0	1	9	0	2	4	2	0	1	3	8
5	3	1	0	1	8	7	2	5	4	4	7	1	3	9
F	D	K	R	H	Y	E	E	H	N	J	V	V	Z	D

MClá
Clave (tan larga como el mensaje)
MClá + Clave
Criptograma

Fuente: <http://bit.ly/2cqBu8D>

Cifrado: (carácter del texto en claro + key) + mod 27

Descifrado: (carácter del criptograma - key) + mod 27

- Ejemplo:
 - En los ordenadores, el OTP aleatorio de longitud infinita se combina mediante XOR con el texto en claro. Ejemplo:

Texto en claro	1	1	0	0	1	0	1	1	0	0	0	1	1	0	1	0	0	1	1	1	0	1	1	\oplus
OTP	1	0	0	1	1	0	1	0	1	0	1	1	0	1	0	0	1	1	0	0	1	0	=	
Criptograma	0	1	0	1	0	0	0	1	1	0	1	0	1	1	1	0	1	0	1	0	0	1	\oplus	
OTP	1	0	0	1	1	0	1	0	1	0	1	1	0	1	0	1	0	0	1	1	0	0	=	
Texto en claro	1	1	0	0	1	0	1	1	0	0	0	1	1	0	1	0	0	1	1	1	0	1	1	

- Inconvenientes del cifrado Vernam:
 - las letras del OTP (o bits si se usa en ordenador) han de generarse aleatoriamente
 - el OTP no se vuelve a usar

Relación de ejercicios

1. Considerando el alfabeto inglés (sin incluir la ñ) y un desplazamiento de 3 posiciones para el proceso de cifrado o descifrado, aplicar la técnica de sustitución Caesar para cifrar el siguiente texto:

“EL PATIO DE MI CASA ES PARTICULAR”

SOLUCIÓN: HO SDWLR GH PL FDVD HV SDUWLFXODU

Relación de ejercicios

2. Dado el criptograma $C = \text{"FMIRZIRMHS E PE EW MKREXYVE HI WIKYVMHEH HI PE MRJSVQEGMSR"}$ descifrar el contenido del mismo, sabiendo, además, que hay que usar la técnica de sustitución Caesar con un desplazamiento de 4 posiciones modulo $n=26$ (Alfabeto inglés)

SOLUCIÓN: BIENVENIDO A LA ASIGNATURA DE SEGURIDAD
DE LA INFORMACIÓN

Relación de ejercicios

3. El siguiente algoritmo aplicará una sustitución monoalfabética, pero esta vez teniendo en cuenta la siguiente regla: $C_i = M_i + K_i \text{ mod } 26$ donde K representa una clave de longitud L . El objetivo es cifrar el texto original usando el alfabeto inglés

¿Cuál sería el criptograma del mensaje $M = \text{"HOLA AMIGOS"}$ usando una clave $K = \text{CIFRA}$?

Nota: se empieza a contar desde la posición 0 (A del alfabeto)

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

H	O	L	A	A	M	I	G	O	S
7	14	11	0	0	12	8	6	14	18
C	I	F	R	A	C	I	F	R	A
+2	+8	+5	+17	+0	+2	+8	+5	+17	+0
J	W	Q	R	A	O	Q	L	F	S
9	22	16	17	0	14	16	11	31→5	18

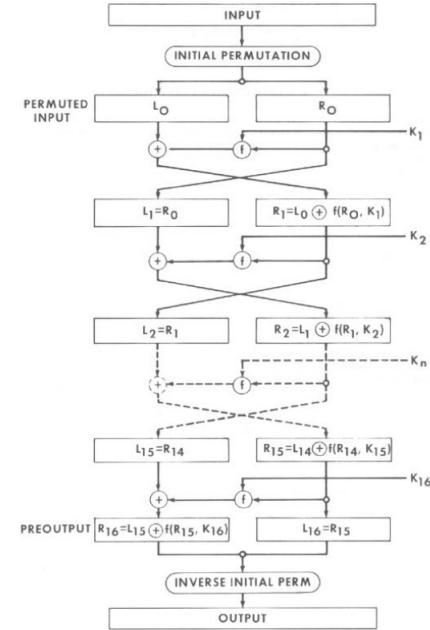
SOLUCIÓN: JWQR AOQLFS

Relación de ejercicios

4. Mediante la técnica Railfence, determinar el criptograma correspondiente al mensaje “*Nos han descubierto, debemos huir*” con una profundidad $P=7$ (alfabeto inglés) y sin contar espacios

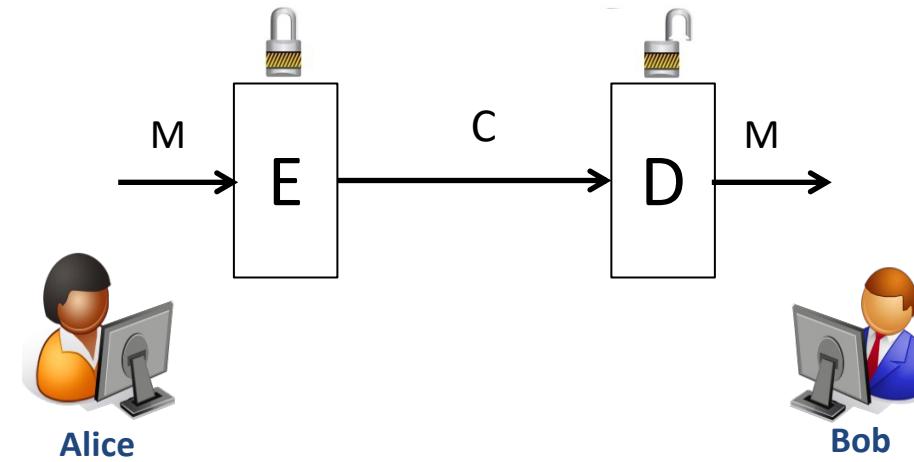
SOLUCIÓN: Nihobesusuroihctmrasoenedbde

Algoritmos simétricos

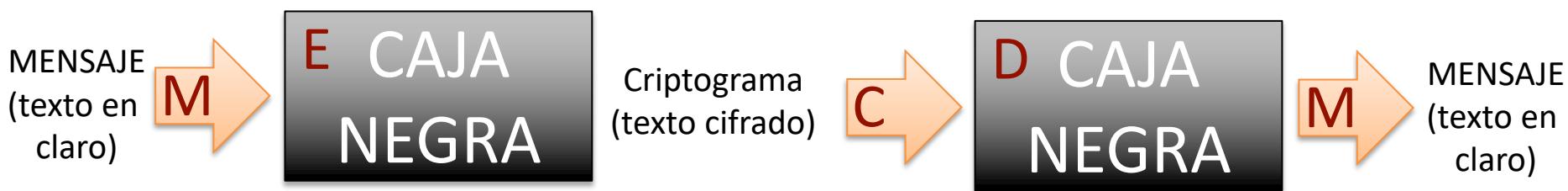


Fundamentos

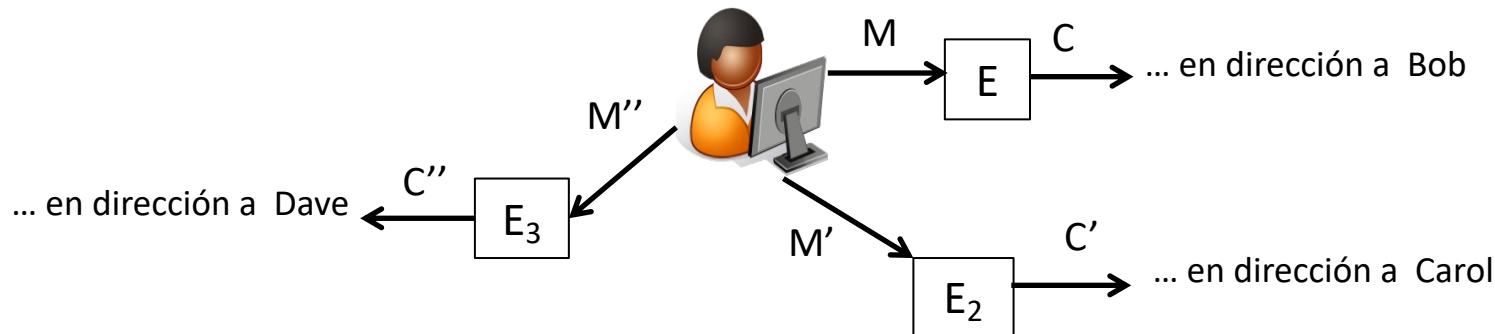
- En la mayoría de los ejemplos de la sección anterior la comunicación entre los usuarios puede representarse como sigue:



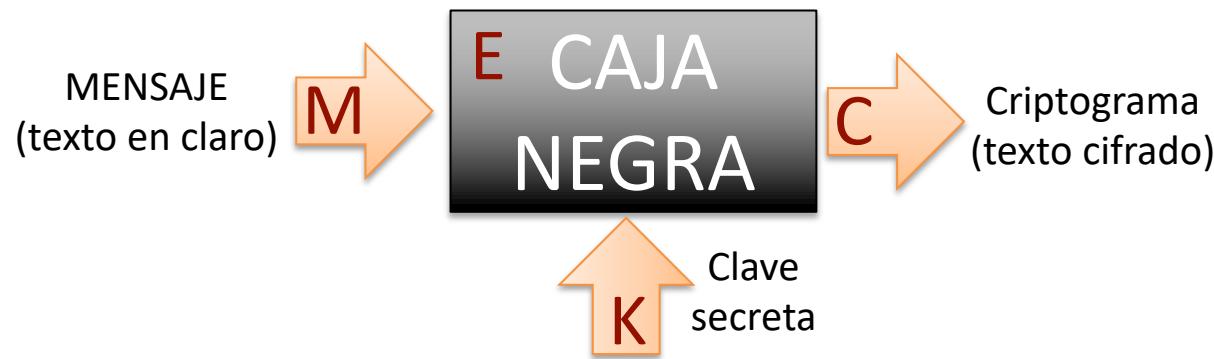
- Equivalente a lo que ya vimos antes:



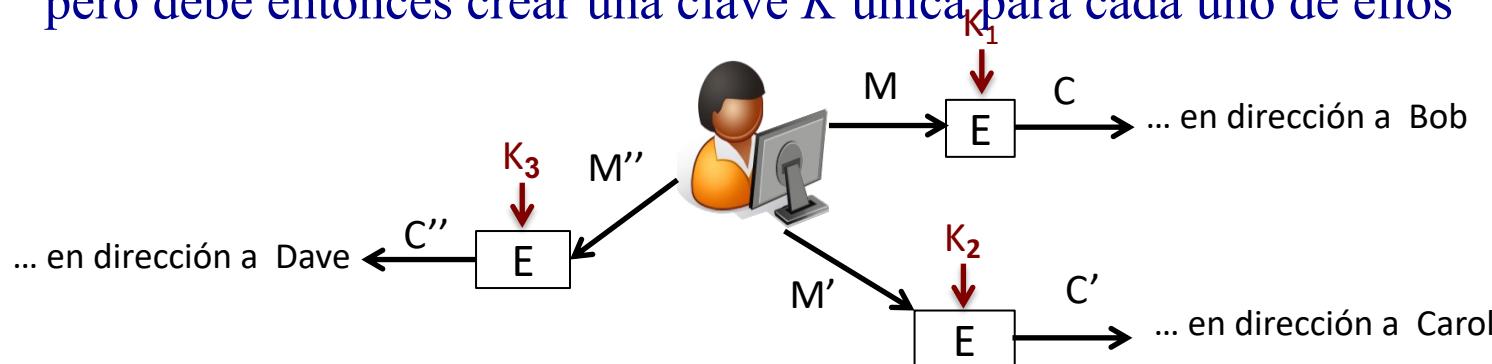
- El esquema anterior es útil siempre que se mantengan en “secreto” la transformación E y su inversa D
 - Esto es factible para un intercambio de información entre dos usuarios específicos (por ejemplo, *Alice* y *Bob*)
 - Sin embargo, esta forma de funcionamiento resulta **no escalable**
 - Cuando *Alice* necesita comunicar con alguien distinto de *Bob*, tendría que usar un algoritmo distinto (o una caja negra distinta) de tipo E como muestra la figura inferior
 - Esto también significa que *Alice* necesitará aplicar un algoritmo distinto (o una caja negra distinta) de tipo E PARA CADA USUARIO con quien contacta



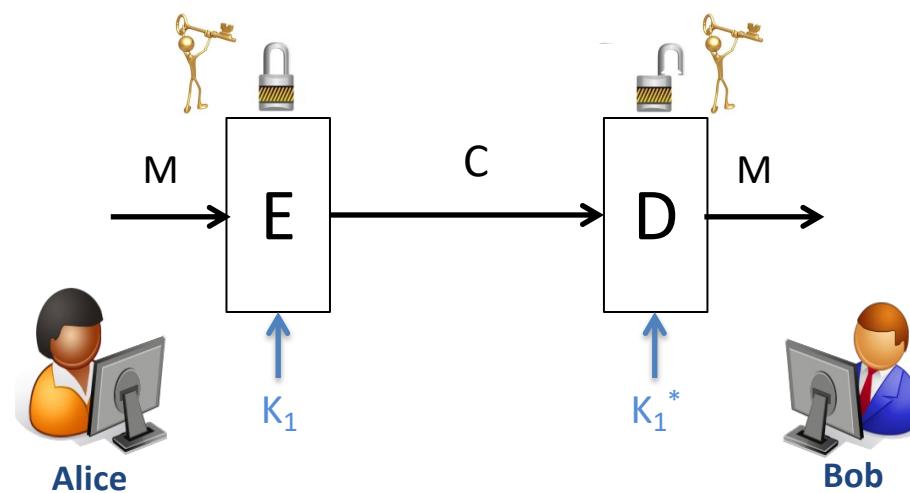
- El problema anterior se puede solucionar aplicando una **CLAVE SECRETA** K y aplicada con el algoritmo de cifrado E



- De esta forma, *Alice* puede usar el mismo algoritmo (o la misma caja negra) E en sus comunicaciones y con todos los usuarios (*Bob*, *Carol*, *Dave*, ...), pero debe entonces crear una clave K única para cada uno de ellos



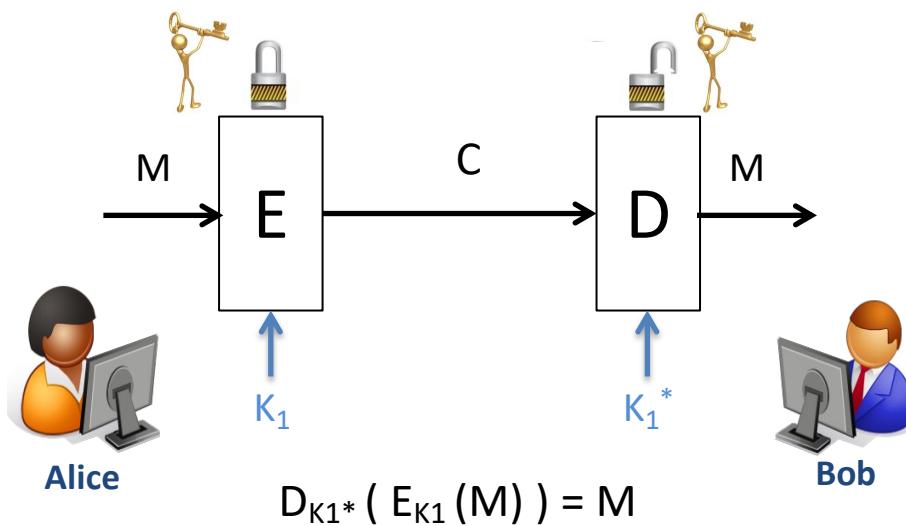
- Por tanto, el mismo algoritmo de descifrado D será usado por todos los receptores, pero cada uno necesitará la clave correspondiente de descifrado ($K_1^*, K_2^*, K_3^* \dots$)
- En resumen, para la comunicación específica entre *Alice* y *Bob*:



$$D_{K1^*} (E_{K1} (M)) = M$$

Recordatorio...

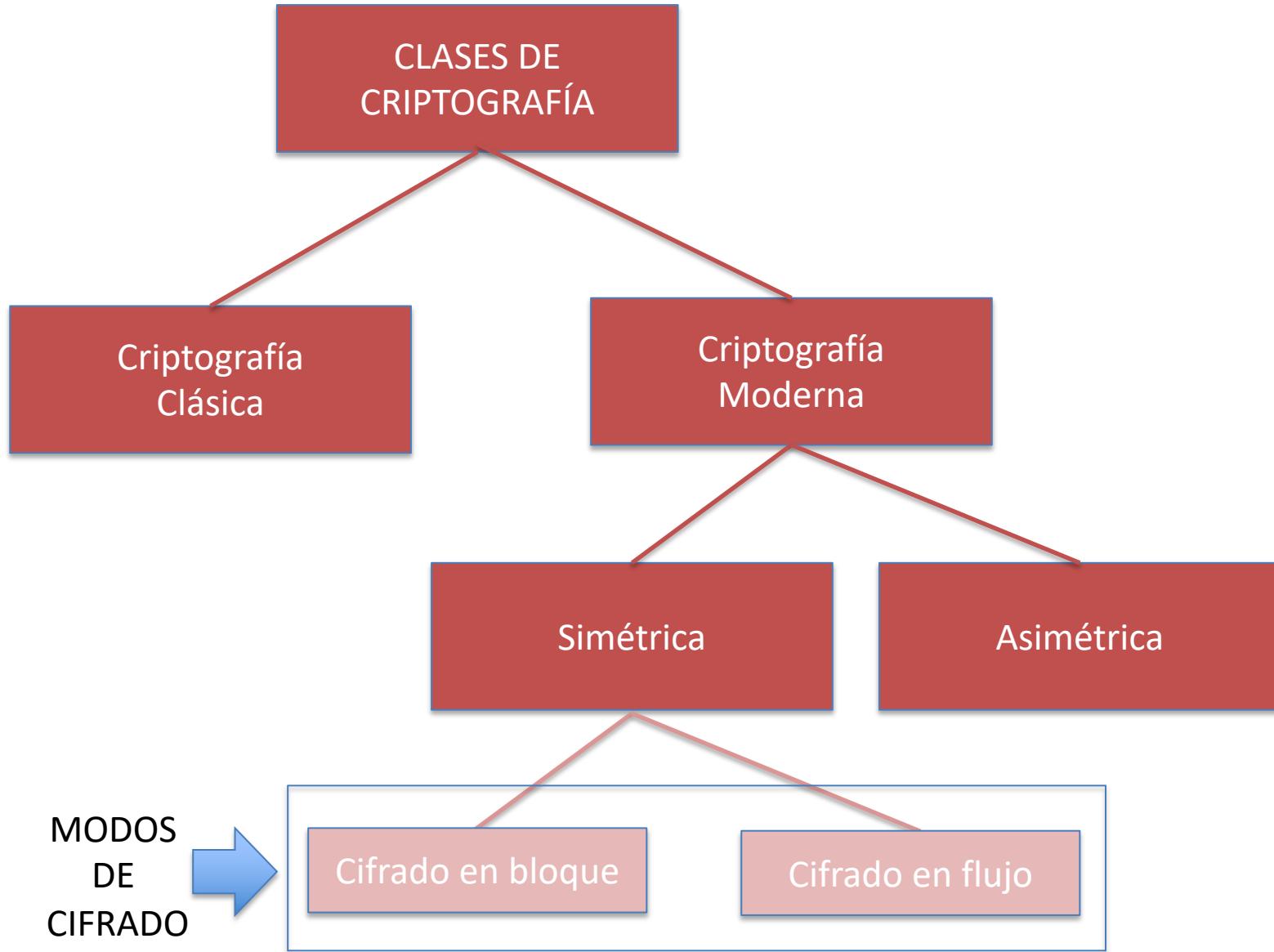
- En esta situación, el mismo algoritmo de descifrado D será usado por todos los receptores, pero cada uno necesitará la clave correspondiente de descifrado ($K_1^*, K_2^*, K_3^* \dots$)
- En resumen, para la comunicación específica entre *Alice* y *Bob*:



- Si los algoritmos E y D son públicos, entonces el secreto se encuentra en K_1 y K_1^*
- **Pero, ¿por qué son los algoritmos públicos? ...**



- Por lo tanto, en las nuevas condiciones anteriores,
es posible hacer públicos los algoritmos E y D
 - De hecho, se pueden evaluar públicamente para detectar posibles fallos
 - En caso de no tener fallos, entonces se pueden introducir en herramientas comerciales, etc.
 - Esto se formaliza en el **segundo principio de Kerckhoffs**:
 - *“The system must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience”*
- Por lo tanto, la seguridad del sistema dependerá finalmente de que Alice y Bob mantengan en secreto las claves secretas K y K^*
 - Los **algoritmos simétricos** son aquellos en los que K y K^* son la misma clave, y se denomina **clave de sesión**
 - En los **algoritmos asimétricos**, las claves K y K^* son distintas



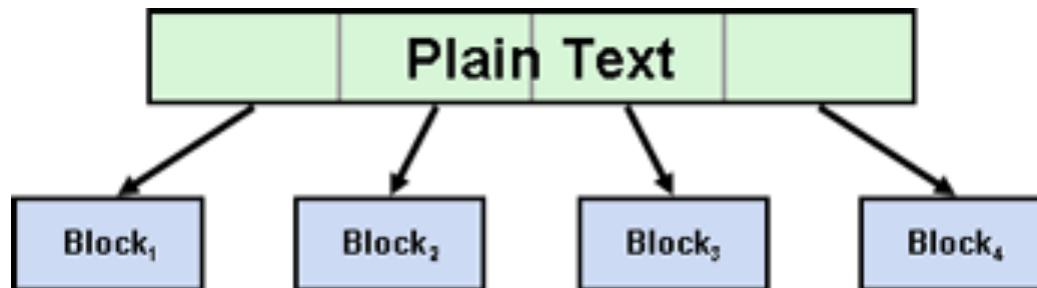
Modos de cifrado - cifrado en bloques

- A partir de la expresión

$$E(M) = C$$

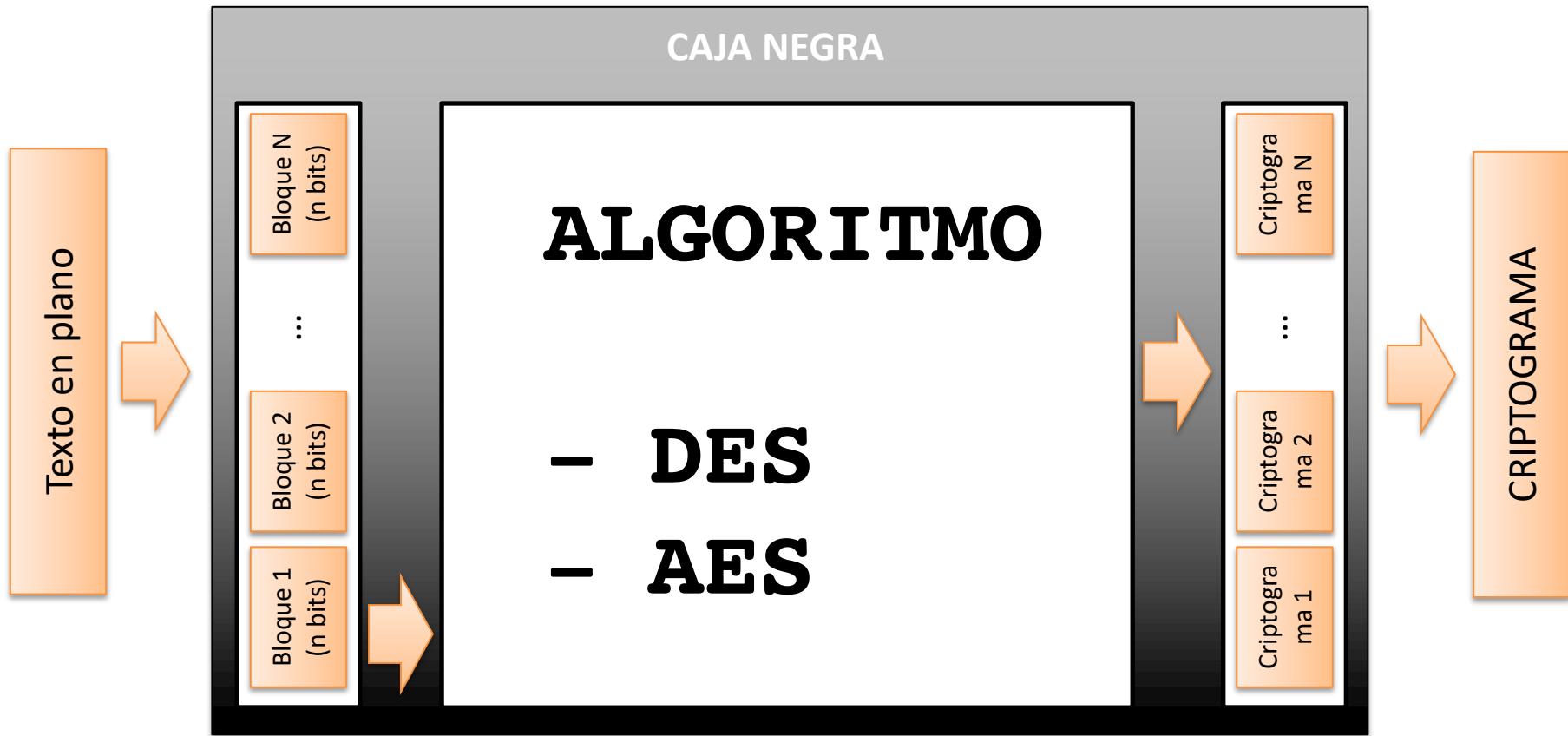
se podría pensar que el algoritmo de cifrado procesa todo el mensaje de una sola vez

- Sin embargo, por cuestiones de diseño, es raro que ocurra eso
- De hecho, son muchos los algoritmos que necesitan procesar el mensaje M en bloques de n bits, denominándose entonces **cifrados en bloque**



- La longitud específica n de los bloques viene determinada por el propio diseño interno del algoritmo

Si lo vemos como cajas negras

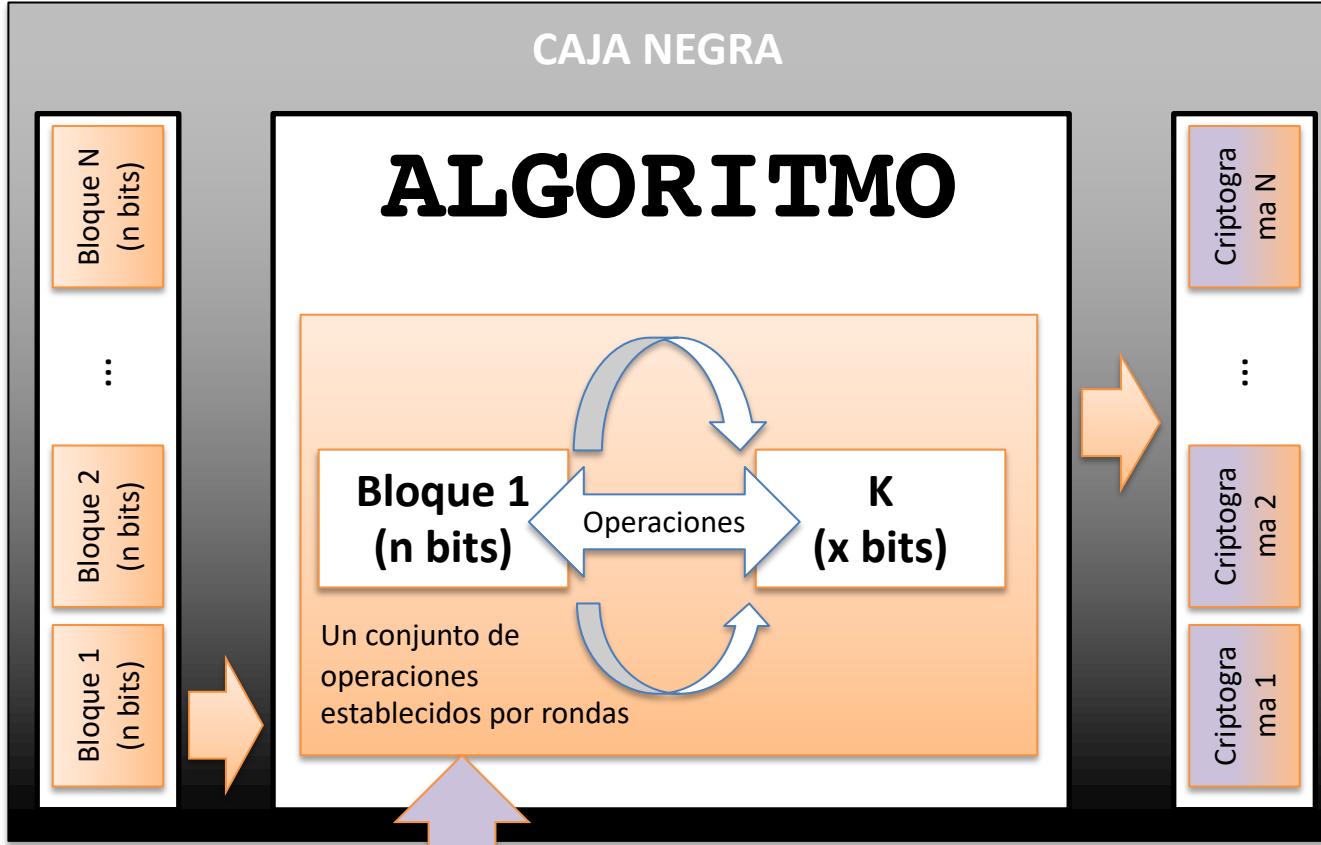


Pero, ¿cuándo se produce el cifrado?

CAJA NEGRA

ALGORITMO

Texto en plano



Se produce el cifrado cuando:

- el bloque de datos se entremezcla con el secreto (es decir, con los datos de la clave K), aplicando la operación **XOR**
- Cuanto más operaciones adicionales haya, más ruido se producirá en el criptograma final y más robusto será éste (frente ataques)

Otra forma de verlo (1)

- Por tanto:

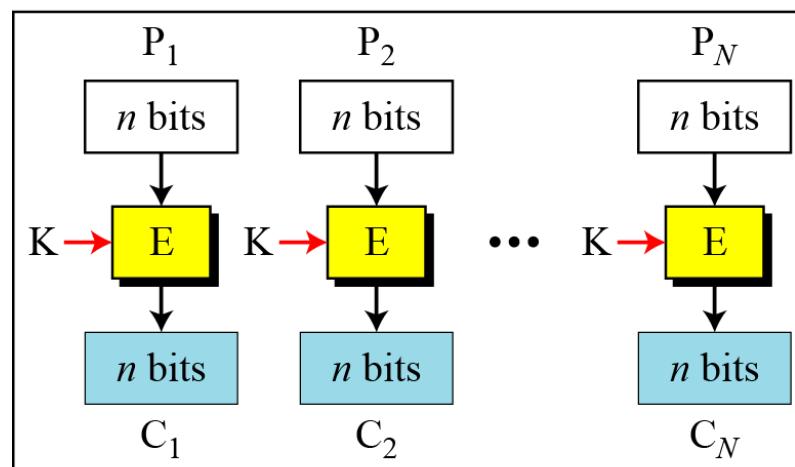
E: Encryption

P_i : Plaintext block i

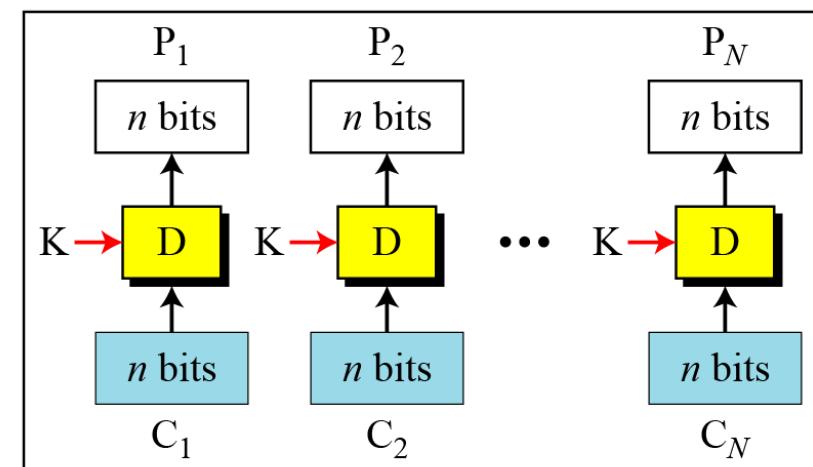
K: Secret key

D: Decryption

C_i : Ciphertext block i

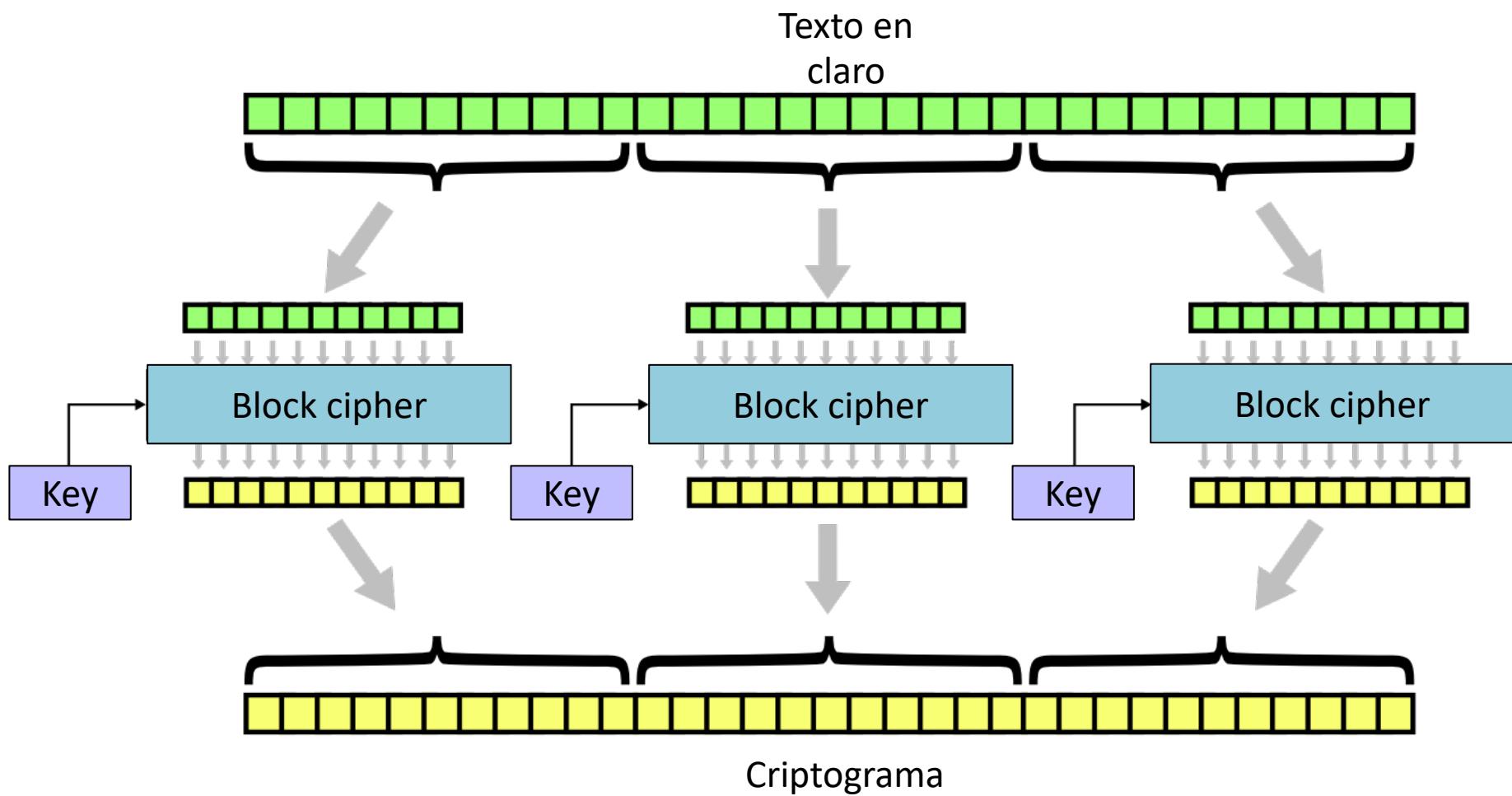


Encryption



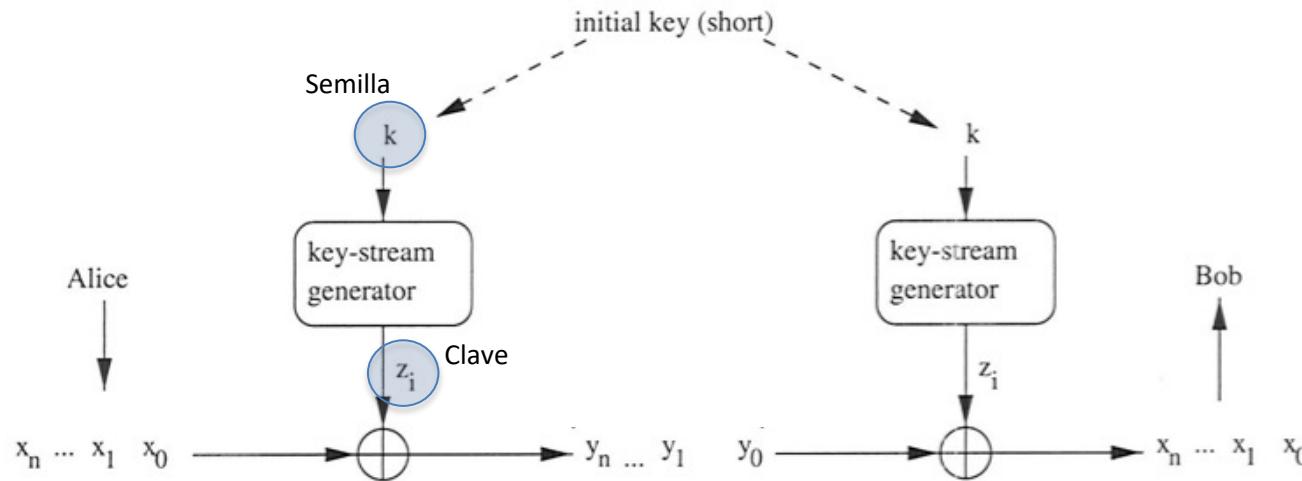
Decryption

Otra forma de verlo (2)

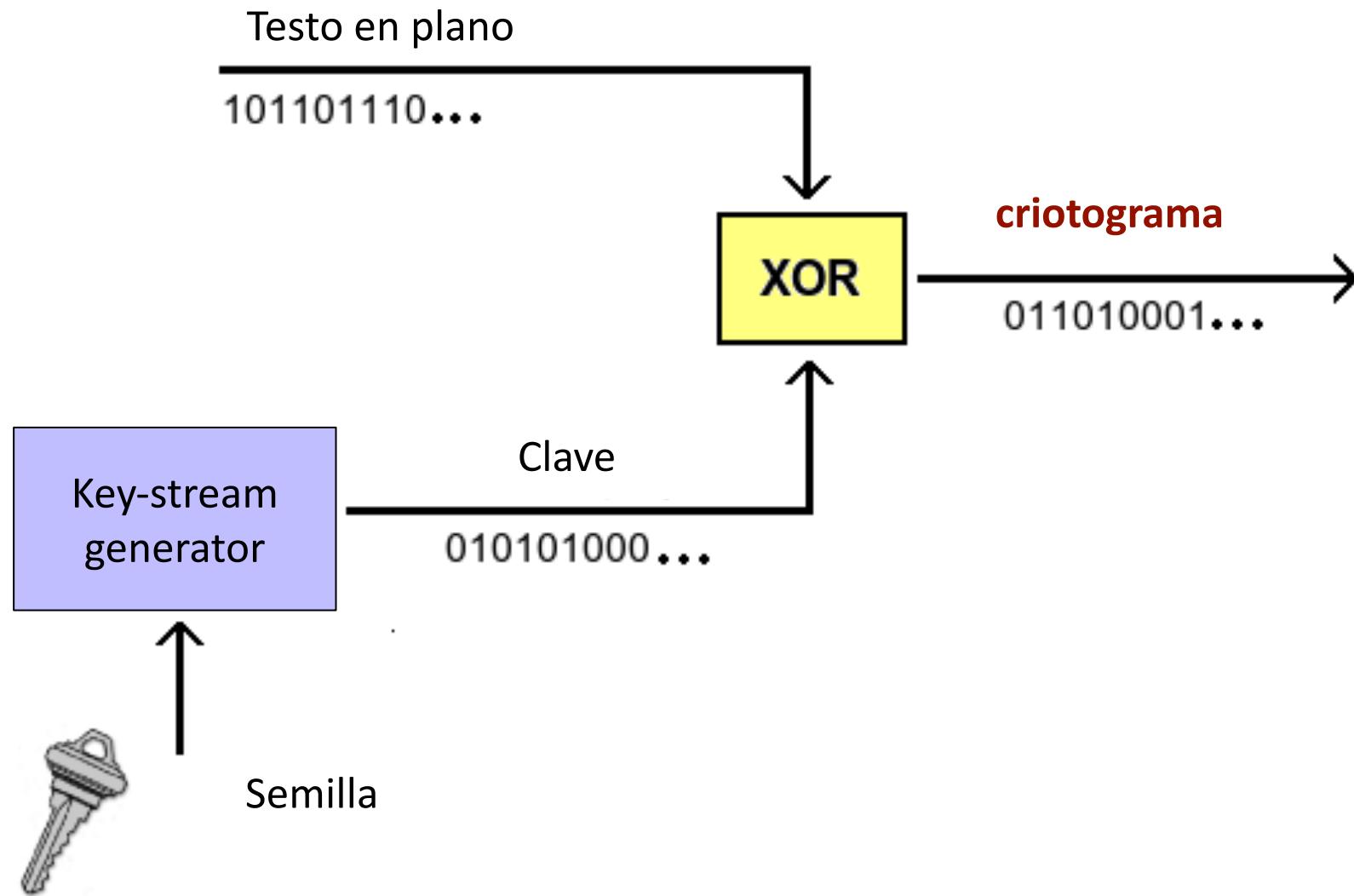


Modos de cifrado - cifrado en flujo

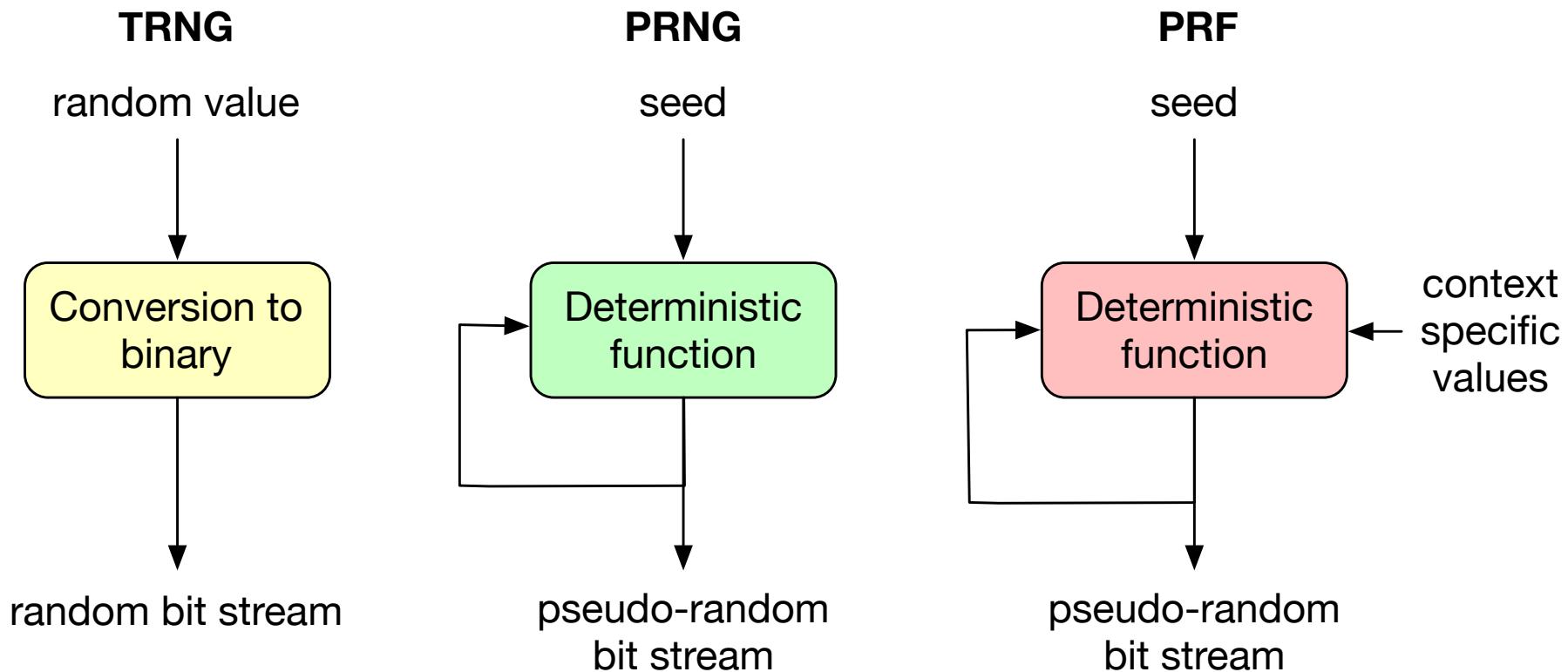
- Otros algoritmos, en lugar de procesar M particionándolo en bloques, necesitan procesarlo bit a bit, denominándose entonces **cifrados en flujo**
- Para ello, se opera en **XOR**:
 - cada bit del mensaje se realiza un XOR con el bit correspondiente del flujo de clave
 - El flujo de clave depende de la clave inicial **K (la semilla)**



Otra forma de verlo (1)



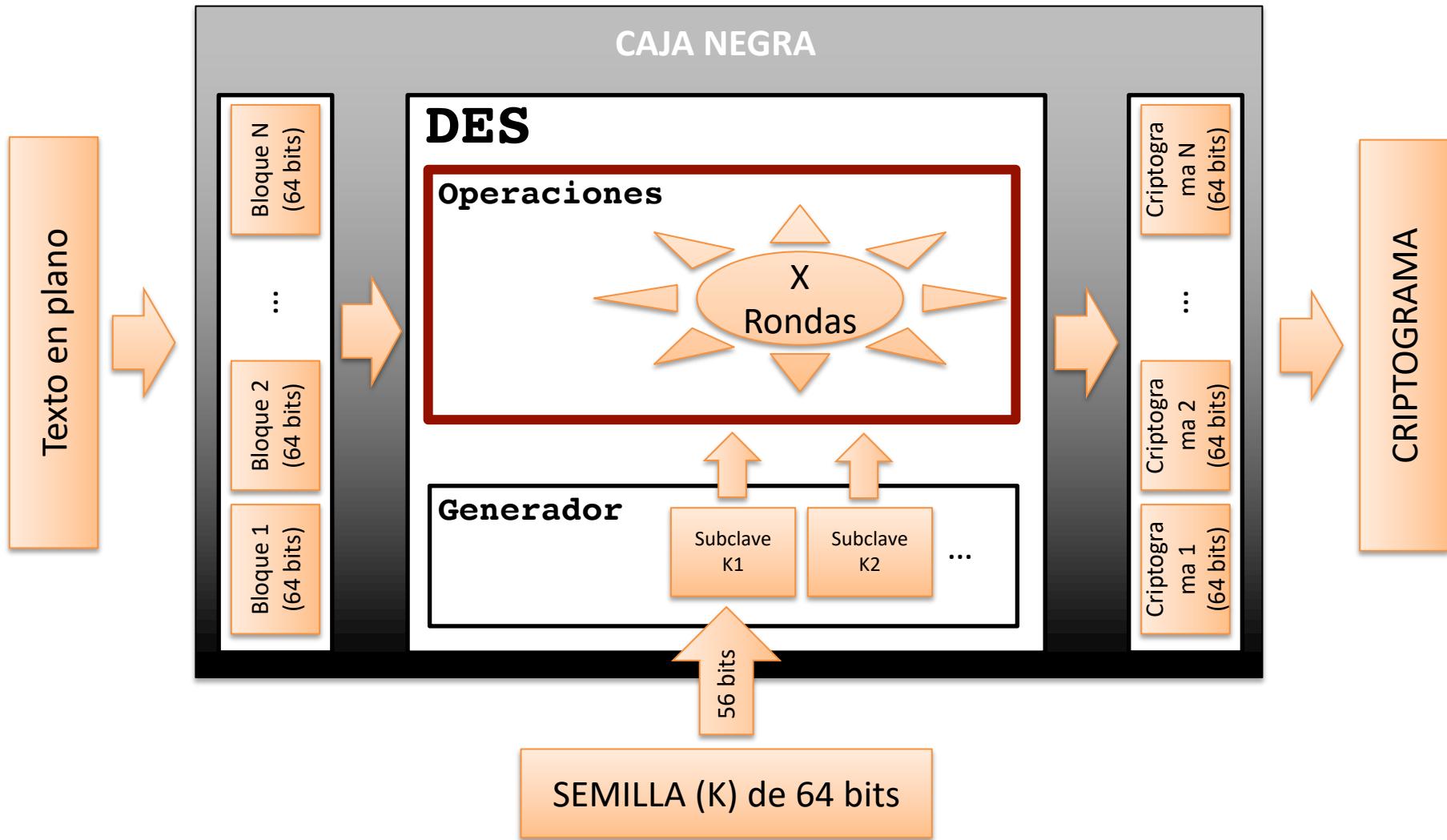
Tipos de generadores pseudo-randoms



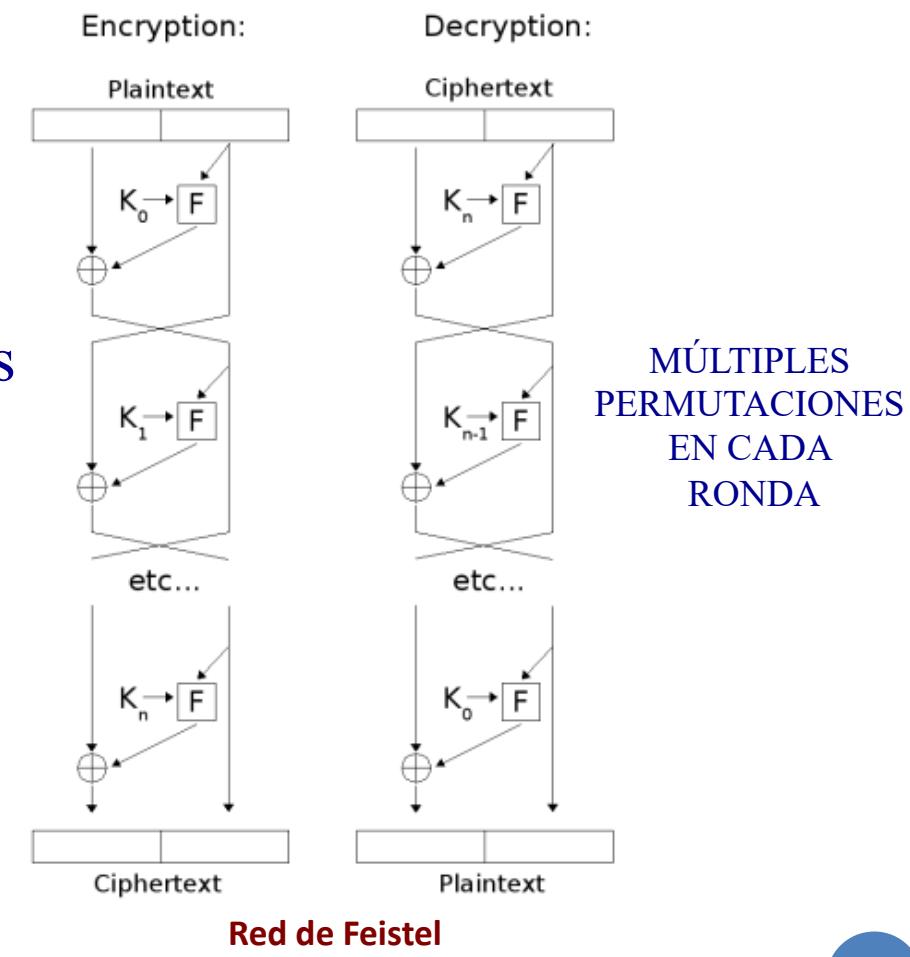
- Un **true random number generator (TRNG)**:
 - Toma como entrada un valor aleatorio
 - Su valor se puede extraer del entorno físico del ordenador (patrones de sincronización de las teclas, actividad eléctrica del disco, movimientos del ratón, valores del reloj del sistema, ...)
- Un **pseudorandom number generator (PRNG)**:
 - Toma como entrada un valor fijo (semilla) y produce una secuencia abierta utilizando un algoritmo determinista
 - La semilla suele ser generada por el TRNG
- Una **pseudorandom function (PRF)**:
 - se utiliza para producir una cadena pseudoaleatoria de bits de cierta longitud fija (por ejemplo, claves simétricas y nonces) + información de contexto

Algoritmo DES (Data Encryption Standard)

- *DES* es un algoritmo de cifrado simétrico que:
 - Usa bloques de **texto en claro de 64 bits**, y produce **bloques cifrados de igual tamaño**
 - Usa una **clave** que también es **de 64 bits** (8 octetos) de longitud
 - El último bit de cada octeto de la clave se usa como bit de paridad, por lo que la longitud efectiva de la clave (a efectos de seguridad) es de, en realidad, **56 bits**
- Diseñado por *IBM* para la competición del *National Bureau of Standards* (ahora *NIST*), en la que se solicitaban propuestas de algoritmos que pudiesen usarse como estándares para:
 - *cifrado de datos en transmisión*
 - *cifrado de datos en almacenamiento*por parte de el Gobierno americano, las empresas privadas y, en general, de cualquier tipo de usuario

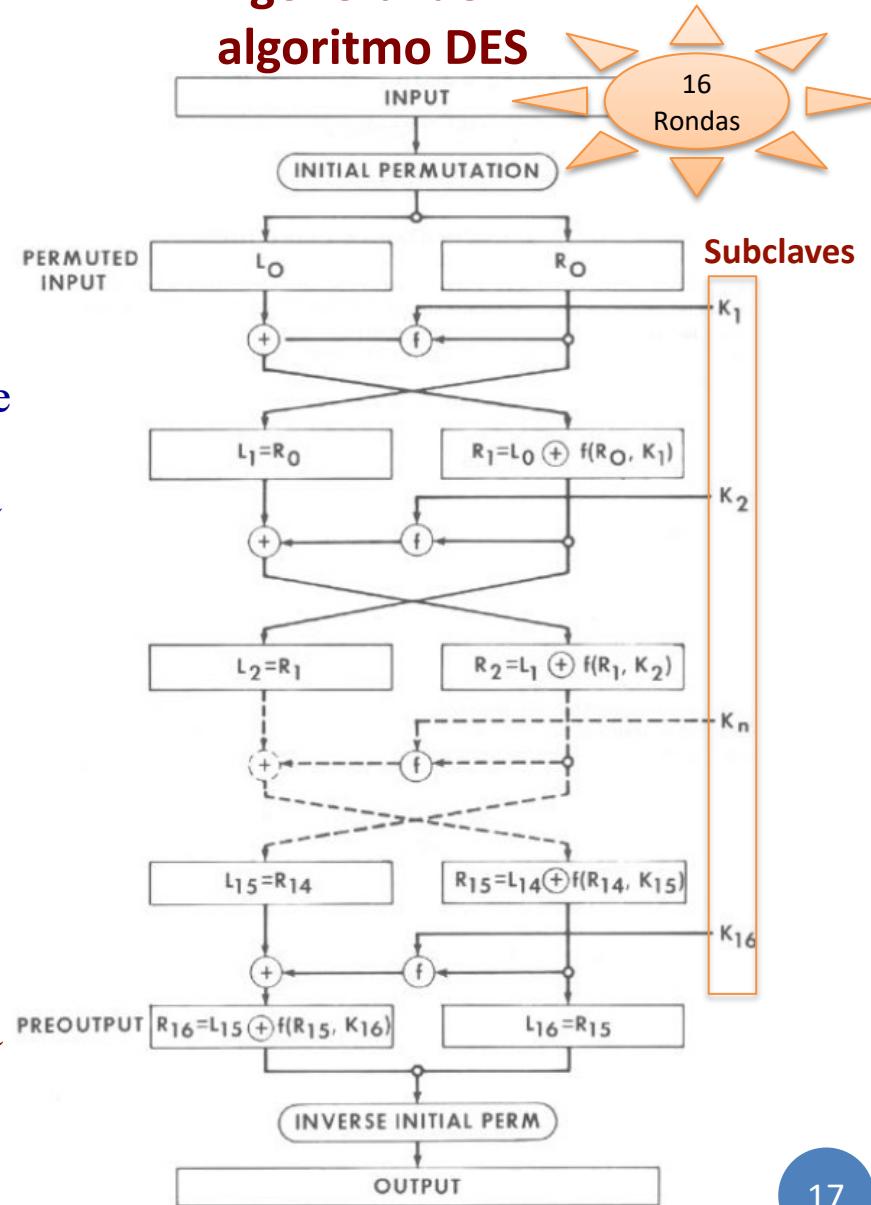


- Para el desarrollo del *DES*, *IBM* partió de *Lucifer*, un algoritmo propio desarrollado con anterioridad y usado principalmente en entornos bancarios
- Lucifer se basaba en el uso de una técnica denominada **red de Feistel**, y usaba una longitud de clave de 128 bits



- La idea de la red de Feistel queda reflejada en el propio ***DES***, como muestra la figura
- El esquema corresponde a la operación de cifrado (**16 etapas/rondas**) que se realiza para cada uno de los bloques del texto en claro
 - donde cada ronda aplica una subclave de 48 bits, generada por un generador interno y propio de DES y mediante una semilla de 56 bits
- ¿Qué extraemos de este esquema?
 1. Se aplica **16 rondas de operaciones (permutaciones y XOR ⊕)** para el cifrado
 2. Se usa **16 subclaves en cada etapa** – luego, ¡ las subclaves son el secreto !
 3. La robustez de DES se encuentra en la cantidad de permutaciones que se realizan dentro del algoritmo

Esquema general del algoritmo DES





64 bits

64 bits

Texto a Cifrar

Semilla

Clave Privada

64 bits

64 bits

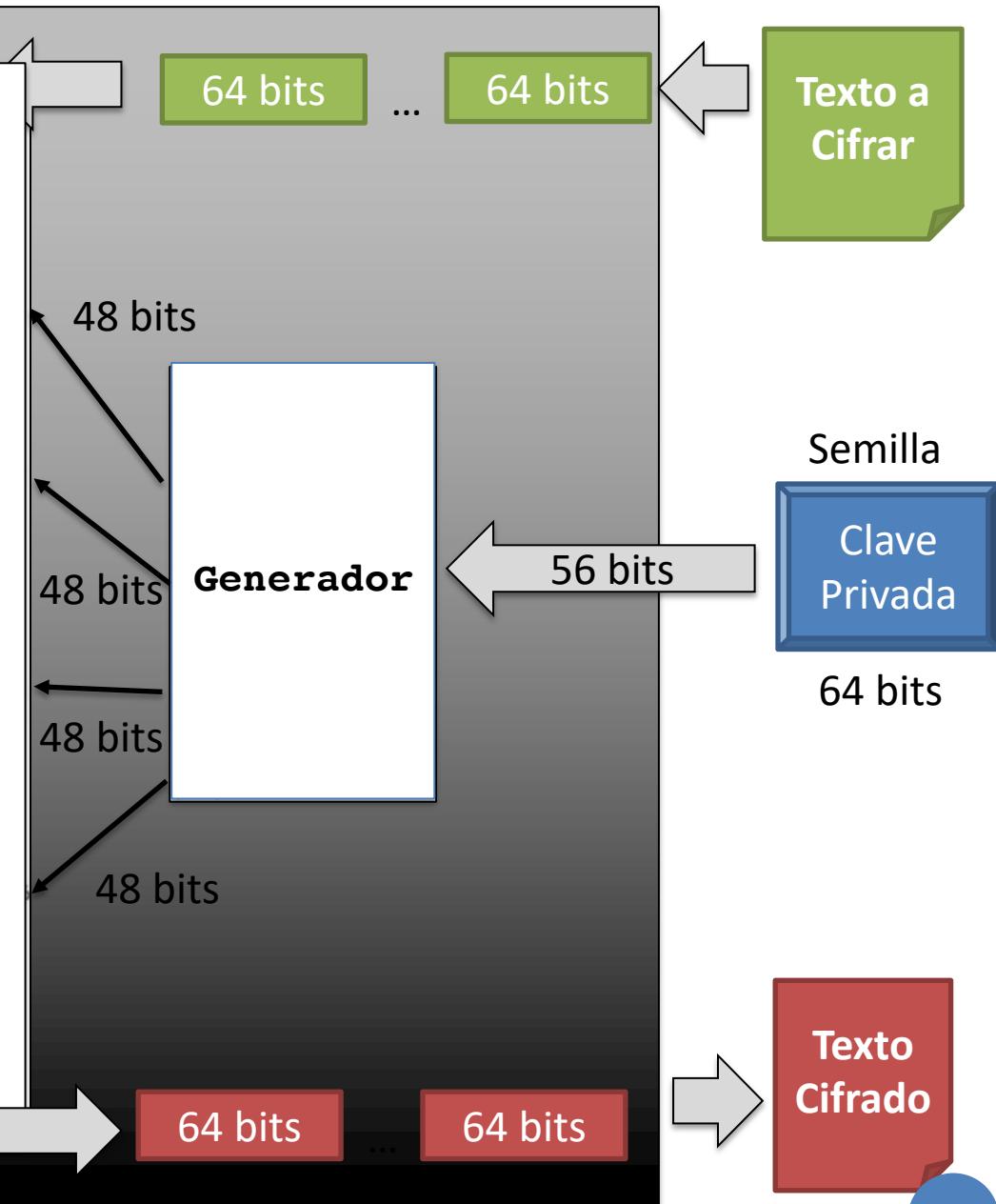
64 bits

Texto Cifrado

F

Se realizan una serie de operaciones internas:

- 1) Expandir los datos de entrada (siguiendo un proceso estandarizado) a 48 bits para que se pueda operar con las subclaves
- 2) CIFRAR con XOR (48 bits + 48 bits)
- 3) Comprimir el criptograma a 32 bits siguiendo un proceso estandarizado



INCISO - Efecto avalancha

- Una característica eficiente de *DES*, y deseable en cualquier algoritmo de cifrado, es el **efecto avalancha**
 - Es decir, un cambio pequeño en el texto en claro, o en la clave, produce un cambio significativo en el texto cifrado
 - Si, por el contrario, el cambio fuera pequeño, el cripto-analista tendría mucha ventaja, porque se reduciría el número de posibles textos en claro o de posible claves
 - En otras palabras: impide reducir el espacio de búsqueda de claves para un ataque de fuerza bruta
- Ejemplo de efecto avalancha en *DES*:

Plaintext: 0000000000000000

Key: 22234512987ABB23

Ciphertext: 4789FD476E82A5F1

Plaintext: 0000000000000001

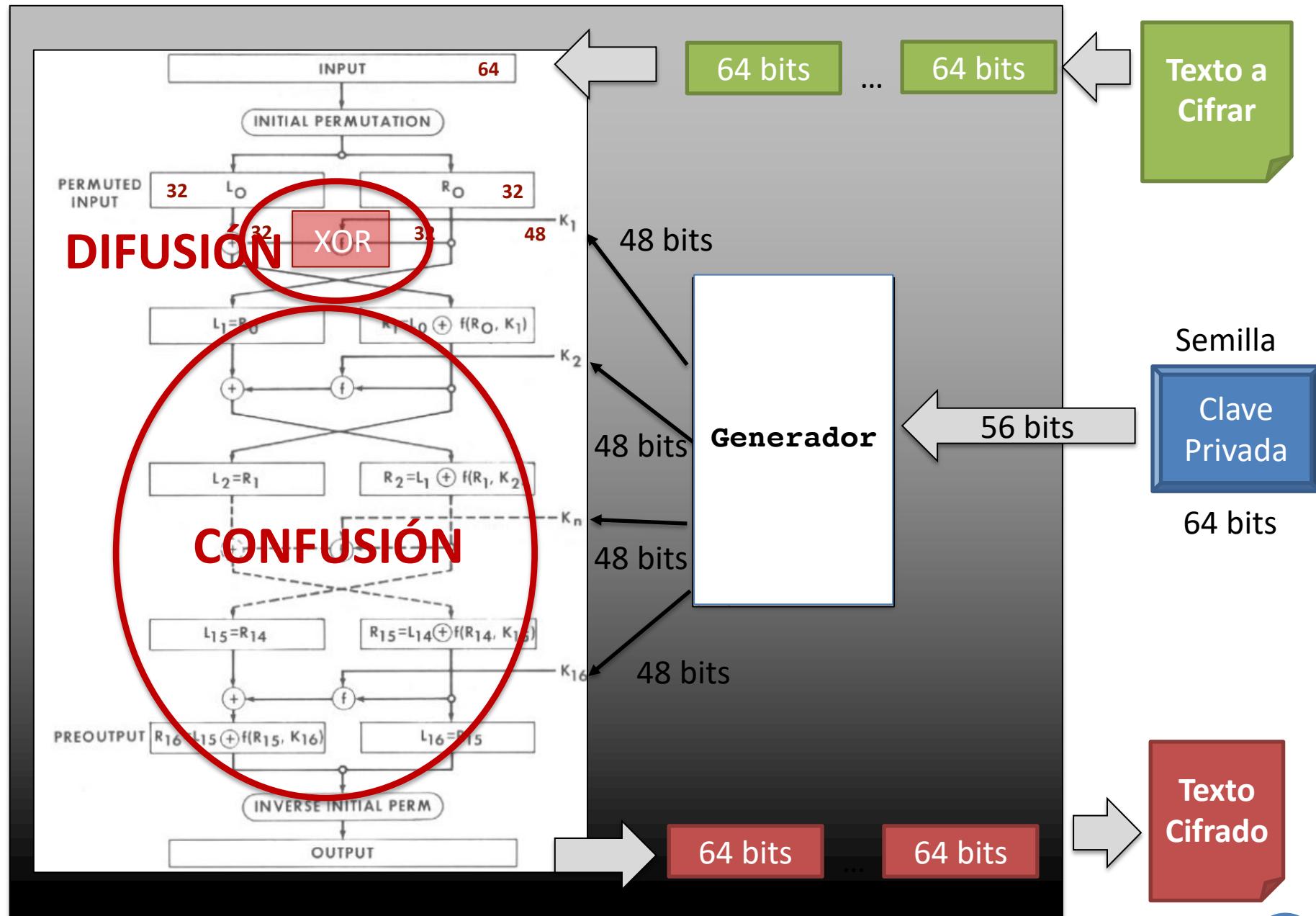
Key: 22234512987ABB23

Ciphertext: 0A4ED5C15A63FEA3

INCISO - Efecto avalancha

- El concepto de efecto avalancha se deriva de las tesis de Claude Shannon, el padre de la **Teoría de la Información** en su artículo:
 - *Communication Theory of Secrecy Systems*, 1949
- En ese artículo definía, entre otros conceptos, las propiedades de **difusión** y **confusión** para evitar (o dificultar) los ataques basados en análisis estadísticos
 - **difusión**: cada carácter del texto cifrado ha de depender de diferentes partes de la clave → ej: $M \text{ XOR } K$
 - **confusión**: la relación entre el texto cifrado y la clave ha de ser tan complicada como sea posible → ej: *PERMUTACIONES*
- Nota: es un criterio que se aplica a cualquier algoritmo de cifrado (DES, 3DES, IDEA, Camellia, etc.)





Seguimos con el algoritmo DES ...

- Hasta el momento no se conoce ningún ataque al algoritmo *DES* en sí mismo, y que haya sido completamente efectivo
- Por otro lado, un **ataque exhaustivo a la clave** (*brute-force attack*) podría parecer impracticable si suponemos, por ejemplo, una operación de descifrado por microsegundo
 - con longitud de clave de 56 bits, existen 2^{56} posibles claves (= 7.2×10^{16})



Algoritmo DES

- Sin embargo, se puede suponer que el criptoanalista va a disponer de capacidad de descifrado con microprocesadores en paralelo, y, por lo tanto, la situación cambia drásticamente:

Key size (bits)	Number of alternative keys	Time required at 1 decryption/ μs	Time required at 10^6 decryptions/ μs
32	$2^{32} = 4.3 \times 10^9$	$2^{31} \mu s = 35.8$ minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55} \mu s = 1142$ years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127} \mu s = 5.4 \times 10^{24}$ years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167} \mu s = 5.9 \times 10^{36}$ years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	$2 \times 10^{26} \mu s = 6.4 \times 10^{12}$ years	6.4×10^6 years

- Por lo anterior, se deduce que la longitud de clave del *DES* resulta demasiado corta si el criptoanalista dispone del hardware adecuado

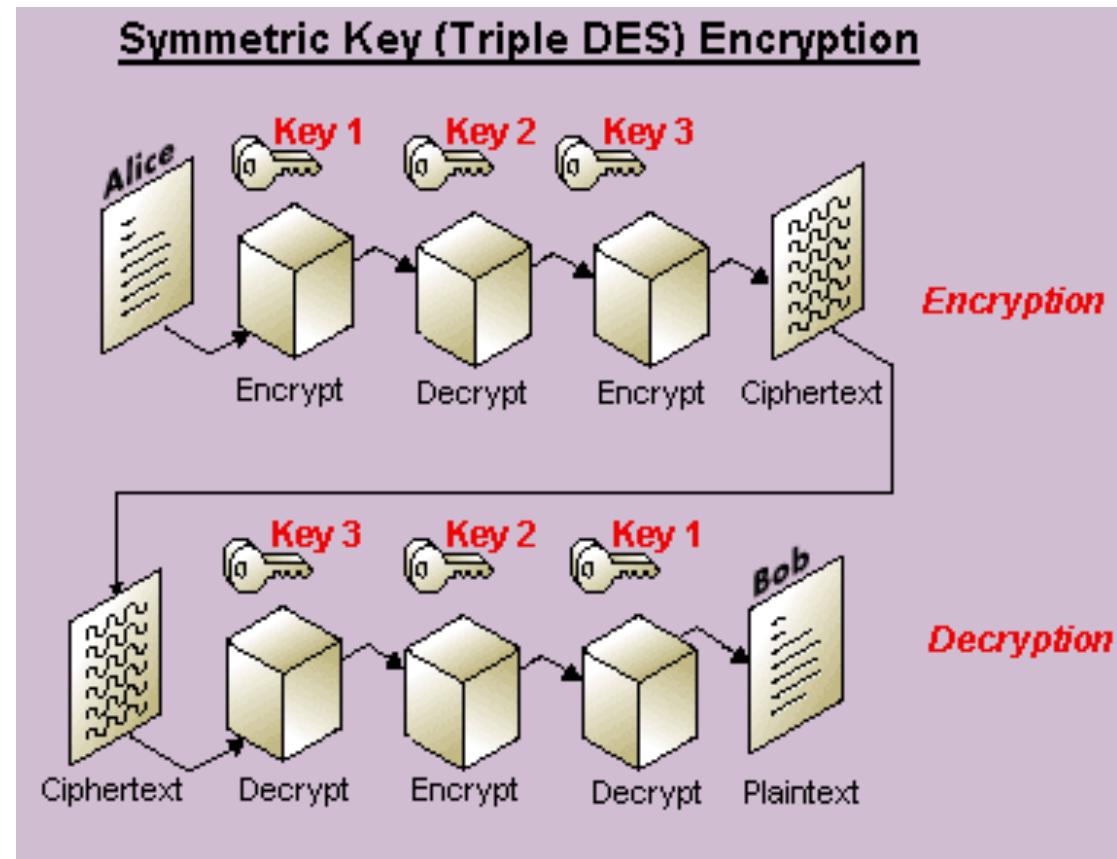


DES Key Search Machine

6 estaciones SUN-2, 27 placas de circuitos, 1800 chips customizados,
24 unidades de búsqueda por cada chip

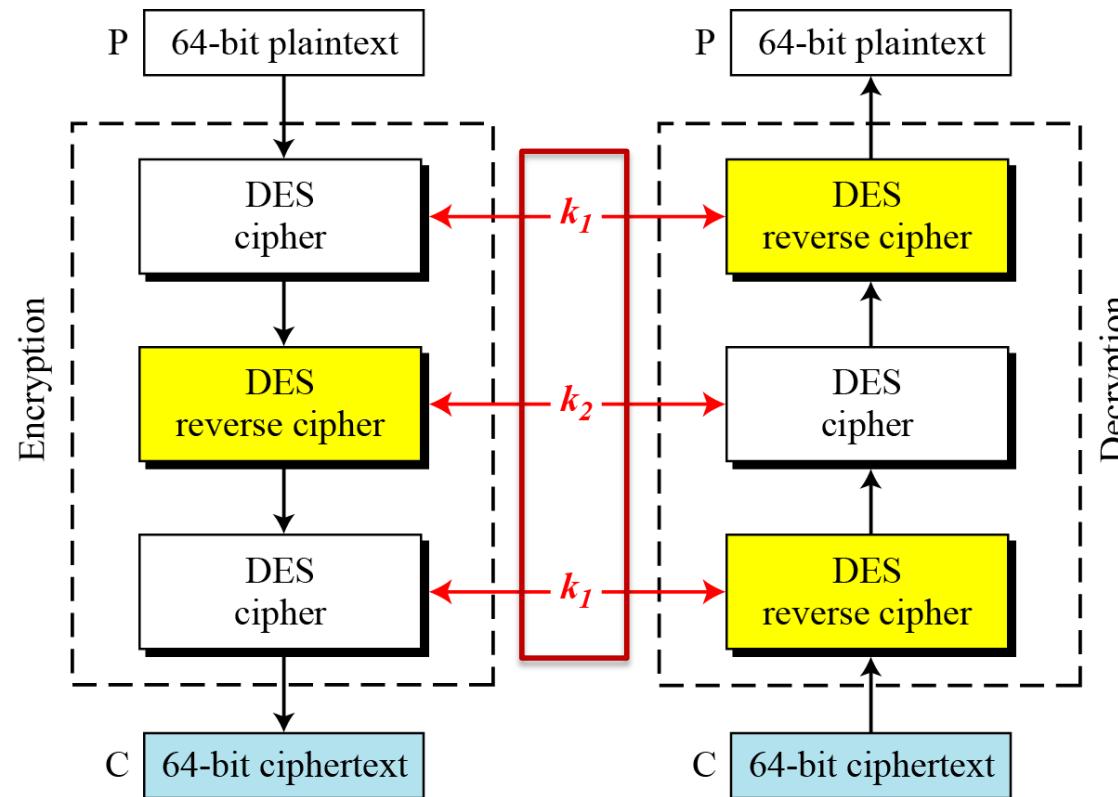
Algoritmo Triple-DES

- Para aprovechar las ventajas del *DES*, y a la vez contrarrestar su escasa longitud de clave, los organismos de estandarización han adoptado el criptosistema ***Triple-DES*** (ó ***3DES***)
- Consiste en usar una secuencia de tres operaciones *DES*, con **3 claves distintas** (168 bits)



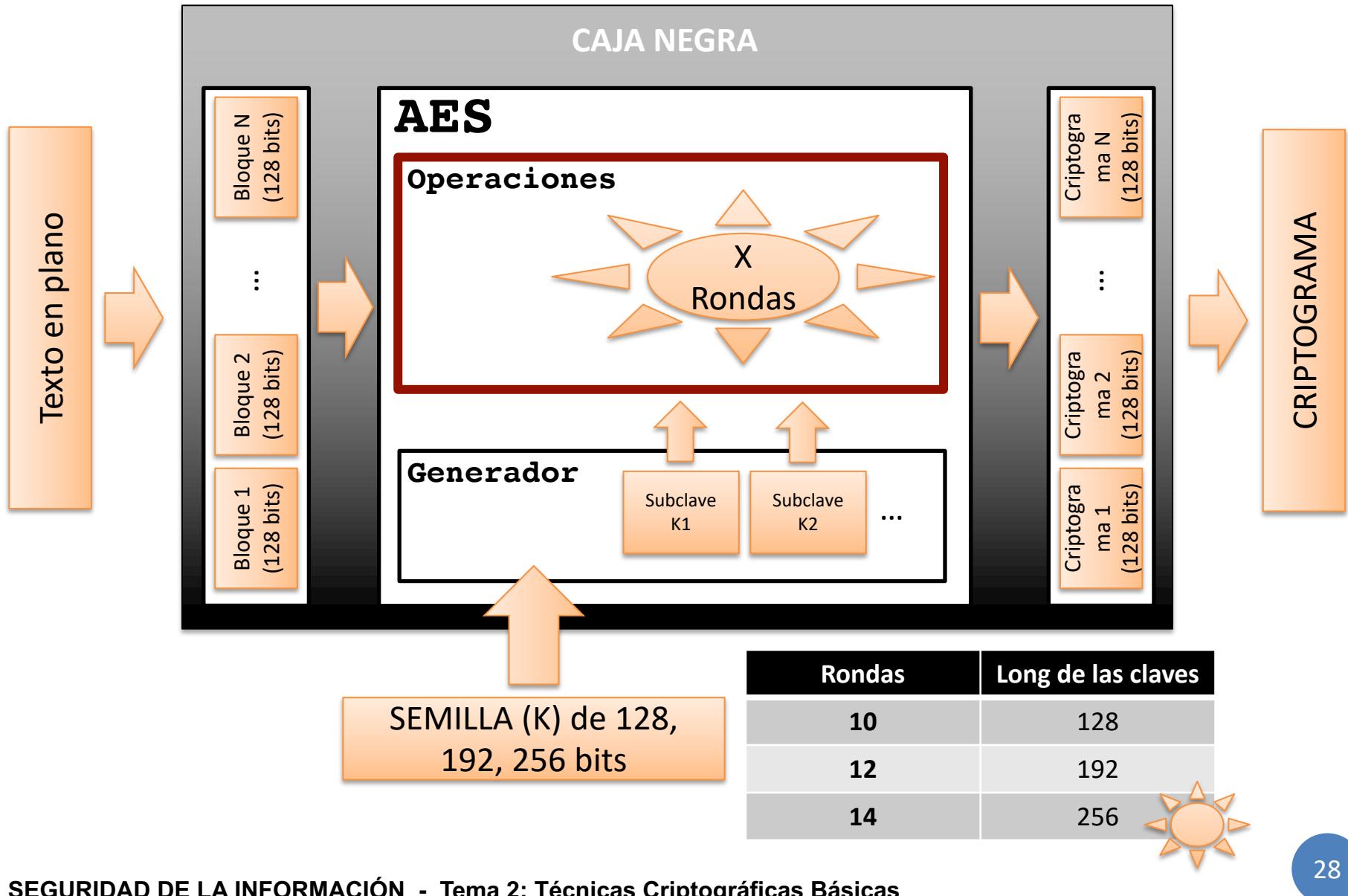
Algoritmo Triple-DES (variante de 2 keys)

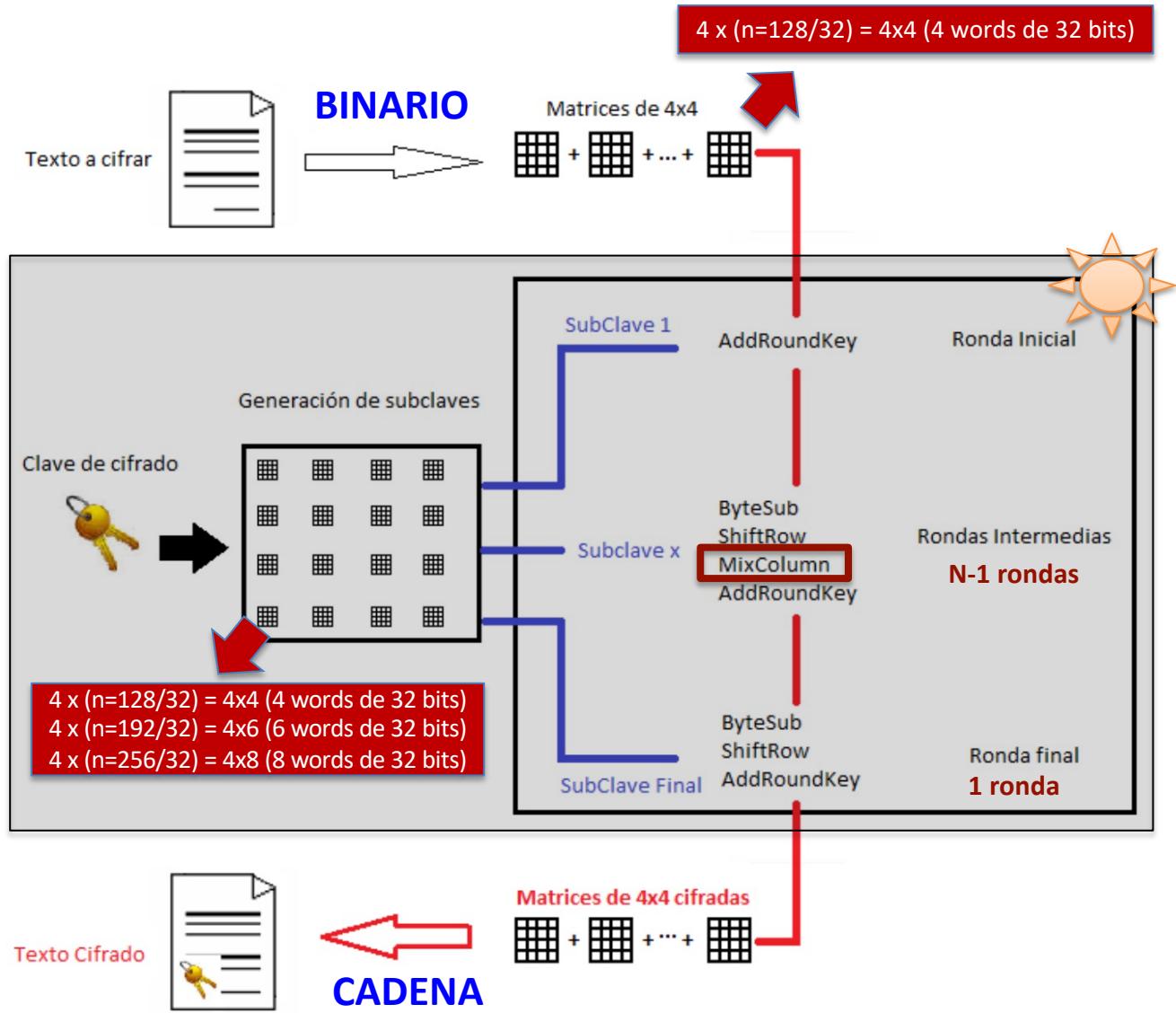
- Existe una variante en la que se utilizan sólo 2 claves, pero este esquema ha sido objeto de ataques, de forma que la **K1 = K3 (112 bits)**



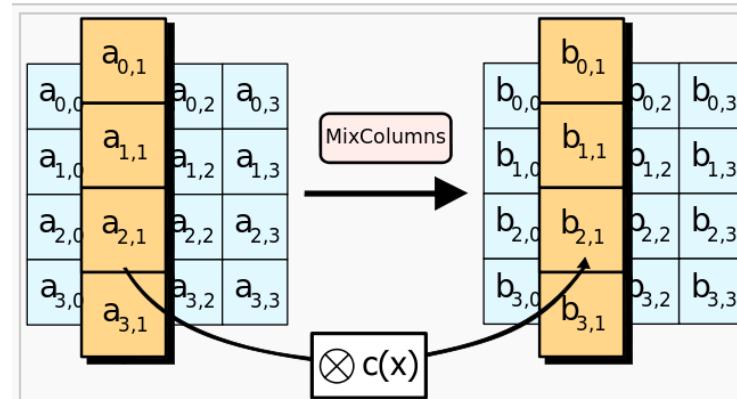
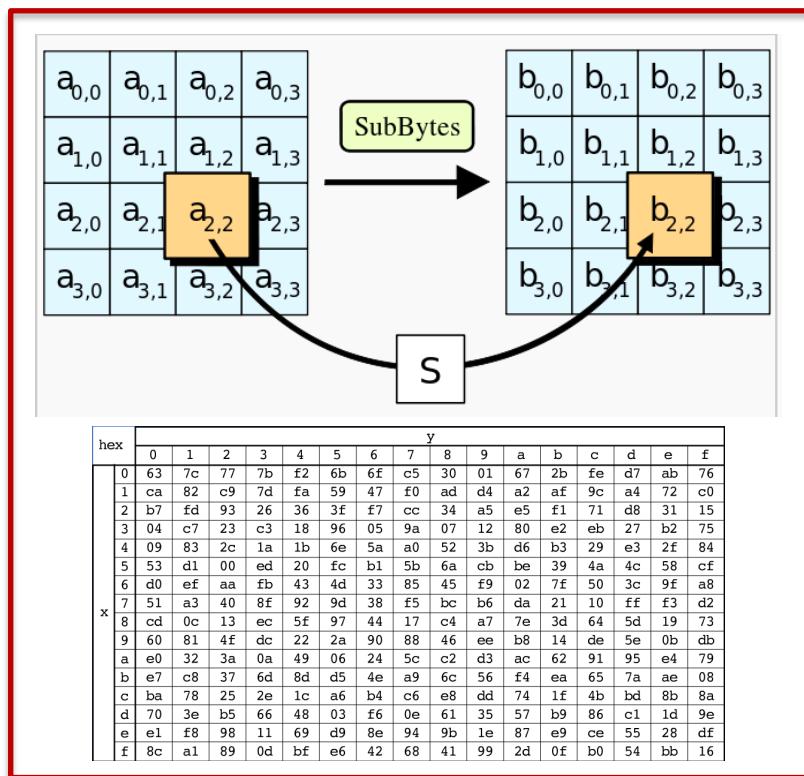
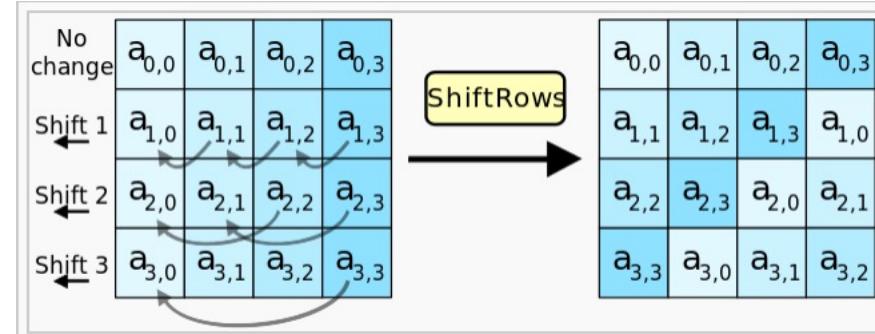
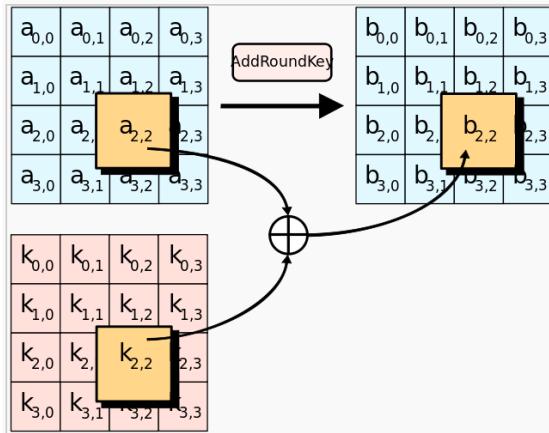
Algoritmo AES (Advanced Encryption Standard)

- AES fue publicado por NIST en 2001 como estándar de **cifrado simétrico en bloque** para sustituir a DES, especialmente en aplicaciones comerciales
 - su nombre original es *Rijndael*, por sus autores *Rijmen* y *Daemen*
 - utiliza **una clave de 128, 192 o 256 bits**
 - la longitud n de cada bloque de datos en los que se subdivide M es **128 bits**
- AES no utiliza una red de Feistel, sino una
 - **red de sustitución-permutación-operaciones aritméticas**
 - Cada etapa de AES se compone de cuatro funciones distintas:
 - **sustitución de byte**
 - **permutación**
 - **operaciones aritméticas en campo finito**
 - **XOR**

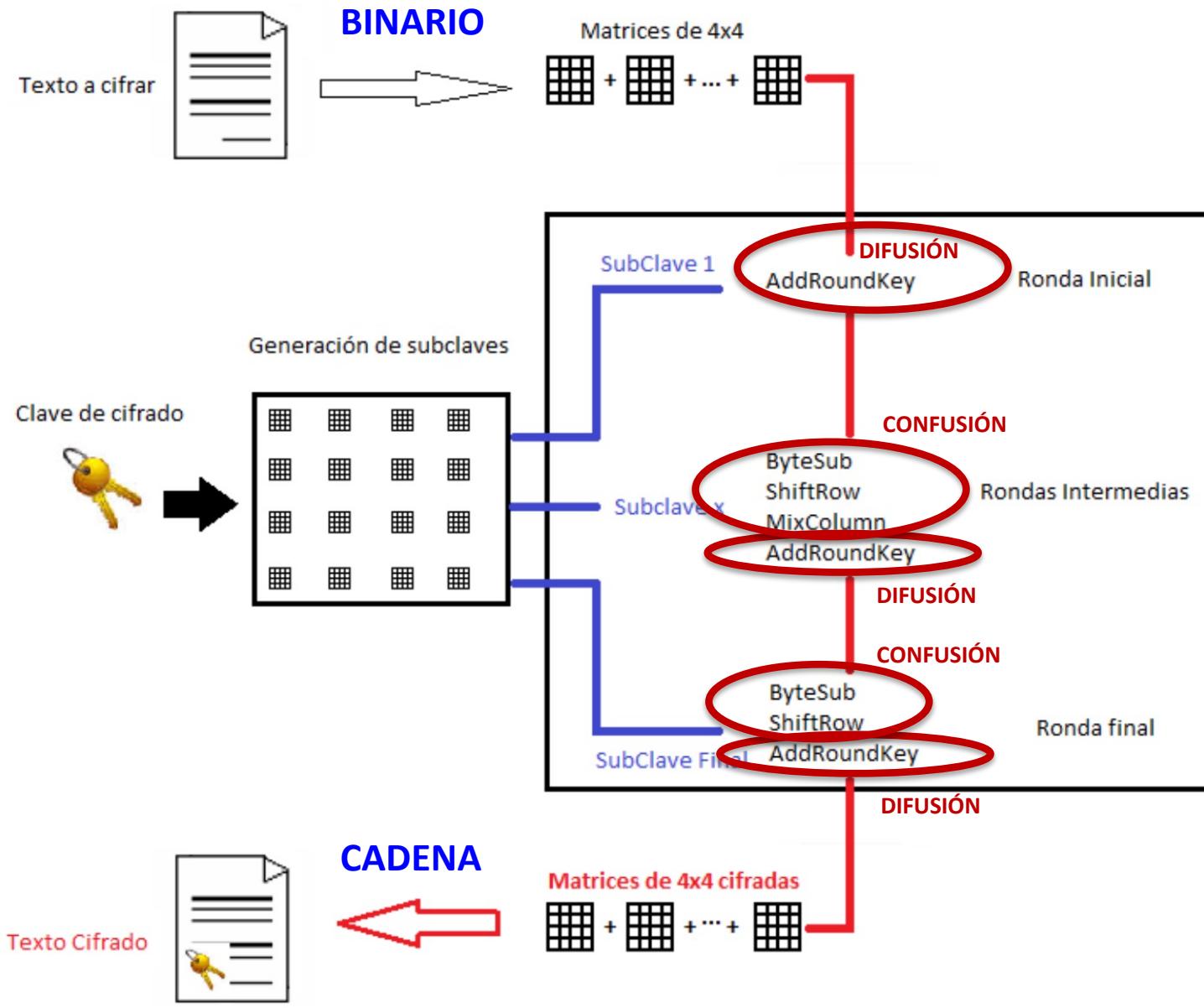




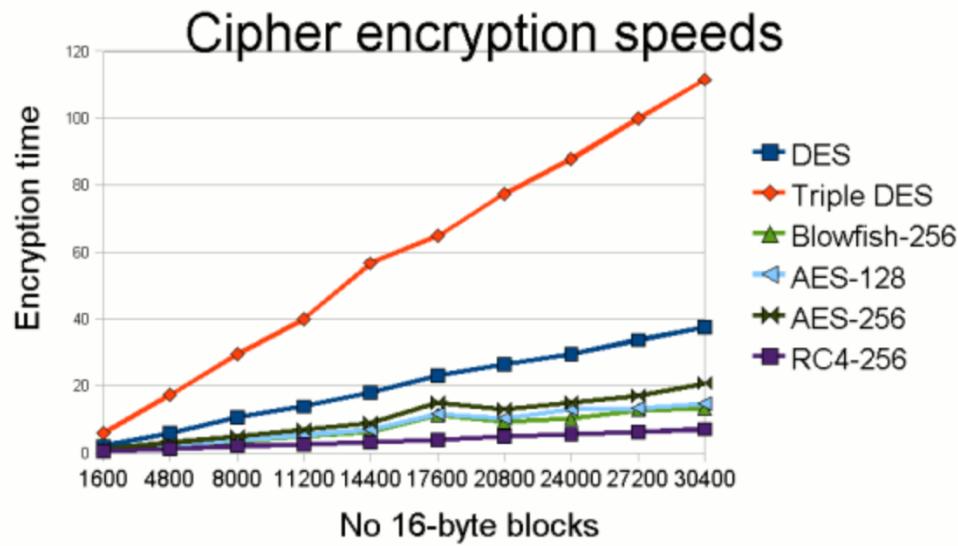
- La figura muestra el proceso de cifrado en AES para **128 bits** de clave (**10 etapas**)
- Otras posibilidades:
 - **192 bits** (**12 etapas**)
 - **256 bits** (**14 etapas**)



MixColumn multiplica la matriz que se está computando con matrices preestablecidas



- Rendimiento de AES comparado con otros algoritmos simétricos



Date	Minimum of Strength	Symmetric Algorithms
2010 (Legacy)	80	2TDEA*
2011 - 2030	112	3TDEA
> 2030	128	AES-128
>> 2030	192	AES-192
>>> 2030	256	AES-256

- Existen algunas recomendaciones generales:
 - En general, la **longitud mínima de clave** para un cifrado simétrico en bloque debería ser **128 bits**
 - **El tamaño mínimo de los bloques de datos** dependerá de la aplicación específica en la que se use el algoritmo, pero en la mayoría de ocasiones el mínimo **debería ser 128 bits**
 - La **cantidad máxima de información a cifrar con una misma clave** debería limitarse a $2^{n/2}$, donde n es el tamaño del bloque de datos
 - En cuanto a los algoritmos: ...

CLAVES:
 ≥ 128 bits

Bloques:
 ≥ 128 bits

Rekeying:
cada $2^{n/2}$

Otros algoritmos simétricos relevantes

– Blowfish

- Requiere una **clave de entre 32 y 448 bits** (pero sólo se recomienda su uso con **más de 80 bits**)
- Se utiliza en algunas configuraciones de IPSEC
- La longitud de cada **bloque de datos es de 64 bits**, demasiado pequeño para algunas aplicaciones
 - Por ese motivo, sólo se recomienda en uso heredado (*legacy use*)

– Kasumi

- Requiere una **clave de 128 bits** de longitud, y la longitud de los **bloque de datos es de 64 bits**
- Se usa en UMTS (con el nombre UIA1) y en GSM (con el nombre A5/3)
- Presenta una serie de problemas que no afectan a su uso práctico en esas aplicaciones
 - sin embargo, no se recomienda para aplicaciones futuras

– Camellia

- Requiere una **clave de 128 bits**, pero también da soporte a claves **de 192 y 256 bits**
 - Las versiones con 192 o 256 bits de clave son un 33% más lentas que la de 128 bits
- Se utiliza como **uno de los posibles cifrados en TLS**
- Por el momento, no se han encontrado ataques efectivos a este algoritmo

- Existen algunas recomendaciones generales:

- En general, la **longitud mínima de clave** para un cifrado simétrico en bloque debería ser **128 bits**
- **El tamaño mínimo de los bloques de datos** dependerá de la aplicación específica en la que se use el algoritmo, pero en la mayoría de ocasiones el mínimo **debería ser 128 bits**
- La **cantidad máxima de información a cifrar con una misma clave** debería limitarse a $2^{n/2}$, donde n es el tamaño del bloque de datos
- En cuanto a los algoritmos:

CLAVES:
>= 128 bits

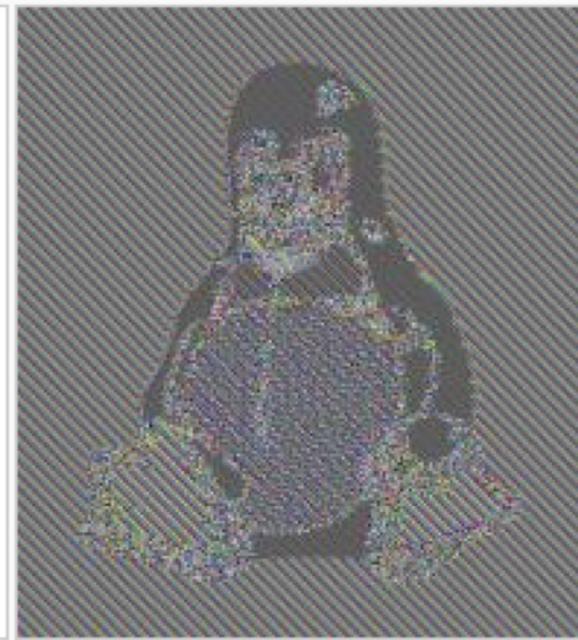
Bloques:
>= 128 bits

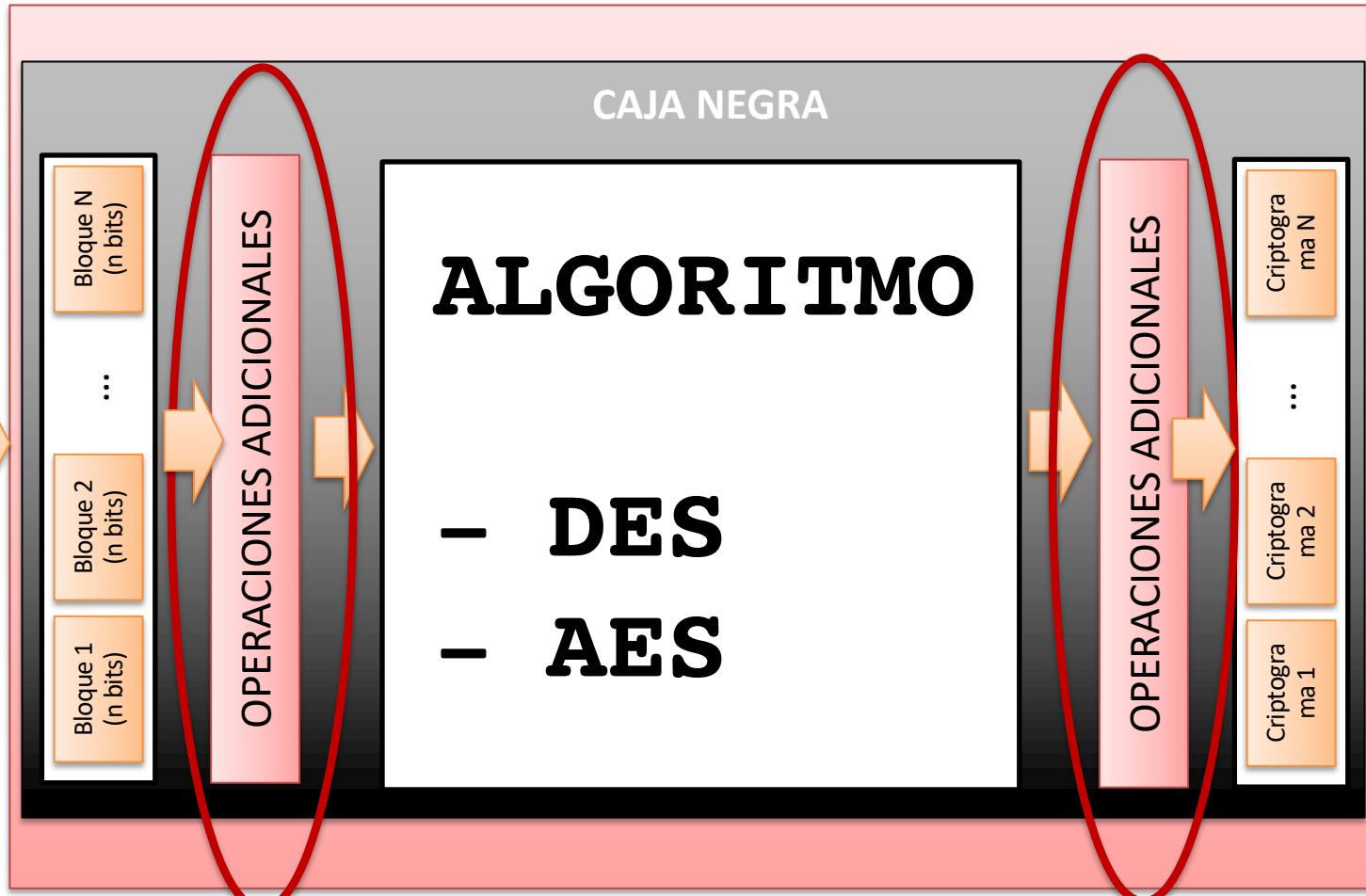
Rekeying:
cada $2^{n/2}$

Primitive	Recommendation	
	Legacy	Future
AES	✓	✓
Camellia	✓	✓
Three-Key-3DES	✓	✗
Two-Key-3DES	✓	✗
Kasumi	✓	✗
Blowfish \geq 80-bit keys	✓	✗
DES	✗	✗

Modos de operación para algoritmos simétricos

- Un modo de operación es una técnica para mejorar el efecto final de un algoritmo criptográfico
 - También se usa para adaptar el algoritmo a un tipo de aplicación concreta
- En ningún caso supone la modificación del algoritmo de cifrado en sí, sino de la forma en que se opera con los bloques de datos

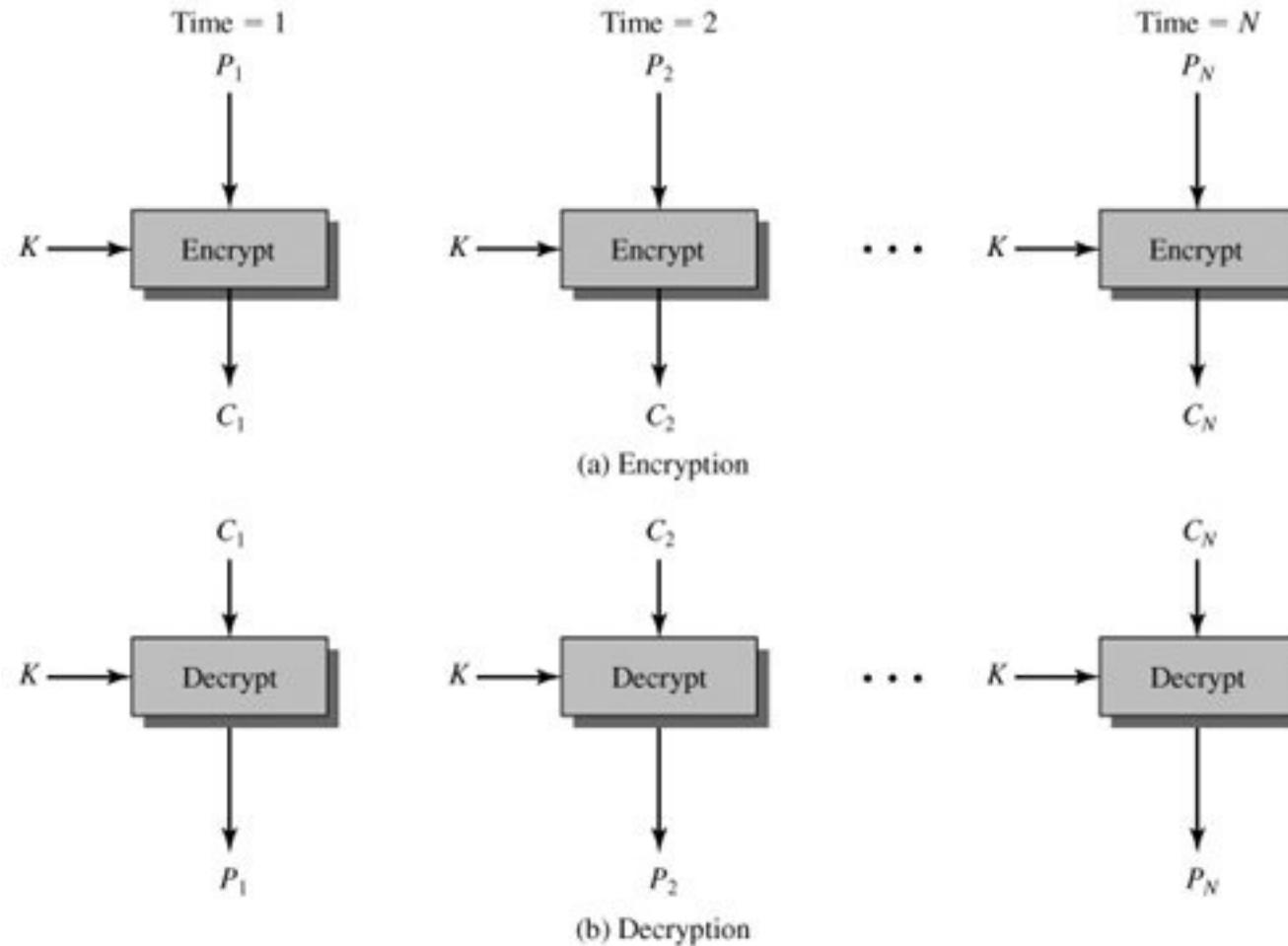


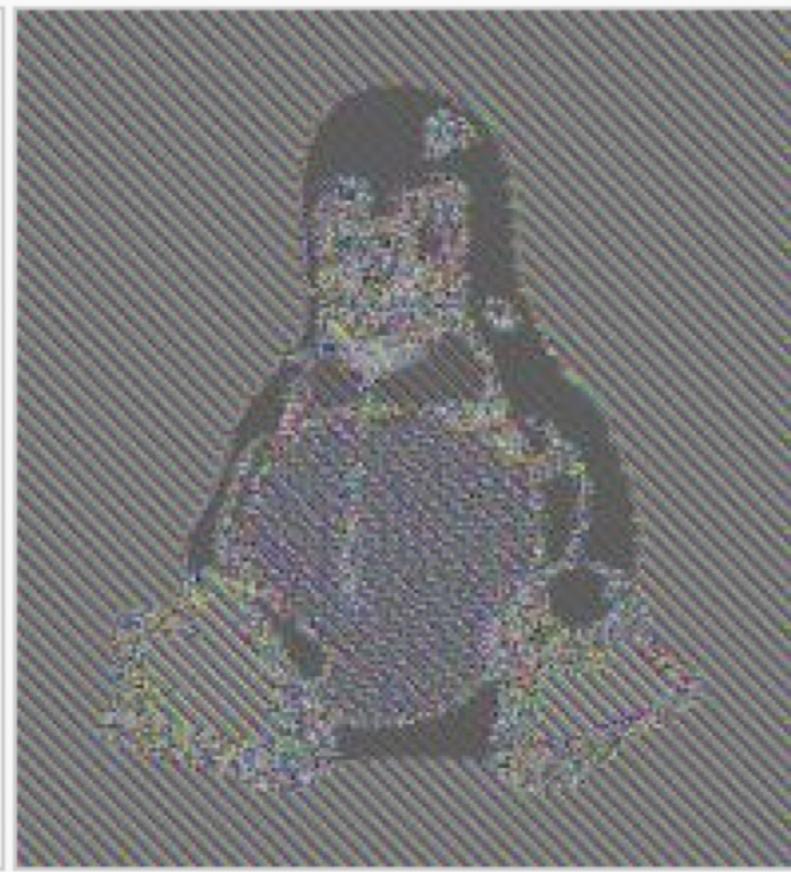


Modos de operación para algoritmos simétricos

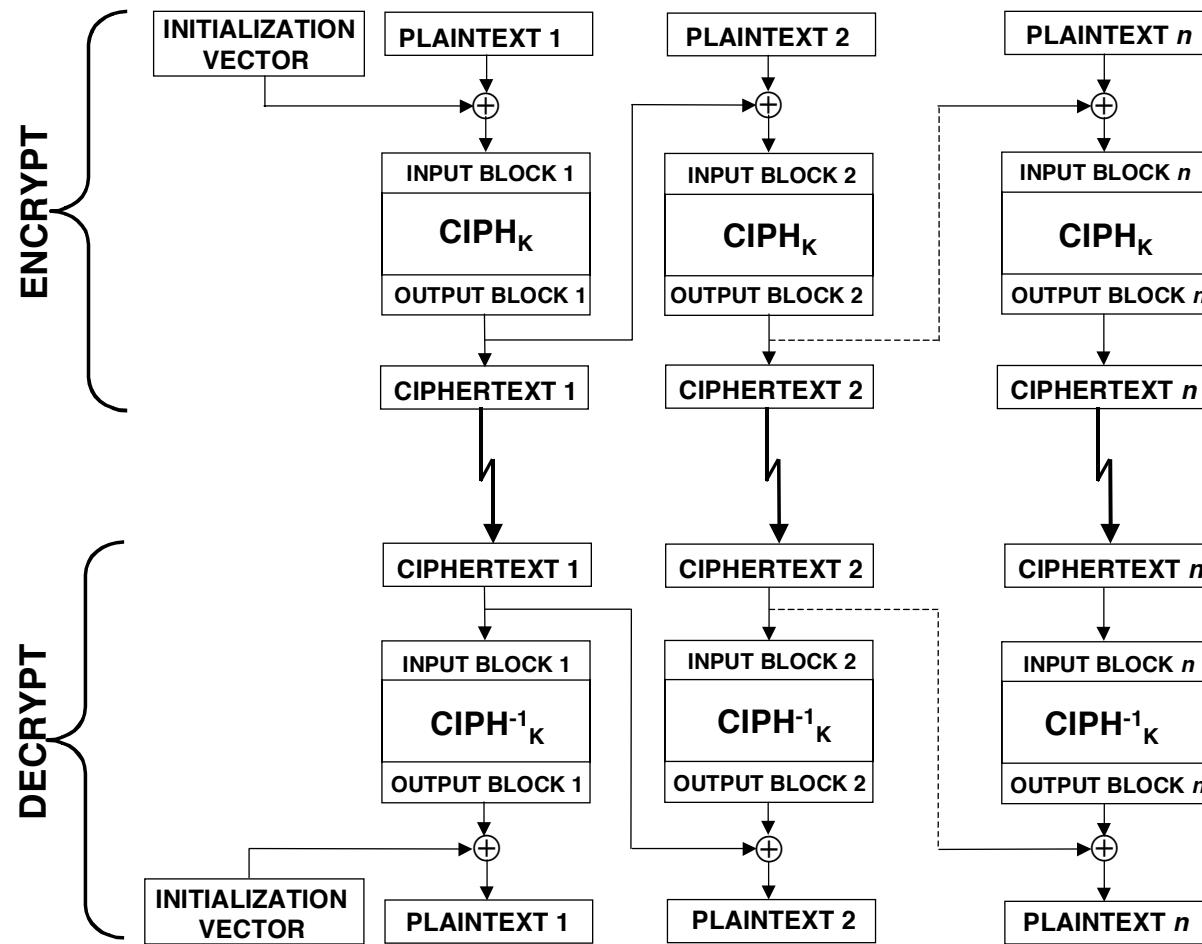
- Hay diferentes modos de operación, o lo que es lo mismo, diferentes formas de aplicar un mensaje M a un algoritmo de cifrado en bloque
 - cada una de ellas con sus ventajas y desventajas
- NIST ha definido cinco modos de operación, y cualquiera de ellos se puede utilizar con cualquier algoritmo simétrico (DES , $3DES$, AES , ...)
 - Electronic Codebook (ECB)
 - Cipher Block Chaining (CBC)
 - Cipher Feedback (CFB)
 - Output Feedback (OFB)
 - Counter (CTR)
 - **Galois-CTR (GCM)** – muy importante por su aplicación actual en la última versión de TLS

- Electronic Codebook (ECB)



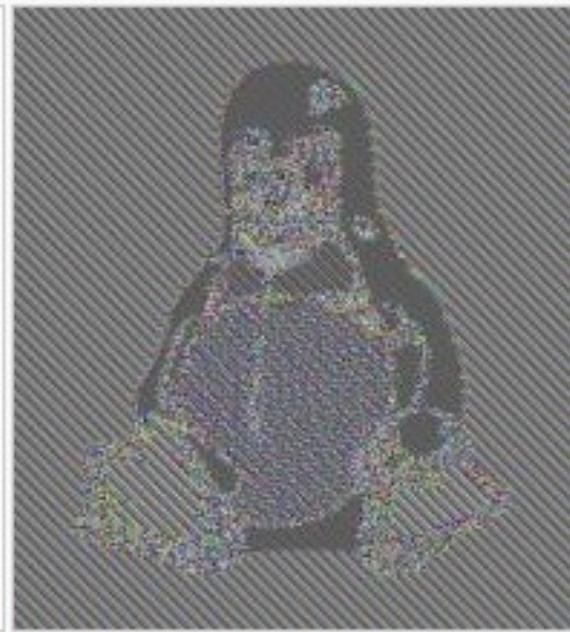


- Cipher Block Chaining (CBC)

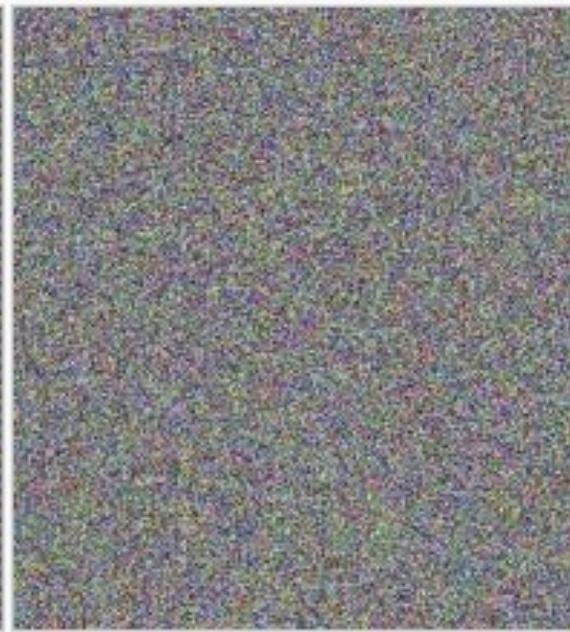




Original image

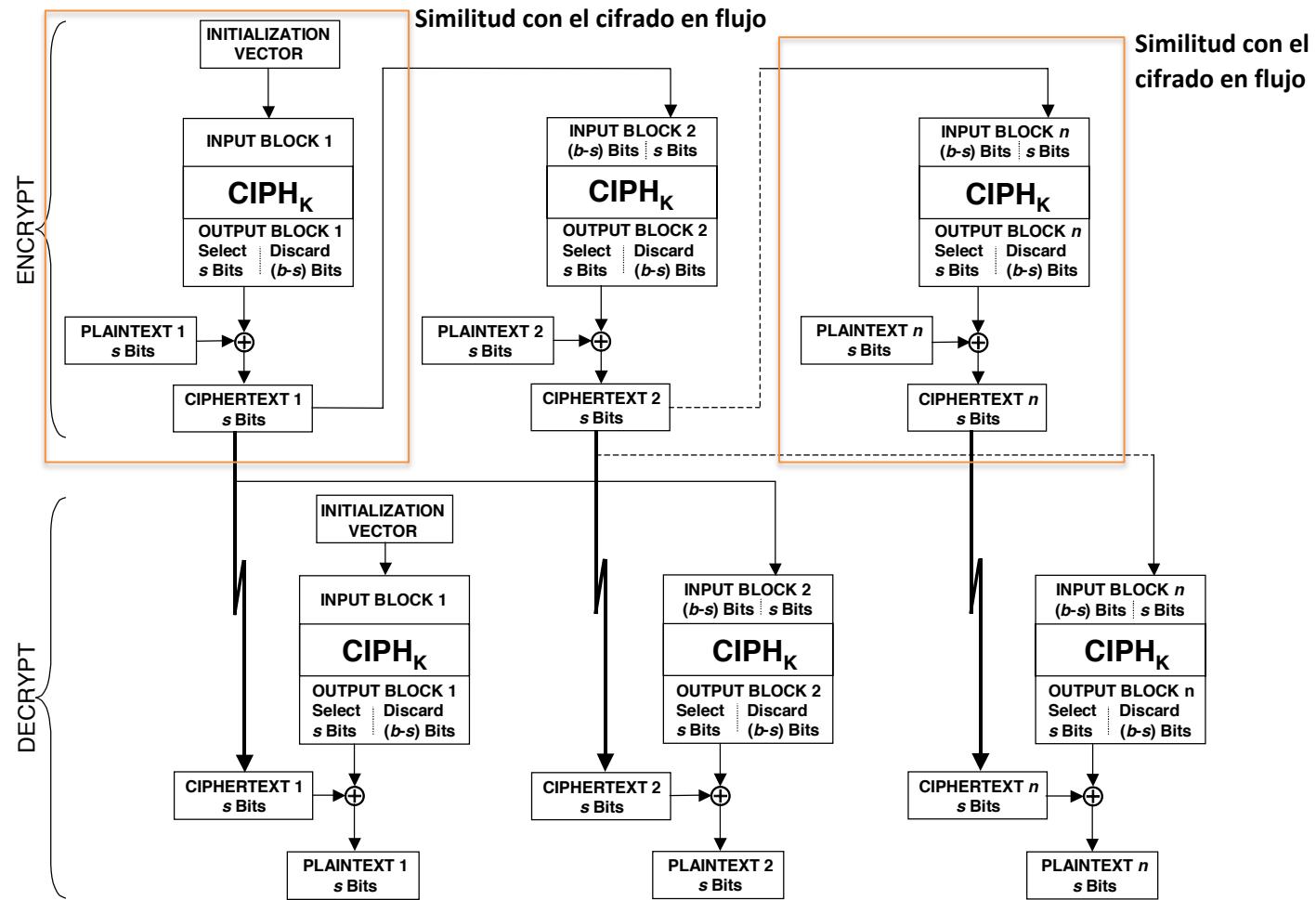


Encrypted using ECB mode

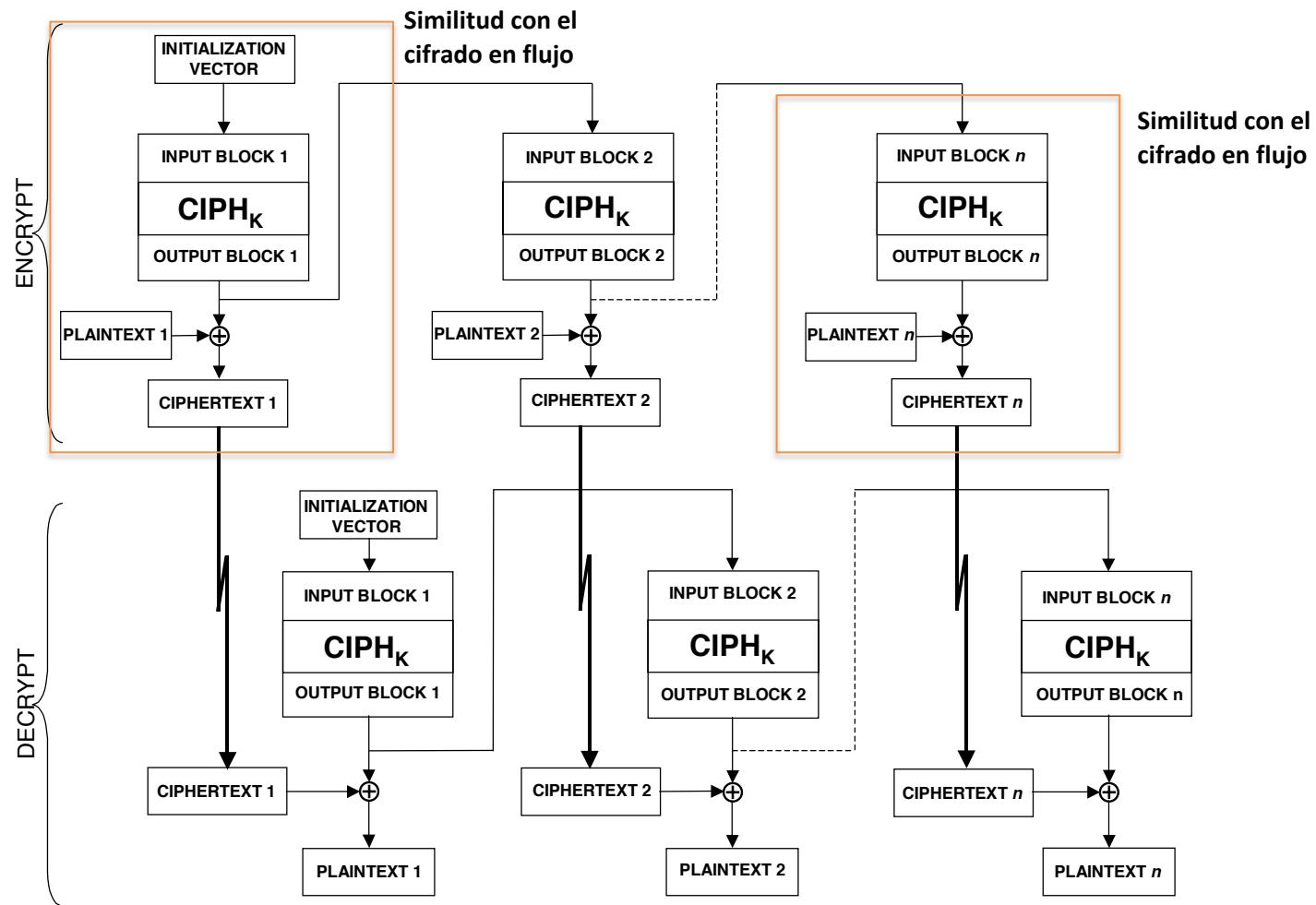


Modes other than ECB result in pseudo-randomness

• Cipher Feedback (CFB)

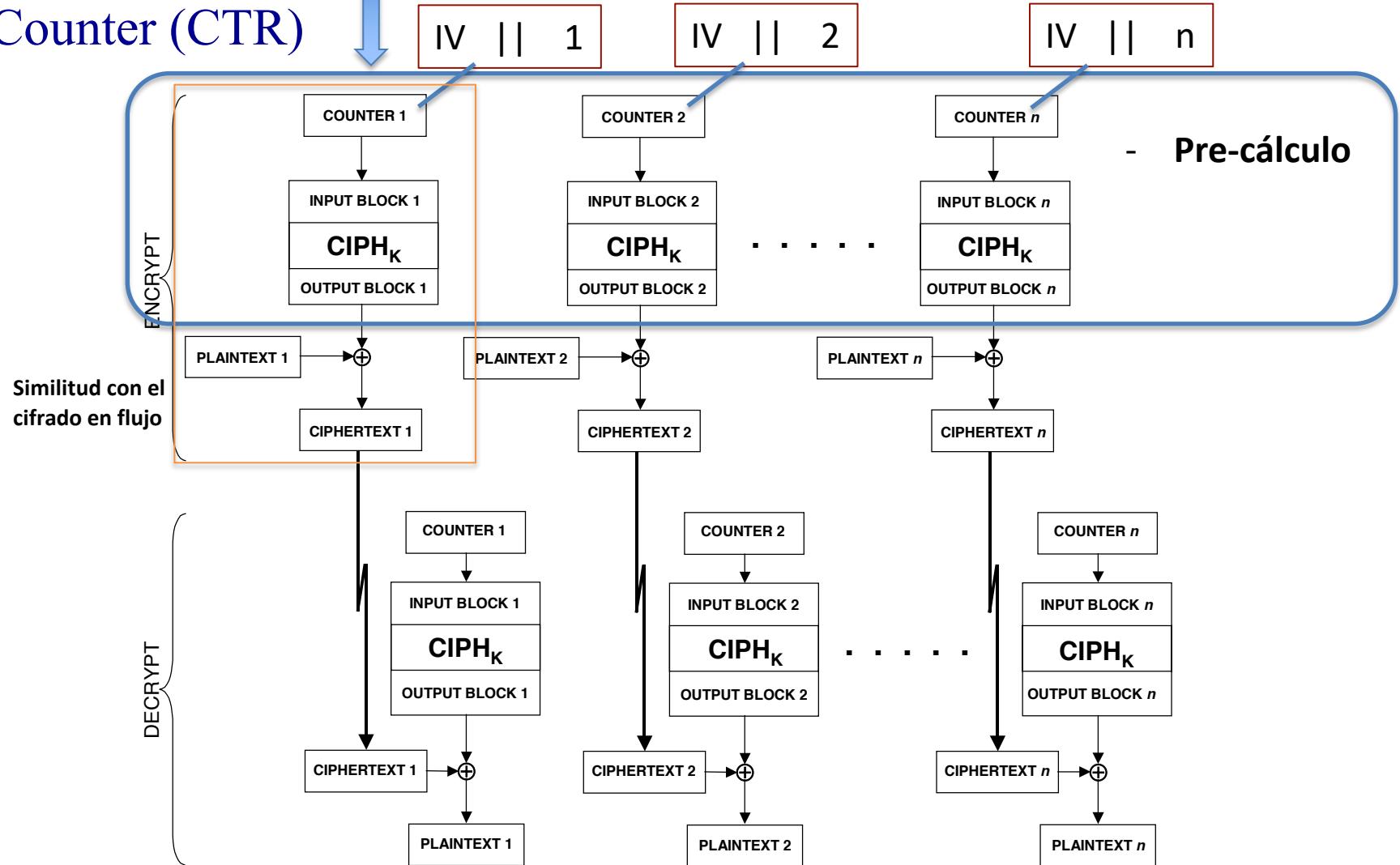


• Output Feedback (OFB)

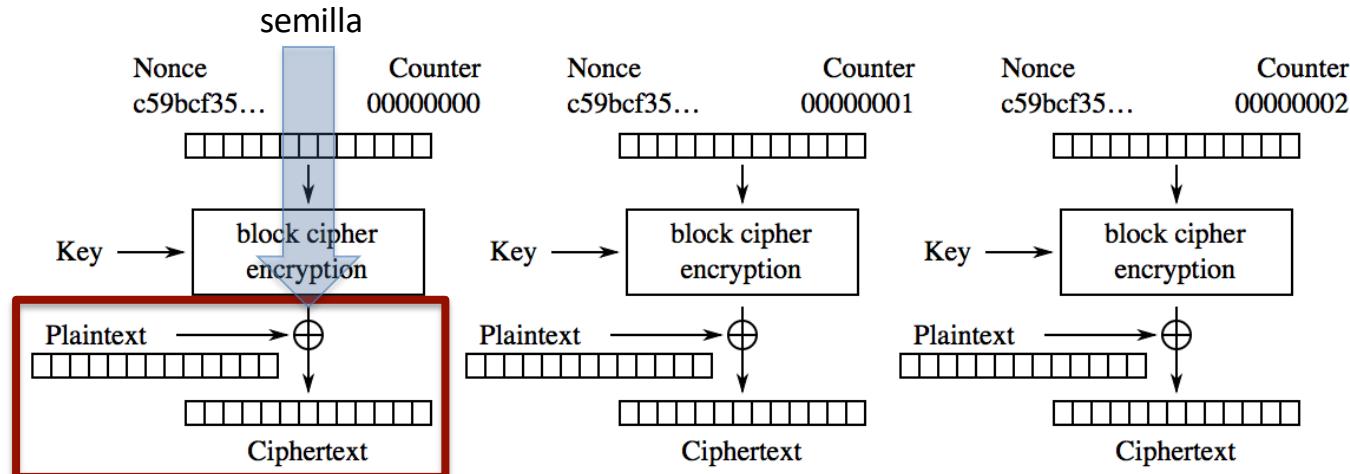


Normalmente se concatena el IV al contador

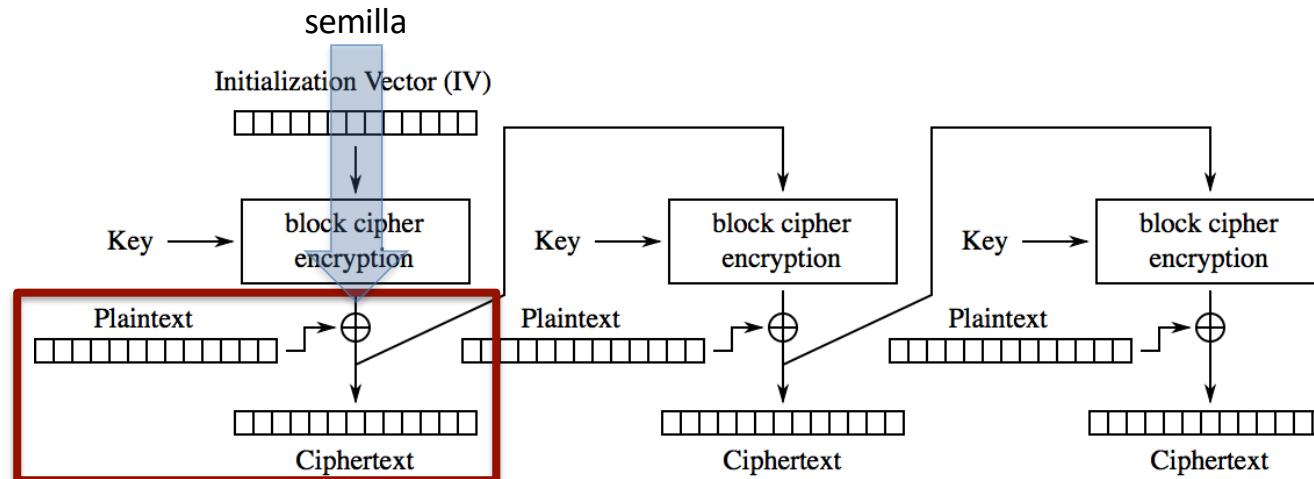
- Counter (CTR)



Otra forma de verlo - ¿ A qué se podría asemejar esto ?



Counter (CTR) mode encryption

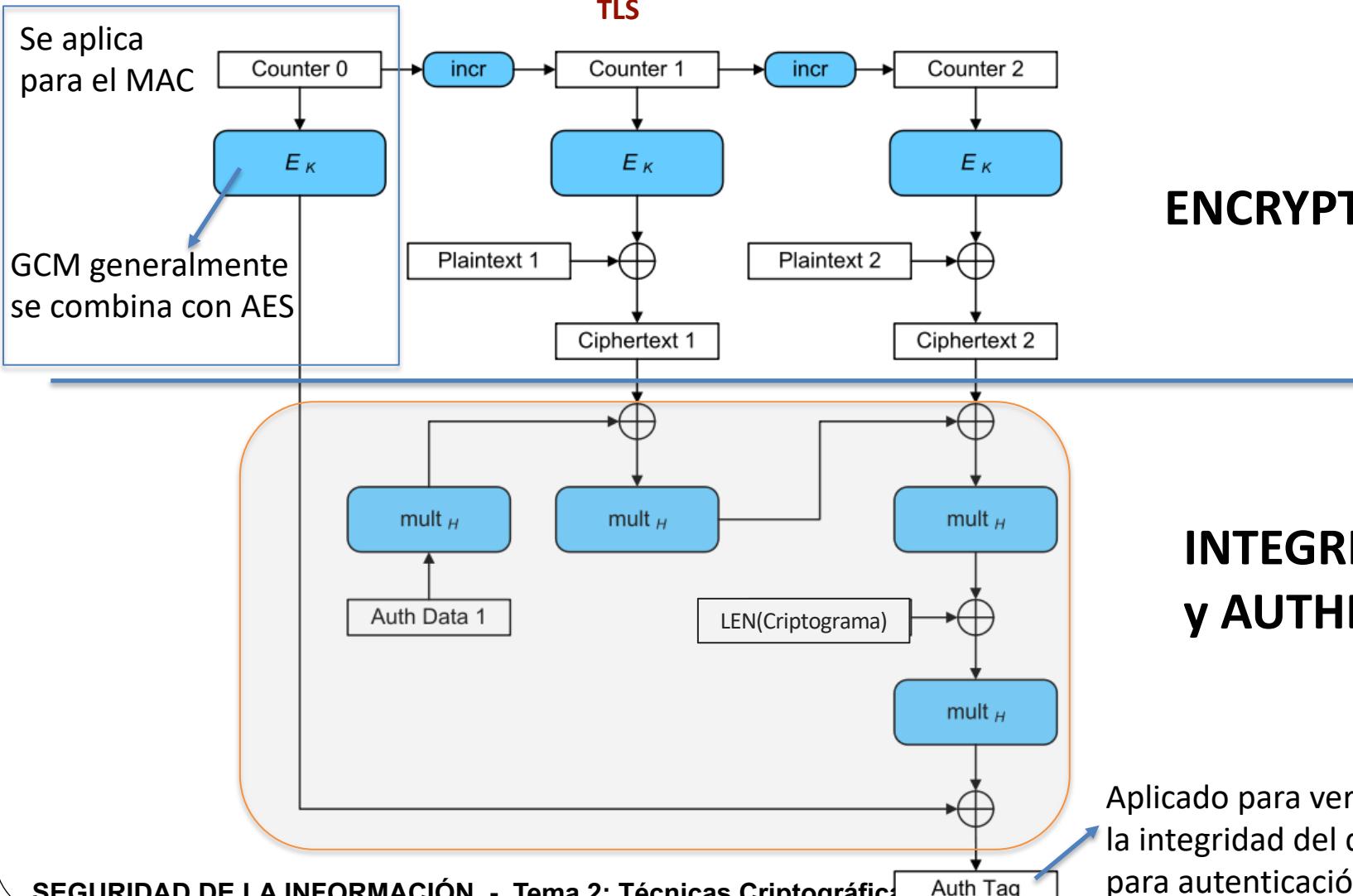


Cipher Feedback (CFB) mode encryption

Extra: Modos de operación con integridad

• Galois-CTR (GCM)

- Funciona de forma similar que el modo CTR pero usa Carter-Wegman MAC en un campo de Galois
- **Es rápido y eficiente, y está soportado por el suite de cifrado dado por TLS**

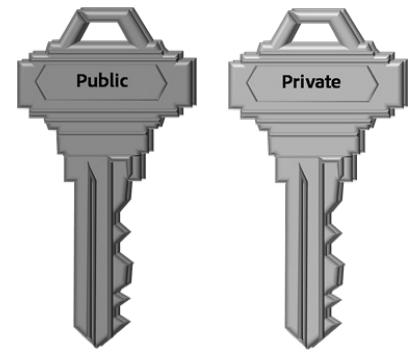


Ventajas y desventajas de los algoritmos simétricos

- **Ventajas:**
 - Los algoritmos simétricos se pueden diseñar para alcanzar un **alto rendimiento** (alto caudal de información cifrada)
 - En HW se pueden alcanzar del orden de cientos de Mbytes/sec
 - En SW se pueden alcanzar del orden Mbytes/sec
 - Se pueden **componer** para producir cifrados más fuertes
 - Recuerda: CIFRADO POR PRODUCTO
 - Se pueden utilizar como base para **construir otros mecanismos** criptográficos como:
 - **Funciones hash y**
 - **Generadores pseudo-aleatorios de números** (☺ i ya lo has visto !)
 - Los algoritmos simétricos necesitan:
 - **Claves K relativamente cortas**

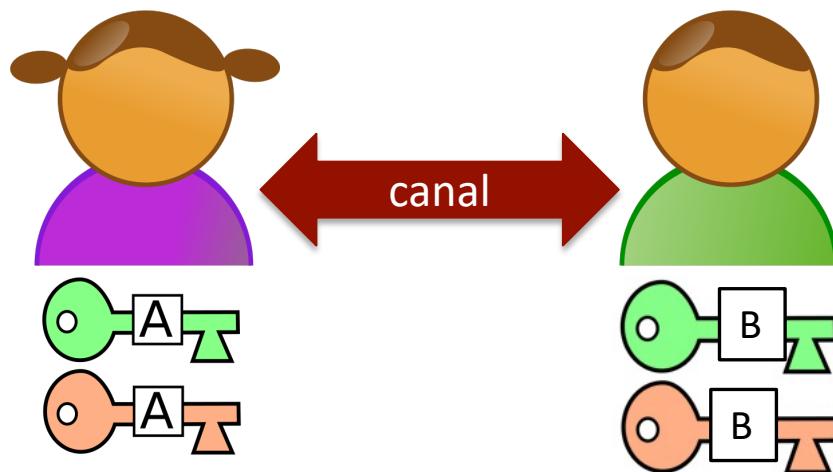
- **Desventajas:**
 - En una comunicación entre dos usuarios, estos han de **acordar, a priori, la clave K** con la que cifrarán/descifrarán sus comunicaciones
 - La clave ha de permanecer estrictamente en secreto, por lo que sólo la han conocer esos dos usuarios que se comunican
 - Si los dos usuarios están **físicamente lejanos** entre sí, acordar la clave puede convertirse en una tarea difícil
 - ¿Qué medio suficientemente seguro habrán de utilizar si no es posible una reunión presencial entre ambos?
 - Además, a efectos de seguridad, es recomendable que la clave K entre dos usuarios se cambie con cierta frecuencia, lo que complica el problema
 - En una red grande (de muchos usuarios) habrá demasiadas claves que administrar
 - Para una comunidad de n usuarios, el número de claves en el sistema será de:
 - $(n * (n-1)) / 2$; ej: 100 usuarios → 4950 claves
 - Probablemente, hará falta una **tercera parte confiable (ej. un administrador de claves)** para ayudar a los usuarios en las tareas de administración de claves

Algoritmos asimétricos (o de clave pública)



Introducción

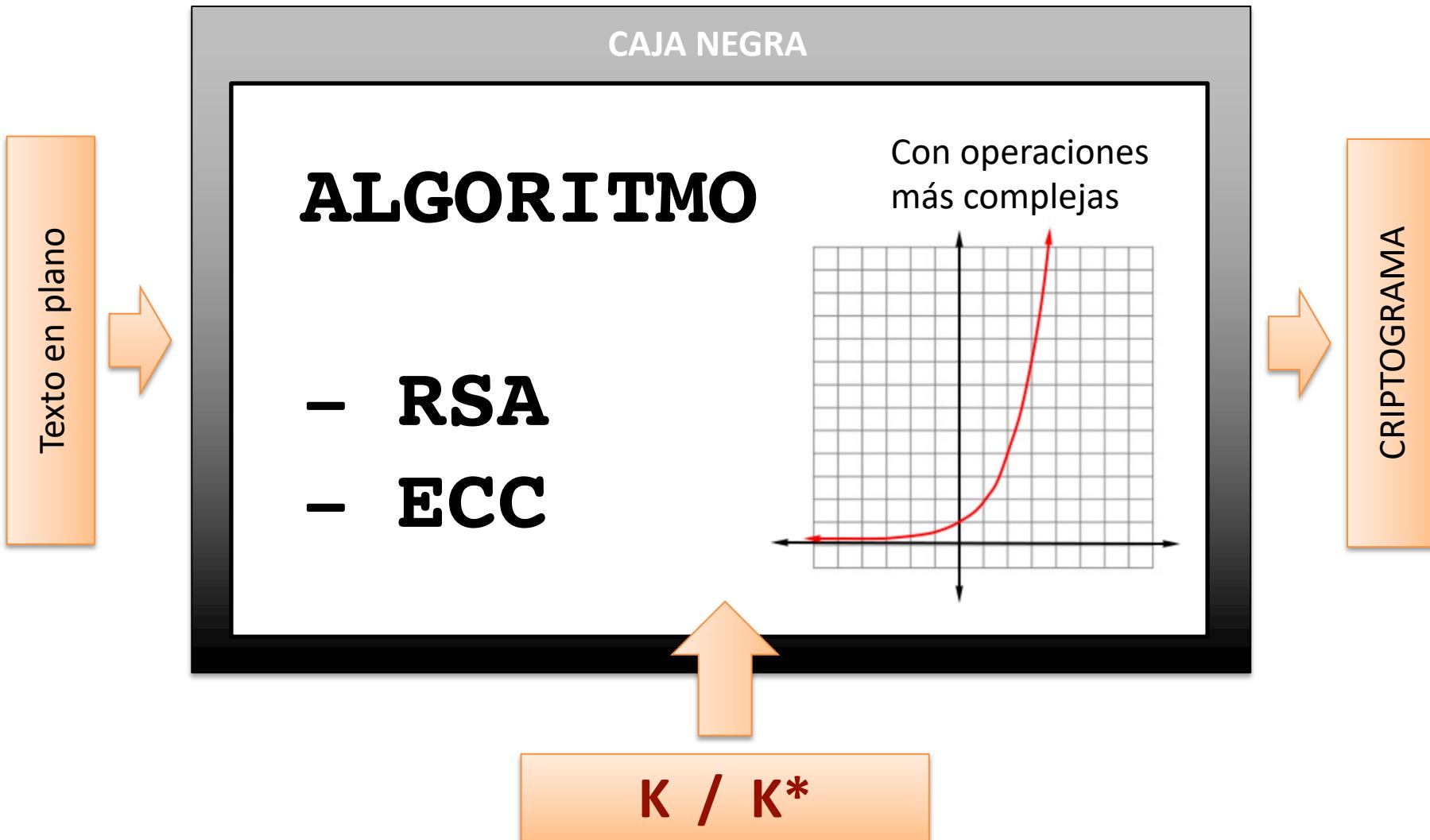
- El concepto de criptografía de clave pública (o asimétrica) fue inventado en 1976 por *Diffie y Hellman*, e independientemente por *Merkle*, para dar solución a algunos de los problemas de los criptosistemas simétricos



- La asimetría reside en que, **las claves K y K^* son distintas**, al contrario de lo que ocurría en el caso simétrico

Introducción

Si lo vemos como cajas negras



Introducción

- **K / K*:**
 - Las claves se utilizan por pares, de tal forma que cada usuario U posee dos claves:
 - Una **clave pública**, conocida por todos los usuarios
 - Una **clave privada**, conocida sólo por U
- Operaciones con K/K*:
 - Una clave se usa para cifrar, y la otra para descifrar
 - Es decir, si se cifra un mensaje con una de las claves, esa misma no servirá para descifrar, sino que necesariamente habrá que usar la otra
- Existen tres funcionalidades básicas con criptografía asimétrica



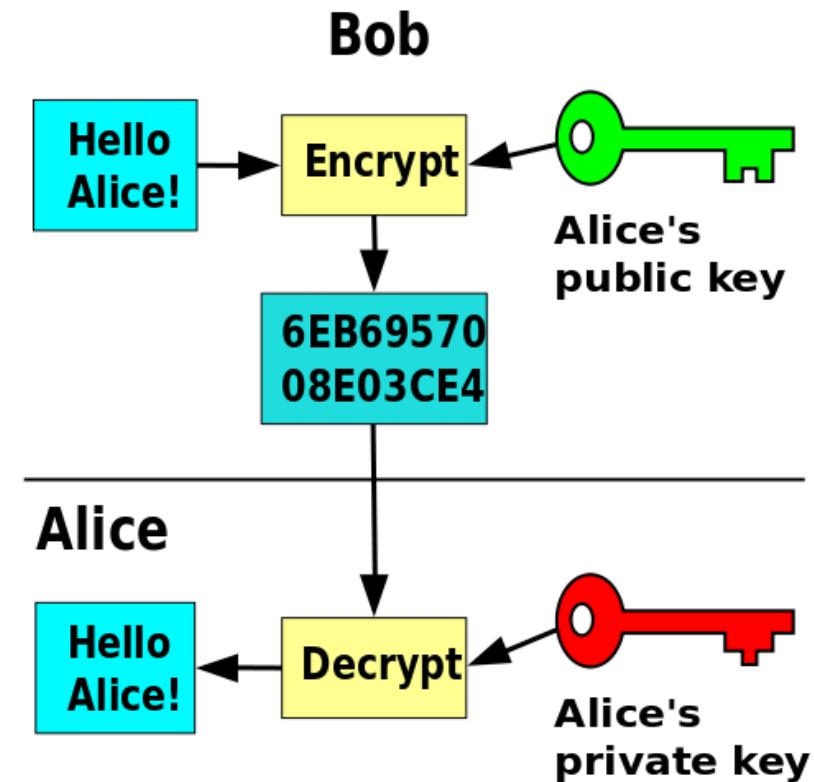
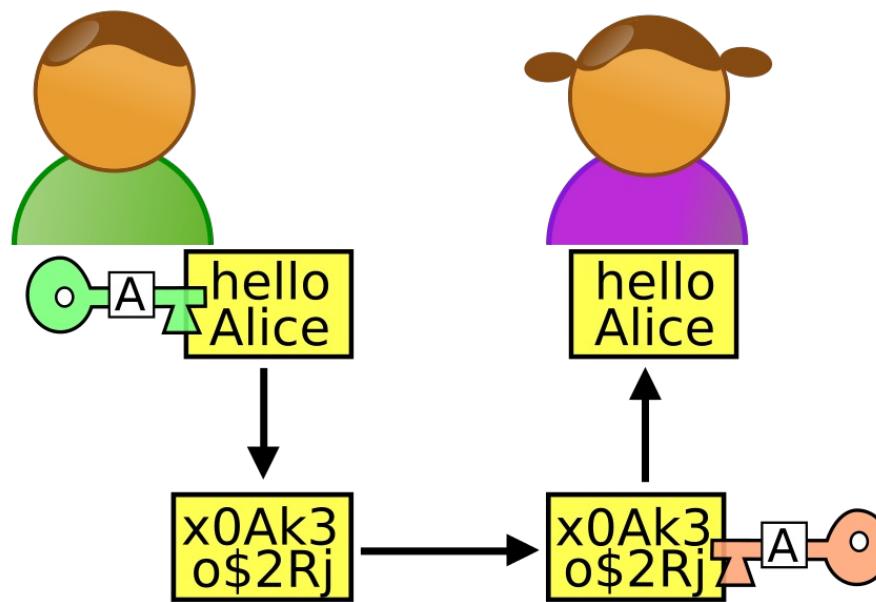
CIFRADO

FIRMA DIGITAL

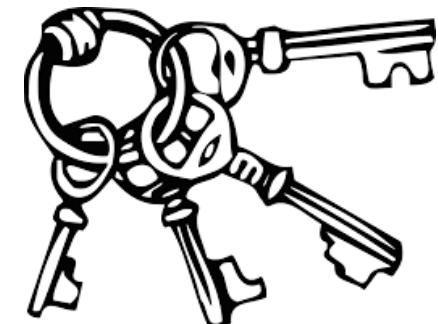
INTERCAMBIO DE
CLAVES

Cifrado/descifrado

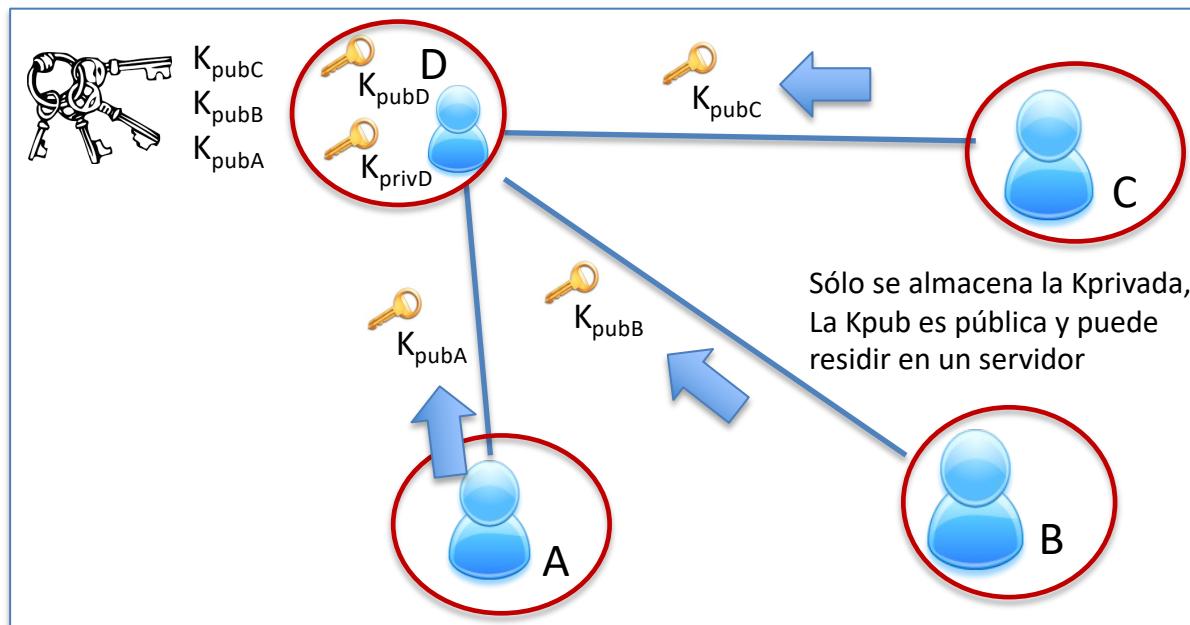
- En el caso de que *Bob* necesite del servicio de confidencialidad para su comunicación con *Alice*, el **cifrado/descifrado** se realiza de la siguiente forma:



- Del esquema anterior se desprende que *Alice* y *Bob* no necesitan acordar a priori ninguna clave (a diferencia de los algoritmos simétricos)
- Pero *Bob* ha de conocer la clave pública de *Alice*
 - Y también la clave pública de cada uno de los usuarios con los que desee contactar
- Para ello, existen varias soluciones para compartir la clave:
 - 1) Alice se lo proporciona mediante un conexión *peer-to-peer*
 - 2) Alice se lo proporciona por una de las vías más comunes: email, WhatsApp, sms, etc.
 - 3) Alice se lo deja en un repositorio común: foro, servidor de claves, etc.
- En cualquiera de los casos, *Bob* debe almacenar todas las claves públicas de todos los usuarios en un **key-ring** personal



- De lo anterior, también se deduce que, usando un criptosistema de clave pública, y para una comunidad de n usuarios:
 - el número de claves en el sistema será **$2n$**
 - en lugar de $(n * (n-1))/2$ como era el caso simétrico



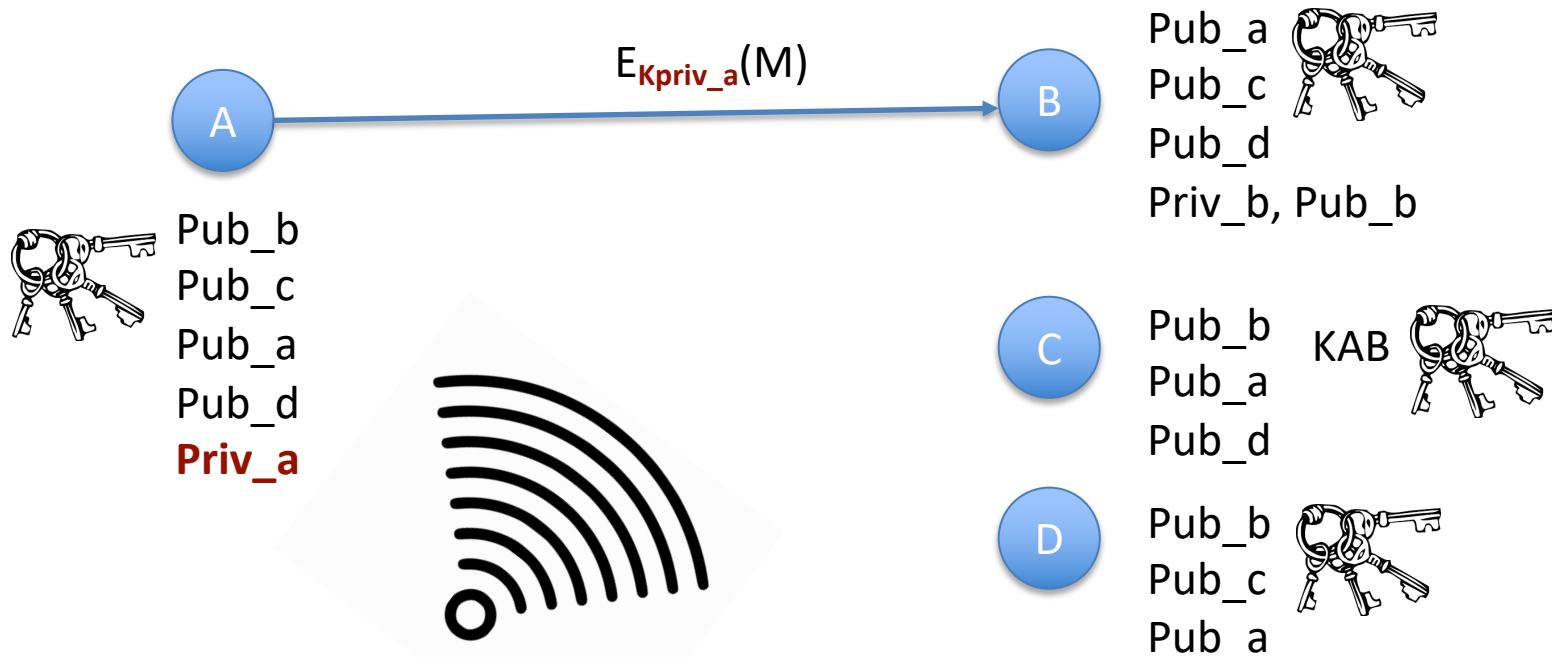
- Otras características relevantes que deberías saber:
 - Es **computacionalmente imposible deducir la clave privada** del usuario U a partir de su clave pública
 - Cualquier usuario con la **clave pública de U puede cifrar un mensaje hacia U** , pero no descifrarlo
 - Cifrar el mensaje con la clave pública es como poner el correo en un buzón (todo el mundo puede hacerlo)
 - Sólo U , con la correspondiente **clave privada, puede descifrar el mensaje**
 - Descifrar el mensaje con la clave privada es como coger el correo del buzón
 - Sólo el que tiene la llave del buzón puede hacerlo



LO QUE UNA HACE, LA OTRA LA DESHACE

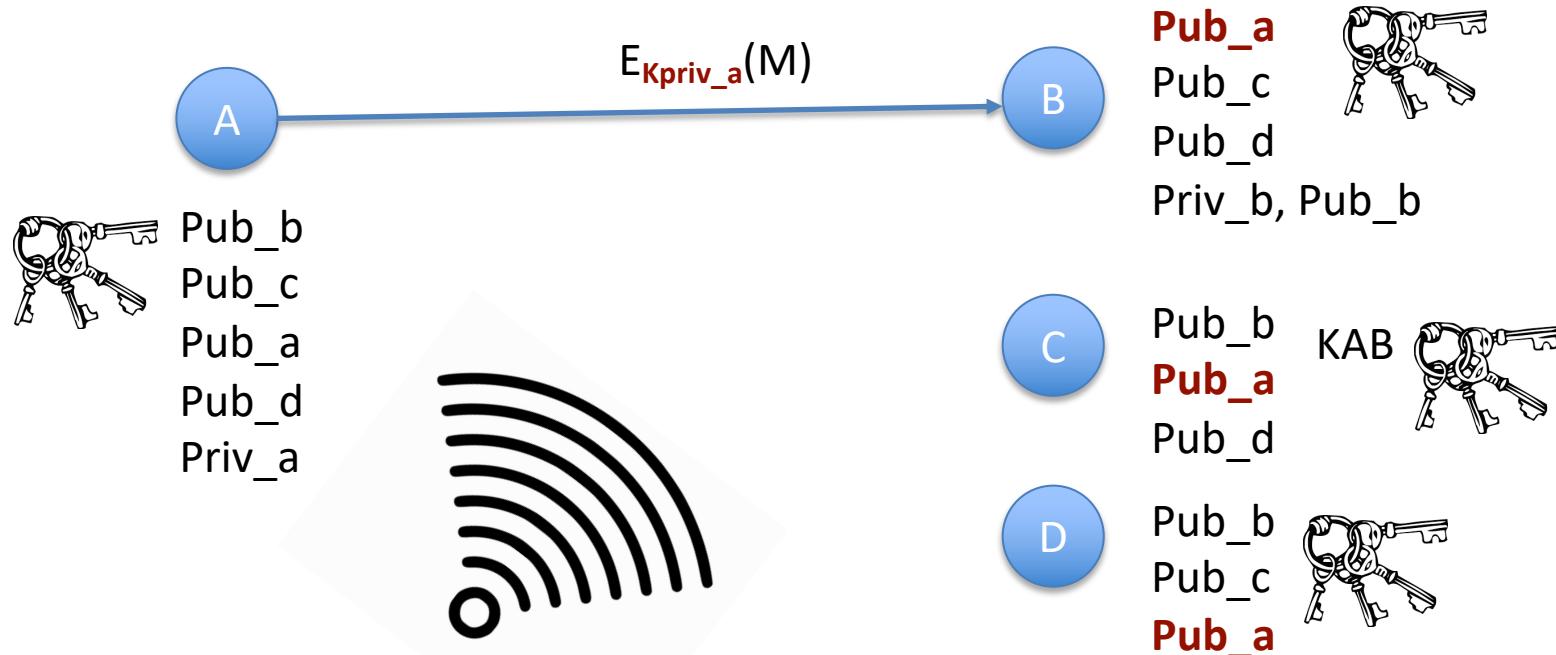
Algunos casos de ejemplos - Queremos confidencialidad

CASO A: A envía a B un mensaje con su clave privada



Algunos casos de ejemplos - Queremos confidencialidad

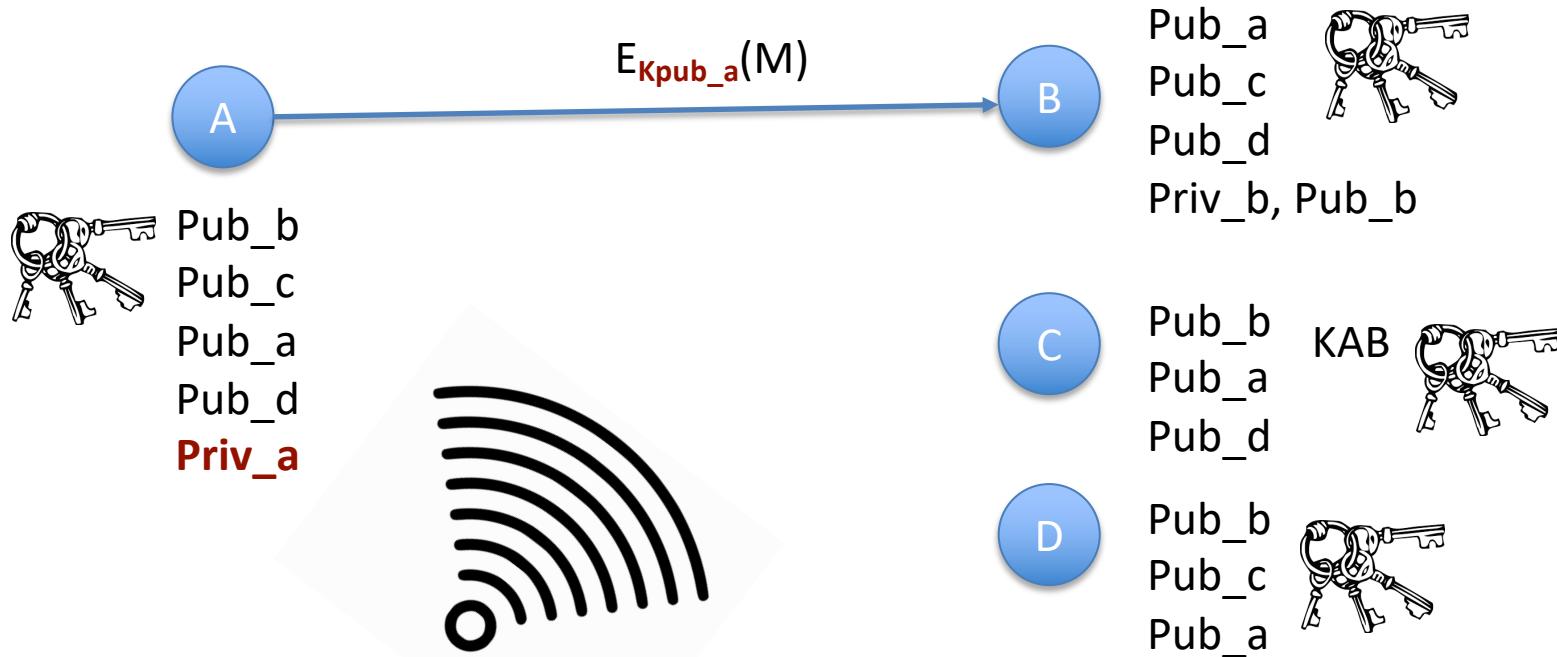
CASO A: A envía a B un mensaje con su clave privada - **¡¡ NO !!**



Si se cifra el mensaje con la clave privada → **NO hay confidencialidad** porque no sólo B tiene la Kpub_a, sino también C y D, por lo que ellos también podrían leer el mensaje

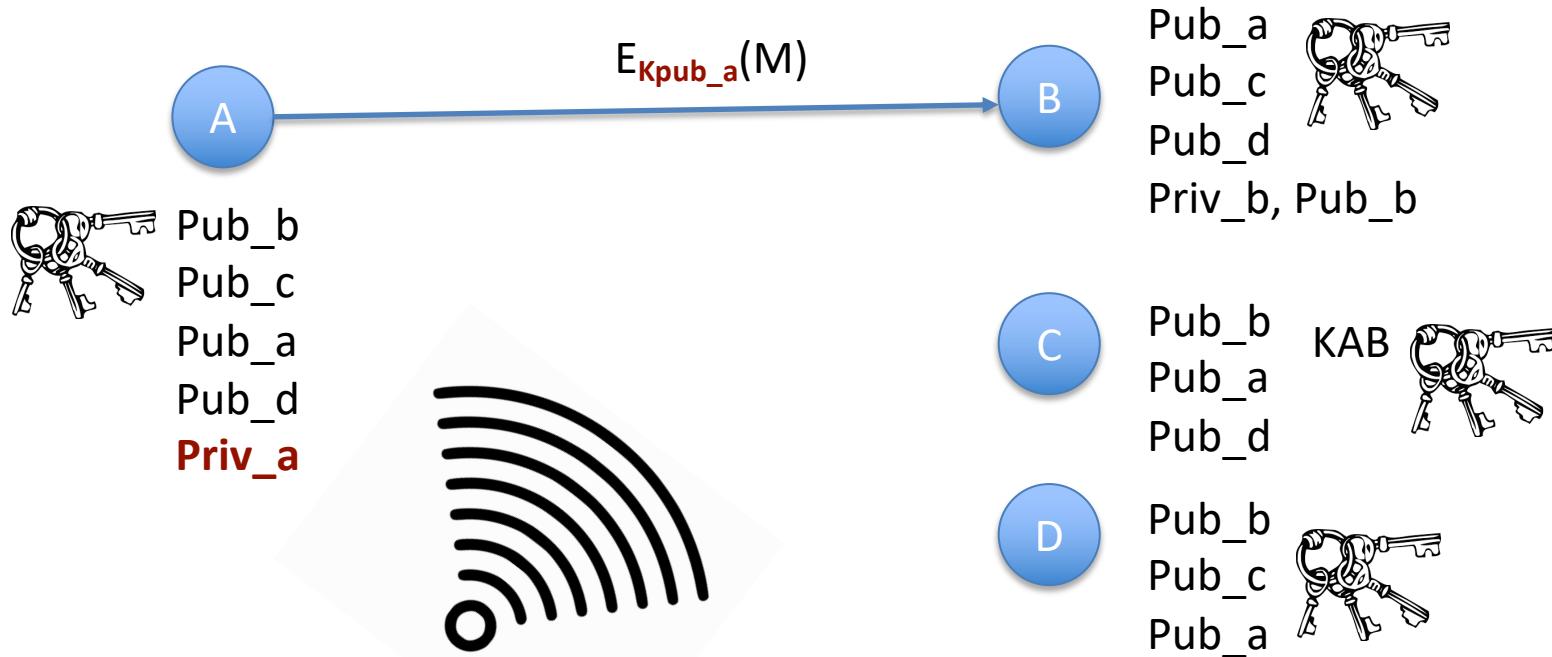
Algunos casos de ejemplos - Queremos confidencialidad

CASO B: A envía a B un mensaje con su clave pública



Algunos casos de ejemplos - Queremos confidencialidad

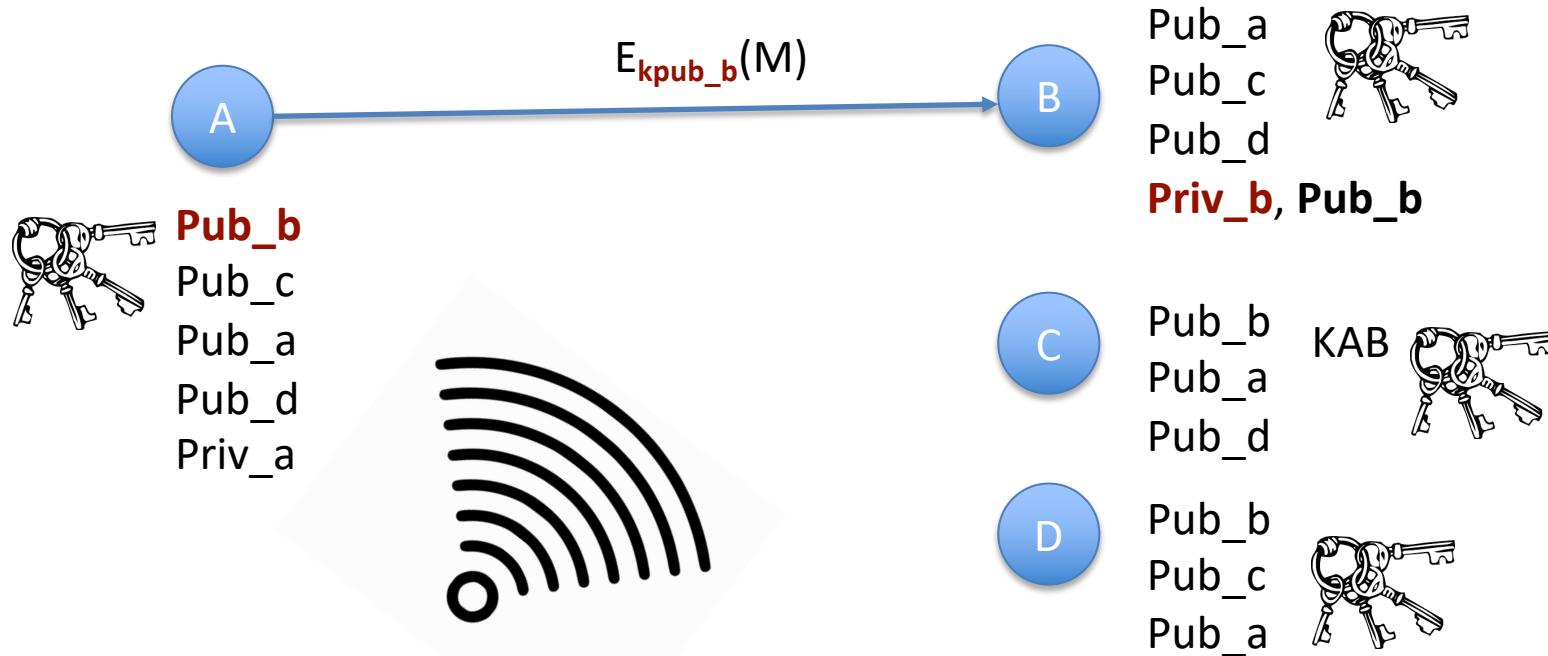
CASO B: A envía a B un mensaje con su clave pública - **¡¡ NO !!**



Si se cifra el mensaje con su propia clave pública → ¡¡ Ni la otra parte ni el resto de comunicación podrán leer el mensaje !! Porque el que tiene la clave privada es el que ha enviado el mensaje

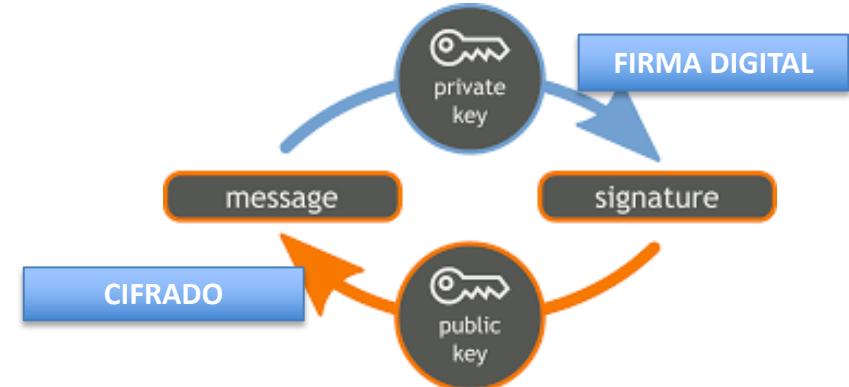
Algunos casos de ejemplos - Queremos confidencialidad

CASO C: A envía a B un mensaje con la clave pública de B - **¡¡ SÍ !!**



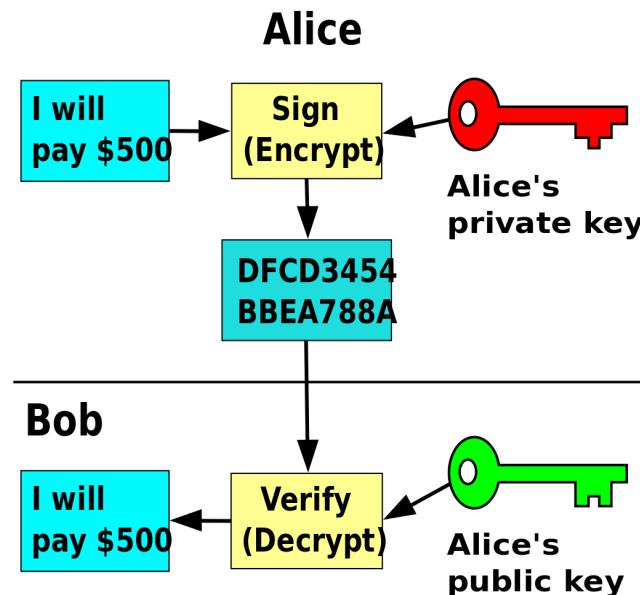
Si se cifra el mensaje con la clave pública de B → **HAY
confidencialidad** porque sólo B tiene la Kpriv_b asociada a Kpub_b, y C y D no podrán leer el mensaje

- Por lo tanto, hemos visto tres **ventajas inmediatas de la criptografía de clave pública** con respecto a la simétrica
 1. Cuando dos usuarios se comunican confidencialmente **no necesitan acordar una clave a priori**
 2. Por lo anterior, **no resulta problemático que estén físicamente lejanos y no puedan reunirse presencialmente**
 3. El **número de claves** en el sistema **se reduce** sustancialmente
- Como se ve en la figura, existe aún una **ventaja adicional** tanto o más importante que las anteriores
 - Esa ventaja se deriva de la **dualidad de funcionamiento** de algunos (no todos) algoritmos de clave pública



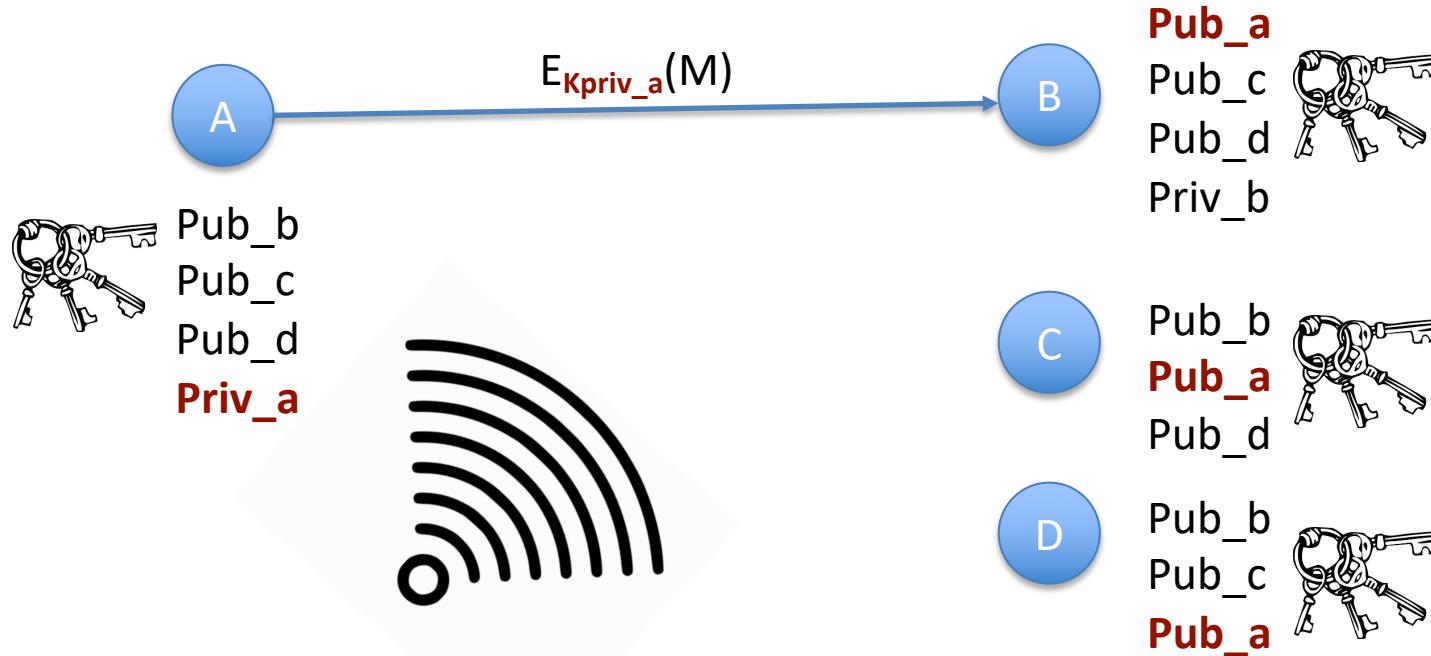
Firma digital

- *Alice* cifra el mensaje usando su propia clave privada, y cualquiera que tenga la clave pública de *Alice* podrá descifrar el criptograma
- Cuando *Bob* descifra el criptograma y obtiene el mensaje original, le queda garantizado que el mensaje viene de *Alice*
 - Porque *Alice* es la única que pudo hacer la operación de cifrado (ya que sólo ella posee la clave privada que generó el criptograma)



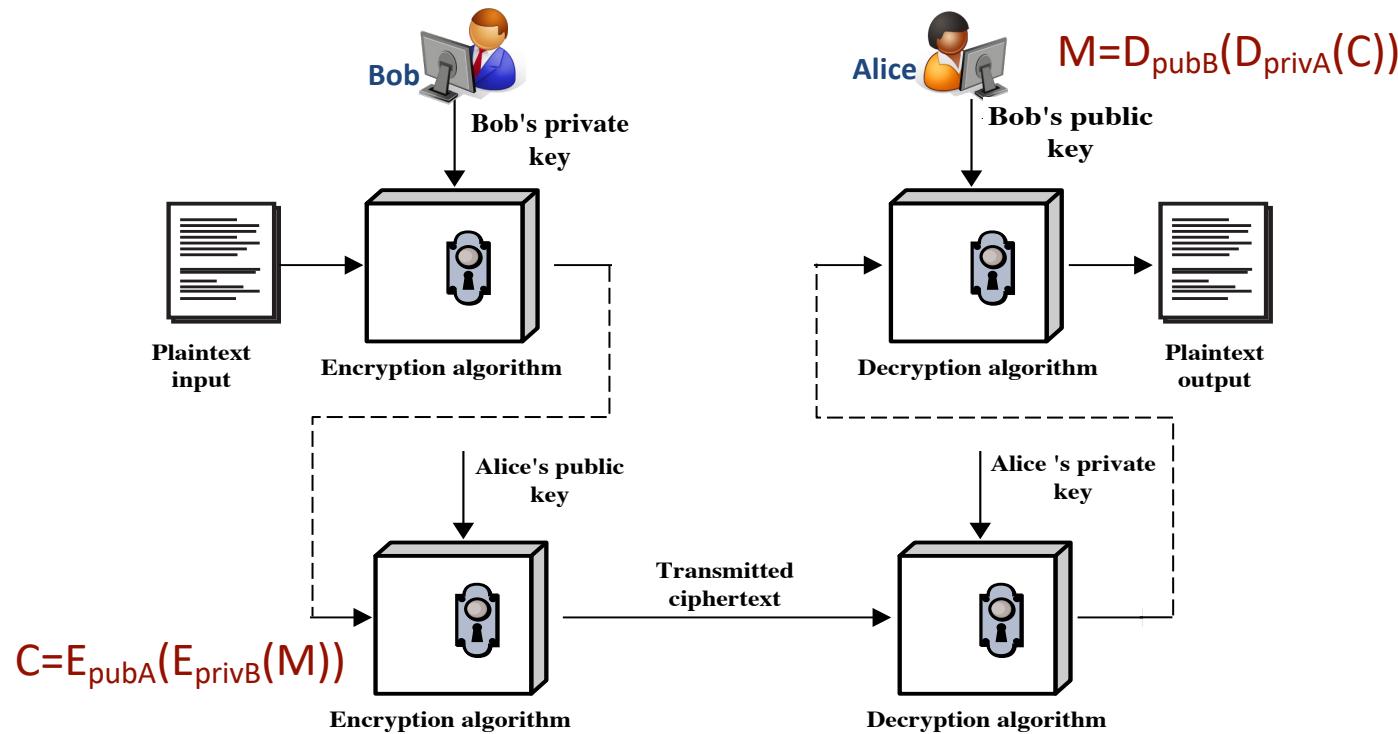
Algunos casos de ejemplos - Queremos autenticación

CASO A: A envía a B un mensaje firmado con su clave privada - **¡¡ SÍ !!**



Firma digital + Cifrado

- Adicionalmente, es posible usar en secuencia las operaciones de firma digital y cifrado/descifrado, obteniendo autenticidad y confidencialidad en un mismo envío



Desventajas de criptografía de clave pública

- A pesar de estas ventajas, los algoritmos de clave pública tienen una **desventaja** de peso, por ejemplo, aplican **claves de gran tamaño**
 - Ejemplo de clave pública:

```
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 2d 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
```



- Además, los algoritmos de clave pública se basan en **funciones matemáticas complejas**
 - En lugar de las convencionales sustituciones, permutaciones y sumas de los criptosistemas simétricos
- Ambos hechos hacen que **el rendimiento** de estos algoritmos sea **sustancialmente menor** que el de los simétricos
 - En general, se puede afirmar que son unos 1000 veces más lentos

Desventajas de criptografía de clave pública

```
$ openssl speed rc4  
To get the most accurate results, try to run this  
program when this computer is idle.  
Doing rc4 for 3s on 16 size blocks: 73270739 rc4's in 2.99s  
Doing rc4 for 3s on 64 size blocks: 19548456 rc4's in 2.99s  
Doing rc4 for 3s on 256 size blocks: 5017905 rc4's in 2.99s  
Doing rc4 for 3s on 1024 size blocks: 1274653 rc4's in 2.98s  
Doing rc4 for 3s on 8192 size blocks: 159407 rc4's in 2.97s
```

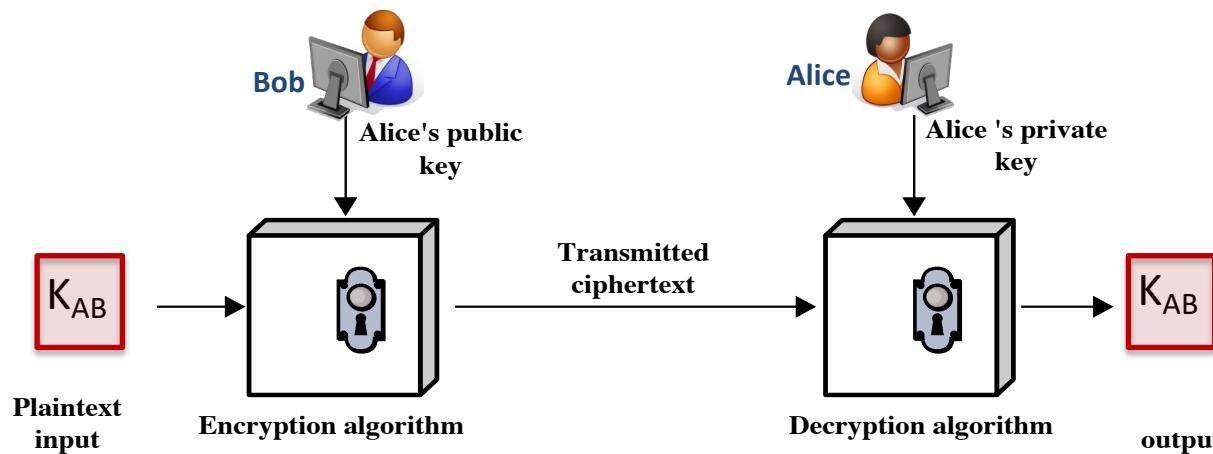
```
$ openssl speed aes  
To get the most accurate results, try to run this  
program when this computer is idle.  
Doing aes-128 cbc for 3s on 16 size blocks: 30108378 aes-128 cbc's in 2.97s  
Doing aes-128 cbc for 3s on 64 size blocks: 7712443 aes-128 cbc's in 2.96s  
Doing aes-128 cbc for 3s on 256 size blocks: 1953741 aes-128 cbc's in 2.98s  
Doing aes-128 cbc for 3s on 1024 size blocks: 490976 aes-128 cbc's in 2.98s  
Doing aes-128 cbc for 3s on 8192 size blocks: 61237 aes-128 cbc's in 2.98s  
Doing aes-192 cbc for 3s on 16 size blocks: 26695873 aes-192 cbc's in 2.98s  
Doing aes-192 cbc for 3s on 64 size blocks: 6930418 aes-192 cbc's in 2.98s  
Doing aes-192 cbc for 3s on 256 size blocks: 1729199 aes-192 cbc's in 2.97s  
Doing aes-192 cbc for 3s on 1024 size blocks: 444845 aes-192 cbc's in 2.98s  
Doing aes-192 cbc for 3s on 8192 size blocks: 52989 aes-192 cbc's in 2.97s  
Doing aes-256 cbc for 3s on 16 size blocks: 23329778 aes-256 cbc's in 2.97s  
Doing aes-256 cbc for 3s on 64 size blocks: 5958585 aes-256 cbc's in 2.98s  
Doing aes-256 cbc for 3s on 256 size blocks: 1565944 aes-256 cbc's in 2.97s  
Doing aes-256 cbc for 3s on 1024 size blocks: 377290 aes-256 cbc's in 2.97s  
Doing aes-256 cbc for 3s on 8192 size blocks: 47844 aes-256 cbc's in 2.94s
```

```
$ openssl speed rsa  
To get the most accurate results, try to run this  
program when this computer is idle.  
Doing 512 bit private rsa's for 10s: 79651 512 bit private RSA's in 9.98s  
Doing 512 bit public rsa's for 10s: 1079143 512 bit public RSA's in 9.95s  
Doing 1024 bit private rsa's for 10s: 22746 1024 bit private RSA's in 9.96s  
Doing 1024 bit public rsa's for 10s: 460663 1024 bit public RSA's in 9.96s  
Doing 2048 bit private rsa's for 10s: 4362 2048 bit private RSA's in 9.96s  
Doing 2048 bit public rsa's for 10s: 174994 2048 bit public RSA's in 9.97s  
Doing 4096 bit private rsa's for 10s: 729 4096 bit private RSA's in 9.98s  
Doing 4096 bit public rsa's for 10s: 50938 4096 bit public RSA's in 9.98s
```

```
$ openssl speed dsa  
To get the most accurate results, try to run this  
program when this computer is idle.  
Doing 512 bit sign dsa's for 10s: 125836 512 bit DSA signs in 9.99s  
Doing 512 bit verify dsa's for 10s: 114530 512 bit DSA verify in 9.99s  
Doing 1024 bit sign dsa's for 10s: 54566 1024 bit DSA signs in 10.00s  
Doing 1024 bit verify dsa's for 10s: 46194 1024 bit DSA verify in 10.00s  
Doing 2048 bit sign dsa's for 10s: 18965 2048 bit DSA signs in 10.00s  
Doing 2048 bit verify dsa's for 10s: 16315 2048 bit DSA verify in 10.00s
```

Intercambio de claves → Criptografía híbrida

- Debido a su bajo rendimiento, se ha ideado una tercera funcionalidad para estos criptosistemas (además de cifrado/descifrado y firma digital): el **intercambio de claves**
 - *Paso 1: Bob y Alice usan el algoritmo asimétrico para la transmisión (cifrado/descifrado) de la clave secreta K_{AB}*
 - *Paso 2: Ambos usarán K_{AB} para, posteriormente, cifrar sus comunicaciones con un algoritmo simétrico*



- El resultado final es un **criptosistema híbrido**
 - Uso de un criptosistema de clave pública + un criptosistema simétrico

Intercambio de claves → Criptografía híbrida

C. Simétrica

R+:

- los algoritmos son más simples y demandan menos recursos

S-:

- las claves son pequeñas y se pueden derivar por fuerza bruta
- Las claves son inseguras y requieren frecuentes procesos de rekeying

C. Asimétrica

S+:

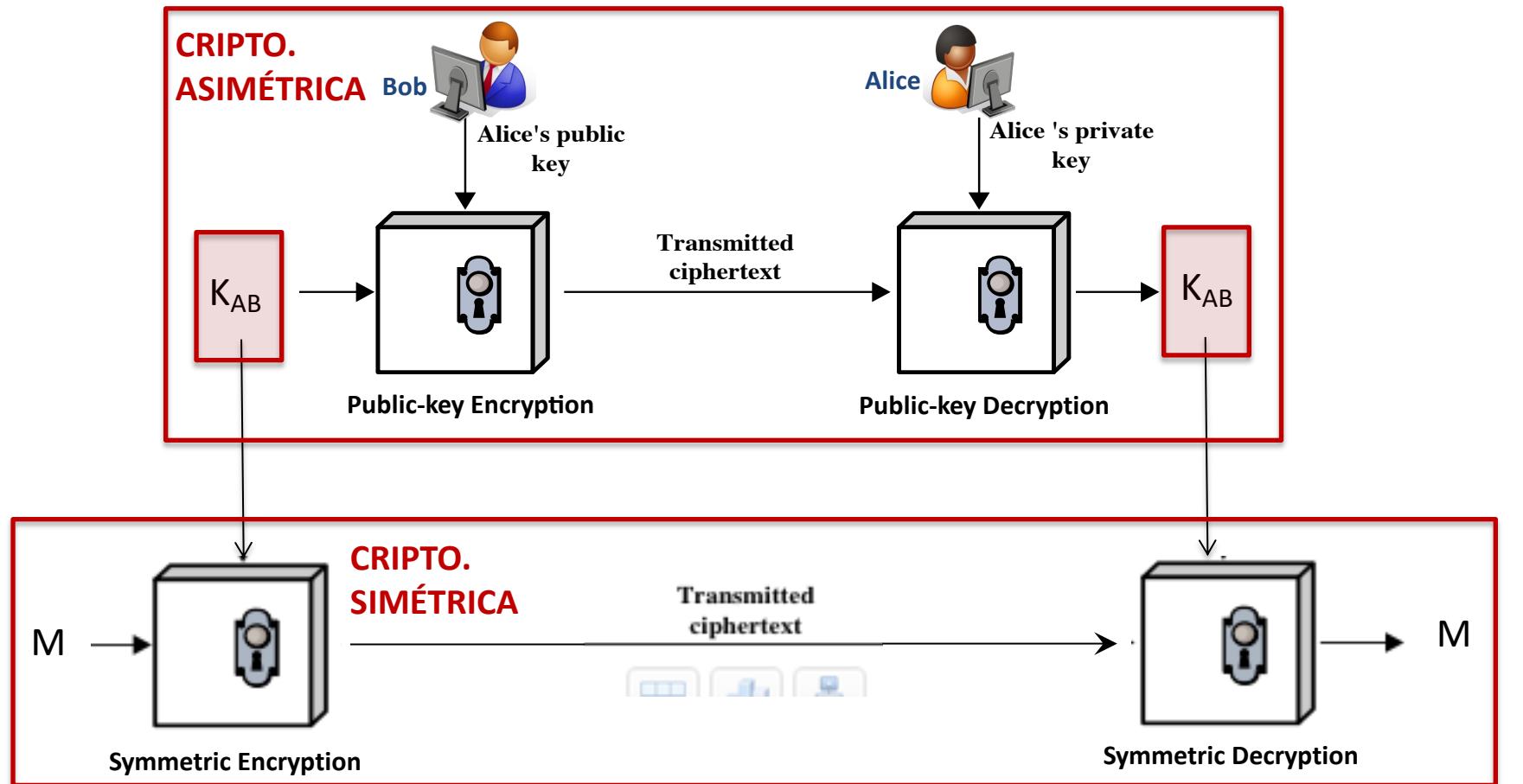
- las claves son de tamaño mayor, y la Kpriv no se puede (computalmente y matemáticamente hablando) derivar de la Kpub
- Los algoritmos son más robustos frente ataques y permite enviar datos seguros por cualquier medio

R-:

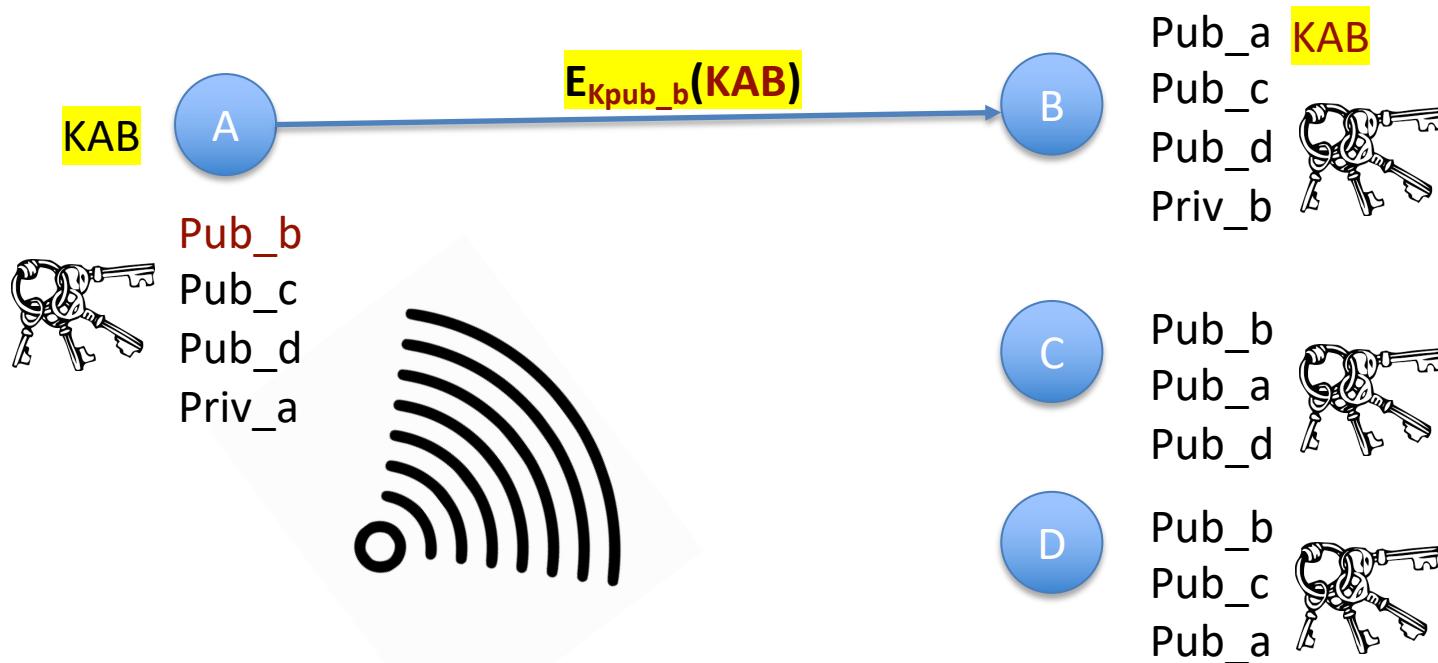
- los algoritmos son complejos y las claves son grandes, lo que demanda recursos para computar el crifrado

Intercambio de claves → Criptografía híbrida

- Una evolución del planteamiento anterior es el siguiente:
 - Enviar simultáneamente la clave de sesión y el mensaje cifrado con esa misma clave para **aprovechar los canales de comunicación**

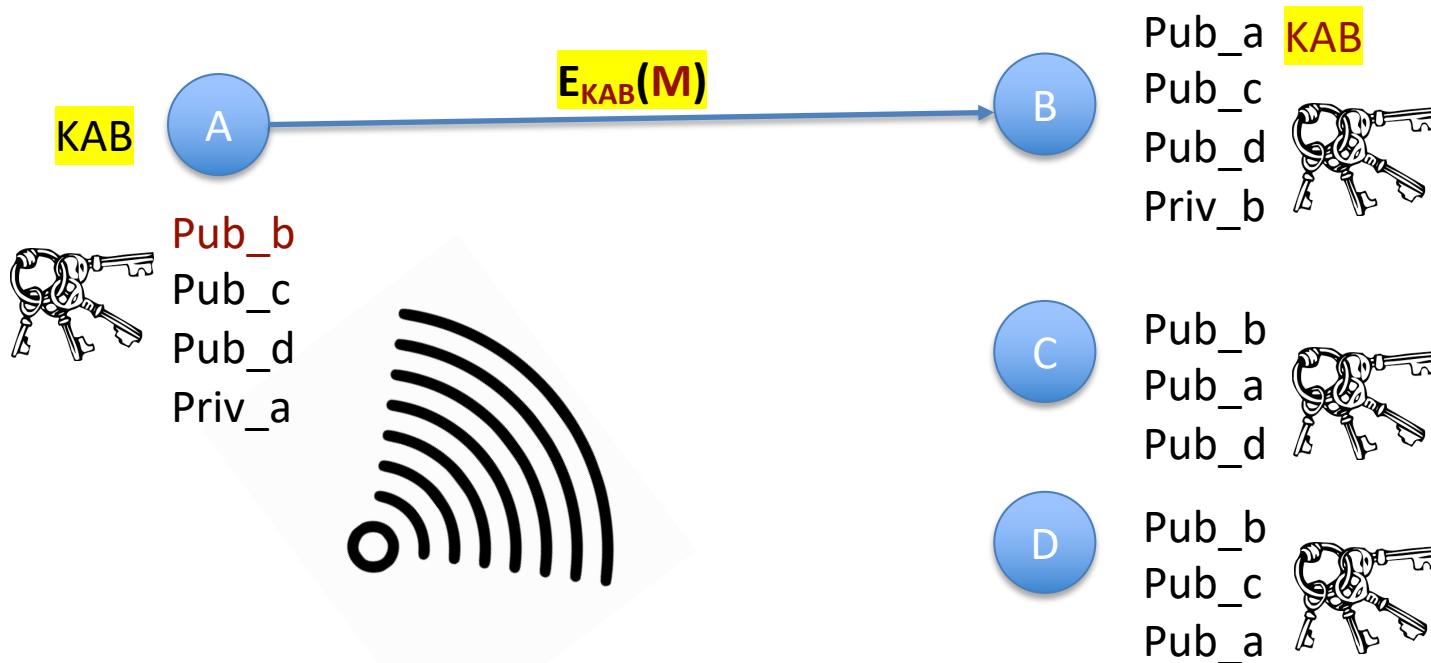


Algunos casos de ejemplos - Queremos criptografía híbrida



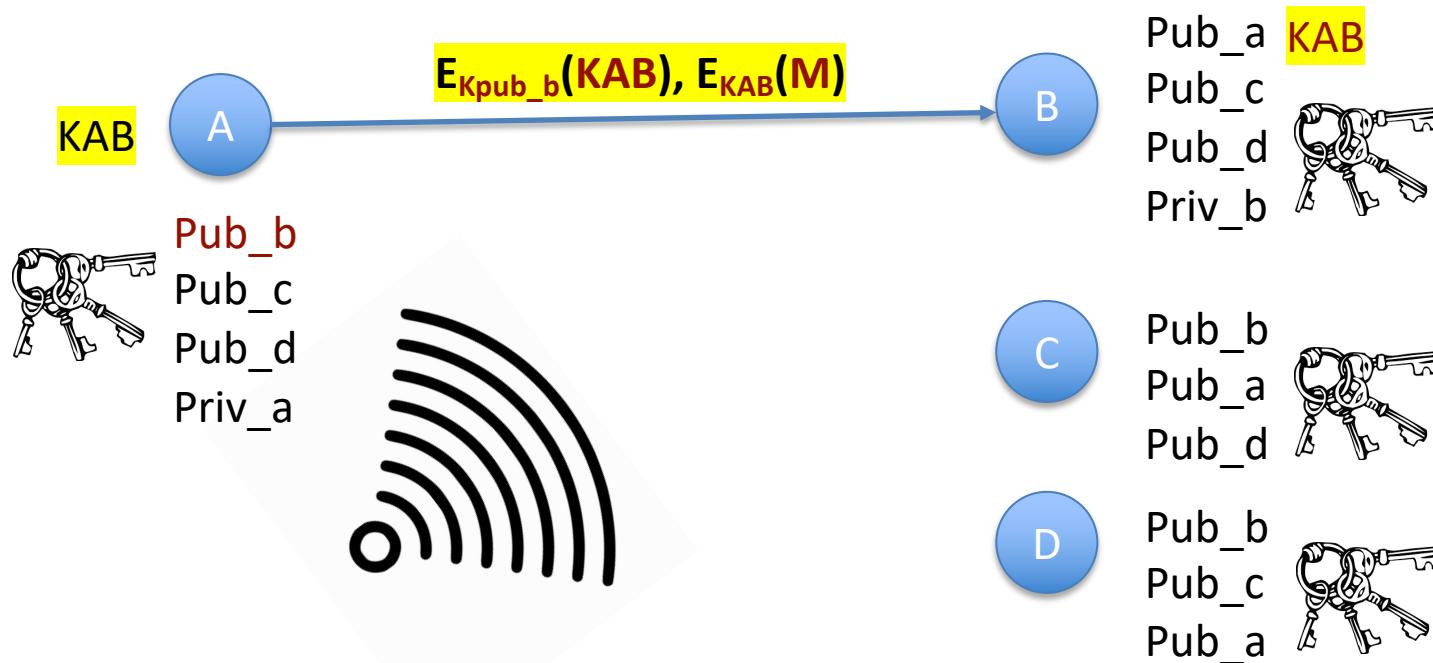
Fase 1: PROTEGER la KAB

Algunos casos de ejemplos - Queremos criptografía híbrida



Fase 2: PROTEGER el mensaje M

Algunos casos de ejemplos - Queremos criptografía híbrida



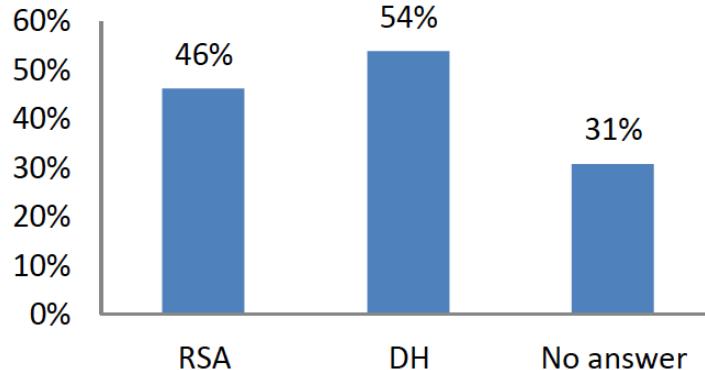
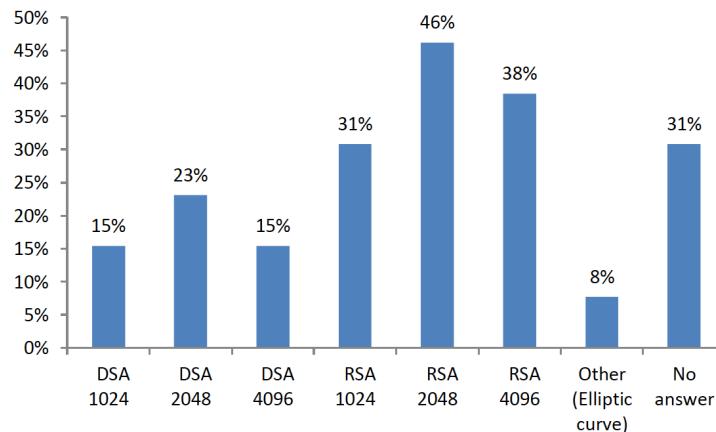
Fase 1 + 2: para proteger el
canal de comunicaciones

Funcionalidades de la criptografía de clave pública

- Como se muestra en la siguiente tabla, no todos los algoritmos de clave pública son capaces de realizar las tres funcionalidades mencionadas:

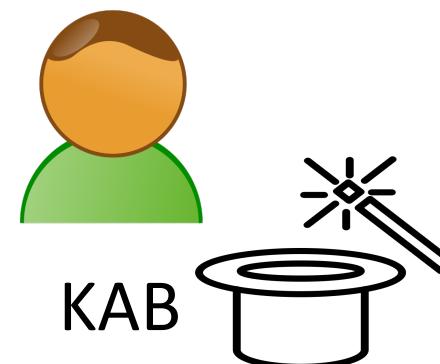
Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

DSS: Digital Signature Standard, e incorpora DSA (Digital Signature Algorithm)



Algoritmo Diffie-Hellman

- Diseñado en 1976 y se conoce como el “**algoritmo Diffie-Hellman (DH) de intercambio de clave**”
 - Permite a dos usuarios cualesquiera **intercambiar**, de forma confidencial, una **clave secreta K (o de sesión)** para posteriormente cifrar de forma simétrica los mensajes entre ellos dos
 - Criptografía híbrida ☺
- Básicamente, DH permite que dos entidades (A y B) puedan generar una clave KAB de forma simultánea, y sin enviarla por el canal de comunicaciones



Global Public Elements

q prime number

α $\alpha < q$ and α a primitive root of q

User A Key Generation

Select private X_A $X_A < q$

Calculate public Y_A $Y_A = \alpha^{X_A} \text{ mod } q$

User B Key Generation

Select private X_B $X_B < q$

Calculate public Y_B $Y_B = \alpha^{X_B} \text{ mod } q$

Calculation of Secret Key by User A

$$K = (Y_B)^{X_A} \text{ mod } q$$

Calculation of Secret Key by User B

$$K = (Y_A)^{X_B} \text{ mod } q$$

1) Para ello, A y B necesitan establecer y **compartir valores comunes**, como un valor q primo y una raíz primitiva α de q

2) Tanto A como B generan sus claves privadas ($X_{A/B}$) y públicas ($Y_{A/B}$) teniendo en cuenta que: $Y_A \equiv \alpha^{X_A} \pmod{q}$, donde $0 \leq X_A \leq (q-1)$, X_A es el **logaritmo discreto de Y_A** , y se representa $dlog_{\alpha, q}(Y_A)$

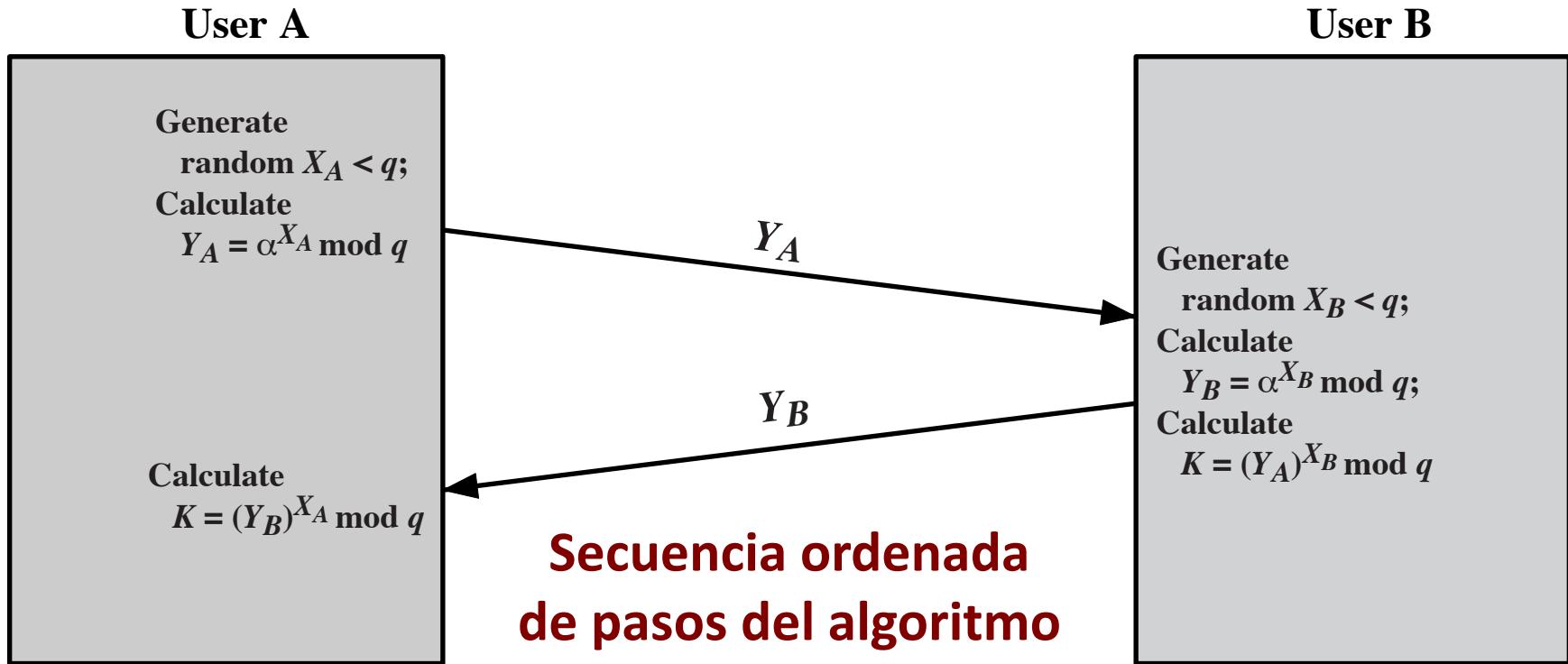


Luego, la efectividad del algoritmo depende de la **dificultad de computar logaritmos discretos**

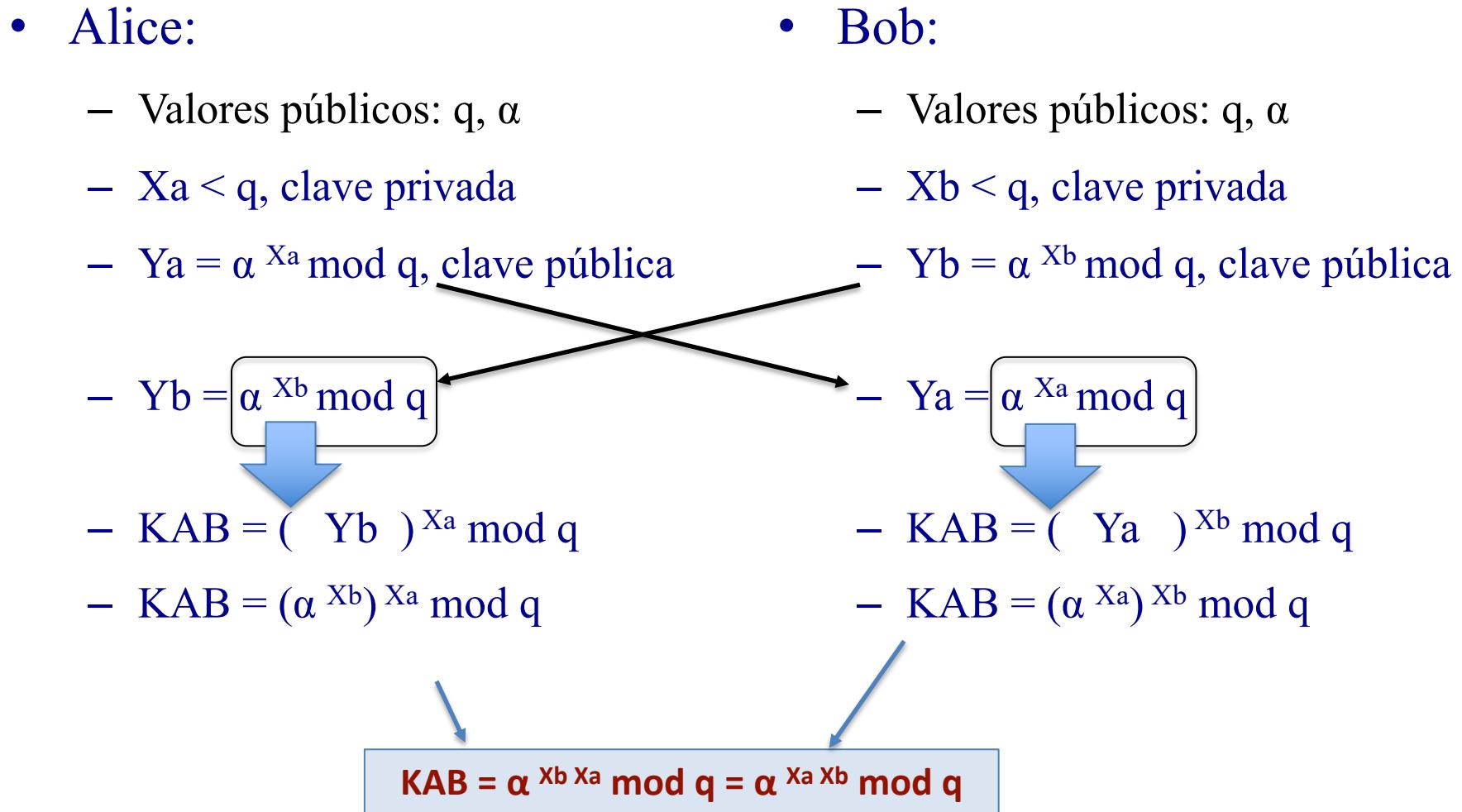
3) Tanto A como B comparten sus respectivas claves públicas (Y_A / Y_B)

4) Tanto A como B generan de forma "mágica" la clave de sesión teniendo en cuenta: **$K_{AB} = (Y_B)^{X_A} \text{ mod } q$**

Otra forma de verlo ...

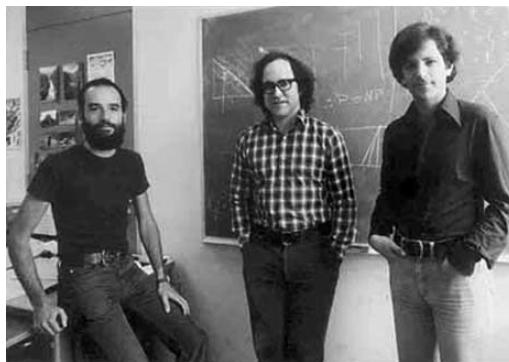


Otra forma de verlo ...



Algoritmo RSA

- Actualmente, es el criptosistema asimétrico o de clave pública más usado en la vida real
 - Su nombre procede de sus inventores: *Rivest, Shamir y Adleman* del Instituto Tecnológico de Massachusetts (MIT), y fue definido en 1977
- El criptosistema se basa de una idea bastante “simple”:
 - *“Es muy fácil multiplicar dos números enteros primos grandes, pero extremadamente difícil hallar la factorización de ese producto”*
 - Puede darse a conocer dicho producto sin que nadie descubra esos números de procedencia, a no ser que conozca al menos uno de ellos



Se piensa que RSA será seguro mientras no se conozcan “formas rápidas” de descomponer un número grande en producto de primos

- Parámetros necesarios:

- encontrar dos números primos grandes p y q , y calcular

$$n = p * q ; p \neq q$$

- se calcula $\phi(n)$, de forma que

$$\phi(n) = (p - 1) * (q - 1)$$



Para extraer los primos,
donde $\phi(n)$ se define como
el número de enteros positivos
menores o iguales a n y coprimos de n

- se elige aleatoriamente un número grande e tal que

$$\text{MCD } (\mathbf{e}, \phi(n)) = 1; e < \phi(n)$$

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

- Se determina el número d que cumple

$$e * \mathbf{d} \equiv 1 \pmod{\phi(n)}$$

(donde d debe ser el multiplicador inverso de e mod $\phi(n)$)

Calcular
las
claves

- n, d y e (y por supuesto, p y q) son la base del sistema
 - n módulo
 - d clave privada
 - e clave pública
- Las claves tienen el siguiente uso:
 - la **clave privada**: para descifrar o firmar mensajes
 - la **clave pública**: para cifrar o verificar la firma
- Para cifrar: $C = M^e \pmod{n}$
- Para descifrar: $M = C^d \pmod{n}$
- RSA se trata, por lo tanto, de un algoritmo exponencial



$$C^d = (M^e)^d \equiv M^{ed} \pmod{n}$$

- Ejemplo del cálculo de la clave pública y privada:

① Seleccionar primos: $p = 17$, $q = 11$

① Calcular $n = pq = 17 \times 11 = 187$

② Calcular $\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$

③ Seleccionar e : $\text{mcd}(e, 160) = 1$ y $e < 160$
se selecciona $e = 7$

⑤ Determinar d : $d \cdot e \equiv 1 \pmod{160}$

$$d = 23 \text{ dado que } 23 \times 7 = 161 \pmod{160} = 1$$

⑥ Clave pública = 7 y módulo = 187

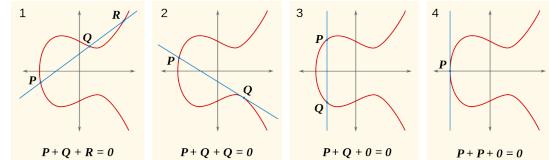
⑦ Clave privada = 23

- El texto original, M (y también el cifrado, C) debe tomarse como un **número decimal**
 - De esta forma, si se trabaja con el código ASCII:
 $M = \text{"Hola"}$ equivaldría al valor decimal 72 111 108 097
- Además, como se está trabajando en aritmética modular, todos los valores usados deben ser menores que el **módulo n**
 - Esto significa que **si el valor decimal del mensaje** es superior al módulo, $M > n$, entonces **M debe ser troceado en otros menores que n**
 - Suponiendo $n = 100000$, daría lugar a los bloques

$$m_0 = 72111, m_1 = 10809 \text{ y } m_2 = 7$$

ALGO MUY
PARECIDO A LO QUE
SE HACE CON EL
CIFRADO EN BLOQUE

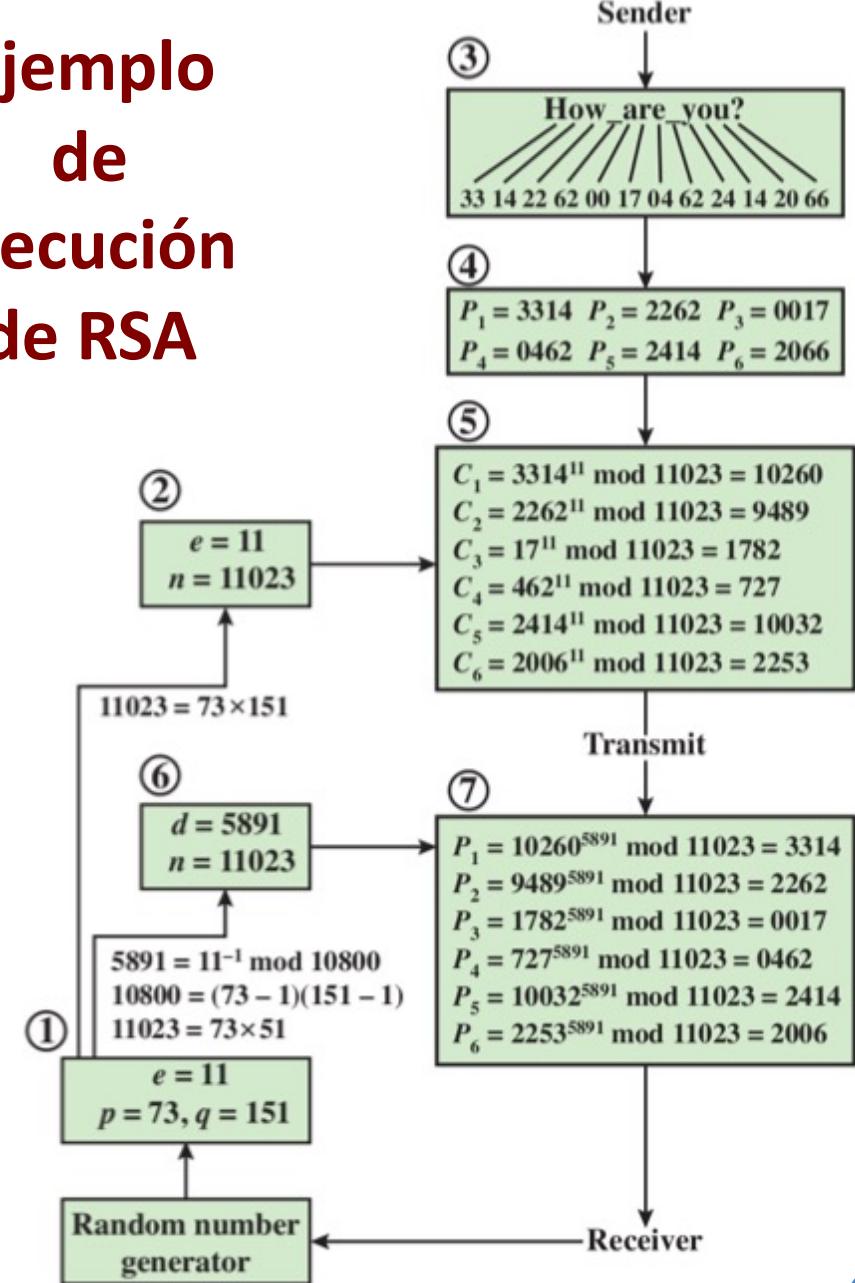
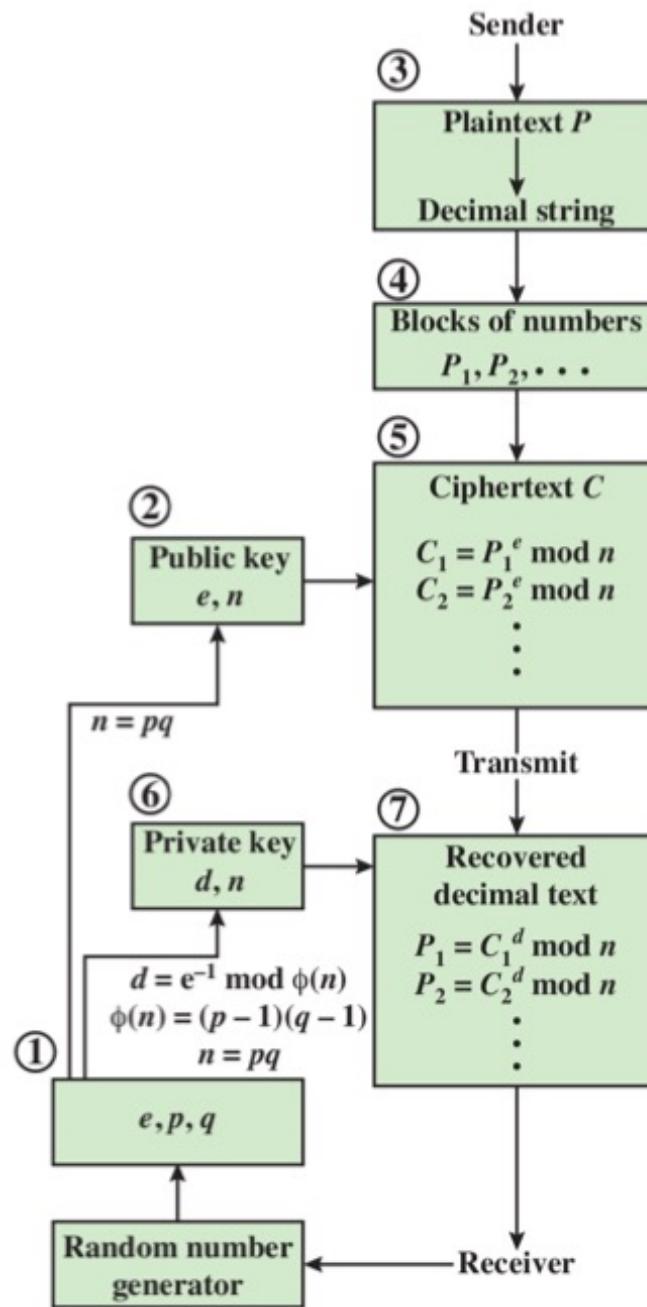
- Cosas que hay que tener en cuenta:
 - Los bloques m_i no deben/pueden ser pequeños, pues entonces el criptoanalista puede construirse una tabla donde tenga todos los posibles m_i y sus respectivos c_i
 - También hay que tener cuidado al elegir el tamaño
 - **Cuanto mayor sea, más seguro, pero también más lento** será el sistema en sus cálculos
 - Para seleccionar p ó q debe testearse si es primo o no con cualquiera de los tests existentes (se recomienda *Miller-Rabin*)
 - Se han propuesto sistemas en los que se busca un valor **e muy pequeño**, con lo cual hacer más rápido el proceso de cifrado
 - Ejemplo: ECC



Tamaño de clave simétrica (bits)	Tamaño de clave RSA y DH (bits)	Tamaño de clave de ECC (bits)
128	1024	160
192	2048	224
256	3072	256
	7680	384
	15360	521

Recomendación dada por el NIST para el tamaño de claves

Ejemplo de ejecución de RSA



Padding

- Los cifrados en bloque están diseñados para trabajar con mensajes compuestos de bloques de un tamaño específico
 - Ejemplo: AES-128 trabaja con bloques de 128 bits (16 bytes)
 - Problema: Supongamos que usamos AES-128 (16 bytes), ¿Qué ocurre cuando queremos cifrar un mensaje que ocupa, por ejemplo, 20 bytes?
 - Tendremos un primer bloque de 16 bytes, y un segundo bloque de 4 bytes
 - El primer bloque lo podemos cifrar sin problemas
 - Al segundo bloque tenemos que añadirle algo al final (“**padding**”)

Padding

- Esquemas de Padding:
 - ISO 10126: añadir bytes aleatorios, excepto el último, que indicará la longitud del padding
 - |12 63 12 65 E7 82 A7 C1|B7 02 9E 29 4E 8C 7B 05|
 - ISO/IEC 7816-4: añadir ceros, excepto el primero, que siempre tendrá el valor 80:
 - |12 63 12 65 e7 82 a7 c1|b7 02 9e 80 00 00 00 00|
 - Zero Padding: simplemente añadir ceros
 - |12 63 12 65 e7 82 a7 c1|b7 02 9e 00 00 00 00 00|
 - Problema: si el mensaje original acaba en alguna secuencia de ceros, no es posible determinar dónde empieza el padding
 - PKCS#5, PKCS#7: Si necesitamos N bytes de padding, usamos N veces el valor N
 - |12 63 12 65 e7 82 a7 c1|b7 02 9e 05 05 05 05 05|

Otras primitivas criptográficas

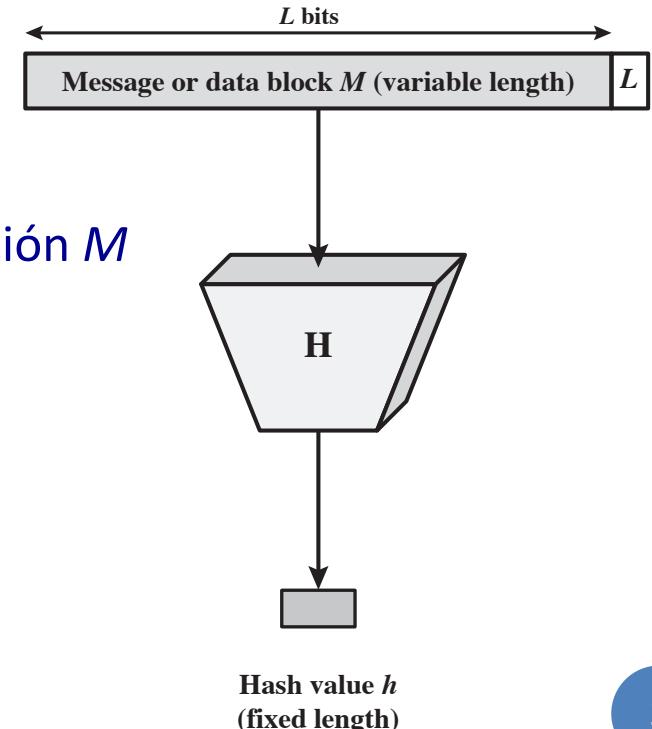


Funciones Hash

- Una función unidireccional (o de sentido único) es un bloque de construcción para muchos protocolos criptográficos
 - Son fáciles de computar, pero muy difíciles de invertir
 - Por lo tanto, no son útiles para el cifrado
 - porque un mensaje cifrado mediante una función unidireccional no se podría descifrar

- Una función hash es una función que:

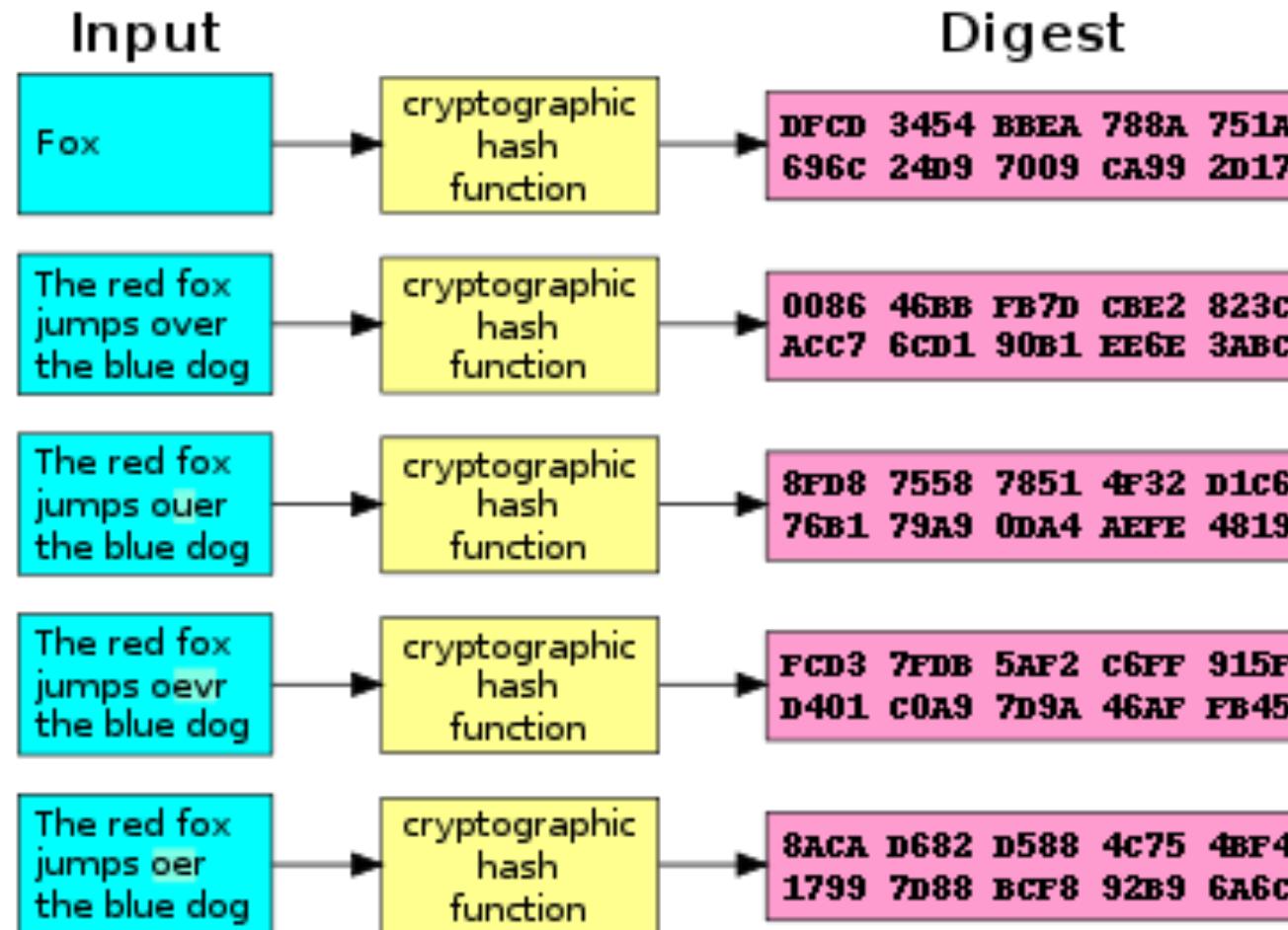
- acepta como entrada un bloque de información M (preimagen) de longitud variable
- produce un bloque de salida h (**valor hash**) de longitud fija
 - Ese valor se denomina huella digital de M



- Una función hash criptográfica es un algoritmo con el que es fácil computar el valor hash a partir de una preimagen, pero para el que resulta computacionalmente imposible:
 1. encontrar la preimagen M para un valor hash h predeterminado (propiedad: **unidireccionalidad**), y
 2. encontrar dos bloques M y M' que produzcan el mismo valor hash h (propiedad: **libre de colisiones**), tal que $h = h'$
- De lo anterior, se puede determinar que dos entradas M (ej. “*hola*”) y M' (ej. “*hora*”) a una función hash criptográfica, producen dos salidas h y h' completamente diferentes
 - De hecho, se asegura que un cambio en un único bit de la preimagen M da lugar a *un cambio de, como media, la mitad de los bits de salida tal que $h \neq h'$*



- Efecto de una función hash (caso de SHA-1):



Ahora lo pruebas tú ...

- Accede a <https://cryptii.com> y prueba con varios textos de diferentes tamaños

The screenshot shows two separate instances of the cryptii.com interface side-by-side.

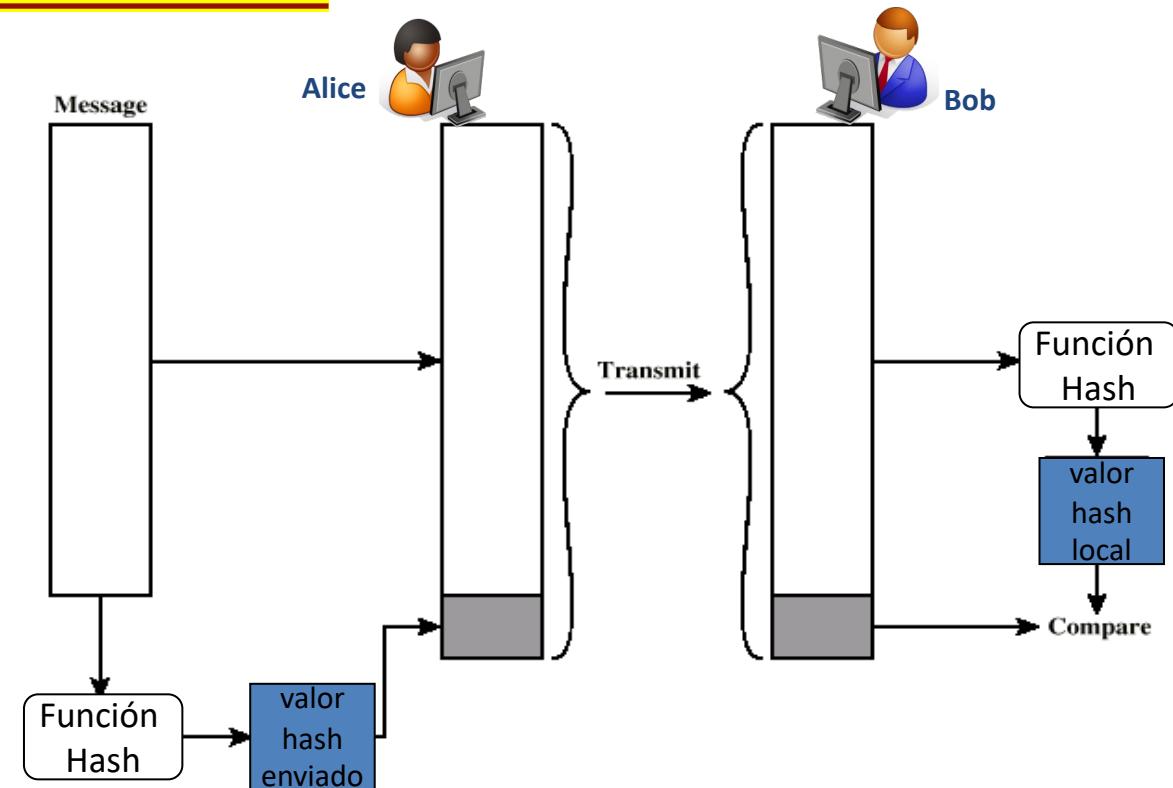
Top Instance:

- Text:** Hola, NO te toca a ti !
- Algorithm:** SHA-256 (selected)
- Output:** Hexadecimal bytes: 1b 29 b9 aa ce 7b ac 2b f1 dc 77 7a 75 ce b5 f5 44 e3 bb 73 a7 b2 8c b4 bd 66 75 7d c1 9b 82 61

Bottom Instance:

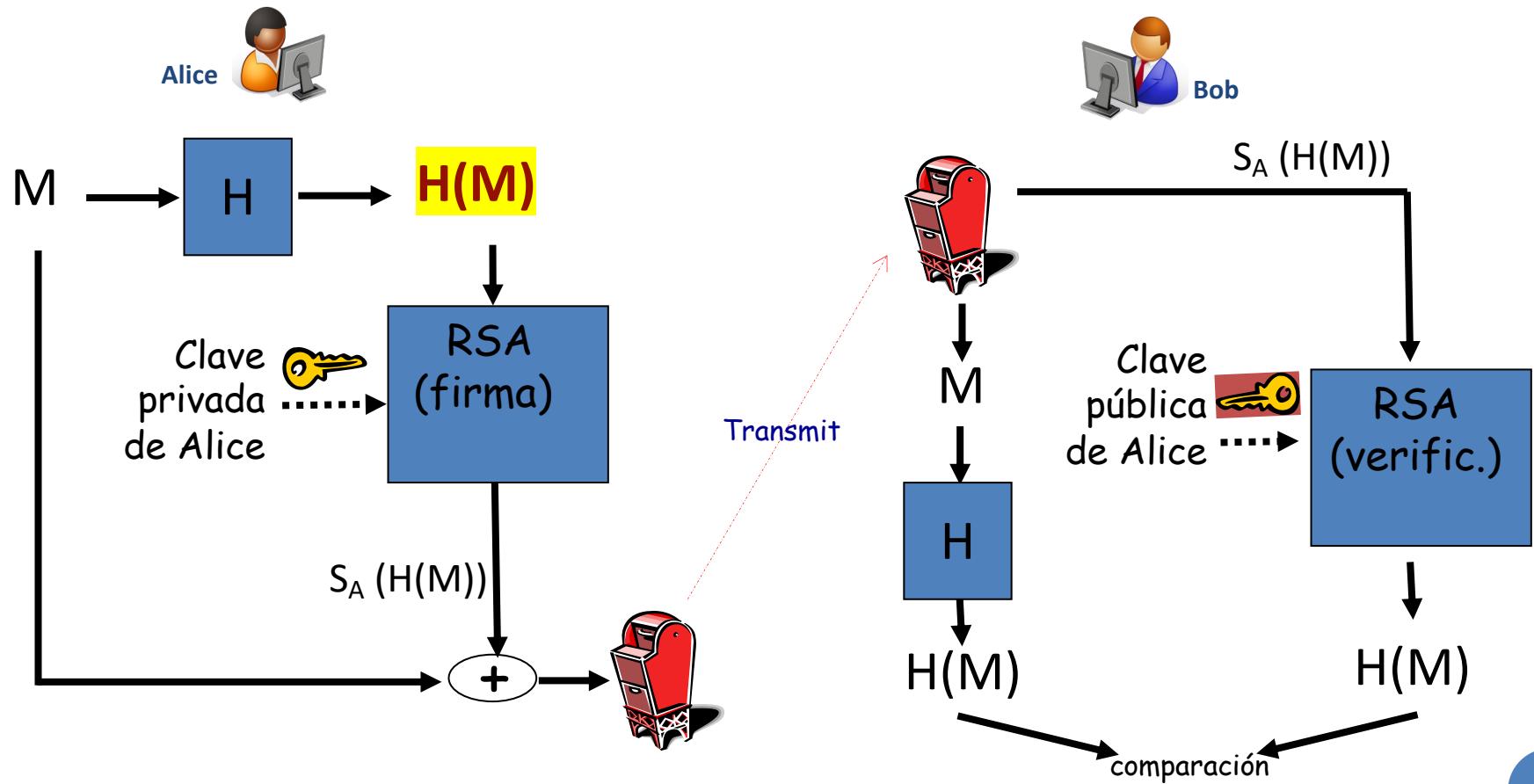
- Text:** Hola, te toca a ti !
- Algorithm:** SHA-256 (selected)
- Output:** Hexadecimal bytes: 37 d0 ae bd eb 78 ab 7a 90 1b b2 9a ea aa b3 a5 1f b6 3c 33 bb 02 90 15 71 bd 8d 2b ce c0 48 f5

- La utilidad más inmediata de una función hash es proporcionar el servicio de **integridad de datos**
- Ejemplo de uso:

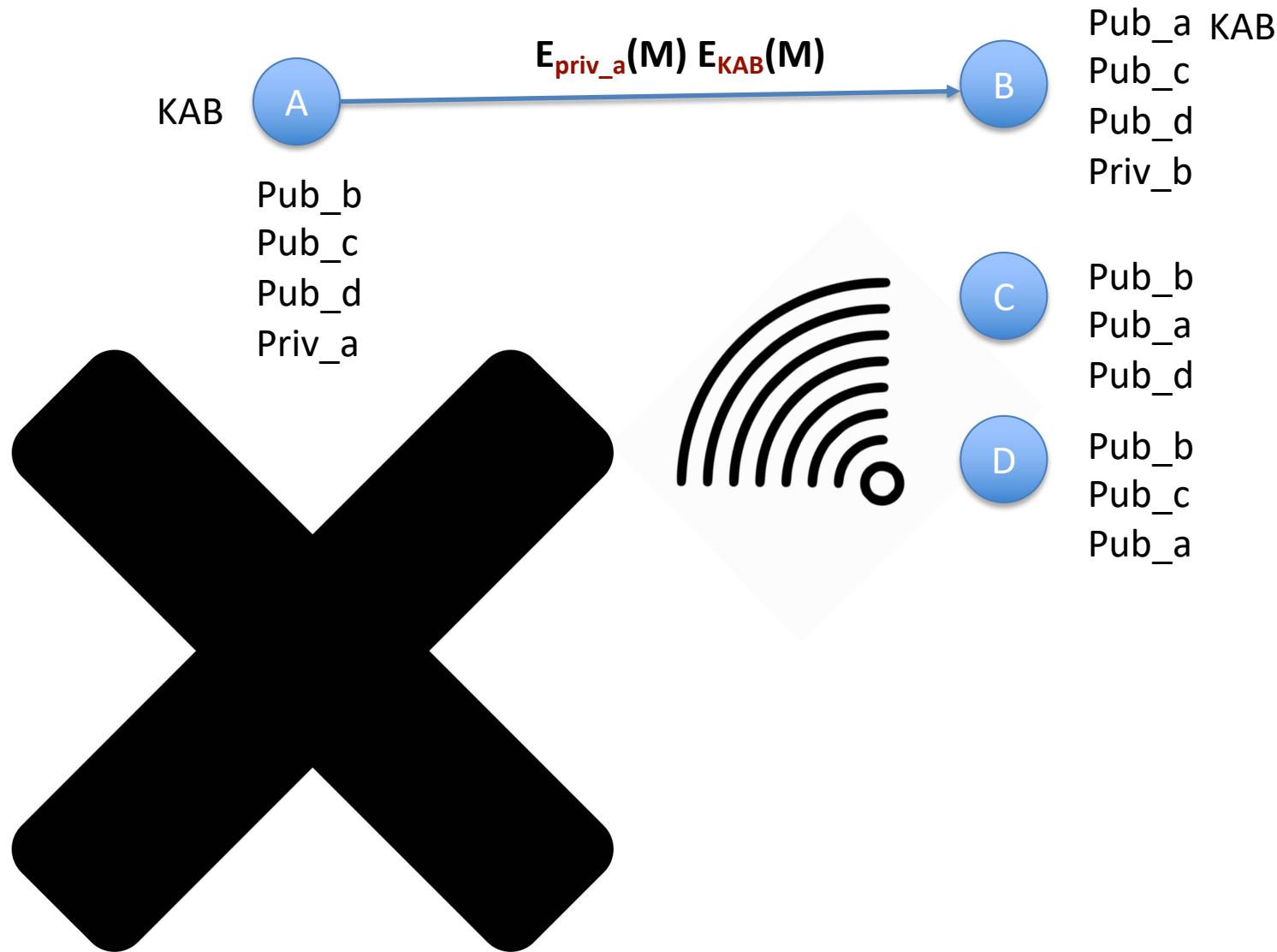


- Sin embargo, se observa que ***no hay autenticidad***
 - *Bob* no tiene garantías de que *Alice* sea el origen

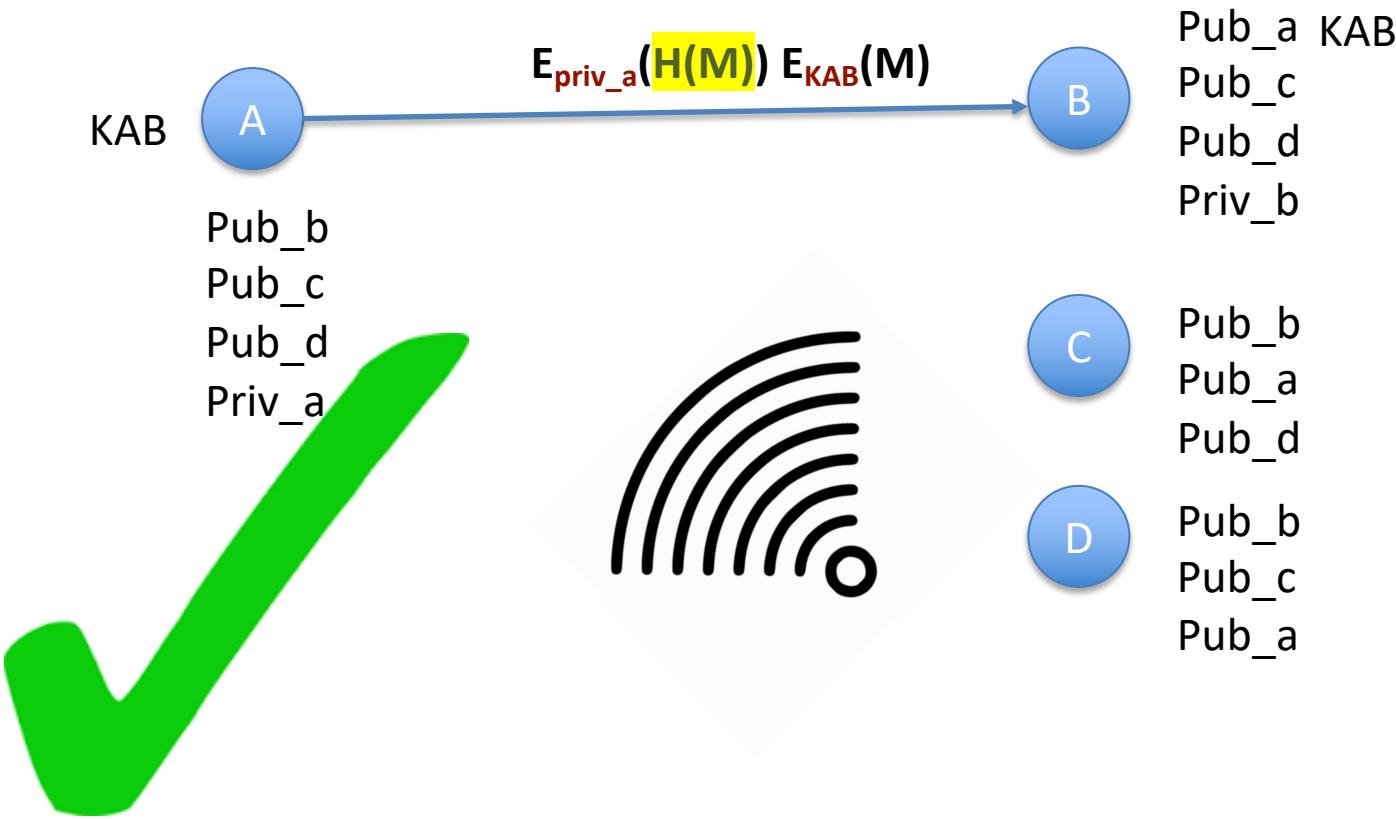
- El uso combinado de las funciones hash con la criptografía de clave pública mejora sustancialmente la eficiencia de uso de esta última (sobre todo en el procedimiento específico de firma digital)



Inciso



Inciso

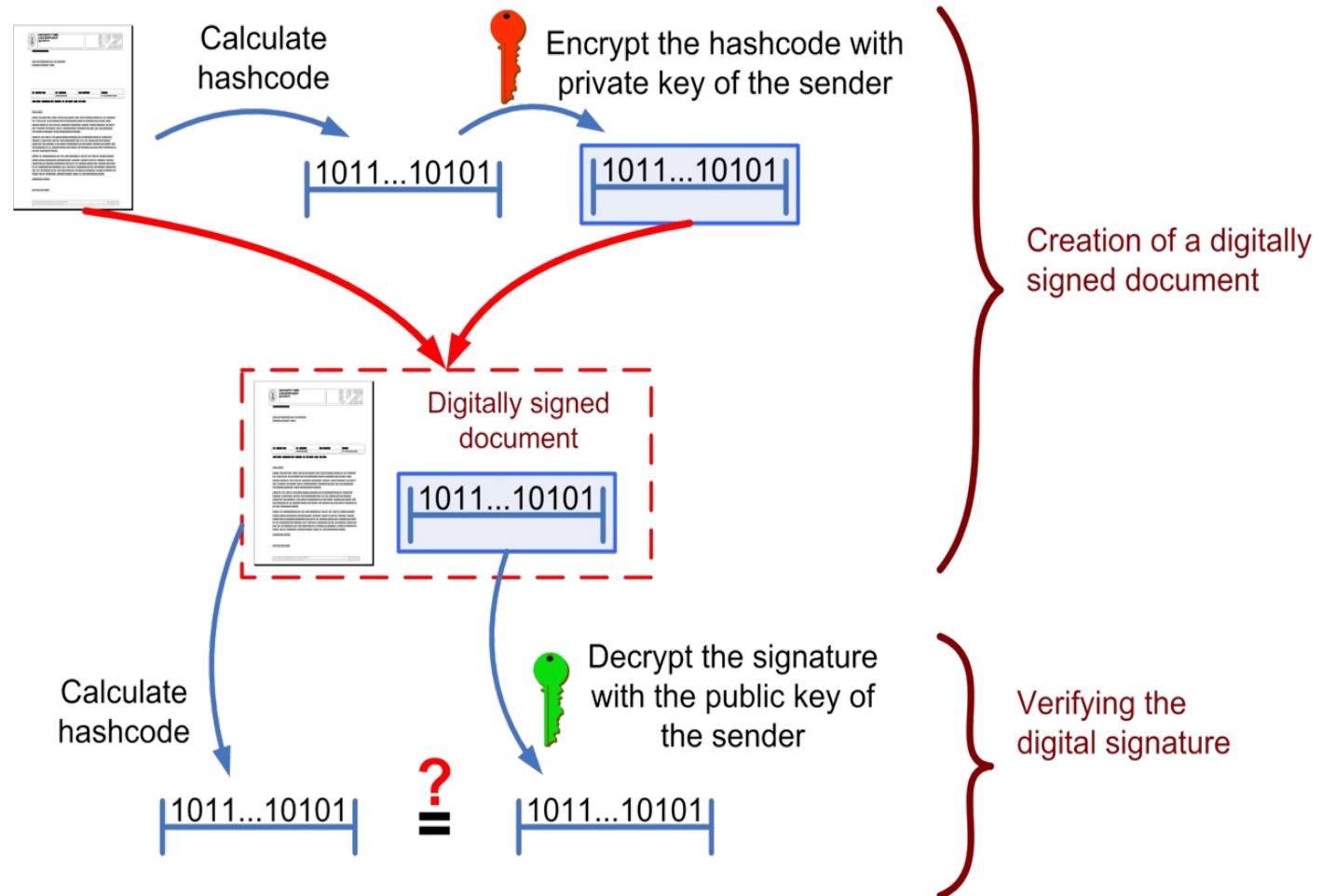


NO SE DEBE USAR C. ASIMÉTRICA PARA CIFRAR MENSAJES.
REALMENTE SE APLICA EN ESTOS CASOS:

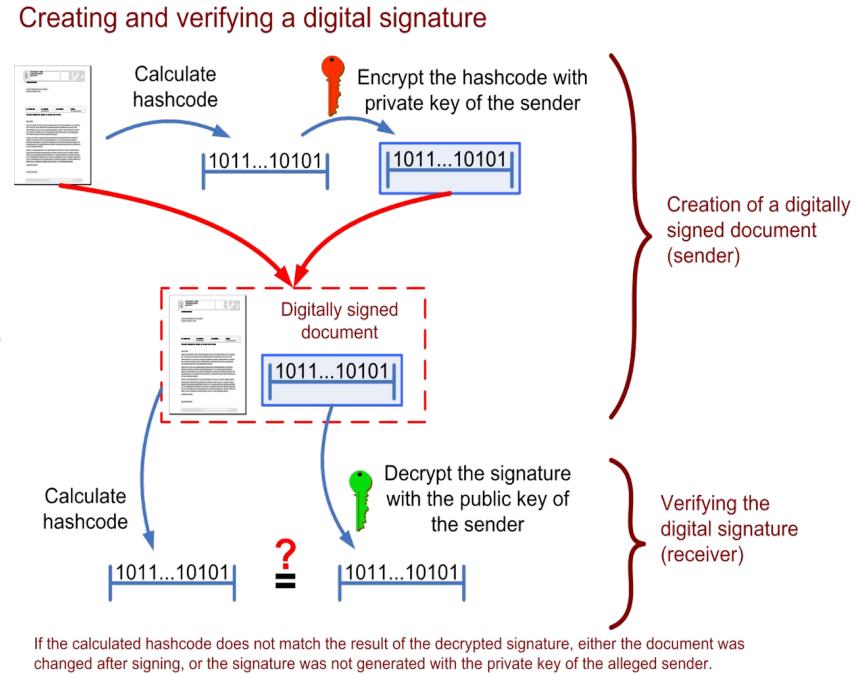
- CRIPT. HÍBRIDA → PARA PROTEGER LA CLAVE DE SESIÓN
- FIRMA DIGITAL → PERO APLICADO A EL HASH DEL MENSAJE

Debido al TAMAÑO
de los mensajes

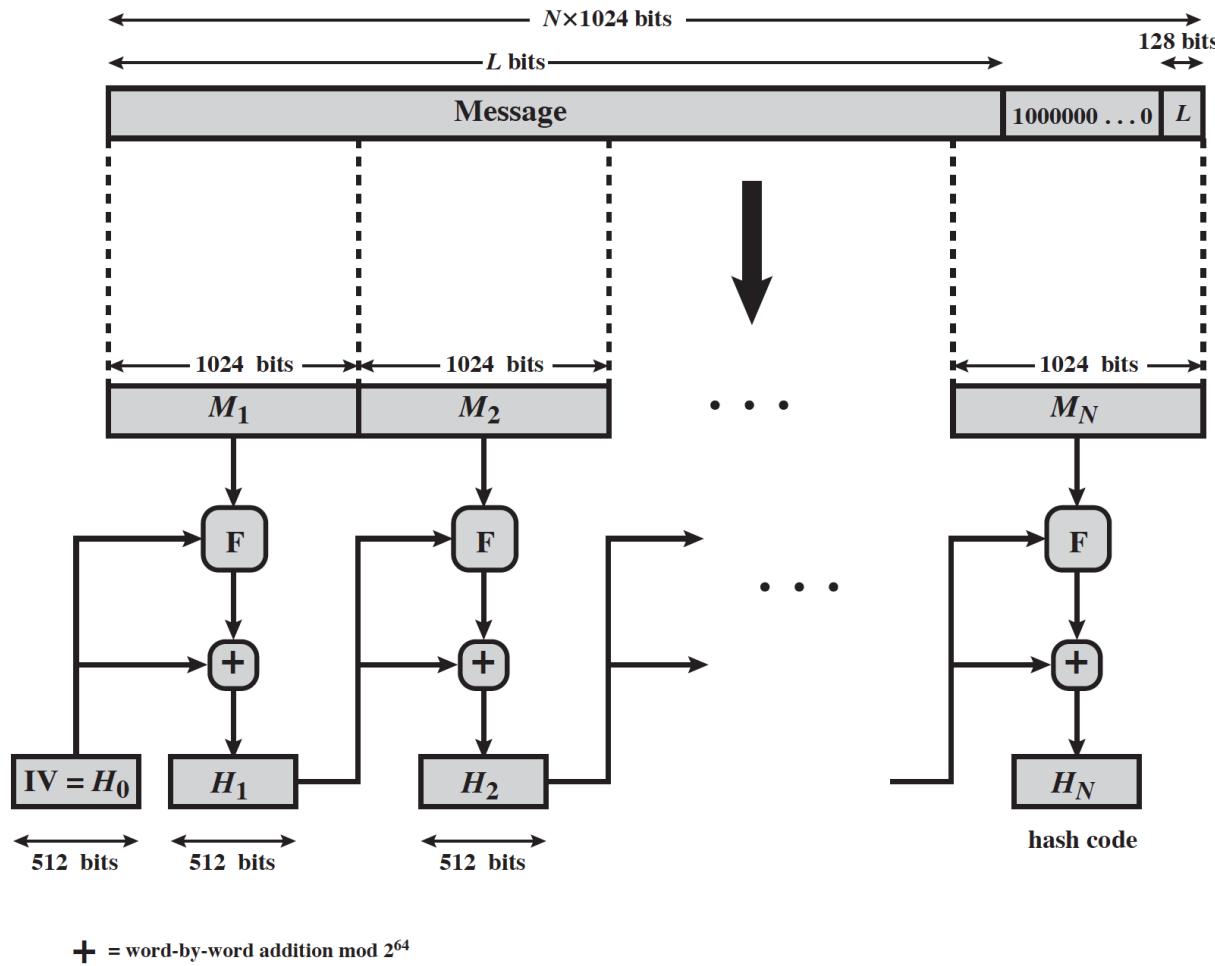
Creating and verifying a digital signature



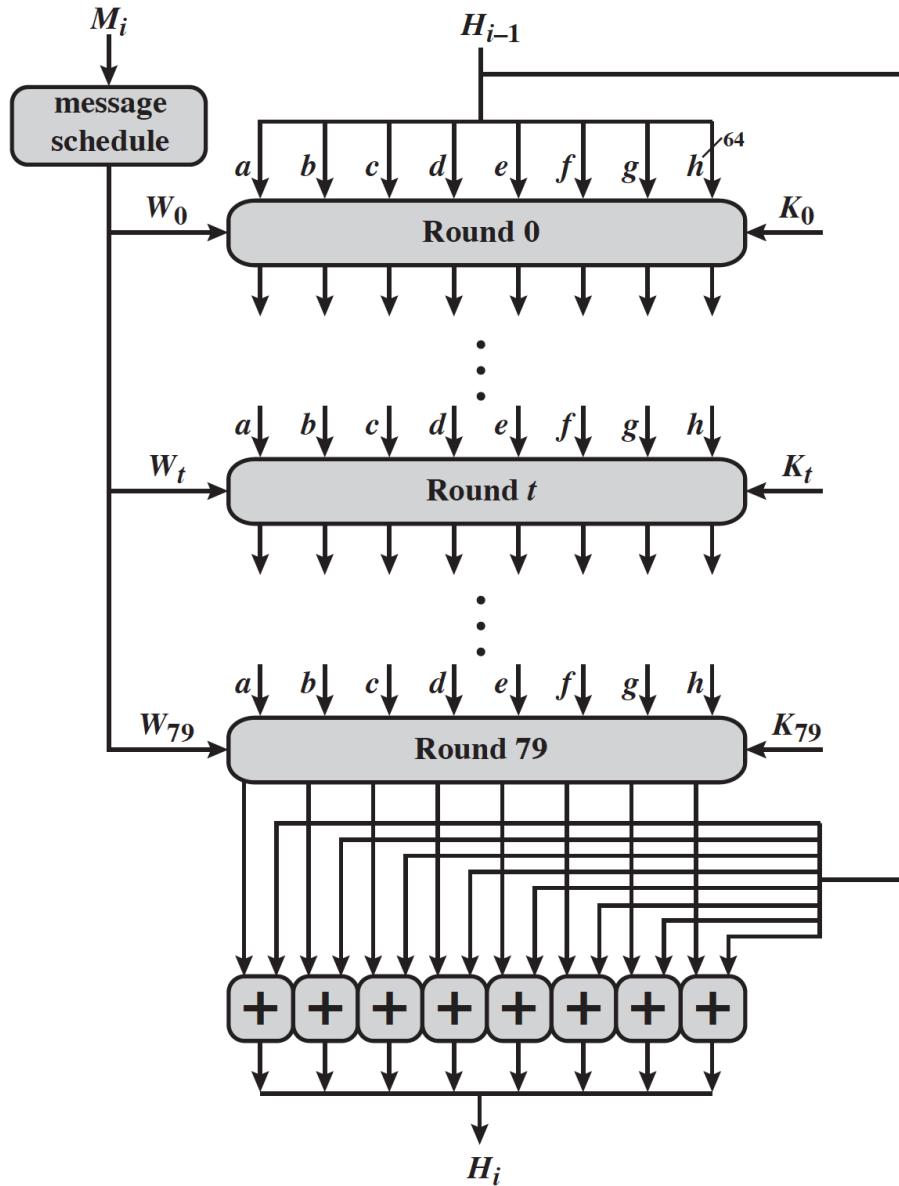
- Los beneficios del esquema:
 - la firma se puede mantener separada del documento
 - los requisitos de almacenamiento y firma son mucho menores
- Un sistema de archivos puede usar este tipo de esquema para verificar la existencia de documentos sin tener que almacenar sus contenidos



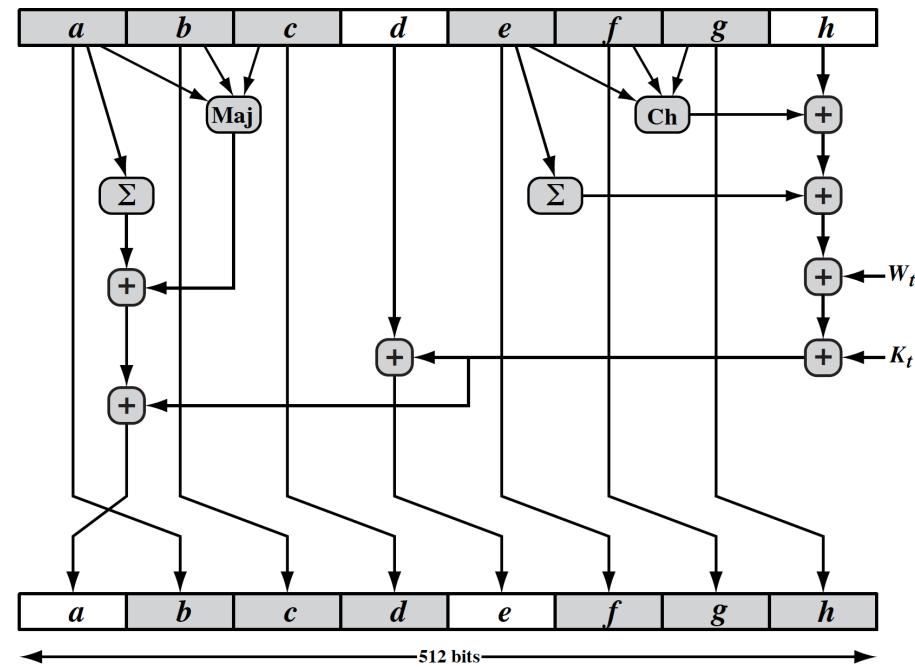
- Ejemplo de función hash: *SHA-512 (SHA-2)*



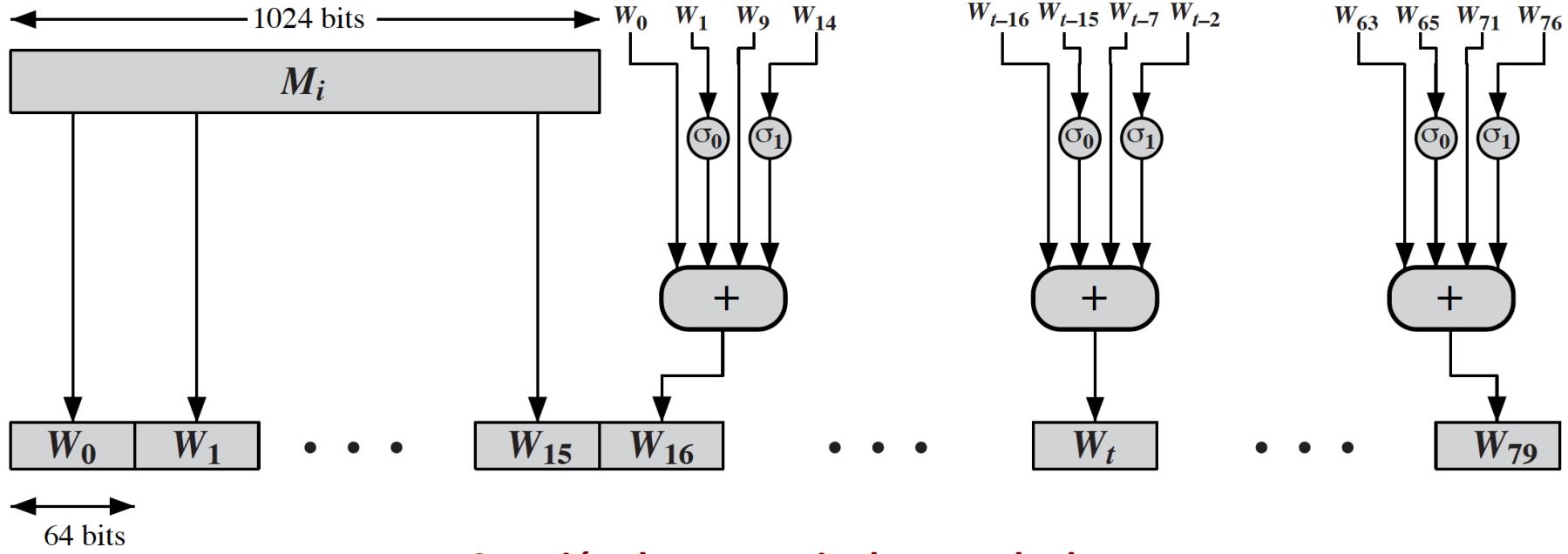
**Esquema
general del
algoritmo
SHA-512**



Procesamiento de un
bloque de 1024 bits
en SHA-2



Operación elemental (una etapa) en SHA-2



**Creación de secuencia de entrada de
 $W_0..W_{79}$ para procesamiento del bloque
 M_i en SHA-2**

428a2f98d728ae22	7137449123ef65cd	b5c0fbcfec4d3b2f	e9b5dba58189dbbc
3956c25bf348b538	59f111f1b605d019	923f82a4af194f9b	ab1c5ed5da6d8118
d807aa98a3030242	12835b0145706fbe	243185be4ee4b28c	550c7dc3d5ffb4e2
72be5d74f27b896f	80deb1fe3b1696b1	9bdc06a725c71235	c19bf174cf692694
e49b69c19ef14ad2	efbe4786384f25e3	0fc19dc68b8cd5b5	240calcc77ac9c65
2de92c6f592b0275	4a7484aa6ea6e483	5cb0a9dcbd41fdb4	76f988da831153b5
983e5152ee66dfab	a831c66d2db43210	b00327c898fb213f	bf597fc7beef0ee4
c6e00bf33da88fc2	d5a79147930aa725	06ca6351e003826f	142929670a0e6e70
27b70a8546d22ffc	2e1b21385c26c926	4d2c6dfc5ac42aed	53380d139d95b3df
650a73548baf63de	766a0abb3c77b2a8	81c2c92e47edaee6	92722c851482353b
a2bfe8a14cf10364	a81a664bbc423001	c24b8b70d0f89791	c76c51a30654be30
d192e819d6ef5218	d69906245565a910	f40e35855771202a	106aa07032bbd1b8
19a4c116b8d2d0c8	1e376c085141ab53	2748774cdf8eeb99	34b0bcb5e19b48a8
391c0cb3c5c95a63	4ed8aa4ae3418acb	5b9cca4f7763e373	682e6ff3d6b2b8a3
748f82ee5defb2fc	78a5636f43172f60	84c87814a1f0ab72	8cc702081a6439ec
90beffa23631e28	a4506cebde82bde9	bef9a3f7b2c67915	c67178f2e372532b
ca273eceeaa26619c	d186b8c721c0c207	eada7dd6cde0eb1e	f57d4f7fee6ed178
06f067aa72176fba	0a637dc5a2c898a6	113f9804bef90dae	1b710b35131c471b
28db77f523047d84	32caab7b40c72493	3c9ebe0a15c9bebc	431d67c49c100d4c
4cc5d4becb3e42b6	597f299cf657e2a	5fc6fab3ad6faec	6c44198c4a475817

Constantes $K_0..K_{79}$ en SHA-2

Funciones hash

- Estas son algunas de las funciones hash más conocidas:
 - MD-5
 - A pesar de estar muy extendida, **MD-5 no se debería considerar segura**, porque se puede atacar criptoanalíticamente, encontrando colisiones en un margen de varios segundos
 - RIPEMD-128
 - **No se puede considerar segura** en la actualidad con independencia de los **ataques criptoanalíticos** que puedan reducir la complejidad global
 - SHA-1
 - Se utiliza extensivamente pero **ya no se considera segura**
 - El criptoanálisis de **MD-5** se ha aplicado también a **SHA-1** (dado que son de la misma familia) y se han encontrado colisiones no sólo del punto de vista teórico sino también práctico

SHA-1

Criptoanalistas 'rompen' SHA1

theregister.co.uk 17-Feb-2005

 Twittear



Share

El rumor que se ha estado corriendo, es ahora oficial, el algoritmo popular SHA-1 ha estado siendo atacado exitosamente por investigadores en China y Estados Unidos. Se ha descubierto una "colisión" en la versión completa en 2^{69} operaciones de hash, haciendo posible intentar un ataque de fuerza bruta exitoso con las computadoras más potentes de la actualidad.

Esto no significa desastre en términos prácticos, ya que la cantidad de poder computacional y conocimiento matemático necesario para poder llevar a cabo un ataque exitoso es elevado. Pero se ha demostrado que el algoritmo SHA-1 no está más allá del alcance de las supercomputadoras, como se había creído o por lo menos esperado. Teóricamente, se requeriría aproximadamente 2^{80} operaciones para poder encontrar una colisión.

Usando versiones de rendondeo-reducido del algoritmo, y la técnica del equipo, fue posible atacar el SHA-1 con menos de 2^{33} operaciones. Usando la misma técnica, el algoritmo completo SHA-0 pudo ser atacado en 2^{39} operaciones.

SHA-1 se había presumido ser más seguro que MD5, en donde las colisiones fueron encontradas el último año por algunas personas que reportaron el descubrimiento reciente. Además, en el último año, se han encontrado colisiones en SHA-0 por un equipo francés.

Security

'First ever' SHA-1 hash collision calculated. All it took were five clever brains... and 6,610 years of processor time

Tired old algo underpinning online security must die now



23 Feb 2017 at 18:33, John Leyden, Thomas Claburn and Chris Williams

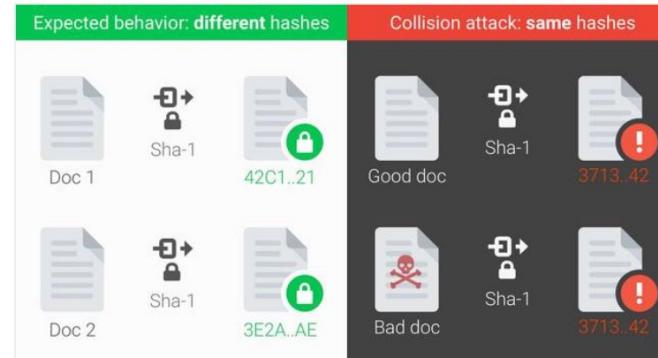


Google researchers and academics have today demonstrated it is possible – following years of number crunching – to produce two different documents that have the same SHA-1 hash signature.

This proves what we've long suspected: that SHA-1 is weak and can't be trusted. This is bad news because the SHA-1 hashing algorithm is used across the internet, from Git repositories to file deduplication systems to HTTPS certificates used to protect online banking and other websites. SHA-1 signatures are used to prove that blobs of data – which could be software source code, emails, PDFs, website certificates, etc – have not been tampered with by miscreants, or altered in any other way.

Now researchers at CWI Amsterdam and bods at Google have managed to alter a PDF without changing its SHA-1 hash value. That makes it a lot easier to pass off the meddled-with version as the legit copy. You could alter the contents of, say, a contract, and make its hash match that of the original. Now you can trick someone into thinking the tampered copy is the original. The hashes are completely the same.

Specifically, the team has successfully crafted what they say is a practical technique to generate a SHA-1 hash collision. As a hash function, SHA-1 takes a block of information and produces a short 40-character summary. It's this summary that is compared from file to file to see if anything has changed. If any part of the data is altered, the hash value should be different. Now, in the wake of the research revealed today, security mechanisms and defenses still relying on the algorithm have been effectively kneecapped.



Google's illustration how changes made to a file can sneak under the radar by not changing the hash value

The gang spent two years developing the technique. It took 9,223,372,036,854,775,808 SHA-1 computations, 6,500 years of CPU time, and 110 years of GPU time, to get to this point. The team is made up of Marc Stevens (CWI Amsterdam), Elie Bursztein (Google), Pierre Karpman (CWI Amsterdam), Ange Albertini (Google), and Yarik Markov (Google), and their paper on their work can be found here [PDF]. Its title is: "The first collision for full SHA-1."

For all the gory details, and the tech specs of the Intel CPU and Nvidia GPU number-crunchers used, you should check out the team's research paper. On a basic level, the collision-finding technique involves breaking the data down into small chunks so that changes, or disturbances, in one set of chunks is countered by twiddling bits in other chunks. A disturbance vector [PDF] is used to find and flip the right bits.

A description of Google's SHA-1 colliding PDFs can be found here. We note that the files essentially each contain a large JPEG, and the hash collision is focused on that image data. We also note that you don't have to burn another few thousand years of CPU and GPU time to create more SHA-1 collisions for simple files: thanks to Google's computations, and quirks of the PDF file format, you can from here on out produce PDFs that are visually different but still have the same SHA-1 hash value. This online tool that popped up today will easily help you create colliding PDF files.

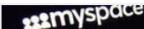
In other words, it is now trivial for anyone to alter PDFs, webpages, and certain other simple documents, and keep the SHA-1 hash values the same, thanks to Google and co's research.



Sunny View School
The Costa del Sol's Leading Private British School >



Free IT Service Desk Tool
Handle your IT Support better. Start with a Free Service Desk!
freshservice.com/free->

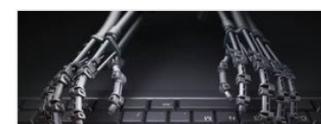


myspace
Search DISCOVER Featured 

Speaking in Tech: A chat with Web 2.0 MySpace worm dude Samy Kamkar



The Register's guide to protecting your data when visiting the US



SHA1 collider

Quick-and-dirty PDF maker using the collision from the SHAttered paper.

Choose two image files (must be JPG, roughly the same aspect ratio). For now, each file must be less than 64kB.

Aspect ratio 512 x 512

Seleccionar archivo nada seleccionado

Seleccionar archivo nada seleccionado

Enviar

(this one is actually even simpler -- just a single comment with the entire first file in it, hence the 64k limit)

Complaints to [@steike](#).

<https://alf.nu/SHA1>

SHATTERED

We have broken SHA-1 in practice.

This industry cryptographic hash function standard is used for digital signatures and file integrity verification, and protects a wide spectrum of digital assets, including credit card transactions, electronic documents, open-source software repositories and software updates.

It is now practically possible to craft two colliding PDF files and obtain a SHA-1 digital signature on the first PDF file which can also be abused as a valid signature on the second PDF file.

For example, by crafting the two colliding PDF files as two rental agreements with different rent, it is possible to trick someone to create a valid signature for a high-rent contract by having him or her sign a low-rent contract.

[Infographic](#) | [Paper](#)

Attack proof

Here are two PDF files that display different content, yet have the same SHA-1 digest.

SHAttered
The first ever SHA-1 collision attack paper

SHAttered
The first ever SHA-1 collision attack paper

CWI Google

CWI Google

PDF 1 | PDF 2

File tester

Upload any file to test if they are part of a collision attack. Rest assured that we do not store uploaded files.

Drag some files here
Or, if you prefer...
Choose a file to upload

<https://shattered.io>

Hola a tod@s

Hola a tod@s



2 PDFs (a.pdf y b.pdf) con el resultado de la colisión

SHATTERED

We have broken SHA-1 in practice.

This industry cryptographic hash function standard is used for digital signatures and file integrity verification, and protects a wide spectrum of digital assets, including credit card transactions, electronic documents, open-source software repositories and software updates.

It is now practically possible to craft two colliding PDF files and obtain a SHA-1 digital signature on the first PDF file which can also be abused as a valid signature on the second PDF file.

For example, by crafting the two colliding PDF files as two rental agreements with different rent, it is possible to trick someone to create a valid signature for a high-rent contract by having him or her sign a low-rent contract.

[Infographic](#) | [Paper](#)

Attack proof

Here are two PDF files that display different content, yet have the same SHA-1 digest.

SHAttered
The first ever SHA-1 collision attack paper

SHAttered
The first ever SHA-1 collision attack paper

CWI Google

CWI Google

PDF 1 | PDF 2

File tester

Upload any file to test if they are part of a collision attack. Rest assured that we do not store uploaded files.

Collision found in a.pdf
Collision found in b.pdf
Reset

SHA-1

- Según <https://shattered.io>, se puede crear colisiones a diferentes tipos de artefactos:
 - Digital Certificate signatures
 - Email PGP/GPG signatures
 - Software vendor signatures
 - Software updates
 - ISO checksums
 - Backup systems
 - Deduplication systems
 - GIT (a version control system (VCS) for tracking changes in computer files and coordinating interactive work based on files)
 - ...

– SHA-2

- En la actualidad hay una familia de cuatro algoritmos hash:
 - *SHA-224, SHA-256, SHA-384* y *SHA-512*
- Para *SHA-224/SHA-256* (resp. *SHA-384/SHA-512*) se han encontrado algunas colisiones reducidas

– SHA-3

- La competición organizada por *NIST* para elegir el estándar *SHA-3* finalizó en Octubre de 2012, seleccionando el algoritmo *Keccak*

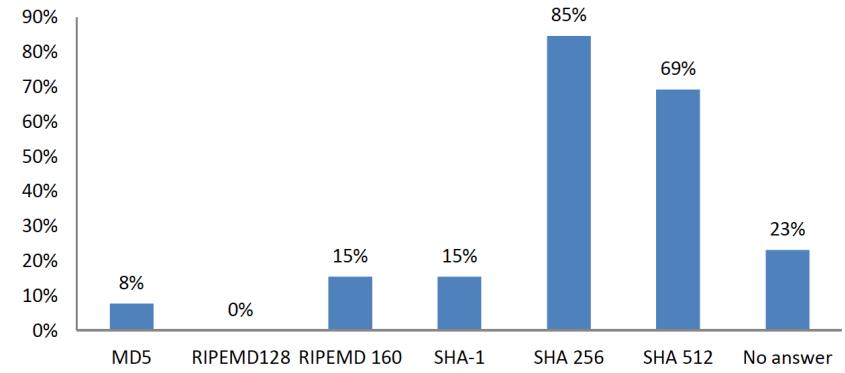
– Whirlpool

- Produce una salida de 512 bits, y no pertenece a la familia *MD-X*
- Se construye a partir de métodos similares a los usados en *AES*, y es una buena alternativa de uso para garantizar la diversidad de algoritmos

- Comentarios generales:

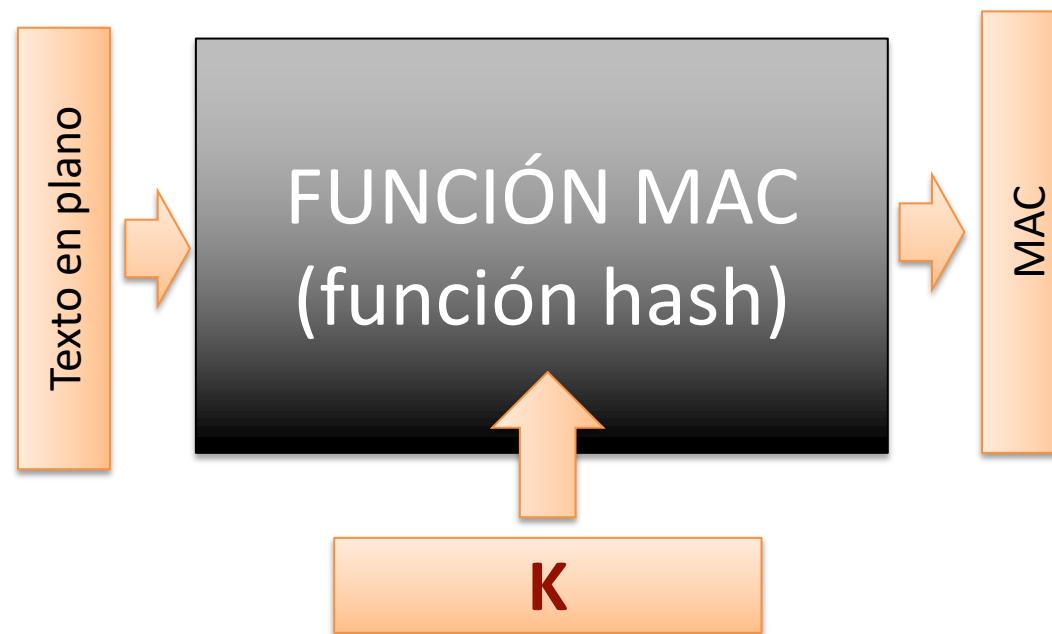
- El desarrollo de la funciones hash es, probablemente, el área de la Criptografía que ha atraído más atención durante la pasada década
- La mayoría de las funciones hash existentes derivan mucho de los criterios de diseño de la función hash *MD-4* (familia *MD-X*)
 - Esta familia incluye *MD-4*, *MD-5*, *RIPEMD-128*, *RIPEMD-160*, *SHA-1* y *SHA-2*
- **Las salidas de las funciones hash deberían ser como mínimo de 160 bits de longitud para las aplicaciones heredadas, y de 256 bits para todas las aplicaciones nuevas**

Primitive	Output Lenfth	Recommendation Legacy	Recommendation Future
SHA-2	256, 384, 512	✓	✓
SHA-3	?	✓	✓
Whirlpool	512	✓	✓
SHA-2	224	✓	✗
RIPEMD-160	160	✓	✗
SHA-1	160	✓	✗
MD-5	128	✗	✗
RIPEMD-128	128	✗	✗

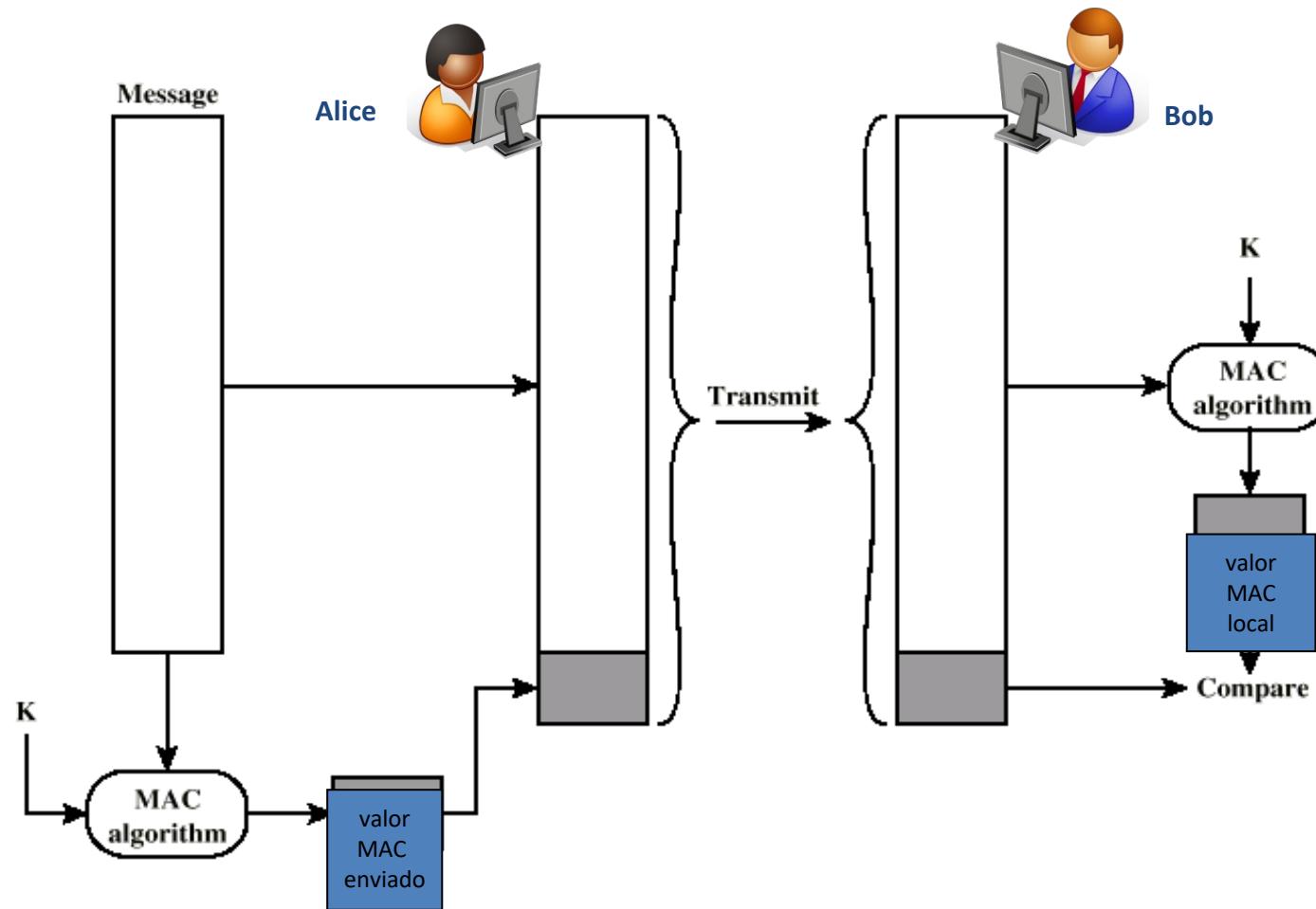


Funciones MAC

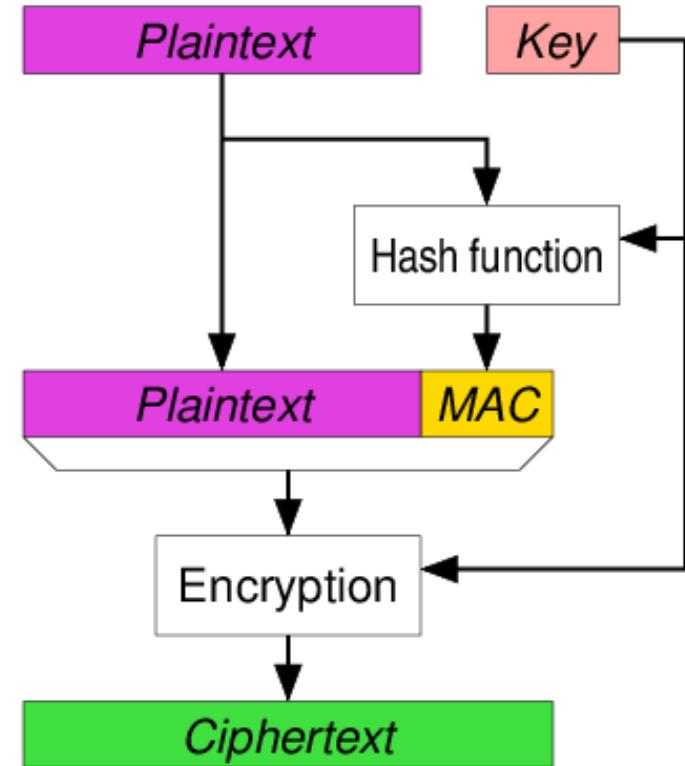
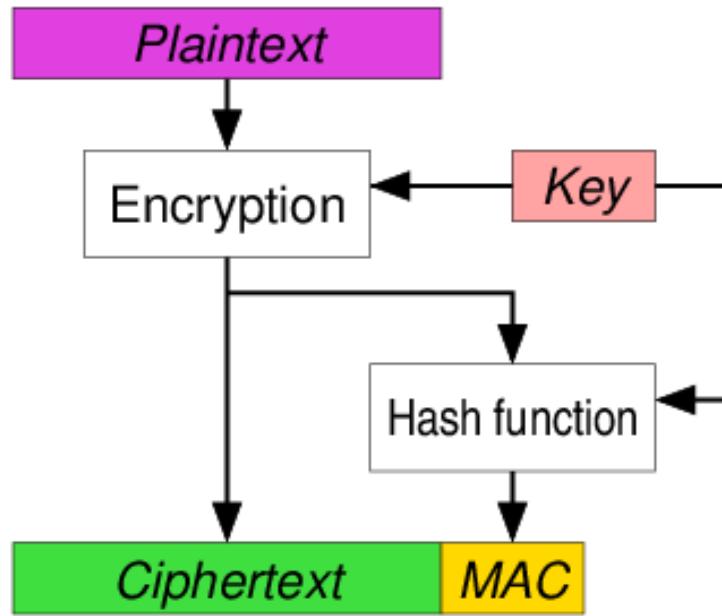
- Un código de autenticación de mensaje, o función MAC, es un concepto que evoluciona a partir del concepto de función hash
- Concretamente, una función MAC toma como entrada un mensaje M y una clave K , y produce un valor hash (que en este caso se denomina **valor MAC**)



- Ejemplo de uso:



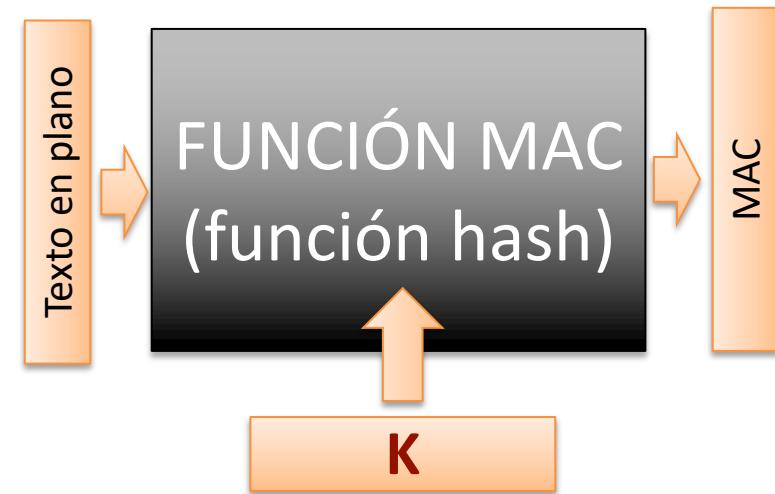
Dicho de otro modo...



- El esquema anterior soluciona el problema anteriormente mencionado de que la función hash, aunque proporciona integridad de datos, no proporcionan autenticación

- Ahora la autenticación se consigue porque *Alice* y *Bob* comparten la clave simétrica **K**
- *Bob* está convencido de que la información proviene de *Alice* porque es la única (además de él) que conoce **K**
- Luego se garantiza:

AUTENTICACIÓN



- Las funciones MAC más conocidas entran en dos categorías:
 - Basadas en funciones hash (por ejemplo: *HMAC*, *UMAC*)
 - Basadas en algoritmos de cifrado en bloque (por ejemplo: *EMAC*, *AMAC*, *CMAC*)

Ahora lo pruebas tú ...

- Accede a <https://cryptii.com> y prueba con varios textos de diferentes tamaños

The screenshot shows three examples of using the HMAC feature on the Cryptii website:

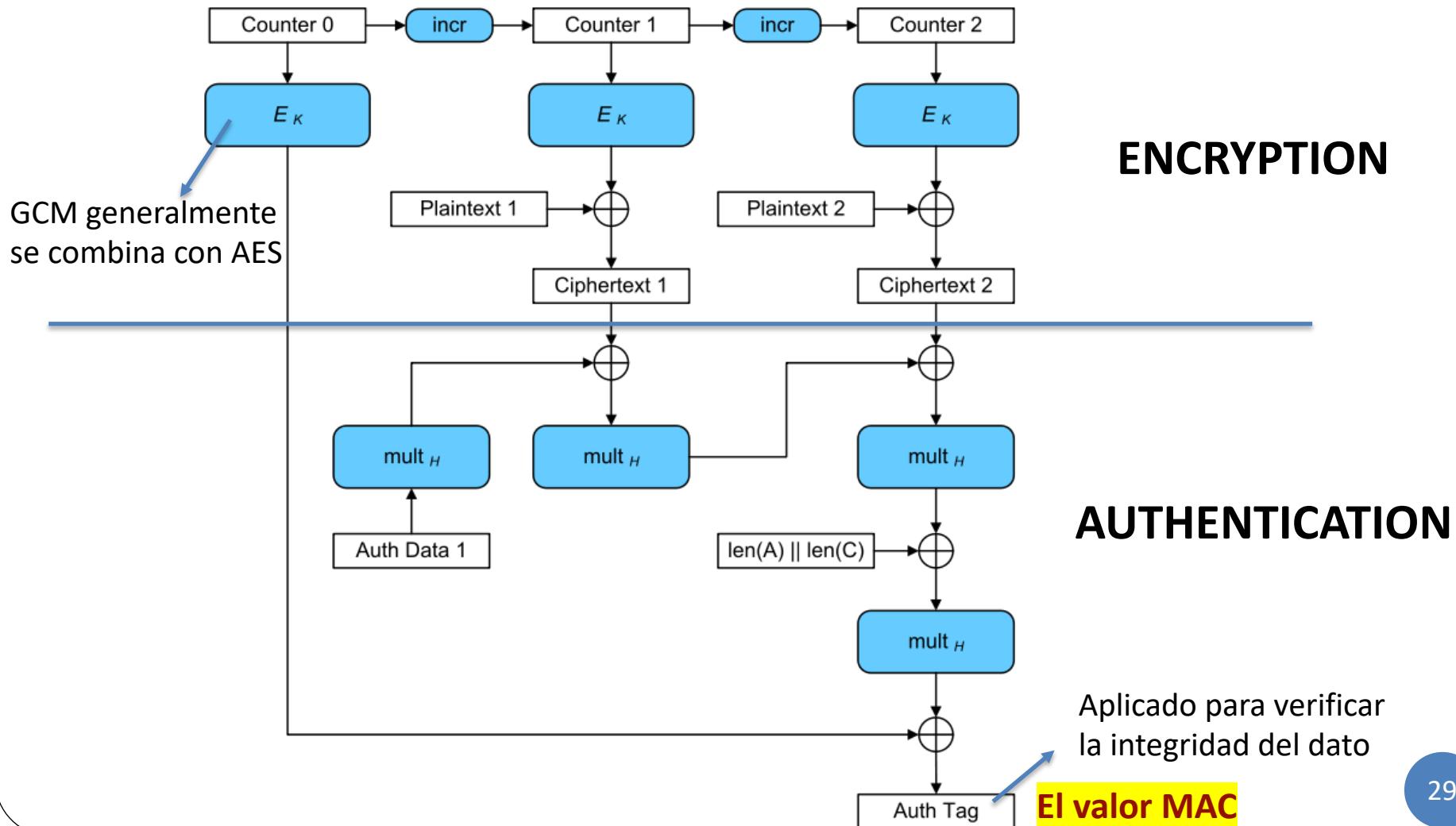
- Example 1:** Input text: "Hola, te toca a ti !". HMAC interface: Key: 62 72 79 70 74 69 69. Algorithm: SHA-256. Bytes output: 0b a4 1c 9a b6 c9 05 bf 34 ff ea 3a a8 06 62 a2 6c 27 90 e7 19 18 74 b2 ad 97 88 0e a1 6e 14 0e.
- Example 2:** Input text: "Hola, te toca a ti !". HMAC interface: Key: 63 72 79 70 74 69 69. Algorithm: SHA-256. Bytes output: 39 73 b7 da 85 62 a8 54 07 e4 3b fe 0f b7 4f fd 53 13 00 fd a2 b7 7e db 62 d3 c0 ef 79 b8 95 8e.
- Example 3:** Input text: "Hola, NO te toca a ti !". HMAC interface: Key: 62 72 79 70 74 69 69. Algorithm: SHA-256. Bytes output: 10 3a e2 e7 2d 2c 5b c6 cd 0d fe 34 77 67 f8 97 16 72 00 4e 73 96 d8 4c b7 cf b9 07 0b ee 32 96. Note: Encoded 32 bytes.

- 1) Cambia el texto
- 2) Cambia el valor de la clave

Inciso - recordatorio

• Galois-CTR (GCM)

- Funciona de forma similar que el modo CRT pero usa Carter-Wegman MAC en un campo de Galois
- Es rápido y eficiente, y está soportado por el suite de cifrado dado por TLS 1.3



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

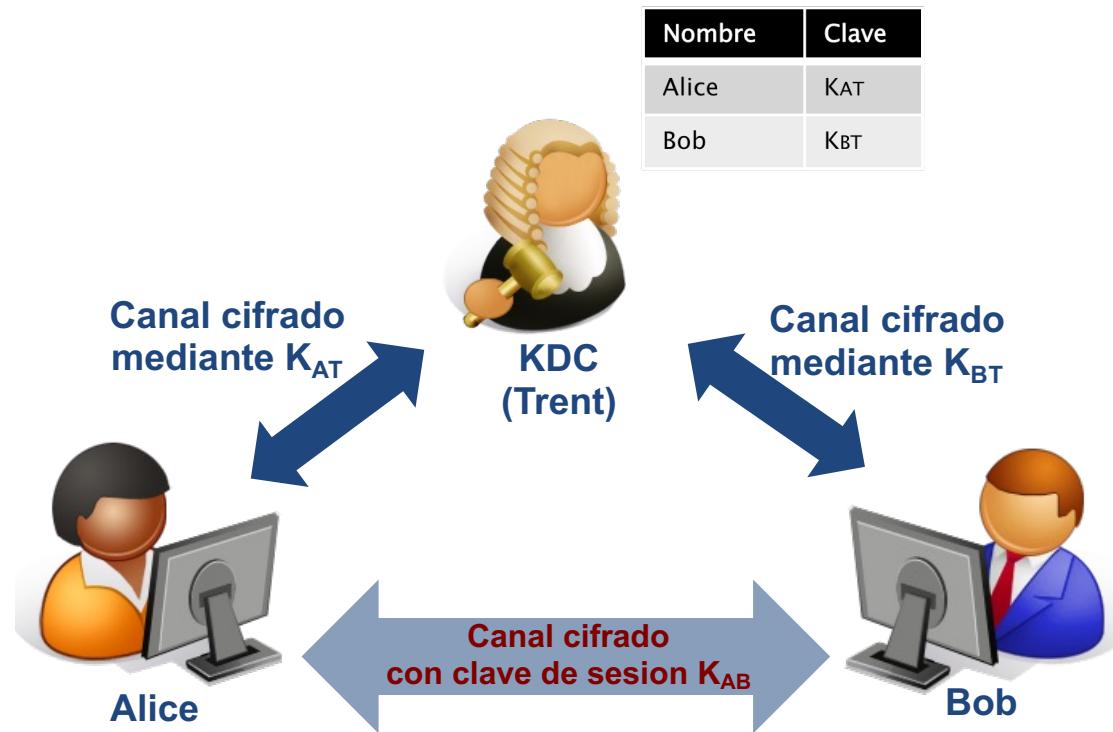
- **Gestión de las “claves”**
 - Protocolos de distribución de claves simétricas
 - Mecanismos e infraestructuras de administración de claves públicas
 - El caso del DNI-e
 - Mecanismo de Single Sign-On para Autenticación
- **Mecanismos de Control de Acceso**
 - DAC, MAC, RBAC, ABAC
 - Otros
- **Protocolos 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

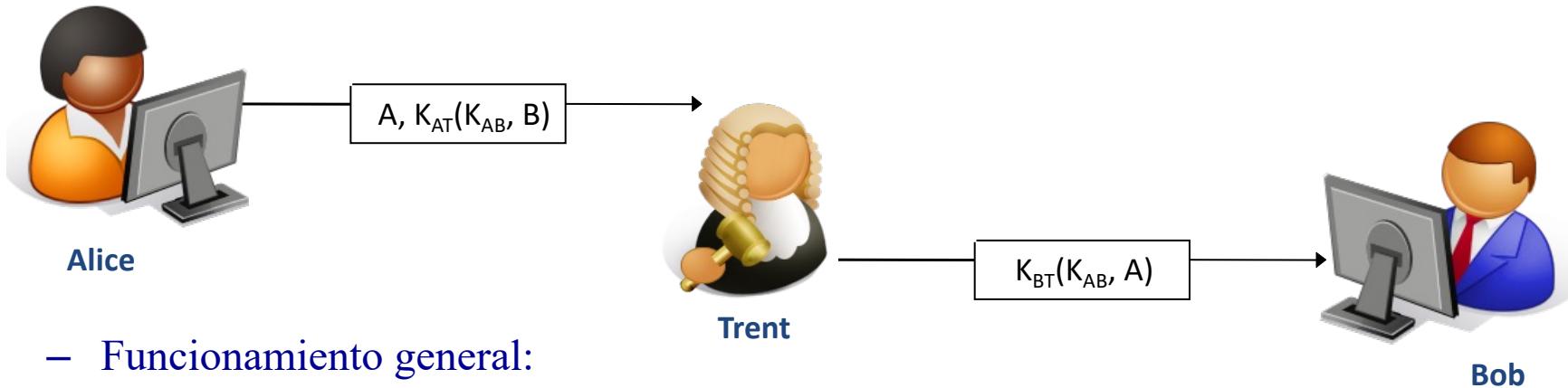
Canal cifrado
mediante K_{AT}, K_{BT}



Canal cifrado
mediante K_{AB}

KDC: modelos y protocolos - Modelo Simple

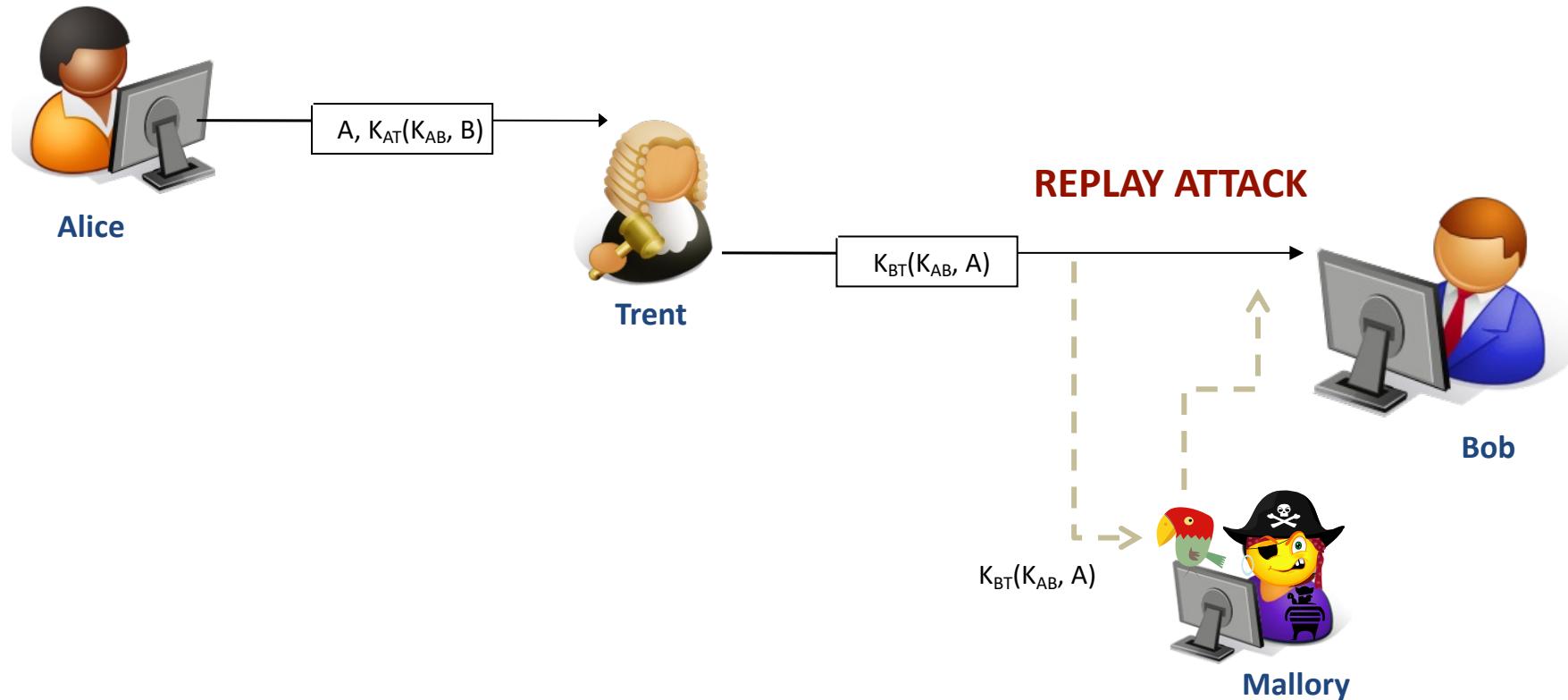
- El protocolo “**La Rana de la Boca Grande**” es un ejemplo de modelo simple para la distribución de claves:



- 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}

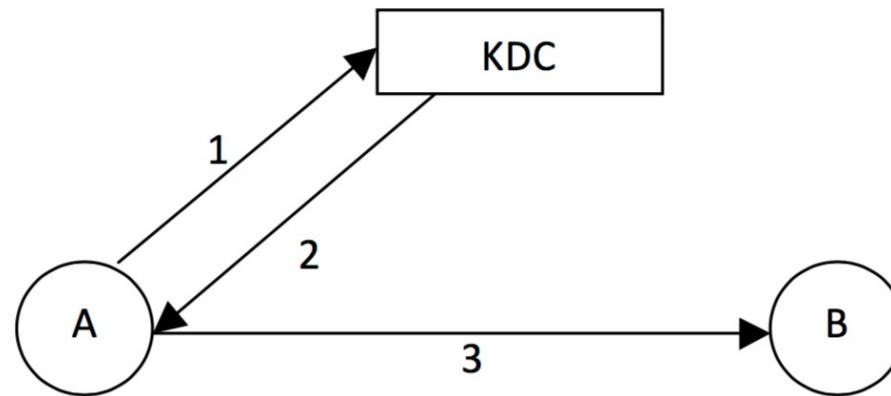
KDC: modelos y protocolos - Modelo Simple

- 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

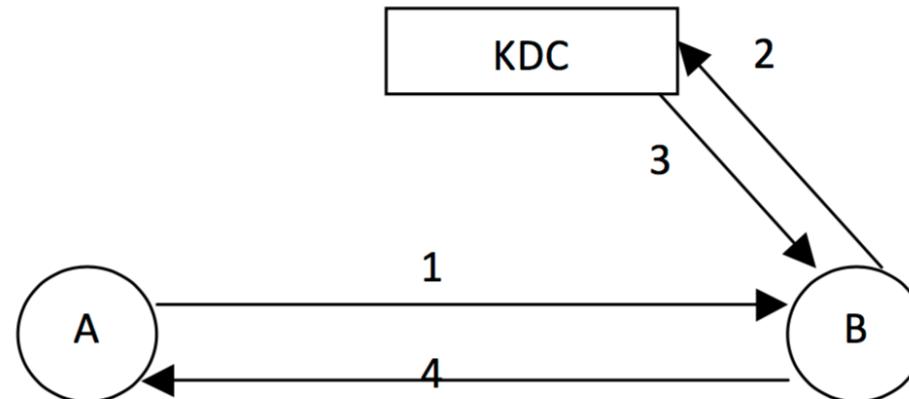
Freshnesses

KDC: modelos y protocolos - Modelos Genéricos

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

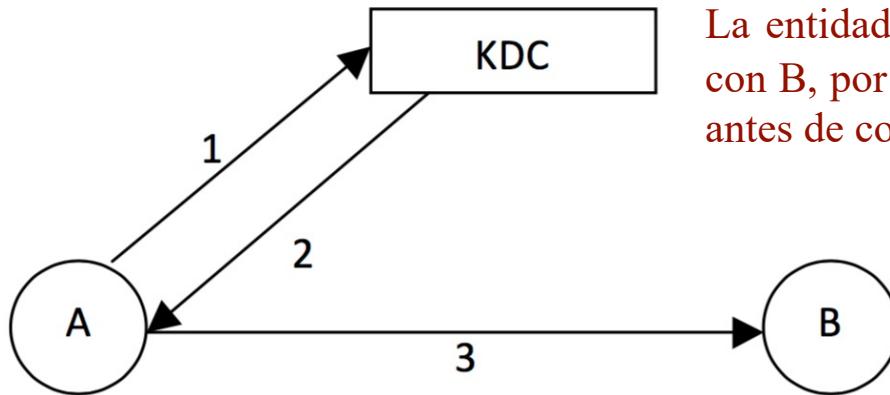


- Modelo **PUSH** para la distribución de claves:



KDC: modelos y protocolos

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



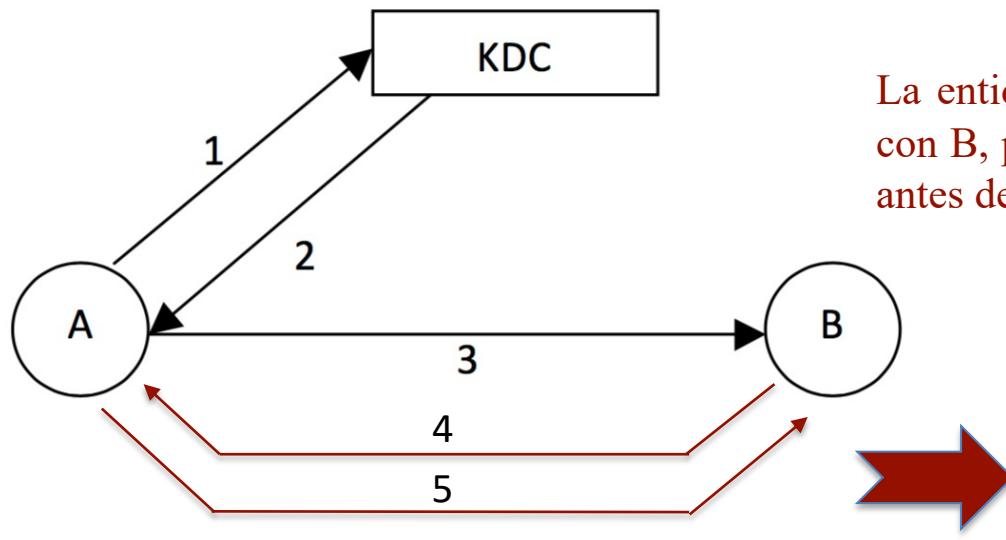
La entidad A desea tener comunicación segura con B, por lo que contacta con el KDC primero antes de comunicarse con B

- 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

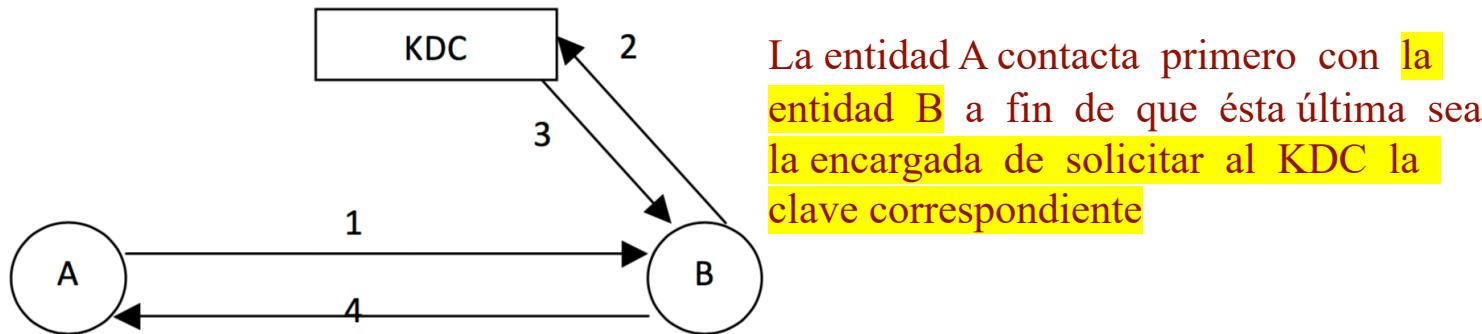
**desafío-respuesta
“challenge-response”**

- Funcionamiento general:

- **Paso 4:** B utiliza la K_{AB} para cifrar un valor único N_2 y se lo envía a A
- **Paso 5:** A recibe el valor N_2 , le aplica una transformación $f(N_2)$, lo cifra con K_{AB} y lo transmite a B

KDC: modelos y protocolos

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

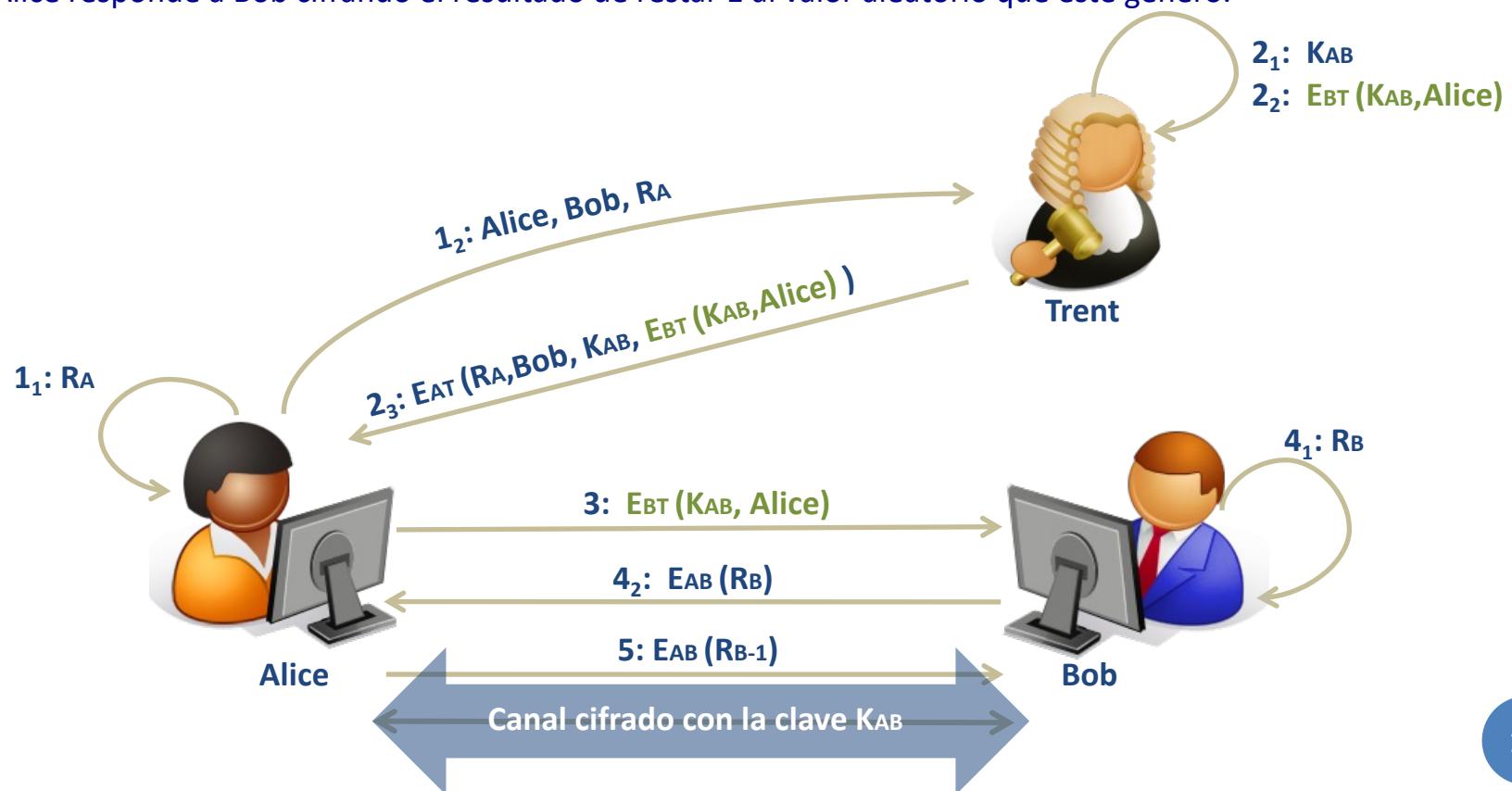


- Funcionamiento general:

- **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

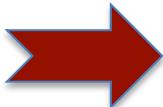
• Protocolo de Needham-Schroeder

- 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

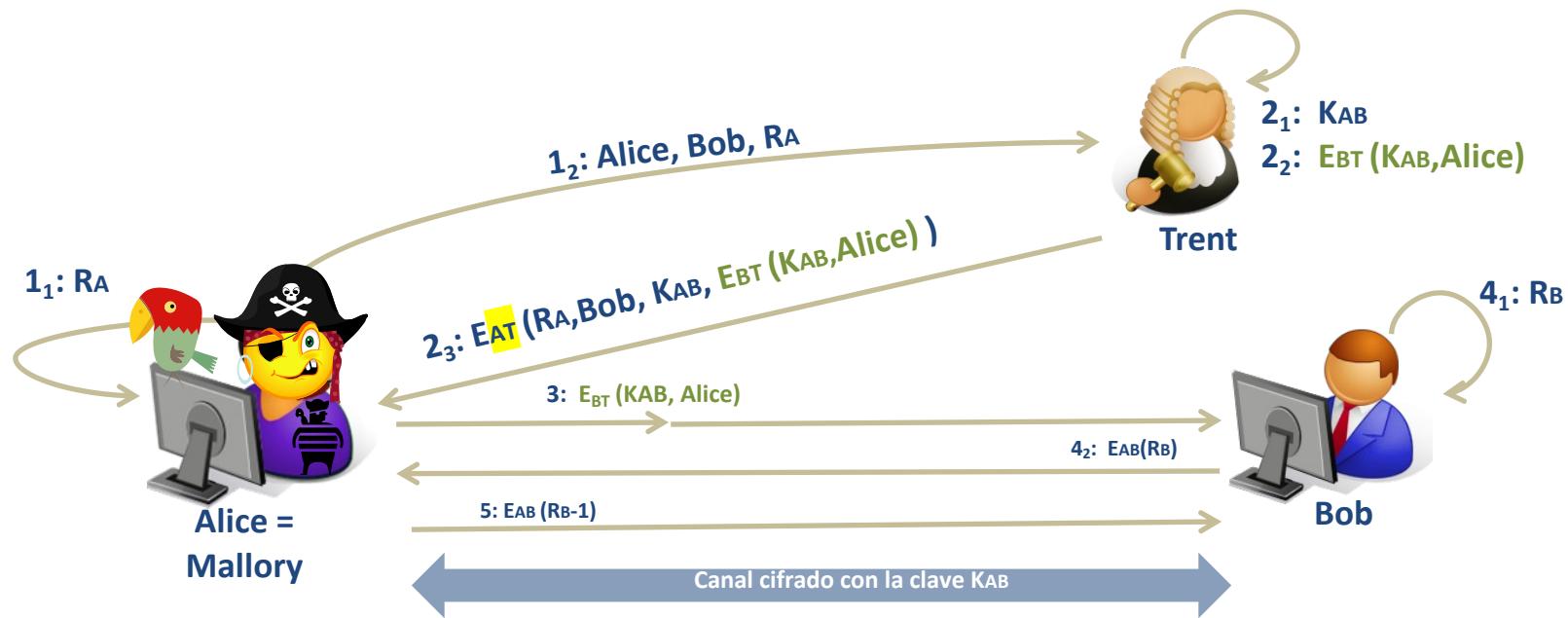
- Este protocolo tiene más **fallos de seguridad**, que fueron descubiertos años después de su funcionamiento



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

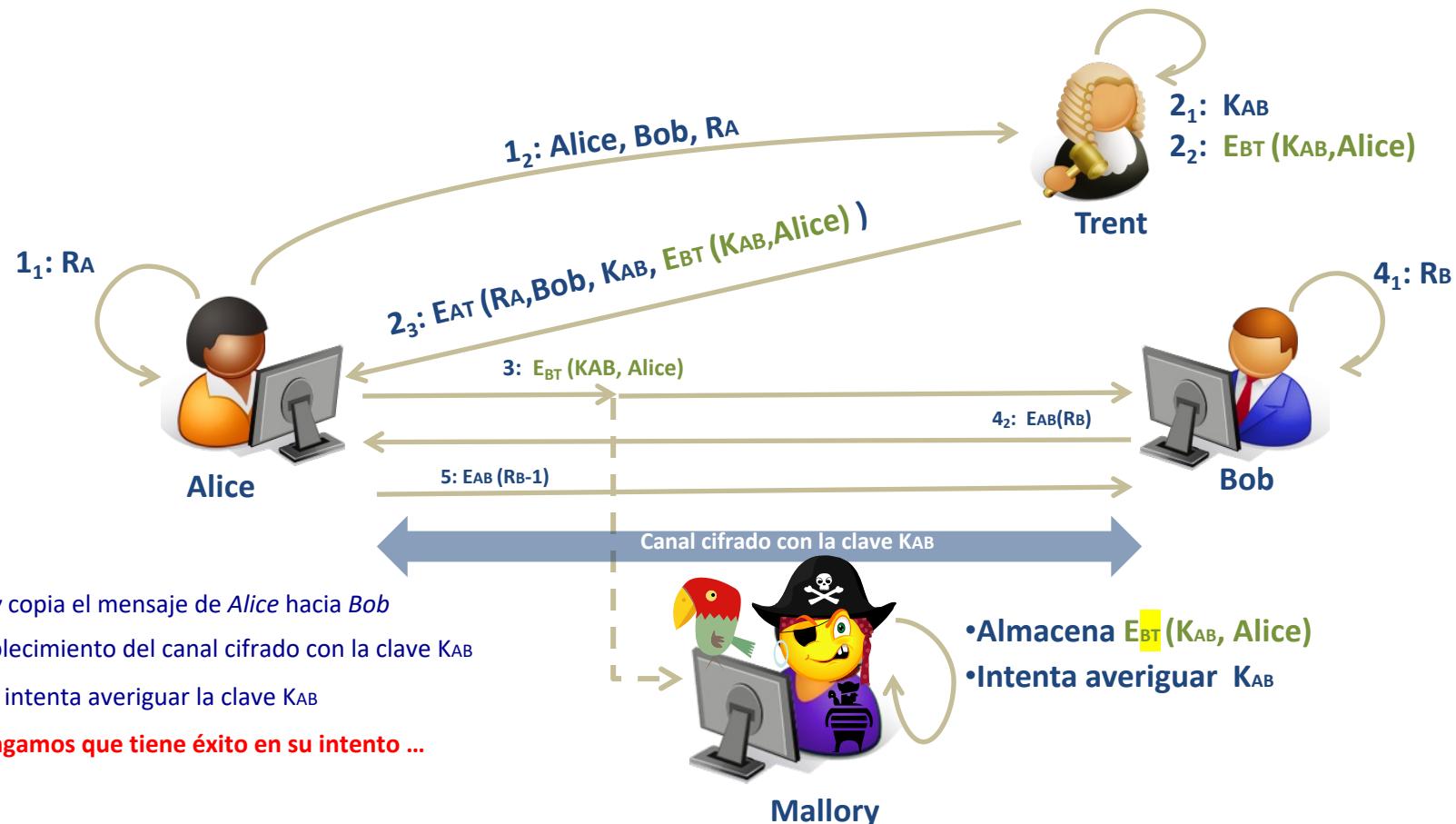


0: Mallory copia cualquier mensaje de Alice hacia Trent en el pasado y deriva la clave K_{AT} . A partir de aquí, todos los mensajes quedan comprometidos

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

supongamos que tiene éxito en su intento ...

• Ataque 2



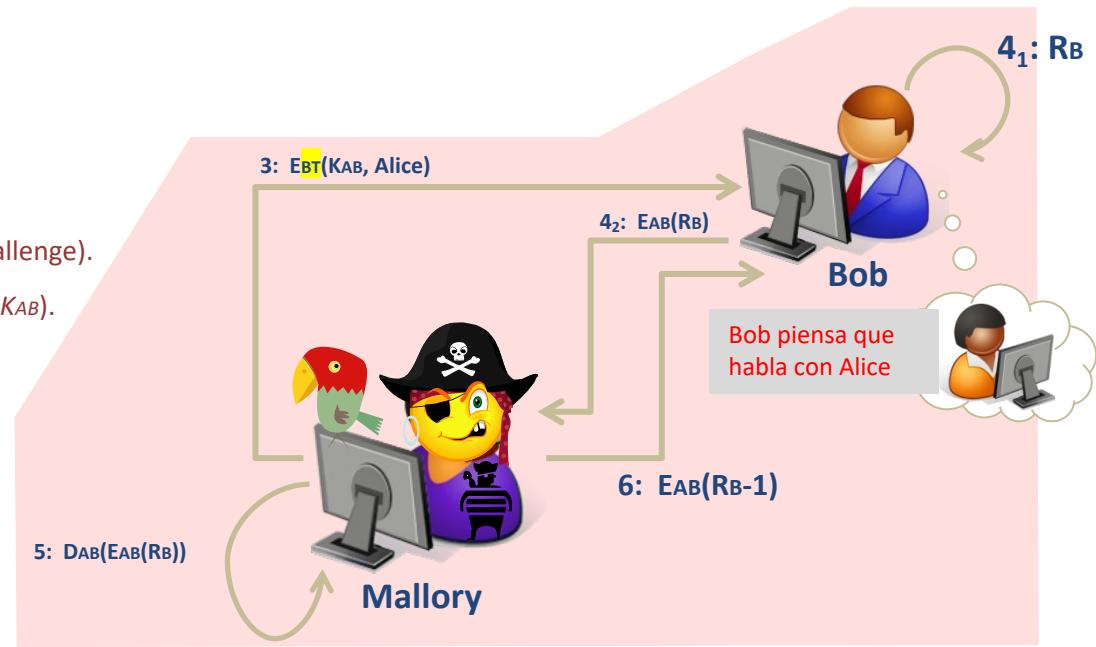
- continuación...

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

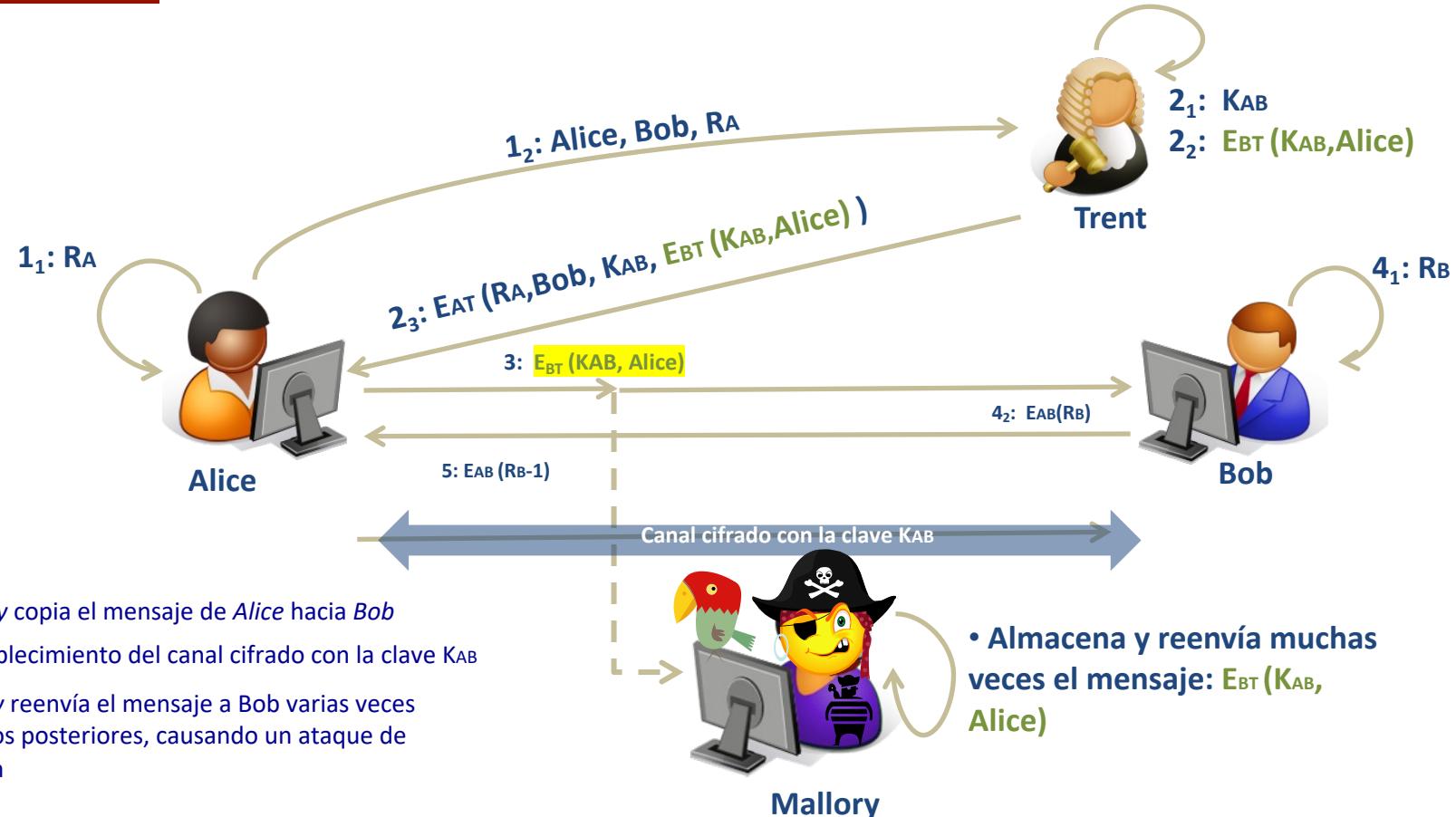
4: Bob responde a Mallory con un valor aleatorio (challenge).

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

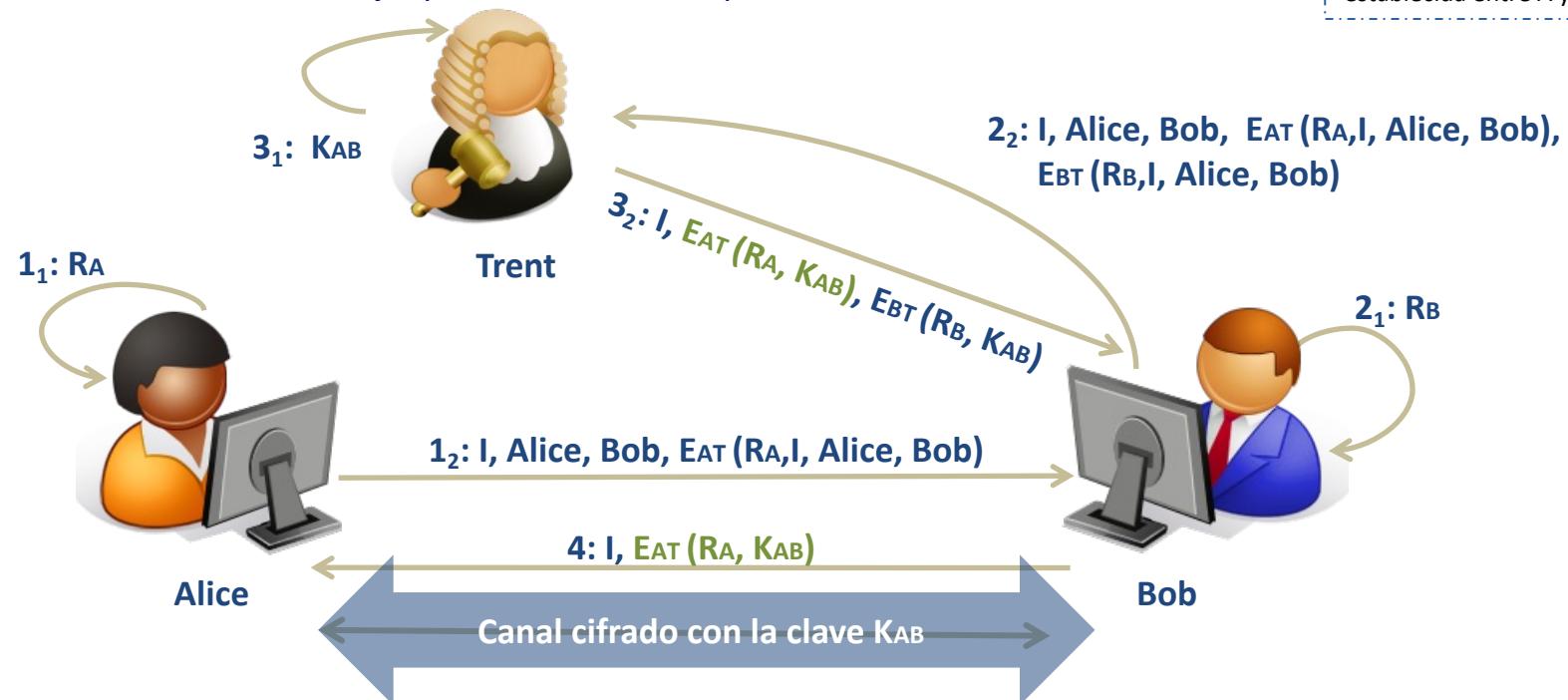
1: Alice genera el valor aleatorio R_A $\langle 1_1 \rangle$ y se lo envía hacia Bob, dentro de un mensaje cifrado con la clave que comparte con Trent $\langle 1_2 \rangle$.

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:

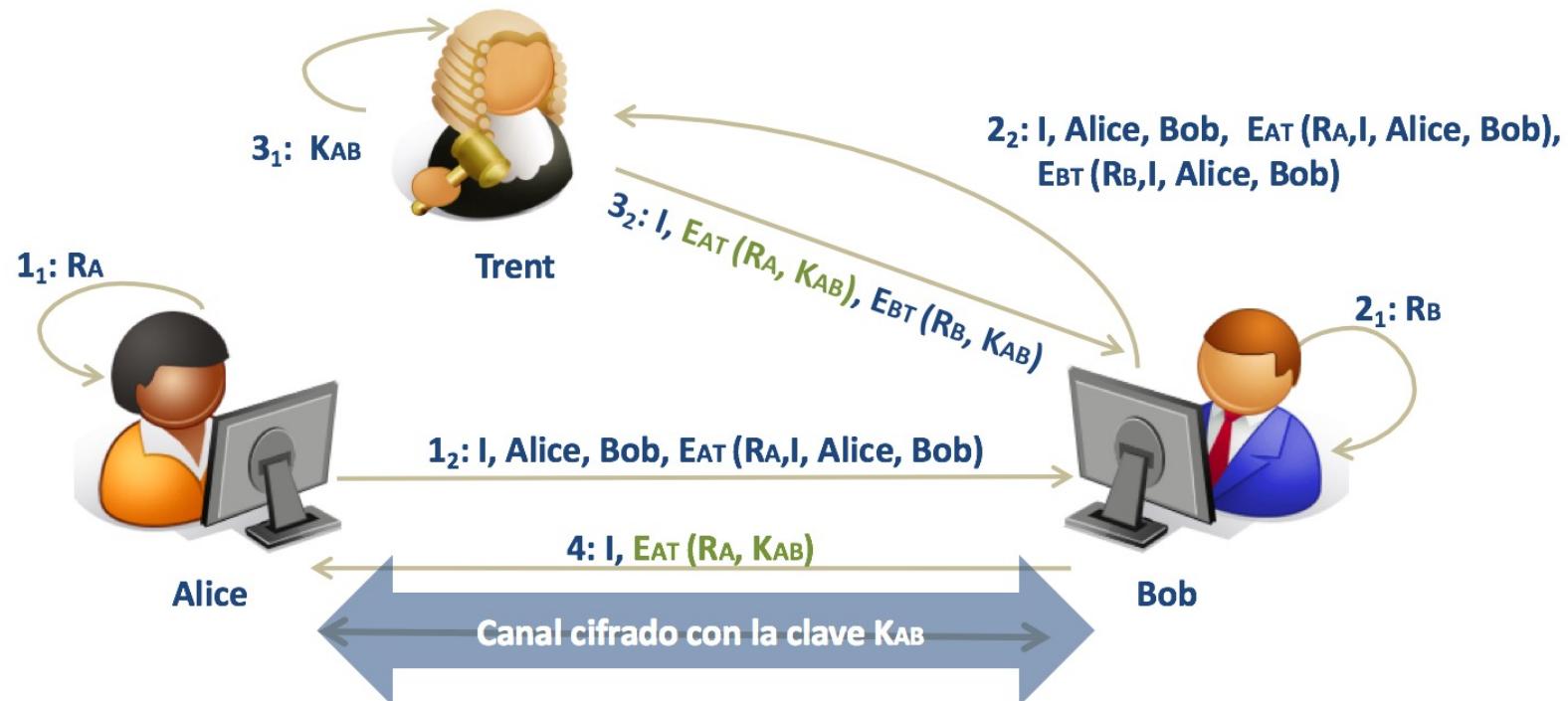
$A \rightarrow B : I, A, B, E_{KAT}\{N_a, I, A, B\}$

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

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

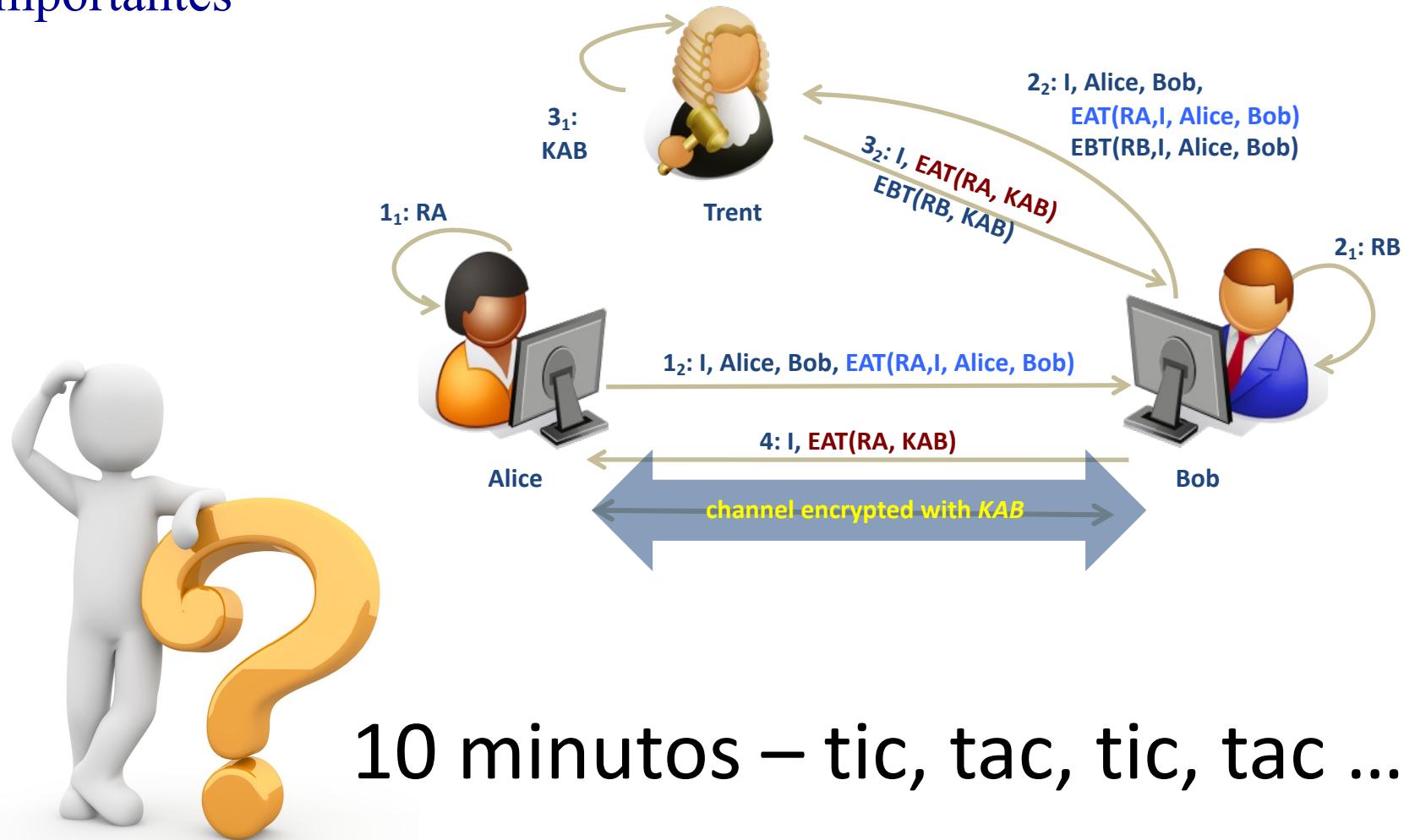
$B \rightarrow A : I, E_{KAT}\{K_{AB}, N_a\}$

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

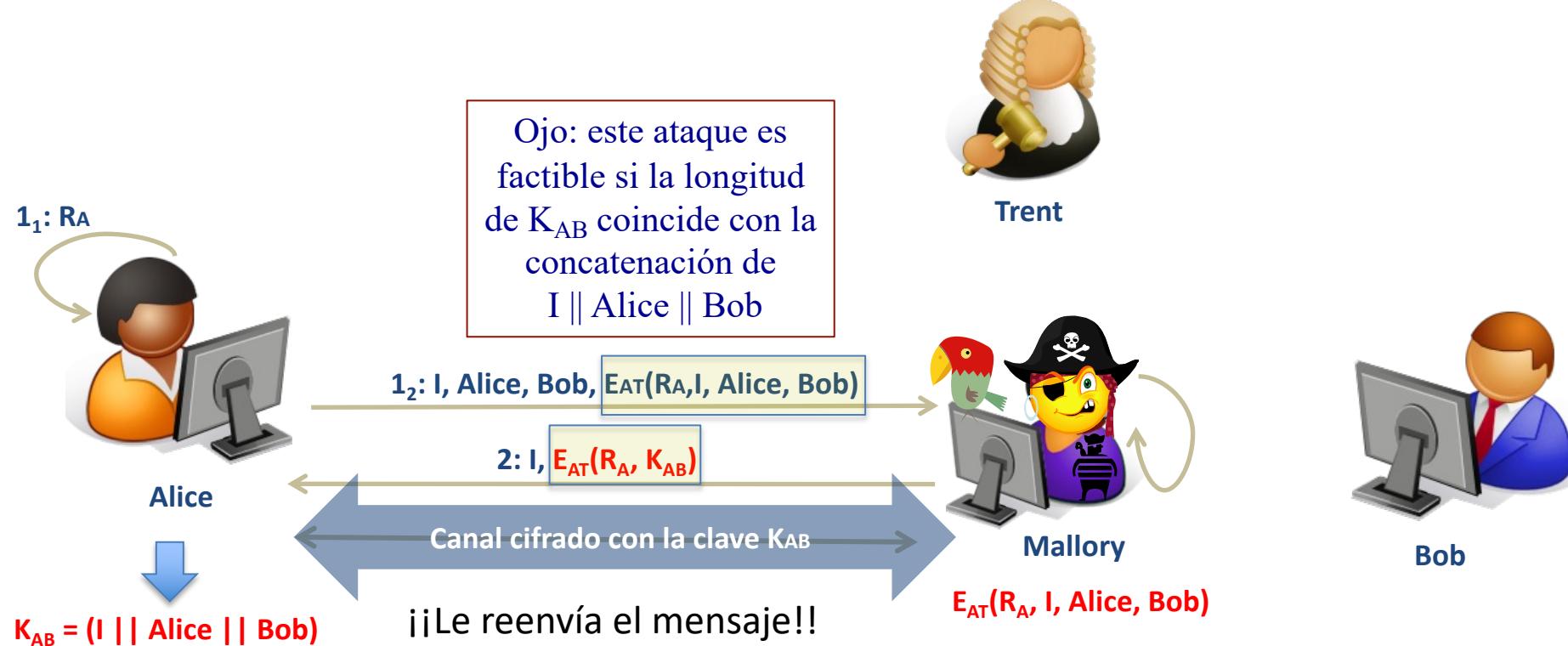


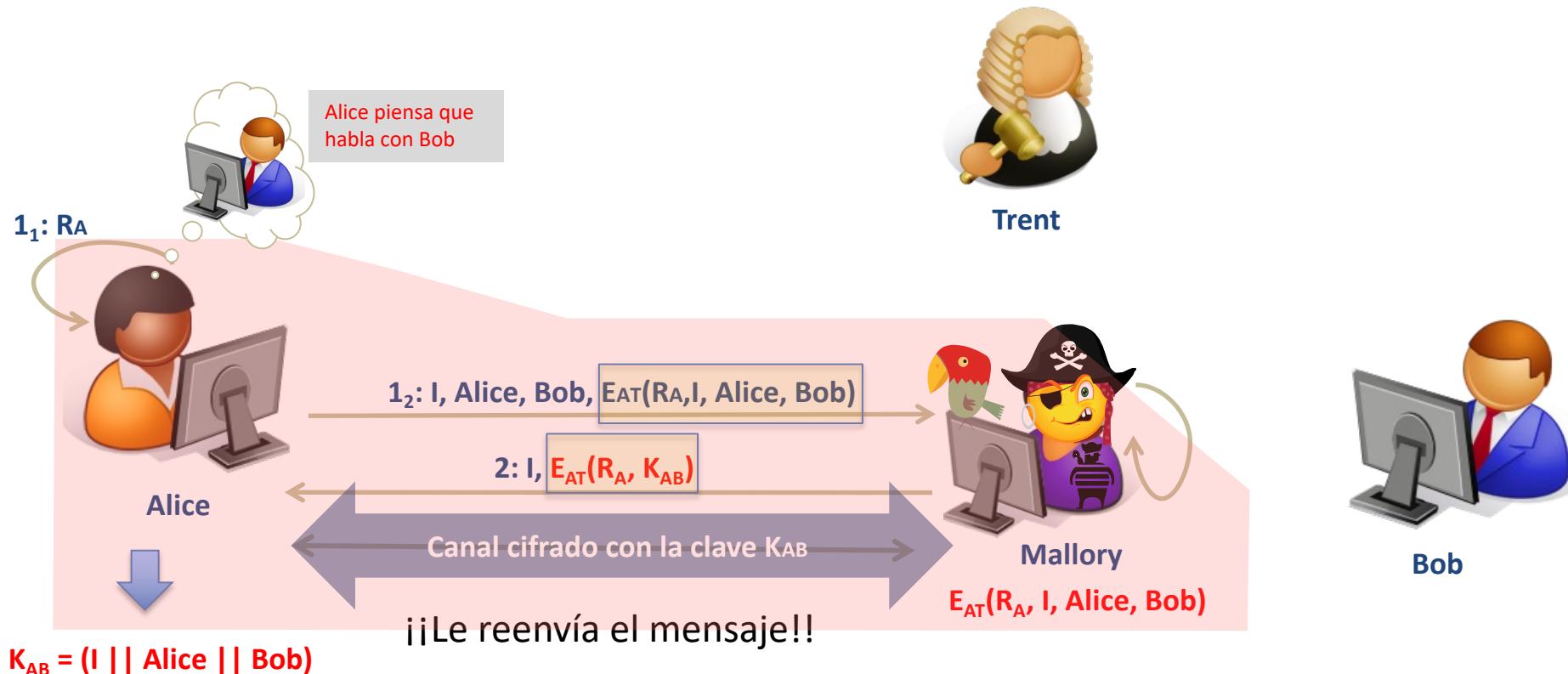
Otway-Rees

- Sin embargo, el protocolo presenta dos vulnerabilidades importantes



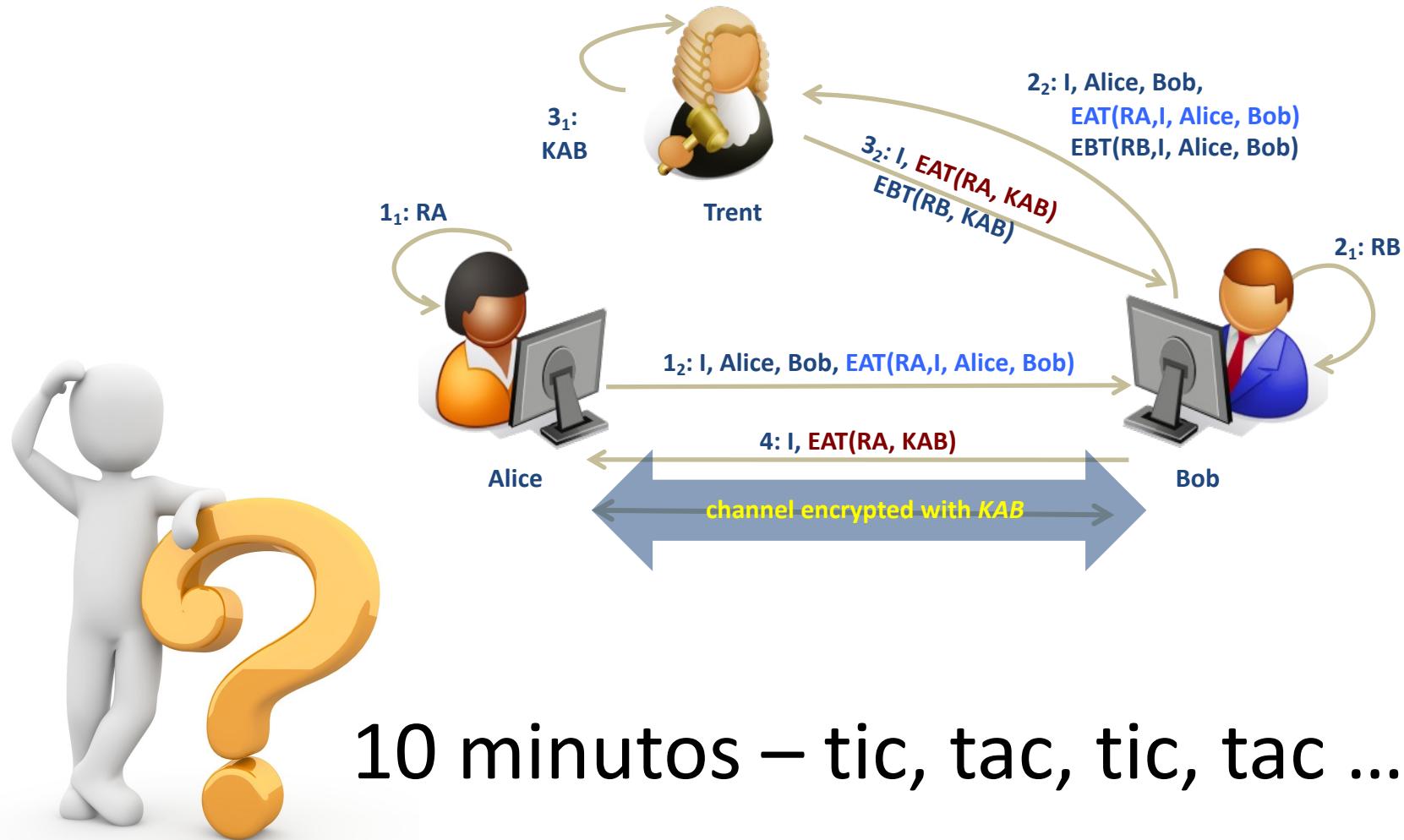
– Primer agujero de seguridad:





Otway-Rees

- ¿ Y el segundo agujero ?

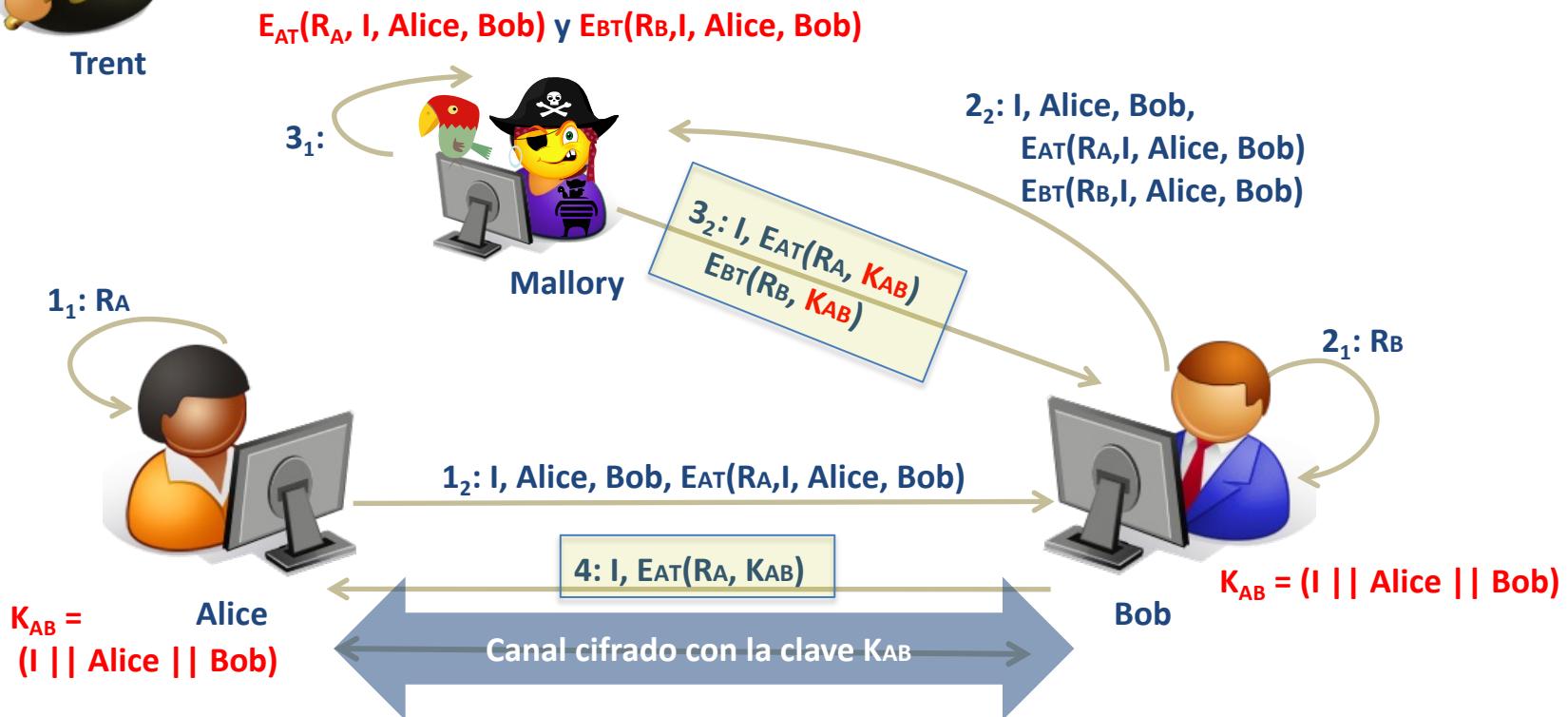


– Segundo agujero de seguridad:



Trent

¡¡Le reenvía los mismos mensajes!!

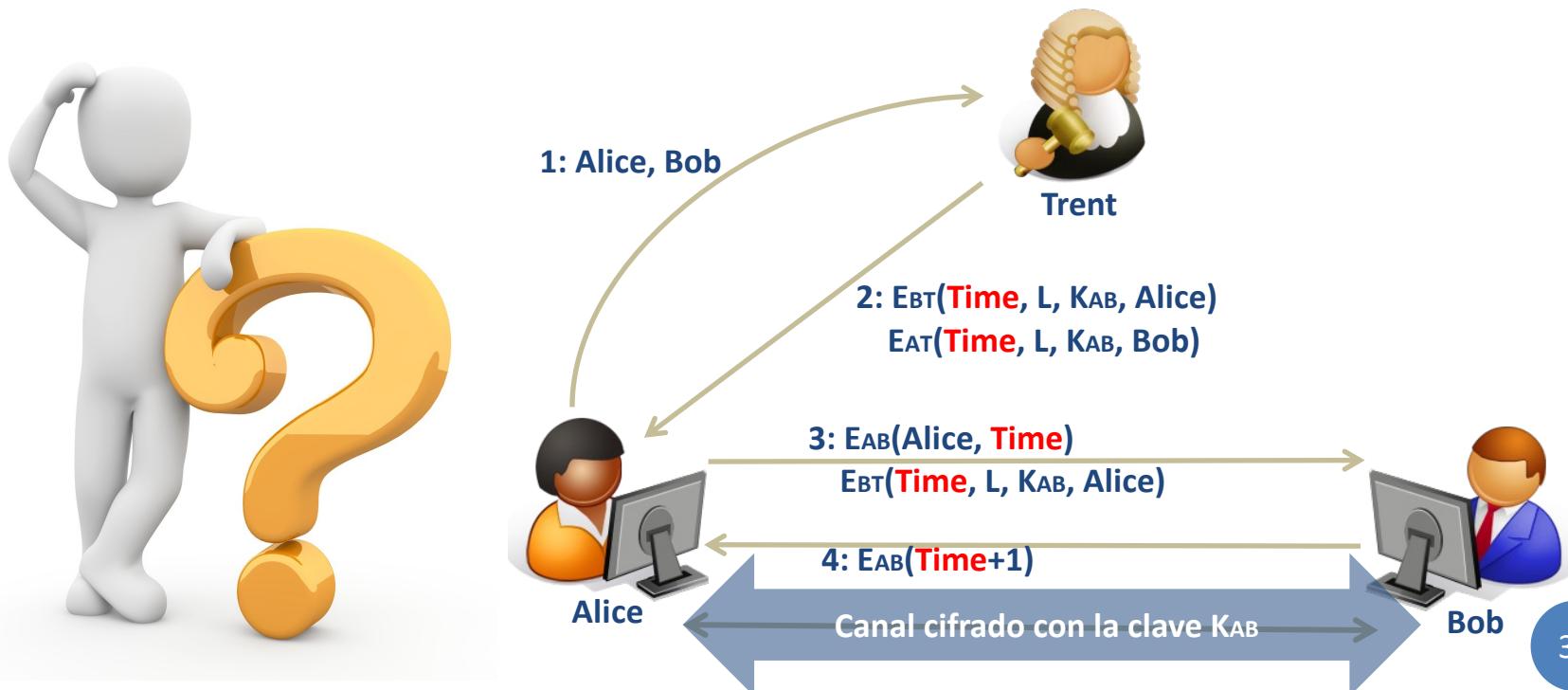


– Segundo agujero de seguridad:



• 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.

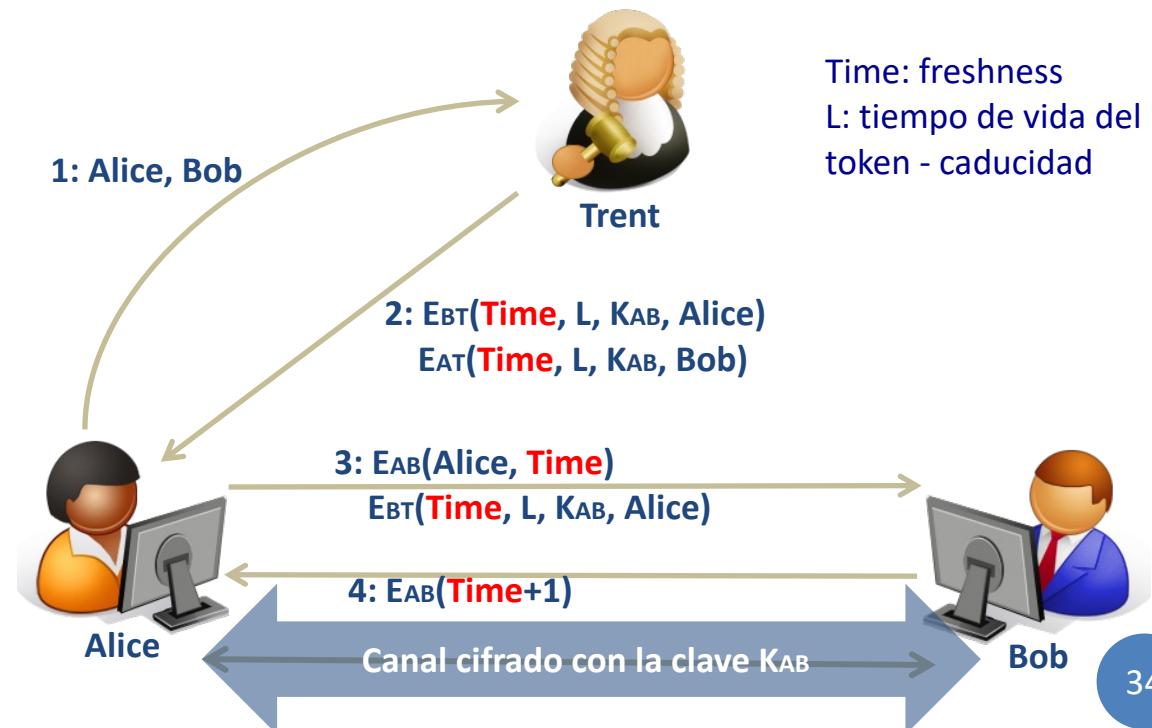


• Protocolo Kerberos

- 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.

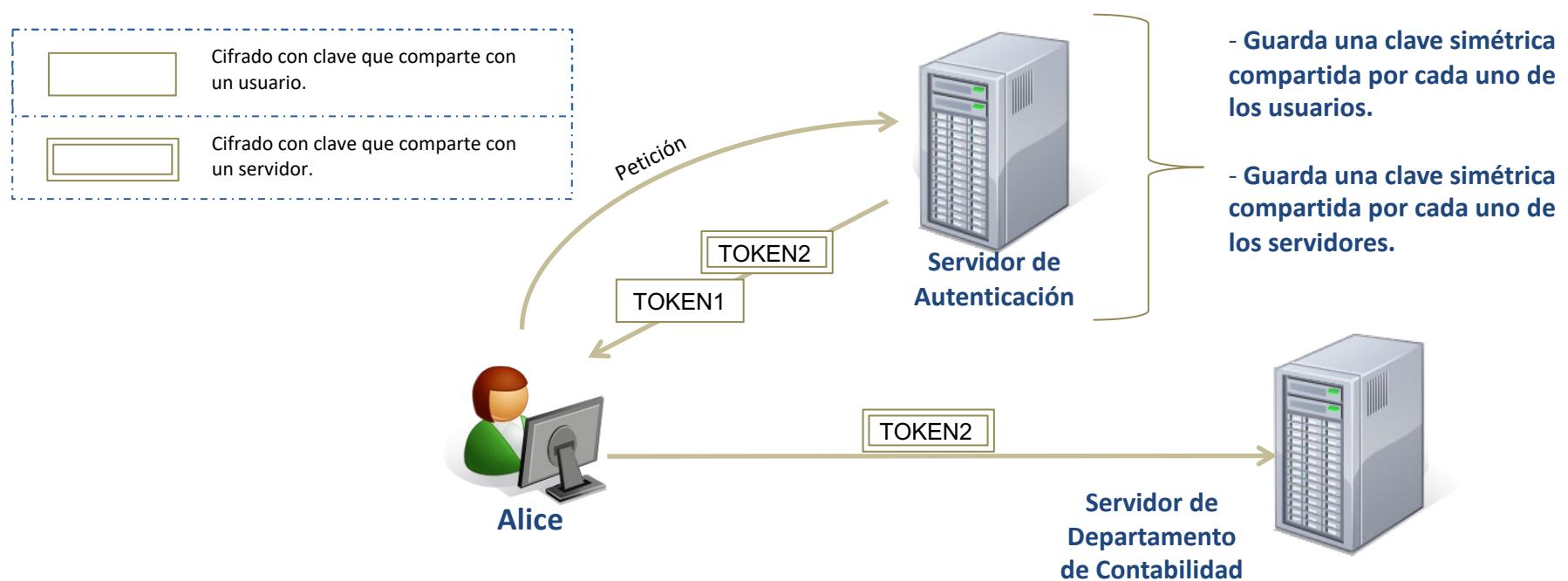
- Asume que los relojes de todos los sistemas están sincronizados
- En la práctica se sincronizan en el rango de unos pocos minutos

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

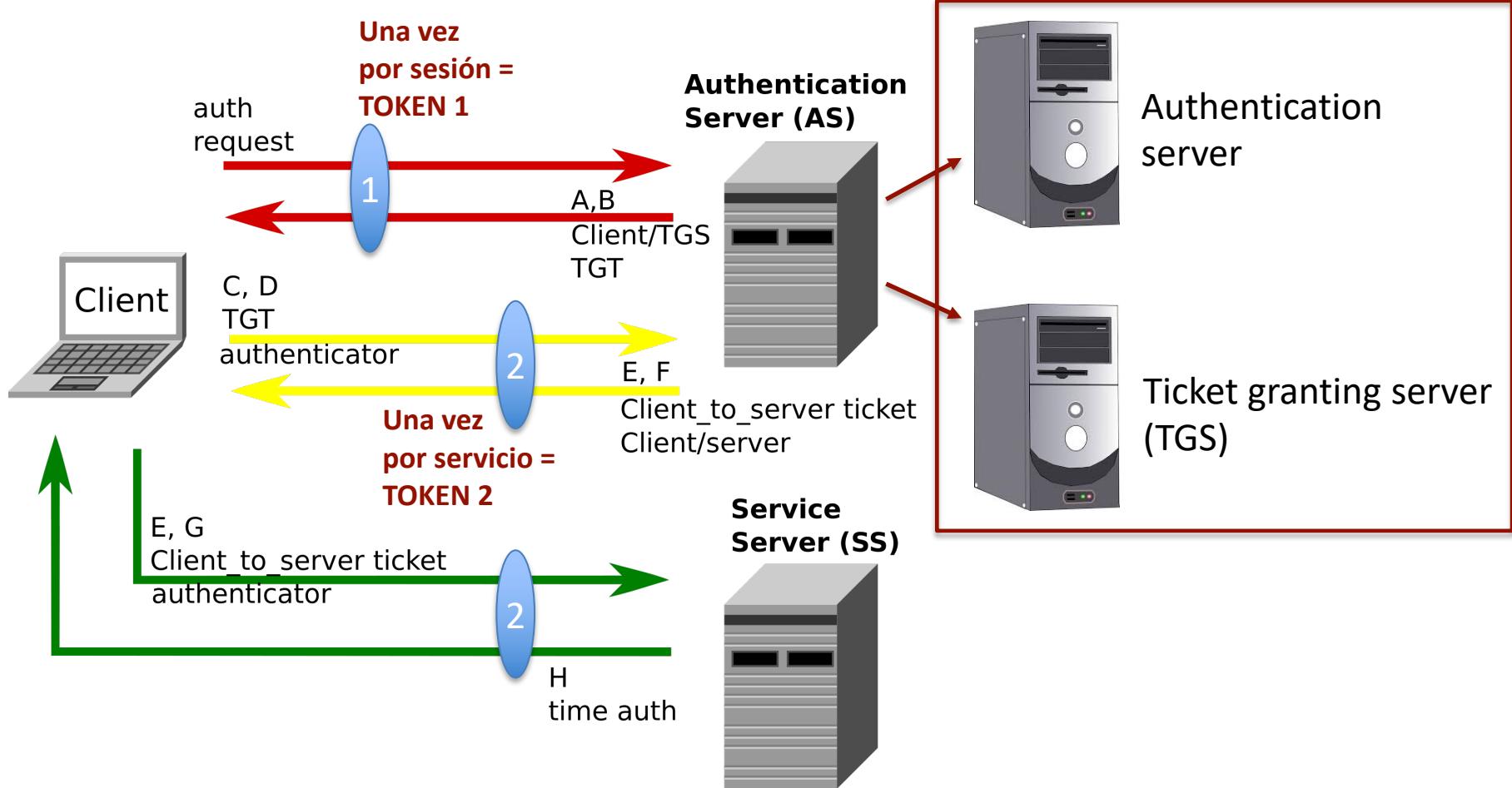


Kerberos

- 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

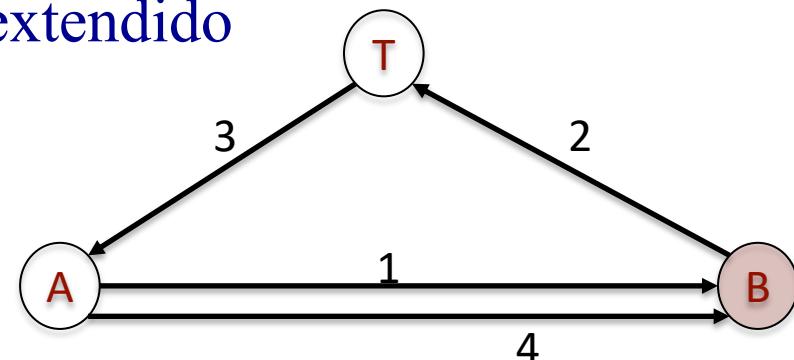


Kerberos

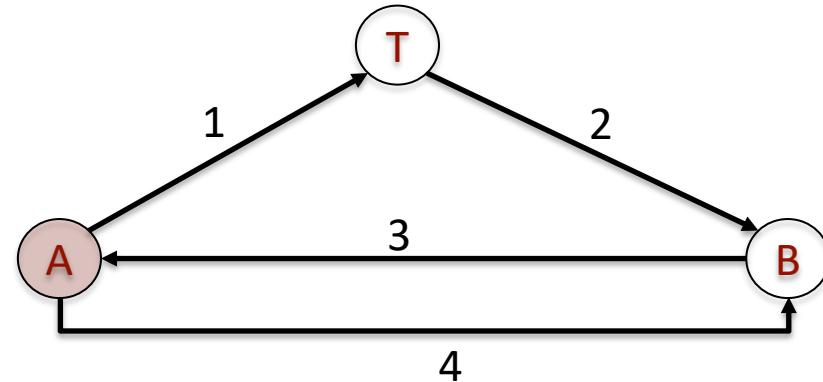


- Aparte de los protocolos anteriores, hay otros que combinan múltiples tipos de estrategias (a nivel de modelos como de mecanismos de seguridad):

– PUSH con PULL → PUSH extendido



– PULL con PUSH → PULL extendido



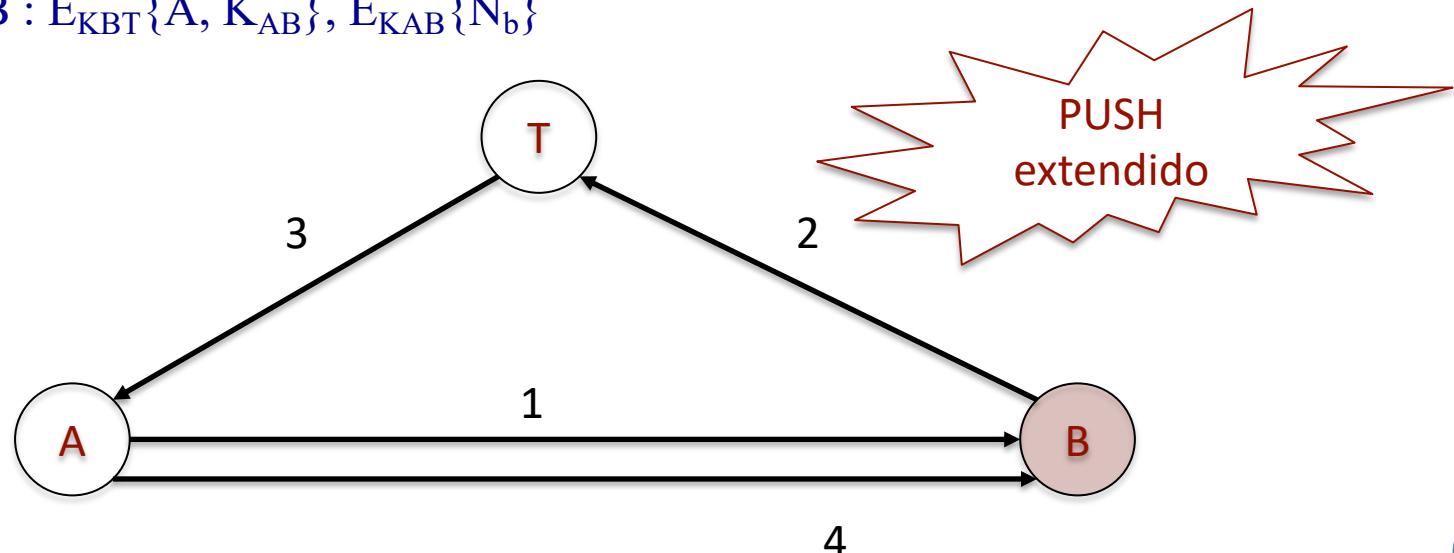
- **Yahalom**
 - Objetivo: permitir a Trent generar la clave de sesión K_{AB} y enviar dicha clave a Alice (de manera directa) y a B (de manera indirecta)

$A \rightarrow B : A, N_a$

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

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

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



- 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*

$A \rightarrow B : A, N_a$

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

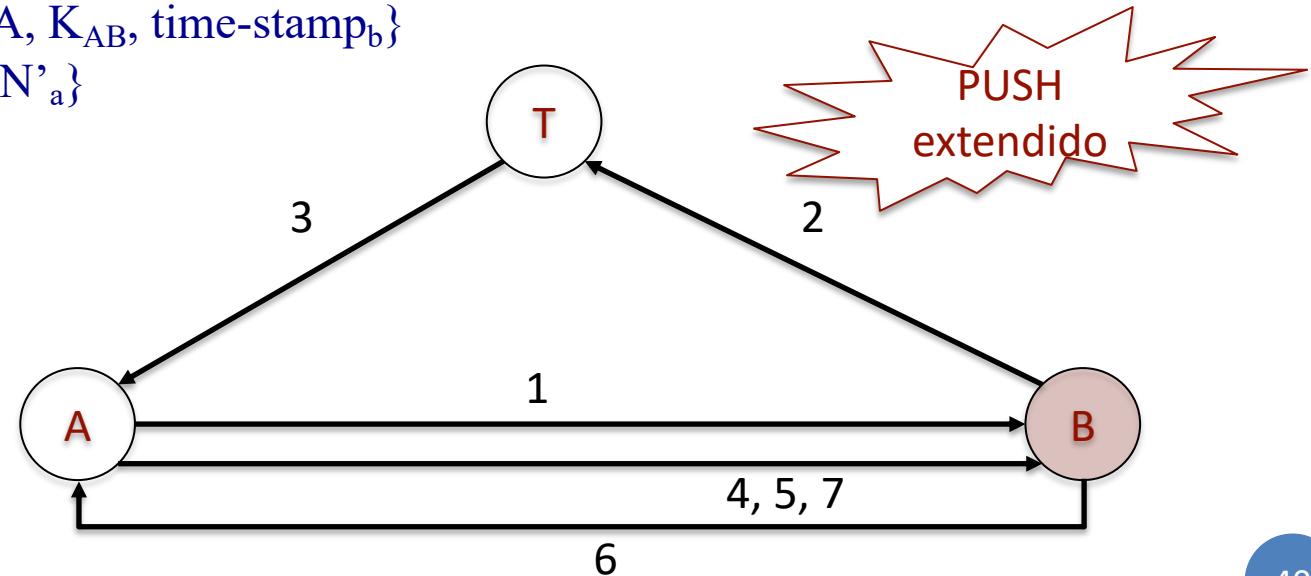
$T \rightarrow A : EK_{AT}\{B, K_{AB}, N_a, \text{time-stamp}_b\}, EK_{BT}\{A, K_{AB}, \text{time-stamp}_b\}, N_b$

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

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

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

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



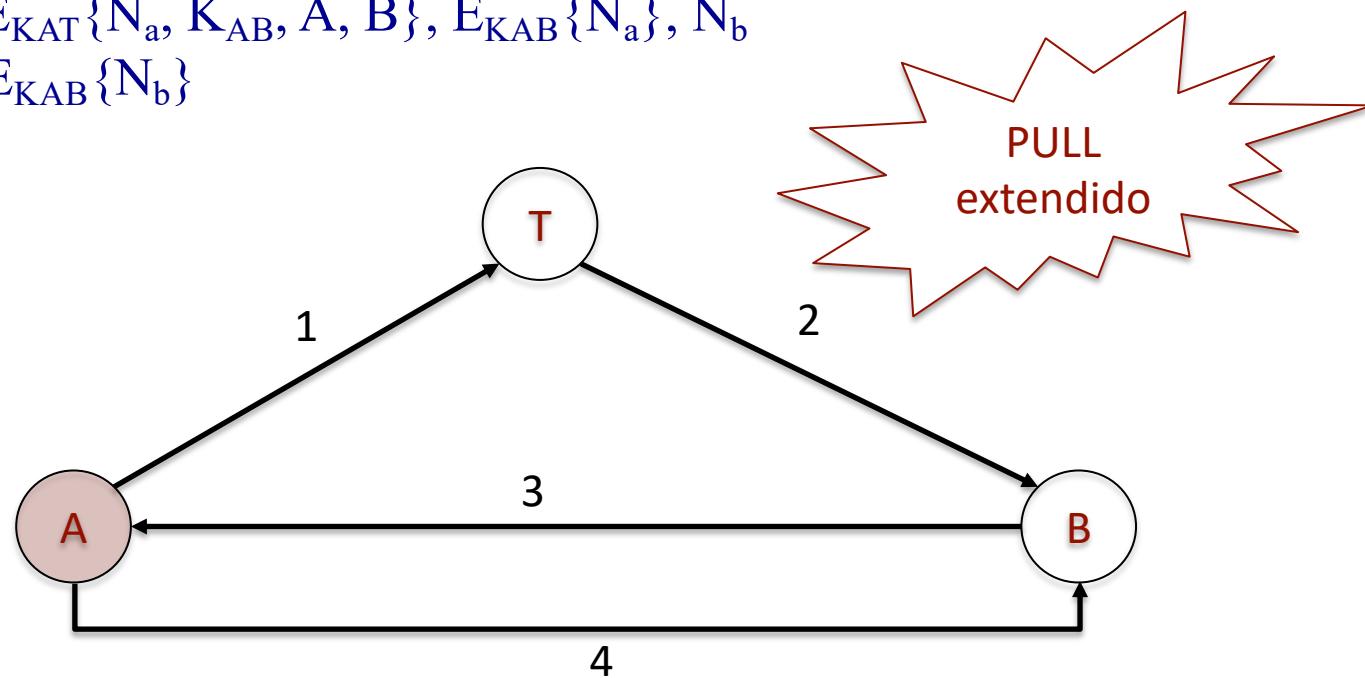
- **Kao Chow**

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

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

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

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

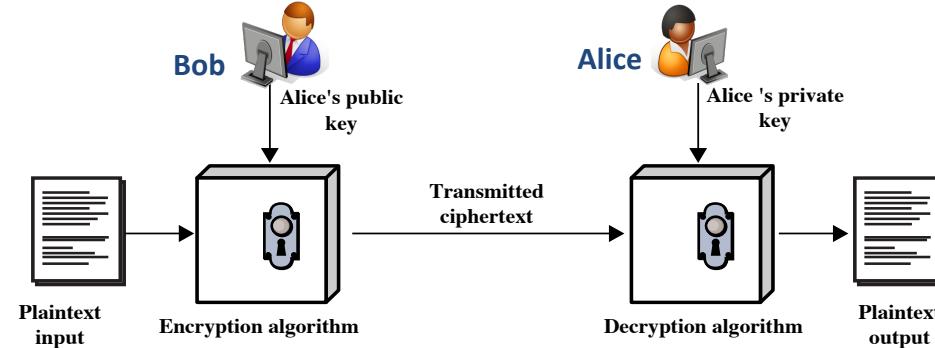


- Hemos visto que existen diversos protocolos para solucionar el problema de la administración/intercambio de claves
 - El protocolo a elegir depende del escenario y de lo que se quiere proteger
 - ¿Se quiere aprovechar los canales de comunicación?
 - ¿Quién ha de contactar primero con el KDC: Alice 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?
 -

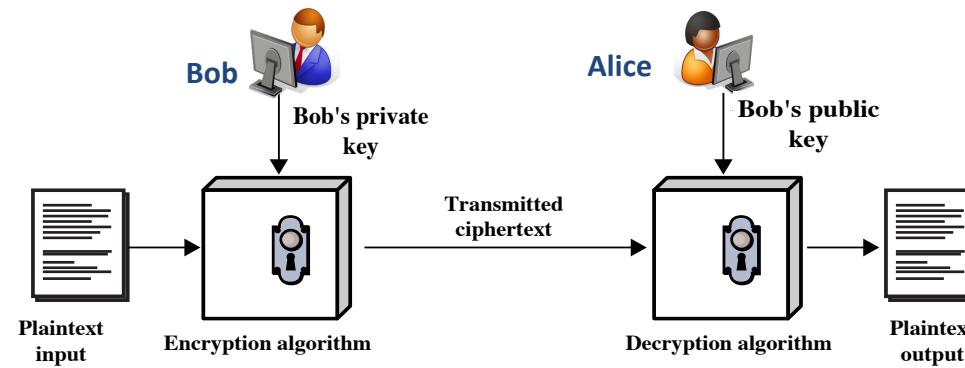
- Usar un KDC no está exento de potenciales problemas:
 1. el KDC posee suficiente **información para suplantar a cualquier usuario**
 - y si un intruso llega hasta él todos los documentos cifrados que circulan por la red se hacen vulnerables
 2. el KDC representa un **único punto de fallos** (o ataques)
 - 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?

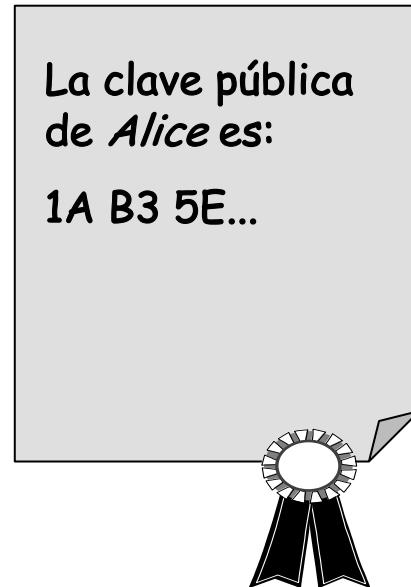


- ¿Cómo sabe *Alice* en este escenario si la clave pública de *Bob* es genuina?



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:



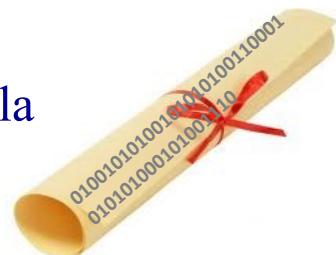
Certificados digitales

- 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



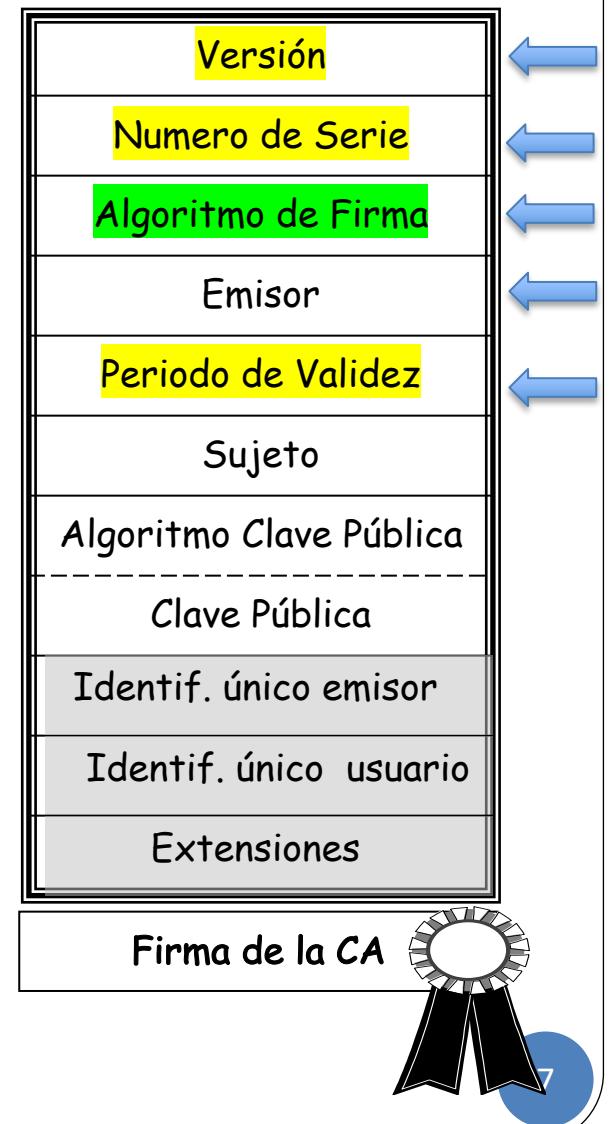
Certificados digitales

- 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

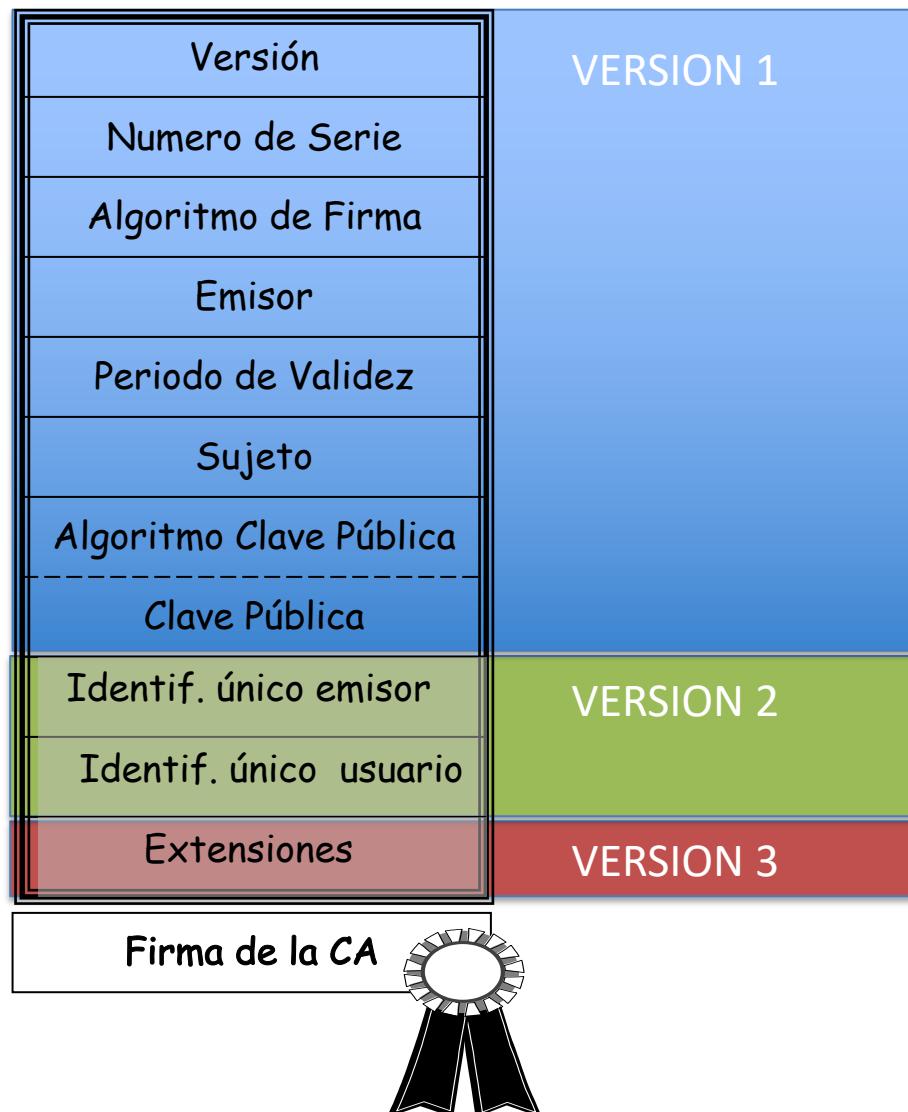


Certificados digitales

- 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)

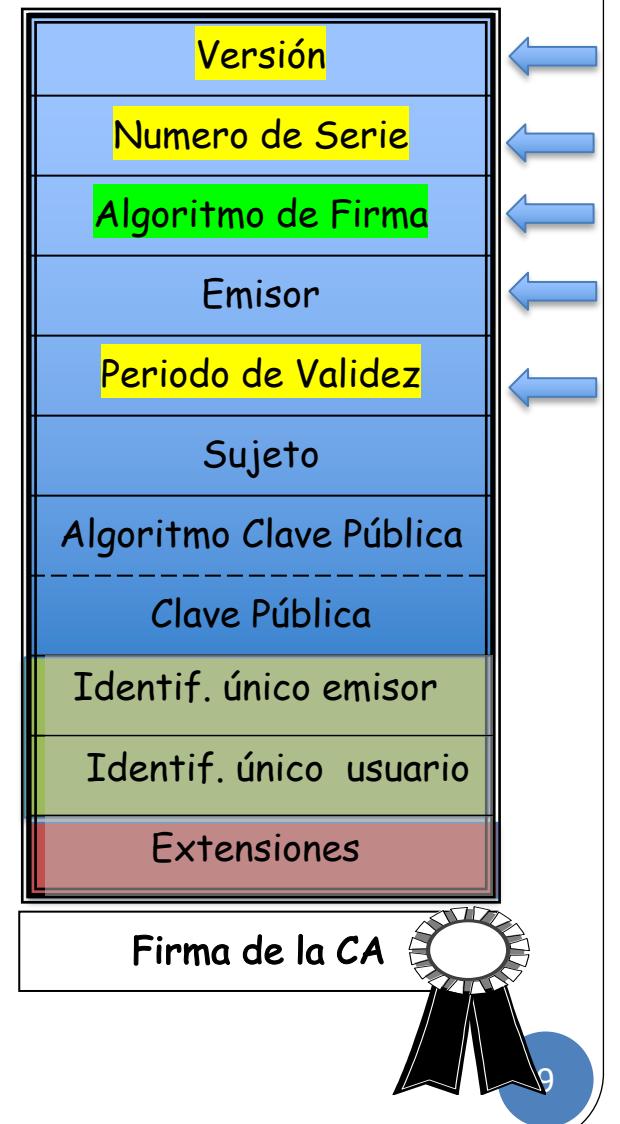


Certificados digitales



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

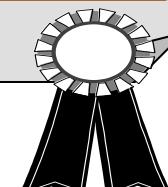
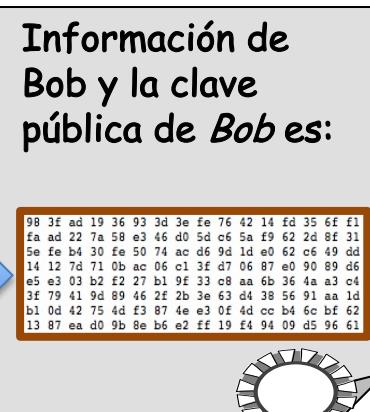
– *Algoritmo Clave Pública:*

- 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:

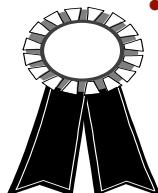
- Dirección de email, dirección postal

– *Extensiones:*

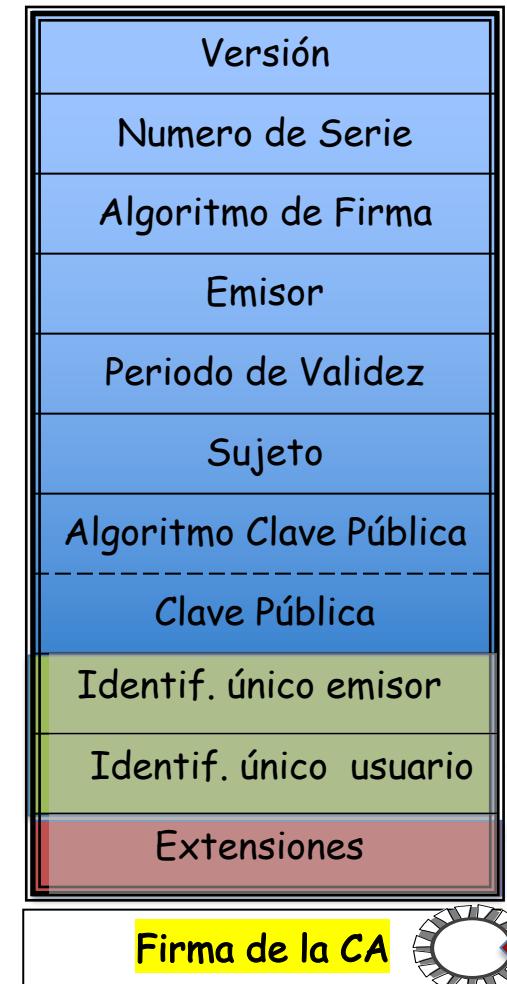
- Campo **opcional** para almacenar información de distinto tipo (versión 3)

– *Firma:*

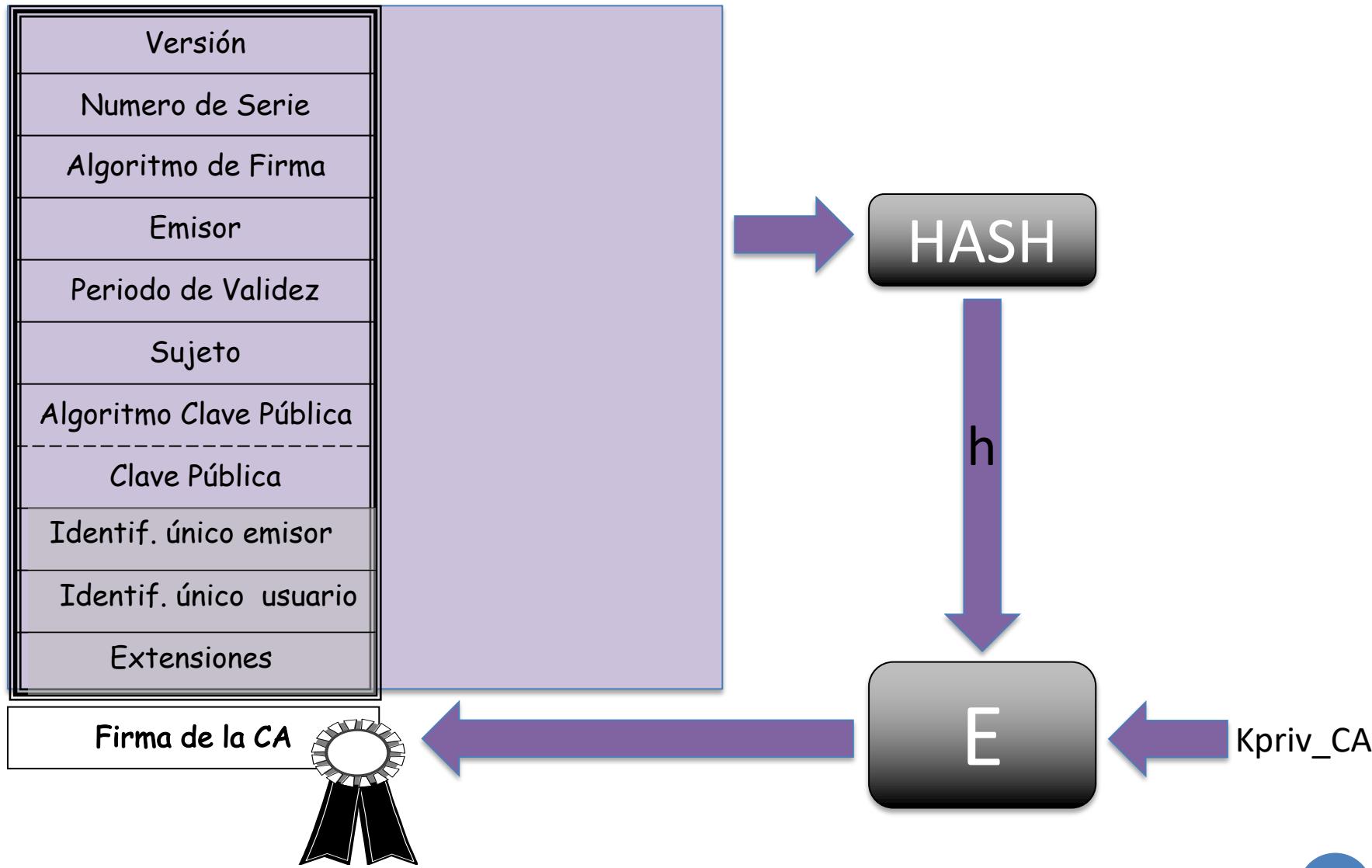
- **Firma digital de la CA sobre el valor hash del conjunto de los demás campos del certificado**



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



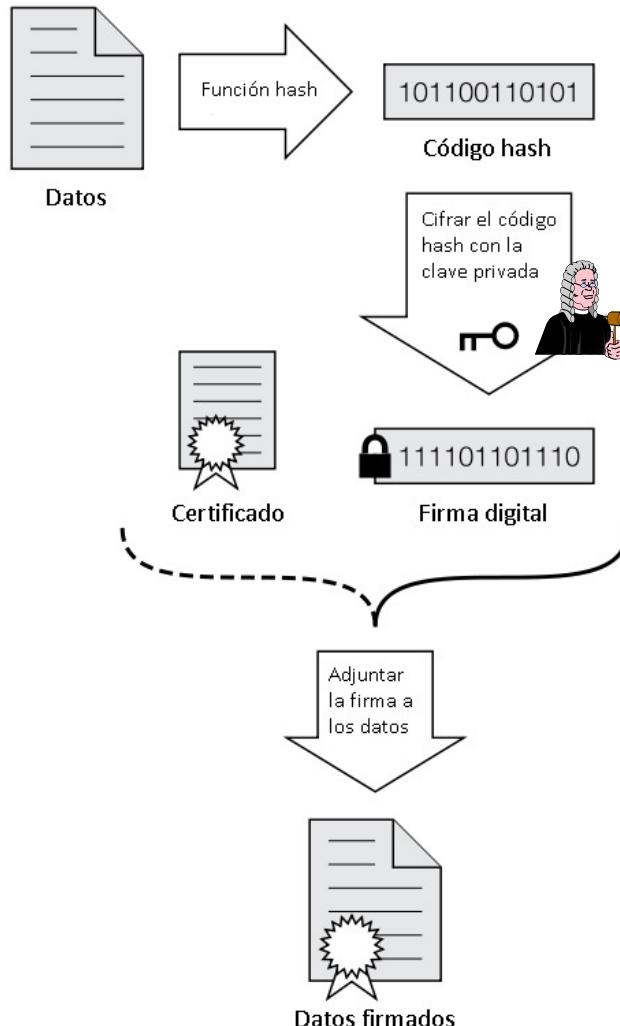
Certificados digitales



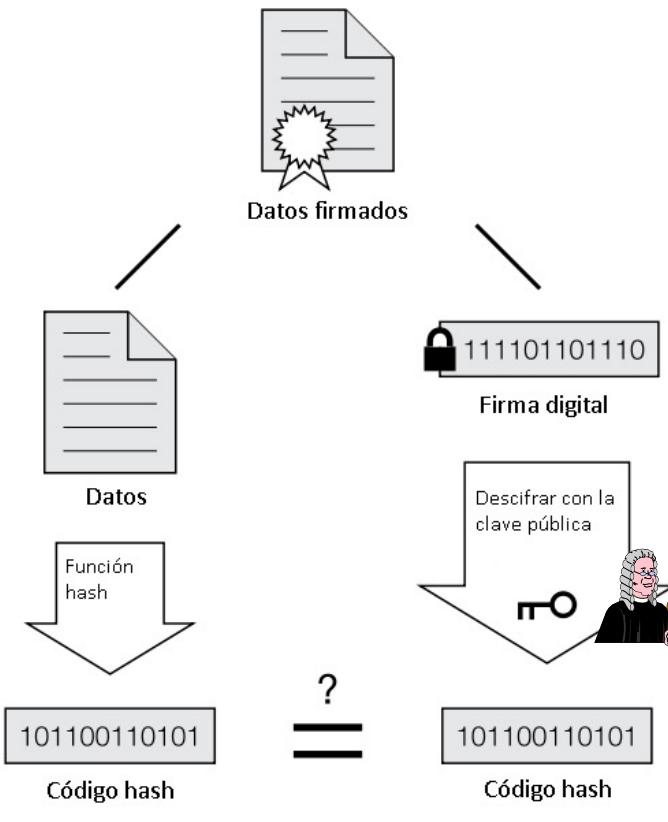
Certificados digitales



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

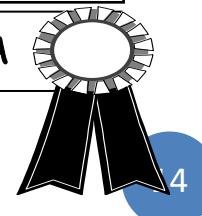
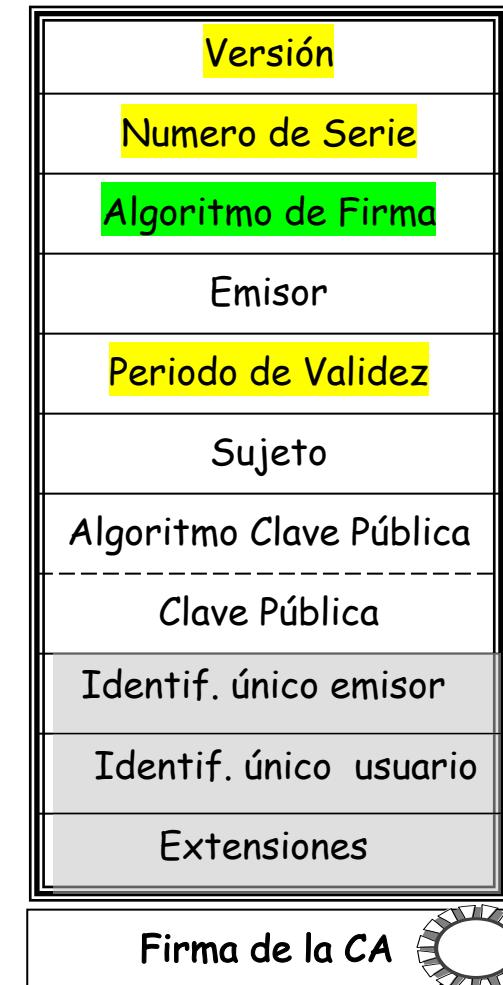
Versión 3

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

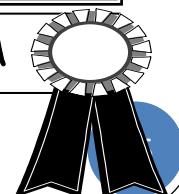
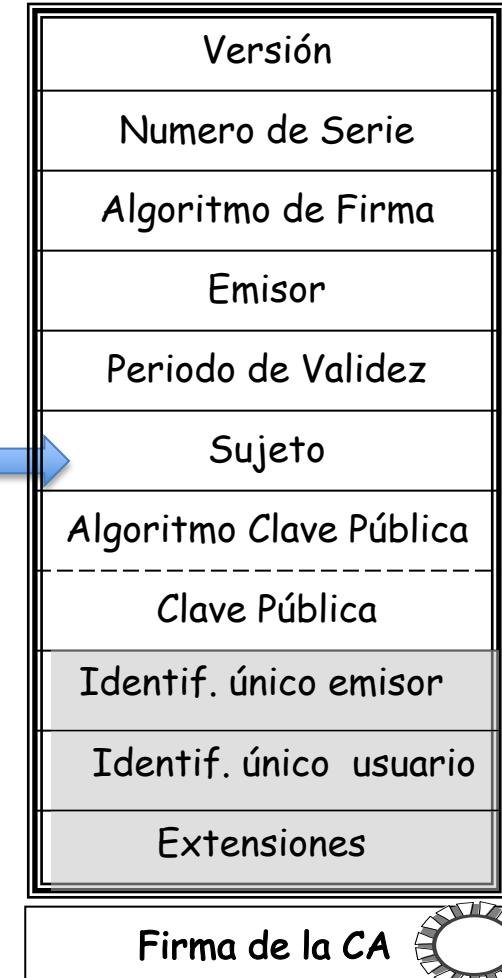
Versión 3

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

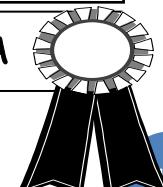
Clave Pública

Identif. único emisor

Identif. único usuario

Extensiones

Firma de la CA



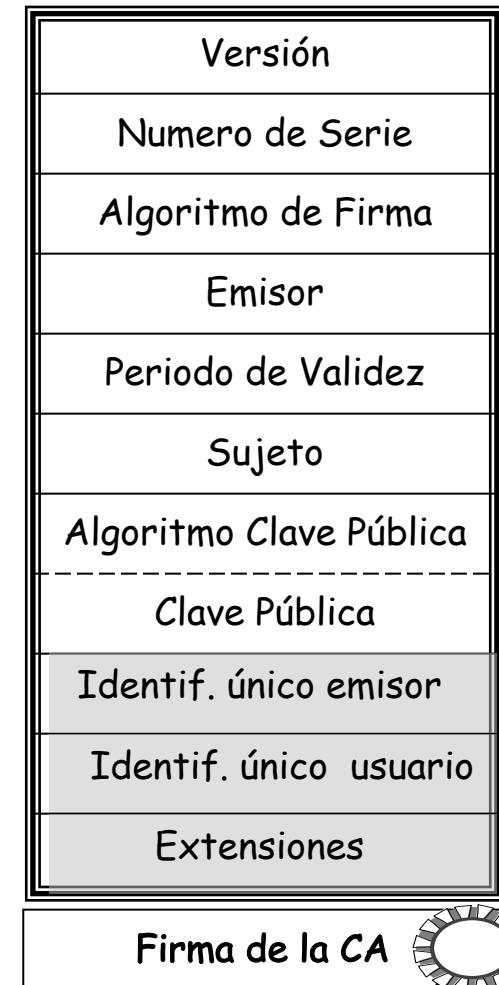
Ejemplo de un certificado digital - en MAC

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

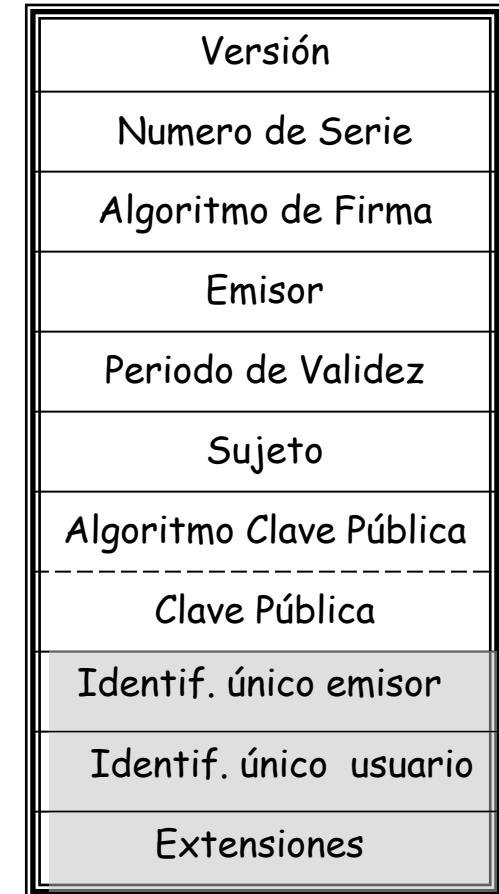
Información de la clave pública

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 ...

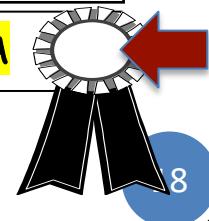
Extensión	Restricciones básicas (2.5.29.19)
Crítico	NO
Entidad de certificación	
Extensión	Identificador de clave del sujeto (2.5.29.14)
Crítico	NO
Nombre de la clave	24 85 28 DC 50 C1 BD 39 5D D0 D5 72 A5 09 1B F4 73 9D AF 7F
Extensión	Identificador de clave de entidad emisora (2.5.29.35)
Crítico	NO
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

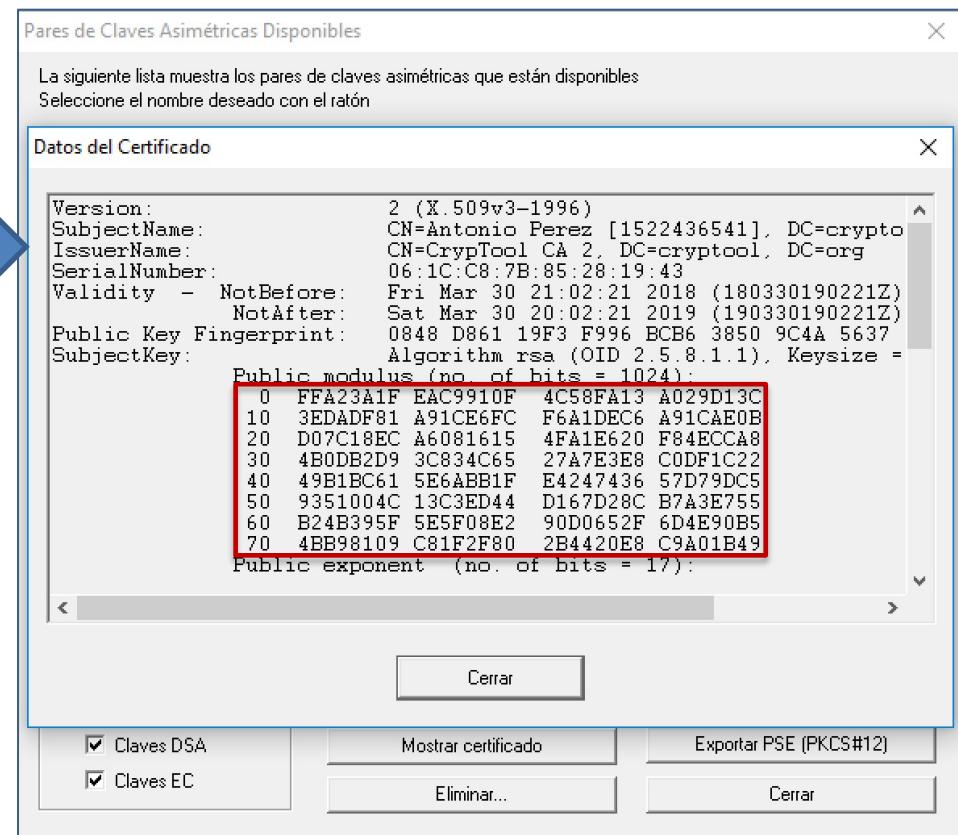
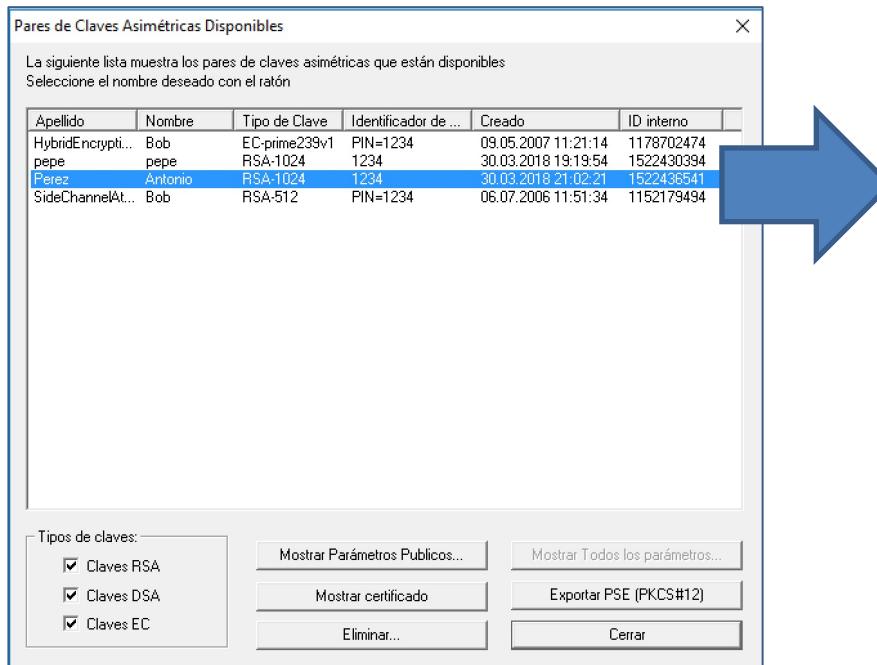


Firma de la CA



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

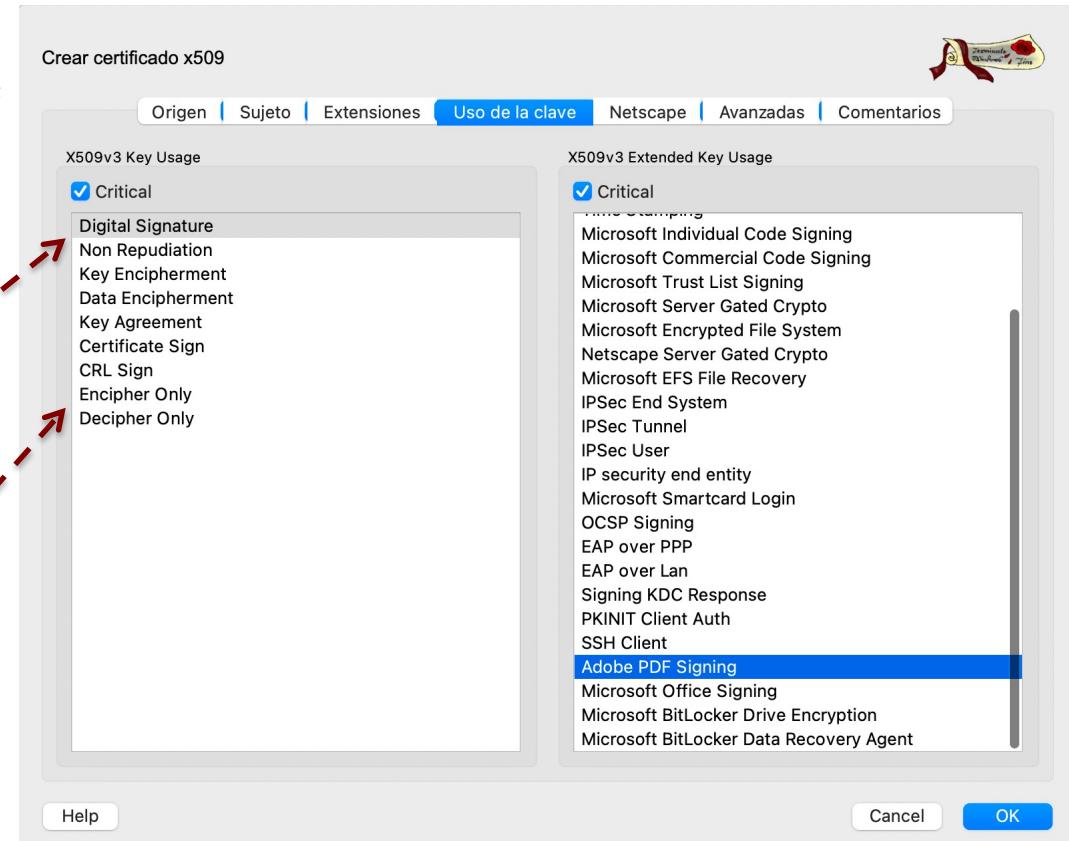


Uso de claves en los certificados digitales

- 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

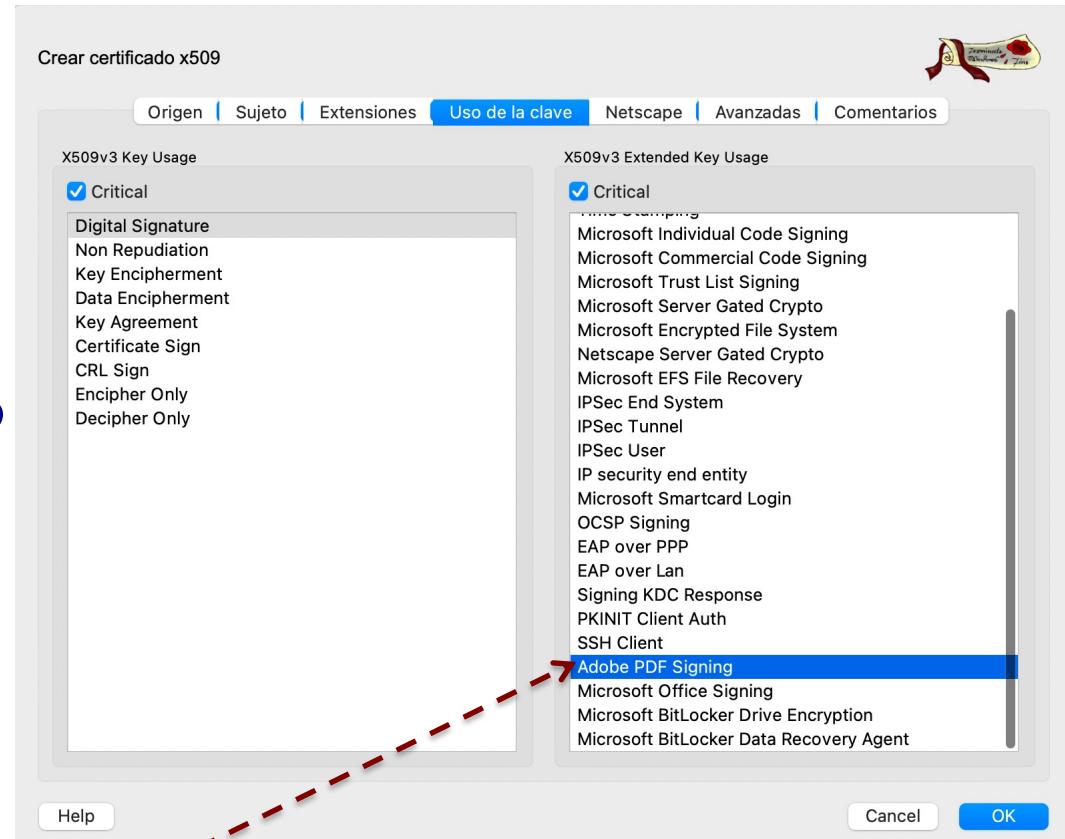


Con critical lo que hacemos es forzar el proceso

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

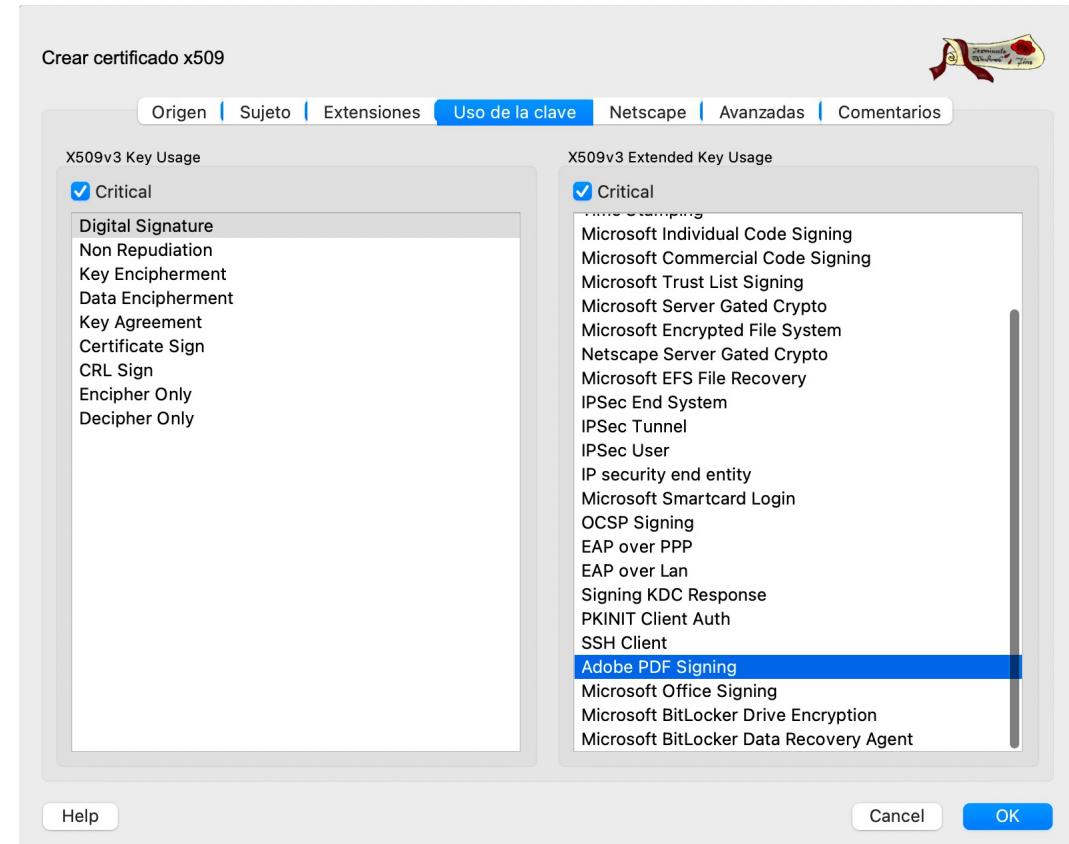
Por ejemplo, para firmar documentos PDF



Con critical lo que hacemos es forzar el proceso

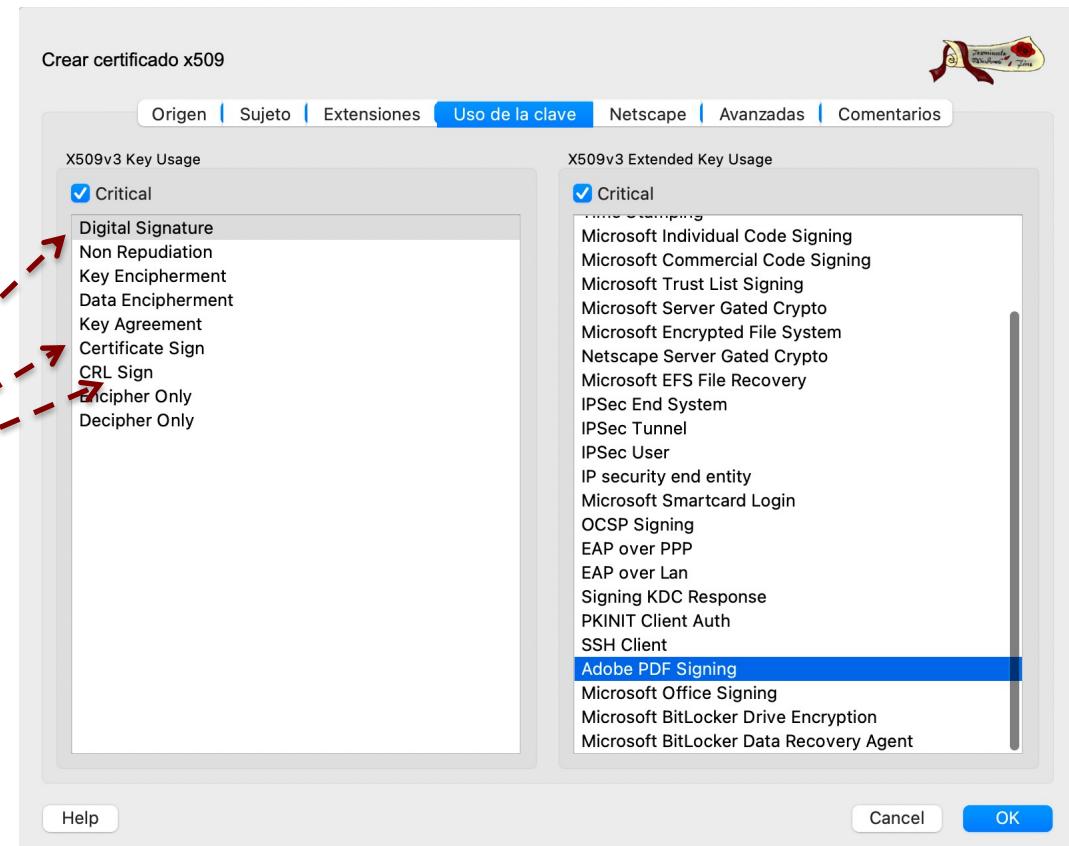
Uso de claves en los certificados digitales

- 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

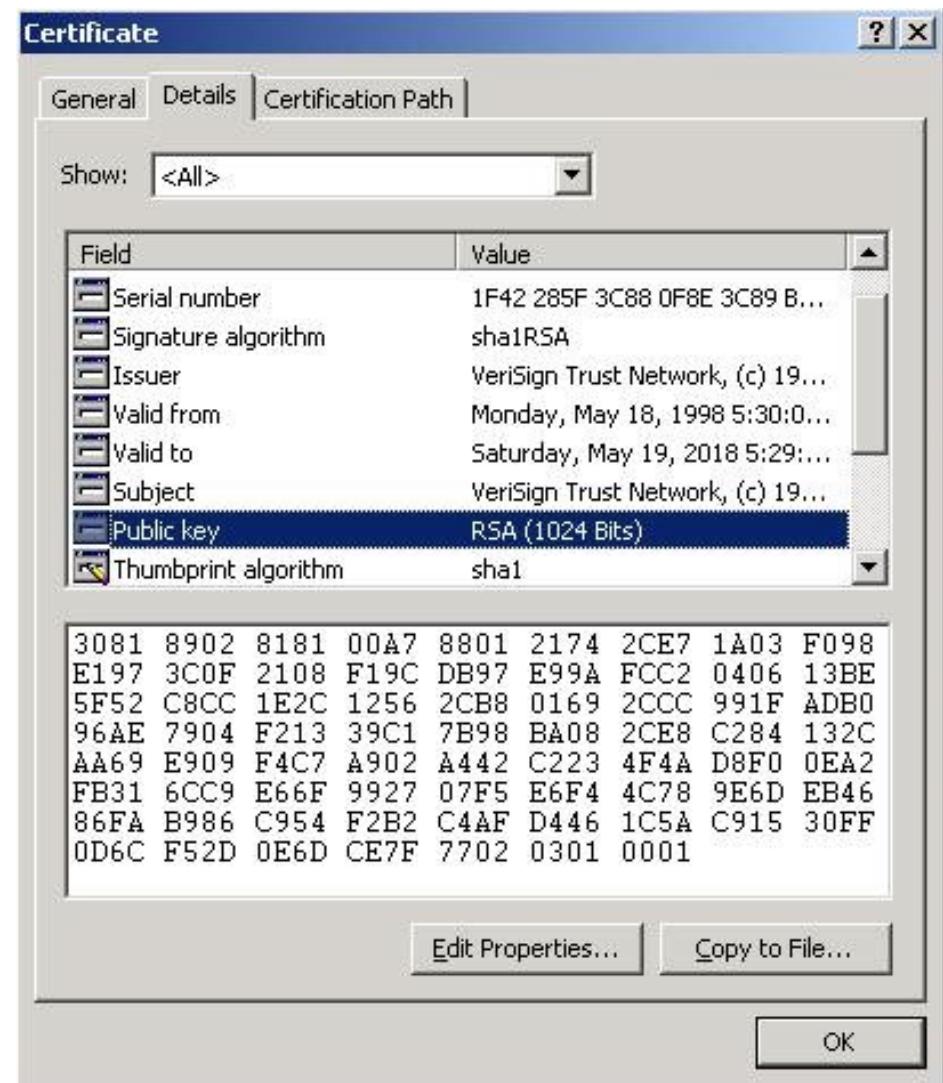
- Supongamos que somos una entidad CA certificando a otra CA
 - ¿Qué tipo de uso de clave se le debería asignar a la nueva CA por defecto?



LUEGO TODA ESTA INFORMACIÓN TAMBIÉN ESTÁ EN EL CERTIFICADO

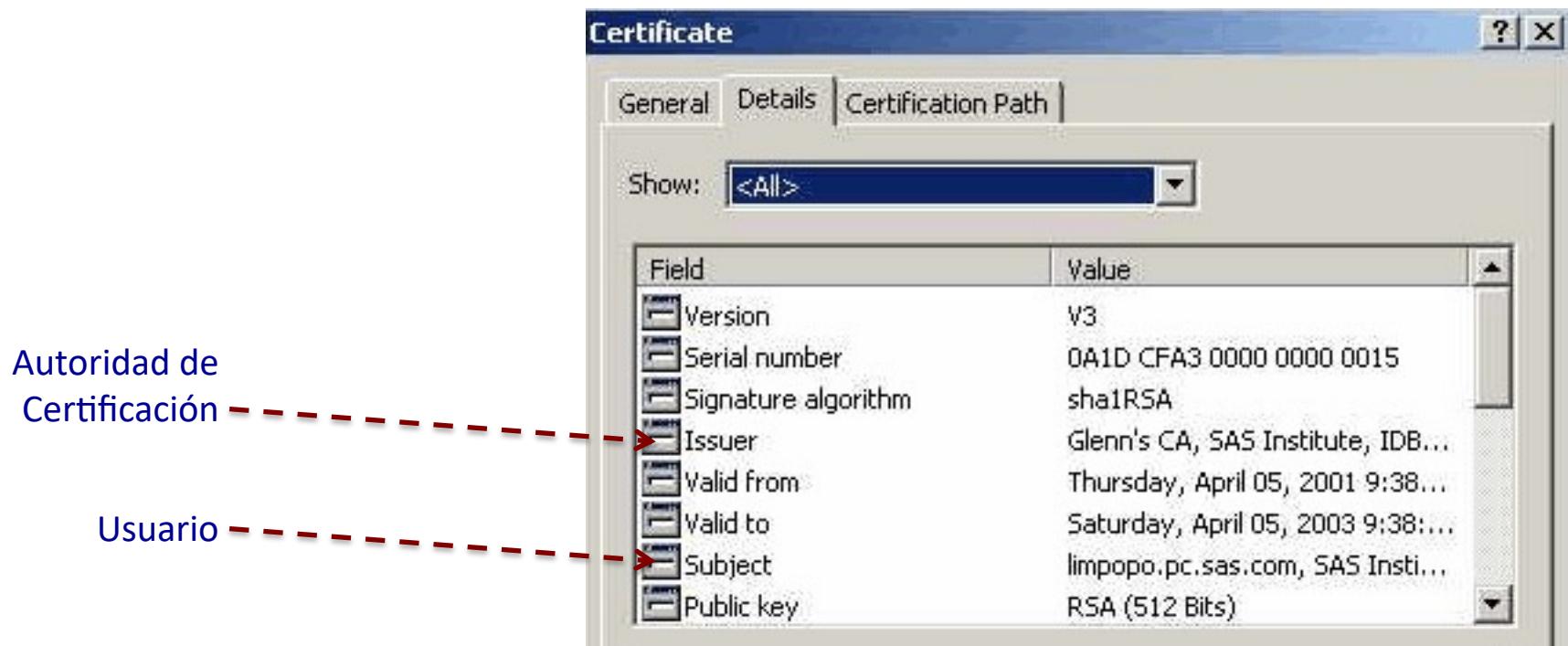
Certificados digitales en navegadores

- 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/>

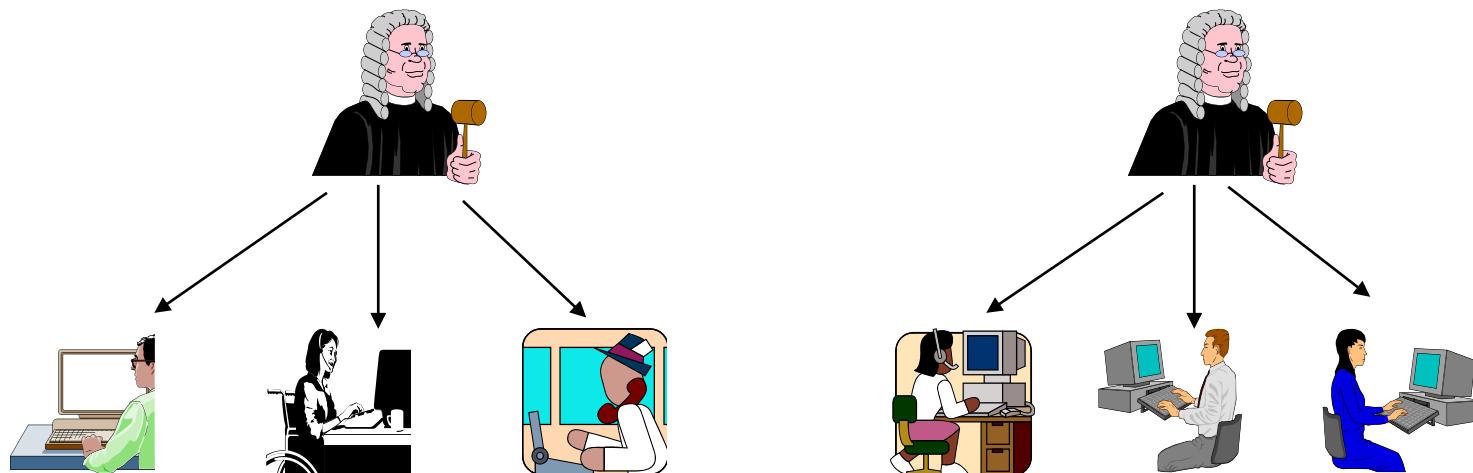
The screenshot shows the official website for Let's Encrypt. At the top left is the Let's Encrypt logo, which consists of a stylized orange lock icon above the text "Let's Encrypt". To the right of the logo is a navigation bar with links for "Documentation", "Get Help", "Donate", "About Us", and "Languages". 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" on the left and "Sponsor" on the right. 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**



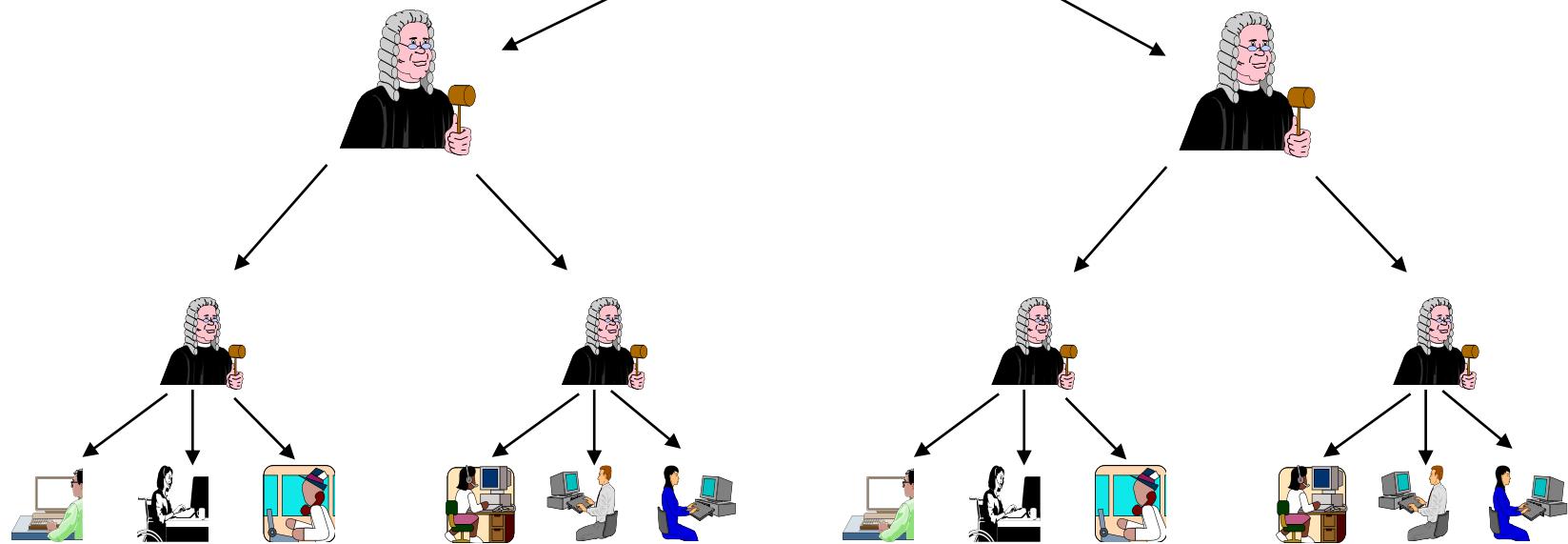
Cadena de confianza



VeriSign®

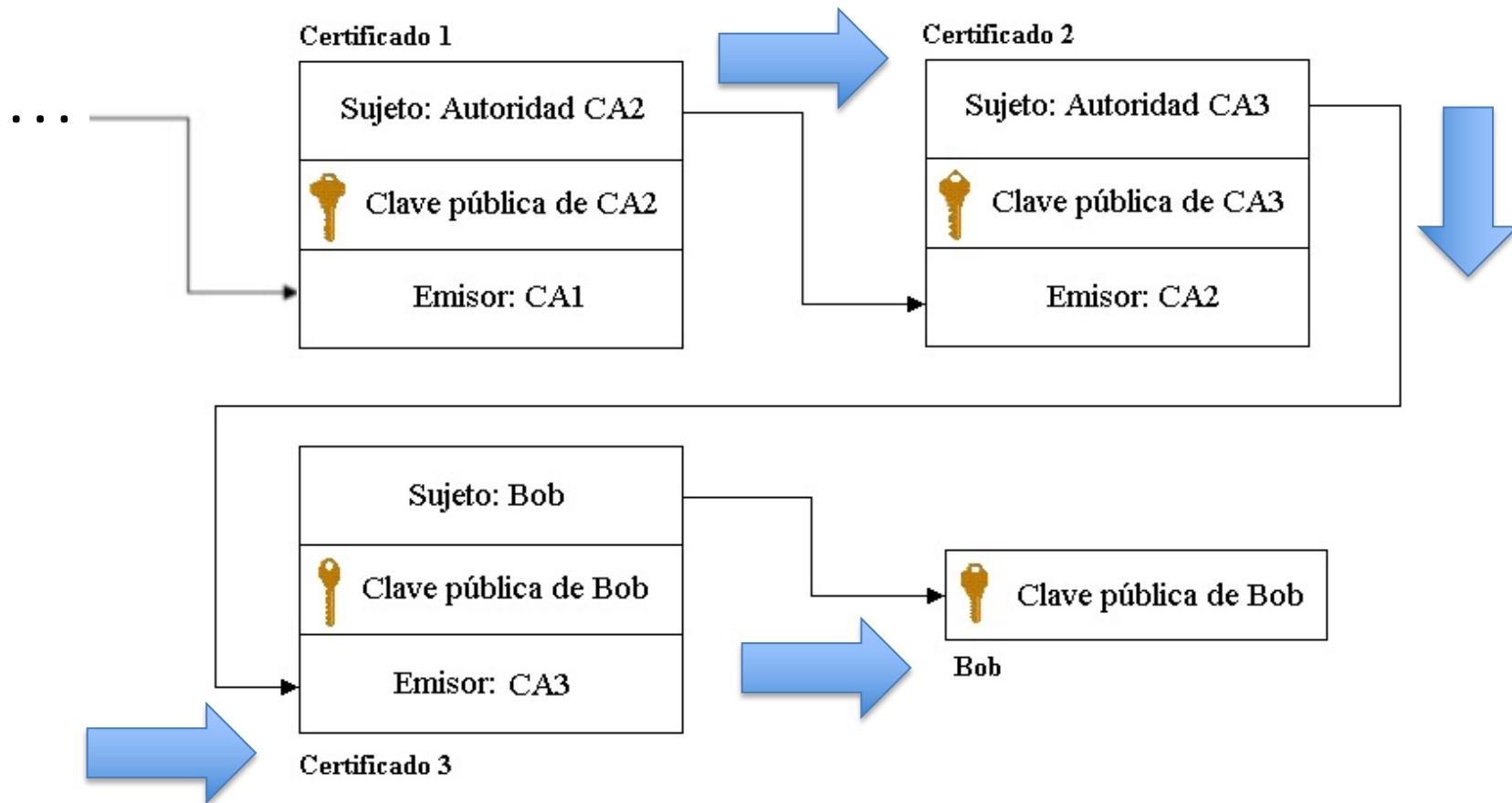


cAcert

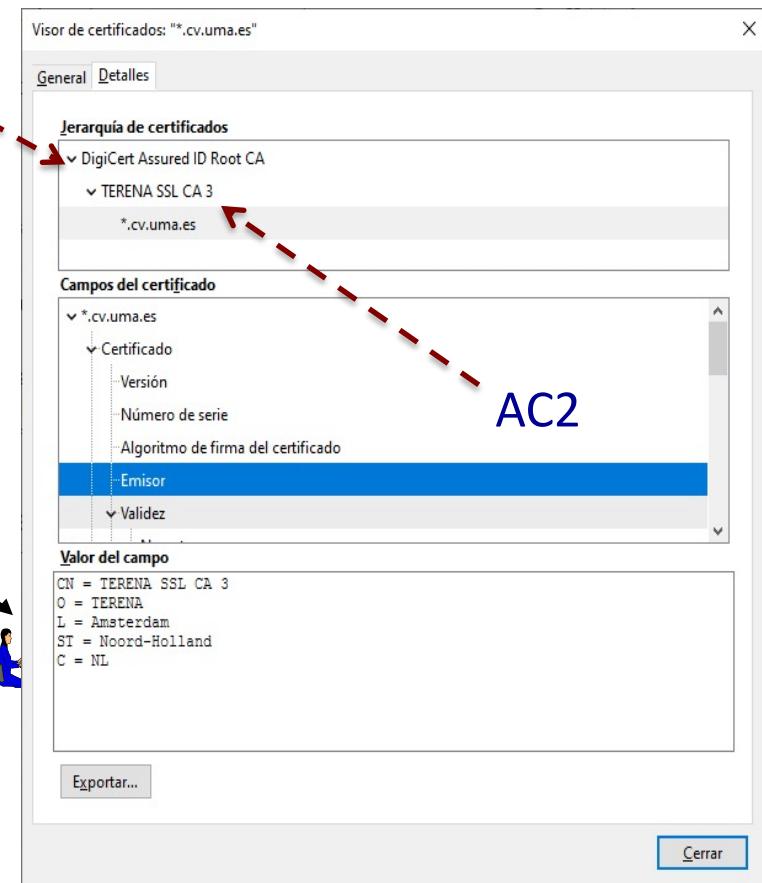
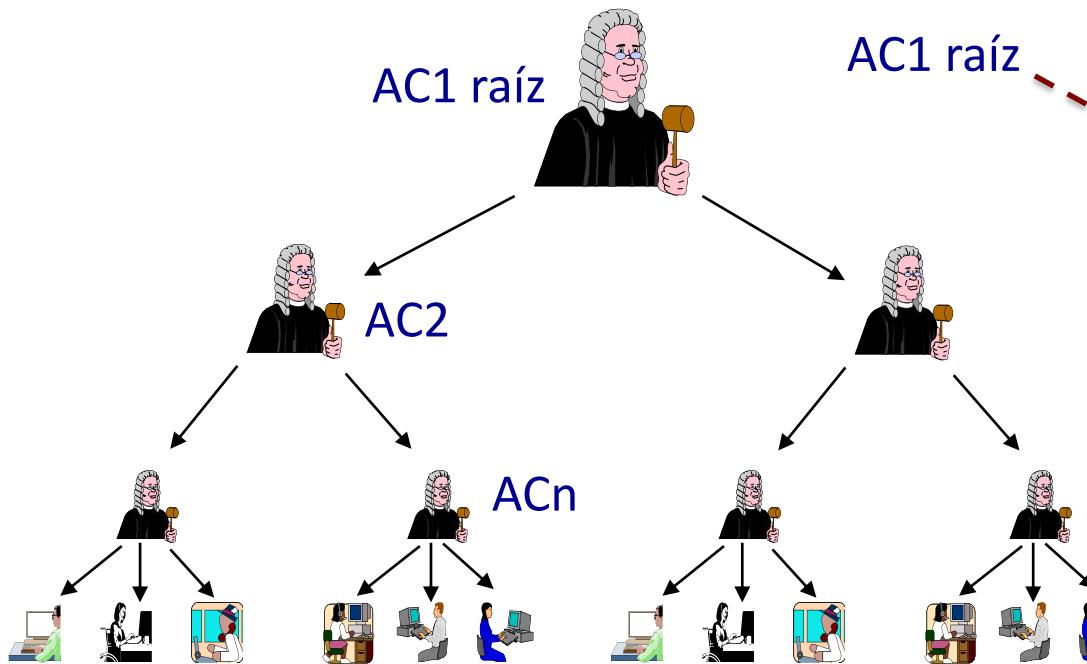


Cadena de confianza

- 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

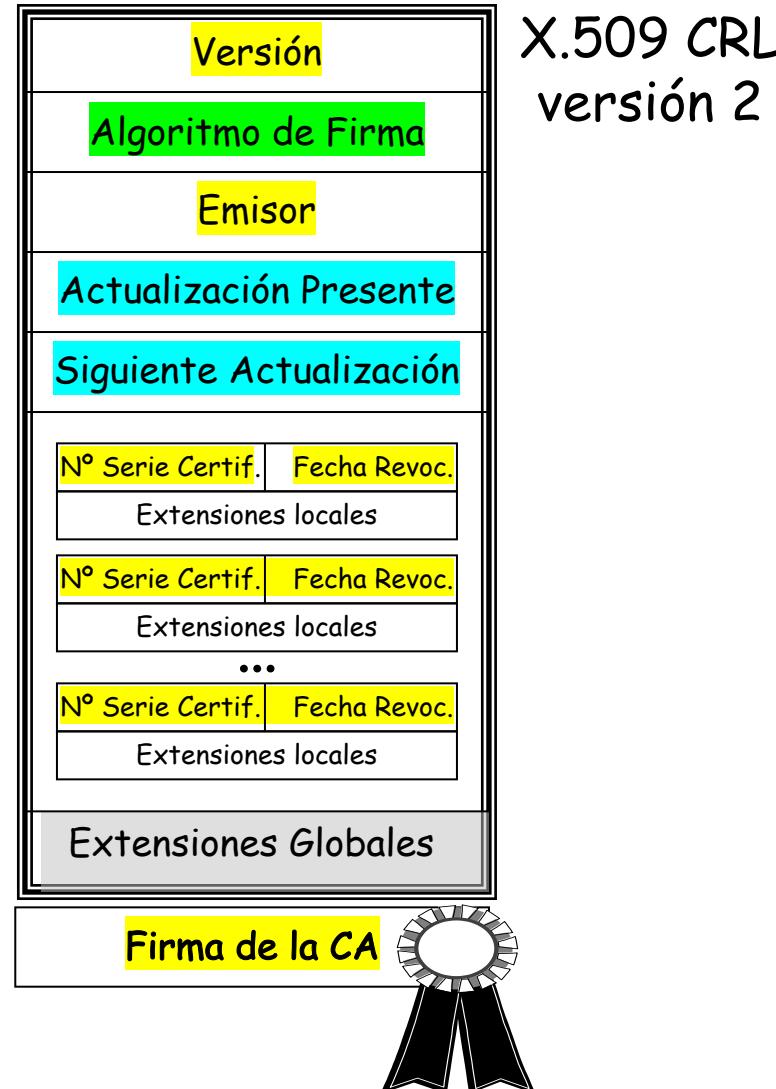
- Revocación de certificados
 - Puede ser recomendable **invalidar** (revocar) un certificado antes de la fecha de expiración cuando:
 - la clave pública deja de ser válida
 - el usuario identificado en el certificado no se considera por más tiempo un usuario con potestad sobre la clave privada correspondiente
 - 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

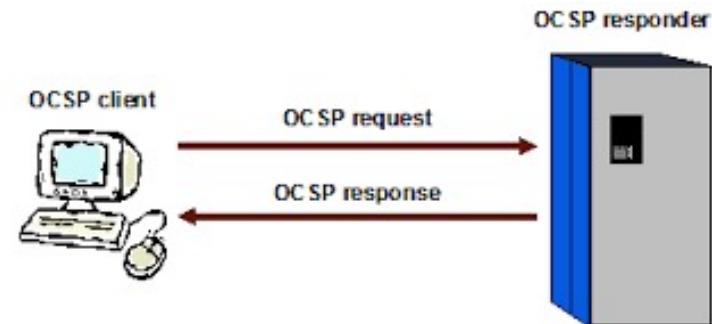
- Estructura de una CRL, según el estándar X.509:



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



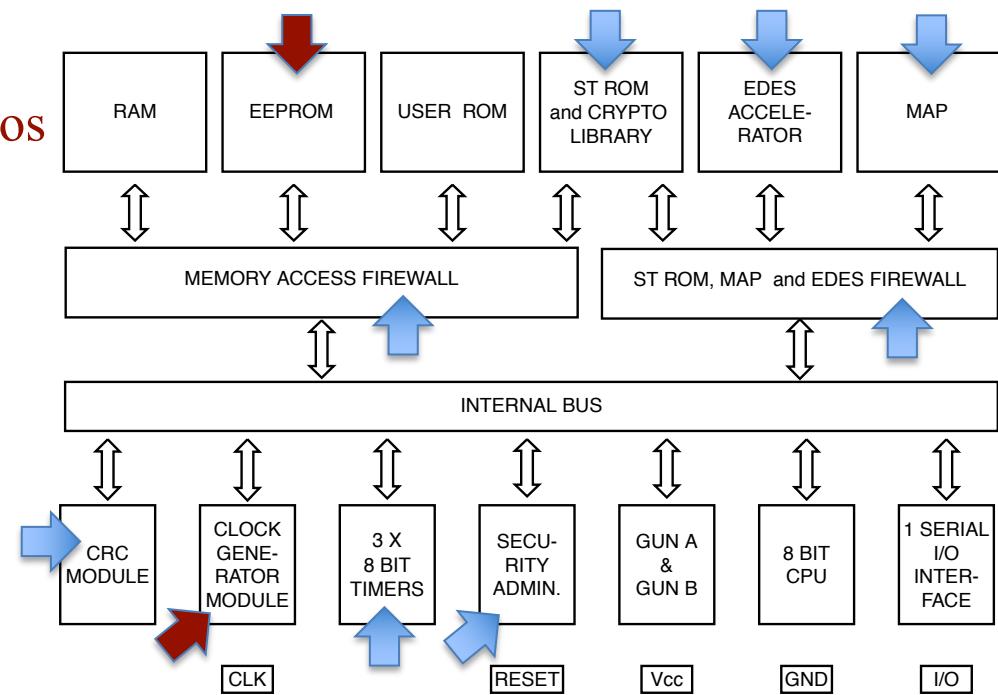
DNI Electrónico (DNI-e)

- 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)

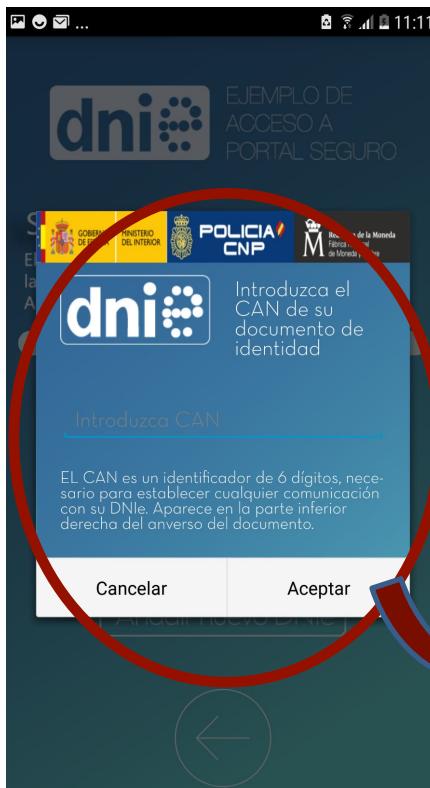
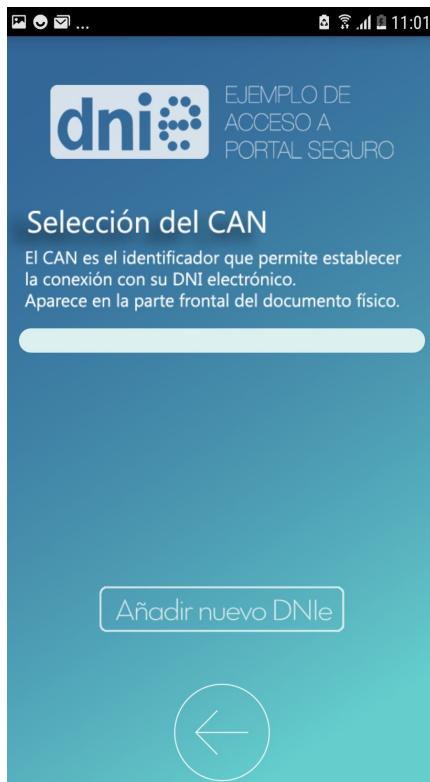
- Dos opciones para acceder al chip



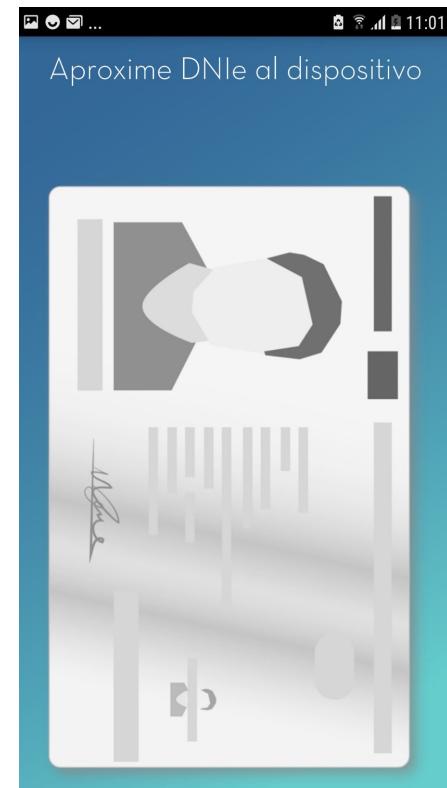
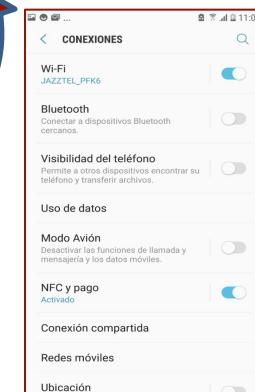
Big buck bunny image: <http://www.bigbuckbunny.org/index.php/about/>
Creative Commons CC by 3.0



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 POLICIA

DNI y Pasaporte

Cuerpo Nacional de Policía

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

POLICÍA NACIONAL

sede electrónica Cuerpo Nacional de Policía

Idiomas ▾ | Inicio | Mapa web | Contacto

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



Real Casa de la Moneda
Fábrica Nacional
de Moneda y Timbre



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

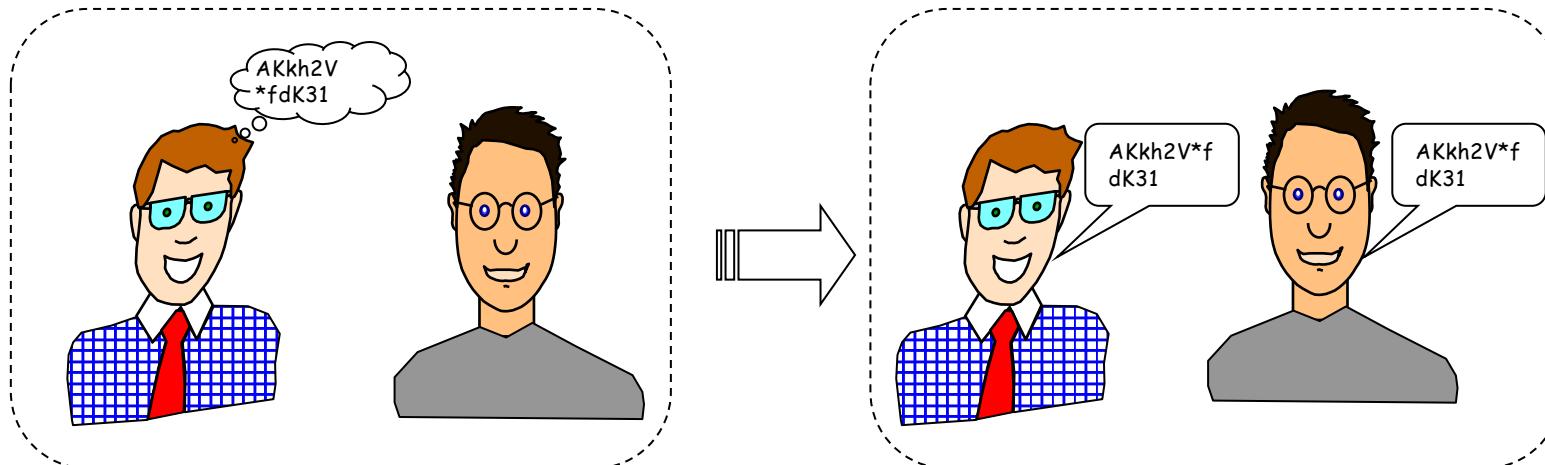


- Algo que yo SOY
 - P.ej. Autenticación biométrica



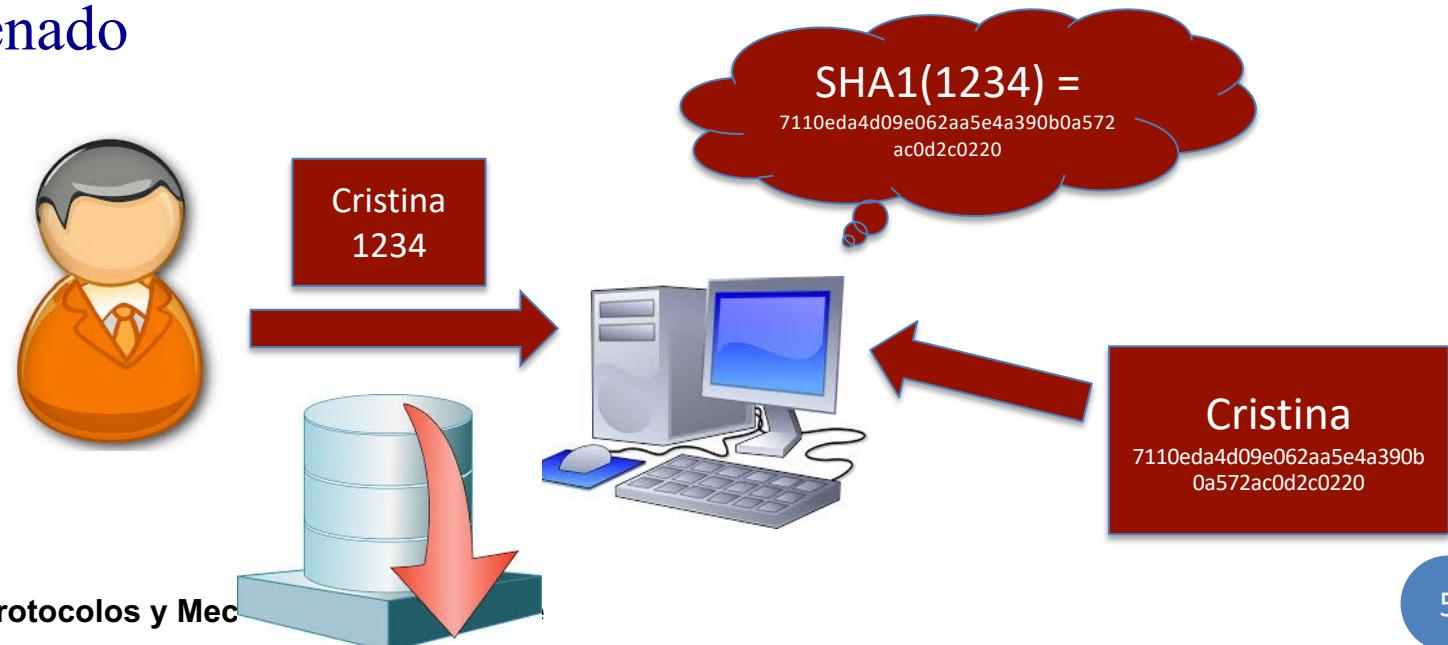
Tipos de autenticación de usuarios - CONOZCO

- 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



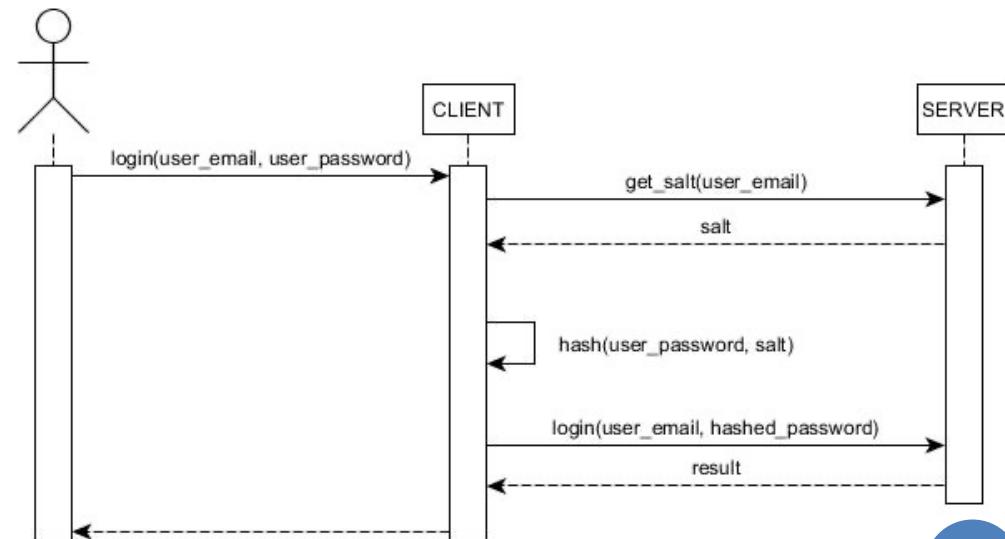
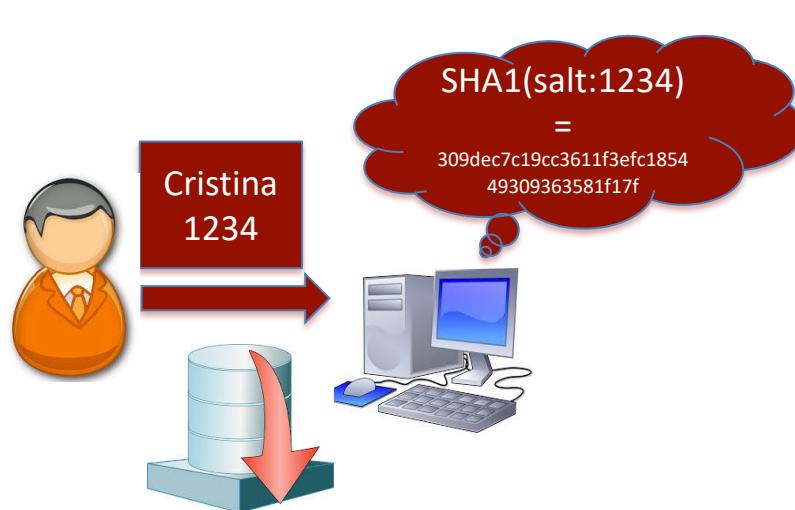
Salt: Contraseñas con Salt

- 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

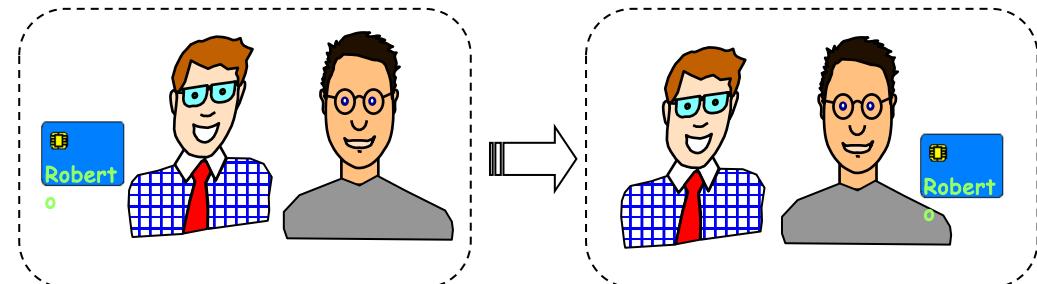
- **bcrypt** es una función hash que combina contraseñas + Salt, y está basada en el cifrado Blowfish
 - Resistente a ataques de fuerza bruta por ser una función hash adaptativa
 - Por implicar el factor tiempo el cual es dependiente de un número de iteraciones que se quieran hacer
 - Cuanto más interacciones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



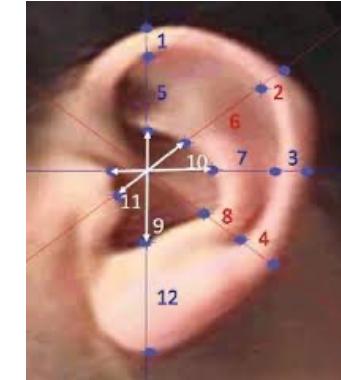
Ejemplo: DNI Electrónico (DNI-e)

- 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

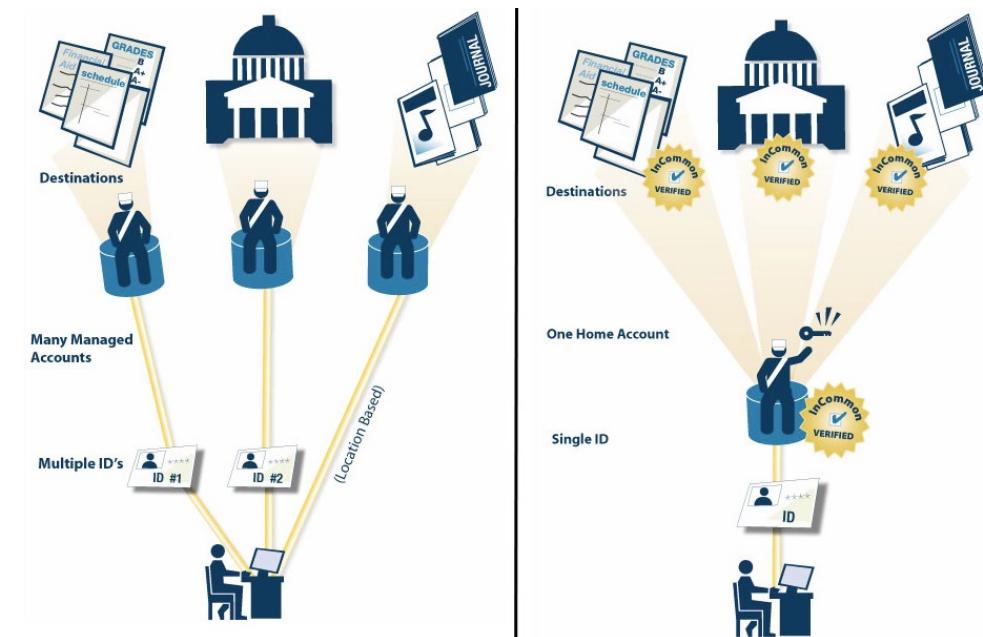
Autenticación basado en códigos QR

- 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



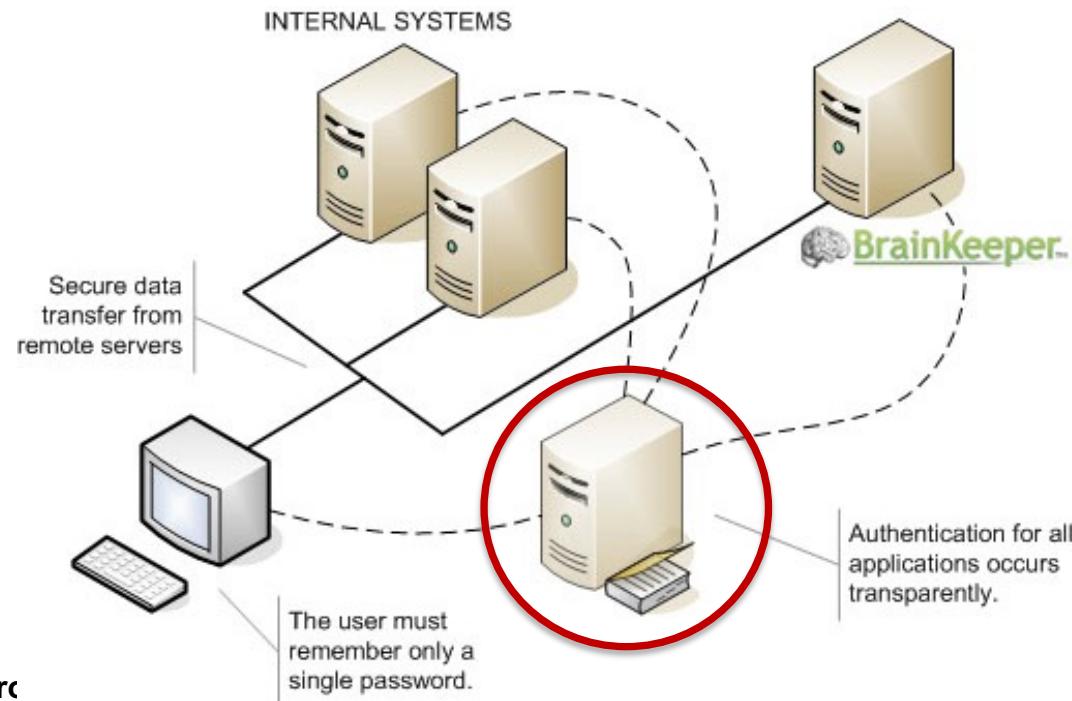
Single Sign-On (SSO)

- 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



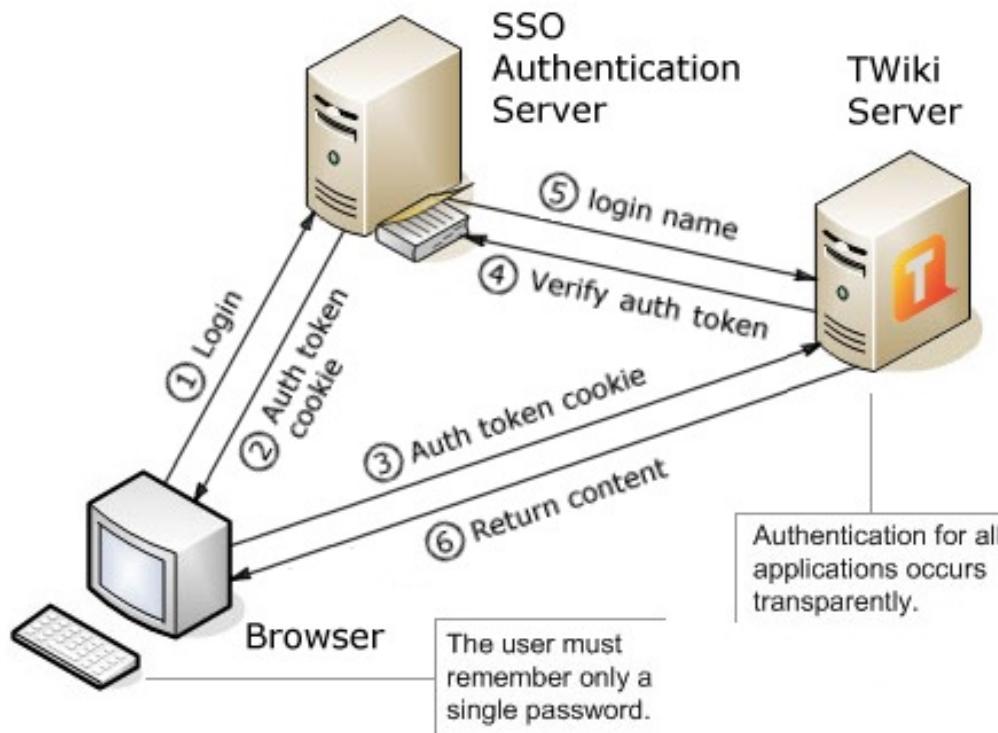
Single Sign-On (SSO)

- 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

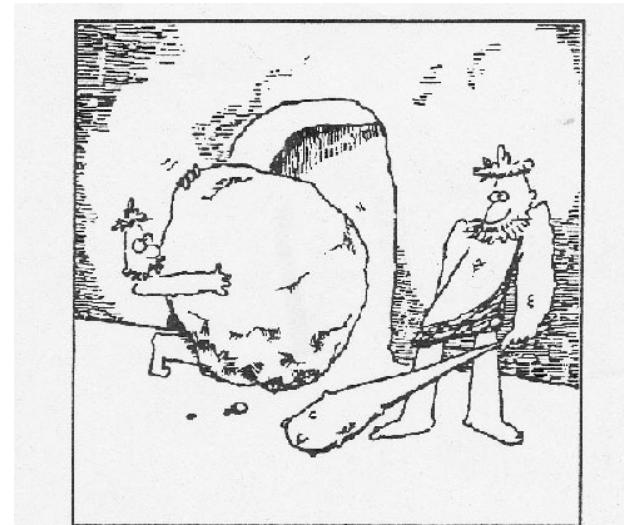


Single Sign-On (SSO)

- 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



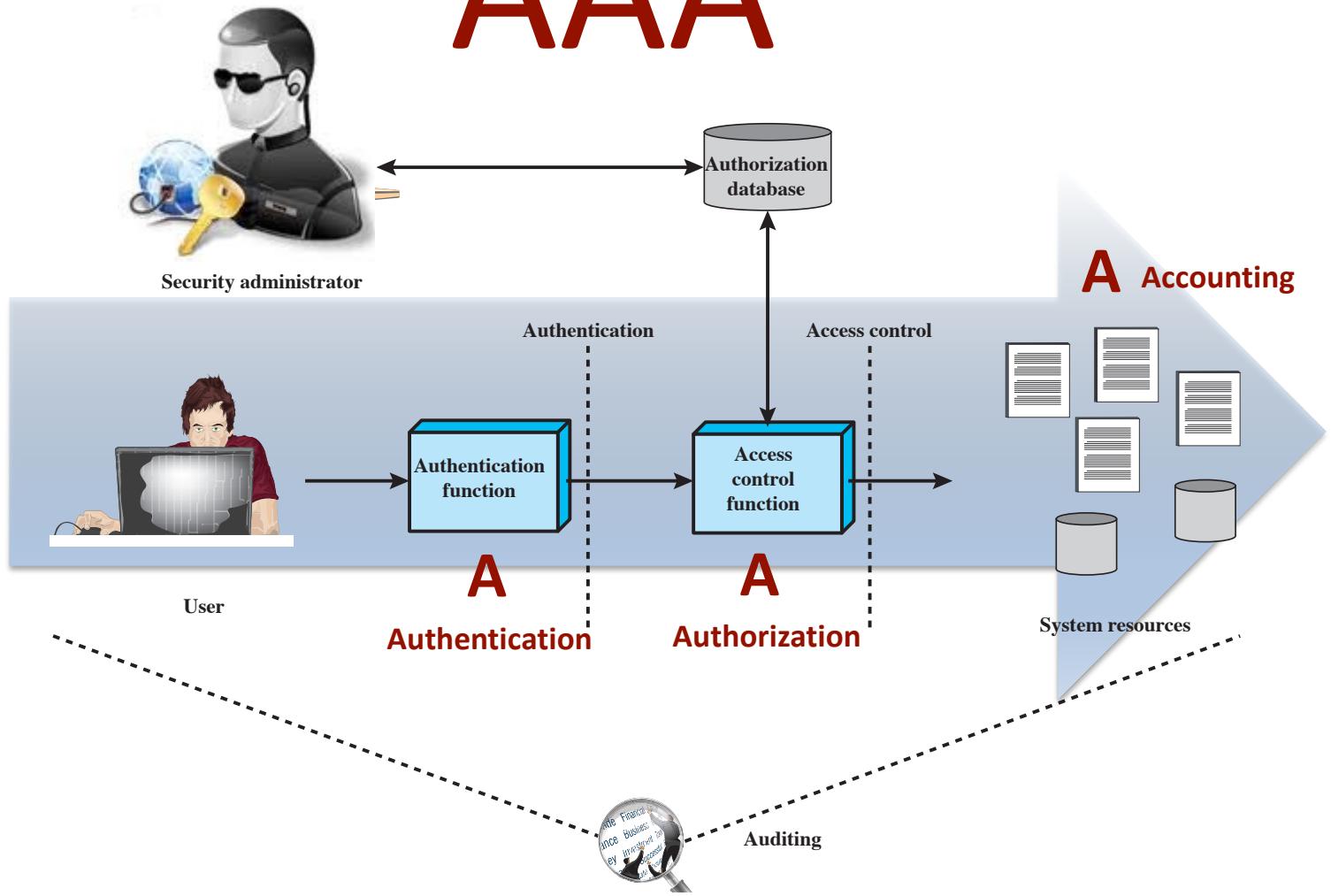
Control de acceso

- 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



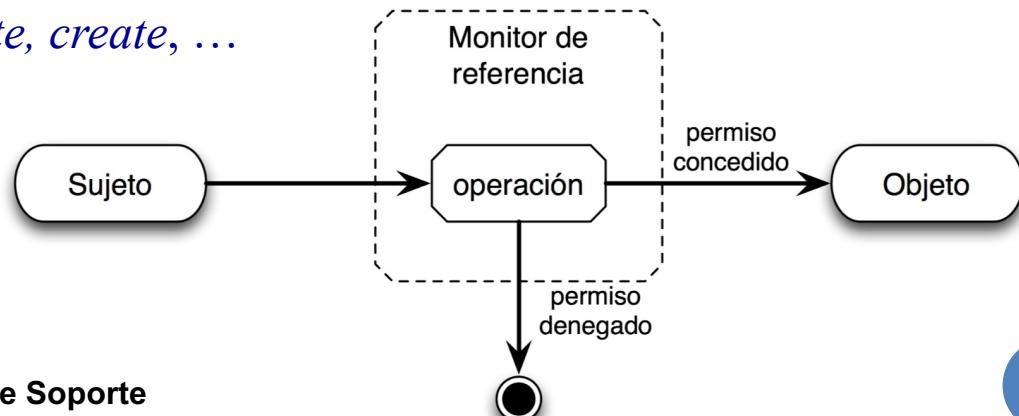
Control de acceso

- 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, ...*

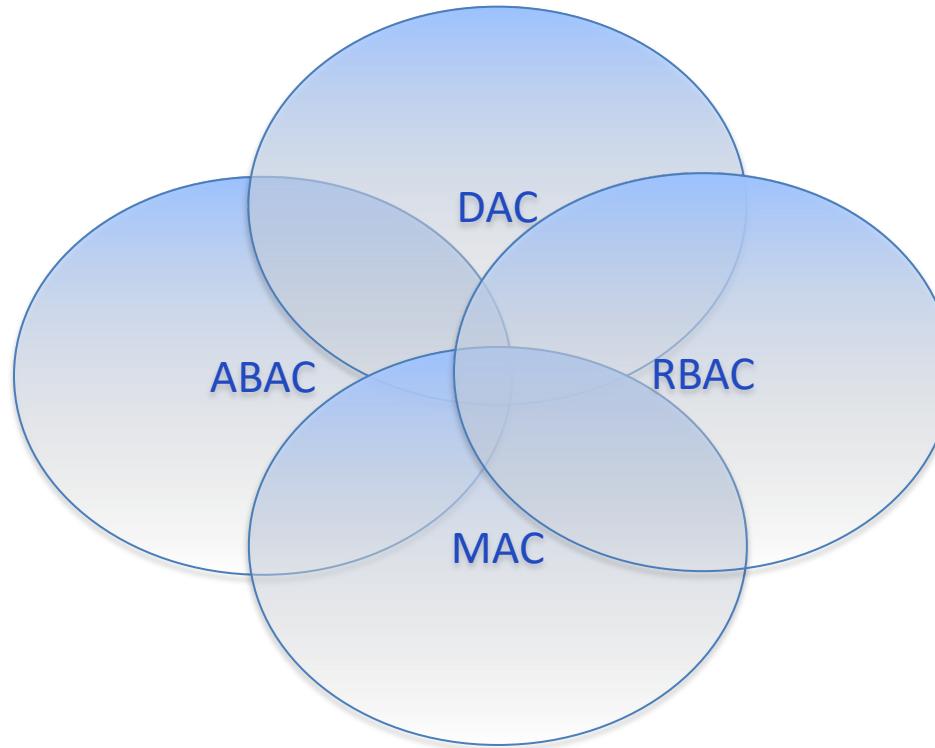


Mecanismos de control de acceso

- 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
 - ...

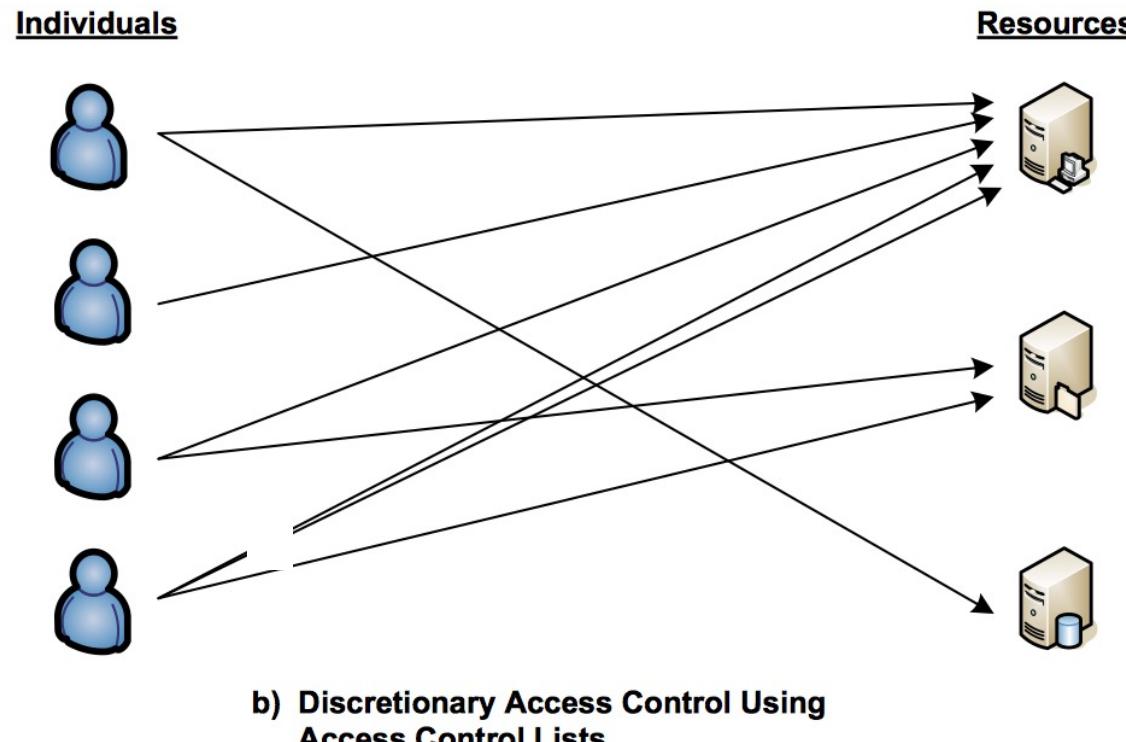
Mecanismos de control de acceso

- 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



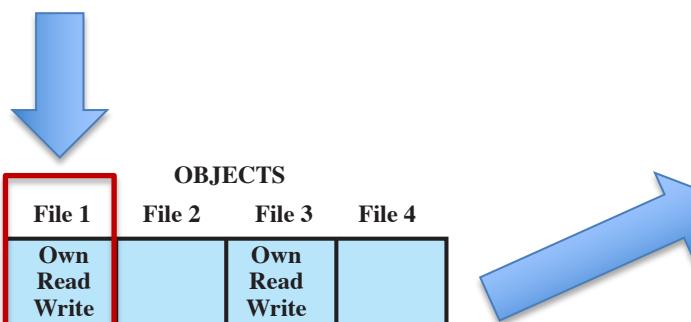
DAC (Discretionary Access Control)

- 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

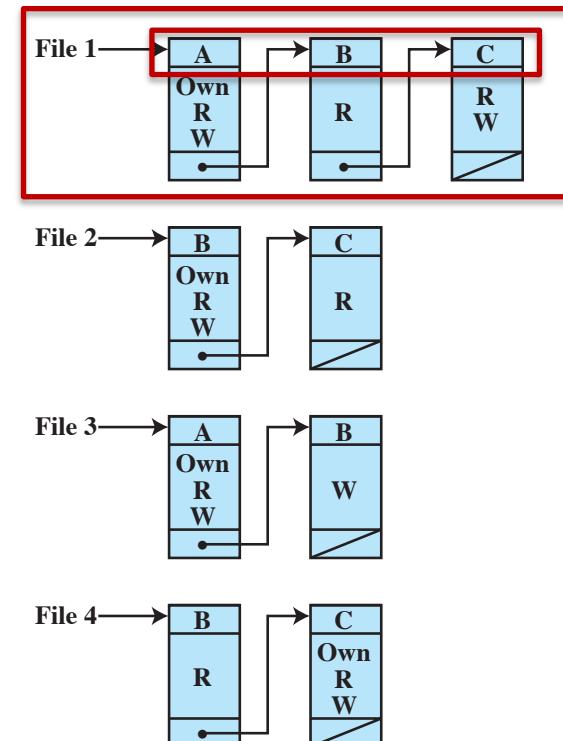
		OBJECTS			
		File 1	File 2	File 3	File 4
SUBJECTS	User A	Own Read Write		Own Read Write	
	User B	Read	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write

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

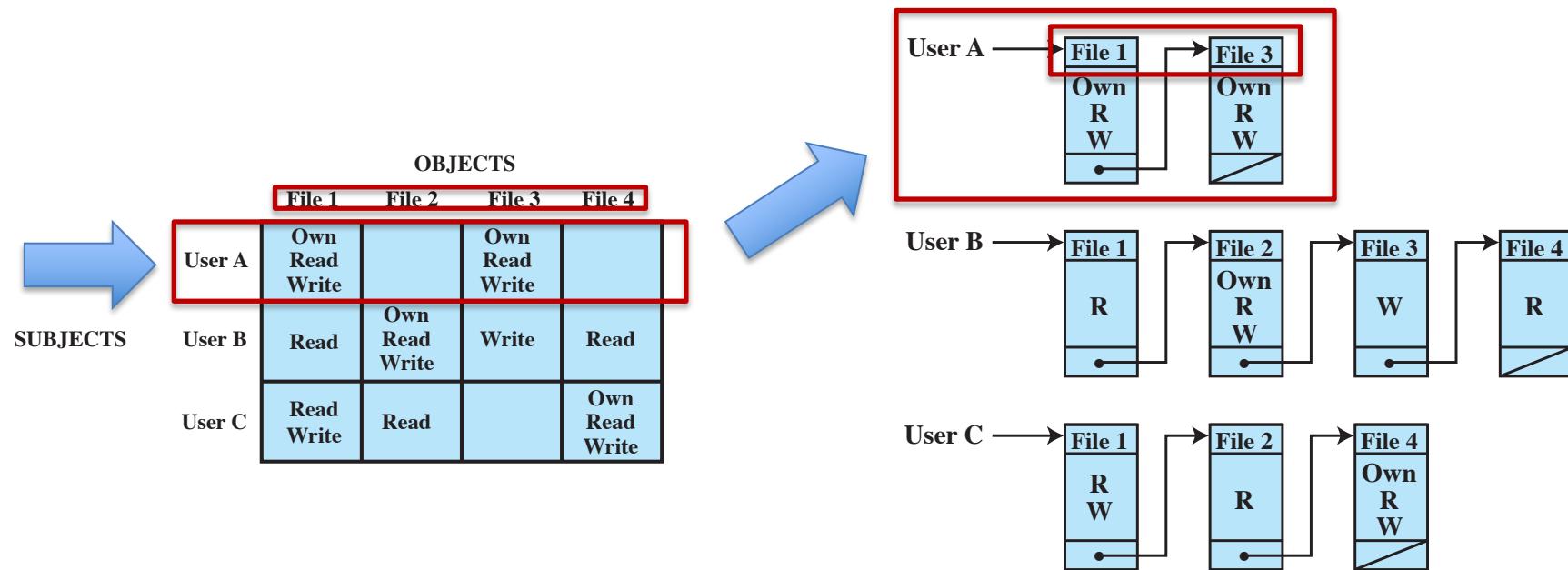


		OBJECTS				
		File 1	File 2	File 3	File 4	
SUBJECTS		User A	Own Read Write		Own Read Write	
		User B	Read	Own Read Write	Write	Read
		User C	Read Write	Read		Own Read Write



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

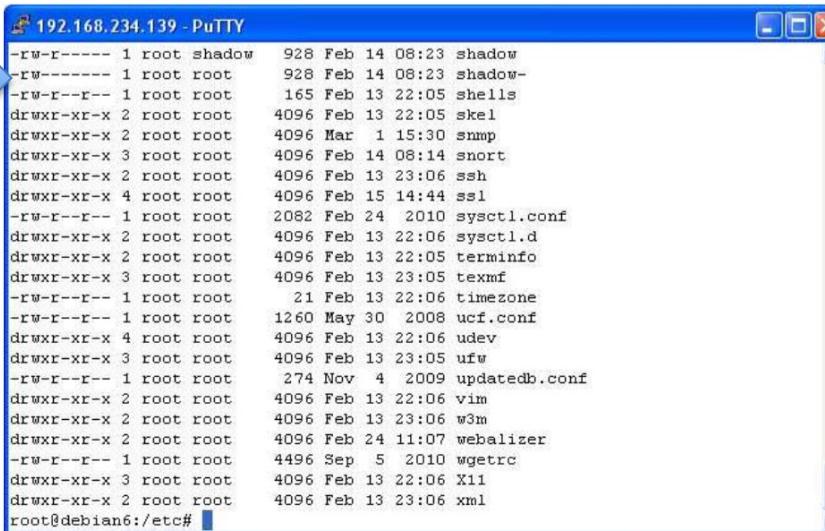
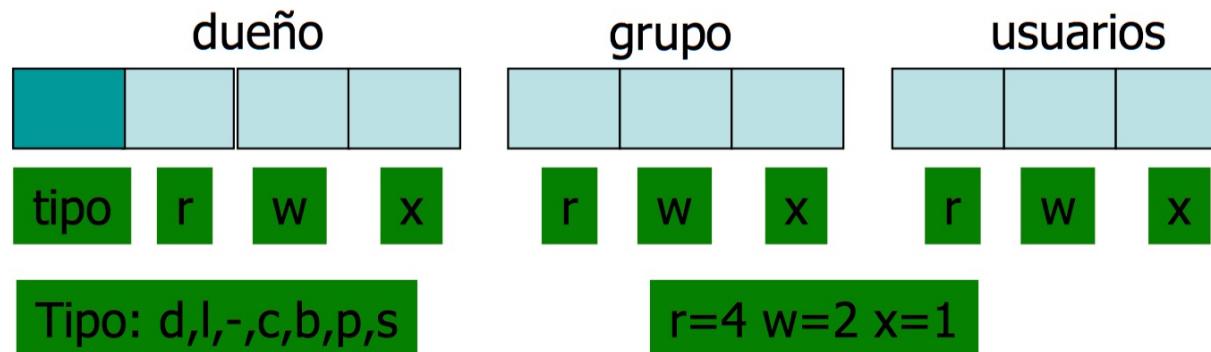
DAC (Discretionary Access Control)

- **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)

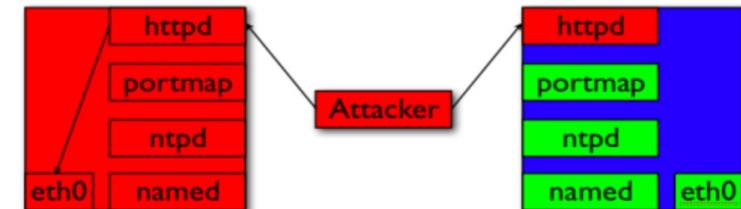
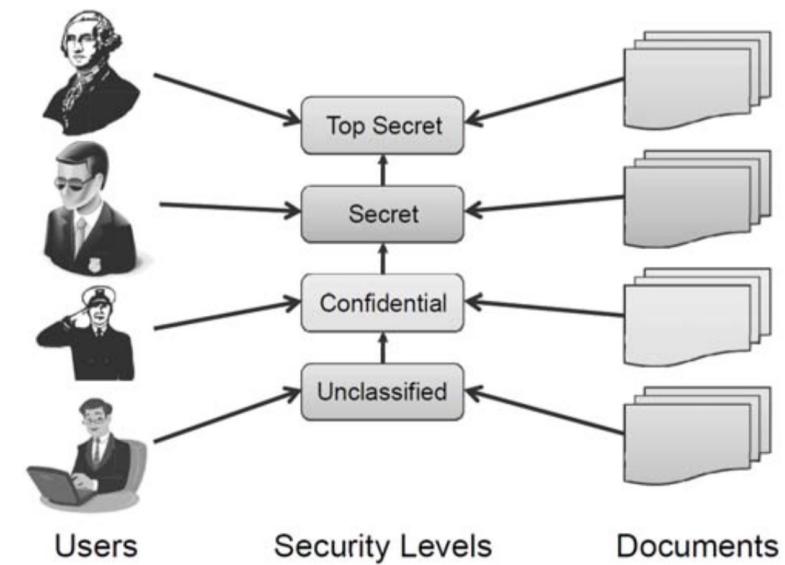
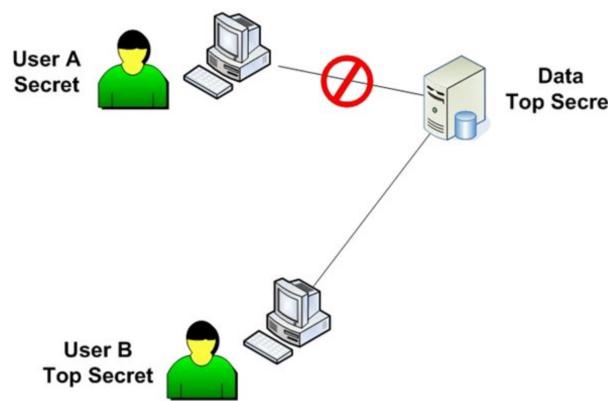
- Ejemplo de permisos básicos en UNIX, usando DAC



```
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)

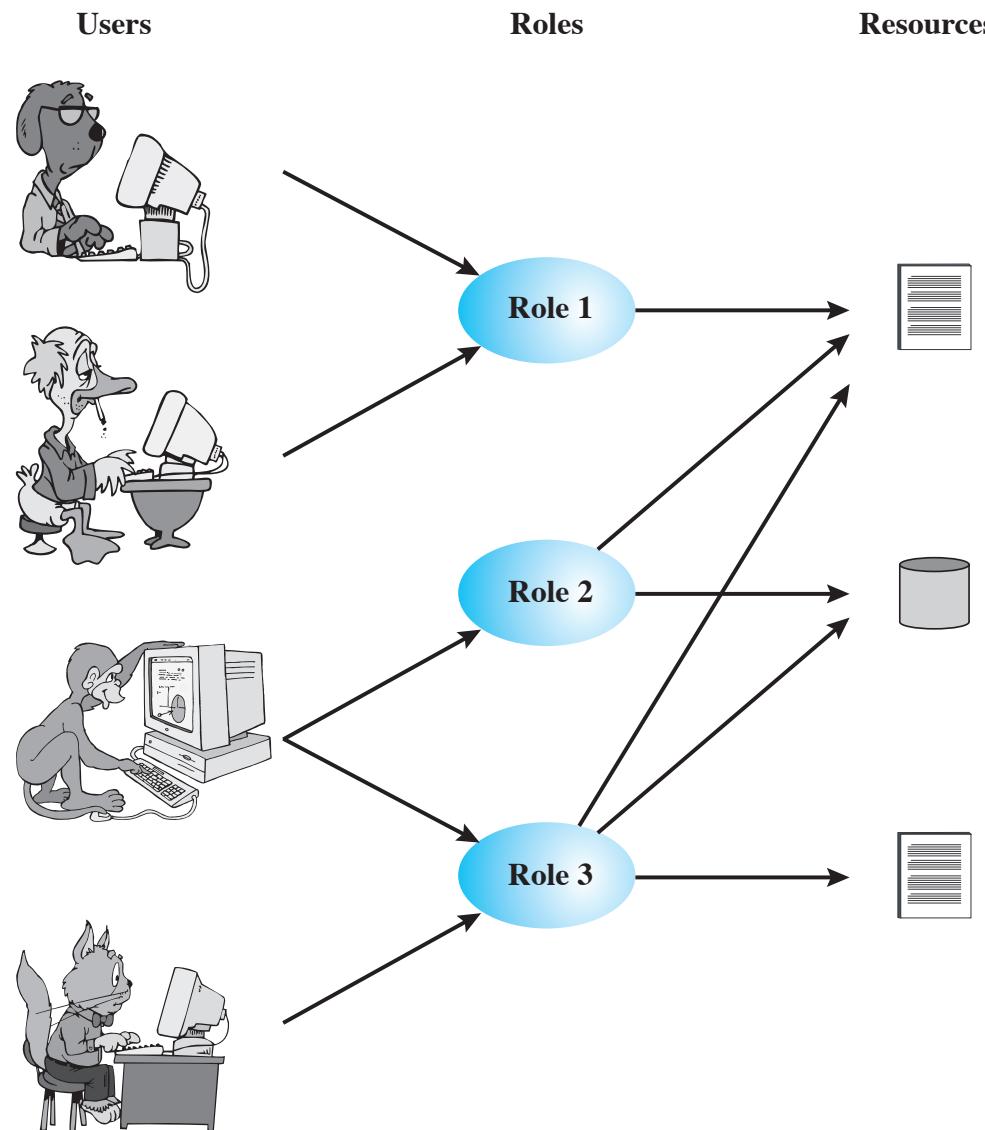
- Se basa en comparar etiquetas de seguridad (que indican la criticidad de los recursos) con las autorizaciones de seguridad (que indican las entidades que pueden acceder a ciertos recursos)
- Por lo tanto, a cada recurso se le asigna una etiqueta de seguridad, que de alguna forma lo clasifica
 - top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

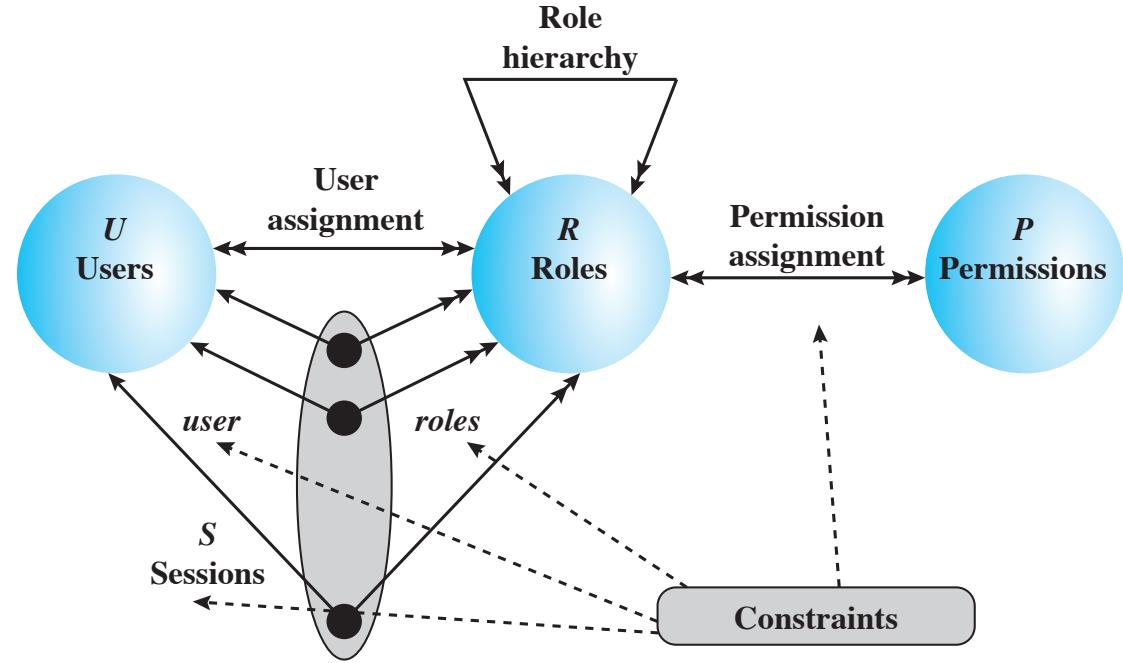
	R ₁	R ₂	• • •	R _n
U ₁	X			
U ₂	X			
U ₃		X		X
U ₄				X
U ₅				X
U ₆				X
•				
U _m	X			

	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



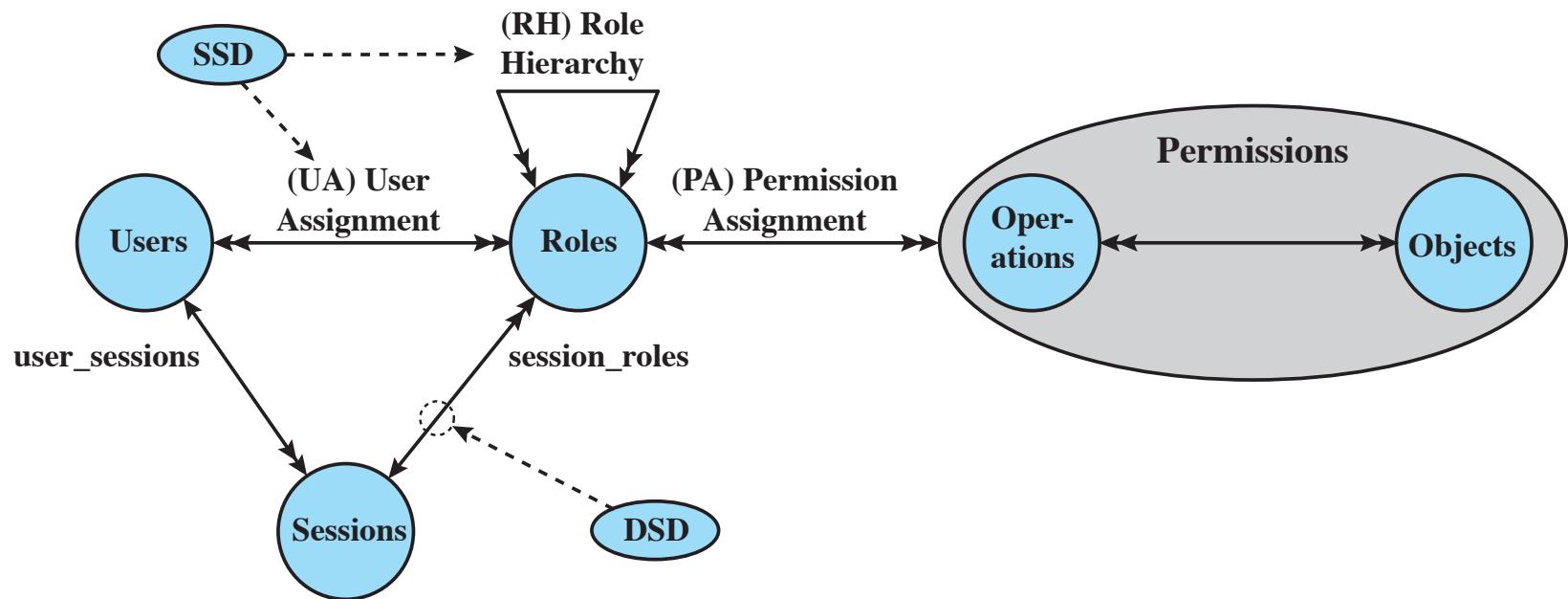
- Las relaciones muchos-a-muchos entre usuarios y roles, y entre roles y permisos, proporcionan **flexibilidad y granularidad** a la hora de gestionar las condiciones de acceso a un usuario
 - Tal una gestión que no ocurre con DAC

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)

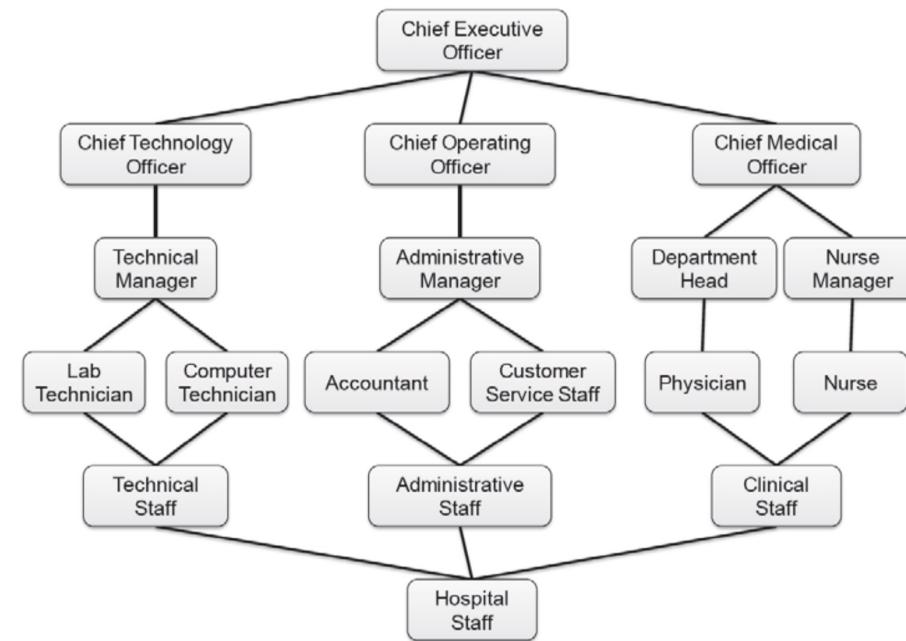
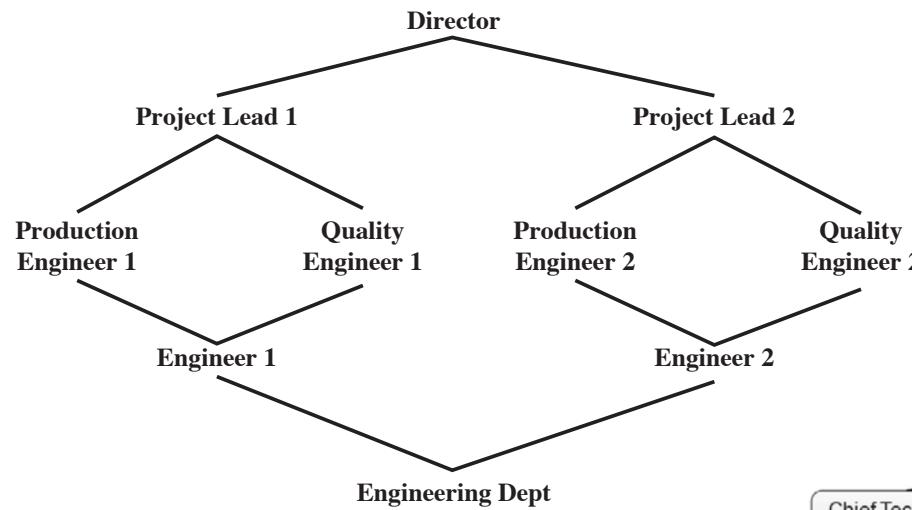
- El modelo **RBAC estándar de NIST** introduce algunas extensiones al RBAC tradicional, como:
 - **Static Separation of Duties (SSD)**: para definir roles mutuamente excluyentes
 - **Dynamic Separation of Duties (DSD)**: para definir restricciones sobre los roles que un usuario puede activar en una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

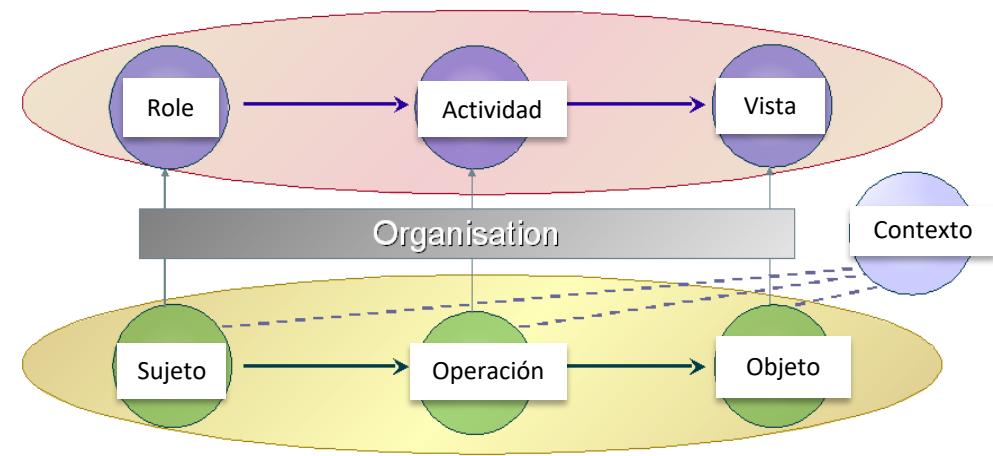
- CapBAC basa su modelo en un conjunto de **roles** y **atributos**,
 - Atributos → capacidades
 - Roles → funciones
- El acceso sólo es posible si el usuario recibe del proveedor del recurso un **token** (un “certificado de autorización”) que demuestra su “capability” para realizar determinadas acciones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

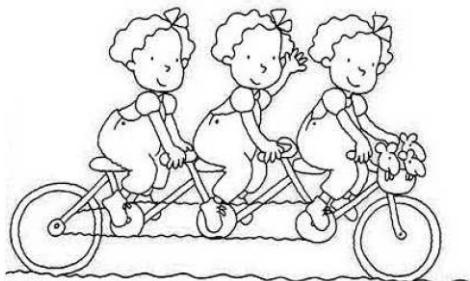


Protocolos criptográficos avanzados



- 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



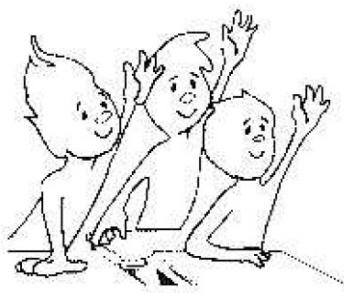
Canal Subliminal



Lanzamiento de monedas



Efecto Mente



Protocolos
Criptográficos



Firma de Contratos



Transferencia
Inconsciente



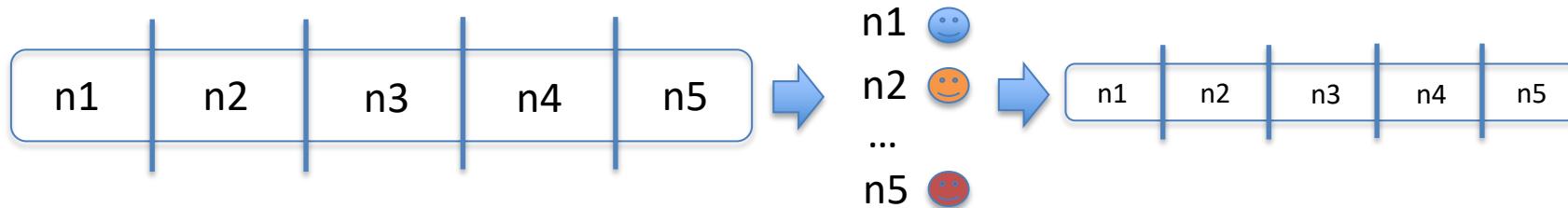
Demostraciones de
Conocimiento Nulo
Criptografía-ULL

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

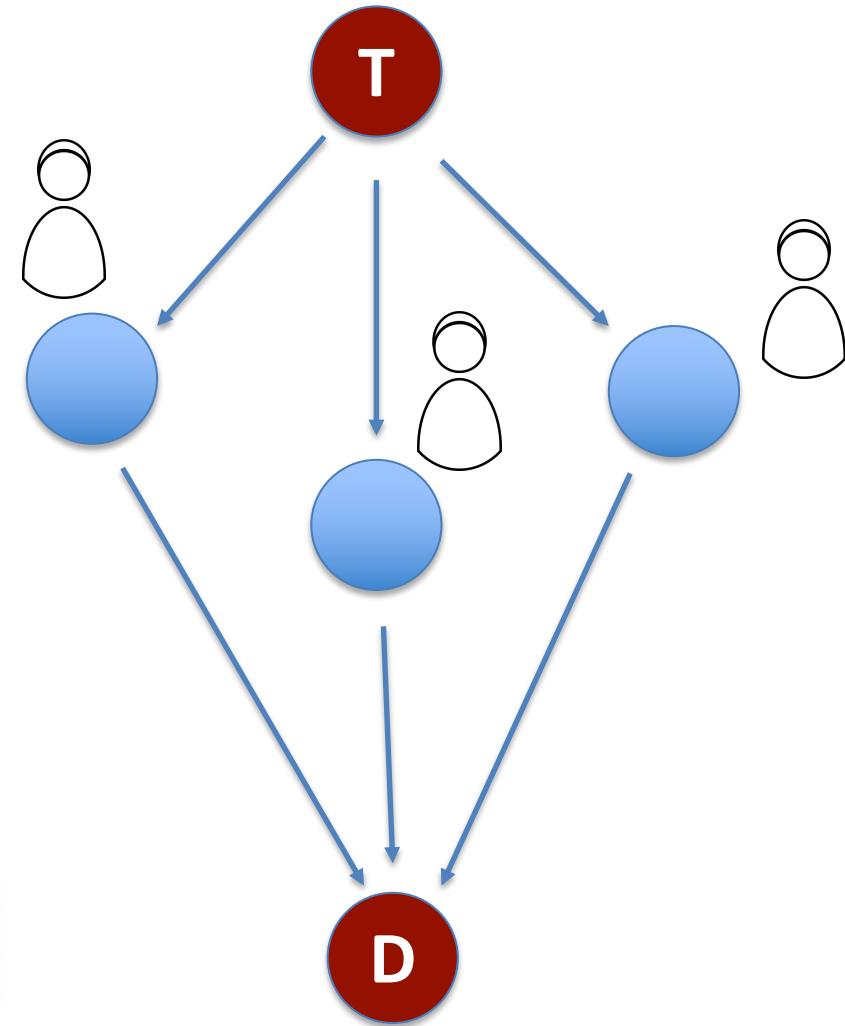


Protocolo de división de secretos - ejemplo 1

- 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:
$$q(x_i) = y_i, \forall i$$
- 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

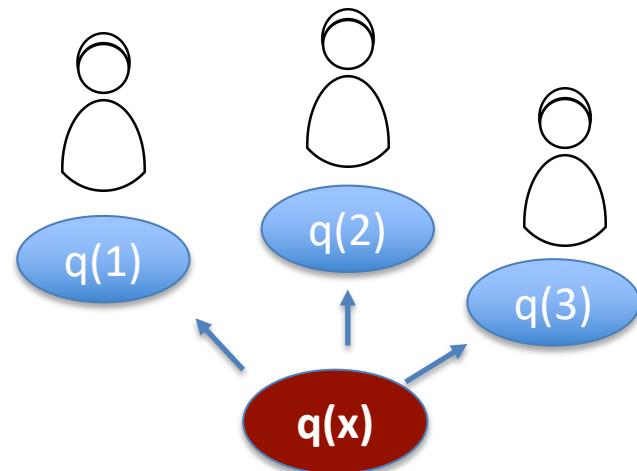
- Más concretamente, se eligen aleatoriamente los coeficientes de un polinomio de grado $k-1$:

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

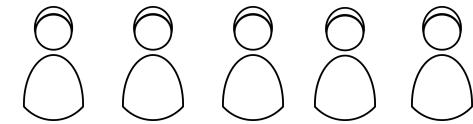
donde $\mathbf{a}_0 = \mathbf{D}_0 = \mathbf{D}$ y evaluamos:

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

- 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:

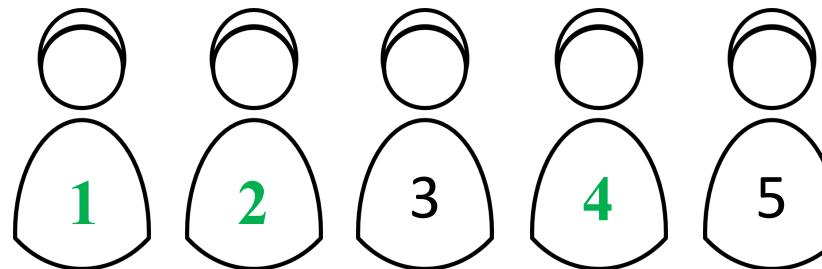
$$q(1) = 14 \rightarrow Alice$$

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

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

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

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



- 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

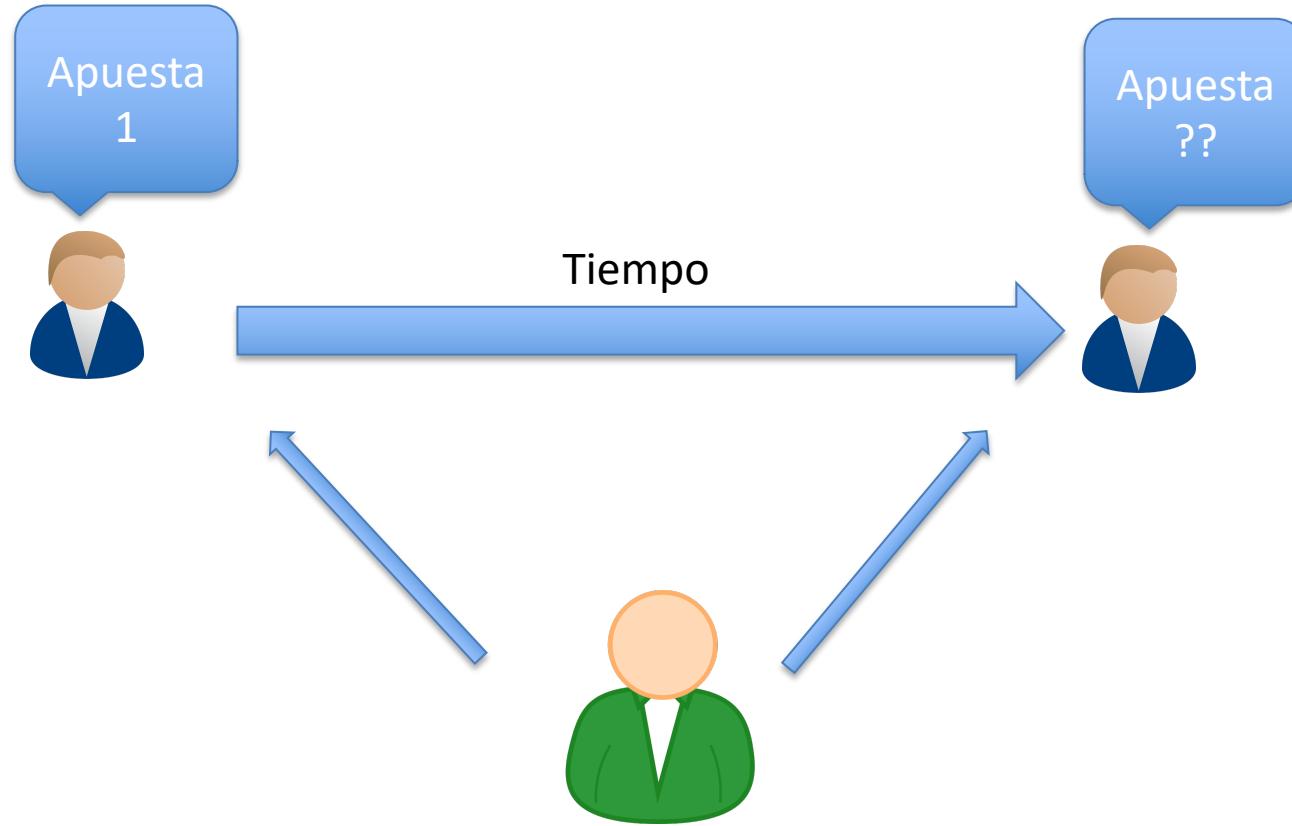
Protocolos de bit-commitment

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



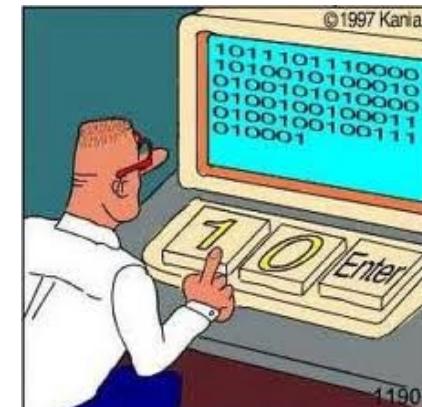
Protocolos de bit-commitment



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

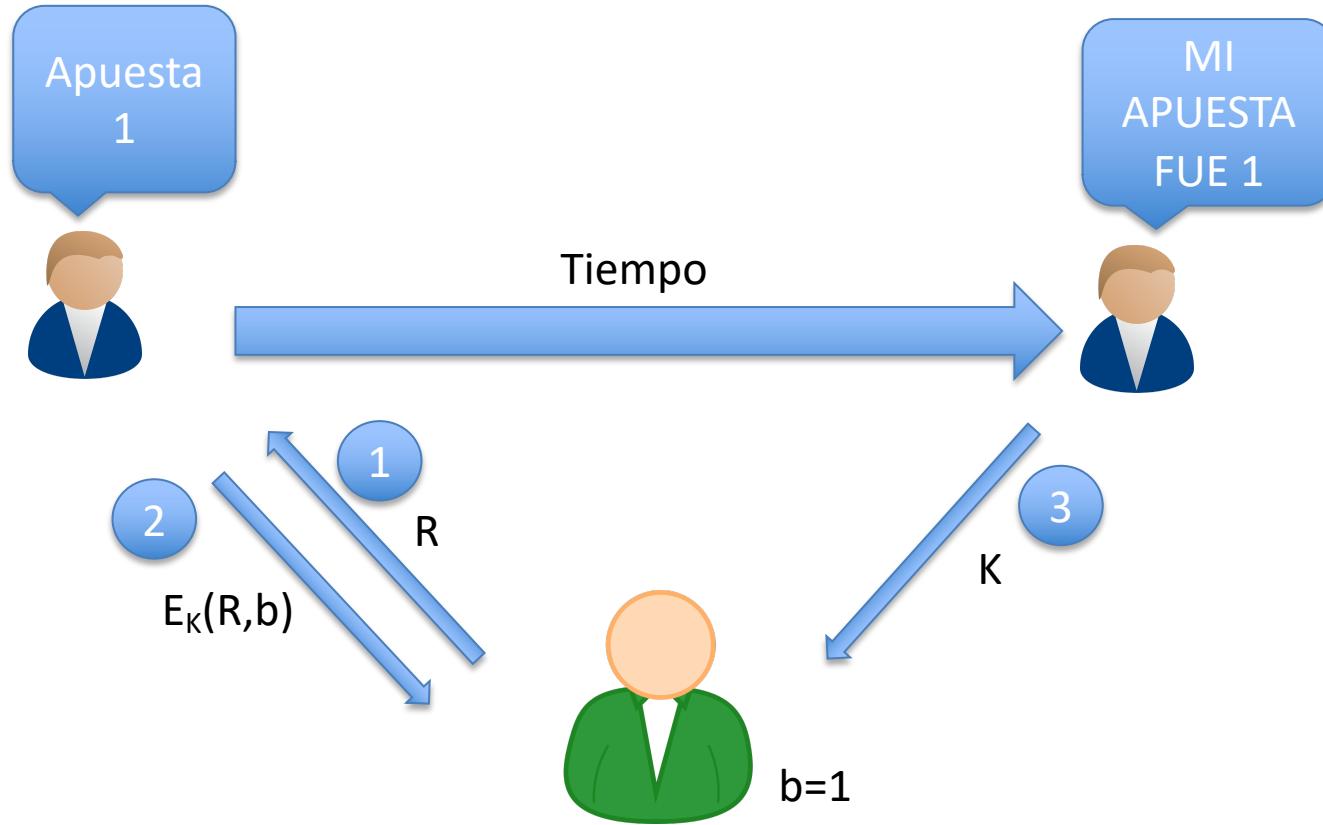
③ Alice envía a Bob la clave K

$$Alice \rightarrow Bob: K$$

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



Bob debe asegurar que Alice no hace trampa

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

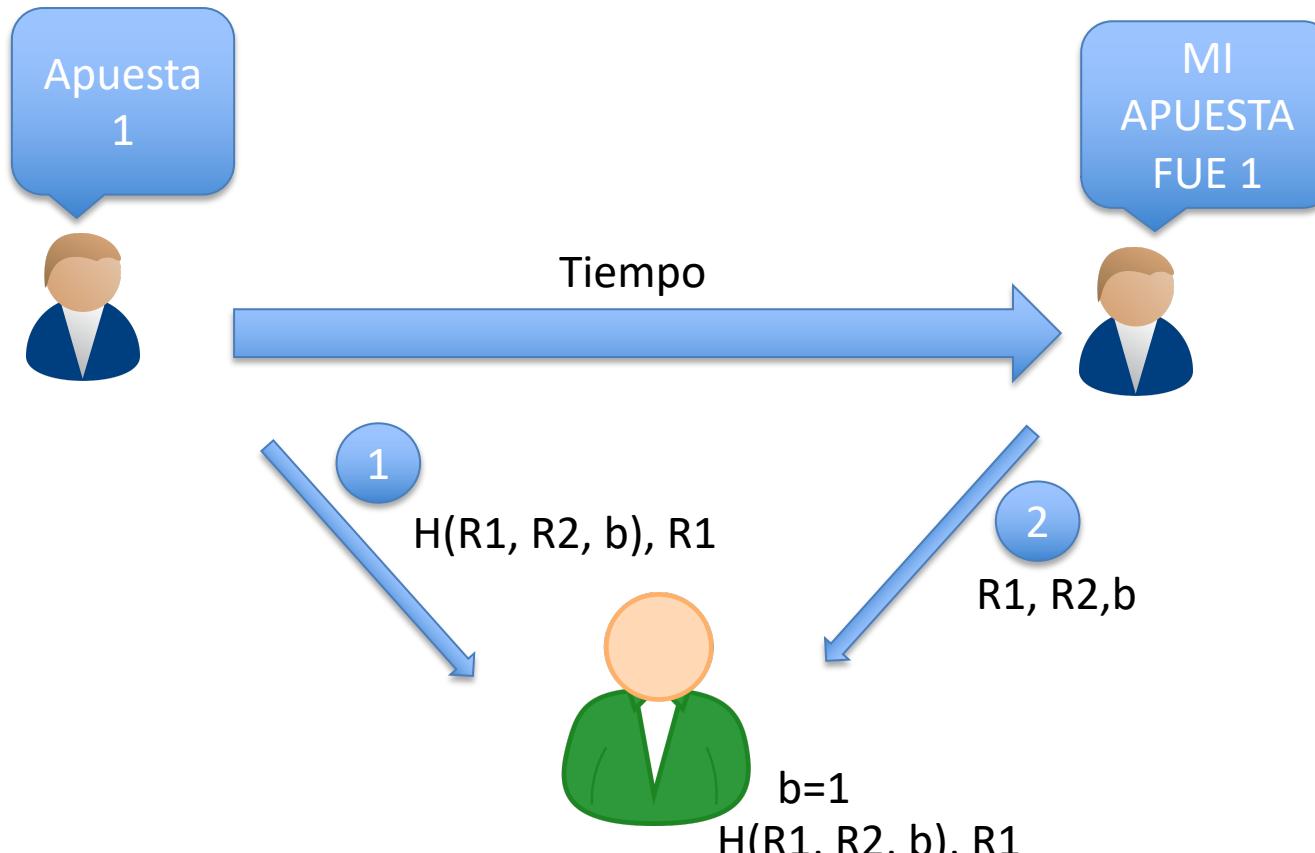
③ *Alice* envía a *Bob* el mensaje original

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}

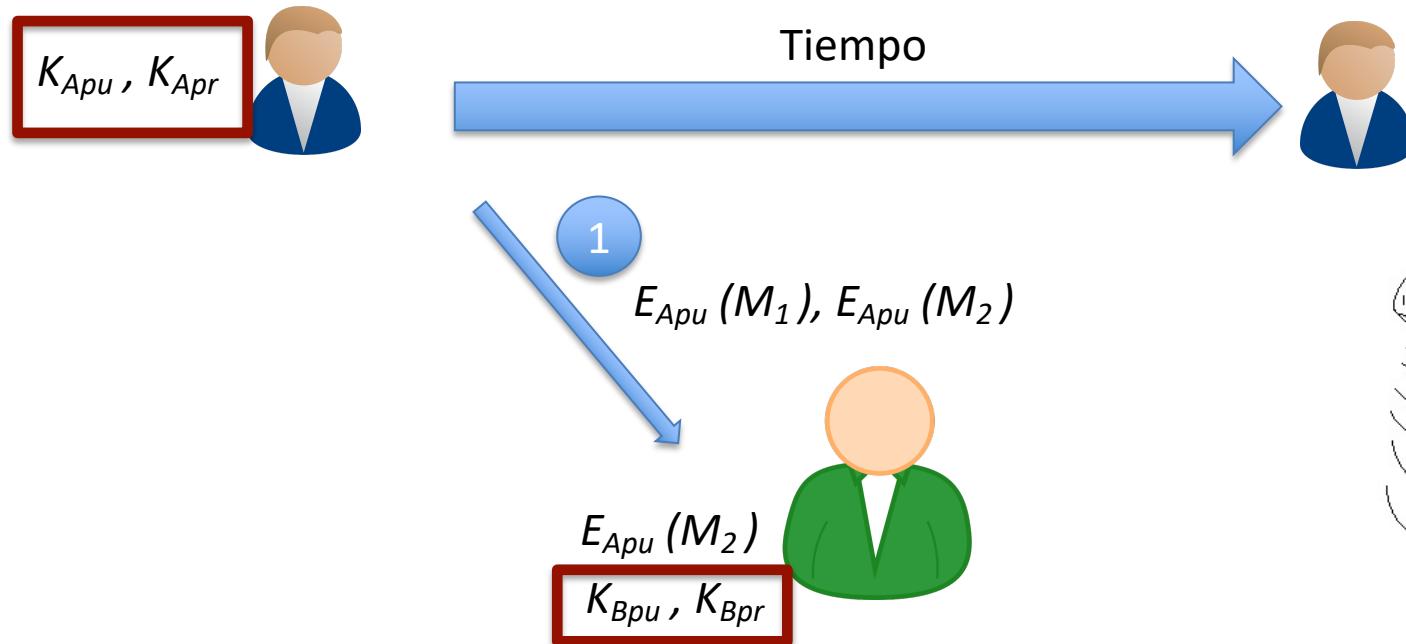
Bob: K_{Bpu}, K_{Bpr}

- ② *Alice* genera dos mensajes, uno indicando “cara” y otro “cruz”. Estos mensajes contienen además alguna cadena aleatoria, para verificar posteriormente su autenticidad.

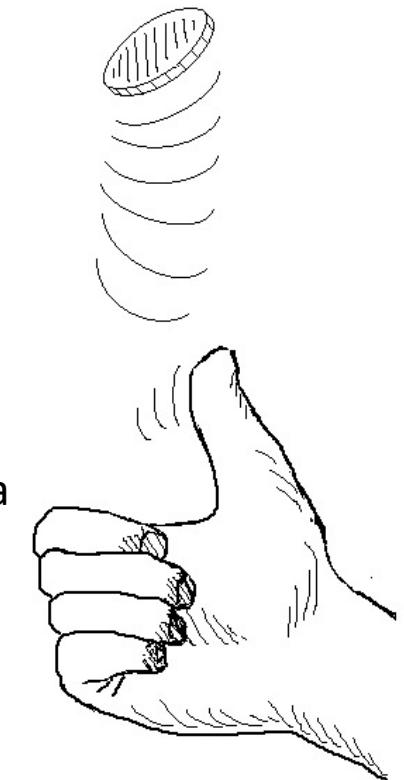
Alice cifra ambos mensajes con su clave pública y los envía a *Bob* en un orden aleatorio

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*

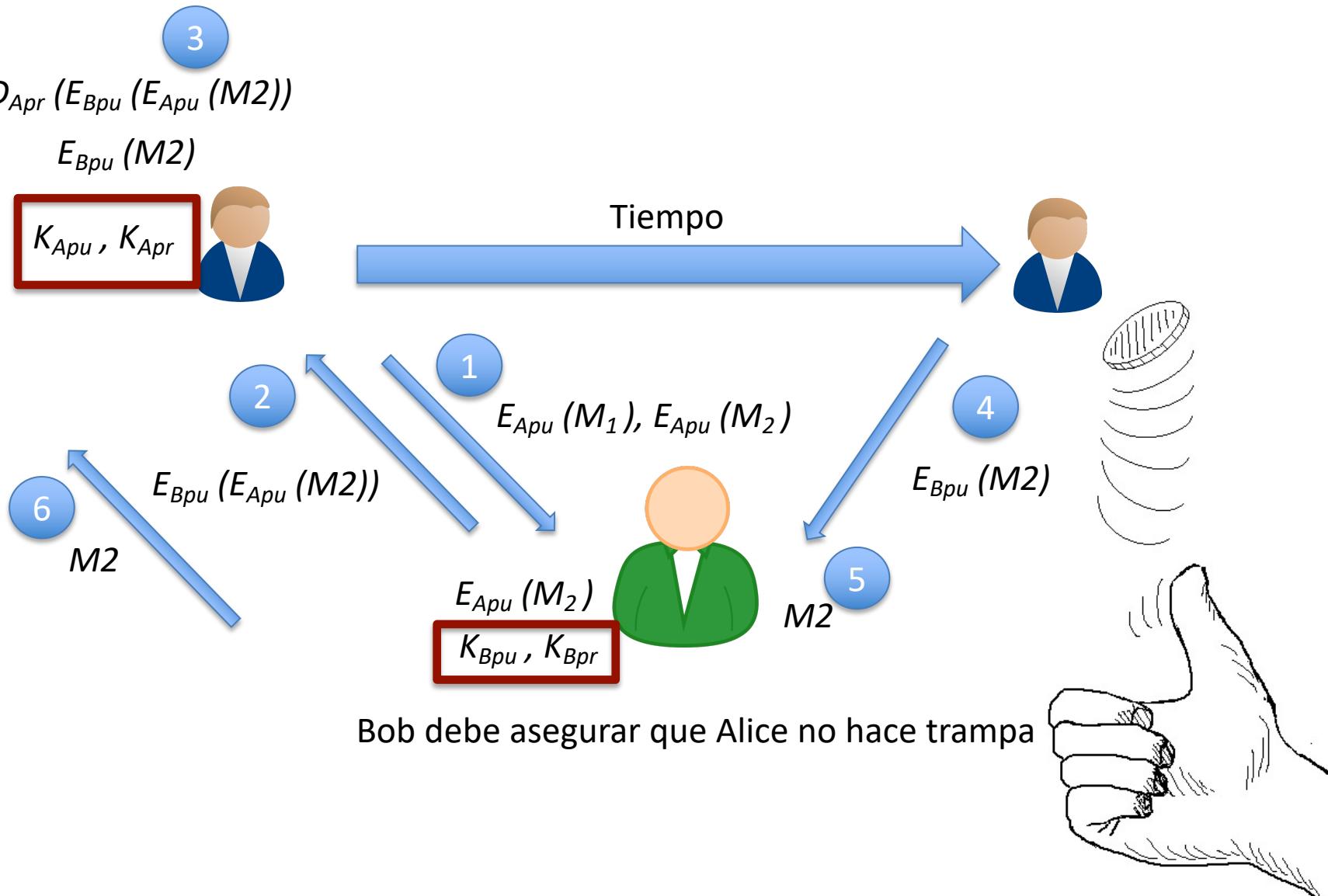
$Bob: D_{Bpr}(E_{Bpu}(M))$

$Bob \rightarrow Alice: M$

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

SEGURIDAD DE LA INFORMACIÓN

TEMA 4 – PARTE 1

**SEGURIDAD Y PRIVACIDAD
EN APLICACIONES TELEMÁTICAS**

Indice del tema

- Otras herramientas de seguridad – *red team and blue team*
 - Herramientas de *red team*
 - Sistemas operativos
 - Herramientas
 - Herramientas de *blue team*
 - Seguridad en email: PGP y S/MIME
 - Conexión remota segura
 - Herramientas de cifrado
- Seguridad en pagos electrónicos
 - Conceptos generales
 - Protocolo SET
 - Protocolo Cybercash
 - Protocolo iKP
 - Protocolo Millicent

Indice del tema

- Privacidad de los usuarios en aplicaciones
 - Conceptos generales
 - Privacidad basada en esquemas avanzados de firma digital
 - Privacidad basada en protocolos criptográficos y de enrutamiento

OTRAS HERRAMIENTAS DE SEGURIDAD

- RED TEAM AND BLUE TEAM

Red Team

- *Red team* consiste en un equipo especializado en realizar actividades ofensivas
- Buscan vulnerabilidades y atacan a su propio sistema para:
 - Probar la eficacia de las soluciones y herramientas de seguridad y su robustez frente a ataques
 - Analizar las posibles fugas de información que puedan existir, y conexiones o redirecciones remotas
 - Determinar posibles comportamientos y técnicas de futuros atacantes contra su propio sistema
 - Etc.
- Metodología de trabajo:
 - Diseñan ataques específicos de acuerdo a la arquitectura de su sistema
 - Identifican herramientas de hacking para explotar vulnerabilidades
 - Ejecutan los ataques pre-diseñados
 - Extraen debilidades y vulnerabilidades para reportárselo al equipo Blue

Blue Team

- *Red team* consiste en un equipo especializado en realizar actividades defensivas
- Tratan, evitan o mitigan vulnerabilidades por:
 - Proporcionar herramientas, soluciones o estrategias defensivas
 - Analizan los informes detallados del equipo Red para identificar las vulnerabilidades y mitigarlas
 - Monitorizan las actividades del sistema y ofrecen planes de actuación
 - Etc.
- Metodología de trabajo:
 - Recopilan información del sistema y evalúan riesgos
 - Identifican soluciones de seguridad o diseñan nuevas soluciones para evitar riesgos
 - Monitorizan constantemente la seguridad del sistema por evaluar los eventos producidos
 - Trabajan en la continua mejora de la seguridad de un sistema
 - Etc.

RED TEAM

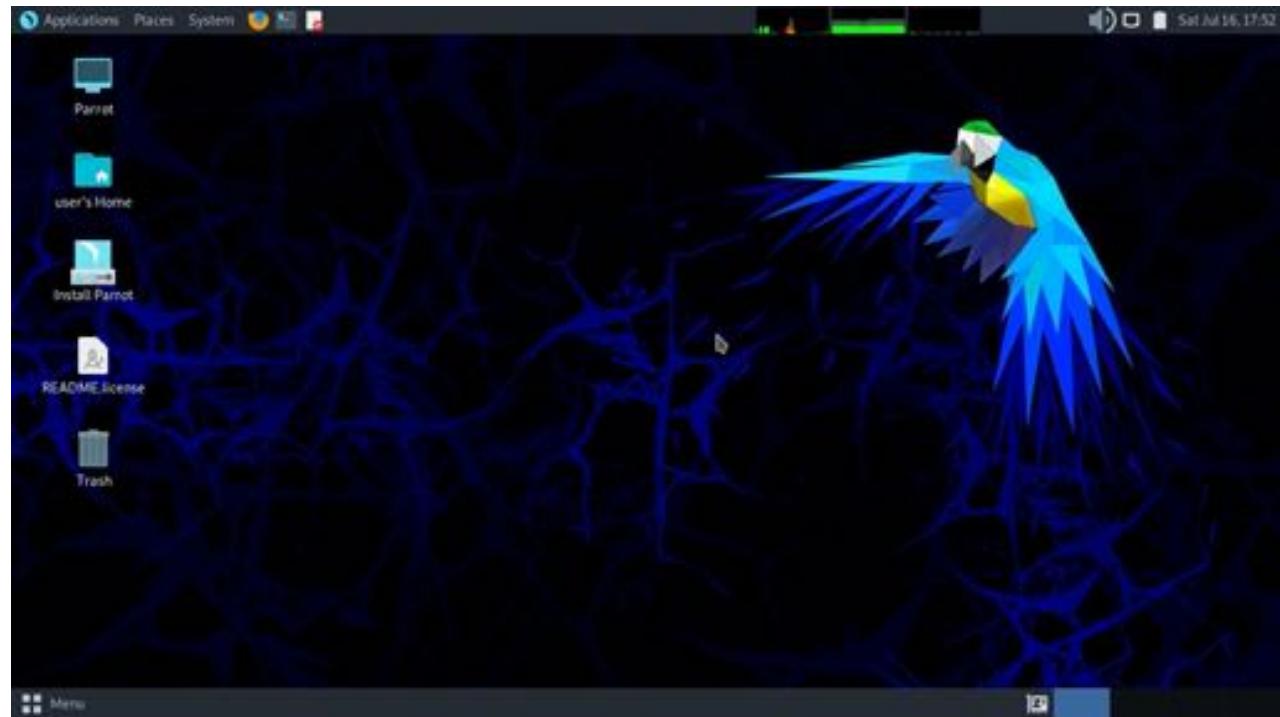
Kali linux

- Distribución Debian GNU/Linux diseñada principalmente para la auditoría y seguridad informática, ofrece un conjunto relevante de herramientas de seguridad



Parrot Security OS

- Distribución Debian GNU/Linux diseñada específicamente para realizar pruebas de penetración, evaluación y análisis de vulnerabilidades, análisis forense de sistemas, preservación del anonimato y análisis y prueba de criptografía

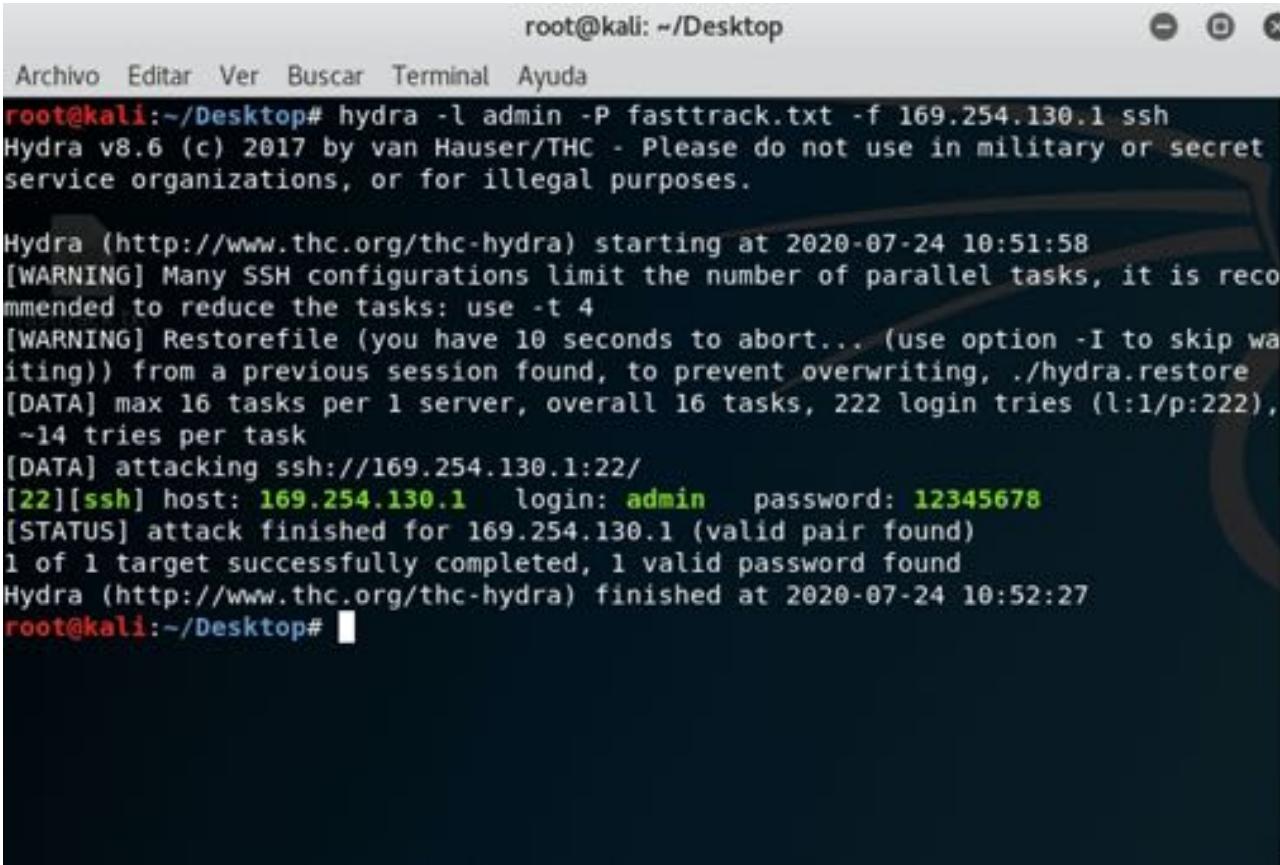


Algunas herramientas en Linux

- **Aircrack-ng** - herramienta para descifrar claves 802.11 TPA-PSA y WEP
- **Burpsuite** - herramienta integrada para probar aplicaciones web
- **Hydra** – herramienta para derivar credenciales de seguridad e iniciar sesión
- **John (the Ripper)** - herramienta derivar contraseñas
- **Maltego** – herramienta de inteligencia y forense
- **Metasploit framework** – un suite de herramientas para realizar pruebas de seguridad extremadamente flexible
- **Nmap** - herramienta de mapeo de redes
- **Owasp-ZAP** - herramienta de prueba de aplicaciones web
- **Wireshark** - La principal herramienta de análisis de protocolos de red
- **Lego** (la versión actualizada de Sparta) - herramienta de pruebas de penetración de la infraestructura de red
- **Scapy** – herramienta de manipulación de paquetes de red
- Etc.

Hydra - fuerza bruta

- Es un cracker de inicio de sesión que se basa en lanzar un ataque fuerza bruta por usar un diccionario de posibles contraseñas



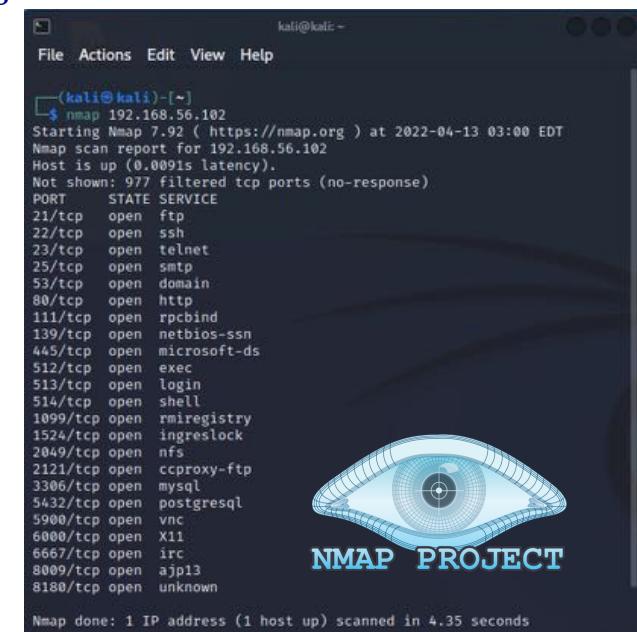
A terminal window titled "root@kali: ~/Desktop" showing the output of a Hydra attack. The terminal has a standard window title bar with minimize, maximize, and close buttons. The menu bar includes "Archivo", "Editar", "Ver", "Buscar", "Terminal", and "Ayuda". The main area displays the command entered and its execution.

```
root@kali:~/Desktop# hydra -l admin -P fasttrack.txt -f 169.254.130.1 ssh
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or secret
service organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2020-07-24 10:51:58
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recom
mended to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip wa
iting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 222 login tries (l:1/p:222),
~14 tries per task
[DATA] attacking ssh://169.254.130.1:22/
[22][ssh] host: 169.254.130.1    login: admin    password: 12345678
[STATUS] attack finished for 169.254.130.1 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2020-07-24 10:52:27
root@kali:~/Desktop#
```

Nmap y Zenmap - rastrear y recopilar información

- Nmap permite el *fingerprinting*
 - El fingerprinting consiste recolectar información de un sistema y conocer sus vulnerabilidades, y suele dejar rastro
- **Nmap:** herramienta de escaneo de puertos, rangos de IP, información del sistema operativo
 - Los análisis se realizan en línea de comando o mediante una GUI (Zenmap)
 - Compatible con Windows, Linux, MacOS y Solaris
 - Opciones de nmap:
 - **nmap IP/localhost:** puertos abiertos y servicios
 - **nmap -sP IP/localhost:** realizar un simple ping
 - **nmap -p T:1-65535 -T4 IP/localhost:** puertos (o un rango) en modo agresivo (-T4)
 - **nmap -p T:X-Y,Z -T4 IP/localhost:** puertos concretos
 - **nmap -sV -T4 IP/localhost:** puertos, sus servicios y versiones
 - ...

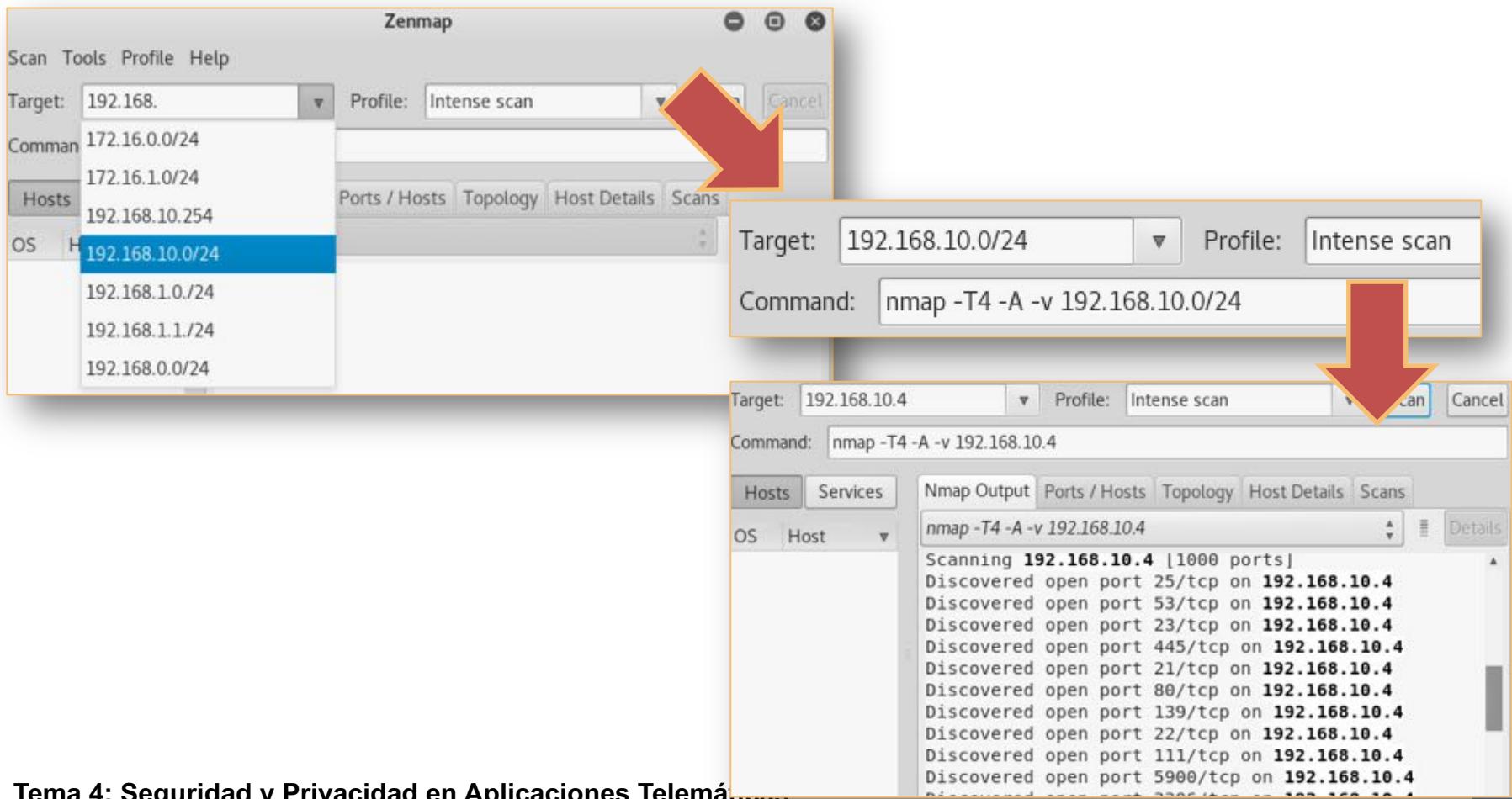


```
(kali㉿kali)-[~] $ nmap 192.168.56.102
Starting Nmap 7.92 ( https://nmap.org ) at 2022-04-13 03:00 EDT
Nmap scan report for 192.168.56.102
Host is up (0.0091s latency).
Not shown: 977 filtered tcp ports (no-response)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  cproxy-ftp
3306/tcp  open  mysql
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
8009/tcp  open  ajp13
8180/tcp  open  unknown

Nmap done: 1 IP address (1 host up) scanned in 4.35 seconds
```

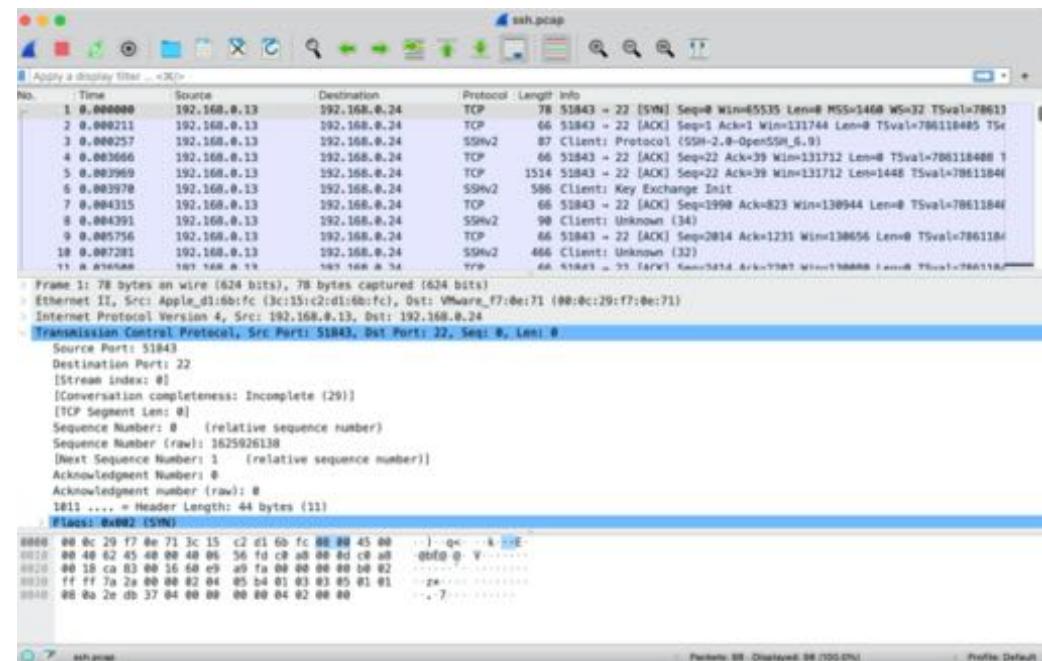
Nmap y Zenmap - rastrear y recopilar información

- Zenmap: open-source cuyo objetivo es ejecutar instrucciones nmap mediante una interfaz de usuario



Wireshark - captura de tráfico de red o sniffing

- Disector de red a cargo de analizar protocolos de red
 - Multiplataforma, y funciona en Windows, Linux, macOS, Solaris, FreeBSD, NetBSD y muchos otros
 - Captura en vivo y análisis fuera de línea
 - Soporte de descifrado para muchos protocolos, incluyendo IPsec, ISAKMP, Kerberos, SNMPv3, SSL/TLS, WEP y WPA/WPA2
 - Capacidad para exportar en XML, PostScript®, CSV o texto sin formato



Scapy - manipulación de paquetes

- Scapy es una librería desarrollada en Python, que tiene su propio interprete de línea de comandos, con la capacidad de crear, modificar, enviar y capturar paquetes de red

The screenshot shows the Scapy v2.4.3 terminal interface. It starts with a welcome message from Lao-Tze: "Craft packets like it is your last day on earth." Below this, a large ASCII art representation of the Chinese character for 'Scapy' is displayed. The terminal then shows a command being entered: `>>> send(IP(src="145.167.7.9",dst="169.254.130.252"))`. The response indicates that one packet was sent.

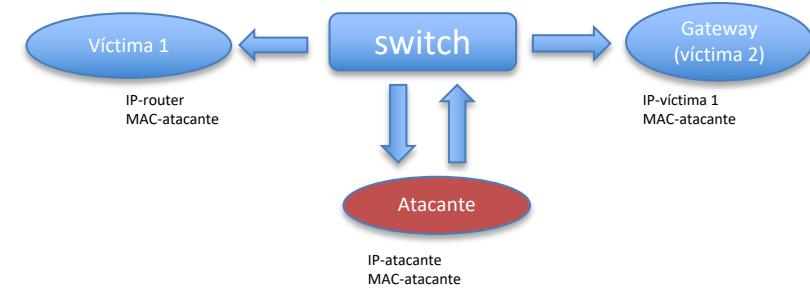
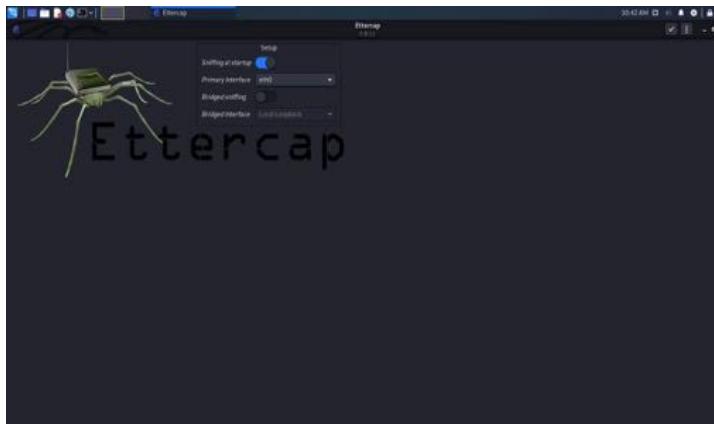
```
Scapy v2.4.3
File Actions Edit View Help
kaliKali:$ sudo scapy
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().
WARNING: No route found for IPv6 destination :: (no default route?)

          aSPY//YASa
          apyyyyCV/////////YCa
          sY/////////YSpCs  scpCY//Pp
          ayp ayyyyySCP//Pp      syY//C
          AYAsAYYYYYYYY///Ps      cY//S
          pCCCCY//p      cSSp y//Y
          SPPP//a      pP///AC//Y
          A//A      cyP///C
          p///Ac      sC///a
          P///YCpc      A//A
          sccccp///pSP//p      p//Y
          sY/////////y caa      S//P
          cayCayP//Ya      pV/Ya
          sY/PsY///YCc      aC//Yp
          sc sccaCY//PCyapaapYCP//YSs
          spCPY////VSpS
          ccaacs
                                         using IPython 7.14.0
>>> send(IP(src="145.167.7.9",dst="169.254.130.252"))
.
Sent 1 packets.
>>> 
```

- Compatible con múltiple plataforma: Linux, Mac OS x y Windows

Ettercap-graphical

- Ettercap es una herramienta que intercepta las comunicaciones mediante ARP spoofing (envenenamiento de la red por ARP) creando ataques de tipo “Man-in-the-Middle”
 - Es una herramienta capaz de analizar todo tráfico atrayendo todo el tráfico de red a la máquina del atacante en lugar del router
 - Básicamente, lo que hace la máquina del atacante es anunciar que su “IP” es la del router real pero la dirección MAC es justo de la máquina del atacante



MITRE ATT&CK

- MITRE es un repositorio (muy conocido) que ofrece un conjunto de técnicas de ataque y tácticas en diferentes ámbitos, a nivel de empresa, para sistemas móviles y sistemas de control industrial

The screenshot displays the MITRE ATT&CK matrix interface. At the top, there's a navigation bar with links for 'Matrices', 'Tactics', 'Techniques', 'Data Sources', 'Mitigations', 'Groups', 'Software', 'Campaigns', 'Resources', and search and contribute buttons. Below the navigation is a sidebar titled 'MATRICES' with a tree view of platforms: Enterprise, PRE, Windows, macOS, Linux, Cloud, Network, Containers, Mobile, and ICS. The main area shows a grid of tactics and their associated techniques. Tactics include Reconnaissance, Resource Development, Initial Access Techniques, Execution, Persistence, Privilege Escalation, Defense Evasion, and Credential Access. Techniques are further broken down into subtechniques like 'Acquire Infrastructure' under Reconnaissance or 'Abuse Elevation Control Mechanism' under Persistence. A specific technique, 'T1500', is highlighted in the Reconnaissance section.

<https://attack.mitre.org>

BLUE TEAM

Seguridad en email: PGP y S/MIME

- El correo electrónico es la aplicación más ampliamente utilizada en la gran mayoría de los entornos distribuidos
- El crecimiento en su uso ha conllevado una mayor necesidad de seguridad, y, más concretamente, de la integración de servicios de **autenticación y confidencialidad**
- Entre las soluciones de seguridad en e-mail disponibles en la actualidad, hay dos que destacan por su amplio uso:
 - PGP
 - S/MIME



PGP (Pretty Good Privacy)

- PGP es una solución diseñada por Phil Zimmerman en 1991
 - Básicamente, PGP consiste en seleccionar algoritmos criptográficos ya existentes en integrarlas en una única aplicación SW independiente del S.O.
 - Integra los algoritmos **RSA**, **DSS**, **Diffie-Helmann (ElGamal)**, **CAST-128**, **IDEA**, **3DES**, **SHA-1**
 - Proporciona servicios de **autenticidad y confidencialidad** que se pueden usar para:
 - **aplicaciones de e-mail**
 - **almacenamiento de ficheros**
 - Existen versiones libres para Windows, UNIX y Mac, además de versiones comerciales
- Incluso IETF ha realizado avances con PGP:
 - *RFC 3156: MIME Security with OpenPGP*



PGP (Pretty Good Privacy)

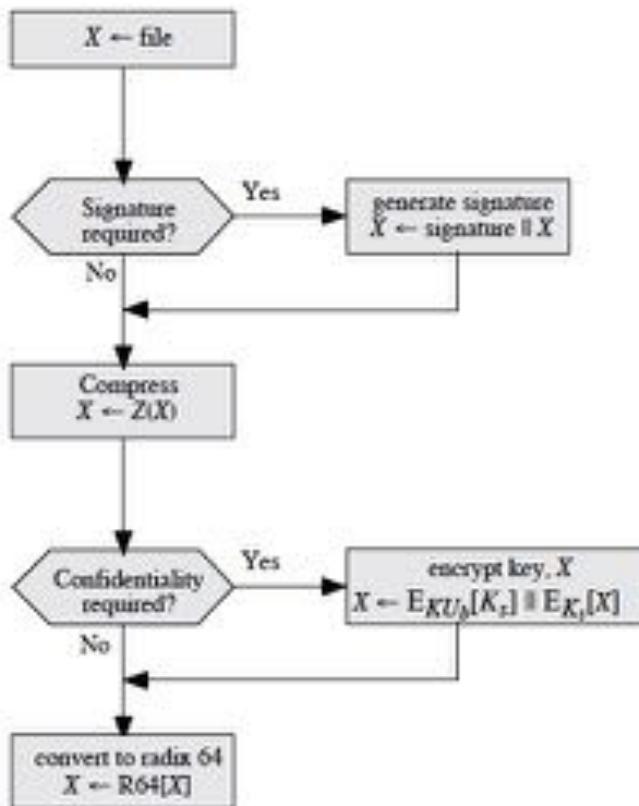
- Las operaciones de PGP incluyen, además de autenticación y confidencialidad, las operaciones de compresión y de compatibilidad de e-mail

Function	Algorithms Used	Description
Digital signature	DSS/SHA or RSA/SHA	A hash code of a message is created using SHA-1. This message digest is encrypted using DSS or RSA with the sender's private key and included with the message.
Message encryption	CAST or IDEA or Three-key Triple DES with Diffie-Hellman or RSA	A message is encrypted using CAST-128 or IDEA or 3DES with a one-time session key generated by the sender. The session key is encrypted using Diffie-Hellman or RSA with the recipient's public key and included with the message.
Compression	ZIP	A message may be compressed, for storage or transmission, using ZIP.
Email compatibility	Radix 64 conversion	To provide transparency for email applications, an encrypted message may be converted to an <u>ASCII string using radix 64 conversion</u> .

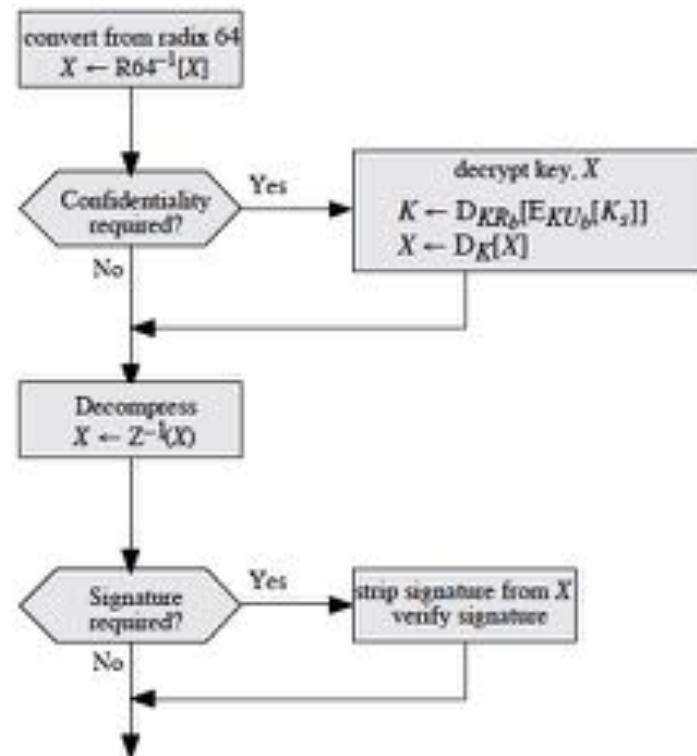


PGP (Pretty Good Privacy)

- Esquema de transmisión y recepción de mensajes PGP:

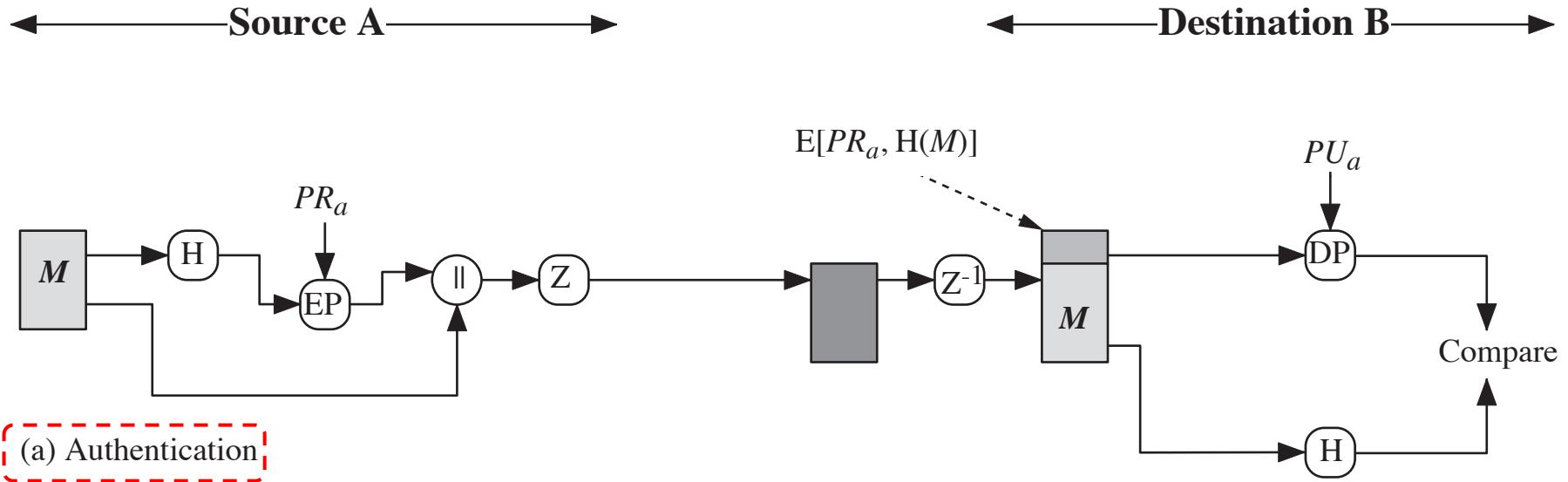


(a) Generic Transmission Diagram (from A)



(b) Generic Reception Diagram (to B)

PGP (Pretty Good Privacy)



K_s = session key used in symmetric encryption scheme

PR_a = private key of user A, used in public-key encryption scheme

PU_a = public key of user A, used in public-key encryption scheme

EP = public-key encryption

DP = public-key decryption

EC = symmetric encryption

DC = symmetric decryption

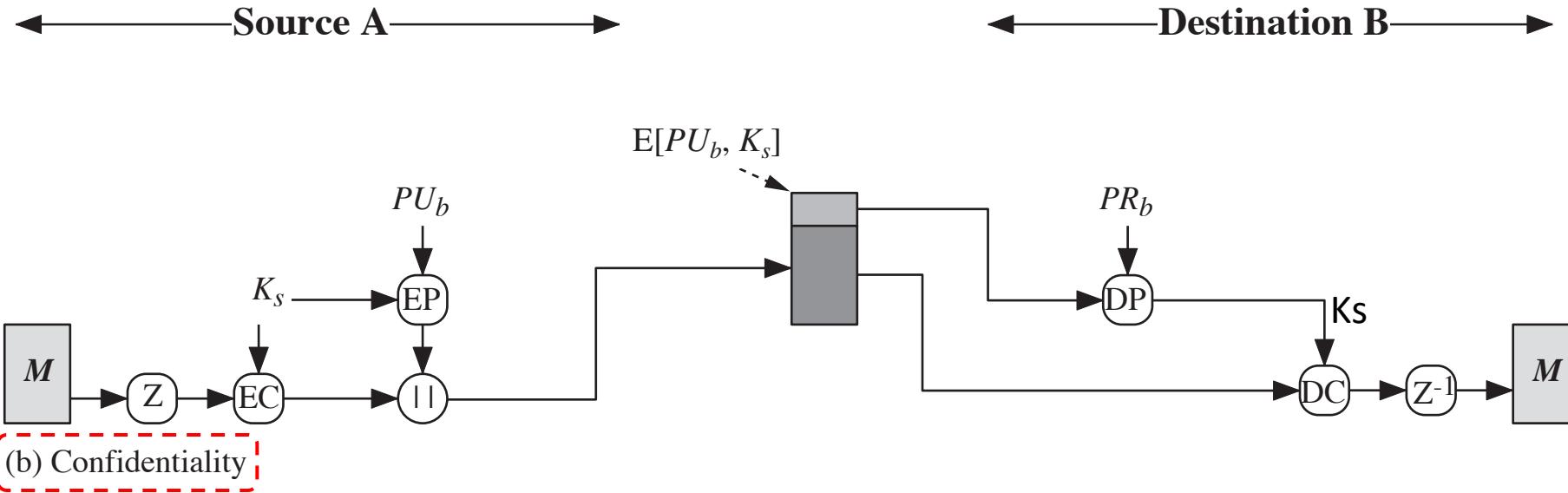
H = hash function

\parallel = concatenation

Z = compression using ZIP algorithm

R64 = conversion to radix 64 ASCII format

PGP (Pretty Good Privacy)



K_s = session key used in symmetric encryption scheme

PR_a = private key of user A, used in public-key encryption scheme

PU_a = public key of user A, used in public-key encryption scheme

EP = public-key encryption

DP = public-key decryption

EC = symmetric encryption

DC = symmetric decryption

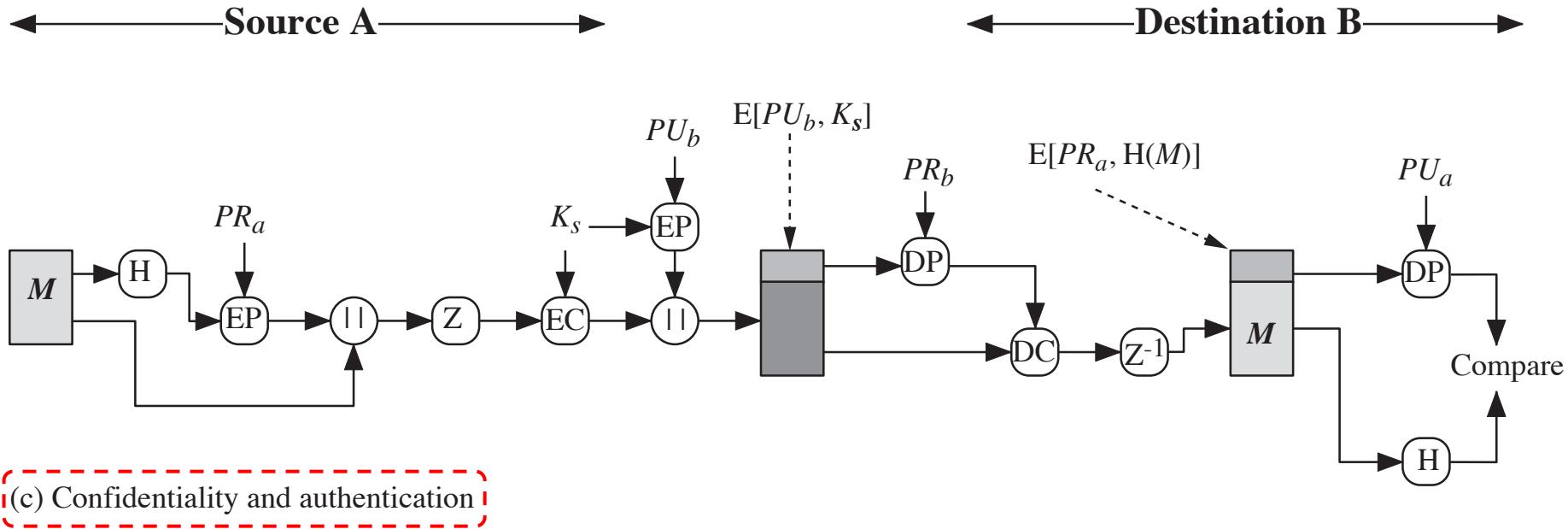
H = hash function

\parallel = concatenation

Z = compression using ZIP algorithm

R64 = conversion to radix 64 ASCII format

PGP (Pretty Good Privacy)



K_s = session key used in symmetric encryption scheme

PR_a = private key of user A, used in public-key encryption scheme

PU_a = public key of user A, used in public-key encryption scheme

EP = public-key encryption

DP = public-key decryption

EC = symmetric encryption

DC = symmetric decryption

H = hash function

$||$ = concatenation

Z = compression using ZIP algorithm

R64 = conversion to radix 64 ASCII format

PGP (Pretty Good Privacy)

- PGP proporciona, para cada usuario U , dos estructuras de datos:
 - **private-key ring**: para almacenar los pares <clave pública, clave privada> del propio usuario U
 - **public-key ring**: para almacenar las claves públicas de los otros usuarios con los que U se comunica

Private-Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•
T_i	$PU_i \bmod 2^{64}$	PU_i	$E(H(P_i), PR_i)$	User i
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•

Public-Key Ring

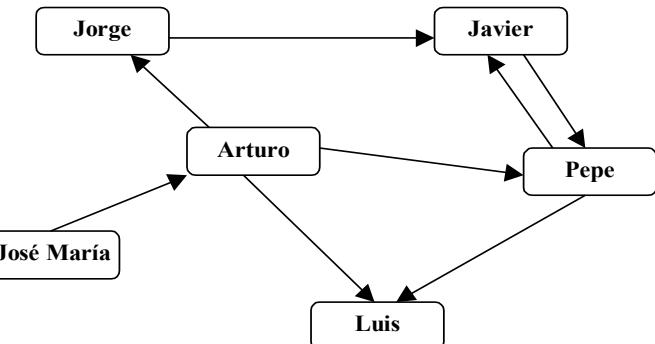
PGP (Pretty Good Privacy)

- Cada línea de la tabla del public-key ring puede considerarse en sí misma como un tipo de **certificado digital** (certificado de clave pública) **no estándar**
 - PGP no usa el estándar X.509, sino un formato propio



Public-Key Ring								
Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
T _i	$PU_i \bmod 2^{64}$	PU_i	trust_flag _i	User i	trust_flag _i			
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	

- Tampoco basa su funcionamiento en la existencia de una PKI jerárquica de Autoridades de Certificación, sino en un **modelo de PKI en malla**
 - o sea, no existen Autoridades de Certificación al uso, pero **cada usuario del sistema puede emitir certificados** al respecto de las claves públicas de los demás usuarios
 - de ahí los valores de confianza (**trust**) incluidos en el public-key ring



PGP con OpenPGP

- El OpenPGP Working Group se creó en 1997 y gracias en parte al Internet Engineering Task Force para definir el estándar
 - OpenPGP es una aplicación estándar y libre que permite cifrar emails usando criptografía de clave pública y un específico formato
- Está basado en el PGP original



[Docs] [txt|pdf] [draft-ietf-openpg...] [Diff1] [Diff2] [Errata]

Updated by: 5581 PROPOSED STANDARD
Network Working Group Errata Exist
Request for Comments: 4880 J. Callas
Obsoletes: 1991, 2440 PGP Corporation
Category: Standards Track L. Donnerhacke
IKS GmbH
H. Finney
PGP Corporation D. Shaw
R. Thayer
November 2007

OpenPGP Message Format

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Abstract

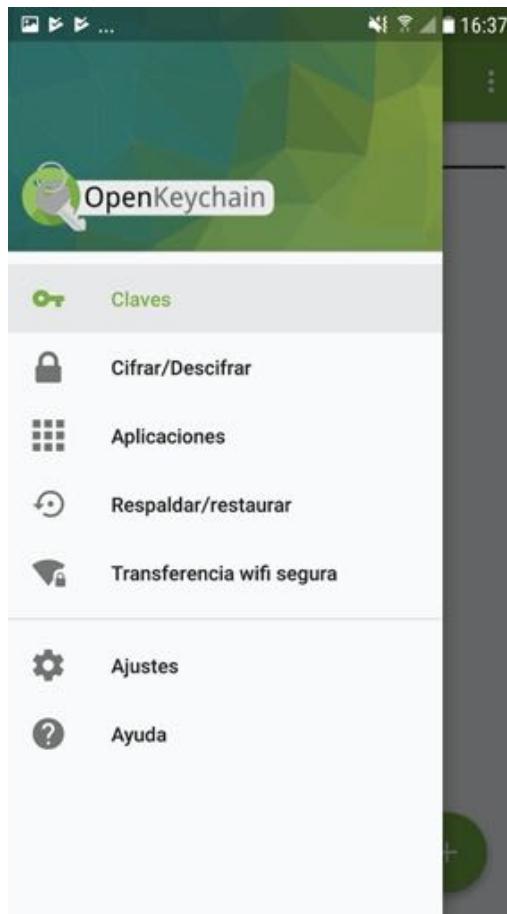
This document is maintained in order to publish all necessary information needed to develop interoperable applications based on the OpenPGP format. It is not a step-by-step cookbook for writing an application. It describes only the format and methods needed to read, check, generate, and write conforming packets crossing any network. It does not deal with storage and implementation questions. It does, however, discuss implementation issues necessary to avoid security flaws.

OpenPGP software uses a combination of strong public-key and symmetric cryptography to provide security services for electronic communications and data storage. These services include confidentiality, key management, authentication, and digital signatures. This document specifies the message formats used in OpenPGP.

PGP con OpenPGP

- Existen varias aplicaciones que soportan OpenPGP:
 - Windows
 - Outlook: gpg4o1, Gpg4win, p=p
 - Thunderbird: enigmail
 - Mac
 - Apple mail: GPGTools
 - Mutt
 - Thunderbird: enigmail
 - Android
 - K-9 mail: Openkeychain
 - P=p
 - R2Mail2
 - iOS
 - iPGMail
 - Linux
 - Evolution: Seahorse
 - Kmail:Kleopatra
 - Mutt
 - Thunderbird: enigmail
 - Solaris
 - Mutt
 - Thunderbird: enigmail
 -

OpenKeychain



The screenshot shows the "Configuración" (Configuration) screen of the OpenKeychain app. At the top, it says "Servidores de claves OpenPGP" with a toggle switch turned on. Below this, it lists "Administrador servidores de claves OpenPGP" with "3 servidores de claves; preferido: hkps://keyserver.ubuntu.com". There are three more toggle switches below: "keybase.io" (on), "Web Key Directory" (on), and "Habilitar Tor" (off). Further down, there are settings for "Habilitar otro proxy" (off), "Servidor proxy" (set to 127.0.0.1), "Puerto de proxy" (set to 8118), and "Tipo de proxy" (set to HTTP).

Texto y ficheros

iPGMail

Carrier 1:39 PM

Done **Create Private Key** **Create**

Passphrase
Confirm Passphrase

Key Size:

Expiration:

Real Name: John Q. Smith

Email Addr: user@domain.com

Carrier 1:39 PM

Edit **Public** **Private** **+**

Public Keys

0	1	2	3	4	5	6	7	8	9	C	F	I	K	N	Q	T	W	Z
adele-en <adele-de@gnupg.de> 4D486CC8 RSA(2048) 2013-07-08 2017-07-08																		
boo <boo@hoo.com> D4C8EB20 RSA(2048) 2013-07-23 (no exp)																		
foobar <foo@bar.com> CC33E7CC RSA(2048) 2013-07-29 (no exp)																		
FooFoo <foo@bar.com> 818F5EE0 RSA(1024) 2012-01-03 (no exp)																		
Wyllys <wyllys@me.com> 47E3234C RSA(2048) 2011-06-11 (no exp)																		

Keys **Compose** **Decode** **Files** **Settings**

Carrier 1:40 PM

Clear **Sign** **Encrypt** **Both** **Upload**

From: foobar <foo@bar.com>

To: info@ipgmail.com

Attach: Select Attachments

Great job!

Keys **Compose** **Decode** **Files** **Settings**

Carrier 11:05 AM

Done **Public Key Details**

Public Key Info

Key ID: 47E3234C
User ID: Wyllys <wyllys@me.com>
Fingerprint: A5D0 EFBC 4A52 B587 E68A 2451 7FEA 1CCB 47E3 234C
Created: 2011-06-11
Expiration:
Algorithm: RSA (enc/sign)
Image: N/A
Key Len: 2048

UID Info

Wyllys Ingersoll <wyllys@gmail.com>
47E3234C 2011-06-11 -----

UID Info

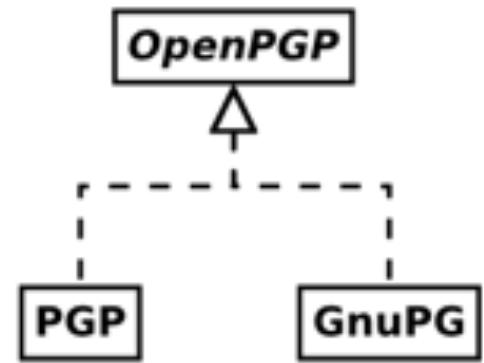
iPGMail Support <info@ipgmail.com>
47E3234C 2011-09-21 -----

UID Info

Wyllys <wyllys@me.com>

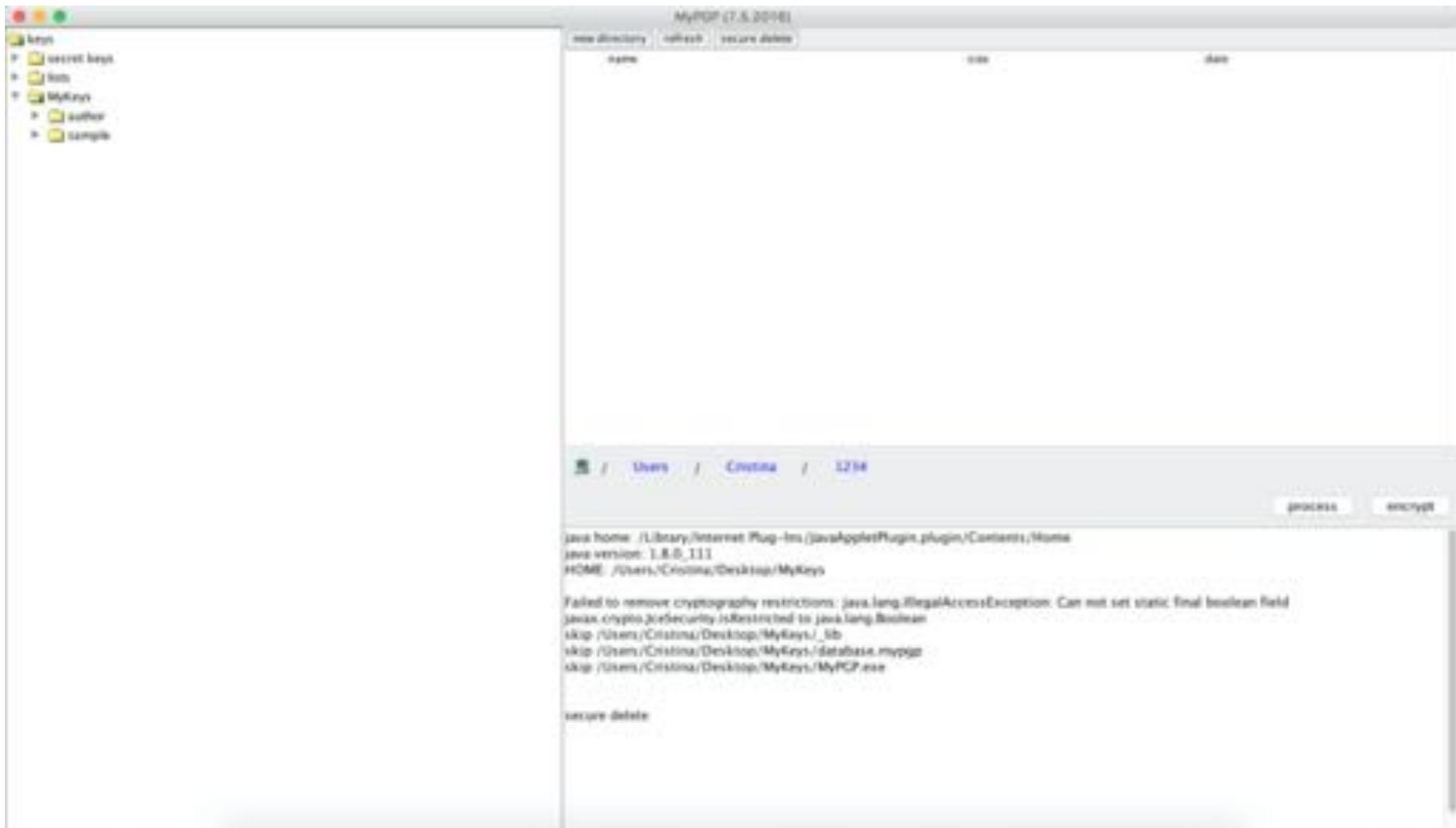
GnuPG

- GnuPG es una implementación del estándar OpenPGP que deriva del software criptográfico PGP desarrollado por Philip Zimmermann
- Con GnuPG se puede realizar las siguientes acciones:
 - gestionar y generar claves públicas y privadas
 - visualizar y distribuir las claves públicas, exportándolas e importándolas
 - cifrar y descifrar documentos
 - firmar y validar documentos



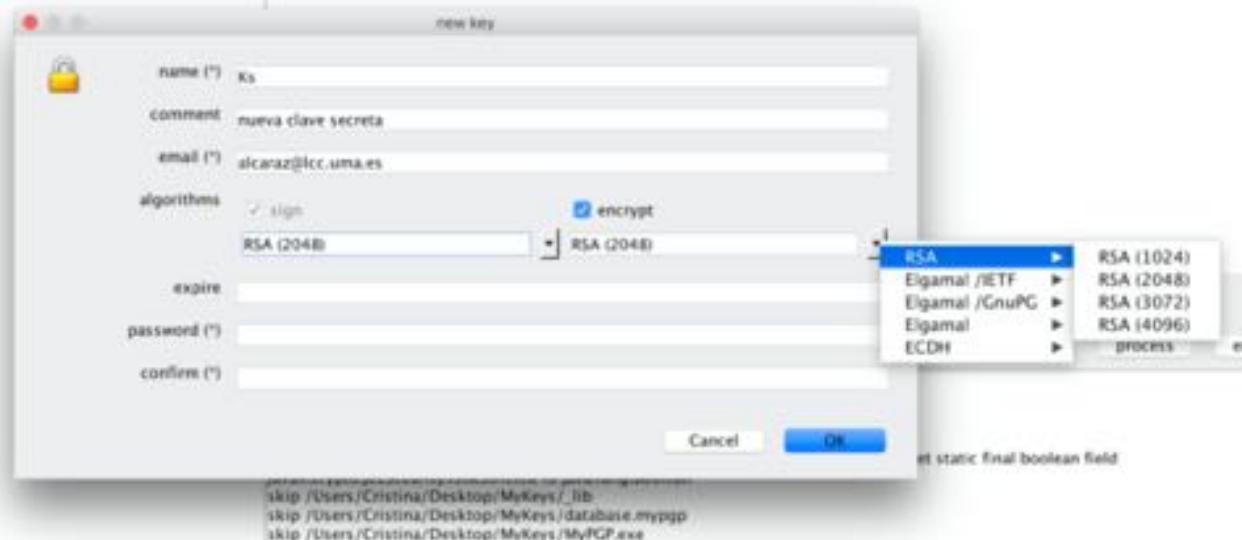
<https://www.gnupg.org>

MyPGP



MyPGP

Creando una pareja de claves (privada y pública):



The main interface of the MyPGP application displays a tree view of keys and a file browser.

Tree View (left):

- keys
 - secret keys
 - Ks <calcaraz@lcc.uma.es> (nueva clave secreta)
 - Publius Cornelius Scipio Africanus <publius@roma.imperium> (password: password)
 - lists
 - sample
- MyKeys
 - Ks <calcaraz@lcc.uma.es> (nueva clave secreta)
 - author
 - sample

File Browser (right):

- new directory
- name: Captura d

Key List (center):

- secret keys
 - Ks <calcaraz@lcc.uma.es> (nueva clave secreta)
 - [28.11.2016] Ks <calcaraz@lcc.uma.es> (nueva clave secreta)
 - fingerprint: d539 9c93 8305 846f 8ffe ff43 e208 be32 8a8d c86e
 - RSA (2048) / RSA (2048)
 - encrypt: AES 256, AES 192, AES 128, CAST5 (128), 3DES (112/168)
 - [8A8DC86E] fingerprint: RSA (2048) / RSA (2048)
 - encrypt: AES 256, AES 192, AES 128, CAST5 (128), 3DES (112/168)
 - Publius Cornelius Scipio Africanus <publius@roma.imperium> (password: password)
 - [12.12.2014] Publius Cornelius Scipio Africanus <publius@roma.imperium> (password: password)
 - fingerprint: ace0 b6a1 38a3 1cff 91e4 0f87 cd60 b3e5 f241 1ec2
 - RSA (2048) / RSA (2048)
 - encrypt: AES 256, AES 192, AES 128, CAST5 (128), 3DES (112/168)
 - [F2411EC2] fingerprint: RSA (2048) / RSA (2048)
 - encrypt: AES 256, AES 192, AES 128, CAST5 (128), 3DES (112/168)

Bottom Navigation:

 - > lists
 - > MyKeys

MyPGP

Si se visualiza la clave privada:

```
-----BEGIN PGP PRIVATE KEY BLOCK-----
Version: BCPG v1.55
Comment: MyPGP (7.5.2016)

10PGBF0c8cCABCAClL2mdceC4KvyPR0W2Y9Kbm08u+g/+0C+rFy90g28P7a6JIE
L0xhERq0FFFeeCiHix/h70mKLH+n5Ga@mH0Hx2xyevvvvT/HwKKqGFbyPt98i6boW
5p5a/k1mmgx1Pw5CaNckoE8+TnAyNDu/6E/cPNWbqWn7nEBJ6dcG3ogZGweG/6hz
14ksNakc2rcCRIf0ZLwntB8IuiFLjhKUxOwawNx+GjY8pLhgiiYtDRr5Zizj1AYH
dKbnwhLRzR0XgpKJ5i1HwG1N9BTfVUykgagoiqthzDnHjctzcR0J0105n32vD6gko
0x+fqUOXi1vvxfPv796UL49E76P5aEKUueLAEBAAH+COMiq7i2TuGUwsbaU26m
6X7hjUj6tskb8Bjh+wpa9k8bPLVfZz7Glv30D7mt2icVKYhnyHn3VfR4054XTNrN
q0wArhhgnXnFog9e6aMw8ie8eEUJ+qqaOn3qgtVw9TPNJRqqF8oypi1uBYDrr2D
20/iy2ub32jTy5dzLYKbT/LSA+GP5czQ3TMpNIYd5Pq1oMwjioid6/Pxd4NKu5PY
TsIC63qIIqVJcWtVnspICN0j5eVbQRVWESvb480tVxkcW7MsLWaKLp0Ke653s5w
5/LVi2ATedLaR8E8KNivCSg4orzi+bwsL/TDJHcxRGZ1Hs+XkVLukU2as0Plbizz
MTrh4M1K6pzEp/P4+FGLGre5wxluFFBQV0B8jW7Bzu3U6PvuvJn@x7TAFOjnCzOfs
1Z3vbwN29y9IIxlosTWW82dG8q8I3ae+LD2k09+uknWcalRgkcuQgi0wFBemGiSg
/1jRv40XvQUYtQaqg46PhoeU0XP7uGz67jYjn@tP+HmIV0d00Lttqz4dw148B3jjq
trdf67zYKV541Xeqzb0T4xwIwaG4xZA6JeoT92NbuhSRUVW4Ts3jCmWd/bDjcx
CSMVc5jhcqodulBG6MqmDnvE4HccMqYmSFg2d1+K7G/arg8tNRhkqg1RIP0dIeV
moXWE2LJ2lBoOCNPw2651ph1TcJ+se/+AO/crtVmql9awdY1o4K0jIQze+nR8E/5
tkST0THqJTI4271hK0m10SzluzfQX450lN41zH150P98AHGFdFckJXiHJ2monHS
t@l/E5Y1YscfY3USe3YkVvSBxjaip10Md4RC69yR7MpieoLGSNe4Kj7rTyRGYNM
7w6eXv1UUmqJFK42JU8V1/PYKfG9W/5ca1Cqwc9GK79QJ0QJ1RNqjT2qCbCwf3xf
+Lh9k8uShoYgtC1lcyaBYTwk1YXjhekB8y2Muwd1hLmVzPiAobnVld=EqY2xhdMq
c2VjcmV8Y5m3ATUEfwEIAB83FAlg8cCEFGwMGcwkIBwMCBRYCAwEGF0gJCgsCAh4B
AAoJE011vJKKJchufTah/RHuverbJ+GJRK9MSID848Rb0XhfQyL1KsAgsCqAsK/7
B8XvrZnE/B89G1CIG29ghn3PB48HQNYZT8qwMn13rDAduoNh5rCt0SfC0fTL8Gs
a40RGEmYw57h/sFVCBH/VFRbyjdZJUaBe7vvnx/05vcqEa5pd4NBj0WKt7HF8FAr
0vynzJ1+IdCvthQ9vq01k1+49T94Xmz7Kmk/r/q3j3VjJU5trh5ukAvYkmkDBm/G
HDM2qyFt9EowTtpbraWhjN8xarVzc3KvX15/rHa3Im7b16w005pCoAY680j9xy1
1PAmZQW2zehuA4mfqIR3xRZjQEKhzqIavUbeonpNjjidABUEwDxwIAEIAKm8duWt
7dhS1E+UpEHxF7/eY0OM1eu/M4GF0oVxZPRT7V1fd/isshbFMy85B+1GN1sk93R+
vFFfw91XT+SSh913sEQ1Rbb+ZqsA0Z2zdsDQovTFdc0vPZhng9E1lkNln0MU060p
tQk+d7gtJG/aoBE1kfxxh+UMgs6z2fXL0MBkScP+GMVHxb0LIKam2bTn0Lb8TLQ
5apZ9eN0C+1dB=6enBvxthKxw4d2zIHBTYKqKjrvjB89pmvc9ux7FZJYEJWENu9
6HF+nGw5s12cW1Ax2BXwmtdAd8yT1TgB/Gycd5tsT0Po12p7261Jnayf93YihmgNu
5QMnqB3512rMXUAEOEAAf4JAwiruXN04ZTCxsAGB82KNS+Vl6NhxUsh7Bpru6ut
w1u6s7RNWbMpug2znRKu8k1Vkb7XRkt1Z0dkZ0j10PX9cta1H4mMuYNG2mLoTJ
RCp3bT07EMPhAbmzH+071YE+INYczAHuV0sLw2db08T5MKKT91DrCX0w61FuYIA
jE5nfnbog3gm7mxz5hk0VJ1npl0e4inziGK1d1j0Z+B/RVQJV90n2d2Lx81hgmL8
60rnpmTzjgN1JW54uf3aV81tvIC9lu7Hw70PoP1zSEM9j9DEP01MLDq6h3v/7N1i
qyRTrPszaHzz8+M9uUX9+r1YsBq3BvWCRowiu5Ngqkyz0/76w2Y2saiVlqp3DEifI
4/lAw/fsdhVt49sMiE0d1pxExksZxcfVMtA2NbibiET6n556okfEgMo9aiXcG0H/Y
1Nos910F48+1TILT10mIxzVaY1zx+qRM0j52KLk1ITHE6+xNxXutufxM1PYD92Y
KDqWhOnDjN2AKGqeWx6JY0hniG8kSxZ7BAP1qww1MErPmbR1WZg+3eDdzf4MwUq
UKPuzfoQajL9pHy0eCtMKwJ5q1qiaenzulqtj18bUCl6pXhmFRYYEYL4l3eg1vByg
bCcDCLA7loqJ+7x1m0/dbW5TV65nC19P58Q12B1Atln1ur3RVmy@DC8k/nhKo3k
Exr65A05ycGEQo2I1dma/6ew3DkwkJC98XHQ+M7xzM0cvNC7YSWES2wfE2310xz/
1H7MaMT45Hg565YfMU9ULiwDgh80KTVsw/vx88A6RBZ61j4bK938d11zL+lwNXUP
tsuwjBD+zY12C1BxdZHWpmlCSt/0VqjNSXh726mr4G4bf2v+u1p2zWU3MFoE8j
Ee9NwjhbFmkwbRBvwB0o3Zw59hdwuUjZUuR+tyFHEb4UnUR74o/WBt4kBHwQYAQgA
CQUCWdxwI0ibDAAKCR01CL4y1o31biwuB/9GbTTfxjWvzrFicAa90GuRSvvFs/7F
r40MuKXXcKoPnw65Sgll5c4YkKPRgzmxxk1C6uLo8e1t2mxwNWwKTNoy4vNsY/LD
HNgs0/1FN5UKpmlbcRmkkPTX0gruk6MT2q5/b5gnAAxNwWItpU1UDDEfxh3LP/H
QEJM0D9dL7+bo1nrqHtu1b1Mwlv2ZDR0M+qLx5780G1bfu+8+bBKDAfrFwNIjohx
idJBHfup@v19FSXEmGPexJW1Iuz55ccYS60uXH1yuw3Xqg5jyLHTMwGZ51DU1+D6
IOgzSolr17R6Wlm0cLo38++RyIdnLSF+4DfZB6+78UW5INg16CcqahXw
==3c+
-----END PGP PRIVATE KEY BLOCK-----
```

MyPGP



S/MIME (Secure/Multipurpose Internet Mail Extension)

- **S/MIME** es una mejora en el ámbito de seguridad del formato MIME para correo electrónico
 - el cual a su vez es una mejora de SMTP
- Algunos de los documentos que describen S/MIME son:
 - RFC 5652: Cryptographic Message Syntax (CMS)
 - RFC 5750: Secure/Multipurpose Internet Mail - Version 3.2 -Certificate Handling
 - RFC 5751: Secure/Multipurpose Internet Mail Extensions -Version 3.2 - Message Specification



S/MIME (Secure/Multipurpose Internet Mail Extension)

- Aunque tanto PGP como S/MIME están en vías de llegar a estándar, todo apunta a que S/MIME se va a consolidar como estándar para uso comercial
 - mientras que PGP quedará para uso personal
- En términos de funcionalidad, S/MIME es similar a PGP en el sentido de que ambos ofrecen la posibilidad de **firmar y/o cifrar mensajes**
- S/MIME usa **certificados de clave pública** con formato **X.509v3**
 - Sigue también el **modelo de PKI híbrido** entre la jerarquía estricta de Autoridades de Certificación y el modelo en malla

S/MIME (Secure/Multipurpose Internet Mail Extension)

- Utiliza los algoritmos criptográficos de la siguiente tabla:

Function	Requirement
Create a message digest to be used in forming a <u>digital signature</u> .	MUST support SHA-1. Receiver SHOULD support MD5 for backward compatibility.
Encrypt message digest to form a <u>digital signature</u> .	Sending and receiving agents MUST support DSS. Sending agents SHOULD support RSA encryption. Receiving agents SHOULD support verification of RSA signatures with key sizes 512 bits to 1024 bits.
Encrypt <u>session key</u> for transmission with a message.	Sending and receiving agents SHOULD support Diffie-Hellman. Sending and receiving agents MUST support RSA encryption with key sizes 512 bits to 1024 bits.
Encrypt message for transmission with a one-time session key.	Sending and receiving agents MUST support encryption with tripleDES. Sending agents SHOULD support encryption with AES. Sending agents SHOULD support encryption with RC2/40.
Create a message <u>authentication code</u> .	Receiving agents MUST support HMAC with SHA-1. Sending agents SHOULD support HMAC with SHA-1.

Thunderbird

Seguridad

Para enviar y recibir mensajes firmados o cifrados, debe especificar tanto un certificado para firma digital como uno para cifrado.

Firmado digital
Usar este certificado para firmar los mensajes que envíe:

 Firmar mensajes digitalmente

Cifrado
Usar este certificado para cifrar/descifrar mensajes enviados a Vd.:

Cifrado elegido para enviar mensajes:
 Nunca (no usar cifrado)
 Siempre (no podrá enviar si algún receptor carece de certificado)

Certificados

Redacción: Prueba S/MIME

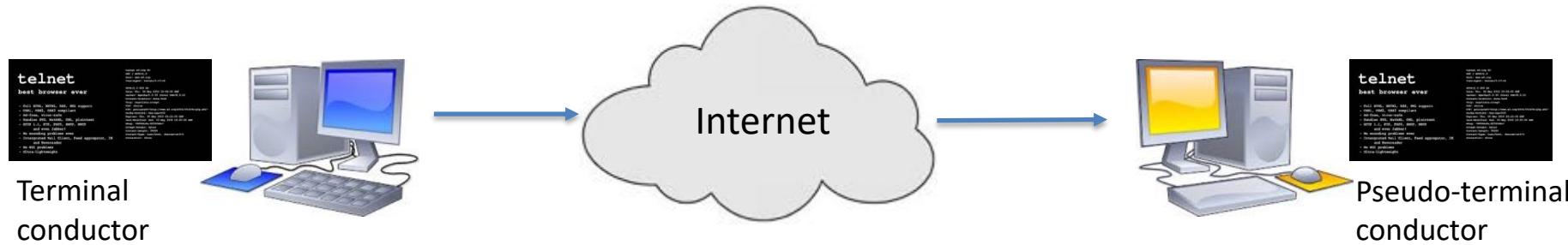
Archivo Editar Ver Insertar Formato Opciones Herramientas Ayuda
Enviar | Ortografía Adjuntar Seguridad Guardar
De:
Para:
 Cifrar este mensaje
 Firmar digitalmente este mensaje

smime.p7m

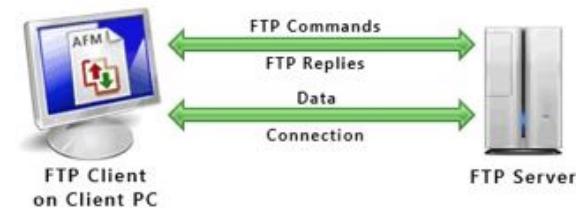
```
1 0€ACK *tHt+
2 SOHBELEOTX €0€STXSOHNUL1, SOH$0, SOH STXISOHNULO^0,1VTO ACKETXUEOTI
3 ACKETXUEOTB$DC3ACKMalaga1$T0
4 ACKETXUEOTBE1.DC3ACKMalaga1FF0
5 ACKETXUEOT
6 DC3ETXUMA1FF0
7 ACKETXUEOTVTO DC3ETXUMA1DC20DEACKETXUEOTETXDC3 Cristina 1!0USE
8 SOH SOHSYNDC2alcaraz@lcc.uma.esSTXSOHSTX0
9 ACK *tHt+
10 SOH$0SOH$0ENONULUO'e', 'NAK0'fi-c·Ó*, -jÓj (»..d£ò"DC1 [2v, SUB00ùSOI
11 \Íg..iÑADC4iè80CANiÀs<GDLÉ=ÛX: FF/Sé€S]56NAKIDOLEF6Jg\4Sf]Óauai}ETBK
12 SOHBELSOH$0DC4ACKBS*tHt+
13 ETXBELEOTBSIEP*t'aG €EOT,
14 @`FS3ñ-*piSUB&2,D/[0,483G^Z't^!1@GShfq23í]GSLjETX«"w=iBELVACKO&ç_
15 =iDC4aÀ@e)1=NscP2^ÓVÜB<"Vúcf0&"US«"dÀí-ióùEvY6èr[DC1ZcÈ\3[írt
```

Telnet (TELetype NETwork) y FTP (File Transfer Protocol)

- Características funcionales:
 - **Telnet** (puerto 23): facilita el acceso remoto a otros sistemas y sobre TCP, de forma que el terminal local aparenta ser el terminal del sistema remoto



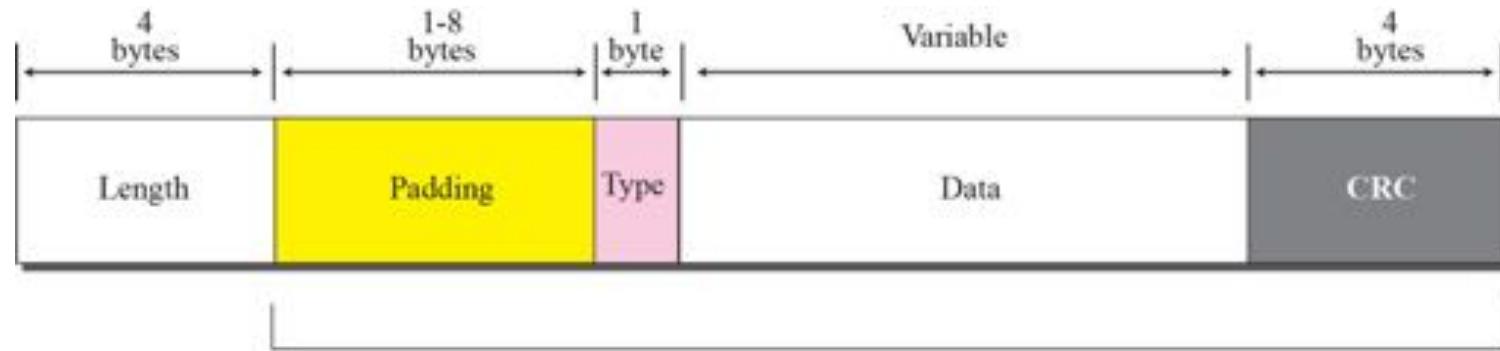
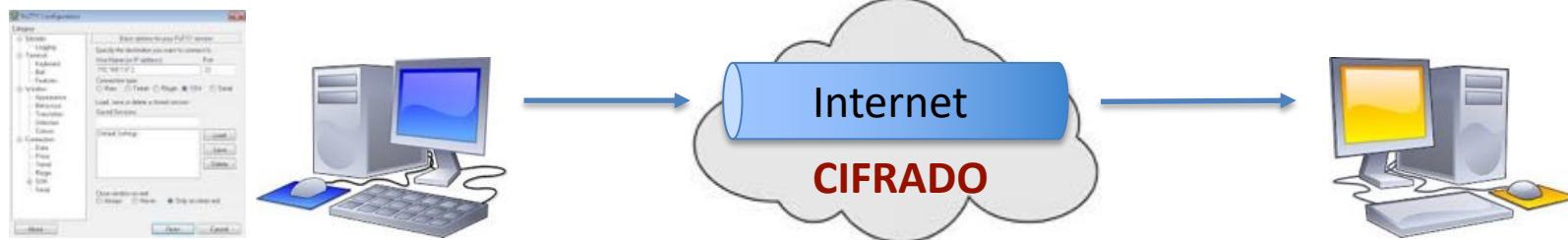
- **FTP** (puerto 20/21): permite la transferencia de ficheros entre diferentes recursos remotos



- Problemas:
 - La información transferida y las acciones remotas se realizan en claro, por lo que no se recomienda su uso

SSH: Secure Shell (v2)

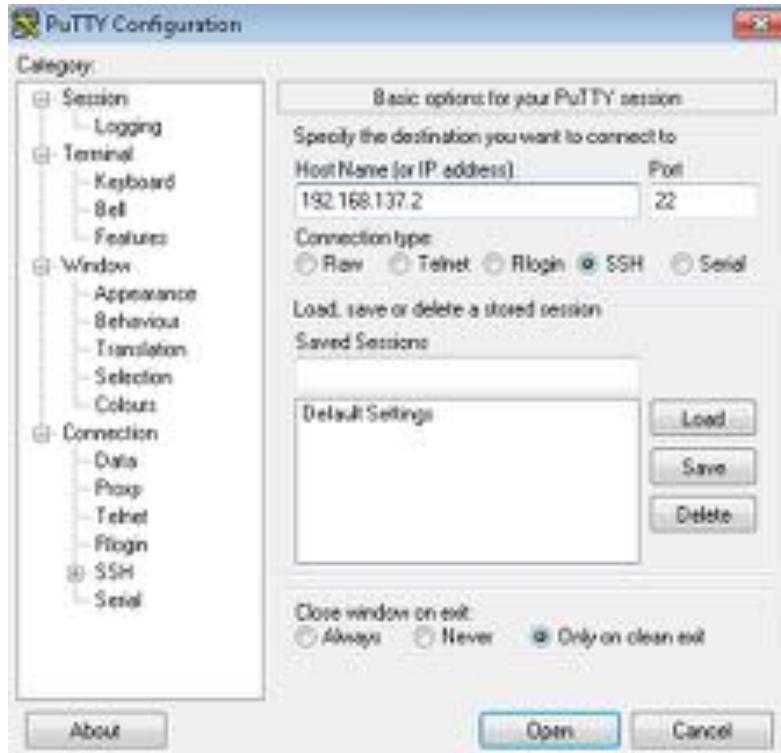
- SSH es una herramienta similar al telnet funcionando en el puerto **22**
- Permite el acceso remoto sobre TCP a otros sistemas usando el concepto de cliente/servidor pero cifrando las transacciones usando **criptografía pública**



SSH: Secure Shell (v2)

- Funcionamiento general:
 - **Capa de aplicación:**
 - Gestiona la autenticación del cliente haciendo uso de un usuario/contraseña o aplicando criptografía de clave pública
 - **Capa de transporte:**
 - Gestiona e intercambia las claves iniciales
 - Establece los modos de cifrado y de comprensión
 - **Capa de red:**
 - Establece una “conexión directa” entre el cliente-servidor y redirige el tráfico entre estos puntos de conexión
 - Modo túnel (en base a cifrado simétrico negociado en la capa de transporte)
- Mitiga o evita ataques específicos:
 - **spoofing de IP o suplantación de identidad**
 - nodos remotos intentan suplantar la identidad de otro nodo de la red
 - **spoofing de DNS** en donde el atacante trata de suplantar el nombre del servidor

SSH: Secure Shell (v2)



PuTTY

OpenSSH

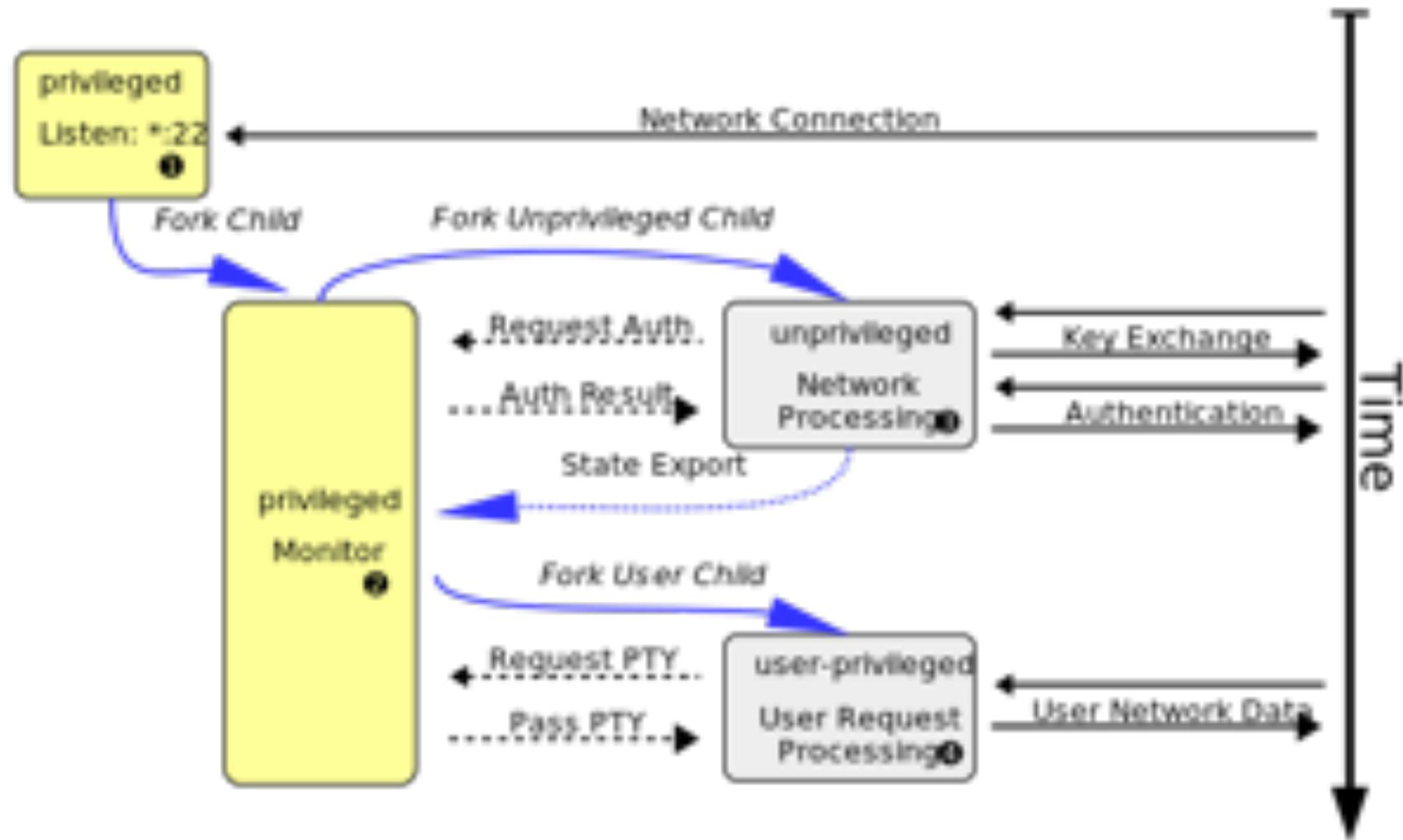
```
vdi-6E28:~ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/
[Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/vdi-6E28/.ssh/id_rsa.
Your public key has been saved in /home/vdi-6E28/.ssh/id_rsa.pub.
The key fingerprint is:
[SHA256]GX02RJH2boy@we6Pl4B1emRo0zTxY2deghkt1uK3vA
The key's randomart image is:
----[RSA 2048]----
|   .+0oo o |
|   ++0oo,++ |
|   ...B .+0o |
|   o++0e,+ |
|   S+0400 - |
|   . .0.0.. |
|   0 - - - |
|   . + 0.. |
|   . E ... |
----[SHA256]----
```

SSH: Secure Shell (v2)

- SSH puede ser usado también para transferir ficheros como una alternativa a FTP, conocido como SFTP (SSH File Transfer Protocol) y SCP (Secure Copy)
- **SFTP (FTP sobre SSH) # FTPS (FTP sobre TSL)**
 - Funcionamiento de SFTP:
 - Se puede basar de diferentes modos de autentificación para conectar con el servidor SFTP:
 - **Modo básico:** usuario y contraseña
 - **Modo avanzado:** usando las claves públicas de SSH, previamente generadas, y compartiendo dichas claves públicas con el servidor SFTP
 - » De esta forma, cuando el cliente quiere establecer conectividad con el sistema remoto, el proceso software del cliente tendrá que transmitir su clave pública al servidor para su autenticación
 - Todas las conexiones SFTP están cifradas a nivel de red

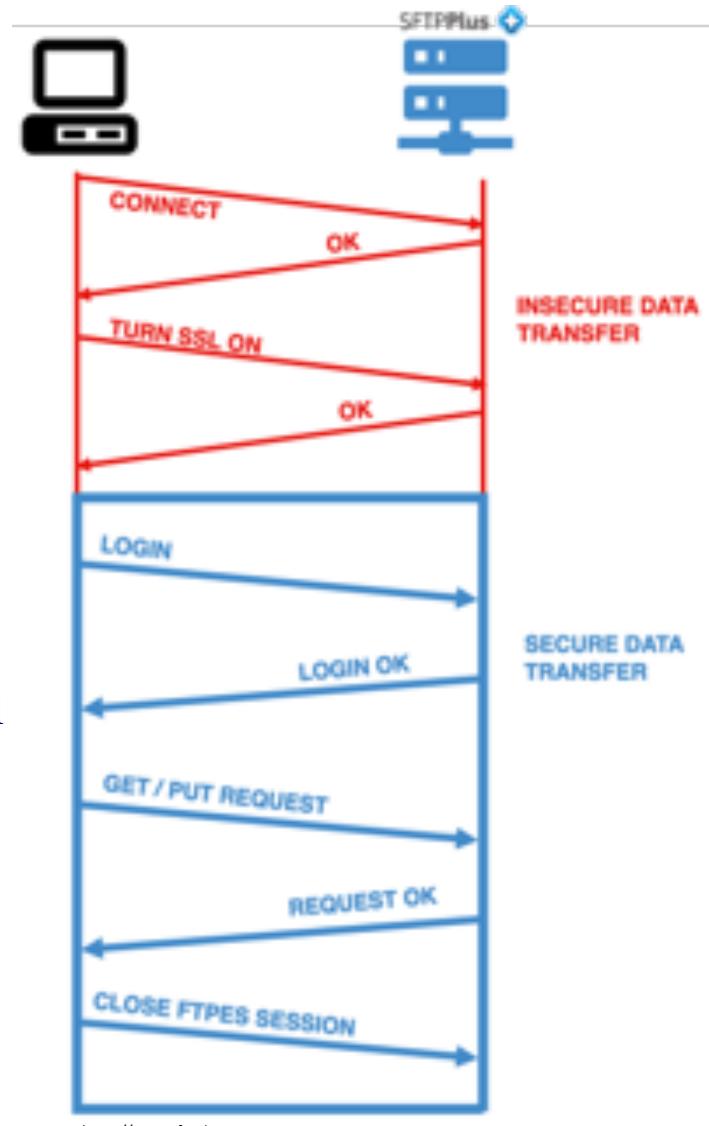


SSH: Secure Shell (v2)



FTPS (FTP Secure)

- FTPS proporciona un soporte adecuado para TLS (Transport Layer Secure)
- Funcionamiento:
 - Se basa de usuario, contraseña y certificados, de forma que:
 - Las credenciales de seguridad (usuario y contraseña) son cifrados a lo largo de la **conexión FTPS**
 - Para ello, el cliente primero verifica que el **certificado del servidor** es correcto y de confianza, y posteriormente se negocia una clave de session (ver Tema 5)



Herramientas de cifrado open-source

- **TrueCrypt:**

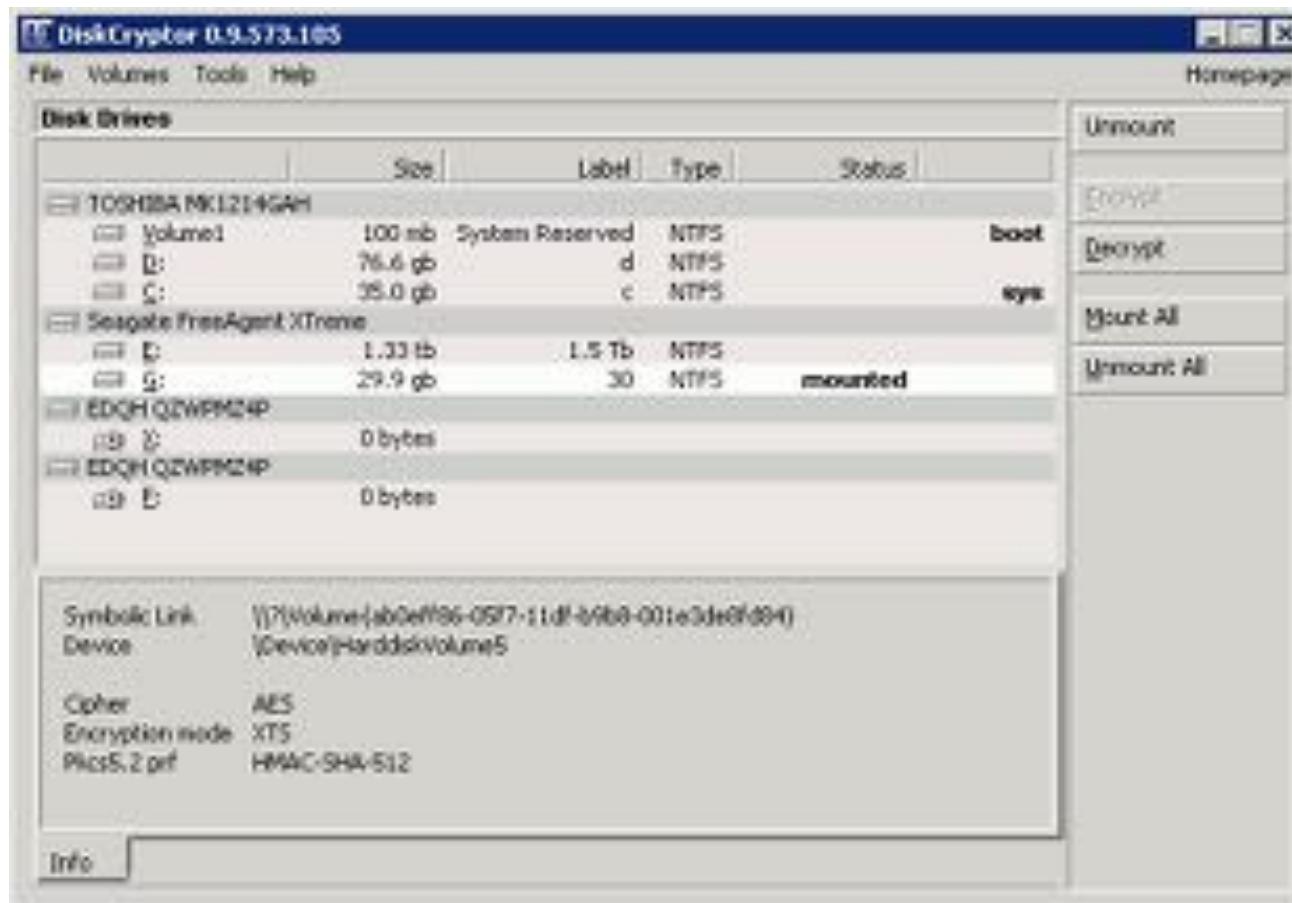
- herramienta de cifrado de discos duro disponible para Windows (XP/2000/2003) y Linux, haciendo uso de AES-256, Blowfish, CAST5, Serpent, Triple DES y Twofish
- También permite ocultación de particiones haciendo uso de cifrado y aleatoriedad de la información



Herramientas de cifrado open-source

- **DiskCryptor:**

- similar a TrueCrypt pero con capacidad de cifrar dispositivos de almacenamiento externo USB
- También incluye algoritmos de cifrado como AES, Twofish y Serpent

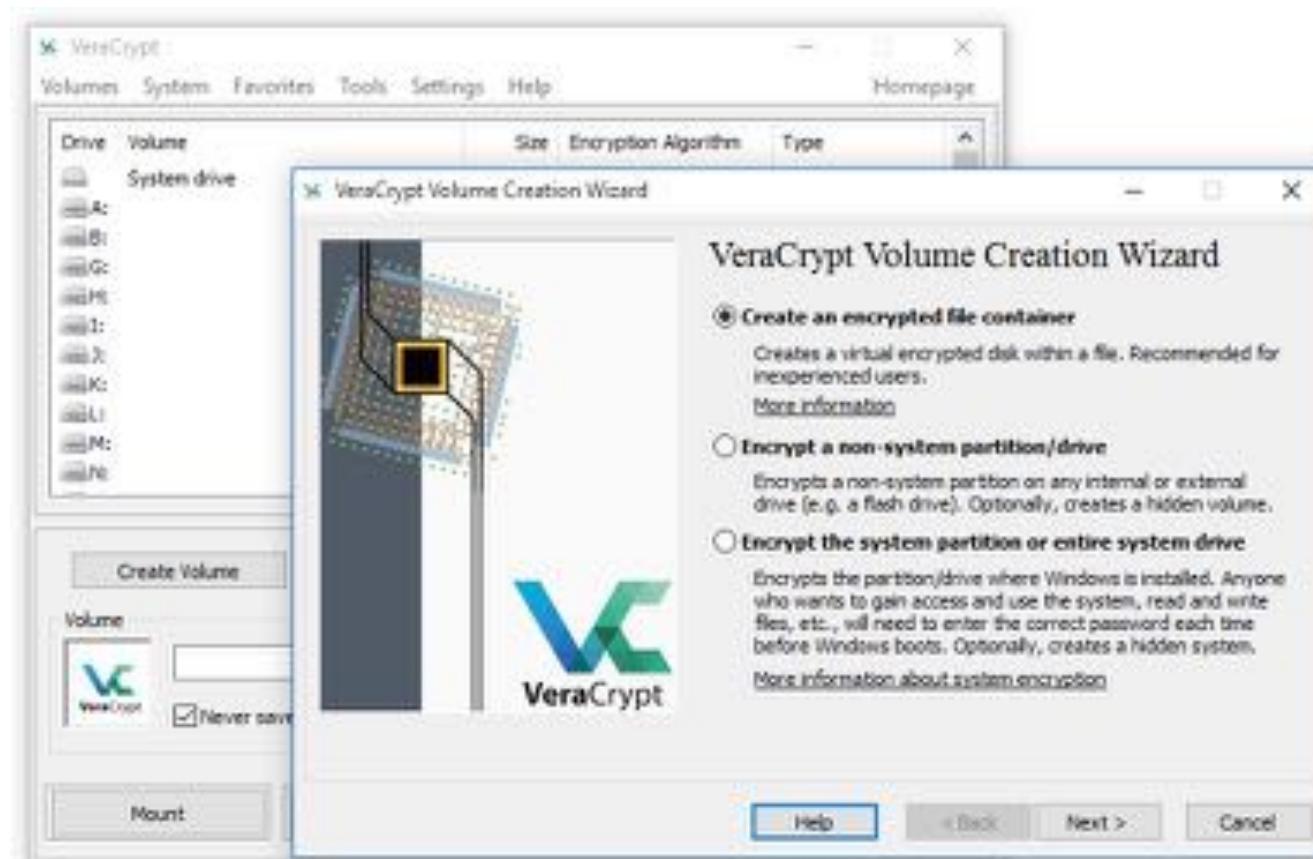


Herramientas de cifrado open-source

-

VeraCrypt:

- similar a TrueCrypt pero con la diferencia que incluye un número específico de iteraciones para el cifrado, incrementando la lentitud del sistema durante los procesos de lectura y escritura en el disco
- Como TrueCrypt, aplica AES, Twofish y Serpent



Herramientas de cifrado open-source

- **OpenStego**: aplica técnicas de **Esteganografía** (del griego "cubierto" u "oculto", y "escritura") para ocultar información haciendo uso de imágenes o cualquier archivo multimedia
- **OpenPuff**: es similar a OpenStego para Windows con soporte para BMP, JPG, MP3, WAV y MPG4



C0200' 80-
21B2C809 CB3EE8EF DF038D
04143B75 4F571C
B57C659E C820E
F743D 9A361
9A51

SEGURIDAD EN PAGOS ELECTRÓNICOS

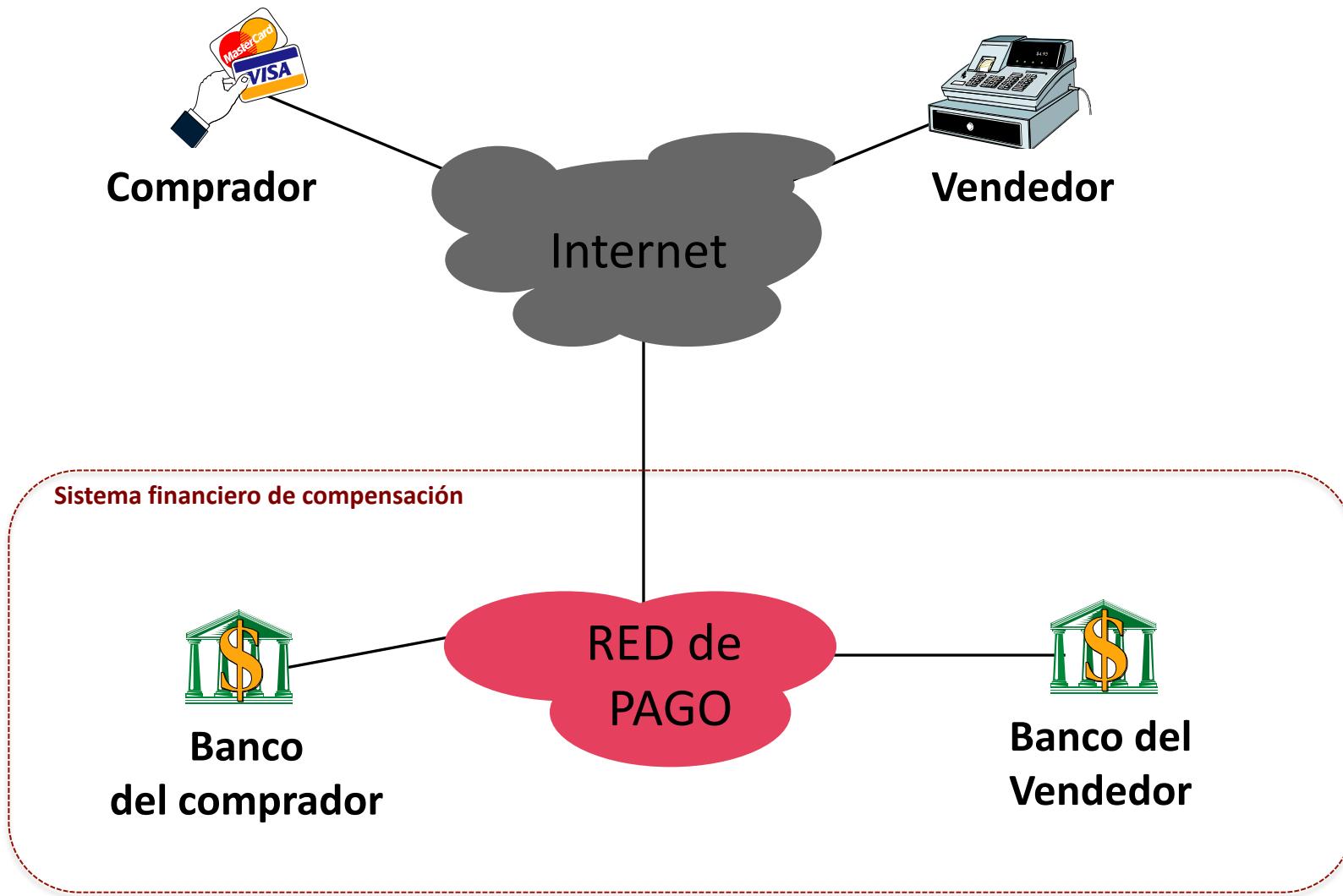


Introducción

- En el ámbito del e-commerce se han desarrollado esquemas de pago electrónico que proporcionan en el mundo digital la misma heterogeneidad y funcionalidad que los sistemas de pago tradicionales
- En la mayoría de los sistemas de pagos electrónicos disponibles, los pagos se realizan a través de **redes abiertas como el Internet**
 - pero la correspondencia entre los pagos electrónicos y la transferencia del valor real es realizada y garantizada por los bancos, a través de los **sistemas financieros de compensación**
 - estos sistemas utilizan para su funcionamiento las redes cerradas de las instituciones bancarias, las cuales son consideradas comparativamente más seguras



Introducción



Introducción

- En general, los pagos electrónicos involucran a un **comprador y a un vendedor**
 - pero pueden existir sistemas y protocolos que involucren a TTPs
 - y también puede existir algún tipo de entidad que ejerza de mediador para la **resolución de disputas**
 - por lo general, las disputas se resuelven fuera del sistema de pago, y en muchos casos el protocolo ni siquiera especifica cómo gestionarlas



Introducción

- Existen varias formas de clasificar los sistemas de pagos electrónicos, y dependen de:
 - **cuando el vendedor contacta con el banco** para verificar el proceso de pago (ej. si está autorizado)
 - **cuando el comprador procede con la transacción y carga de dinero** en la cuenta del vendedor
 - **la cantidad de dinero** implicada en cada transacción



Introducción - Cuando el vendedor contacta con el banco

- Los sistemas de pagos electrónicos se pueden clasificar según **cuando el vendedor contacta con el banco**:
 - **On-line**: antes de enviar el producto, el vendedor contacta con la entidad financiera para verificar la validez del pago del comprador
 - **Off-line**: cierto tiempo después de que el vendedor haya aceptado el pago y enviado el producto, realiza el depósito del dinero que le ha dado el comprador
 - para que la entidad financiera lo verifique y lo ingrese en su cuenta
 - es decir, el vendedor no contacta con el banco durante el proceso de compra-venta



Introducción -

Cuando el comprador procede con la transacción

- Existe otra forma de clasificar los pagos electrónicos, atendiendo al momento en que se retira el dinero de la cuenta del **comprador**:
 - **Sistemas de pre-pago**: el comprador ve decrementada su cuenta bancaria antes de realizar la compra
 - este método se correspondería con los sistemas de **monedero electrónico y tarjetas telefónicas**
 - éste sería el sistema más análogo al papel moneda tradicional
 - **Sistemas de pago instantáneo**: cuando al comprador se le realiza el cargo en cuenta justo en el momento de realizar la compra
 - se correspondería con los sistemas actuales de pagos con **tarjeta de débito**
 - **Sistemas de post-pago**: cuando el comprador realiza la compra, el banco asegura al vendedor que se le hará efectiva la cantidad acordada
 - pero el comprador sólo verá decrementada su cuenta cierto tiempo después de haberse realizado la compra



Introducción - La cantidad de dinero

- Otro criterio de clasificación es **según la cantidad implicada en la transacción**. De esta forma se clasifican los pagos electrónicos como:
 - **Macropagos**: cualquier pago superior a 10 euros
 - **Pagos**: la cantidad está comprendida entre 1 y 10 euros
 - **Micropagos**: cualquier pago inferior a 1 euro
- Normalmente, los **pagos inferiores a 10 euros** presentan el problema del coste de implementación
 - no tendría sentido utilizar un sistema de pago cuyo coste económico sea de orden de magnitud o superior al importe de la transacción



Introducción

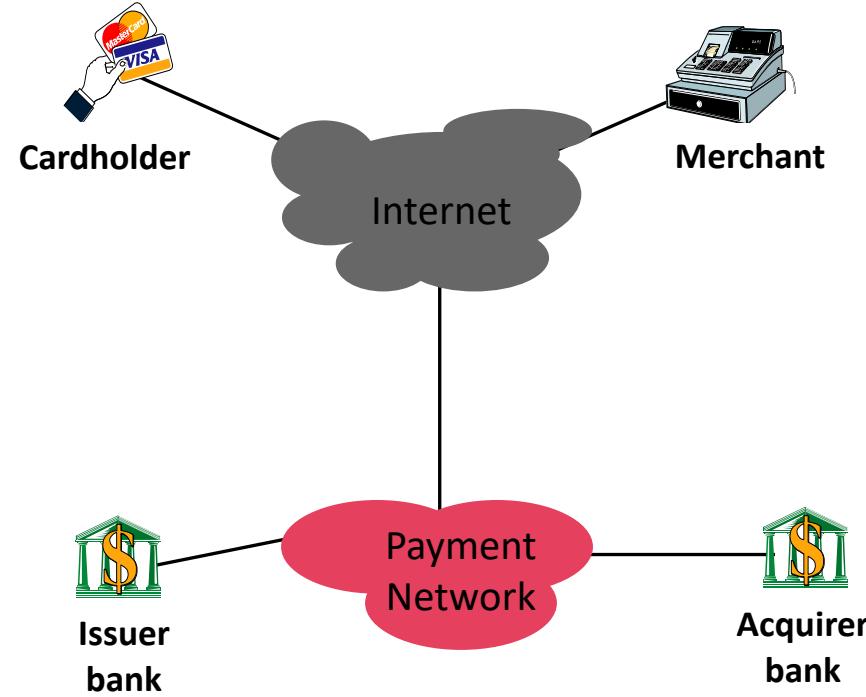
- Hay muchos protocolos que gestionan el proceso de pago online, tales como:

- **On-line y trazables:**

- First Virtual
 - CyberCash
 - iKP
 - SET
 - ...

- **Micropagos:**

- PayPal
 - Google Checkout
 - Amazon Payments
 - iTunes Store
 - ...

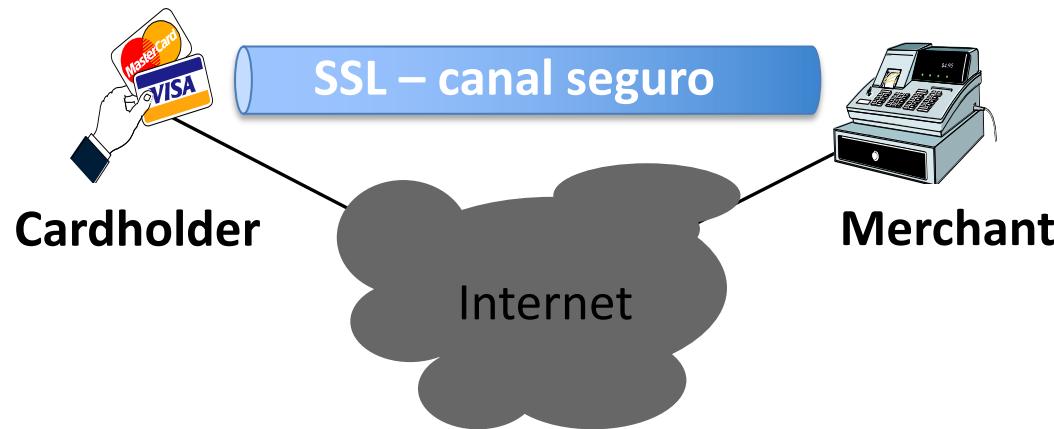


Introducción

- Sin embargo, existen numerosos problemas de seguridad en estos medios de pagos:
 - Ataques de escucha
 - Suplantación de identidad (cliente o comerciante)
 - Generación de dato falso
 - Modificación del dato
 - Etc.
- Para evitar estos problemas de seguridad, los protocolos suelen implementar:
 - Primitivas criptográficas
 - Mecanismos de autenticación y autorización de usuarios
 - Firma digitales
 - Certificados digitales
 - Etc.

Introducción - un poco de historia ...

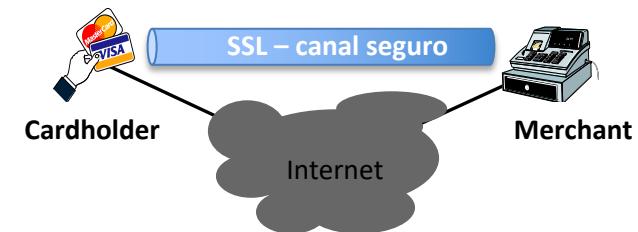
- Inicialmente se aplicaba el **protocolo Secure Sockets Layer (SSL)** como medida de protección de los canales de comunicación



- SSL es un protocolo de propósito general (como TLS) para establecer conexiones seguras
 - **No es un protocolo de pago**, pero se usaba por seguridad
 - SSL lo creó originalmente Netscape (1994).
 - La última versión: SSLv3 – ¡¡ESTÁ YA OBSOLETA!!

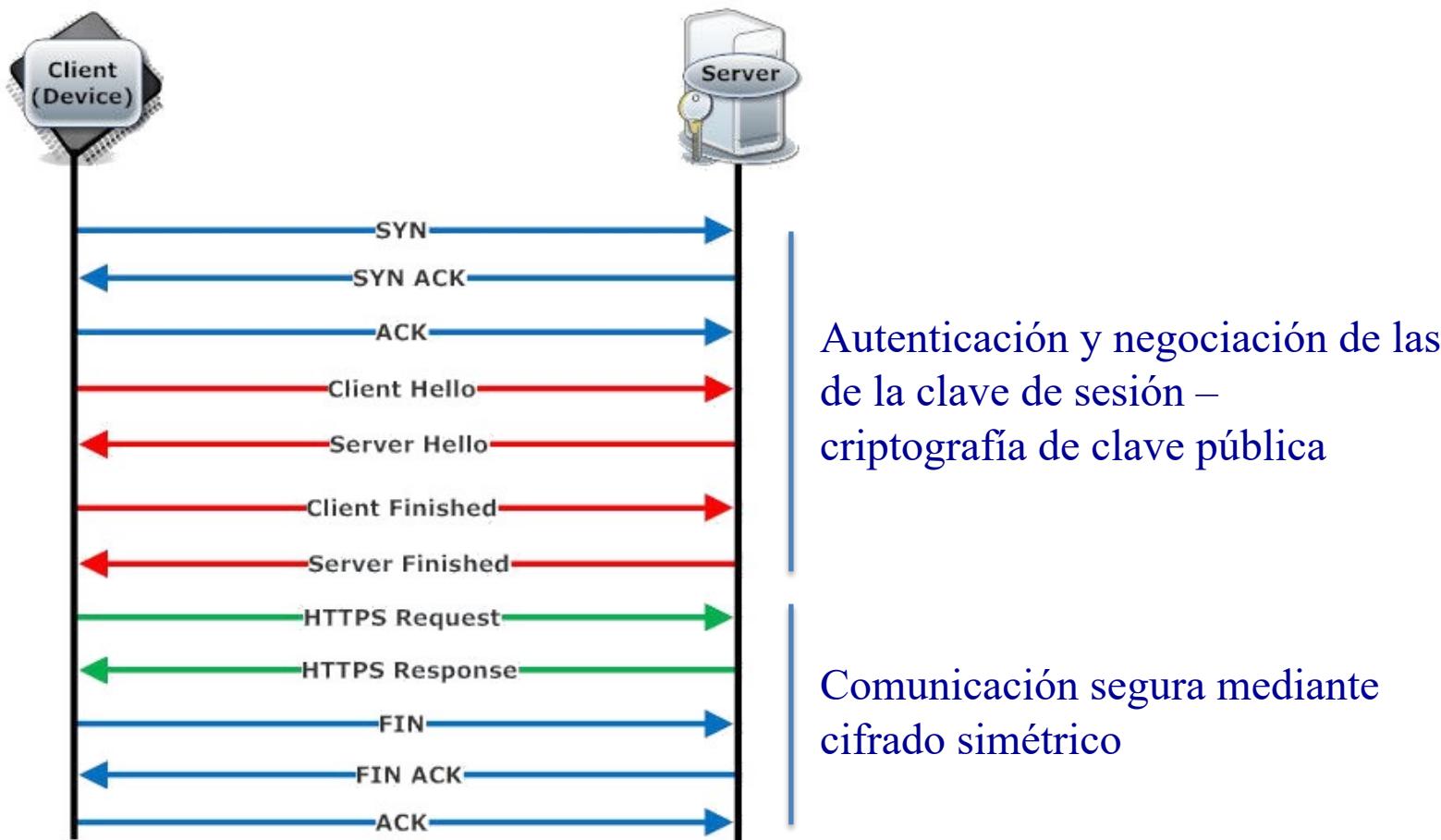
Introducción - un poco de historia ...

- SSL inicialmente se aplicaba porque proporcionaba el uso de la cripto. híbrida:
 - criptografía asimétrica para:
 - la autenticación usando certificados digitales
 - la negociación de las claves de sesión con RSA o Diffie-Hellman
 - criptografía simétrica para:
 - el cifrado de punto a punto usando DES, 3DES, RC2, RC4 o IDEA
- SSL autentifica al servidor y al cliente
 - utilizando certificados digitales X.509 v3
 - la autenticación con certificados es completamente opcional para ambas partes, el servidor y el cliente
- SSL también asegura la integridad de los datos
 - mediante MAC y una clave secreta para dicha MAC
 - pero uso MD5 o SHA-1 para la MAC ☹



Introducción - un poco de historia ...

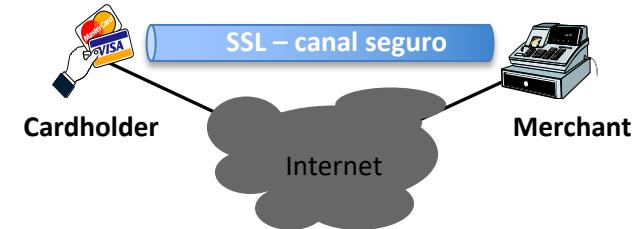
- Originalmente, SSL proveía **confidencialidad en el pago electrónico**
 - Garantizaba la creación de un canal seguro entre cliente y servidor



Introducción - un poco de historia ...

- Sin embargo, SSL presentaba algunos **problemas**:
 - Sólo protege transacciones entre dos puntos
 - mientras que una transacción electrónica basada en una tarjeta de crédito involucra al menos a un banco
 - SSL no protege al comprador frente al vendedor
 - el vendedor puede obtener información de la tarjeta que podría utilizar en un futuro de forma ilícita
 - No hay mecanismos de autentificación de tarjetas
 - no está la entidad bancaria quien autentica la entidad interesada
 - No hay mecanismos de facturación o de gestión de recibos
 - cualquier reclamación queda a la buena voluntad del vendedor

Por tanto, se requerían de otros tipos de protocolos más específicos ...



Protocolo SET (Secure Electronic Transactions)

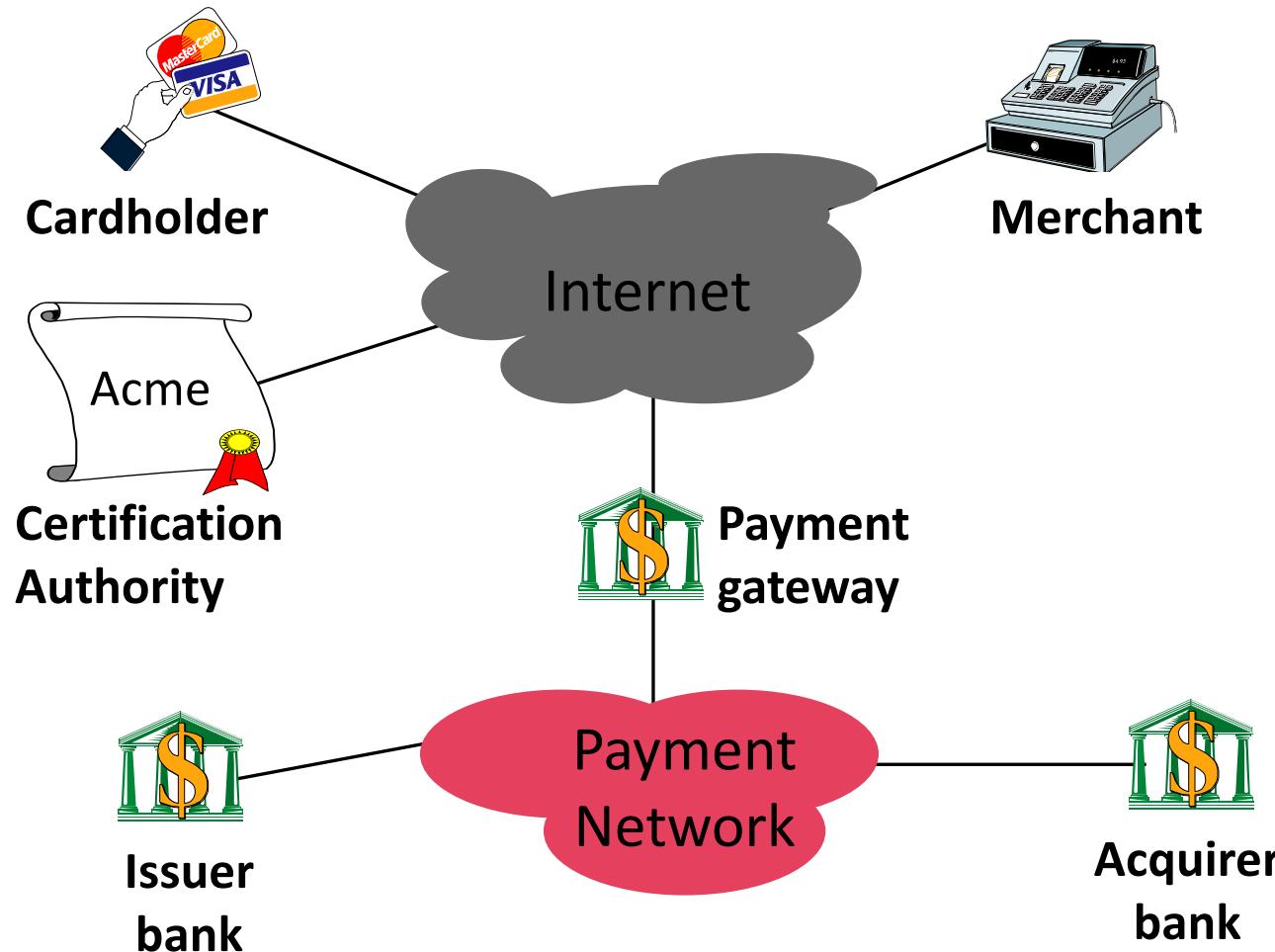


- Protocolo desarrollado en 1996 por VISA y Mastercard (+ American Express), en colaboración con:
 - IBM
 - Microsoft
 - Verisign
 - RSA
 - Netscape
 - GTE
- El objetivo era proporcionar seguridad a las **transacciones electrónicas basadas en tarjetas de crédito**
 - para poder reducir el fraude mercantil
 - y garantizar el pago a través de esas mismas redes



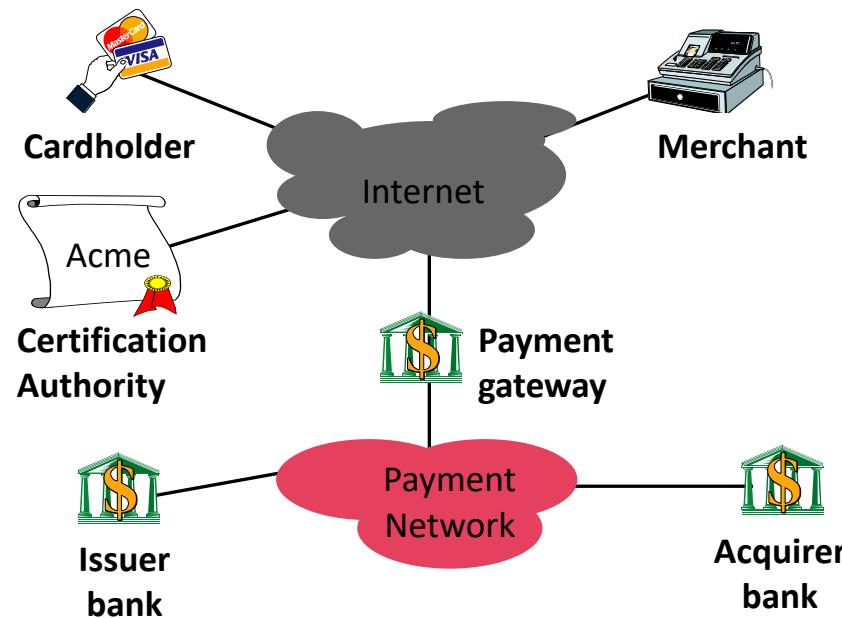
Protocolo SET (Secure Electronic Transactions)

- Respeta la infraestructura de pago existente:



Protocolo SET (Secure Electronic Transactions)

- A diferencia de SSL, fue diseñado para el comercio electrónico
- Sin embargo, no es un sistema de pago en sí mismo, sino un **conjunto de protocolos de seguridad y de formatos estándar**
 - que permiten a los usuarios usar de una forma segura a través de Internet, la infraestructura ya existente de tarjetas de crédito



Protocolo SET (Secure Electronic Transactions)

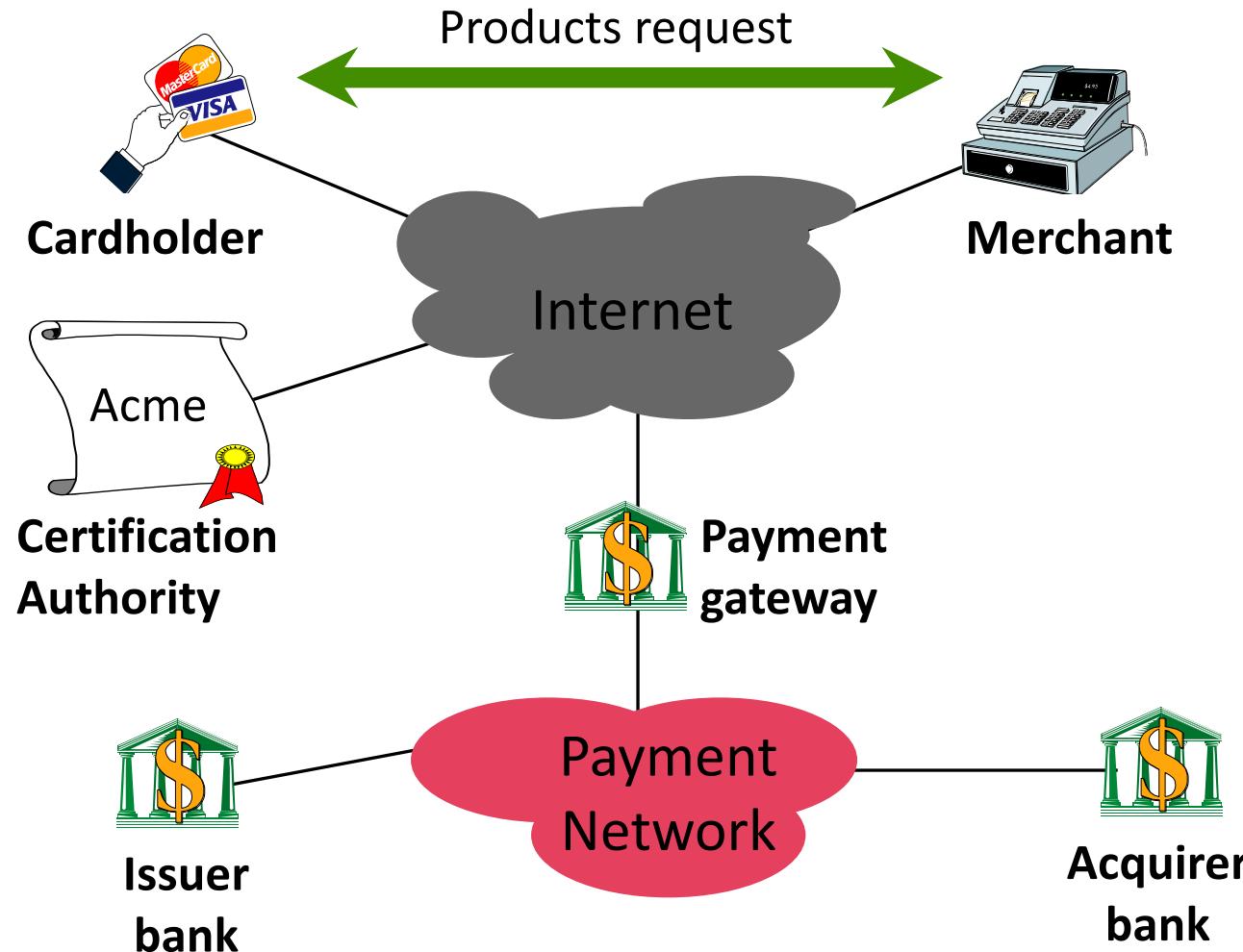
- SET proporciona un conjunto de servicios de seguridad:
 - **Confidencialidad** en las comunicaciones entre las entidades que intervienen en la transacción
 - **Autenticación**, a través del uso de certificados digitales X.509
 - Todas las entidades, incluyendo el cliente, el vendedor y la pasarela de pago, han de tener certificados X.509 [... OBLIGATORIO]
 - Es necesario el servicio de una o más Autoridades de Certificación
 - **Privacidad**, porque la información sólo está disponible para las diferentes entidades cuando y donde es necesario
 - **Integridad** por las firmas digitales
 - **Reduce las disputas** debido al no repudio
 - **Autorización** de pago y, por tanto, confirmación de la transacción y garantía de pago al vendedor

Protocolo SET (Secure Electronic Transactions)

- Pasos del protocolo SET:
 1. **Petición de producto**
 2. **Inicialización:**
 - envío de certificados
 3. **Información del pedido e instrucciones de pago:**
 - descripción de la compra
 4. **Petición de autorización:**
 - vendedor-pasarela y pasarela-banco emisor
 5. **Aprobación de autorización:**
 - el banco emisor autoriza el pago
 6. **Finalización:** el vendedor reclama la cantidad a la pasarela
 - Petición de compensación hacia el banco del vendedor

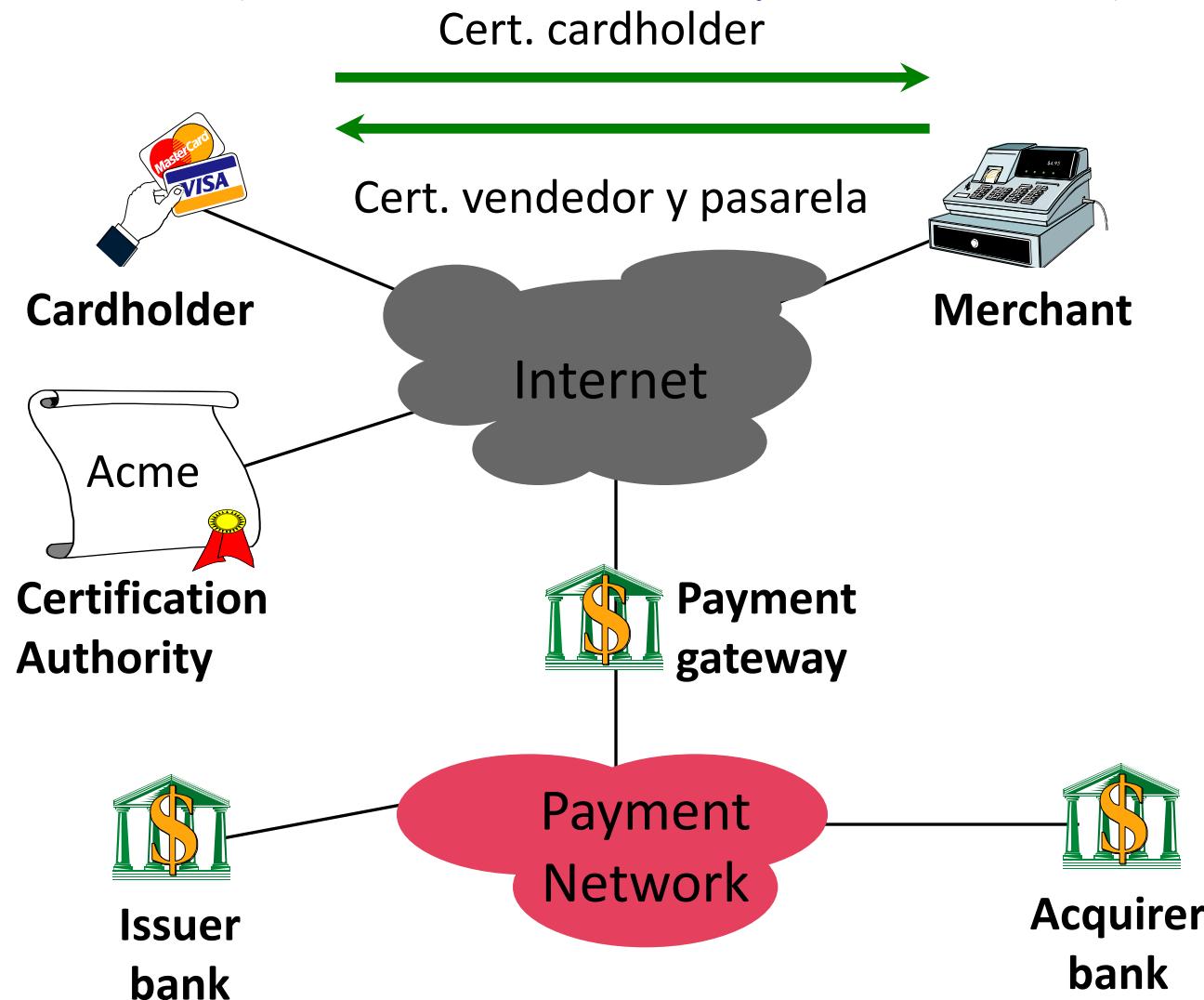
Protocolo SET (Secure Electronic Transactions)

1. Petición del producto



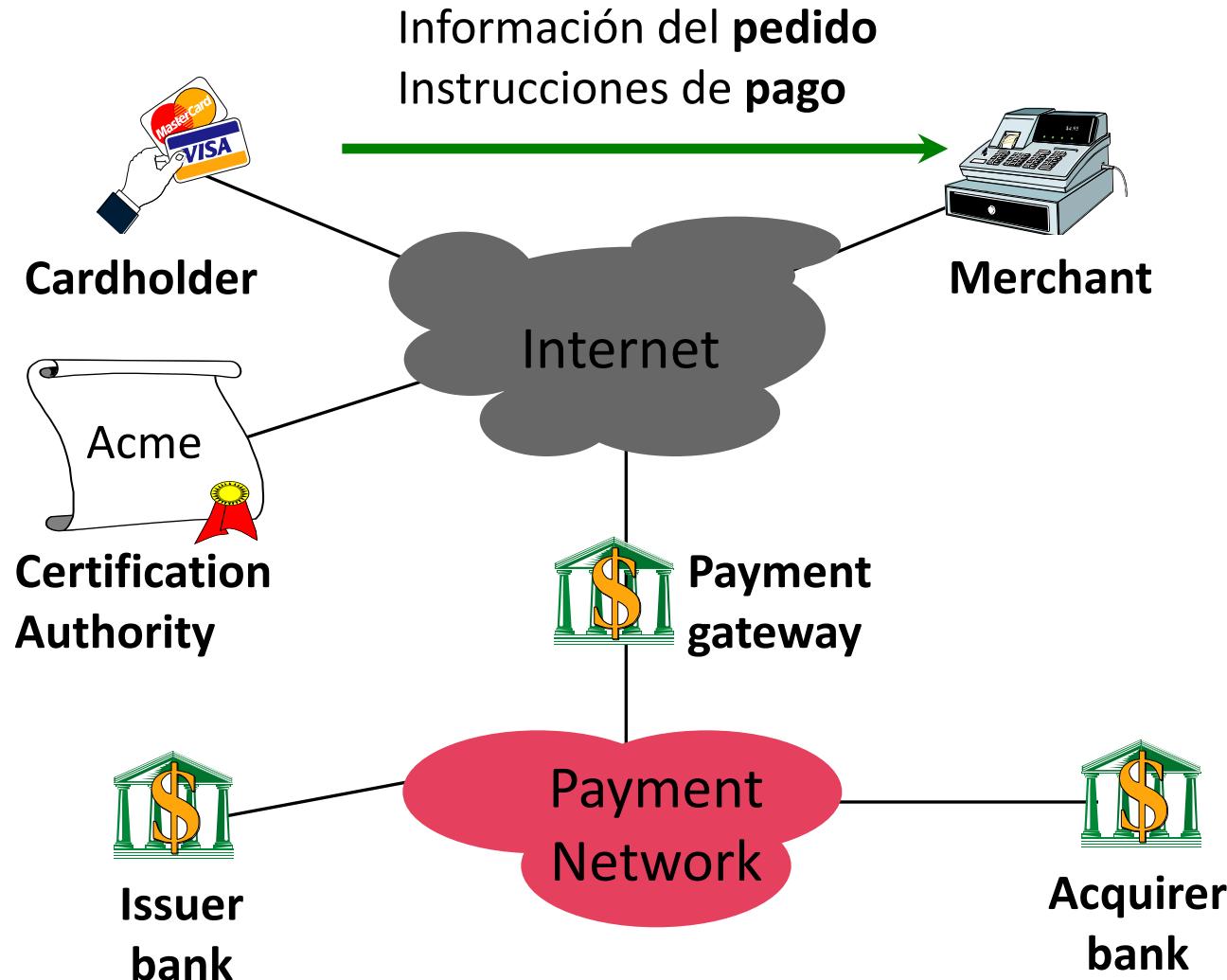
Protocolo SET (Secure Electronic Transactions)

2. Inicialización (envío de certificados y autenticación)



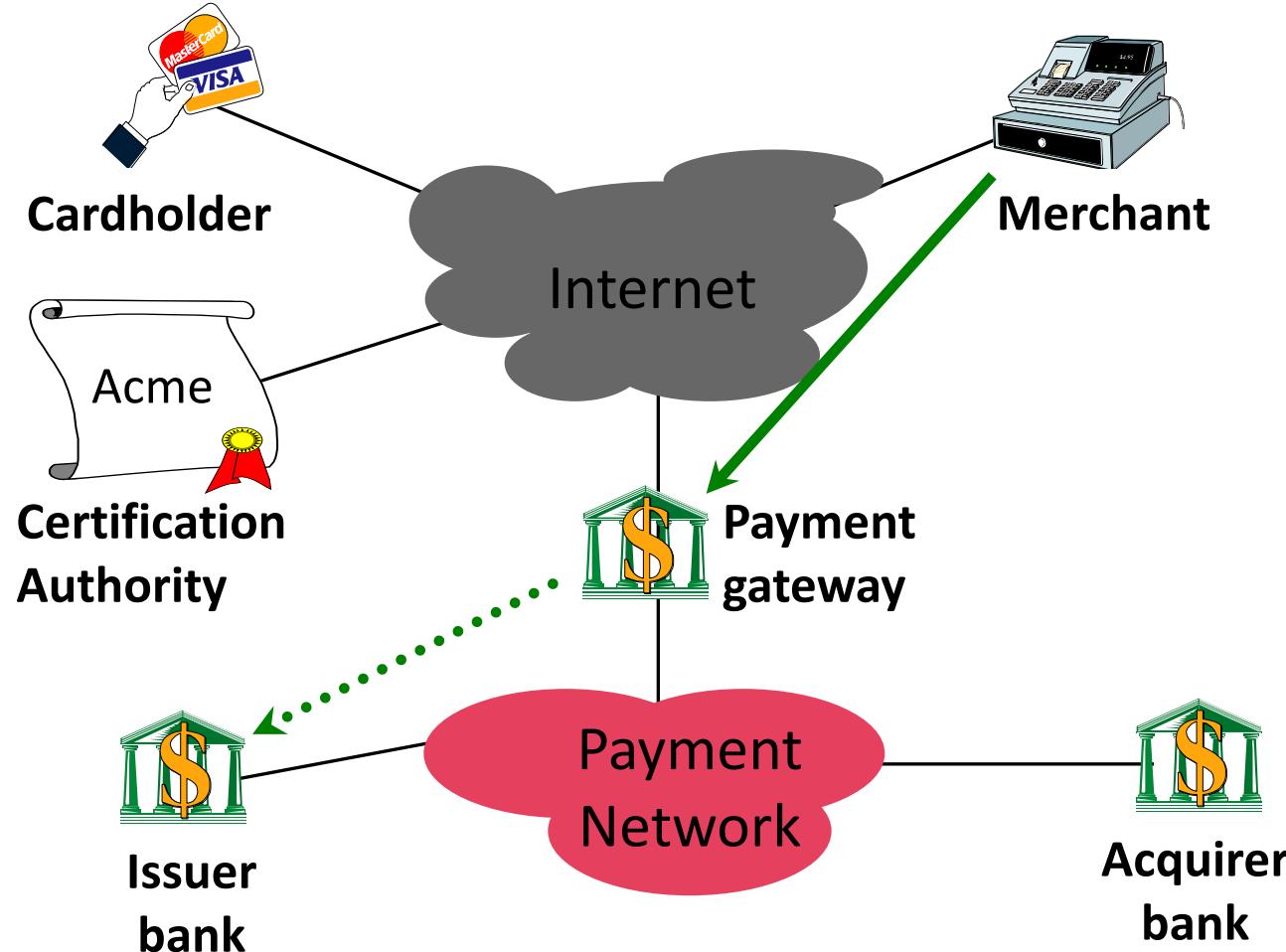
Protocolo SET (Secure Electronic Transactions)

3. Información del pedido e instrucciones de pago



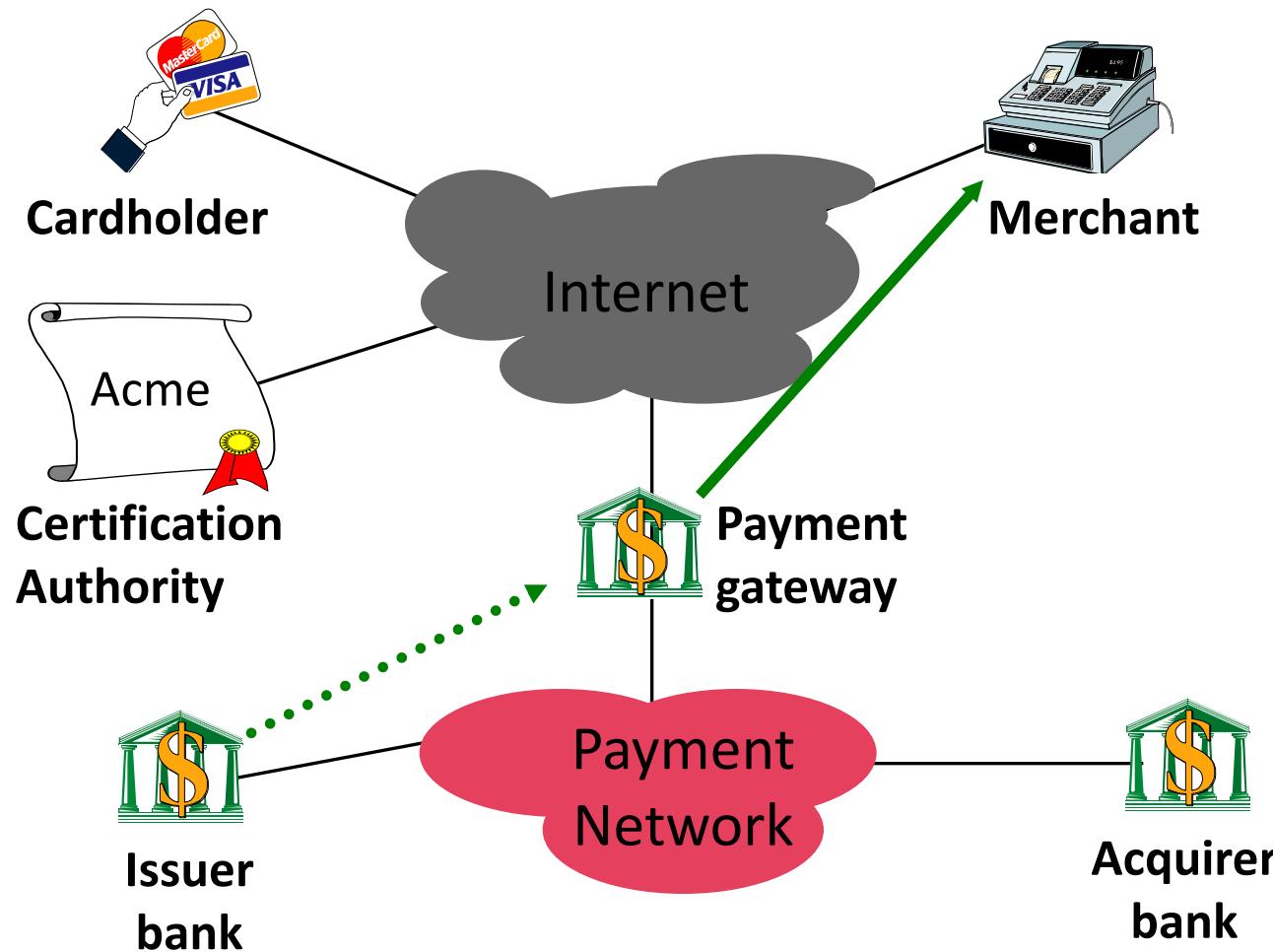
Protocolo SET (Secure Electronic Transactions)

4. Petición de autorización



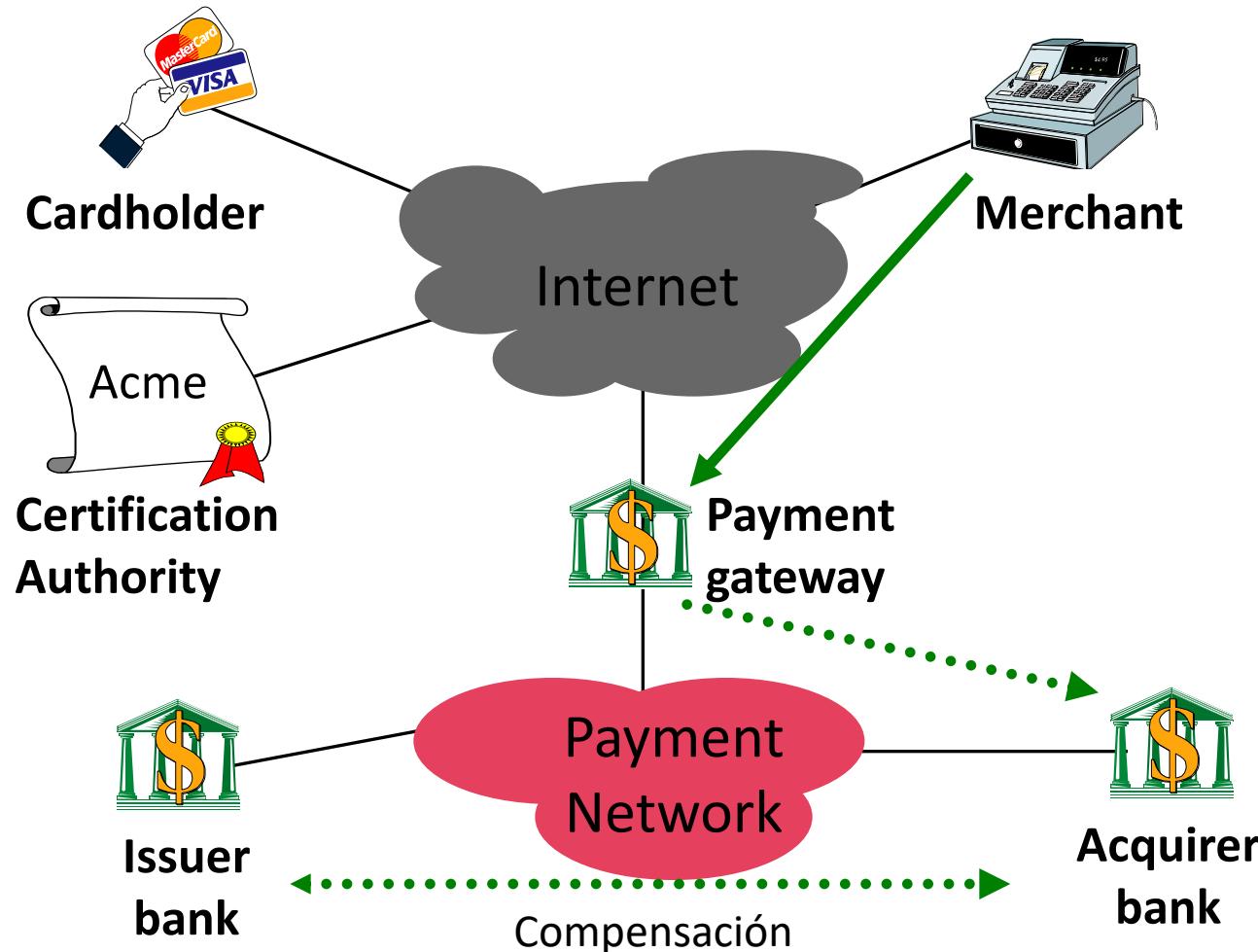
Protocolo SET (Secure Electronic Transactions)

5. Aprobación de autorización



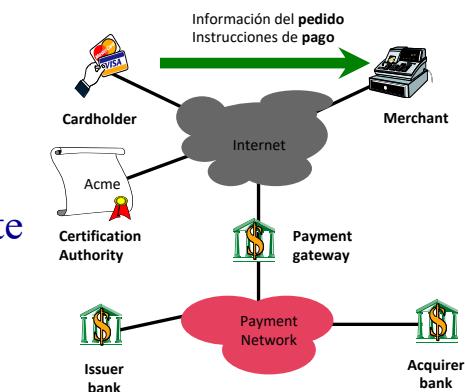
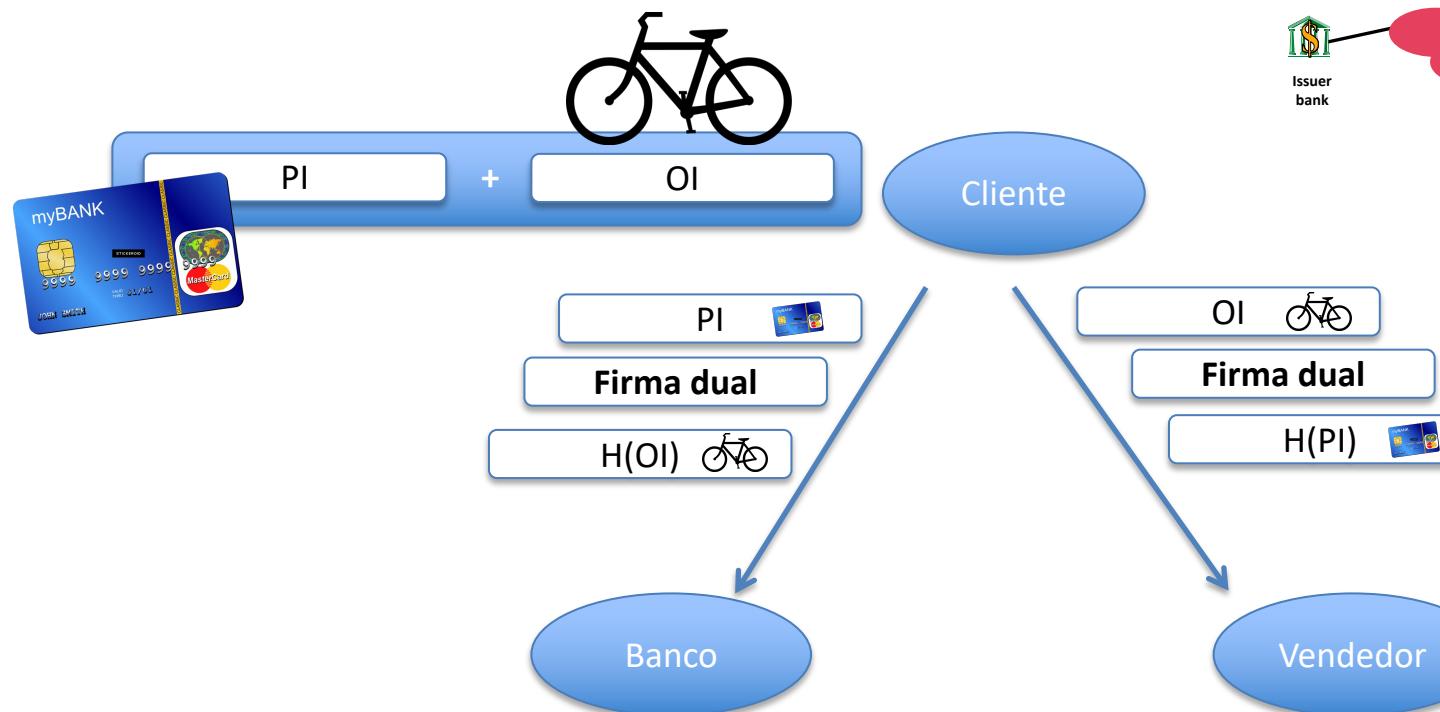
Protocolo SET (Secure Electronic Transactions)

6. Finalización



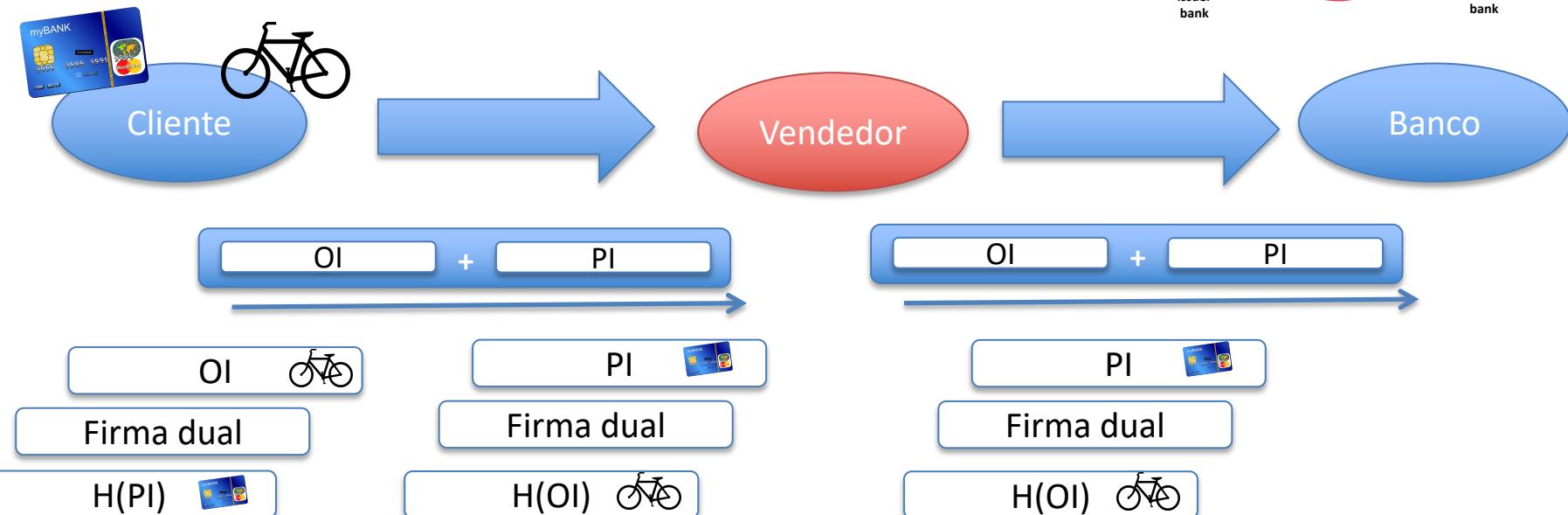
FIRMA DUAL (en SET)

- SET introduce una importante innovación técnica: la **firma dual**
 - El propósito de este tipo de firma es **enlazar** dos mensajes que han de ir a receptores distintos:
 - la información del **pago** (Payment Information) al banco
 - la información del **pedido** (Order Information) al comerciante



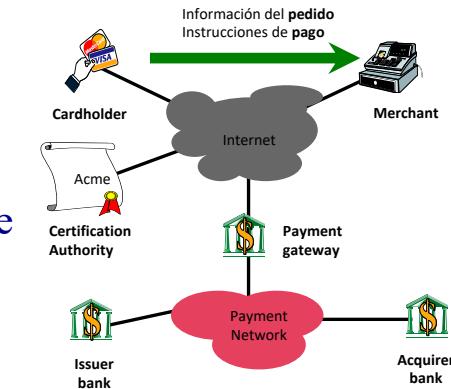
FIRMA DUAL (en SET)

- SET introduce una importante innovación técnica: la **firma dual**
 - El propósito de este tipo de firma es **enlazar** dos mensajes que han de ir a receptores distintos:
 - la información del **pago** (Payment Information) al banco
 - la información del **pedido** (Order Information) al comerciante



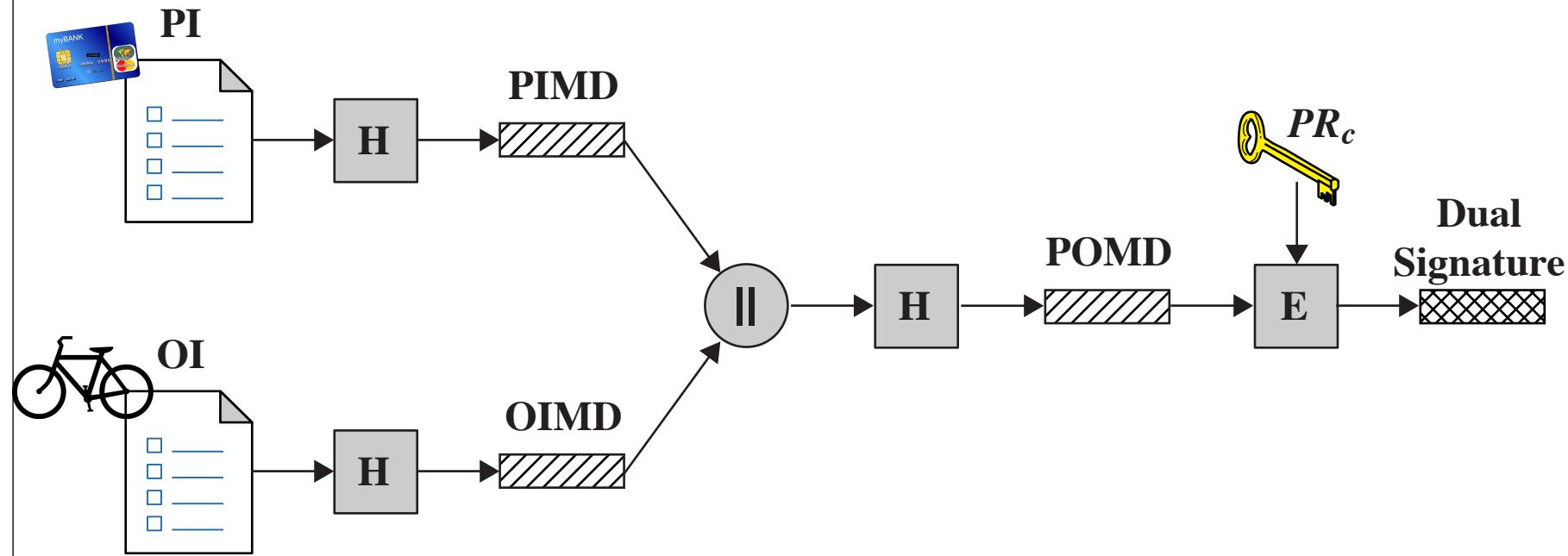
FIRMA DUAL (en SET)

- SET introduce una importante innovación técnica: la **firma dual**
 - El propósito de este tipo de firma es **enlazar** dos mensajes que han de ir a receptores distintos:
 - la información del **pago** (Payment Information) al banco
 - la información del **pedido** (Order Information) al comerciante
- Con la firma dual se ofrece mayor **privacidad** al cliente si ambos ítems (PI y OI) se mantienen por separado:
 - ni el comerciante necesita conocer el número de tarjeta del cliente
 - ni el banco necesita conocer los detalles del pedido del cliente
 - pero también es necesario que ambos ítems queden enlazados de alguna forma, para una posible **resolución de disputas** posterior



¡¡HAY NO-REPUDIO!!

FIRMA DUAL (en SET)



PI = Payment Information

OI = Order Information

H = Hash function (SHA-1)

|| = Concatenation

PIMD = PI message digest

OIMD = OI message digest

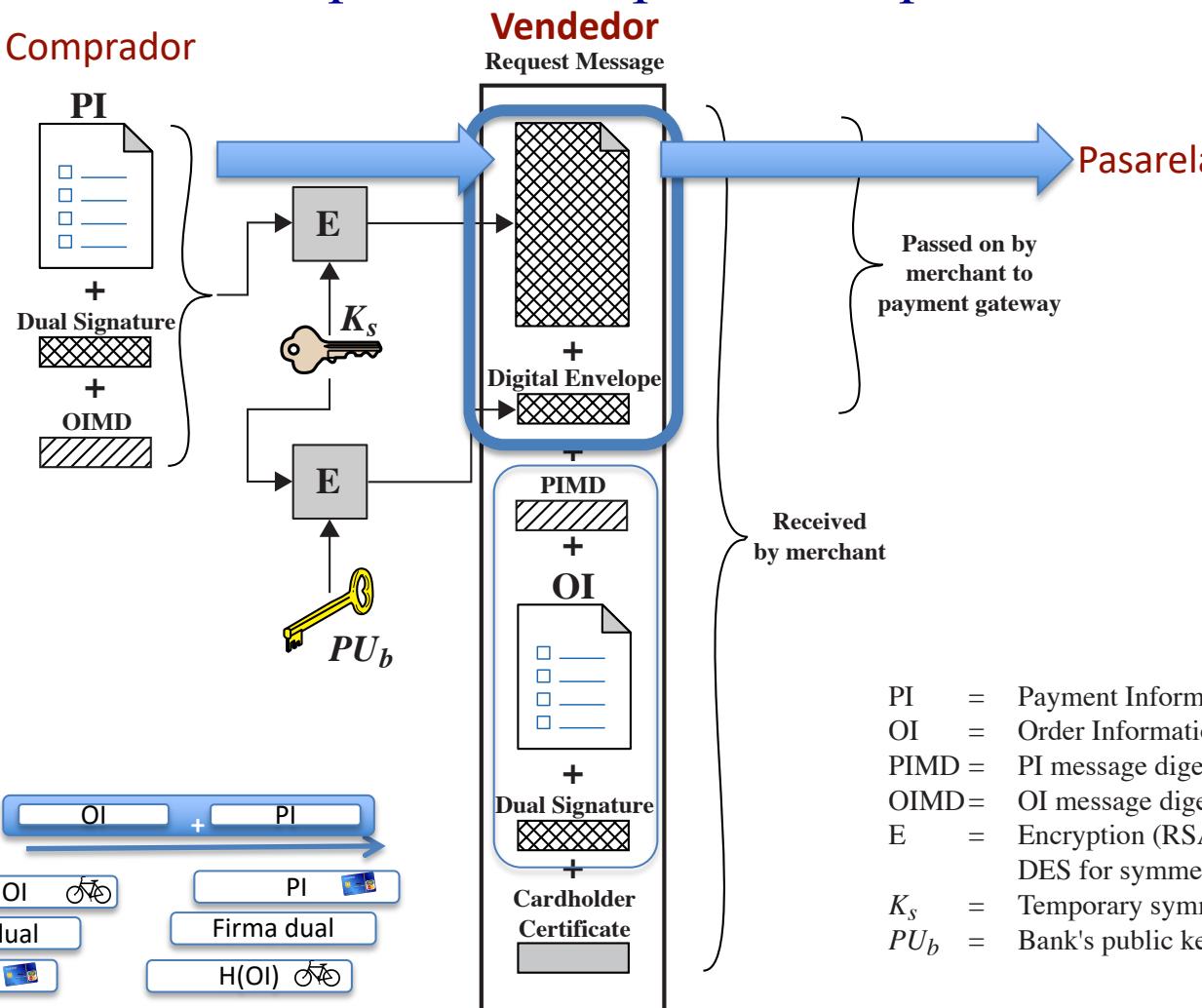
POMD = Payment Order message digest

E = Encryption (RSA)

PR_c = Customer's private signature key

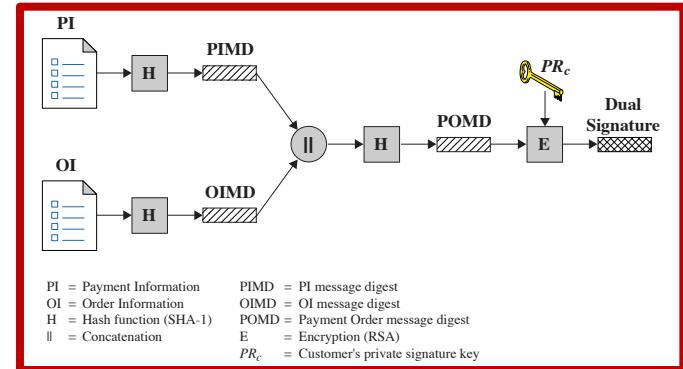
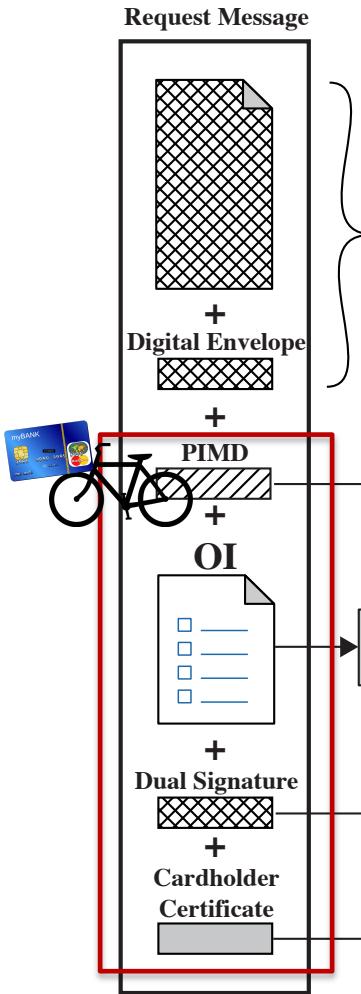
FIRMA DUAL (en SET)

- Petición de compra enviada por el comprador al vendedor:



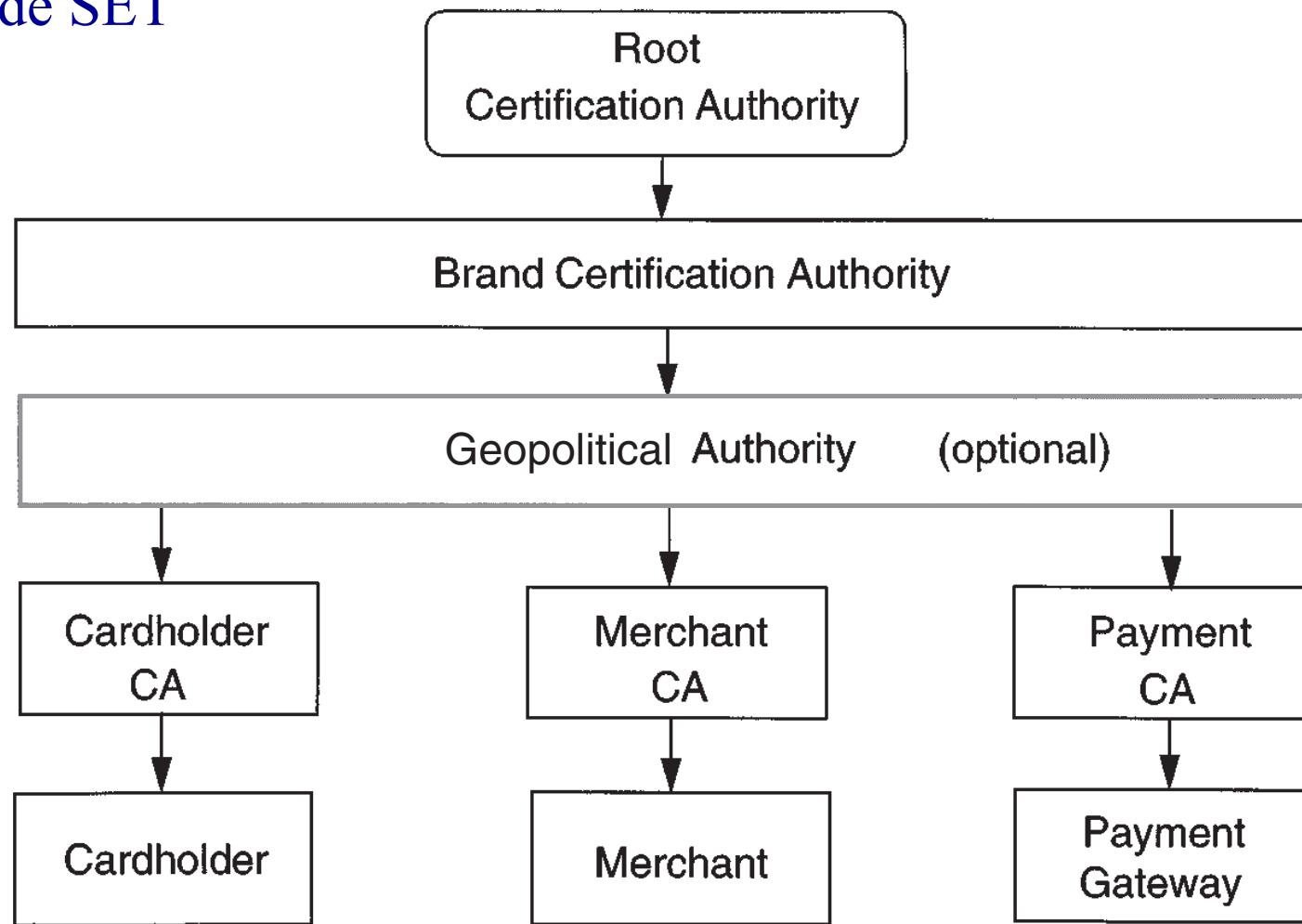
FIRMA DUAL (en SET)

- Verificación del vendedor:



Protocolo SET (Secure Electronic Transactions)

- PKI de SET



Protocolo SET (Secure Electronic Transactions)

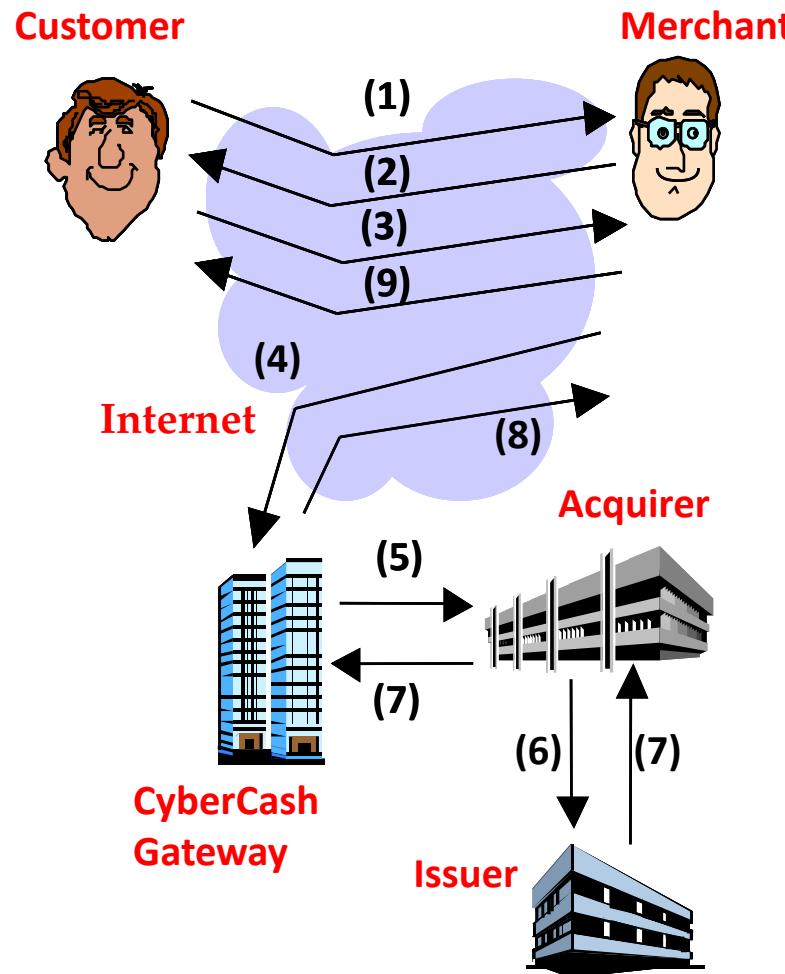
- Ventajas de uso del SET:
 - Muy seguro y bien diseñado
 - Garantiza:
 - autenticación, confidencialidad, integridad y no-repudio
 - Garantiza privacidad
 - evita que el vendedor acceda a los datos de la tarjeta
 - evita que el banco acceda a la información de los productos comprados
- Desventajas de uso del SET:
 - Es dependiente de algoritmos específicos (RSA, DES, SHA1)
 - Gestión compleja de certificados digitales
 - fuerte esfuerzo para la implantación (especialmente para el vendedor)
 - No adaptado a micropagos

Protocolo Cybercash

- Se basa en el uso de una **pasarela propia** que gestiona los pagos electrónicos
 - permite el uso de cualquier tipo de tarjeta
 - integra el software de cliente (**cyberwallet**) con la red financiera del banco del comprador
- Se realiza la **autenticación** de todas las entidades y el **cifrado** de los datos relativos al pago



Protocolo Cybercash



1. Purchase order (description).
2. Payment request (price).
3. Payment order (signed by the CyberWallet).
4. Redirection of the payment order.
5. Verification of the order. Authorization request.
6. Request for authorization of the issuer bank.
7. Authorization reply (acquirer and gateway).
8. Sending the **encrypted bills** (customer and merchant)
9. Redirection of the **customer's bill**.

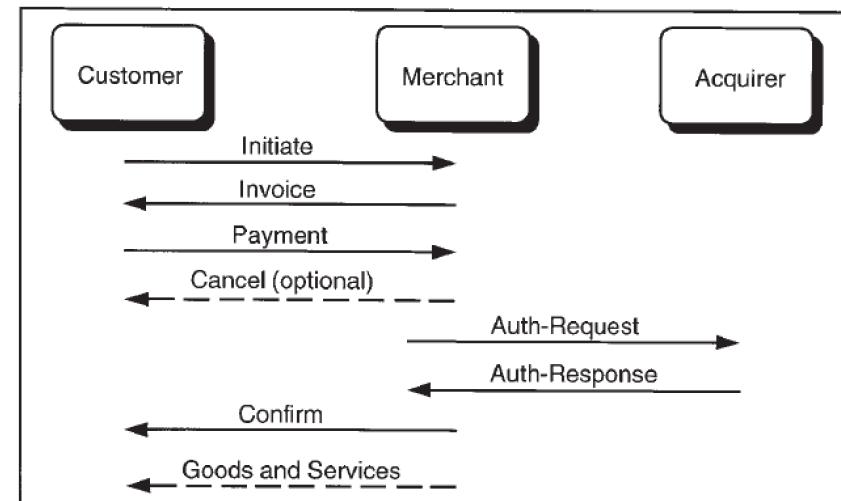
Protocolo Cyberscash

- Sus principales problemas son:
 - Parte de la información del cliente es conocida por la pasarela “privada”, por lo que se pueden analizar los hábitos del cliente
 - **problema de privacidad** que no existía en SET
 - Hace uso de **DES y RSA (1024 bits)**
- Tras caer en bancarrota, Verisign adquirió los derechos sobre la marca y el protocolo de pago
- Posteriormente, Paypal compró la solución a Verisign



Protocolos i-Key Protocol (iKP)

- iKP ($i = 1, 2, 3$) es una familia de protocolos de pago, basado en criptografía de clave pública, y desarrollado por IBM
- La implantación de los protocolos se realiza de forma gradual para conseguir **un pago seguro (multiparte – si se aplica 2/3KP)** completo, requiriendo una **infraestructura de certificación avanzada**
 - Estos protocolos (1KP, 2KP, 3KP) se diferencian entre sí dependiendo del nº de entidades que poseen el certificado digital
- El funcionamiento del pago es equivalente a otros protocolos



Protocolos i-Key Protocol (iKP)

- Los elementos intercambiados en una transacción iKP, y los campos formados por la combinación de tales elementos, son:

Item	Description
CAN	Customer's account number (e.g., credit card number)
ID _M	Merchant ID; identifies merchant to acquirer
TID _M	Transaction ID; uniquely identifies the transaction
DESC	Description of the goods; includes payment information such as credit card holder's name and bank identification number
SALT _C	Random number generated by C; used to randomize DESC and thus ensure privacy of DESC on the M to A link
NONCE _M	Random number generated by a merchant to protect against replay
DATE	Merchant's current date/time
PIN	Customer's PIN which, if present, can be optionally used in 1KP to enhance security
Y/N	Response from card issuer; Yes/No or authorization code
R _C	Random number chosen by C to form CID
CID	A customer pseudo-ID which uniquely identifies C; computed as CID = H(R _C , CAN)
V	Random number generated in 2KP and 3KP by merchant; used to bind the Confirm and Invoice message flows

Item	Description
Common	Information held in common by all parties: PRICE, ID _M , TID _M , DATE, NONCE _M , CID, H(DESC, SALT _C), [H(V)]
Clear	Information transmitted in the clear: ID _M , TID _M , DATE, NONCE _M , H(Common), [H(V)]
SLIP	Payment instructions: PRICE, H(Common), CAN, R _C , [PIN]
EncSlip	Payment instruction encrypted with the public key of the acquirer: PK _A (SLIP)
CERT _X	Public-key certificate of X, issued by a CA
Sig _A	Acquirer's signature: SK _A [H(Y/N, H(Common))]
Sig _M	Merchant's signature in Auth-Request: SK _M [H(H(Common), [H(V)])]
Sig _C	Cardholder's signature: SK _C [H(EncSlip, H(Common))]

Importante: no hay que estudiar ni memorizar estos datos

Protocolos i-Key Protocol (iKP)

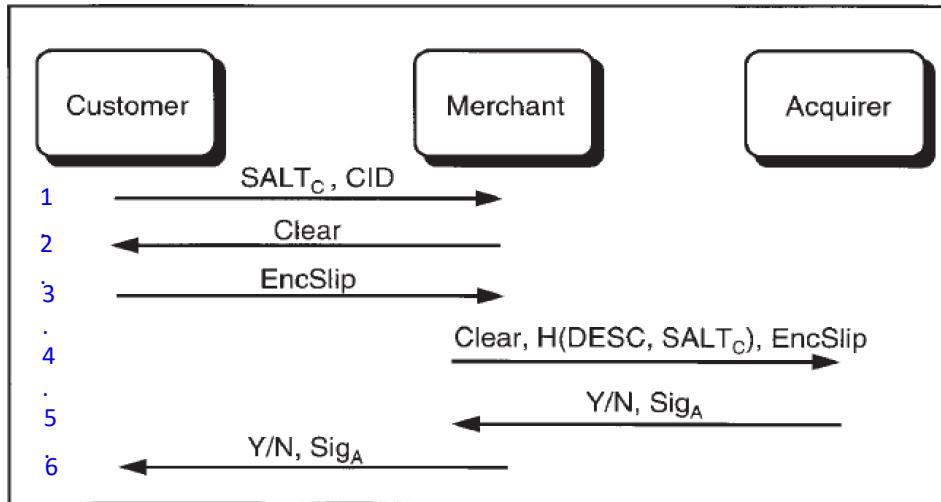
- El **protocolo 1KP** es el más básico
 - Sólo el banco necesita poseer (y distribuir) su certificado digital, CERT_A .
- La información de partida de cada una de las entidades es:

Item	Description
CAN	Customer's account number (e.g., credit card number)
ID_M	Merchant ID; identifies merchant to acquirer
TID_M	Transaction ID; uniquely identifies the transaction
DESC	Description of the goods; includes payment information such as credit card holder's name and bank identification number
SALT_C	Random number generated by C; used to randomize DESC and thus ensure privacy of DESC on the M to A link
NONCE_M	Random number generated by a merchant to protect against replay
DATE	Merchant's current date/time
PIN	Customer's PIN which, if present, can be optionally used in 1KP to enhance security
Y/N	Response from card issuer; Yes/No or authorization code
R_C	Random number chosen by C to form CID
CID	A customer pseudo-ID which uniquely identifies C; computed as $\text{CID} = H(R_C, \text{CAN})$
V	Random number generated in 2KP and 3KP by merchant; used to bind the Confirm and Invoice message flows

Actor	Information Items
Customer	DESC, CAN, PK_{CA} , [PIN], CERT_A
Merchant	DESC, PK_{CA} , CERT_A
Acquirer	SK_A , CERT_A

Protocolos i-Key Protocol (iKP)

- Los pasos del protocolo 1KP son:



Item	Description
Common	Information held in common by all parties: PRICE, ID_M , TID_M , DATE, $NONCE_M$, CID, $H(DESC, SALT_C)$, $[H(V)]$
Clear	Information transmitted in the clear: ID_M , TID_M , DATE, $NONCE_M$, $H(\text{Common})$, $[H(V)]$
SLIP	Payment instructions: PRICE, $H(\text{Common})$, CAN, R_C , [PIN]
EncSlip	Payment instruction encrypted with the public key of the acquirer: PK_A (SLIP)
CERT_X	Public-key certificate of X, issued by a CA
Sig_A	Acquirer's signature: $SK_A[H(Y/N, H(\text{Common}))]$
Sig_M	Merchant's signature in Auth-Request: $SK_M[H(H(\text{Common}), [H(V]))]$
Sig_C	Cardholder's signature: $SK_C[H(\text{EncSlip}, H(\text{Common}))]$

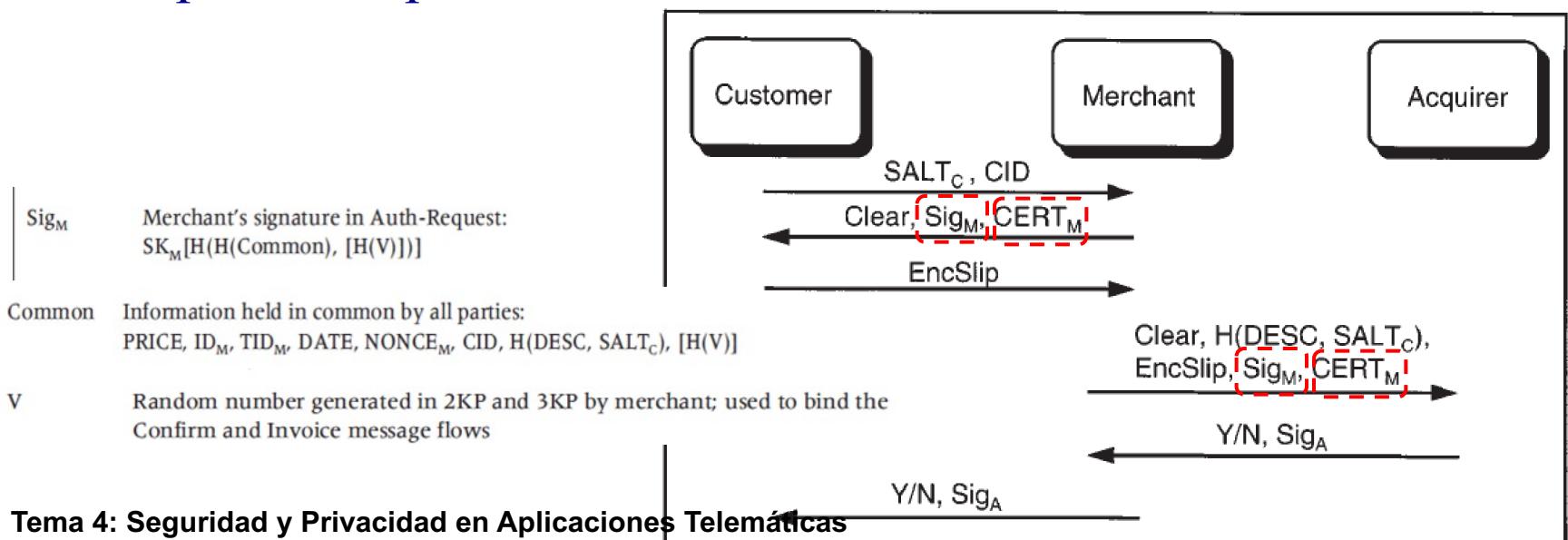
- Las desventajas de uso de 1KP son:

- El cliente se autentica utilizando sólo un número de tarjeta de crédito y, opcionalmente, un PIN, en lugar de firmas digitales
- El vendedor no se autentica ni ante el cliente ni ante al banco
- Ni el vendedor ni el cliente proporcionan evidencias de intervención en la transacción

Protocolos i-Key Protocol (iKP)

- En el **protocolo 2KP**, además del banco, cada vendedor necesita tener un par *<clave pública, clave privada>*, y está obligado a distribuir su certificado $CERT_M$ al cliente y al banco
- La información de partida de cada una de las entidades es:
- Los pasos del protocolo son:

Actor	Information Items
Customer	DESC, CAN, PK_{CA} , $CERT_A$
Merchant	DESC, PK_{CA} , $CERT_A$, SK_M , $CERT_M$
Acquirer	PK_{CA} , SK_A , $CERT_A$



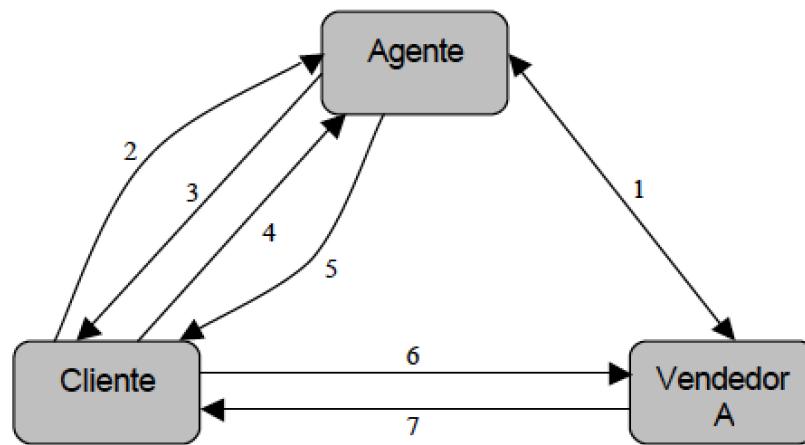
Micropagos

- En algunos escenarios hay que transferir una cantidad muy pequeña (**micropago**), y, por ello, hay que buscar la forma más eficiente y económica posible de hacerlo
 - minimizando el tráfico y los recursos utilizados, para que los costes de realizar el pago sean mínimos en comparación el pago en sí mismo
- Para reducir los costes e pueden utilizar varias soluciones:
 - servicios de prepago
 - autorizaciones off-line (ej. pagos en efectivos)
 - agrupación de facturas para el micropago por lotes
 - soluciones “online” un poco más complejas usando criptografía
 - en este caso, la criptografía de clave pública no es la más adecuada puesto que resulta costosa, e incluso, algunos criptosistemas simétricos pueden ser cuestionables
 - cada vez más se aplica el empleo de funciones hash
 - » sin embargo, conlleva también a la imposibilidad de garantizar el e no-repudio

Protocolo Millicent

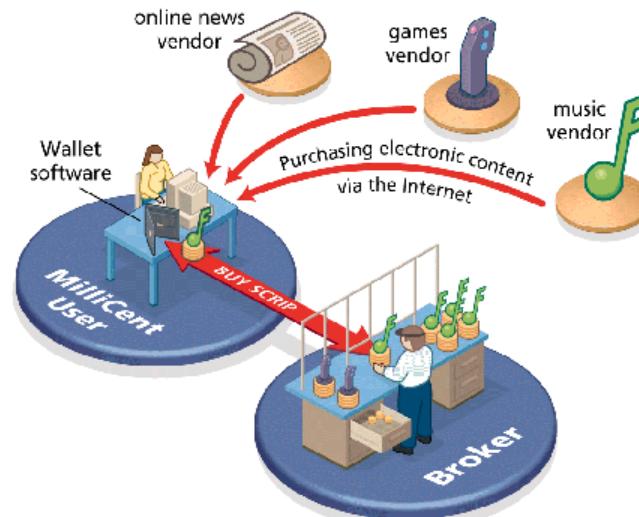
- Un ejemplo de protocolo para gestionar el micropago es **Millicent**:
 - basado en criptografía simétrica y NO utiliza procesamiento online
- Además de clientes y comerciantes, en Millicent existe la figura del **agente de negocios** (posiblemente una institución financiera)
- El sistema utiliza una forma de moneda electrónica, el **scrip**
 - los scrips vienen a ser “cupones electrónicos” que representan dinero, con los que el comprador obtiene la mercancía del vendedor
 - Para un cliente no sería eficiente comprar lotes de scrips para cada vendedor sino más bien para un conjunto de vendedores
 - se puede suponer que, durante un periodo de tiempo las compras de un cliente a varios comerciantes alcanzarán un importe equivalente a un macropago
- La función principal del **agente de negocios / broker** es la de vender a cada cliente, y dentro de un mismo lote mixto, scrips de distintos vendedores

Protocolo Millicent



1. Compra-Venta de scrips de A
2. Envío de scrips de agente
3. Envío de scrips de A
4. Envío de producto

2. Compra de scrips de agente (macropago)
4. Compra de scrips de A (micropago mediante scrips de agente)
6. Petición producto + micropago mediante scrips de A



Protocolo Millicent

- El modelo multiparte (cliente, vendedor y bróker) ayuda a tener cierto grado de **anonimato** por parte del comprador:
 - el agente conoce la identidad del comprador y su número de tarjeta de crédito, pero nunca llega a conocer qué producto compra
 - el vendedor sabe lo que el cliente compra, pero desconoce su identidad
- Hay otros muchos sistemas de micropago, como:
 - Subscrip
 - Kleline
 - Flattr
 - M-coin
 - etc.

PRIVACIDAD DE LOS USUARIOS EN APLICACIONES



Conceptos generales

- La **privacidad** puede definirse como:
 - *el derecho de los individuos y entidades de proteger, salvaguardar y controlar el acceso, almacenamiento, distribución y uso de información sobre su “propia persona”*
 - ¿Qué información es necesario proteger?
 - Dependerá de lo que el usuario considere información privada
 - Identidad, localización, preferencias, rutinas, ...
- **Confidencialidad NO es equivalente a privacidad**
 - Confidencialidad es relativa a los datos mientras que la privacidad es relativa a las personas
 - Por tanto: **Privacidad ≠ Confidencialidad**
 - Confidencialidad := mantener datos en secreto
 - Privacidad := mantener la integridad de la persona
- Hay varias formas de violar la privacidad, como:
 - Rastreo de actividad en la red
 - Análisis de tráfico

Conceptos generales - rastreo de act. en la red

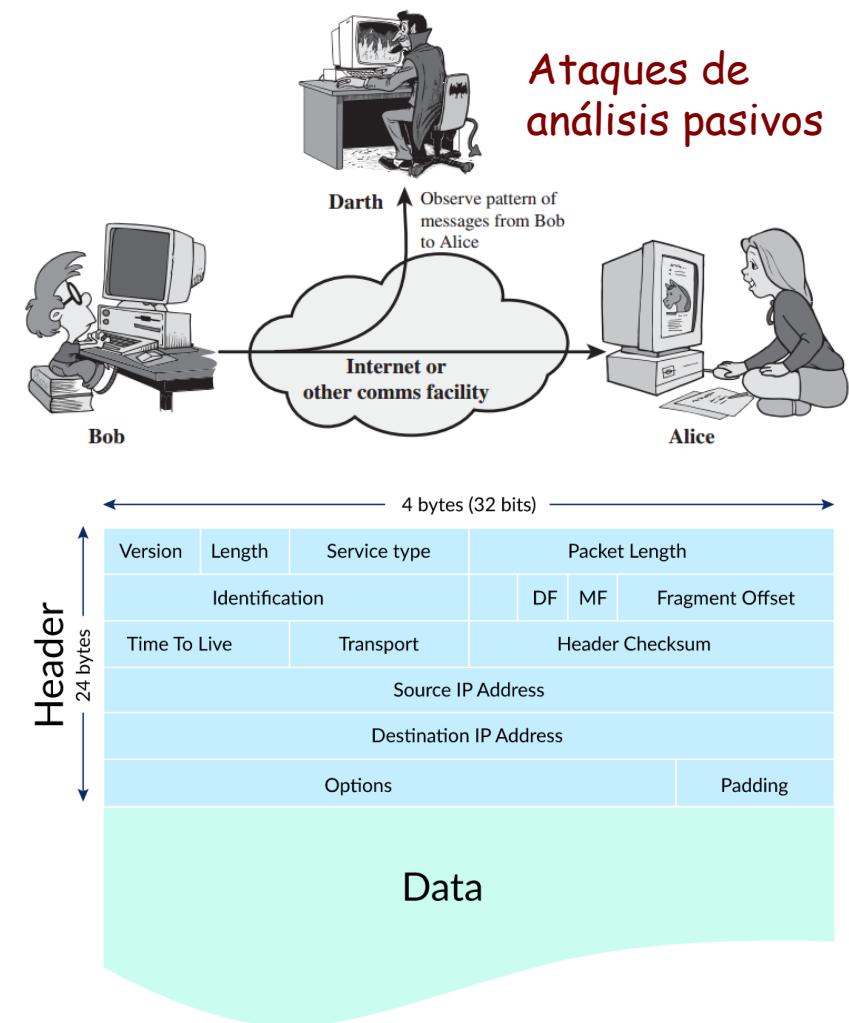
- **Todo lo que se sube a la red es público** y se pierde el control sobre esa información 😞 ... ¡ así que cuidado !
 - Cualquiera puede encontrar los datos y utilizarlos para múltiples fines
 - Despidos, robos, discriminación, incriminación, ...
 - Las **redes sociales** ha facilitado la adquisición de información personal



The screenshot shows a blog post from the website 'experto laboral online'. The post title is 'Me han despedido estando de baja por unas fotos en Facebook ¿Es legal?'. The date of the post is '11 de febrero de 2014'. The author is 'por Experto Laboral Online'. The image in the post shows a person with long blonde hair sitting at a desk, looking at a laptop screen. A camera is also visible on the desk.

Conceptos generales - análisis de tráfico

- El análisis de tráfico permite a observadores externos obtener información a través de las comunicaciones de un usuario
 - Si la comunicación está en claro se puede acceder a la carga útil
 - El cifrado no elimina el problema porque el cifrado sólo protege los datos, y las cabeceras no se pueden cifrar (ej. dirección IP origen y destino), y además se puede estimar aspectos que no se pueden controlar con el cifrado como el tamaño, número de paquetes, frecuencia y tiempos de envío, ...



Conceptos generales - análisis de tráfico

Low Disk Space: Your storage disk is almost full. 3.0G is available.

Guest upload is turned off Log In

tcp1337-netcat-chat.pcapng 1.4 kb · 12 packets · more info

Start typing a Display Filter ✓ Apply Clear Filters ▾

No.	No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	0.000000	10.185.220.101	10.185.220.139	TCP	62	2764 → 1337 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1
2	0.000242	0.000242	10.185.220.139	10.185.220.101	TCP	62	1337 → 2764 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM=1
3	0.000313	0.000313	10.185.220.101	10.185.220.139	TCP	54	2764 → 1337 [ACK] Seq=1 Ack=1 Win=64240 Len=0
4	1.831059	1.831059	10.185.220.101	10.185.220.139	TCP	59	2764 → 1337 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=5
5	1.832836	1.832836	10.185.220.139	10.185.220.101	TCP	60	1337 → 2764 [ACK] Seq=1 Ack=6 Win=65535 Len=0
6	9.291432	9.291432	10.185.220.139	10.185.220.101	TCP	66	1337 → 2764 [PSH, ACK] Seq=1 Ack=6 Win=65535 Len=12
7	9.491448	9.491448	10.185.220.101	10.185.220.139	TCP	54	2764 → 1337 [ACK] Seq=6 Ack=13 Win=64228 Len=0
8	15.405658	15.405658	10.185.220.139	10.185.220.101	TCP	60	1337 → 2764 [FIN, ACK] Seq=13 Ack=6 Win=65535 Len=0
9	15.405740	15.405740	10.185.220.101	10.185.220.139	TCP	54	2764 → 1337 [ACK] Seq=6 Ack=14 Win=64228 Len=0

Internet Protocol Version 4, Src: 10.185.220.139, Dst: 10.185.220.101
Transmission Control Protocol, Src Port: 1337, Dst Port: 2764, Seq: 1, Ack: 6, Len: 0
Source Port: 1337
Destination Port: 2764
[Stream index: 0]
[Conversation completeness: Complete, WITH_DATA (31)]
[TCP Segment Len: 0]
Sequence Number: 1 (relative sequence number)
Sequence Number (raw): 3979006224
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 6 (relative ack number)
Acknowledgment number (raw): 2042183646
0101 = Header Length: 20 bytes (5)
Flags: 0x010 (ACK)
Window: 65535
[Calculated window size: 65535]
[Window size scaling factor: -2 (no window scaling used)]
Checksum: 0x5999 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
[Timestamps]
[SEQ/ACK analysis]

Low Disk Space: Your storage disk is almost full. 3.0G is available.

Guest upload is turned off Log In

tcp1337-netcat-chat.pcapng 1.4 kb · 12 packets · more info

Start typing a Display Filter ✓ Apply Clear Filters ▾

No.	No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	0.000000	10.185.220.101	10.185.220.139	TCP	62	2764 → 1337 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1
2	0.000242	0.000242	10.185.220.139	10.185.220.101	TCP	62	1337 → 2764 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM=1
3	0.000313	0.000313	10.185.220.101	10.185.220.139	TCP	54	2764 → 1337 [ACK] Seq=1 Ack=1 Win=64240 Len=0
4	1.831059	1.831059	10.185.220.101	10.185.220.139	TCP	59	2764 → 1337 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=5
5	1.832836	1.832836	10.185.220.139	10.185.220.101	TCP	60	1337 → 2764 [ACK] Seq=1 Ack=6 Win=65535 Len=0
6	9.291432	9.291432	10.185.220.139	10.185.220.101	TCP	66	1337 → 2764 [PSH, ACK] Seq=1 Ack=6 Win=65535 Len=12
7	9.491448	9.491448	10.185.220.101	10.185.220.139	TCP	54	2764 → 1337 [ACK] Seq=6 Ack=13 Win=64228 Len=0
8	15.405658	15.405658	10.185.220.139	10.185.220.101	TCP	60	1337 → 2764 [FIN, ACK] Seq=13 Ack=6 Win=65535 Len=0
9	15.405740	15.405740	10.185.220.101	10.185.220.139	TCP	54	2764 → 1337 [ACK] Seq=6 Ack=14 Win=64228 Len=0

Internet Protocol Version 4, Src: 10.185.220.139, Dst: 10.185.220.101
Transmission Control Protocol, Src Port: 1337, Dst Port: 2764, Seq: 1, Ack: 6, Len: 0
Source Port: 1337
Destination Port: 2764
[Stream index: 0]
[Conversation completeness: Complete, WITH_DATA (31)]
[TCP Segment Len: 0]
Sequence Number: 1 (relative sequence number)
Sequence Number (raw): 3979006224
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 6 (relative ack number)
Acknowledgment number (raw): 2042183646
0101 = Header Length: 20 bytes (5)
Flags: 0x010 (ACK)
Window: 65535
[Calculated window size: 65535]
[Window size scaling factor: -2 (no window scaling used)]
Checksum: 0x5999 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
[Timestamps]
[SEQ/ACK analysis]

Follow TCP: tcp.stream eq 0 in tcp1337-netcat-chat.pcapng
Show only this stream | Filter out this stream

test
return test

Low Disk Space: Your storage disk is almost full. 3.0G is available.

Guest upload is turned off Log In

tcp1337-netcat-chat.pcapng 1.4 kb · 12 packets · more info

Start typing a Display Filter ✓ Apply Clear Filters ▾

No.	No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	0.000000	10.185.220.101	10.185.220.139	TCP	62	2764 → 1337 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1
2	0.000242	0.000242	10.185.220.139	10.185.220.101	TCP	62	1337 → 2764 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM=1
3	0.000313	0.000313	10.185.220.101	10.185.220.139	TCP	54	2764 → 1337 [ACK] Seq=1 Ack=1 Win=64240 Len=0
4	1.831059	1.831059	10.185.220.101	10.185.220.139	TCP	59	2764 → 1337 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=5
5	1.832836	1.832836	10.185.220.139	10.185.220.101	TCP	60	1337 → 2764 [ACK] Seq=1 Ack=6 Win=65535 Len=0
6	9.291432	9.291432	10.185.220.139	10.185.220.101	TCP	66	1337 → 2764 [PSH, ACK] Seq=1 Ack=6 Win=65535 Len=12
7	9.491448	9.491448	10.185.220.101	10.185.220.139	TCP	54	2764 → 1337 [ACK] Seq=6 Ack=13 Win=64228 Len=0
8	15.405658	15.405658	10.185.220.139	10.185.220.101	TCP	60	1337 → 2764 [FIN, ACK] Seq=13 Ack=6 Win=65535 Len=0
9	15.405740	15.405740	10.185.220.101	10.185.220.139	TCP	54	2764 → 1337 [ACK] Seq=6 Ack=14 Win=64228 Len=0

Internet Protocol Version 4, Src: 10.185.220.139, Dst: 10.185.220.101
Transmission Control Protocol, Src Port: 1337, Dst Port: 2764, Seq: 1, Ack: 6, Len: 0
Source Port: 1337
Destination Port: 2764
[Stream index: 0]
[Conversation completeness: Complete, WITH_DATA (31)]
[TCP Segment Len: 0]
Sequence Number: 1 (relative sequence number)
Sequence Number (raw): 3979006224
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 6 (relative ack number)
Acknowledgment number (raw): 2042183646
0101 = Header Length: 20 bytes (5)
Flags: 0x010 (ACK)
Window: 65535
[Calculated window size: 65535]
[Window size scaling factor: -2 (no window scaling used)]
Checksum: 0x5999 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
[Timestamps]
[SEQ/ACK analysis]

Entire Conversation ASCII Hex Dump Wrap long lines

Ladder Diagram Open in new window Done

Fuente: cloudshark.org

Conceptos generales

- Existen dos **enfoques complementarios** para proteger la privacidad

- **Enfoque legislativo:**

- Creación de leyes que impongan **sanciones** para limitar prácticas abusivas de las empresas
 - En Europa tenemos el Reglamento General de Protección de Datos (**GDPR** - General Data Protection Regulation)

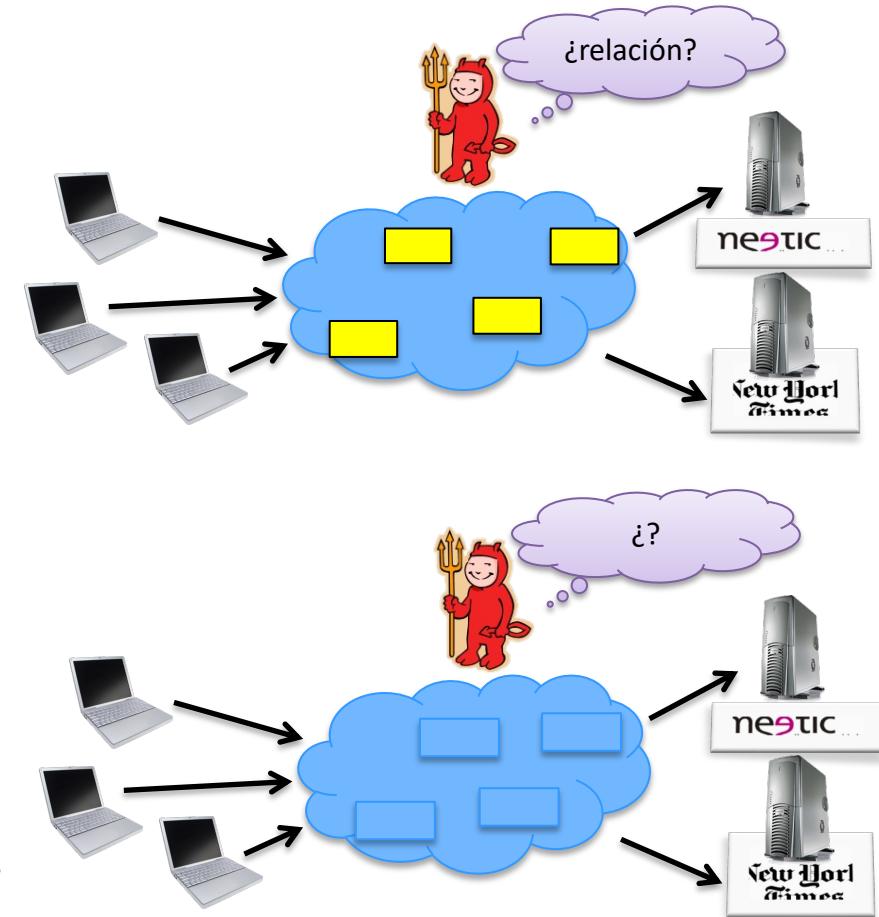


- **Enfoque tecnológico:**

- Existen mecanismos de preservación de la privacidad, principalmente basados en algoritmos criptográficos y probabilísticos, y también en conceptos basados en aleatorización

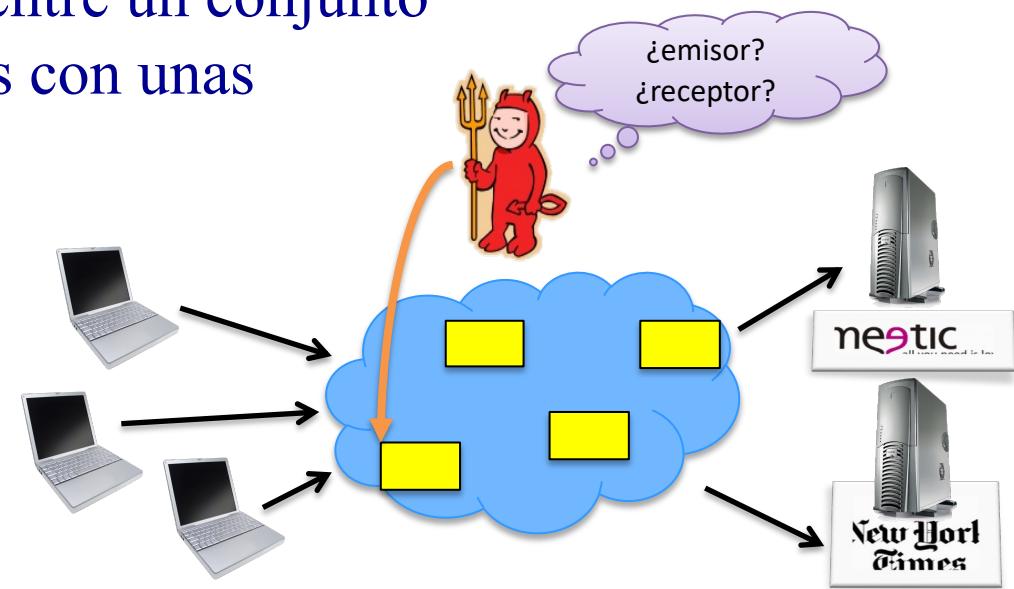
Conceptos generales

- Propiedades de la privacidad:
 - **No-vinculación / no enlazabilidad** (unlinkability): se refiere a la incapacidad de un atacante para relacionar dos mensajes o entidades observadas
 - **No-observabilidad** (unobservability): se refiere a la imposibilidad de distinguir la presencia de mensajes, ni la identidad de la entidades



Conceptos generales

- Relacionado con el concepto de privacidad, está el concepto de **anonimato**:
 - Anonimato se puede definir como “*el estado de no ser identificable entre un grupo de sujetos*”
 - Las técnicas de anonimato tratan de hacer indistinguible (u ocultar) a un individuo entre un conjunto suficiente de identidades con unas características similares



Conceptos generales

- El anonimato depende de la técnica aplicada y se distinguen 4 clases de técnicas:
 - **Pseudónimos:**
 - Se basa de técnicas para ocultar la identidad de un usuario a través de pseudónimos
 - Sin embargo, el uso continuado de pseudónimos pueden ser vinculantes, es decir, que se puede derivar la identidad del usuario, y todas las operaciones previas realizadas
 - **Anonimato rastreable:**
 - Ofrecer anonimato, pero en caso de necesidad se puede revelar la identidad del usuario
 - Ej. el vendedor y el banco: dependiendo de la honestidad de estas partes se puede o no revelar la identidad del usuario principal
 - **Anonimato no rastreable:**
 - Trata de resolver el problema del anonimato pero garantiza que la identidad de los usuarios no se va a revelar
 - **Anonimato no rastreable y no vinculante:**
 - Además de garantizar que la identidad de los usuarios no se puede revelar, la condición de no vinculante asegura que el conjunto de las operaciones realizadas por un usuario no se puedan vincular

Conceptos generales

- Para conseguir una o varias de las propiedades de privacidad mencionadas anteriormente, las soluciones suelen basarse en el uso de algunas de las siguientes técnicas:
 - **Esquemas avanzados de firma digital**
 - **Protocolos criptográficos y de enrutado**
 - Evitan que la dirección de red o el camino que siguen los paquetes puedan identificar a las partes comunicantes
 - **Técnicas de ofuscación**
 - Son mecanismos basados principalmente en la generalización o supresión de información para limitar la precisión de la información que se revela

Privacidad basada en
esquemas avanzados de firma digital

Esquemas avanzados de firma digital

- A partir del concepto básico de firma digital surgen esquemas más avanzados de firma digital con objetivos más ambiciosos:
 - **Firma ciega**
 - El firmante firma mensajes para otros usuarios, pero desconoce el contenido de los mensajes que firma
 - La firma se puede verificar posteriormente – **anonimato rastreable**
 - **Firma de grupo**
 - Cualquier miembro del grupo firma mensajes de forma anónima en nombre del grupo
 - En caso de disputa, una entidad determinada puede revelar la identidad del firmante – **anonimato rastreable**
 - **Firma de anillo**
 - Similar al anterior, pero el anonimato es total; es decir, no es posible saber la identidad del firmante bajo ninguna circunstancia – **anonimato no rastreable ni vinculante**

Firma ciega

- Existen aplicaciones (p. ej. el voto electrónico), donde una de las propiedades a preservar es el anonimato del usuario
- Concretamente, con la firma ciega, lo que se consigue es que el mensaje M generado por *Bob* sea firmado por *Alice* sin que esta conozca el contenido del mensaje
- La firma ciega resultante puede ser verificada públicamente con posterioridad, como una firma digital normal



Firma ciega

- El esquema de firma ciega se puede implementar con diferentes algoritmos de clave pública, entre ellos, RSA
 - Sean e y d , las claves pública y privada de *Alice*
 - Sea n el módulo RSA
 - Y sea r un número aleatorio $\text{mod } n$
(r es el *blinding factor*)



1. Bob: r
2. Bob: $M' = M \cdot r^e \pmod{n}$
3. Bob \rightarrow Alice: M'
4. Alice \rightarrow Bob: $(M')^d \pmod{n} = [M^d \cdot (r^e)^d \pmod{n}] = [M^d \cdot r \pmod{n}]$
5. Bob: $M^d \cdot r \cdot r^{-1} \pmod{n} = [M^d \pmod{n}]$

Firma de grupo

- Al contrario que con los esquemas tradicionales de firma, en los que sólo hay un firmante:
 - los esquemas de firma en grupo permiten que cualquier miembro de un grupo firme en nombre del grupo
 - Ej. un empleado de un banco firma en nombre del banco
- La figura del **administrador del grupo** controla quién pertenece al mismo y también emite la clave de firma del grupo
 - o sea, la clave con la que cualquier miembro firma en nombre del grupo



Firma de grupo

- De lo anterior, se puede deducir que en la utilización de estos esquemas de firma hay tres tipos de **participantes**:
 - Administrador del grupo
 - Miembro del grupo
 - Verificador (receptor del documento firmado)



Firma de grupo

- Un esquema de firma de grupo debe satisfacer las siguientes **propiedades iniciales** para cumplir la condición de “anonimato rastreable”:
 - Sólo los miembros del grupo pueden firmar mensajes de forma correcta (infalsifiable)
 - A excepción del administrador del grupo nadie puede descubrir:
 - qué miembro del grupo ha firmado el mensaje (anonimato)
 - si dos firmas han sido emitidas por el mismo miembro del grupo (no-vinculación)
 - Los miembros no pueden evitar la **apertura de la firma** por parte del administrador, ni firmar por otro

Firma de grupo

- Un esquema de firma de grupo consta de cuatro **procedimientos** distintos:
 - ① **Establecimiento** → es un protocolo entre el administrador y los miembros del grupo. Su ejecución origina:
 - la clave pública Y del grupo
 - las claves privadas individuales X_i de cada miembro del grupo
 - una clave secreta Z de administración para el administrador
 - ② **Firma** → es un algoritmo que: $S = E_{X_i}(M)$
 - tiene como entrada un mensaje M , y la clave privada de uno de los miembros del grupo
 - devuelve la firma S sobre el mensaje M

Firma de grupo

③ Verificación → es un algoritmo que: $E_Y(S) == M$

- recibe como entrada un mensaje M , una firma S , y la clave pública Y del grupo
- devuelve información de M si la firma S es correcta o no

④ Apertura → es un algoritmo que:

- recibe como entrada una firma S y la clave secreta de administración
- devuelve la identidad del miembro del grupo que realizó la firma S además de una prueba de este hecho
- Se asume que todas las comunicaciones entre el administrador y los miembros se llevan a cabo de forma segura

Firma de grupo

- **Ejemplo 1 - ejemplo básico de diseño:**
 - El administrador proporciona a cada miembro del grupo una lista de claves privadas
 - $L_1 = \{X_{1,1}, \dots, X_{1,n}\}$, $L_2 = \{X_{2,1}, \dots, X_{2,n}\}$, ..., $L_n = \{X_{n,1}, \dots, X_{n,n}\}$,
 - Esas listas son disjuntas, tal que: $L_1 \neq L_2$
 - El administrador divulga en un directorio público y en orden aleatorio, la lista completa de claves públicas correspondientes
 - $Y = L_1 = \{Y_{1,1}, \dots, Y_{1,n}\}$, $L_2 = \{Y_{2,1}, \dots, Y_{2,n}\}$, ..., $L_n = \{Y_{n,1}, \dots, Y_{n,n}\}$,
 - Esa lista completa hace las veces de clave pública del grupo
 - Cada miembro puede firmar un mensaje con una de las claves privadas de su lista
 - $L_1 = \{\cancel{X_{1,1}}, \cancel{X_{2,1}}, \cancel{X_{3,1}}, \dots, X_{1,n}\}$
 - Sólo utilizará la clave privada una vez para evitar la vinculación
 - El receptor puede verificar esa firma con la correspondiente clave pública (obtenida del directorio)
 - $X_{1,1} \rightarrow Y_{1,1}$
 - El administrador conoce todas las claves privadas, por lo que, en caso de ser necesario, puede saber fácilmente qué miembro del grupo realizó la firma



Firma de grupo

- **Ejemplo 2:**
 - Sea e el exponente público del grupo (es decir, Y)
 - Sea C_p el conjunto (p_1, p_2, \dots, p_L) de valores primos relativos a e
 - Sea n el módulo compuesto
- $$n = p_1 \cdot p_2 \cdot \dots \cdot p_L$$
- Se generan módulos individuales n_i para cada miembro
 - $$n_i = p_{i1} \cdot p_{i2}$$
- Se calculan los exponentes privados d_i (es decir, X_i) para cada miembro en base al exponente e y a los n_i

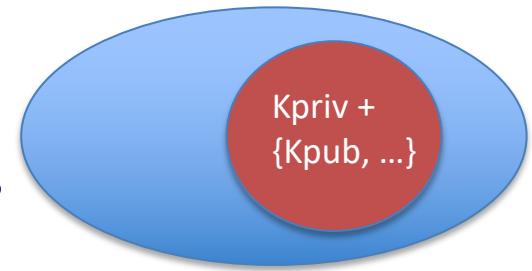
$$d_i = e^{-1} \bmod \theta(n_i)$$

Firma de grupo

- La **firma** por parte del miembro k se realiza así:
 - $S = M^{dk} \text{ mod } n_k$
- La **verificación** por parte de cualquier usuario se realiza así:
 - $M' = S^e \text{ mod } n$
- La **apertura de la firma**, en caso de que sea necesario, la realiza el administrador
 - Se realiza probando uno a uno la firma donde el administrador aplica la clave privada de cada miembro
 - $S_1 = M^{d1} \text{ (mod } n_1)$
 - $S_2 = M^{d2} \text{ (mod } n_2)$
 - ...
 - $S_N = M^{dn} \text{ (mod } n_n)$

Firma de anillo

- Al contrario que las firmas de grupo, las firmas de anillo:
 - no tienen ni administradores de grupo, ni procedimiento de establecimiento, ni procedimiento de revocación del anonimato del firmante, ni coordinación, ni procedimiento para distribuir claves
- Cualquier usuario puede elegir un conjunto de posibles firmantes incluyéndose él/ella mism@, sin la aprobación ni ayuda de esos otros miembros del conjunto
 - computa **la firma** por sí mism@, **usando sólo su clave privada y las claves públicas de los demás**
- Es decir, los otros miembros del conjunto pueden tener desconocimiento total de que sus claves públicas se están usando para ese proceso de firma
 - con el que quizás ni siquiera estén de acuerdo



Privacidad basada en
protocolos criptográficos y de enruteado

Protocolos criptográficos y de enrutado

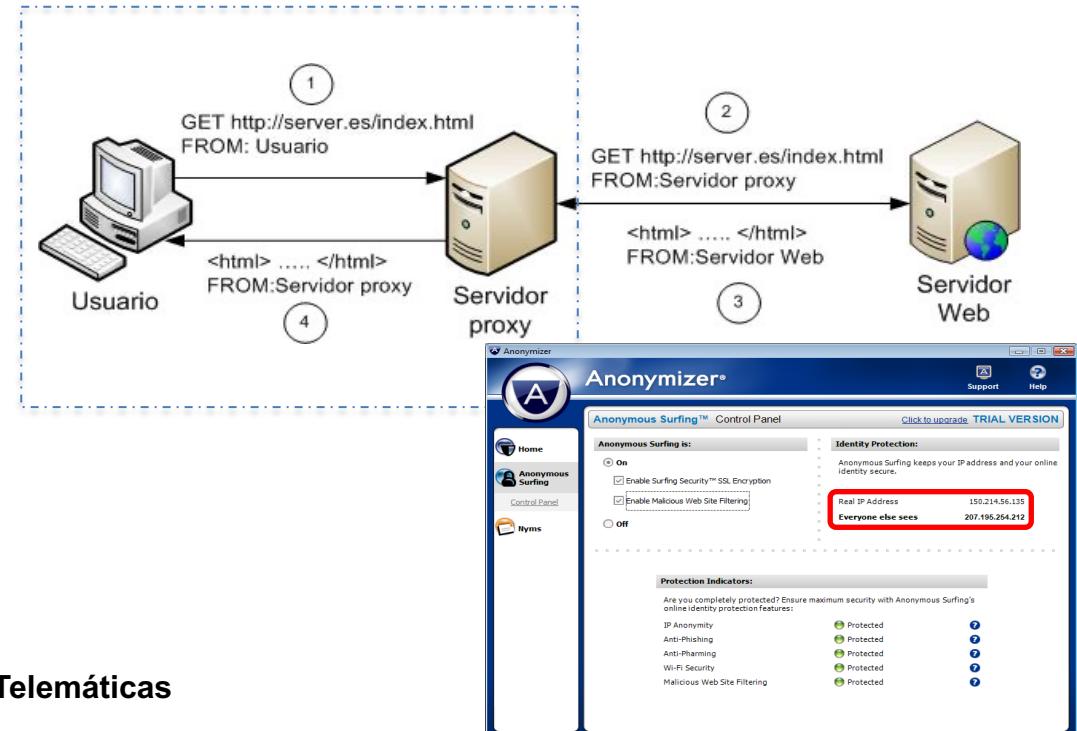
- Tratan de **proteger el tráfico** frente a entidades que se dedican a observar las comunicaciones
- Existen soluciones basadas en:
 - a) uso de proxy
 - b) uso de mezcladores
 - c) conocimiento parcial de la ruta
 - d) creación de gruposademás de algunas **soluciones híbridas**



Protocolos criptográficos y de enrutado

- a) **Basadas en el uso de proxy**: un servidor proxy hace de intermediario en la comunicación, aceptando conexiones de los clientes y reenviándolas
- ocultando así la dirección IP del verdadero origen, y proporcionando **anonimato respecto al destino** porque cree que el origen de la comunicación es el proxy

- Limitaciones:
 - No protegen frente a otros atacantes externos
 - El proxy puede ser honesto pero curioso



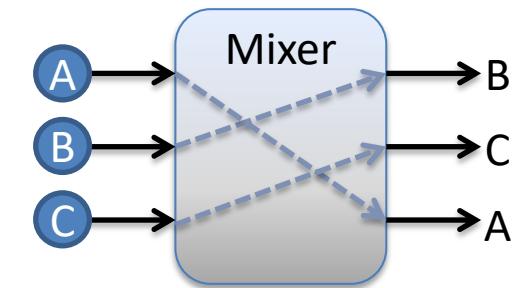
Protocolos criptográficos y de enrutado

b) **Basadas en el uso de mezcladores (mixers)**: es un tipo de proxy capaz de **ocultar la relación** entre los mensajes que recibe y los que envía, y son dispositivos de almacenamiento y envío

- el usuario envía el mensaje a través del mixer
- este lo almacenan durante cierto tiempo, con el fin de que pueda ser mezclado con otros mensajes recibidos, saliendo del mixer en un orden diferente
- el esquema funciona sólo si el número de mensajes almacenados temporalmente por el mixer es lo suficientemente grande

- **Limitaciones:**

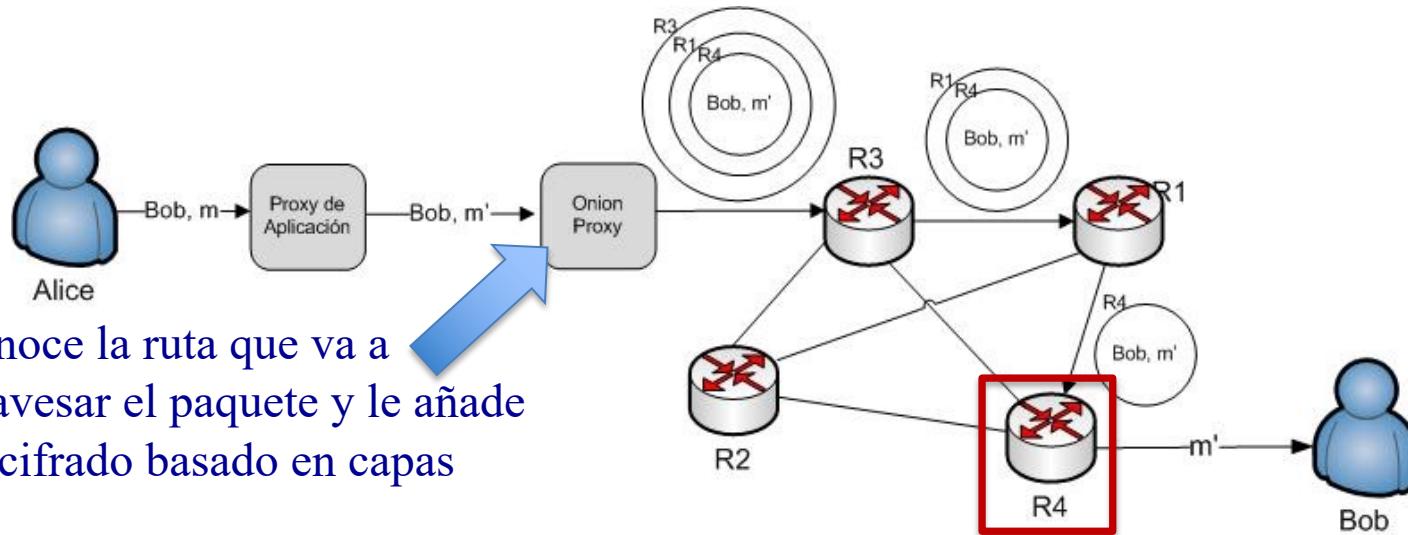
- Los retrasos introducidos pueden llegar a ser grandes
- El mezclador puede ser deshonesto u honesto pero curioso



Protocolos criptográficos y de enrutado

c) Basadas en el conocimiento parcial de la ruta

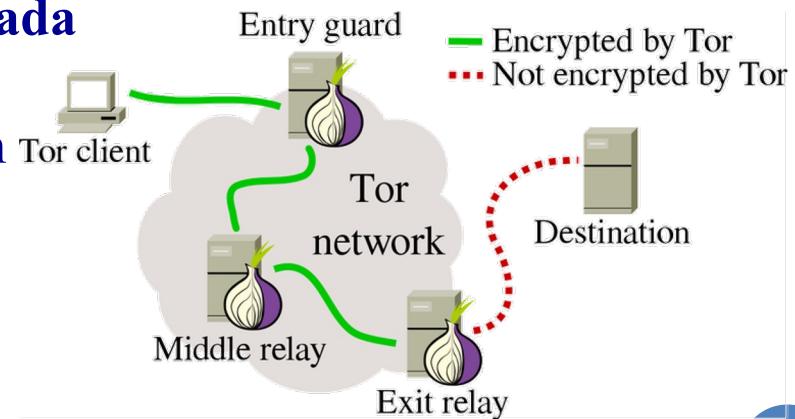
- **Onion Routing:** el onion proxy crea un paquete cifrado en capas, que se irán “pelando” a medida que atraviese el camino de onion routers



- Ataques en un extremo ☹
 - El atacante necesita controlar un extremo de la red

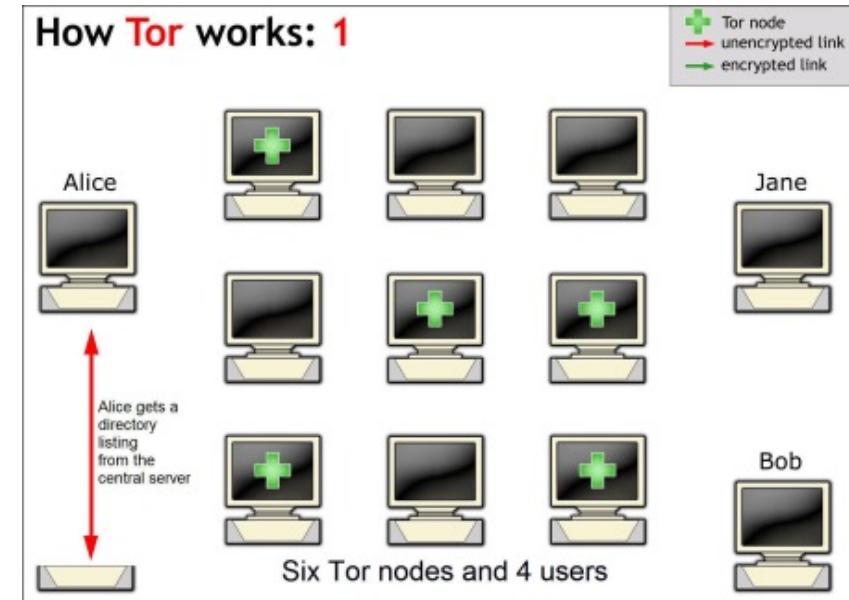
Protocolos criptográficos y de enrutado

- **TOR**: es la segunda generación de **Onion Routing**
 - evita algunas deficiencias del diseño original, añadiendo control de congestión, comprobación de integridad, etc.
 - permite navegar a través de una **ruta privada** de la red TOR, creada **de manera aleatoria** entre los distintos repetidores (onion routers) de la red
 - se negocian **nuevos caminos** a menudo y, tras su uso, las claves se borran para proporcionar *forward secrecy*
 - se aplican **guardianes de entrada** (**el onion proxy**) para reducir posibles ataques de correlación
 - **servidores ocultos** que sólo se pueden acceder desde dentro de la red Tor



Protocolos criptográficos y de enrutado

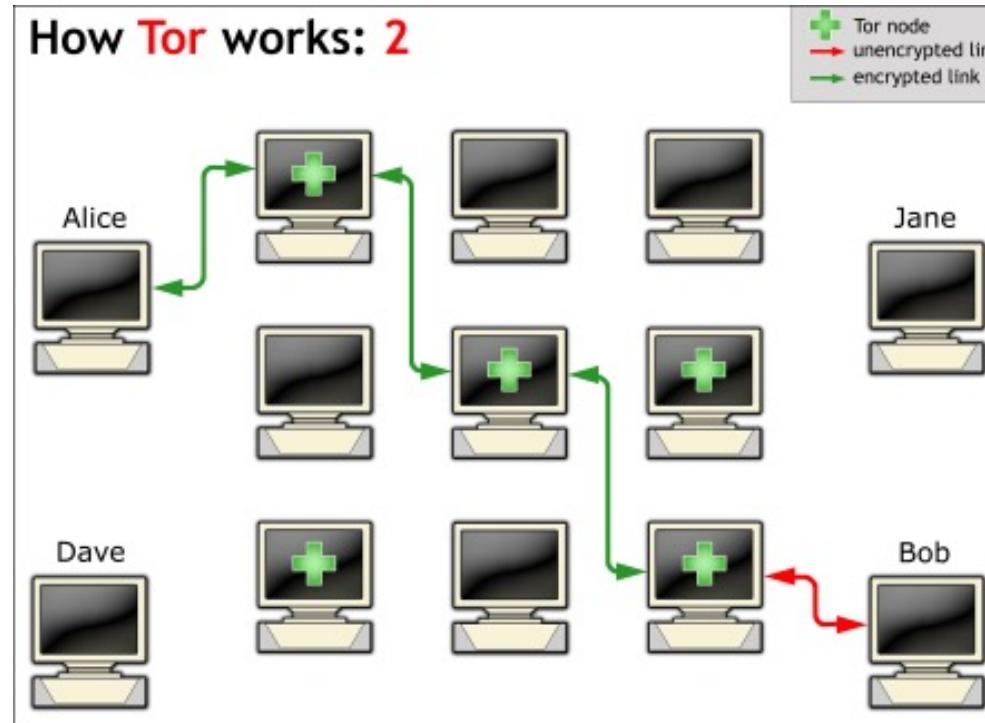
- Funcionamiento:
 - Enrutamiento → los paquetes se envían a través de varios *routers onions*, los cuales son elegidos de forma “aleatoria” y previamente por un “servidor central”
 - Todos los nodos Tor son elegidos al azar y ningún nodo puede ser utilizado dos veces



<https://geekytheory.com/que-es-y-como-funciona-la-red-tor>

Protocolos criptográficos y de enrutado

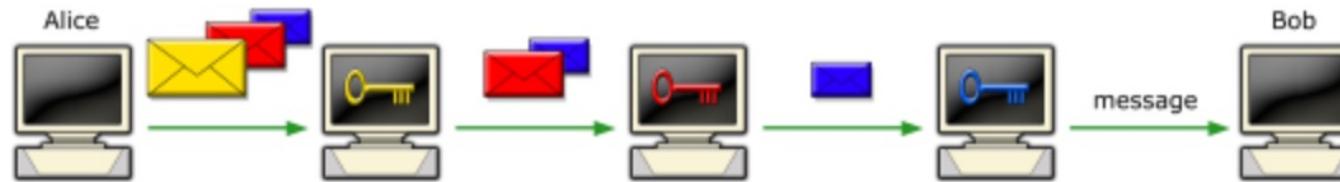
- Se conecta a un nodo aleatorio a través de una conexión cifrada
 - Una vez que el camino ya es conocido desde el origen, cada conexión y salto en la red deberá ser cifrada, excepto con el penúltimo nodo de la comunicación, el cual hará una conexión NO cifrada



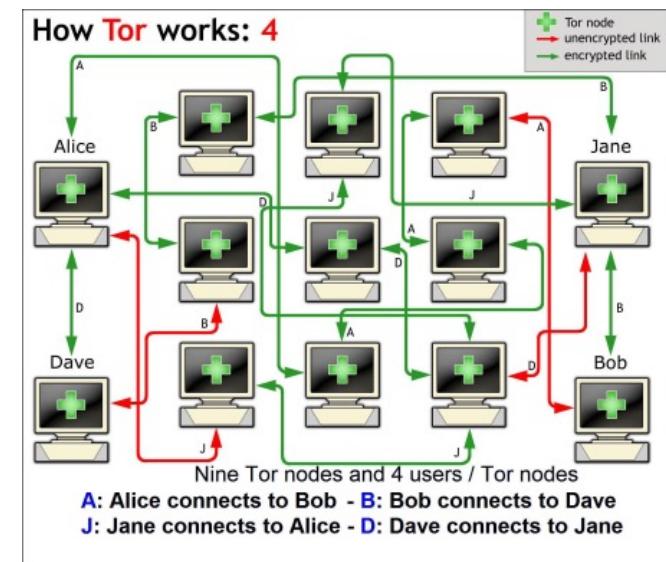
<https://geekytheory.com/que-es-y-como-funciona-la-red-tor>

Protocolos criptográficos y de enrutado

- El enrutado y el cifrado es “asimétrico” (*Onion Routing*):
 - Ej.: Alice lo primero que hace es cifrar el mensaje con la clave pública del último router onion de la lista, para que éste último lo pueda descifrar; y así con todos los demás



- Para evitar el **análisis de tráfico pasivo**, cada 10 minutos se cambian los nodos de la conexión Tor, escogiendo nuevos nodos



Protocolos criptográficos y de enrutado

– Limitaciones

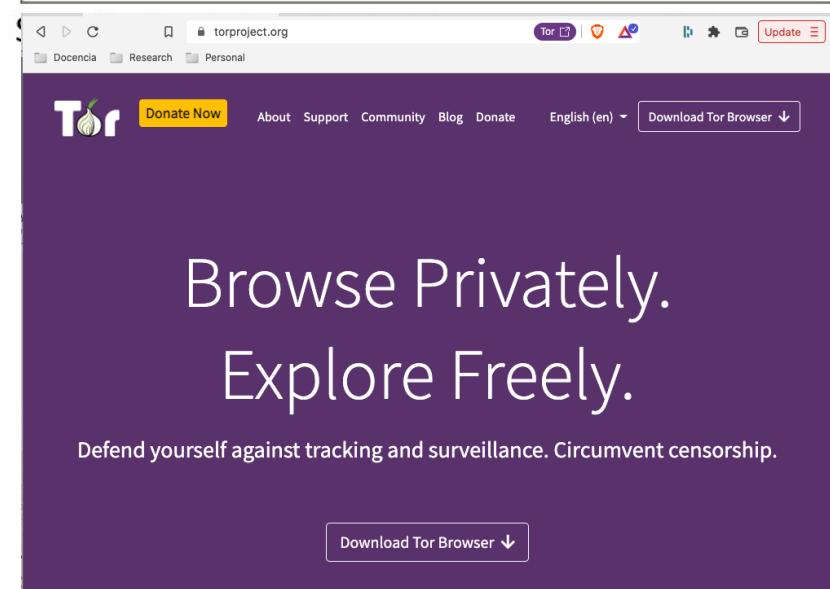
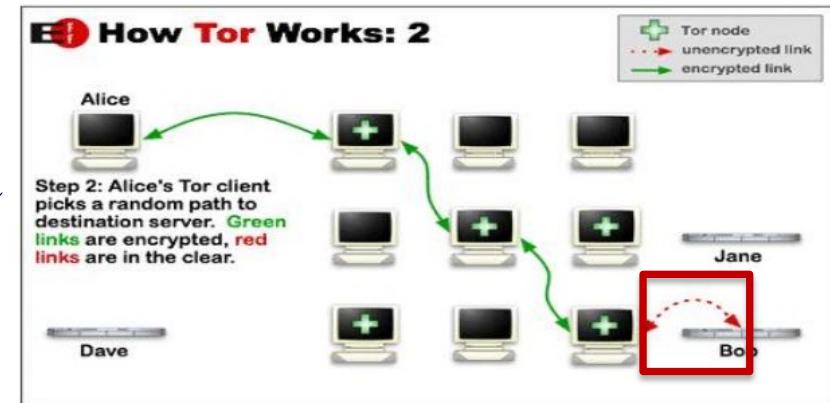
- Lento por su arquitectura:
 - conectividad basada en routers y uso de la criptografía de clave pública
- Garantiza el anonimato, pero no garantiza la “privacidad de los datos”

– Existen varias formas de acceder a la red Tor:

- Utilizar el navegador compatible con Tor
- También para móviles
- Utilizar una extensión del navegador

<https://geekytheory.com/que-es-y-como-funciona-la-red-tor>

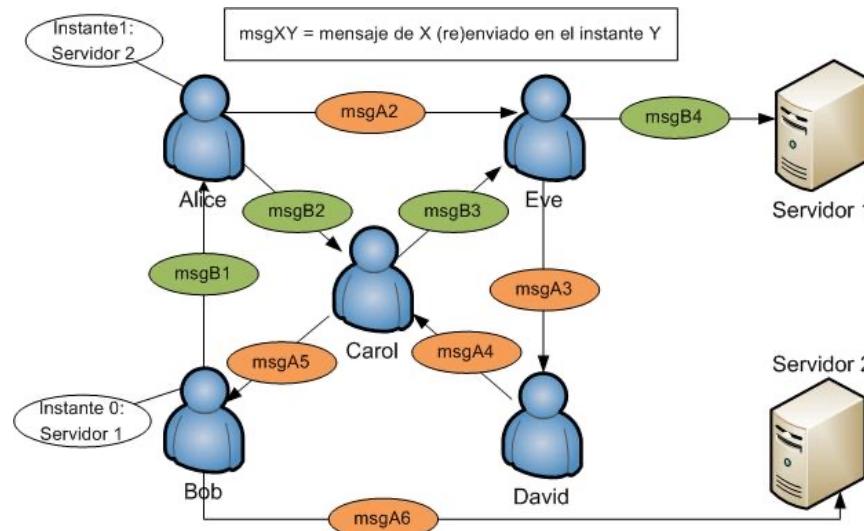
Tor in brief – 2/3



Protocolos criptográficos y de enrutado

d) Basadas en la creación de grupos

- **Crowds:** los emisores se agrupan creando una “multitud” en la que todos enrutan de manera aleatoria los paquetes recibidos de sus compañeros
 - cada nodo sólo conoce el destino y el nodo del que recibe el paquete
 - los nodos comparten claves con sus vecinos para cifrar los mensajes
 - el camino seguido por el mensaje es almacenado temporalmente en cada nodo en el que se transita para saber enrutar de vuelta (con la respuesta)



Protocolos criptográficos y de enrutado

- **Hordes**: es un esquema muy similar al de Crowds con algunas diferencias, principalmente en la forma de enrutar las respuestas de los mensajes
 - en este caso se hace un broadcast de la respuesta desde el router a todos los miembros del grupo donde se encuentra el originador del mensaje
 - esta diferencia supone una mejora significativa en términos de rendimiento (en término de tiempo) frente a Crowds
 - sin embargo, es menos robusto que Crowds ya que el mensaje de broadcast da ciertos indicios sobre la localización del usuario, y sobrecarga al sistema

Protocolos criptográficos y de enrutado

	Arquitectura	Latencia
Proxy		Baja
Mezcladores		Alta
Red de mezcladores	Centralizada	Muy alta
Enrutado de cebolla		Media-baja
Tor		
Crowds	Distribuida	Media-baja

SEGURIDAD DE LA INFORMACIÓN

TEMA 5 – PARTE A

SEGURIDAD EN REDES TCP/IP

Índice del tema

- Seguridad en la Capa de Transporte
- Seguridad en la Capa de Internet
- Firewalls en Redes
- Seguridad en la Capa de Acceso a Red: el caso de las Redes Inalámbricas

Introducción



- La expansión de la WWW en los 90 y su utilidad para realizar transacciones a nivel web, plantean nuevos riesgos de seguridad
 - en lado del cliente Web,
 - en el lado del servidor Web,
 - **la propia información entre el cliente y el servidor** ←

Amenazas		Consecuencias
Integridad	Modificación de los datos de usuario, troyano en el navegador, modificación del mensaje en tránsito, etc.	Pérdida de la información, afecta al dispositivo, desencadenamiento de otros ataques
Confidencialidad	Escuchas en el canal de comunicaciones, robo de información, robo de configuración de red, etc.	Pérdida de información y privacidad
Denegación de servicio	Interrumpir procesos de usuario, interrupciones en el lado del cliente o del servidor, etc.	Interrupción, inaccesibilidad
Autenticación	Suplantación de identidad, falsificación de datos	Creer en datos falsos o en entidades ilegítimas

Introducción



- El IETF formó a mediados de los 90 un Grupo de Trabajo denominado *Web Transaction Security (WTS)*
 - su objetivo: desarrollar los requisitos y las especificaciones para la provisión de servicios de seguridad en transacciones Web

Web Transaction Security (wts) (concluded WG)

[Documents](#) | [Charter](#) | [History](#) | [List Archive »](#) | [Tools WG Page »](#)

Description of Working Group

The goal of the Web Transaction Security Working Group is to develop requirements and a specification for the provision of security services to Web transaction, e.g., transactions using HyperText Transport Protocol (HTTP). This work will proceed in parallel to and independently of the development of non-security features in the HTTP Working Group. The working group will prepare two documents for submission as Internet Drafts; an HTTP Security Requirements Specification, and an HTTP Security Protocol Specification. The latter will be submitted as a Standards Track RFC.

Goals and Milestones

Jul 1995	HTTP Security Requirements finalized at the Stockholm IETF. Submit HTTP Security Specification proposal(s) as Internet-Drafts.
Dec 1995	HTTP Security Specification finalized at the Dallas IETF, submit to IESG for consideration as a Proposed Standard.
Done	HTTP Security Requirements submitted as Internet-Draft.

Note: The data for concluded WGs is occasionally incorrect.

Group

Name: Web Transaction Security
Acronym: wts
Area: Security Area (sec)
State: Concluded
Charter: [charter-ielf-wts-01](#) (Approved)

Personnel

Chair: [Charlie Kaufman <charlie_kaufman@notesdev.ibm.com>](#)
Area Director: ?

Mailing List

Address: www-security@nsmx.rutgers.edu
To Subscribe: www-security-request@nsmx.rutgers.edu
Archive: <http://www-ns.rutgers.edu/www-security>

Introducción



- Este grupo se centró en el desarrollo de una solución en la capa de aplicación, y
 - diseñó el protocolo **SHTTP - Secure HyperText Transfer Protocol**, especificado en los documentos RFC que se observan abajo

Web Transaction Security (wts) (concluded WG)

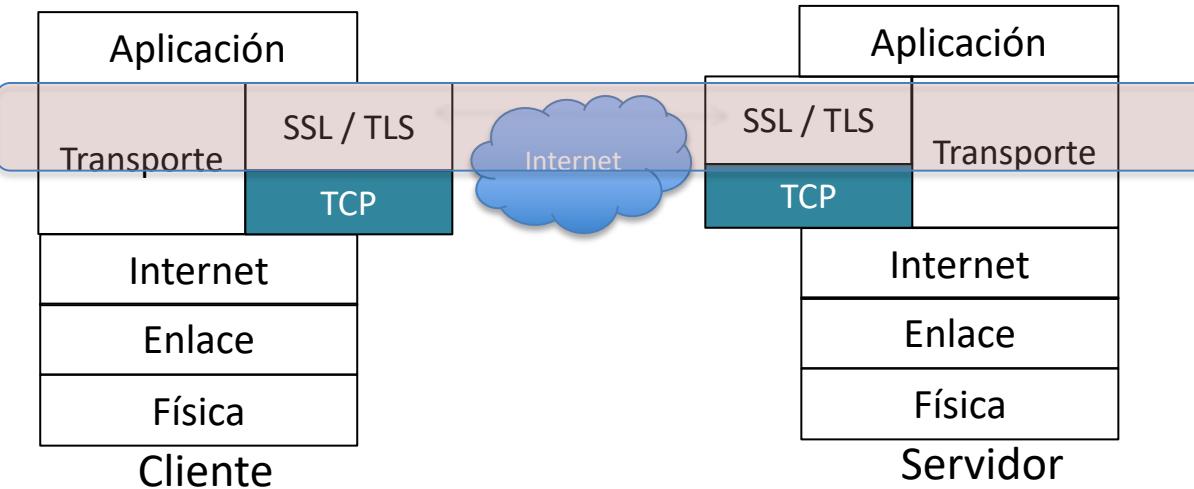
[Documents](#) | [Charter](#) | [History](#) | [List Archive »](#) | [Tools WG Page »](#)

Document	Title	Date	Status	IPR	Area Director
RFCs					
RFC 2084 (draft-ietf-wts-requirements)	Considerations for Web Transaction Security	1997-01	RFC 2084 (Informational)		
RFC 2659 (draft-ietf-wts-shtml)	Security Extensions For HTML	1999-08	RFC 2659 (Experimental)		
RFC 2660 (draft-ietf-wts-shttp)	The Secure HyperText Transfer Protocol	1999-08	RFC 2660 (Experimental)		
Related Documents					

Introducción



- Por otro lado, en las mismas fechas, los desarrolladores de Netscape abordaron el problema, pero desde la capa de transporte
 - como una solución intermedia (ni en la capa alta ni en las bajas)
- El resultado fue el protocolo **SSL - Secure Sockets Layer**, una subcapa entre la de aplicación y la de transporte
 - más concretamente, SSL se sitúa por encima de TCP dado que este es orientado a la conexión y proporciona fiabilidad



- el objetivo del protocolo SSL es, por tanto, crear **conexiones seguras y fiables** y transmitir datos a través de esas conexiones
- la última versión producida fue la v3.0



EVOLUCIÓN DE SSL/TLS

Introducción (un poco de historia ☺ ...)



Cuándo	Quién	Qué
A mediados de 1994	Netscape	SSL v1.0
A finales de 1994	Netscape	SSL v2.0
1995	Netscape	SSL v2.0 (actualizada)
1996	Netscape	SSL v3.0 (obsoleta)



Netscape

- **SSL es una iniciativa de Netscape para abordar los problemas de seguridad del Internet**
- Sustituye por completo el protocolo SHTTP, como un intento de proteger los canales de comunicación vía el Internet

Introducción (un poco de historia ☺ ...)



Cuándo	Quién	Qué
A mediados de 1994	Netscape	SSL v1.0
A finales de 1994	Netscape	SSL v2.0
1995	Netscape	SSL v2.0
1996	Netscape	SSL v3.0 (obsoleta)
1999	IETF	TLS v1.0 (SSL v3.1) – RFC: 2246

MISIÓN

This document specifies Version 1.0 of the Transport Layer Security (TLS) protocol. The TLS protocol provides communications privacy over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery.

- **TLS (Transport Layer Security) es una iniciativa de IETF para estandarizar SSL**
- Se definió por primera vez (TLS 1.0) en 1999, en el **RFC 2246**. Como se menciona en ese RFC:

Network Working Group
Request for Comments: 2246 ← RFC
Category: Standards Track

T. Dierks
Ceritcom
C. Allen
Ceritcom
January 1999

The TLS Protocol
Version 1.0 ← año

Status of this Memo ← versión

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice
Copyright (C) The Internet Society (1999). All Rights Reserved.

Abstract
This document specifies Version 1.0 of the Transport Layer Security (TLS) protocol. The TLS protocol provides communications privacy over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery. ← descripción

Table of Contents

1. Introduction	3
2. Goals	4
3. Goals of this document	5
4. Presentation language	5
4.1. Basic block size	6
4.2. Miscellaneous	6
4.3. Vectors	6
4.4. Numbers	7
4.5. Enumerateds	7
4.6. Constructed types	8
4.6.1. Variants	9
4.7. Cryptographic attributes	10
4.8. Constants	11
5. HMAC and the pseudorandom function	11
6. The TLS Record Protocol	13
6.1. Connection states	14

Introducción (un poco de historia ☺ ...)



Cuándo	Quién	Qué
A mediados de 1994	Netscape	SSL v1.0
A finales de 1994	Netscape	SSL v2.0
1995	Netscape	SSL v2.0
1996	Netscape	SSL v3.0 (obsoleta)
1999	IETF	TLS v1.0 (SSL v3.1) – RFC: 2246

- **TLS (Transport Layer Security) es una iniciativa de IETF para estandarizar SSL**
- Se definió por primera vez (TLS 1.0) en 1999, en el **RFC 2246**. Como se menciona en ese RFC:

“the differences between this protocol and SSL 3.0 are not dramatic, but they are significant to preclude interoperability between TLS 1.0 and SSL 3.0”

- Sin embargo, para entender estas diferencias es necesario primero ver cómo funciona el protocolo SSL (el cual es equivalente a TLS)

Introducción (un poco de historia ☺ ...)



Cuándo	Quién	Qué
A mediados de 1994	Netscape	SSL v1.0
A finales de 1994	Netscape	SSL v2.0
1995	Netscape	SSL v2.0
1996	Netscape	SSL v3.0 (obsoleta)
1999	IETF	TLS v1.0 (SSL v3.1) – RFC: 2246
2006	IETF	TLS v1.1 (SSL v3.2) – RFC: 4346
2008	IETF	TLS v1.2 (SSL v3.3) – RFC: 5246
2014	IETF	TLS v1.3 (draft 1) – (SSL v3.4) – RFC: 8446
2018	IETF	TLSv 1.3

[RFC Home] [TEXT|PDF|HTML] [Tracker] [IPR] [Errata] [Info page]

Obsoleted by: 5246, 6066
Updated by: 5746
Network Working Group
Request for Comments: 4366
Obsoletes: 3546
Updates: 4346
Category: Standards Track

PROPOSED STANDARD
Errata Exist
S. Blake-Wilson
M. Nystrom
RSA Security
D. Hopwood
Independent Consultant
J. Mukkavilli
Transactionware
T. Wright
Vodafone
April 2006

Transport Layer Security (TLS) Extensions

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2006).

Abstract

This document describes extensions that may be used to add functionality to Transport Layer Security (TLS). It provides both generic extension mechanisms for the TLS handshake client and server hellos, and specific extensions using these generic mechanisms.

The extensions may be used by TLS clients and servers. The extensions are backwards compatible; communication is possible between TLS clients that support the extensions and TLS servers that do not support the extensions, and vice versa.

[RFC Home] [TEXT|PDF|HTML] [Tracker] [IPR] [Errata] [Info page]

Obsoleted by: 8446
Updated by: 5746, 5878, 6176, 7465, 7507, 7568, 7627, 7685, 7985, 7919, 8447, 9155
Network Working Group
Request for Comments: 5246
Obsoletes: 3268, 4346, 4366
Updates: 4492
Category: Standards Track

PROPOSED STANDARD
Errata Exist
T. Dierks
Independent
E. Rescorla
RTFM, Inc.
August 2008

The Transport Layer Security (TLS) Protocol Version 1.2

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Abstract

This document specifies Version 1.2 of the Transport Layer Security (TLS) protocol. The TLS protocol provides communications security over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery.

Fuente: <https://www.rfc-editor.org/rfc/rfc4366>

Fuente: <https://www.rfc-editor.org/rfc/rfc5246>

[RFC Home] [TEXT|PDF|HTML] [Tracker] [IPR] [Errata] [Info page]

PROPOSED STANDARD
Errata Exist
E. Rescorla
Mozilla
August 2018

Internet Engineering Task Force (IETF)
Request for Comments: 8446
Obsoletes: 5077, 5246, 6961
Updates: 5705, 6066
Category: Standards Track
ISSN: 2070-1721

The Transport Layer Security (TLS) Protocol Version 1.3

Abstract

This document specifies version 1.3 of the Transport Layer Security (TLS) protocol. TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

This document updates RFCs 5075 and 6066, and obsoletes RFCs 5077, 5246, and 6961. This document also specifies new requirements for TLS 1.2 implementations.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in [Section 2 of RFC 7841](#).

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at <https://www.rfc-editor.org/info/rfc8446>.

SSL - Secure Sockets Layer



- El protocolo SSL es un **protocolo cliente/servidor** que proporciona un conjunto de servicios de seguridad para garantizar prevención contra:
 - “ataques de escucha (*eavesdropping*)”,
 - “ataques de manipulación (*tampering*)” y
 - “ataques de falsificación (*forgery*)”

This document specifies Version 1.0 of the Transport Layer Security (TLS) protocol. The TLS protocol provides communications privacy over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery.

Fuente: <https://www.ietf.org/rfc/rfc2246.txt>

- Esto significa que para abordar:
 - Ataques de escuchas es necesario: **confidencialidad**
 - Ataques de manipulación: **integridad**
 - Ataques de falsificación (de ID): **autenticación**

SSL - Secure Sockets Layer



- Efectivamente el **protocolo SSL** tenía como misión garantizar:
 - **Autenticación**
 - tanto de las entidades origen y destino de los datos
 - usando, por ejemplo, certificados (aunque opcional) y MAC
 - **Confidencialidad** de la conexión
 - **Integridad** de la conexión
- Para ello, el **protocolo SSL** tenía también como misión garantizar una conexión segura entre un cliente y un servidor,
 - por lo que consiste básicamente en un pase de mensajes previos antes de iniciar la conexión



SSL - Secure Sockets Layer



- Más concretamente, SSL (v3.0) empleaba:
 - **criptografía de clave pública**
 - autenticación de las entidades
 - establecimiento de clave
 - **criptografía simétrica**
 - autenticación de los datos (mensajes)
 - cifrado de los mensajes
- Para el intercambio de clave SSLv3.0 aplicaba:
 - RSA
 - Diffie-Hellmann
 - Fortezza KEA (sólo hasta SSL v3.0)
- A pesar de la utilización de criptografía de clave pública,
SSL no proporciona el servicio de no-repudio
 - ni no-repudio de origen ni no-repudio de entrega

**Criptografía
Híbrida ☺**

SSL - Secure Sockets Layer

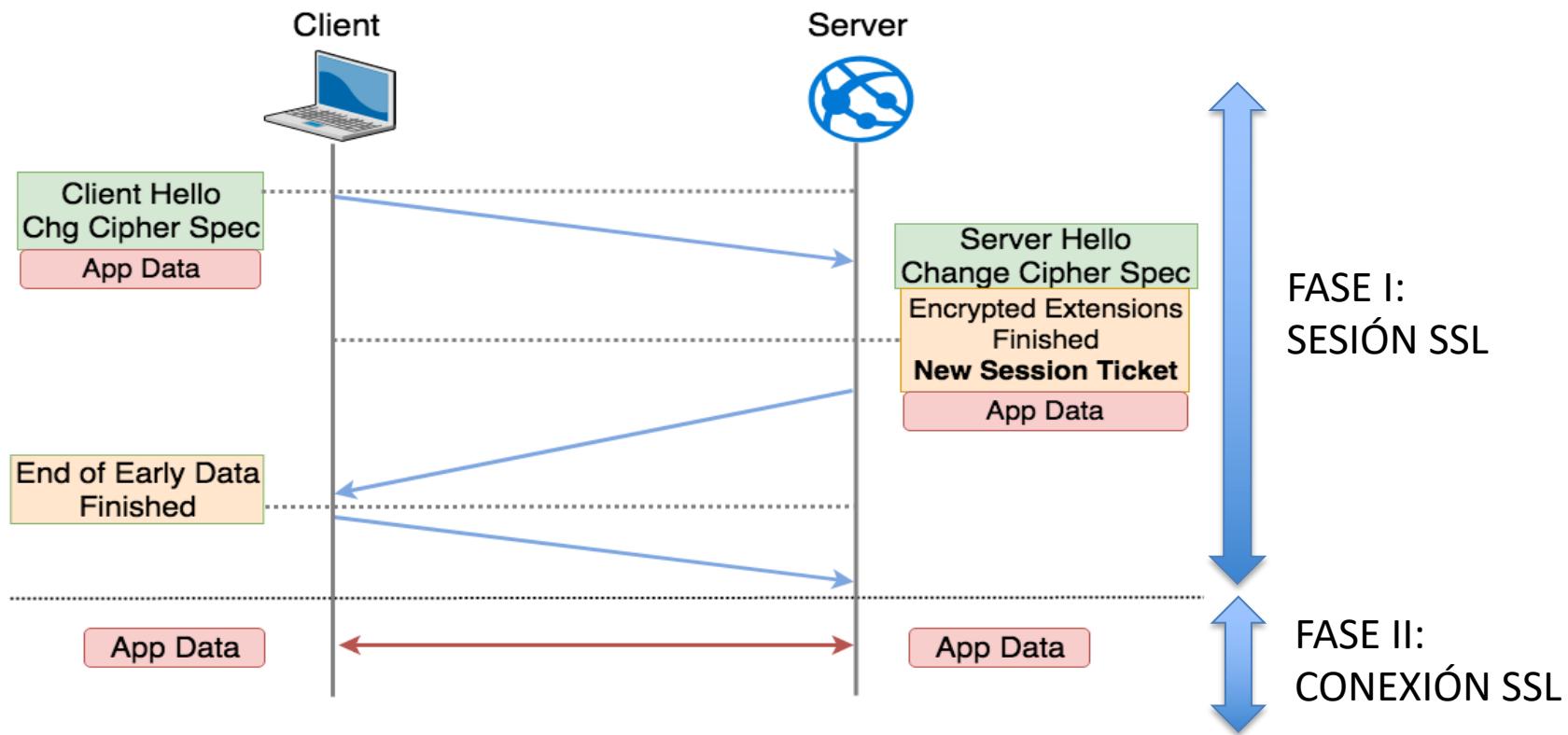


- Una ventaja del protocolo SSL es que es independiente del protocolo de la capa de aplicación
 - es decir, cualquier protocolo de aplicación basado en TCP se puede beneficiar de SSL (éste le dota de los servicios de seguridad mencionados)

Protocolo	Descripción	Puerto
https	HTTP sobre SSL/TLS	443
ldaps	LDAP sobre SSL/TLS	636
ftps-data	FTP Data sobre SSL/TLS	989
ftps	FTP Control sobre SSL/TLS	990
telnets	Telnet sobre SSL/TLS	992
Impas	IMAP4 sobre SSL/TLS	993
Pop3s	POP3 sobre SSL/TLS	995
...



- El protocolo SSL emplea, entre otros, estos dos conceptos:
 - **Sesión SSL**: asociación entre el cliente y el servidor en la que se negocian los parámetros de seguridad para todas las conexiones de esa sesión
 - **Conexión SSL**: realización de la transmisión de datos entre el cliente y el servidor, protegida criptográficamente según lo negociado en la sesión



SSL - Secure Sockets Layer



- El ámbito de la funcionalidad de SSL es, por tanto, doble:
 1. **Establecer una conexión segura** entre los puntos que se comunican, y garantizar conexión:
 - Autenticada
 - Confidencial entre un cliente y un servidor (“*conexión segura punto a punto*”)
 2. **Utilizar esa conexión para transmitir** de forma segura los datos del nivel de aplicación entre el emisor y el receptor



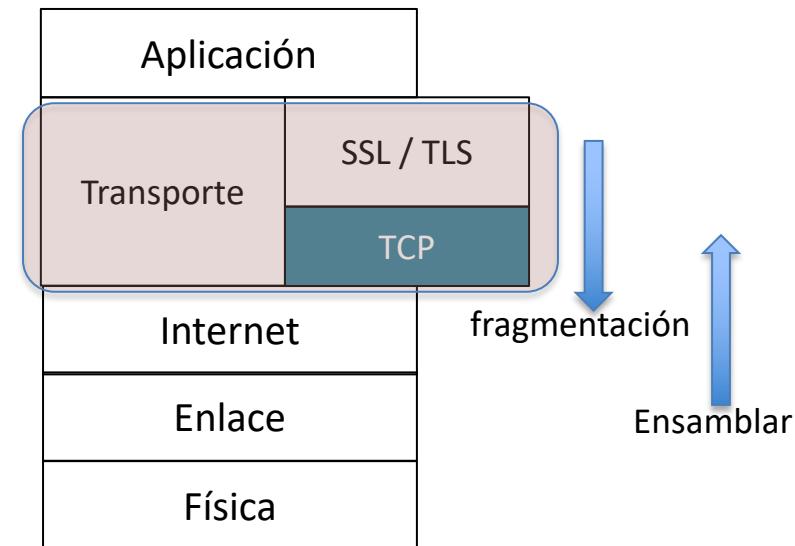
Negociación de
parámetros de seguridad

Comunicación segura

SSL - Secure Sockets Layer



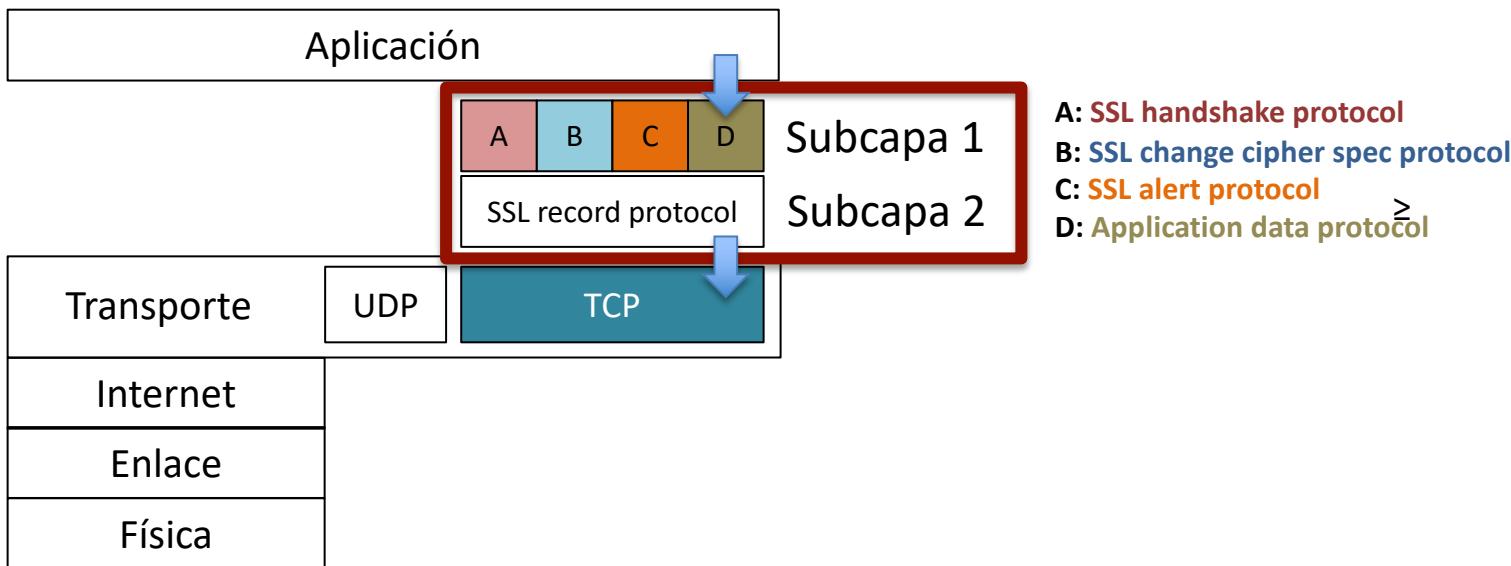
- Para una conexión segura es necesario que SSL considere las misiones propias de la capa de transporte, y especialmente aquellas asignadas a TCP
 - Fragmentar los paquetes de forma que siga una transmisión ordenada entre el origen y el destino
 - Ensamblar los paquetes de forma que se pueda construir los mensaje de manera ordenada
- Esta transmisión requiere a su vez de:
 - Dividir los datos en fragmentos más manejables
 - Procesarlos de forma individual
 - cada fragmento tratado se denomina **SSL record**



SSL - Secure Sockets Layer



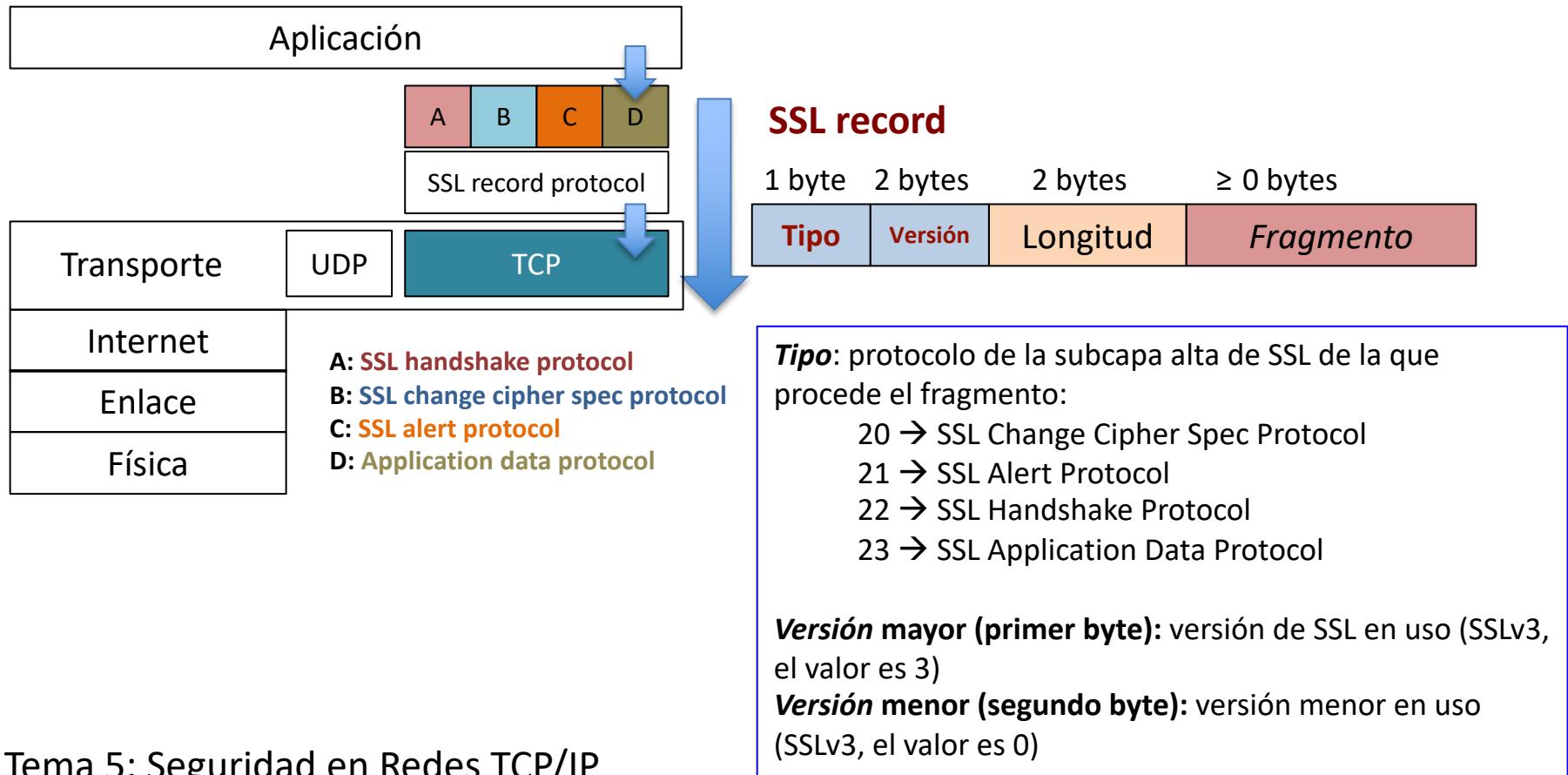
- Para llevar a cabo esa doble funcionalidad, SSL consta de dos subcapas y varios subprotocolos, como se observa en la siguiente figura:



SSL - Secure Sockets Layer



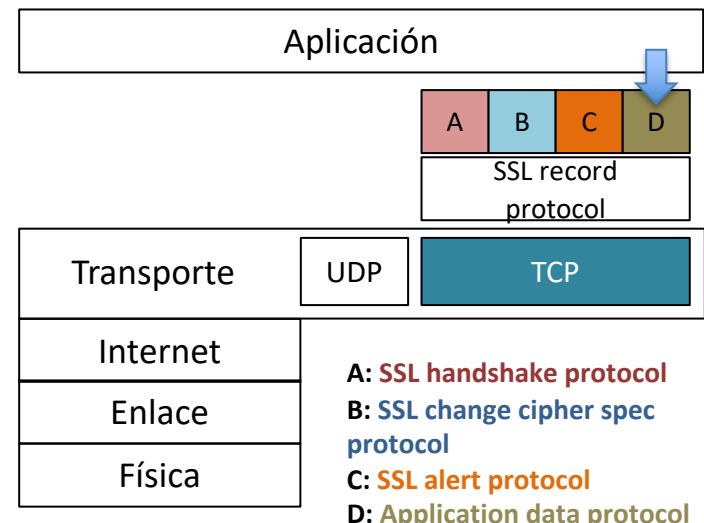
- Para llevar a cabo esa doble funcionalidad, SSL consta de dos subcapas y varios subprotocolos, como se observa en la siguiente figura:



SSL - Secure Sockets Layer



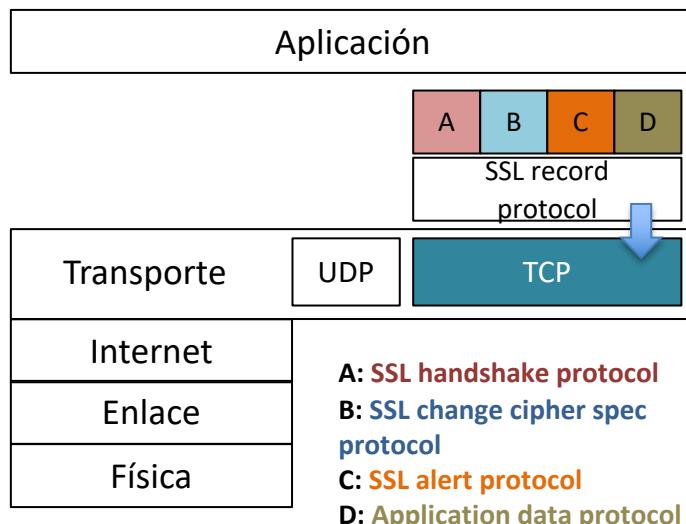
- La subcapa alta contiene:
 - **(22) SSL Handshake Protocol:** permite que los puntos de comunicación:
 - se autentiquen mutuamente, y que además,
 - se negocien un **cipher suite** y
 - (opcionalmente) un método de compresión
 - **(20) SSL Change Cipher Spec Protocol:** permite a los puntos de comunicación activar el cipher suite
 - **(21) SSL Alert Protocol:** permite a los puntos de comunicación indicar posibles problemas potenciales e intercambiar los correspondientes mensajes de alerta
 - **(23) SSL Application Data Protocol:** es el propio protocolo de la capa de aplicación (ej: HTTP) y alimenta al SSL Record Protocol



SSL - Secure Sockets Layer



- La subcapa baja contiene:
 - **SSL Record Protocol:** fragmenta los datos de la capa de aplicación y los procesa de forma individual



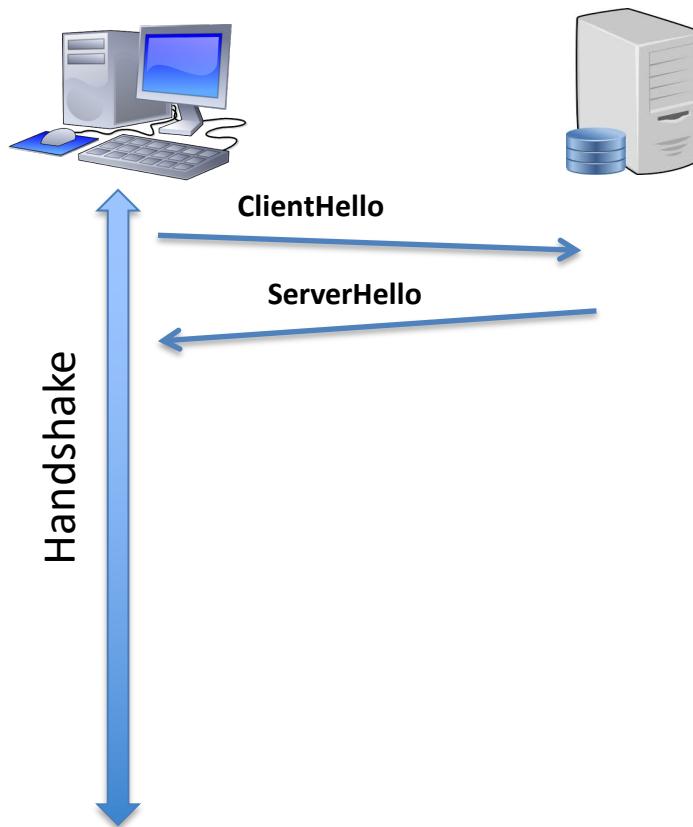
SSL record

1 byte	2 bytes	2 bytes	≥ 0 bytes
Tipo	Versión	Longitud	<i>Fragmento</i>

SUBCAPA 1

(SESIÓN ENTRE LOS PUNTOS)

SSL - Secure Sockets Layer

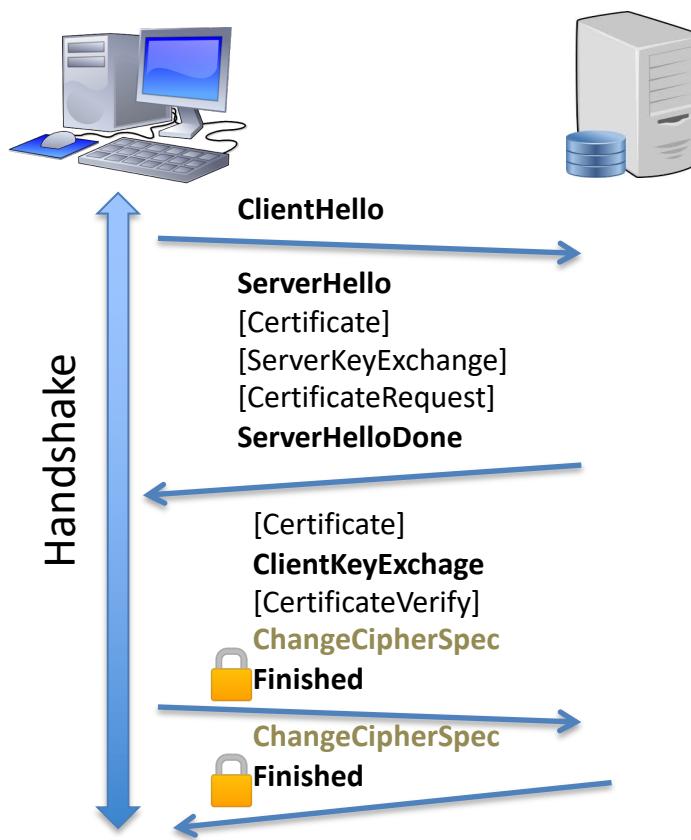


[] – significa opcional

Fase 1: se establece los parámetros de seguridad, incluyendo la versión del protocolo, la sesión de ID, el **suite de cifrado**, un número aleatorio y el método de comprensión (opcional, e incluso, en SSLv3.0 no se especifica, pero sí en TLSv1.2, pero ya en la TLSv1.3 vuelve a ser opcional)

Name of the Cipher Suite	Key Exchange	Cipher	Hash
<i>SSL_NULL_WITH_NULL_NULL</i>	NULL	NULL	NULL
<i>SSL_RSA_WITH_NULL_MD5</i>	RSA	NULL	MD5
<i>SSL_RSA_WITH_NULL_SHA</i>	RSA	NULL	SHA
<i>SSL_RSA_EXPORT_WITH_RC4_40_MD5</i>	RSA_EXPORT	RC4_40	MD5
<i>SSL_RSA_WITH_RC4_128_MD5</i>	RSA	RC4_128	MD5
<i>SSL_RSA_WITH_RC4_128_SHA</i>	RSA	RC4_128	SHA
<i>SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5</i>	RSA_EXPORT	RC2_CBC_40	MD5
<i>SSL_RSA_WITH_IDEA_CBC_SHA</i>	RSA	IDEA_CBC	SHA
<i>SSL_RSA_EXPORT_WITH_DES40_CBC_SHA</i>	RSA_EXPORT	DES40_CBC	SHA
<i>SSL_RSA_WITH_DES_CBC_SHA</i>	RSA	DES_CBC	SHA
<i>SSL_RSA_WITH_3DES_EDE_CBC_SHA</i>	RSA	3DES_EDE_CBC	SHA
<i>SSL_DH_DSS_EXPORT_WITH_DES40_CBC_SHA</i>	DH_DSS_EXPORT	DES40_CBC	SHA
<i>SSL_DH_DSS_WITH_DES_CBC_SHA</i>	DH_DSS	DES_CBC	SHA
<i>SSL_DH_DSS_WITH_3DES_EDE_CBC_SHA</i>	DH_DSS	3DES_EDE_CBC	SHA
<i>SSL_DH_RSA_EXPORT_WITH_DES40_CBC_SHA</i>	DH_RSA_EXPORT	DES40_CBC	SHA
<i>SSL_DH_RSA_WITH_DES_CBC_SHA</i>	DH_RSA	DES_CBC	SHA
<i>SSL_DH_RSA_WITH_3DES_EDE_CBC_SHA</i>	DH_RSA	3DES_EDE_CBC	SHA
<i>SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA</i>	DHE_DSS_EXPORT	DES40_CBC	SHA
<i>SSL_DHE_DSS_WITH_DES_CBC_SHA</i>	DHE_DSS	DES_CBC	SHA
<i>SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA</i>	DHE_DSS	3DES_EDE_CBC	SHA
<i>SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA</i>	DHE_RSA_EXPORT	DES40_CBC	SHA
<i>SSL_DHE_RSA_WITH_DES_CBC_SHA</i>	DHE_RSA	DES_CBC	SHA
<i>SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA</i>	DHE_RSA	3DES_EDE_CBC	SHA
<i>SSL_DH_anon_EXPORT_WITH_RC4_40_MD5</i>	DH_anon_EXPORT	RC4_40	MD5
<i>SSL_DH_anon_WITH_RC4_128_MD5</i>	DH_anon	RC4_128	MD5
<i>SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA</i>	DH_anon	DES40_CBC	SHA
<i>SSL_DH_anon_WITH_DES_CBC_SHA</i>	DH_anon	DES_CBC	SHA
<i>SSL_DH_anon_WITH_3DES_EDE_CBC_SHA</i>	DH_anon	3DES_EDE_CBC	SHA
<i>SSL_FORTEZZA_KEA_WITH_NULL_SHA</i>	FORTEZZA_KEA	NULL	SHA
<i>SSL_FORTEZZA_KEA_WITH_FORTEZZA_CBC_SHA</i>	FORTEZZA_KEA	FORTEZZA_CBC	SHA
<i>SSL_FORTEZZA_KEA_WITH_RC4_128_SHA</i>	FORTEZZA_KEA	RC4_128	SHA

SSL - Secure Sockets Layer



Fase 1: se establece los parámetros de seguridad, incluyendo la versión del protocolo, la sesión de ID, el **suite de cifrado**, un número aleatorio y el método de comprensión (opcional, e incluso, en SSLv3.0 no se especifica, pero sí en TLSv1.2, pero ya en la TLSv1.3 vuelve a ser opcional)

Fase 2: el servidor **puede enviar un certificado (opcional)**, intercambia los parámetros necesarios para gestionar la clave secreta y puede solicitar un certificado al cliente

Fase 3: el cliente envía el certificado si es solicitado y los parámetros de seguridad necesarios para computar la clave de sesión

Si el cliente envía el certificado, entonces necesita enviar un certificado firmado para la verificación de la entidad origen

Fase 4: Se establece el suite de cifrado y se termina el proceso de handshake

INTERCAMBIO DE CLAVES

SSL - Secure Sockets Layer



¿Cómo se crea la clave de sesión?

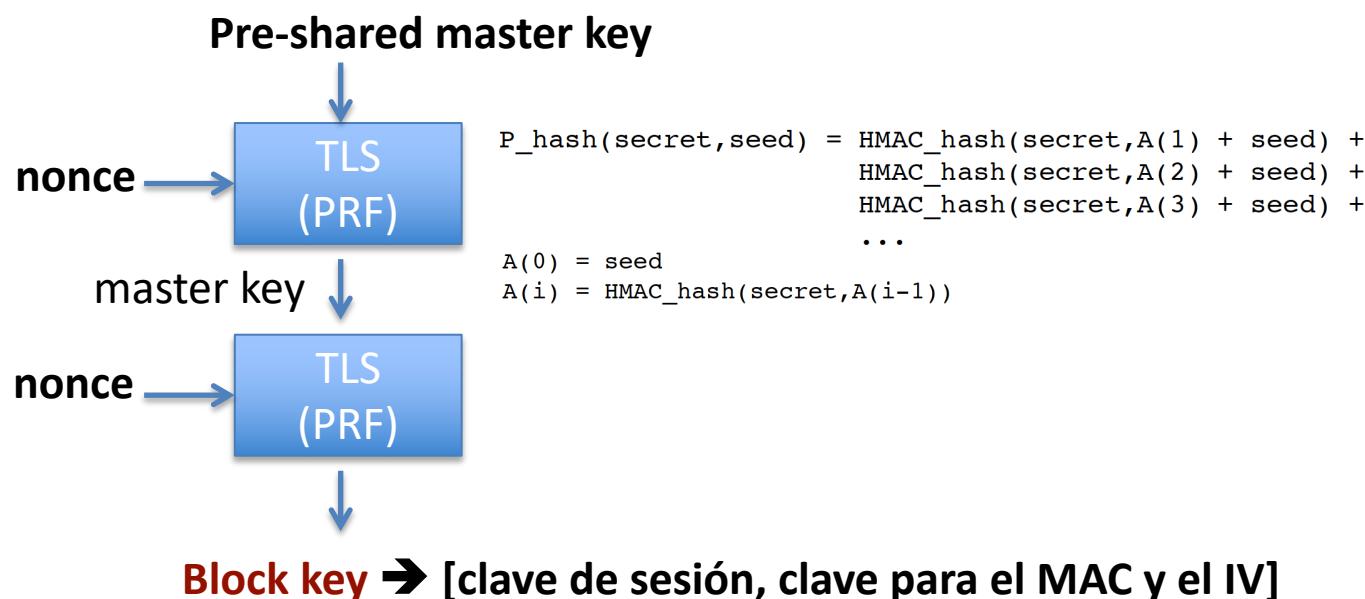
Sin embargo, para proteger la conexión entre el cliente y el servidor necesitamos calcular:

1. la **clave de sesión**
2. una **clave para el MAC**
3. un **IV** para computar el modo de operación

SSL - Secure Sockets Layer



- Los parámetros anteriores se calculan en función de varios parámetros de seguridad:
 - una semilla**
 - un valor aleatorio** (nonce)
 - una función pseudorandom** (PRF –pseudorandom function)



SSL - Secure Sockets Layer - intercambio de claves



Client_random

Paso 1:

- generar números aleatorios (la **salt** para generar después los parámetros de seguridad comentados anteriormente),
- establecer el resto de parámetros de seguridad (ej. tipo de algoritmos de intercambio de clave),
- enviar toda esta información a la otra parte

ClientHello = ({TLS_DHE_RSA_WITH_AES_256_CBC_SHA...}, client_random, ...)

Server_random

ServerHello = (TLS_DHE_RSA_WITH_AES_256_CBC_SHA, server_random)

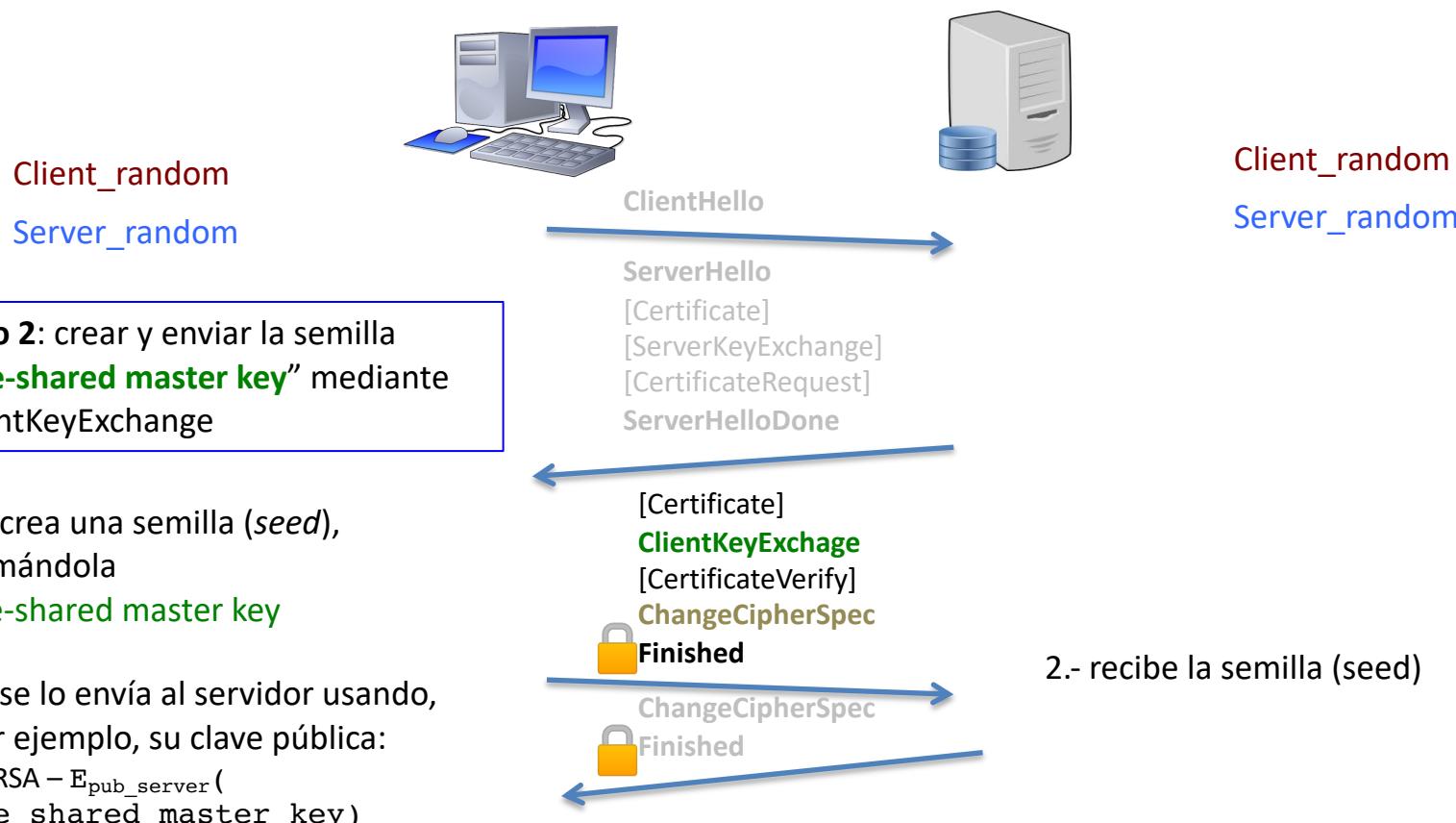
[Certificate]
[ServerKeyExchange]
[CertificateRequest]
ServerHelloDone

[Certificate]
ClientKeyExchange
[CertificateVerify]
ChangeCipherSpec

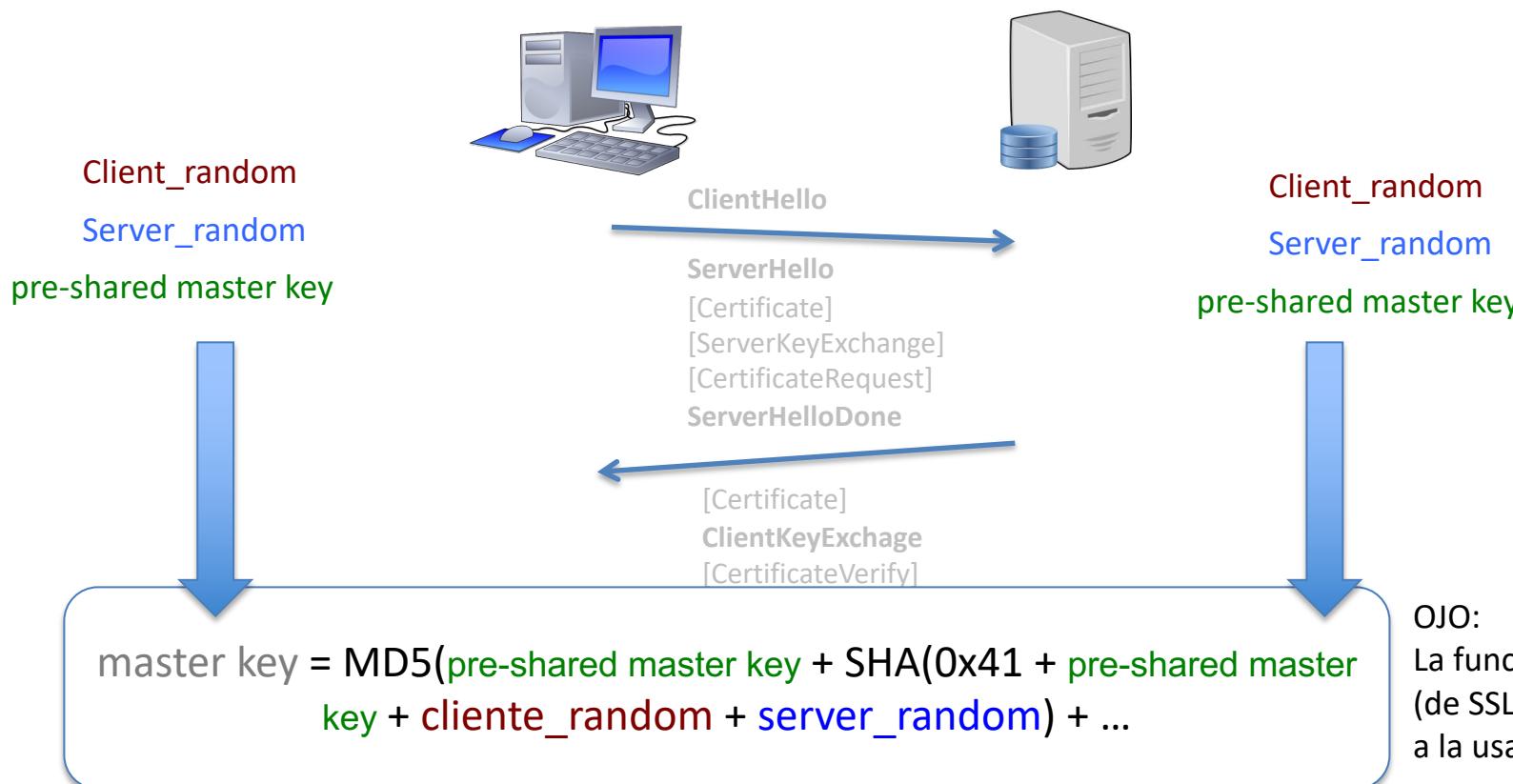


Finished
ChangeCipherSpec
Finished

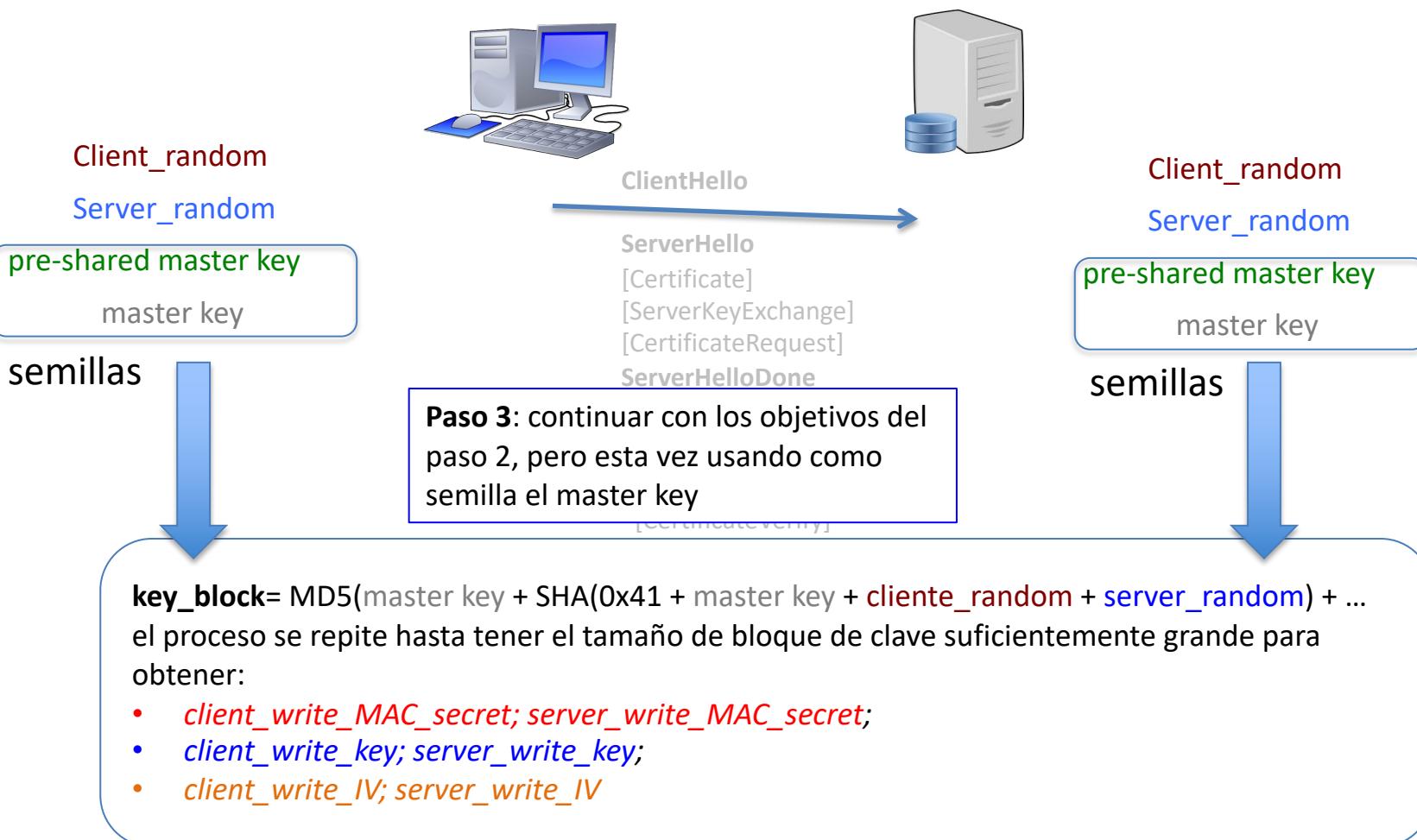
SSL - Secure Sockets Layer - intercambio de claves



SSL - Secure Sockets Layer - intercambio de claves



SSL - Secure Sockets Layer - intercambio de claves



SSL - Secure Sockets Layer - intercambio de claves

- Tanto SSL como TLS usan también DHE (**Diffie Helman efímero**):
 - Sin embargo, antes de entrar en el DHE, recordemos cómo funciona DH

- Alice:

- Valores públicos: q, α
- $X_a < q$, clave privada
- $Y_a = \alpha^{X_a} \text{ mod } q$, clave pública

- $Y_b = \alpha^{X_b} \text{ mod } q$
- $K_{AB} = (Y_b)^{X_a} \text{ mod } q$
- $K_{AB} = (\alpha^{X_b})^{X_a} \text{ mod } q$

- Bob:

- Valores públicos: q, α
- $X_b < q$, clave privada
- $Y_b = \alpha^{X_b} \text{ mod } q$, clave pública

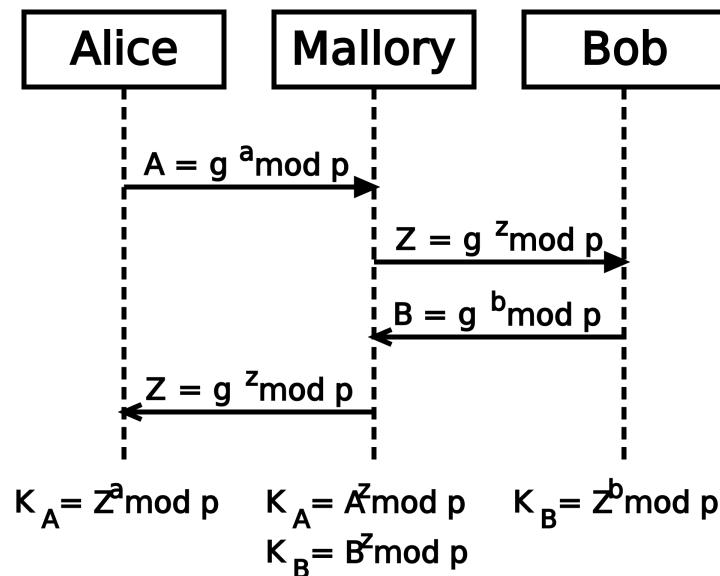
- $Y_a = \alpha^{X_a} \text{ mod } q$
- $K_{AB} = (Y_a)^{X_b} \text{ mod } q$
- $K_{AB} = (\alpha^{X_a})^{X_b} \text{ mod } q$

Recordatorio...

$$K_{AB} = \alpha^{X_b X_a} \text{ mod } q = \alpha^{X_a X_b} \text{ mod } q$$

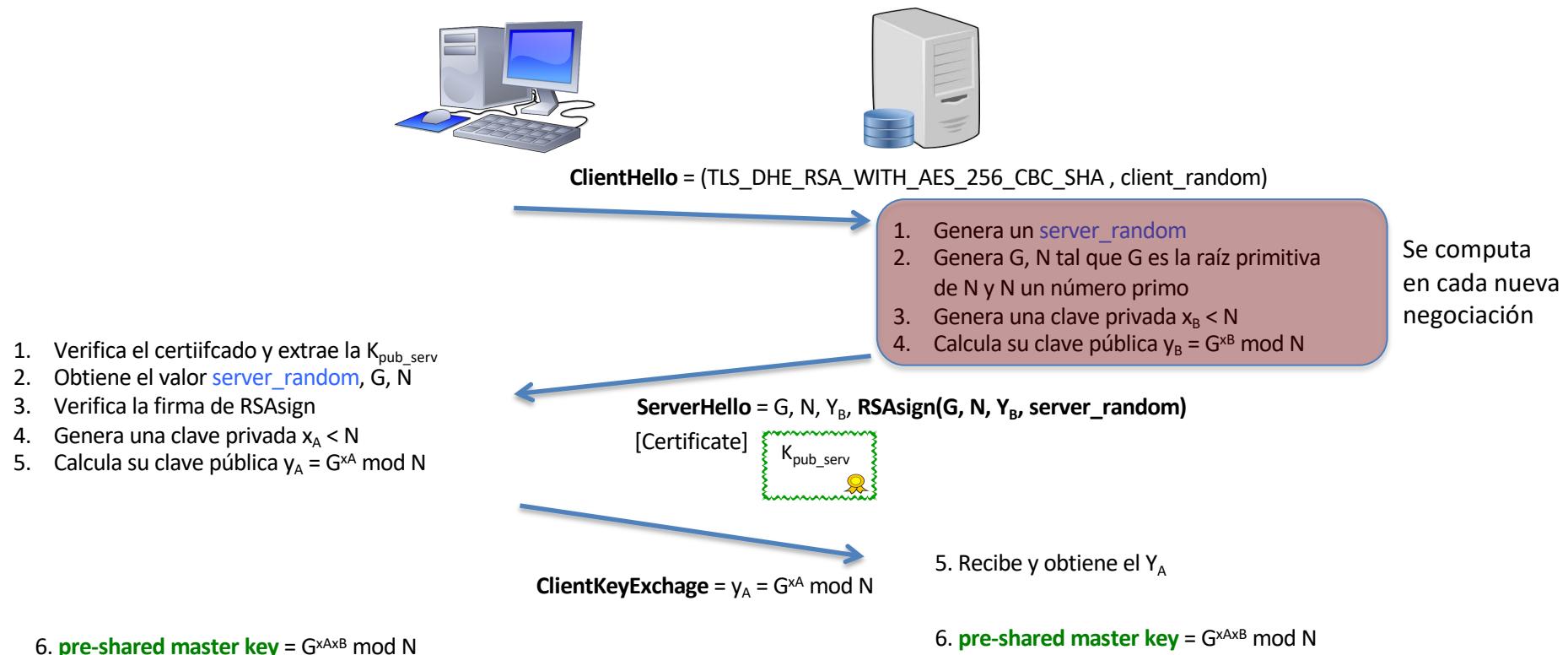
SSL - Secure Sockets Layer - intercambio de claves

- Tanto SSL como TLS usan también DHE (**Diffie Helman efímero**):
 - Tanto el cliente como el servidor generan sus valores secretos en **cada negociación** en lugar de usar valores estáticos para el intercambio
 - P. ej. usar sólo la clave privada en RSA y de manera permanente
 - Esta negociación constante se consigue el **Perfect Forward Secrecy** (PFS) en la creación del secreto compartido
 - El PFS evita el riesgo de MiTM
 - Si la clave privada de RSA (del servidor) se filtra o hay un MiTM en las negociaciones de DH entre dos entidades, entonces el MiTM puede derivar las claves de sección que se establezcan entre ambas entidades legítimas



SSL - Secure Sockets Layer - intercambio de claves

- DHE (Diffie Helman Efímero) con RSA



PROTOCOLOS DE SESIÓN

SSL Handshake Protocol



- Se utiliza antes de transmitir ningún dato de la capa de aplicación
- Es la parte más compleja de SSL porque permite al servidor y al cliente:
 - autenticarse mutuamente
 - negociar un algoritmo de cifrado y una función MAC
 - así como las claves a usar para proteger los datos del SSL record
- Consta de una serie de mensajes intercambiados entre el cliente y el servidor, con el formato:

SSL record

1 byte 2 bytes 2 bytes ≥ 0 bytes



Tipo: 22 → SSL Handshake Protocol

CONTENIDO DEL HANDSHAKE

1 byte	3 bytes	≥ 0 bytes
<i>Tipo</i>	<i>Longitud</i>	<i>Contenido</i>

- Por lo tanto, cada mensaje tiene 3 campos:
 - *Tipo* (1 byte): indica uno de 10 posibles mensajes (ver siguiente tabla)
 - *Longitud* (3 bytes): longitud del mensaje en bytes
 - *Contenido* (≥ 0 bytes): parámetros asociados con el mensaje (ver también siguiente tabla)

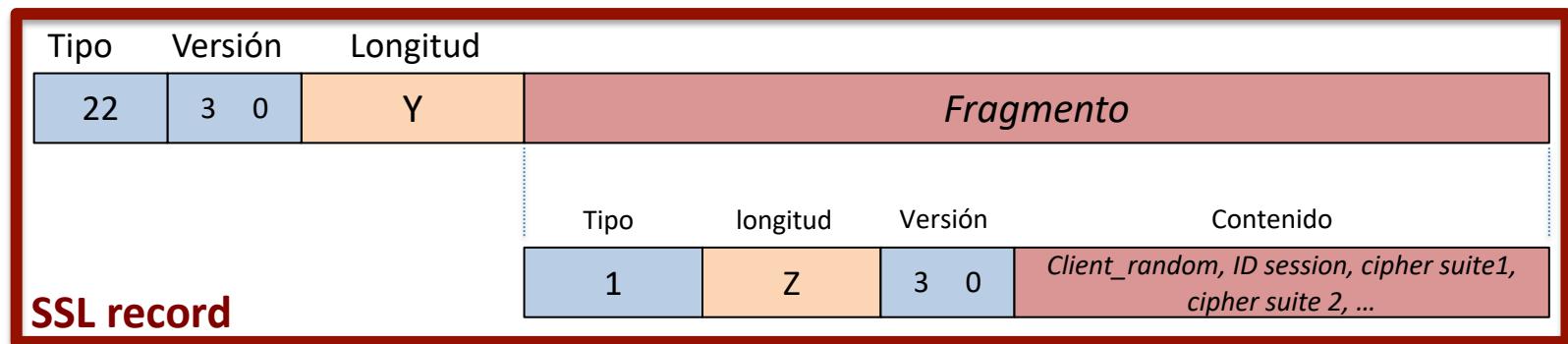
SSL Handshake Protocol

Mensajes	Parámetros asociados con los contenidos (contenido)	Tipo
Hello_request	null	Tipo = 0 Lo solicita el servidor para renegociar una sesión
Client_hello	Versión, random_cliente, sesión ID, cipher suite, método de compresión	Tipo = 1
Server_hello	Versión, random_servidor, sesión ID, cipher suite, método de compresión	Tipo = 2
Certificate	Cadena de certificados X.509	Tipo = 11
Server_key_exchange	Parámetros de seguridad necesarios para computar la clave de sesión (ej. usando DH como seed inicial) y firma del hash(cliente_random + servidor_random + parámetros de seguridad)	Tipo = 12
Certificate_request	Tipo de certificados y autoridades	Tipo = 13
Server_hello_done	Null	Tipo = 14
Client_key_exchange	El pre-shared master key cifrado con RSA, o añade los parámetros de DH/Fortezza para computar el master key en cada una de las partes	Tipo = 16
Certificate_verify	Se aplica cuando se solicita el certificado. Consiste en firmar el hash(master key + hash(todos los mensajes intercambiados hasta el momento))	Tipo = 15
Finished 	Es el cifrado del hash(master_key + hash(hash(todos los mensajes intercambiados hasta el momento) + ID_sender))	Tipo = 20

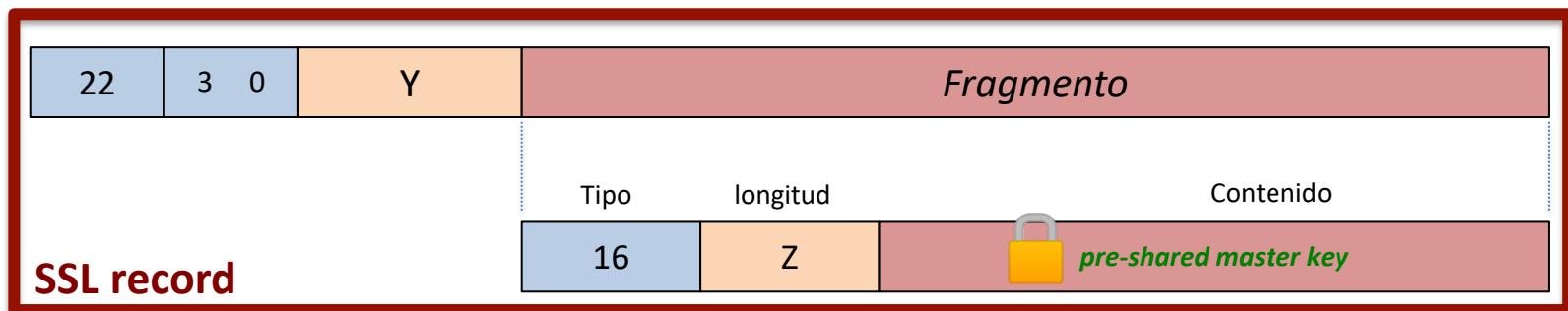
SSL Handshake Protocol



- El formato de algunos mensajes serían:
 - ClientHello



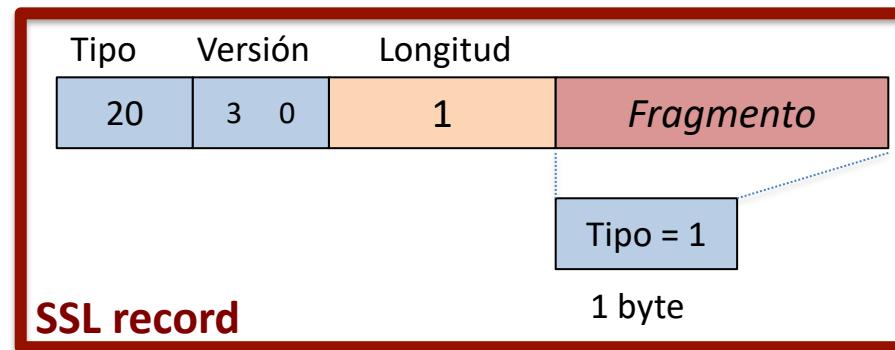
- ClientKeyExchange (RSA)



SSL Change Cipher Spec Protocol



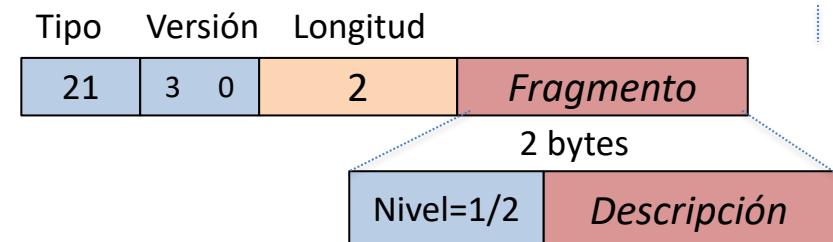
- Es un protocolo muy simple que consta de un solo mensaje de un sólo byte con valor 1 que permite activar el *cipher suite*



SSL Alert Protocol



- Se usa para comunicar al otro punto de comunicación las alertas relacionadas con SSL, y cada mensaje de este protocolo consta de 2 bytes
 - Estos mensajes también se comprimen (opcional) y se cifran de acuerdo con lo establecido en la sesión

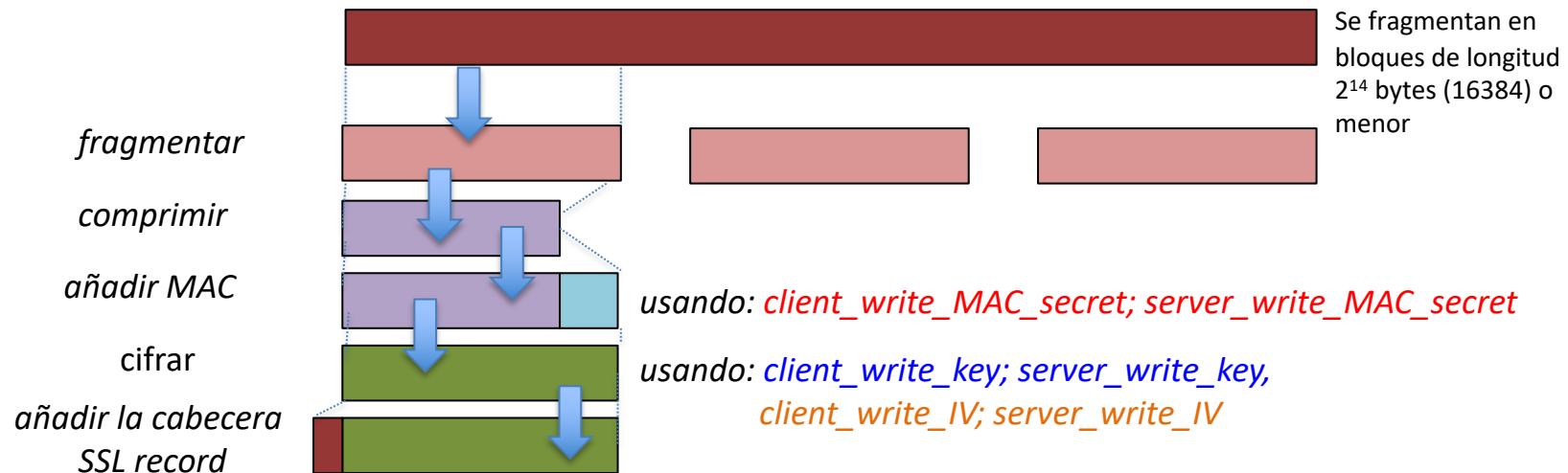


- El primer byte toma el valor 1 (warning) o 2 (fatal) para informar de la severidad del mensaje
 - Si el nivel es fatal, SSL termina la conexión de forma inmediata
 - Otras conexiones de la misma sesión pueden continuar pero no se producen nuevas conexiones dentro de la misma sesión
- El segundo byte contiene un código que indica la alerta específica
 - Ejemplos: *unexpected_message*, *bad_record_mac*, *decompression_failure*, *illegal_parameter*, ...

SSL Record Protocol



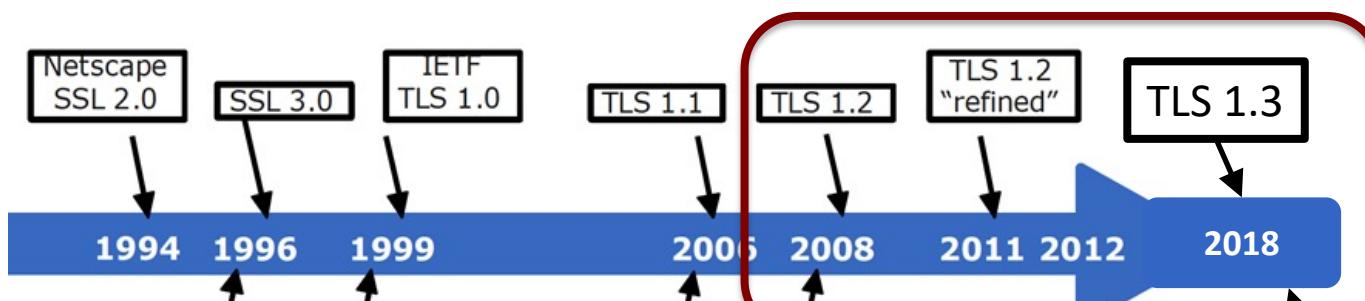
- Toma los datos de la subcapa alta:
 - *los fragmenta en bloques manejables,*
 - *los comprime de forma opcional,*
 - *añade el MAC,*
 - *cifra el paquete, y*
 - *añade una cabecera SSL record*
- El resultado final se trasmite en un segmento TCP



- En el destino, los datos recibidos son descifrados, verificados, descomprimidos y reensamblados antes de entregarlos a la capa de aplicación

TLSV1.2 Y TLSV1.3

Transport Layer Security



Nos centraremos en las dos últimas versiones de TLS

complete redesign

- minor changes
- no interoperation with SSL3
- can downgrade connections to SSL3

2006 2008 2011 2012 2018

- MD5-SHA-1 → SHA-256
- authenticated encryption e.g. AES in CCM mode
- protection against CBC-attacks
- implicit IV → explicit IV

MAC Message Authentication Code

IETF Internet Engineering Task Force

CBC Cipher Block Chaining

IV Initialization Vector

MD5 Message Digest Algorithm

SHA Secure Hash Algorithm

AES Advanced Encryption Standard

CCM Counter with CBC-MAC

Major changes

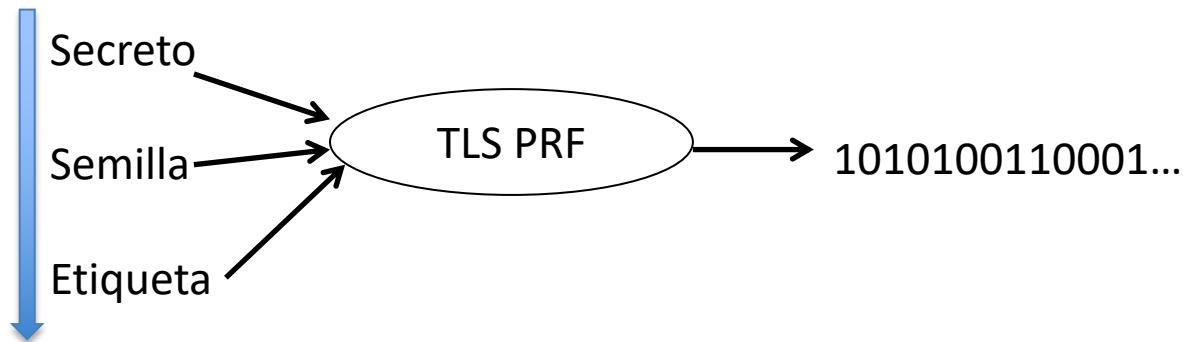
Operation modes are reduced, only AEAD (GCM and AES-CBC-MAC / AES-CCM) are supported, only PFS is applied to generate “pre-shared master key” does not use compression, shorter handshake, 0-RTT

“Authenticated Encryption with Addition Data” (AEAD)

TLSv1.2



- TLS 1.2 introduce cambios muy significativos:
 - Transacciones del handshake:
 - el master-key se computa con SHA-256
 - en vez de usar un generador PRF con MD5 y SHA-1 como lo usa TLSv1.0 y TLSv1.1
 - Concretamente, TLSv1.2 y TLSv1.3 calculan las claves de la siguiente forma:
master key (48 bytes) = PRF-SHA-256(pre-shared master key, “master_key”, client_random + server_random) sabiendo que



key block = PRF(master key, “key_expansion”, client_random + server_random) y de esta bloque de clave se deriva:
 $\text{client_write_MAC_secret}; \text{server_write_MAC_secret}; \text{client_write_key}; \text{server_write_key}; \text{client_write_IV}; \text{server_write_IV}$

- permite incluir una lista de “extensiones” en los mensajes ClientHello/ServerHello
 \rightarrow + detalles sobre los certificados, parámetros de seguridad, autorizaciones, tipos de certificados, etc.

TLSv1.2

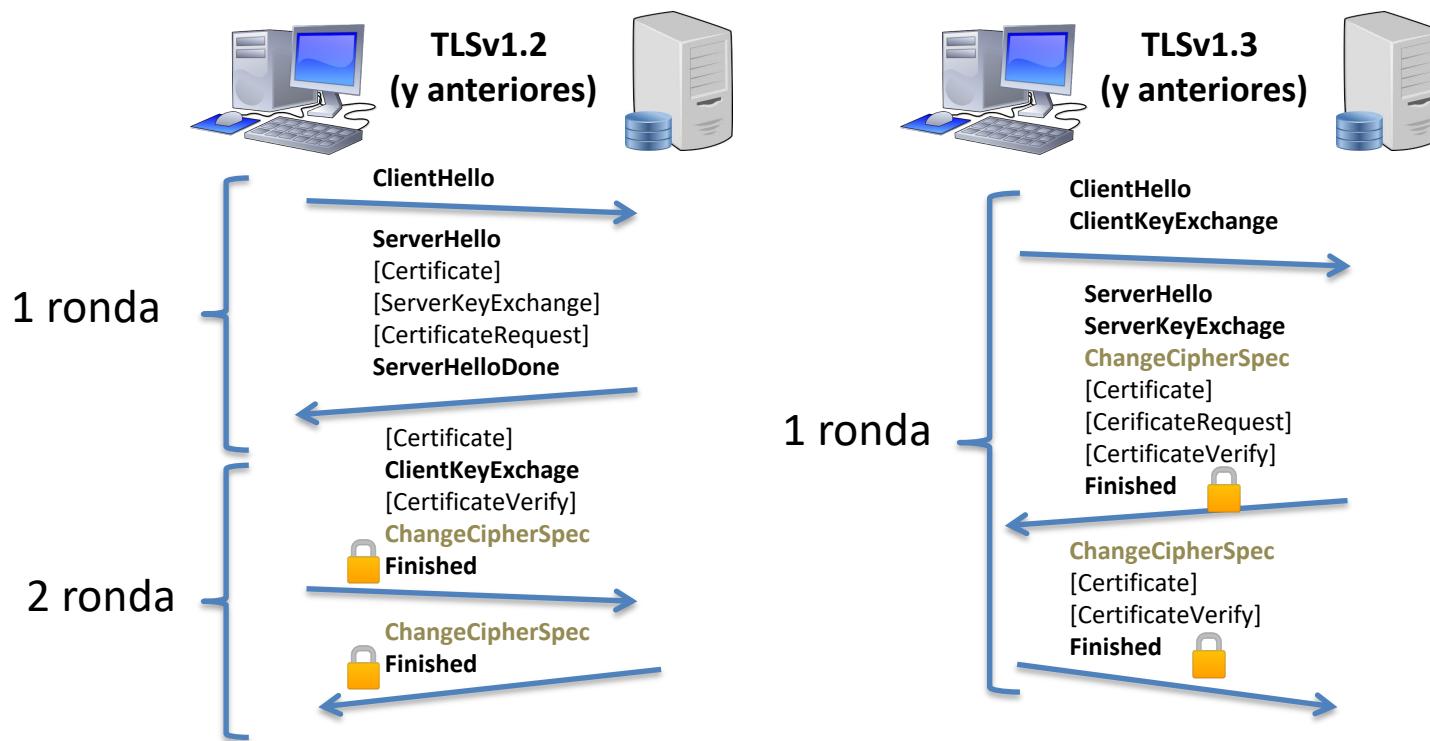


- Cipher suite:
 - Quita DES e IDEA, y añade AES
 - Añade criptografía de clave pública basada en curvas elípticas con ECDHE para la negociación de claves
 - Introduce el concepto de “Authenticated Encryption with Addition Data” (AEAD)
 - AES-CBC-MAC / AES-CCM
 - AES-GCM (Galois-Counter Mode)
 - » Funciona de forma similar que el modo CRT pero usa Carter-Wegman **MAC** en un campo de Galois
 - » Es rápido y eficiente

TLSv1.3

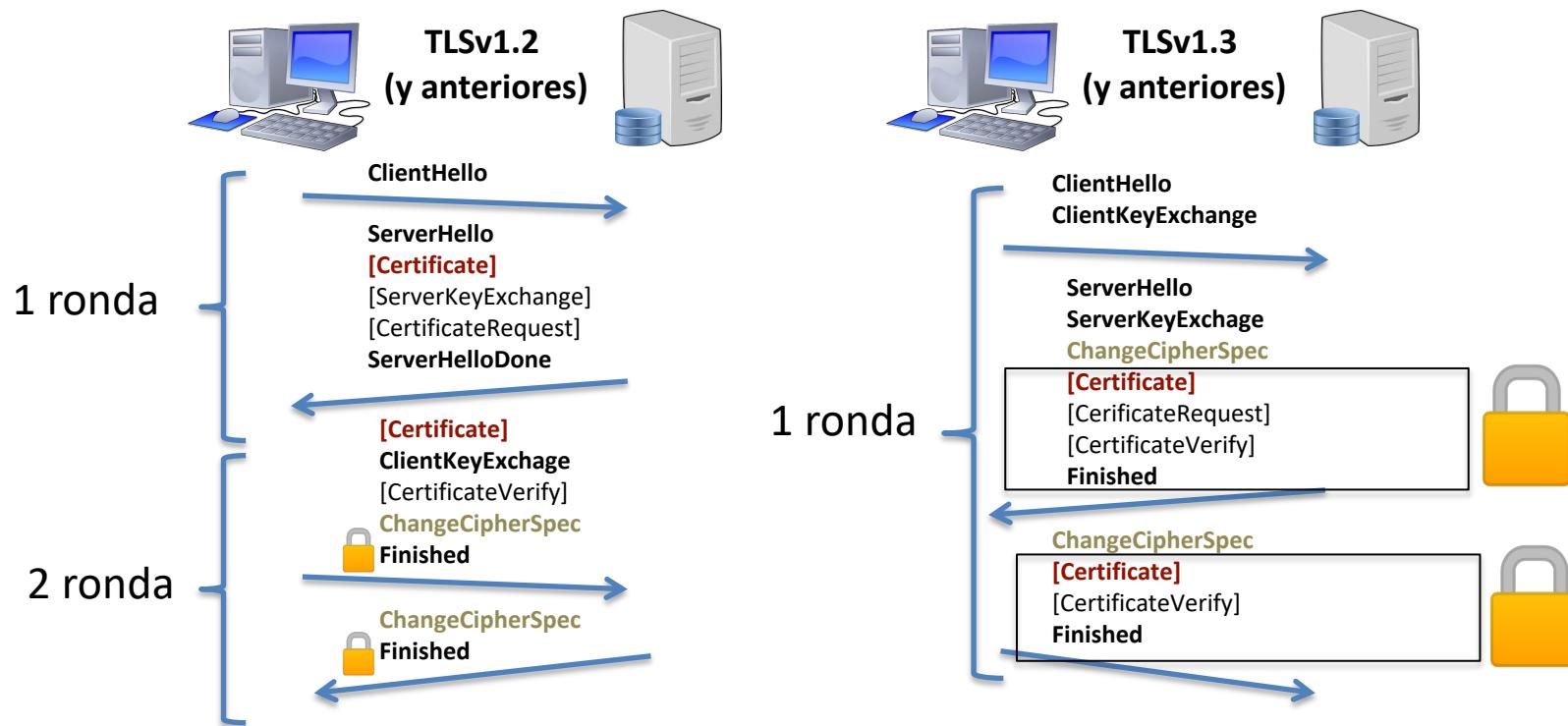


- TLS 1.3 introduce cambios muy significativos en términos de rendimiento y seguridad:
 - Un único Round-Trip Time (RTT)



TLSv1.3

- Además, TLSv1.3 es más seguro:



TLSv1.3



- TLS 1.3 introduce cambios muy significativos en términos de rendimiento y seguridad:
 - Un único Round-Trip Time (RTT)
 - 0-RTT para reconexión por usar las credenciales de seguridad de la sesión anterior – PSK (pre-shared master key)
 - Prevalece sesiones del tipo Perfect Forward Secrecy – DHE-RSA / ECDHE
 - Reduce el número de modos de operación soportados, limitándolo a CBC y AEAD: GCM y CCM

TLS_RSA_WITH_CAMELLIA_128_CBC_SHA256
TLS_DH_DSS_WITH_CAMELLIA_128_CBC_SHA256
TLS_DH_RSA_WITH_CAMELLIA_128_CBC_SHA256
TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA256
TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA256
TLS_DH_anon_WITH_CAMELLIA_128_CBC_SHA256
TLS_RSA_WITH_CAMELLIA_256_CBC_SHA256
TLS_DH_DSS_WITH_CAMELLIA_256_CBC_SHA256
TLS_DH_RSA_WITH_CAMELLIA_256_CBC_SHA256
TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA256
TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA256
TLS_DH_anon_WITH_CAMELLIA_256_CBC_SHA256
TLS_EMPTY_RENEGOTIATION_INFO_SCSV
TLS_FALLBACK_SCSV
TLS_ECDH_ECDSA_WITH_NULL_SHA
TLS_ECDH_ECDSA_WITH_RC4_128_SHA
TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA

TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_ECDSA_WITH_NULL_SHA
TLS_ECDHE_ECDSA_WITH_RC4_128_SHA
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_RSA_WITH_NULL_SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_RSA_WITH_RC4_128_SHA
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
...
...

DTLS

DTLS (Datagram Transport Layer Security)



- Es conveniente comentar que existe un protocolo llamado **DTLS** (Datagram Transport Layer Security) definido en el RFC 6347
 - Se utiliza para los protocolos basados en datagramas
 - Es decir, para los que se ejecutan por encima de UDP
 - Se creó en 2006, aunque la última versión es de Enero de 2012
 - Esta tomando un papel relevante en escenarios que se requiera comunicación en tiempo real o restringidos (IoT)

 Datatracker Groups Documents Meetings Other User Document search

DTLS In Constrained Environments (dice) Concluded WG

[About](#) [Documents](#) [Meetings](#) [History](#) [Photos](#) [Email expansions](#) [List archive](#) [Tools »](#)

Document	Date	Status	IPR	AD / Shepherd
RFC (1 hit)				
RFC 7925 (was draft-ietf-dice-profile) Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things	2016-07 61 pages	Proposed Standard RFC		Stephen Farrell Zach Shelby

Atom feed: [All changes](#) [Significant](#) [Subscribe to changes](#) [!\[\]\(91f1365a772570bcecc722ff94a30ae2_img.jpg\) Export as CSV](#)

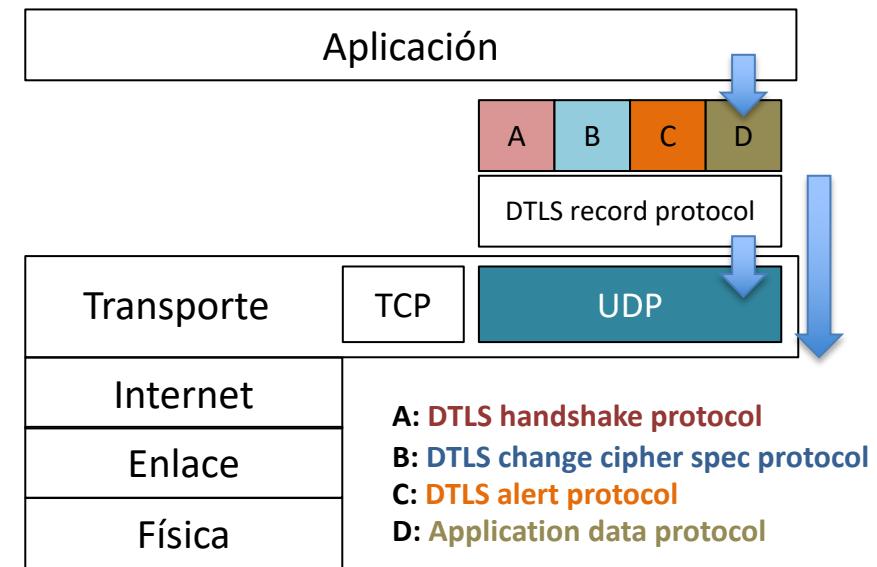
[ISOC](#) [IETF Trust](#) [RFC Editor](#) [IRTF](#) [IESG](#) [IETF](#) [IAB](#) [IASA & IAOC](#) [IETF Tools](#) [IANA](#)

About | IETF Datatracker | Version 6.89.2 | 2018-12-19 | Report a bug: Tracker | Email | Python 2.7.15 | Django 1.11.17

<https://datatracker.ietf.org/wg/dice/documents/>

DTLS (Datagram Transport Layer Security)

- Como se puede ver la estructura es similar al del TLS, pero con la diferencia que la comunicación va sobre UDP
 - esto significa que los datagramas son transmitidos de forma no fiable (puede haber pérdidas o ensamblado no ordenado)
- Para evitar el punto anterior:
 - incluir mecanismos para controlar el flujo de la información
 - un explícito número de secuencia en cada **DTLS record**



Transport Layer Security



Transport Layer Security (tls)

[About](#) [Documents](#) [Meetings](#) [History](#) [Photos](#) [Email expansions](#) [List archive »](#) [Tools »](#)

WG Name Transport Layer Security

Acronym tls

Area Security Area (sec)

State Active

Charter [charter-ietf-tls-06](#) Approved

Status Update [Show update](#) (last changed 2018-11-07)

Dependencies [Document dependency graph \(SVG\)](#)

Additional - [Github](#)

Resources - [Home Page](#)

- [Wiki](#)

Personnel Chairs Christopher Wood

Joseph Salowey

Sean Turner

Area Director Benjamin Kaduk

Mailing list Address tls@ietf.org

To subscribe <https://www.ietf.org/mailman/listinfo/tls>

Archive <https://mailarchive.ietf.org/arch/browse/tls/>

Jabber chat Room address <xmpp:tls@jabber.ietf.org?join>

Logs <https://jabber.ietf.org/logs/tls/>

Charter for Working Group

The TLS (Transport Layer Security) working group was established in 1996 to standardize a 'transport layer' security protocol. The basis for the work was SSL (Secure Socket Layer) v3.0 [RFC6101]. The TLS working group has completed a series of specifications that describe the TLS protocol v1.0 [RFC2246], v1.1 [RFC4546], v1.2 [RFC5346], and v1.3 [RFC8446], and DTLS (Datagram TLS) v1.0 [RFC4347], v1.2 [RFC6347], and v1.3 [draft-ietf-tls-dtls13], as well as extensions to the protocols and ciphersuites.

The working group aims to achieve three goals. First, improve the applicability and suitability of the TLS family of protocols for use in emerging protocols and use cases. This includes extensions or changes that help protocols better use TLS as an authenticated key exchange protocol, or extensions that help protocols better leverage TLS security properties, such as Exported Authenticators. Extensions that focus specifically on protocol extensibility are also in scope. This goal also includes protocol changes that reduce TLS resource consumption without affecting security. Extensions that help reduce TLS handshake size meet this criterion.

The second working group goal is to improve security, privacy, and deployability. This includes, for example, Delegated Credentials and Encrypted SNI. Security and privacy goals will place emphasis on the following:

- Encrypt the ClientHello SNI (Server Name Indication) and other application-sensitive extensions, such as ALPN (Application-Layer Protocol Negotiation).
- Identify and mitigate other (long-term) user tracking or fingerprinting vectors enabled by TLS deployments and implementations.

The third goal is to maintain current and previous version of the (D)TLS protocol as well as to specify general best practices for use of (D)TLS, extensions to (D)TLS, and cipher suites. This includes recommendations as to when a particular version should be deprecated. Changes or additions to older versions of (D)TLS whether via extensions or ciphersuites are discouraged and require significant justification to be taken on as work items.

The working group will also place a priority in minimizing gratuitous changes to (D)TLS.

Transport Layer Security



ietf.org Datatracker Groups Documents Meetings Other User

Transport Layer Security (tls)

About Documents Meetings History Photos Email expansions List archive » Tools »

Document		Date	Status
Active Internet-Drafts (14 hits)			
draft-ietf-tls-ctls-01 Compact TLS 1.3	2020-11-02	I-D Exists WG Document	
draft-ietf-tls-dtls-connection-id-08 Connection Identifiers for DTLS 1.2	2020-11-02	AD Evaluation::AD Followup for 82 days Submitted to IESG for Publication: Proposed Standard	
draft-ietf-tls-dtls13-39 The Datagram Transport Layer Security (DTLS) Protocol Version 1.3	2020-11-02	AD Evaluation::Revised I-D Needed for 23 days Submitted to IESG for Publication: Proposed Standard	
draft-ietf-tls-esni-08 TLS Encrypted Client Hello	2020-10-16	I-D Exists WG Document Mar 2021	
draft-ietf-tls-exported-authenticator-13 Exported Authenticators in TLS	2020-06-26	Waiting for Writeup::Revised I-D Needed for 46 days Submitted to IESG for Publication: Proposed Standard Reviews: genart, opsdir, secdir	
draft-ietf-tls-external-psk-guidance-01 Guidance for External PSK Usage in TLS	2020-11-02	I-D Exists WG Document	
draft-ietf-tls-external-psk-importer-06 Importing External PSKs for TLS	2020-12-03	Waiting for Writeup::AD Followup for 46 days Submitted to IESG for Publication: Proposed Standard Reviews: genart, opsdir, secdir Jan 2021	
draft-ietf-tls-hybrid-design-01 Hybrid key exchange in TLS 1.3	2020-10-15	I-D Exists WG Document	
draft-ietf-tls-md5-sha1-deprecate-04 Deprecating MD5 and SHA-1 signature hashes in TLS 1.2	2020-10-09	Waiting for Writeup::AD Followup for 21 days Submitted to IESG for Publication: Proposed Standard Reviews: genart, intdir, iotdir, opsdir, secdir Jul 2020	
draft-ietf-tls-oldversions-deprecate-09 Deprecating TLSv1.0 and TLSv1.1	2020-11-09	Waiting for Writeup for 6 days Submitted to IESG for Publication: Best Current Practice Reviews: genart, opsdir, secdir	
draft-ietf-tls-rfc8446bis-00 The Transport Layer Security (TLS) Protocol Version 1.3	2020-10-05	I-D Exists WG Document	
draft-ietf-tls-subcerts-09 Delegated Credentials for TLS	2020-06-26	I-D Exists Waiting for WG Chair Go-Ahead: Proposed Standard Sep 2020	
draft-ietf-tls-ticketrequests-07 TLS Ticket Requests	2020-12-03	IESG Evaluation for 1 day IESG telechat: 2020-12-17 Submitted to IESG for Publication: Proposed Standard Reviews: genart, opsdir, secdir Nov 2020	
draft-ietf-tls-tlsflags-03 A Flags Extension for TLS 1.3	2020-07-03	I-D Exists WG Document: Proposed Standard Nov 2020	

Transport Layer Security (tls)

About Documents Meetings History Photos Email expansions List archive » Tools »

WG

Name	Transport Layer Security
Acronym	tls
Area	Security Area (sec)
State	Active
Charter	charter-ietf-tls-06 Approved
Status Update	Show update (last changed 2018-11-07)
Dependencies	Document dependency graph (SVG)
Additional Resources	- Github - Home Page - Wiki

Personnel

Chairs	Christopher Wood Joseph Salowey Sean Turner
Area Director	Benjamin Kaduk

Mailing list

Address	tls@ietf.org
To subscribe	https://www.ietf.org/mailman/listinfo/tls
Archive	https://mailarchive.ietf.org/arch/browse/tls

Jabber chat

Room address	xmpp:tls@jabber.ietf.org?join
Logs	https://jabber.ietf.org/logs/tls/

Charter for Working Group

The TLS (Transport Layer Security) working group was established in 1996 to standardize the TLS protocol. The protocol has evolved through several versions, including v1.0 [RFC2246], v1.1 [RFC4346], v1.2 [RFC5546], and v1.3 [RFC8446]. The working group aims to achieve three goals. First, improve the applicability of extensions that help protocols better leverage TLS security properties, such as forward security. Extensions that help reduce TLS handshake size meet this criterion. The second working group goal is to improve security, privacy, and deployability. - Encrypt the ClientHello SNI (Server Name Indication) and other application-layer protocols. - Identify and mitigate other (long-term) user tracking or fingerprinting vectors. The third goal is to maintain current and previous versions of the (D)TLS protocol, additions to older versions of (D)TLS whether via extensions or ciphersuites and other mechanisms. The working group will also place a priority in minimizing gratuitous changes.

SEGURIDAD EN LA CAPA DE INTERNET



Seguridad en la Capa de Internet

- En 1994, la *Internet Architecture Board (IAB)* publicó un informe titulado “*Security in the Internet Architecture*” (RFC1636)
 - En este informe se identificaba la necesidad de proporcionar **seguridad a la infraestructura de red**
 - En este informe se aconseja la incorporación de mecanismos de
 - cifrado y
 - autenticaciónpara la siguiente versión de IP (IPv6)

The screenshot shows the header and abstract of RFC1636. The header includes links for RFC Home, TEXT, PDF, HTML, Tracker, IPR, and Info page. It also indicates the document is informational. The abstract discusses the purpose of the workshop, its date (February 8-10, 1994), and the status of the memo (unlimited distribution).

[RFC Home] [TEXT] [PDF] [HTML] [Tracker] [IPR] [Info page]

INFORMATIONAL

Network Working Group
Request for Comments: 1636
Category: Informational

R. Braden
ISI
D. Clark
MIT Laboratory for Computer Science
S. Crocker
Trusted Information Systems, Inc.
C. Huitema
INRIA, IAB Chair
June 1994

Report of IAB Workshop on
Security in the Internet Architecture
February 8-10, 1994

Status of this Memo

This memo provides information for the Internet community. This memo does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Abstract

This document is a report on an Internet architecture workshop, initiated by the IAB and held at USC Information Sciences Institute on February 8-10, 1994. This workshop generally focused on security issues in the Internet architecture.

This document should be regarded as a set of working notes containing ideas about security that were developed by Internet experts in a broad spectrum of areas, including routing, mobility, realtime service, and provider requirements, as well as security. It contains some significant diversity of opinions on some important issues. This memo is offered as one input in the process of developing viable security mechanisms and procedures for the Internet.

<https://www.rfc-editor.org/rfc/rfc1636>

Seguridad en la Capa de Internet

- A partir de ese momento se elaboraron, bajo el nombre **IPSec** (RFC 4301), las especificaciones y las funcionalidades de seguridad en la capa de Internet para el modelo TCP/IP
 - no sólo teniendo en cuenta IPv6, sino **también para que sirviera para la propia IPv4**

[RFC Home] [TEXT|PDF|HTML] [Tracker] [IPR] [Errata] [Info page]

PROPOSED STANDARD
Errata Exist
S. Kent
K. Seo
BBN Technologies
December 2005

Updated by: [6040](#), [7619](#)
Network Working Group
Request for Comments: 4301
Obsoletes: [2401](#)
Category: Standards Track

Security Architecture for the Internet Protocol

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2005).

Abstract

This document describes an updated version of the "Security Architecture for IP", which is designed to provide security services for traffic at the IP layer. This document obsoletes [RFC 2401](#) (November 1998).

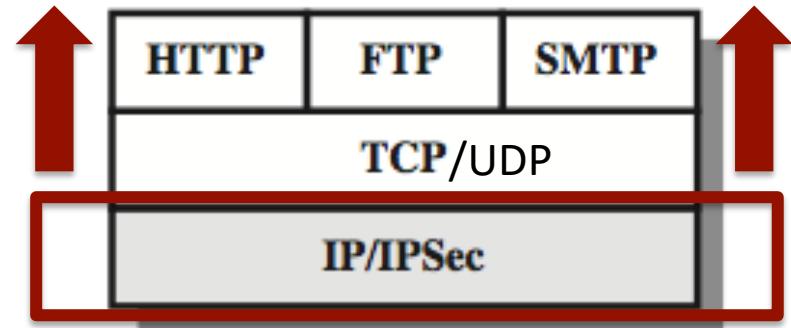
Dedication

This document is dedicated to the memory of Charlie Lynn, a long-time senior colleague at BBN, who made very significant contributions to the IPsec documents.

<https://www.rfc-editor.org/rfc/rfc4301>

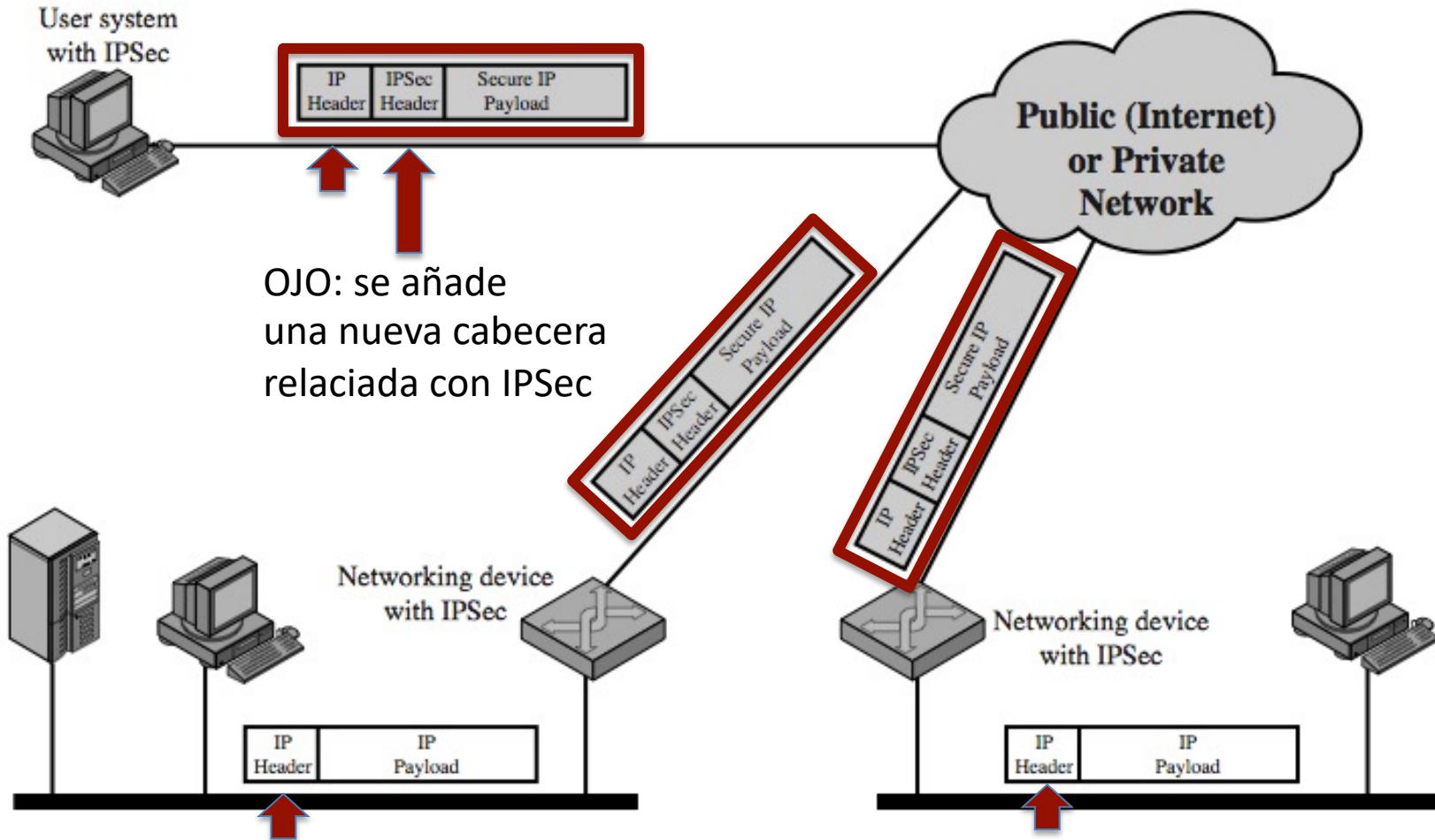
Seguridad en la Capa de Internet

- Implementando la seguridad al nivel de IP, una empresa garantiza la protección de **todas sus aplicaciones**
 - necesiten éstas seguridad o no



- Su aplicación se puede extender a múltiples tipos de escenarios:
 - Conectividad segura entre sucursales a través de Internet
 - Acceso remoto seguro vía el Internet
 - Establecimiento de conectividad extranet e intranet con socios
 - Aplicaciones de comercio electrónico
 - etc.

Seguridad en la Capa de Internet



Seguridad en la Capa de Internet

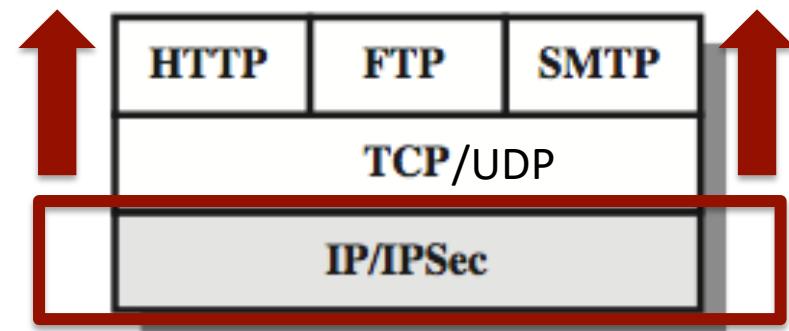
- Como en TLS, la seguridad en IPSec se centra en proporcionar:
 - Autenticación del origen de los mensajes
 - Integridad
 - Confidencialidad
 - Intercambio de claves entre los puntos que se comunican

Esto se consigue con:

- MAC
- cifrado y
- algoritmos para el intercambio de clave (ej. DH)
- IPSec **NO** proporciona:
 - servicios de no-repudio, como SSL/TLS, y
 - protección frente a ataques DoS, aunque sí proporciona una forma de protección ante ataques de repetición

Seguridad en la Capa de Internet

- Como se ha comentado anteriormente, al funcionar por debajo del nivel de transporte, es transparente a las aplicaciones
- Por lo tanto, es transparente a los usuarios finales:
 - no es necesario informar a los usuarios sobre el uso de mecanismos de seguridad
 - no hace falta que gestionen claves

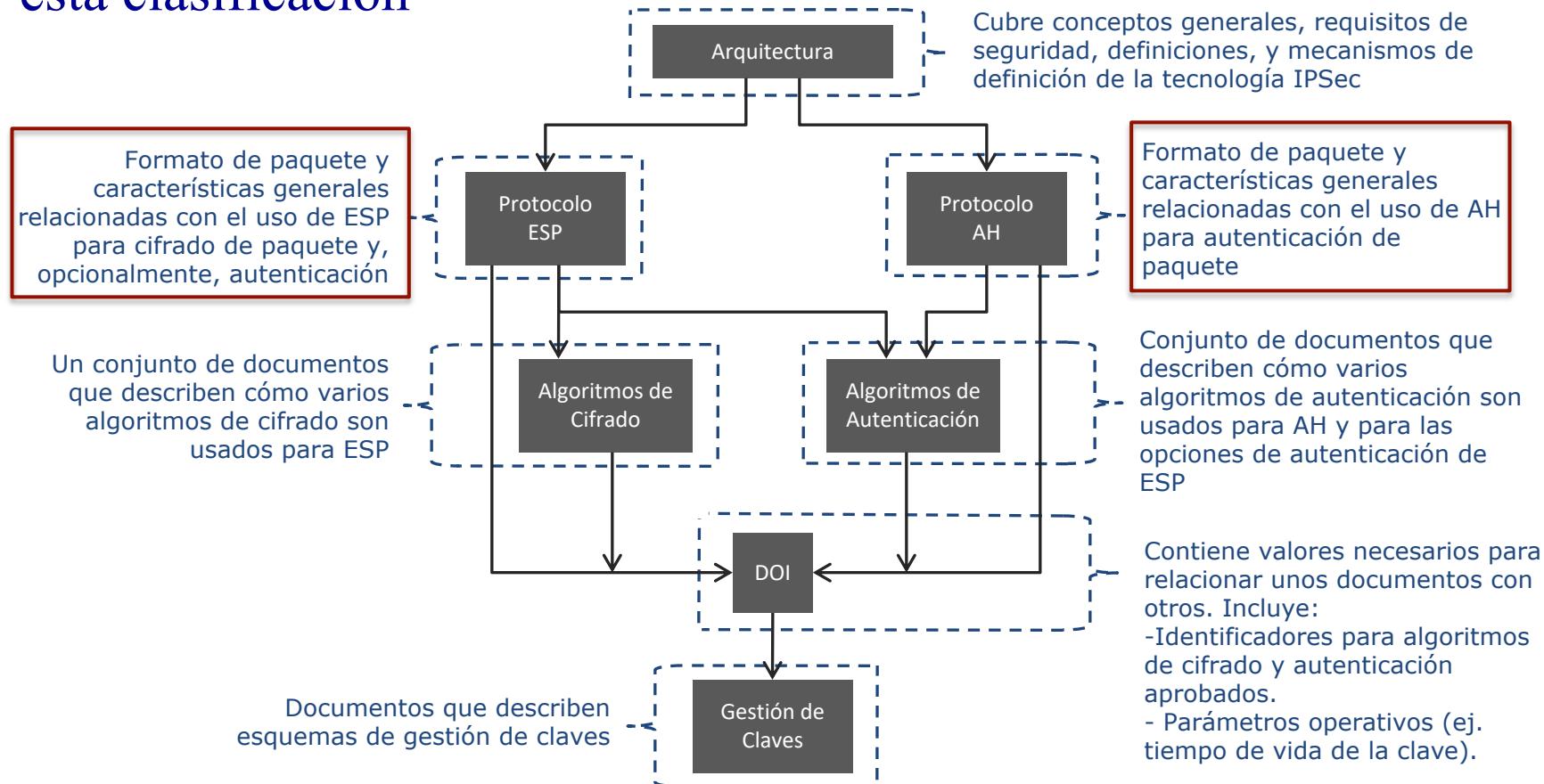


Seguridad en la Capa de Internet

- Para la comunicación segura entre dos puntos, IPSec utiliza los siguientes protocolos:
 - **ESP (Encapsulating Security Payload)**
 - Garantiza **confidencialidad, integridad y autenticación (opcional)**
 - Principalmente proporciona protección frente a lecturas ilícitas de tráfico en red
 - También incluye un número de secuencia que proporciona una forma de **protección ante ataques de repetición**
 - **AH (Authentication Header)**
 - Garantiza **integridad y autenticación** del origen de datos
 - También incluye un número de secuencia que proporciona una forma de **protección ante ataques de repetición** (lo mismo que ESP)
 - **IKE (Internet Key Exchange)**
 - Protocolo específico para **generar y distribuir claves** para ESP y AH
 - También autentica la identidad del sistema remoto
- Cuando usando estos tres protocolos:
 - Antes de que dos puntos se comuniquen de forma segura, tienen que acordar qué **parámetros de seguridad** se van a aplicar

Seguridad en la Capa de Internet

- De hecho, los documentos de IETF al respecto de IPSec siguen esta clasificación



DOI: Domain of Interpretation

Seguridad en la Capa de Internet

- La siguiente tabla muestra algunos de los códigos de protocolos de Internet, asignados por IANA (Internet Assigned Numbers Authority)

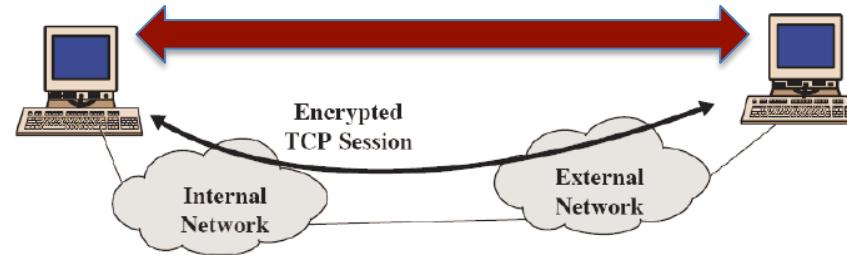
Some IP protocol codes	
Protocol code	Protocol Description
1	ICMP — Internet Control Message Protocol
2	IGMP — Internet Group Management Protocol
4	IP within IP (a kind of encapsulation)
6	TCP — Transmission Control Protocol
17	UDP — User Datagram Protocol
41	IPv6 — next-generation TCP/IP
47	GRE — Generic Router Encapsulation (used by PPTP)
50	IPsec: ESP — Encapsulating Security Payload
51	IPsec: AH — Authentication Header



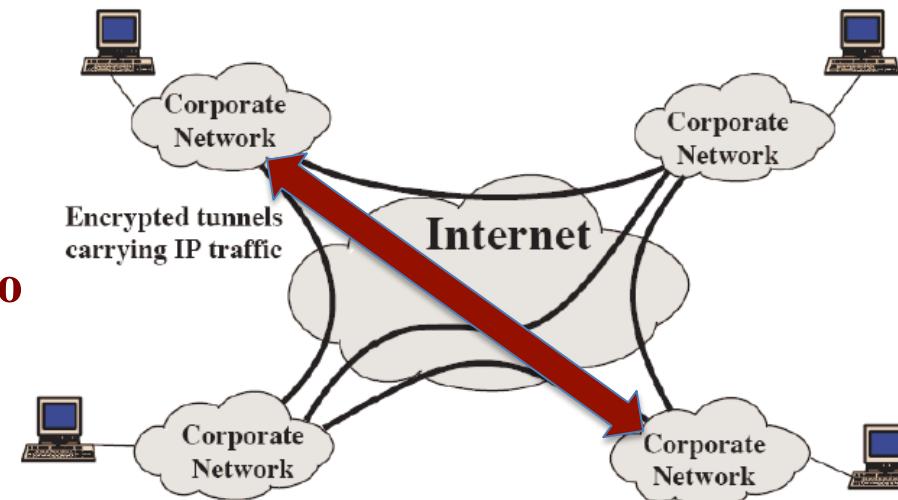
- Lista completa en:
<http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml>

Modos de IPSec

- Modo transporte:
 - Se usa normalmente para comunicaciones punto a punto entre **dos hosts**
 - Proporciona protección a la carga útil del paquete IP (**IP payload**)
 - es decir, a los protocolos de la capa superior - TCP, UDP, ICMP, ...

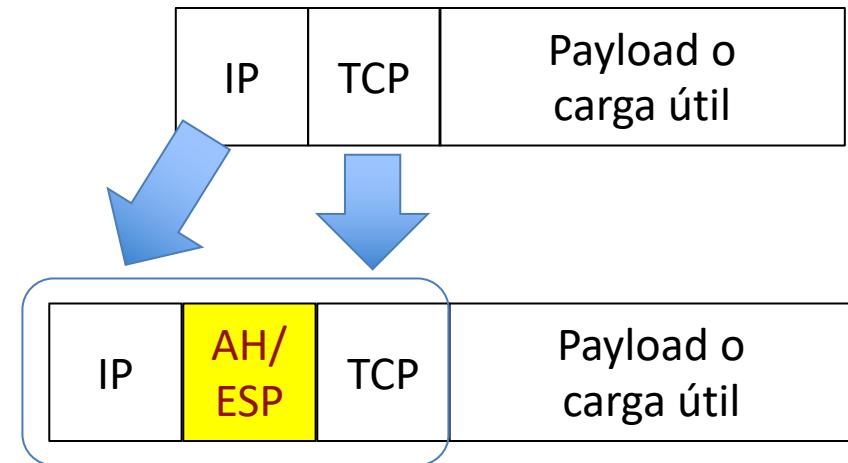


- Modo túnel:
 - Se usa cuando los puntos a comunicar son **gateways** de seguridad o **routers**
 - Proporciona **protección a todo el paquete IP**



Modos de IPSec

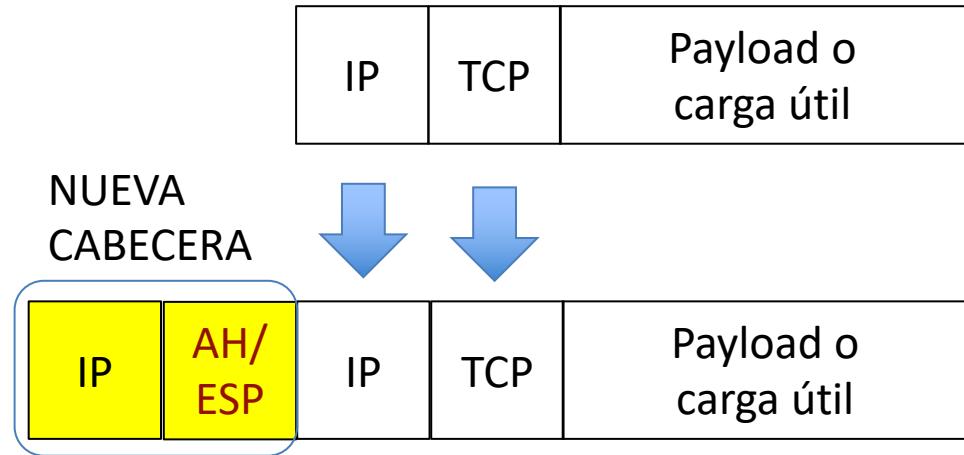
- **El modo transporte**
IPsec sólo protege la carga del datagrama IP
 - insertando la cabecera IPsec (AH, ESP) entre la cabecera IP y la cabecera del protocolo de capas superiores



- *IP origen – host*
- *IP destino - host*
- Cifrado
- Autenticación
- Integridad

Modos de IPSec

- El modo túnel encapsula TODO el datagrama IP dentro de un nuevo datagrama IP
 - donde añade la IP de los routers/gateways y la información del protocolo IPsec (AH, ESP)
- Por tanto, la protección es en todo el paquete IP:
 - añadiendo las cabecera AH o ESP al paquete IP original, y
 - creando una cabecera IP nueva
- De esta forma el paquete IP original (paquete interno) se “**encapsula**” y viaja por el túnel sin que ninguno de los routers intermedios pueda saber **ni el origen ni el destino final de los datos**

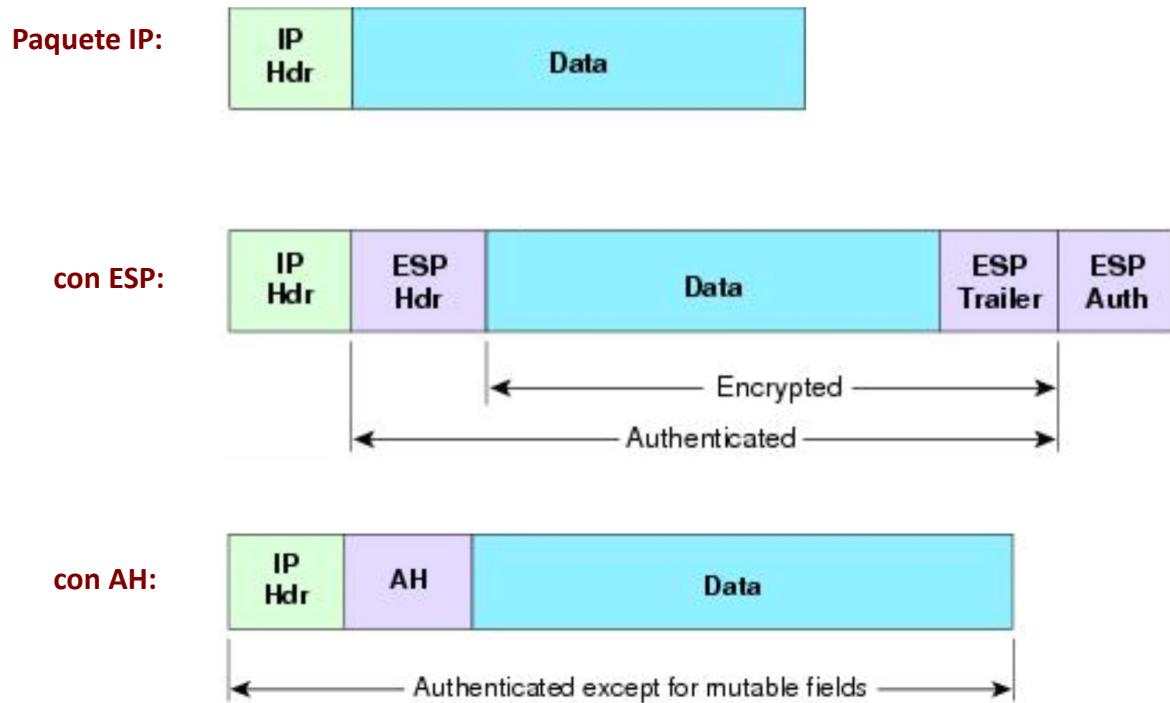


- *IP origen – router*
- *IP destino - router*
- Cifrado
- Autenticación
- Integridad

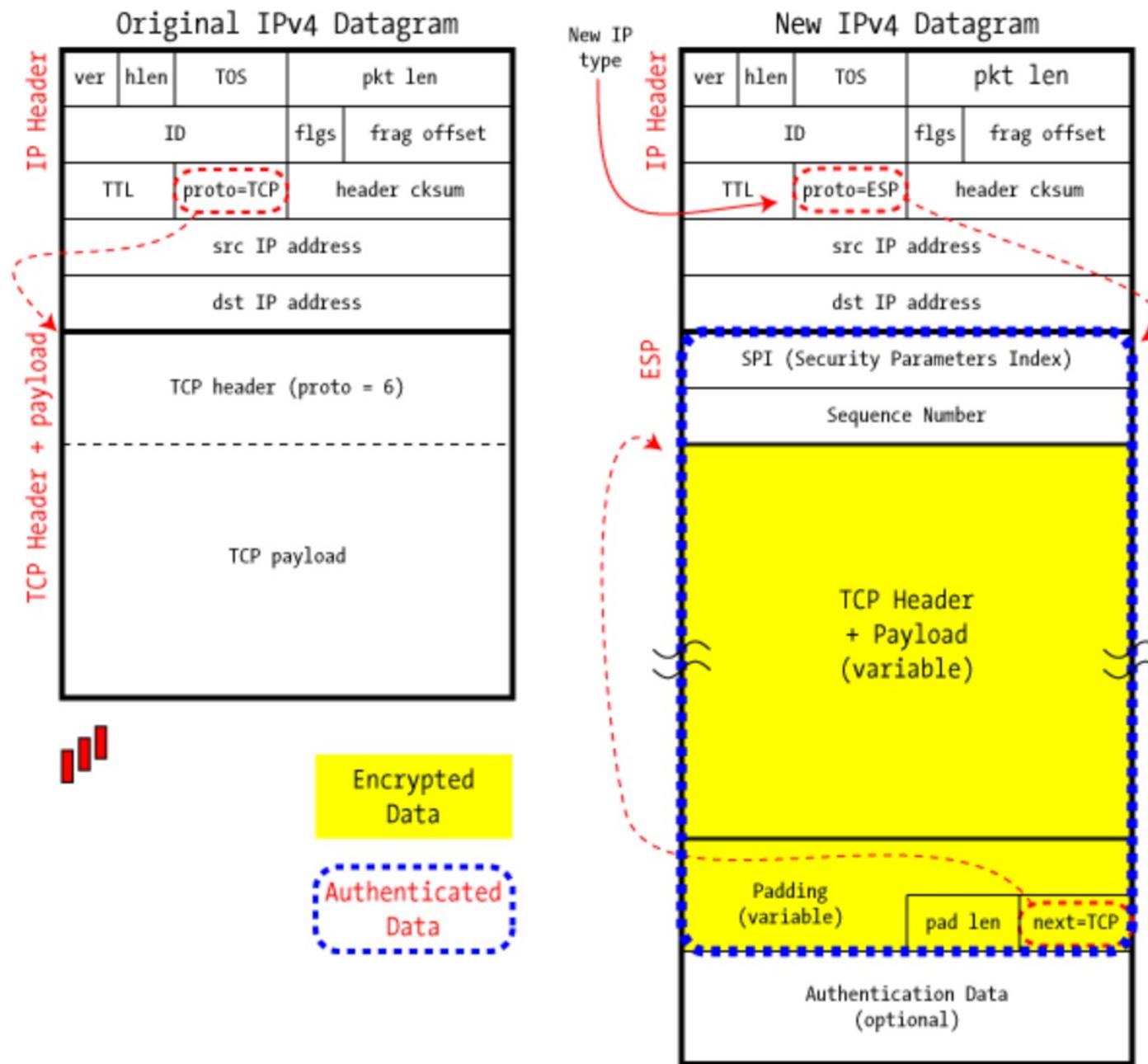
Modo transporte

- En este modo de uso:

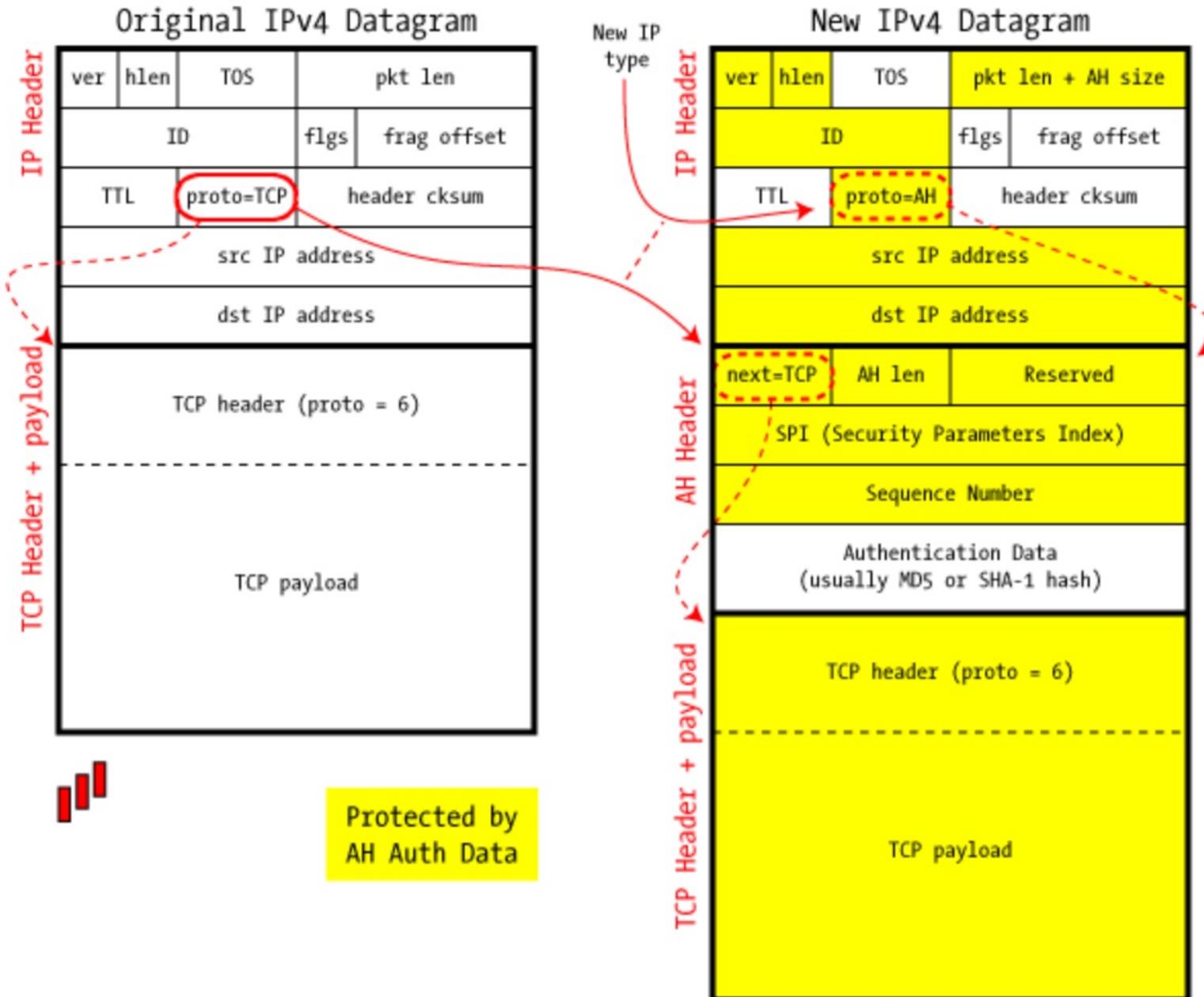
- **si se utiliza ESP:** se cifra el *payload* y opcionalmente lo autentica, pero no la cabecera IP
- **si se utiliza AH:** se autentica el *payload* y algunas porciones de la cabecera IP



IPSec in ESP Transport Mode



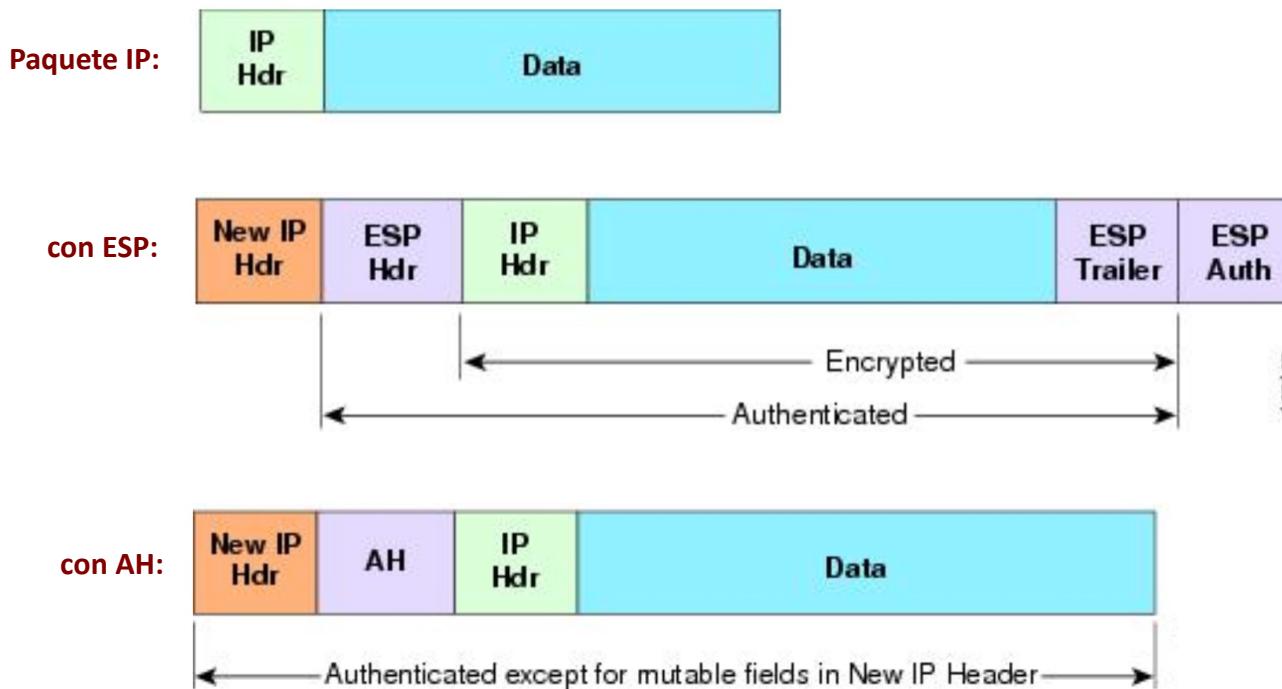
IPSec in AH Transport Mode



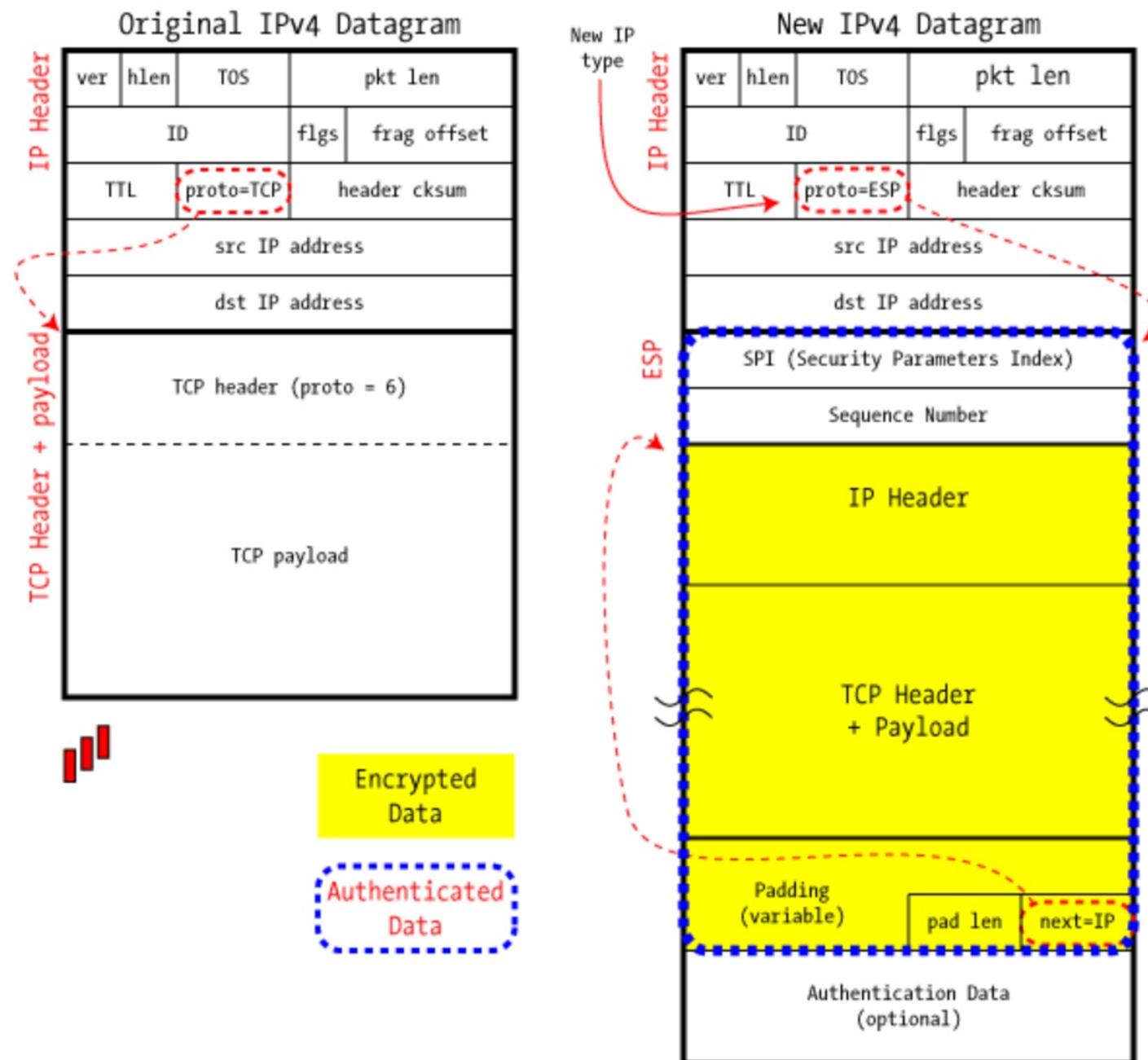
Modo túnel

- En este modo de uso:

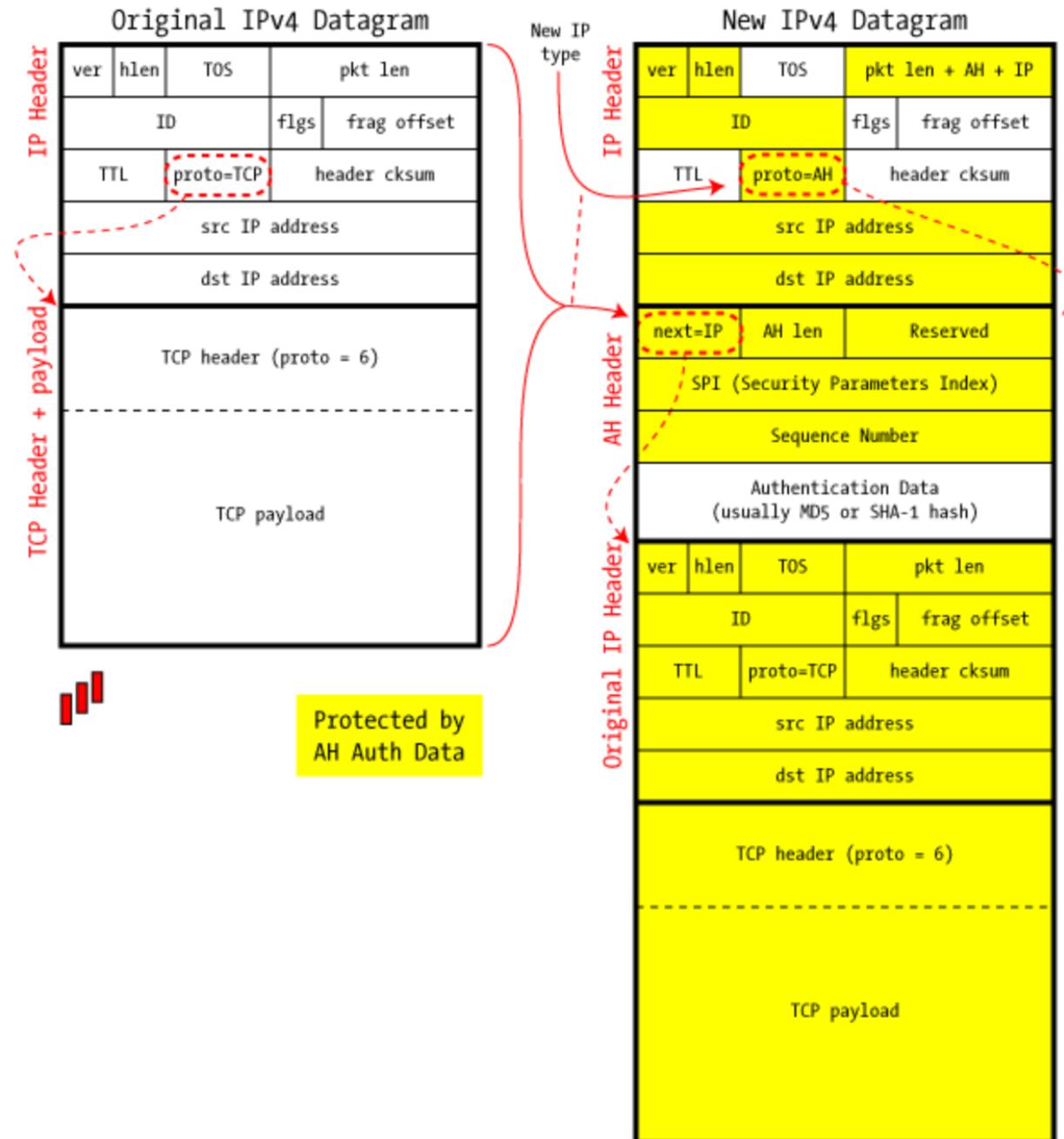
- si se usa **ESP**: se cifra y opcionalmente autentica todo el paquete IP original (paquete interno), incluyendo la cabecera de ese paquete original
- si se usa **AH**: se autentica todo el paquete original y algunas partes de la nueva cabecera externa



IPSec in ESP Tunnel Mode

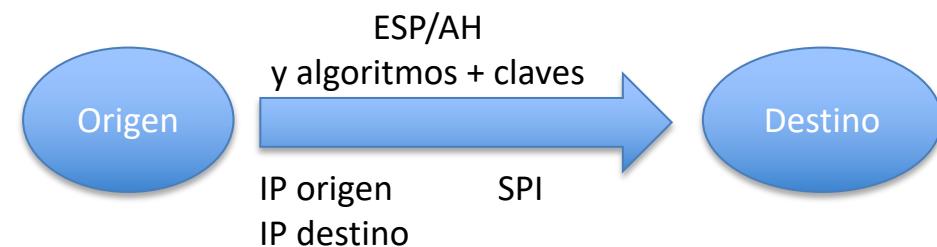


IPSec in AH Tunnel Mode



Asociaciones de seguridad en IPSec

- Para activar IPSec es necesario configurar:
 - El origen y el destino de los paquetes IPSec
 - El modo de autenticación de los mensajes
 - P. ej. el HMAC basado en MD5 o SHA
 - El algoritmo de cifrado
 - P. ej: DES, 3DES, AES y Blowfish
 - El índice de parámetro de seguridad (**SPI - Security Parameter Index**)
 - Es un número de 32 bits único para cada asociación de seguridad definida para ESP / AH
 - Un número de secuencia única de paquetes para controlar los ataques *replay*
 - *Sequence Number* – de las transparencias anteriores
 - Sólo se aceptan paquetes que tienen un número actual de secuencia o posterior, las anteriores se descartan
- A toda esta información se le conoce con el nombre de **Asociación de seguridad (SA – Security Associations)**
 - OJO, sólo se protege un sentido
 - El emisor y el receptor deben aplicar la misma SA, pero teniendo en cuenta el destino y el origen



No.	Time	Source	Destination	Protocol	Length	Info
134	215.661944	190.0.0.1	190.0.0.14	ESP	134	ESP (SPI=0x00000070)
135	216.661895	190.0.0.1	190.0.0.14	ESP	134	ESP (SPI=0x00000070)
136	217.663971	190.0.0.1	190.0.0.14	ESP	134	ESP (SPI=0x00000070)
137	218.681817	190.0.0.1	190.0.0.14	ESP	134	ESP (SPI=0x00000070)
138	219.681772	190.0.0.1	190.0.0.14	ESP	134	ESP (SPI=0x00000070)
139	220.681692	190.0.0.1	190.0.0.14	ESP	134	ESP (SPI=0x00000070)
140	230.684156	190.0.0.1	190.0.0.15	ESP	122	ESP (SPI=0x00000071)
141	231.683968	190.0.0.1	190.0.0.15	ESP	122	ESP (SPI=0x00000071)
142	232.683915	190.0.0.1	190.0.0.15	ESP	122	ESP (SPI=0x00000071)
143	233.683761	190.0.0.1	190.0.0.15	ESP	122	ESP (SPI=0x00000071)

► Frame 143: 122 bytes on wire (976 bits), 122 bytes captured (976 bits)
 ► Ethernet II, Src: Dell_4a:d7:0a (00:11:43:4a:d7:0a), Dst: 00:0c:29:00:00:15 (00:00:00:00:00:15)
 ▼ Internet Protocol Version 4, Src: 190.0.0.1, Dst: 190.0.0.15
 0100 = Version: 4
 0101 = Header Length: 20 bytes
 ► Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECN)
 Total Length: 108
 Identification: 0x0003 (3)
 ► Flags: 0x02 (Don't Fragment)
 Fragment offset: 0
 Time to live: 64
 Protocol: Encap Security Payload (50)
 ► Header checksum: 0xbe4c [validation disabled]
 Source: 190.0.0.1
 Destination: 190.0.0.15
 [Source GeoIP: Unknown]
 [Destination GeoIP: Unknown]
 ▼ Encapsulating Security Payload
 ESP SPI: 0x00000071 (113)
 ESP Sequence: 6

0000	00 00 00 00 00 15 00 11 43 4a d7 0a 08 00 45 00E. l..@2
0010	00 6c 00 03 40 00 40 32 be 4c be 00 00 01 be 00	.L.....
0020	00 0f 00 00 00 71 00 00 00 06 08 00 0a 18 7e 64	...q.....~d
0030	00 04 3b a9 f9 43 44 8f 0b 00 08 09 0a 0b 0c 0d	...;..CD.....
0040	0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d
0050	1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d	.. !#\$% &'()*+,-./012345 67.....
0060	2e 2f 30 31 32 33 34 35 36 37 01 02 02 01 ab f8	...0)g/. l.
0070	af 87 f4 4f 29 67 2f c4 6c c8	

Asociaciones de seguridad en IPSec

- Las SAs se almacenan en una **base de datos de asociaciones de seguridad (SAD)**
- Cada entrada en la SAD existen varios campos:
 - *Security Parameter Index (SPI)*:
 - Es un valor de 32 bits para identificar a una SA particular
 - *AH Information*:
 - Algoritmo de autenticación, claves y otros parámetros relacionados con AH
 - *ESP Information*:
 - Algoritmo de cifrado y autenticación, claves y otros parámetros relacionados con ESP
 - *Lifetime of the SA*:
 - Un intervalo o un contador después del cual habrá que reemplazar la SA
- Algunas SAD también definen:
 - El tipo de modo (túnel o transporte)

Políticas de seguridad en IPSec

- Sin embargo, SA sólo especifica “*el modo en el que se protegerá el dato IPSec*”. Para definir “***el modo en cómo va a viajar el tráfico entre dos puntos***”, se requiere de una **política de seguridad (SP – Security Policy)** que se almacena en una **SPD (Security Policy Database)**
- Un SP define:
 - Las direcciones de origen y destino a proteger
 - En modo transporte, éstas serán las mismas direcciones que aquellas definidas en la SA
 - En modo túnel NO son las mismas
 - Los protocolos y puertos a proteger
 - Algunas implementaciones no permiten la definición de protocolos específicos a proteger
 - Si no se especifica nada, se protege todo el tráfico
 - El modo de protección: túnel o transporte

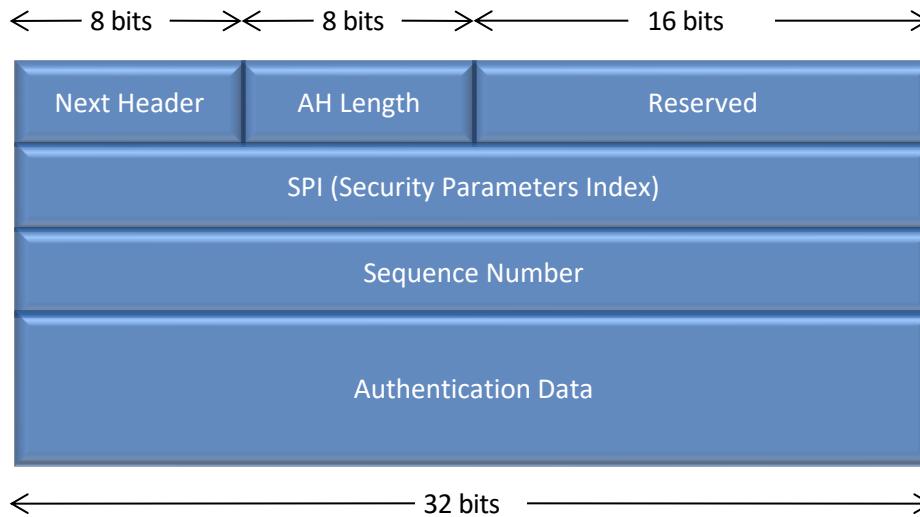
Políticas de seguridad en IPSec

- Un ejemplo de SPD es el siguiente:

Protocol	Local IP	Port	Remote IP	Port	Action	Comment
UDP	1.2.3.101	500	*	500	BYPASS	IKE
ICMP	1.2.3.101	*	*	*	BYPASS	Error messages
*	1.2.3.101	*	1.2.3.0/24	*	PROTECT: ESP intransport-mode	Encrypt intranet traffic
TCP	1.2.3.101	*	1.2.4.10	80	PROTECT: ESP intransport-mode	Encrypt to server
TCP	1.2.3.101	*	1.2.4.10	443	BYPASS	TLS: avoid double encryption
*	1.2.3.101	*	1.2.4.0/24	*	DISCARD	Others in DMZ
*	1.2.3.101	*	*	*	BYPASS	Internet

Cabeceras AH y ESP

- Authentication Header - AH



Next Header (8 bits): Identifica el tipo de cabecera inmediatamente posterior a esta cabecera

AH Length (8 bits): Longitud de la Cabecera de Autenticación en palabras de 32 bits, menos 2 palabras

Reserved (16 bits): Para uso futuro

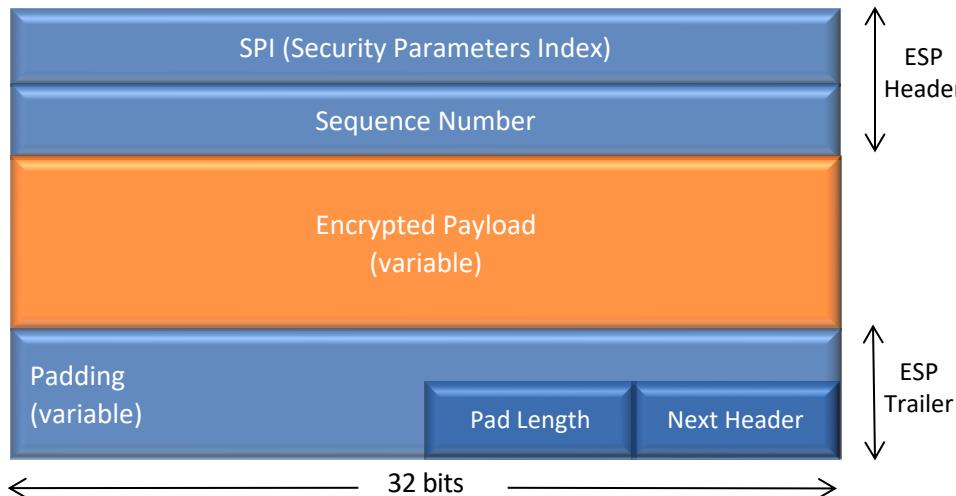
SPI (Security Parameters Index) (32 bits): Identifica una “asociación de seguridad”

Sequence Number (32 bits): contador para evitar ataques de repetición

Authentication Data (variable): Campo de longitud variable (debe ser un número de palabras de 32 bits) que contiene el valor de comprobación de integridad (valor MAC) para este paquete

Cabeceras AH y ESP

- Encapsulating Security Payload – ESP (sólo cifrado)



SPI (Security Parameters Index) (32 bits): Identifica una “asociación de seguridad”

Sequence Number (32 bits): **contador para evitar ataques de repetición**

Encrypted Payload (32 bits): Segmento de nivel de transporte (modo transporte) o paquete IP (modo túnel) protegido por medio de cifrado

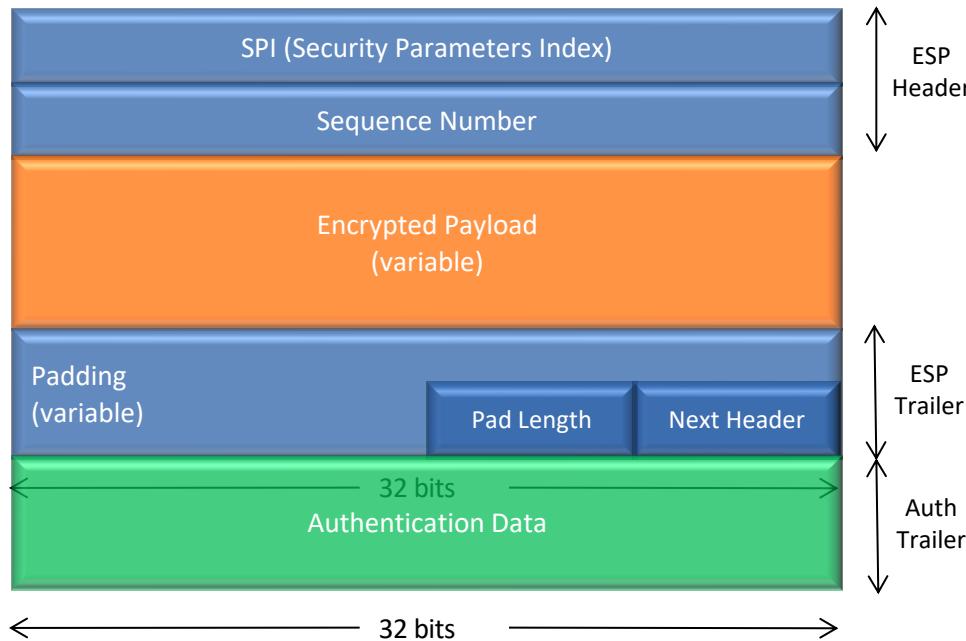
Padding (0-255 bytes): Espacio adicional incluido porque los algoritmos de encriptación basados en bloque pueden requerir espacios diferentes

Pad Length (8 bits): longitud del “Padding”

Next Header (8 bits): guarda el tipo de la siguiente cabecera (IP, TCP, UDP, etc.)

Cabeceras AH y ESP

- Encapsulating Security Payload – ESP
(cifrado + autenticación de dato+integridad):



Cabeceras AH y ESP

- Las distintas opciones al seleccionar tipos de cabeceras:

	AH	ESP (encryption only)	ESP (encryption plus authentication)
Connectionless integrity	✓		✓
Data origin authentication	✓		✓
Rejection of replayed packets	✓	✓	✓
Confidentiality		✓	✓
Limited traffic flow confidentiality		✓	✓

Ejemplo: modo transporte



192.168.1.100



192.168.2.100

- *setkey* es el comando para establecer la comunicación segura. Concretamente, lee las órdenes asociadas al SA o SAD de un fichero cuando **se invoca** con:
 - # setkey -f /etc/ipsec.conf (el fichero)
- Se comprueba la viabilidad de la acción *setkey* por lanzar:
 - # setkey -D
 - # setkey -DP

```

#!/usr/sbin/setkey -f

# Configuración for 192.168.1.100

# Vaciar las SAD y SPD
flush;
spdflush;

# Atención: Emplee estas claves sólo para pruebas
# ¡Debería generar sus propias claves!

```

```

# SAs para AH empleando claves largas de 128 bits
add 192.168.1.100 192.168.2.100 ah 0x200 -A hmac-md5 \
0xc0291ff014dccdd03874d9e8e4cdf3e6;
add 192.168.2.100 192.168.1.100 ah 0x300 -A hmac-md5 \
0x96358c90783bbfa3d7b196ceabe0536b;

```

```

# SAs para ESP empleando claves largas de 192 bits (168 + 24 paridad)
add 192.168.1.100 192.168.2.100 esp 0x201 -E 3des-cbc \
0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831;
add 192.168.2.100 192.168.1.100 esp 0x301 -E 3des-cbc \
0xf6ddb555acfd9d77b03ea3843f2653255afe8eb5573965df;

```

```

# Políticas de seguridad
spdadd 192.168.1.100 192.168.2.100 any -P out ipsec
    esp/transport//require
    ah/transport//require;

```

```

spdadd 192.168.2.100 192.168.1.100 any -P in ipsec
    esp/transport//require
    ah/transport//require;

```

SPI



Se limpia el sistema de SAs y SPs antiguas (el SAD y SPD)

AH



SAs: AH y la clave (en ASCII, "", hexadecimal) para el HMAC

- **-A:** alg. de autenticación
- **-E:** alg. de cifrado
- **-C:** alg. de compresión

ESP



SAs: ESP y la clave para el cifrado

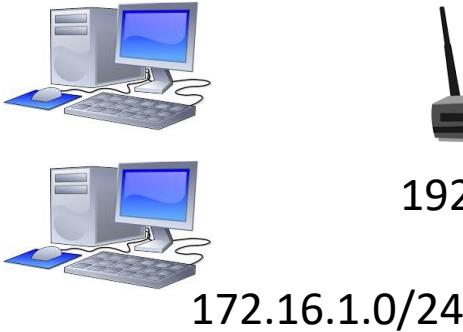
Protocolo y puerto



SPDs: para ambos lados

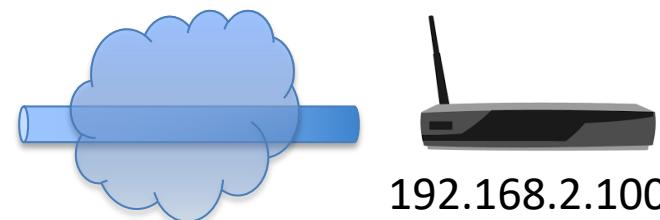
Dirección de la acción de la política

Ejemplo: modo túnel

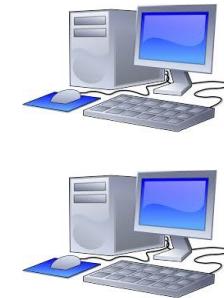


192.168.1.100

172.16.1.0/24



192.168.2.100



172.16.2.0/24

```
#!/usr/sbin/setkey -f

# Vaciar las SAD y SPD
flush;
spdflush;

# SAs para ESP realizando cifrado con claves largas de 192 bit (168 + 24 paridad)
# y autenticación empleando claves largas de 128 bits
add 192.168.1.100 192.168.2.100 esp 0x201 -m tunnel -E 3des-cbc \
0x7aeaca3f87d060a12f4a4487d5a5c3355920fae69a96c831 \
-A hmac-md5 0xc0291ff014dccdd03874d9e8e4cdf3e6;

add 192.168.2.100 192.168.1.100 esp 0x301 -m tunnel -E 3des-cbc \
0xf6ddb555acfd9d77b03ea3843f2653255afe8eb5573965df \
-A hmac-md5 0x96358c90783bbfa3d7b196ceabe0536b;

# Políticas de seguridad
spdadd 172.16.1.0/24 172.16.2.0/24 any -P out ipsec
      esp/tunnel/192.168.1.100-192.168.2.100/require;

spdadd 172.16.2.0/24 172.16.1.0/24 any -P in ipsec
      esp/tunnel/192.168.2.100-192.168.1.100/require;
```

Protocolo Internet Key Exchange (IKE)

- Como se puede observar en la figura anterior, cuando no existe la SA, hay que negociarla.
 - De eso se encarga el protocolo **IKE - Internet Key Exchange**
- IKE se encarga de la:
 - **autenticación de las partes de la comunicación y**
 - **el establecimiento de la clave secreta**
- IKE utiliza:
 - **certificados X.509** para la autenticación, y
 - el algoritmo de **Diffie-Hellman** para establecer la clave secreta
- IKE se basa a su vez en los protocolos:
 - **Oakley** (Key Exchange Protocol)
 - **ISAKMP** (Internet Security Association and Key Management Protocol)

Protocolo Internet Key Exchange (IKE)

Estructura interna de la suite IPSEC:

AH = Authentication Header

API = Application Programming Interface

DOI = Domain of Interpretation

ESP = Encapsulated Security Payload

ISAKMP = Internet Security Association
and Key Management Protocol

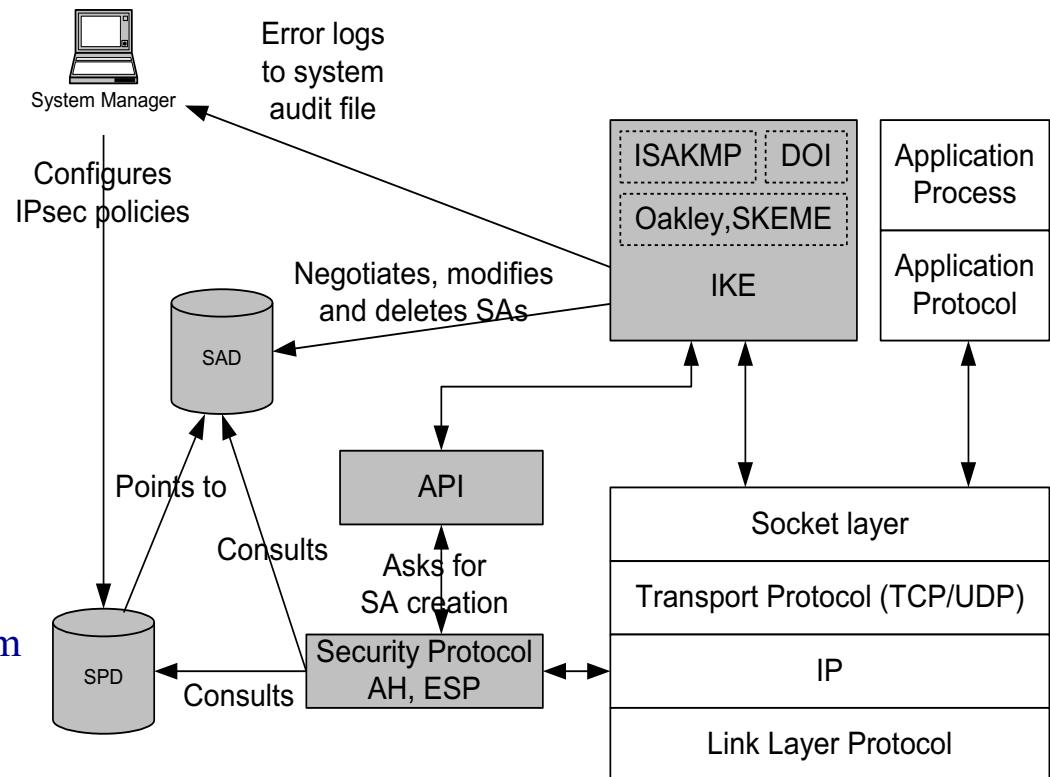
Oakley = Key Exchange Protocol

SA = Security Association

SAD = Security Association Database

SKEME = Secure Key Exchange Mechanism

SPD = Security Policy Database



- Con ISAKMP, IKE funciona en dos fases:
 - Fase 1: autentica cada parte
 - Fase 2: negocia y establece las SAs de IPSec

Protocolo Internet Key Exchange (IKE)

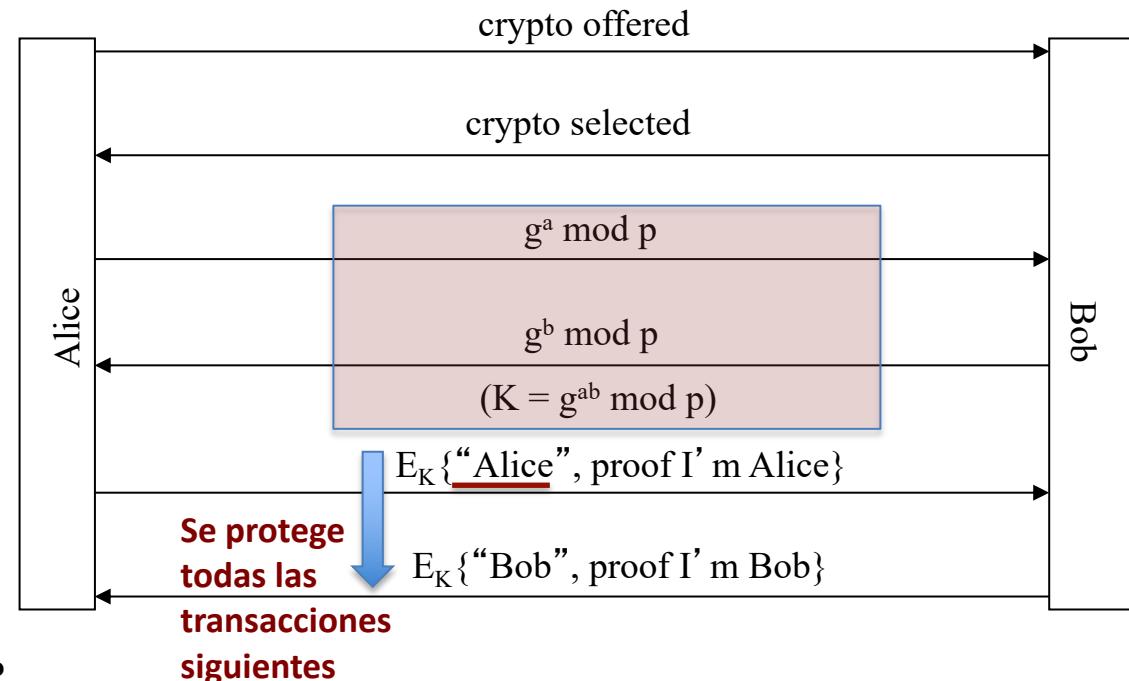
- Fase 1: **autentica cada parte**
 - La autenticación de las partes de la comunicación
 - suele basarse en claves compartidas
 - claves RSA
 - certificados x.509
 - Esta fase soporta dos modos de autenticación: **agresivo y principal**
 - El modo agresivo:
 - proceso simple y sólo usa la mitad de los mensajes para finalizar el proceso rápidamente
 - Sin embargo, no soporta la protección completa de las identidades de cada parte de la comunicación y transmite la identidad del cliente en claro
 - El modo principal:
 - realiza el proceso de autenticación de forma lenta y segura

Protocolo Internet Key Exchange (IKE)

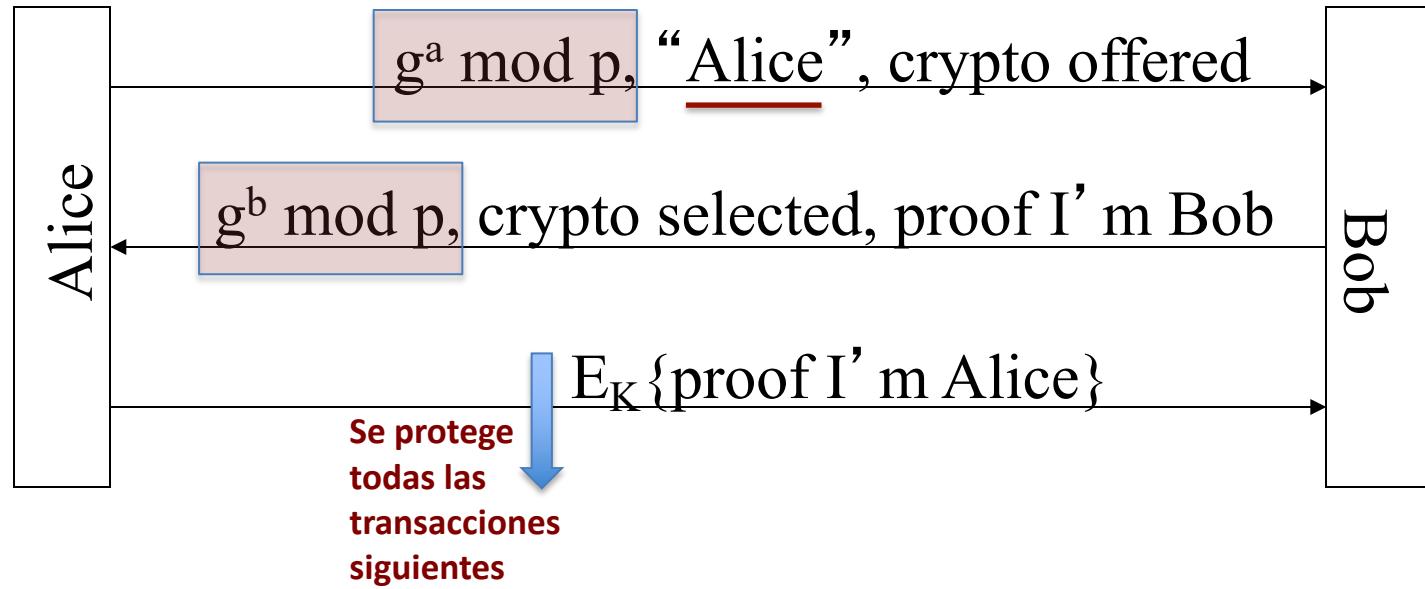
- Fase 2: el nuevo SA ISAKMP es empleado para **negociar** y establecer los SAs de IPSec
 - En esta fase el protocolo IKE intercambia propuestas de SA
 - negocia asociaciones de seguridad basándose en el ISAKMP SA inicial, y
 - establece la clave de sesión
 - Las claves de las SA se derivan de
 - las claves de la primera fase, los nonces y los SPI, o usando un nuevo DH

Protocolo IKE - Fase 1 a Fase 2: Modo Principal

- El modo principal negocia una ISAKMP SA que se usa para crear las SAs de IPSec
- Tres pasos
 - Autenticación
 - Negociación de las SA
 - Diffie-Hellman e intercambio de nonce



Protocolo IKE - Fase 1 a Fase 2: Modo Agresivo

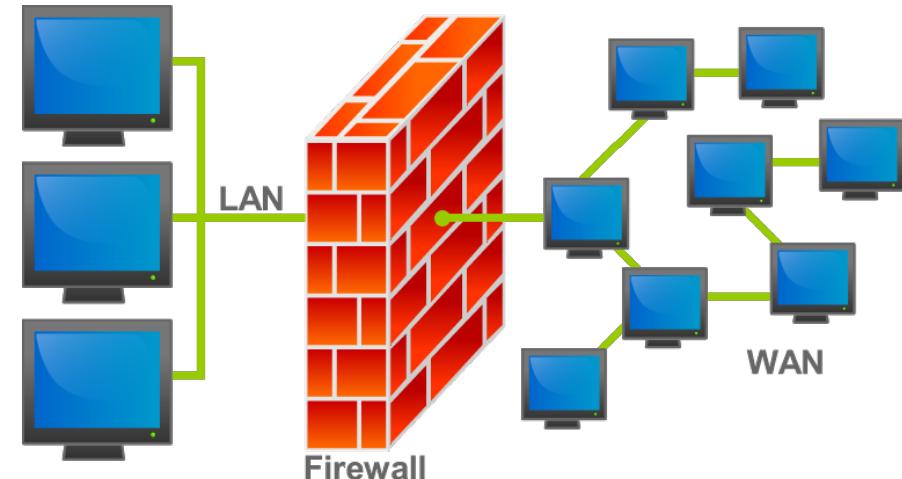


FIREWALLS EN REDES

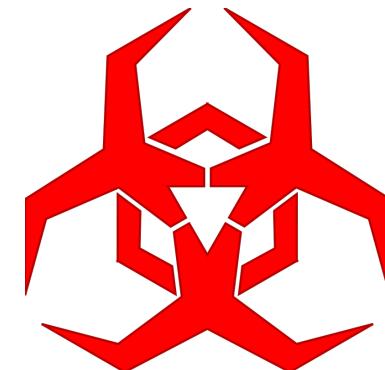
¿Qué es un cortafuego?

Firewalls

- Un cortafuego (firewall) es un sistema (software o hardware) que establece un conjunto de políticas de control

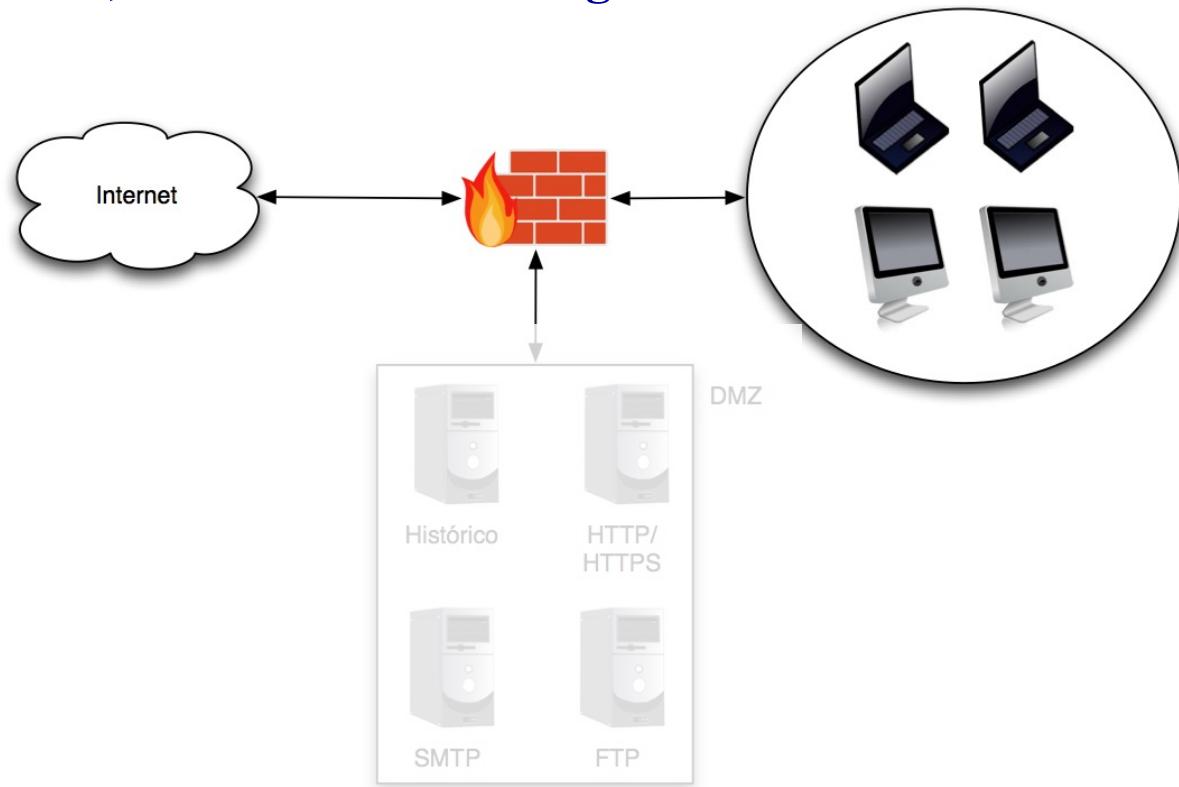


- Considerada la primera herramienta de seguridad para las redes y surge a finales de 1980 (aprox. a principios de los noventa)
 - cuando el Internet comienza a aplicarse a nivel de usuario y cuando surgen los primeros ataques a nivel de red como el *gusano Morris* (por su autor Robert Morris)
 - El malware Morris apareció en 1988 e infectó aprox. 6000 de los 60.000 servidores conectados a la red
 - aunque no era extremadamente peligroso afectó muchas máquinas del mundo



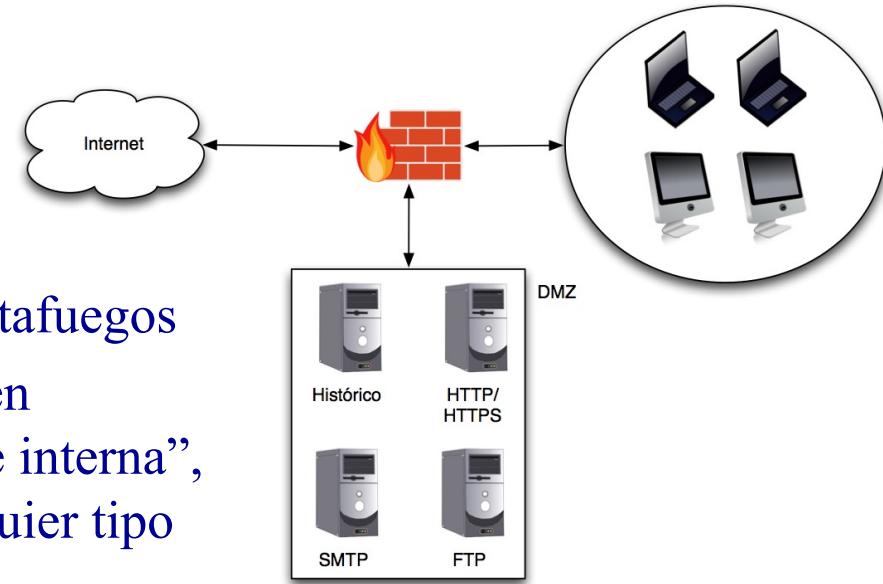
Firewalls

- Un firewall define:
 - El espacio protegido, denominado **perímetro de seguridad**, suele ser propiedad de la misma organización, y
 - la protección se realiza generalmente contra una red externa (ej. el Internet) no confiable, llamada **zona de riesgo**



Firewalls

- **Zonas desmilitarizadas (De-Militarized Zones, DMZ):**
 - Añaden un nivel específico de seguridad en las arquitecturas de cortafuegos
 - Situando una subred DMZ (basado en servidores) entre las redes “externa e interna”, de forma que aísla y/o protege cualquier tipo de acceso a los hosts del sistema
- Se establecen configuraciones específicas de firewall tal que:
 - La conexión de las redes externas al DMZ son permisivas
 - La conexión del DMZ a la intranet son prohibitivas



Tipos de firewalls

- Hay cuatro generaciones de firewalls principales:
 - **Primera generación:**
 - filtrado de paquetes (*packet filter*)
 - **Segunda generación:**
 - cortafuegos de estado (*stateful inspection*)
 - **Tercer generación:**
 - cortafuegos de aplicación (*application filtering*)
 - **Cuarta generación:**
 - cortafuegos de nueva generación (*Next-Generation Firewall, NGFW*)

Tipos de firewalls

- **Filtrado de paquetes (*packet filter*):**

- Definido en 1988 por Digital Equipment Corporation (DEC) para inspeccionar cada paquete entrante/saliente, usando reglas sencillas de filtrado y a parámetros específicos como: IP del emisor, IP del destino, tipo de protocolo, puerto, DNS

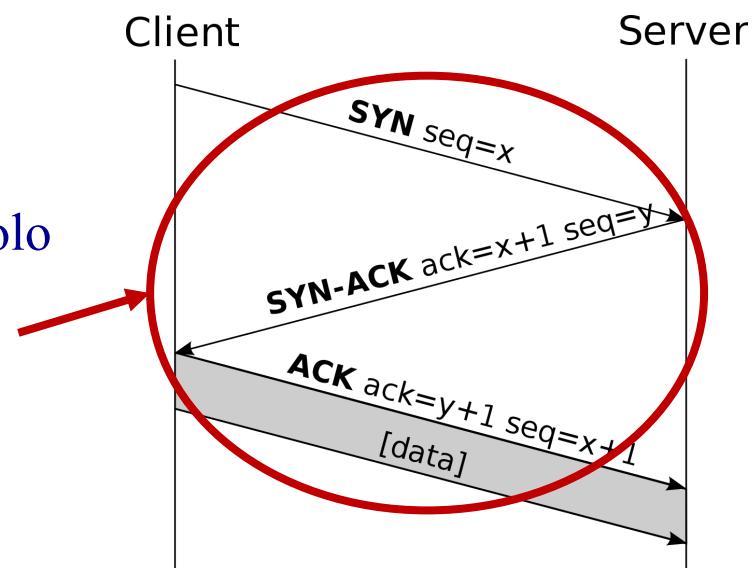
SOLO MIRA EN LA CABECERA DEL
PAQUETE (IPs, puertos, MAC,
protocolo, etc.)

Tipos de firewalls

- **Cortafuegos de estado (*stateful inspection*):**
 - Definido en 1989-1990 por AT&T Bell para hacer lo mismo que los cortafuegos de primera generación, pero considerando el estado de cada paquete los **flags**: SYN, ACK, ...

SOLO MIRA EN LA CABECERA DEL
PAQUETE (IPs, puertos, MAC,
protocolo, etc.) + **FLAGS**

- Los estados ayudan a:
 - Controlar el estado de cada protocolo (ej. conexiones TCP)

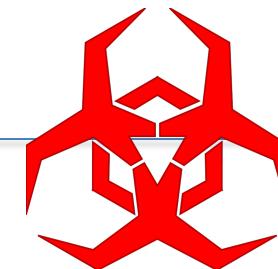


Tipos de firewalls

- **Cortafuegos de aplicación (*application filtering*):**
 - El objetivo era mejorar las capacidades de los cortafuegos de segunda generación y permitir:
 - **DPI “Deep Packet Inspection”** para detectar irregularidades o infección malware en los contenidos de los paquetes
 - Combinar el firewall con otros sistemas de protección integrados como:
 - Antimalware y sistemas de detección y prevención de intrusiones
 - VPN, TLS

SOLO MIRA EN LA CABECERA DEL PAQUETE (IPs, puertos, MAC, protocolo, etc.) + **FLAGS**

DPI



Tipos de firewalls

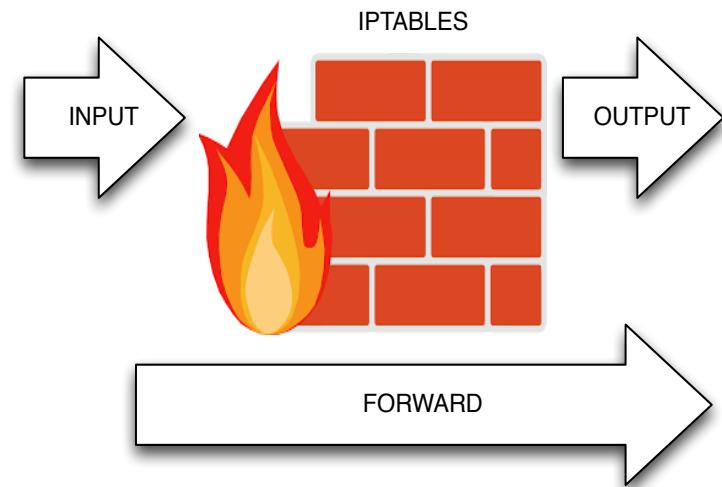
- **Cortafuegos de nueva generación (*Next-Generation Firewall, NGFW*):**
 - El objetivo era mejorar para incorporar nuevos servicios:
 - Gestión y monitorización de usuarios y aplicaciones (email, app de descarga, compartición de recursos y colaborativos,...)
 - Eliminación de apps no deseadas para reducir la superficie de ataque (especialmente los APTs)
 - Validación de aplicaciones, el aislamiento de los datos, el control sobre las aplicaciones no deseadas, etc.
 - Protección a los sistemas de red en la nube
 - Control desde plataformas móviles
 - Etc.

IPTABLES: sintaxis y políticas

- IPtables ilustra la siguiente sintaxis de comandos:

```
[root@localhost alumno]# #iptables -A <chain> -j <target>
```

- <chain>: define una cadena de reglas que pueden ser del tipo INPUT, OUTPUT y FORWARD



- -A <chain>: agrega la regla a la cadena
 - -D <chain>: elimina la regla
 - -I <chain>: inserta una nueva regla en una posición determinada
- -j <target>: especifica el fin de la regla; es decir, qué hacer si un paquete coincide con la regla, como, por ejemplo: ACCEPT y DROP

IPTABLES: sintaxis y políticas

- **Eliminar cualquier tráfico telnet entrante:**
iptables -I INPUT -p tcp --dport 23 -j DROP
- **Eliminar cualquier tráfico web saliente:**
iptables -I OUTPUT -p tcp --dport 80 -j DROP
- **Eliminar cualquier tráfico saliente a la IP 192.168.0.1**
iptables -I OUTPUT -p tcp --dest 192.168.0.1 -j DROP
- **Permitir cualquier tráfico web entrante:**
iptables -I INPUT -p tcp --dport 80 -j ACCEPT
- **Permitir tráfico entrante al puerto 25 (del servidor SMTP):**
iptables -A INPUT -s 0.0.0.0/0 -p tcp --dport 25 -j ACCEPT
- **Permitir tráfico pop3 entrante:**
iptables -A INPUT -s 0.0.0.0/0 -p tcp --dport 110 -j ACCEPT

IPTABLES: sintaxis y políticas

- **Permitir tráfico HTTPS (443) entrante desde la IP 11.3.2.1**
iptables -I INPUT -s 11.3.2.1 -p tcp --dport 443 -j ACCEPT
- **Denegar el tráfico saliente a la red 192.2.4.0-192.2.4.255:**
iptables -I OUTPUT -d 192.2.4.6.0/24 -j DROP
- **Bloquear cualquier tráfico saliente de un particular dominio o host:**
 - 1) host -t a elpais.es ➔ elpais.es tiene IP:78.120.152.203
 - 2) iptables -A OUTPUT -d 78.120.152.203 -j DROP

IPTABLES: flush y políticas por defecto

- Política establecida por defecto:
 - Para cada cadena <chain> se define una política inicial que puede ser ACCEPT o DROP, y para ello se usa la opción –P

```
### FLUSH de reglas
```

```
iptables -F  
iptables -X  
iptables -Z  
iptables -t nat -F
```

```
### Políticas por defecto
```

```
iptables -P INPUT ACCEPT  
iptables -P OUTPUT ACCEPT  
iptables -P FORWARD ACCEPT  
iptables -t nat -P PREROUTING ACCEPT  
iptables -t nat -P POSTROUTING ACCEPT
```

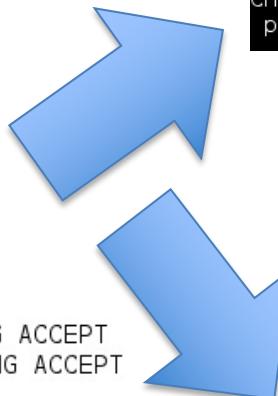
IPTABLES: flush y políticas por defecto

FLUSH de reglas

```
iptables -F  
iptables -X  
iptables -Z  
iptables -t nat -F
```

Políticas por defecto

```
iptables -P INPUT ACCEPT  
iptables -P OUTPUT ACCEPT  
iptables -P FORWARD ACCEPT  
iptables -t nat -P PREROUTING ACCEPT  
iptables -t nat -P POSTROUTING ACCEPT
```



```
[alumno@router ~]$ su  
Contraseña:  
[root@router alumno]# iptables -nvL  
Chain INPUT (policy ACCEPT 20845 packets, 27M bytes)  
pkts bytes target prot opt in out source destination  
  
Chain FORWARD (policy ACCEPT 84 packets, 5676 bytes)  
pkts bytes target prot opt in out source destination  
  
Chain OUTPUT (policy ACCEPT 14649 packets, 1370K bytes)  
pkts bytes target prot opt in out source destination
```

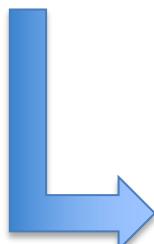
```
[root@router alumno]# iptables -t nat -nvL  
Chain PREROUTING (policy ACCEPT 3 packets, 718 bytes)  
pkts bytes target prot opt in out source destination  
  
Chain INPUT (policy ACCEPT 3 packets, 718 bytes)  
pkts bytes target prot opt in out source destination  
  
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)  
pkts bytes target prot opt in out source destination  
  
Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)  
pkts bytes target prot opt in out source destination
```

IPTABLES: INPUT, OUTPUT, FORWARD - RESTRICTIVO

- **Ejemplo 1: NO permitir tráfico entrante y saliente, y para todas las redes (incluyendo al Router)**

```
[root@router alumno]# iptables -P INPUT DROP
[root@router alumno]# iptables -P OUTPUT DROP
[root@router alumno]# ping 192.168.0.15
PING 192.168.0.15 (192.168.0.15) 56(84) bytes of data.
^C
--- 192.168.0.15 ping statistics ---
0 packets transmitted, 0 received

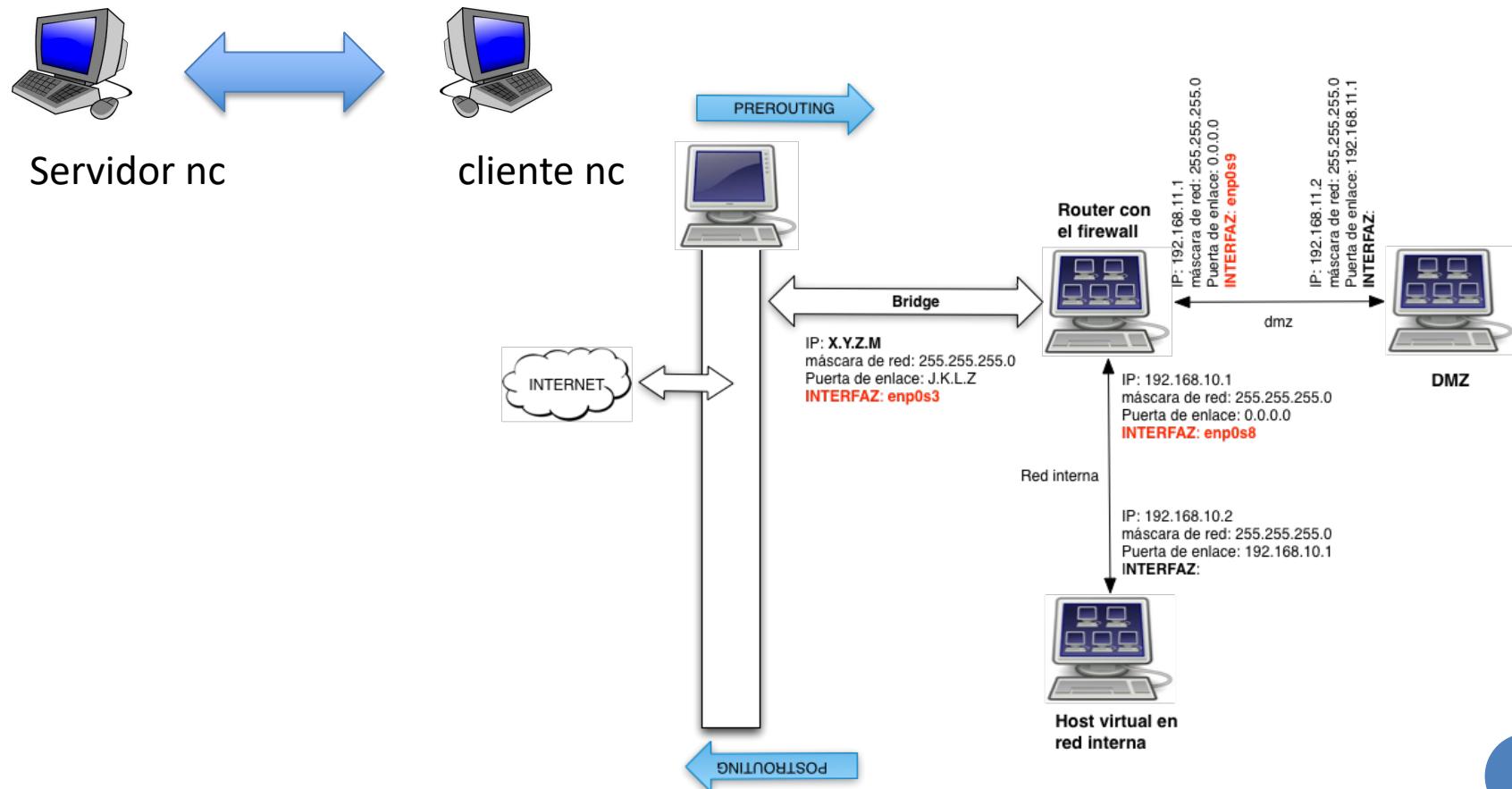
[root@router alumno]# ping 192.168.11.1
PING 192.168.11.1 (192.168.11.1) 56(84) bytes of data.
^C
--- 192.168.11.1 ping statistics ---
0 packets transmitted, 0 received
```



```
[root@router sysctl.d]# iptables -nL
Chain INPUT (policy DROP)
target     prot opt source          destination
Chain FORWARD (policy ACCEPT)
target     prot opt source          destination
Chain OUTPUT (policy DROP)
target     prot opt source          destination
```

IPTABLES: INPUT, OUTPUT, FORWARD

- Ejemplo 2: crear un cliente-servidor netcat en las máquinas instaladas en la red DMZ y la red interna para que se comuniquen, sólo y únicamente, por el puerto 55555



IPTABLES: INPUT, OUTPUT, FORWARD

- **Ejemplo 2:** crear un cliente-servidor netcat en las máquinas instaladas en la red DMZ y la red interna para que se comuniquen, sólo y únicamente, por el puerto 55555

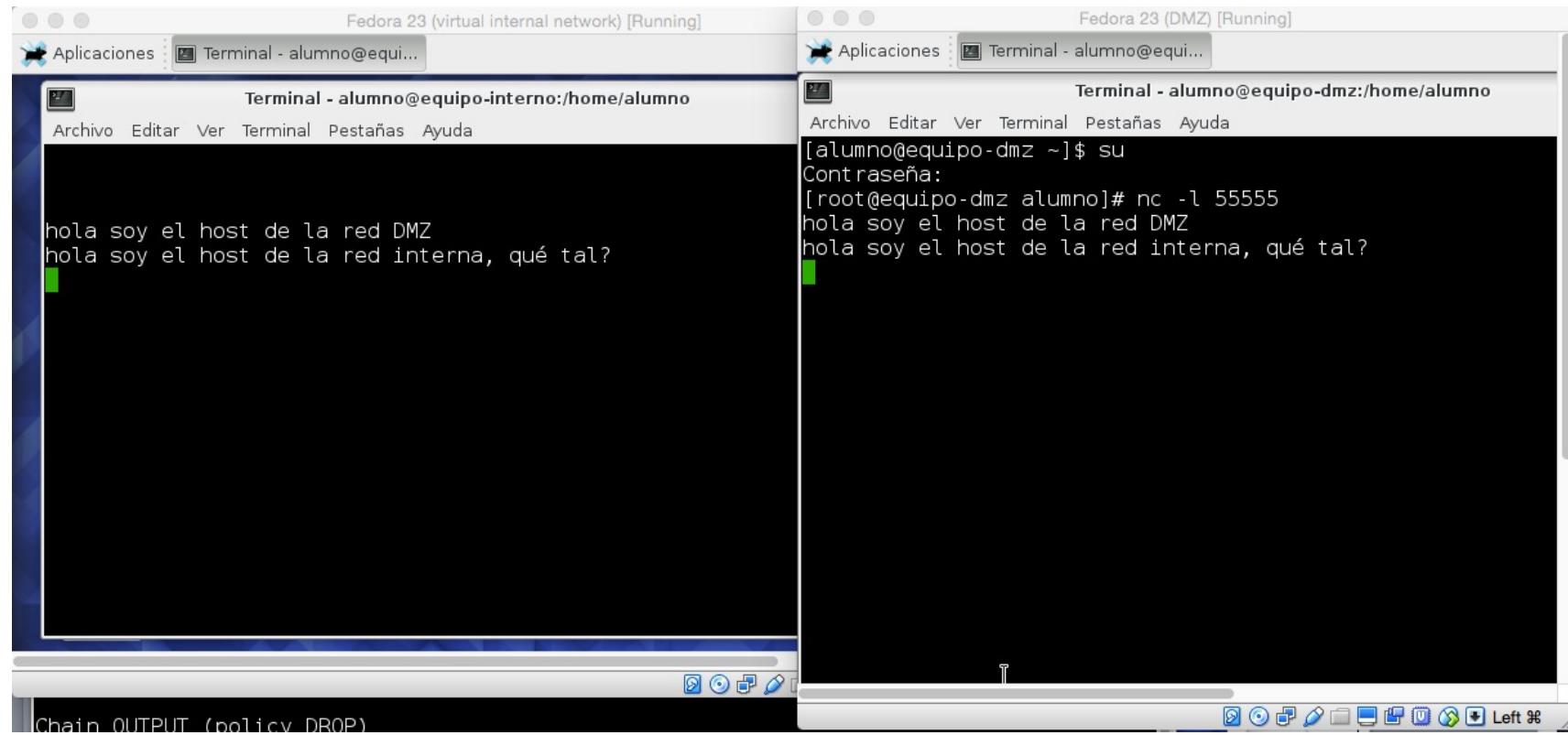
```
### ASEGURAR EL FORWARDING
sysctl -w net.ipv4.ip_forward=1
echo "net.ipv4.ip_forward = 1" > /etc/sysctl.d/ip_forwarding.conf

### FLUSH de reglas
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

### Políticas por defecto
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD DROP
iptables -P FORWARD DROP
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT

## PROPORCIONAR LA COMUNICACIÓN ENTRE LAN-LAN PERO SOLO POR EL PUERTO 55555
iptables -A FORWARD -i enp0s9 -o enp0s8 -p tcp --dport 55555 -j ACCEPT
iptables -A FORWARD -i enp0s8 -o enp0s9 -p tcp --sport 55555 -j ACCEPT
```

IPTABLES: INPUT, OUTPUT, FORWARD



Cliente nc– Router

Servidor nc – Red DMZ

IPTABLES: PREROUTING Y POSTROUTING

- La comunicación hacia o desde redes públicas requiere:

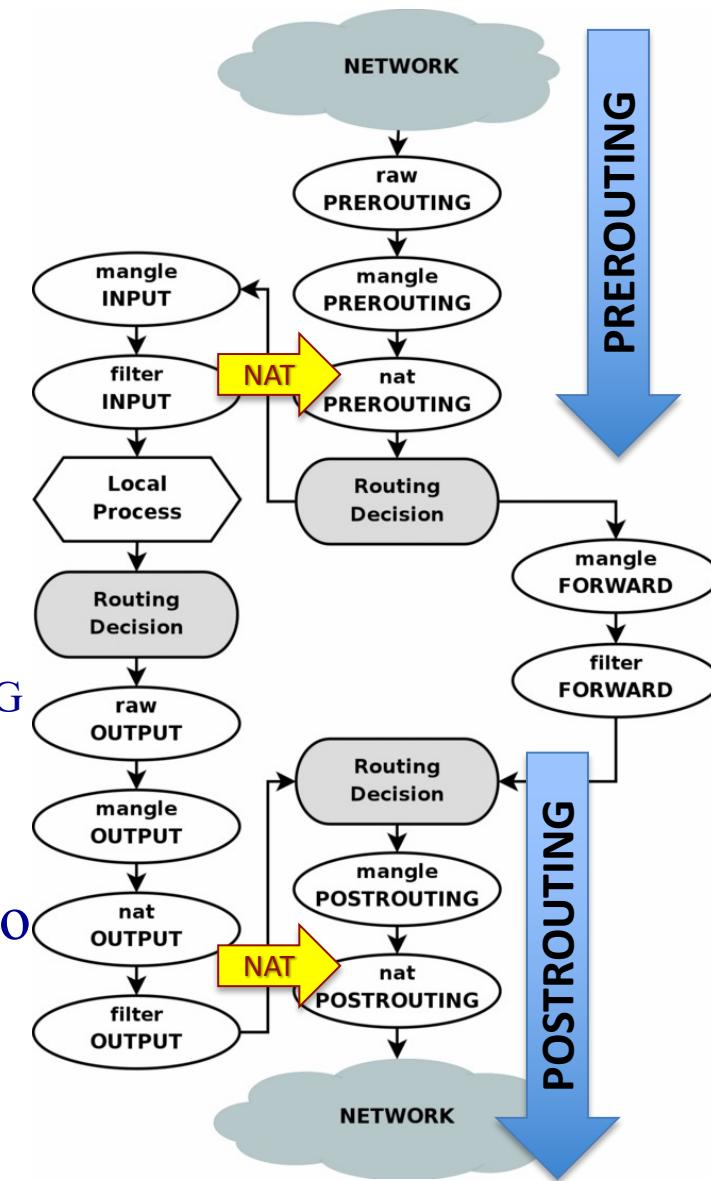
– PREROUTING:

- EXTERIOR → INTERIOR
 - NAT: exterior → interior
- PREROUTING → INPUT / FORWARD

– POSTROUTING

- INTERIOR → EXTERIOR
 - NAT: interior → exterior
(requiere ENMASCARAMIENTO)
- OUTPUT / FORWARD → POSTROUTING

- Como se aprecia en la figura, al trabajar con paquetes que vienen de o van hacia redes externas, hay que aplicar NAT (Network Address Translation)



IPTABLES: PREROUTING Y POSTROUTING

- La comunicación hacia o desde redes públicas requiere:
 - **POSTROUTING:** INTERIOR → EXTERIOR

- Proceso de **enmascaramiento** para poder trabajar con el NAT

```
[root@router alumno]# iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE
```



- El orden de las reglas importa:
 - Primero **FORWARD/OUTPUT** y
 - Despues el **POSTROUTING**

- **PREROUTING:** EXTERIOR → INTERIOR

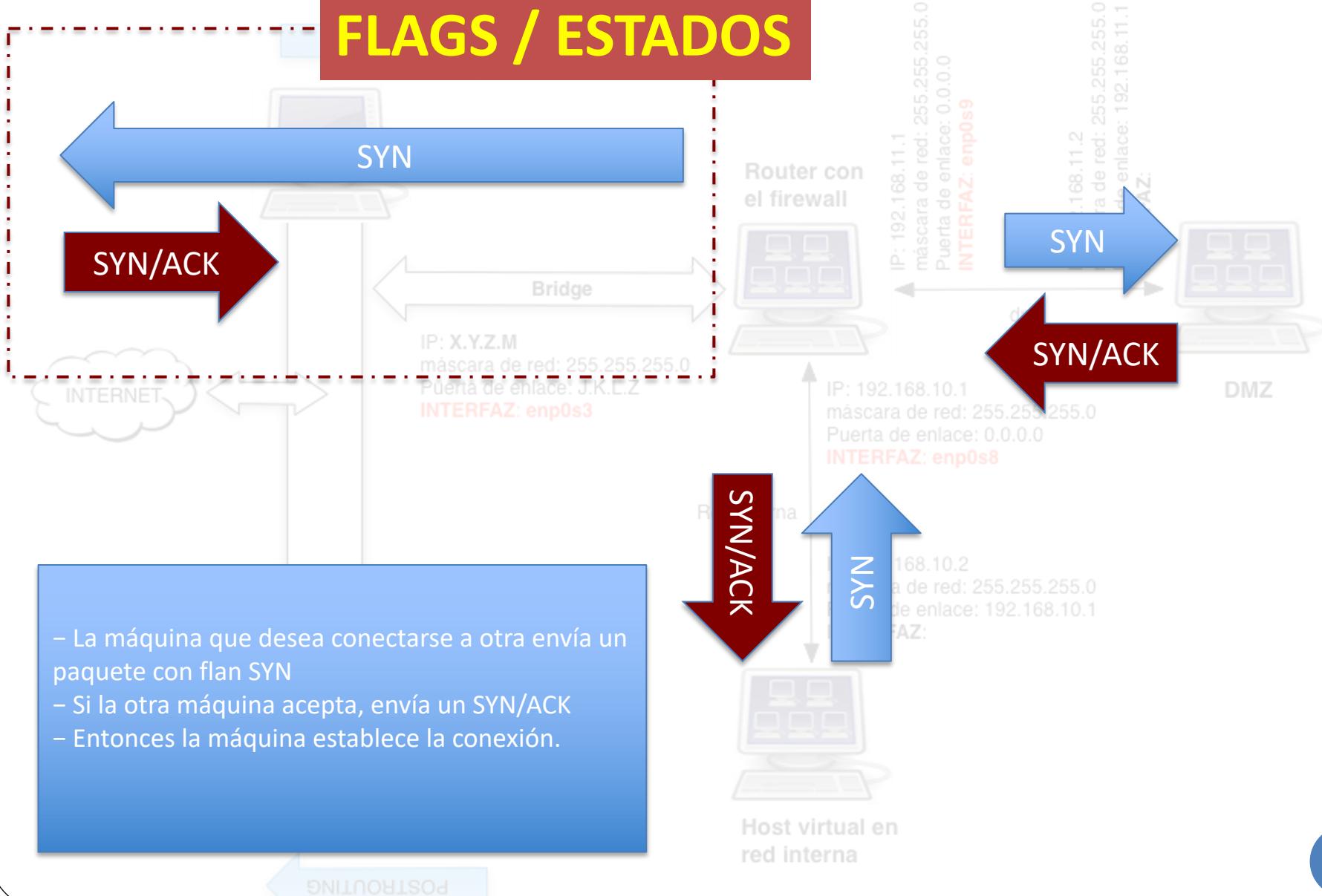


```
[root@router alumno]# iptables -t nat -A PREROUTING -i enp0s3 -d X.Y.Z.M -p tcp  
--dport 443-j DNAT --to 192.168.11.2:443
```

- El orden de las reglas importa:
 - Primero el **PREROUTING** y
 - Despues **FORWARD/INPUT**

Three-way-handshake en TCP/IP y problema a controlar

FLAGS / ESTADOS

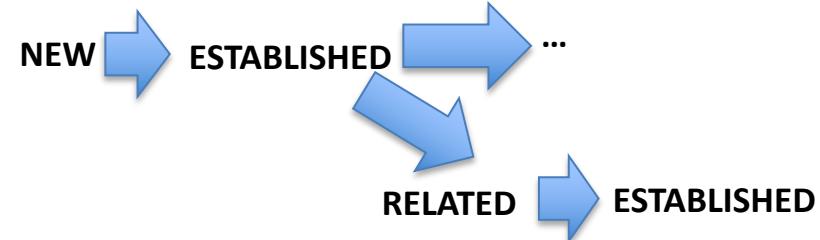


Three-way-handshake en TCP/IP y problema a controlar



- NEW: se ha establecido una nueva conexión y se requiere comunicación bidireccional (ej. SYN -- SYN/ACK)
- ESTABLISHED: el paquete está asociado con una conexión ya establecida pero se requiere que haya paquetes en ambas direcciones (ej. SYN -- SYN/ACK)
- RELATED: el paquete está comenzando una nueva conexión pero está asociada a una conexión ya existente, como puede ser una comunicación FTP

NEW → ESTABLISHED



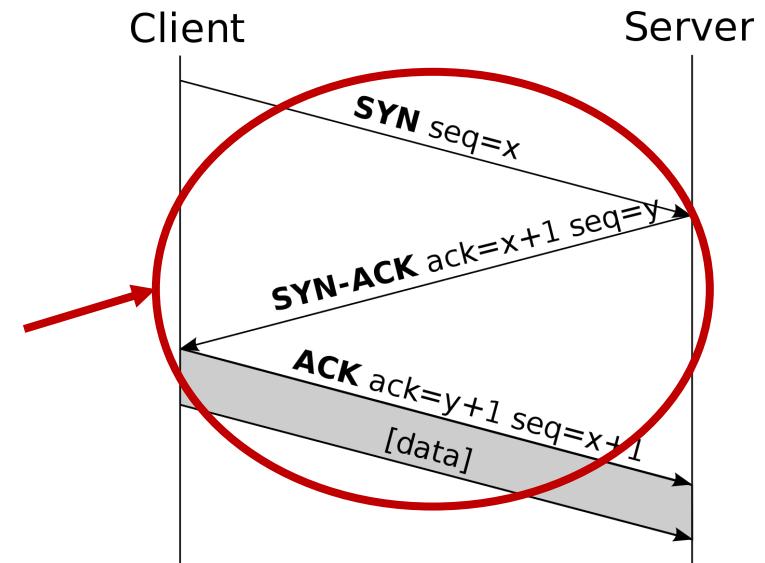
Three-way-handshake en TCP/IP y problema a controlar

- Un ejemplo: el 3-way-handshake de TCP
 - El firewall debe recordar estos estados y controlar en este caso el:

- SYN
- SYN/ACK
- ACK

Tal que SYN sería el NEW,
y el SYN/ACK y ACK serían
los RELATED

FLAGS / ESTADOS



Three-way-handshake en TCP/IP y problema a controlar

FLAGS / ESTADOS



Políticas por defecto

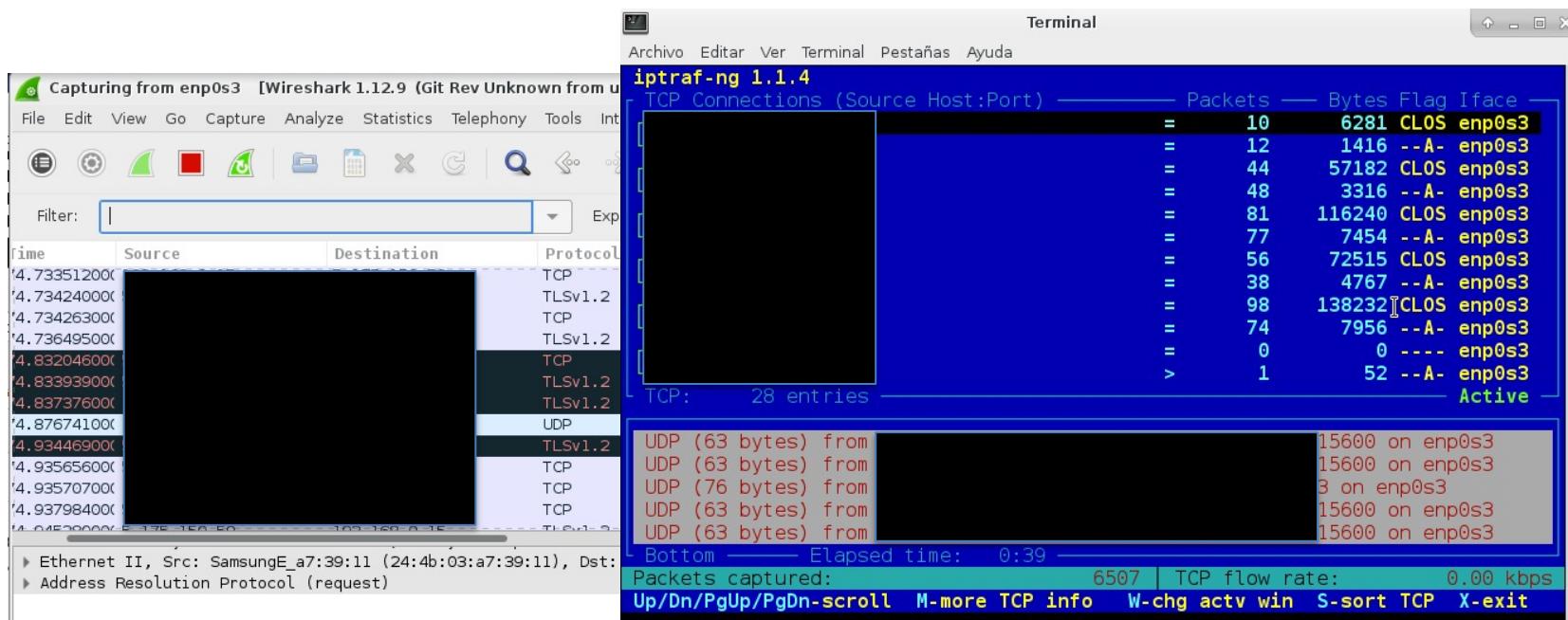
```
iptables -P INPUT ACCEPT  
iptables -P OUTPUT ACCEPT  
iptables -P FORWARD DROP  
iptables -t nat -P PREROUTING ACCEPT  
iptables -t nat -P POSTROUTING ACCEPT
```

Permitiendo comunicación con la red externa

```
iptables -A FORWARD -i enp0s9 -o enp0s3 -j ACCEPT  
iptables -A FORWARD -i enp0s3 -o enp0s9 -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT  
iptables -A FORWARD -i enp0s8 -o enp0s3 -j ACCEPT  
iptables -A FORWARD -i enp0s3 -o enp0s8 -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT  
iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE
```

```
iptables -nvL
```

Herramientas de monitorización



```
0000 ff ff ff ff ff ff [root@router pub]# tcpdump -D
0010 08 00 06 04 00 00 1.enp0s3 [Up, Running]
0020 00 00 00 00 00 00 2.enp0s8 [Up, Running]
0030 00 00 00 00 00 00 3.enp0s9 [Up, Running]
4.any (Pseudo-device that captures on all interfaces) [Up, Running]
5.lo [Up, Running, Loopback]
6.bluetooth-monitor (Bluetooth Linux Monitor)
7.usbmon1 (USB bus number 1)
[root@router pub]# tcpdump -i 3
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp0s9, link-type EN10MB (Ethernet), capture size 262144 bytes
18:39:45.534659 IP [REDACTED] 934 > mad06s09-in-f3.1e100.net.https: Flags
., seq 1367477101:1367477132, ack 1352877774, win 507, options [nop,nop,TS val
38372341 ecr 3259937716], length 31
18:39:45.534906 IP [REDACTED] 34 > mad06s09-in-f3.1e100.net.https: Flags
., seq 31, ack 1, win 507, options [nop,nop,TS val 38372341 ecr 3259937716],
length 0
18:39:45.534938 IP [REDACTED] > mad06s09-in-f3.1e100.net.https: Flags
```

SEGURIDAD EN LA CAPA DE ACCESO A RED: EL CASO DE LAS REDES INALÁMBRICAS

Consideraciones generales

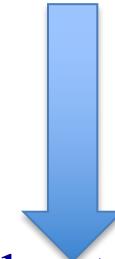
- Existe una amplia variedad de tecnologías y tipos de redes inalámbricas, entre las que destacan **Wi-Fi**, Bluetooth, WiMAX, Zigbee, etc.
 - Los requisitos de seguridad de estas redes son los mismos que en el caso de las redes cableadas
- Sin embargo, hay algunas amenazas de seguridad que aumentan cuando se consideran las redes inalámbricas
 - y además, hay otras amenazas que son propias/específicas de estos entornos
- La fuente de riesgo más significativa en las redes inalámbricas es el **medio de comunicación** subyacente
 - pero también hay riesgos de seguridad en los propios **protocolos inalámbricos** cuando no se diseñan apropiadamente

Consideraciones generales

- A grandes rasgos, el entorno inalámbrico tiene tres puntos de ataque:
 - cliente (o estación),
 - punto de acceso (AP) y
 - medio de transmisión
- Las posibles amenazas generales son:
 - Robo de identidad (o sea, de la dirección MAC del dispositivo)
 - Man-in-the-middle
 - Denegación de servicio
 - Inyección en la red
 -

Consideraciones generales

- Las **medidas de seguridad** se pueden clasificar de acuerdo a:
 - la transmisión inalámbrica
 - los puntos de acceso (AP)
 - los elementos de interconexión (ej., routers)
- Medidas de seguridad relacionadas con las **transmisiones inalámbricas**:
 - las principales amenazas son:
 - la copia de mensajes – *sniffing / eavesdropping*
 - la alteración – *tampering / manipulación*
 - la inserción de mensajes – *generación / falsificación*
 - la interrupción de los mismos – *denegación de servicio*
 - las contramedidas son:
 - **técnicas de ocultación o engaño** (ej. aleatorización en el routing)
 - **cifrado** (ej. AES-CBC)
 - **autenticación** (ej. MAC)
 - **métodos contra DoS**, ej. uso de nonce ☺



Consideraciones generales

- Las **medidas de seguridad** se pueden clasificar de acuerdo a:
 - la transmisión inalámbrica
 - **el puntos de acceso (AP)**
 - los elementos de interconexión (ej., routers)
- Medidas de seguridad relacionadas con el **AP**:
 - la mayor amenaza que involucra al AP es el acceso no autorizado a la red
 - la forma de evitarlo es usando **mecanismos de autenticación** para los dispositivos que se quieren conectar a la red



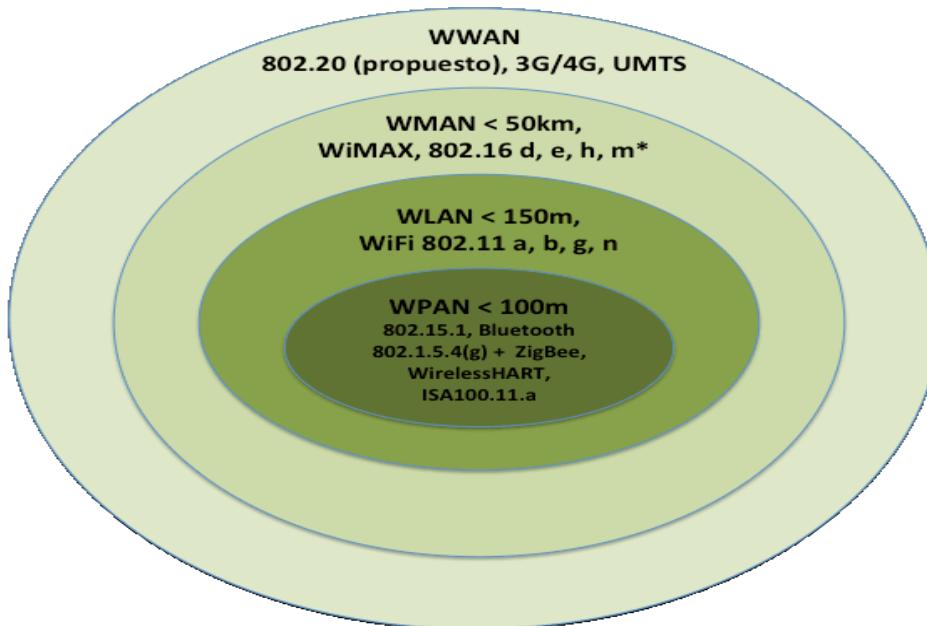
Consideraciones generales

- Las **medidas de seguridad** se pueden clasificar de acuerdo a:
 - la transmisión inalámbrica
 - el puntos de acceso (AP)
 - **los elementos de interconexión (ej., routers)**
- Medidas de seguridad relacionadas con **los elementos de la red**:
 - **cifrado**
 - integrado en los routers inalámbricos para el tráfico entre routers
 - **deshabilitar el broadcast de identificación**
 - sólo los dispositivos autorizados podrán conocer la identidad de los routers
 - **dejar que sólo equipos específicos se conecten** a la red
 - sólo direcciones cuyas direcciones MAC sean conocidas
 - **cambiar el identificador (el SID)**
 - sobre todo el por defecto que el router trae de fábrica
 - **cambiar el password**
 - El por defecto o el preestablecido para la administración del router
 - Hacer frecuente *rekeying*



Redes inalámbricas IEEE 802.11

- IEEE 802 es un comité que ha desarrollado estándares para diferentes tipos de redes LAN y WAN
- IEEE 802.11 es un capítulo de ese comité, y su objetivo es el desarrollo de un protocolo y las especificaciones de transmisión para LANs inalámbricas



Redes inalámbricas IEEE 802.11

- Diferencias entre una red cableada y una red inalámbrica:

	Cableada	Inalámbrica
Ventajas	Robustez Ancho de banda Equipos baratos Fiables al contexto (ruido, vibración, interferencias u obstáculos)	Rapidez y bajo coste de instalación Bajo coste de mantenimiento Movilidad Conexión para múltiples usuarios/dispositivos
Inconvenientes	Coste de mantenimiento Vulnerabilidad a amenazas físicas (principalmente) Dificultad para control en local	Vulnerables a múltiples tipos de amenazas No fiable para contextos inestables, ruidosos o con altas interferencias Coexistencia para interactuar varias redes inalámbricas

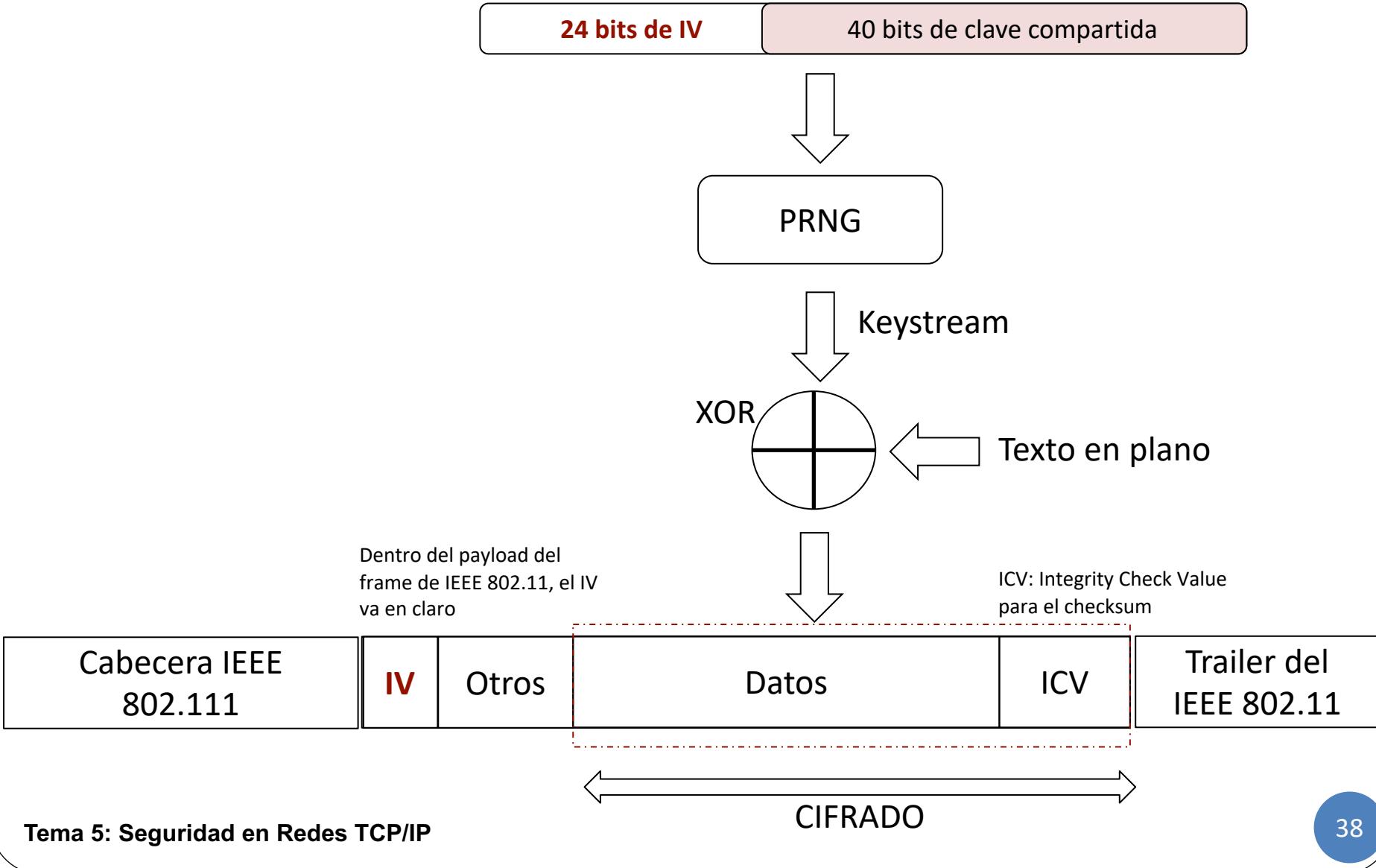
Redes inalámbricas IEEE 802.11

- En lo que a **seguridad** se refiere, hay dos características importantes que distinguen las LAN cableadas y las inalámbricas:
 - en una LAN cableada hay una forma intrínseca de **autenticación** de las estaciones porque están directamente interconectadas a esa red
 - una LAN cableada proporciona un cierto grado de **confidencialidad de los datos** porque la recepción de los datos está limitada a las estaciones conectadas a esa red
- Estas diferencias ponen de manifiesto la necesidad de servicios y mecanismos de seguridad más robustos en las LAN inalámbricas
 - La especificación original de IEEE 802.11 requería, por tanto, un conjunto de características de seguridad:
 - **Privacidad de los datos (confidencialidad) y autenticación**, por lo que se definió el protocolo **WEP (Wired Equivalent Privacy)**

IEEE 802.11: WEP (Wired Equivalent Privacy)

- El protocolo WEP se especificó con el objetivo de proporcionar unos **niveles de seguridad y confidencialidad comparables al de las LAN cableadas**
 - limitado a la comunicación entre la estación y el punto de acceso
- Se basa:
 - en la especificación de una clave de 64 bits que comparten los dispositivos de la red, y de esos 64 bits:
 - 40 corresponden a la clave secreta, y
 - **24 bits para el vector de inicialización (IV)**
 - Además, WEP se basa en el algoritmo de cifrado RC4 (algoritmo en flujo)
- WEP tiene varias vulnerabilidades:
 - **uso de IV débiles**, que posibilitan que, a partir de un número de paquetes cifrados, recuperar la clave secreta
 - Debido a la reutilización de IVs y a la corta longitud del IV

Redes inalámbricas IEEE 802.11



IEEE 802.11i: WPA (Wi-Fi Protected Access)

- Ante esa debilidad, IEEE introdujo una serie de mejoras recogidas e integradas dentro de la versión **IEEE 802.11i**
 - y, por tanto, dentro del protocolo **WPA (Wi-Fi Protected Access)**
- Las mejoras de WPA son:
 - El protocolo **TKIP (Temporal Key Integrity Protocol)**
 - Inicialmente se basaba de RC4, pero usando IVs de 48 bits (el doble de los 24 bits del IV de WEP)
 - Como la mejora de TIK seguía proporcionando los mismos problemas de seguridad, se propuso más tarde el uso del **algoritmo de cifrado AES**
 - También propuso la adopción del protocolo de autenticación **802.1X** (desarrollado inicialmente para LAN cableadas), conocido como **EAP (Extensible Authentication Protocol)**

IEEE 802.11i: WPA (Wi-Fi Protected Access)

- 802.11i asegura tres principales servicios de seguridad:
 - **Autenticación**
 - Autenticación entre un cliente (la estación) y el servidor de autenticación (AS)
 - El AS proporciona autenticación mutua y generación de claves temporales para la confidencialidad e integridad
 - Las claves temporales se usan entre el cliente y el AP
 - **Control de acceso - basado en “puertos”**
 - Controla el acceso de acuerdo a puertos
 - *“Hasta que el cliente no se autentique y no tenga las claves, no podrá acceder al sistema”*, por lo que el AP bloquea todos los puertos excepto el requerido para el IEEE 802.1X / EAP
 - **Confidencialidad con integridad de mensaje**
 - Una vez autenticado el cliente y generados las claves, como se indica arriba, el cliente y el AP pueden intercambiar datos de manera segura
 - Los datos se cifran con TKIP / AES y aplica una función MAC que asegura que los datos no han sido alterados

IEEE 802.11i: WPA (Wi-Fi Protected Access)

- Concretamente:
 - Autenticación
 - A través de un servidor de autenticación en el AP o en un servicio externo, ej. RADIUS
 - Aplica el protocolo EAP para la autenticación
 - MAC/MIC (Message Integrity Code) usando HMAC con SHA-1 o MD5, TKIP-MIC, etc.
 - Control de acceso - basado en “puertos”
 - Controla el acceso de acuerdo a puertos
 - Confidencialidad con integridad de mensaje
 - Aplica TKIP-RC4, CCM (AES-CTR)
 - Generación de clave
 - Aplica HMAC-SHA-1

IEEE 802.11i: WPA (Wi-Fi Protected Access)

- Las operaciones de seguridad (autenticación, control de acceso y confidencialidad) en el 802.11i se distribuyen entre 5 fases distintas:

1. Descubrimiento

- el AP utiliza mensajes llamados *beacons* y *probe responses* para anunciar su política de seguridad
- la estación los utiliza para identificar al AP y se asocia con él seleccionando un *cipher suite* y un mecanismo de autenticación

2. Autenticación

- la estación y el servidor de autenticación (AS) verifican la identidad del otro
- hasta que la autenticación no ha finalizado, el AP bloquea cualquier tráfico entre la estación y el AS, salvo que el tráfico esté relacionado con el propio proceso de autenticación

IEEE 802.11i: WPA (Wi-Fi Protected Access)

3. Generación y distribución de claves

- el AS y la estación realizan diferentes operaciones para generar claves criptográficas, las cuales se almacenan en el AP y la estación
- Esto quiere decir que la comunicación segura se realiza entre el AP y la estación:
 - Estación → AP → Estación

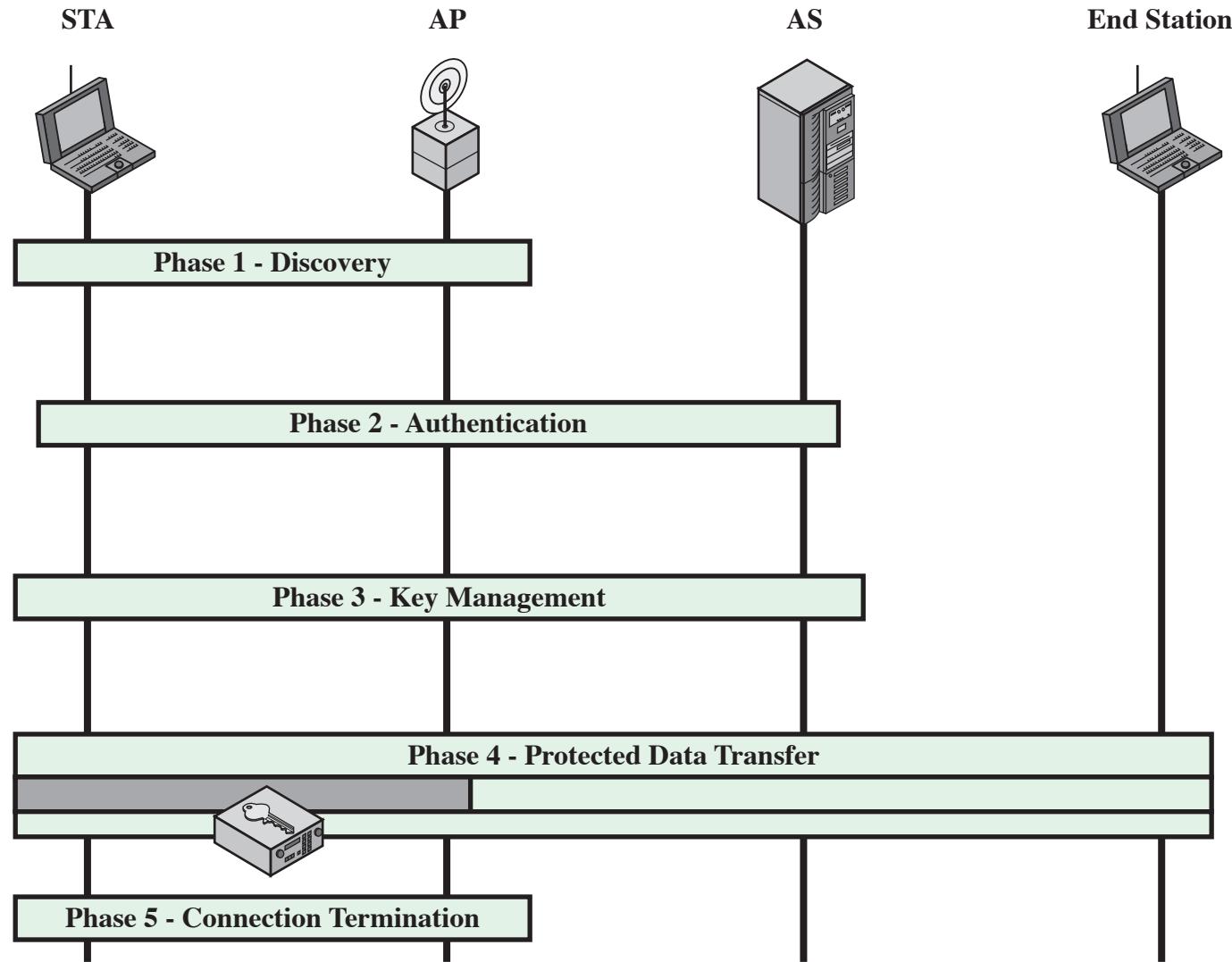
4. Transferencia segura de datos

- Como se indica arriba, la estación origen y la estación final intercambian datos a través del AP, pero la transferencia sólo se realiza de forma segura (cifrada) entre la estación de origen y el AP
 - Estación → AP → Estación

5. Finalización de la conexión

- Entre la estación y el AP

IEEE 802.11i: WPA (Wi-Fi Protected Access)



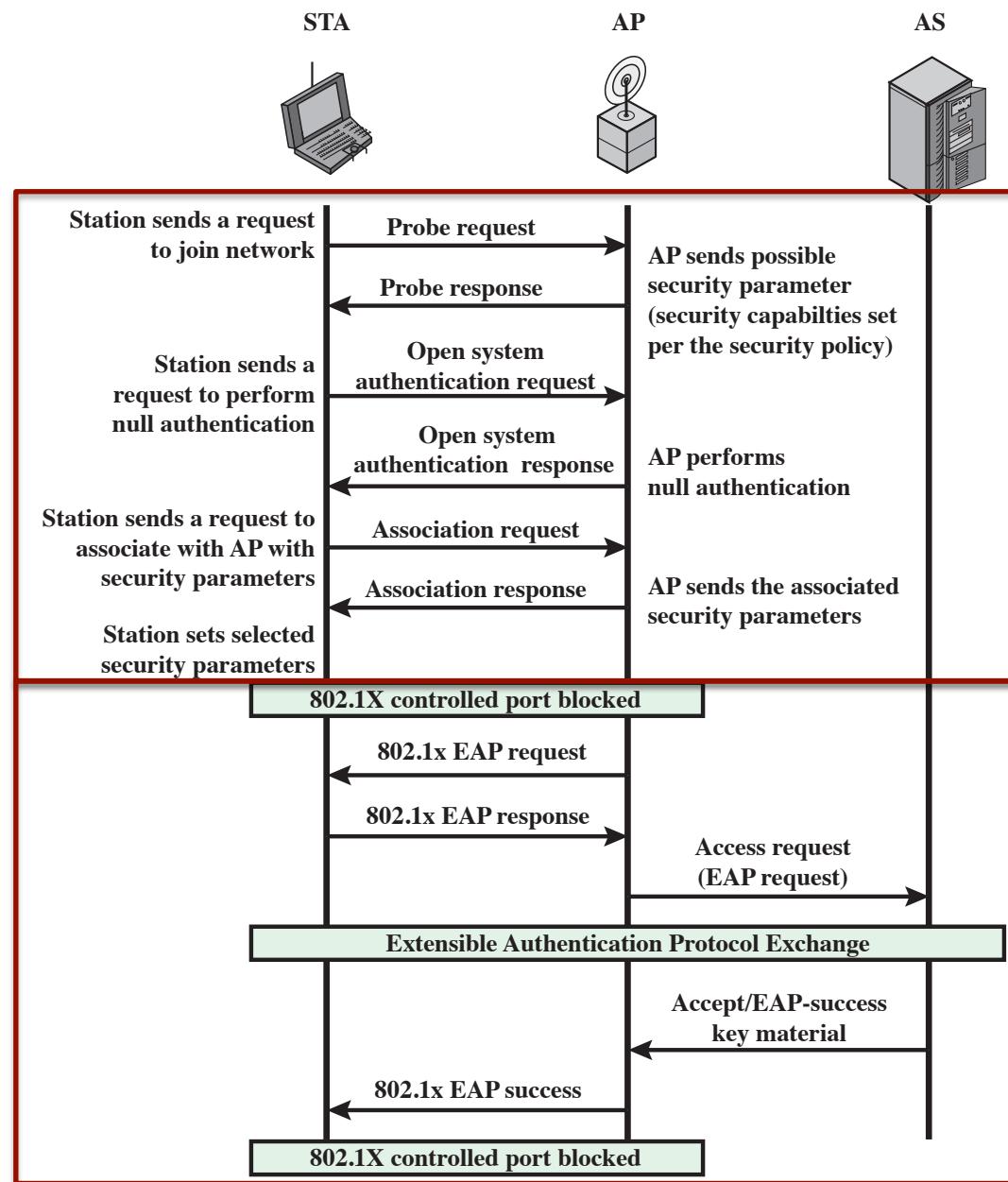
IEEE 802.11i: WPA (Wi-Fi Protected Access)

- Detalles de la **fase de autenticación**:
 - permite la autenticación mutua entre la estación y un servidor de autenticación (AS), como por ejemplo: un servidor RADIUS
 - la autenticación está diseñada para permitir sólo a estaciones autorizadas el uso de la red y garantizarles que están comunicando con una red legítima
 - el protocolo de autenticación utilizado es el **EAP (Extensible Authentication Protocol)**, definido en el estándar 802.1X
 - este estándar define los términos: *suplicant* (el cliente), *authenticator* (el AP) y *authentication server* (el AS)
 - el *authentication server* puede ser un servidor por sí mismo, o puede ejecutarse dentro del propio *authenticator* (el AP)

IEEE 802.1X / EAP

Descubrimiento

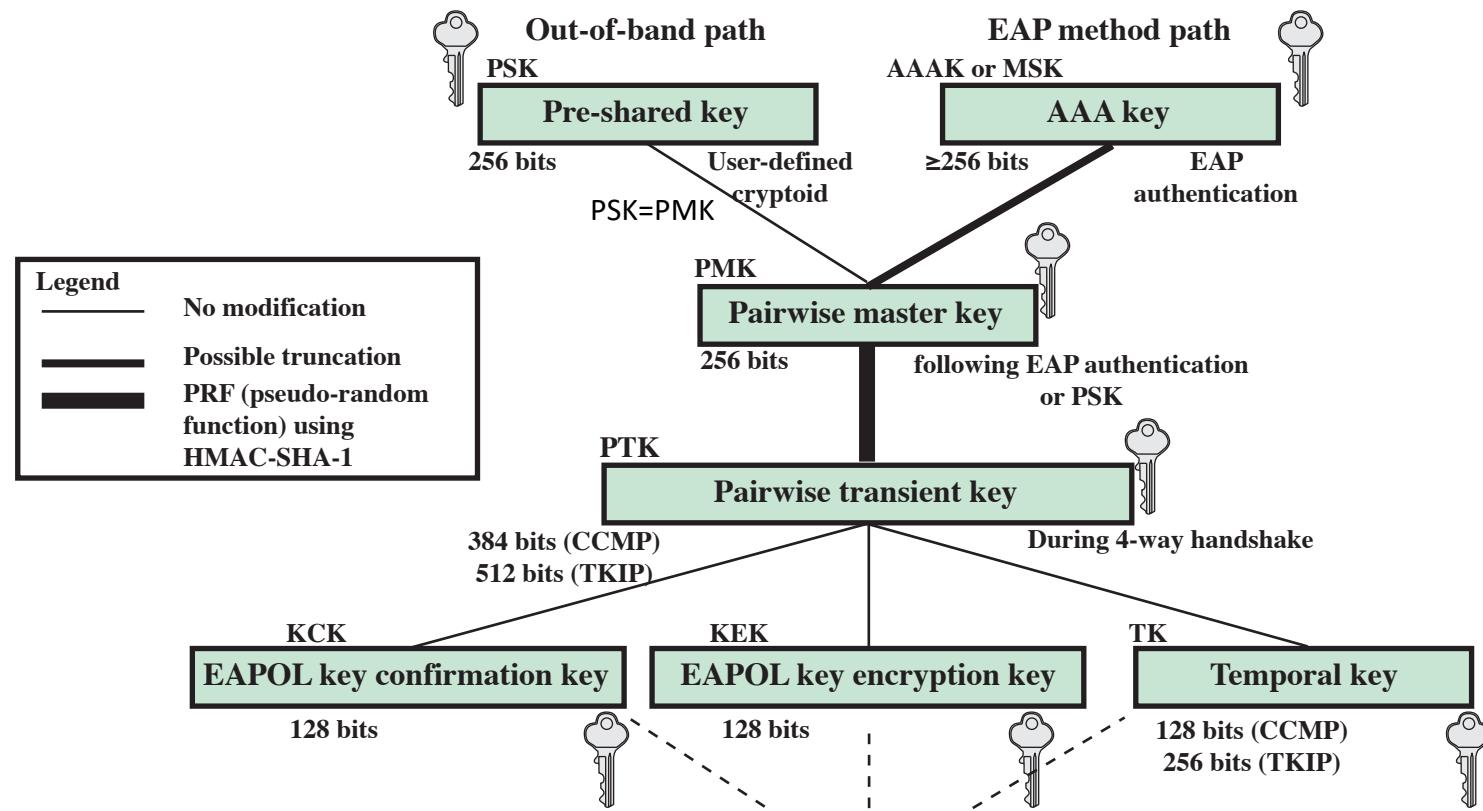
Autenticación con el AS



IEEE 802.11i Phases of Operation:
Capability Discovery, Authentication, and Association

IEEE 802.11i: WPA (Wi-Fi Protected Access)

- Detalles de la fase de generación de claves:



$$\text{CCMP} = 128 (\text{KCK}) + 128 (\text{KEK}) + 128 (\text{TK})$$

$$\text{TKIP} = 128 (\text{KCK}) + 128 (\text{KEK}) + 256 (\text{TK})$$

IEEE 802.11i: WPA (Wi-Fi Protected Access)

- Como se aprecia de la figura anterior, la **fase de generación de claves** se basa en la composición global de varias claves organizadas según una jerarquía
- Es decir, se crean claves temporales de sesión:
 - La clave **PMK** (Pairwise Master Key), la cual depende del método de autenticación
 - Si se aplica una PSK (Pre-Shared Key), entonces: PMK = PSK, cuyo valor puede derivar de una frase secreta de 8-23 caracteres, o una cadena de 256-bit
 - Solución adecuada para entornos pequeños sin servidor configurado
 - Si se usa un servidor AS, entonces PMK puede derivar de la MK (Master Key) del SA
 - Con la PMK se genera una clave temporal para cifrado denominada **PTK** (Pairwise Transient Key).
 - Su longitud depende del protocolo de cifrado (TKIP-512, CCMP-384)
 - La PTK está basado de varias claves dedicadas:
 - KCK (Key Confirmation Key – 128 bits): usada para la **autenticación de mensajes (MIC)**
 - KEK (Key Encryption Key – 128 bits): usada para garantizar la **confidencialidad de los datos**
 - TK (Temporary Key – 256/128 bits): clave para realizar otras operaciones de cifrado datos en modo TKIP o CCMP