4. Les xifres de bloc

Una alternativa a les xifres de flux són les **xifres de bloc**. Aquestes xifres s'inclouen també dins dels criptosistemes de clau compartida ja que la clau que s'utilitza per a xifrar i desxifrar és la mateixa i la comparteixen emissor i receptor. La diferència bàsica entre el xifratge en flux i el xifratge en bloc és la utilització de la memòria en els algorismes de xifratge.

Ja hem vist en el capítol anterior que el xifrat de flux utilitza una clau diferent per cada bit d'informació. Aquesta clau depèn de l'estat inicial del generador, però també de l'estat del generador en el moment de xifrar un bit concret. Per tant, dos bits iguals es poden xifrar de maneres diferents depenent de l'estat en què es trobi el generador. En el xifratge en bloc això no passa ja que les xifres en bloc actuen sense memòria, i per tant el text xifrat només depèn del text en clar i de la clau. D'aquesta manera, dos blocs de text en clar iguals es xifren sempre de la mateixa manera quan s'utilitza la mateixa clau. Caldrà estudiar aquest fet en detall ja que si no es corregeix, els sistemes de xifrat que en resulten són força vulnerables, ja que es poden inserir o esborrar blocs de text xifrat sense que es pugui detectar. A més, el fet que dos blocs de text en clar quedin xifrats d'una mateixa manera, pot donar pistes per a una possible criptoanàlisi de tipus estadístic.

Pel que fa a la seva utilització, les xifres de bloc són força utilitzades ja que aconsegueixen una velocitat acceptable de xifratge. En concret, el xifrador en bloc més utilitzat és l'AES (Advanced Encryption Standard) ja que està establert com a estàndard per el NIST des de l'any 2002.

4.1 Definició de les xifres de bloc

Les xifres de bloc són un dels elements més importants en criptografia i es fan servir en diferents contextos. D'una banda, es poden fer servir directament en esquemes de xifrat per tal de proporcionar confidencialitat. D'altra banda però, també es fan servir com a primitives bàsiques en altres esquemes criptogràfics, com ara els generadors pseudoaleatoris, les funcions hash o els codis d'autenticació de missatges (coneguts per les seves sigles en anglès, MAC de *Message Authentication Codes*).

Una **xifra de bloc** és una funció que rep un bloc b d'n bits de text en clar i retorna un text xifrat c també d'n bits:

$$c = E_k(b)$$

Diem que n és, aleshores, la **mida de bloc** del criptosistema.

Noteu que la funció rep com a paràmetre el valor k, que representa la clau. La mida de la clau és la longitud en bits de k.

Per tal d'assegurar que al desxifrar un text xifrat amb E (amb una mateixa clau k) obtenim el text original, la funció E ha de ser invertible. Així doncs, les xifres de bloc diposen també d'una funció de desxifrat, que realitza el procés invers de la de xifrat:

$$b = D_k(c)$$

La majoria de vegades que fem servir un criptosistema de bloc voldrem xifrar contingut que supera la mida del bloc del criptosistema utilitzat. En aquests casos, el que es fa és partir el text que cal xifrar, m, en diversos blocs, $m_1, m_2, \ldots$, cada un dels quals té la llargada corresponent al bloc per a xifrar (n bits), i xifrar cadascun dels fragments. El procediment a seguir per xifrar cadascun dels fragments queda determinat per el **mode d'operació**.

4.1.1 Modes d'operació

El mode d'operació més senzill és coneix com a **ECB** (de l'anglès, *Electronic Code Book*) i consisteix a xifrar cada un dels blocs del missatge en clar, m_i, de manera individual, fent servir la mateixa clau. Així, s'obtenen els blocs xifrats c_i, que es concatenen per formar el text xifrat c. La Figura 4.1 esquematitza el procés de xifrat en mode ECB.

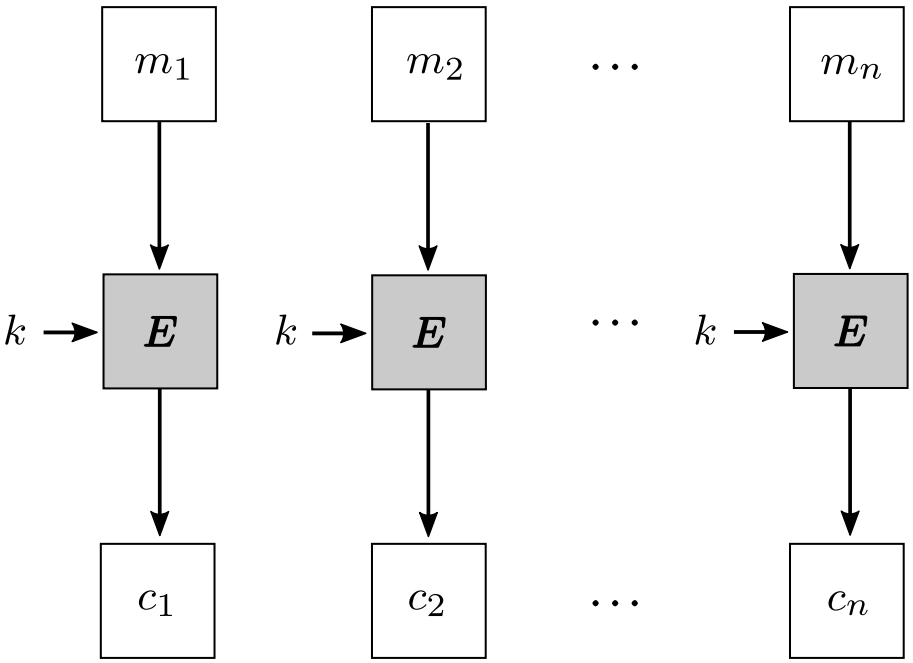


Figura 4.1: Esquema de xifrat amb el mode ECB.

Les propietats principals que ens ofereix el mode ECB són:

1. Els blocs de text en clar idèntics resulten en blocks xifrats també idèntics (si es fa servir la mateixa clau).
2. Cada bloc es xifra de manera independent als altres blocs.

3. Permet accés aleatori al contingut, és a dir, és possible desxifrar un bloc i sense haver de desxifrar els anteriors.
4. Els errors no es propaguen: un error en un bloc afecta només a aquell bloc.

Com a conseqüència immediata d'aquestes propietats, el mode ECB és vulnerable a certs atacs. D'una banda, per la propietat 1) un atacant que observi el text xifrat pot aprendre directament si el text original conté blocs iguals. A més, aquesta propietat també pot facilitar els atacs de tipus estadístic per a obtenir la clau k. Així mateix, el mode ECB no és capaç d'amagar els patrons en les dades. D'altra banda, per la propietat 2), un atacant pot reordenar el text xifrat, fent que al desxifrar-se s'obtingui el text en clar reordenat, sense que el receptor pugui detectar el canvi. Addicionalment, un atacant també pot inserir blocs de text xifrat o eliminar-ne, sense que el desxifrat posterior falli.

Per tal d'exemplificar les conseqüències de fer servir el mode ECB per a xifrar dades de mida superior al bloc, procedim a xifrar una imatge amb aquest mode, i a visualitzar el text xifrat resultant també en forma d'imatge (veure Figura 4.2.

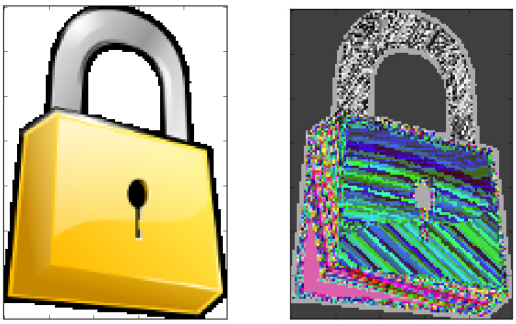


Figura 4.2: Exemple de xifrat d'una imatge amb ECB.

La imatge de l'esquerra correspon a la imatge en clar i la de la dreta és el resultat de xifrar la primera imatge fent servir el mode d'operació ECB. Com es pot apreciar, tot i que detalls concrets de la imatge original no es revel·len en la versió xifrada (per exemple, el color), la silueta de la imatge queda perfectement reconeixible.

Exercici 4.1 Suposem un esquema de xifrat de bloc amb mida de bloc de 2 bits i mida de clau també de 2 bits que implementa la següent funció de xifrat E:

Entrada	k	Sortida	Entrada	k	Sortida
00	00	11	00	01	00
01	00	10	01	01	01
10	00	01	10	01	10
11	00	00	11	01	11
00	10	01	00	11	10
01	10	11	01	11	00
10	10	00	10	11	11
11	10	10	11	11	01

Xifreu el missatge $m = 1001100100110000$ amb $k = 10$ fent servir la funció de xifrat E i el mode d'operació ECB.

El mode **CBC** (de l'anglès, *Cipher Block Chaining*) consisteix en l'encadenament dels blocs per al xifratge, de manera que es crea una dependència del xifratge de cada bloc amb l'immediatament anterior. De nou, cada bloc es xifra amb la mateixa clau k, però el text que es xifra no és directament el bloc en clar, sinó el resultat d'una XOR entre el bloc en clar i el bloc xifrat anterior. La Figura 4.3 esquematitza el funcionament del mode CBC.

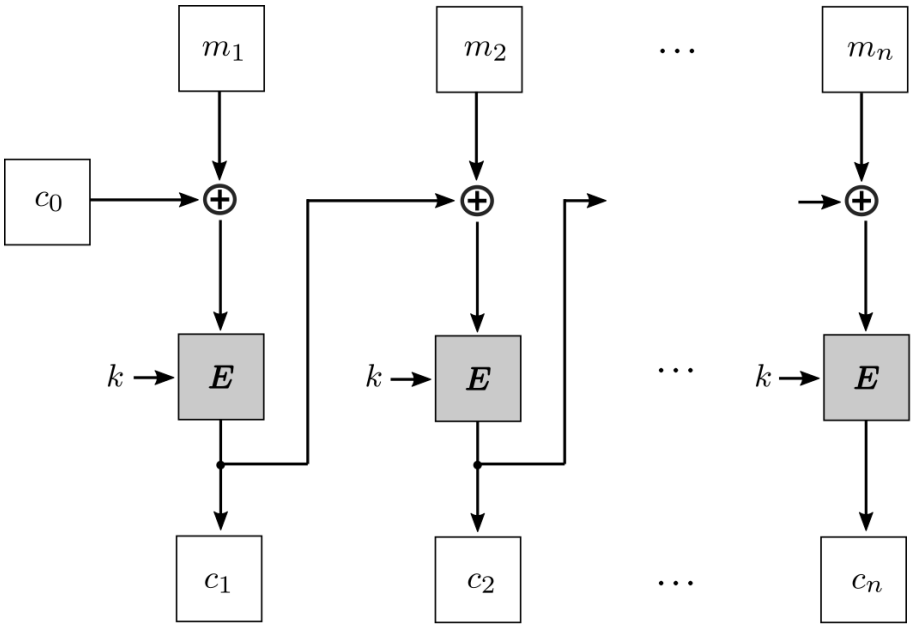


Figura 4.3: Esquema de xifrat amb el mode CBC.

Suposem un xifratge de bloc amb una clau k, una funció de xifratge E i una de desxifratge D. Si $m_1, \ldots, m_m$ són els blocs de text en clar que cal xifrar, mitjançant el sistema CBC el xifratge del bloc m_i es porta a terme de la manera següent:

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

Per a fer-ne el desxifratge també ens cal partir del text xifrat anterior, i aleshores hem d'executar l'operació següent:

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

Per a xifrar el primer bloc necessitarem un bloc inicial aleatori, c_0, que no cal que sigui secret. Aleshores, incloent aquest nou vector inicial en el xifratge podrem obtenir dos textos en clar iguals però xifrats de manera diferent; així, encara que emprem la mateixa clau, k, només ens caldrà canviar el vector inicial, c_0, que, a més, pot incorporar una marca temporal.

En contraposició amb el mode ECB, si un atacant canvia l'ordre dels blocs xifrats amb CBC, aleshores el procés de desxifrat no es realitza correctament. Addicionalment, un error en un bloc xifrat afecta el desxifrat d'aquell bloc, però també del següent. Noteu que els blocs successius es desxifren ja correctament.

Amb aquesta estructura, el mode CBC aconsegueix ocultar els patrons del text en clar molt millor que el mode ECB. Si repetim el procediment de xifrar la imatge del cadenat fent servir ara el mode CBC, podem observar com ara en la Figura 4.4 no podem intuir el perfil de la imatge a partir de la imatge xifrada.

Exercici 4.2 Xifreu el mateix missatge m amb la funció E definida en l'exercici 4.1 i la clau $k = 10$, fent servir ara el mode d'operació CBC amb el vector inicial 10.

El mode de xifratge **CFB** (de l'anglès, *Cipher Feedback*) utilitza indirectament el xifrador de bloc, com veurem a continuació. Per això, la llargada dels blocs que s'han de xifrar no cal que sigui la mateixa que la

Figura 4.4: Exemple de xifrat d'una imatge amb CBC.

dels blocs del criptosistema amb què actua, sinó que pot ser més petita. L'esquema general de funcionament d'aquest mètode es mostra a la Figura 4.5.

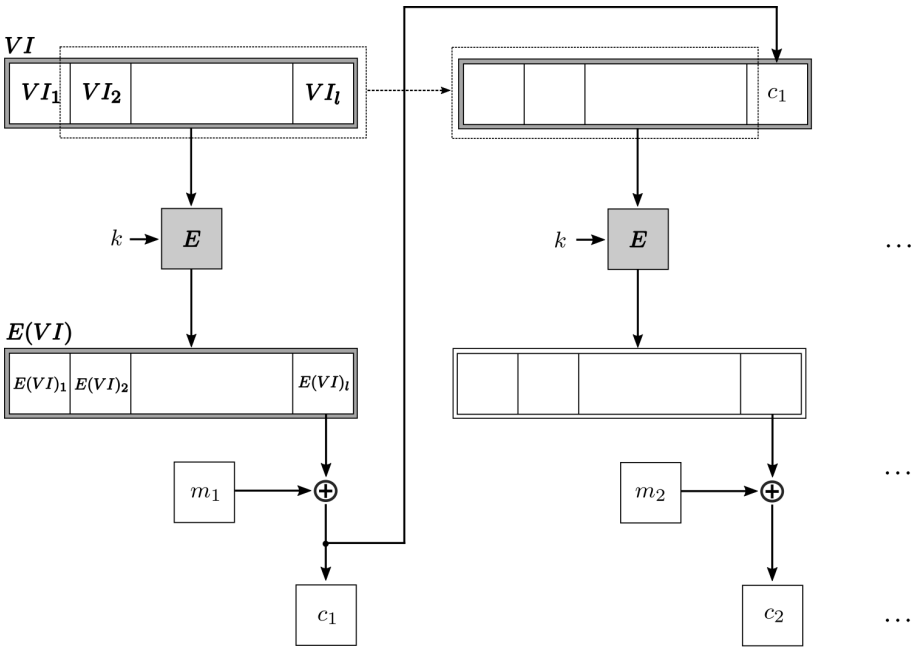


Figura 4.5: Esquema de xifrat amb el mode CFB.

Donat $m = m_1 m_2 \ldots$, en què m és el missatge de text en clar, i $m_1, m_2, \ldots$ representen els blocs de longitud n que formen el missatge, si considerem el vector inicial VI com una concatenació d'l blocs de longitud n, és a dir, $VI = VI_1 VI_2 \ldots VI_l$, on VI_i i té n bits de llargada, podrem calcular el xifratge del vector VI, $E(VI)$, mitjançant el criptosistema de bloc.

El resultat tindrà la mateixa llargada que VI i, per tant, el podrem descompondre de la mateixa manera que aquell:

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

Finalment, ja podrem xifrar el primer bloc de text en clar, m_1, fent la suma bit a bit amb el darrer bloc, $E(VI)_l$:

$$c_1 = m_1 \oplus E(VI)_l;$$

obtenim així el primer bloc xifrat de longitud n, c_1.

Per a xifrar el segon bloc, m_2 , tornarem a fer el mateix procés, però aquesta vegada prendrem com a vector inicial el vector format pels fragments següents:

$$VI = VI_2 VI_3 \ldots VI_l c_1,$$

és a dir, hem desplaçat els blocs d'n bits cap a l'esquerra per afegir-hi el bloc c_1 i descartar-ne el VI_1. D'aquesta manera, el segon bloc de text xifrat l'obtenim fent l'operació següent:

$$c_2 = E(VI_b)_l \oplus m_2.$$

El procés es repeteix al llarg dels blocs de text que es vol xifrar: per al bloc següent es desplacen els blocs del vector inicial anterior, $VI_b, \ldots$ a l'esquerra per afegir-hi el darrer bloc de text xifrat obtingut i anar aplicant el que ja hem descrit anteriorment.

Exercici 4.3 Xifreu el mateix missatge m amb la funció E definida en l'exercici 4.1 i la clau $k = 10$, fent servir ara el mode d'operació CFB amb el vector inicial 10.

El mode de xifratge **OFB** (de l'anglès, *Output Feedback*) utilitza el criptosistema de bloc com a generador pseudoaleatori. És un sistema molt semblant a l'anterior; l'única diferència que presenta és que el vector inicial es realimenta directament amb el resultat del xifratge de bloc abans de fer la suma bit a bit amb el bloc de text en clar, com es pot veure a la Figura 4.6.

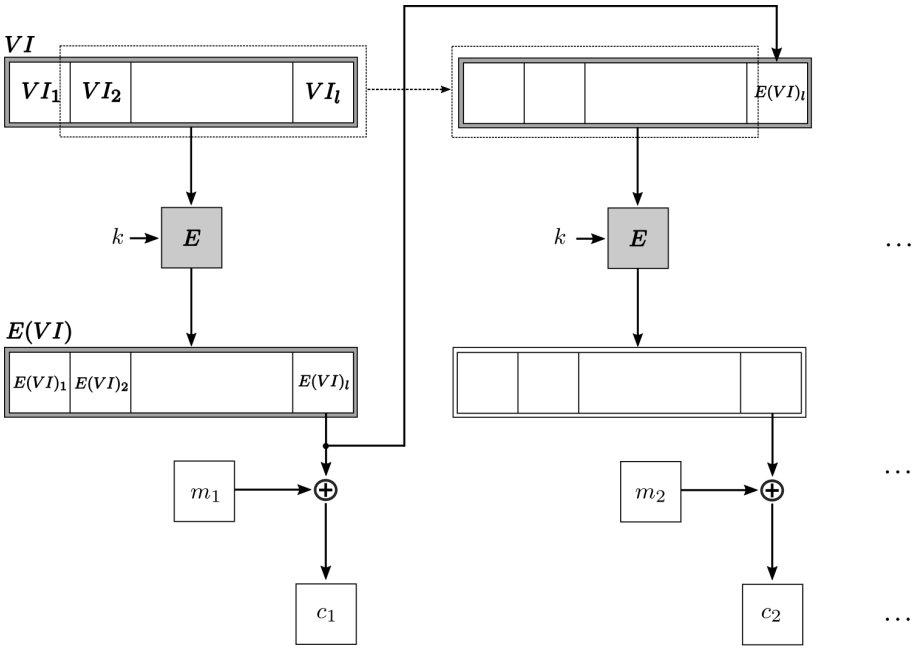


Figura 4.6: Esquema de xifrat amb el mode OFB.

Com que el xifrador de bloc actua com un generador pseudoaleatori, cal que els criptosistemes de bloc que emprem amb el mode OFB compleixin les característiques requerides per als generadors pseudoaleatoris, tant pel que fa a la impredictibilitat de la seqüència resultant com a la complexitat lineal.

Exercici 4.4 Xifreu el mateix missatge m amb la funció E definida en l'exercici 4.1 i la clau $k = 10$, fent servir ara el mode d'operació OFB amb el vector inicial 10.

El mode **CTR** (de l'anglès, *counter*) és similar a l'OFB, convertint també el criptosistema de bloc amb un xifrador de flux. La seqüència de xifrat es genera xifrant successius valors d'un comptador (d'aquí en sorgeix el seu nom), que pot ser qualsevol funció que tingui un període gran (veure Figura 4.7).

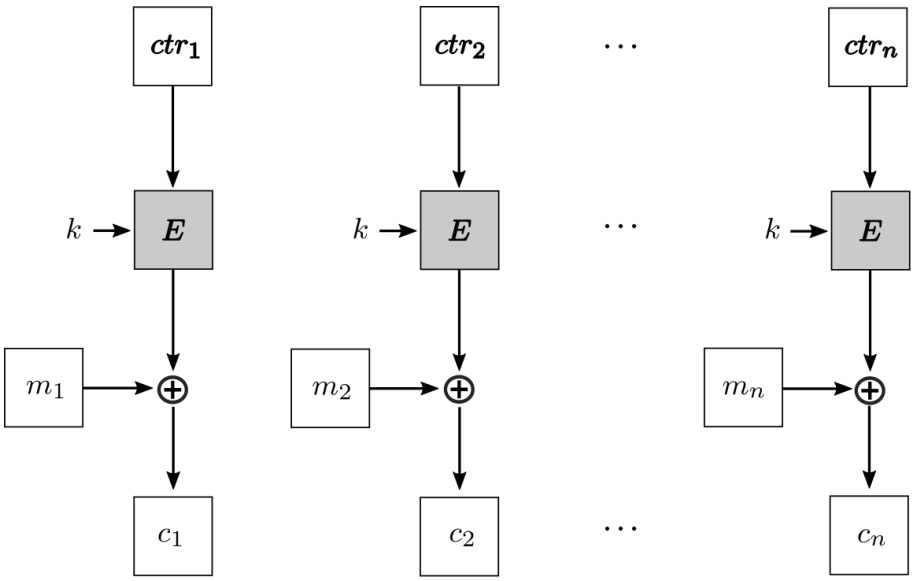


Figura 4.7: Esquema de xifrat amb el mode CTR.

Un ús habitual és fer servir un valor de nonce aleatori concatenat amb un comptador que s'incrementi d'un en un. Així, per exemple, si la mida de bloc del xifrador a utilitzar és de 128 bits, se selecciona una nonce de 64 bits i un comptador de 64 bits. Per a xifrar el primer bloc, es concatena la nonce amb el comptador inicialitzat a 0. Per a cada nou bloc, el comptador s'incrementa en 1. D'aquesta manera, es poden xifrar 2^{64} blocs amb la mateixa nonce.

El principal avantatge d'aquest mode d'operació és que permet paral·lelitzar tant el procés de xifrat com el de desxifrat, el que el fa addient per funcionar en dispositius amb més d'un processador. A més, permet accés aleatori (com el mode ECB).

Exercici 4.5 Xifreu el mateix missatge m amb la funció E definida en l'exercici 4.1 i la clau $k = 10$, fent servir ara el mode d'operació CTR amb el vector inicial 10.

4.2 El criptosistema AES

L'any 1998, els criptògrafs belgues Vincent Rijmen i Joan Daemen van desenvolupar l'algorisme anomenat (en reconeixement dels autors) criptosistema de Rijndael. Aquest criptosistema va ser triat pel NIST com a AES (de l'anglès, *Advanced Encryption Standard*) l'any 2000, reemplaçant el DES.

De fet, el Rijndael és una família d'algorismes de xifrat amb diferents mides de clau i de bloc. En concret, el Rijndael defineix blocs i claus de mida mínima 128 i màxima de 256, acceptant múltiples de 32 bits. L'AES n'és només un subconjunt, amb mida de bloc fixada a 128 bits.

El **criptosistema AES** xifra blocs de text en clar de **128 bits** de longitud. La longitud de les claus de xifratge que aquest criptosistema empra pot variar entre **128, 192 o 256** bits. Les operacions criptogràfiques es basen en un grup finit d'ordre 2^8.

4.2.1 Descripció del funcionament

El funcionament de l'AES es mostra en la Figura 4.8. Es basa en una transformació inicial seguida d'un nombre d'iteracions que varien entre 10 i 14, segons la longitud de la clau.

El nombre d'iteracions	El nombre d'iteracions que es mostren en el gràfic és $n-1$ perquè la iteració final, tot i que es considera iteració, no conté la funció mixColumn.

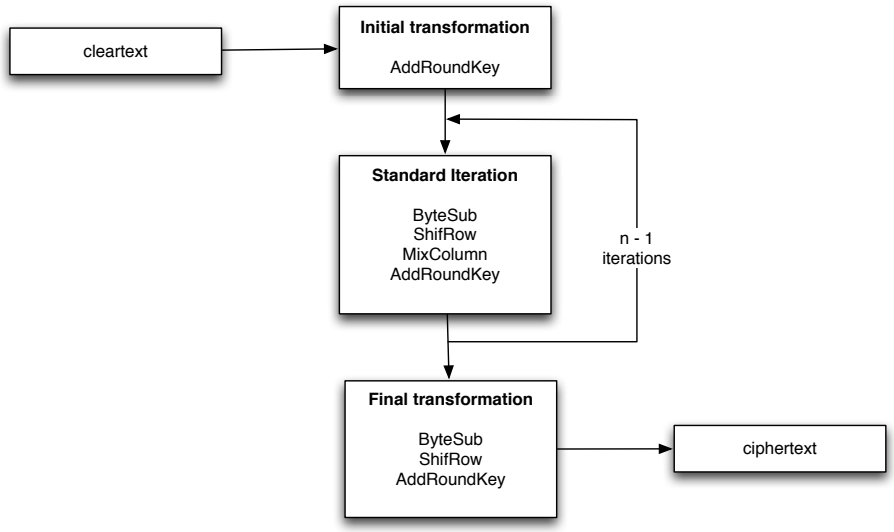


Figura 4.8: Estructura de l'AES.

La taula següent mostra el nombre exacte d'iteracions Nr en funció del nombre de paraules de 32 bits que té la clau que s'utilitza per xifrar (Nk):

$Nk=4$	$Nk=6$	$Nk=8$
10	12	14

La unitat bàsica d'informació amb què treballa l'AES és el byte. Totes les cadenes de bits (textos en clar i claus) es representen amb matrius de bytes. Per exemple, una cadena de 128 bits de text en clar:
$m = m_1 m_2 \cdots m_{127} m_{128}$

es representarà amb 16 bytes de la següent manera:
$a_{0,0} = m_1 m_2 m_3 m_4 m_5 m_6 m_7 m_8$
$a_{1,0} = m_9 m_{10} m_{11} m_{12} m_{13} m_{14} m_{15} m_{16}$
$\cdots$

$a_{3,3} = m_{121}m_{122}m_{123}m_{124}m_{125}m_{126}m_{127}m_{128}$

i aquests bytes es poden expressar de forma matricial:

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

Les diferents funcions que executa l'AES (per exemple, AddRoundKey, ByteSub, etc.) tenen com a entrada i com a sortida una matriu de bytes com l'anterior.

Les matrius intermèdies amb què treballa el criptosistema AES s'anomenen **matrius d'estat**. Les matrius d'estat són matrius 4×4 i cada element de la matriu és un byte. Els elements de cada estat es denoten per s_{ij}, on i determina la fila i j la columna.

Les operacions "suma" i "producte" de bytes que executa l'AES no són les operacions convencionals que coneixem. En concret, l'AES considera els bytes en una representació de polinomi. Cada byte b es pot representar amb 8 bits:

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

Aquest conjunt de bits es pot expressar com els coeficients d'un polinomi de grau 7:

$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0 = \sum_{i=0}^{7} b_i x^i$. Per exemple, el byte 01100011 té com a representació el polinomi $x^6 + x^5 + x + 1$

Per tal de simplificar la notació, representarem els bytes en notació hexadecimal. Així, l'element 01100011 en base binària es representarà per un 63 en base hexadecimal, ja que $01100011_{(2} = 99_{(10} = 63_{(16}$.

Donades aquestes representacions, considerem que la "suma" i el "producte" es defineixen de la manera següent.

Siguin les representacions binàries dels bytes $x = (x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0)$ i $y = (y_7, y_6, y_5, y_4, y_3, y_2, y_1, y_0)$.

Definim l'operació suma:
$x \oplus y = (x_7 \oplus y_7, x_6 \oplus y_6, x_5 \oplus y_5, x_4 \oplus y_4, x_3 \oplus y_3, x_2 \oplus y_2, x_1 \oplus y_1, x_0 \oplus y_0)$
on $\oplus$ denota l'operació XOR bit a bit.[1]

D'altra banda, definim l'operació producte:
$x \otimes y = (x_7x^7 + x_6x^6 + x_5x^5 + x_4x^4 + x_3x^3 + x_2x^2 + x_1x + x_0)(y_7x^7 + y_6x^6 + y_5x^5 + y_4x^4 + y_3x^3 + y_2x^2 + y_1x + b_0) \pmod{x^8 + x^4 + x^3 + x + 1}$.

Exemple 4.1 Càlcul de "suma" i "producte":

Donats els bytes x i y:

$x = 57_{(16} = 01010111_{(2} = x^6 + x^4 + x^2 + x + 1$

$y = 83_{(16} = 10000011_{(2} = x^7 + x + 1$,

calculem la suma i el producte de bytes:

[1] Recordeu que l'operació XOR queda definida per: $1 \oplus 0 = 0 \oplus 1 = 1, 1 \oplus 1 = 0 \oplus 0 = 0$.

$x \oplus y = 57_{(16} \oplus 83_{(16} = D4_{(16}$,

ja que: $01010111_{(2} \oplus 10000011_{(2} = 11010100_{(2} = D4_{(16}$.

D'altra banda, pel "producte" tenim:
$x \otimes y = 57_{(16} \otimes 83_{(16} = C1_{(16}$,

ja que:

$$(x^6+x^4+x^2+x+1)\cdot(x^7+x+1) \pmod{x^8+x^4+x^3+x+1} =$$

$$= (x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1) \pmod{x^8+x^4+x^3+x+1} =$$

$$= x^7+x^6+1 = 11000001_{(2} = C1_{(16}.$$

Un cop vistes aquestes representacions, ja podem passar a veure el funcionament de l'algorisme.

4.2.2 Detall d'una iteració

En el gràfic del funcionament general de l'algorisme es mostra com l'AES realitza, primer, una transformació inicial del text d'entrada, aplicant la funció AddRoundKey. Després, s'executen $n-1$ iteracions, cadascuna de les quals aplica les funcions ByteSub, ShiftRow, MixColumn i AddRoundKey. Finalment, es realitza una transformació final que executa tres de les quatre funcions anteriors, deixant d'aplicar la funció MixColumn.

A més d'aquestes operacions, en la transformació inicial el text en clar s'ha de convertir en una matriu d'estat, que serà utilitzada per la funció AddRoundKey. De manera similar, la transformació final transforma la sortida de la funció AddRoundKey (que és una matriu d'estat) en el text xifrat final.

Passem a descriure cada una de les funcions que s'executen en cada iteració.

4.2.3 Funció AddRoundKey

La funció AddRoundKey s'utilitza tant en les transformacions inicial i final com en les iteracions estàndard.

La funció **AddRoundKey** fa una suma XOR de la matriu d'estat amb cada byte de la subclau $K(i)$ corresponent. En el cas de la transformació inicial tenim, $i = 0$; per tant, utilitzem la primera subclau $K(0)$.

Les subclaus

L'índex i denota la subclau de 128 bits que es fa servir en la i-èsima iteració tenint en compte que $K(0)$ serà la subclau que es farà servir per a la transformació inicial. Podeu trobar la descripció de com s'obtenen les subclaus a partir de la clau inicial de xifratge en el subapartat 4.2.7 d'aquest capítol.

Exemple 4.2 Càlcul de la funció AddRoundKey

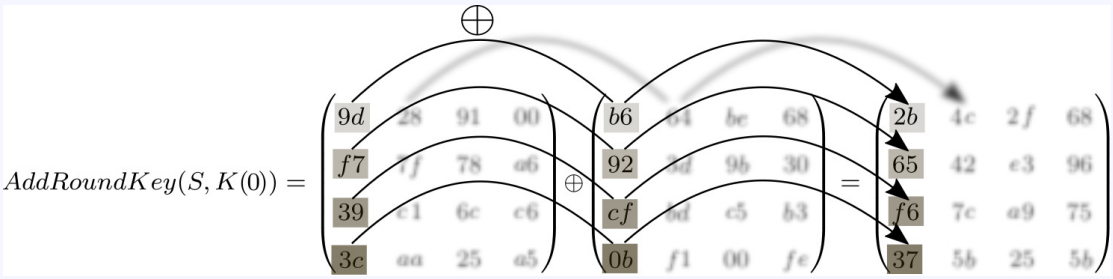
Considerem la subclau: $K(0) = b692cf0b643dbdf1be9bc5006830b3fe$

i la matriu d'estat $S = \begin{pmatrix} 9d & 28 & 91 & 00 \\ f7 & 7f & 78 & a6 \\ 39 & c1 & 6c & c6 \\ 3c & aa & 25 & a5 \end{pmatrix}$

El resultat d'aplicar la funció AddRoundKey serà:

$$AddRoundKey(S,K(0)) = \begin{pmatrix} 9d & 28 & 91 & 00 \\ f7 & 7f & 78 & a6 \\ 39 & c1 & 6c & c6 \\ 3c & aa & 25 & a5 \end{pmatrix} \oplus \begin{pmatrix} b6 & 64 & be & 68 \\ 92 & 3d & 9b & 30 \\ cf & bd & c5 & b3 \\ 0b & f1 & 00 & fe \end{pmatrix} = \begin{pmatrix} 2b & 4c & 2f & 68 \\ 65 & 42 & e3 & 96 \\ f6 & 7c & a9 & 75 \\ 37 & 5b & 25 & 5b \end{pmatrix}$$

Fixeu-vos que la suma XOR de les matrius correspon a la suma XOR de cada una de les seves entrades. Així, per exemple, la primera posició de la transformació val $2B$, ja que $9D \oplus B6 = 10011101 \oplus 10110110 = 2B$.



4.2.4 Funció ByteSub

La funció **ByteSub** aplica una substitució no lineal dels bytes de la matriu d'estat.

La funció ByteSub[2] rep com a entrada una matriu d'estat A, hi aplica una transformació S i obté una altra matriu d'estat B, de manera que $b_{ij} = S(a_{ij})$. La transformació de cada byte de la matriu es realitza de manera independent.

Les caixes S de l'AES

Les caixes S de l'AES són una matriu de 256 elements que s'utilitza com una taula de consulta. Normalment es representa com una matriu de 16 files i 16 columnes. Si representem cada byte a processar amb dos caracters hexadecimals xy, aleshores el valor x indica la fila i el valor y la columna de la posició on es troba el byte resultant.

Taula de consulta	Una taula de consulta (en anglès, *lookup table*) és una estructura de dades que substitueix una execució algorísmica per una operació d'indexació. Normalment l'objectiu d'utilitzar taules de consulta és reduir el temps d'obtenció del resultat esperat.

[2]La funció **ByteSub** apareix amb aquesta denominació a la proposta inicial del criptosistema de Rijndael. A la publicació de l'AES en l'estàndard FIP-197, la funció s'anomena SubBytes. Sigui quin sigui el nom que se li doni, en els dos casos és la mateixa funció.

	$x0$	$x1$	$x2$	$x3$	$x4$	$x5$	$x6$	$x7$	$x8$	$x9$	xa	xb	xc	xd	xe	xf
$0y$	63	$7c$	77	$7b$	$f2$	$6b$	$6f$	$c5$	30	1	67	$2b$	fe	$d7$	ab	76
$1y$	ca	82	$c9$	$7d$	fa	59	47	$f0$	ad	$d4$	$a2$	af	$9c$	$a4$	72	$c0$
$2y$	$b7$	fd	93	26	36	$3f$	$f7$	cc	34	$a5$	$e5$	$f1$	71	$d8$	31	15
$3y$	4	$c7$	23	$c3$	18	96	5	$9a$	7	12	80	$e2$	eb	27	$b2$	75
$4y$	9	83	$2c$	$1a$	$1b$	$6e$	$5a$	$a0$	52	$3b$	$d6$	$b3$	29	$e3$	$2f$	84
$5y$	53	$d1$	0	ed	20	fc	$b1$	$5b$	$6a$	cb	be	39	$4a$	$4c$	58	cf
$6y$	$d0$	ef	aa	fb	43	$4d$	33	85	45	$f9$	2	$7f$	50	$3c$	$9f$	$a8$
$7y$	51	$a3$	40	$8f$	92	$9d$	38	$f5$	bc	$b6$	da	21	10	ff	$f3$	$d2$
$8y$	cd	$0c$	13	ec	$5f$	97	44	17	$c4$	$a7$	$7e$	$3d$	64	$5d$	19	73
$9y$	60	81	$4f$	dc	22	$2a$	90	88	46	ee	$b8$	14	de	$5e$	$0b$	db
ay	$e0$	32	$3a$	$0a$	49	6	24	$5c$	$c2$	$d3$	ac	62	91	95	$e4$	79
by	$e7$	$c8$	37	$6d$	$8d$	$d5$	$4e$	$a9$	$6c$	56	$f4$	ea	65	$7a$	ae	8
cy	ba	78	25	$2e$	$1c$	$a6$	$b4$	$c6$	$e8$	dd	74	$1f$	$4b$	bd	$8b$	$8a$
dy	70	$3e$	$b5$	66	48	3	$f6$	$0e$	61	35	57	$b9$	86	$c1$	$1d$	$9e$
ey	$e1$	$f8$	98	11	69	$d9$	$8e$	94	$9b$	$1e$	87	$e9$	ce	55	28	df
fy	$8c$	$a1$	89	$0d$	bf	$e6$	42	68	41	99	$2d$	$0f$	$b0$	54	bb	16

Exemple 4.3 Càlcul de la funció ByteSub

$$S = \begin{pmatrix} b5 & b1 & b9 & b5 \\ c9 & cc & c5 & c8 \\ 17 & 11 & 1b & 15 \\ 9e & 99 & 92 & 9d \end{pmatrix}$$

Calculem la transformació de la primera entrada de la matriu, $S_{00} = b5$. Busquem el valor de primera component, b a les files de la taula de les caixes S, i el valor de la segona component 5 a les columnes. Això ens indica que el valor que hi ha a la intersecció serà el valor resultant, en aquest cas el $d5$. Si fem el mateix procés amb tots els elements de la matriu tenim com a resultat:

$$ByteSub(S) = \begin{pmatrix} d5 & c8 & 56 & d5 \\ dd & 4b & a6 & e8 \\ f0 & 82 & af & 59 \\ 0b & ee & 4f & 5e \end{pmatrix}$$

4.2.5 Funció ShiftRow

La funció **ShiftRow** desplaça les files de la matriu d'estat de manera que la fila zero es deixa igual, la fila 1 es desplaça una posició a l'esquerra, la fila 2 es desplaça dues posicions a l'esquerra i la fila 3, tres posicions a l'esquerra.

Exemple 4.4 Càlcul de la funció ShiftRow

Si suposem la matriu d'estat:

$$S = \begin{pmatrix} d5 & c8 & 56 & d5 \\ dd & 4b & a6 & e8 \\ f0 & 82 & af & 59 \\ 0b & ee & 4f & 5e \end{pmatrix}$$

Podem realitzar el càlcul de la funció ShiftRow tal com es mostra a la figura següent, deixant la fila zero de la matriu sense modificar i desplaçant les files 1, 2 i 3, una, dues i tres posicions, respectivament:

$$ShiftRow(S) = ShiftRow\left(\begin{pmatrix} d5 & e8 & 56 & d5 \\ dd & 4b & a6 & e8 \\ f0 & 82 & af & 59 \\ 0b & ee & 4f & 5e \end{pmatrix}\right) = \begin{pmatrix} d5 & c8 & 56 & d5 \\ 4b & a6 & e8 & dd \\ af & 59 & f0 & 82 \\ 5e & 0b & ee & 4f \end{pmatrix}$$

La matriu d'estat resultant de la transformació serà doncs:

$$ShiftRow(S) = \begin{pmatrix} d5 & c8 & 56 & d5 \\ 4b & a6 & e8 & dd \\ af & 59 & f0 & 82 \\ 5e & 0b & ee & 4f \end{pmatrix}$$

4.2.6 Funció MixColumns

La funció **MixColumns** barreja les columnes de la matriu d'estat a partir d'operacions polinomials.

Concretament, aquesta funció considera les columnes de la matriu d'estat com polinomis de grau 3. Cada columna es multiplica pel polinomi $c(x) =$ "03"$x^3 +$ "01"$x^2 +$ "01"$x +$ "02" i el resultat es redueix mòdul $x^4 + 1$. Aquest producte dels polinomis es pot escriure com un producte de matrius:

$$\begin{pmatrix} s'_{0j} \\ s'_{1j} \\ s'_{2j} \\ s'_{3j} \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} s_{0j} \\ s_{1j} \\ s_{2j} \\ s_{3j} \end{pmatrix}$$

Tingueu en compte que les operacions "suma" i "producte" entre els elements de la matriu i els del vector columna són les operacions $\oplus$ i $\otimes$ definides en el subapartat anterior.

El polinomi $c(x)$ és coprimer amb $x^4 + 1$ i, per tant, invertible. D'aquesta manera, l'operació MixColumns es pot desfer multiplicant cada columna per el polinomi $d(x)$ tal que: $c(x) \otimes d(x) = a(x)$

El polinomi $d(x)$ és doncs "0B"$x^3 +$ "0D"$x^2 +$ "09"$x +$ "0E".

Exemple 4.5 Càlcul de la funció MixColumns

Suposem una matriu d'estat: $S = \begin{pmatrix} d5 & c8 & 56 & d5 \\ 4b & a6 & e8 & dd \\ af & 59 & f0 & 82 \\ 5e & 0b & ee & 4f \end{pmatrix}$

Per a obtenir la transformació de la primera columna calcularem:

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} d5 \\ 4b \\ af \\ 5e \end{pmatrix}$$

Això ens donarà un vector columna de quatre bytes determinats pels valors següents:

$$\begin{pmatrix} (02 \otimes d5) \oplus (03 \otimes 4b) \oplus (01 \otimes af) \oplus (01 \otimes 5e) \\ (01 \otimes d5) \oplus (02 \otimes 4b) \oplus (03 \otimes af) \oplus (01 \otimes 5e) \\ (01 \otimes d5) \oplus (01 \otimes 4b) \oplus (02 \otimes af) \oplus (03 \otimes 5e) \\ (03 \otimes d5) \oplus (01 \otimes 4b) \oplus (01 \otimes af) \oplus (02 \otimes 5e) \end{pmatrix}$$

Per exemple, vegem quant val la segona posició del vector columna:

$$(01 \otimes d5) \oplus (02 \otimes 4b) \oplus (03 \otimes af) \oplus (01 \otimes 5e)$$

Si passem els valors hexadecimals a representació polinòmica (passant per la seva representació binària) tenim:

Hexadecimal	Binari	Polinomi
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 ara fem els càlculs, resulta:

$$(\text{“}01\text{”} \otimes \text{“}D5\text{”}) = (1)(x^7+x^6+x^4+x^2+1) \pmod{x^8+x^4+x^3+x+1} = x^7+x^6+x^4+x^2+1 \rightarrow 11010101$$

$$(\text{“}02\text{”} \otimes \text{“}4B\text{”}) = (x)(x^6+x^3+x+1) \pmod{x^8+x^4+x^3+x+1} = x^7+x^4+x^2+x \rightarrow 10010110$$

$$(\text{“}03\text{”} \otimes \text{“}AF\text{”}) = (x+1)(x^7+x^5+x^3+x^2+x+1) \pmod{x^8+x^4+x^3+x+1} = x^7+x^6+x^5+x^3+x \rightarrow 11101010$$

$$(\text{“}01\text{”} \otimes \text{“}5E\text{”}) = (1)(x^6+x^4+x^3+x^2+x) \pmod{x^8+x^4+x^3+x+1} = x^6+x^4+x^3+x^2+x \rightarrow 01011110$$

Finalment, fem la XOR:

$$11010101 \oplus 10010110 \oplus 11101010 \oplus 01011110 \oplus 11110111 \rightarrow f7$$

Concretament, el resultat de tots els elements de la primera columna és:

$$\begin{pmatrix} (02 \otimes d5) \oplus (03 \otimes 4b) \oplus (01 \otimes af) \oplus (01 \otimes 5e) \\ (01 \otimes d5) \oplus (02 \otimes 4b) \oplus (03 \otimes af) \oplus (01 \otimes 5e) \\ (01 \otimes d5) \oplus (01 \otimes 4b) \oplus (02 \otimes af) \oplus (03 \otimes 5e) \\ (03 \otimes d5) \oplus (01 \otimes 4b) \oplus (01 \otimes af) \oplus (02 \otimes 5e) \end{pmatrix} = \begin{pmatrix} 9d \\ f7 \\ 39 \\ 3c \end{pmatrix}$$

I el resultat de la funció MixColumns sobre tota la matriu d'estat és:

$$MixColumns(S) = \begin{pmatrix} 9d & 28 & 91 & 00 \\ f7 & 7f & 78 & a6 \\ 39 & c1 & 6c & c6 \\ 3c & aa & 25 & a5 \end{pmatrix}$$

4.2.7 Generació de subclaus

A l'igual que la majoria de criptosistemes en bloc, l'algorisme de Rijndael treballa amb diferents subclaus en cada iteració. Aquestes subclaus s'obtenen per l'aplicació d'una funció d'ampliació a la clau de xifratge inicial.

La funció d'expansió genera, a partir de les Nk paraules de 32 bits de clau de xifratge, $K = (K_0, K_1, \ldots, K_{Nk-1})$, una clau estesa $W = (W_0, W_1, \ldots, W_{4(Nr+1)-1})$ que conté $4(Nr+1)$ paraules de 32 bits. Cada iteració de

l'algorisme de xifrat farà servir 4 paraules de 32 bits i caldran 4 paraules addicionals per a la inicialització. Si denotem per $K(i)$ cada una de les subcadenes de W de 4 paraules de 32 bits tindrem que $K(i)$ és la subclau que s'utilitza en la i-èssima iteració. Gràficament les subclaus de cada iteració en relació amb la clau estesa es poden expressar com:

$$W = (\underbrace{W_0, W_1, W_2, W_3,}_{K(0)} \underbrace{W_4, W_5, W_6, W_7,}_{K(1)} \quad \ldots \quad \underbrace{W_{4Nr}, \ldots, W_{4(Nr+1)-1}}_{K(Nr)})$$

Els paràmetres Nk **i** Nr

Recordem que els paràmetres (Nk, Nr), que representen respectivament la mida de la clau en paraules de 32 bits i el número d'iteracions, poden prendre els valors $(4, 10)$, $(6, 12)$ i $(8, 14)$.

Així, la transformació inicial utilitza la subclau $K(0)$ formada per les primeres 4 paraules de W i en cada una de les Nr iteracions s'utilitzen 4 paraules. D'aquesta manera, per valors d'Nk de 4, 6 i 8 es generaran, respectivament, claus exteses W de 44, 52 i 60 paraules de 32 bits (que corresponen a 1408, 1664 i 1920 bits).

L'algorisme d'expansió de clau consta de dues fases:

- Fase d'inicialització, on la clau de xifratge és copia integrament a les primeres posicions de la clau extesa. És a dir:

$$W_i = K_i, \forall i = 0, \ldots, Nk-1$$

- Fase d'expansió, on s'agafa l'última paraula calculada i s'extén. L'algorisme que implementa aquesta fase queda descrit pel següent pseudocodi:

```
for (i = Nk ; i < 4(Nr + 1); i++)
        temp = W_{i-1}
        if i = 0 mod Nk then
                temp = SubWord(RotWord(temp)) ⊕ Rcon[i/Nk]
        else if ((Nk > 6) and (i mod Nk = 4)) then
                temp = SubWord(temp)
        endif
        W_i = W_{i-Nk} ⊕ temp
```

La fase d'expansió fa servir dues funcions: SubWord i RotWord. La funció SubWord és la mateixa funció que ByteSub (definida anteriorment). La funció RotWord simplement fa una permutació cíclica a la paraula de 4 bytes, és a dir, si tenim $[a0, a1, a2, a3]$ com a entrada, la sortida serà $[a1, a2, a3, a0]$. D'altra banda també es fa servir la constant $Rcon[i]$ que val $Rcon[i] = [x^{i-1}, \text{“00”}, \text{“00”}, \text{“00”}]$. Recordeu que x en hexadecimal val “02” ja que correspon a la representació en binari de 00000010.

L'esquema següent resumeix el procés d'expansió de claus per al cas $Nk = 4$, és a dir, per a claus de 128 bits. En aquest cas, si considerem la clau $K = (K_0 K_1 K_2 K_3)$, aleshores els valors $W_0 \ldots W_3$ contindrien la clau inicial K, i la resta de valors (fins a W_{43}) es calcularien en funció d'aquestes quatre paraules inicials.

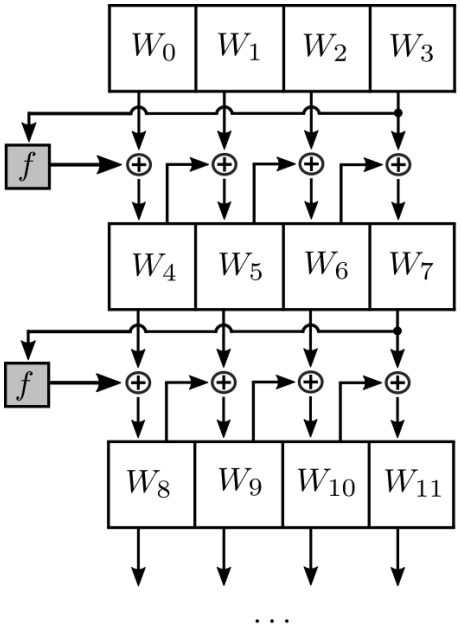


Figura 4.9: Esquema d'expansió de claus de l'AES per a $Nk = 4$

Noteu que l'esquema inclou la funció f, que correspon a aplicar $SubWord(RotWord(temp)) \oplus Rcon[i/Nk]$ sobre el valor que es rep a l'entrada.

Exemple 4.6 Càlcul de l'expansió de claus

Suposem que la longitud de la clau és de 128 bits, és a dir, $Nk = 4$ paraules de 32 bits i que la clau de xifrat (representada en hexadecimal[a]) correspon a:

$$K = \underbrace{\texttt{00 01 02 03}}_{K_0}\ \underbrace{\texttt{04 05 06 07}}_{K_1}\ \underbrace{\texttt{08 09 0A 0B}}_{K_2}\ \underbrace{\texttt{0C 0D 0E 0F}}_{K_3}$$

Amb aquests paràmetres tenim que el nombre d'iteracions és $Nr = 10$. Això vol dir que la clau extesa W tindrà $4 \cdot (10+1) = 44$ paraules de 32 bits.

Denotant per $K(i)$ la clau que es fa servir a l'i-èssima iteració. Els primers bytes de la clau extesa són els mateixos que els de la clau de xifratge:
$W_0 = \texttt{00 01 02 03}$
$W_1 = \texttt{04 05 06 07}$
$W_2 = \texttt{08 09 0A 0B}$
$W_3 = \texttt{0C 0D 0E 0F}$
Per tant:
$K(0) = W_0W_1W_2W_3 = \texttt{00010203 04050607 08090A0B 0C0D0E0F} = K$ Aquestes quatre paraules són les que es fan servir en la transformació inicial de l'algorisme.

La segona subclau serà:

$$
\begin{aligned}
W_4 &= W_0 \oplus SubWord(RotWord(W_3)) \oplus Rcon[1] \\
&\quad SubWord(RotWord(W_3)) = RotWord(\texttt{0C 0D 0E 0F}) = \texttt{0D 0E 0F 0C} \\
&\quad SubWord(\texttt{0D 0E 0F 0C}) = (\texttt{D7 AB 76 FE}) \\
&\quad W_4 = \texttt{00 01 02 03} \oplus \texttt{D7 AB 76 FE} \oplus \texttt{01 00 00 00} = \texttt{D6 AA 74 FD} \\
W_5 &= W_1 \oplus W_4 = \texttt{04 05 06 07} \oplus \texttt{D6 AA 74 FD} = \texttt{D2 AF 72 FA} \\
W_6 &= W_2 \oplus W_5 = \texttt{08 09 0A 0B} \oplus \texttt{D2 AF 72 FA} = \texttt{DA A6 78 F1} \\
W_7 &= W_3 \oplus W_6 = \texttt{0C 0D 0E 0F} \oplus \texttt{DA A6 78 F1} = \texttt{D6 AB 76 FE}
\end{aligned}
$$

Per tant, la subclau $K(1) = \texttt{D6 AA 74 FD D2 AF 72 FA DA A6 78 F1 D6 AB 76 FE}$.
La resta de la clau ampliada es calcula de la mateixa manera.

[a] Recordeu que cada caràcter hexadecimal permet representar 4 bits (és a dir, valors des de 0 fins a 15).

Exercici 4.6 Suposem que la clau de xifratge de 192 bits d'un xifrador AES expressada en hexadecimal és la següent:
`8E 73 B0 F7 DA 0E 64 52 C8 10 F3 2B 80 90 79 E5 62 F8 EA D2 52 2C 6B 7B`. Doneu-ne les dues primeres subclaus, és a dir, $K(0)$ i $K(1)$.

Exercici 4.7 Donat un xifrador Rijndael amb clau de xifratge K i un bloc de text per xifrar 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
```

Quantes iteracions cal fer per xifrar aquest bloc de text en clar amb aquesta clau? Quina és la matriu d'estat a l'inici de la segona iteració?

4.2.8 Desxifrat

En el subapartats anteriors hem definit amb tot detall les operacions de xifratge de l'AES. Totes les funcions que s'utilitzen en el procés de xifratge (ByteSub, ShiftRow, MixColumn i AddRoundKey) són invertibles i, per tant, se'n pot definir la corresponent funció inversa.

Si les funcions definides en el xifratge s'apliquen en l'ordre oposat al que s'executen en el procés de xifratge, obtenim el procés de desxifratge del criptosistema.

4.3 Resum

En aquest capítol hem descrit el funcionament i les característiques principals dels esquemes de xifratge de flux i de bloc.

Pel que fa al xifratge de flux, hem estudiat les propietats que ha de tenir una seqüència aleatòria perquè es pugui utilitzar com a seqüència de xifratge. Hem presentat igualment diferents tipus de generadors per a obtenir seqüències pseudoaleatòries. Hem assenyalat que els registres de desplaçament realimentats linealment (LFSR) eren els més interessants perquè són fàcils d'estudiar, tot i que, com ja hem apuntat, no n'aconsellem l'aplicació en criptografia perquè la seva criptoanàlisi és força senzilla. Finalment, hem estudiat dos generadors que es fan servir avui en dia en productes habituals, l'A5 i el Trivium.

En relació a les xifres de bloc, en primer lloc n'hem descrit la seva estructura general. Després, hem passat a detallar com es poden fer servir les xifres de bloc per a xifrar textos de mida superior al bloc, descrivint diferents modes d'operació: ECB, CBC, CFB, OFC i CTR. Finalment, hem presentat el criptosistema de bloc més utilitzat avui en dia, l'AES, tot detallant-ne tant l'arquitectura com les funcions internes que fa servir.

4.4 Solucions dels exercicis

Exercici 4.1:

En primer lloc, procedim a separar el missatge en blocs de 2 bits, la mida de bloc de la funció de xifrat:

$m = 10\,01\,10\,01\,00\,11\,00\,00$

Després procedim a aplicar la funció de xifrat a cada bloc individual, i concatenem els resultats:

$c = 00\,11\,00\,11\,01\,10\,01\,01$

Exercici 4.2:

En primer lloc, procedim a separar el missatge en blocs de 2 bits. Després, per cada bloc, realitzem una xor amb el bloc xifrat anterior (fent servir el vector inicial com a bloc xifrat anterior per al primer bloc, M_1). Finalment, apliquem el xifrador de bloc sobre la sortida de la xor. El procés a seguir és doncs:

Bloc	$M_i \oplus C_{i-1}$	$C_i = E(M_i \oplus C_{i-1})$
$M_1 = 10$	$M_1 \oplus C_0 = 10 \oplus 10 = 00$	$E(00) = 01$
$M_2 = 01$	$M_2 \oplus C_1 = 01 \oplus 01 = 00$	$E(00) = 01$
$M_3 = 10$	$M_3 \oplus C_2 = 10 \oplus 01 = 11$	$E(11) = 10$
$M_4 = 01$	$M_4 \oplus C_3 = 01 \oplus 10 = 11$	$E(11) = 10$
$M_5 = 00$	$M_5 \oplus C_4 = 00 \oplus 10 = 10$	$E(10) = 00$
$M_6 = 11$	$M_6 \oplus C_5 = 11 \oplus 00 = 11$	$E(11) = 10$
$M_7 = 00$	$M_7 \oplus C_6 = 00 \oplus 10 = 10$	$E(10) = 00$
$M_8 = 00$	$M_8 \oplus C_7 = 00 \oplus 00 = 00$	$E(00) = 01$

El text xifrat correspon a la concatenació dels blocs xifrats: 0101101000100001.

Exercici 4.3:

En aquest cas, la mida de bloc del criptosistema és de 2 bits, pel que els blocs de text a xifrar poden ser com a molt de 2 bits. Agafem doncs blocs de text a xifrar de 2 bits i procedim a realitzar el procés de xifrat. Particionem el missatge M en blocs de 2 bits, i fem una xor de cada bloc amb el resultat de xifrar el bloc anterior, utilitzant el vector inicial com a bloc anterior per a la primera iteració:

Bloc	$E(C_{i-1})$	$C_i = E(C_{i-1}) \oplus M_i$
$M_1 = 10$	$E(C_0) = E(10) = 00$	$M_1 \oplus E(C_0) = 10 \oplus 00 = 10$
$M_2 = 01$	$E(C_1) = E(10) = 00$	$M_2 \oplus E(C_1) = 01 \oplus 00 = 01$
$M_3 = 10$	$E(C_2) = E(01) = 11$	$M_3 \oplus E(C_2) = 10 \oplus 11 = 01$
$M_4 = 01$	$E(C_3) = E(01) = 11$	$M_4 \oplus E(C_3) = 01 \oplus 11 = 10$
$M_5 = 00$	$E(C_4) = E(10) = 00$	$M_5 \oplus E(C_4) = 00 \oplus 00 = 00$
$M_6 = 11$	$E(C_5) = E(00) = 01$	$M_6 \oplus E(C_5) = 11 \oplus 01 = 10$
$M_7 = 00$	$E(C_6) = E(10) = 00$	$M_7 \oplus E(C_6) = 00 \oplus 00 = 00$
$M_8 = 00$	$E(C_7) = E(00) = 01$	$M_8 \oplus E(C_7) = 00 \oplus 01 = 01$

El text xifrat correspon a la concatenació dels blocs xifrats: 1001011000100001.

Exercici 4.4:

En aquest cas, la mida de bloc del criptosistema és de 2 bits, pel que els blocs de text a xifrar poden ser com a molt de 2 bits. Agafem doncs blocs de text a xifrar de 2 bits i procedim a realitzar el procés de xifrat. Particionem el missatge M en blocs de 2 bits, i fem una xor de cada bloc M_i amb el resultat de xifrar v_i, on $v_i = E(v_{i-1})$, amb $v_0 = VI$:

Bloc	$v_i = E(v_{i-1})$	$C_i = v_i \oplus M_i$
$M_1 = 10$	$v_1 = E(v_0) = E(10) = 00$	$v_1 \oplus M_1 = 00 \oplus 10 = 10$
$M_2 = 01$	$v_2 = E(v_1) = E(00) = 01$	$v_2 \oplus M_2 = 01 \oplus 01 = 00$
$M_3 = 10$	$v_3 = E(v_2) = E(01) = 11$	$v_3 \oplus M_3 = 11 \oplus 10 = 01$
$M_4 = 01$	$v_4 = E(v_3) = E(11) = 10$	$v_4 \oplus M_4 = 10 \oplus 01 = 11$
$M_5 = 00$	$v_5 = E(v_4) = E(10) = 00$	$v_5 \oplus M_5 = 00 \oplus 00 = 00$
$M_6 = 11$	$v_6 = E(v_5) = E(00) = 01$	$v_6 \oplus M_6 = 01 \oplus 11 = 10$
$M_7 = 00$	$v_7 = E(v_6) = E(01) = 11$	$v_7 \oplus M_7 = 11 \oplus 00 = 11$
$M_8 = 00$	$v_8 = E(v_7) = E(11) = 10$	$v_8 \oplus M_8 = 10 \oplus 00 = 10$

El text xifrat correspon a la concatenació dels blocs xifrats: 1000011100101110.

Exercici 4.5:

En aquest cas, com que la mida de bloc és molt petita, farem servir directament un comptador que s'incrementa d'un en un, sense incorporar cap nonce. Noteu que el comptador només té 4 valors, pel que la seqüència és repeteix. En una situació real, cal evitar aquest fet ja que compromet la seguretat del sistema.

Procedim doncs a particionar el missatge M en blocs de 2 bits, i fem una xor de cada bloc M_i amb el resultat de xifrar v_i, on v_i és un comptador cíclic que s'inicia amb el valor 00 i s'incrementa per cada nou bloc a xifrar:

Bloc	$v_i = E(i-1 \mod 4)$	$C_i = v_i \oplus M_i$
$M_1 = 10$	$v_1 = E(00) = 01$	$v_1 \oplus M_1 = 01 \oplus 10 = 11$
$M_2 = 01$	$v_2 = E(01) = 11$	$v_2 \oplus M_2 = 11 \oplus 01 = 10$
$M_3 = 10$	$v_3 = E(10) = 00$	$v_3 \oplus M_3 = 00 \oplus 10 = 10$
$M_4 = 01$	$v_4 = E(11) = 10$	$v_4 \oplus M_4 = 10 \oplus 01 = 11$
$M_5 = 00$	$v_5 = E(00) = 01$	$v_5 \oplus M_5 = 01 \oplus 00 = 01$
$M_6 = 11$	$v_6 = E(01) = 11$	$v_6 \oplus M_6 = 11 \oplus 11 = 00$
$M_7 = 00$	$v_7 = E(10) = 00$	$v_7 \oplus M_7 = 00 \oplus 00 = 00$
$M_8 = 00$	$v_8 = E(11) = 10$	$v_8 \oplus M_8 = 10 \oplus 00 = 10$

El text xifrat correspon a la concatenació dels blocs xifrats: 1110101101000010.

Exercici 4.6:

Atès que la clau de xifratge és de 192 bits, el nombre de paraules de 32 bits de la clau val $Nk = 6$; per tant, haurem d'aplicar l'algorisme per al cas $Nk \leq 6$.

Els primers bits de la clau estesa són exactament els mateixos bits de la clau de xifratge:

$W_0 =$ `8E 73 B0 F7`
$W_1 =$ `DA 0E 64 52`
$W_2 =$ `C8 10 F3 2B`
$W_3 =$ `80 90 79 E5`
$W_4 =$ `62 F8 EA D2`
$W_5 =$ `52 2C 6B 7B`

Per tant:

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

Si apliquem l'algorisme per al cas $Nk \leq 6$ amb els valors W_i anteriors obtenim:

$$\begin{aligned}
W_6 =& W_0 \oplus SubWord(RotWord(W_5)) \oplus Rcon[1] \\
& RotWord(W_5) = RotWord(\texttt{52 2C 6B 7B}) = \texttt{2C 6B 7B 52} \\
& SubWord(\texttt{2C 6B 7B 52}) = (\texttt{71 7F 21 00}) \\
W_6 =& \texttt{8E 73 B0 F7} \oplus \texttt{71 7F 21 00} \oplus \texttt{01 00 00 00} = \\
=& \texttt{8E 73 B0 F7} \oplus \texttt{70 7F 21 00} = \texttt{FE 0C 91 F7} \\
W_7 =& W_1 \oplus W_6 = \texttt{DA 0E 64 52} \oplus \texttt{FE 0C 91 F7} = \texttt{24 02 F5 A5} \\
W_8 =& W_2 \oplus W_7 = \texttt{C8 10 F3 2B} \oplus \texttt{24 02 F5 A5} = \texttt{EC 12 06 8E} \\
W_9 =& W_3 \oplus W_8 = \texttt{80 90 79 E5} \oplus \texttt{EC 12 06 8E} = \texttt{6C 82 7F 6B} \\
W_{10} =& W_4 \oplus W_9 = \texttt{62 F8 EA D2} \oplus \texttt{6C 82 7F 6B} = \texttt{0E 7A 95 B9} \\
W_{11} =& W_5 \oplus W_{10} = \texttt{52 2C 6B 7B} \oplus \texttt{0E 7A 95 B9} = \texttt{5C 56 FE C2}
\end{aligned}$$

Per tant, la subclau:
$K(1) = \texttt{FE 0C 91 F7 24 02 F5 A5 EC 12 06 8E 6C 82 7F 6B 0E 7A 95 B9 5C 56 FE C2}$.

Exercici 4.7:

Caldrà fer deu iteracions per a xifrar aquest bloc de text en clar, ja que tant la longitud de la clau és de 16 bytes; per tant, $Nk = 4$.

En la transformació inicial s'aplica la transformació addRoundKey. En el nostre cas:

$$AddRoundKey(S,K(0)) = \begin{pmatrix} 32 & 88 & 31 & e0 \\ 43 & 5a & 31 & 37 \\ f6 & 30 & 98 & 07 \\ a8 & 8d & a2 & 34 \end{pmatrix} \oplus \begin{pmatrix} 2b & 28 & ab & 09 \\ 7e & ae & f7 & cf \\ 15 & d2 & 15 & 4f \\ 16 & a6 & 88 & 3c \end{pmatrix} = \begin{pmatrix} 19 & a0 & 9a & e9 \\ 3d & f4 & c6 & f8 \\ e3 & e2 & 8d & 48 \\ be & 2b & 2a & 08 \end{pmatrix} = S_1$$

El resultat de la primera iteració correspondrà a executar les funcions ByteSub, ShiftRow, MixColumns i AddRoundKey. El resultat de la funció ByteSub sobre la matriu d'estat S_1 és:

$$ByteSub\left(\begin{pmatrix} 19 & a0 & 9a & e9 \\ 3d & f4 & c6 & f8 \\ e3 & e2 & 8d & 48 \\ b3 & 2b & 2a & 08 \end{pmatrix}\right) = \begin{pmatrix} d4 & e0 & b8 & 1e \\ 27 & bf & b4 & 41 \\ 11 & 98 & 5d & 52 \\ ae & f1 & e5 & 30 \end{pmatrix} = S_2$$

El resultat de la funció ShiftRow sobre la matriu d'estat S_2 és:

$$ShiftRow\left(\begin{pmatrix} d4 & e0 & b8 & 1e \\ 27 & bf & b4 & 41 \\ 11 & 98 & 5d & 52 \\ ae & f1 & e5 & 30 \end{pmatrix}\right) = \begin{pmatrix} d4 & e0 & b8 & 1e \\ bf & b4 & 41 & 27 \\ 5d & 52 & 11 & 98 \\ 30 & ae & f1 & e5 \end{pmatrix} = S_3$$

El resultat de la funció MixColumns sobre la matriu d'estat S_3 és:

$$MixColumns\left(\begin{pmatrix} d4 & e0 & b8 & 1e \\ bf & b4 & 41 & 27 \\ 5d & 52 & 11 & 98 \\ 30 & ae & f1 & e5 \end{pmatrix}\right) = \begin{pmatrix} 04 & e0 & 48 & 28 \\ 66 & cb & f8 & 06 \\ 81 & 19 & d3 & 26 \\ e5 & 9a & 7a & 4c \end{pmatrix} = S_4$$

Ara ens cal calcular el valor de la clau de la segona iteració, és a dir, el valor $K(1)$. Per fer-ho, aplicarem l'algorisme d'expansió de claus descrit en l'apartat 4.2.7, que ens permetrà obtenir la matriu:

$$\begin{pmatrix} a0 & 88 & 23 & 2a \\ fa & 54 & a3 & 6c \\ fe & 2c & 39 & 76 \\ 17 & b1 & 39 & 05 \end{pmatrix}$$

Una descripció gràfica per a la generació d'aquesta subclau la podeu trobar en aquest enllaç.

Finalment, el resultat de la funció AddRoundKey sobre la matriu d'estat S_4 resulta:

$$\begin{pmatrix} 04 & e0 & 48 & 28 \\ 66 & cb & f8 & 06 \\ 81 & 19 & d3 & 26 \\ e5 & 9a & 7a & 4c \end{pmatrix} \oplus \begin{pmatrix} a0 & 88 & 23 & 2a \\ fa & 54 & a3 & 6c \\ fe & 2c & 39 & 76 \\ 17 & b1 & 39 & 05 \end{pmatrix} = \begin{pmatrix} a4 & 68 & 6b & 02 \\ 9c & 9f & 5b & 6a \\ 7f & 35 & ea & 50 \\ f2 & 2b & 43 & 49 \end{pmatrix}$$

Així, el valor de la matriu d'estat a l'inici de la segona iteració valdrà:

$$S = \begin{pmatrix} a4 & 68 & 6b & 02 \\ 9c & 9f & 5b & 6a \\ 7f & 35 & ea & 50 \\ f2 & 2b & 43 & 49 \end{pmatrix}$$

4.5 Bibliografia

Christophe De Cannière and Bart Preneel (2005). *Trivium - Specifications*. Technical report.

Joan Daemen and Vincent Rijmen (2002). *The Design of Rijndael, AES - The Advanced Encryption Standard*. Springer-Verlag (238 pp.)

GSMA (2017). *GSMA The Mobile Economy 2017*.
`https://www.gsmaintelligence.com/research/`

James L. Massey (1969). *Shift-register synthesis and BCH decoding*. IEEE transactions on Information Theory, 15(1), 122-127.

Alfred Menezes, Paul van Oorschot, and Scott Vanstone (1996). *Handbook of Applied Cryptography*. CRC Press.
`http://cacr.uwaterloo.ca/hac/`

NIST (2001). *Advanced Encryption Standard (AES)*. Federal Information Processing Standards Publication 197.
`http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf`

Christof Paar and Jan Pelzl (2009). *Understanding Cryptography: A Textbook for Students and Practitioners*. Springer.

Andrew Rukhin,Juan Soto, James Nechvatal, et al. (2010). *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications* . NIST Special Publication.
`http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf`

Thomas Stockinger (2005). *GSM network and its privacy - the A5 stream cipher*.