

Criptosistemas de clave compartida: cifrado en bloque

Jordi Herrera Joancomartí

PID_00214913
Módulo 4

Índice


Introducción	5
Objetivos	6
1. Estructura del cifrado en bloque	7
1.1. Los principios de confusión y difusión de Shannon	7
1.2. Características comunes	7
1.3. Modos de cifrado en bloque	8
2. Criptosistemas de cifrado en bloque	13
2.1. El estándar DES	13
2.1.1. Descripción del funcionamiento	13
2.1.2. Detalle de una iteración	14
2.1.3. Generación de subclaves	17
2.1.4. Descifrado	19
2.1.5. Particularidades del criptosistema	19
2.2. El criptosistema IDEA	20
2.2.1. Descripción del funcionamiento	20
2.2.2. Detalle de una iteración	21
2.2.3. Generación de subclaves	21
2.2.4. Particularidades del criptosistema	22
2.3. Propuestas de AES	23
2.4. El criptosistema Rijndael	25
2.4.1. Descripción del funcionamiento	25
2.4.2. Detalle de una iteración	28
2.4.3. Generación de subclaves	33
2.4.4. Descifrado	35
3. Ataques al cifrado en bloque	36
3.1. Ataque del punto intermedio	36
3.2. Cifrado triple	37
3.3. Criptoanálisis diferencial	37
4. Gestión de claves	38
Resumen	39
Actividades	41
Ejercicios de autoevaluación	41
Solucionario	42

Glosario 44


Bibliografía 44

Introducción

Una alternativa al cifrado en flujo es lo que se denomina **cifrado en bloque**. Este tipo de cifrado se incluye también dentro del grupo de criptosistemas de clave compartida, ya que el emisor y el receptor comparten la misma clave que se emplea a la vez para cifrar y descifrar. La diferencia básica entre el cifrado en flujo y el cifrado en bloque es el uso de memoria en los algoritmos de cifrado. Se podría hacer una analogía entre la criptografía y la codificación, en la cual el cifrado en flujo sería el equivalente a los códigos de convolución, mientras que el cifrado en bloque corresponde a la codificación en bloque.



El cifrado de flujo se estudia en el módulo "Cifras de clave compartida: cifras de flujo" de esta asignatura.

El cifrado en flujo utiliza una clave diferente para cada bit de información, que depende tanto del estado inicial del generador como de su estado en el momento de cifrar el bit. De este modo, dos bits iguales pueden quedar cifrados de maneras diferentes según el estado en que se halle el generador. Eso no sucede en el cifrado en bloque, porque las cifras en bloque actúan sin memoria y el texto cifrado sólo puede depender del texto en claro y de la clave. En el cifrado en bloque, dos textos en claro iguales se cifran siempre del mismo modo cuando se utiliza la misma clave. 

Este hecho ha de tratarse con cuidado, porque los sistemas de cifrado que resultan son bastante vulnerables y, si no se corrigen, se podrían insertar o borrar bloques de texto cifrado sin que se detectara. Además, el hecho de que dos bloques de texto en claro queden cifrados del mismo modo puede dar pistas a un posible criptoanálisis de tipo estadístico.

Los cifradores en bloque se emplean bastante, ya que consiguen velocidades de cifrado aceptables. En concreto, el cifrador de bloque más utilizado es el Data Encryption Standard (DES), que se ha establecido como estándar. Actualmente, sin embargo, en muchos ámbitos se suele elegir el Advanced Encryption Standard (AES), ya que se considera que el DES (que fue creado a finales de los setenta) no ofrece las mismas garantías de seguridad que presentaba cuando surgió, teniendo en cuenta la rapidez con que se ha desarrollado la tecnología informática estos últimos años.

Objetivos

En los materiales didácticos facilitados en este módulo hallaréis las herramientas necesarias para alcanzar los objetivos siguientes:

1. Comprender el esquema general de las cifras en bloque y sus características principales.
2. Entender los principios de *confusión* y *difusión* y los diferentes modos de cifrado en bloque.
3. Conocer el funcionamiento de los criptosistemas en bloque más relevantes.
4. Obtener unos conocimientos básicos en criptoanálisis de sistemas de cifrado en bloque.

1. Estructura del cifrado en bloque

1.1. Los principios de confusión y difusión de Shannon

Para que un criptosistema sea seguro, la información que proporcione el texto cifrado debe ser la mínima posible. Ello implica, en particular, evitar mantener en el texto cifrado las propiedades estadísticas del texto en claro.

En un trabajo de formalización de la teoría de la información, Shannon propuso las dos técnicas que tienen que seguir los criptosistemas de cifrado en bloque si pretenden evitar eventuales ataques basados en métodos estadísticos.

Las dos técnicas propuestas por Shannon en su trabajo son:

- La **confusión**, que incluye sustituciones a fin de que la relación entre la clave y el texto cifrado sea lo más complicada posible.
- La **difusión**, que utiliza transformaciones que disipan las propiedades estadísticas del texto en claro entre el texto cifrado.

1.2. Características comunes

Desde un punto de vista general, todos los esquemas de cifrado en bloque tienen la misma estructura básica de funcionamiento. A partir de un bloque de texto en claro B de longitud fijada y de una clave K , ejecutan determinadas operaciones más o menos complicadas hasta obtener el texto cifrado correspondiente Y , tal que:

$$Y = E_k(B).$$

Además, dichas operaciones han de permitir obtener el mismo texto en claro a partir del texto cifrado Y y la clave K , si se lleva a cabo el proceso de descifrado:

$$B = D_k(Y).$$

Las operaciones que se ejecutan sobre el bloque que se va a cifrar sólo combinan los propios elementos del bloque y la clave, pues estos métodos de cifrado

no incorporan memoria, de modo que si la clave K es la misma, dos bloques idénticos generarán textos cifrados idénticos.

La estructura básica de los criptosistemas en bloque modernos consta normalmente de dos transformaciones, una de entrada y otra de salida, entre las cuales se efectúa un número determinado de iteraciones de una función f no lineal, que combina los elementos que forman parte del bloque de texto en claro con los elementos que forman parte de la clave. De hecho, en general, la clave K se utiliza para generar una serie de subclaves K_i a partir de una cierta función f_K bastante complicada, y son estas subclaves las que actúan en cada iteración.

El modo básico de funcionamiento de un criptosistema en bloque se conoce con la sigla ECB. Dado que la mayoría de veces la longitud del texto M a cifrar será superior a la longitud del bloque con el que el criptosistema va a trabajar, el texto M se parte en varios bloques M_1, M_2, \dots , y cada bloque se cifra con una misma clave K para obtener el texto cifrado resultante: $Y = Y_1 Y_2 \dots$

El nombre de ECB no es casual; alude al hecho de que, una vez fijada una clave K , a cada bloque de texto en claro corresponde un bloque de texto cifrado concreto, como si para cada clave K tuviéramos un diccionario y hubiera que ir buscando qué palabra (bloque) de texto cifrado corresponde a cada palabra del texto en claro.

Aunque el modo ECB parece el método más natural, presenta bastantes inconvenientes desde una perspectiva de seguridad: el hecho de que el cifrado de dos bloques sea totalmente independiente lo vuelve vulnerable a ciertos ataques. Por ejemplo:

- a) Un atacante podría borrar bloques de texto cifrado sin que ello afectara al proceso de descifrado del resto, y el receptor no lo podría detectar.
- b) El atacante podría insertar bloques de texto cifrado.
- c) El que los mismos bloques queden cifrados siempre del mismo modo puede facilitar los ataques de tipo estadístico para obtener la clave K .

1.3. Modos de cifrado en bloque

Con lo dicho hasta el momento, vemos que los cifrados en bloque pueden ser útiles para cifrar informaciones cortas, como identificadores, contraseñas, claves, etc.; en definitiva, mensajes en que la longitud del texto que se va a cifrar

Ejemplos de criptosistemas en bloque...

... son las cifras de sustitución, ya sea ésta simple u homofónica, y las cifras de transposición.

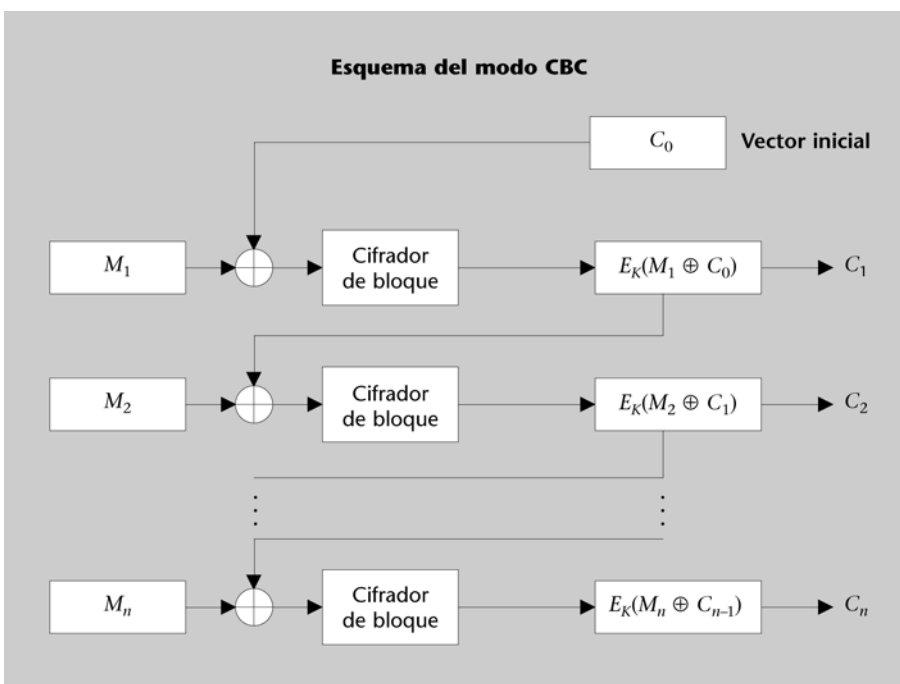
ECB es la sigla de *Electronic Code Book*.

no sobrepase la longitud del bloque. En cambio, se desaconseja totalmente la utilización del cifrado en bloque por el modo ECB en mensajes largos y, en especial, en textos que repiten determinados patrones. Si deseamos usar los cifrados en bloque en este tipo de textos habrá que aplicar alguno de los modos de cifrado que explicamos a continuación. **!**

Modo CBC

El modo CBC consiste en encadenar los bloques de modo que se cree una dependencia del cifrado de cada bloque con respecto al anterior inmediato, tal como muestra la figura siguiente:

CBC es la sigla de *Cipher Block Chaining*.



Supongamos un cifrado en bloque con una clave K , una función de cifrado E y una de descifrado D . Si M_1, \dots, M_n son los bloques de texto en claro por cifrar con el sistema CBC, el cifrado del bloque M_i se lleva a cabo de la manera siguiente:

$$C_i = E_K(M_i \oplus C_{i-1}).$$

Asimismo, para su descifrado, también hay que partir del texto cifrado anterior y efectuar la operación siguiente:

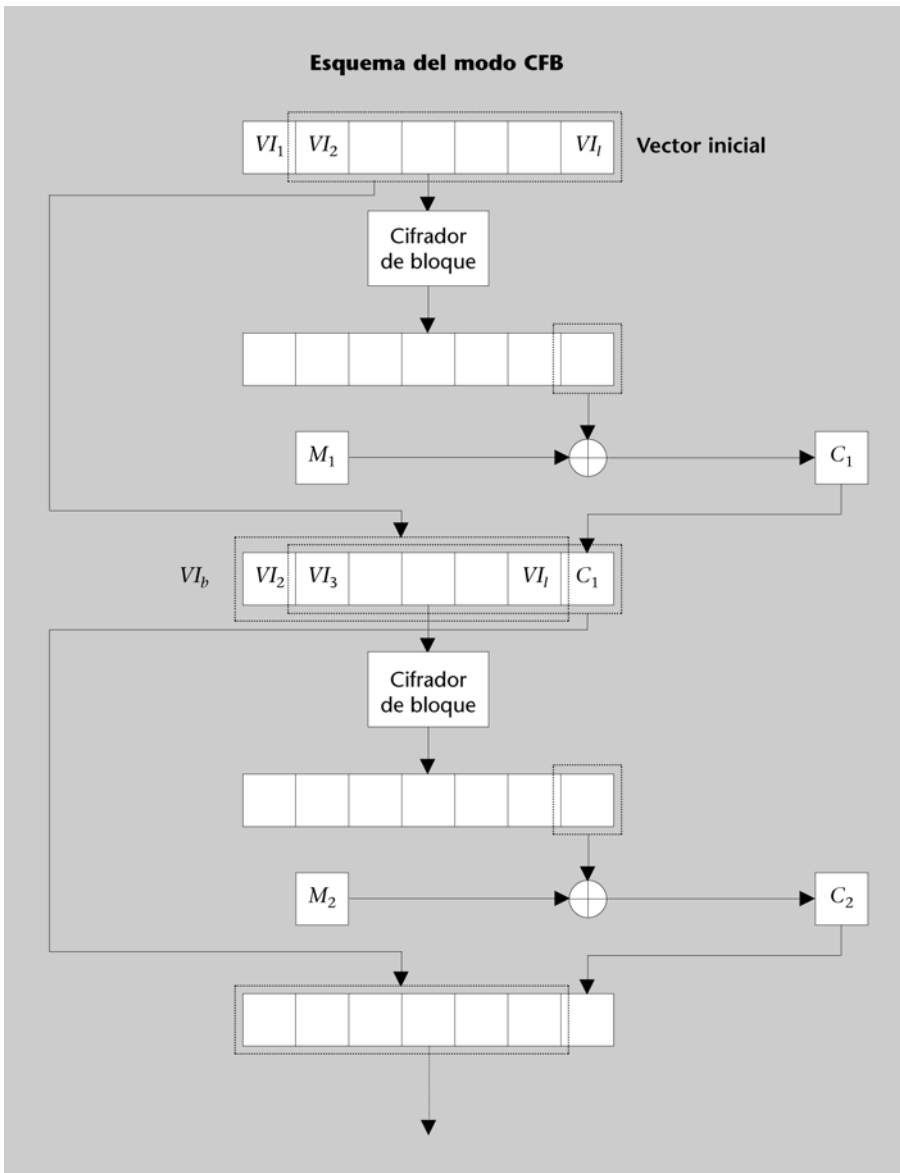
$$\begin{aligned} D_K(C_i) \oplus C_{i-1} &= D_K(E_K(M_i \oplus C_{i-1})) \oplus C_{i-1} = \\ &= (M_i \oplus C_{i-1}) \oplus C_{i-1} = M_i \end{aligned}$$

Para cifrar el primer bloque introduciremos en el cifrado un bloque inicial aleatorio C_0 , no necesariamente secreto. De este modo, aun utilizando la misma clave K , al ir cambiando el vector inicial C_0 –que además puede incorporar una marca temporal–, para un mismo texto en claro podemos obtener dos cifrados diferentes.

Modo CFB

El modo de cifrado CFB utiliza indirectamente el cifrador en bloque. Por ello, no es necesario que la longitud de los bloques que se van a cifrar sea la misma que la de los bloques del criptosistema con que actúa, sino que puede ser más corta. El esquema general de funcionamiento de este método se muestra en la figura siguiente:

CFB es el acrónimo de *Cipher Feedback*.



Dado el mensaje de texto en claro $M = M_1M_2\dots$ donde M_1, M_2, \dots son bloques de n bits de longitud, si consideramos el vector inicial VI como una concate-

nación de l bloques de n bits de longitud cada uno, $VI = VI_1 VI_2 \dots VI_l$, el criptosistema en bloque permite calcular el cifrado del vector VI , $E(VI)$. El resultado tendrá la misma longitud que VI y, por lo tanto, podrá ser descompuesto del mismo modo:

$$E(VI) = E(VI)_1 E(VI)_2 \dots E(VI)_l$$

Así, obtenemos el primer bloque cifrado, C_1 , de longitud n , sumando bit a bit el primer bloque de texto en claro, M_1 con el último bloque, $E(VI)_l$:

$$C_1 = M_1 \oplus E(VI)_l;$$

Para cifrar el segundo bloque, M_2 , repetimos el proceso, pero esta vez tomaremos como vector inicial el vector formado por los fragmentos siguientes:

$$VI_b = VI_2 VI_3 \dots VI_l C_1,$$

Es decir, desplazamos los bloques de n bits hacia la izquierda para añadir el bloque C_1 y descartar el VI_1 y obtenemos el segundo bloque de texto cifrado a partir de la operación siguiente:

$$C_2 = E(VI_b)_l \oplus M_2.$$

El proceso se repite para todos los bloques de texto que se vayan a cifrar: para cada bloque se desplazan los bloques del vector inicial anterior, $VI_b \dots$ hacia la izquierda, se añade el último bloque de texto cifrado obtenido y se ejecutan las operaciones descritas.

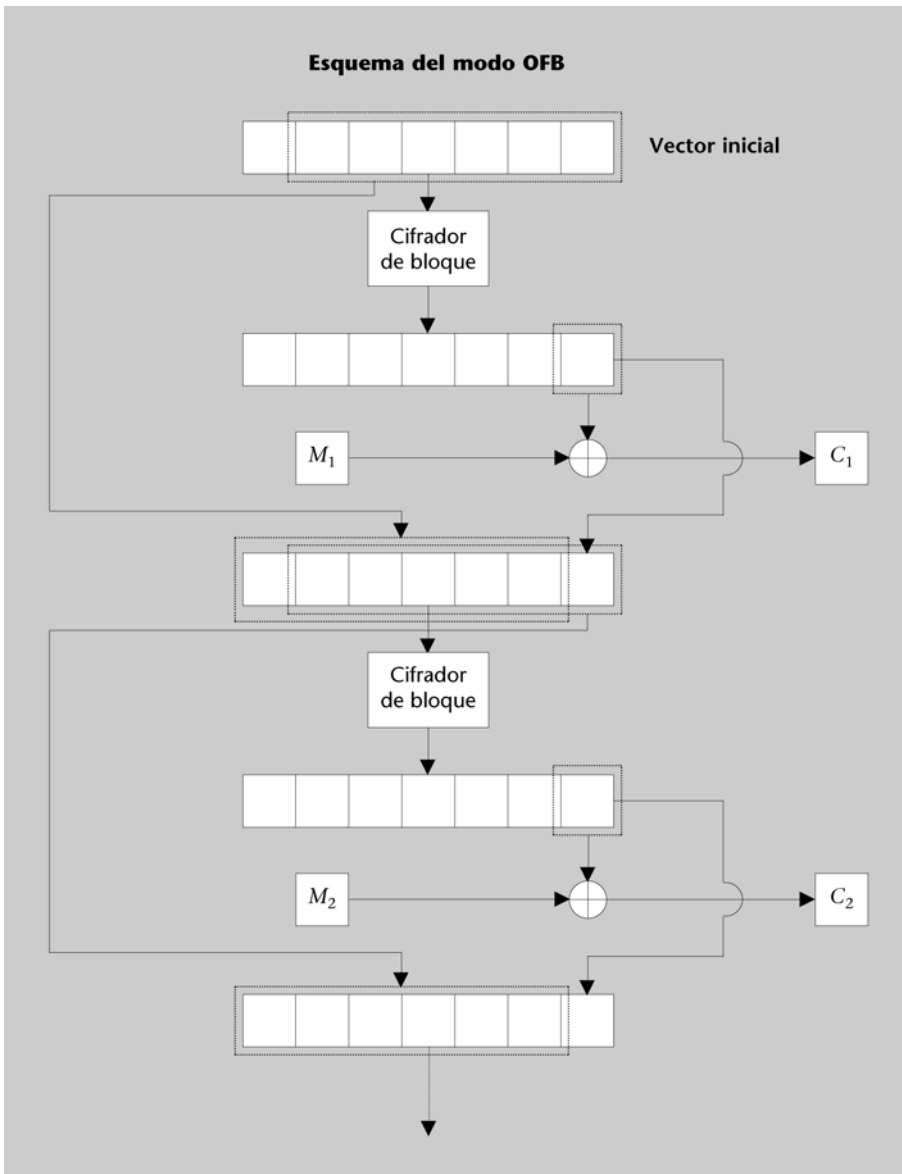
Modo OFB

El modo de cifrado OFB utiliza el criptosistema en bloque como generador pseudoaleatorio. Es un sistema muy parecido al anterior; la única diferencia que presenta es que el vector inicial se realimenta directamente con el resultado del cifrado en bloque antes de hacer la suma bit a bit con el bloque de texto en claro, como se puede ver en la figura de la página siguiente.

Al actuar el cifrador de bloque como generador pseudoaleatorio, los criptosistemas en bloque que utilicemos con el modo OFB deben cumplir los requisitos exigidos para los generadores pseudoaleatorios, tanto con respecto a la impredecibilidad de la secuencia resultante como con respecto a su complejidad lineal.

OFB es el acrónimo de
Output Feedback.

Los generadores pseudoaleatorios se tratan en el apartado 1 del módulo "Criptosistemas de clave compartida: cifrado en flujo" de esta asignatura.



2. Criptosistemas de cifrado en bloque

2.1. El estándar DES

En 1977, el National Bureau of Standards (NBS), una sección del Departamento de Defensa de Estados Unidos, publicó un criptosistema estándar creado con la finalidad de proteger cualquier tipo de datos: el DES. En el desarrollo de este sistema participaron la empresa IBM y la National Security Agency (NSA).

DES es la sigla de
Data Encryption Standard.

DES es un criptosistema de cifrado en bloque que cifra bloques de datos de 64 bits de longitud con una clave de 56 bits. Este criptosistema consigue cumplir tanto el principio de confusión como el de difusión gracias a las acciones de las cajas S que contiene.

La descripción de las cajas S se presenta en el subapartado 2.1.2 de este módulo.

2.1.1. Descripción del funcionamiento

El funcionamiento del algoritmo DES de cifrado y descifrado queda determinado en la figura siguiente. Se suministra al algoritmo un bloque de entrada M sobre el cual se aplica una permutación inicial, σ , de donde se obtiene $T_0 = \sigma(M)$. Tras 16 iteraciones de la función f , que explicaremos a continuación, el resultado se invierte por medio de la permutación de salida σ^{-1} . Las permutaciones σ y σ^{-1} se recogen en dos tablas:

La interpretación de estas tablas...

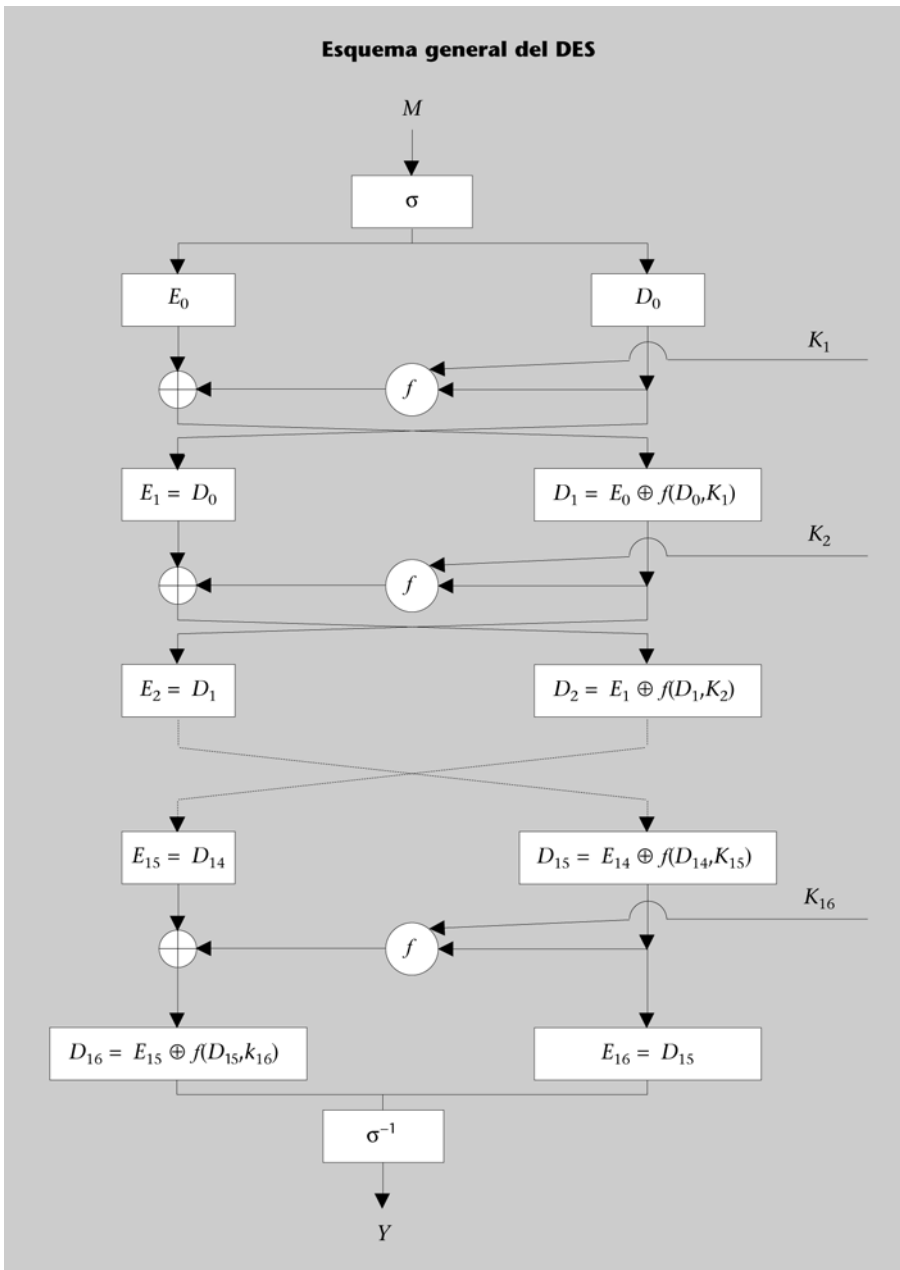
... es muy sencilla. Por ejemplo, la tabla correspondiente a la permutación σ nos indica que el primer bit de salida corresponde al bit 58 de entrada, el segundo, al 50 de entrada, en el tercero al 42, etc.

- Tabla de permutaciones σ :

Tabla de permutaciones σ							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

- Tabla de permutaciones σ^{-1} :

Tabla de permutaciones σ^{-1}							
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25



2.1.2. Detalle de una iteración


Entre las dos permutaciones, σ y σ^{-1} , el algoritmo efectúa 16 iteraciones de la función f que combinan sustituciones y transposiciones.

Sea T_i el resultado de la i -ésima iteración, de 64 bits de longitud. Podemos dividir T_i en dos mitades iguales de 32 bits: $T_i = E_i D_i$, de modo que la parte izquierda sea $E_i = t_1 \dots t_{32}$, y la derecha $D_i = t_{33} \dots t_{64}$. Entonces, para cada iteración se verificarán las condiciones siguientes:

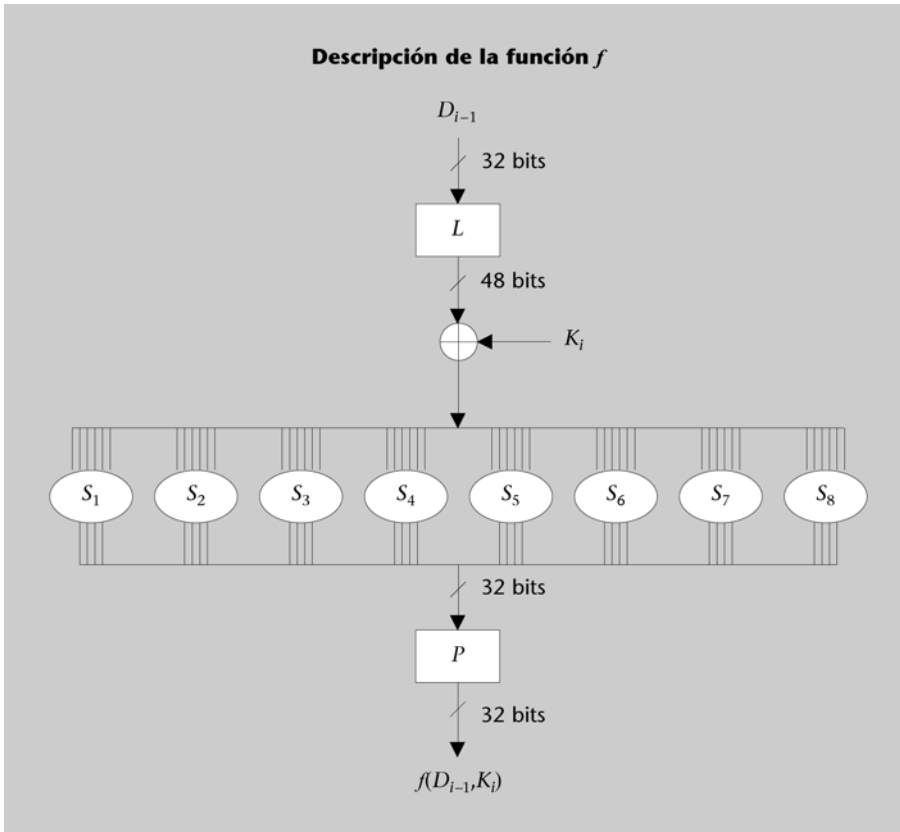
- $E_i = D_{i-1}$,
- $D_i = E_{i-1} \oplus f(D_{i-1}, K_i)$,

donde K_i es una clave de 48 bits de longitud que describiremos más adelante.

El cálculo de K_i se desarrolla en el subapartado 2.1.3 de este módulo.

Observemos que en el esquema general del DES, para obtener la salida final Y no se trata de cambiar de lado las partes en la última iteración, sino que hay que obtener la inversa de la permutación inicial, σ^{-1} . Así, el mismo algoritmo sirve al mismo tiempo para cifrar y para descifrar. 

La figura siguiente ayuda a entender el valor de la función $f(D_{i-1}, K_i)$ obtenido:



En primer lugar se transforma D_{i-1} para que tenga una longitud de 48 bits*. Para extender D_{i-1} se han repetido algunos de sus bits por medio de la transformación de expansión, L , que queda determinada en la tabla siguiente:

* Recordad que D_{i-1} es la mitad derecha de T_{i-1} , y sólo tiene 32 bits.

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

En la tabla...
 ... el primer bit de salida corresponde al bit 32 de entrada; el segundo bit de salida, al 1 de entrada, etc.

A continuación, siguiendo el esquema del DES, sumamos bit a bit la mitad expandida $L(D_{i-1})$ con la clave K_i y descomponemos el resultado de esta operación en ocho bloques de 6 bits cada uno, es decir:

$$L(D_{i-1}) \oplus K_i = B_1 B_2 \dots B_8$$

Cada bloque B_j se usa como entrada de lo que se denomina una caja S . Cada caja S_j recibe su bloque correspondiente de 6 bits $B_j = b_1b_2b_3b_4b_5b_6$ y devuelve un bloque de 4 bits de longitud, de acuerdo con la tabla siguiente:

		Columna															
Caja S	Fila	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_1	0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

El criterio de asignación de una fila y una columna de una caja a B_j es el siguiente: el índice j fija la caja S_j , el entero correspondiente a b_1b_6 selecciona la fila y el entero que corresponde a $b_2b_3b_4b_5$ determina la columna.

Después se toman los bloques resultantes de las cajas S y se concatenan hasta obtener un bloque de 32 bits de longitud. Finalmente, se aplica sobre este último

Ejemplo de transformación en bloque

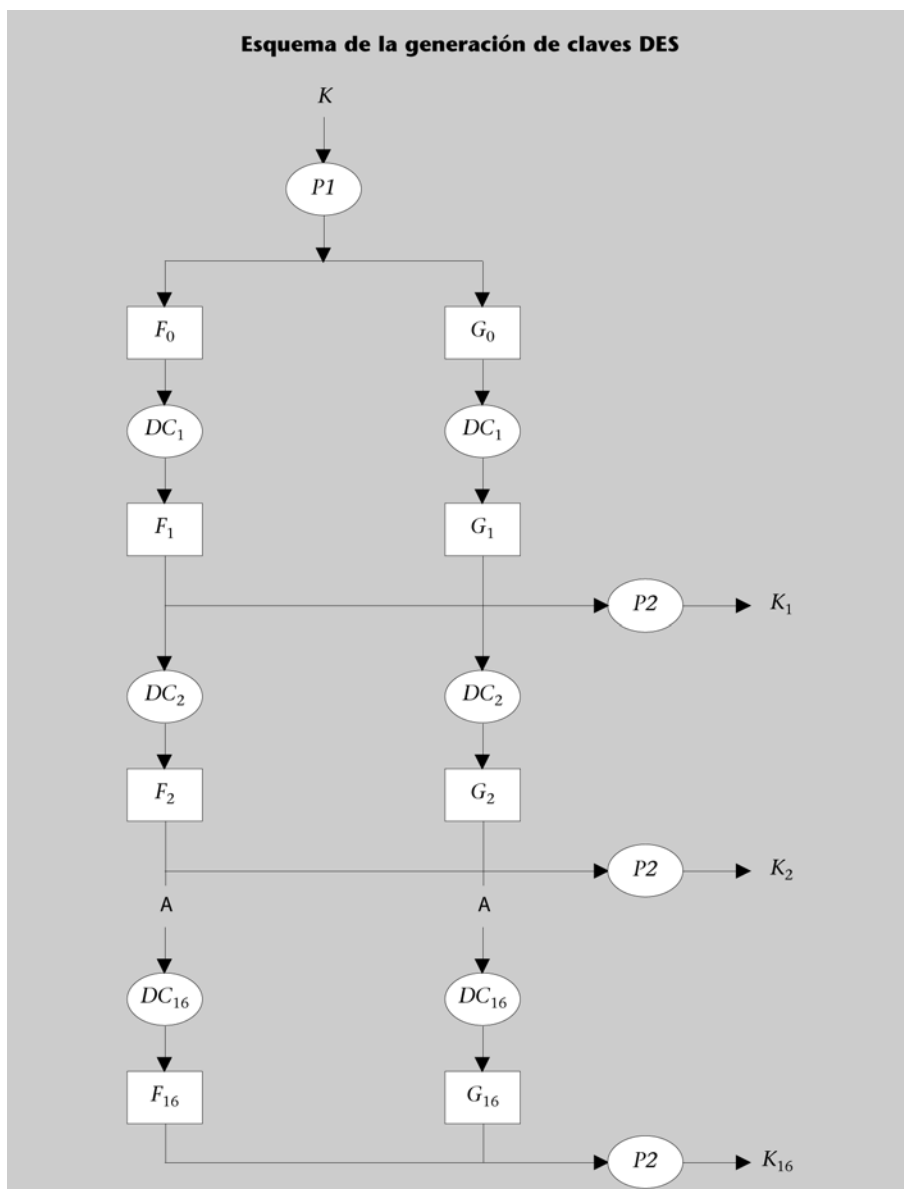
Para transformar un bloque dado, $B_1 = 011001$, S_1 devolverá el valor de la fila 1 (ya que $01_2 = 1$) y la columna 12 (ya que $1100_2 = 12$). Si nos fijamos en la tabla de las cajas S hay un 9, que en binario es 1001.

bloque la permutación P definida en la siguiente tabla con el fin de obtener el valor $f(D_{i-1}, K_i)$.

Tabla de permutaciones P			
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

2.1.3. Generación de subclaves

Al iniciar la descripción del criptosistema DES hemos señalado que operaba con una clave K de 56 bits de longitud. En cambio, el algoritmo descrito utiliza 16 subclaves K_i de 48 bits. Veamos cómo obtener estas subclaves a partir de la principal. El algoritmo de generación de subclaves se ilustra en la figura siguiente:




A la clave inicial K de 56 bits se han añadido 8 bits más en las posiciones 8, 16, ..., 64, que corresponden a las paridades. La permutación $P1$ descarta los bits de paridad e invierte los 56 restantes. El resultado de hacer la permutación $P1(K)$ es dividir K en dos mitades, F_0 y G_0 , de 28 bits cada una. Los bloques F_0 y G_0 se desplazan hacia la izquierda para obtener cada subclave K_i . Si representamos por F_i y G_i los valores utilizados para obtener K_i : 

Tabla de permutaciones $P1$

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Nota

En esta tabla no figuran las posiciones añadidas 8, 16..., 64.

- $F_i = DC_i(F_{i-1})$,
- $G_i = DC_i(G_{i-1})$,

donde DC_i es un desplazamiento circular hacia la izquierda de tantas posiciones como determina la tabla de desplazamiento siguiente:

Tabla de desplazamientos DC

Número de interacción i	Desplazamiento i a la izquierda
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

Finalmente, la subclave K_i se obtiene a partir de la expresión:

$$K_i = P2(F_i G_i),$$

donde $P2$ es la permutación especificada en la siguiente tabla:

Tabla de permutaciones $P2$					
14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

2.1.4. Descifrado

Hasta aquí hemos descrito el proceso de cifrado del criptosistema DES. Ahora usaremos el mismo algoritmo para descifrar los datos de un bloque Y cifrado con una clave K . Pero en este caso las subclaves generadas se utilizarán en las iteraciones que se muestran en la figura del esquema general del DES, aunque en orden inverso. Es decir, en la primera iteración se utiliza K_{16} , en la segunda K_{15} , y sucesivamente hasta llegar a K_1 para la última iteración.

2.1.5. Particularidades del criptosistema

El criptosistema DES se puede implementar tanto en *software* como en *hardware*. Naturalmente, su implementación en *hardware* le confiere una velocidad de cifrado y de descifrado muy superior a las implementaciones *software*.

Cabe destacar que el DES puede trabajar con todos los modos de cifrado de bloques (CBC, CFB, OFB) gracias a la complejidad de sus operaciones, que le permiten actuar como un generador pseudoaleatorio.

Ya desde la aparición del DES en 1976, incluso antes de su estandarización, surgieron voces críticas sobre este criptosistema. Las dos debilidades más graves que se le atribuyen son la corta longitud de la clave, de sólo 56 bits, y la arbitrariedad de las cajas S , que podría obedecer a la existencia de una clave maestra que permitiera descifrar cualquier mensaje sin tener la clave original.

Con respecto a la longitud de la clave, Diffie y Hellman consideraron que se podría construir un ordenador con una arquitectura de procesadores paralelos que llegara a romper el DES en un día haciendo una búsqueda exhaustiva de claves. Parece ser que estas estimaciones, y otras realizadas al principio de los años ochenta, no se cumplieron, cuando menos en medios amplios, ya que el coste que suponía construir un ordenador de estas características era muy elevado. Pero el tiempo ha pasado y los avances en materia de tecnología informática han sido muy importantes, tanto, que han permitido romper sistemas como el DES en modo ECB.

Los distintos modos de cifrado en bloque CBC, CFB y OFB se tratan en el subapartado 1.3 de este módulo.

Fragilidad de la cifra de 56 bits

Actualmente ha quedado demostrada la fragilidad de los sistemas que trabajan con claves de 56 bits, como el DES, que se ha conseguido romper en 22 horas y 15 minutos gracias a un gran supercomputador, eso sí, con la ayuda añadida de unos cien mil ordenadores personales conectados por Internet. La clave del mensaje descifrado era 8558891AB0C851B6, y el mensaje que escondía decía: "Strong cryptography makes the world a safer place" (es decir, 'la criptografía fuerte hace del mundo un lugar más seguro').

Para resolver el problema de la longitud de la clave y continuar usando el DES como criptosistema seguro, se utiliza el cifrado triple, que en el caso del DES es conocido como triple DES.

Ved las características del cifrado triple en el subapartado 3.2 de este módulo.

2.2. El criptosistema IDEA

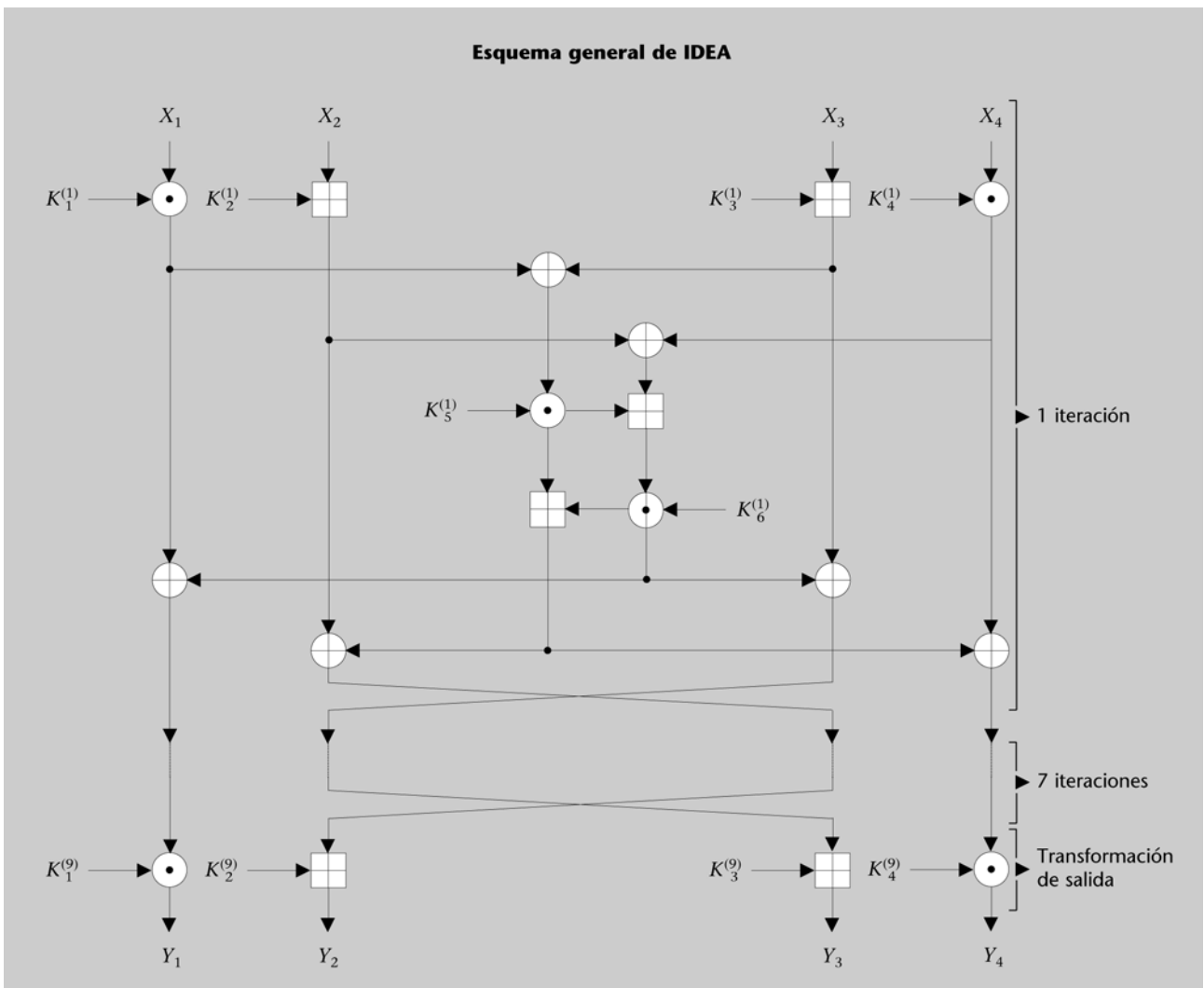
El criptosistema IDEA fue desarrollado por J. Massey y X. Lai en 1990. Es un sistema que cifra bloques de texto en claro de 64 bits de longitud por medio de una clave de 128 bits. Su funcionamiento se basa en ocho iteraciones idénticas seguidas de una transformación de salida.

IDEA es la sigla de *International Data Encryption Algorithm*.

En este criptosistema quedan garantizados los principios de confusión y difusión. Con respecto a los procesos de cifrado y descifrado, el algoritmo que se utiliza es el mismo, pero la obtención de las subclaves con que operan es diferente.

2.2.1. Descripción del funcionamiento

Podemos ver el esquema general de cifrado del IDEA en la figura siguiente:



Para que IDEA sea seguro, se ejecutan tres operaciones básicas de grupos algebraicos diferentes que actúan sobre bloques de 16 bits:

- 1) Operación XOR bit a bit de dos subbloques de 16 bits, que en la figura anterior se representa por \oplus .
- 2) Suma de enteros módulo 2^{16} , donde cada subbloque de 16 bits es tratado como un entero con representación binaria. En el esquema, esta operación se representa por \boxplus .
- 3) Multiplicación de enteros módulo $2^{16} + 1$. En este caso, el subbloque también se interpreta como la expresión en base dos de un entero, excepto en el caso del subbloque formado por todo ceros, que corresponde al valor entero 2^{16} . La operación se representa por \odot .

Operaciones con $2^{16} \bmod (2^{16} + 1)$

Fijémonos en el resultado de la operación siguiente:

$$(0, \dots, 0) \odot (1, 0, \dots, 0) = (1, 0, \dots, 0, 1).$$

Este resultado se debe a que:

$$2^{16} 2^{15} \bmod (2^{16} + 1) = 2^{15} + 1.$$

2.2.2. Detalle de una iteración

Como se muestra en el esquema de cifrado de IDEA, los 64 bits de entrada se separan en cuatro bloques de 16 bits de longitud cada uno: X_1, X_2, X_3, X_4 . Estos bloques, combinados con las claves $K_1^{(i)}, \dots, K_6^{(i)}$, en la i -ésima iteración correspondiente acaban dando los bloques de texto cifrado Y_1, Y_2, Y_3, Y_4 .

2.2.3. Generación de subclaves

En el criptosistema IDEA, como en muchos criptosistemas de bloque, se generan subclaves a partir de la clave inicial. En este caso, dados los 128 bits de la clave inicial, generaremos 52 subclaves de 16 bits. Para cada iteración de cifrado utilizaremos 6 subclaves, y 4 subclaves más para la transformación de salida.

Los procesos de cifrado y descifrado son esencialmente el mismo, ya que sólo varían las subclaves utilizadas. A continuación vamos a describir cómo se obtienen las claves para el cifrado, y obtendremos las del descifrado a partir de éstas:

- 1) La clave del usuario K se amplía hasta obtener una clave K_{ext} de 832 bits de longitud, que será la concatenación de las subclaves de cifrado que necesitamos, es decir:

$$K_{\text{ext}} = K_1^{(1)}, K_2^{(1)}, \dots, K_6^{(1)}, K_1^{(2)}, \dots, K_6^{(2)}, \dots, K_1^{(8)}, \dots, K_6^{(8)}, K_1^{(9)}, K_2^{(9)}, K_3^{(9)}, K_4^{(9)}.$$

La ampliación de la clave K a la clave ampliada K_{ext} se logra de la manera siguiente:

- a) La clave inicial K , de 128 bits, forma íntegramente los primeros 128 bits de la clave ampliada. De esta manera se obtienen las primeras ocho subclaves de 16 bits:

$$K_1^{(1)}, K_2^{(1)}, \dots, K_6^{(1)}, K_1^{(2)}, K_2^{(2)}.$$

b) Para obtener los 128 bits siguientes se efectúa una rotación hacia la izquierda de 25 posiciones de la clave inicial K ; de este modo se generan las ocho subclaves siguientes:

$$K_3^{(2)}, \dots, K_6^{(2)}, K_1^{(3)}, \dots, K_4^{(3)}.$$

c) El resto de subclaves se obtiene a partir de rotaciones de 25 posiciones hacia la izquierda del bloque de 128 bits obtenido en la rotación anterior. La operación se repite hasta obtener la última subclave, $K_4^{(9)}$.

2) Las claves utilizadas para el descifrado se obtienen a partir de las relaciones con las claves del cifrado, que se muestran en la tabla siguiente:

Claves de cifrado	Claves de descifrado
$K_1^{(1)}, K_2^{(1)}, K_3^{(1)}, K_4^{(1)}, K_5^{(1)}, K_6^{(1)}$	$K_1^{(9)-1}, -K_2^{(9)}, -K_3^{(9)}, K_4^{(9)-1}, K_5^{(8)}, K_6^{(8)}$
$K_1^{(2)}, K_2^{(2)}, K_3^{(2)}, K_4^{(2)}, K_5^{(2)}, K_6^{(2)}$	$K_1^{(8)-1}, -K_3^{(8)}, -K_2^{(8)}, K_4^{(8)-1}, K_5^{(7)}, K_6^{(7)}$
$K_1^{(3)}, K_2^{(3)}, K_3^{(3)}, K_4^{(3)}, K_5^{(3)}, K_6^{(3)}$	$K_1^{(7)-1}, -K_3^{(7)}, -K_2^{(7)}, K_4^{(7)-1}, K_5^{(6)}, K_6^{(6)}$
$K_1^{(4)}, K_2^{(4)}, K_3^{(4)}, K_4^{(4)}, K_5^{(4)}, K_6^{(4)}$	$K_1^{(6)-1}, -K_3^{(6)}, -K_2^{(6)}, K_4^{(6)-1}, K_5^{(5)}, K_6^{(5)}$
$K_1^{(5)}, K_2^{(5)}, K_3^{(5)}, K_4^{(5)}, K_5^{(5)}, K_6^{(5)}$	$K_1^{(5)-1}, -K_3^{(5)}, -K_2^{(5)}, K_4^{(5)-1}, K_5^{(4)}, K_6^{(4)}$
$K_1^{(6)}, K_2^{(6)}, K_3^{(6)}, K_4^{(6)}, K_5^{(6)}, K_6^{(6)}$	$K_1^{(4)-1}, -K_3^{(4)}, -K_2^{(4)}, K_4^{(4)-1}, K_5^{(3)}, K_6^{(3)}$
$K_1^{(7)}, K_2^{(7)}, K_3^{(7)}, K_4^{(7)}, K_5^{(7)}, K_6^{(7)}$	$K_1^{(3)-1}, -K_3^{(3)}, -K_2^{(3)}, K_4^{(3)-1}, K_5^{(2)}, K_6^{(2)}$
$K_1^{(8)}, K_2^{(8)}, K_3^{(8)}, K_4^{(8)}, K_5^{(8)}, K_6^{(8)}$	$K_1^{(2)-1}, -K_3^{(2)}, -K_2^{(2)}, K_4^{(2)-1}, K_5^{(1)}, K_6^{(1)}$
$K_1^{(9)}, K_2^{(9)}, K_3^{(9)}, K_4^{(9)}$	$K_1^{(1)-1}, -K_2^{(1)}, -K_3^{(1)}, K_4^{(1)-1}$

K^{-1} representa el inverso multiplicativo módulo $2^{16} + 1$ de K^* , $-K$ es el inverso aditivo**, y $K_1^{(i)}, \dots, K_6^{(i)}$, son las claves del cifrado de la iteración i -ésima.

* Eso significa que $K \odot K^{-1} = 1$.
 ** Eso significa que $-K \boxplus K = 0$.

2.2.4. Particularidades del criptosistema

Las características más importantes del criptosistema IDEA son:

- 1) La longitud de su clave (128 bits) imposibilita un criptoanálisis por búsqueda exhaustiva.
- 2) No utiliza nada parecido a las tan criticadas cajas S del DES, sino que simplemente ejecuta operaciones algebraicas de grupos diferentes. Además, como algunas de dichas operaciones son modulares, puede ser más rápido que otros criptosistemas que se basen en operaciones bit a bit.

3) Desde el punto de vista de la seguridad, no se conoce ningún ataque capaz de romper este criptosistema, que es resistente al criptoanálisis diferencial (sólo se han podido romper por esta técnica las dos primeras iteraciones).

4) El criptosistema se incluye en el *software* PGP, que permite enviar correo electrónico de manera segura.

2.3. Propuestas de AES

Ante el aumento de la potencia de cálculo de los ordenadores y, por otra parte, a causa de la controversia existente sobre la seguridad del DES en relación con la longitud de su clave, se hacía necesario sustituirlo. El National Institute of Standards and Technology (NIST) del gobierno norteamericano hizo un llamamiento en septiembre de 1997 para que se presentaran propuestas para un nuevo estándar de cifrado con el fin de seleccionar el Advanced Encryption Standard (AES).

En la petición de propuestas, el NIST solicitó un algoritmo de cifrado en bloque que fuera utilizable tanto en entornos gubernamentales como en entornos comerciales. Se especificó que tendría que soportar los modos estándares CBC, CFB, OFB, que tendría que ser “significativamente” más eficiente que el triple DES, y que la clave tendría que ser de longitud variable (128, 192 ó 256 bits, como mínimo), y los bloques de cifrado, de longitud 128 bits.

El 20 de agosto de 1998, el NIST hizo pública la primera lista de quince candidatos a AES. Mencionamos a continuación las quince propuestas seleccionadas, y especificamos el nombre de sus autores y algunas de las características más relevantes:

Nombre de la cifra	Fabricante y características
CAST-256	Entrust Technologies, Inc. Clave de 256 bits de longitud Cifrado en bloque de 128 bits 6 iteraciones
Crypton	Future Systems, Inc. Clave variable de hasta 256 bits Cifrado en bloque de 128 bits 12 iteraciones
DEAL	Richard Outerbridge y Lars Knudsen Clave variable de 128, 192 o 256 bits Cifrado en bloque de 128 bits 16 iteraciones
DFC	Centre National pour la Recherche Scientifique (CNRS) Clave variable de hasta 256 bits Cifrado en bloque de 128 bits 8 iteraciones
E2	Nippon Telegraph and Telephone Corporation (NTT) Clave de 128, 192 y 256 bits Cifrado en bloque de 128 bits 12 iteraciones

Nombre de la cifra	Fabricante y características
Frog	TecApro Internacional S.A. Clave de 40 a 1.000 bits (en múltiplos de 8) Cifrado en bloque desde 64 hasta 1.024 bits 8 iteraciones
HPC	Rich Schroepel Longitud de la clave de cualquier número de bits Cifrado en bloque de cualquier medida de bits Número variable de iteraciones
Loki97	Lawrie Brown, Josef Pieprzyk y Jennifer Seberry Clave de 128, 192 ó 256 bits de longitud Cifrado en bloque de 128 bits 16 iteraciones
Magenta	Deutsche Telekom AG Cifrado en bloque de 128 bits 6 iteraciones
Mars	IBM Clave de longitud variable Cifrado en bloque de 128 bits
RC6	Ronald Rivest, MJB Robshaw, R. Sidney, Y.L. Yin Clave de 16, 24 ó 32 bytes Cifrado en bloque de 128 bits 20 iteraciones
Rijndael	Joan Daemen y Vincent Rijmen Clave variable de 128, 192 ó 256 bits Cifrado en bloque de 128, 192 ó 256 bits El número de iteraciones depende de la longitud de la clave y de los bloques. Oscila entre 10 y 14 iteraciones
SAFER+	James L. Massey Clave 128, 192 ó 256 bits Cifrado en bloque de 128, 192 ó 256 bits El mismo número de iteraciones que SAFER
Serpent	Ross Anderson, Eli Bianzuelo y Lars Knudsen Clave de 256 bits Cifrado en bloque de 128 bits 32 iteraciones
Twofish	Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chirs Hall y Niels Ferguson Clave de 128, 192 ó 256 bits Cifrado en bloque de 128 bits 16 iteraciones

En agosto de 1999, el NIST publicó la lista de los cinco finalistas de entre las quince propuestas. Eran el Mars, RC6, Rijndael, Serpent y Twofish.

El 2 de octubre de 2000 el NIST hizo pública la decisión de seleccionar el criptosistema Rijndael como *Advanced Encryption Standard*. Los motivos que el NIST dio para seleccionar el AES fueron su combinación de seguridad, rendimiento, eficiencia, facilidad de implementación y flexibilidad. En particular, destacaron su eficiencia, tanto en implementaciones en *hardware* como en *software*.

Finalmente, el 26 de mayo de 2002 el FIPS anunció la aprobación del *Advanced Encryption Standard* bajo el código FIPS-197. Este estándar establece el Rijndael

como algoritmo simétrico de cifrado que las organizaciones del gobierno de Estados Unidos (y otros) pueden utilizar para proteger información sensible.

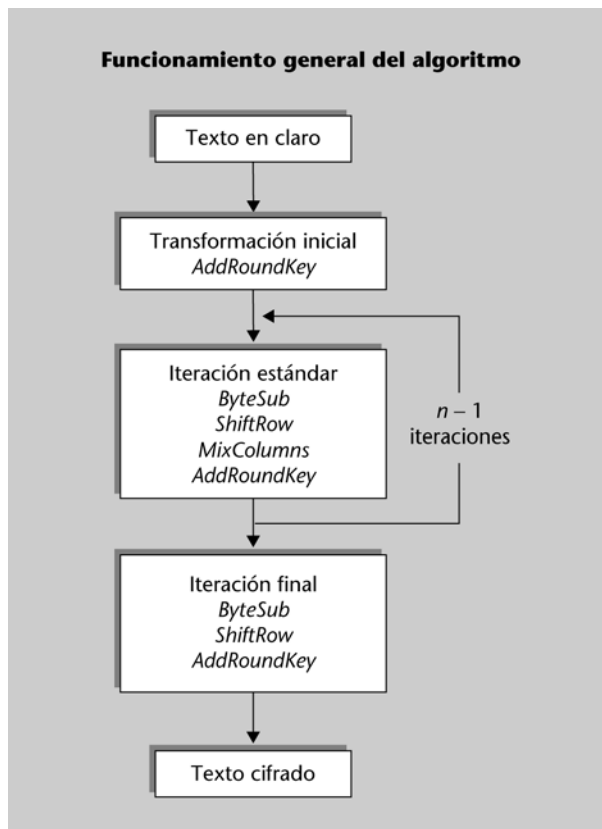
2.4. El criptosistema Rijndael

En 1998, los criptógrafos belgas Vincent Rijmen y Joan Daemen desarrollaron el algoritmo llamado (en reconocimiento de sus autores) criptosistema de Rijndael. Este criptograma fue el escogido por el NIST como AES.

El criptosistema de Rijndael cifra bloques de texto en claro de 128, 192 ó 256 bits de longitud, aunque el estándar aprobado por el NIST sólo utiliza bloques de 128 bits. La longitud de las claves de cifrado que este criptosistema utiliza también puede variar entre 128, 192 ó 256 bits. Las operaciones criptográficas se basan en un grupo finito de orden 2^8 .

2.4.1. Descripción del funcionamiento

El funcionamiento del criptosistema de Rijndael se muestra en la figura siguiente. Se basa en una transformación inicial seguida de un número de iteraciones que varía entre 10 y 14, según la longitud tanto del bloque que se cifra como de la clave.



El número de iteraciones

El número de iteraciones que se muestran en el gráfico es $n - 1$ porque la iteración final, aunque se considera iteración, no contiene la función *mixColumns*.

La tabla siguiente muestra el número exacto de iteraciones N en función del número de palabras de 32 bits que tienen los bloques de texto en claro (N_b), y de la clave (N_k):

		Longitud del bloque (N_b)		
		4	6	8
Longitud de la clave N_k	4	10	10	14
	6	12	12	14
	8	14	14	14

Bloques de 128 y 256 bits

Para cifrar bloques de 128 bits con claves de 128 bits se tendrán que hacer 10 iteraciones, mientras que para cifrar bloques de 256 bits con claves de 192 bits el número de iteraciones será de 14.

La unidad básica de información con que trabaja el criptosistema de Rijndael es el byte. Todas las cadenas de bits (textos en claro y claves) se representan con matrices de bytes. Por ejemplo, una cadena de 128 bits de texto en claro:

$$input_0 \ input_1 \ input_2 \ \dots \ input_{126} \ input_{127}$$

se representará con 16 bytes de la manera siguiente:

$$a_0 = [input_0 \ input_1 \ input_2 \ input_3 \ input_4 \ input_5 \ input_6 \ input_7]$$

$$a_1 = [input_8 \ input_9 \ input_{10} \ input_{11} \ input_{12} \ input_{13} \ input_{14} \ input_{15}]$$

...

$$a_{15} = [input_{120} \ input_{121} \ input_{122} \ input_{123} \ input_{124} \ input_{125} \ input_{126} \ input_{127}]$$

Estos bytes se pueden expresar en forma matricial:

$$\begin{bmatrix} a_0 & a_4 & a_8 & a_{12} \\ a_1 & a_5 & a_9 & a_{13} \\ a_2 & a_6 & a_{10} & a_{14} \\ a_3 & a_7 & a_{11} & a_{15} \end{bmatrix}$$

Las diferentes funciones que ejecuta el criptosistema de Rijndael (por ejemplo, *AddRoundKey*, *ByteSub*, etc.) tienen como entrada y como salida una matriz de bytes como la anterior.

Las matrices intermedias con que trabaja el criptosistema de Rijndael se llaman *matrices de estado*. Las **matrices de estado** tienen 4 filas y N_b columnas y cada elemento de la matriz es un byte. Los elementos de cada estado se denotan por s_{ij} , donde i determina la fila y j la columna.

Las operaciones “suma” y “producto” de bytes que ejecuta el criptosistema Rijndael **no** son las operaciones convencionales que conocemos. En concreto, el algoritmo Rijndael considera los bytes en una representación de polinomio. Cada byte b se puede representar con 8 bits:

$$b = [b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0], \text{ donde } b_i \in \{0, 1\}$$

Este conjunto de bits se puede expresar como los coeficientes de un polinomio de grado 7:

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0 = \sum_{i=0}^7 b_i x^i$$

Con el fin de simplificar la notación, representaremos los bytes en notación hexadecimal. Así, el elemento 01100011 en base binaria se representará por un 63 en base hexadecimal, ya que $0110\ 0011_{(2)} = 99_{(10)} = 63_{(16)}$.

Las operaciones “suma” y “producto” en estas representaciones se definen de la manera siguiente.

Sean las representaciones binarias de los bytes $x = (a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0)$ e $y = (b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0)$.

Definimos la operación “suma”:

$$x \oplus y = (a_7 \oplus b_7, a_6 \oplus b_6, a_5 \oplus b_5, a_4 \oplus b_4, a_3 \oplus b_3, a_2 \oplus b_2, a_1 \oplus b_1, a_0 \oplus b_0)$$

donde \oplus denota la operación XOR bit a bit.

Por otra parte, definimos la operación “producto”:

$$x \otimes y = (a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0) \cdot (b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0) \pmod{(x^8 + x^4 + x^3 + x + 1)}$$

Ejemplo de cálculo de “suma” y “producto”:

Dados los bytes x e y :

$$x = 57 = 01010111 = x^6 + x^4 + x^2 + x + 1 \text{ e } y = 83 = 10000011 = x^7 + x + 1$$

calculamos la suma y el producto de bytes:

- $x \oplus y = 57 \oplus 83 = D4$, ya que: $01010111 \oplus 10000011 = 11010100 = D4$.
- $x \otimes y = 57 \otimes 83 = C1$, ya que:

$$\begin{aligned} (x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) \pmod{(x^8 + x^4 + x^3 + x + 1)} &= \\ = (x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1) \pmod{(x^8 + x^4 + x^3 + x + 1)} &= \\ = x^7 + x^6 + 1 = 11000001 = C1. \end{aligned}$$

Una vez vistas estas representaciones, ya podemos pasar a ver el funcionamiento del algoritmo.

Antes de la primera iteración, el texto en claro se tiene que convertir en una matriz de estado. A continuación, el criptosistema de Rijndael ejecuta una transformación denominada *AddRoundKey*.

Ejemplo de representación polinómica

El byte 01100011 tiene como representación el polinomio $x^6 + x^5 + x + 1$.

La operación XOR

Recordad que la operación XOR queda definida por:

$$\begin{aligned} 1 \oplus 0 &= 0 \oplus 1 = 1, \\ 1 \oplus 1 &= 0 \oplus 0 = 0. \end{aligned}$$

Aritmética modular

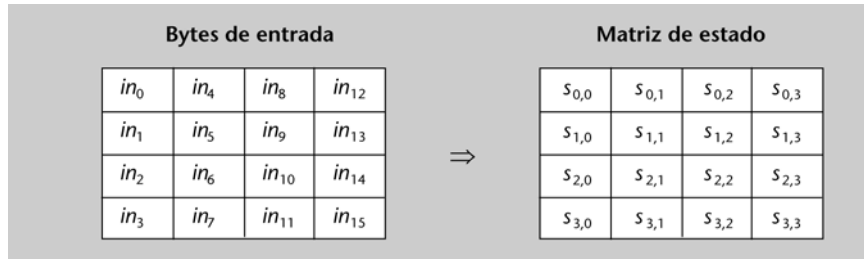
La notación *mod* utilizada en la definición de la operación “producto” se debe al hecho de trabajar con cuerpos finitos. En el subapartado 1.1 del módulo de “Cifras de clave pública” hay una descripción de aritmética modular para elementos enteros. En este caso hacemos lo mismo, pero en vez de trabajar con números enteros, trabajamos con polinomios. De cara a los cálculos que encontraréis en este subapartado, podéis prescindir de los conceptos teóricos y utilizar cualquier herramienta informática como cálculo simbólico, que permite trabajar directamente con aritmética modular.

Los ejemplos de este apartado

Para todos los ejemplos de las funciones del criptosistema de Rijndael de este módulo supondremos el caso $N_b = 4$, es decir que los bloques de texto para cifrar son de 128 bits. De hecho, esta longitud es la que especifica el AES.

Transformación inicial

En el gráfico siguiente se muestra la transformación del texto en claro en una matriz de estado para un caso con $N_b = 4$ (es decir, bloques para cifrar de 128 bits):



Sobre esta matriz de estado se aplica la función *AddRoundKey*.

La función *AddRoundKey* hace una suma XOR de la matriz de estado con cada byte de la subclave $K(i)$ correspondiente. En el caso de la transformación inicial tenemos $i = 0$; por lo tanto, utilizamos la primera subclave $K(0)$.

Las subclaves

$K(i)$ denota la subclave de $(32 \cdot N_b)$ bits que se utiliza en la i -ésima iteración teniendo en cuenta que $K(0)$ será la subclave que se utilizará para la transformación inicial. Podéis encontrar la descripción de cómo se obtienen las subclaves a partir de la clave inicial de cifrado en el subapartado 2.4.3 de este módulo.

Ejemplo de cálculo de la función *AddRoundKey*

Consideremos la subclave $K(0)$ y la matriz de estado S :

$K(0) = \text{B6 92 CF 0B 64 3D BD F1 BE 9B C5 00 68 30 B3 FE}$

$$S = \begin{bmatrix} 9D & 28 & 91 & 00 \\ F7 & 7F & 78 & A6 \\ 39 & C1 & 6C & C6 \\ 3C & AA & 25 & A5 \end{bmatrix}$$

El resultado de aplicar la función *AddRoundKey* será:

$$\text{AddRoundKey}(S, K(0)) = \begin{bmatrix} 9D & 28 & 91 & 00 \\ F7 & 7F & 78 & A6 \\ 39 & C1 & 6C & C6 \\ 3C & AA & 25 & A5 \end{bmatrix} \oplus \begin{bmatrix} B6 & 64 & BE & 68 \\ 92 & 3D & 9B & 30 \\ CF & BD & C5 & B3 \\ 0B & F1 & 00 & FE \end{bmatrix} = \begin{bmatrix} 2B & 4C & 2F & 68 \\ 65 & 42 & E3 & 96 \\ F6 & 7C & A9 & 75 \\ 37 & 5B & 25 & 5B \end{bmatrix}$$

Fijaos que la suma XOR de las matrices corresponde a la suma XOR de cada una de sus entradas. Así, por ejemplo, la primera posición de la transformación vale 2B, ya que $9D \oplus B6 = 10011101 \oplus 10110110 = 2B$.

2.4.2. Detalle de una iteración

En el gráfico del funcionamiento general del algoritmo se muestra que las $n - 1$ primeras iteraciones ejecutan las funciones *ByteSub*, *ShiftRow*, *MixColumns* y *AddRoundKey*, mientras que la última iteración ejecuta las primeras tres funciones, pero no ejecuta la función *MixColumns*.

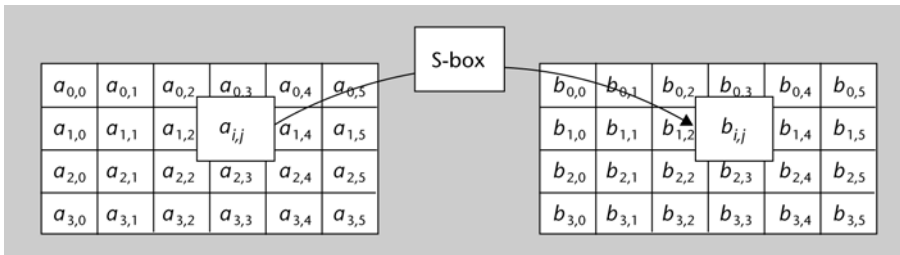
Pasamos a describir cada una de las funciones que se ejecutan en cada iteración.

La función *ByteSub* aplica una sustitución no lineal de los bytes de la matriz de estado.

Notación

La función *ByteSub* aparece con esta denominación en la propuesta inicial del criptosistema de Rijndael. En la publicación del AES en el estándar Fip-197, la función se llama *SubBytes*. Se cual sea el nombre que se dé, en los dos casos es la misma función.

La función recibe como entrada una matriz de estado A , aplica una transformación S y obtiene otra matriz de estado B , de manera que $b_{ij} = S(a_{ij})$, tal como muestra el esquema siguiente:



Más concretamente, la tabla de sustitución S , que actúa para cada bit de la matriz de estado, es invertible y se compone de dos transformaciones:

1) Dado el byte x , se expresa en forma polinómica y se calcula su inverso multiplicativo del polinomio resultante en el anillo de polinomios módulo $(x^8 + x^4 + x^3 + x + 1)$. Es decir, se calcula el polinomio x^{-1} , tal que:

$$x \cdot x^{-1} = 1 \pmod{(x^8 + x^4 + x^3 + x + 1)}.$$

Si $x = 0$ consideraremos $x^{-1} = 0$.

2) El polinomio x^{-1} resultante se transforma en la notación de byte y se aplica la transformación afín siguiente:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} x^{-1} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Ejemplo de cálculo de la función ByteSub

Supongamos que la matriz de estado vale:

$$S = \begin{bmatrix} B5 & B1 & B9 & B5 \\ C9 & CC & C5 & C8 \\ 17 & 11 & 1B & 15 \\ 9E & 99 & 92 & 9D \end{bmatrix}$$

Calculamos la transformación de la primera entrada de la matriz, $S_{11} = B5$. El byte B5 expresado en bits queda: 10110101. A continuación lo pasamos a la expresión polinómica: $x^7 + x^5 + x^4 + x^2 + 1$. Calculamos el inverso de este polinomio módulo $x^8 + x^4 + x^3 + x + 1$ es decir, el polinomio $b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$ tal que:

$$(x^7 + x^5 + x^4 + x^2 + 1)(b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0) = 1 \pmod{(x^8 + x^4 + x^3 + x + 1)}$$

Eso nos da el polinomio $x^6 + x^5 + x^4 + x^2 + 1$ que tiene una representación en binario 01110101. Si ahora aplicamos el producto de la matriz de sustitución:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Observad que...
 ... la posición del valor x^{-1} como vector columna tiene el bit menos significativo en la parte superior.

La transformación de la primera entrada de la matriz de estado para la función *ByteSub* vale 11010101. Si pasamos a notación hexadecimal, nos queda D5.

El resultado de aplicar los cálculos a cada entrada de la matriz de estado nos da la matriz siguiente:

$$ByteSub(S) = \begin{bmatrix} D5 & C8 & 56 & D5 \\ DD & 4B & A6 & E8 \\ F0 & 82 & AF & 59 \\ 0B & EE & 4F & 5E \end{bmatrix}$$

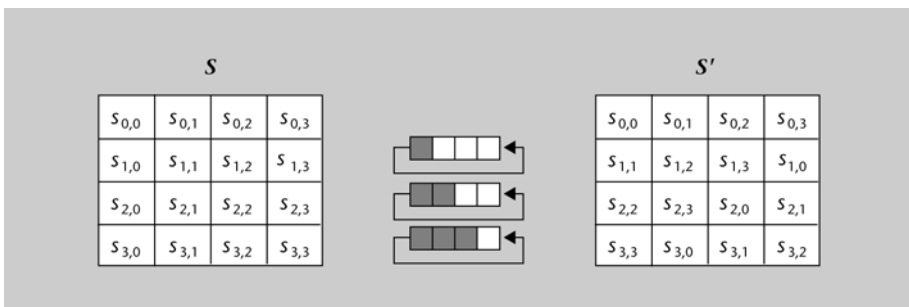
La función *ShiftRow* desplaza las filas de la matriz de estado de manera que la fila cero se deja igual, la fila 1 se desplaza $C1$ bytes a la izquierda, la fila 2 se desplaza $C2$ bytes a la izquierda y la fila 3, $C3$ bytes a la izquierda. Los valores $C1$, $C2$ y $C3$ dependen de la longitud del bloque N_b .

Estos valores se especifican en la tabla siguiente:

N_b	$C1$	$C2$	$C3$
4	1	2	3
6	1	2	3
8	1	3	4

Ejemplo de cálculo de la función *ShiftRow*

Supongamos que la longitud de la clave es de 128 bits, es decir, $N_b = 4$. Como siempre, consideramos que la longitud de los bloques es también de 128 bits, es decir $N_b = 4$. Eso quiere decir que $C1 = 1$, $C2 = 2$ y $C3 = 3$ y, por lo tanto, la transformación del estado para la función *ShiftRow* se puede expresar gráficamente como:



Por lo tanto, la matriz de estado S vale:

$$S = \begin{bmatrix} D5 & C8 & 56 & D5 \\ DD & 4B & A6 & E8 \\ F0 & 82 & AF & 59 \\ 0B & EE & 4F & 5E \end{bmatrix}$$

La matriz de estado transformada por la función *ShiftRow* será:

$$\text{ShiftRow}(S) = \begin{bmatrix} D5 & C8 & 56 & D5 \\ 4B & A6 & E8 & DD \\ AF & 59 & F0 & 82 \\ 5E & 0B & EE & 4F \end{bmatrix}$$

La función **MixColumns** mezcla las columnas de la matriz de estado a partir de operaciones polinomiales.

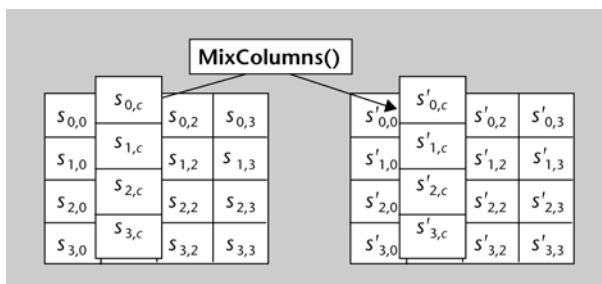
Concretamente, esta función considera las columnas de la matriz de estado como polinomios de grado 3. Cada columna se multiplica por el polinomio $c(x) = '03'x^3 + '01'x^2 + '01'x + '02'$ y el resultado se reduce al módulo $x^4 + 1$. El polinomio $c(x)$ es primo con respecto a $x^4 + 1$ y, por lo tanto, invertible. Este producto de los polinomios se puede escribir como un producto de matrices:

$$\begin{bmatrix} s'_{0j} \\ s'_{1j} \\ s'_{2j} \\ s'_{3j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0j} \\ s_{1j} \\ s_{2j} \\ s_{3j} \end{bmatrix} \quad \forall 0 \leq j \leq N_b$$

Tened en cuenta que las operaciones “suma” y “producto” entre los elementos de la matriz y los del vector columna son las operaciones \oplus y \otimes definidas en el subapartado anterior. Gráficamente, la función *MixColumns* hace la transformación siguiente:

¡Cuidado con el producto!

Daos cuenta de que este producto no es el producto estándar de polinomios, sino el definido en el subapartado 2.4.1 de este módulo.



Ejemplo de cálculo de la función *MixColumns*

Supongamos que la longitud de los bloques es de 128 bits, es decir $N_b = 4$ y que la longitud de la clave es también de 128 bits, es decir $N_k = 4$. Además, la matriz de estado vale:

$$S = \begin{bmatrix} D5 & C8 & 56 & D5 \\ 4B & A6 & E8 & DD \\ AF & 59 & F0 & 82 \\ 5E & 0B & EE & 4F \end{bmatrix}$$

Para obtener la transformación de la primera columna calcularemos:

$$S = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} D5 \\ 4B \\ AF \\ 5E \end{bmatrix}$$

Eso nos dará un vector columna de cuatro bytes determinados por los valores siguientes:

$$\begin{bmatrix} (02 \otimes D5) \oplus (03 \otimes 4B) \oplus (01 \otimes AF) \oplus (01 \otimes 5E) \\ (01 \otimes D5) \oplus (02 \otimes 4B) \oplus (03 \otimes AF) \oplus (01 \otimes 5E) \\ (01 \otimes D5) \oplus (01 \otimes 4B) \oplus (02 \otimes AF) \oplus (03 \otimes 5E) \\ (03 \otimes D5) \oplus (01 \otimes 4B) \oplus (01 \otimes AF) \oplus (02 \otimes 5E) \end{bmatrix}$$

Por ejemplo, veamos cuánto vale la segunda posición del vector columna:

$$(01 \otimes D5) \oplus (02 \otimes 4B) \oplus (03 \otimes AF) \oplus (01 \otimes 5E)$$

Si pasamos los valores hexadecimales a representación polinómica (pasando por su representación binaria) tenemos:

Hexadecimal	Binario	Polinomio
01	00000001	1
D5	11010101	$x^7 + x^6 + x^4 + x^2 + 1$
02	00000010	x
4B	01001011	$x^6 + x^3 + x + 1$
03	00000011	$x + 1$
AF	10101111	$x^7 + x^5 + x^3 + x^2 + x + 1$
5E	01011110	$x^6 + x^4 + x^3 + x^2 + x$

Si ahora hacemos los cálculos, resulta:

$$('01' \otimes 'D5') = (1)(x^7 + x^6 + x^4 + x^2 + 1) \text{ mod } x^8 + x^4 + x^3 + x + 1 = x^7 + x^6 + x^4 + x^2 + 1 \Rightarrow 11010101$$

$$('02' \otimes '4B') = (x)(x^6 + x^3 + x + 1) \text{ mod } x^8 + x^4 + x^3 + x + 1 = x^7 + x^4 + x^2 + x \Rightarrow 10010110$$

$$('03' \otimes 'AF') = (x + 1)(x^7 + x^5 + x^3 + x^2 + x + 1) \text{ mod } x^8 + x^4 + x^3 + x + 1 = x^7 + x^6 + x^5 + x^3 + x \Rightarrow 11101010$$

$$('01' \otimes '5E') = (1)(x^6 + x^4 + x^3 + x^2 + x) \text{ mod } x^8 + x^4 + x^3 + x + 1 = x^6 + x^4 + x^3 + x^2 + x \Rightarrow 01011110$$

Finalmente, hacemos la XOR:

$$11010101 \oplus 10010110 \oplus 11101010 \oplus 01011110 = 11110111 \Rightarrow F7$$

Concretamente, el resultado de todos los elementos de la primera columna es:

$$\begin{bmatrix} (02 \otimes D5) \oplus (03 \otimes 4B) \oplus (01 \otimes AF) \oplus (01 \otimes 5E) \\ (01 \otimes D5) \oplus (02 \otimes 4B) \oplus (03 \otimes AF) \oplus (01 \otimes 5E) \\ (01 \otimes D5) \oplus (01 \otimes 4B) \oplus (02 \otimes AF) \oplus (03 \otimes 5E) \\ (03 \otimes D5) \oplus (01 \otimes 4B) \oplus (01 \otimes AF) \oplus (02 \otimes 5E) \end{bmatrix} = \begin{bmatrix} 9D \\ F7 \\ 39 \\ 3C \end{bmatrix}$$

Y el resultado de la función *MixColumns* sobre toda la matriz de estado es:

$$MixColumns(S) = \begin{bmatrix} 9D & 28 & 91 & 00 \\ F7 & 7F & 78 & A6 \\ 39 & C1 & 6C & C6 \\ 3C & AA & 25 & A5 \end{bmatrix}$$

Para terminar, la última función que se aplica en cada iteración es la función *AddRoundKey*, que ya hemos descrito.

2.4.3. Generación de subclaves

Al igual que la mayoría de criptosistemas en bloque, el algoritmo de Rijndael trabaja con diferentes subclaves en cada iteración. Estas subclaves se obtienen por la aplicación de una función de ampliación en la clave de cifrado inicial.

La función de ampliación genera una clave ampliada $W = (W_0, W_1, \dots, W_{N_b(N_r+1)-1})$ a partir de las N_K palabras de 32 bits de clave de cifrado $K = (K_0, K_1, \dots, K_{N_K-1})$, que contiene $N_b(N_r + 1)$ palabras de 32 bits. Cada $K(i)$, que denota cada una de las subcadenas de W de N_b palabras de 32 bits, es la subclave que se utiliza en la i -ésima iteración. Gráficamente, las subclaves de cada iteración en relación con la clave ampliada se pueden expresar de la manera siguiente:

$$W = (\underbrace{W_0, W_1, \dots, W_{N_b-1}}_{K(0)}, \underbrace{W_{N_b}, \dots, W_{2N_b-1}}_{K(1)}, \dots, \underbrace{W_{N_b(N_r)}, \dots, W_{N_b(N_r+1)-1}}_{K(N_r)})$$

En la transformación inicial se utiliza la subclave $K(0)$, formada por las primeras N_b palabras de W y en cada una de las N_r iteraciones se utilizan N_b palabras.

Más concretamente, la función de ampliación de claves *KeyExpansion* viene definida por el algoritmo siguiente, según si $N_k \leq 6$ o $N_k > 6$:

a) Caso $N_k \leq 6$. El algoritmo consta de dos etapas:

- **Inicialización**, fase en que la clave de cifrado se copia íntegramente en las primeras posiciones de la clave ampliada; es decir: $W_i = K_i$ para $i = 0, \dots, N_k - 1$.
- **Fase de ampliación**, en la que se toma la última palabra calculada y se amplía. Esta fase tiene dos pasos:

```

temp = W_{i-1}
si i = 0 (mod N_k) entonces
    temp = SubWord(RotWord(temp)) ⊕ Rcon[⌊i/N_k⌋]
fini
W_i = W_{i-N_k} ⊕ temp

```

b) Caso $N_k > 6$. El algoritmo consta de dos etapas:

- **Inicialización**, en el que la clave de cifrado se copia íntegramente en las primeras posiciones de la clave ampliada; es decir: $W_i = K_i$ para $i = 0, \dots, N_K - 1$.

- **Fase de ampliación**, en la que se toma la última palabra calculada y se amplía. Esta fase tiene dos pasos:

```

temp = Wi-1
si i = 0 (mod Nk) entonces
    temp = SubWord(RotWord(temp)) ⊕ Rcon⌊i/Nk⌋
sino
    si i = 4 (mod N4) entonces
        temp = SubWord(temp)
    fin si
fin si
Wi = Wi-Nk ⊕ temp

```

En ambos casos, la fase de ampliación utiliza dos subfunciones, la *SubWord* y la *RotWord*. La función **SubWord** es la misma función que *ByteSub* (definida en el subapartado anterior). La función **RotWord** simplemente ejecuta una permutación cíclica en la palabra de 4 bytes, es decir, si tiene una entrada $[a_0, a_1, a_2, a_3]$, la salida será: $[a_1, a_2, a_3, a_0]$. Por otra parte, también se utiliza la constante $Rcon[i]$, que vale $Rcon[i] = [x^{i-1}, '00', '00', '00']$. Recordad que x en hexadecimal vale '02', ya que corresponde a la representación en binario de 00000010.

Ejemplo de generación de subclaves

Supongamos que la longitud de los bloques es de 128 bits, es decir $N_b = 4$ y que la clave de cifrado en hexadecimal vale:

```

K = 00010203   04050607   08090A0B   0C0D0E0F
    K0         K1         K2         K3

```

Es decir, la longitud de la clave es también de 128 bits, $N_k = 4$ (cuatro palabras de 32 bits).

Con estos parámetros y dada la tabla del subapartado 2.4.1, el número de iteraciones será $N_r = 10$. Esto quiere decir que la clave ampliada W tendrá $4 \cdot (10 + 1) = 44$ palabras de 32 bits, donde $K(i)$ denota la clave que se utiliza en la i -ésima iteración.

Los primeros bytes de la clave ampliada son los mismos que los de la clave de cifrado:

```

W0 = 00 01 02 03
W1 = 04 05 06 07
W2 = 08 09 0A 0B
W3 = 0C 0D 0E 0F

```

Por lo tanto: $K(0) = W_0W_1W_2W_3 = 00010203\ 04050607\ 08090A0B\ 0C0D0E0F = K$.

Estos cuatro bytes son los que se utilizan en la transformación inicial como el algoritmo.

La segunda subclave será:

```

W4 = W0 ⊕ SubWord(RotWord(W3)) ⊕ Rcon[1]
    RotWord(W3) = RotWord(0C 0D 0E 0F) = 0D 0E 0F 0C
    SubWord(0D 0E 0F 0C) = (D7 AB 76 FE)
W4 = 00 01 02 03 ⊕ D7 AB 76 FE ⊕ 01 00 00 00 = D6 AA 74 FD
W5 = W1 ⊕ W4 = 04 05 06 07 ⊕ D6 AA 74 FD = D2 AF 72 FA
W6 = W2 ⊕ W5 = 08 09 0A 0B ⊕ D2 AF 72 FA = DA A6 78 F1
W7 = W3 ⊕ W6 = 0C 0D 0E 0F ⊕ DA A6 78 F1 = D6 AB 76 FE

```

Por lo tanto, la subclave $K(1) = D6\ AA\ 74\ FD\ D2\ AF\ 72\ FA\ DA\ A6\ 78\ F1\ D6\ AB\ 76\ FE$.

El resto de la clave ampliada se calcula de la misma manera.

2.4.4. Descifrado

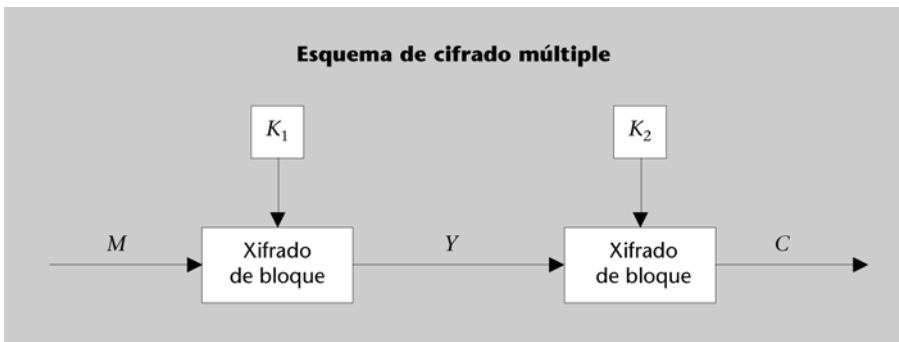
Hemos definido con todo detalle las operaciones de cifrado del criptosistema de Rijndael. Todas las funciones que se utilizan en el proceso de cifrado (*ByteSub*, *ShiftRow*, *MixColumn* y *AddRoundKey*) son invertibles y, correspondientemente, se puede definir su función inversa.

Si las funciones definidas en el cifrado se aplican en el orden opuesto al que se ejecutan en el proceso de cifrado, obtenemos el proceso de descifrado del criptosistema.

3. Ataques al cifrado en bloque

3.1. Ataque del punto intermedio

Con el fin de aumentar el espacio de las claves para frustrar un criptoanálisis de búsqueda exhaustiva, una alternativa es encadenar diferentes cifrados de bloque en que cada uno utilice una clave K_i diferente, como muestra la figura siguiente:



Esta técnica de cifrado múltiple requiere la utilización de un criptosistema –como el DES– que no forme un grupo algebraico. Esta última condición es necesaria porque si la función de cifrado formara un grupo algebraico, la operación de cifrar un mensaje con una clave K_1 y volver a cifrar el resultado con otra clave K_2 sería la misma que cifrar el mensaje sólo una sola vez con una tercera clave K_3 . 🚫

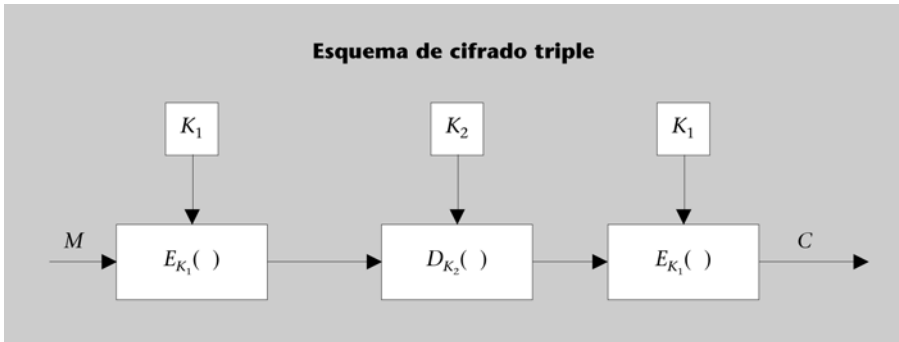
Aunque podríamos pensar que el cifrado múltiple debe incrementar mucho el espacio de las claves, en realidad no lo hace tanto como parece. Se ha demostrado que un cifrado de dos etapas del DES se puede romper con una búsqueda exhaustiva de sólo 2^{n+1} iteraciones, en vez de las 2^{2n} que parecerían más lógicas. Ello se debe a la aplicación del criptoanálisis conocido como *ataque del punto intermedio*.

Supongamos que K_1 y K_2 son claves de n bits. El **ataque del punto intermedio** calcula $D_{K_2}(C)$ para los 2^n valores posibles de K_2 . A continuación se calcula $E_{K_1}(M)$ para los 2^n valores posibles de K_1 . A la fuerza debe haber un valor de K_1 y uno de K_2 tales que cumplan $E_{K_1}(M) = Y = D_{K_2}(C)$. Éstos son los valores correctos de las claves.

Si sumamos el total de operaciones efectuadas entre el primer y en el segundo pasos, obtenemos $2^n + 2^n = 2^{n+1}$ operaciones.

3.2. Cifrado triple

Para evitar ataques como el del punto intermedio y conseguir a la vez una duplicación efectiva del espacio de las claves, se utiliza el cifrado triple. El **cifrado triple** combina la función de cifrado en bloque con la de descifrado del mismo bloque con claves diferentes, como muestra la figura siguiente:



Cada caja representa el cifrador de bloque. El cifrador aplica a ambos extremos del mensaje la función de cifrado con la clave E_{K_1} , mientras que en la caja de en medio se aplica la función de descifrado D_{K_2} . Las ecuaciones correspondientes son:

- $C = E_{K_1}(D_{K_2}(E_{K_1}(M)))$,
- $M = D_{K_1}(E_{K_2}(D_{K_1}(C)))$.

De este modo se consigue doblar el número de bits de la clave del criptosistema y evitar a la vez el ataque del punto intermedio.

Actualmente, la longitud de la clave del DES (56 bits) se considera demasiado corta y casi todas las aplicaciones implementan lo que se conoce como *triple DES*, que no es más que tomar el DES como cifrador para el esquema de cifrado triple.

3.3. Criptoanálisis diferencial


El criptoanálisis diferencial es un ataque estadístico de texto en claro conocido que se utiliza contra sistemas que se basan en iteraciones de una transformación criptográfica débil. El objetivo es reducir el número de operaciones necesarias para romper el criptosistema mediante una búsqueda exhaustiva. Cada criptosistema requiere un esquema de ataque concreto.

La idea general del criptoanálisis diferencial es ver cómo se propaga por el algoritmo de cifrado una diferencia conocida de texto en claro. Dados dos textos en claro M y M' y una diferencia conocida $D = M \oplus M'$, se obtienen los textos cifrados correspondientes $C = E_K(M)$ y $C' = E_K(M')$, y se estudia la relación entre D y $D_c = C \oplus C'$.

Un dato

El criptoanálisis diferencial, que no se hizo público hasta 1991, ya debía de ser conocido por los investigadores de IBM que desarrollaron el DES, pues, curiosamente, este criptosistema, que fue creado durante los años setenta, resiste el ataque de aquél.

4. Gestión de claves

La característica más importante del cifrado con clave compartida es que se utiliza la misma clave para cifrar y para descifrar y la comparten el emisor y el receptor. Esto supone un riesgo, porque ya se sabe que el secreto mejor guardado es el que no se comparte. Por este motivo hay que diseñar sistemas de gestión de claves que se basen en estructuras jerárquicas cuyas claves se usen para cifrar otras claves de un nivel inferior. Los **principales tipos de claves** según su importancia y su función, son: 

1) **Clave maestra.** Clave de jerarquía superior, que no se utiliza nunca para cifrar mensajes, sino solamente para cifrar claves primarias y secundarias. Esta clave se guarda en claro en un dispositivo seguro.

2) **Clave primarias o secundaria.** Claves de jerarquía inferior a la clave maestra y que se guardan cifradas con la clave maestra. Los tipos más importantes son:

a) **Clave de generación de claves,** que se utiliza para generar aleatoriamente claves de sesión o vectores de inicialización.

b) **Clave de cifrado de claves,** que se utiliza para cifrar otras claves con el fin de protegerlas.

3) **Clave de sesión o de comunicación,** que se envía por el canal de comunicación al empezar una sesión, cifrada con una clave secundaria de cifrado de claves. Se caracteriza porque cifra aquella comunicación y sólo aquella, y cuando la comunicación ha acabado, la clave se borra.

4) **La clave de cifrado de archivos,** que se puede considerar de sesión, usada para cifrar archivos completos. Se caracteriza porque se guarda en la cabecera del archivo que cifra.

No hace falta decir que la piedra angular de toda la seguridad en un sistema jerárquico como el que se ha descrito recae en la clave maestra, que es la que cifra las otras claves de rango inferior, por lo cual debe generarse y guardarse de manera segura. Normalmente, tanto la clave maestra como el propio algoritmo de cifrado de claves se guardan en un módulo de seguridad de memoria no volátil (como por ejemplo, circuitos integrados), que puede incorporar también otras medidas de seguridad, como por ejemplo, la autodestrucción en caso de que se intente manipular el sistema.

Resumen

En este módulo hemos descrito el funcionamiento general de las cifras en bloque actuales. En concreto, hemos tratado de los aspectos siguientes:

- 1) Hemos definido los conceptos de *confusión* y *difusión* y también los métodos de cifrado posibles de un criptosistema en bloque.
- 2) Hemos detallado el funcionamiento de cuatro modos de cifrado: el ECB, el CBC, el CFB y el OFB.
- 3) Hemos descrito el funcionamiento de tres criptosistemas en bloque: DES, IDEA y el AES. Hemos mencionado sus características principales, explicado los procesos de cifrado y descifrado y la generación de las subclaves, y detallado sus iteraciones.
- 4) Hemos hecho referencia a algunas técnicas de criptoanálisis del cifrado en bloque y a la manera de evitar algunas de ellas.
- 5) Finalmente, hemos explicado el problema de la gestión de las claves y los esquemas teóricos existentes para resolverla.

Actividades

1. Implementad el DES en modo CBC.
2. Implementad el IDEA con ocho bóvedas de cifrado.
3. Implementad el AES.

Ejercicios de autoevaluación

1. Supongamos que la clave principal de 64 bits de un cifrador DES es la siguiente:

01010001 01010001 01010100 01000101 01000011 01000000 01001100 01001111

Hallad la primera subclave de 48 bits, es decir, K_1 .

2. Suponed que las claves utilizadas con el DES consisten sólo en las letras A-Z y tienen ocho caracteres de longitud. Dad una estimación del tiempo que se tardaría en probar todas las claves utilizando una búsqueda exhaustiva, suponiendo que cada clave puede ser probada en un microsegundo. Haced lo mismo para claves que contengan ocho caracteres restringidos a mayúsculas, minúsculas y dígitos decimales.

3. Suponemos que la clave principal de 128 bits de un cifrador IDEA es la siguiente:

01001001 01000100 01000101 01000001 00100000 01100101 01110011 00100000
01101100 01100001 00100000 01100011 01101100 01100001 01110110 01100101

Hallad los 16 bits que forman la segunda clave de la cuarta iteración, es decir, $K_{\frac{4}{2}}$.

4. Para la misma clave inicial del ejercicio de autoevaluación 3, hallad la tercera clave de descifrado correspondiente a la sexta iteración.

5. Supongamos que la clave de cifrado de 192 bits de un cifrador Rijndael expresada en hexadecimal es la siguiente:

8E 73 B0 F7 DA 0E C8 10 F3 2 B 80 90 79 E5 62 F8 EA D2 52 2C 6B 7B

Dad las dos primeras subclaves, es decir, $K(0)$ y $K(1)$.

6. Dado un cifrador Rijndael con clave de cifrado K y un bloque de texto para cifrar B :

$K = 2B\ 7E\ 15\ 16\ 28\ AE\ D2\ A6\ AB\ F7\ 15\ 88\ 09\ CF\ 4F\ 3C$

$B = 32\ 43\ F6\ A8\ 88\ 5A\ 30\ 8D\ 31\ 31\ 98\ A2\ E0\ 37\ 07\ 34$

¿Cuántas iteraciones hay que realizar para cifrar este bloque de texto en claro con esta clave? ¿Cuál es la matriz de estado en el inicio de la segunda iteración?

Solucionario

Ejercicios de autoevaluación

1. En primer lugar, tenemos que aplicar la permutación $P1$ en la clave de entrada. Esto nos da la secuencia de 56 bits siguiente:

0000 0000 1111 1111 0000 0000 0000 1001 0000 1100 1100 1100 0000 0111

Por lo tanto, al dividirla en dos partes tenemos:

- $F_0 = 0000\ 0000\ 1111\ 1111\ 0000\ 0000\ 0000$
- $G_0 = 1001\ 0000\ 1100\ 1100\ 1100\ 0000\ 0111$

Si aplicamos el desplazamiento circular DC_1 , que, tal como nos muestra la tabla DC , supone un desplazamiento hacia la izquierda, tendremos:

- $F_1 = 0000\ 0001\ 1111\ 1110\ 0000\ 0000\ 0000$
- $G_1 = 0010\ 0001\ 1001\ 1001\ 1000\ 0000\ 1111$

Finalmente, aplicamos la permutación $P2$ y obtenemos:

$K_1 = 101000\ 001001\ 001001\ 000010\ 101101\ 010100\ 100101\ 000100$

2. Si cada clave está formada por ocho caracteres de un alfabeto de 26 letras, el número total de claves diferentes es: $26^8 = 2,088270 \cdot 10^{11}$. Por lo tanto, si suponemos que cada clave se puede probar en un microsegundo, 10^{-6} segundos, se tardarían unos 208.827 segundos, es decir, unas 58 horas para hacer una búsqueda exhaustiva.

Si, además de las mayúsculas, añadimos las minúsculas y los dígitos decimales, los valores que obtenemos son: $(26 + 26 + 10)^8 = 2,183401 \cdot 10^{14}$ claves. Entonces, necesitaremos 218.340.105 segundos, o 60.650 horas o 2.527 días de tiempo.

Como podemos comprobar, el espacio de claves no se puede restringir a caracteres con sentido.

3. Como en cada bóveda del criptosistema IDEA se utilizan seis claves, la segunda de la cuarta vuelta será la vigésima clave ($3 \cdot 6 + 2$). Los primeros 128 bits de la clave principal generan las ocho primeras claves. Por lo tanto, la vigésima clave es generada por las posiciones 49, 50..., 64 del tercer grupo de bloques de 128 bits de clave inicial. Como cada bloque queda desplazado 25 posiciones a la derecha, tenemos que el segundo bloque empieza en la posición 26 y el tercero, en la 51. Por lo tanto, la segunda clave de la cuarta iteración la forman los bits de las posiciones 99, 100, 101..., 114 de la clave inicial, es decir, 10110001 10000101 (el valor 99 se obtiene de $51 + 48$).

4. Si nos fijamos en la tabla de obtención de las claves de descifrado, vemos que en la sexta iteración las claves las da $K_1^{(4)-1}$, $-K_3^{(4)}$, $-K_2^{(4)}$, $K_4^{(4)-1}$, $K_5^{(3)}$, $K_6^{(3)}$, donde el superíndice i representa la i -ésima iteración de cifrado. Por consiguiente, la tercera clave que nos piden es $-K^{(4)}$. Teniendo en cuenta el resultado del ejercicio anterior, 10110001 10000101, lo que buscamos es su inverso aditivo módulo 2^{16} , que es 01001110 01111011.

5. Dado que la clave de cifrado es de 192 bits, el número de palabras de 32 bits de la clave vale $N_K = 6$; por lo tanto, tendremos que aplicar el algoritmo para el caso $N_K \leq 6$.

Los primeros bits de la clave extendida son exactamente los mismos bits de la clave de cifrado:

$W_0 = 8E\ 73\ B0\ F7$
 $W_1 = DA\ OE\ 64\ 52$
 $W_2 = C8\ 10\ F3\ 2\ B$
 $W_3 = 80\ 90\ 79\ E5$
 $W_4 = 62\ F8\ EA\ D2$
 $W_5 = 52\ 2C\ 6\ B\ 7B$

Por lo tanto:

$K(0) = W_0W_1W_2W_3W_4W_5 =$
 $= 8E73B0F7\ DA\ OE\ 64\ 52\ C810F32B\ 809079E5\ 62F8EAD2\ 522C6B7B =$
 $= K$

Si aplicamos el algoritmo para el caso $N_K \leq 6$ con los valores W_i anteriores obtenemos:

$W_6 = W_0 \oplus \text{SubWord}(\text{RotWord}(W_5)) \oplus \text{Rcon}[1]$
 $\text{RotWord}(W_5) = \text{RotWord}(52\ 2C\ 6B\ 7B) = 2C\ 6B\ 7B\ 52$
 $\text{SubWord}(2C\ 6B\ 7B\ 52) = (71\ 7F\ 21\ 00)$
 $W_6 = 8E\ 73\ B0\ F7 \oplus 71\ 7F\ 00 \oplus 01\ 00\ 00\ 00 =$
 $= 8E\ 73\ B0\ F7 \oplus 70\ 7F\ 00 = FE\ OC\ 91\ F7$

$$\begin{aligned}
 W_7 &= W_1 \oplus W_6 = \text{DA 0E 64 52} \oplus \text{FE 0C 91 F7} = \text{24 02 F5 A5} \\
 W_8 &= W_2 \oplus W_7 = \text{C8 10 F3 2B} \oplus \text{24 02 F5 A5} = \text{EC 12 06 8E} \\
 W_9 &= W_3 \oplus W_8 = \text{80 90 79 E5} \oplus \text{EC 12 06 8E} = \text{6C 82 7F 6B} \\
 W_{10} &= W_4 \oplus W_9 = \text{62 F8 EA D2} \oplus \text{6C 82 7F 6B} = \text{0E 7A 95 B9} \\
 W_{11} &= W_5 \oplus W_{10} = \text{52 2C 6B 7B} \oplus \text{0E 7A 95 B9} = \text{5C 56 FE C2}
 \end{aligned}$$

Por lo tanto, la subclave:

$$K(1) = \text{FE 0C 91 F7 24 02 F5 A5 EC 12 06 8E 6C 82 7F 6B 0E 7A 95 B9 5C 56 FE C2}$$

6. Habrá que hacer diez iteraciones para cifrar este bloque de texto en claro, ya que tanto la longitud del bloque de cifrado como la longitud de la clave es de 16 bytes; por lo tanto, $N_b = 4$ y $N_k = 4$.

En la transformación inicial se aplica la transformación *AddRoundKey*. En nuestro caso:

$$\text{AddRoundKey}(S, K(0)) = \begin{bmatrix} 32 & 88 & 31 & E0 \\ 43 & 5A & 31 & 37 \\ F6 & 30 & 98 & 07 \\ A8 & 8D & A2 & 34 \end{bmatrix} \oplus \begin{bmatrix} 2B & 28 & AB & 09 \\ 7E & AE & F7 & CF \\ 15 & D2 & 15 & 4F \\ 16 & A6 & 88 & 3C \end{bmatrix} = \begin{bmatrix} 19 & A0 & 9A & E9 \\ 3D & F4 & C6 & F8 \\ E3 & E2 & 8D & 48 \\ BE & 2B & 2A & 08 \end{bmatrix} = S1$$

El resultado de la primera iteración corresponderá a ejecutar las funciones *ByteSub*, *ShiftRow*, *MixColumns* y *AddRoundKey*. El resultado de la función *ByteSub* sobre la matriz de estado *S1* es:

$$\text{ByteSub} \begin{bmatrix} 19 & A0 & 9A & E9 \\ 3D & F4 & C6 & F8 \\ E3 & E2 & 8D & 48 \\ BE & 2B & 2A & 08 \end{bmatrix} = \begin{bmatrix} D4 & E0 & B8 & 1E \\ 27 & BF & B4 & 41 \\ 11 & 98 & 5D & 52 \\ AE & F1 & E5 & 30 \end{bmatrix} = S2$$

El resultado de la función *ShiftRow* sobre la matriz de estado *S2* es:

$$\text{ShiftRow} \begin{bmatrix} D4 & E0 & B8 & 1E \\ 27 & BF & B4 & 41 \\ 11 & 98 & 5D & 52 \\ AE & F1 & E5 & 30 \end{bmatrix} = \begin{bmatrix} D4 & E0 & B8 & 1E \\ BF & B4 & 41 & 27 \\ 5D & 52 & 11 & 98 \\ 30 & AE & F1 & E5 \end{bmatrix} = S3$$

El resultado de la función *MixColumns* sobre la matriz de estado *S3* es:

$$\text{MixColumns} \begin{bmatrix} D4 & E0 & B8 & 1E \\ BF & B4 & 41 & 27 \\ 5D & 52 & 11 & 98 \\ 30 & AE & F1 & E5 \end{bmatrix} = \begin{bmatrix} 04 & E0 & 48 & 28 \\ 66 & CB & F8 & 06 \\ 81 & 19 & D3 & 26 \\ E5 & 9A & 7A & 4C \end{bmatrix} = S4$$

Calculamos el valor de la clave de la segunda iteración:

$$\begin{bmatrix} A0 & 88 & 23 & 2A \\ FA & 54 & A3 & 6C \\ FE & 2C & 39 & 76 \\ 17 & B1 & 39 & 05 \end{bmatrix}$$

Finalmente, el resultado de la función *AddRoundKey* sobre la matriz de estado *S4* resulta:

$$\begin{bmatrix} 04 & E0 & 48 & 28 \\ 66 & CB & F8 & 06 \\ 81 & 19 & D3 & 26 \\ E5 & 9A & 7A & 4C \end{bmatrix} \oplus \begin{bmatrix} A0 & 88 & 23 & 2A \\ FA & 54 & A3 & 6C \\ FE & 2C & 39 & 76 \\ 17 & B1 & 39 & 05 \end{bmatrix} = \begin{bmatrix} A4 & 68 & 6B & 02 \\ 9C & 9F & 5B & 6A \\ 7C & 35 & EA & 50 \\ F2 & 2B & 43 & 49 \end{bmatrix}$$

Así, el valor de la matriz de estado al inicio de la segunda iteración valdrá:

$$S = \begin{bmatrix} A4 & 68 & 6B & 02 \\ 9C & 9F & 5B & 6A \\ 7F & 35 & EA & 50 \\ F2 & 2B & 43 & 49 \end{bmatrix}$$

Glosario

Advanced Encryption Standard *m* Criptosistema Rijndael, que cifra bloques de 128 bits por medio de una clave que puede variar la longitud entre 128, 192 ó 256 bits.

AES véase Advanced Encryption Standard.

CBC véase Cipher Bloc Chaining.

CFB véase Cipher Feedback.

Cipher Bloc Chaining *m* Modo de cifrado en bloque en que se crea un encadenamiento de los bloques, de manera que el cifrado de un bloque depende del anterior por medio de un bloque inicial aleatorio para el cifrado.

Cipher Feedback *m* Modo de cifrado en bloque en que la longitud de los bloques de texto no tiene que coincidir con la de los bloques del criptosistema.

confusión *f* Técnica que tiene por objetivo que la relación entre la clave y el texto cifrado sea tan complicada como se pueda.

Data Encryption Standard *m* Criptosistema de cifrado en bloque que cifra bloques de datos de 64 bits de longitud por medio de una clave de 56 bits y la acción de cajas S.

DES véase Data Encryption Standard.

difusión *f* Técnica que tiene por objetivo la disipación de las propiedades estadísticas del texto en claro mediante el texto cifrado.

ECB véase Electronic Code Book.

Electronic Code Book *m* Modo de cifrado en bloque en que el cifrado de los bloques es independiente el uno del otro y se lleva a cabo con una misma clave.

IDEA véase International Data Encryption Algorithm.

International Data Encryption Algorithm *m* Criptosistema de cifrado en bloque que cifra bloques de texto en claro de 64 bits de longitud por medio de una clave de 128, por medio de ocho iteraciones idénticas y una transformación de salida.

OFB véase Output Feedback.

Output Feedback *m* Modo de cifrado en bloque en que el vector inicial se realimenta directamente con el resultado del cifrado en bloque.

Triple DES *m* Protocolo de cifrado triple que utiliza el DES como base para obtener un cifrador con un espacio de claves superior.

Bibliografía

Biham, E.; Shamir, A. (1991). "Differential Cryptanalysis of DES-like Cryptosystems". *Journal of cryptology* (vol. 4, núm. 1, pág. 3-72).

Lay, X.; Massey, J. (1990). "A proposal for a new block encryption standard". *Advances in cryptology - Eurocrypt'90* (pág. 389-404).

Meier, W. (1993). "On the security of the IDEA Block cipher". *Advances in Cryptology - Eurocrypt'93* (pág. 371-385).

Merkle, R.; Hellman, M. (1981). "On the security of multiple encryption". *Communications of the ACM* (vol. 24, núm. 7, pág. 465-467).

Meyer, C. H.; Matyas, S. M. (1982). *Cryptography: a new dimension in computer data security*. Nueva York: John Wiley & Sons.

National Institute of Standards and Technology (NIST) (1977). "Data encryption standard". *Federal information processing standards* (núm. 46). Washington: NIST.

National Institute of Standards and Technology (NIST)(1981). "DES, modes of Operation". *Federal Information Processing Standards* (núm. 81). Washington: NIST.

National Institute of Standards and Technology (NIST) (2001). "Advanced Encryption Standard (AES)". *Federal information processing standards* (núm. 197). Washington: NIST.