Criptografía de clave secreta

Fernando Martínez
fernando.martinez@upc.edu

Departament de Matemàtiques • Universitat Politècnica de Catalunya

3 de marzo de 2016

Criptografía de clave secreta o simétrica

Se utiliza la misma clave para cifrar y descifrar.
La clave ha de mantenerse en secreto.

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.**

Alleged RC4

- Algoritmo de cifrado de flujo desarrollado en 1987 por Ron Rivest para RSA Data Security.
- Es considerado secreto industrial y el algoritmo no es conocido públicamente.
- En septiembre de 1994 fue publicado de forma anónima un código (conocido como "Alleged RC4") cuyos resultados son compatibles con los obtenidos con el software proporcionado por RSA Data Security.

A grandes rasgos se puede describir de la siguiente forma:

1. A partir de una clave de 1 a 256 bytes (8 a 2048 bits) inicializa una tabla de 256 estados $S_0, S_1, \ldots, S_{255}$ (S-box).
2. Esta tabla se usa para generar una lista de bytes pseudo-aleatorios cuyo tamaño coincide con la del texto a cifrar.
3. Estos bytes se combinan mediante la función `XOR` con el texto en claro. El resultado, es el texto cifrado.

Alleged RC4

- Algoritmo de cifrado de flujo desarrollado en 1987 por Ron Rivest para RSA Data Security.
- Es considerado secreto industrial y el algoritmo no es conocido públicamente.
- En septiembre de 1994 fue publicado de forma anónima un código (conocido como "Alleged RC4") cuyos resultados son compatibles con los obtenidos con el software proporcionado por RSA Data Security.

A grandes rasgos se puede describir de la siguiente forma:

1. A partir de una clave de 1 a 256 bytes (8 a 2048 bits) inicializa una tabla de 256 estados $S_0, S_1, \ldots, S_{255}$ (S-box).
2. Esta tabla se usa para generar una lista de bytes pseudo-aleatorios cuyo tamaño coincide con la del texto a cifrar.
3. Estos bytes se combinan mediante la función `XOR` con el texto en claro. El resultado, es el texto cifrado.

Alleged RC4: Descripción detallada (I)

- Cuenta con 256 S-box, $S_0, S_1, \ldots, S_{255}$, que contienen una permutación de los enteros 0 hasta 255. Además, hace uso de dos contadores, i y j, inicializados a 0.
- Para cada byte que entra se genera un byte pseudo-aleatorio con el cual se combina mediante la función `XOR` dando lugar al byte cifrado.
  Generaración del byte pseudo-aleatorio.

$$
\begin{aligned}
&i = (i + 1) \mod 256 \\
&j = (j + S_i) \mod 256 \\
&\text{Intercambiar los valores de } S_i \text{ y } S_j \\
&t = (S_j + S_i) \mod 256 \\
&K = S_t
\end{aligned}
$$

- El byte K se combina mediante la función `XOR` con el byte del texto correspondiente.

Alleged RC4: Descripción detallada (y II)

Inicialización de los S-box.

Los S-box toman los valores $S_0 = 0, S_1 = 1, \ldots, S_{255} = 255$.

Se crea un array de 256 bytes cuyos valores son los bytes de la clave. Si la clave no es lo suficientemente grande se repiten los valores hasta completar el array $K_0, K_1, \ldots, K_{255}$.

Se define una variable auxiliar $n = 0$.

Entonces:

```
For i = 0 to 255
        n = (n + S_i + K_i)  mod 256
        Intercambiar los valores de S_i y S_n
End for
```

DES: Data Encryption Standard

Cifrado de bloques de 64 bits y clave de 56 bits.

1973 El NBS (National Bureau of Standards) solicita públicamente propuestas de sistemas criptográficos, que deben cumplir los criterios:

- Proporcionar un alto nivel de seguridad y ser eficiente.
- Residir la seguridad en la clave y no en el secreto del algoritmo.
- Ser adaptable para ser utilizado en diversas aplicaciones.
- Ser barato de implementar en dispositivos electrónicos.

1974 Segundo llamamiento: IBM presenta LUCIFER. La NSA (National Security Agency) propone una serie de cambios que son aceptados.

1975 El 17 de marzo el NBS publica los detalles del DES.

1976 El 23 de noviembre es adoptado por el gobierno USA para la transmisión y almacenamiento de información no clasificada. Se revisará cada cinco años.

1981 Diversos organismos privados lo adoptan como estándard.

1983 Se ratifica como estándard sin problemas.

DES: Historia (y II)

1987 La NSA se opone a una nueva ratificación pero, por motivos económicos, finalmente se renueva.

1992 Por falta de alternativas se renueva otra vez.

1997 El 17 de junio el DES es roto. A principios de año RSA lanza el reto y al cabo de cuatro meses es alcanzada la solución después de examinar, aproximadamente, el 25% de las claves.

1998 El 26 de febrero el DES vuelve a ser roto. Sólo han sido necesarios 39 días y se han examinado, aproximadamente, el 85% de las claves.

1998 El 17 de julio la Electronic Frontier Foundation (EFF) presenta su DES craker que puede romper el DES utilizando la fuerza bruta en un tiempo medio de 4.5 días. Su coste: 220000\$.[1]

1999 El 19 de enero la EFF rompe el DES en menos de 23 horas.

2001 Es sustituido por el AES (Advanced encryption standard), aunque se mantiene el 3DES.

[1] Actualmente se puede construir una máquina que rompa el DES en un día por unos pocos miles de euros.

DES: Descripción del algoritmo a alto nivel (I)

1. Dado un bloque x, se le aplica una permutación inicial π,

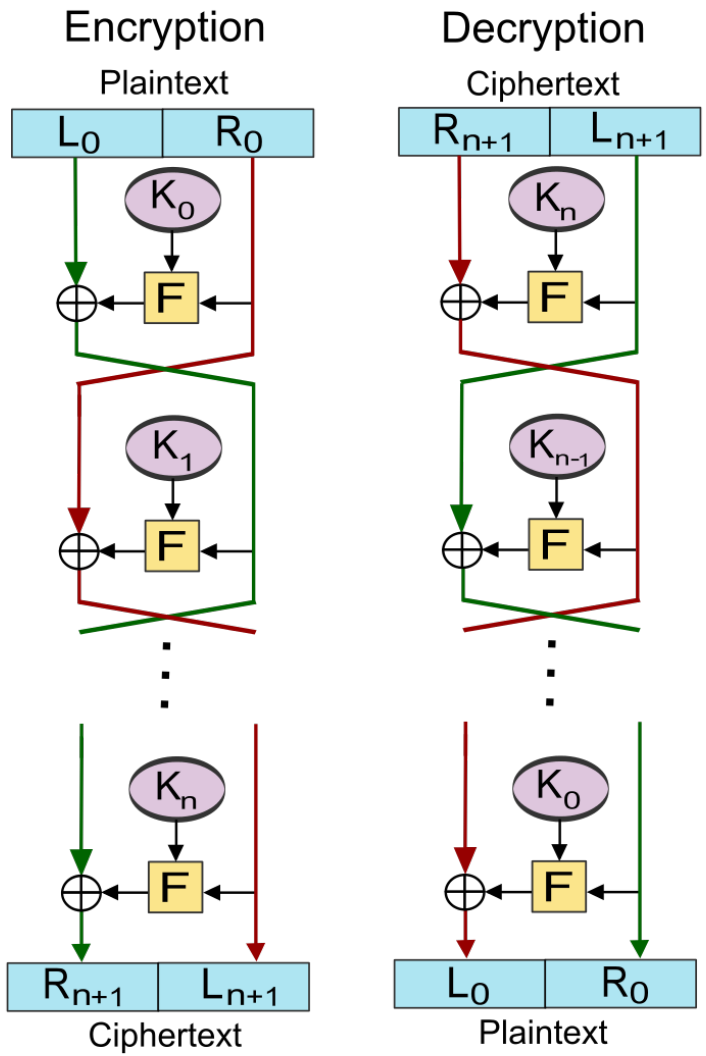
$$x_0 = \pi(x) \equiv L_0 R_0,$$

2. Se realizan 16 iteraciones del tipo *(Feistel cipher)*:

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

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

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



DES: Descripción del algoritmo a alto nivel (y II)

Si definimos:

- θ función que intercambia parte derecha e izquierda de un bloque de 64 bits,
- $\lambda_{f_i} : x \longrightarrow y$

$$L_y = L_x \oplus f(R_x, k_i),$$
$$R_y = R_x,$$

entonces el DES se puede escribir en forma más compacta como

$$\pi^{-1}\lambda_{f_{16}}\theta\lambda_{f_{15}}\theta\lambda_{f_{14}}\theta...\theta\lambda_{f_3}\theta\lambda_{f_2}\theta\lambda_{f_1}\pi,$$

y para descifrar

$$\pi^{-1}\lambda_{f_1}\theta\lambda_{f_2}\theta\lambda_{f_3}\theta...\theta\lambda_{f_{14}}\theta\lambda_{f_{15}}\theta\lambda_{f_{16}}\pi.$$

DES: Generación de subclaves

1. Dada una clave k de 56 bits (una vez eliminados los bits de paridad) se le aplica una permutación P_1,

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

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

$$\begin{aligned} C_i &= LS_i(C_{i-1}), \\ D_i &= LS_i(D_{i-1}), \\ k_i &= P_2(C_i D_i), \end{aligned}$$

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; 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 $B = E(x) \oplus k \equiv B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8$, 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. $C \equiv C_1 C_2 C_3 C_4 C_5 C_6 C_7 C_8$ es un bloque de 32 bits al que se le aplica una permutación P,
5. $f(x, k) = P(C)$.

DES: S-boxes: 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: S-boxes

1. Cada fila es una permutación de los enteros 0,..,15.
2. Ningún S-box es una función lineal o afín de las entradas.
3. Cambiando un bit en la entrada de un S-box se modifican como mínimo dos bits a la salida.
4. Para cualquier S-box y para cualquier B (B bloque de 6 bits) $S(B)$ y $S(B \oplus 001100)$ difieren como mínimo en dos bits.
5. Para cualquier S-box, B y $e, f \in \{0, 1\}$, $S(B) \neq S(B \oplus 11ef00)$.
6. Para cualquier S-box y B, si fijamos uno de los seis bits de entrada y nos centramos en un bit determinado de la salida, el número de entradas que dan lugar a un 0 en dicho bit ha de ser aproximadamente igual al número de entradas que dan un 1 en dicho bit a la salida.

DES: Permutación P

1. Los cuatro bits de salida de un S-box en la ronda i son distribuidos de forma que dos afecten a bits intermedios de S-boxes en la ronda siguiente y los otros dos afecten a bits de los extremos.
2. Los cuatro bits de salida de un S-box afectan a seis S-boxes diferentes. Además no hay dos que afecten al mismo S-box.
3. Si un bit de un S-box afecta a un bit intermedio de otro S-box, entonces un bit del segundo S-box no puede afectar a un bit intermedio del primer S-box.

DES: Seguridad

1. Tamaño de la clave: Se considera muy pequeño.
2. Las razones de la elección de las S-box no son públicas.
3. No se conoce ninguna técnica de criptoanálisis para atacarlo más eficiente que la fuerza bruta.
4. El criptoanálisis diferencial permite atacar versiones débiles del DES.
5. El DES, *muy probablemente*, no tiene estructura de grupo. Se puede aumentar la seguridad mediante aplicaciones sucesivas del DES con distintas claves.
   Dos variantes son el triple DES ($E_{k_1} D_{k_2} E_{k_3}$) y la función `crypt(3)`.

CRYPT(3)

Era la función la encargada de cifrar los passwords[2] en unix y estaba basada en el DES.

```
hobbes:1gDPklM/j4y.i:504:100:Calvin & Hobbes:/home/hobbes:/bin/bash
```

Modifica el DES introduciendo el *salt* que es una variable de 12 bits. El *salt* se aplica tras la expansión E de la función f y consiste en:

permutar los bits 1 y 25 si el primer bit del salt es 1,

⋮

permutar los bits 12 y 36 si el decimosegundo bit del salt es 1.

[2] Ahora cada entrada en el fichero `/etc/shadow` puede ser de la forma idsalt$encrypted, id identifica el algoritmo usado: 1→MD5, 5→SHA256, 6→SHA512

AES: Advanced Encryption Standard (I)

El 12 de septiembre de 1997 en departamento de comercio del National Institute of Standars and Technology (NIST), antiguo NBS, hace un llamamiento público para la presentación de algoritmos candidatos al Advanced Encryption Standard (AES).

Requisitos mínimos

1. El algoritmo debe ser de clave secreta (simétrico).
2. El algoritmo debe ser un algoritmo de bloque.
3. El algoritmo debe ser capaz de soportar las combinaciones clave-bloque de los tamaños 128-128, 192-128 y 256-128.

AES: Advanced Encryption Standard (y II)

Criterios de evaluación

1. Seguridad: Es el factor más importante a la hora de evaluar los candidatos.
2. Coste:
   1. El algoritmo debe ser accesible a todo el mundo y de libre distribución.
   2. El algoritmo debe ser computacionalmente eficiente tanto en *hardware* como en *software*.
   3. El algoritmo debe utilizar la menor memoria posible tanto en *hardware* como en *software*.
3. Características de implementación del algoritmo:
   1. El algoritmo debe ser fácilmente implementable en distintas plataformas tanto en *hardware* como en *software*.
   2. El algoritmo debe acomodarse a diferentes combinaciones clave-bloque además de las mínimas requeridas.
   3. El algoritmo debe ser de diseño simple.

Candidatos al AES

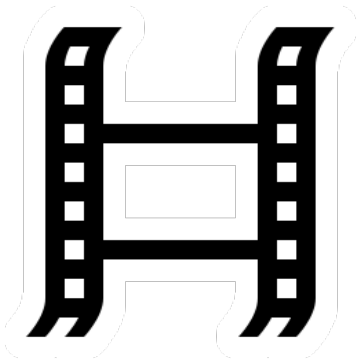
1. **CAST-256**, Entust Technologies, Inc. (C. Adams).
2. **CRYPTON**, Future Systems, Inc. (Chae Hoon Lim).
3. **DEAL**, L. Knudsen, R. Outerbridge.
4. **DFC**, CNRS-Ecole Normale Superiere (S. Vaudenay).
5. **E2**, NTT *Nippon Telegraph and Telephone Corporation* (M. Kanda).
6. **FROG**, TecApro International S.A. (D. Georgoudis, Leroux, Chaves).
7. **HPC**, R. Schoeppel.
8. **LOKI97**, L. Brown, J. Pieprzyk, J.Seberry.
9. ~~**MAGENTA**~~, Deutshe Telekom AG (K. Huber).
10. **MARS***, IBM (N. Zunic).
11. **RC6***, RSA Laboratories (Rivest, M. Robshaw, Sidney, Yin).
12. **RIJNDAEL***, J. Daemen[3], V. Rijmen.
13. **SAFER+**, Cylink Corporation (L. Chen).
14. **SERPENT***, R. Anderson, E. Biham, L. Knudsen.
15. **TWOFISH***, B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, N. Ferguson.

[3] También participó en el diseño de SHA3.

RIJNDAEL-AES: Advanced Encryption Standard [4]

Algoritmo simétrico de bloque de 128 *(192, 256)* bits y clave de 128, 192 o 256 bits.

RIJNDAEL-AES



[4] http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

RIJNDAEL-AES: Advanced Encryption Standard

Opera con bytes, GF(2^8) (polinomio irreducible $x^8 + x^4 + x^3 + x + 1$), y con palabras de 4 bytes, polinomios con coeficientes en GF(2^8).

GF(2^8) generado por $03 = x + 1$

03	05	0F	11	33	55	FF	1A	2E	72	96	A1	F8	13	35	5F
E1	38	48	D8	73	95	A4	F7	02	06	0A	1E	22	66	AA	E5
34	5C	E4	37	59	EB	26	6A	BE	D9	70	90	AB	E6	31	53
F5	04	0C	14	3C	44	CC	4F	D1	68	B8	D3	6E	B2	CD	4C
D4	67	A9	E0	3B	4D	D7	62	A6	F1	08	18	28	78	88	83
9E	B9	D0	6B	BD	DC	7F	81	98	B3	CE	49	DB	76	9A	B5
C4	57	F9	10	30	50	F0	0B	1D	27	69	BB	D6	61	A3	FE
19	2B	7D	87	92	AD	EC	2F	71	93	AE	E9	20	60	A0	FB
16	3A	4E	D2	6D	B7	C2	5D	E7	32	56	FA	15	3F	41	C3
5E	E2	3D	47	C9	40	C0	5B	ED	2C	74	9C	BF	DA	75	9F
BA	D5	64	AC	EF	2A	7E	82	9D	BC	DF	7A	8E	89	80	9B
B6	C1	58	E8	23	65	AF	EA	25	6F	B1	C8	43	C5	54	FC
1F	21	63	A5	F4	07	09	1B	2D	77	99	B0	CB	46	CA	45
CF	4A	DE	79	8B	86	91	A8	E3	3E	42	C6	51	F3	0E	12
36	5A	EE	29	7B	8D	8C	8F	8A	85	94	A7	F2	0D	17	39
4B	DD	7C	84	97	A2	FD	1C	24	6C	B4	C7	52	F6	01	

RIJNDAEL-AES: Advanced Encryption Standard

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

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

RIJNDAEL-AES: Descripción del algoritmo de cifrado a alto nivel

1. `AddRoundKey(State, RoundKey`$_0$`)`
2. `Round(State, RoundKey`$_i$`)`, $i = 1, \ldots, N_r - 1$:
   1. `ByteSub(State)`
   2. `ShiftRow(State)`
   3. `MixColumn(State)`
   4. `AddRoundKey(State, RoundKey`$_i$`)`
3. `FinalRound(State, RoundKey`$_{N_r}$`)`:
   1. `ByteSub(State)`
   2. `ShiftRow(State)`
   3. `AddRoundKey(State, RoundKey`$_{N_r}$`)`

RIJNDAEL-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):

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

RIJNDAEL-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

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

$\Longrightarrow$

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

RIJNDAEL-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 $b(x) = c(x) \otimes a(x)$,

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

RIJNDAEL-AES: `AddRoundKey`

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

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

RIJNDAEL-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 de los elementos de W se definen recursivamente utilizando la función `SubByte`, desplazamientos cíclicos y $\oplus$.
- Usa la función `RotByte` que devuelve una palabra cuyos bytes se han desplazado cíclicamente una posición.
- Utiliza unas constantes cuyos valores son

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

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

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

RIJNDAEL-AES: Generación de subclaves (y II)

$N_k \leq 6$

```
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;
  }
}
```

$N_k > 6$

```
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];
   else if (i % Nk == 4)
       temp = SubByte(temp);
   W[i] = W[i - Nk] ^ temp;
 }
}
```

RIJNDAEL-AES: Un algoritmo de descifrado (I)

1. `AddRoundKey(State, InvRoundKey`$_{N_r}$`)`
2. `Round(State, InvRoundKey`$_i$`)`, $i = N_r - 1, \ldots, 1$:
   1. `InvByteSub(State)`
   2. `InvShiftRow(State)`
   3. `InvMixColumn(State)`
   4. `AddRoundKey(State, InvRoundKey`$_i$`)`
3. `FinalRound(State, InvRoundKey`$_{N_0}$`)`:
   1. `InvByteSub(State)`
   2. `InvShiftRow(State)`
   3. `AddRoundKey(State, InvRoundKey`$_{N_0}$`)`

RIJNDAEL-AES: Un algoritmo de descifrado (II)

- Las funciones `InvByteSub`, `InvShiftRow`, `InvMixColumn` son las inversas de `ByteSub`, `ShiftRow`, `MixColumn` respectivamente.
- Las subclaves `InvRoundKey` vienen dadas por:
  - $\texttt{InvRoundKey}_0 = \texttt{RoundKey}_0$
  - $\texttt{InvRoundKey}_i = \texttt{InvMixColumn}(\texttt{RoundKey}_i)$, $i = 1, \ldots, N_r - 1$
  - $\texttt{InvRoundKey}_{N_r} = \texttt{RoundKey}_{N_r}$

RIJNDAEL-AES: Un algoritmo de descifrado (y III)

- **InvByteSub**. Si $\mathbf{y} = \mathbf{Ax} + \mathbf{B}$, invirtiendo $\mathbf{x} = \mathbf{A}^{-1}(\mathbf{y} - \mathbf{B})$. Después se ha de sustituir cada byte diferente de 0x00 por su inverso en $GF(2^8)$.

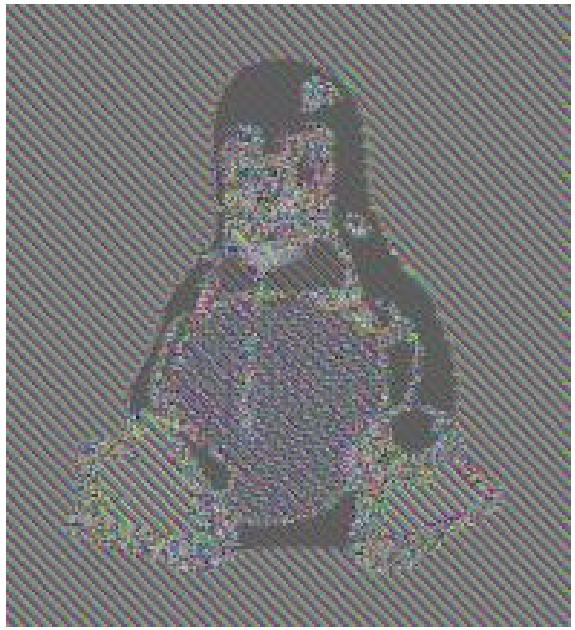
$$\mathbf{A}^{-1} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

- **InvShiftRow**. Se han de rotar las filas el mismo número de posiciones pero en sentido contrario.
- **InvMixColumn**. Se multiplica por el polinomio inverso del anterior $d(x) = \texttt{0x0B}\, x^3 + \texttt{0x0D}\, x^2 + \texttt{0x09}\, x + \texttt{0x0E}$. Matricialmente:

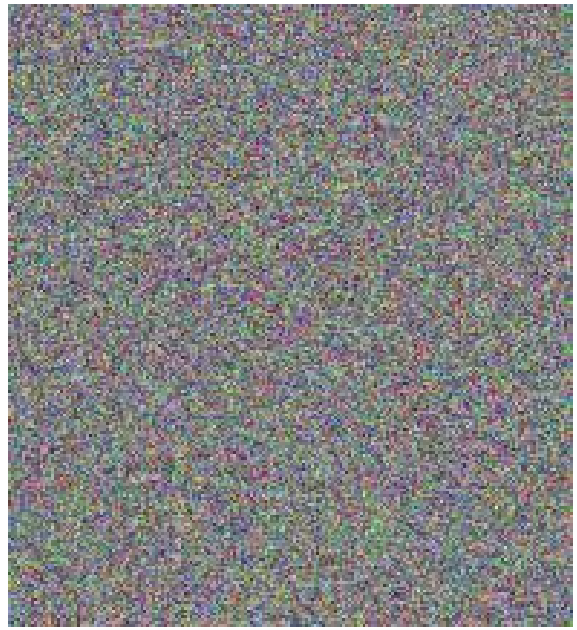
$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} \texttt{0x0E} & \texttt{0x0B} & \texttt{0x0D} & \texttt{0x09} \\ \texttt{0x09} & \texttt{0x0E} & \texttt{0x0B} & \texttt{0x0D} \\ \texttt{0x0D} & \texttt{0x09} & \texttt{0x0E} & \texttt{0x0B} \\ \texttt{0x0B} & \texttt{0x0D} & \texttt{0x09} & \texttt{0x0E} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}.$$

Modos de operación

(a) Original[1]



(b) ECB



(c) Otro

[1] Tux the Penguin. Created in 1996 by Larry Ewing with The GIMP.

Modos de operación[6] (I)

ECB **Electronic CodeBook**

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

Es paralelizable.

CBC **Cipher Block Chaining** Se inicializa c_0 aleatorio,

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

1. c_0 puede ser público. Variando c_0 podemos cifrar el mismo mensaje con la misma clave obteniendo criptogramas diferentes.
2. Bloques iguales van a criptogramas diferentes.
3. Un error en la transmisión afecta al descifrar a dos bloques del mensaje.
4. Puede usarse como MAC[5] (message authentification code), variar un bit en un bloque del mensaje afecta al resto de los bloques cifrados.

[5] `http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf`

[6] NIST Special Publication 800-38A: Recommendation for Block Cipher Modes of Operation http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

Modos de operación (II)

CFB **Cipher FeedBack** Se inicializa c_0 aleatorio,

$$c_i = m_i \oplus E_k(c_{i-1});$$

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

Además de las propiedades del modo CBC:

1. Se utiliza la misma función para cifrar y descifrar.
2. No es necesario *padding*.

OFB **Output FeedBack** Se inicializa s_0 aleatorio,

$$c_i = m_i \oplus E_k(s_{i-1}), \qquad s_i = E_k(s_{i-1}));$$

$$m_i = c_i \oplus E_k(s_{i-1}).$$

1. Un error en la transmisión afecta al descifrar a un bloque del mensaje.
2. Se utiliza la misma función para cifrar y descifrar.
3. Se puede usar como cifrado de flujo.

Modos de operación (III)

CTR **Counter Mode** Dados $T_1, \ldots, T_n$

$$c_i = m_i \oplus E_k(T_i);$$

$$m_i = c_i \oplus E_k(T_i).$$

1. Se utiliza la misma función para cifrar y descifrar.
2. No es necesario *padding*.
3. Los T_i deben ser diferentes para cada bloque y mensaje.
4. Los T_i se pueden generar cómo se quieran (sin repeticiones).
5. Se pueden precalcular los valores $E_k(T_i)$.
6. Se puede usar como cifrado de flujo.
7. Se pueden descifrar bloques del mensaje de forma independiente.
8. Es paralelizable.

Modos de operación (y IV)

GMAC **Galois Message Authentication Code**[7]

Sea $y_0 = 0 \dots 0$ un bloque de 128 bits y H la clave de autentificación.

$$y_i = (m_i \oplus y_{i-1}) \bullet H$$

El símbolo $\bullet$ representa la multiplicación en $\mathrm{GF}(2^{128})$ (polinomio irreducible $x^{128} + x^7 + x^2 + x + 1$).

[7] Asociado al CTR como método de autentificación.