# The ABC of Criptography and its Algorithms [1]

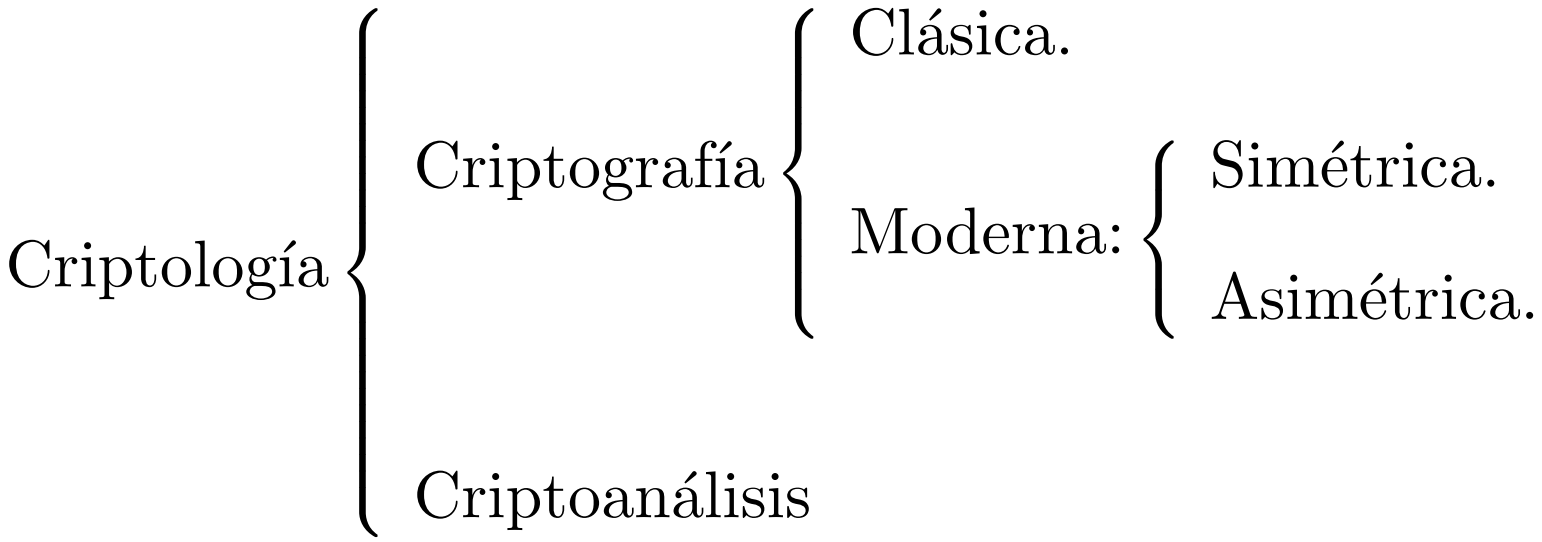
**Fernando Martínez**

**fernando@ma2.upc.es**

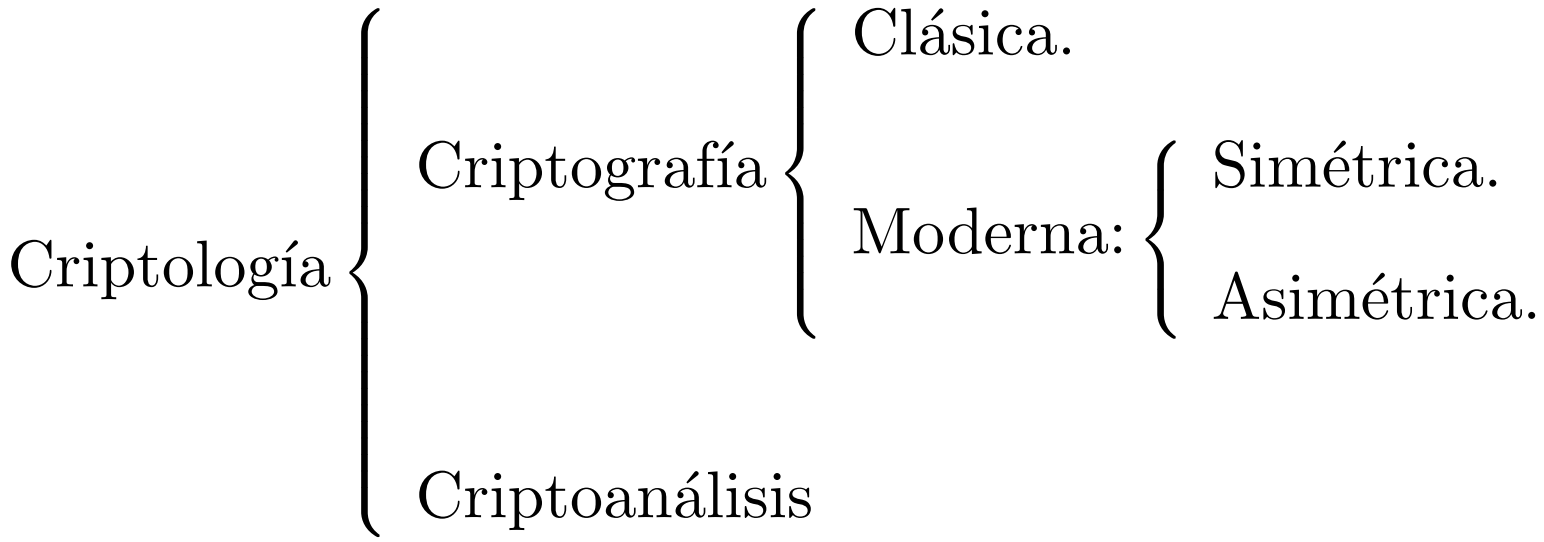
**Matemàtica Aplicada II**

[1] A. Juan Hormigo

# Criptología



# Criptología




**Incondicionalmente seguro**: La probabilidad de que un criptograma provenga de un mensaje determinado es igual a la probabilidad a priori del mensaje.

# Criptología

$$\text{Criptología} \begin{cases} \text{Criptografía} \begin{cases} \text{Clásica.} \\ \text{Moderna:} \begin{cases} \text{Simétrica.} \\ \text{Asimétrica.} \end{cases} \end{cases} \\ \text{Criptoanálisis} \end{cases}$$

**Incondicionalmente seguro**: La probabilidad de que un criptograma provenga de un mensaje determinado es igual a la probabilidad a priori del mensaje.

**Computacionalmente seguro**: Si con *recursos limitados* no puede ser criptoanalizado.

# Criptografía de clave secreta

Se utiliza la misma clave para cifrar y descifrar.

# Criptografía de clave secreta

Se utiliza la misma clave para cifrar y descifrar.

- Cifrado de flujo: La transformación de cifrado varía de símbolo a símbolo. **RC4.**

# Criptografía de clave secreta

Se utiliza la misma clave para cifrar y descifrar.

- Cifrado de flujo: La transformación de cifrado varía de símbolo a símbolo. **RC4.**

- Cifrado de bloque: el mensaje se divide en bloques de igual longitud a los que se les aplica la misma transformación de cifrado. **DES, AES.**

# DES: Data Encryption Standard

`http://csrc.nist.gov/fips/` FIPS 46, 74,81
Cifrado de bloques de 64 bits y clave de 56 bits.

# DES: Data Encryption Standard

`http://csrc.nist.gov/fips/` FIPS 46, 74,81
Cifrado de bloques de 64 bits y clave de 56 bits.

- Se publica en 1975 y es adoptado en 1976 por el gobierno USA para la transmisión y almacenamiento de información no clasificada.

# DES: Data Encryption Standard

`http://csrc.nist.gov/fips/` FIPS 46, 74,81
Cifrado de bloques de 64 bits y clave de 56 bits.

- Se publica en 1975 y es adoptado en 1976 por el gobierno USA para la transmisión y almacenamiento de información no clasificada.
- En 1981 diversos organismos privados lo adoptan como estándard.

# DES: Data Encryption Standard

`http://csrc.nist.gov/fips/` FIPS 46, 74,81
Cifrado de bloques de 64 bits y clave de 56 bits.

- Se publica en 1975 y es adoptado en 1976 por el gobierno USA para la transmisión y almacenamiento de información no clasificada.
- En 1981 diversos organismos privados lo adoptan como estándard.
- En 1987 la NSA se opone a que se siga manteniendo como estándard pero, por motivos económicos, finalmente se renueva.

# DES: Data Encryption Standard

`http://csrc.nist.gov/fips/` FIPS 46, 74,81
Cifrado de bloques de 64 bits y clave de 56 bits.

- Se publica en 1975 y es adoptado en 1976 por el gobierno USA para la transmisión y almacenamiento de información no clasificada.
- En 1981 diversos organismos privados lo adoptan como estándard.
- En 1987 la NSA se opone a que se siga manteniendo como estándard pero, por motivos económicos, finalmente se renueva.
- En 1997, tras 4 meses de cálculo se descifra un mensaje cifrado con el DES. Hoy, el récord está en menos de 23 horas.

# DES: Data Encryption Standard

`http://csrc.nist.gov/fips/` FIPS 46, 74,81
Cifrado de bloques de 64 bits y clave de 56 bits.

- Se publica en 1975 y es adoptado en 1976 por el gobierno USA para la transmisión y almacenamiento de información no clasificada.
- En 1981 diversos organismos privados lo adoptan como estándard.
- En 1987 la NSA se opone a que se siga manteniendo como estándard pero, por motivos económicos, finalmente se renueva.
- En 1997, tras 4 meses de cálculo se descifra un mensaje cifrado con el DES. Hoy, el récord está en menos de 23 horas.
- Será sustituido por el AES (Advanced encryption standard) a partir del 2000. **Rijndael**

# DES: Descripción del algoritmo a alto nivel

1. Dado un bloque $x$, se le aplica una permutación inicial $\pi$,

$$x_0 = \pi(x) \equiv L_0 R_0,$$

# DES: Descripción del algoritmo a alto nivel

1. Dado un bloque $x$, se le aplica una permutación inicial $\pi$,

$$x_0 = \pi(x) \equiv L_0 R_0,$$

2. Se realizan 16 iteraciones del tipo:

$$L_i = R_{i-1},$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, k_i),$$

# DES: Descripción del algoritmo a alto nivel

1. Dado un bloque $x$, se le aplica una permutación inicial $\pi$,

$$x_0 = \pi(x) \equiv L_0 R_0,$$

2. Se realizan 16 iteraciones del tipo:

$$L_i = R_{i-1},$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, k_i),$$

3. Se aplica la permutación inversa de $\pi$ a $R_{16}L_{16}$,

$$y = \pi^{-1}(R_{16}L_{16}).$$

# DES: Generación de subclaves

1. Dada una clave $k$ de 56 bits se le aplica una permutación $P_1$,

$$P_1(k) \equiv C_0 D_0.$$

# DES: Generación de subclaves

1. Dada una clave $k$ de 56 bits se le aplica una permutación $P_1$,

$$P_1(k) \equiv C_0 D_0.$$

2. Para $1 \leq i \leq 16$ se calcula

$$C_i = LS_i(C_{i-1}), \quad D_i = LS_i(D_{i-1}),$$
$$k_i = P_2(C_i D_i),$$

$LS_i$ es una rotación a la izquierda de una posición si $i = 1, 2, 9, 16$ o de dos posiciones para cualquier otro valor de $i$.
$P_2$ es una $permutación$ de compresión, de los 56 bits se eligen 48 en un orden determinado.

# DES: La función $f$

Depende de dos parámetros, el primero es un bloque de 32 bits y el segundo es una subclave de 48 bits y devuelve un bloque de 32 bits, $f(x, k)$.

1. $x$ se expande a un bloque de 48 bits, $E(x)$, consistente en los 32 bits de $x$ permutados más otros 16 bits, también de $x$, repetidos.

# DES: La función $f$

Depende de dos parámetros, el primero es un bloque de 32 bits y el segundo es una subclave de 48 bits y devuelve un bloque de 32 bits, $f(x, k)$.

1. $x$ se expande a un bloque de 48 bits, $E(x)$, consistente en los 32 bits de $x$ permutados más otros 16 bits, también de $x$, repetidos.

2. Se calcula $B = E(x) \oplus k \equiv B_1B_2B_3B_4B_5B_6B_7B_8$, los $B_i$ son bloques de 6 bits.

# DES: La función $f$

Depende de dos parámetros, el primero es un bloque de 32 bits y el segundo es una subclave de 48 bits y devuelve un bloque de 32 bits, $f(x, k)$.

1. $x$ se expande a un bloque de 48 bits, $E(x)$, consistente en los 32 bits de $x$ permutados más otros 16 bits, también de $x$, repetidos.

2. Se calcula $B = E(x) \oplus k \equiv B_1B_2B_3B_4B_5B_6B_7B_8$, los $B_i$ son bloques de 6 bits.

3. Se calcula $C_i = S_i(B_i)$, las $S_i$ (S-boxes) son aplicaciones que a un bloque de 6 bits le asocian un bloque de 4 bits.

# DES: La función $f$

Depende de dos parámetros, el primero es un bloque de 32 bits y el segundo es una subclave de 48 bits y devuelve un bloque de 32 bits, $f(x, k)$.

1. $x$ se expande a un bloque de 48 bits, $E(x)$, consistente en los 32 bits de $x$ permutados más otros 16 bits, también de $x$, repetidos.

2. Se calcula $B = E(x) \oplus k \equiv B_1B_2B_3B_4B_5B_6B_7B_8$, los $B_i$ son bloques de 6 bits.

3. Se calcula $C_i = S_i(B_i)$, las $S_i$ (S-boxes) son aplicaciones que a un bloque de 6 bits le asocian un bloque de 4 bits.

4. $C \equiv C_1C_2C_3C_4C_5C_6C_7C_8$ es un bloque de 32 bits al que se le aplica una permutación $P$,

# DES: La función $f$

Depende de dos parámetros, el primero es un bloque de 32 bits y el segundo es una subclave de 48 bits y devuelve un bloque de 32 bits, $f(x, k)$.

1. $x$ se expande a un bloque de 48 bits, $E(x)$, consistente en los 32 bits de $x$ permutados más otros 16 bits, también de $x$, repetidos.

2. Se calcula $B = E(x) \oplus k \equiv B_1B_2B_3B_4B_5B_6B_7B_8$, los $B_i$ son bloques de 6 bits.

3. Se calcula $C_i = S_i(B_i)$, las $S_i$ (S-boxes) son aplicaciones que a un bloque de 6 bits le asocian un bloque de 4 bits.

4. $C \equiv C_1C_2C_3C_4C_5C_6C_7C_8$ es un bloque de 32 bits al que se le aplica una permutación $P$,

5. $f(x, k) = P(C)$.

# S-box

$S_1$

| | 0110 | | | | | | | | | | 1010 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| 01 | 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 |
| 10 | 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| 11 | 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |

$$B_1 = 101101 \qquad S_1(B_1) = 1 = 0001,$$
$$\tilde{B}_1 = 110100 \qquad S_1(\tilde{B}_1) = 9 = 1001.$$

# DES: Seguridad

1. Tamaño de la clave: Se considera muy pequeño.

# DES: Seguridad

1. Tamaño de la clave: Se considera muy pequeño.

2. No se conoce ninguna técnica de criptoanálisis para atacarlo más eficiente que la fuerza bruta.

# DES: Seguridad

1. Tamaño de la clave: Se considera muy pequeño.

2. No se conoce ninguna técnica de criptoanálisis para atacarlo más eficiente que la fuerza bruta.

3. El DES no tiene estructura de grupo. Se puede aumentar la seguridad mediante aplicaciones sucesivas del DES con distintas claves. Triple DES ($E_{k_1} D_{k_2} E_{k_3}$).

# RIJNDAEL- AES: Advanced Encryption Standard

`http://csrc.nist.gov/encryption/aes/`

Algoritmo simétrico de bloque capaz de soportar las combinaciones clave-bloque de los tamaños 128-128, 192-128 y 256-128.

# RIJNDAEL- AES: Advanced Encryption Standard

`http://csrc.nist.gov/encryption/aes/`

Algoritmo simétrico de bloque capaz de soportar las combinaciones clave-bloque de los tamaños 128-128, 192-128 y 256-128.

- Realiza operaciones a nivel de byte, GF($2^8$), y en términos de palabras de 4 bytes, polinomios a coeficientes en GF($2^8$).

# RIJNDAEL- AES: Advanced Encryption Standard

`http://csrc.nist.gov/encryption/aes/`

Algoritmo simétrico de bloque capaz de soportar las combinaciones clave-bloque de los tamaños 128-128, 192-128 y 256-128.

- Realiza operaciones a nivel de byte, GF($2^8$), y en términos de palabras de 4 bytes, polinomios a coeficientes en GF($2^8$).

- $N_b$ número de bits del bloque dividido por 32.
  $N_k$ número de bits de la clave dividido por 32.

# RIJNDAEL- AES: Advanced Encryption Standard

- El número de rondas, $N_r$, depende de la longitud de la clave y del bloque.

| $N_r$ | $N_b = 4$ | $N_b = 6$ | $N_b = 8$ |
|---|---|---|---|
| $N_k = 4$ | 10 | 12 | 14 |
| $N_k = 6$ | 12 | 12 | 14 |
| $N_k = 8$ | 14 | 14 | 14 |

# RIJNDAEL- AES: Advanced Encryption Standard

- El número de rondas, $N_r$, depende de la longitud de la clave y del bloque.

| $N_r$ | $N_b = 4$ | $N_b = 6$ | $N_b = 8$ |
|---|---|---|---|
| $N_k = 4$ | 10 | 12 | 14 |
| $N_k = 6$ | 12 | 12 | 14 |
| $N_k = 8$ | 14 | 14 | 14 |

- Las diferentes transformaciones actúan sobre un resultado intermedio, `State`, formado por una matriz $4 \times N_b$ de bytes:

| | | | | |
|---|---|---|---|---|
| $m_{0,0}$ | $m_{0,1}$ | $m_{0,2}$ | $m_{0,3}$ | $\ldots$ |
| $m_{1,0}$ | $m_{1,1}$ | $m_{1,2}$ | $m_{1,3}$ | $\ldots$ |
| $m_{2,0}$ | $m_{2,1}$ | $m_{2,2}$ | $m_{2,3}$ | $\ldots$ |
| $m_{3,0}$ | $m_{3,1}$ | $m_{3,2}$ | $m_{3,3}$ | $\ldots$ |

# AES: Descripción del algoritmo a alto nivel

1. `AddRoundKey(State, RoundKey`$_1$`)`

2. `Round(State, RoundKey`$_i$`)`, $i = 1, \ldots, N_r - 1$:

   (a) `ByteSub(State)`
   (b) `ShiftRow(State)`
   (c) `MixColumn(State)`
   (d) `AddRoundKey(State, RoundKey`$_i$`)`

3. `FinalRound(State, RoundKey`$_{N_r}$`)`:

   (a) `ByteSub(State)`
   (b) `ShiftRow(State)`
   (c) `AddRoundKey(State, RoundKey`$_{N_r}$`)`

# **AES:** `ByteSuB`

Transformación no lineal de sustitución de bytes (S-box).

1. Toma el inverso multiplicativo en GF($2^8$).

2. Aplica la transformación afín sobre GF($2$):

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

# **AES:** `ShiftRow`

Las filas de `State` se desplazan cíclicamente, la primera no sufre desplazamiento, la segunda se desplaza $C_1$ posiciones, la tercera $C_2$ y la cuarta $C_3$:

| $N_b$ | $C_1$ | $C_2$ | $C_3$ |
|---|---|---|---|
| 4 | 1 | 2 | 3 |
| 6 | 1 | 2 | 3 |
| 8 | 1 | 3 | 4 |

| | | | | |
|---|---|---|---|---|
| $m_{0,0}$ | $m_{0,1}$ | $m_{0,2}$ | $m_{0,3}$ | $\ldots$ |
| $m_{1,0}$ | $m_{1,1}$ | $m_{1,2}$ | $m_{1,3}$ | $\ldots$ |
| $m_{2,0}$ | $m_{2,1}$ | $m_{2,2}$ | $m_{2,3}$ | $\ldots$ |
| $m_{3,0}$ | $m_{3,1}$ | $m_{3,2}$ | $m_{3,3}$ | $\ldots$ |

$\Longrightarrow$

| | | | | |
|---|---|---|---|---|
| $m_{0,0}$ | $m_{0,1}$ | $m_{0,2}$ | $m_{0,3}$ | $\ldots$ |
| $m_{1,1}$ | $m_{1,2}$ | $m_{1,3}$ | $\ldots$ | $m_{1,0}$ |
| $m_{2,2}$ | $m_{2,3}$ | $\ldots$ | $m_{2,0}$ | $m_{2,1}$ |
| $m_{3,3}$ | $\ldots$ | $m_{3,0}$ | $m_{3,1}$ | $m_{3,2}$ |

# **AES:** `MixColumn`

Las columnas de `State` son consideradas polinomios sobre GF($2^8$) y multiplicadas módulo $x^4 + 1$ por el polinomio:

$$c(x) = \texttt{0x03}\, x^3 + \texttt{0x01}\, x^2 + \texttt{0x01}\, x + \texttt{0x02}.$$

Si $b(x) = c(x) \otimes a(x)$,

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} \texttt{0x02} & \texttt{0x03} & \texttt{0x01} & \texttt{0x01} \\ \texttt{0x01} & \texttt{0x02} & \texttt{0x03} & \texttt{0x01} \\ \texttt{0x01} & \texttt{0x01} & \texttt{0x02} & \texttt{0x03} \\ \texttt{0x03} & \texttt{0x01} & \texttt{0x01} & \texttt{0x02} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

# **AES:** `AddRoundKey`

Consiste en un `XOR` entre `State` y `RounKey`.

$$\texttt{State} \oplus \texttt{RounKey}$$

# AES: Generación de subclaves (I)

La clave se extiende a una lista de palabras de 4 bytes que llamaremos $W$ y que contiene $N_b(N_r + 1)$ palabras. Los primeros $N_k$ elementos de $W$ corresponden a la clave, el resto se definen recursivamente utilizando la función `SubByte`, desplazamientos cíclicos y $\oplus$. La recurrencia depende de la longitud de la clave.

$N_k \leq 6$

```
KeyExpansion(byte Key[4*Nk] word W[Nb*(Nr+1)])
{
  for(i = 0; i < Nk; i++)
    W[i] = (Key[4*i],Key[4*i+1],Key[4*i+2],Key[4*i+3]);

  for(i = Nk; i < Nb * (Nr + 1); i++)
  {
    temp = W[i - 1];
    if (i % Nk == 0)
            temp = SubByte(RotByte(temp)) ^ Rcon[i / Nk];
    W[i] = W[i - Nk] ^ temp;
  }
}
```

# AES: Generación de subclaves (y II)

La función `RotByte` devuelve una palabra cuyos bytes se han desplazado cíclicamente una posición.

Las constantes de cada ronda vienen definidas por

$$Rcon[i] = (RC[i], \texttt{0x00}, \texttt{0x00}, \texttt{0x00}),$$

siendo $RC[i]$ un elemento de GF($2^8$) definido por

$$RC[1] = \texttt{0x01}, \qquad RC[i] = \texttt{0x02} \bullet RC[i-1].$$

# Modos de operación

- **ECB: Electronic CodeBook**

$$c_i = E_k(m_i), \qquad m_i = D_k(c_i).$$

# Modos de operación

- **ECB: Electronic CodeBook**

$$c_i = E_k(m_i), \qquad m_i = D_k(c_i).$$

- **CBC: Cipher Block Chaining**

  Se inicializa $c_0$ aleatorio,

$$c_i = E_k(m_i \oplus c_{i-1}), \qquad m_i = D_k(c_i) \oplus c_{i-1}.$$

# Modos de operación

- **ECB: Electronic CodeBook**

$$c_i = E_k(m_i), \qquad m_i = D_k(c_i).$$

- **CBC: Cipher Block Chaining**

  Se inicializa $c_0$ aleatorio,

$$c_i = E_k(m_i \oplus c_{i-1}), \qquad m_i = D_k(c_i) \oplus c_{i-1}.$$

  Se puede utilizar como MAC (Message Authentication Code)

# Criptografía de clave pública

Se utilizan dos claves, una para cifrar y otra para descifrar.

# Criptografía de clave pública

Se utilizan dos claves, una para cifrar y otra para descifrar.

Estos criptosistemas se basan en la utilización de funciones unidireccionales (*one-way functions*) con puerta trasera *trap-door*.

# Criptografía de clave pública

Se utilizan dos claves, una para cifrar y otra para descifrar.

Estos criptosistemas se basan en la utilización de funciones unidireccionales ($one\text{-}way\ functions$) con puerta trasera $trap\text{-}door$.

- Función unidireccional: Es una aplicación $f$ tal que existe un algoritmo *polinómico* para calcular $f(x)$ pero no existen algoritmos polinómicos para calcular su inversa $f^{-1}(y)$.

# Criptografía de clave pública

Se utilizan dos claves, una para cifrar y otra para descifrar.

Estos criptosistemas se basan en la utilización de funciones unidireccionales ($one\text{-}way\ functions$) con puerta trasera $trap\text{-}door$.

- Función unidireccional: Es una aplicación $f$ tal que existe un algoritmo *polinómico* para calcular $f(x)$ pero no existen algoritmos polinómicos para calcular su inversa $f^{-1}(y)$.

- Puerta trasera para una función unidireccional $f$ es una información que permite diseñar un algoritmo *polinómico* para calcular $f^{-1}(y)$.

# Criptografía de clave pública

$$\text{Clave pública} \begin{cases} \text{IF} \\ \\ \text{DL} \begin{cases} \text{Cuerpos finitos:} \begin{cases} \mathbb{Z}_p. \\ \\ \mathbb{F}_{2^n}. \end{cases} \\ \\ \text{Curvas elípticas.} \end{cases} \end{cases}$$

# RSA: Rivest-Shamir-Adleman 1978

Los mensajes y criptogramas los consideraremos elementos de $\mathbb{Z}_n$, siendo $n = pq$, $p$, $q$ primos.

# RSA: Rivest-Shamir-Adleman 1978

Los mensajes y criptogramas los consideraremos elementos de $\mathbb{Z}_n$, siendo $n = pq$, $p$, $q$ primos.

- Clave pública: $\{e, n\}$, $e$ arbitrario tal que $(e, \phi(n)) = 1$, $\phi(n) = (p-1)(q-1)$.

# RSA: Rivest-Shamir-Adleman 1978

Los mensajes y criptogramas los consideraremos elementos de $\mathbb{Z}_n$, siendo $n = pq$, $p$, $q$ primos.

- Clave pública: $\{e, n\}$, $e$ arbitrario tal que $(e, \phi(n)) = 1$, $\phi(n) = (p-1)(q-1)$.

- Clave privada: $\{d, p, q\}$, $d$ inverso de $e$ módulo $\phi(n)$.

# RSA: Rivest-Shamir-Adleman 1978

Los mensajes y criptogramas los consideraremos elementos de $\mathbb{Z}_n$, siendo $n = pq$, $p$, $q$ primos.

- Clave pública: $\{e, n\}$, $e$ arbitrario tal que $(e, \phi(n)) = 1$, $\phi(n) = (p-1)(q-1)$.

- Clave privada: $\{d, p, q\}$, $d$ inverso de $e$ módulo $\phi(n)$.

- Cifrado: $c \equiv m^e \mod n$.

# RSA: Rivest-Shamir-Adleman 1978

Los mensajes y criptogramas los consideraremos elementos de $\mathbb{Z}_n$, siendo $n = pq$, $p$, $q$ primos.

- Clave pública: $\{e, n\}$, $e$ arbitrario tal que $(e, \phi(n)) = 1$, $\phi(n) = (p-1)(q-1)$.
- Clave privada: $\{d, p, q\}$, $d$ inverso de $e$ módulo $\phi(n)$.
- Cifrado: $c \equiv m^e \mod n$.
- Descifrado: $m \equiv c^d \mod n$.

# RSA: Rivest-Shamir-Adleman 1978

Los mensajes y criptogramas los consideraremos elementos de $\mathbb{Z}_n$, siendo $n = pq$, $p$, $q$ primos.

- Clave pública: $\{e, n\}$, $e$ arbitrario tal que $(e, \phi(n)) = 1$, $\phi(n) = (p-1)(q-1)$.
- Clave privada: $\{d, p, q\}$, $d$ inverso de $e$ módulo $\phi(n)$.
- Cifrado: $c \equiv m^e \mod n$.
- Descifrado: $m \equiv c^d \mod n$.
- Firma: $f \equiv m^d \mod n$.

# RSA: Rivest-Shamir-Adleman 1978

Los mensajes y criptogramas los consideraremos elementos de $\mathbb{Z}_n$, siendo $n = pq$, $p$, $q$ primos.

- Clave pública: $\{e, n\}$, $e$ arbitrario tal que $(e, \phi(n)) = 1$, $\phi(n) = (p-1)(q-1)$.
- Clave privada: $\{d, p, q\}$, $d$ inverso de $e$ módulo $\phi(n)$.
- Cifrado: $c \equiv m^e \mod n$.
- Descifrado: $m \equiv c^d \mod n$.
- Firma: $f \equiv m^d \mod n$.
- Verificación de la firma: $f^e \equiv m \mod n$.

# RSA en la práctica

- Clave pública: $\{e, n\}$, $e = 2^{16} + 1$, $n = pq$, $p$ y $q$ primos tales que $(e, \phi(n)) = 1$.
- Clave privada: $\{d, p, q, d_p, d_q, P_q, Q_p\}$,
  $d$ inverso de $e$ módulo $\phi(n)$,
  $d_p \equiv d \mod p - 1$, $d_q \equiv d \mod q - 1$,
  $P_q$ inverso de $p$ módulo $q$, $Q_p$ inverso de $Q$ módulo $p$.
- Cifrado: $c \equiv m^e \mod n$.
- Descifrado:

$$m_1 \equiv c^{d_p} \mod p,$$
$$m_2 \equiv c^{d_q} \mod q,$$
$$m \equiv m_1 Q_p q + m_2 P_q p \mod n.$$

- Firma:

$$f_1 \equiv m^{d_p} \mod p,$$
$$f_2 \equiv m^{d_q} \mod q,$$
$$f \equiv f_1 Q_p q + f_2 P_q p \mod n.$$

- Verificación de la firma: $f^e \equiv m \mod n$.

# RSA: Cuestiones

- Cálculo de m.c.d. e inverso modular:

# RSA: Cuestiones

- Cálculo de m.c.d. e inverso modular: algoritmo extendido de Euclides.

# RSA: Cuestiones

- Cálculo de m.c.d. e inverso modular: algoritmo extendido de Euclides.
- Búsqueda de números primos:

# RSA: Cuestiones

- Cálculo de m.c.d. e inverso modular: algoritmo extendido de Euclides.
- Búsqueda de números primos: test probabilístico de Miller-Rabin.

# RSA: Cuestiones

- Cálculo de m.c.d. e inverso modular: algoritmo extendido de Euclides.
- Búsqueda de números primos: test probabilístico de Miller-Rabin.
- *Autentificación* de los usuarios:

# RSA: Cuestiones

- Cálculo de m.c.d. e inverso modular: algoritmo extendido de Euclides.
- Búsqueda de números primos: test probabilístico de Miller-Rabin.
- *Autentificación* de los usuarios: certificados digitales.

# RSA: Cuestiones

- Cálculo de m.c.d. e inverso modular: algoritmo extendido de Euclides.
- Búsqueda de números primos: test probabilístico de Miller-Rabin.
- *Autentificación* de los usuarios: certificados digitales.
- Seguridad:

# RSA: Cuestiones

- Cálculo de m.c.d. e inverso modular: algoritmo extendido de Euclides.
- Búsqueda de números primos: test probabilístico de Miller-Rabin.
- *Autentificación* de los usuarios: certificados digitales.
- Seguridad:
  - ⋆ Cálculo de raíces:

# RSA: Cuestiones

- Cálculo de m.c.d. e inverso modular: algoritmo extendido de Euclides.
- Búsqueda de números primos: test probabilístico de Miller-Rabin.
- *Autentificación* de los usuarios: certificados digitales.
- Seguridad:
  - ⋆ Cálculo de raíces: Se cree que es equivalente a factorizar $n$ pero podría ser más fácil.

# RSA: Cuestiones

- Cálculo de m.c.d. e inverso modular: algoritmo extendido de Euclides.
- Búsqueda de números primos: test probabilístico de Miller-Rabin.
- *Autentificación* de los usuarios: certificados digitales.
- Seguridad:
  - ⋆ Cálculo de raíces: Se cree que es equivalente a factorizar $n$ pero podría ser más fácil.
  - ⋆ Factorización de $n$

# RSA: Cuestiones

- Cálculo de m.c.d. e inverso modular: algoritmo extendido de Euclides.
- Búsqueda de números primos: test probabilístico de Miller-Rabin.
- *Autentificación* de los usuarios: certificados digitales.
- Seguridad:
  - ⋆ Cálculo de raíces: Se cree que es equivalente a factorizar $n$ pero podría ser más fácil.
  - ⋆ Factorización de $n$ ó cálculo de $\phi(n)$.

# RSA: Factorización

- Factorización por división. $\bigcirc\,(\sqrt{n})$
- Factorización de Fermat. $\bigcirc\left(\frac{n+1}{2}-\sqrt{n}\right)$
- $\rho$-método de Pollard. $\bigcirc\left(n^{1/4}\log^2 n\right)$.
- Método $p-1$. $\bigcirc\left(n^{1/4}\right)$.
- Curvas elípticas. $\bigcirc\left(n^{\sqrt{(2+\circ(1))\log\log n/\log n}}\right)$.
- Criba cuadrática. $\bigcirc\left(n^{(1+\circ(1))\sqrt{\log\log n/\log n}}\right)$.
- Criba del cuerpo de números. $\bigcirc\left(\exp(c(\log n)^{1/3}(\log\log n)^{2/3})\right)$, $c=\sqrt[3]{\frac{64}{9}}\approx 1.923$.

# RSA: Recomendaciones

1. El tamaño de $p$ y $q$ debe de ser de unos 512 bits y el de $n$ no debe ser inferior a 1024 bits, recomendándose módulos de tamaño mayor.

# RSA: Recomendaciones

1. El tamaño de $p$ y $q$ debe de ser de unos 512 bits y el de $n$ no debe ser inferior a 1024 bits, recomendándose módulos de tamaño mayor.

2. $p - q$ no debe de ser pequeño.

# RSA: Recomendaciones

1. El tamaño de $p$ y $q$ debe de ser de unos 512 bits y el de $n$ no debe ser inferior a 1024 bits, recomendándose módulos de tamaño mayor.

2. $p - q$ no debe de ser pequeño.

3. $p$ y $q$ deben ser primos fuertes. $p$ es un número primo fuerte si es primo y además:

   (a) $p - 1$ tiene un factor primo grande, $r$;
   (b) $p + 1$ tiene un factor primo grande;
   (c) $r - 1$ tiene un factor primo grande.

Paral·lelització del criptosistema RSA. Jordi Giné León. PFC Enginyeria en informàtica dirgit per X. Martorell. Febrer 1999.

# RSA: Criba cuadrática (I)

Dados $n = pq$, $y$ y $z$ tales que $y \neq \pm z \mod n$ y $y^2 \equiv z^2 \mod n$ entonces $(y - z, n) = p$ ó $q$.

# RSA: Criba cuadrática (I)

Dados $n = pq$, $y$ y $z$ tales que $y \neq \pm z \mod n$ y $y^2 \equiv z^2 \mod n$ entonces $(y - z, n) = p$ ó $q$.

Se generan $x_1, \ldots, x_m$. Para cada número se cálcula $x_i^2 \mod n$ que se intenta factorizar sobre una base de primos $\{2, 3, \ldots, p_k\}$. En el caso de ser posible tendremos

$$x_i^2 \mod n = \Pi_{j=1}^{k} p_j^{e_j}.$$

# RSA: Criba cuadrática (I)

Dados $n = pq$, $y$ y $z$ tales que $y \neq \pm z \mod n$ y $y^2 \equiv z^2 \mod n$ entonces $(y - z, n) = p$ ó $q$.

Se generan $x_1, \ldots, x_m$. Para cada número se cálcula $x_i^2 \mod n$ que se intenta factorizar sobre una base de primos $\{2, 3, \ldots, p_k\}$. En el caso de ser posible tendremos

$$x_i^2 \mod n = \Pi_{j=1}^{k} p_j^{e_j}.$$

Cuando obtengamos $k + 1$, por eliminación gausiana podremos escribir uno de los vectores $(e_1, \ldots, e_k)$ como combinación lineal (módulo 2) de los otros. Entonces

$$\Pi_{i=1}^{m} \left(x_i^2\right)^{b_i} \equiv \Pi_{j=1}^{k} p_j^{c_j} \mod n,$$

siendo $b_i$ 0 ó 1 (dependiendo de si pertenece o no a la combinación lineal) y $c_j$ la suma de las multiplicidades de los primos.

# RSA: Criba cuadrática (y II)

Definiendo

$$y \equiv \Pi_{i=1}^{m} x_i^{b_i} \mod n \quad \text{y} \quad z \equiv \Pi_{j=1}^{k} p_j^{c_j/2} \mod n$$

entonces

$$y^2 \equiv z^2 \mod n.$$

# RSA: Criba cuadrática (y II)

Definiendo

$$y \equiv \Pi_{i=1}^{m} x_i^{b_i} \mod n \quad \text{y} \quad z \equiv \Pi_{j=1}^{k} p_j^{c_j/2} \mod n$$

entonces

$$y^2 \equiv z^2 \mod n.$$

Con $m = 100\,000\,000$ y $k = 200\,000$ la criba se puede realizar en unos pocos segundos (5-10) obteniendo poquísimos números cuyo cuadrado sea factorizable.

# PRESS RELEASE
# CWI, Amsterdam - August 26, 1999
# RSA-155 (512-bit number)

About 300 fast SGI workstations and Pentium PCs did the work, mostly on nights and weekends, over the course of seven months. The algorithm used was the General Number Field Sieve. The algorithm has two parts: a sieving step and a matrix reduction step. The sieving step was the part that the 300 computers worked on: about 8000 MIPS-years over 3.7 months. (This is the step that Shamir's TWINKLE device can speed up.) The matrix reduction step took 224 CPU hours (and about 3.2 Gig of memory) on the Cray C916 at the SARA Amsterdam Academic Computer Center.

The entire effort was 50 times easier than breaking DES.

# ElGamal

Los mensajes y criptogramas los consideraremos elementos de $\mathbb{Z}_p$, $p$ primo. Además, sean $g$ generador de $\mathbb{Z}_p^*$, $2 \leq a \leq p-2$ y $\alpha \equiv g^a \mod p$.

# ElGamal

Los mensajes y criptogramas los consideraremos elementos de $\mathbb{Z}_p$, $p$ primo. Además, sean $g$ generador de $\mathbb{Z}_p^*$, $2 \leq a \leq p-2$ y $\alpha \equiv g^a \mod p$.

- Clave pública: $\{p, g, \alpha\}$.

# ElGamal

Los mensajes y criptogramas los consideraremos elementos de $\mathbb{Z}_p$, $p$ primo. Además, sean $g$ generador de $\mathbb{Z}_p^*$, $2 \leq a \leq p-2$ y $\alpha \equiv g^a \mod p$.

- Clave pública: $\{p, g, \alpha\}$.

- Clave privada: $a$.

# ElGamal

Los mensajes y criptogramas los consideraremos elementos de $\mathbb{Z}_p$, $p$ primo. Además, sean $g$ generador de $\mathbb{Z}_p^*$, $2 \leq a \leq p-2$ y $\alpha \equiv g^a \mod p$.

- Clave pública: $\{p, g, \alpha\}$.

- Clave privada: $a$.

- Cifrado:

  ⋆ Se elige $k$ aleatorio $1 \leq k \leq p-2$.
  ⋆ Se calcula $\gamma \equiv g^k \mod p$ y $\delta \equiv m\alpha^k \mod p$.
  ⋆ El criptograma es el par $(\gamma, \delta)$.

# ElGamal

Los mensajes y criptogramas los consideraremos elementos de $\mathbb{Z}_p$, $p$ primo. Además, sean $g$ generador de $\mathbb{Z}_p^*$, $2 \leq a \leq p-2$ y $\alpha \equiv g^a \mod p$.

- Clave pública: $\{p, g, \alpha\}$.

- Clave privada: $a$.

- Cifrado:

  ⋆ Se elige $k$ aleatorio $1 \leq k \leq p-2$.
  ⋆ Se calcula $\gamma \equiv g^k \mod p$ y $\delta \equiv m\alpha^k \mod p$.
  ⋆ El criptograma es el par $(\gamma, \delta)$.

- Descifrado: $m \equiv \gamma^{p-1-a}\delta \mod p$.

# DSS: Digital Signature Standard (I)

`http://csrc.nist.gov/fips/` FIPS 186

Modificación de ElGamal propuesto por el NIST en 1991 y adoptado como estándard en 1994.

## DSS: Generación de claves

Cada usuario realiza los siguientes pasos:

1. Elige $q$ primo, $2^{159} < q < 2^{160}$ (160 bits) y $p$ primo de 1024 bits tales que $q|(p-1)$.

2. Elige $h \in \mathbb{Z}_p^*$ y calcula $g = h^{\frac{p-1}{q}} \mod p$. Si $g = 1$ se toma otro $h$.

3. Elige $x \in [2, q-1]$ aleatorio y calcula $y = g^x \mod p$.

4. La clave pública es $\{p, q, g, y\}$, la clave privada es $x$.

# DSS: Digital Signature Standard (II)

## DSS: Firma

Para firmar un mensaje $m$ se realizan los siguientes pasos:

1. Elige $k \in [2, q-1]$ aleatorio.
2. Calcula $r = (g^k \mod p) \mod q$.
3. Calcula $k^{-1} \mod q$.
4. Calcula $s = k^{-1}(SHA1(m) + xr) \mod q$. Si $s = 0$ se toma otro $k$.
5. La firma es el par $(r, s)$.

# DSS: Digital Signature Standard (y III)

## DSS: Verificación de la firma

Para verificar que la firma $(r, s)$ del mensaje $m$ es de A se realizan los siguientes pasos:

1. Se busca la clave pública de A, $\{p, q, g, y\}$.
2. Calcula $w = s^{-1} \mod q$.
3. Calcula $u_1 = SHA1(m)w \mod q$.
4. Calcula $u_2 = rw \mod q$.
5. Calcula $v = (g^{u_1} y^{u_2} \mod p) \mod q$.
6. La firma es válida si $v = r$.

# Certificados digitales (I)

Un certificado digital es un fichero digital intransferible y no modificable emitido por una tercera parte de confianza (AC).

# Certificados digitales (I)

Un certificado digital es un fichero digital intransferible y no modificable emitido por una tercera parte de confianza (AC).

Un certificado digital que siga el standard X509v3 contiene la siguiente información:

- Identificación del titular del certificado: Nombre, dirección, etc.
- Clave pública del titular del certificado.
- Fecha de validez.
- Identificación del emisor del certificado.

# Certificados digitales (y II)

```
issuer :/C=ES /ST=Catalunya /L=Bcn /O=SECURITY BCN
        /OU=seccio d'empreses /CN=David Guerrero Vidal
        /Email=guerrero@grec.upc.es
subject:/C=ES /ST= /O= /OU= /CN=Calvin & Hobbes
        /Email=calvin@hobbes
serial :15
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number: 21 (0x15)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=ES, ST=Catalunya, L=Bcn,
                O=SECURITY BCN, OU=seccio d'empreses,
                CN=David Guerrero Vidal
                /Email=guerrero@grec.upc.es
        Validity
            Not Before: Nov 18 15:15:31 1998 GMT
            Not After : Nov 13 15:15:31 1999 GMT
        Subject: C=ES, ST=, O=, OU=, CN=Calvin & Hobbes
                 /Email=calvin@hobbes
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
```

```
                Modulus (1024 bit):
                    00:9e:74:de:c9:1a:6b:f4:fe:d1:04:30:58:7e:8b:
                    51:7a:98:23:e9:45:a9:c2:a7:7c:f8:f8:b5:9a:a2:
                    ea:c1:99:68:ba:f7:c3:d8:06:05:1b:6a:47:a1:44:
                    5c:2c:a6:e0:4b:6f:ce:02:c4:06:32:20:34:be:13:
                    97:39:a3:aa:6f:2f:41:a7:bc:14:c8:f3:0c:ad:9d:
                    09:63:8a:f5:eb:60:5b:06:a6:01:fb:1a:07:b2:c6:
                    39:48:bb:b7:00:56:4e:20:6d:87:3f:67:0b:2f:f4:
                    b0:5f:74:7f:90:6b:b4:47:6f:56:1a:b5:c5:42:54:
                    9b:e5:e3:00:e2:4f:e3:14:47
                Exponent: 65537 (0x10001)
    Signature Algorithm: md5WithRSAEncryption
        3b:2b:e9:ff:48:48:35:ab:30:5c:e2:d1:88:c9:29:8b:bc:09:
        b2:58:80:17:9c:e7:08:0a:7d:8a:5e:46:a8:83:3b:ee:84:de:
        62:e3:ea:51:cb:92:bc:fa:db:90:bd:cd:9f:25:d4:4a:48:63:
        ac:b8:93:f9:dc:9c:cf:ef:fd:45

-----BEGIN CERTIFICATE-----
MIICOzCCAeUCARUwDQYJKoZIhvcNAQEEBQAwgaYxCzAJBgNVBAYTAkVTMRIwEAYD
VQQIEwlDYXRhbHVueWExDDAKBgNVBAcTA0JjbjEVMBMGA1UEChMMU0VDVVJJVFkg
QkNOMRowGAYDVQQLExFzZWNjaW8gZCdlbXByZXNlczEdMBsGA1UEAxMURGF2aWQg
R3VlcnJlcm8gVmlkYWwxIzAhBgkqhkiG9w0BCQEWFGd1ZXJyZXJvQGdyZWMudXBj
LmVzMB4XDTk4MTExODE1MTUzMVoXDTk5MTExMzE1MTUzMVowZjELMAkGA1UEBhMC
RVMxCTAHBgNVBAgTADEJMAcGA1UEChMAMQkwBwYDVQQLEwAxGDAWBgNVBAMUD0Nh
bHZpbiAmIEhvYmJlczEcMBoGCSqGSIb3DQEJARYNY2FsdmluQGhvYmJlczCBnzAN
BgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAnnTeyRpr9P7RBDBYfotRepgj6UWpwqd8
+Pi1mqLqwZlouvfD2AYFG2pHoURcLKbgS2/OAsQGMiA0vhOXOaOqby9Bp7wUyPMM
rZ0JY4r162BbBqYB+xoHssY5SLu3AFZOIG2HP2cLL/SwX3R/kGu0R29WGrXFQlSb
5eMA4k/jFEcCAwEAATANBgkqhkiG9w0BAQQFAANBADsr6f9ISDWrMFzi0YjJKYu8
CbJYgBec5wgKfYpeRqiDO+6E3mLj6lHLkrz625C9zZ8l1EpIY6y4k/ncnM/v/UU=
-----END CERTIFICATE-----
```

# Lanzamiento de monedas (I)

- $B$ lanza una moneda.
- $A$ intenta adivinar el resultado.
- $B$ le comunica el resultado a $A$. Si éste último ha acertado, gana.

Se pide que $A$ no conozca el resultado antes de emitir su pronóstico y que $B$ no sea capaz cambiar el resultado una vez que conozca la predicción de $A$.

# Lanzamiento de monedas (II)

1. $A$ elige $p$ y $q$ primos congruentes con 3 módulo 4 y envía $n = pq$ a $B$.

2. $B$ elige $r < \frac{n}{2}$ aleatorio. calcula $z \equiv r^2 \mod n$. Por último envía $z$ a $A$.

3. $A$ calcula las cuatro raíces de $z$ módulo $n$, $\pm x$, $\pm y$. Sean $r_1 = \min(x, -x \mod n)$ y $r_2 = \min(y, -y \mod n)$. $r$ es $r_1$ o $r_2$ ya que es menor que $\frac{n}{2}$.

4. $A$ pronostica que $r$ es igual a $r_1$ o $r_2$ y envía su pronóstico a $B$.

5. $B$ revela $r$ a $A$.

Si $A$ ha acertado gana en caso contrario pierde. $B$ no puede modificar el valor de $r$ ya que no conoce la descomposición de $n$.

# Problema de la mochila

Consideremos la sucesión $\{1, 5, 8, 9, 10, 15, 16, 19, 20, 23\}$ y pensemos cada letra del alfabeto como una cadena binaria de cinco bits

A=00000,..., I=01001,..., N=01110, O=01111,..., S=10011,...

Para cifrar la palabra SI el proceso a seguir sería:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 5 | 8 | 9 | 10 | 15 | 16 | 19 | 20 | 23 | |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 59 |

¿Cual sería el texto en claro correspondiente al cifrado 100?

Por fuerza bruta tendríamos que probar

$$\binom{10}{1} + \binom{10}{2} + \cdots + \binom{10}{10} = \sum_{k=1}^{10} \binom{10}{k} = 2^{10} - 1$$

combinaciones. En general, si tenemos $n$ términos en la lista el número de combinaciones posibles es $2^n - 1$.

# Problema fácil de la mochila

Elección de una sucesión supercreciente $a_n > \sum_{i=1}^{n-1} a_i$, por ejemplo:

$$\{1, 3, 5, 11, 22, 45, 88, 180, 357, 712\}.$$

Si queremos descifrar el criptograma 1356 el proceso a seguir es:

| | | |
|---|---|---|
| 1356 | mayor que 712, | 1356 - 712 |
| 644 | entre 357 y 712, | 644 - 357 |
| 287 | entre 180 y 357, | 287 - 180 |
| 107 | entre 88 y 180, | 107 - 88 |
| 19 | entre 11 y 22, | 19 - 11 |
| 8 | entre 5 y 11, | 8 - 5 |
| 3 | entre 3 y 5, | 3 - 3 |
| 0 | | |

y el criptograma corresponde a la palabra NO:

| 1 | 3 | 5 | 11 | 22 | 45 | 88 | 180 | 357 | 712 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |

En este caso para obtener el resultado necesitamos 10 comparaciones. En general, se necesitan tantas como elementos tenga la sucesión. Además la solución es única.

## Transformación para que no parezca un problema sencillo

Multiplicamos la sucesión por un número $t$, tomamos módulo $M$, ($M > \sum_{i=1}^{n} a_i$ y $(t, M) = 1$) y la reordenamos.

Si tomamos $U = 100$ y $M = 1511$

$$\{1, 3, 5, 11, 22, 45, 88, 180, 357, 712\}$$
$$\rightarrow \{100, 300, 500, 1100, 689, 1478, 1245, 1379, 947, 183\}$$

y si la reordenamos:

$$\{100, 183, 300, 500, 689, 947, 1100, 1245, 1379, 1478\}.$$

Para descifrar el criptograma 6185 el proceso a seguir sería:

1. $100^{-1} \equiv 136 \pmod{1511}$.

2. $6185 \cdot 136 \equiv 1044 \pmod{1511}$.

3. Se resuelve el problema de la mochila para 1044 con la sucesión supercreciente, 1044=712+180+88+45+11+5+3.

4. Se aplica la permutación correspondiente para obtener el mensaje original,

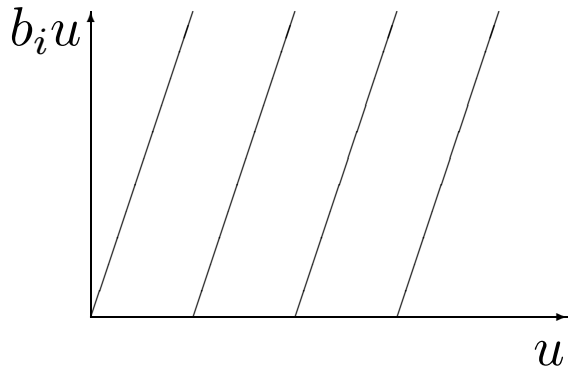
| 100 | 183 | 300 | 500 | 689 | 947 | 1100 | 1245 | 1379 | 1478 |
|---|---|---|---|---|---|---|---|---|---|
| | 712 | 3 | 5 | | | 11 | 88 | 180 | 45 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |

   que es NO.

# Cálculo de $t$ y $M$

$a_i$ supercreciente, $b_i U = a_i \mod M$, $U$ inverso de $t$ módulo $M$.

La función $b_i u \mod M$ será de la forma:




Teniendo en cuenta que $b_1 U \equiv a_1 \mod M$ y que $a_1$ es muy pequeño comparado con $M$, entonces $U$ estará cerca de un mínimo de la función que hemos dibujado. Lo mismo pasa para los primeros $b_i$.

A. Shamir. A polynomial-time algorithm for breaking the basic Merkle-Hellman cryptosistems. *IEEE Trans. Inform. Theory*, **IT-30**:699-704, 1984.

# Bibliografía


- Simmons, G.J. (Ed.) *Contemporary Cryptology. The Science of Information Integrity*. Ed. IEEE Press, 1992.

- Stinson, D.R. *Cryptography: Theory and Practice*. Ed. CRC Press, 1995.

- Schneier, B. *Applied Cryptography. Protocols, Algorithms, and Source Code in C*. Ed. John Wiley and Sons, 1996.

- Menezes, A.J., Oorschot, P.C., Vanstone, S.A. *Handbook of applied cryptography*. Ed. CRC Press, 1997

- Pastor, J., Sarasa, M.A. *Criptografia digital*. Ed. Prensas Universitarias de Zaragoza, 1998

- Yan, S.Y. *Number Theory for Computing*. Springer, 2000.

# Links

- http://www.counterpane.com/hotlist.html
- http://www.counterpane.com/biblio/
- http://escert.upc.es
- ftp://ftp.funet.fi/pub/crypt/
- http://www-ma2.upc.es/~cripto/