

# Seguretat Informàtica (SI)

Tema 2. Criptografía

Davide Careglio

Fuentes: Jordi Nin, "Cryptography", Computer Security, 2014  
Jaime Delgado, "Cryptography", Computer Security, 2018

# Temario

---

- ▶ Tema 1. Introducción
  - ▶ Tema 2. Criptografía
  - ▶ Tema 3. Infraestructura PKI
- 
- ▶ Tema 4. Seguridad en la red
  - ▶ Tema 5. Seguridad en las aplicaciones
- 
- ▶ Tema 6. Seguridad en los sistemas operativos
  - ▶ Tema 7. Análisis forense

# Temario

---

- ▶ Tema 1. Introducción
- ▶ **Tema 2. Criptografía**
- ▶ Tema 3. Infraestructura PKI
  
- ▶ Tema 4. Seguridad en la red
- ▶ Tema 5. Seguridad en las aplicaciones
  
- ▶ Tema 6. Seguridad en los sistemas operativos
- ▶ Tema 7. Análisis forense

# Tema 2. Índice

---

1. Motivación
2. Definición de criptografía
3. Criptosistemas históricos
4. Tipos
  1. Criptografía privada
  2. Criptografía pública
5. Firma digital
6. Algunos principios matemáticos
7. Algoritmos más conocidos

## 2.1 – Motivación

Que puede hacer típicamente un ataque

---

- ▶ **Eavesdrop**

- ▶ Interceptar mensajes

- ▶ **Insertion**

- ▶ Insertar mensajes en una conexión

- ▶ **Impersonation**

- ▶ Hacerse pasar por otro alterando campos de los datos (por ejemplo @IP origen) para acceder a determinados servicios

- ▶ **Hijacking**

- ▶ Meterse en una conexión activa quitando uno de los dos extremos y hacerse pasar por este

- ▶ **Denial of Service**

- ▶ inhabilitar un servicio mediante el envío de gran cantidad de solicitudes desde uno o mas ordenadores (generalmente zombis) hasta saturar los dispositivos y servicios

## 2.1 – Motivación

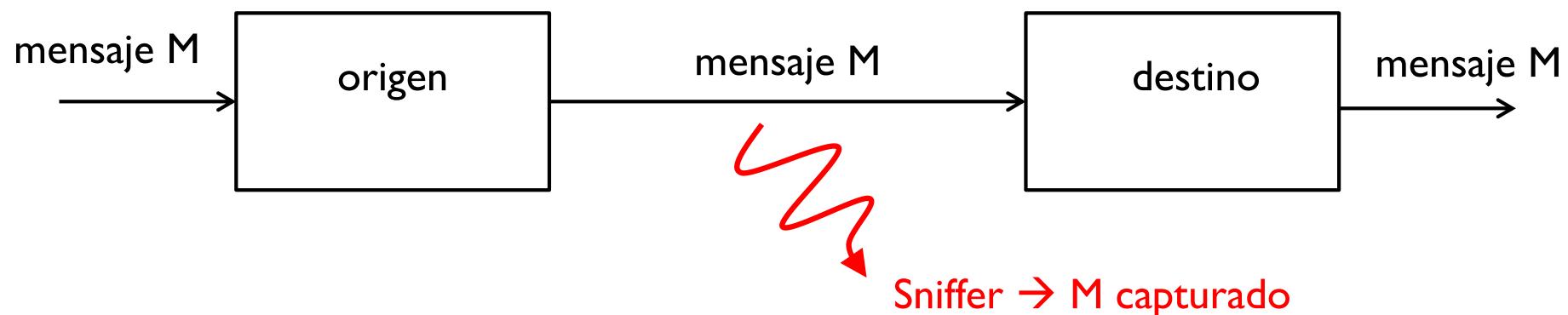
### Que puede proteger la criptografía

---

- ▶ Directamente
  - ▶ **Confidencialidad:** solo origen y destino deben poder entender el mensaje
  - ▶ **Autenticación:** origen y destino deben poder confirmar la identidad del otro
  - ▶ **Integridad** del mensaje: origen y destino quieren poder asegurar que el mensaje se recibe sin alterar y que nadie más lo haya podido recibir
- ▶ Indirectamente
  - ▶ **Acceso y disponibilidad:** los servicios y las aplicaciones deben ser accesibles y disponibles para los usuarios

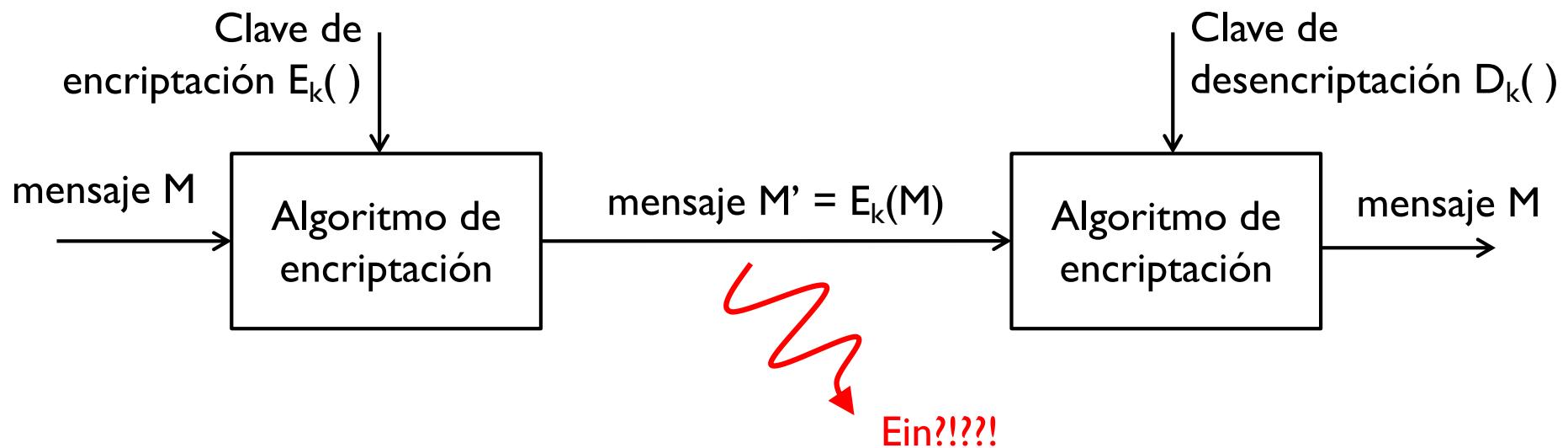
## 2.2 - Definición de criptografía

- ▶ Del griego *kryptο* (oculta) y *grapho* (escritura)
- ▶ Literalmente escritura oculta
- ▶ Se ocupa de las técnicas de cifrado o codificado destinadas a alterar la representación lingüística de un mensaje con el fin de hacerlo ininteligible a receptores no autorizados



## 2.2 - Definición de criptografía

- ▶ Del griego *kryptο* (oculta) y *grapho* (escritura)
- ▶ Literalmente escritura oculta
- ▶ Se ocupa de las técnicas de cifrado o codificado destinadas a alterar la representación lingüística de un mensaje con el fin de hacerlo ininteligible a receptores no autorizados



## 2.3 - Criptosistemas históricos

---

- ▶ Los códigos históricos son
  - ▶ Transposición
  - ▶ Substitución

## 2.3 - Criptosistemas históricos

### Transposición

---

- ▶ Ejemplo
  - ▶  $M = \text{CRYPTOGRAPHY}$  se convierte en  $E_k(M) = \text{YCROPTAGRYPH}$
  - ▶  $D_k(E_k(M)) = \text{CRYPTOGRAPHY}$
- ▶ ¿Qué hace el algoritmo de cifrado?
- ▶ ¿Cuál es la clave?

## 2.3 - Criptosistemas históricos

### Transposición

---

- ▶ Ejemplo
  - ▶  $M = \text{CRYPTOGRAPHY}$  se convierte en  $E_k(M) = \text{YCROPTAGRYPH}$
  - ▶  $D_k(E_k(M)) = \text{CRYPTOGRAPHY}$
- ▶ Algoritmo de cifrado
  - ▶ Rotar las letras hacia la derecha de una posición dentro de cada grupo bien definido del mensaje
  - ▶ El tamaño del grupo es la clave
- ▶ ¿Cuál es la clave?
  - ▶ Clave fija  $k = 3$

## 2.3 - Criptosistemas históricos

### Substitución

---

- ▶ Ejemplo:
  - ▶  $M = \text{CRYPTOGRAPHY}$  se convierte en  $E_k(M) = \text{FUBSWRJUDSKB}$
  - ▶  $D_k(E_k(M)) = \text{CRYPTOGRAPHY}$
  
- ▶ ¿Qué hace el algoritmo de cifrado?
- ▶ ¿Cuál es la clave?

## 2.3 - Criptosistemas históricos

### Substitución

---

- ▶ Ejemplo:
  - ▶  $M = \text{CRYPTOGRAPHY}$  se convierte en  $E_k(M) = \text{FUBSWRJUDSKB}$
  - ▶  $D_k(E_k(M)) = \text{CRYPTOGRAPHY}$
- ▶ ¿Que hace el algoritmo de cifrado?
  - ▶ Substitución de una letra con otra un número determinado de posiciones más adelante en el abecedario
  - ▶ El número de posiciones es la clave
- ▶ ¿Cuál es la clave?
  - ▶ Clave fija  $k = 3$

## 2.4 – Tipos y características

---

- ▶ El algoritmo de encriptación y desencriptación es conocido
- ▶ Lo que es secreto es la clave
- ▶ En los ejemplos anteriores, k es el factor desconocido en el cifrado

## 2.4 – Tipos y características

### Shannon best practices

---

- ▶ Idea de “confusión y difusión” definida por Shannon como condición necesaria para un cifrado seguro y práctico
- ▶ Confusión
  - ▶ Hacer que la relación entre clave e mensaje cifrado sea la más compleja posible (esconde la relación entre clave y mensaje cifrado)
  - ▶ Hacer realmente difícil encontrar la clave aunque se tuviera a disposición un gran número de mensajes no cifrados y mensajes cifrados con una misma clave
  - ▶ Es decir, si se cambiara un solo bit de la clave, el mensaje cifrado debería cambiar completamente
- ▶ Difusión
  - ▶ Hacer que el mensaje cifrado dependa del no cifrado de una manera muy compleja (esconde la relación entre mensaje y mensaje cifrado)
  - ▶ Es decir, si se cambiara aunque solo un bit del bloque no cifrado, el bloque cifrado debería cambiar completamente

## 2.4 – Tipos y características

---

- ▶ **Criptografía privada**
  - ▶ También conocida como criptografía simétrica
  - ▶ Origen y destino usan la misma clave secreta
- ▶ **Criptografía publica**
  - ▶ También conocida como criptografía asimétrica
  - ▶ Se usan dos claves, una pública y una privada

# Tema 2. Índice

---

1. Motivación
2. Definición de criptografía
3. Criptosistemas históricos
4. Tipos
  1. Criptografía privada
  2. Criptografía pública
5. Firma digital
6. Algunos principios matemáticos
7. Algoritmos más conocidos

## 2.4.1 - Criptografía privada

---

- ▶ También conocida como criptografía simétrica
- ▶ Origen y destino usan la misma clave secreta
- ▶ Única técnica de cifrado públicamente conocida hasta junio de 1976
- ▶ Actualmente se usan métodos basados en cifrado en **bloques** y cifrado de **flujo**
- ▶ Se necesita el intercambio de la clave entre los dos extremos a través de un sistema seguro
  - ▶ Hoy en día existen métodos de intercambio de claves de forma segura sobre un medio no seguro
  - ▶ Por ejemplo el Diffie-Hellman que veremos más adelante

## 2.4.1 - Criptografía privada

---

- ▶ **Cifrado en bloques**
  - ▶ Se define un grupo de bits, llamado bloque, que tiene una transformación invariante que solo depende de la clave
  - ▶ Por ejemplo el bloque de bits 1100 se transforma siempre en el 0111
- ▶ **Estándares más conocidos**
  - ▶ One Time Pad (OTP)
  - ▶ Data Encryption Standard (DES)
  - ▶ 3DES
  - ▶ Advanced Encryption Standard (AES)

## 2.4.1 - Criptografía privada

### One Time Pad (OTP)

- ▶ Cada bloque de bits del mensaje es encriptado usando una clave secreta aleatoria de la misma longitud que el bloque

```
SENDING
-----
message: 0 0 1 0 1 1 0 1 0 1 1 1 ...
pad:      1 0 0 1 1 1 0 0 1 0 1 1 ...
XOR      -----
cipher:  1 0 1 1 0 0 0 1 1 1 0 0 ...
```

```
RECEIVING
-----
cipher:  1 0 1 1 0 0 0 1 1 1 0 0 ...
pad:      1 0 0 1 1 1 0 0 1 0 1 1 ...
XOR      -----
message: 0 0 1 0 1 1 0 1 0 1 1 1 ...
```

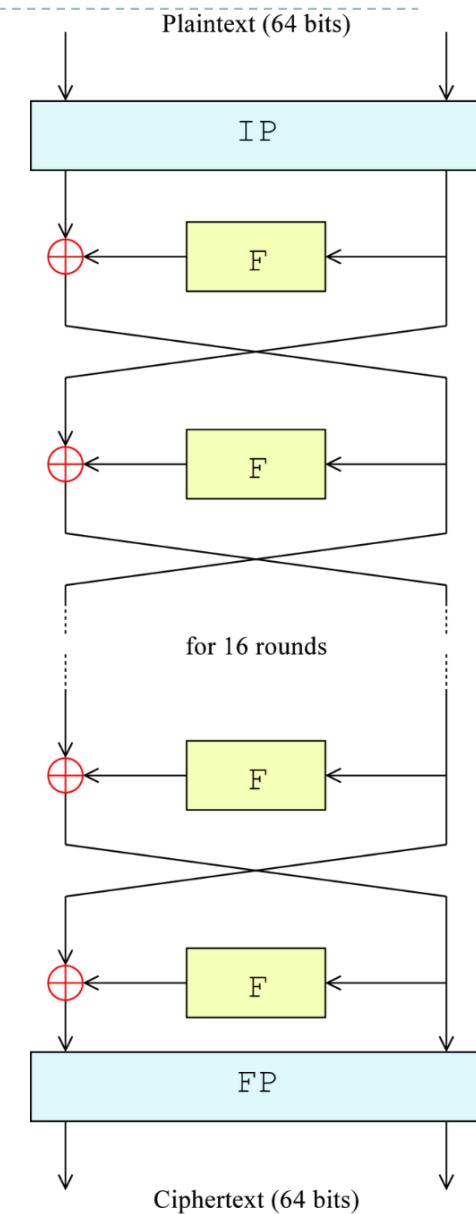
- ▶ Si la clave es realmente aleatoria, del mismo tamaño que un bloque del mensaje y se usa una única vez, el mensaje cifrado es imposible de descifrar sin conocer la clave
- ▶ Usado durante la guerra fría para la comunicación entre EEUU y URSS

## 2.4.1 - Criptografía privada

### Data Encryption Standard (DES)

- ▶ Elaborado por IBM y estandarizado en el 1976
- ▶ Los bloques son de 64 bits
- ▶ La clave es de 56 bits (+ 8 de paridad)
- ▶ El algoritmo es conocido (en la figura)
  - ▶ Consiste de una permutación inicial IP, 16 rondas F iguales y una permutación final FP (función inversa a IP)
  - ▶ Y de 16 rondas F iguales (se llaman F porque se aplica una función llamada Feistel)
  - ▶ Para cada ronda se usa una subclave diferente generada a partir de la clave de 56 bits
- ▶ El descifrado usa el mismo algoritmo, la única diferencia es que las subclaves usadas en cada ronda F se aplican en orden inverso
- ▶ Simplifica la implementación (sobre todo hardware)

Fuentes: imagen: Wikipedia

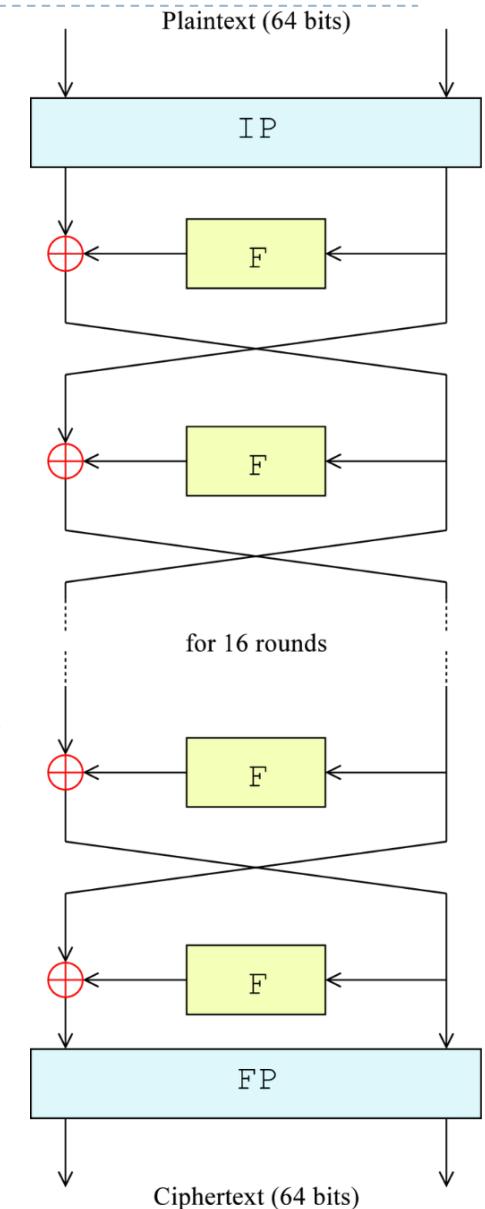


## 2.4.1 - Criptografía privada

### Data Encryption Standard (DES)

- ▶ La permutación inicial (IP) y la permutación final (FP) no son criptográficamente significativa
  - ▶ Se incluyeron para facilitar la carga de los bloques en el hardware de los 70
- ▶ Las 16 rondas F son todas idénticas
  - ▶ Antes de empezar las rondas, el bloque se divide en dos semibloques de 32 bits (S1 y S2) que se procesan alternativamente
  - ▶ Por un lado, el semibloque S1 se duplica
  - ▶ A una copia de S1 se aplica la función Feistel con una subclave y el resultado se combina (XOR) con el semibloque S2
  - ▶ La otra copia de S1 va directamente a la siguiente ronda
  - ▶ En la siguiente ronda, se intercambian los procesos entre S1 y S2 y se usa otra subclave
  - ▶ Y así durante 16 rondas

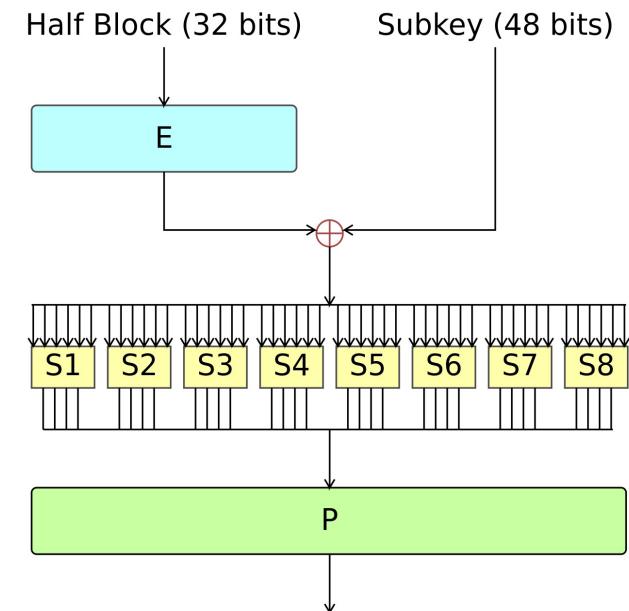
Fuentes: imagen: Wikipedia



## 2.4.1 - Criptografía privada

### Data Encryption Standard (DES)

- ▶ La función Feistel consiste de 4 etapas
  - ▶ **Expansión:** el semibloque de 32 bits se expande a 48 duplicando algunos bits
  - ▶ **Mezcla:** operación de XOR con una subclave de 48 bits
  - ▶ **Sustitución:** se dividen los 48 bits en 8 sub-bloques (8 S-cajas) de 6 bits. A cada sub-bloque, se aplica una transformación no lineal especificada por una tabla de búsqueda que transforma los 6 bits iniciales en 4 bits de salida
  - ▶ **Permutación:** los 32 bits de salida de las 8 S-cajas se reordenan según una permutación fija
- ▶ La sustitución y la permutación son las operaciones que proporcionan “confusión y difusión”
- ▶ Las subclaves usadas se generan a partir de la clave inicial según un algoritmo determinado (este si que no lo explico)



Fuentes: imagen: Wikipedia

## 2.4.1 - Criptografía privada

### Data Encryption Standard (DES)

---

- ▶ Fue la técnica de encriptación aceptada por la NSA de EEUU
- ▶ Su esquema base así como sus mejoras fueron usadas hasta el 26 de mayo de 2002 cuando fue reemplazado por el AES
  
- ▶ El ataque más práctico para romper el cifrado DES es la **fuerza bruta**, es decir usar una por una todas las combinaciones posibles de claves que son  $2^{56}$
- ▶ De hecho, se especula que la NSA impuso una clave de solo 56 bits porque en aquellos tiempos sola la NSA tenía la capacidad computacional necesaria para romper el cifrado DES
- ▶ Hoy en día se puede descifrar un DES en pocos minutos (en 1998 se demostró que era rompible en 2 días)

## 2.4.1 - Criptografía privada

### 3DES

---

- ▶ Después de descubrir que el DES era “fácilmente” rompible, se propuso pasar al 3DES
- ▶ El 3DES aplica el DES tres veces con tres claves distintas, haciendo así incrementar la clave hasta los 168 bits ( $3 \times 56$  bits)
- ▶ De momento no hay vulnerabilidad conocida
  - ▶ La fuerza bruta no es computacionalmente viable porque se necesitan procesar  $2^{168}$  combinaciones
- ▶ Pero es extremadamente lento
- ▶ AES puede llegar a ser 6 veces más rápido que el 3DES

## 2.4.1 - Criptografía privada

### Advanced Encryption Standard (AES)

---

- ▶ Reemplazo del DES como estándar de la NSA de EEUU desde 2002
- ▶ También conocido como cifrado Rijndael por sus dos autores Joan Daemen y Vincent Rijman que lo publicaron el 26 de noviembre de 2001
- ▶ Basado en una red de sustitución y permutación
- ▶ Relativamente fácil de implementar y usa poca memoria
- ▶ Hoy en día se usa a gran escala (por ejemplo en WPA2 en 802.11)
- ▶ Se usan bloques de 128 bits y tamaños de claves de 128, 192 o 256 bits

## 2.4.1 - Criptografía privada

### Advanced Encryption Standard (AES)

- ▶ Reemplazo del DES como estándar de la NSA de EEUU desde 2002
- ▶ También conocido como cifrado Rijndael por sus dos autores Joan Daemen y Vincent Rijman que lo publicaron el 26 de noviembre de 2001

The screenshot shows a web-based configuration interface for a COMTREND ADSL Router. The left sidebar contains navigation links: Device Info, Advanced Setup, Wireless (selected), Basic, Security, MAC Filter, Wireless Bridge, Advanced, Station Info, Diagnostics, and Management. The main content area is titled "Wireless -- Security". It includes a descriptive text about configuring wireless security features, a dropdown menu for "Network Authentication" set to "WPA2-PSK", a text input field for "WPA Pre-Shared Key" containing a masked password, a link to "Click here to display", and a numeric input field for "WPA Group Rekey Interval" set to 168. Below these are dropdown menus for "WPA Encryption" set to "AES" and "WEP Encryption" set to "Disabled". A "Save/Apply" button is at the bottom right.

## 2.4.1 - Criptografía privada

### Advanced Encryption Standard (AES)

---

- ▶ El algoritmo es conocido
  - ▶ Se organizan los 128 bits en una matriz de 4x4 bytes (16 bytes x 8 = 128 bits)

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

## 2.4.1 - Criptografía privada

### Advanced Encryption Standard (AES)

- ▶ El algoritmo es conocido
  - ▶ Se organizan los 128 bits en una matriz de 4x4 bytes (16 bytes x 8 = 128 bits)
  - ▶ Se hace un primera operación de combinación de cada byte con la clave modificada según una determinada operación

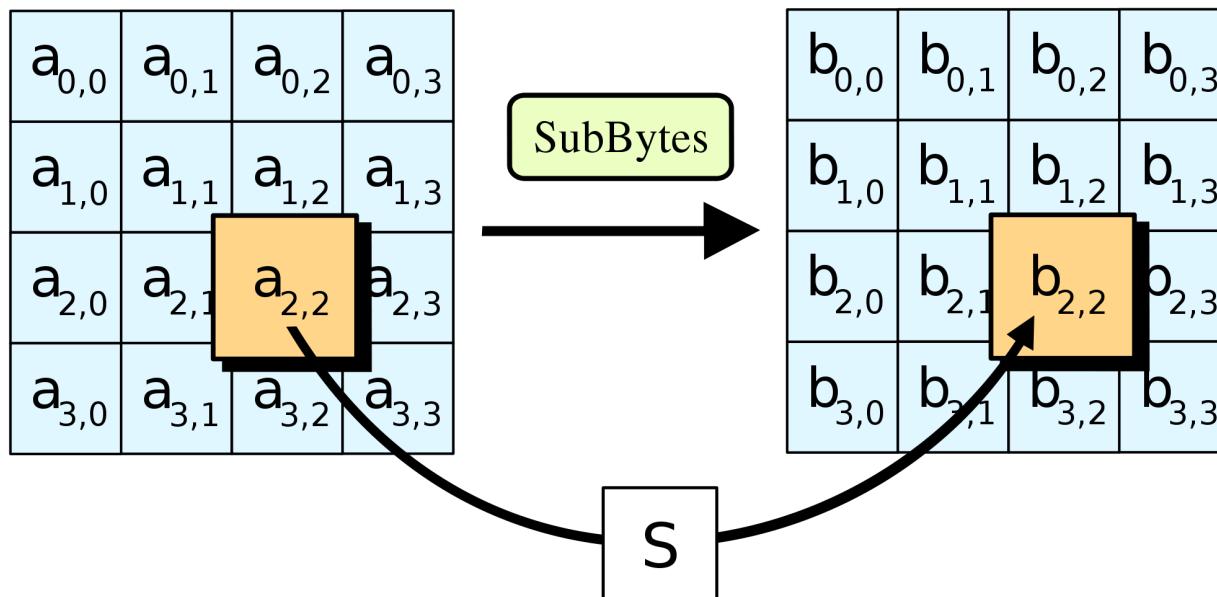
$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$A_{3,1}$	$a_{3,2}$	$a_{3,3}$

+ RoundKey

## 2.4.1 - Criptografía privada

### Advanced Encryption Standard (AES)

- ▶ El algoritmo es conocido
  - ▶ Se hacen luego 10 (clave de 128 bits), 12 (192 bits) o 14 (256 bits) rondas, cada una con estos pasos
    - I. SubBytes: En cada ronda se hace una sustitución de cada byte por otro según una tabla conocida S



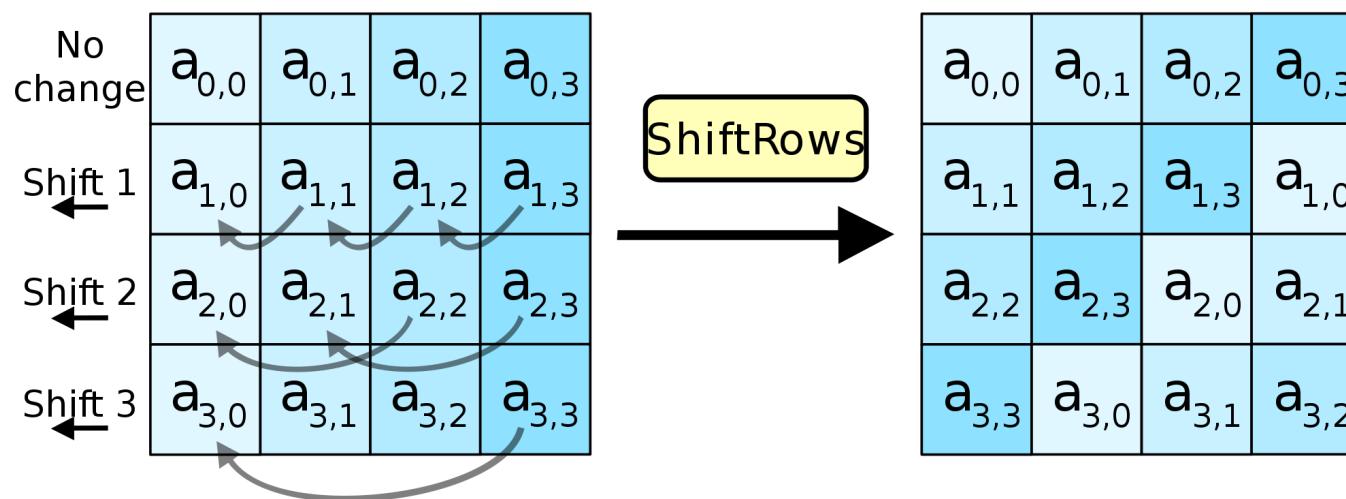
## 2.4.1 - Criptografía privada

### Advanced Encryption Standard (AES)

- ▶ El algoritmo es conocido

- ▶ Se hacen luego 10 (clave de 128 bits), 12 (192 bits) o 14 (256 bits) rondas, cada una con estos pasos

2. ShiftRows: Las últimas tres líneas de bytes de la matriz se desplazan un cierto número de posiciones



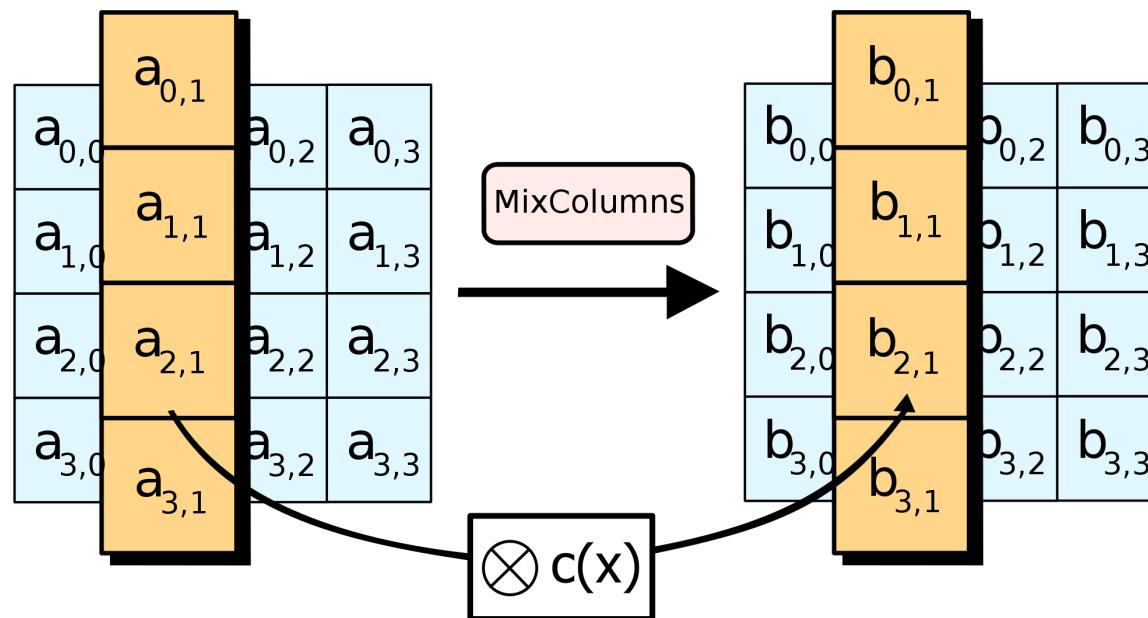
## 2.4.1 - Criptografía privada

### Advanced Encryption Standard (AES)

- ▶ El algoritmo es conocido

- ▶ Se hacen luego 10 (clave de 128 bits), 12 (192 bits) o 14 (256 bits) rondas, cada una con estos pasos

3. MixColumns: Se combinan los 4 bytes de cada columna usando una transformación lineal conocida  $c(x)$



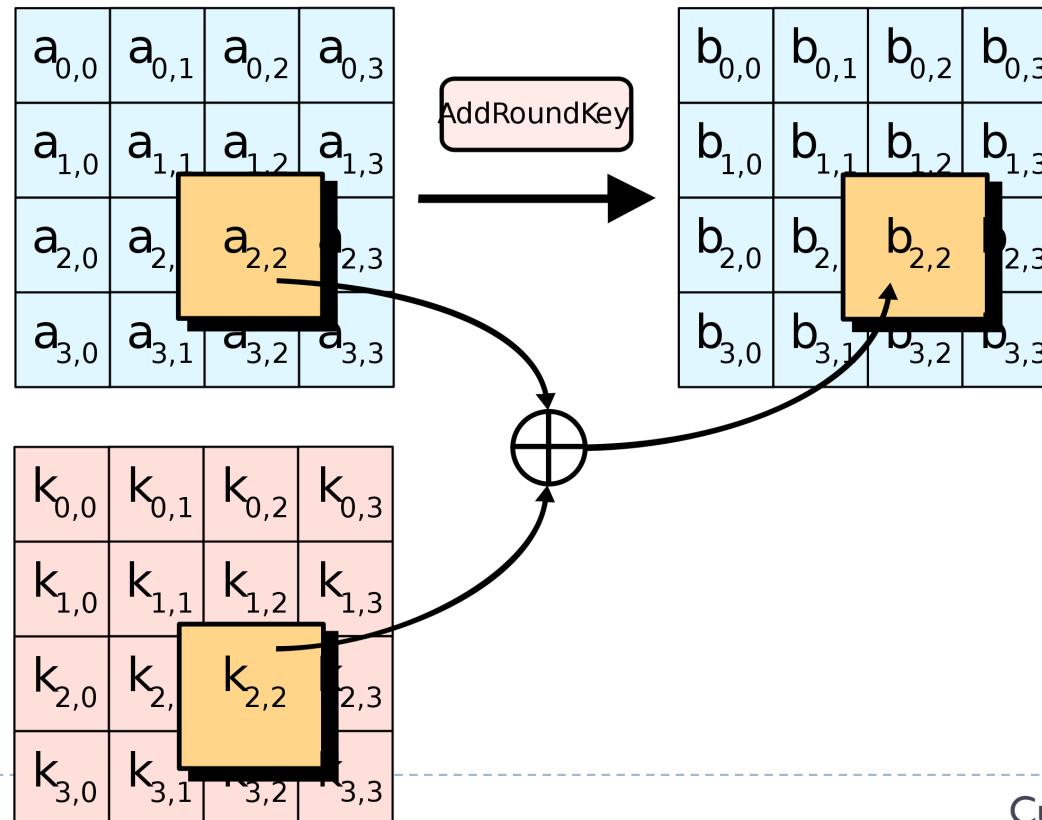
## 2.4.1 - Criptografía privada

### Advanced Encryption Standard (AES)

- ▶ El algoritmo es conocido

- ▶ Se hacen luego 10 (clave de 128 bits), 12 (192 bits) o 14 (256 bits) rondas, cada una con estos pasos

4. AddRoundKey: Se combina cada bytes de la matriz con la clave modificada según la ronda



## 2.4.1 - Criptografía privada

### Advanced Encryption Standard (AES)

---

- ▶ El algoritmo es conocido
  - ▶ Se hace una última etapa donde se aplican una última vez los pasos 1, 2 y 4
    1. SubBytes: En cada ronda se hace una substitución de cada byte por otro según una tabla conocida S
    2. ShiftRows: Las últimas tres líneas de bytes de la matriz se desplazan un cierto número de posiciones
    4. AddRoundKey: Se combina cada bytes de la matriz con la clave modificada según la ronda

## 2.4.1 - Criptografía privada

### Advanced Encryption Standard (AES)

#### ▶ Ejemplo de implementación en c

- ▶ <https://github.com/kokke/tiny-AES-c/blob/master/aes.c>

571 lines (480 sloc) | 18.7 KB

Raw Blame History

```
1  /*
2
3 This is an implementation of the AES algorithm, specifically ECB, CTR and CBC mode.
4 Block size can be chosen in aes.h - available choices are AES128, AES192, AES256.
5
35 ****
36 /* Includes: */
37 ****
38 #include <string.h> // CBC mode, for memset
39 #include "aes.h"
40
41 ****
42 /* Defines: */
43 ****
44 // The number of columns comprising a state in AES. This is a constant in AES. Value=4
45 #define Nb 4
46
47 #if defined(AES256) && (AES256 == 1)
48     #define Nk 8
49     #define Nr 14
50 #elif defined(AES192) && (AES192 == 1)
51     #define Nk 6
52     #define Nr 12
53 #else
54     #define Nk 4      // The number of 32 bit words in a key.
55     #define Nr 10     // The number of rounds in AES Cipher.
56
57         break;
58     }
59     bi = 0;
60 }
61
62     buf[i] = (buf[i] ^ buffer[bi]);
63 }
64 }
65
66 #endif // #if defined(CTR) && (CTR == 1)
67 }
```

## 2.4.1 - Criptografía privada

### Cifrado de flujo

---

- ▶ Para algunas aplicaciones, el cifrado en bloques es inapropiada porque los flujos de datos se producen en tiempo real en pequeños fragmentos (por ejemplo telefonía).
  - ▶ Técnicas de cifrado que realizan el cifrado incrementalmente, convirtiendo el mensaje en claro en mensaje cifrado bit a bit.
- 
- ▶ Estándares más conocidos
    - ▶ RC4 (usado en WEP de 802.11)
    - ▶ GSM (3G)
      - ▶ A5/1
      - ▶ A5/2
        - intencionalmente débil, hackeado al cabo de un mes de su publicación
        - prohibida la implementación en móviles desde 7/2007
    - ▶ Snow 3G (usado en 4G)
    - ▶ Snow V (para 5G)

## 2.4.1 - Criptografía privada

### Problemas

---

- ▶ **Distribución de la clave**
  - ▶ Los usuarios deben intercambiarse la clave antes de empezar la comunicación
  - ▶ Hay que usar un método seguro para conseguirlo
- ▶ **Gestión de la clave**
  - ▶ Si hay  $n$  usuarios, cada pareja debe intercambiarse una clave, con lo que se van a necesitar  $n(n-1)/2$  claves
- ▶ **Firma digital**
  - ▶ No es posible tener una firma propia digital ya que cada clave es compartida entre dos usuarios

# Tema 2. Índice

---

1. Motivación
2. Definición de criptografía
3. Criptosistemas históricos
4. Tipos
  1. Criptografía privada
  2. Criptografía pública
5. Firma digital
6. Algunos principios matemáticos
7. Algoritmos más conocidos

## 2.4.2 - Criptografía pública

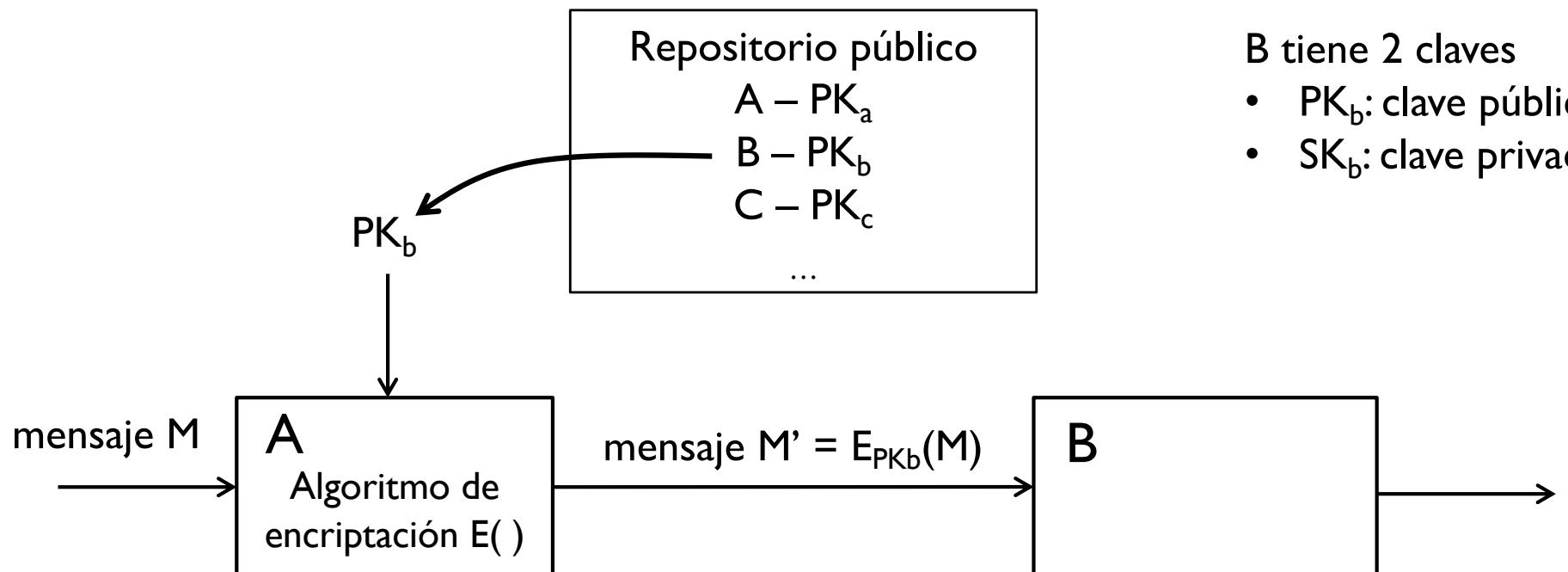
---

- ▶ También conocida como criptografía asimétrica
- ▶ Cada extremo posee dos claves, una pública y una privada
  - ▶ Las claves públicas deben estar disponibles para todos (repositorio de claves públicas)
  - ▶ Las claves privadas las mantienen secretas los usuarios
  - ▶ Las dos claves se generan a través de un algoritmo de generación de claves
- ▶ Los algoritmos de cifrado y descifrado son conocidos (públicos)
- ▶ De esta forma no se necesita el intercambio de claves para encriptar y desencriptar los mensajes

## 2.4.2 - Criptografía pública

### ▶ Funcionamiento en el origen

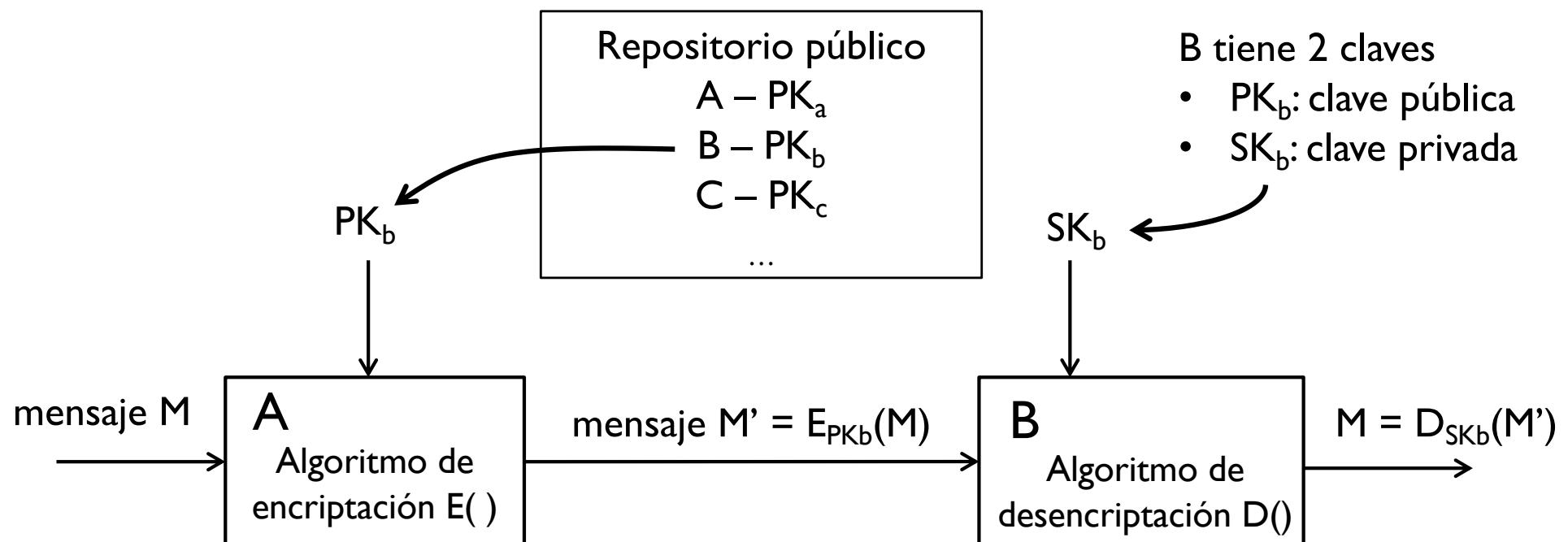
- ▶ El origen A quiere transmitir un mensaje M al destino B
- ▶ A encuentra la clave pública  $PK_b$  de B en un directorio público
- ▶ A computa  $M' = E_{PK_b}(M)$  donde E es un algoritmo de encriptación público
- ▶ A envía el mensaje  $M'$  a B



## 2.4.2 - Criptografía pública

### ▶ Funcionamiento en el destino

- ▶ El destino B recupera su clave privada  $SK_b$
- ▶ B computa  $D_{SKb}(M') = M$  donde D es un algoritmo de desencriptación público
- ▶ B lee el mensaje en claro M



## 2.4.3 - Criptografía híbrida

---

- ▶ Se emplean ambos cifrados... ¿Cómo?

## 2.4.3 - Criptografía híbrida

---

- ▶ Se emplean ambos cifrados
  - ▶ Se usa un cifrado asimétrico para enviar la clave del cifrado simétrico al destino usando la clave pública del destino
  - ▶ Se usa el cifrado simétrico para encriptar luego los mensajes
- ▶ Ejemplos
  - ▶ PGP, GnuPG, TLS y SSH usan un sistema de cifrado híbrido
  - ▶ La clave de la comunicación (clave simétrica) es cifrada con la clave pública del destino y el mensaje es cifrado con la clave simétrica. Se junta todo en un mismo paquete y se envía
  - ▶ El destino usa su clave privada para descifrar la clave simétrica y luego descifra el mensaje con esta
  - ▶ De esta forma la clave puede cambiar por cada comunicación

## 2.4.3 - Criptografía híbrida

---

- ▶ **Se emplean ambos cifrados**
  - ▶ Se usa un cifrado asimétrico para enviar la clave del cifrado simétrico al destino usando la clave pública del destino
  - ▶ Se usa el cifrado simétrico para encriptar luego los mensajes
- ▶ **¿Por qué?**
  - ▶ El cifrado asimétrico es seguro pero computacionalmente más complejo de ejecutar que el cifrado simétrico
  - ▶ Cifrar con el método asimétrico puede resultar prohibitivo (muy lento, inaceptable) cuando hay que intercambiar grandes cantidades de información

# Tema 2. Índice

---

1. Motivación
2. Definición de criptografía
3. Criptosistemas históricos
4. Tipos
  1. Criptografía privada
  2. Criptografía pública
5. Firma digital
6. Algunos principios matemáticos
7. Algoritmos más conocidos

## 2.5 - Firma digital

---

- ▶ ¿Cuál es el objetivo de una firma?
- ▶ ¿Y de una firma digital?

## 2.5 - Firma digital

---

- ▶ ¿Cuál es el objetivo de una firma?
- ▶ ¿Y de una firma digital?
- ▶ Se necesita verificar que la firma es autentica, es decir ha firmado realmente la persona que dice que ha firmado

## 2.5 - Firma digital

- ▶ ¿Cuál es el objetivo de una firma?
- ▶ ¿Y de una firma digital?
- ▶ **Se necesita verificar que la firma firmado realmente la persona que la realizó.**

### VISTO BUENO DEL INFORME ANUAL DE EVALUACIÓN

#### I. INFORME DEL DIRECTOR/A

Valoración de la consecución de los objetivos por parte del beneficiario/a durante la anualidad a la que se refiere este informe:

- Favorable: se aconseja la continuidad de la ayuda  
 NO favorable: NO se aconseja la continuidad de la ayuda

Motivación del informe NO favorable:

Grado aproximado de consecución de los objetivos marcados para la anualidad objeto de este informe:

	Excelente	Notable	Aceptable	Insuficiente
Metodología	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Tareas y resultados	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Programa formativo	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Motivación de la calificación:

La valoración del desarrollo de la tesis doctoral hasta este punto es muy positiva. Cabe destacar las numerosas colaboraciones lideradas por el beneficiario con instituciones tanto nacionales (p.e., Telefónica I+D, Fundación i2CAT, ATOS Origin) como internacionales (p.e., Predictive Network Solutions, Brno University of Technology), las cuáles han permitido la preparación de un número substancial de artículos de investigación para su publicación en revistas de prestigio y para su presentación en conferencias. Un ejemplo de estos es el artículo aceptado en la revista indexada en los JCR European Transactions on Telecommunications, o las ponencias en IEEE GLOBECOM o en el Workshop NetCloud 2016.

Firma electrónica del director/a:

Una vez firmado, debe enviar este documento a la COMISIÓN ACADÉMICA para que cumplimente y firme electrónicamente la página 8.  
Es importante que al firmar NO bloquee el documento.



## 2.5 - Firma digital

- ▶ ¿Cuál es el objetivo de una firma?
- ▶ ¿Y de una firma digital?
- ▶ **Se necesita verificar que la firma firmado realmente la persona que la realizó**

### VISTO BUENO DEL INFORME ANUAL DE EVALUACIÓN

#### I. INFORME DEL DIRECTOR/A

Valoración de la consecución de los objetivos por parte del beneficiario/a durante la anualidad a la que se refiere este informe:

- Favorable: se aconseja la continuidad de la ayuda  
 NO favorable: NO se aconseja la continuidad de la ayuda

Motivación del informe NO favorable:

Grado aproximado de consecución de los objetivos marcados para la anualidad objeto de este informe:

	Excelente	Notable	Aceptable	Insuficiente
Metodología	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Tareas y resultados	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Programa formativo	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Motivación de la calificación:

La valoración del desarrollo de la tesis doctoral hasta este punto es muy positiva. Cabe destacar las numerosas colaboraciones lideradas por el beneficiario con instituciones tanto nacionales (p.e., Telefónica I+D, Fundación i2CAT, ATOS Origin) como internacionales (p.e., Predictive Network Solutions, Brno University of Technology), las cuáles han permitido la preparación de un número substancial de artículos de investigación para su publicación en revistas de prestigio y para su presentación en conferencias. Un ejemplo de estos es el artículo aceptado en la revista indexada en los JCR European Transactions on Telecommunications, o las ponencias en IEEE GLOBECOM o en el Workshop NetCloud 2016.

Firma electrónica del director/a:

Una vez firmado, debe enviar este documento a la COMISIÓN ACADÉMICA para que cumplimente y firme electrónicamente la página 8.  
Es importante que al firmar NO bloquee el ordenador.



## 2.5 - Firma digital

---

- ▶ La idea de clave privada/pública de la criptografía asimétrica también se usa para la firma digital pero de forma inversa
- ▶ Si  $U$  quiere firmar un mensaje/documento  $M$ , simplemente aplica el algoritmo  $E$  con su clave privada de forma que el mensaje firmado es  $S = E_{SK_u}(M)$
- ▶ Para verificar que el que ha firmado es realmente  $U$ , cualquier usuario puede aplicar el algoritmo de desencriptación usando la clave pública de  $U$  sobre el mensaje cifrado y comparar el resultado con el mensaje no cifrado, es decir verificar que  $D_{PK_u}(S) = M$

# Tema 2. Índice

---

1. Motivación
2. Definición de criptografía
3. Criptosistemas históricos
4. Tipos
  1. Criptografía privada
  2. Criptografía pública
5. Firma digital
6. Algunos principios matemáticos
7. Algoritmos más conocidos

## 2.6 – Algunos principios matemáticos

---

- ▶ Como se puede garantizar que no se descubra una clave secreta a partir de valores que se intercambien los dos extremos
- ▶ Dos soluciones principales
  - ▶ Logarítmica/exponenciación discreta
  - ▶ Curvas elípticas

## 2.6 – Algunos principios matemáticos

---

- ▶ Como se puede garantizar que no se descubra una clave secreta a partir de valores que se intercambien los dos extremos
- ▶ Dos soluciones principales
  - ▶ Logarítmica/exponenciación discreta
  - ▶ Curvas elípticas

## 2.6.1 – Logarítmica/exponenciación discreta

---

- ▶ Se define un grupo cíclico finito  $G$ , de orden  $n$
- ▶ Se conoce como logaritmo discreto de  $y$  en base  $g$  (con  $y, g \in G$ ) a la solución  $x$  de la ecuación  $g^x = y$
- ▶ O de forma equivalente

$$x = \log_g(y) \Leftrightarrow g^x = y$$

## 2.6.1 – Logarítmica/exponenciación discreta

---

- ▶ Se define un grupo cíclico finito  $G$ , de orden  $n$
- ▶ Se conoce como logaritmo discreto de  $y$  en base  $g$  (con  $y, g \in G$ ) a la solución  $x$  de la ecuación  $g^x = y$
- ▶ O de forma equivalente

$$x = \log_g(y) \Leftrightarrow g^x = y$$

- ▶ La exponenciación discreta es sencilla en términos computacionales (existe un algoritmo que veremos)
- ▶ El logaritmo discreto en cambio es irresoluble en un tiempo razonable y no da un resultado único (da varias soluciones posibles)
- ▶ Estas operaciones crean lo que se llaman Trapdoor Function

## 2.6.1 – Logarítmica/exponenciación discreta

### Ejemplo de exponenciación discreta

---

- ▶ Se define un grupo cíclico finito  $G$ , de orden  $n = 23$ 
  - ▶ Todos los números del 0 al 22
- ▶  $g = 4, x = 5$
- ▶ Calcular  $y$  como  $g^x = y$ 
  - ▶  $g,y$  deben pertenecer al grupo cíclico  $G$

## 2.6.1 – Logarítmica/exponenciación discreta

### Ejemplo de exponenciación discreta

---

- ▶ Se define un grupo cíclico finito  $G$ , de orden  $n = 23$ 
  - ▶ Todos los números del 0 al 22
- ▶  $g = 4, x = 5$
- ▶ Calcular  $y$  como  $g^x = y$ 
  - ▶  $g, y$  deben pertenecer al grupo cíclico  $G$
- ▶  $y = 4^5 = 1024$

## 2.6.1 – Logarítmica/exponenciación discreta

### Ejemplo de exponenciación discreta

---

- ▶ Se define un grupo cíclico finito  $G$ , de orden  $n = 23$ 
  - ▶ Todos los números del 0 al 22
- ▶  $g = 4, x = 5$
- ▶ Calcular  $y$  como  $g^x = y$ 
  - ▶  $g, y$  deben pertenecer al grupo cíclico  $G$
- ▶  $y = 4^5 = 1024$ , pero 1024 no pertenece a  $G$

## 2.6.1 – Logarítmica/exponenciación discreta

### Ejemplo de exponenciación discreta

---

- ▶ Se define un grupo cíclico finito  $G$ , de orden  $n = 23$ 
  - ▶ Todos los números del 0 al 22
- ▶  $g = 4, x = 5$
- ▶ Calcular  $y$  como  $g^x = y$ 
  - ▶  $g, y$  deben pertenecer al grupo cíclico  $G$
- ▶  $y = 4^5 = 1024$ , pero 1024 no pertenece a  $G$
- ▶  $G$  es un grupo cíclico finito, quiere decir que hay que hacer el modulo 23 de 1024
  - ▶ Es decir dividir 1024 por 23 y quedarse con el resto
- ▶  $y = 1024 \text{ mod } 23 = 12$

## 2.6.1 – Logarítmica/exponenciación discreta

### Ejemplo de exponenciación discreta

---

- ▶ Se define un grupo cíclico finito  $G$ , de orden  $n = 23$ 
  - ▶ Todos los números del 0 al 22
- ▶  $g = 4, x = 5$
- ▶ Calcular  $y$  como  $g^x = y$  **Easy-Peasy!!!**
  - ▶  $g, y$  deben pertenecer al grupo cíclico  $G$
- ▶  $y = 4^5 = 1024$ , pero 1024 no pertenece a  $G$
- ▶  $G$  es un grupo cíclico finito, quiere decir que hay que hacer el modulo 23 de 1024
  - ▶ Es decir dividir 1024 por 23 y quedarse con el resto
- ▶  $y = 1024 \text{ mod } 23 = 12$

## 2.6.1 – Logarítmica/exponenciación discreta

### Ejemplo de logaritmo discreto

---

- ▶ Se define un grupo cíclico finito  $G$ , de orden  $n = 15$
- ▶ Ahora el logaritmo con  $g = 2, y = 1$
- ▶ Determinar  $x$  como  $g^x = y$ , (con  $g,y \in G$ )

## 2.6.1 – Logarítmica/exponenciación discreta

### Ejemplo de logaritmo discreto

---

- ▶ Se define un grupo cíclico finito  $G$ , de orden  $n = 15$
  - ▶ Ahora el logaritmo con  $g = 2, y = 1$
  - ▶ Determinar  $x$  como  $g^x = y, (con g,y \in G)$
- 
- ▶  $x = \log_g(y) = \log_2 1 = 0 \rightarrow$  efectivamente  $2^0 = 1$

## 2.6.1 – Logarítmica/exponenciación discreta

### Ejemplo de logaritmo discreto

---

- ▶ Se define un grupo cíclico finito  $G$ , de orden  $n = 15$
  - ▶ Ahora el logaritmo con  $g = 2, y = 1$
  - ▶ Determinar  $x$  como  $g^x = y, (con g,y \in G)$
- 
- ▶  $x = \log_g(y) = \log_2 1 = 0 \rightarrow$  efectivamente  $2^0 = 1$
  - ▶ Ya esta, fácil, no?

## 2.6.1 – Logarítmica/exponenciación discreta

### Ejemplo de logaritmo discreto

---

- ▶ Se define un grupo cíclico finito  $G$ , de orden  $n = 15$
  - ▶ Ahora el logaritmo con  $g = 2, y = 1$
  - ▶ Determinar  $x$  como  $g^x = y, (\text{con } g, y \in G)$
- 
- ▶  $x = \log_g(y) = \log_2 1 = 0 \rightarrow \text{efectivamente } 2^0 = 1$
  - ▶ Ya esta, fácil, no? Un momento...

## 2.6.1 – Logarítmica/exponenciación discreta

### Ejemplo de logaritmo discreto

---

- ▶ Se define un grupo cíclico finito  $G$ , de orden  $n = 15$
  - ▶ Ahora el logaritmo con  $g = 2, y = 1$
  - ▶ Determinar  $x$  como  $g^x = y, (con g,y \in G)$
- 
- ▶  $x = \log_g(y) = \log_2 1 = 0 \rightarrow$  efectivamente  $2^0 = 1$
  - ▶ Ya esta, fácil, no? Un momento... hablamos de un grupo cíclico...

## 2.6.1 – Logarítmica/exponenciación discreta

### Ejemplo de logaritmo discreto

---

- ▶ Se define un grupo cíclico finito  $G$ , de orden  $n = 15$
  - ▶ Ahora el logaritmo con  $g = 2, y = 1$
  - ▶ Determinar  $x$  como  $g^x = y, (con g, y \in G)$
- 
- ▶  $x = \log_g(y) = \log_2 1 = 0 \rightarrow$  efectivamente  $2^0 = 1$
  - ▶ Ya esta, fácil, no? Un momento... hablamos de un grupo cíclico...
  - ▶ También  $x = 4$  es una solución ya que  $2^4 = 16 \text{ mod } 15 = 1$

## 2.6.1 – Logarítmica/exponenciación discreta

### Ejemplo de logaritmo discreto

---

- ▶ Se define un grupo cíclico finito  $G$ , de orden  $n = 15$
  - ▶ Ahora el logaritmo con  $g = 2, y = 1$
  - ▶ Determinar  $x$  como  $g^x = y, (con g,y \in G)$
- 
- ▶  $x = \log_g(y) = \log_2 1 = 0 \rightarrow$  efectivamente  $2^0 = 1$
  - ▶ Ya esta, fácil, no? Un momento... hablamos de un grupo cíclico...
  - ▶ También  $x = 4$  es una solución ya que  $2^4 = 16 \text{ mod } 15 = 1$
  - ▶ También  $x = 8$  es una solución ya que  $2^8 = 256 \text{ mod } 15 = 1$
  - ▶ También  $x = 12$  es una solución

## 2.6.1 – Logarítmica/exponenciación discreta

### Ejemplo de logaritmo discreto

---

- ▶ Se define un grupo cíclico finito  $G$ , de orden  $n = 15$
  - ▶ Ahora el logaritmo con  $g = 2, y = 1$
  - ▶ Determinar  $x$  como  $g^x = y, (con g,y \in G)$
- 
- ▶  $x = \log_g(y) = \log_2 1 = 0 \rightarrow$  efectivamente  $2^0 = 1$
  - ▶ Ya esta, fácil, no? Un momento... hablamos de un grupo cíclico...
  - ▶  $x = 0, 4, 8, 12, \dots$  son todas soluciones validas... ¿cuántas hay?

## 2.6.1 – Logarítmica/exponenciación discreta

### Ejemplo de logaritmo discreto

---

- ▶ Se define un grupo cíclico finito  $G$ , de orden  $n = 15$
  - ▶ Ahora el logaritmo con  $g = 2, y = 1$
  - ▶ Determinar  $x$  como  $g^x = y, (con g,y \in G)$
- 
- ▶  $x = \log_g(y) = \log_2 1 = 0 \rightarrow$  efectivamente  $2^0 = 1$
  - ▶ Ya esta, fácil, no? Un momento... hablamos de un grupo cíclico...
  - ▶  $x = 0, 4, 8, 12, \dots$  son todas soluciones validas... ¿cuántas hay? **Número muy elevado si se elige bien  $G$**

## 2.6.1 – Logarítmica/exponenciación discreta

### Ejemplo de logaritmo discreto

---

- ▶ Se define un grupo cíclico finito  $G$ , de orden  $n = 15$
  - ▶ Ahora el logaritmo con  $g = 2, y = 1$
  - ▶ Determinar  $x$  como  $g^x = y, (con g,y \in G)$
- 
- ▶  $x = \log_g(y) = \log_2 1 = 0 \rightarrow$  efectivamente  $2^0 = 1$
  - ▶ Ya esta, fácil, no? Un momento... hablamos de un grupo cíclico...
  - ▶  $x = 0, 4, 8, 12, \dots$  son todas soluciones validas... ¿cuántas hay? **Número muy elevado si se elige bien  $G$**
  - ▶ La única solución posible es probarlas todas
    - ▶ Complejidad exponencial respecto al número de dígitos de  $n$
    - ▶ No es computacionalmente viable con ordenadores digitales

## 2.6 – Algunos principios matemáticos

---

- ▶ Como se puede garantizar que no se descubra una clave secreta a partir de valores que se intercambien los dos extremos
- ▶ Dos soluciones principales
  - ▶ Logarítmica/exponenciación discreta
  - ▶ Curvas elípticas → veremos luego

## 2.7 – Algoritmos más conocidos

---

- ▶ Diffie-Hellman
  - ▶ Exponenciación binaria
- ▶ RSA
  - ▶ Números coprimos
  - ▶ Multiplicador modular inverso
- ▶ ElGamal
- ▶ RSA para Firma Digital
- ▶ Función Hash

## 2.7 – Algoritmos más conocidos

---

- ▶ **Diffie-Hellman**
  - ▶ Exponenciación binaria
- ▶ **RSA**
  - ▶ Números coprimos
  - ▶ Multiplicador modular inverso
- ▶ **ElGamal**
- ▶ **RSA para Firma Digital**
- ▶ **Función Hash**

## 2.7.1 - Diffie-Hellman

---

- ▶ Usado para
  - ▶ Generación de clave privada en la criptográfica simétrica
  - ▶ Parte del mecanismo de cifrado asimétrico de algunos algoritmos

## 2.7.1 - Diffie-Hellman

---

- ▶ Mecanismo usado para compartir una clave privada
- ▶ Los usuarios Alice y Bob
  - ▶ Eligen un grupo cíclico finito  $G$  de orden  $n$  ( $n$  debe ser número primo)
  - ▶ Un generador  $\alpha \in G$
  - ▶ Estos valores se pueden intercambiar sin cifrar

## 2.7.1 - Diffie-Hellman

---

- ▶ Mecanismo usado para compartir una clave privada
- ▶ Los usuarios Alice y Bob
  - ▶ Eligen un grupo cíclico finito  $G$  de orden  $n$  ( $n$  debe ser número primo)
  - ▶ Un generador  $\alpha \in G$
  - ▶ Estos valores se pueden intercambiar sin cifrar
- ▶ Alice
  - ▶ Elige un número  $a \in G$
  - ▶ Computa el valor  $\alpha^a \text{ mod } n$
  - ▶ Envía el resultado a Bob
- ▶ Bob
  - ▶ Elige un número  $b \in G$
  - ▶ Computa el valor  $\alpha^b \text{ mod } n$
  - ▶ Envía el resultado a Alice

## 2.7.1 - Diffie-Hellman

---

- ▶ Mecanismo usado para compartir una clave privada
- ▶ Los usuarios Alice y Bob
  - ▶ Eligen un grupo cíclico finito  $G$  de orden  $n$  ( $n$  debe ser número primo)
  - ▶ Un generador  $\alpha \in G$
  - ▶ Estos valores se pueden intercambiar sin cifrar
- ▶ Alice
  - ▶ Elige un número  $a \in G$
  - ▶ Computa el valor  $\alpha^a \text{ mod } n$
  - ▶ Envía el resultado a Bob
- ▶ Bob
  - ▶ Elige un número  $b \in G$
  - ▶ Computa el valor  $\alpha^b \text{ mod } n$
  - ▶ Envía el resultado a Alice
- ▶ Alice
  - ▶ Recibe  $\alpha^b \text{ mod } n$
  - ▶ Computa  $(\alpha^b \text{ mod } n)^a \text{ mod } n = X$
- ▶ Bob
  - ▶ Recibe  $\alpha^a \text{ mod } n$
  - ▶ Computa  $(\alpha^a \text{ mod } n)^b \text{ mod } n = X$

## 2.7.1 - Diffie-Hellman

---

- ▶ Mecanismo usado para compartir una clave privada
- ▶ Los usuarios Alice y Bob
  - ▶ Eligen un grupo cíclico finito  $G$  de orden  $n$  ( $n$  debe ser número primo)
  - ▶ Un generador  $\alpha \in G$
  - ▶ Estos valores se pueden intercambiar sin cifrar
- ▶ Alice
  - ▶ Elige un número  $a \in G$
  - ▶ Computa el valor  $\alpha^a \text{ mod } n$
  - ▶ Envía el resultado a Bob
- ▶ Bob
  - ▶ Elige un número  $b \in G$
  - ▶ Computa el valor  $\alpha^b \text{ mod } n$
  - ▶ Envía el resultado a Alice
- ▶ Alice
  - ▶ Recibe  $\alpha^b \text{ mod } n$
  - ▶ Computa  $(\alpha^b \text{ mod } n)^a \text{ mod } n = X$
- ▶ Bob
  - ▶ Recibe  $\alpha^a \text{ mod } n$
  - ▶ Computa  $(\alpha^a \text{ mod } n)^b \text{ mod } n = X$

## 2.7.1 - Diffie-Hellman

- ▶ Mecanismo usado para compartir una clave privada
  - ▶ Los usuarios Alice y Bob
    - ▶ Eligen un grupo cíclico finito  $G$  de orden  $n$  ( $n$  debe ser número primo)
    - ▶ Un generador  $\alpha \in G$
    - ▶ Estos valores se pueden intercambiar sin cifrar
  - ▶ Alice
    - ▶ Elige un número  $a \in G$
    - ▶ Computa el valor  $\alpha^a \text{ mod } n$
    - ▶ Envía el resultado a Bob
  - ▶ Bob
    - ▶ Elige un número  $b \in G$
    - ▶ Computa el valor  $\alpha^b \text{ mod } n$
    - ▶ Envía el resultado a Alice
- Números privados,  
solo Alice conoce  $a$ , solo Bob conoce  $b$

## 2.7.1 - Diffie-Hellman

### Ejemplo

- ▶ Mecanismo usado para compartir una clave privada
- ▶ Los usuarios A y B
  - ▶ Eligen un grupo cíclico finito  $G$  de orden  $n = 53$
  - ▶ Un generador  $\alpha \in G = 2$
  - ▶ Estos valores se pueden intercambiar sin cifrar
- ▶ A
  - ▶ Elige un número  $a \in G = 29$
  - ▶ Computa el valor  $2^{29} \text{ mod } 53 = 45$
  - ▶ Envía 45 el resultado a B
- ▶ B
  - ▶ Recibe 45
  - ▶ Computa  $45^{19} \text{ mod } 53 = 21$
- ▶ A
  - ▶ Recibe 12
  - ▶ Computa  $12^{29} \text{ mod } 53 = 21$
- ▶ B
  - ▶ Recibe 12
  - ▶ Computa  $12^{19} \text{ mod } 53 = 21$
- ▶ B
  - ▶ Envía 12 el resultado a A

## 2.7.1 - Diffie-Hellman

### Ejemplo

- |  |   |
|--|---|
| <ul style="list-style-type: none"><li>▶ A<ul style="list-style-type: none"><li>▶ Elige un número <math>a \in G = 29</math></li><li>▶ Computa el valor <math>2^{29} \text{ mod } 53 = 45</math></li><li>▶ Envía 45 el resultado a B</li></ul></li><li>▶ B<ul style="list-style-type: none"><li>▶ Elige un número <math>b \in G = 19</math></li><li>▶ Computa el valor <math>2^{19} \text{ mod } 53 = 12</math></li><li>▶ Envía 12 el resultado a A</li></ul></li><li>▶ Se usa el principio matemático visto antes</li><li>▶ Exponente fácil, logaritmo intratable computacionalmente</li><li>▶ Alguien podría perfectamente conocer 2, 12 y 53, pero no puede saber que 19 es el valor correcto, podrían ser otros<ul style="list-style-type: none"><li>▶ Habría que saber determinar <math>x</math> de <math>2^x \text{ mod } 53 = 12</math></li><li>▶ Uno de estos valores de <math>x</math> es 19, pero no es el único</li></ul></li></ul> | <ul style="list-style-type: none"><li>▶ A<ul style="list-style-type: none"><li>▶ Recibe 12</li><li>▶ Computa <math>12^{29} \text{ mod } 53 = 21</math></li></ul></li><li>▶ B<ul style="list-style-type: none"><li>▶ Recibe 45</li><li>▶ Computa <math>45^{19} \text{ mod } 53 = 21</math></li></ul></li></ul> |
|--|---|

## 2.7.1 - Diffie-Hellman

### Posibles ataques: ataque pasivo

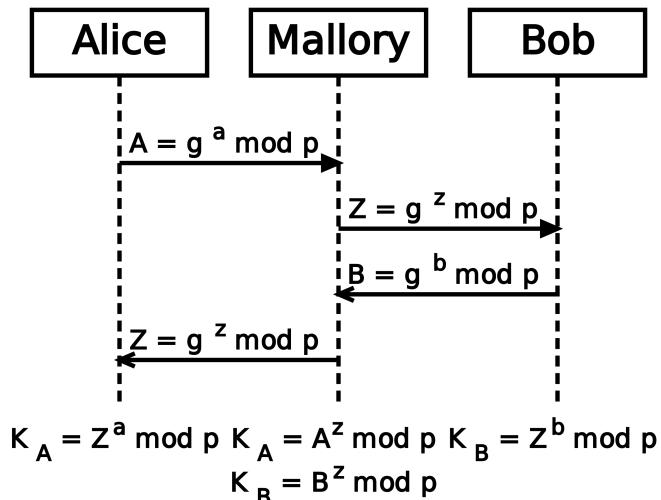
---

- ▶ Obtener  $a$  o  $b$  a partir de los otros valores calculando el logaritmo discreto y probando todas las soluciones
- ▶ Este es un problema (que se cree de momento) intratable computacionalmente siempre que el modulo  $n$ 
  - ▶ Sea un número primo grande >200 dígitos (vulnerabilidad logjam)
    - ▶ Se puede usar el algoritmo number field sieve
    - ▶ Este algoritmo consiste de 4 pasos (computacionalmente complejos e intratable)
    - ▶ Pero si se conoce  $G$ , se pueden pre-calcular los 3 primeros pasos (y tardar lo que tarda) y luego solo hay que determinar el último paso (menos complejo) con los valores públicos que se intercambian  $A$  y  $B$
  - ▶ Que no cumplan ciertas características debilitantes
    - ▶ Se usan número primo de Sophie Germain, del estilo  $n = 2q + 1$ , donde  $n$  y  $q$  son primos
    - ▶ Valores conocidos de  $q$  son  $2, 3, 5, 11, \dots, 2618163402417 \times 2^{1290000} - 1$

## 2.7.1 - Diffie-Hellman

Posibles ataques: ataque activo

- ▶ No proporciona autenticación, se supone que los dos extremos han pasado anteriormente por una etapa de certificación
- ▶ Si no se usa, es posible hacer un ataque de tipo Man-in-the-middle
  - ▶ Malory se hace pasar por Bob cuando habla con Alice (y establecen una clave privada común)
  - ▶ Malory se hace pasar por Alice cuando habla con Bob (y establecen una clave privada común)



Fuentes imagen: Wikipedia

## 2.7.1 - Diffie-Hellman

### Ejercicio

---

- ▶ Mecanismo usado para compartir una clave privada
- ▶ Los usuarios A y B
  - ▶ Eligen un grupo cíclico finito  $G$  de orden  $n = 11$
  - ▶ Un generador  $\alpha \in G = 3$
  - ▶ Estos valores se pueden intercambiar sin cifrar
- ▶ A
  - ▶ Elige un número  $a \in G = 7$
- ▶ B
  - ▶ Elige un número  $b \in G = 8$

## 2.7.1 - Diffie-Hellman

### Ejercicio

---

- ▶ Mecanismo usado para compartir una clave privada
- ▶ Los usuarios A y B
  - ▶ Eligen un grupo cíclico finito  $G$  de orden  $n = 11$
  - ▶ Un generador  $\alpha \in G = 3$
  - ▶ Estos valores se pueden intercambiar sin cifrar
- ▶ A
  - ▶ Elige un número  $a \in G = 7$
  - ▶ Computa el valor  $3^7 \text{ mod } 11 = 9$
  - ▶ Envía 9 el resultado a B
- ▶ B
  - ▶ Recibe 5
  - ▶ Computa  $5^7 \text{ mod } 11 = 3$
- ▶ A
  - ▶ Recibe 9
  - ▶ Computa  $9^8 \text{ mod } 11 = 3$
- ▶ B
  - ▶ Envía 5 el resultado a A

## 2.7.1 - Diffie-Hellman

### Ejercicio

---

- ▶ Mecanismo usado para compartir una clave privada
- ▶ Los usuarios A y B
  - ▶ Eligen un grupo cíclico finito  $G$  de orden  $n = 211$
  - ▶ Un generador  $\alpha \in G = 5$
  - ▶ Estos valores se pueden intercambiar sin cifrar
- ▶ A
  - ▶ Elige un número  $a \in G = 27$
- ▶ B
  - ▶ Elige un número  $b \in G = 20$

## 2.7.1 - Diffie-Hellman

### Ejercicio

---

- ▶ Mecanismo usado para compartir una clave privada
- ▶ Los usuarios A y B
  - ▶ Eligen un grupo cíclico finito  $G$  de orden  $n = 211$
  - ▶ Un generador  $\alpha \in G = 5$
  - ▶ Estos valores se pueden intercambiar sin cifrar
- ▶ A
  - ▶ Elige un número  $a \in G = 27$
  - ▶ Computa el valor  $5^{27} \text{ mod } 211 = ?$
- ▶ B
  - ▶ Elige un número  $b \in G = 20$
  - ▶ Computa el valor  $5^{20} \text{ mod } 211 = ?$

## 2.7.1 - Diffie-Hellman

¿Algún problema?

---

- ▶ Calcular  $5^{27} \text{ mod } 211$
- ▶ Complicado cuando los números son demasiado grandes
- ▶ Pero ...

## 2.7.1 - Diffie-Hellman

¿Algún problema?

---

- ▶ Calcular  $5^{27} \text{ mod } 211$
- ▶ Complicado cuando los números son demasiado grandes
- ▶ Pero ...
- ▶ La aritmética modular simplifica estos cálculos

## 2.7.1 - Diffie-Hellman

¿Algún problema?

---

- ▶ Calcular  $5^{27} \text{ mod } 211$
- ▶ Complicado cuando los números son demasiado grandes
- ▶ Pero ...
- ▶ La aritmética modular simplifica estos cálculos
  - ▶ Algoritmo de exponenciación binaria

## 2.7.1 - Exponenciación binaria

---

- ▶ Se usa para computar números exponenciales sin tratar números excesivamente grandes

---

**Exponentiation by squaring (a,z,n)**  $x = a^z \bmod n$

---

**begin**

```
|   x = 1;  
|   z1 = binary representation of z;  
|   // starting by the most significant bit  
foreach bit zi1 ∈ z1 do  
    |   x = x2 mod n;  
    |   // multiply x by a if zi1 is equal to one  
    |   if zi1 == 1 then  
    |       |   x = x · a mod n  
|  
return x
```

---

## 2.7.1 - Exponenciación binaria

- ▶ Hay que calcular  $5^{27} \bmod 211$
- ▶ Se calcula el binario de 27 que es 11011

Exponentiation by squaring (a,z,n)  $x = a^z \bmod n$

```
begin
    x = 1;
    z1 = binary representation of z;
    // starting by the most significant bit
    foreach bit z1i ∈ z1 do
        x = x2 mod n;
        // multiply x by a if z1i is equal to one
        if z1i == 1 then
            x = x · a mod n
    return x
```

## 2.7.1 - Exponenciación binaria

- ▶ Hay que calcular  $5^{27} \bmod 211$
- ▶ Se calcula el binario de 27 que es 11011
- ▶ Al principio  $x = 1$
- ▶  $z^1 = 11011$

Exponentiation by squaring (a,z,n)  $x = a^z \bmod n$

```
begin
    x = 1;
    z1 = binary representation of z;
    // starting by the most significant bit
    foreach bit zi1 ∈ z1 do
        x = x2 mod n;
        // multiply x by a if zi1 is equal to one
        if zi1 == 1 then
            x = x · a mod n
    return x
```

## 2.7.1 - Exponenciación binaria

- ▶ Hay que calcular  $5^{27} \bmod 211$
- ▶ Se calcula el binario de 27 que es 11011
- ▶ Al principio  $x = 1$
- ▶  $z^1 = 11011$
- ▶ Bucle  $i = 0$  to 4
  - ▶  $z^1_0 = 1$
  - ▶  $x = 1^2 \bmod 211 = 1$  (operación de squaring)
  - ▶ Si  $z^1_0 == 1$  (cierto)
    - ▶  $x = 1 \times 5 \bmod 211 = 5$  (operación de multiply)

Exponentiation by squaring (a,z,n)  $x = a^z \bmod n$

```
begin
    x = 1;
    z1 = binary representation of z;
    // starting by the most significant bit
    foreach bit  $z_i^1 \in z^1$  do
        x = x2 mod n;
        // multiply x by a if  $z_i^1$  is equal to one
        if  $z_i^1 == 1$  then
            x = x · a mod n
    return x
```

## 2.7.1 - Exponenciación binaria

- ▶ Hay que calcular  $5^{27} \bmod 211$
- ▶ Se calcula el binario de 27 que es 11011
- ▶ Al principio  $x = 1$
- ▶  $z^1 = 1\boxed{1}011$
- ▶ Bucle  $i = 0$  to 4
  - ▶  $z^1_1 = 1$
  - ▶  $x = 5^2 \bmod 211 = 25$  (operación de squaring)
  - ▶ Si  $z^1_1 == 1$  (cierto)
    - ▶  $x = 25 \times 5 \bmod 211 = 125$  (operación de multiply)

Exponentiation by squaring (a,z,n)  $x = a^z \bmod n$

```
begin
    x = 1;
    z1 = binary representation of z;
    // starting by the most significant bit
    foreach bit  $z_i^1 \in z^1$  do
        x =  $x^2 \bmod n$ ;
        // multiply x by a if  $z_i^1$  is equal to one
        if  $z_i^1 == 1$  then
            x =  $x \cdot a \bmod n$ 
    return x
```

## 2.7.1 - Exponenciación binaria

- ▶ Hay que calcular  $5^{27} \bmod 211$
- ▶ Se calcula el binario de 27 que es **11011**
- ▶ Al principio  $x = 1$
- ▶  $z^1 = 1 \boxed{1} 0 1 1$
- ▶ Bucle  $i = 0$  to 4
  - ▶  $z^1_2 = 0$
  - ▶  $x = 125^2 \bmod 211 = 15625 \bmod 211 = 11$  (operación de squaring)
  - ▶ Si  $z^1_2 == 1$  (falso)

Exponentiation by squaring (a,z,n)  $x = a^z \bmod n$

```
begin
    x = 1;
    z1 = binary representation of z;
    // starting by the most significant bit
    foreach bit  $z_i^1 \in z^1$  do
        x = x2 mod n;
        // multiply x by a if  $z_i^1$  is equal to one
        if  $z_i^1 == 1$  then
            x = x · a mod n
    return x
```

## 2.7.1 - Exponenciación binaria

- ▶ Hay que calcular  $5^{27} \bmod 211$
- ▶ Se calcula el binario de 27 que es 11011
- ▶ Al principio  $x = 1$
- ▶  $z^1 = 110\boxed{1}$
- ▶ Bucle  $i = 0$  to 4
  - ▶  $z^1_3 = 1$
  - ▶  $x = 11^2 \bmod 211 = 121$  (operación de squaring)
  - ▶ Si  $z^1_3 == 1$  (cierto)
    - ▶  $x = 121 \times 5 \bmod 217 = 605 \bmod 211 = 183$  (operación de multiply)

Exponentiation by squaring (a,z,n)  $x = a^z \bmod n$

```
begin
    x = 1;
    z1 = binary representation of z;
    // starting by the most significant bit
    foreach bit  $z_i^1 \in z^1$  do
        x = x2 mod n;
        // multiply x by a if  $z_i^1$  is equal to one
        if  $z_i^1 == 1$  then
            x = x · a mod n
    return x
```

## 2.7.1 - Exponenciación binaria

- ▶ Hay que calcular  $5^{27} \bmod 211$
- ▶ Se calcula el binario de 27 que es 11011
- ▶ Al principio  $x = 1$
- ▶  $z^1 = 11011$
- ▶ Bucle  $i = 0$  to 4
  - ▶  $z^1_4 = 1$
  - ▶  $x = 183^2 \bmod 211 = 33489 \bmod 211 = 151$  (operación de squaring)
  - ▶ Si  $z^1_4 == 1$  (cierto)
    - ▶  $x = 151 \times 5 \bmod 211 = 755 \bmod 211 = 122$  (operación de multiply)
- ▶ El resultado de  $5^{27} \bmod 211$  es 122

---

Exponentiation by squaring (a,z,n)  $x = a^z \bmod n$

```
begin
    x = 1;
    z1 = binary representation of z;
    // starting by the most significant bit
    foreach bit  $z_i^1 \in z^1$  do
        x =  $x^2 \bmod n$ ;
        // multiply x by a if  $z_i^1$  is equal to one
        if  $z_i^1 == 1$  then
            x =  $x \cdot a \bmod n$ 
    return x
```

---

## 2.7.1 - Exponenciación binaria

- ▶ Hay que calcular  $5^{27} \bmod 211$
- ▶ El resultado de  $5^{27} \bmod 211$  es 122

- ▶ Existen herramientas (online) que, usando este algoritmo (y veremos otros), pueden calcular este resultado

- ▶ Un ejemplo es

<https://www.dcode.fr/modular-exponentiation>

- ▶ Podéis usar estas herramientas para comprobar vuestras cálculos mientras os preparáis para los controles

---

Exponentiation by squaring ( $a,z,n$ )  $x = a^z \bmod n$

```
begin
    x = 1;
    z1 = binary representation of z;
    // starting by the most significant bit
    foreach bit z1i ∈ z1 do
        x = x2 mod n;
        // multiply x by a if z1i is equal to one
        if z1i == 1 then
            x = x · a mod n
    return x
```

---

## 2.7.1 - Diffie-Hellman

### Ejercicio

---

- ▶ Mecanismo usado para compartir una clave privada
- ▶ Los usuarios A y B
  - ▶ Eligen un grupo finito  $G$  de orden  $n = 211$
  - ▶ Un generador  $\alpha \in G = 5$
  - ▶ Estos valores se pueden intercambiar sin cifrar
- ▶ A
  - ▶ Elige un número  $a \in G = 27$
  - ▶ Computa el valor  $5^{27} \text{ mod } 211 = 122$
  - ▶ Envía 122 el resultado a B
- ▶ B
  - ▶ Elige un número  $b \in G = 20$

## 2.7.1 - Diffie-Hellman

### Ejercicio

---

- ▶ Mecanismo usado para compartir una clave privada
- ▶ Los usuarios A y B
  - ▶ Eligen un grupo finito  $G$  de orden  $n = 211$
  - ▶ Un generador  $\alpha \in G = 5$
  - ▶ Estos valores se pueden intercambiar sin cifrar
- ▶ A
  - ▶ Elige un número  $a \in G = 27$
  - ▶ Computa el valor  $5^{27} \text{ mod } 211 = 122$
  - ▶ Envía 122 el resultado a B
- ▶ B
  - ▶ Recibe 148
  - ▶ Computa  $148^{27} \text{ mod } 211 = 144$
- ▶ A
  - ▶ B
    - ▶ Recibe 122
    - ▶ Computa  $122^{20} \text{ mod } 211 = 144$
- ▶ B
  - ▶ Calcula la clave privada

## 2.7 – Algoritmos más conocidos

---

- ▶ Diffie-Hellman
  - ▶ Exponenciación binaria
- ▶ RSA
  - ▶ Números coprimos
  - ▶ Multiplicador modular inverso
- ▶ ElGamal
- ▶ RSA para Firma Digital
- ▶ Función Hash

## 2.7.2 - RSA

---

- ▶ Generador de clave pública y privada en criptografía asimétrica
- ▶ Algoritmo descrito en 1977 por Ron Rivest, Adi Shamir y Leonard Adleman del MIT
  - ▶ RSA: Rivest, Shamir, Adleman
- ▶ Primer y más utilizado algoritmo de este tipo y es válido tanto para cifrar como para firmar digitalmente
- ▶ El funcionamiento se basa en el producto de dos números primos grandes elegidos al azar y mantenidos en secreto
- ▶ Actualmente estos números primos son del orden de  $10^{300}$  (cuanto más grandes, más difícil de encontrarlos)

## 2.7.2 - RSA

---

- ▶ Se eligen dos números primos muy grandes  $p$  y  $q$
- ▶ Se computa  $n = p \cdot q$ , donde  $n$  será la base del modulo (grupo  $\mathbb{Z}_n$ )
- ▶ Se computa la función de Euler  $\Phi(n) = (p-1) \cdot (q-1)$
- ▶ Se elige un entero  $e$  menor que  $\Phi(n)$  y que sea coprimo de  $\Phi(n)$ 
  - ▶  $e$  es el exponente de la clave publica
- ▶ Se determina  $d$  como el multiplicador modular inverso de  $e$  mod  $\Phi(n)$ 
  - ▶  $d$  es el exponente de la clave privada
- ▶ La clave publica es  $(n,e)$
- ▶ La clave privada es  $(n,d)$ 
  - ▶  $p, q$  y  $\Phi(n)$  también deben mantenerse privados

## 2.7.2 – RSA

### Cifrado/descifrado

---

- ▶ A tiene clave publica  $(n,e)$  y privada  $(n,d)$
- ▶ Si B quiere enviar un mensaje  $m$  a A,
- ▶ B busca la clave publica de A  $(n,e)$  y crea el mensaje encriptado c

$$c = m^e \bmod n$$

- ▶ A recibe el mensaje y desencripta usando su clave privada  $(n,d)$

$$m = c^d \bmod n$$

## 2.7.2 - RSA

---

- ▶ Se eligen dos números primos muy grandes  $p$  y  $q$
- ▶ Se computa  $n = p \cdot q$ , donde  $n$  será la base del modulo (grupo  $\mathbb{Z}_n$ )
- ▶ Se computa la función de Euler  $\Phi(n) = (p-1) \cdot (q-1)$
- ▶ Se elige un entero  $e$  menor que  $\Phi(n)$  y que sea coprimo de  $\Phi(n)$ 
  - ▶  $e$  es el exponente de la clave publica
- ▶ Se determina  $d$  como el multiplicador modular inverso de  $e$  mod  $\Phi(n)$ 
  - ▶  $d$  es el exponente de la clave privada
- ▶ La clave publica es  $(n,e)$
- ▶ La clave privada es  $(n,d)$ 
  - ▶  $p, q$  y  $\Phi(n)$  también deben mantenerse privados

## 2.7.2 – RSA

### Números coprimos

---

- ▶ Dos números  $a$  y  $b$  son coprimos si no tienen ningún factor primo en común
- ▶ Dicho de otra manera
  - ▶ Si no tienen otro divisor común más que 1
  - ▶ Equivalentemente son coprimos , si y solo si, su máximo común divisor es igual a 1,  $(\text{mcd}(a, b) = 1)$

## 2.7.2 – RSA

### Ejemplo de números coprimos

---

- ▶ Son coprimos si no tienen otro divisor común más que 1
  
- ▶ 45 y 6 son coprimos?

## 2.7.2 – RSA

### Ejemplo de números coprimos

---

- ▶ Son coprimos si no tienen otro divisor común más que 1
  
- ▶ 45 y 6 son coprimos?
  
- ▶ Divisores de 45: 1, 3 y 5
- ▶ Divisores de 6: 1, 2 y 3

## 2.7.2 – RSA

### Ejemplo de números coprimos

---

- ▶ Son coprimos si no tienen otro divisor común más que 1
  
- ▶ 45 y 6 son coprimos?
  
- ▶ Divisores de 45: 1, 3 y 5
- ▶ Divisores de 6: 1, 2 y 3
  
- ▶ Tienen 1 y 3 en común
  - ▶ El máximo común divisor de hecho es 3
- ▶ No son coprimos!

## 2.7.2 – RSA

### Ejemplo de números coprimos

---

- ▶ Son coprimos si no tienen otro divisor común más que 1
  
- ▶ 120 y 23 son coprimos?

## 2.7.2 – RSA

### Ejemplo de números coprimos

---

- ▶ Son coprimos si no tienen otro divisor común más que 1
- ▶ 120 y 23 son coprimos?
- ▶ Divisores de 120: 1, 2, 3 y 5
- ▶ Divisores de 23: 1 y 23
- ▶ Son coprimos ya que solo tienen el 1 en común

## 2.7.2 – RSA

### Ejemplo de números coprimos

---

- ▶ Son coprimos si no tienen otro divisor común más que 1
- ▶ 120 y 23 son coprimos?
- ▶ Resolución más rápida:
- ▶ 23 es un número primo, simplemente hay que verificar que  $120/23 = 5,21739\dots \rightarrow$  no es un número entero
- ▶ 23 no es un divisor de 120, por lo tanto son coprimos

## 2.7.2 – RSA

---

- ▶ Se eligen dos números primos muy grandes  $p$  y  $q$
- ▶ Se computa  $n = p \cdot q$ , donde  $n$  será la base del modulo (grupo  $\mathbb{Z}_n$ )
- ▶ Se computa la función de Euler  $\Phi(n) = (p-1) \cdot (q-1)$
- ▶ Se elige un entero  $e$  menor que  $\Phi(n)$  y que sea coprimo de  $\Phi(n)$ 
  - ▶  $e$  es el exponente de la clave publica
- ▶ Se determina  $d$  como el multiplicador modular inverso de  $e$  mod  $\Phi(n)$ 
  - ▶  $d$  es el exponente de la clave privada
- ▶ La clave publica es  $(n,e)$
- ▶ La clave privada es  $(n,d)$ 
  - ▶  $p, q$  y  $\Phi(n)$  también deben mantenerse privados

## 2.7.2 – RSA

### Multiplicador modular inverso

---

- ▶ Se determina  $d$  como el multiplicador modular inverso de  $e \text{ mod } \Phi(n)$

$$d = e^{-1} \text{ mod } \Phi(n)$$

- ▶ Dicho de otra manera  $(d \cdot e) \text{ mod } \Phi(n)$  es igual a 1
  - ▶ El multiplicador modular inverso de  $e \text{ mod } \Phi(n)$  existe solo si  $e$  y  $\Phi(n)$  son coprimos
- ▶ ¿Como se calcula este inverso?
  - ▶ Un poco de cálculos ahora

## 2.7.2 – RSA

Ejemplo de multiplicador modular inverso

---

- ▶  $e = 27, \Phi(n) = 392$
- ▶ Calcular  $d = 27^{-1} \bmod 392$

## 2.7.2 – RSA

Ejemplo de multiplicador modular inverso

---

- ▶  $e = 27, \Phi(n) = 392$
- ▶ Calcular  $d = 27^{-1} \bmod 392$

$392 = 27 \times 14 + 14 \rightarrow$  se calculan cociente y residuo

## 2.7.2 – RSA

Ejemplo de multiplicador modular inverso

---

- ▶  $e = 27, \Phi(n) = 392$
- ▶ Calcular  $d = 27^{-1} \bmod 392$

$392 = 27 \times 14 + 14 \rightarrow$  divisor y residuo se expanden en la siguiente

$$27 = 14 \times 1 + 13$$

## 2.7.2 – RSA

### Ejemplo de multiplicador modular inverso

---

- ▶  $e = 27, \Phi(n) = 392$
- ▶ Calcular  $d = 27^{-1} \bmod 392$

$$392 = 27 \times 14 + 14$$

$27 = 14 \times 1 + 13 \rightarrow$  divisor y residuo se expanden en la siguiente

$$14 = 13 \times 1 + 1$$

## 2.7.2 – RSA

### Ejemplo de multiplicador modular inverso

---

- ▶  $e = 27, \Phi(n) = 392$
- ▶ Calcular  $d = 27^{-1} \bmod 392$

$$392 = 27 \times 14 + 14$$

$$27 = 14 \times 1 + 13$$

$$14 = 13 \times 1 + 1 \rightarrow \text{hasta que quede 1 como residuo}$$

## 2.7.2 – RSA

### Ejemplo de multiplicador modular inverso

---

- ▶  $e = 27, \Phi(n) = 392$
- ▶ Calcular  $d = 27^{-1} \bmod 392$

$$392 = 27 \times 14 + 14$$

$$27 = 14 \times 1 + 13$$

$$14 = 13 \times 1 + 1$$

$1 = 14 - 1 \times 13 \rightarrow$  ahora se gira la última para que quede como  $1 =$

## 2.7.2 – RSA

### Ejemplo de multiplicador modular inverso

---

- ▶  $e = 27, \Phi(n) = 392$
- ▶ Calcular  $d = 27^{-1} \bmod 392$

$$392 = 27 \times 14 + 14$$

$$27 = 14 \times 1 + 13$$

$$14 = 13 \times 1 + 1$$

$$1 = 14 - 1 \times 13$$

$$13 = 27 - 1 \times 14 \rightarrow y la penúltima para que quede como 13 =$$

## 2.7.2 – RSA

### Ejemplo de multiplicador modular inverso

---

- ▶  $e = 27, \Phi(n) = 392$
- ▶ Calcular  $d = 27^{-1} \bmod 392$

$$392 = 27 \times 14 + 14$$

$$27 = 14 \times 1 + 13$$

$$14 = 13 \times 1 + 1$$

$$1 = 14 - 1 \times 13$$

13 = 27 - 1 × 14 → y se substituye el 13 de la primera por esta ecuación

$$\rightarrow 1 = 14 - 1 \times (27 - 1 \times 14)$$

$$\rightarrow 1 = 14 - 1 \times 27 + 1 \times 14$$

$$\rightarrow 1 = 2 \times 14 - 1 \times 27$$

## 2.7.2 – RSA

### Ejemplo de multiplicador modular inverso

---

- ▶  $e = 27, \Phi(n) = 392$
- ▶ Calcular  $d = 27^{-1} \bmod 392$

$$392 = 27 \times 14 + 14$$

$$27 = 14 \times 1 + 13$$

$$14 = 13 \times 1 + 1$$

$$1 = 14 - 1 \times 13$$

$$13 = 27 - 1 \times 14$$

$$\rightarrow 1 = 14 - 1 \times (27 - 1 \times 14)$$

$$\rightarrow 1 = 14 - 1 \times 27 + 1 \times 14$$

$$\rightarrow 1 = 2 \times 14 - 1 \times 27 \rightarrow \text{deben quedar divisor y residuo de la primera}$$

## 2.7.2 – RSA

### Ejemplo de multiplicador modular inverso

---

- ▶  $e = 27, \Phi(n) = 392$
- ▶ Calcular  $d = 27^{-1} \bmod 392$

$$392 = 27 \times 14 + 14$$

$$27 = 14 \times 1 + 13$$

$$14 = 13 \times 1 + 1$$

$$1 = 14 - 1 \times 13$$

$$13 = 27 - 1 \times 14$$

$$\rightarrow 1 = 14 - 1 \times (27 - 1 \times 14)$$

$$\rightarrow 1 = 14 - 1 \times 27 + 1 \times 14$$

$$\rightarrow 1 = 2 \times 14 - 1 \times 27 \rightarrow \text{nos quedamos con esta}$$

## 2.7.2 – RSA

### Ejemplo de multiplicador modular inverso

---

- ▶  $e = 27, \Phi(n) = 392$
- ▶ Calcular  $d = 27^{-1} \bmod 392$

$$392 = 27 \times 14 + 14$$

$$27 = 14 \times 1 + 13$$

$$14 = 13 \times 1 + 1$$

$$1 = 2 \times 14 - 1 \times 27$$

## 2.7.2 – RSA

### Ejemplo de multiplicador modular inverso

---

- ▶  $e = 27, \Phi(n) = 392$
- ▶ Calcular  $d = 27^{-1} \bmod 392$

$$392 = 27 \times 14 + 14$$

$$27 = 14 \times 1 + 13$$

$$14 = 13 \times 1 + 1$$

$$1 = 2 \times 14 - 1 \times 27$$

$$14 = 392 - 14 \times 27 \rightarrow \text{se gira la primera para que quede como } 14 =$$

## 2.7.2 – RSA

### Ejemplo de multiplicador modular inverso

---

- ▶  $e = 27, \Phi(n) = 392$
- ▶ Calcular  $d = 27^{-1} \bmod 392$

$$392 = 27 \times 14 + 14$$

$$27 = 14 \times 1 + 13$$

$$14 = 13 \times 1 + 1$$

$$1 = 2 \times 14 - 1 \times 27$$

14 = 392 – 14 x 27 → y se substituye el 14 de la primera por esta ecuación

$$\rightarrow 1 = 2 \times (392 - 14 \times 27) - 1 \times 27$$

$$\rightarrow 1 = 2 \times 392 - 28 \times 27 - 1 \times 27$$

$$\rightarrow 1 = 2 \times 392 - 29 \times 27$$

## 2.7.2 – RSA

### Ejemplo de multiplicador modular inverso

---

- ▶  $e = 27, \Phi(n) = 392$
- ▶ Calcular  $d = 27^{-1} \bmod 392$

$$392 = 27 \times 14 + 14$$

$$27 = 14 \times 1 + 13$$

$$14 = 13 \times 1 + 1$$

$$1 = 2 \times 14 - 1 \times 27$$

$14 = 392 - 14 \times 27 \rightarrow$  y se substituye el 14 de la primera por esta ecuación

$$\rightarrow 1 = 2 \times (392 - 14 \times 27) - 1 \times 27$$

$$\rightarrow 1 = 2 \times 392 - 28 \times 27 - 1 \times 27$$

$$\rightarrow 1 = 2 \times 392 - 29 \times 27 \rightarrow$$
 deben quedar los valores de  $e$  y  $\Phi(n)$

## 2.7.2 – RSA

### Ejemplo de multiplicador modular inverso

---

- ▶  $e = 27, \Phi(n) = 392$
- ▶ Calcular  $d = 27^{-1} \bmod 392$

$$392 = 27 \times 14 + 14$$

$$27 = 14 \times 1 + 13$$

$$14 = 13 \times 1 + 1$$

$$1 = 2 \times 14 - 1 \times 27$$

$14 = 392 - 14 \times 27 \rightarrow$  y se substituye el 14 de la primera por esta ecuación

$$\rightarrow 1 = 2 \times (392 - 14 \times 27) - 1 \times 27$$

$$\rightarrow 1 = 2 \times 392 - 28 \times 27 - 1 \times 27$$

$$\rightarrow 1 = 2 \times 392 - 29 \times 27 \rightarrow$$
 nos quedamos con esta

## 2.7.2 – RSA

### Ejemplo de multiplicador modular inverso

---

- ▶  $e = 27, \Phi(n) = 392$
- ▶ Calcular  $d = 27^{-1} \bmod 392$
- ▶ De momento hemos llegado a eso

$$1 = 2 \times 392 - 29 \times 27$$

## 2.7.2 – RSA

### Ejemplo de multiplicador modular inverso

---

- ▶  $e = 27, \Phi(n) = 392$
- ▶ Calcular  $d = 27^{-1} \bmod 392$
- ▶ De momento hemos llegado a eso
$$1 = 2 \times 392 - 29 \times 27$$
- ▶ Ahora se aplica el mod 392 a cada elemento
$$1 \bmod 392 = 1$$

## 2.7.2 – RSA

### Ejemplo de multiplicador modular inverso

---

- ▶  $e = 27, \Phi(n) = 392$
- ▶ Calcular  $d = 27^{-1} \bmod 392$
- ▶ De momento hemos llegado a eso

$$1 = 2 \times 392 - 29 \times 27$$

- ▶ Ahora se aplica el mod 392 a cada elemento

$$1 \bmod 392 = 1$$

$$2 \times 392 \bmod 392 = 0$$

## 2.7.2 – RSA

### Ejemplo de multiplicador modular inverso

---

- ▶  $e = 27, \Phi(n) = 392$
- ▶ Calcular  $d = 27^{-1} \bmod 392$
- ▶ De momento hemos llegado a eso

$$1 = 2 \times 392 - 29 \times 27$$

- ▶ Ahora se aplica el mod 392 a cada elemento

$$1 \bmod 392 = 1$$

$$2 \times 392 \bmod 392 = 0$$

$$-29 \times 27 \bmod 392 \rightarrow 392 - 29 = 363 \rightarrow 363 \times 27 \bmod 392$$

Hay que pasar el número negativo a positivo

Se le puede sumar el valor del modulo y pasarlo a positivo: el resultado no cambia ya que es una operación de modulo,

## 2.7.2 – RSA

### Ejemplo de multiplicador modular inverso

---

- ▶  $e = 27, \Phi(n) = 392$
- ▶ Calcular  $d = 27^{-1} \bmod 392$
- ▶ De momento hemos llegado a eso

$$1 = 2 \times 392 - 29 \times 27$$

- ▶ Ahora se aplica el mod 392 a cada elemento

$$1 \bmod 392 = 1$$

$$2 \times 392 \bmod 392 = 0$$

$$-29 \times 27 \bmod 392 \rightarrow 392 - 29 = 363 \rightarrow 363 \times 27 \bmod 392$$

- ▶ Entonces queda como

$$363 \times 27 \bmod 392 = 1 \bmod 392$$

## 2.7.2 – RSA

### Ejemplo de multiplicador modular inverso

---

- ▶  $e = 27, \Phi(n) = 392$
- ▶ Calcular  $d = 27^{-1} \bmod 392$

- ▶ De momento hemos llegado a eso

$$1 = 2 \times 392 - 29 \times 27$$

- ▶ Ahora se aplica el mod 392 a cada elemento

$$1 \bmod 392 = 1$$

$$2 \times 392 \bmod 392 = 0$$

$$-29 \times 27 \bmod 392 \rightarrow 392 - 29 = 363 \rightarrow 363 \times 27 \bmod 392$$

- ▶ Entonces queda como

$$363 \times 27 \bmod 392 = 1 \bmod 392$$

$$\rightarrow 363 \bmod 392 = 27^{-1} \bmod 392$$

$$\rightarrow 363 = 27^{-1} \bmod 392$$

## 2.7.2 – RSA

### Ejemplo de multiplicador modular inverso

---

- ▶  $e = 27, \Phi(n) = 392$
- ▶ Calcular  $d = 27^{-1} \bmod 392$

- ▶ De momento hemos llegado a eso

$$1 = 2 \times 392 - 29 \times 27$$

- ▶ Ahora se aplica el mod 392 a cada elemento

$$1 \bmod 392 = 1$$

$$2 \times 392 \bmod 392 = 0$$

$$-29 \times 27 \bmod 392 \rightarrow 392 - 29 = 363 \rightarrow 363 \times 27 \bmod 392$$

- ▶ Entonces queda como

$$363 = 27^{-1} \bmod 392$$

$$\rightarrow \mathbf{d = 363 = 27^{-1} \bmod 392}$$

## 2.7.2 – RSA

Ejercicio de multiplicador modular inverso

---

- ▶  $e = 23, \Phi(n) = 120$
- ▶ Calcular  $d = 23^{-1} \bmod 120$

## 2.7.2 – RSA

### Ejercicio de multiplicador modular inverso

---

- ▶  $e = 23, \Phi(n) = 120$
- ▶ Calcular  $d = 23^{-1} \bmod 120$
- ▶ Resultado  $d = 47$

## 2.7.2 – RSA

### Ejercicio de multiplicador modular inverso

---

- ▶  $e = 23, \Phi(n) = 120$
- ▶ Calcular  $d = 23^{-1} \bmod 120$
- ▶ Resultado  $d = 47$
- ▶ Como en el caso de la exponenciación binaria, existen herramientas que pueden calcular este resultado
- ▶ Un ejemplo es  
<https://www.dcode.fr/modular-inverse>
- ▶ Podéis usar estas herramientas para comprobar vuestros cálculos mientras os preparáis para los controles

## 2.7.2 – RSA

---

- ▶ Se eligen dos números primos muy grandes  $p$  y  $q$
- ▶ Se computa  $n = p \cdot q$ , donde  $n$  será la base del modulo (grupo  $\mathbb{Z}_n$ )
- ▶ Se computa la función de Euler  $\Phi(n) = (p-1) \cdot (q-1)$
- ▶ Se elige un entero  $e$  menor que  $\Phi(n)$  y que sea coprimo de  $\Phi(n)$ 
  - ▶  $e$  es el exponente de la clave publica
- ▶ Se determina  $d$  como el multiplicador modular inverso de  $e$  mod  $\Phi(n)$ 
  - ▶  $d$  es el exponente de la clave privada
- ▶ La clave publica es  $(n,e)$
- ▶ La clave privada es  $(n,d)$ 
  - ▶  $p, q$  y  $\Phi(n)$  también deben mantenerse privados

## 2.7.2 – RSA

### Ejemplo

---

- ▶ Se eligen dos números primos muy grandes  $p = 61$  y  $q = 53$
- ▶ Se computa  $n = p \cdot q = 61 \cdot 53 = 3223$
- ▶ Se computa  $\Phi(n) = (p-1) \cdot (q-1) = 60 \cdot 52 = 3120$
- ▶ Se elige un entero  $e$  menor que  $\Phi(n)$  y que sea coprimo de  $\Phi(n)$ 
  - ▶  $e = 17$
- ▶ Se determina  $d$  como el multiplicador modular inverso de  $e$  mod  $\Phi(n)$ 
  - ▶  $d = e^{-1} \text{ mod } \Phi(n) = 17^{-1} \text{ mod } 3120 = 2753$
- ▶ La clave publica es  $(n,e) = (3233, 17)$
- ▶ La clave privada es  $(n,d) = (3233, 2753)$

## 2.7.2 – RSA

### Ejemplo cifrado/descifrado

---

- ▶ A tiene clave publica (3233, 17) y clave privada (3233, 2753)

- ▶ Si B quiere enviar un mensaje  $m = 65$  a A,
- ▶ B busca la clave publica de A (3233, 17) y crea el mensaje encriptado c

$$c = m^e \bmod n = 65^{17} \bmod 3233 = 2790$$

- ▶ A recibe el mensaje y desencripta usando su clave privada (3233, 2753)

$$m = c^d \bmod n = 2790^{2753} \bmod 3233 = 65$$

## 2.7.2 – RSA

### Possibles ataques

---

- ▶ Mensajes  $m=0$ ,  $m=1$  o  $m=n-1$  siempre producen cifrados iguales  $c=0$ ,  $c=1$  y  $c=n-1$
- ▶ Si se usan exponentes e pequeños (p.e., 3) y valores pequeños de m, el resultado de m cifrado puede ser menor que n, de manera que no hay componente cíclica, es decir, solo hay un resultado posible
- ▶ Si se sabe que los mensajes son de texto, se puede lanzar un ataque con un diccionario
  - ▶ Es decir se construye un diccionario de textos probables usando la clave pública
  - ▶ Se compara el texto cifrado con este diccionario hasta encontrar uno igual

## 2.7.2 – RSA

### Possibles ataques

---

- ▶ Mensajes  $m=0$ ,  $m=1$  o  $m=n-1$  siempre producen cifrados iguales  $c=0$ ,  $c=1$  y  $c=n-1$
- ▶ Si se usan exponentes  $e$  pequeños (p.e., 3) y valores pequeños de  $m$ , el resultado de  $m$  cifrado puede ser menor que  $n$ , de manera que no hay componente cíclica, es decir, solo hay un resultado posible
- ▶ Si se sabe que los mensajes son de texto, se puede lanzar un ataque con un diccionario
  - ▶ Es decir se construye un diccionario de textos probables usando la clave pública
  - ▶ Se compara el texto cifrado con este diccionario hasta encontrar uno igual
- ▶ Solución
  - ▶ Se usan esquemas de padding (relleno) como RSA-OAEP+
  - ▶ Estos esquemas modifican el texto añadiendo “relleno” para que no se cumplan las condiciones anteriores

## 2.7.2 – RSA

### Posibles ataques

---

- ▶ Una posible manera para descubrir  $d$  es encontrar aquellos números primos grandes  $p$  y  $q$  que multiplicados dan el valor  $n$ 
  - ▶ Ya que  $n$  es público, se podría intentar encontrar  $p$  y  $q$
  - ▶ Este problema se conoce como factorización de números grandes
  - ▶ Una vez encontrados  $p$  y  $q$ , se calcula  $\Phi(n) = (p-1) \cdot (q-1)$  y luego  $d = e^{-1} \bmod \Phi(n)$
- ▶ De momento, no se ha encontrado ningún método en tiempo polinómico para la factorización de enteros grandes
  - ▶ Existen pero métodos matemáticos que requieren computación intensiva que pueden encontrar  $p$  y  $q$
  - ▶ El más reciente es la factorización del RSA-240 (2019)

## 2.7.2 – RSA

### Posibles ataques

---

- ▶ A lo largo de los años se han ido creando varias versiones del RSA que han ido usando  $n$  siempre más grandes, por ejemplo
  - ▶ RSA-100 usa una  $n$  de 100 decimales:  
15226050279225333605356183781326374297180681149613806886579084945801229632589  
52897654000350692006139
  - ▶ RSA-110 usa una  $n$  de 110 decimales:  
35794234179725868774991807832568455403003778024228226193532908190484670252364  
67741151351611204504060317568667
  - ▶ ...
  - ▶ RSA-240 usa una  $n$  de 240 decimales:  
12462036678171878406583504460810659043482037465167880575481878888328966680118  
82108550360395702725087475098647684384586210548655379702539305718912176843182  
86362846948405301614416430468066875699415246993185704183030512549594371372159  
029236099
  - ▶ ...
  - ▶ RSA-2048 usa una  $n$  de 617 decimales (2048 bits)

## 2.7.2 – RSA

### Posibles ataques

---

- ▶ RSA-240 usa una n de 240 decimales
  - ▶ 124620366781718784065835044608106590434820374651678805754818788883289666801188210855036039570272508747509864768438458621054865537970253930571891217684318286362846948405301614416430468066875699415246993185704183030512549594371372159029236099
  - ▶ En el 2019 se ha podido calcular que este número se compone como la multiplicación de estos dos números primos  
**50943595228583991455505102358084371413264838202411147318666029652182120646974670 0620316443478873837606252372049619334517**  
x  
**24462420883831815056781313902400289665380209257893140145204122133655847709517815 525821889773503059066904130204590871447**
- ▶ Para calcular esta factorización se ha necesitado la computación equivalente a 900 años en un 2.1 GHz Intel Xeon Gold 6130 CPU
  - ▶ En realidad se ha usado la computación paralela sobre centenares de maquinas

## 2.7.2 – RSA

### Posibles ataques

---

- ▶ De momento no hay solución a partir del RSA-250 (n de 250 decimales)
- ▶ Pero en el 1994, Peter Shor (MIT) descubrió un algoritmo capaz de factorizar números grandes en un tiempo polinómico usando computación cuántica
- ▶ Factible si hay un suficiente número de qubits
  - ▶ En el 2001, IBM consiguió factorizar 15 en  $3 \times 5$  (7 qubits)
  - ▶ En el 2012, se factorizó 21 en  $7 \times 3$
  - ▶ En el 2014, 56.153 en  $233 \times 241$  (4 qubits)
  - ▶ En el 2018, 4.088.459 en  $2.017 \times 2.027$  (5 y 16 qubits)
  - ▶ En 2021, IBM prueba un ordenador cuántico con 127 qubits

## 2.7.2 – RSA

### Posibles ataques

---

- ▶ De momento no hay solución a partir del RSA-250 (n de 250 decimales)
- ▶ Pero en el 1994, Peter Shor (MIT) descubrió un algoritmo capaz de factorizar números grandes en un tiempo polinómico usando computación cuántica
- ▶ Factible si hay un suficiente número de qubits
  - ▶ En el 2001, IBM consiguió factorizar 15 en  $3 \times 5$  (7 qubits)
  - ▶ En el 2012, se factorizó 21 en  $7 \times 3$
  - ▶ En el 2014, 56.153 en  $233 \times 241$  (4 qubits)
  - ▶ En el 2018, 4.088.459 en  $2.017 \times 2.027$  (5 y 16 qubits)
  - ▶ En 2021, IBM prueba un ordenador cuántico con 127 qubits
- ▶ Estamos lejos de momento de tener una solución para los RSA que se usan realmente
  - ▶ Actualmente se usa RSA-2048 (se calcula fiable hasta 2030)
  - ▶ Pero también se están usando RSA-3072 y RSA-4096

## 2.7.2 – RSA

### Posibles ataques

- ▶ Pero se pueden hacer ataques más ingeniosos

#### RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis

Daniel Genkin

Technion and Tel Aviv University

Adi Shamir

Weizmann Institute of Science

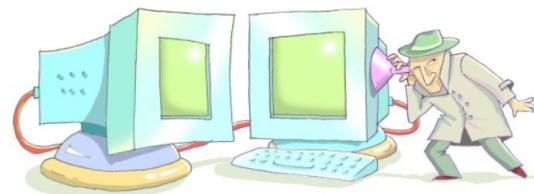
Eran Tromer

Tel Aviv University

assisted by Lev Pachmanov and numerous others

#### Summary

Many computers emit a high-pitched noise during operation, due to vibration in some of their electronic components. These acoustic emanations are more than a nuisance: they can convey information about the software running on the computer and, in particular, leak sensitive information about security-related computations. In a [preliminary presentation](#), we have shown that different RSA keys induce different sound patterns, but it was not clear how to extract individual key bits. The main problem was the very low bandwidth of the acoustic side channel (under 20 kHz using common microphones, and a few hundred kHz using ultrasound microphones), many orders of magnitude below the GHz-scale clock rates of the attacked computers.



Here, we describe a new *acoustic cryptanalysis* key extraction attack, applicable to GnuPG's current implementation of RSA. The attack can extract full 4096-bit RSA decryption keys from laptop computers (of various models), within an hour, using the sound generated by the computer during the decryption of some chosen ciphertexts. We experimentally demonstrate that such attacks can be carried out, using either a plain mobile phone placed next to the computer, or a more sensitive microphone placed 4 meters away.

Beyond acoustics, we demonstrate that a similar low-bandwidth attack can be performed by measuring the electric potential of a computer chassis. A suitably-equipped attacker need merely touch the target computer with his bare hand, or get the required leakage information from the ground wires at the remote end of VGA, USB or Ethernet cables.

## 2.6 – Algunos principios matemáticos

---

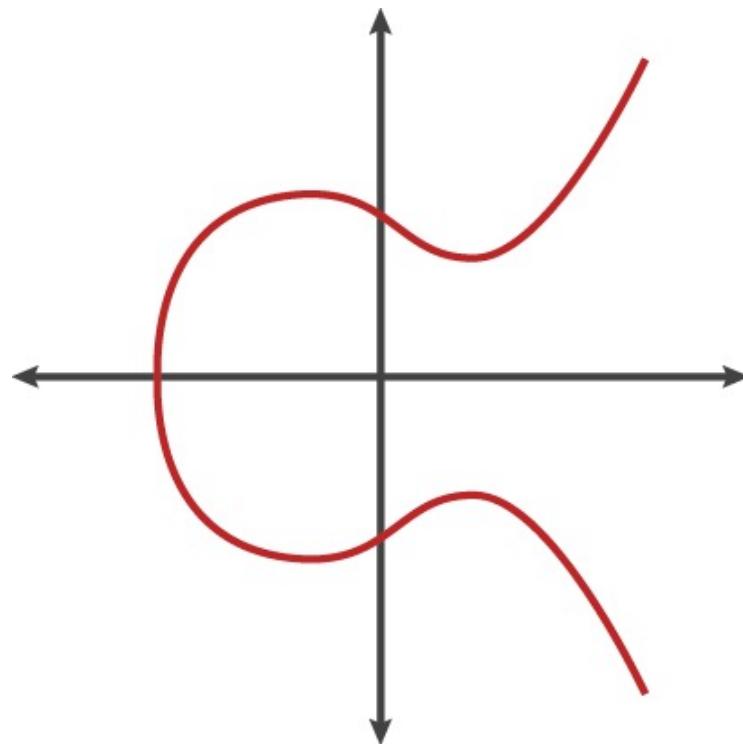
- ▶ Como se puede garantizar que no se descubra una clave secreta a partir de valores que se intercambien los dos extremos
- ▶ Dos soluciones principales
  - ▶ Logarítmica/exponenciación discreta
  - ▶ Curvas elípticas

## 2.6.2 – Curvas elípticas

---

- ▶ Se propuso por primera vez en el 1985
- ▶ La idea es usar una curva elíptica de la forma

$$y^2 = x^3 + ax + b$$



Fuente: <https://blog.cloudflare.com/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/>

## 2.6.2 – Curvas elípticas

---

- ▶ Se propuso por primera vez en el 1985
- ▶ La idea es usar una curva elíptica de la forma

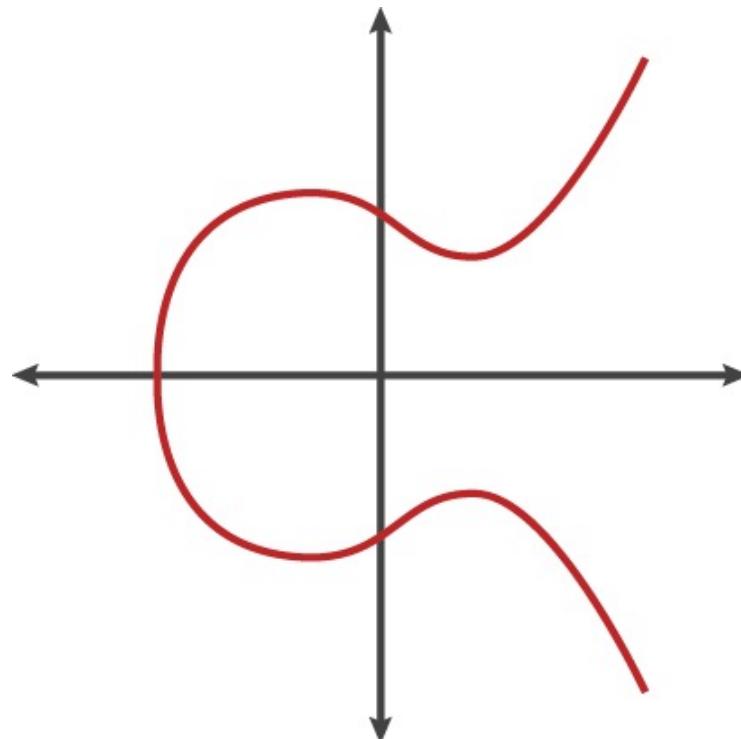
$$y^2 = x^3 + ax + b$$

- ▶ Crear un grupo finito  $x,y$  que satisfagan esta curva
- ▶ Y un grupo de operaciones que tengan las mismas características del logaritmo/exponente discreto
  - ▶ Muy fácil uno, muy difícil el otro
  - ▶ Trapdoor function
- ▶ Juntos crean lo que se llama un grupo abeliano

## 2.6.2 – Curvas elípticas

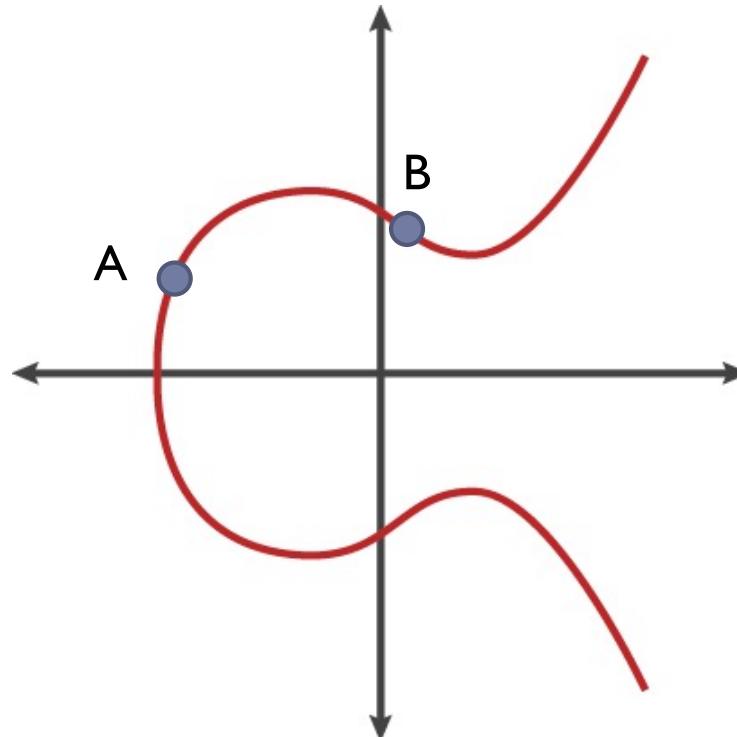
---

- ▶ Estas curvas tienen varias propiedades
  - ▶ Una de estas es que cualquier línea no vertical, intercepta la curva en tres puntos diferentes
  - ▶ Otra es que es simétrica



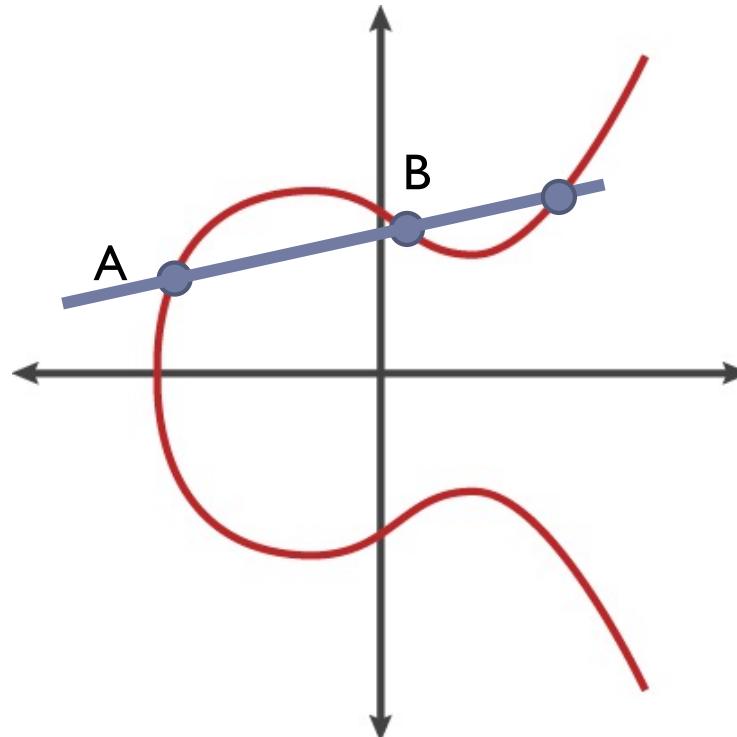
## 2.6.2 – Curvas elípticas

- ▶ Estas curvas tienen varias propiedades
  - ▶ Una de estas es que cualquier línea no vertical, intercepta la curva en tres puntos diferentes
  - ▶ Otra es que es simétrica
- ▶ Operación A dot B =



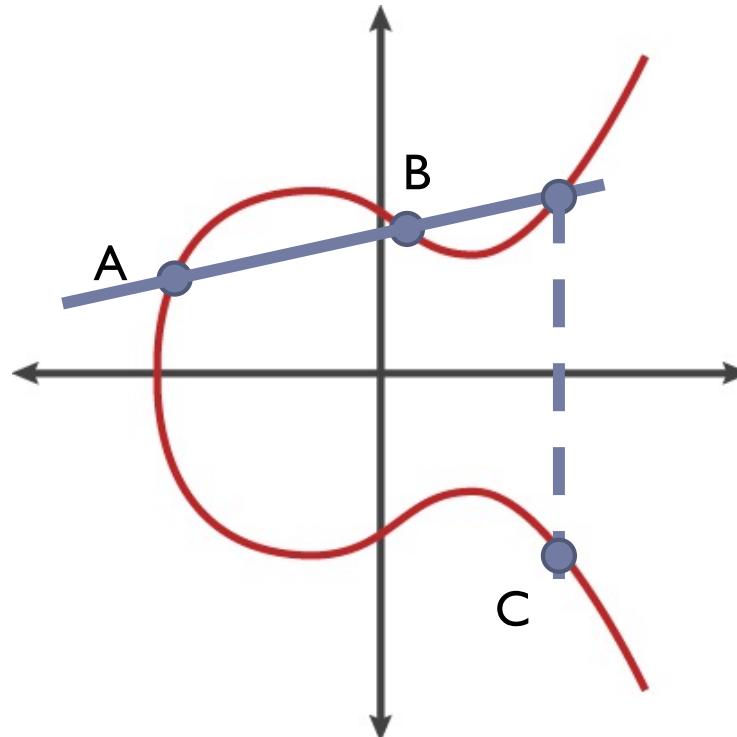
## 2.6.2 – Curvas elípticas

- ▶ Estas curvas tienen varias propiedades
  - ▶ Una de estas es que cualquier línea no vertical, intercepta la curva en tres puntos diferentes
  - ▶ Otra es que es simétrica
- ▶ Operación A dot B =



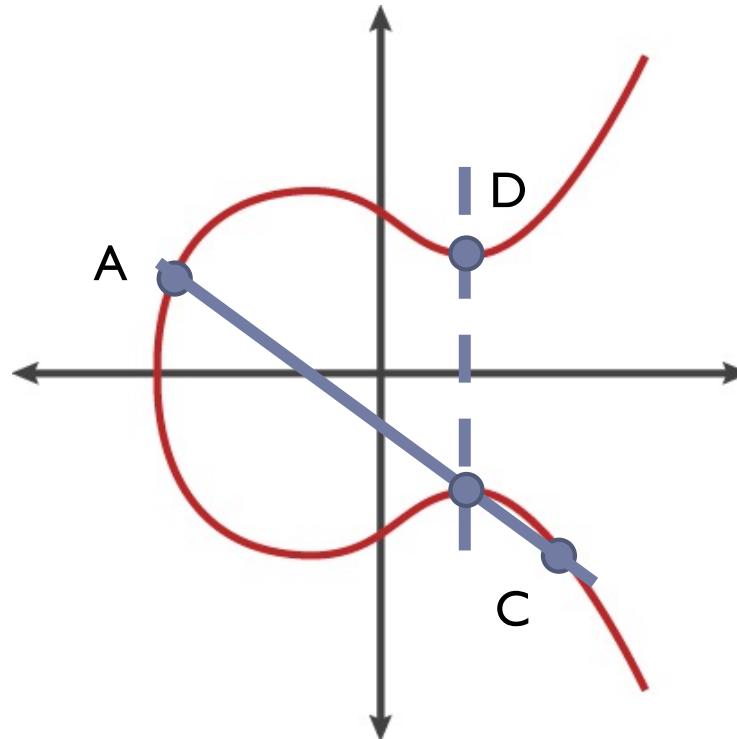
## 2.6.2 – Curvas elípticas

- ▶ Estas curvas tienen varias propiedades
  - ▶ Una de estas es que cualquier línea no vertical, intercepta la curva en tres puntos diferentes
  - ▶ Otra es que es simétrica
- ▶ Operación  $A \dot{+} B = C$



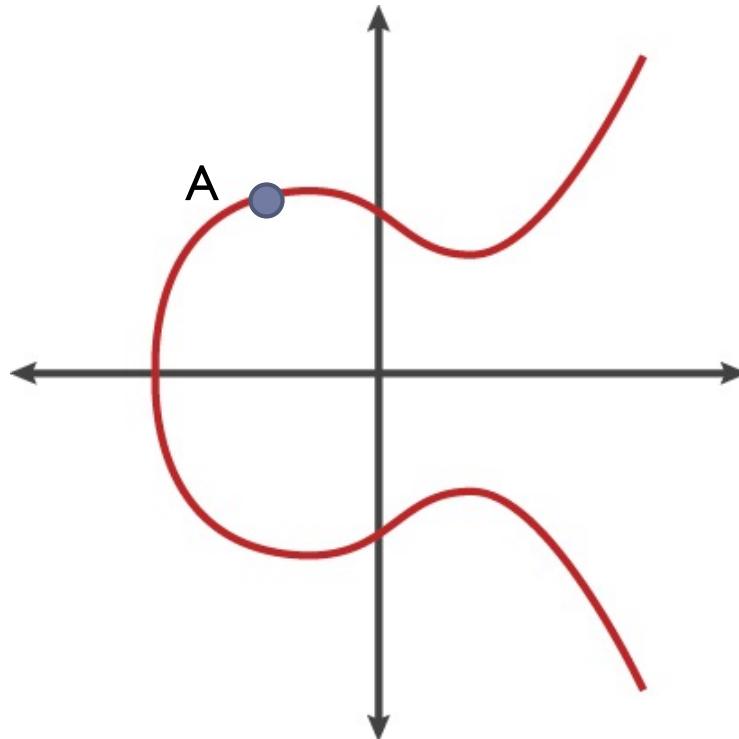
## 2.6.2 – Curvas elípticas

- ▶ Estas curvas tienen varias propiedades
  - ▶ Una de estas es que cualquier línea no vertical, intercepta la curva en tres puntos diferentes
  - ▶ Otra es que es simétrica
- ▶ Operación  $A \cdot C = D$



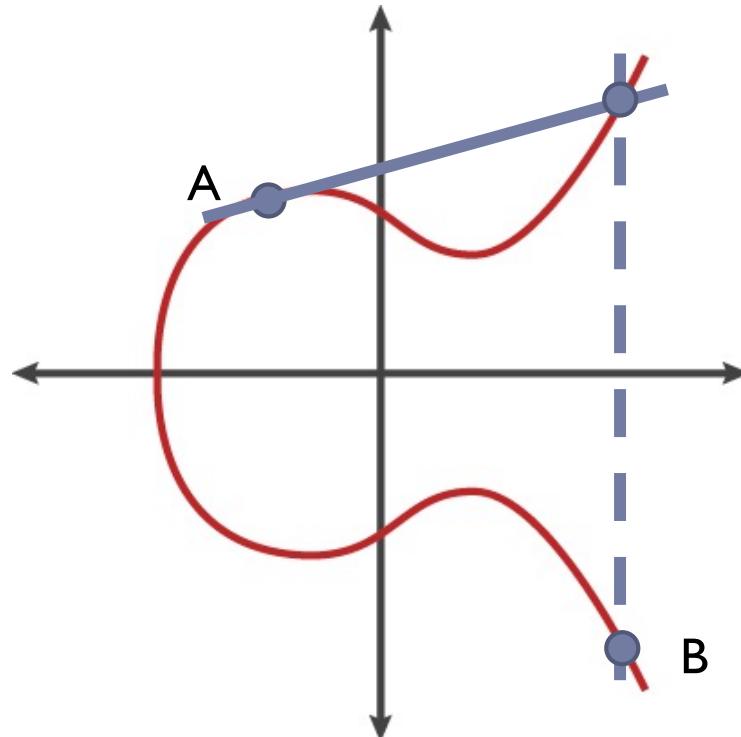
## 2.6.2 – Curvas elípticas

- ▶ Estas curvas tienen varias propiedades
  - ▶ Una de estas es que cualquier línea no vertical, intercepta la curva en tres puntos diferentes
  - ▶ Otra es que es simétrica
- ▶ Operación  $A \cdot A = ?$



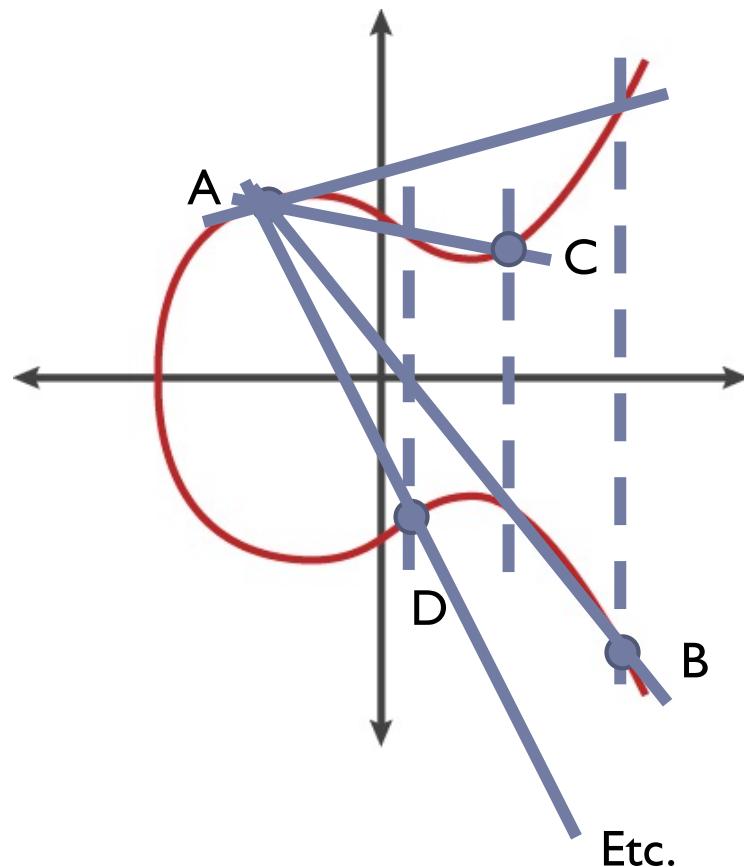
## 2.6.2 – Curvas elípticas

- ▶ Estas curvas tienen varias propiedades
  - ▶ Una de estas es que cualquier línea no vertical, intercepta la curva en tres puntos diferentes
  - ▶ Otra es que es simétrica
- ▶ Operación  $A \text{ dot } A = B$ 
  - ▶ La tangente de A



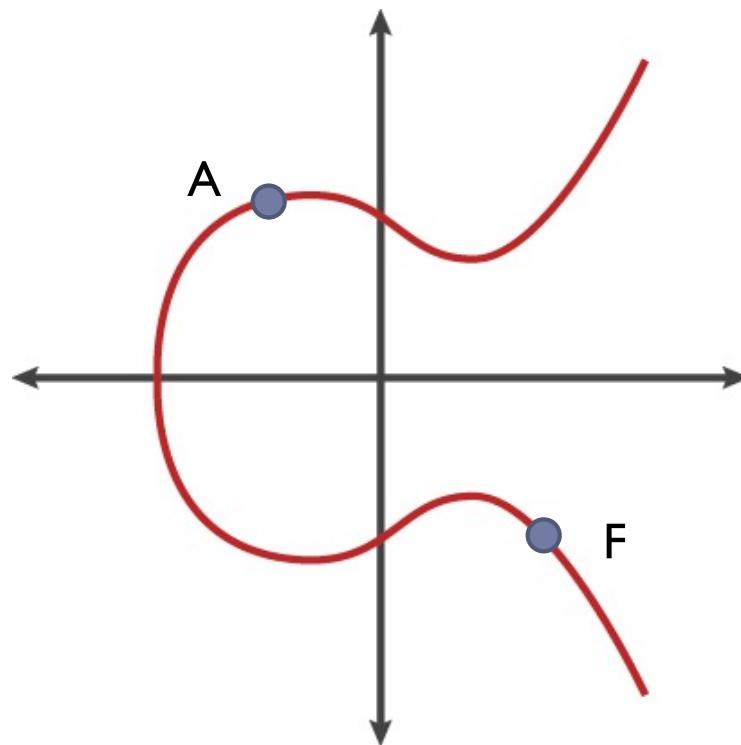
## 2.6.2 – Curvas elípticas

- ▶ ¿Por qué es una trapdoor funtion?
- ▶ Porque a partir de un punto público A y repitiendo esta operación dot, se llega a un punto final F



## 2.6.2 – Curvas elípticas

- ▶ ¿Por qué es una trapdoor funtion?
- ▶ Porque a partir de un punto público A y repitiendo esta operación dot, se llega a un punto final F



- ▶ **Y la pregunta es: ¿a partir de estos puntos sabrías indicar cuantos dot se han hecho?**

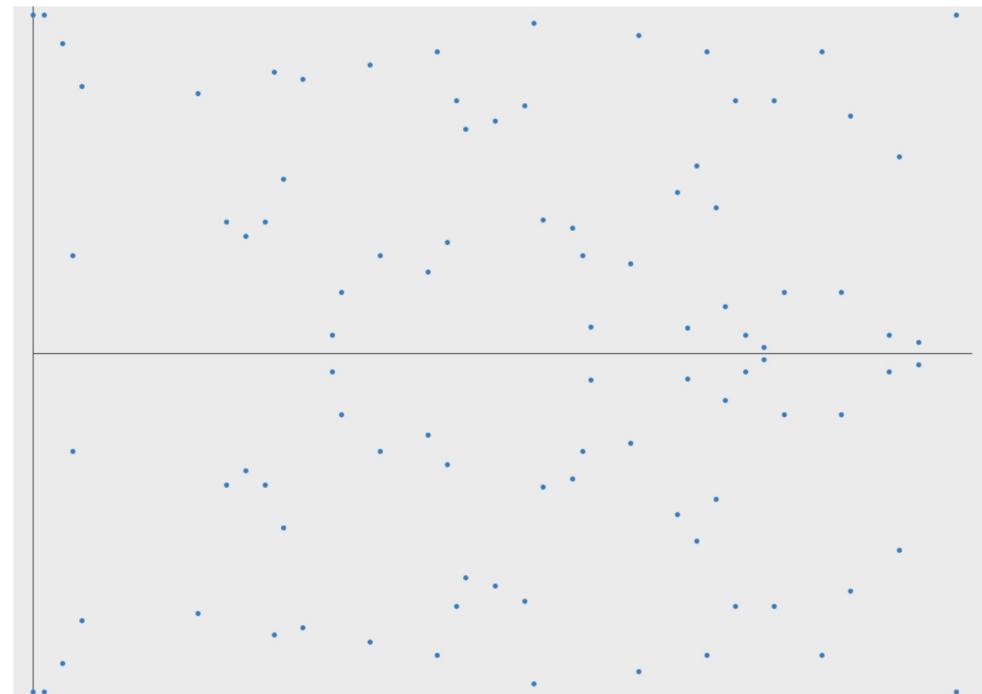
## 2.6.2 – Curvas elípticas

---

- ▶ La clave pública y privada de una curva elíptica se determinan con esta trapdoor function
- ▶ Se elige una curva elíptica, se elige un punto público  $A$  de este curva y un número primo (veremos a que sirve luego)
- ▶ Se elige luego un número privado (la clave privada)  $S_k$  y se calcula la clave pública  $P_k$  como el resultado de hacer  $A$  dot  $A$  un número igual a  $S_k$  veces

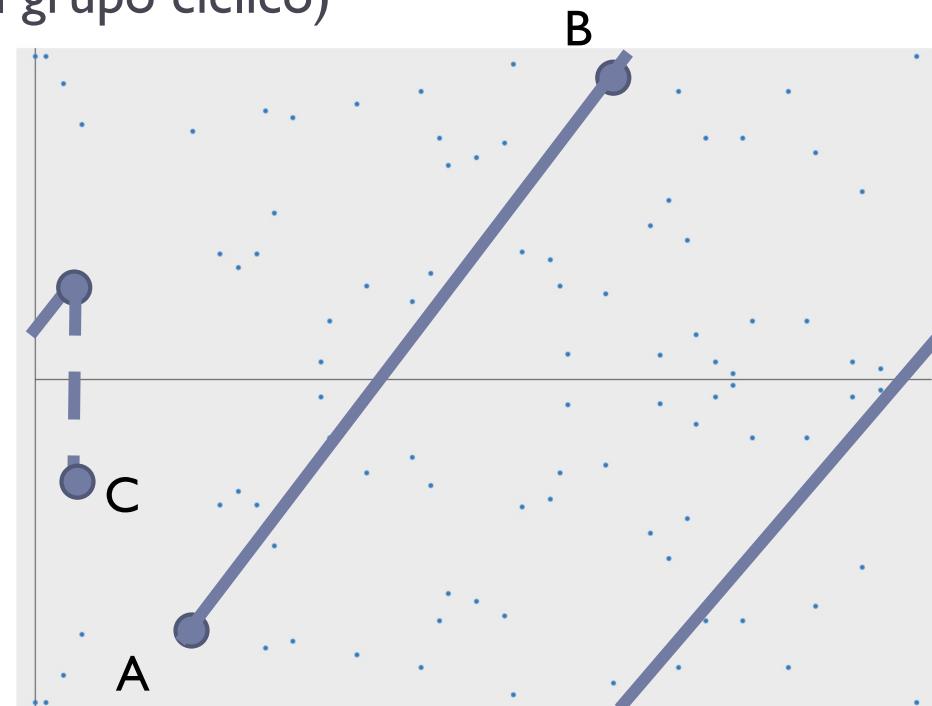
## 2.6.2 – Curvas elípticas

- ▶ Las curvas usadas en estos ejemplos tienen pero valores reales y pueden tener valores muy grandes
- ▶ En realidad, se usan números enteros y hasta un valor máximo
  - ▶ Este valor máximo debe ser un número primo
  - ▶ La curva se “dobla” en los bordes para que no se supere este valor máximo (como el módulo del grupo cíclico)
- ▶ El resultado es una “curva” elíptica donde solo aparecen los valores enteros y envueltos en un espacio limitado



## 2.6.2 – Curvas elípticas

- ▶ Las curvas usadas en estos ejemplos tienen pero valores reales y pueden tener valores muy grandes
- ▶ En realidad, se usan números enteros y hasta un valor máximo
  - ▶ Este valor máximo debe ser un número primo
  - ▶ La curva se “dobla” en los bordes para que no se supere este valor máximo (como el módulo del grupo cíclico)
- ▶ El resultado es una “curva” elíptica donde solo aparecen los valores enteros y envueltos en un espacio limitado
- ▶ Sigue valiendo las mismas propiedades vista antes



## 2.6.2 – Curvas elípticas

---

- ▶ Se propuso por primera vez en el 1985
- ▶ La idea es usar una curva elíptica de la forma
$$y^2 = x^3 + ax + b$$
- ▶ Crear un grupo finito  $x,y$  que satisfagan esta curva
- ▶ Y un grupo de operaciones posibles, creando lo que se llama un grupo abeliano
- ▶ La idea entonces es sustituir el grupo cíclico finito  $G$  por este grupo y usar los mismos algoritmos
  - ▶ Se estima que se pueden usar claves más pequeñas, por ejemplo 256 bits, y obtener el mismo nivel criptográfico que usando RSA con 3072 bit

## 2.6.2 – Curvas elípticas

---

- ▶ De manera que hoy en día existen también
  - ▶ Elliptic Curve Diffie-Hellmann
  - ▶ Elliptic Curve Digital Signature Algorithm
  - ▶ Elliptic Curve Integrated Encryption Scheme
- ▶ Se estima pero que la computación cuántica sería aún más eficiente para romper una criptografía de curva elíptica
  - ▶ Para una de 256 bits se estiman necesarios 2330 qubits y 126 billion gates
  - ▶ Para un RSA de 2048 bits, 4098 qubits y 5.2 trillion gates

## 2.7 – Algoritmos más conocidos

---

- ▶ Diffie-Hellman
  - ▶ Exponenciación binaria
- ▶ RSA
  - ▶ Números coprimos
  - ▶ Multiplicador modular inverso
- ▶ ElGamal
- ▶ RSA para Firma Digital
- ▶ Función Hash

## 2.7.3 - ElGamal

---

- ▶ Se basa en el generador de claves privadas Diffie-Hellman
- ▶ Es un algoritmo de criptografía asimétrica (clave publica/privada)
- ▶ Es de libre uso (no hay patente)
- ▶ Se puede usar en GNU Privacy Guard, PGP y otros sistemas criptográficos

-----BEGIN PGP SIGNATURE-----

Version: GnuPG v2

```
iQlcBAEBCAAGBQJaHrrMAAoJEA5SAxxF+nojApYQAMxICG2ivxBKS9KGpq51bjFo  
kiypqYo2WAMu25v9q/b2kHijzA4VgOSaZ+UrPMgzuHWdXoJPHPvypMFnbYHz95eg  
fNVxBKi/UAbSZBmuktq/wtcyzbfxOi0nu+pGggEZEM5W9KGZ5tQ/q18WwbJgZXnX  
q96YvvSbZltpHIL5Z7LRxOILNtN9oA3NpxeCBRof0vds0uRP1alEwxJZBjj9OpVg  
W9flMe1STrBmPGcTEB8aCo2Buwd9pdwd0oSKkX3Gb9jRgl3CMmGmqHtfWA8NZYXH  
L3Vv10Tar4Yv6oghf1nMHSmrK/+dhkMYqhXTj/JXT9wctsqNCYrefqmoYPLrL4dx  
K6bV66ITNr65Age+BWwc0clmRTs9iPX0g2giLeji7HjB+z/aHvFUyvmLIMGMSIsH  
Uzf7e9lpMoJhDDgz93I/7UfEbXAFb3OGZMvOX90smNgkTXI9dunyMo0XZ9KnsTJK  
CVZoFMQDWr+WxFY1QvW/JZFwYeqDqlZ/nluscPlg/Lf25DtGzHUoBR3Al8qRmDp5  
I4fXYnLfyZFJ2BjsVmzLIFakGoVTWg/akVu3qtzMrKXIGbOdaTuchhFTzOfHitqD  
+zKdKKII7mHpazP5Nn2OIBfPD4UNbGh13yUD7GDxqBjjSSoIKj01RdhdAX058kM8  
8PqrxFVeYn4P5SzTsRurF  
=3i08  
-----END PGP SIGNATURE-----
```

## 2.7.3 – ElGamal

### Generación claves

---

- ▶ Se elige grupo cíclico finito  $G$  de orden  $n$
- ▶ Un elemento  $\alpha$  de este grupo  $\alpha \in G$
- ▶ Un usuario A
  - ▶ Elige un número aleatorio  $a$  (será parte de su clave privada)
  - ▶ Calcula  $\alpha^a \in G$  (esta será la clave pública)
  - ▶  $\alpha, G, \alpha^a \in G$  son todos valores públicos ( $a$  debe mantenerse secreta)
- ▶ Clave pública  $(\alpha, G, \alpha^a \in G)$
- ▶ Clave privada  $(\alpha, G, a)$

## 2.7.3 – ElGamal

### Encriptación

---

- ▶ Si B quiere enviar un mensaje  $m \in \mathbf{G}$  a A, entonces debe
  - ▶ Recibir la clave publica de A ( $\alpha, \mathbf{G}, \alpha^a$ )
  - ▶ Elegir un número aleatorio  $b$  y calcular  $\alpha^b \in \mathbf{G}$  (nota)
  - ▶ Calcular el mensaje cifrado  $c = m \cdot (\alpha^a)^b \in \mathbf{G}$
  - ▶ Enviar a A el mensaje  $(\alpha^b, c)$
- ▶ Nota: se recomienda que  $b$  sea de un único uso

## 2.7.3 – ElGamal

### Desencriptación

---

- ▶ A recibe el mensaje cifrado  $(\alpha^b, c)$ 
  - ▶ Calcula  $x = (\alpha^b)^a \in \mathbb{G}$
  - ▶ Calcula el mensaje en claro  $m = c \cdot x^{-1} \in \mathbb{G}$

## 2.7.3 – ElGamal

### Desencriptación

---

- ▶ A recibe el mensaje cifrado  $(\alpha^b, c)$ 
  - ▶ Calcula  $x = (\alpha^b)^a \in \mathbb{G}$
  - ▶ Calcula el mensaje en claro  $m = c \cdot x^{-1} \in \mathbb{G}$
- ▶ Os acordáis que es eso ( $x^{-1} \in \mathbb{G}$ )?

## 2.7.3 – ElGamal

### Desencriptación

---

- ▶ A recibe el mensaje cifrado  $(\alpha^b, c)$ 
  - ▶ Calcula  $x = (\alpha^b)^a \in G$
  - ▶ Calcula el mensaje en claro  $m = c \cdot x^{-1} \in G$
  
- ▶ Os acordáis que es eso ( $x^{-1} \in G$ )?
- ▶ Multiplicador modular inverso

## 2.7.3 – ElGamal

### Ejemplo

---

- ▶  $G = 13$
- ▶  $\alpha = 2$
- ▶ Un usuario A
  - ▶ Elige un número aleatorio  $a = 9$
  - ▶ Calcula  $\alpha^a \in G \rightarrow 2^9 \text{ mod } 13 = 5$
  - ▶ Clave publica =  $(2, 13, 5)$
  - ▶ Clave privada = 9
    - ▶ También se podría indicar como  $(2, 13, 9)$

## 2.7.3 – ElGamal

### Ejemplo

---

- ▶ Un usuario B quiere enviar el mensaje  $m=11$  a A
  - ▶ Recibir la clave publica de A  $(2, 13, 5)$
  - ▶ Elegir un número aleatorio  $b = 10$  y calcular  $\alpha^b \in G \rightarrow 2^{10} \text{ mod } 13 = 10$
  - ▶ Calcular el mensaje cifrado  $c = m \cdot (\alpha^a)^b \in G$   
 $\rightarrow c = 11 \cdot 5^{10} \text{ mod } 13 = 11 \cdot 12 \text{ mod } 13 = 2$
  - ▶ Enviar a A el mensaje  $(\alpha^b, c) = (10, 2)$

## 2.7.3 – ElGamal Ejemplo

---

- ▶ A recibe el mensaje cifrado  $(\alpha^b, c) = (10, 2)$ 
  - ▶ Calcula  $x = (\alpha^b)^a \in \mathbb{G} = 10^9 \text{ mod } 13 = 12$
  - ▶ Calcula el mensaje en claro  $m = c \cdot x^{-1} \in \mathbb{G}$   
 $\rightarrow m = 2 \cdot 12^{-1} \text{ mod } 13$

## 2.7.3 – ElGamal

### Ejemplo

---

- ▶ A recibe el mensaje cifrado  $(\alpha^b, c) = (10, 2)$

- ▶ Calcula  $x = (\alpha^b)^a \in \mathbb{G} = 10^9 \text{ mod } 13 = 12$

- ▶ Calcula el mensaje en claro  $m = c \cdot x^{-1} \in \mathbb{G}$

- $\rightarrow m = 2 \cdot 12^{-1} \text{ mod } 13$

- $\rightarrow 12^{-1} \text{ mod } 13 \rightarrow \text{MM inverso}$

$$13 = 12 \times 1 + 1$$

$$1 = 13 - 1 \times 12$$

$$1 \text{ mod } 13 = 1$$

$$13 \text{ mod } 13 = 0$$

$$-1 \times 12 \text{ mod } 13 \rightarrow 13 - 1 = 12 \rightarrow 12 \times 12 \text{ mod } 13$$

$$1 = 12 \times 12 \text{ mod } 13$$

$$12 = 12^{-1} \text{ mod } 13$$

$$\rightarrow m = 2 \cdot 12 \text{ mod } 13 = 11$$

## 2.7.3 – ElGamal

### Posibles ataques

---

- ▶ Igual que RSA, ElGamal sufre de los mismos problemas con determinados mensajes
- ▶ Necesita un esquema de padding

## 2.7 – Algoritmos más conocidos

---

- ▶ Diffie-Hellman
  - ▶ Exponenciación binaria
- ▶ RSA
  - ▶ Números coprimos
  - ▶ Multiplicador modular inverso
- ▶ ElGamal
- ▶ RSA para Firma Digital
- ▶ Función Hash

## 2.7.4 - Firma Digital

- ▶ Si  $U$  quiere firmar un mensaje  $M$ , simplemente aplica el algoritmo  $E$  con su clave privada de forma que el mensaje firmado es  $S = E_{SK_U}(M)$
- ▶ Para verificar que el que ha firmado es realmente  $U$ , cualquier usuario puede aplicar el algoritmo de desencriptación usando la clave pública de  $U$  sobre el mensaje cifrado y comparar el resultado con el mensaje no cifrado, es decir verificar que  $D_{PK_U}(S) = M$

### VISTO BUENO DEL INFORME ANUAL DE EVALUACIÓN

#### I. INFORME DEL DIRECTOR/A

Valoración de la consecución de los objetivos por parte del beneficiario/a durante la anualidad a la que se refiere este informe:

- Favorable: se aconseja la continuidad de la ayuda  
 NO favorable: NO se aconseja la continuidad de la ayuda

Motivación del informe NO favorable:

Grado aproximado de consecución de los objetivos marcados para la anualidad objeto de este informe:

	Excelente	Notable	Aceptable	Insuficiente
Metodología	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Tareas y resultados	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Programa formativo	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Motivación de la calificación:

La valoración del desarrollo de la tesis doctoral hasta este punto es muy positiva. Cabe destacar las numerosas colaboraciones lideradas por el beneficiario con instituciones tanto nacionales (p.e., Telefónica I+D, Fundación i2CAT, ATOS Origin) como internacionales (p.e., Predictive Network Solutions, Brno University of Technology), las cuales han permitido la preparación de un número substancial de artículos de investigación para su publicación en revistas de prestigio y para su presentación en conferencias. Un ejemplo de estos es el artículo aceptado en la revista indexada en los JCR European Transactions on Telecommunications, o las ponencias en IEEE GLOBECOM o en el Workshop NetCloud 2016.

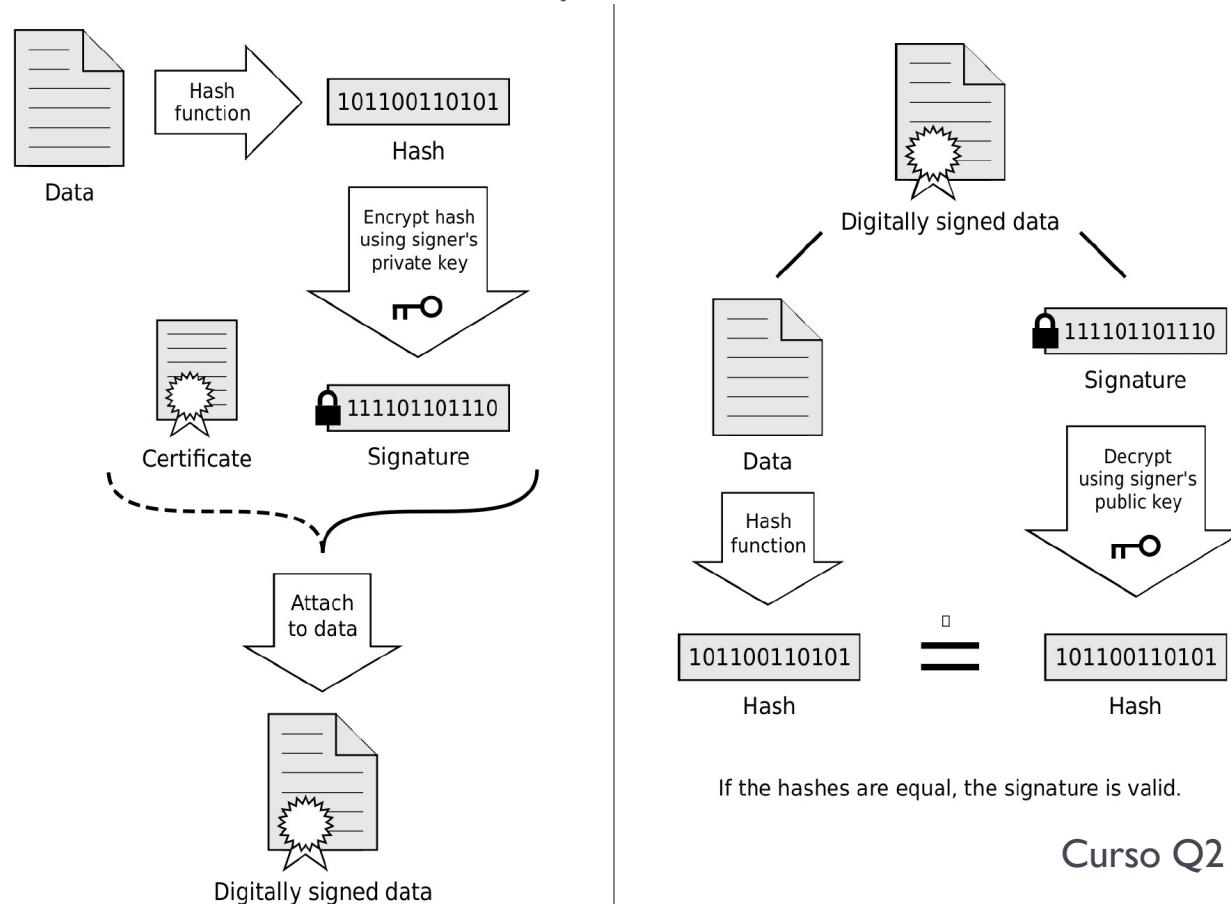
Firma electrónica del director/a:

Una vez firmado, debe enviar este documento a la COMISIÓN ACADÉMICA para que cumplimente y firme electrónicamente la página 8.  
Es importante que al firmar NO bloquee el documento.



## 2.7.4 - Firma Digital

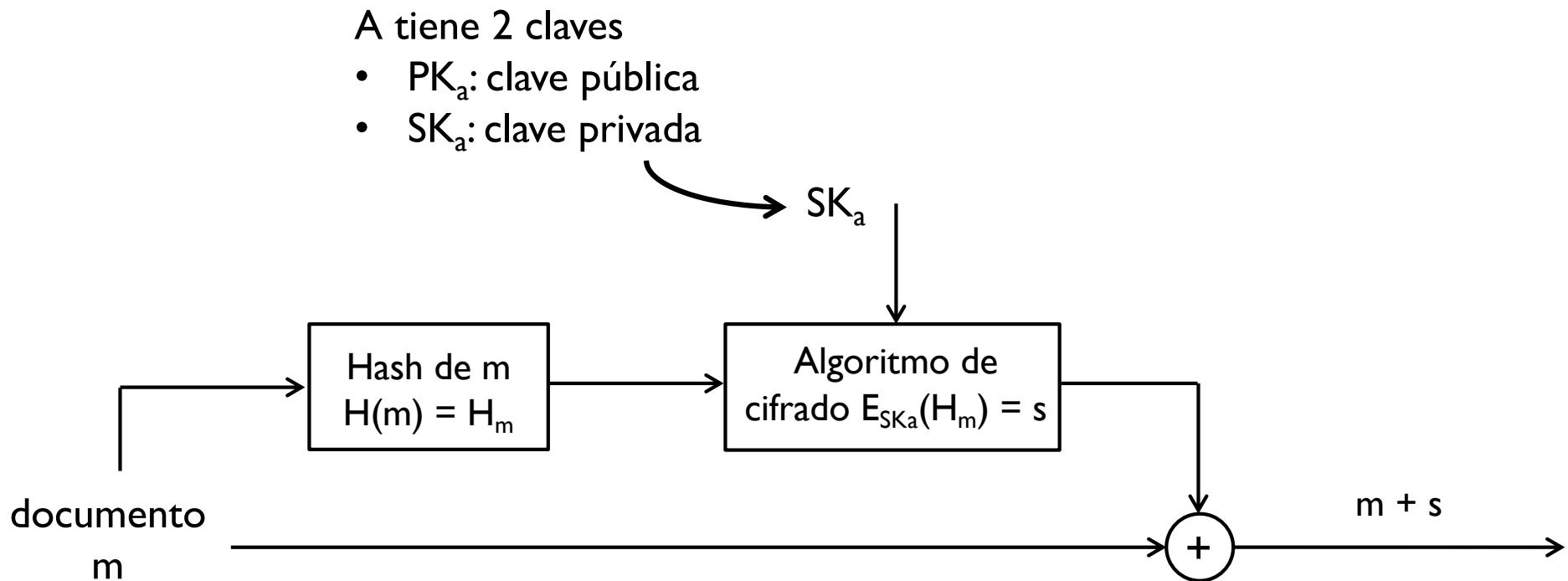
- ▶ Ya que cifrar todo el documento puede resultar computacionalmente costoso, lo que se hace es
  - ▶ Calcular una función de Hash (i.e., resumen de tamaño fijo) sobre el documento
  - ▶ Cifra el resultado de esta función con la clave privada y la añade al documento
  - ▶ Para autenticar la firma, se hace la operación inversa



## 2.7.4 – RSA para Firma Digital

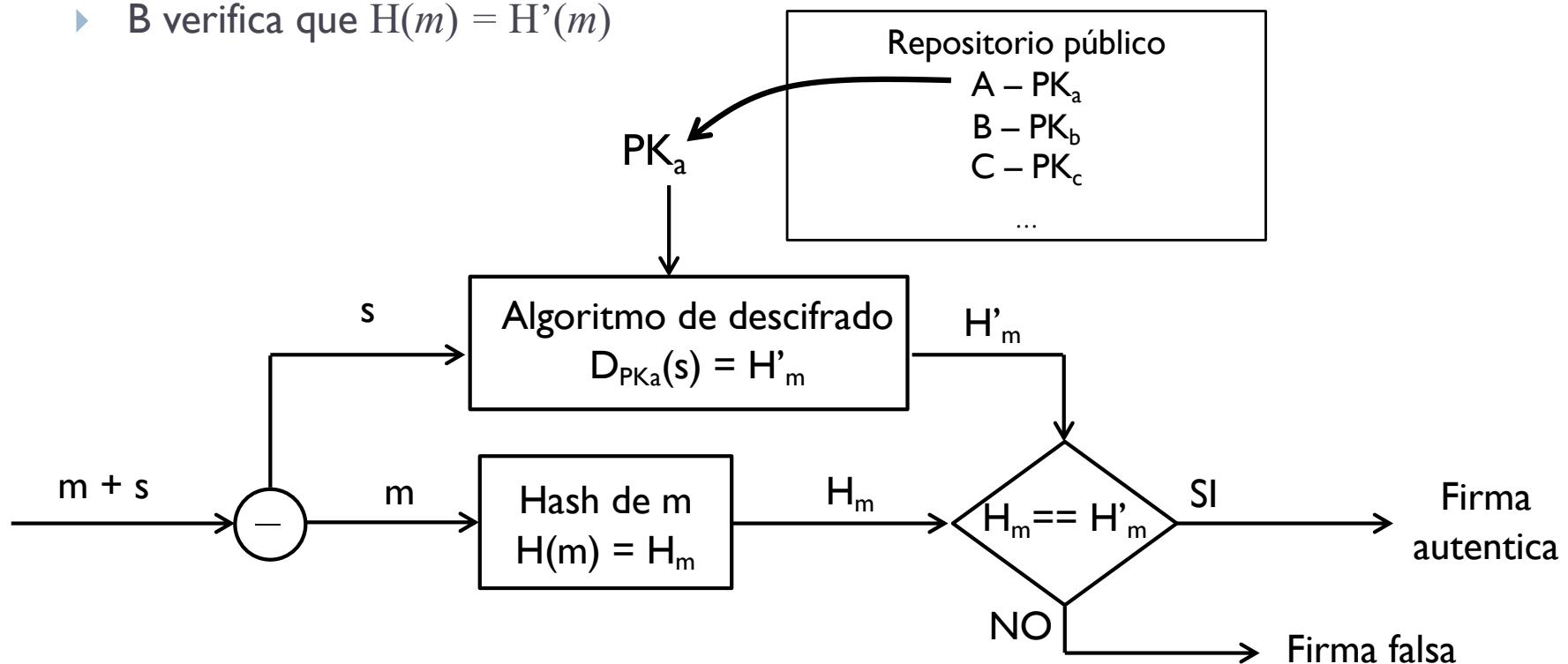
### ▶ Funcionamiento en el firmante A

- ▶ A tiene clave publica  $PK_a = (n,e)$  y privada  $SK_a = (n,d)$
- ▶ A calcula la función Hash  $H(m)$  del documento  $m$
- ▶ A cifra el hash usando la clave privada  $s = (H(m))^d \text{ mod } n$
- ▶ A envía el documento original  $m$  con añadida la firma  $s$



## 2.7.4 – RSA para Firma Digital

- ▶ Verificación de la firma por parte de B
  - ▶ B coge el documento firmado  $m + s$
  - ▶ B separa  $m$  de  $s$
  - ▶ B calcula la función Hash  $H(m)$  del documento  $m$
  - ▶ B descifra la firma con la clave pública de A y encuentra el hash  $H'_m = s^e \text{ mod } n$
  - ▶ B verifica que  $H(m) = H'_m$



## 2.7.4 – Firma Digital

---

- ▶ RSA no es el único algoritmo de cifrado usado en la firma digital
  - ▶ Digital Signature Algorithm (DSA), variante de ElGamal
  - ▶ Elliptic Curve Digital Signature Algorithm (ECDSA)
  - ▶ Edwards-curve Digital Signature Algorithm (EdDSA)

## 2.7.4 – Firma Digital

---

- ▶ Una firma digital no solo permite autenticar una firma
- ▶ **Integridad**
  - ▶ Garantiza la integridad del documento firmado
  - ▶ Si se modifica el documento después de la firma, el proceso de autenticación de la firma verificaría que no es el mismo documento
  - ▶ El Hash  $H(m)$  del documento original firmado por A y el Hash  $H'(m)$  del documento recibido por B después de haber sido modificado serían diferentes
- ▶ **No repudio**
  - ▶ El firmante no puede repudiar el documento firmado
  - ▶ Alguien que ha firmado un documento no puede a posteriori negar que lo haya firmado

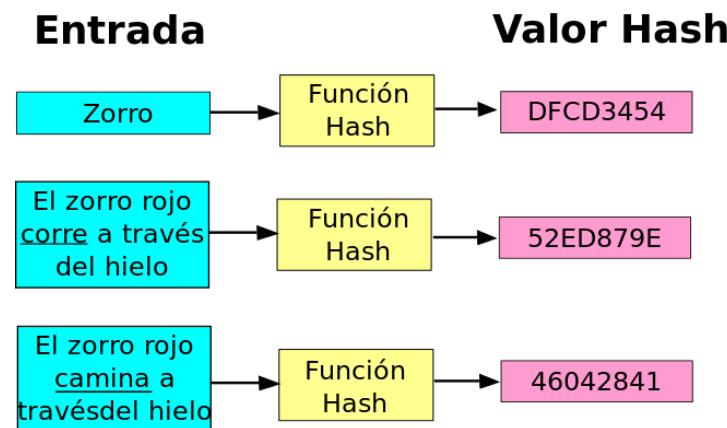
## 2.7 – Algoritmos más conocidos

---

- ▶ Diffie-Hellman
  - ▶ Exponenciación binaria
- ▶ RSA
  - ▶ Números coprimos
  - ▶ Multiplicador modular inverso
- ▶ ElGamal
- ▶ RSA para Firma Digital
- ▶ Función Hash

## 2.7.5 – Función Hash

- ▶ Una función hash es una función computable mediante un algoritmo que tiene como entrada un conjunto de elementos y proporciona una stringa de salida de longitud fija
  - ▶ También se suele llamar función resumen o digest
  - ▶ Un valor hash sirve como representación compacta de la entrada
- ▶ Ejemplo

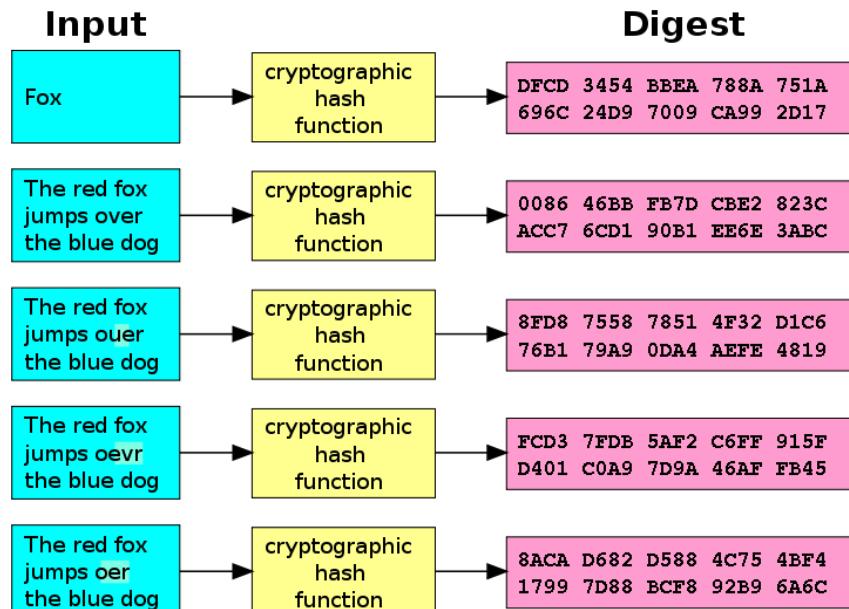


Fuente imagen: wikipedia

## 2.7.5 – Función Hash

### En criptografía

- ▶ Una función hash criptográfica debe tener unas características particulares
  - ▶ Debe ser determinista, es decir una misma entrada debe dar siempre el mismo hash
  - ▶ El hash debe ser fácil de computar
  - ▶ Debe ser inviable generar un mensaje que produzca un valor hash determinado
  - ▶ Debe ser inviable encontrar dos mensajes diferentes con el mismo valor hash
  - ▶ Un pequeño cambio en un mensaje debería cambiar el valor de hash de manera tan extensa que el nuevo valor de hash parece no estar relacionado con el antiguo valor de hash
- ▶ Ejemplo



Fuente imagen: wikipedia

## 2.7.5 – Función Hash

### En criptografía: posibles ataques

---

- ▶ La vulnerabilidad más común es que haya la posibilidad de encontrar una colisión en tiempos razonables
  - ▶ Una colisión se da cuando dos conjuntos diferentes dan el mismo hash
  - ▶ Se define entonces el termino collision resistance
- ▶ La otra posibilidad es calcular el inverso, es decir, conocido el hash, determinar la entrada
  - ▶ Se define entonces el termino preimage resistance

## 2.7.5 – Función Hash

En criptografía: algoritmos más comunes

---

- ▶ Message Digest Algorithm 5 (MD5)
- ▶ Secure Hashing Algorithm (SHA-1, SHA-2 y SHA-3)
- ▶ RACE Integrity Primitives Evaluation Message Digest (RIPEMD)
- ▶ BLAKE2

## 2.7.5 – Función Hash

En criptografía: algoritmos más comunes

---

- ▶ **Message Digest Algorithm 5 (MD5)**
  - ▶ Abril 1992
  - ▶ Hash de 128 bits
  - ▶ Sufre de varias vulnerabilidades, se encuentra una colisión en segundos en un ordenador común
  - ▶ Se usa principalmente como checksum (verifica la integridad de datos)
- ▶ **Secure Hashing Algorithm (SHA-1, SHA-2 y SHA-3)**
- ▶ **RACE Integrity Primitives Evaluation Message Digest (RIPEMD)**
- ▶ **BLAKE2**

Fuente imagen: wikipedia

## 2.7.5 – Función Hash

### En criptografía: algoritmos más comunes

---

- ▶ Message Digest Algorithm 5 (MD5)
- ▶ Secure Hashing Algorithm (SHA-1, SHA-2 y SHA-3)
  - ▶ SHA-1
    - ▶ Serie de algoritmos creada desde el 1995 (NSA)
    - ▶ Hash de 160 bits
    - ▶ Primeras colisiones encontradas en tiempos razonables en el 2005
    - ▶ Se recomendó no usar ya a partir del 2010
  - ▶ SHA-2
    - ▶ Serie de algoritmos creada desde el 2001 (NSA)
    - ▶ 6 diferentes longitudes de Hash: 224, 256, 384 o 512 bits (SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, y SHA-512/256)
    - ▶ Usado en muchos protocolos de seguridad como TLS, SSL, SSH, IPSec
    - ▶ SHA-256 usado en Bitcoin para verificar las transacciones y calcular proof of work/stake
    - ▶ Aunque tienen alguna vulnerabilidad, la complejidad para atacarla sigue siendo inviable actualmente
  - ▶ SHA-3
    - ▶ Serie de algoritmos creada desde el 2015 (estándar NIST)
    - ▶ Hash de tamaño arbitrario: los más comunes son los usados en SHA-2
    - ▶ De momento se siguen usando ambos SHA-3 y SHA-2
- ▶ RACE Integrity Primitives Evaluation Message Digest (RIPEMD)
- ▶ BLAKE2

## 2.7.5 – Función Hash

En criptografía: algoritmos más comunes

---

- ▶ Message Digest Algorithm 5 (MD5)
- ▶ Secure Hashing Algorithm (SHA-1, SHA-2 y SHA-3)
- ▶ RACE Integrity Primitives Evaluation Message Digest (RIPEMD)
  - ▶ Serie de algoritmos desde el 1992 (proyecto europeo RIPE)
  - ▶ Hash de 128, 160, 256 y 320 bits (RIPEMD-128, RIPEMD-160, RIPEMD-256, RIPEMD-320)
  - ▶ Aunque considerado robusto, se usa menos ya que salió después del SHA y suele ser más lento
- ▶ BLAKE2

## 2.7.5 – Función Hash

En criptografía: algoritmos más comunes

---

- ▶ Message Digest Algorithm 5 (MD5)
- ▶ Secure Hashing Algorithm (SHA-1, SHA-2 y SHA-3)
- ▶ RACE Integrity Primitives Evaluation Message Digest (RIPEMD)
- ▶ BLAKE2
  - ▶ Familia de algoritmos desde el 2012
  - ▶ Hash de 224, 256, 384 y 512 bits
  - ▶ Considerado más rápido que los demás, más seguro que SHA-2 y similar a SHA-3
  - ▶ Nueva versión BLAKE3 el 9 de enero 2020
    - ▶ Un único algoritmo, no una familia como BLAKE2
    - ▶ Hash de longitud arbitraria, por defecto 256 bits

# Seguretat Informatica (SI)

Tema 2. Criptografía - Problemas

# Tema 2. Problema 1

---

- ▶ Alice y Bob quiere usar una clave privada para crear un canal seguro usando criptografía DES
  - ▶ Contestar a estas preguntas
- I) ¿Que pueden usar para compartir una clave privada de forma segura?

# Tema 2. Problema 1

---

- ▶ Alice y Bob quiere usar una clave privada para crear un canal seguro usando criptografía DES
  - ▶ Contestar a estas preguntas
- I) ¿Que pueden usar para compartir una clave privada de forma segura?
- El metodo Diffie-Helman
- Hacerlo personalmente
- Usando criptografia pública

# Tema 2. Problema 1

---

- ▶ Alice y Bob quiere usar una clave privada para crear un canal seguro usando criptografía DES
  - ▶ Contestar a estas preguntas
- I) ¿Que pueden usar para compartir una clave privada de forma segura?

Deciden usar Diffie-Helman

- 2) Eligen un grupo cíclico finito  $G$  de 29 y un generador  $\alpha = 2$

Alice elige el número 5

Bob elige el número 12

Describe que valores se intercambian Alice y Bob y que clave privada usaran.

# Tema 2. Problema 1

---

- ▶ Alice y Bob quiere usar una clave privada para crear un canal seguro usando criptografía DES
  - ▶ Contestar a estas preguntas
- I) ¿Que pueden usar para compartir una clave privada de forma segura?

El metodo Diffie-Helman

- 2) Eligen un grupo cíclico finito  $G$  de 29 y un generador  $\alpha = 2$

Alice elige el número 5

Bob elige el número 12

Describe que valores se intercambian Alice y Bob y que clave privada usaran.

Alice calcula  $2^5 \text{ mod } 29 = 3$  y envía 3 a Bob

Bob calcula  $2^{12} \text{ mod } 29 = 7$  y envía 7 a Alice

Alice calcula  $7^5 \text{ mod } 29 = 16$

Bob calcula  $3^{12} \text{ mod } 29 = 16$

16 será la clave privada

# Tema 2. Problema 2

---

- ▶ Alice quiere usar ElGamal para recibir mensajes privados de Bob
  - ▶ Deciden usar el grupo cíclico finito  $G$  de 23 y un  $\alpha = 11$
  - ▶ Contestar a las siguientes preguntas
- 1) Alice elige  $a = 6$  como clave privada. Ayuda Alice a calcular su clave pública
  - 2) Bob quiere enviar el mensaje  $m = 10$  a Alice y elige el número  $b = 3$ .  
Ayuda Bob a calcular el mensaje cifrado  $c$
  - 3) Ayuda Alice a descifrar el mensaje  $c$

# Tema 2. Problema 2

---

- ▶ Alice quiere usar ElGamal para recibir mensajes privados de Bob
  - ▶ Deciden usar el grupo cíclico finito  $G$  de 23 y un  $\alpha = 11$
  - ▶ Contestar a las siguientes preguntas
- 1) Alice elige  $a = 6$  como clave privada. Ayuda Alice a calcular su clave pública
- $$\alpha^a \in G = 11^6 \bmod 23 = 9$$
- Alice envía a Bob la clave pública  $(11, 23, 9)$
- 2) Bob quiere enviar el mensaje  $m = 10$  a Alice y elige el número  $b = 3$ .  
Ayuda Bob a calcular el mensaje cifrado  $c$
- 3) Ayuda Alice a descifrar el mensaje  $c$

# Tema 2. Problema 2

---

- ▶ Alice quiere usar ElGamal para recibir mensajes privados de Bob
  - ▶ Deciden usar el grupo cíclico finito  $G$  de 23 y un  $\alpha = 11$
  - ▶ Contestar a las siguientes preguntas
- 1) Alice elige  $a = 6$  como clave privada. Ayuda Alice a calcular su clave pública
- $$\alpha^a \in G = 11^6 \text{ mod } 23 = 9$$
- Alice envía a Bob la clave pública  $(11, 23, 9)$
- 2) Bob quiere enviar el mensaje  $m = 10$  a Alice y elige el número  $b = 3$ .  
Ayuda Bob a calcular el mensaje cifrado  $c$
- $$\alpha^b \in G = 11^3 \text{ mod } 23 = 20$$
- $$c = m \cdot (\alpha^a)^b \in G = 10 \cdot 9^3 \text{ mod } 23 = 22$$
- Bob envía a Alice el mensaje  $(20, 22)$
- 3) Ayuda Alice a descifrar el mensaje  $c$

# Tema 2. Problema 2

---

- ▶ Alice quiere usar ElGamal para recibir mensajes privados de Bob
- ▶ Deciden usar el grupo cíclico finito  $G$  de 23 y un  $\alpha = 11$
- ▶ Contestar a las siguientes preguntas

1) Alice elige  $a = 6$  como clave privada. Ayuda Alice a calcular su clave pública

$$\alpha^a \in G = 11^6 \bmod 23 = 9$$

Alice envía a Bob la clave pública  $(11, 23, 9)$

2) Bob quiere enviar el mensaje  $m = 10$  a Alice y elige el número  $b = 3$ .

Ayuda Bob a calcular el mensaje cifrado  $c$

$$\alpha^b \in G = 11^3 \bmod 23 = 20$$

$$c = m \cdot (\alpha^a)^b \in G = 10 \cdot 9^3 \bmod 23 = 22$$

Bob envía a Alice el mensaje  $(20, 22)$

3) Ayuda Alice a descifrar el mensaje  $c$

$$x = (\alpha^b)^a \in G = 20^6 \bmod 23 = 16$$

$$x^{-1} \in G = 16^{-1} \bmod 23 = 13$$

$$m = c \cdot x^{-1} \in G = 22 \cdot 13 \bmod 23 = 10$$

## Tema 2. Problema 3

---

- ▶ A usa ElGamal con grupo cíclico finito  $G$  de 991, un  $\alpha = 7$  y su clave privada  $a$  es 323
  - ▶ A recibe el mensaje  $(\alpha^b = 415, c = 862)$
- I) Calcula el mensaje no cifrado

# Tema 2. Problema 3

---

- ▶ A usa ElGamal con grupo cíclico finito  $G$  de 991, un  $\alpha = 7$  y su clave privada  $a$  es 323
- ▶ A recibe el mensaje  $(\alpha^b = 415, c = 862)$

I) Calcula el mensaje no cifrado

$$x = (\alpha^b)^a \in G = 415^{323} \bmod 991 = 850$$

$$x^{-1} \in G = ((\alpha^b)^a)^{-1} \in G = 850^{-1} \bmod 991 = 745$$

$$m = c \cdot x^{-1} \in G = 862 \cdot 745 \bmod 991 = 22$$

# Seguretat Informàtica (SI)

Tema 2. Criptografía

Davide Careglio

Fuentes: Jordi Nin, "Cryptography", Computer Security, 2014  
Jaime Delgado, "Cryptography", Computer Security, 2018