# Protección y Seguridad

# Definiciones

Protección:

Se refiere al mecanismo para controlar el acceso de programas, procesos o usuarios a los recursos definidos del sistema.

Seguridad:

La seguridad se diferencia de la protección en que ésta es una medida de cómo se preserva la integridad de un sistema y sus datos. La seguridad no solo requiere de un buen sistema de protección sino que además considera el entorno externo dentro del cual opera el sistema. Los problemas de seguridad son esencialmente problemas de administración y no de sistema.

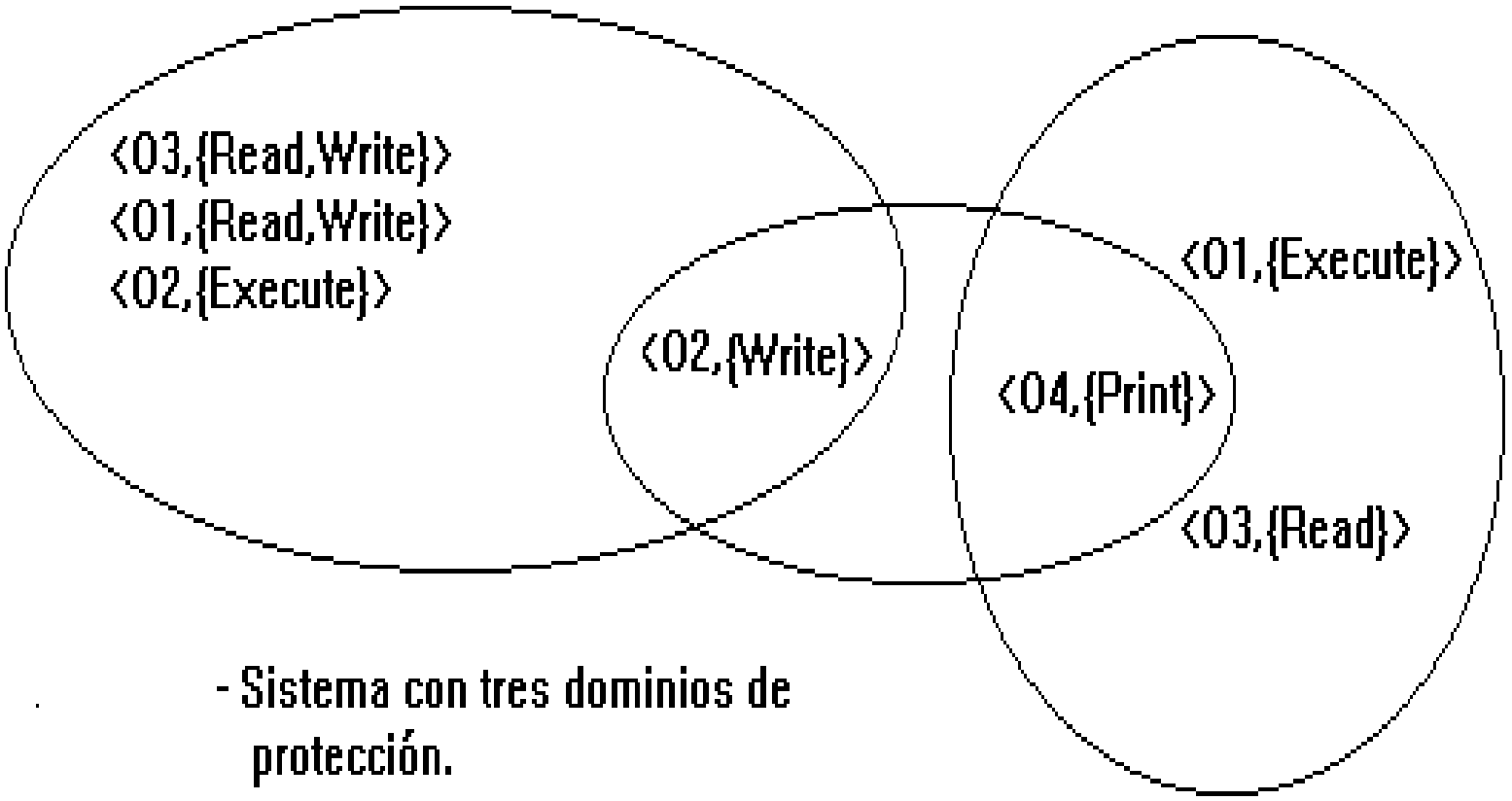
## Elementos de un esquema de protección

DOMINIO: Entidad activa que realiza acciones (un proceso, un usuario, una CPU, etc.)

OBJETO: Entidad pasiva que recibe las acciones realizadas por los dominios (memoria, un periférico, un usuario, etc.)

DERECHO DE ACCESO: Calificación respecto del tipo de acción que puede realizarse (por ej.: lectura, escritura, etc.)

# Dominios de protección y Objetos




- Sistema con tres dominios de protección.

# Implementaciones de esquemas de protección

- MATRIZ DE ACCESOS
- TABLA GLOBAL
- LISTAS DE ACCESO
- LISTAS DE CAPACIDADES
- LOCK/KEY

# Matriz de Accesos

Matriz con los objetos en las columnas y los dominios en las filas. En la intersección de una fila/columna se encuentra el derecho de acceso.

Si el derecho de acceso está en blanco implica que no se puede acceder.

| Objetos / Dominios | Compiladores | Archivos |
|---|---|---|
| Programadores | Ejecutar | Leer |
| System Programmer | Leer/Grabar/Ejecutar | |
| Usuarios | Ejecutar | Leer/Grabar |

# Tabla Global

Un conjunto de uplas (<Dominio, Objeto, Derecho de acceso>) con todos los elementos NO nulos de la Matriz de acceso.

<Programadores, Compilador, Ejecutar>

<Programadores, Archivos, Lectura>

<System Programmer, Compilador, Lectura/ejecución/Grabación>

<Usuarios, Compilador, Ejecutar>

<Usuarios, Archivos, Lectura/grabación>

| Objetos / Dominios | Compiladores | Archivos |
|---|---|---|
| Programadores | Ejecutar | Leer |
| System Programmer | Leer/Grabar/Ejecutar | |
| Usuarios | Ejecutar | Leer/Grabar |

# Listas de Acceso

Se construyen por **objeto** (las LCA - listas de control de acceso- son un subconjunto de éstas)

Compilador < Programadores, Ejecutar>

< System Programmer, Lect./Grab./Ej>

< Usuarios, Ejecutar>

Archivos < Programadores, Lectura>

< Usuarios, Lectura/Grabación>

| Objetos / Dominios | Compiladores | Archivos |
|---|---|---|
| Programadores | Ejecutar | Leer |
| System Programmer | Leer/Grabar/Ejecutar | |
| Usuarios | Ejecutar | Leer/Grabar |

# Listas de Capacidad

| Objetos / Dominios | Compiladores | Archivos |
|---|---|---|
| Programadores | Ejecutar | Leer |
| System Programmer | Leer/Grabar/Ejecutar | |
| Usuarios | Ejecutar | Leer/Grabar |

Se construyen por **dominio** (las LCU - listas de control de usuarios- son un subconjunto de éstas)

| | |
|---|---|
| Programador | < Compilador, Ejecutar> |
| | < Archivos, Leer> |
| Usuarios | < Compilador, Ejecutar> |
| | < Archivos, Leer/Grabar> |
| Syst. Programmer | < Compilador, Lect/Ej/Grab> |

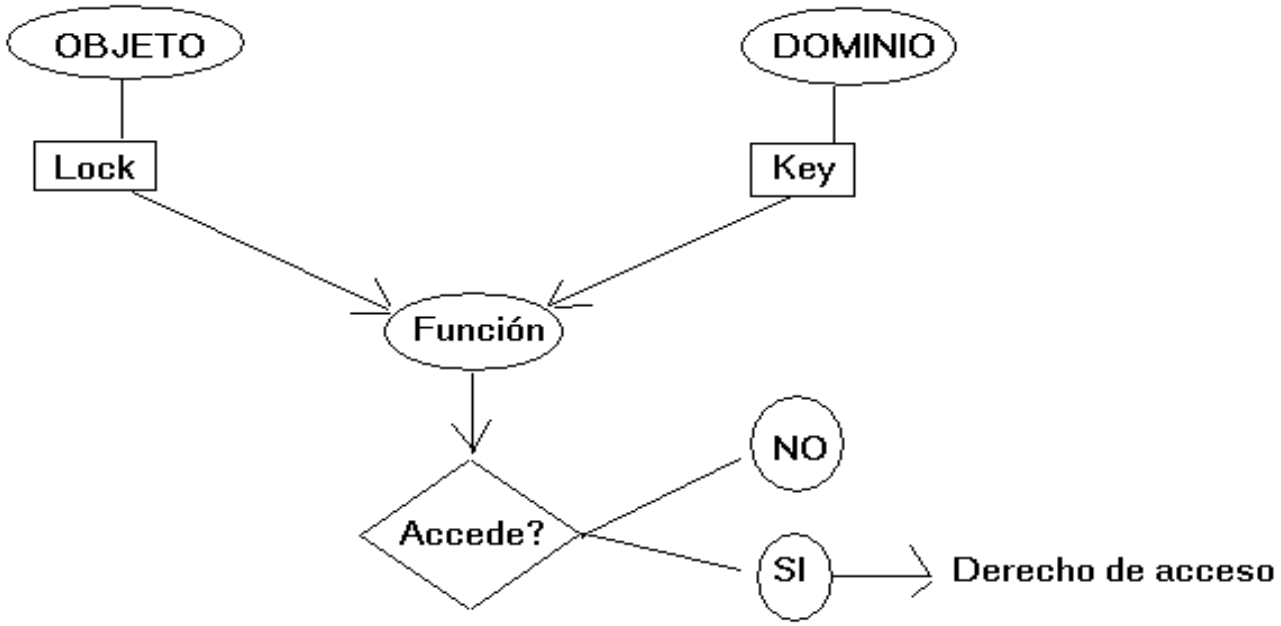
# Listas de Capacidad

La capacidad de llegar al objeto la posee el dominio ya que debe conocer la ubicación del objeto para poder accederlo.

En las LCU recordar que cada usuario debe poseer un puntero al archivo que no le pertenece para poder encontrarlo.

# Mecanismo Lock-Key

Cada Dominio provee su key y cada objeto provee su lock (usualmente son unos pocos bits). Ambos se combinan en una función (usualmente de tipo booleano) y de acuerdo al resultado se otorga o no el acceso.



# Mecanismo Lock-Key (ejemplo)

3 dominios, 2 objetos y la función es la operación AND.
Se accede si el resultado es Falso.

Armemos solo el acceso al objeto Archivos

Programadores < 1 0 > key

Syst. Prog. < 0 1 > key

Usuarios < 1 0 > key

Archivos < 0 1 > lock

| Objetos / Dominios | Compiladores | Archivos |
|---|---|---|
| Programadores | Ejecutar | Leer |
| System Programmer | Leer/Grabar/Ejecutar | |
| Usuarios | Ejecutar | Leer/Grabar |

Aquí solo se construyó el permiso de acceso sin calificarlo, es decir no se sabe aún si accede en modalidad de lectura o grabación -> necesitaré más bits.

# Estructuras Dinámicas

Hasta aquí las estructuras vistas son estáticas. Si queremos que la estructura pueda cambiar en tiempo de ejecución deben agregarse nuevos tipos de derechos de acceso. Veremos algunos, a saber:

- COPY
- OWNER
- SWITCH
- CONTROL

# COPY

Permite que un dominio otorgue el permiso que posee sobre un objeto a otro dominio.

Se denota con un *.

Ejemplo: Permitir que los Programadores puedan grabar archivos

| Objetos / Dominios | Compiladores | Archivos |
|---|---|---|
| Programadores | Ejecutar | Leer |
| System Programmer | Leer/Grabar/Ejecutar | |
| Usuarios | Ejecutar | Leer/Grabar * |

OJO! No puedo otorgar lo que no poseo!!

# OWNER

Otorga la propiedad de un Objeto a un dominio en particular. El dominio puede otorgar permisos sobre ese objeto a otros dominios y además eliminarlos.

Ejemplo: Quitar la autorización dada a los programadores para grabar archivos

| Objetos / Dominios | Compiladores | Archivos |
|---|---|---|
| Programadores | Ejecutar | Leer/Grabar |
| System Programmer | Leer/Grabar/Ejecutar | |
| Usuarios | Ejecutar | Leer/Grabar<br>**OWNER** |

# CONTROL

Permitir que un dominio controle a otro dominio. Se necesita que los dominios ahora sean pasibles de recibir acciones.

Asumiremos que el CONTROL solo permite **eliminar** derechos.

Ejemplo: El System Programmer controla la operatoria total de los otros dominios

| Objetos / Dominios | Compiladores | Archivos | Programadores | Usuarios | Syst Prog |
|---|---|---|---|---|---|
| Programadores | Ejecutar | Leer | | | |
| System Programmer | Leer/Grabar/Ejecutar | | CONTROL | CONTROL | |
| Usuarios | Ejecutar | Leer/Grabar | | | |

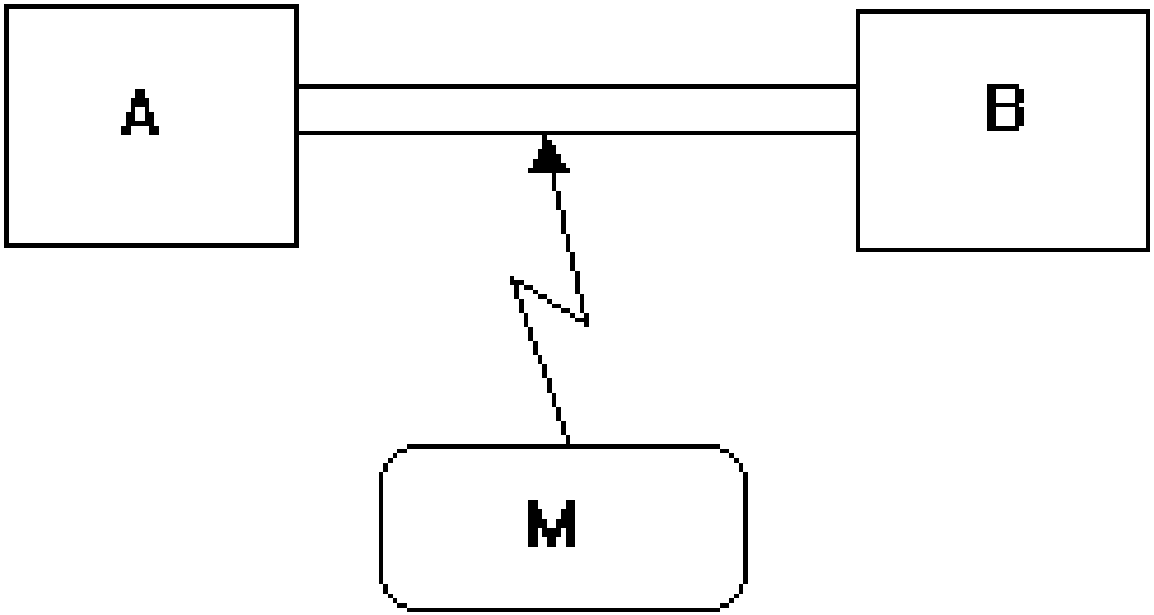
# SWITCH

Permite que un dominio pase a actuar como otro dominio perdiendo sus derechos y asumiendo los derechos del dominio al que se cambió.

Ejemplo: Se desea que los System Programmer puedan probar nuevas aplicaciones desarrolladas para los usuarios para ver si efectivamente funcionan bien.

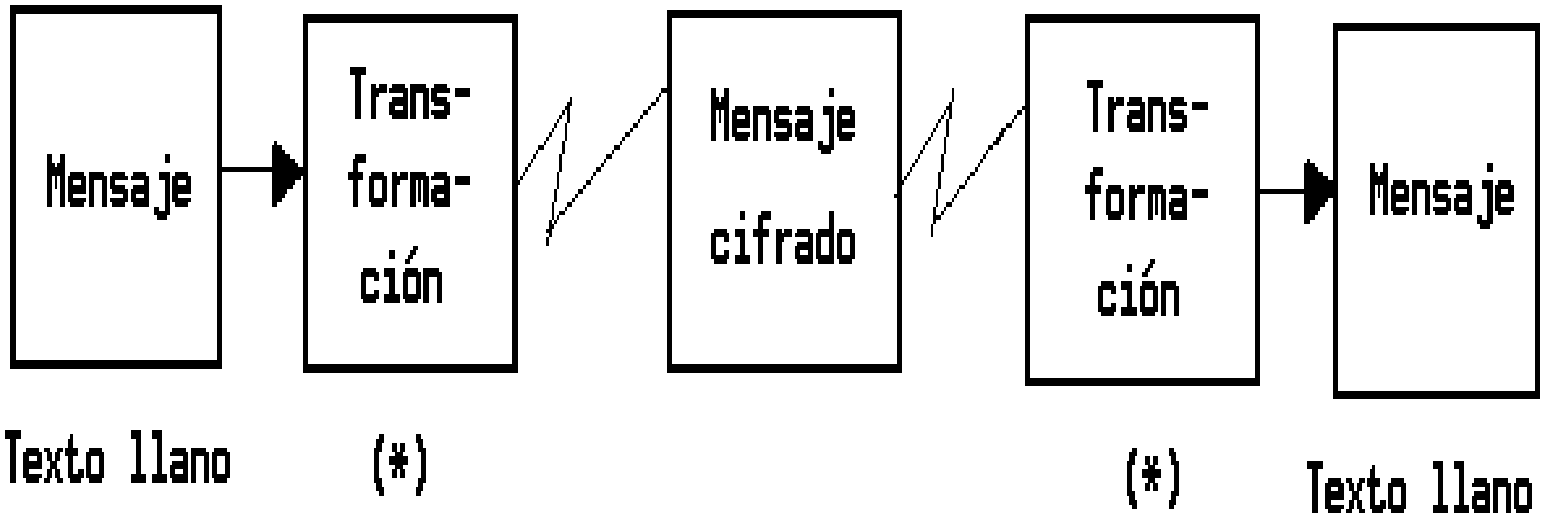
| Objetos / Dominios | Compiladores | Archivos | Programadores | Usuarios | Syst Prog |
|---|---|---|---|---|---|
| Programadores | Ejecutar | Leer | | | |
| System Programmer | Leer/Grabar/Ejecutar | | | SWITCH | |
| Usuarios | Ejecutar | Leer/Grabar | | | |

# SEGURIDAD



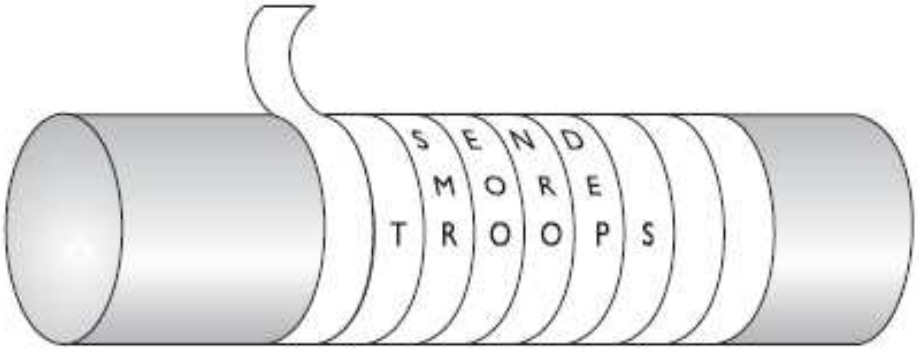

Escucha, genera o modifica datos.

# CRIPTOGRAFÍA



# Criptografía (un poco de historia)

La **escitala** utilizada por los espartanos para enviar mensajes consistía de una tira de papel o tela en la cual se escribía el mensaje que solo podía ser comprendido para aquellos que contaban con la escitala (vara o bastón) del diámetro correcto.




http://www.dma.fi.upm.es/java/matematicadiscreta/aritmeticamodular/criptografia.html

# Criptografía

## CESAR (SUSTITUCIÓN)

Atribuído al emperador Julio Cesar.
Se suma un número entero fijo a las letras del alfabeto, por ejemplo + 3

Ejemplo:

Mensaje: SUPERSECRETO

Mensaje cifrado: VXSHUVHFUHWR

# Criptografía

## SUSTITUCIÓN CON PALABRA CLAVE

Se escriben las letras del alfabeto mezcladas con una palabra clave, por ejemplo:
Clave: SIMON BOLIVAR.
Sin letras repetidas: SIMON BLVAR.

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
Alf. Transformado
S I M O N B L V A R C D E F G H J K P Q T U W X Y Z

Aquí el texto llano CESAR cifrará como MNPSK.

# Criptografía

ONE TIME PAD (Bloque de uso único)

Es un método totalmente seguro. Trabaja como el método de Cesar, pero el entero a sumar varía para cada caracter del texto en forma aleatoria.

El criptograma "SECRETO" podría provenir de:
- MENTIRA con la clave 6 0 15 2 22 23 12, o de:
- REALEZA con la clave 1 0 2 6 0 6 12.

***El problema es que la clave debe ser tan larga como el mensaje, y nunca debe ser reutilizada!!!***

# Criptografía

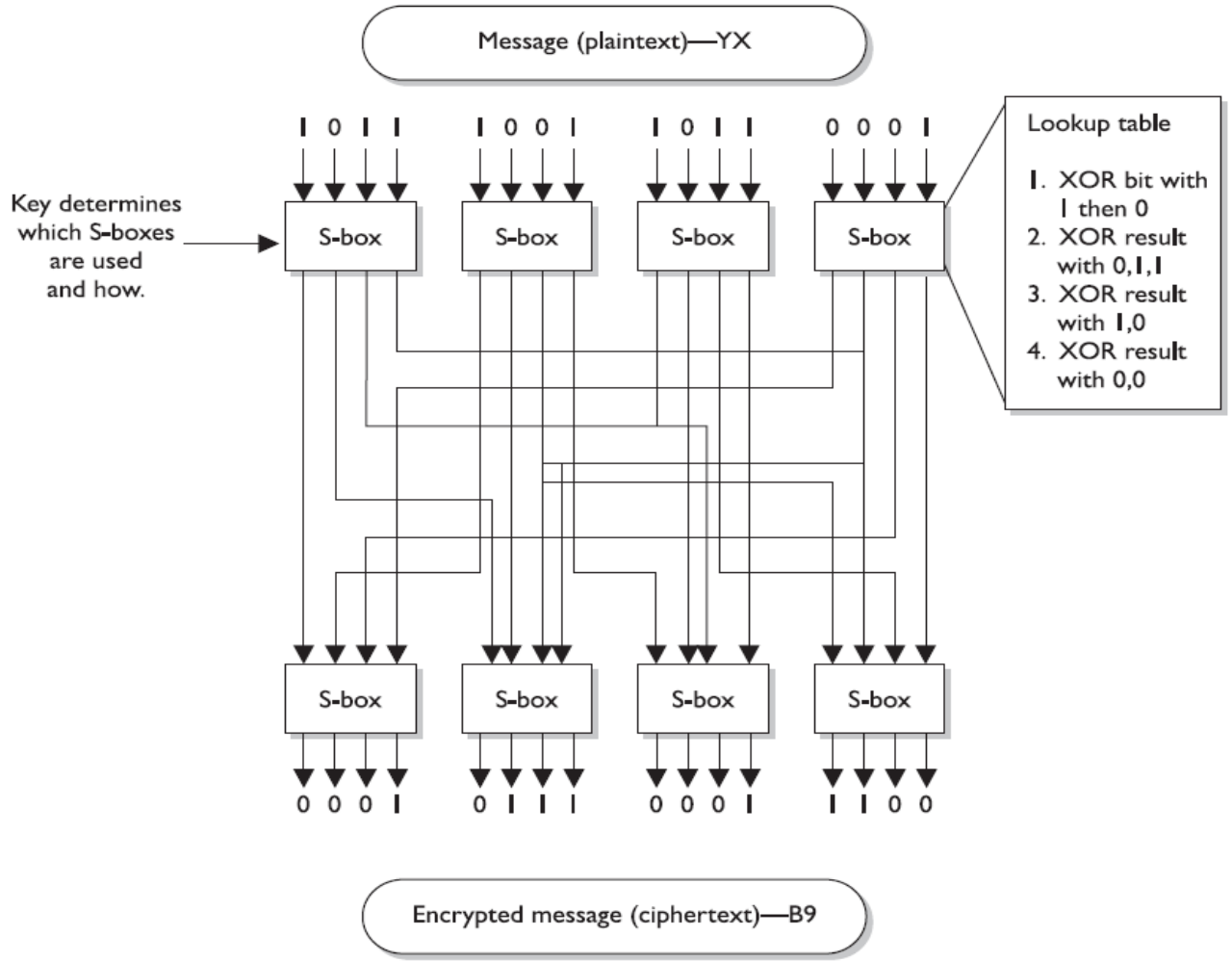
DES (Data Encryption Standard - 1977)

Divide el texto llano en bloques de 64 bits, utiliza una clave de 56 bits y los 8 bits restantes se utilizan para paridad.
Realiza 16 pasos con funciones intermedias de substitución (S-boxes) y transposición, pero su corazón es el XOR.

| XOR | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

# Criptografía

## Ejemplo de funciones de substitución y transposición




Cada S-box tiene una tabla asociada que es utilizada por el algoritmo para indicar cómo deben cifrarse los bits.

# Criptografía

## DISTRIBUCION DE CLAVES

Los métodos criptográficos (DES y One-time Pad) tienen el problema de la distribución de las claves.

Todos los que participan deben conocerlas y por ende existe más de un punto posible de divulgación por parte de un atacante.

Son **simétricos**

# Criptografía

Métodos Asimétricos

Se basan en la existencia de pares de claves: Claves Públicas y Privadas

A y B publican sus claves de encriptamiento Ea y Eb, y mantiene en secreto sus claves de desencriptamiento Da y Db. Con esto, el problema de distribución de claves casi desaparece .
Existe aún el problema de un atacante en el medio de comunicación que intercepte el intercambio de claves y suplante la persona de uno de los emisores (man-in-the-middle). El atacante Z se hace pasar por B.

# Criptografía

Métodos Asimétricos: RSA (Rivest-Shamir-Adleman)

Se buscan dos primos grandes p y q, y se obtiene n = p.q
Se buscan dos números e y d que contemplen la siguiente propiedad:

$$e.d \bmod [(p-1).(q-1)] = 1$$

Se representa el texto llano por medio de un M tal que

$$0 \leq M \leq n-1$$

de forma tal que el encriptamiento resulta:

$$C = M^e \bmod(n)$$

y el desencriptamiento es:

$$M = C^d \bmod(n)$$

# Criptografía

La clave pública es el par (e,n) y la privada el par (d,n). Conocer e y n no da suficiente información como para hallar d (p y q son secretos).

Un espía lo que puede hacer es factorizar n, que es público, para obtener p y q, y luego podrá calcular d, ya que e es también público.

# Criptografía

Veamos un ejemplo con números pequeños:

p=3; q=5; n=15

Se elige e = 3 y d = 11 tal que 3.11 mod [2.4]=33 mod(8) = 1

Si suponemos un mensaje M = 8, resulta:

cifrado $c = 8^3 \bmod (15) = 512 \bmod(15) = 2$

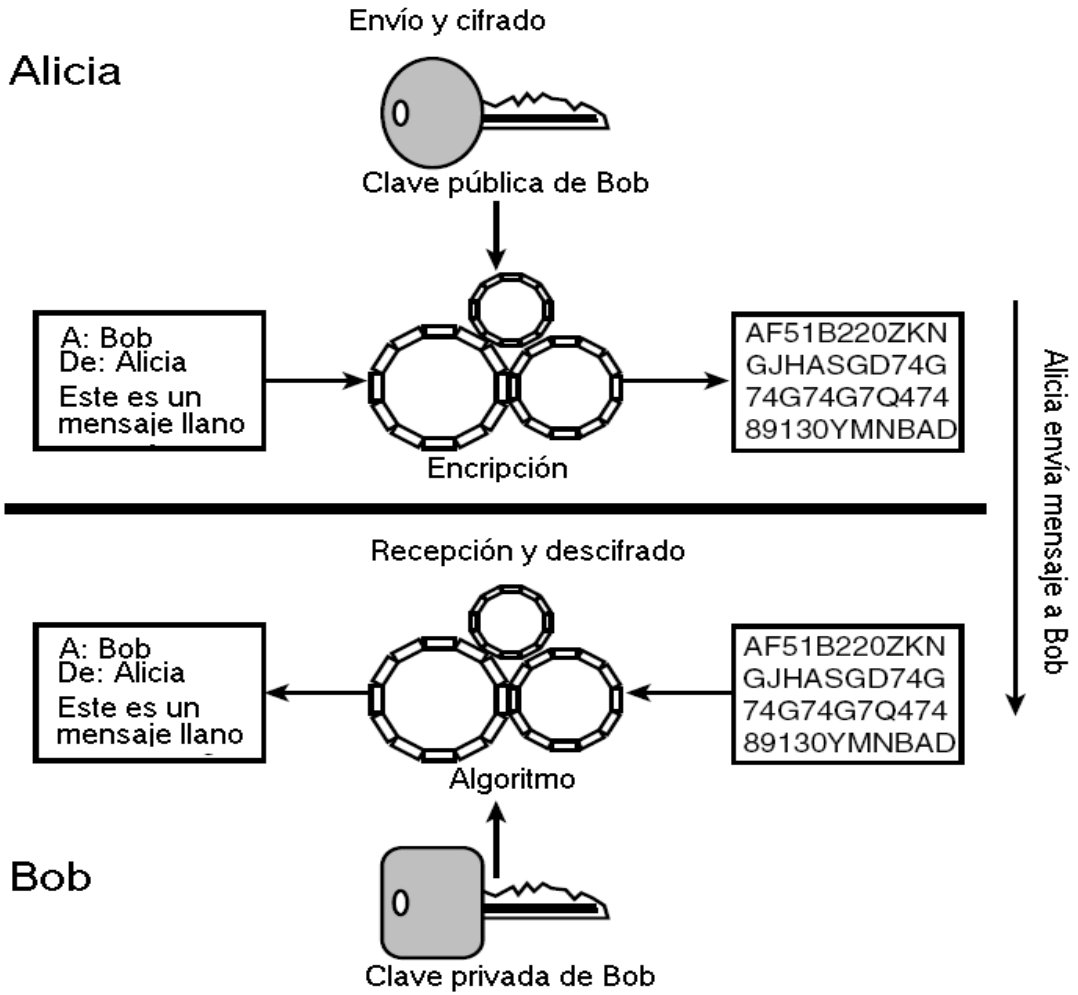
siendo c = 2, el mensaje cifrado que viaja por el medio de comunicación.

En el lugar de destino, se hace:

$M = 2^{11} \bmod(15) = 2048 \bmod(15) = 8$

donde M=8 es el mensaje original desencriptado.

# Criptografía



# Criptografía

## AUTENTICIDAD (FIRMA)

Si pensamos el RSA al revés, por ejemplo, si encriptamos con una clave D secreta, se puede desencriptar con una clave E pública, y nadie podría falsificar el mensaje, ni siquiera el receptor.
Esto podría usarse como "firma" de documentación electrónica.
Pero cifrar un mensaje por RSA es muy costoso en tiempos de cómputo luego.....

# Criptografía

FIRMA DIGITAL

Para un mensaje dado se calcula un “digesto”. El digesto es un valor de tamaño fijo que surge de la aplicación de una función de tipo HASH (ej: MD5, SHA-1).

Estas funciones son de fácil y rápido cálculo y son no reversibles (no se puede obtener el input original a partir del resultado hash).

Ese digesto se cifra utilizando la clave privada del emisor y cualquier receptor conociendo la clave pública del receptor puede verificar la autenticidad del mensaje.

# Criptografía

## Ejemplo de mensaje firmado usando PGP

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Aquí termina la clase de hoy

Graciela

-----BEGIN PGP SIGNATURE-----
Version: PGP 8.0.2 - not licensed for commercial use: www.pgp.com

iQA/AwUBSB0CSQaP1MLVR6eeEQJlVACfUGF1mORDtJF3hZEfTYFxi
RU3eCwAn3d/
vNKStFEJu4YoDyb4zS9Ao/jA
=lb/Z
-----END PGP SIGNATURE-----

# Criptografía

## Verificación de firma

*** PGP SIGNATURE VERIFICATION ***
*** Status:  Good Signature
*** Signer:  Grace Norma Pataro <gpataro@dc.uba.ar> (0xD547A79E)
*** Signed:  03/05/2008 09:24:41 p.m.
*** Verified: 03/05/2008 09:26:33 p.m.
*** BEGIN PGP VERIFIED MESSAGE ***

Aquí termina la clase de hoy

Graciela

*** END PGP VERIFIED MESSAGE ***