

# Redes de Comunicaciones 2

## 3º del Grado en Ingeniería Informática

# Tema 2 : Seguridad y Criptografía

Oscar Delgado – [oscar.delgado@uam.es](mailto:oscar.delgado@uam.es)

Eloy Anguiano – [eloy.anguiano@uam.es](mailto:eloy.anguiano@uam.es)

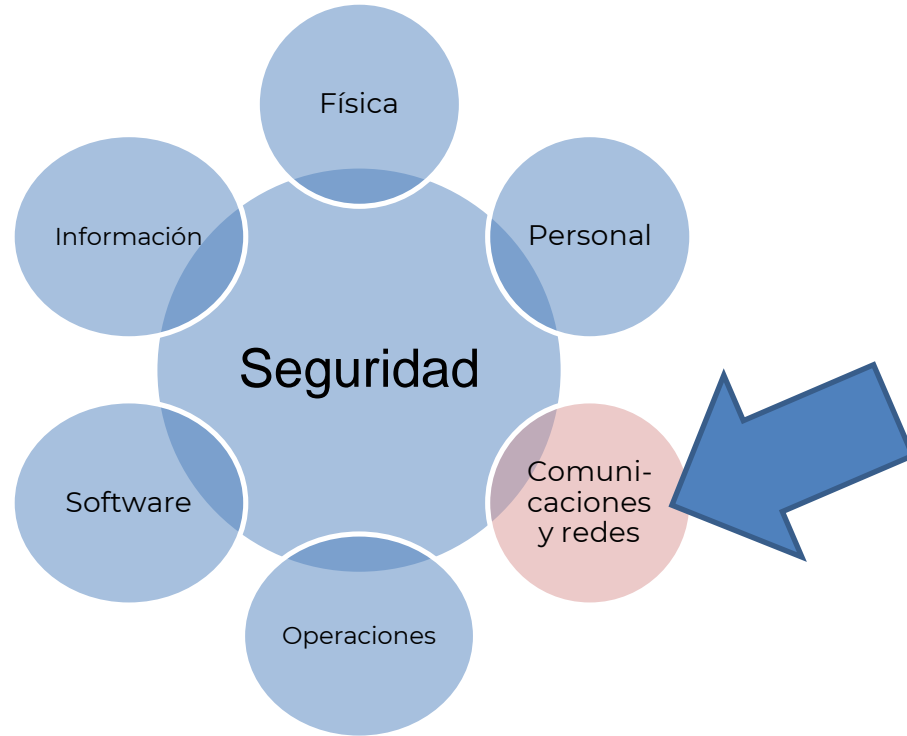
“La seguridad no es un producto,  
es un proceso”

--Bruce Schneier

“La seguridad no es problema de  
tecnología, sino de gente y de  
administración”

--Kevin Mitnick

# Seguridad de los sistemas IT



# Seguridad de la Información

C.I.A. = Confidencialidad, Integridad, Autenticación

Confidencialidad  $\approx$  Secreto

Integridad  $\approx$  Modificación

Autenticación  $\approx$  Identidad

# Seguridad en Redes

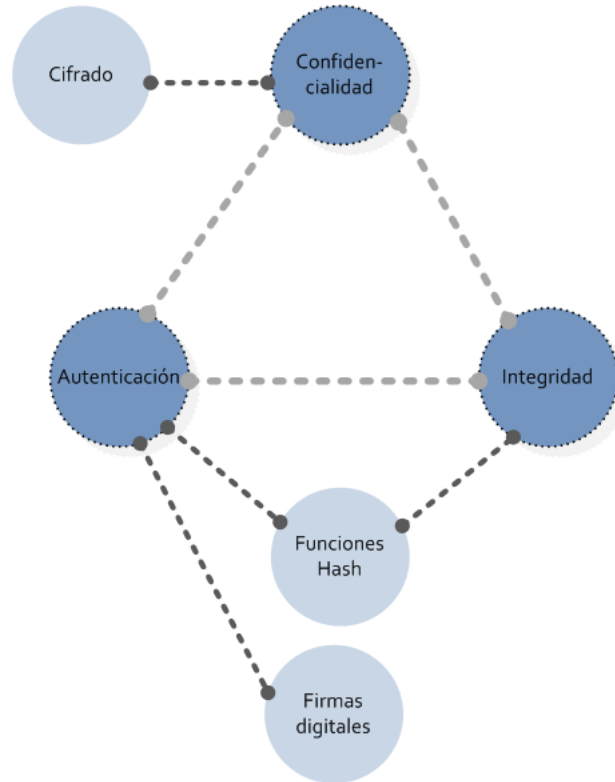


# Seguridad en Redes

- **Confidencialidad:** El receptor y no otro puede “entender” el contenido del mensaje:
  - El emisor **cifra** el mensaje
  - El receptor **descifra** el mensaje
- **Autenticación:** El receptor quiere confirmar la identidad del emisor
- **Integridad:** El receptor quiere asegurarse que el mensaje no ha sido alterado sin detectarlo
- **Disponibilidad:** Los servicios deben ser accesibles y estar disponibles para los usuarios

# Seguridad de la información

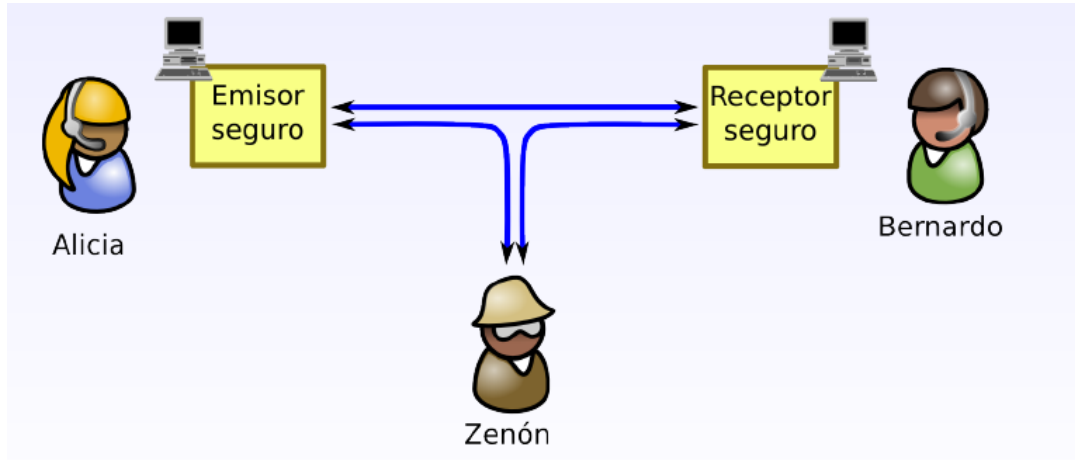
C.I.A.





# Amigos y enemigos

- Alicia y Bernardo son “amigos” y quieren comunicarse de forma “segura”.
- Zenón puede intentar interceptar, borrar o añadir mensajes.



# ¿Qué hacen los enemigos?

- Espiar mensajes ajenos
- Insertar activamente mensajes en la conexión
- Suplantación de Identidad fingiendo la dirección fuente del paquete (u otro campo)
- Secuestro de la conexión en curso poniéndose en el lugar del emisor, del receptor o de ambos
- Provocar una denegación de servicio impidiendo así que el servicio pueda ser usado por otros (e.g., sobrecargando el servicio o recursos usados por éste)

# Criptografía clásica

# Terminología

criptología = criptografía + criptoanálisis

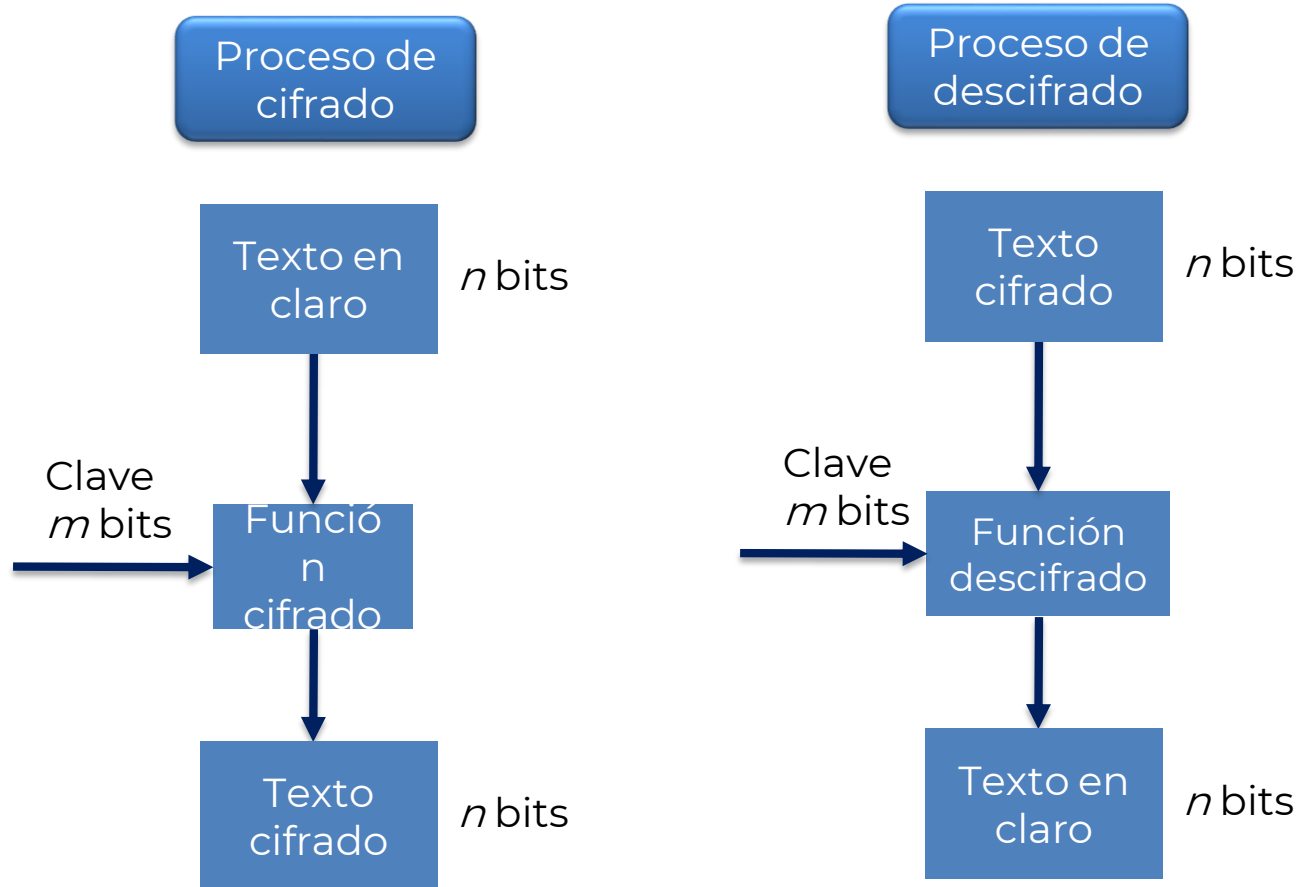
Cifrar, claves

~~Encriptar, llaves~~

Texto en claro

Texto cifrado, criptograma

# Elementos básicos



# Historia de la criptografía



Escítala espartana

# Cifrado de César

$$C = (M - 3) \pmod{26}$$

Original	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
Posición	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Cifrado	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

M = FIRMA LA PAZ



C = CFOJXIXMXW

# Cifrado de César



¿Escítala?

¿Cifrado de César?



# Criptoanálisis

## Cifradores de sustitución monoalfabéticos Análisis de frecuencias

Frecuencia Alta		Frecuencia Media		Frecuencia Baja	
E	13,11	C	4,85	Y	0,79
A	10,60	L	4,42	Q	0,74
S	8,47	U	4,34	H	0,60
O	8,23	M	3,11	Z	0,26
I	7,16	P	2,71	J	0,25
N	7,14	G	1,40	X	0,15
R	6,95	B	1,16	W	0,12
D	5,87	F	1,13	K	0,11
T	5,40	V	0,82	Ñ	0,10

# Criptografía

## Ejercicio

- Desde tu reciente puesto de analista en la T.I.A., se te encarga tu primera misión: descifrar el siguiente mensaje interceptado al enemigo

vlhuhvfdsdcghñhhuhvwrvhjxudohpwhdsuredudvhñfxuvr

# Cifrado de Vernam

## El cifrado “perfecto”

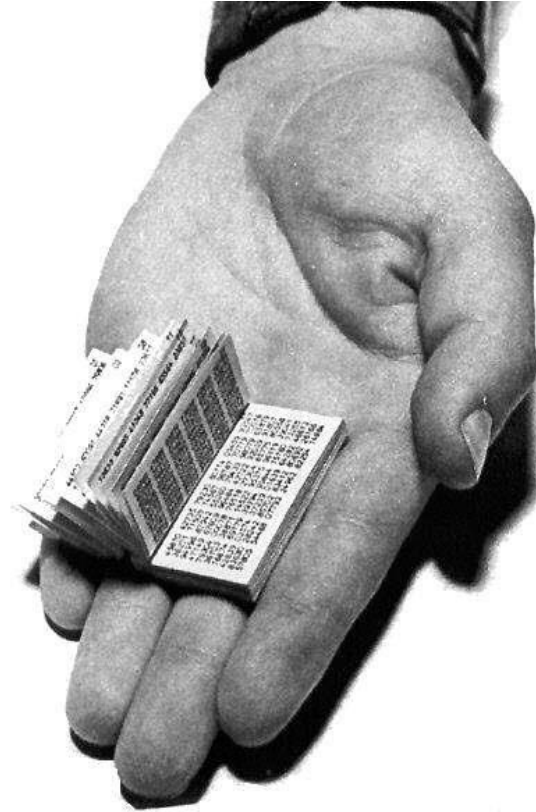
	V	E	R	N	A	M	C	I	P	H	E	R
Texto en claro	21	4	17	13	0	12	2	8	15	7	4	17
Clave aleatoria	76	48	16	82	44	3	58	11	60	5	48	88
Suma	97	52	33	95	44	15	60	19	75	12	52	105
Módulo 26	19	0	7	17	18	15	8	19	23	12	0	1
Texto cifrado	t	a	h	r	s	p	i	t	x	m	a	b

## Condiciones:

- Clave realmente aleatoria
- ¡No reutilizar nunca!
- XOR + clave aleatoria + no reutilizar = One Time Pad (OTP)

# Cifrado de Vernam

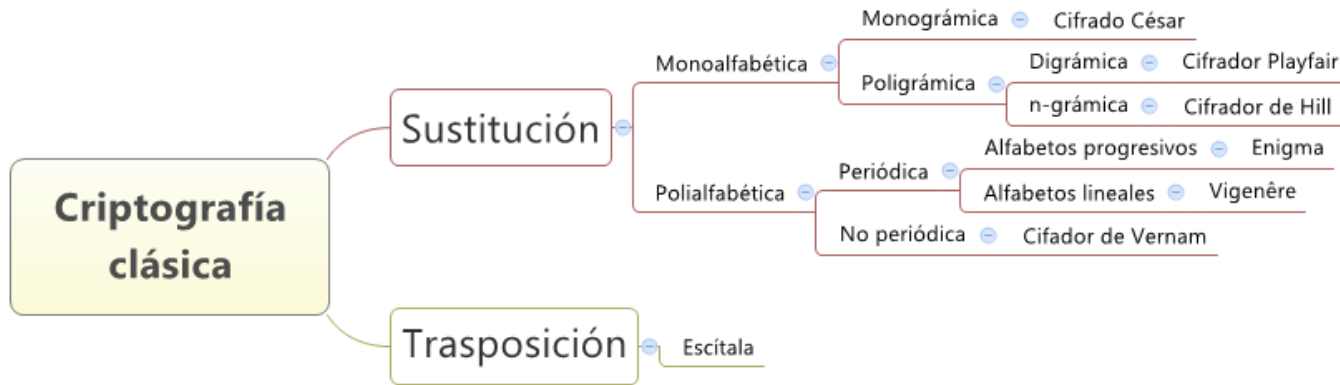
Libro de códigos ruso, capturado por el MI5



# Estaciones de números



# Cifrado de Vernam



# Cifrado de Vigenère

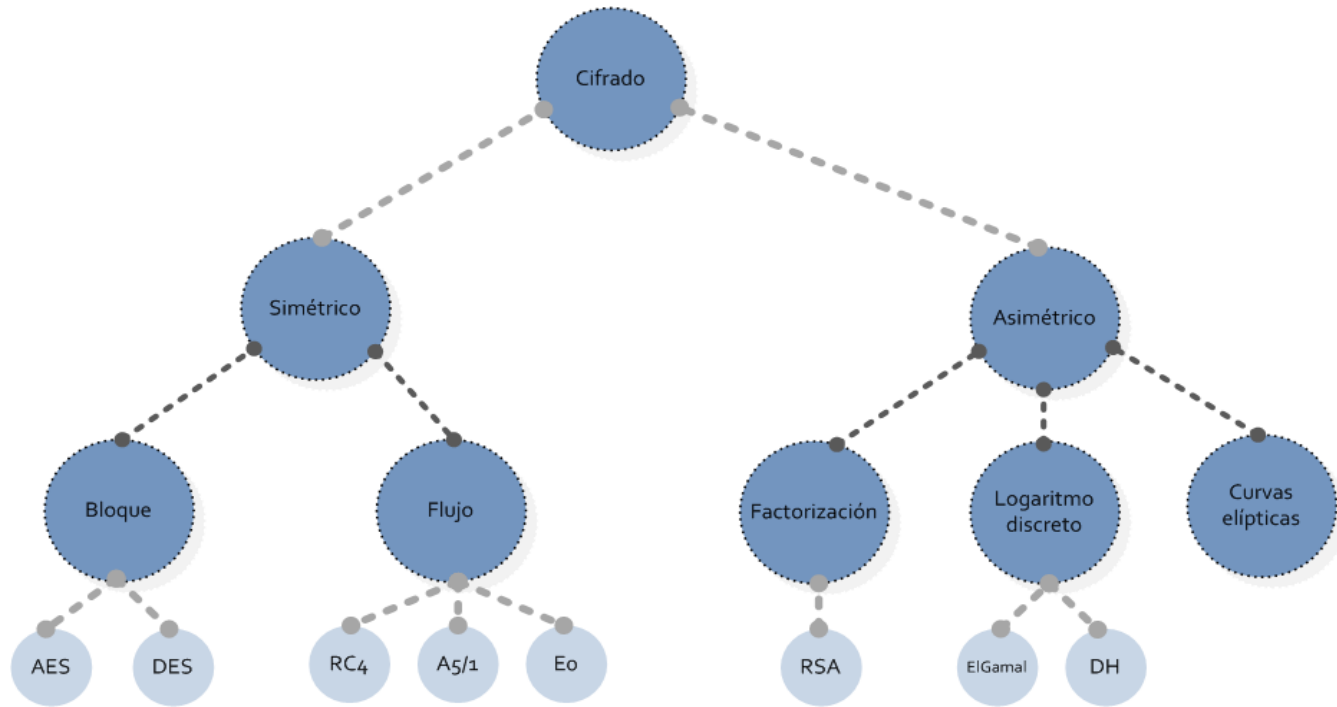
**TEXTO:** TOBEORNOTBETHATISTHEQUESTION ...  
**CLAVE:** RUNRUNRUNRUNRUNRUNRUNRUNRU...  
**CRIFTOGRAMA:** KIOVIEEIGKIOVNURNVJNUVKHVMGZIA

		Texto en claro																									
		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
Clave de cifrado	A	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
	B	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	A
	C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
	D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
	E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
	F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
	G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
	H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
	I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
	J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
	K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
	L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
	M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
	N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
	O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
	P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
	R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
	S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
	T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
	U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
	V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
	W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
	X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
	Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
	Z	Z	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

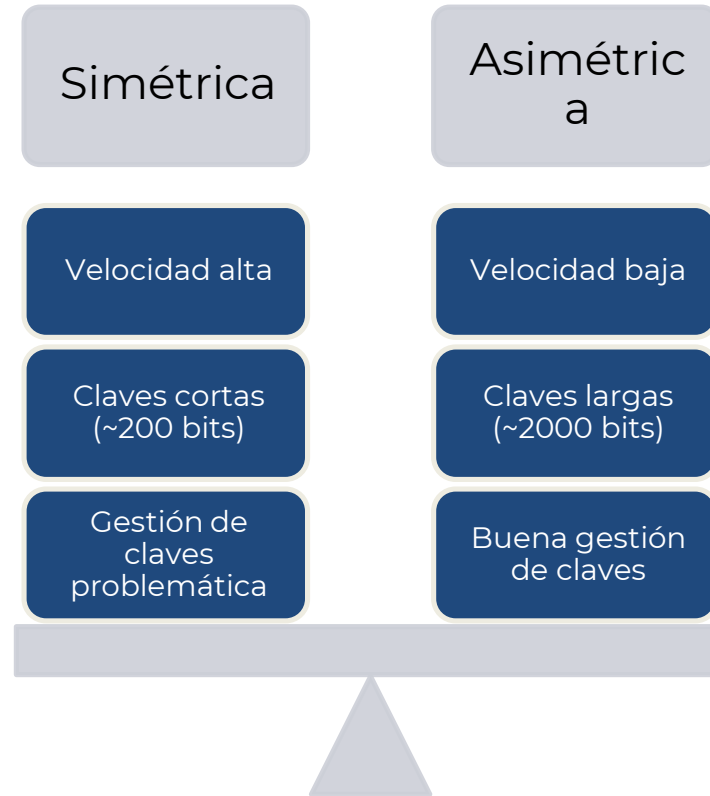
# Criptografía clásica



# Taxonomía del cifrado



# Simétrica vs Asimétrica



# Principios básicos de diseño

Mecanismos  
básicos

**Confusión** – ocultar relaciones entre texto en claro, texto cifrado y clave

**Difusión** – hacer depender la salida el máximo de la entrada

# Confusión

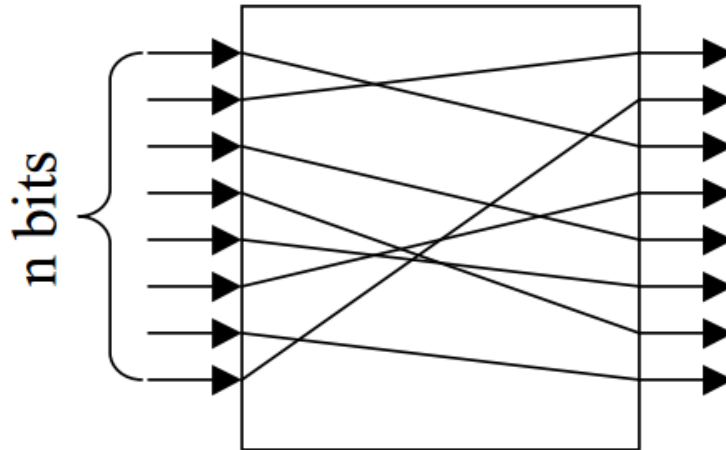
- Una palabra de entrada es sustituida por otra
- El espacio de claves posibles es  $2^n!$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7

Caja S de 4x4

# Difusión

- Se cambia el orden de los bits de una palabra de entrada
- El espacio de claves posibles es  $n!$



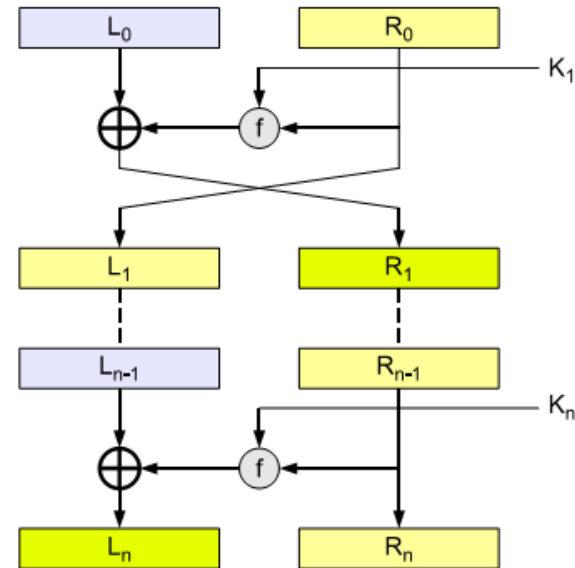
# Difusión – Efecto avalancha

- El cambio de un único bit de entrada debe producir, en término medio, el cambio de la mitad de los bits de la salida.

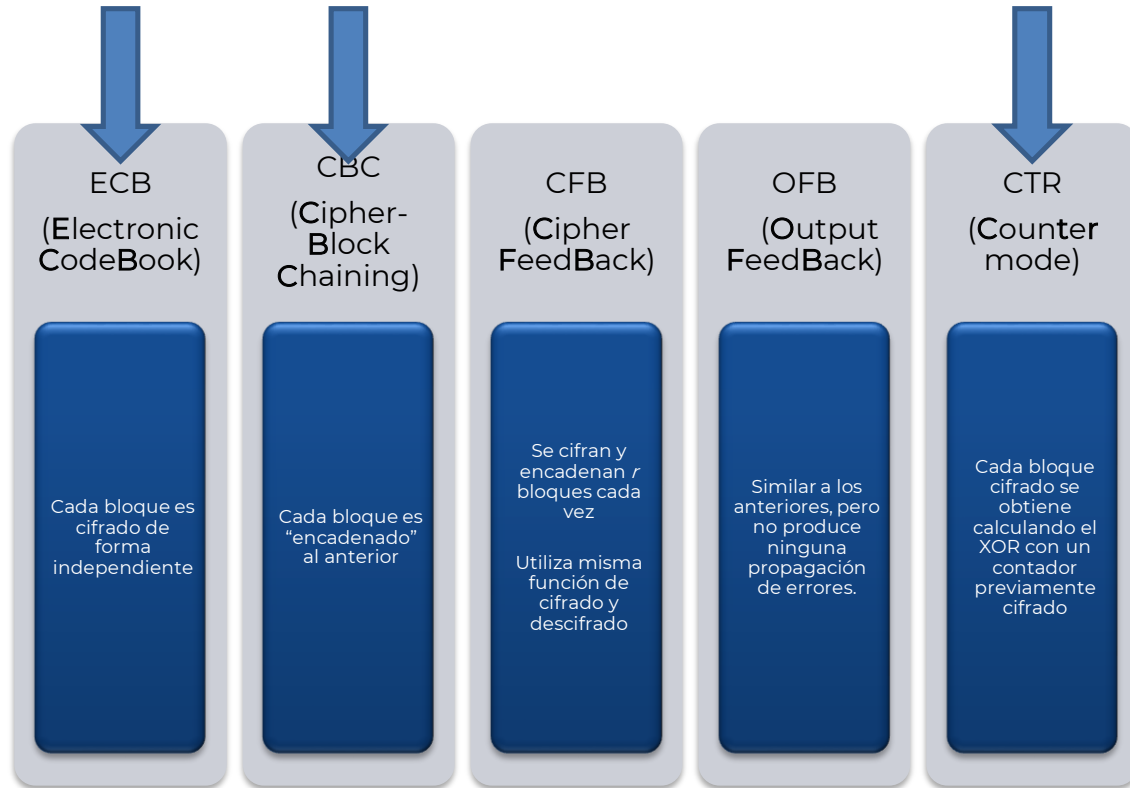
# Redes de Feistel

- $\oplus$  : función XOR
- $f$  función de **sustitución** dependiente de la clave  $K_i$
- Se realiza una **permutación** intercambiando las dos mitades de la salida  $L_i$  y  $R_i$
- El proceso se repite  $n$  veces (iteraciones)

**Descifrado:** simplemente invirtiendo el orden de las subclaves  $K_i$

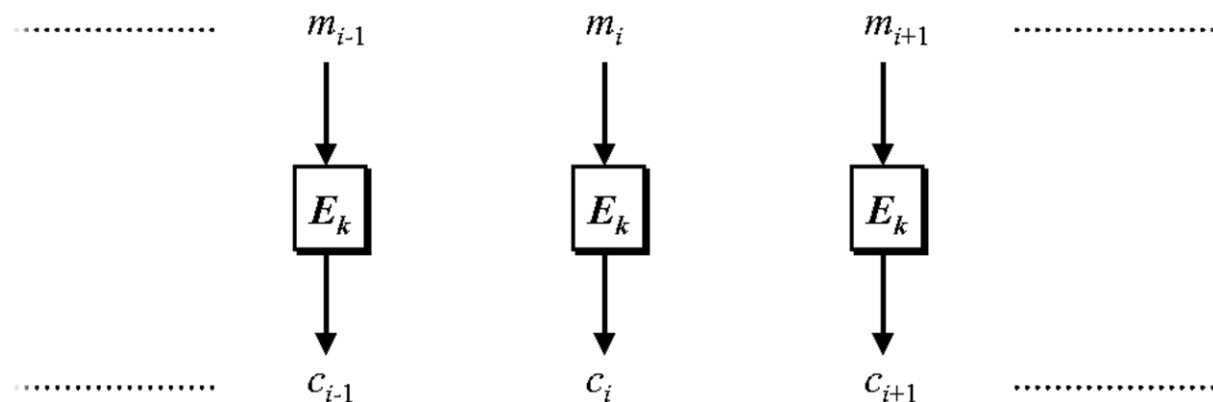


# Modos de operación





# Modo ECB



- No oculta los patrones en el texto de entrada
- Un atacante puede manipular bloques: cambiar el orden, insertar o eliminar

# Modo ECB



Propagación  
limitada de  
errores

Pueden  
reordenarse  
los bloques  
cifrados



Igual texto en  
claro produce  
idéntico texto  
cifrado

Vulnerable a  
*ataque  
semántico.*

# Modo ECB



Imagen en  
claro

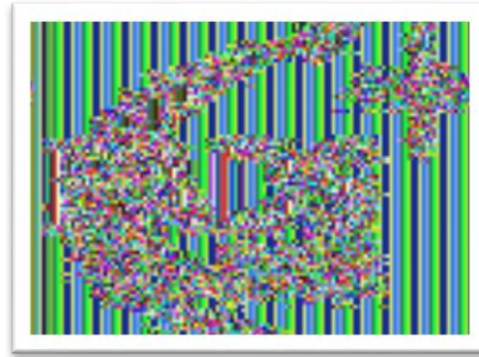
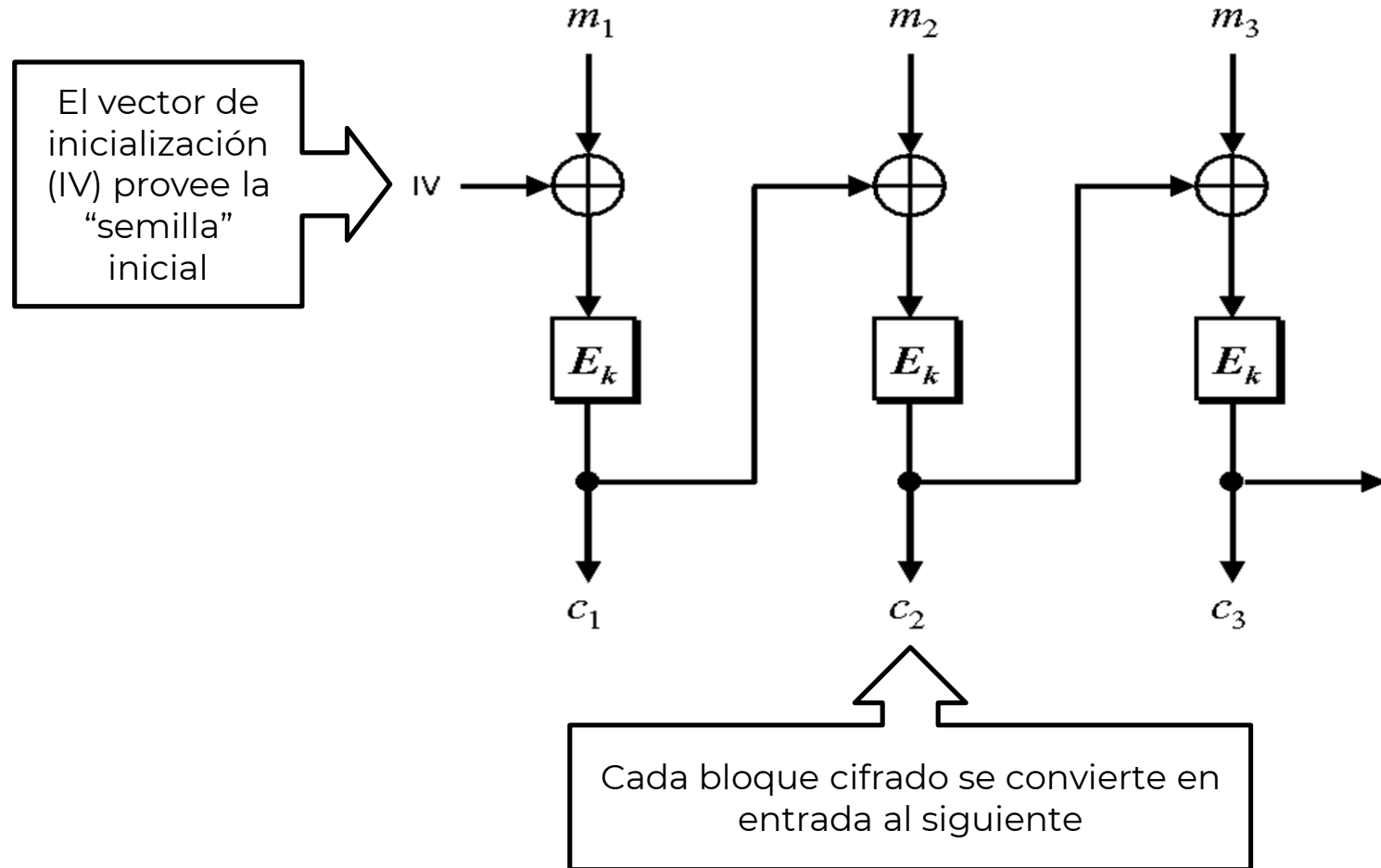


Imagen cifrada  
con el modo  
ECB



Imagen cifrada  
con cualquier  
otro modo

# Modo CBC



# Vector de inicialización

- Los IV no necesitan ser **secretos**, pero sí **impredecibles**
- ¡NO se deben reutilizar IVs con la misma clave!
- Solución: guardar el IV utilizado al inicio del fichero

# Vector de inicialización

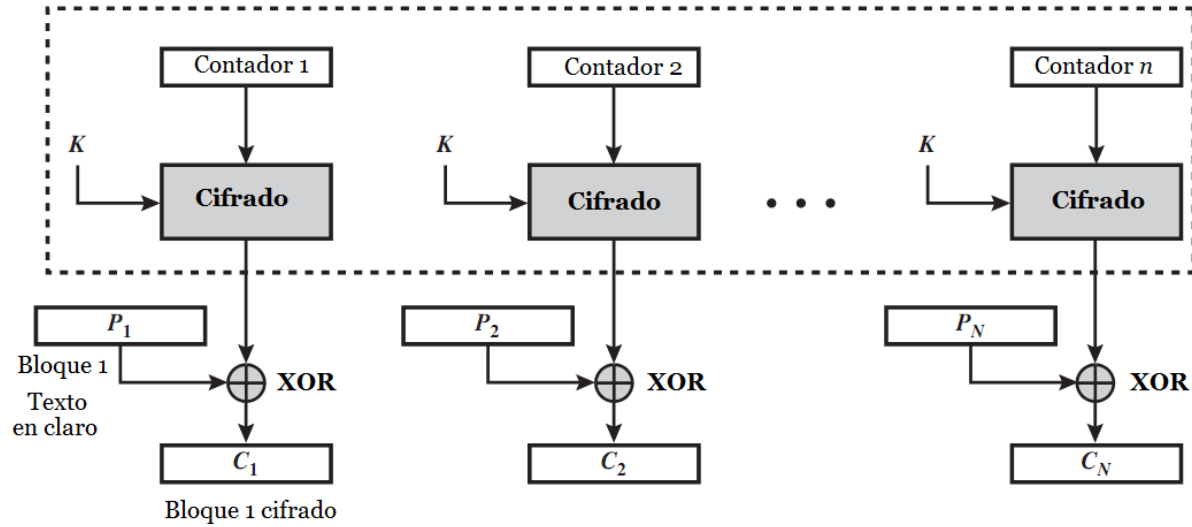
Longitud recomendada

- Objetivo: no repetir nunca IV
- Mínimo: 16 bytes = 128 bits =  $2^{128}$  posibles valores
- Idealmente, generar de forma aleatoria para cada proceso de cifrado: fichero, mensaje, etc...

# Modo CBC

- Uso correcto:
  1. Generar un IV aleatorio
  2. Cifrar en modo CBC
  3. Concatenar IV con salida antes de guardar en disco

# Modo CTR - Contador

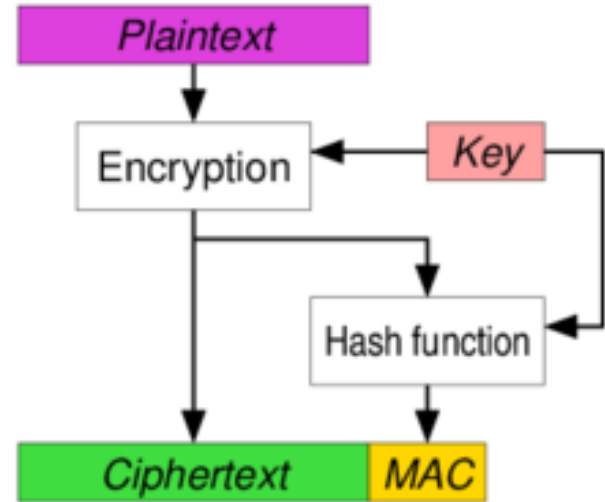


- En esencia, convierte el cifrador de bloques en uno de flujo



# Modos autenticados

- CCM, GCM
- Añaden un tag de autenticación a cada cifrado



# Resumen modos de cifrado

	ECB	CBC	CTR	
			CCM	GCM
Seguridad semántica	✗	✓	✓	✓
Encadenamiento	✗	✓	✓	✓
Propagación errores	✓ Limitada	✓ en descifrado ✗ en cifrado	¡Esencial NO repetir el IV para una misma clave!	
Recuperación de errores	-	2 bloques		

# Modos de cifrado

## Recomendación final

- Los modos autenticados (GCM, CCM) añaden seguridad adicional, pero es ESENCIAL no repetir IV
- Si esto no se puede garantizar, es preferible utilizar CBC (que tampoco debería repetir IV, pero no es tan sensible)

# DES

Finales 1960 – Se incorpora a IBM y comienza a trabajar en cripto

1971 – Primer prototipo Lucifer – Se utiliza por Lloyds en Londres

1973 – IBM propone a Lucifer como candidato a estándar de cifrado

1974 – La NSA entra en juego y “propone” cambios al algoritmo: nuevas cajas S y reducción de la clave de 128 a 56 bits.

1977 – El algoritmo es aceptado como estándar y pasa a denominarse DES (Data Encryption Standard)



Horst Feistel – (1915-1990)

# DES - Polémicas

¿Modificación cajas  
S?

*Criptoanálisis  
diferencial*

¿Reducción del  
tamaño de clave?

*Atacable por fuerza  
bruta*

¿Tienen puertas  
traseras?

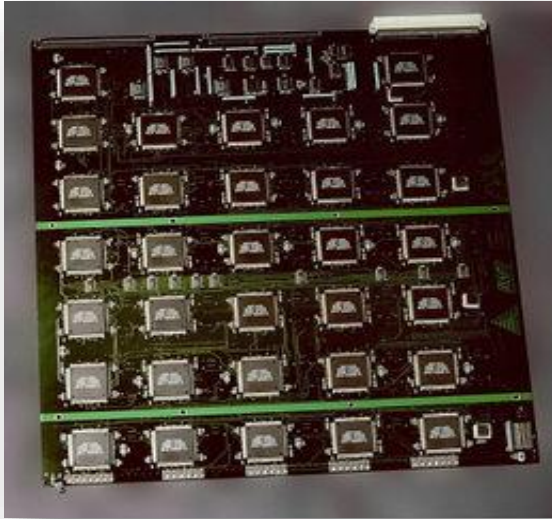
*No, se habrían  
detectado ya*

¿La NSA podía  
descifrarlo?

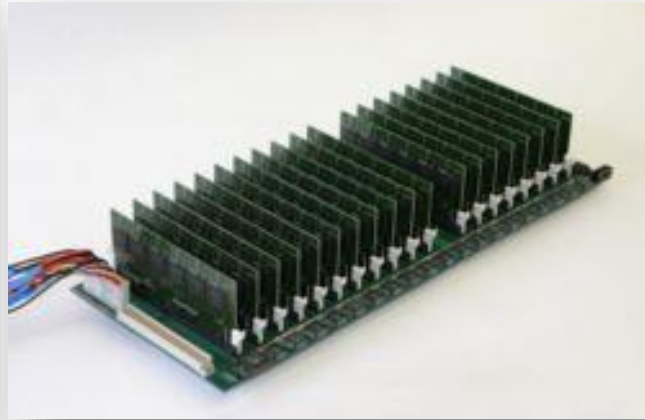
*Sí, pero le llevaba  
varios días (Opinión)*

# DES – Situación actual

DES es INSEGURO



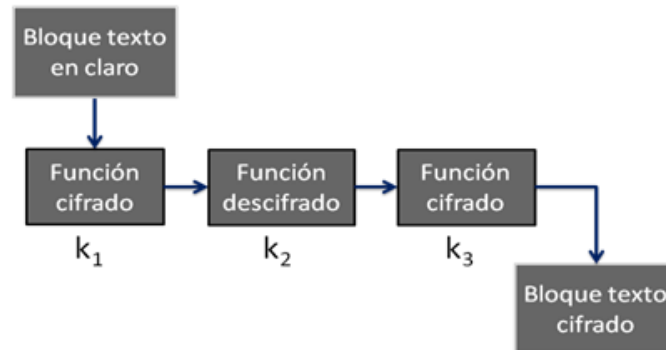
Atacable por hardware



Las más rápidas pueden probar ahora 90 billones de claves por segundo

# 3DES

- Actualmente DES es **completamente inseguro**
- Cambiar software y hardware es caro
- Solución: concatenar varias iteraciones de DES → 3DES



# AES – Advanced Encryption Standard

- Nuevo estándar de cifrado desde 2000
- No es de tipo Feistel, sino que utiliza álgebra de cuerpos finitos (concretamente un cuerpo de Galois,  $GF(2^8)$ ), aunque sí utiliza cajas S.
- Diseñado para ser eficiente en microprocesadores de cualquier ancho de palabra, desde 8 bits, usados en tarjetas inteligentes o microcontroladores, hasta CPUs de 64 bits.
- La NSA elige AES en 2003 para cifrar su propia información clasificada como secreto y alto secreto



# AES – Advanced Encryption Standard

Base matemática	Número de etapas	Cajas S
<ul style="list-style-type: none"><li>• Operaciones algebraicas en cuerpos finitos</li><li>• No es de tipo Feistel</li></ul>	<ul style="list-style-type: none"><li>• Flexible según necesidades del usuario.</li></ul>	<ul style="list-style-type: none"><li>• Usa un conjunto de Cajas S similares a las del DES.</li></ul>

# AES – Advanced Encryption Standard

## Tamaño de palabra

- 8 bits (tarjetas inteligentes y CPUs)

## Tamaño de clave variable

- 128, 192 y 256 bits (estándar) o bien múltiplo de 4 bytes.

## Tamaño del bloque de texto

- 128 bits o múltiplo de 4 bytes

# Ataques por fuerza bruta

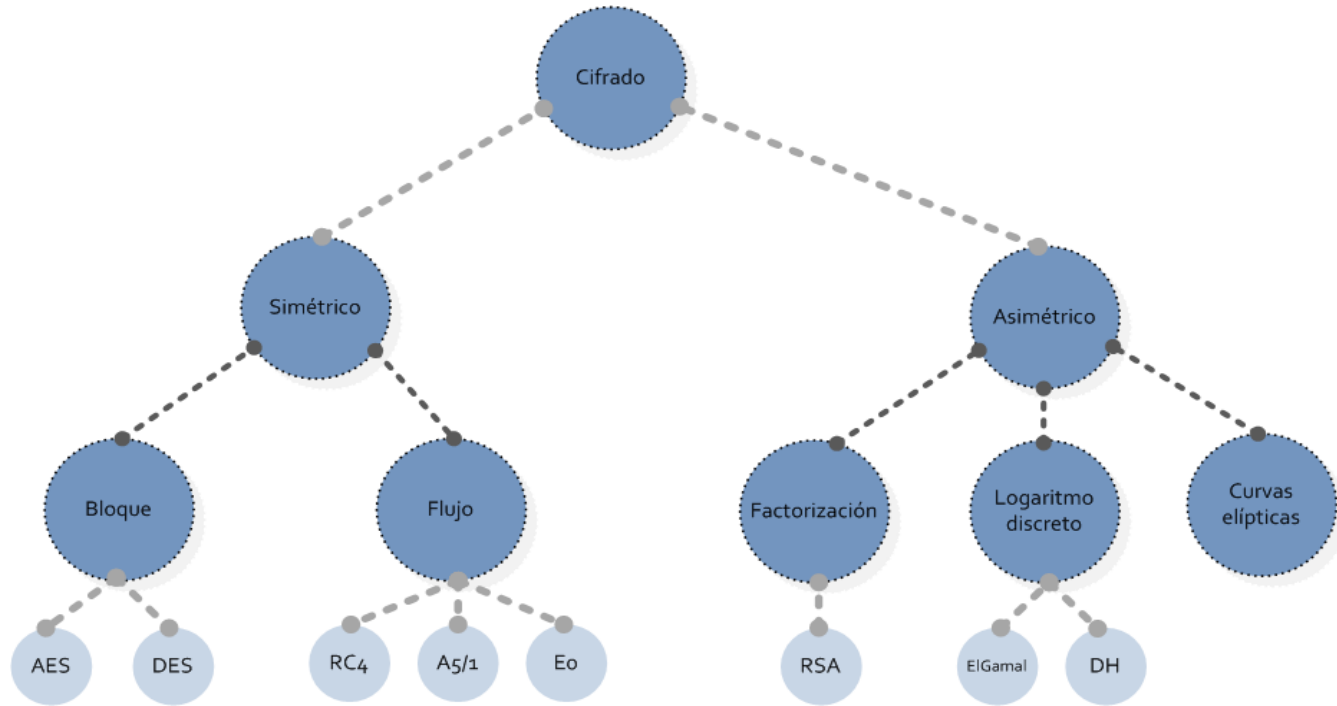
Longitud de la clave	Tiempo necesario para romper la clave
40 bits	2 segundos
48 bits	9 minutos
56 bits	40 horas
64 bits	14 meses
72 bits	305 años
80 bits	78.250 años
96 bits	5.127.160.311 años
112 bits	336.013.578.167.538 años
128 bits	22.020.985.858.787.784.059 años
192 bits	$1.872 \cdot 10^{37}$ años
256 bits	$9.1 \cdot 10^{50}$ años

# Vector de inicialización

## Ejercicio MOD5\_CRYPT0\_JAVA

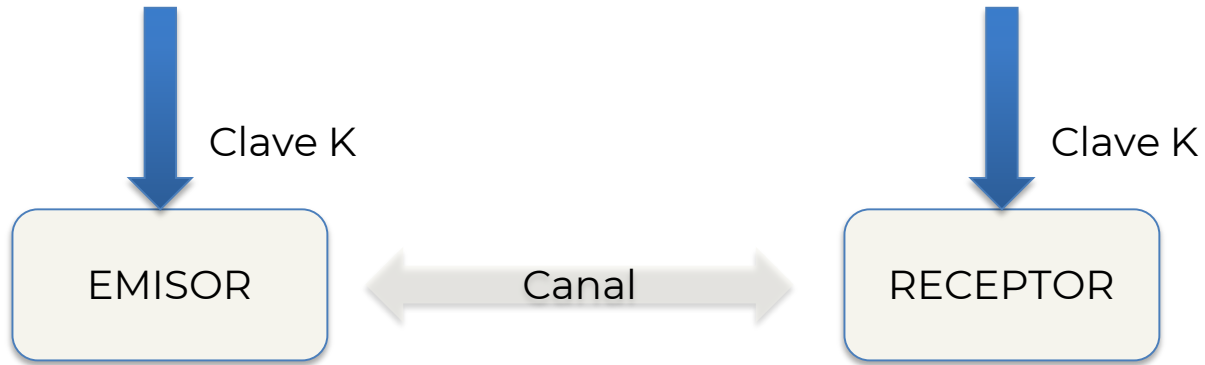
- Tiempo de realización: 30 minutos
- Individual

# Taxonomía



# Cripto simétrica

## Clave simétrica



Inconveniente: **distribución de claves**

# Historia de la cripto asimétrica

1976



Hellman

Diffie

# Historia de la cripto asimétrica

1976

611

IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. IT-22, NO. 6, NOVEMBER 1976

## New Directions in Cryptography

*Invited Paper*

WHITFIELD DIFFIE AND MARTIN E. HELLMAN, MEMBER, IEEE

**Abstract**—Two kinds of contemporary developments in cryptography are examined. Widening applications of teleprocessing have given rise to a need for new types of cryptographic systems, which minimize the need for secure key distribution channels and supply the equivalent of a written signature. This paper suggests ways to solve these currently open problems. It also discusses how the theories of communication and computation are beginning to provide the tools to solve cryptographic problems of long standing.

### I. INTRODUCTION

**W**E STAND TODAY on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high grade

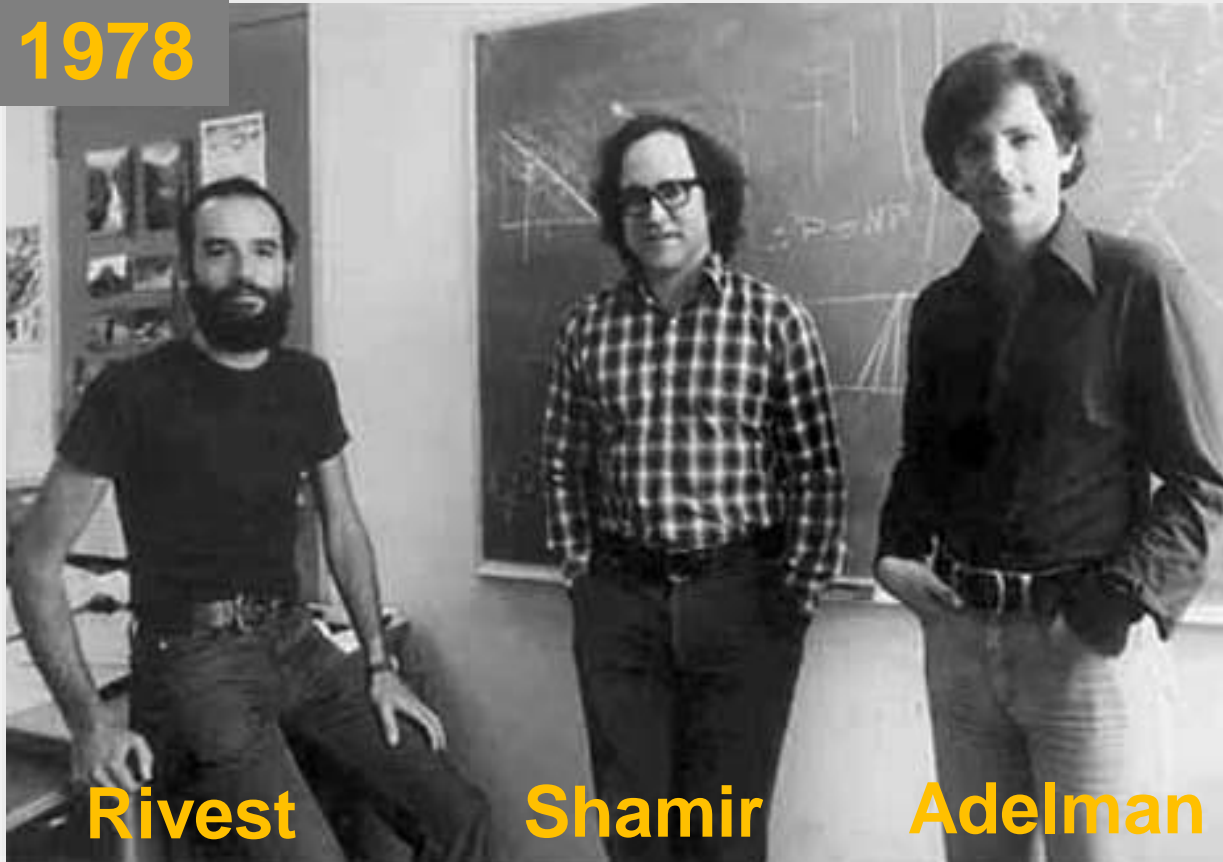
The best known cryptographic problem is that of privacy: preventing the unauthorized extraction of information from communications over an insecure channel. In order to use cryptography to insure privacy, however, it is currently necessary for the communicating parties to share a key which is known to no one else. This is done by sending the key in advance over some secure channel such as private courier or registered mail. A private conversation between two people with no prior acquaintance is a common occurrence in business, however, and it is unrealistic to expect initial business contacts to be postponed long enough for keys to be transmitted by some physical means. The cost and delay imposed by this key distribution problem is a major barrier to the transfer of business communications to large teleprocessing networks.

Section III proposes two approaches to transmitting



# Historia de la cripto asimétrica

**1978**



**Rivest**

**Shamir**

**Adelman**

# Historia de la cripto asimétrica

- Con el tiempo se ha sabido que Clifford Cocks, de los servicios de inteligencia británicos, descubrió estos esquemas en 1973.
- No fueron publicados por considerarse alto secreto.

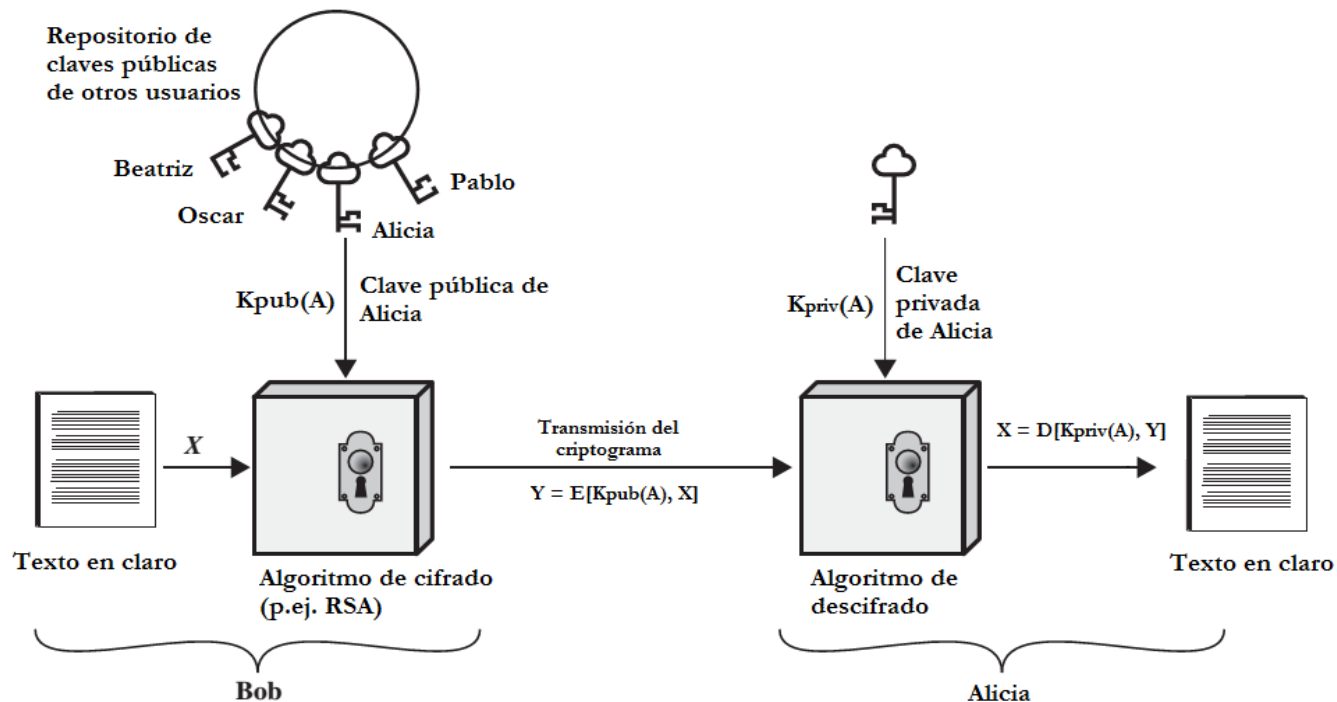
# Elementos básicos

- Dos claves, denominadas clave pública y privada, vinculadas matemáticamente.

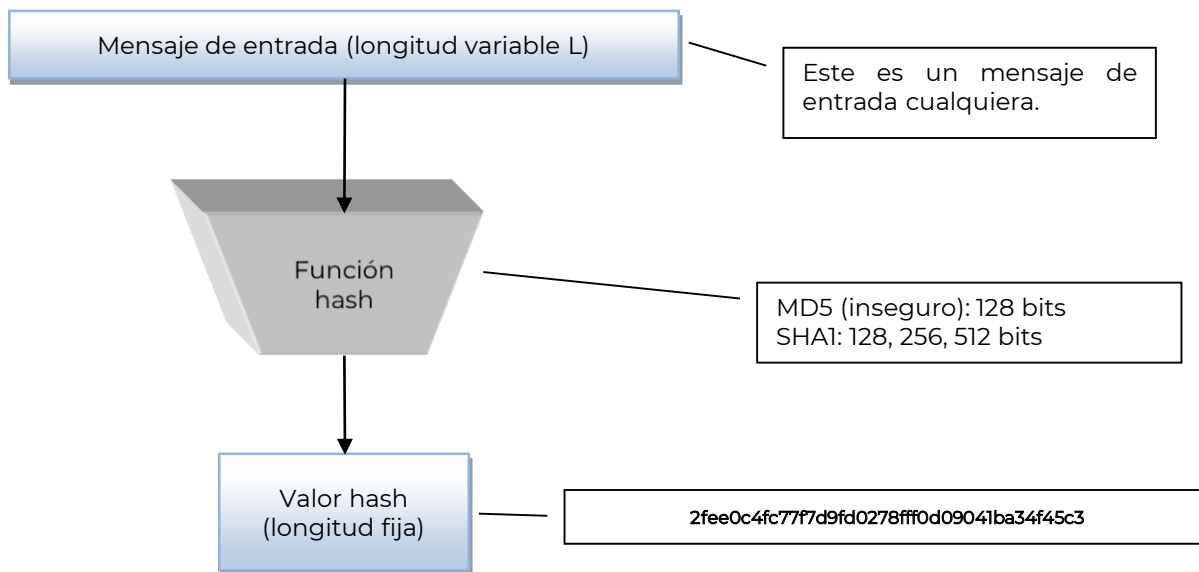
Principio básico:

“Lo que cifra con una de las claves, sólo se puede descifrar con la otra”

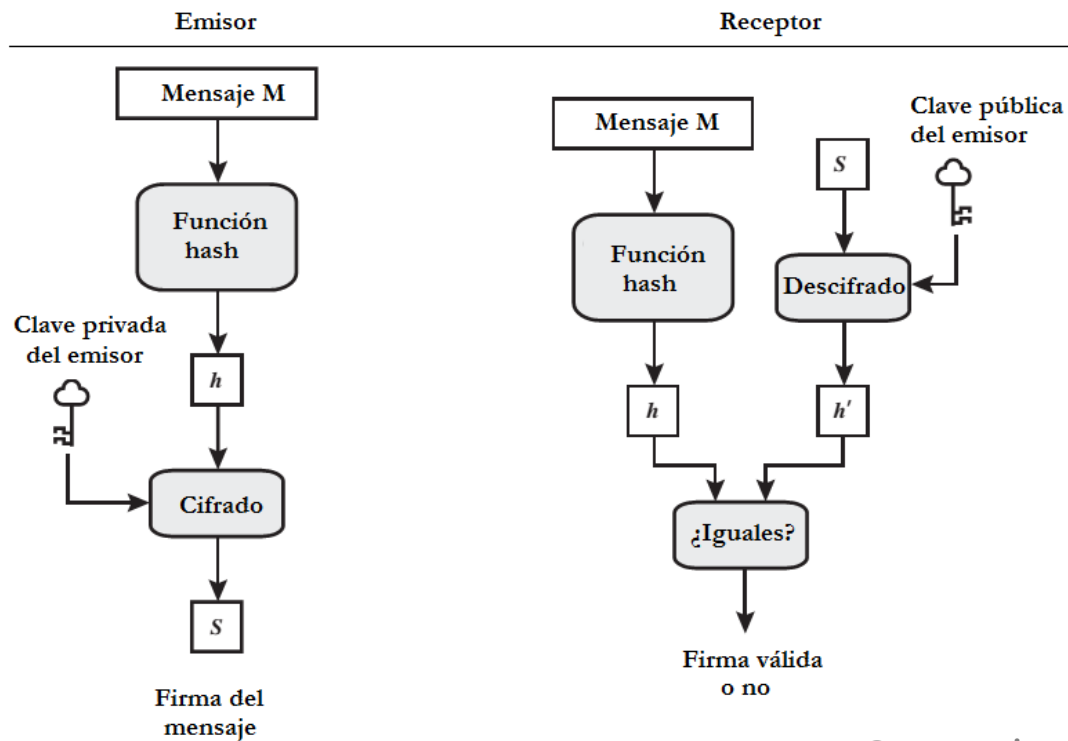
# Elementos básicos



# Funciones hash

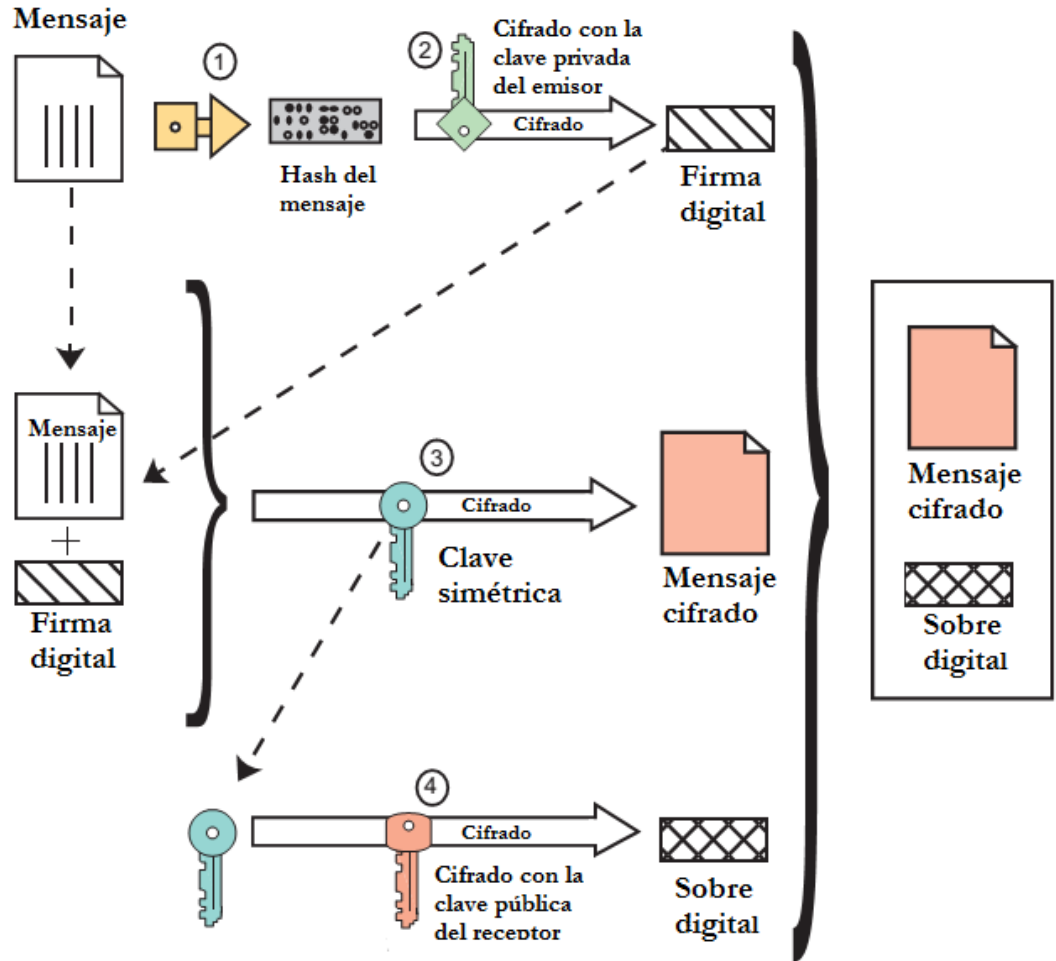


# Firma digital



Garantiza el no repudio

# Esquemas híbridos



# Algoritmos asimétricos

## RSA

- Factorización de enteros

## ElGamal

- Logaritmo discreto

## Curvas elípticas

- Curvas elípticas sobre cuerpos finitos
- Muy eficientes y pequeñas longitudes de claves



# Longitudes de clave

Clave simétrica (bits)	Clave RSA (bits)	Estimación resistencia (años)	Clave ECC (bits)
80	1024	2006-2010	160-223
112	2048	2030	224-255
128	3072	>2030	256-383
256	15360 (!)	-	512+

# Clasificación de información

Clave simétrica	No clasificada	SECRET	TOP SECRET
Cifrado	AES-128	AES-128	AES-256
Funciones hash	SHA-1	SHA-256	SHA-384
Firma digital	RSA-2048	ECC-256	ECC-384

# Clave secreta vs pública: rendimiento

Algoritmo	Longitud de clave (bits)	Velocidad de cifrado (Kb/s)
RC4	128	196.234
DES	128	58.413
3DES	128	18.860
AES	128	121.119
RSA	512	656

Unas 100 veces más lenta

# Algoritmo RSA

## Factorización

$$p \cdot q = n$$



Dado  $n$ , ¿cuáles son  $p$  y  $q$ ?

# Algoritmo RSA

## Aritmética modular

Si trabajamos en  $Z_n = \{0, 1, \dots, (n-1)\}$  se cumplen las siguientes propiedades:

$$(a + b) \bmod n = [(a \bmod n) + (b \bmod n)] \bmod n \quad [1]$$

$$(a - b) \bmod n = [(a \bmod n) - (b \bmod n)] \bmod n \quad [2]$$

$$(a \cdot b) \bmod n = [(a \bmod n) \cdot (b \bmod n)] \bmod n \quad [3]$$

# Algoritmo RSA

## Aritmética modular

Ejemplos:

$$11 \bmod 8 = 3; 15 \bmod 8 = 7$$

$$\begin{aligned} [(11 \bmod 8) + (15 \bmod 8)] \bmod 8 &= [3 + 7] \bmod 8 = 10 \bmod 8 = 2 \\ (11 + 15) \bmod 8 &= 26 \bmod 8 = 2 \end{aligned}$$

$$\begin{aligned} [(11 \bmod 8) \cdot (15 \bmod 8)] \bmod 8 &= [3 \cdot 7] \bmod 8 = 21 \bmod 8 = 5 \\ (11 \cdot 15) \bmod 8 &= 165 \bmod 8 = 5 \end{aligned}$$

# Algoritmo RSA

## Aritmética modular

### Ejemplos (exponenciación)

Para encontrar  $11^7 \bmod 13$ , podemos hacer:

$11^7 \bmod 13 = 19487171 \bmod 13$  (Costoso)

O utilizar las propiedades anteriores:

Sabemos que  $11^7 = 11^4 \cdot 11^2 \cdot 11$  y por la prop. [3]:

$$11^2 = 121 \bmod 13 = 4 \bmod 13$$

$$11^4 = (11^2)^2 \bmod 13 = 4^2 \bmod 13 = 3 \bmod 13$$

$$11^7 = (11 \cdot 4 \cdot 3) \bmod 13 = 132 \bmod 13 = 2 \bmod 13$$

# Algoritmo RSA

## Fundamentos matemáticos

*Función  $\varphi$  de Euler*

$\varphi(n)$  = número de enteros  
positivos **coprimos** con  $n$



# Algoritmo RSA

## Fundamentos matemáticos

*Función  $\varphi$  de Euler*

$\varphi(n)$  = número de enteros  
positivos **coprimos** con  $n$

# Algoritmo RSA

## Generación de claves

1.  $p, q$ , dos números primos grandes
2.  $n = pq, \varphi(n) = (p-1) \cdot (q-1)$
3. Seleccionar  $e$ , con  $\text{mcd}(\varphi(n), e) = 1; 1 < e < \varphi(n)$
4. Buscar  $d \equiv e^{-1} \pmod{\varphi(n)} = d \cdot e \equiv 1 \pmod{\varphi(n)}$

Clave pública =  $\{e, n\}$

Clave privada =  $\{d, n\}$

# Algoritmo RSA

## Generación de claves - Ejemplo

1. Elegimos  $p = 3$  y  $q = 11$
2. Calculamos  $n = p \cdot q = 3 \cdot 11 = 33$ .
3. Calculamos  $\phi(n) = (p-1) \cdot (q-1) = 2 \cdot 10 = 20$ .
4. Elegimos  $e$  tal que  $1 < e < \phi(n)$  y  $e$  y  $n$  sean primos entre sí. Por ejemplo,  $e = 7$ .
5. Calculamos un valor para  $d$  tal que  $(d \cdot 7) \equiv 1 \pmod{20}$ . Una solución es  $d = 3$ .

1. La clave pública es  $\{e, n\} = \{7, 33\}$
2. La clave privada es  $\{d, n\} = \{3, 33\}$

# Algoritmo RSA

## Proceso de cifrado/descifrado

A cifra un mensaje  $m$  para B

1. **Cifrado.** A debe:
  - a. Obtener la clave pública de B,  $(e, n)$
  - b. Representar el mensaje con un entero  $m$  en el intervalo  $[0, n-1]$
  - c. Calcular  $c = m^e \pmod n$  y enviar a B.
2. **Descifrado.** Para recuperar el texto en claro de  $c$ , B debe:
  - a. Usar su clave privada  $d$ , y calcular  $m = c^d \pmod n$

# Algoritmo RSA

## Ejemplo de cifrado

La clave pública es  $\{e, n\} = \{7, 33\}$

La clave privada es  $\{d, n\} = \{3, 33\}$

1. Cifrado del mensaje: USA TU ARMA!
2. Codificamos cada símbolo de forma numérica (ASCII, por ejemplo)
3. Ciframos cada símbolo (U=85 en ASCII)  
 $85 = 19^7 \pmod{33} = 13$

1. El descifrado es, simplemente:  
 $13^3 \pmod{33} = 19$

# Algoritmo RSA

## Disclaimer

1. ¡Hemos visto ejemplos MUY simplificados! NO ofrecen seguridad real.
2. Cifrando símbolo a símbolo convertimos el proceso en un simple cifrado de sustitución, atacable por un análisis de frecuencias.
3. Una posible solución es combinar varios números en bloques (entonces,  $n$  debe ser mayor que el mayor número posible del bloque)

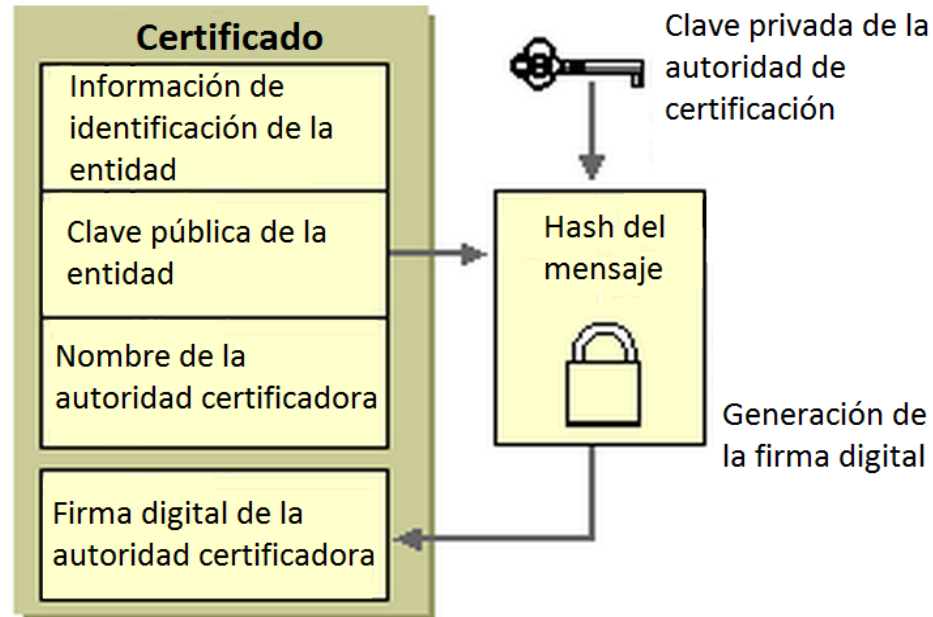
En la práctica, RSA se utiliza en combinación con un cifrador simétrico, cifrando sólo su clave secreta.

# Certificados digitales

## Necesidad

¿Cómo puedo obtener de forma fiable la clave pública de otro usuario?

# Certificados digitales





# Certificados X.509

- **Versión:** hace referencia al número de versión del certificado. Los posibles valores son 1, 2 y 3.
- **Número de serie:** es un número identificador único, asignado por la CA en el momento de la creación del certificado.
- **Identificador del algoritmo de firmado:** identifica el algoritmo empleado para firmar el certificado (como RSA).
- **Nombre del emisor:** identifica la CA que ha emitido y firmado el certificado.
- **Fechas de validez:** por razones de seguridad, un certificado o, en general, una clave criptográfica no puede durar eternamente. Este campo contiene el periodo de tiempo durante el que el certificado es válido, compuesto de una fecha inicial, en la que empieza a poder ser utilizado (es válido) y la fecha después de la cual el certificado deja de serlo.
- **Nombre del sujeto:** campo que identifica a la entidad certificada. Este “nombre” puede ser el nombre real de una persona, o el nombre de una máquina, en el caso de un ordenador.
- **Clave pública:** el campo más importante, que contiene la clave pública en sí.
- **Huella digital:** valor hash del certificado que, por tanto, sirve también para identificarlo de forma única.

# Certificados X.509

## Ejercicio MOD5\_X509

- Tiempo de realización: 5 minutos
- Individual
- Abre el almacén de certificados de tu navegador, y estudia el de alguna CA raíz.
- Ordénalos por fecha de expiración. ¿A partir de qué fecha aproximadamente su longitud de clave pasa de 4096 a 2048 bits?

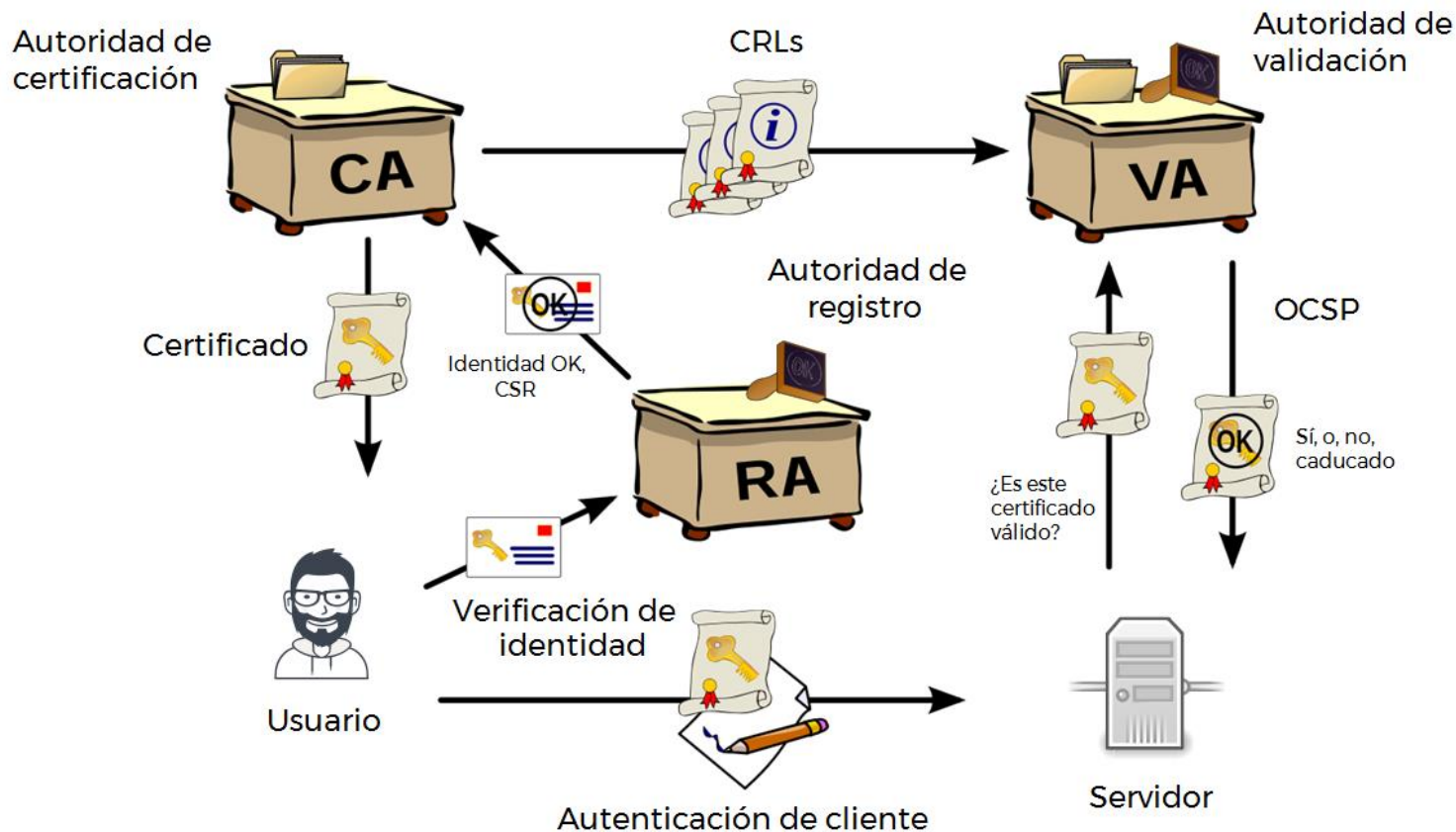
# Certificados X.509

## Ejercicio

### MOD5\_CRYPTO\_ASIMETRICA

- Tiempo de realización: 25 minutos
- Individual

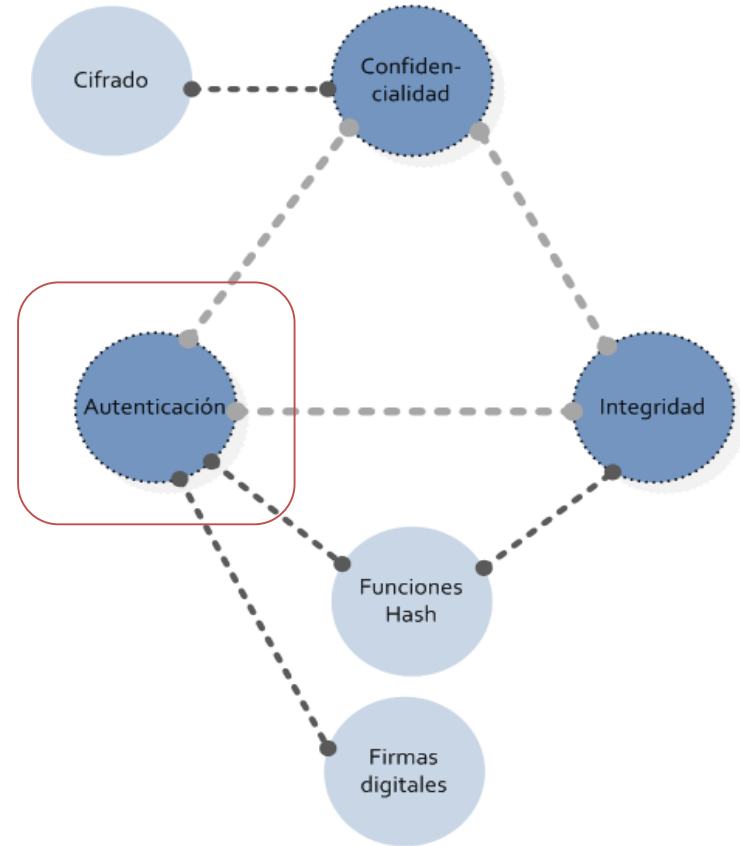
# PKI – Public Key Infrastructure



# Autenticación y gestión de credenciales

# Autenticación

“Verificación de la identidad de otra parte”



# Autenticación

- La “otra parte” puede ser una persona, aplicación, máquina, etc...
- **Identificación:** autenticación específica de personas
- **Autorización:** gestión de los permisos de una parte ya autenticada

# Autenticación

## Algo que sabemos

- Contraseñas
- PINs
- Claves criptográficas

## Algo que tenemos

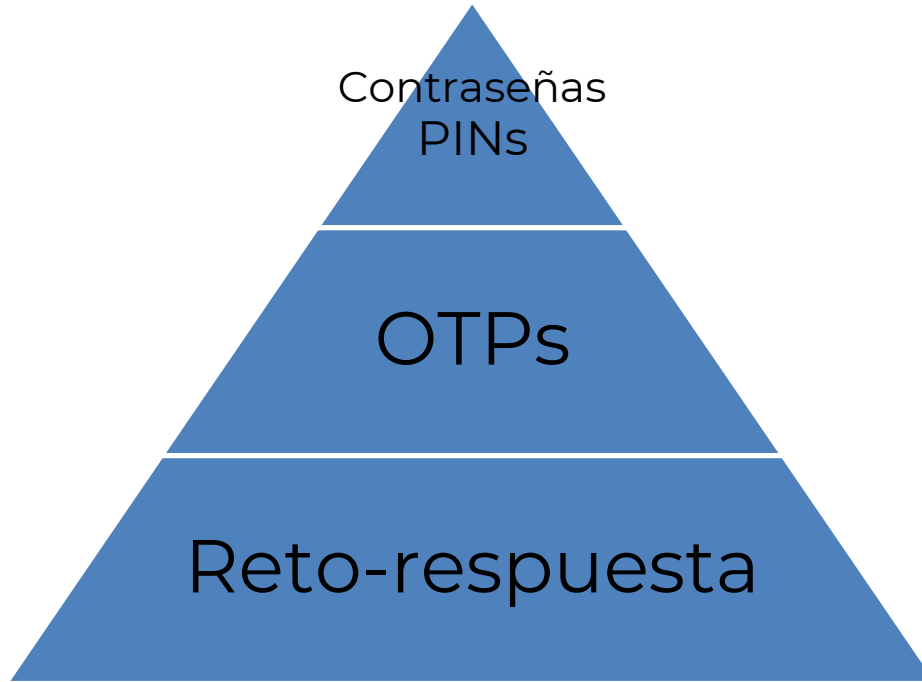
- DNle
- Tarjetas de coordenadas
- Tokens OTPs
- Teléfonos móviles

## Algo que somos

- Firma manuscrita
- Huellas dactilares
- Voz
- Retina



# Autenticación: fortaleza



1 factor (1F) = secreto

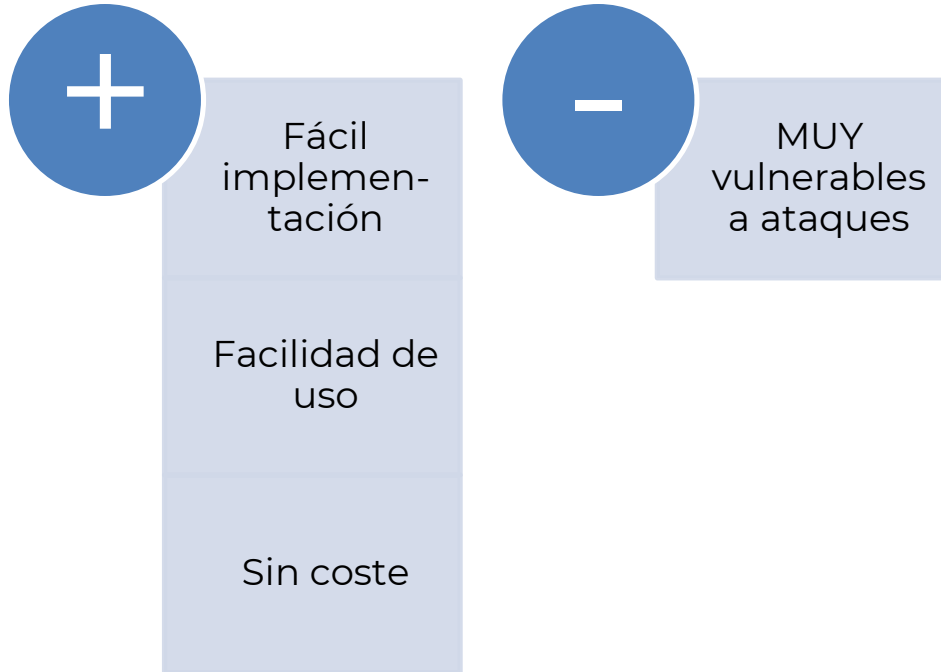
$2F = 1F + \text{token}$   
(Doble factor)

$3F = 2F + \text{rasgo biométrico}$   
(Diffie-Hellman (TLS))

# Contraseñas

- Ambas partes, y solo ellas, conocen un **secreto compartido**
- Primer método, y aún el más utilizado de largo

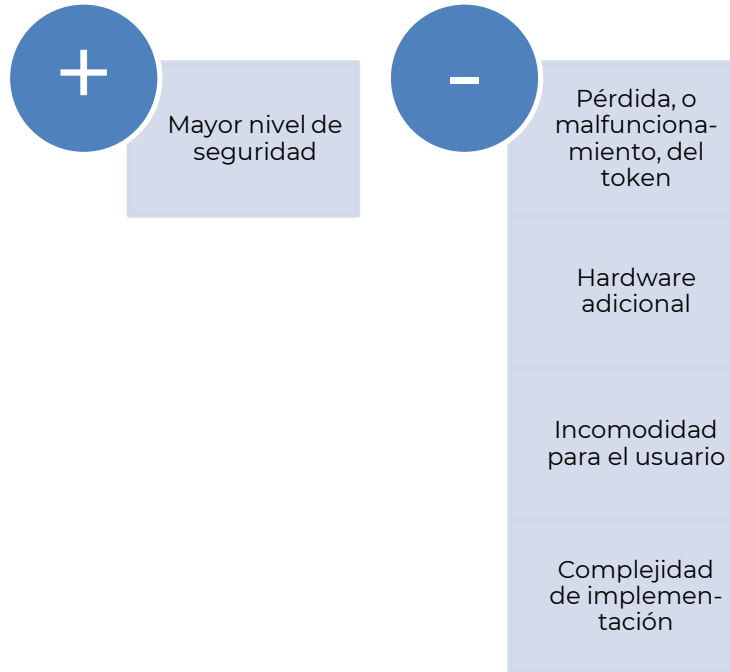
# Contraseñas



## Sistemas 2F

- El usuario debe demostrar, además del conocimiento de un secreto, la posesión de un **token**

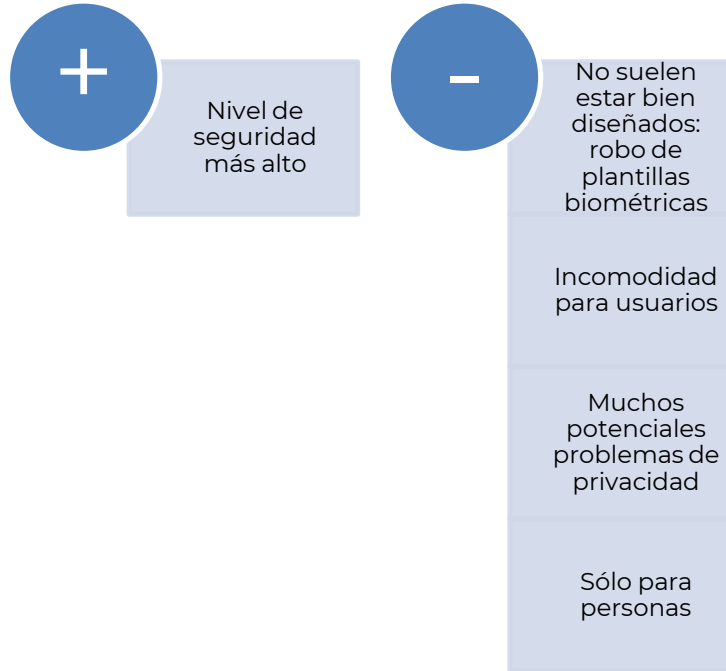
# Sistemas 2F



# Sistemas biométricos

- Utilizan un rasgo biométrico para la identificación del usuario

# Sistemas biométricos



# Contraseñas



# Autenticación por contraseña

- Una de las principales amenazas en toda aplicación Web, sistema o plataforma

## ¿Qué podemos hacer?

- A pesar de los problemas, las contraseñas tienen una larga vida por delante:
  - Son sencillas, baratas y cómodas
  - No existen métodos **cómodos, universales o ubicuos.**

¿Qué podemos hacer?

# Contramedidas

Cliente

Servidor

Medidores de  
fortaleza

Políticas

Almacenamiento  
seguro

# Política de contraseñas

- Típicamente establece:
  - Longitud mínima
  - Frecuencia de cambio
  - Complejidad obligatoria
  - Número permitido de intentos

## Política de contraseñas

- **Complejidad:** al menos 2 números, al menos 2 mayúsculas, al menos 8 caracteres
- **Expiración:** mensual
- **Límite intentos fallidos:** 3

## Política de contraseñas

- Con esas restricciones, las consecuencias suelen ser:
  - Post-it en el monitor
  - Variaciones: Password1, Password2
- Puede ser peor el remedio que la enfermedad

# Política de contraseñas

- Proponer buenos hábitos de generación de contraseñas

Passphrases

P.A.O.

Patrones del  
teclado

Uso de  
gestores de  
contraseñas

# Passphrases

- Componer la contraseña con la primera letra de cada palabra de...
  - Una canción  
“dcsComillbtfy”
  - Frase favorita de una película  
“temspclelyp1982s”
  - ...



## P.A.O.

- Nuestro cerebro recuerda mejor imágenes visuales en escenarios poco habituales
- Método **P**ersona-**A**cción-**O**bjeto
  1. Selecciona un lugar: “Puerta del Sol”
  2. Una persona: “Beyoncé”
  3. Una acción (aleatoria): “cocinar”

## P.A.O.

- Método **P**ersona-**A**cción-**O**bjeto
- 4. Selecciona un objeto: “chincheta”

“Beyoncé cocinando una chincheta  
en la Puerta del Sol”



beycocchipursol

# Diceware

- Variación del método anterior, basado en el uso puro del azar mecánico (dado)
- ✓ Sencillo, seguro, fáciles de recordar
- ✗ Largas al teclear

# Patrones de teclado

- Fáciles de recorder gracias a la memoria muscular
- ¡No triviales!



# Patrones de teclado



## Uso de gestores de contraseñas

- **Ideal:** contraseñas largas, aleatorias, complejas y diferentes para cada servicio
- Imposible de mantener en la práctica
- **Solución:** gestores de contraseñas

## Uso de gestores de contraseñas

- Permiten mantener una cartera de contraseñas MUY complejas y diferentes para cada servicio
- Ejemplos:  
LastPass, KeePass, 1Password

# Uso de gestores de contraseñas

- ¿Punto único de fallo?
- ¿Atado a un proveedor?
- Sí, pero es, desde luego, la solución menos mala



## Hábitos a evitar

- Utilizar una palabra común como semilla:  
“ronaldo”
- Poner en mayúscula la primera letra:  
“Ronaldo”
- Añadir un número, seguramente 1 o 2, al final: “Ronaldo1”
- Añadir algún símbolo habitual (!, @, #, %) al final: “Ronaldo1!”

# Política de contraseñas

- Sirve básicamente para:
  - Concienciar a los usuarios de la importancia de la seguridad y de los peligros existentes.
  - Descargar nuestra responsabilidad en caso de producirse una intrusión por la vulneración de una contraseña

# Ataques al sistema de autenticación

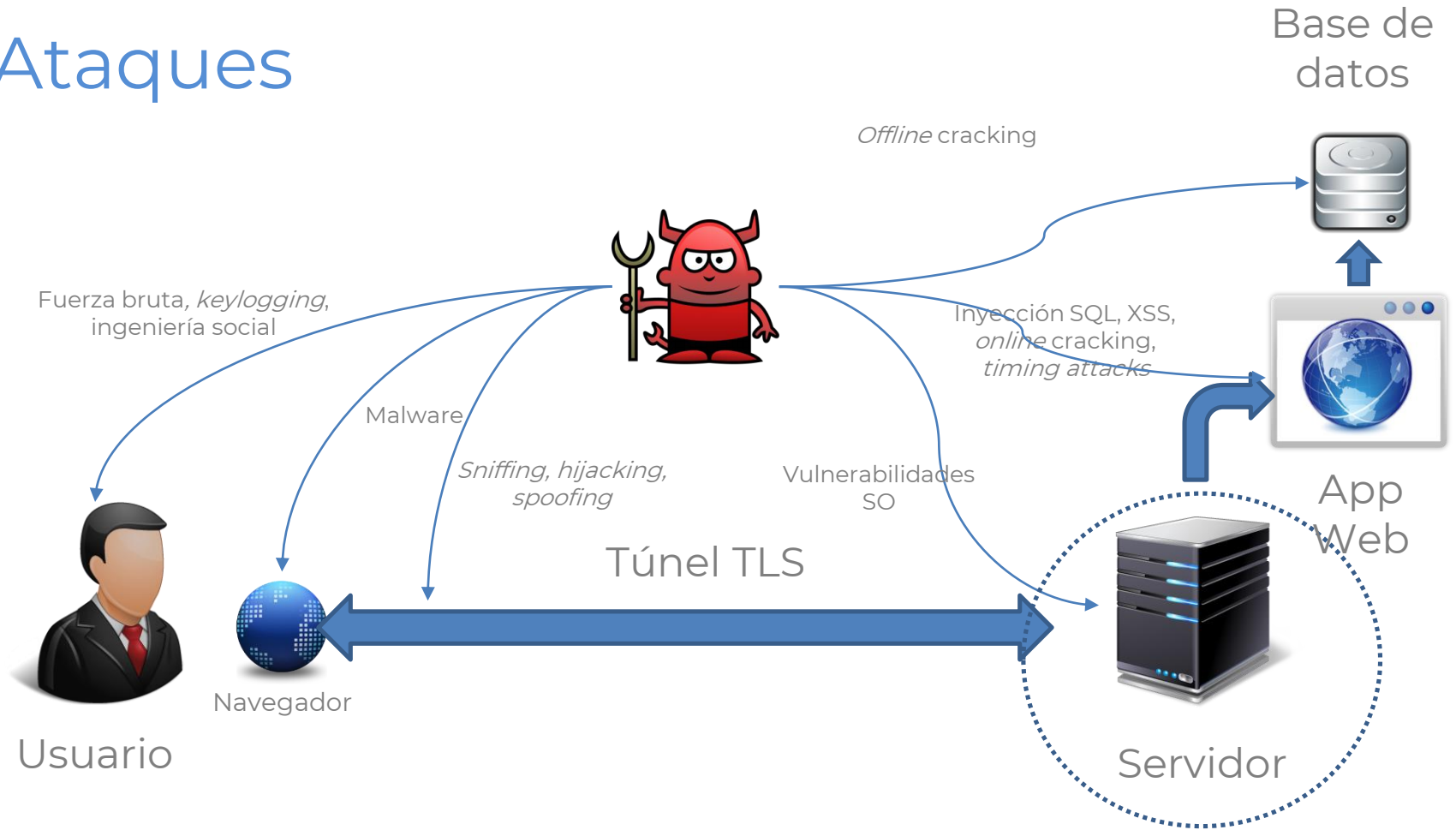
## Ejercicio MOD3\_POLICY

- Tiempo de realización: 20-30 minutos
- Individual

# Contraseñas

## Ataques

# Ataques

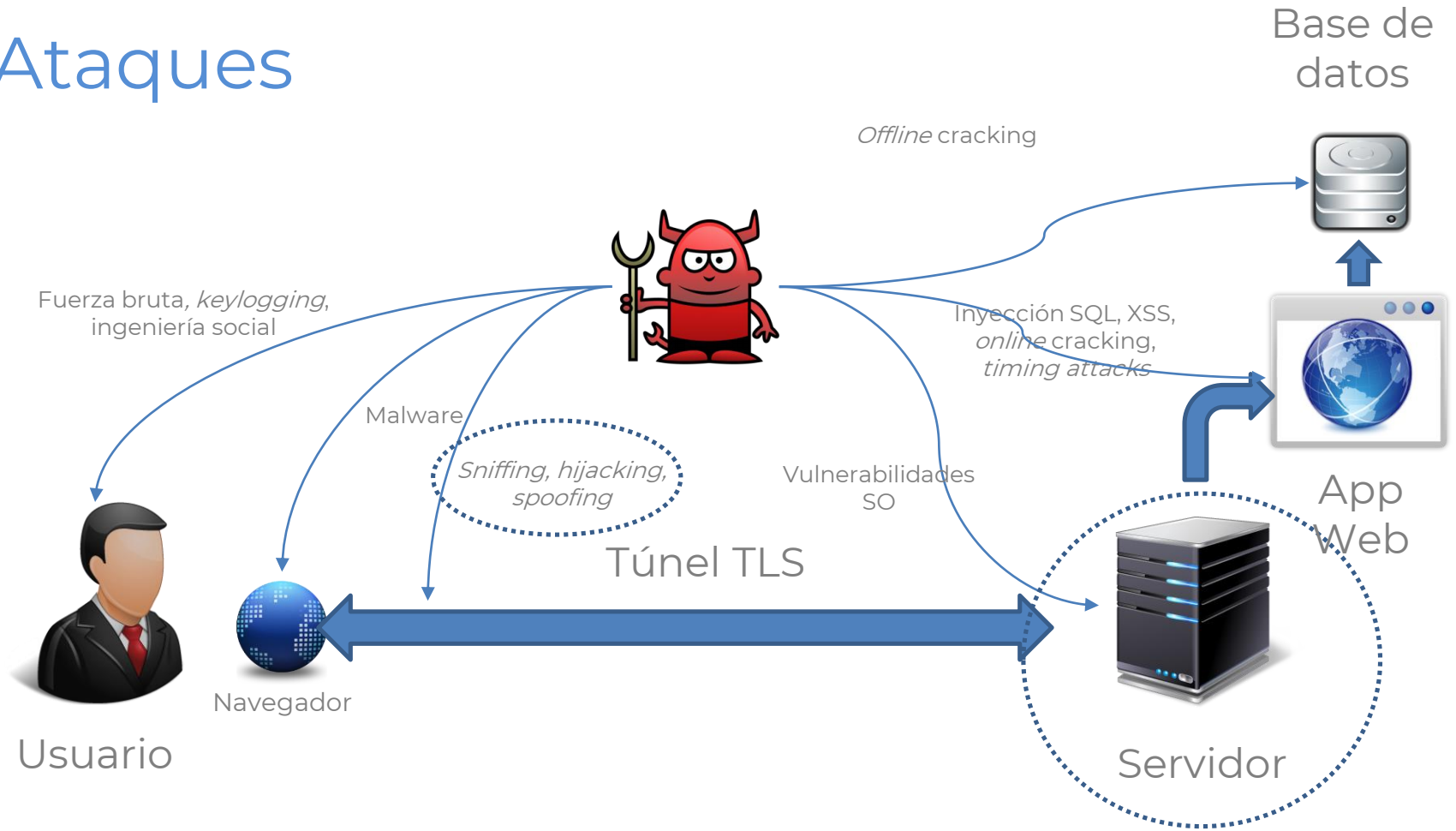


# Ataques al servidor

- Ataques “tradicionales” contra el SO y servicios del servidor

SO	Aplicaciones
<ul style="list-style-type: none"><li>• Buffer overflow</li><li>• Head overflow</li><li>• Errores de configuración</li></ul>	<ul style="list-style-type: none"><li>• Inyección de código SQL, LPAD, comandos</li><li>• XSS</li><li>• Ataques a las APIs</li></ul>

# Ataques



# Ataques a la red

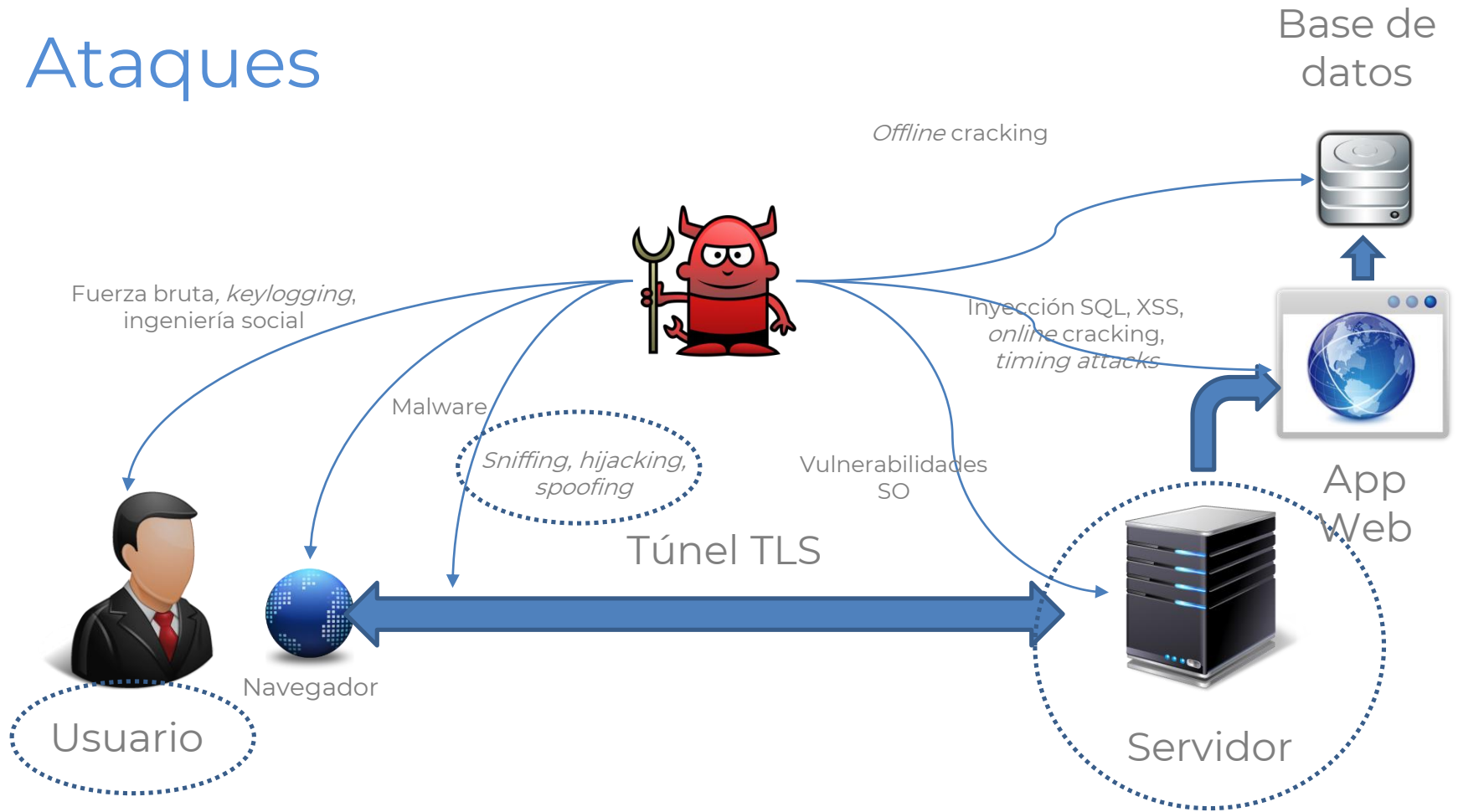
- Ataques contra la comunicación cliente – servidor.
- ¡Pueden funcionar incluso con túneles TLS!



# Ataques a la red

- **Confidencialidad:** Obtener el token del usuario en su transmisión por la red
  - Sniffing
- **Integridad:** modificación de la comunicación cliente-servidor
  - ARP Spoofing, MITM
- **Autenticidad:** suplantar la identidad del usuario una vez autenticado y autorizado
  - Hijacking, Spoofing

# Ataques



# Ataques al cliente

- Ingeniería social: manipulación de personas
  - Ataques de *phishing*
  - Ataque del CEO

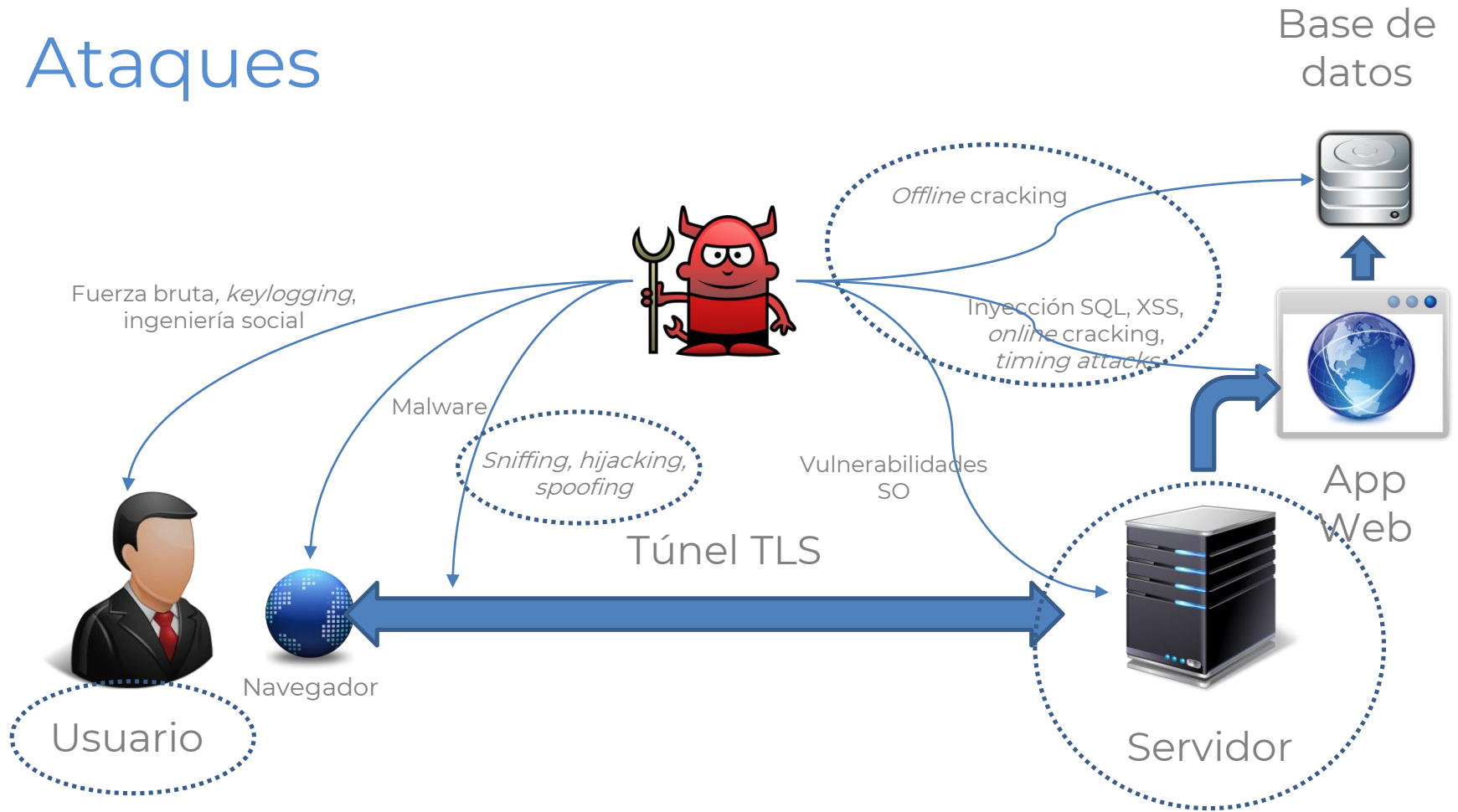
# Ataques al cliente

- (Digital) Trashing: búsqueda de información útil para el atacante
  - Post-it en el monitor o en la mesa, notas en un cuaderno
  - LinkedIn, Facebook, etc...

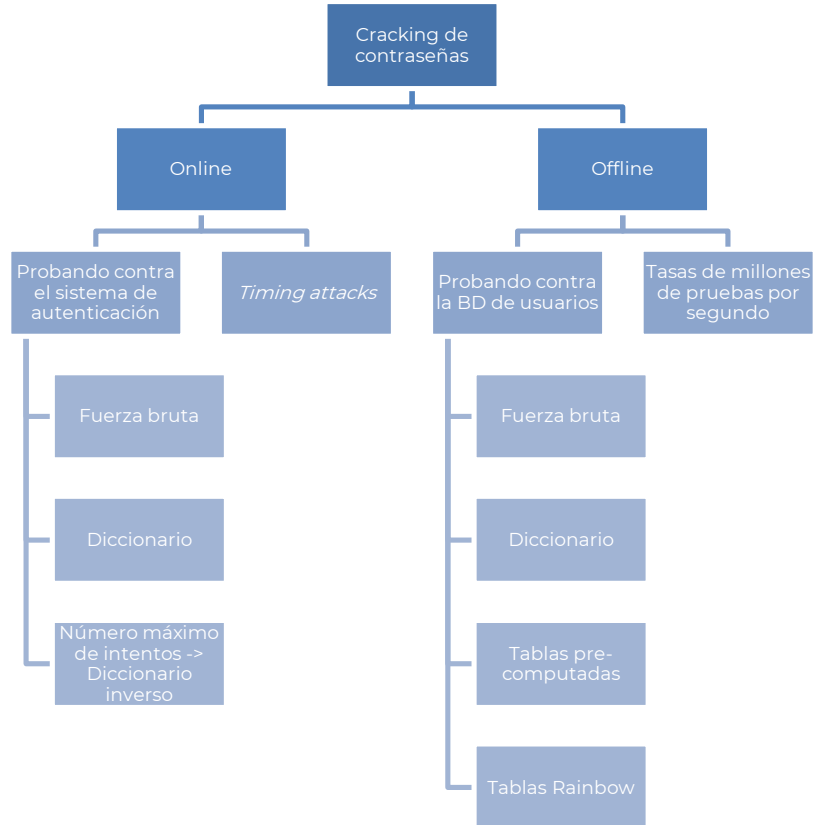
# Ataques al cliente

- **Shoulder Surfing:** “surfear” por encima del hombro del usuario cuando éste escribe su contraseña
  - Cajeros: cámaras para grabar las pulsaciones del PIN

# Ataques



# Cracking de contraseñas



# Espacio de contraseñas

- Los humanos eligen muy malas contraseñas → se puede reducir MUCHO el espacio de búsqueda:
  - Passwords candidatos
  - Diccionario
  - Diccionario con reglas
  - Fuerza bruta
  - Fuerza bruta con reglas



# Uso de diccionarios

- Diccionarios personalizados
  - Nombres de familiares: padres, hijos...
  - Nombre de la novia
  - Nombre de mascotas
  - Fecha de nacimiento
  - Fecha de boda
- ¡Verificar variaciones!
  - Anteponer o postponer números

# Diccionario con reglas

- Aplicar reglas estáticas contra las palabras del diccionario:
  - Añadir o preceder la palabra de un número
  - Capitalizar primera letra
  - Sustituir a por 4, e por 3, etc
  - Invertir la palabra

# Fuerza bruta con reglas

- Explorar sistemáticamente el espacio, pero con reglas que lo reduzcan:
  - Contraseña numérica
  - Solo letras minúsculas
  - Contraseña compuesta por una palabra concatenada consigo misma...

# Contraseñas

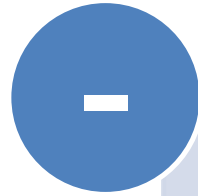
## Ataques online

# Ataques online



Siempre  
son  
posibles

SSL puede  
ser un  
problema  
para su  
detección



Lentitud

Número  
máximo de  
intentos →  
diccionario  
inverso

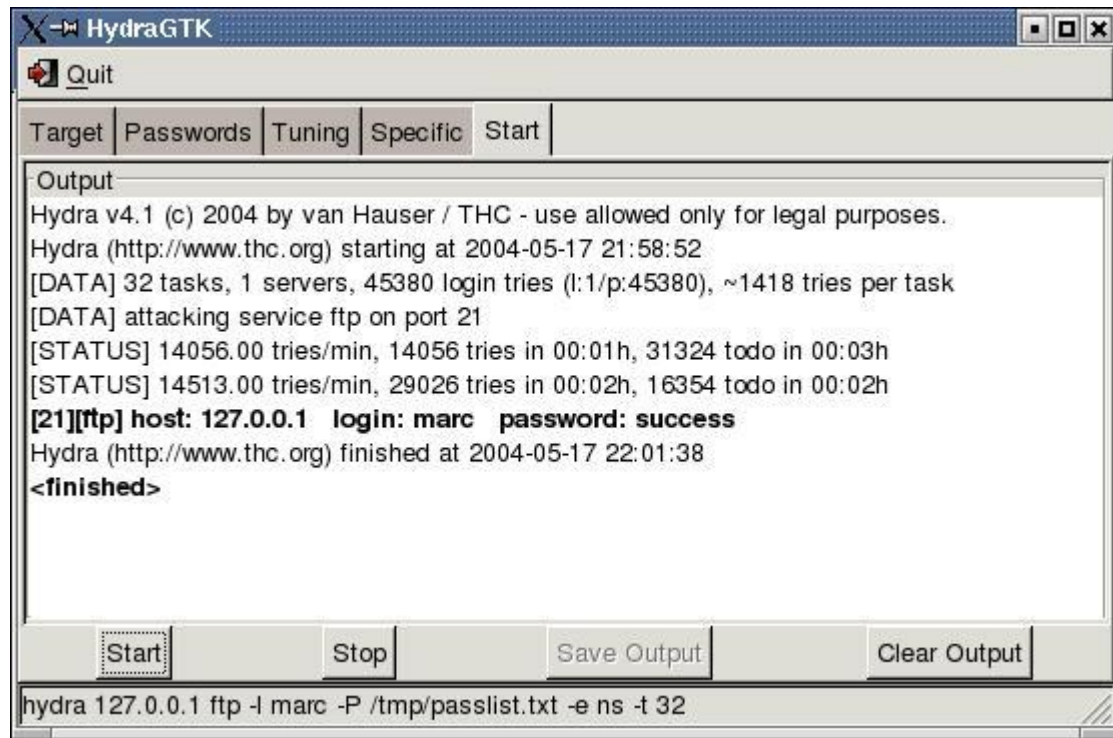
Ataque  
“ruidoso”

# Ataques al sistema de autenticación

## Hydra

Herramienta de referencia

Funciona contra más de 50 protocolos



# Ataques al sistema de autenticación

## Ejercicio MOD3\_HYDRA

- Tiempo de realización: 30 minutos
- Individual

## Ataques de diccionario inverso

- Fijar la contraseña y variar el usuario
- ¿Cuántos usuarios pueden tener como contraseña 'ronaldo'?



# Ataques online: salvaguardas

- Limitar el número de intentos fallidos:
  - Bloquear cuentas tras X intentos fallidos
  - Retrasar unos minutos tras X intentos fallidos
- Activar CAPTCHA cuando se detecte un ataque sistemático

## Ataques online: salvaguardas

- NO usar contraseñas por defecto para los nuevos usuarios
- Distribuir las nuevos contraseñas por un medio seguro, NO por email

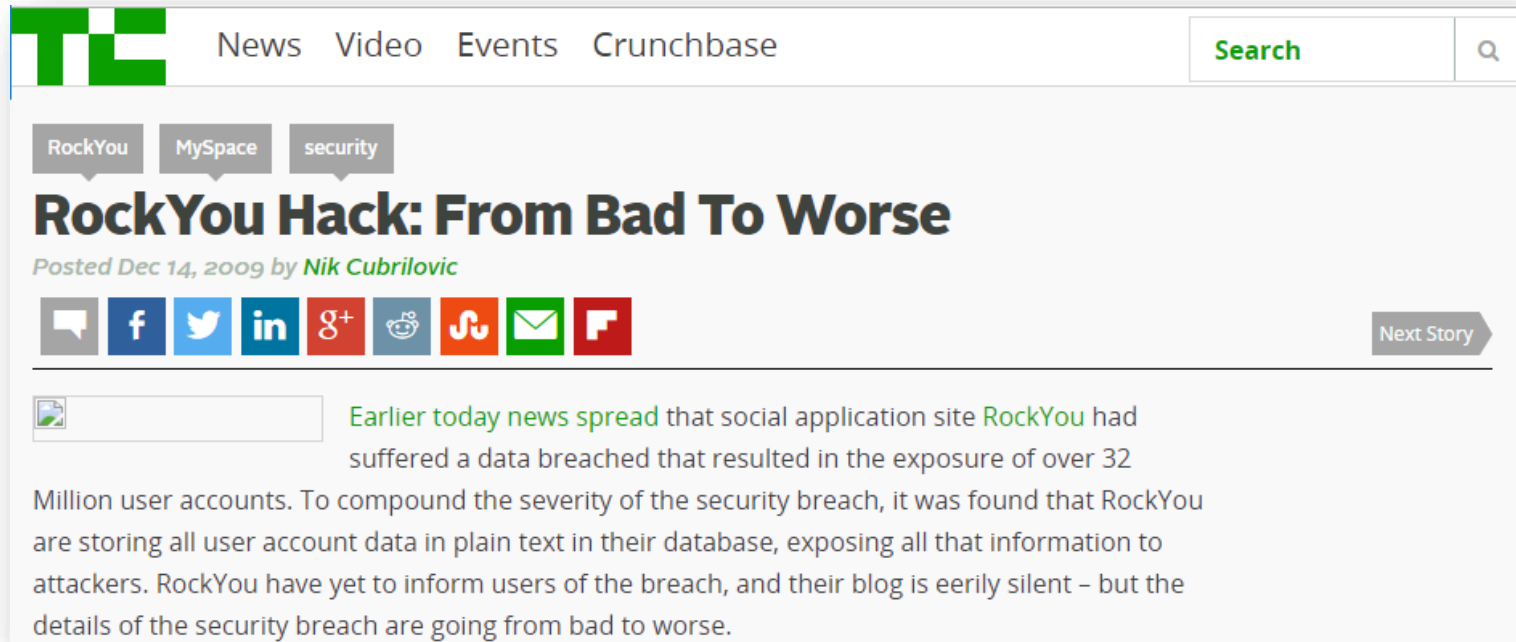
# Contraseñas

Ataques offline

# ¿Cómo guardamos la contraseña en una BD?

- Obviamente, NO en claro...¿no?

# ¿Cómo guardamos la contraseña en una BD?



The screenshot shows a TechCrunch article page. At the top is the TechCrunch logo and navigation links for News, Video, Events, and Crunchbase. A search bar is located on the right. Below the navigation is a category bar with 'RockYou', 'MySpace', and 'security'. The article title is 'RockYou Hack: From Bad To Worse', posted on Dec 14, 2009, by Nik Cubrilovic. A row of social media sharing icons (Twitter, Facebook, LinkedIn, Google+, Reddit, StumbleUpon, Email, Print) is displayed. A 'Next Story' button is on the right. The article text begins with a placeholder image and states: 'Earlier today news spread that social application site RockYou had suffered a data breach that resulted in the exposure of over 32 Million user accounts. To compound the severity of the security breach, it was found that RockYou are storing all user account data in plain text in their database, exposing all that information to attackers. RockYou have yet to inform users of the breach, and their blog is eerily silent – but the details of the security breach are going from bad to worse.'

TechCrunch News Video Events Crunchbase

Search


RockYou MySpace security

## RockYou Hack: From Bad To Worse

Posted Dec 14, 2009 by [Nik Cubrilovic](#)

[Twitter](#) [Facebook](#) [LinkedIn](#) [Google+](#) [Reddit](#) [StumbleUpon](#) [Email](#) [Print](#)

[Next Story](#)

 Earlier today news spread that social application site [RockYou](#) had suffered a data breach that resulted in the exposure of over 32 Million user accounts. To compound the severity of the security breach, it was found that RockYou are storing all user account data in plain text in their database, exposing all that information to attackers. RockYou have yet to inform users of the breach, and their blog is eerily silent – but the details of the security breach are going from bad to worse.

# Robos de BD de usuarios

**The Register**  
*Biting the hand that feeds IT*

DATA CENTRE SOFTWARE SECURITY TRANSFORMATION DEVOPS BUSINESS PERSONAL TECH SCIENCE EMERGENT TECH BOOTNOTES

**Security**

## Dropbox: Leaked DB of 68 million account passwords is real

Login details are strongly hashed and date back to 2012



More like this

Dropbox



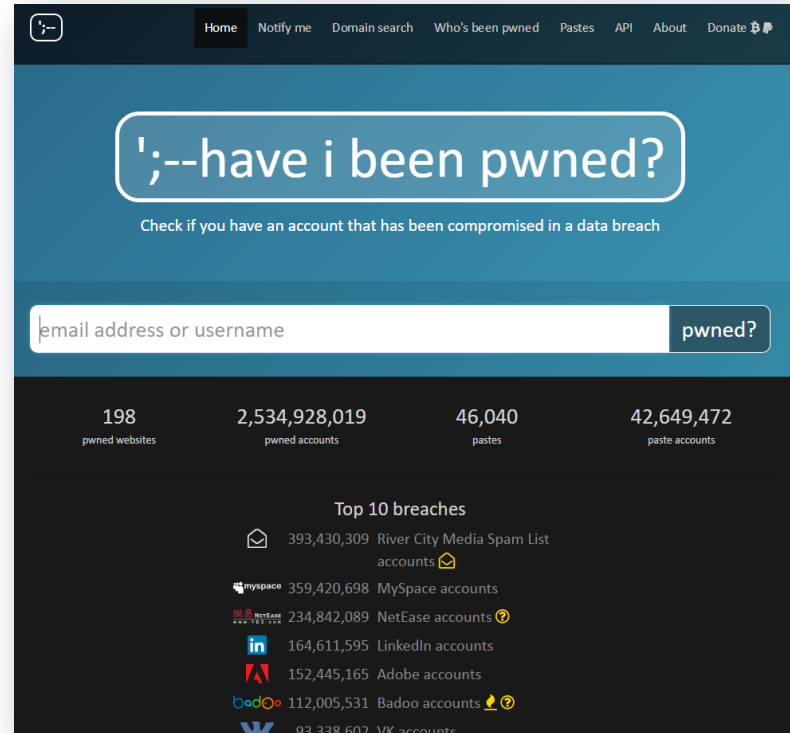
IP Transit  
\$0.20/Mbps



31 Aug 2016 at 01:20 Richard Chirgwin

# Robos de BD de usuarios

<https://haveibeenpwned.com/>

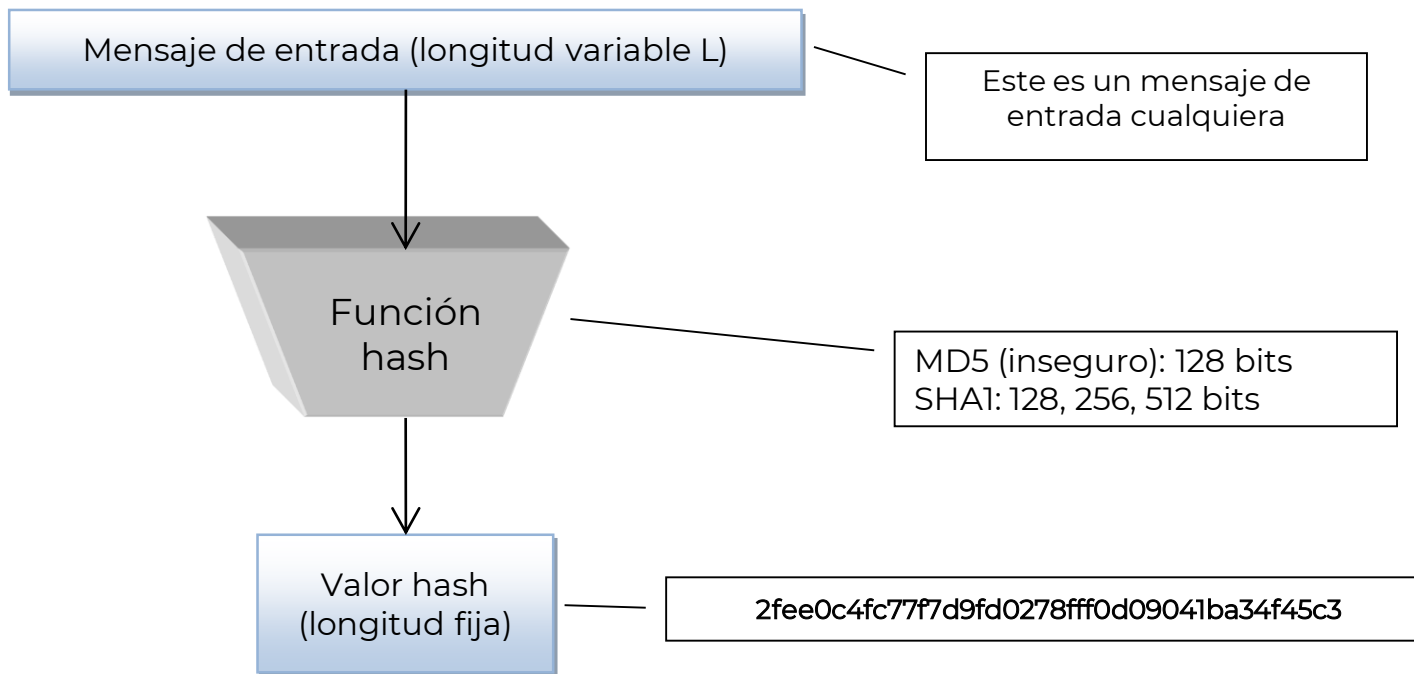


# ¿Cómo guardamos la contraseña en una BD?

- En su lugar se guarda la salida de una función de un único sentido (*función hash*) sobre la contraseña.



# Funciones hash



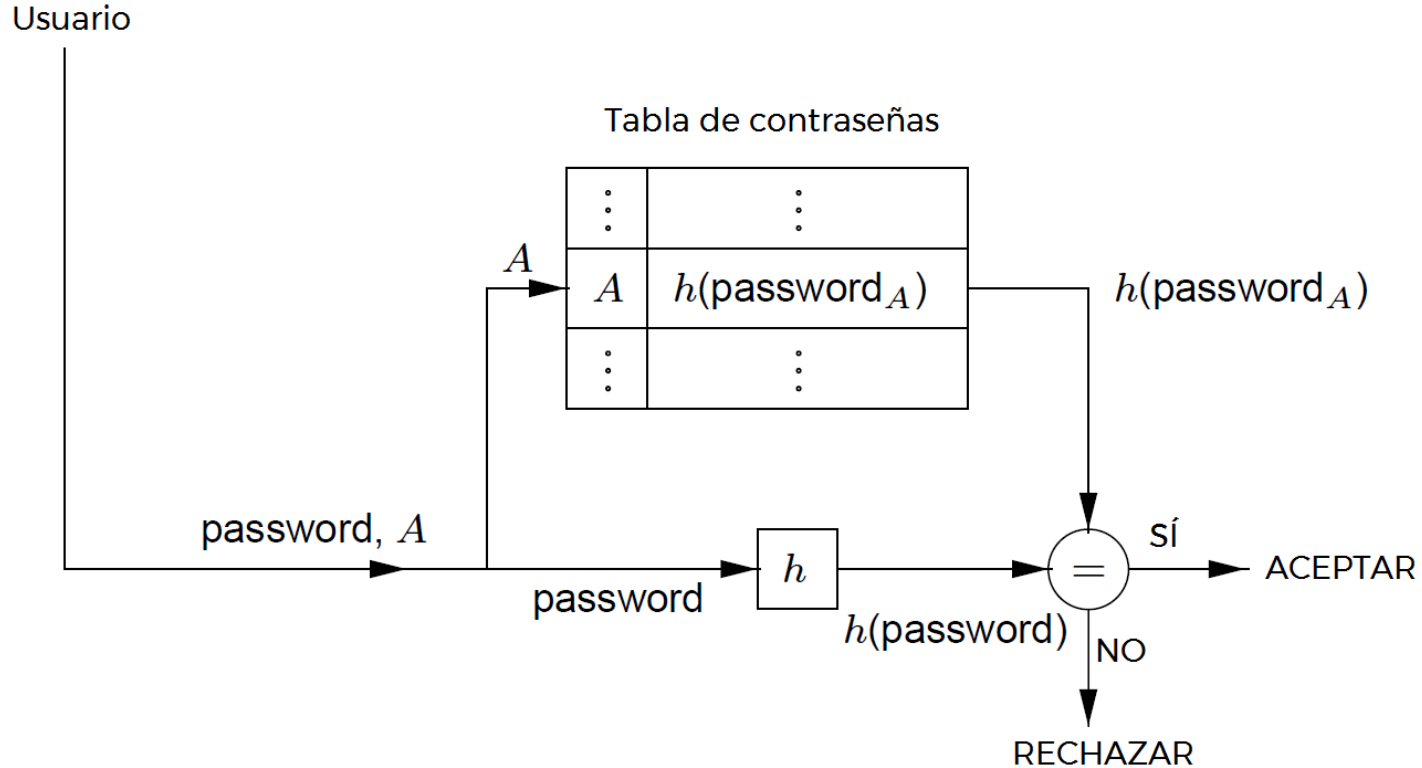
# Funciones hash

Familia	Función	Longitud salida	Estado	Fecha publicación
MD	MD4	128	Insegura	1991
	MD5	128	Insegura	1992
SHA-1	SHA1	160	Insegura	1995
SHA-2	SHA256	256	Segura	2001
	SHA512	512	Segura	2001
SHA-3	SHA3-256	256	Segura	2015
	SHA3-512	512	Segura	2015

# Propiedades

- Hablamos de funciones hash **criptográficas** (NO tablas hash)
- El más mínimo cambio en la entrada produce un hash diferente (efecto avalancha)

# Almacenamiento de contraseñas con hash



## ¿Basta entonces con esto?

- NO, el atacante puede montar un ataque offline con la BD robada
- Toma una posible contraseña, calcula su hash y compara con el almacenado

# Ataques offline

- Ventajas para el atacante
  - Offline
  - No deja logs en el sistema remoto.
  - Extremadamente rápido en realizar las validaciones
  - No hay salvaguardas que limiten el ratio o número total de validaciones

# Fuerza bruta

- Explorar sistemáticamente el espacio de contraseñas estados
- PROBLEMA: el tiempo
- VENTAJA: hay unas combinaciones mucho más probables que otras

# Ataques offline

- ¿Cómo se obtienen los hashes?
  - Habitualmente atacando el servidor y robando directamente la BD



# Tablas pre-computadas

- Pre-calculan todos los hashes posibles, de diccionarios y combinaciones

```
Searching: 5f4dcc3b5aa765d61d8327deb882cf99: FOUND: password5
Searching: 6cbe615c106f422d23669b610b564800: not in database
Searching: 630bf032efe4507f2c57b280995925a9: FOUND: letMEin12
Searching: 386f43fab5d096a7a66d67c8f213e5ec: FOUND: mcdonalds
Searching: d5ec75d5fe70d428685510fae36492d9: FOUND: p@ssw0rd!
```

# Tablas Rainbow

- Enfoque intermedio, que sacrifique velocidad por espacio de almacenamiento.
- Pueden revertir el hash MD5 cualquier contraseña de hasta 8 caracteres inmediatamente.

# Ataques al sistema de autenticación

## Ejercicio MOD3\_CRACKING

- Tiempo de realización: 30 minutos
- Individual

# Contraseñas

## Almacenamiento seguro

# Pongamos un poco de salt...

- Si concatenamos una cadena aleatoria a la contraseña antes de calcular su hash, se dificulta MUCHO todos los ataques anteriores

```
hash("hello") = 2cf24dba5fb0a30e26e83...c1fa7425e73043362938b9824  
hash("hello" + "QxLUF1bgIAdeQX") = 9e209040c863f84a31e71...9fe5ed3b58a75cff2127075ed1  
hash("hello" + "bv5PehSMfV11Cd") = d1d3ec2e6f20fd420d50e...14b8ea157c9e18477aaef226ab
```

# Salt + Hash: generación

1. Se calcula un salt aleatorio para cada contraseña
2. Se hashean juntas:  
$$h = \text{HASH}(\text{salt} \mid \text{contraseña})$$

# Usuario:Hash:Salt

Oscar:2cf24dba5fb0a30e26e83fa7425e73042938b9824:5ed3b58a75

## Salt + Hash: verificación

1. Se recupera el salt del usuario
2. Se recalcula  $h$  con la contraseña proporcionada
3. Se compara el resultado con el almacenado

## Salt + Hash

- El salt no debe ser secreto, pero sí aleatorio (impredecible)



# The **WRONG** way

Reutilizar el salt

- Es común utilizar siempre el mismo valor o generarlo aleatoriamente solo una vez
- CONSECUENCIA: dos usuarios con la misma contraseña tendrán el mismo hash -> habilitamos ataques offline

# The **WRONG** way

Salt muy cortos

- Si el salt es corto, se pueden pre-computar las tablas
- Salt de tres caracteres:

$$95 \cdot 95 \cdot 95 = 95^3 = 857.375 \text{ salts}$$

Si cada tabla ocupa 1MB, necesitaríamos menos de  
1TB = 50€

# The **WRONG** way

## Doble hashing y funciones frikis

- Ejemplos reales de propuestas en foros:

```
md5 (sha1 (password) )
```

```
md5 (md5 (salt) + md5 (password) )
```

```
sha1 (sha1 (password) )
```

```
sha1 (str_rot13 (password + salt) )
```

```
md5 (sha1 (md5 (md5 (password) + sha1 (password) ) +
```

```
md5 (password) ) )
```

# The **WRONG** way

Doble hashing y funciones frikis

- No aportan seguridad real
- Tiempo de cálculo:

$$\text{sha1}(\text{password}) \cong \text{md5}(\text{sha1}(\text{password}))$$

# The **WRONG** way

Doble hashing y funciones frikis

- <https://crackstation.net/hashing-security.htm>

# OK, ¿cuál es la forma correcta?

## Generación

1. Generar un salt largo utilizando un generador de números aleatorios criptográficamente seguro
2. Concatenar el salt a la contraseña y hashear con una función estándar como PBKDF2, bcrypt o bcrypt
3. Guardar salt, hash y número de iteraciones en la BD de usuarios

# OK, ¿cuál es la forma correcta?

## Verificación

1. Recuperar salt y hash de la BD

2. Concatenar salt a la contraseña proporcionada y recalcular hash

3. Comparar resultado con el almacenado en la BD. Si son idénticos, la contraseña es correcta.

# CSPRNG

## Cryptographically Secure Pseudo-Random Generator

---

Plataforma	CSPRNG
PHP	<a href="#"><u>mcrypt_create_iv, openssl_random_pseudo_bytes</u></a>
Java	<a href="#"><u>java.security.SecureRandom</u></a>
Dot NET (C#, VB)	<a href="#"><u>System.Security.Cryptography.RNGCryptoServiceProvider</u></a>
Ruby	<a href="#"><u>SecureRandom</u></a>
Python	<a href="#"><u>os.urandom</u></a>
Perl	<a href="#"><u>Math::Random::Secure</u></a>
C/C++ (Windows API)	<a href="#"><u>CryptGenRandom</u></a>
Cualquier lenguaje en Linux	Lectura de <a href="#"><u>/dev/random</u></a>

---



# Funciones de derivación de claves

## Key Derivation Functions (KDF)

- En general, las KDF se usan para la derivación de claves criptográficas a partir de secretos “débiles”
- Son una especie de “amplificadores” de entropía: no se puede cifrar un texto con una contraseña

# Funciones de derivación de claves

## Key Derivation Functions (KDF)

- Para evitar los ataques, se iteran para hacerlas “lentas”

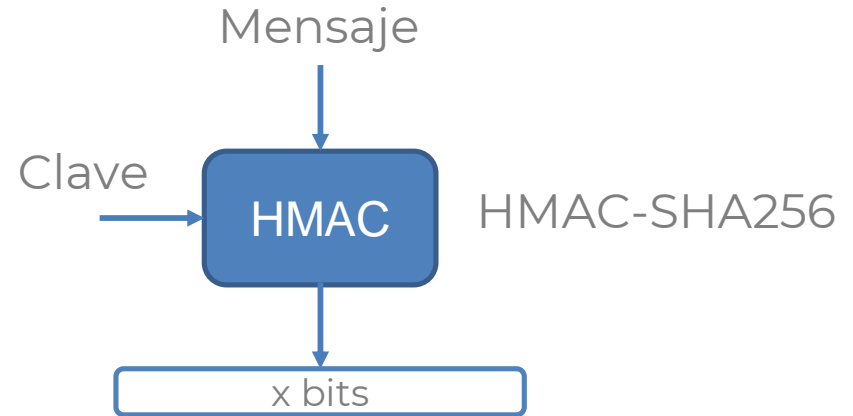
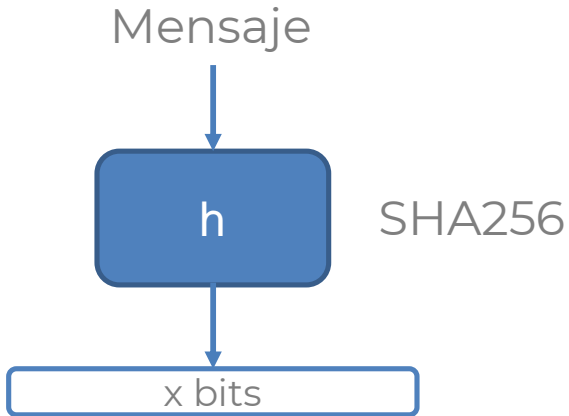
Clave derivada = KDF (contraseña, salt, num\_iteraciones)

- Ejemplos: PBKDF2, scrypt, bcrypt

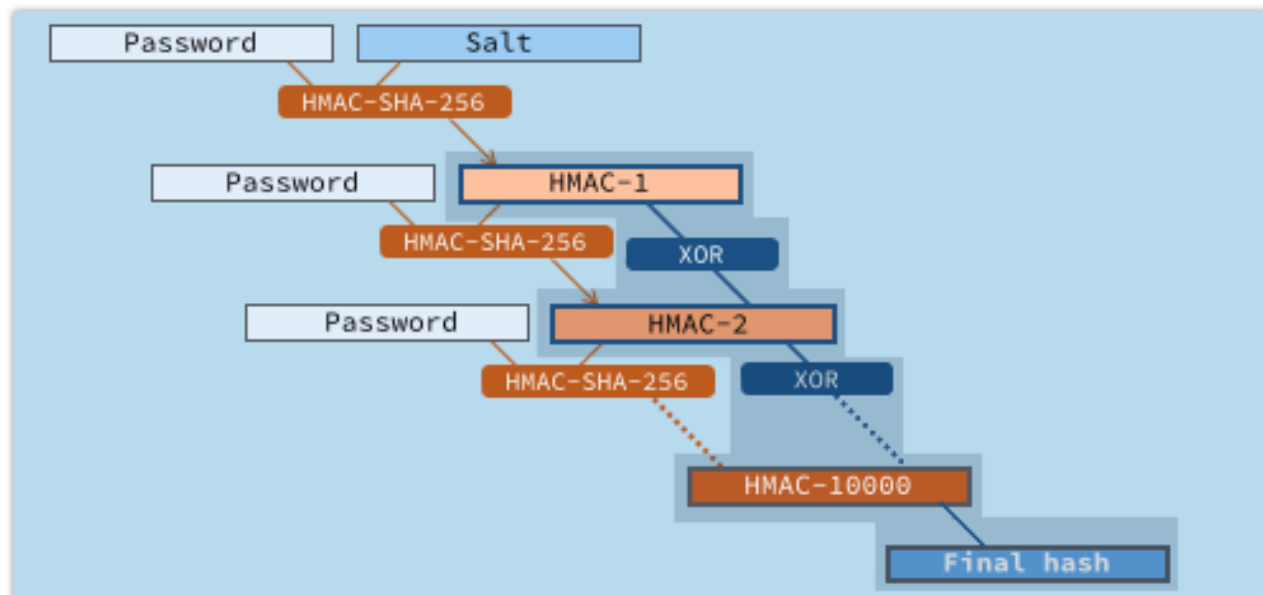
# HMAC

## Hash con clave

- Permite generar “firmas” simétricas



# PBKDF2

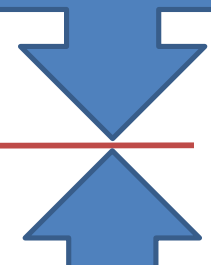


# PBKDF2

## Número de iteraciones

Año	Plataforma	Iteraciones
2000	Estándar	1000
2005	Kerberos	4096
2009	iOS 3	2000
2010	iOS 4	10000
2011	LastPass	100000

Mínimo estricto,  
10000 iteraciones



Idealmente, “tantas  
como te permita tu  
servidor sin afectar a su  
rendimiento”

# PBKDF2

## Tiempos de ataque con GPU

Complejidad contraseña	Entropía estimada (bits)	1000 iteraciones	10000 iteraciones
> 8 caracteres, una mayúscula, un símbolo y un número	33	4 horas 46 minutos	47 horas
8 letras minúsculas aleatorias	37	12 horas	5 días
8 caracteres aleatorios	45	123 días	3 años 5 meses
4 palabras Diceware	52	325 años	3250 años

# PBKDF2

## Uso desde Java

```
public static String createHash(char[] password)
    throws NoSuchAlgorithmException, InvalidKeySpecException
{
    // Generate a random salt
    SecureRandom random = new SecureRandom();
    byte[] salt = new byte[SALT_BYTES];
    random.nextBytes(salt);

    // Hash the password
    byte[] hash = pbkdf2(password, salt, PBKDF2_ITERATIONS, HASH_BYTES);
    // format iterations:salt:hash
    return PBKDF2_ITERATIONS + ":" + toHex(salt) + ":" + toHex(hash);
}
```

# Funciones de derivación de claves

## Otras funciones

Función	Observaciones
PBKDF2	<ul style="list-style-type: none"><li>• Necesita ajustar el número de iteraciones a la mejora de la capacidad computacional media</li><li>• Pueden ser muy eficientemente implementadas en ASICs específicas</li></ul>
bcrypt	Diseñadas para necesitar mucha memoria RAM en su computación, lo que reduce su vulnerabilidad a ataques basados en ASICs y GPUs
scrypt	



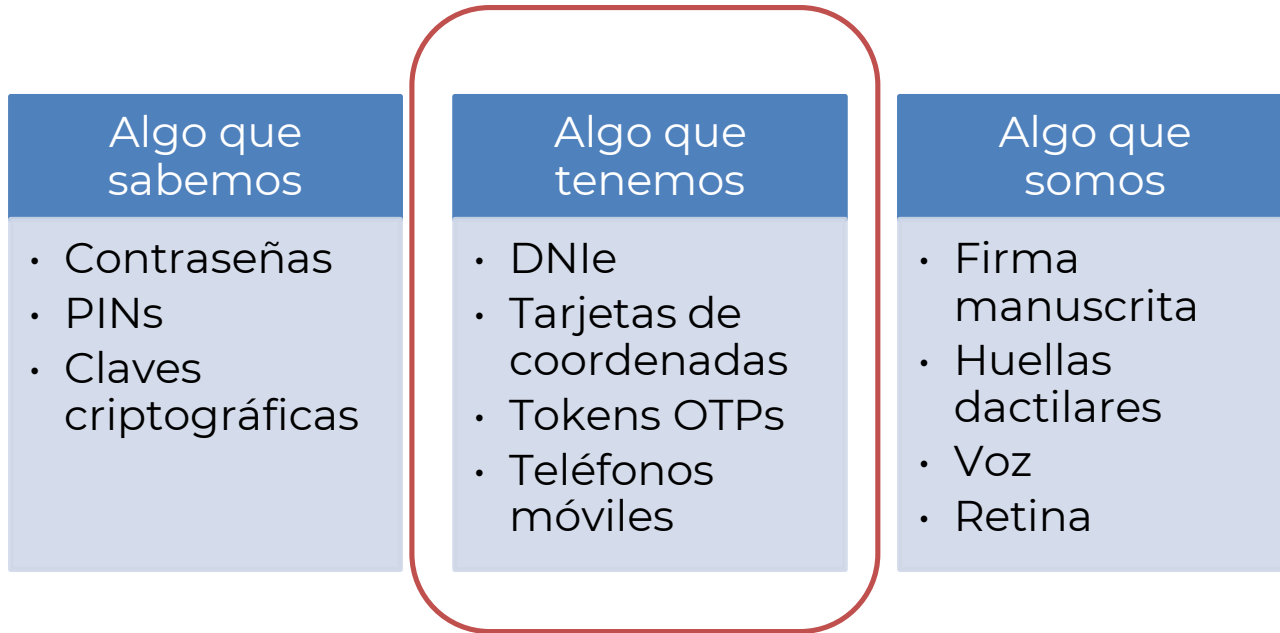
# PBKDF2

## Resumen de uso

- Utiliza un salt de 64 bits único para cada contraseña
- En lugar de SHA1, mejor SHA256 o SHA512
- Al menos 10.000 iteraciones, más si el servidor lo permite

# Sistemas de doble factor

# Autenticación



## Doble factor

- Pretenden superar los problemas de las contraseñas
- Se introduce un nuevo “secreto” fuera de banda (del mundo digital), típicamente un objeto físico

# Tarjetas de coordenadas

Los delincuentes se adaptan a todo: piden una serie de coordenadas futuras

Por favor confirma 8 coordenates de su tarjeta , 2 de cada linea.

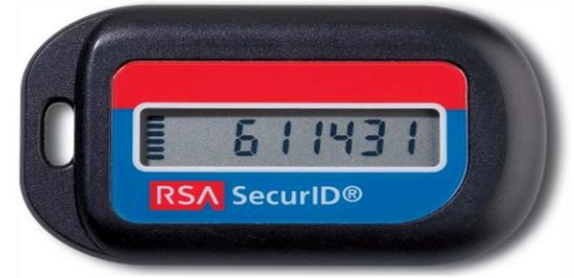
7	*	9
4	5	6
1	2	3
0	Borrar	

	1	2	3	4	5	6	7	8	9	10
A	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
B	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
C	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
D	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

# Tokens OTP

- ✓ Buen nivel de seguridad
- ✗ Incómodo (pérdidas)
- ✗ Coste elevado

¿Y el DNIe?



# SMS

- ✓ Canal fuera de banda
- ✗ Incómodo (retrasos)
- ✗ Coste
- ✗ Vulnerable a ataques
- ✗ Token enviado debe ser diseñado con cuidado

# Ejemplo: doble factor en banca online

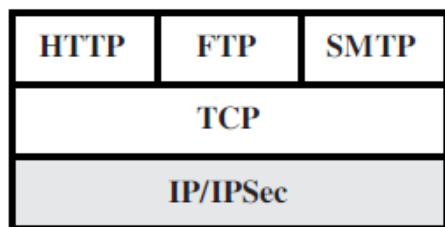
Esquema	Técnica	Ventajas	Inconvenientes
Algo que sabes	Contraseñas	Fáciles de utilizar	Pueden ser capturadas
Algo que posees	Tarjeta de coordenadas	Barata y relativamente sencilla de utilizar	<ul style="list-style-type: none"><li>• Puede perderse u olvidarse</li><li>• Difícil de compartir</li></ul>
	Token hardware	Difícil manipulación	<ul style="list-style-type: none"><li>• Caro</li><li>• Puede perderse u olvidarse</li><li>• Difícil de compartir</li></ul>
	Teléfono móvil	No requiere hardware adicional	<ul style="list-style-type: none"><li>• Puede agotarse la batería o no tener cobertura</li><li>• Coste de la comunicación para el cliente y el banco</li></ul>
Algo que eres	Ritmo de tecleo	<ul style="list-style-type: none"><li>• No requiere hardware adicional</li><li>• Alta aceptabilidad</li></ul>	<ul style="list-style-type: none"><li>• No es fiable</li><li>• Tiempo de registro elevado</li></ul>



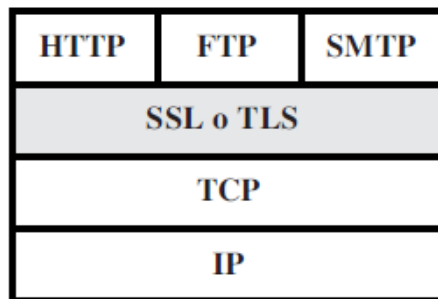
# Arquitecturas de red

## Seguridad

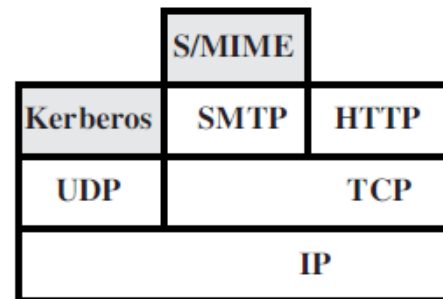
# Pila TCP/IP



(a) Nivel de red



(b) Nivel de transporte



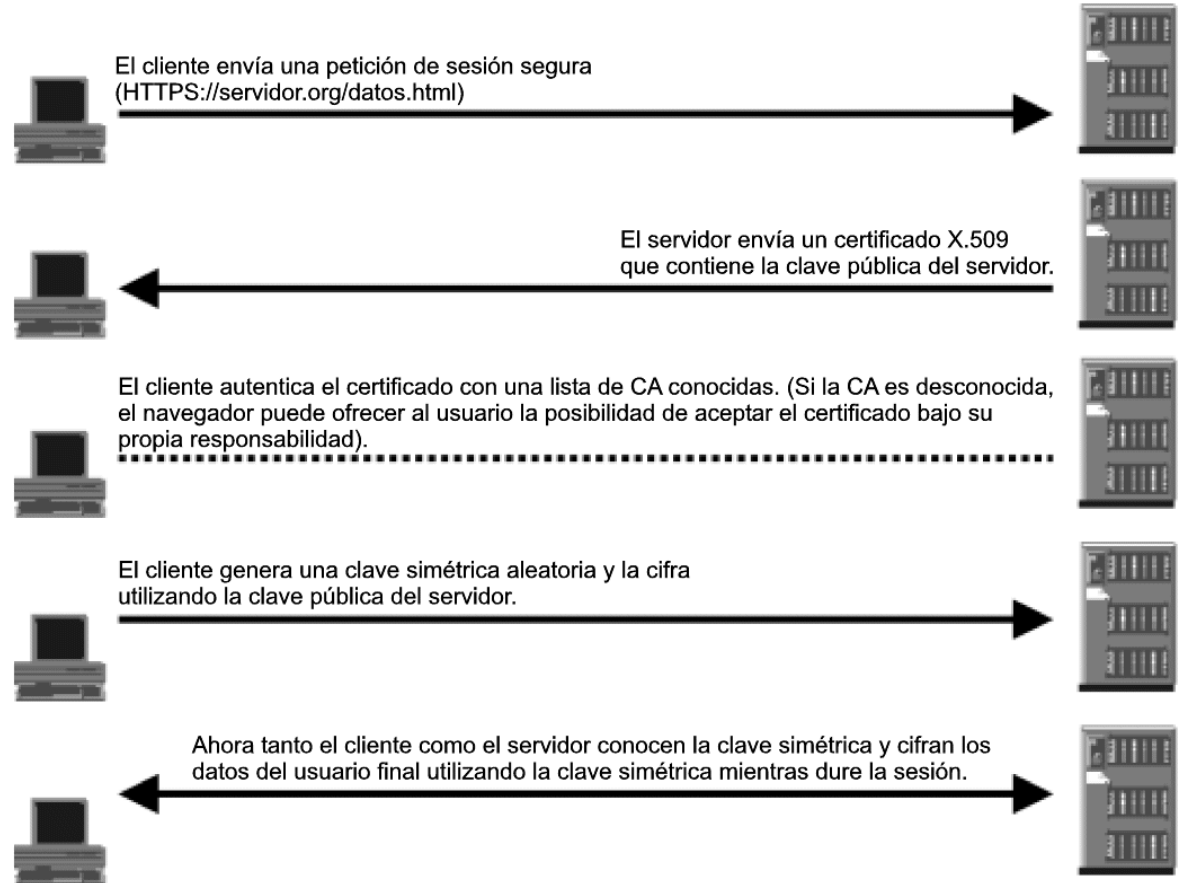
(c) Nivel de aplicación

# SSL, Secure Socket Layer

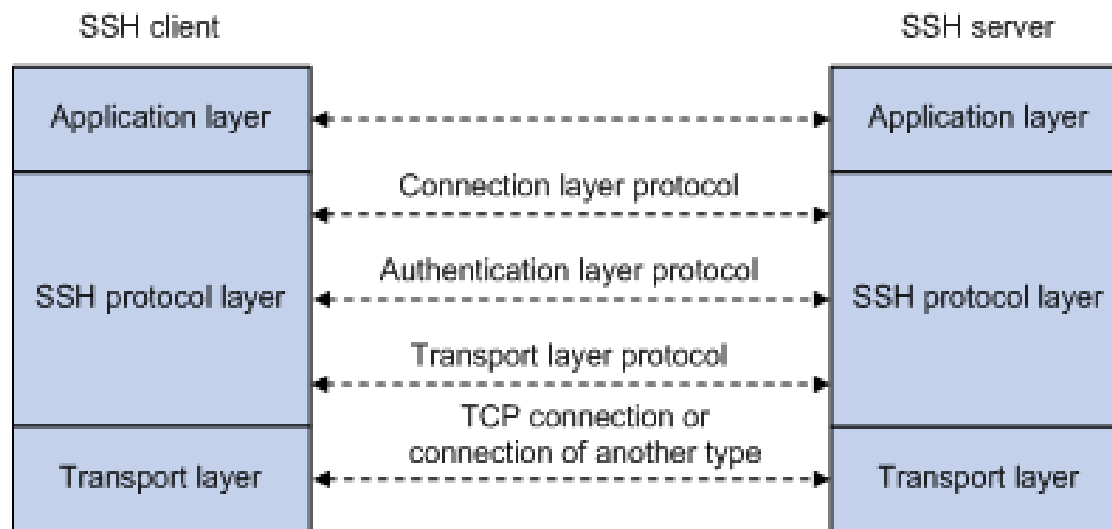
- Desarrollado por Netscape en 1994 (!)
- Multitud de problemas y vulnerabilidades: su sucesor se denomina TLS (Transport Layer Security)
- Ambos protocolos son protocolos de nivel de transporte, por lo que permiten proteger protocolos de nivel de aplicación como Telnet, FTP, SMTP, IMAP o el propio HTTP

# SSL

## Negociación



# SSH



# SSH – Nivel de transporte

Se encarga de:

- La autenticación del servidor.
- Establecimiento de un canal cifrado para garantizar la confidencialidad de la comunicación.
- Comprobación de la integridad de los mensajes.
- Generación de identificador único de sesión.

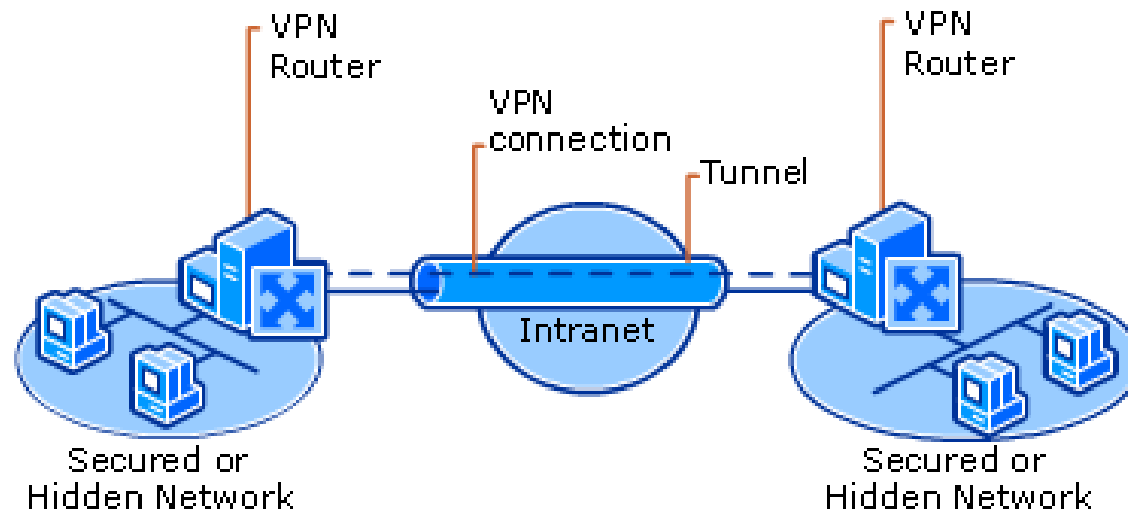
# SSH – Nivel de transporte

**Nivel de autenticación de usuario:** el protocolo ofrece varios mecanismos de autenticación:

- Autenticación basada en el uso de criptografía de clave pública. El usuario deberá disponer, por tanto, de un par de claves pública/privada para poder utilizar esta opción, sin duda la más segura.
- Autenticación tradicional basada en nombre de usuario y contraseña. Esta contraseña se envía, en cualquier caso, cifrada.
- Autenticación basada en la procedencia (dirección IP de origen) de la conexión. Esta opción no es segura y no debería utilizarse en entornos empresariales.

**Nivel de sesión:** se encarga de la asignación de identificadores de sesión, que permiten multiplexar varias comunicaciones distintas a través de un único “túnel” cifrado virtual.

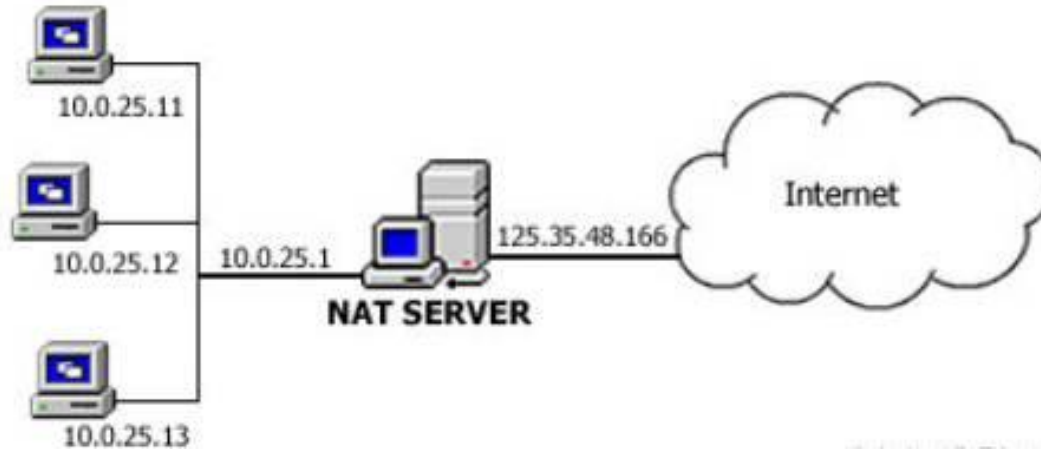
# VPN – Virtual Private Network





# NAT – Network Address Translation

- Su principal aplicación es ocultar el direccionamiento IP interno de una red
- Tres grandes bloques de direcciones IP privadas:
  - De la dirección 10.0.0.0 a la 10.255.255.255
  - De la dirección 172.16.0.0 a la 172.31.255.255
  - De la dirección 192.168.0.0 a las 192.168.255.255



# VPN - Protocolos

- **Point-to-Point Tunneling Protocol (PPTP):**
  - Funciona a nivel de enlace
  - Diseñado para conexiones sencillas, únicas entre cliente y servidor (no permiten conectar dos redes, por ejemplo).
- **Layer 2 Tunneling Protocol (L2TP):**
  - Este protocolo es una combinación del anterior, PPTP, y el antiguo Layer 2 Forwarding Protocol (L2F).
- **IPSec:**
  - Es el estándar más completo, pues permite todo tipo de conexiones, incluyendo túneles que conectan dos redes completas, en lugar de dos computadores.
  - Inconveniente: puede llegar a ser difícil de configurar.

# Criptovirus – (Ransomware)

- Cifra los documentos que encuentra, generalmente ofimáticos, eliminando los originales y dejando un archivo de texto con las instrucciones para recuperarlos

