Criptografia de clau simètrica

Criptosistemes de flux i de bloc

PID_00235161

Jordi Herrera Joancomartí
Cristina Pérez Solà

Temps mínim de dedicació recomanat: 4 hores

Universitat Oberta de Catalunya

Jordi Herrera Joancomartí

Llicenciat en Matemàtiques per la Universitat Autònoma de Barcelona i doctor per la Universitat Politècnica de Catalunya. Els seus àmbits de recerca són la criptografia, les criptomonedes i la tecnologia *blockchain*. Ha publicat nombrosos textos docents i més de cent articles de recerca en revistes i congressos nacionals i internacionals. Ha dirigit nou tesis doctorals i ha estat investigador principal de diversos projectes de recerca nacionals. Ha participat com a avaluador per a agències de recerca de diversos països europeus i també per a la Comissió Europea. Actualment és professor agregat del departament d'Enginyeria de la Informació i les Comunicacions a la Universitat Autònoma de Barcelona.

Cristina Pérez Solà

Doctora en Informàtica per la Universitat Autònoma de Barcelona i la Universitat Catòlica de Lovaina. Actualment és professora dels Estudis d'Informàtica, Multimèdia i Telecomunicacions de la Universitat Oberta de Catalunya. Els seus àmbits de recerca són les criptomonedes basades en *blockchain* i, en especial, els aspectes relacionats amb la seguretat i la privadesa d'aquestes. També està interessada en els problemes de privacitat que sorgeixen arran de l'ús de les xarxes socials i en l'adaptació de tècniques de mineria de dades a la naturalesa específica d'aquest tipus de xarxes.

L'encàrrec i la creació d'aquest recurs d'aprenentatge UOC han estat coordinats per la professora: Helena Rifà Pous

Segona edició: febrer 2021

Av. Tibidabo, 39-43, 08035 Barcelona
Autoria: Jordi Herrera Joancomartí, Cristina Pérez Solà
Producció: FUOC

Índex

Introducció

Ja hem vist en anteriors mòduls didàctics que un criptosistema incondicionalment segur necessita tants bits de clau com bits de text per xifrar. El xifratge de Vernam és el criptosistema que aconsegueix aquesta seguretat incondicional, però el preu que en paga és la ineficiència del xifratge. Aquesta ineficiència recau justament en el fet que la clau ha de tenir la mateixa longitud del text que es vol xifrar. Això comporta que la longitud de les claus sigui molt gran i, per tant, sigui més difícil guardar-les en secret. A més, es dona la paradoxa que si tenim un canal segur per a intercanviar les claus, aleshores també podem utilitzar-lo per a intercanviar els missatges, ja que tenen la mateixa longitud.

Les **xifres de flux** sorgeixen com una aproximació optimitzada al xifratge de Vernam. La idea és construir una clau suficientment llarga (com a mínim de la longitud del missatge) a partir d'una clau inicial curta. Això s'aconsegueix utilitzant el que s'anomena un generador pseudoaleatori. Aquest generador expandeix una clau petita, anomenada llavor; així se n'obté una de molt més llarga. L'operació d'expansió cal que tingui certes característiques, ja que la seqüència que en resultarà és la que s'utilitzarà per a xifrar el text en clar. Caldrà, doncs, veure quines propietats hauran de complir aquestes seqüències i estudiar quin tipus de generadors hi ha per obtenir-les.

Les xifres de flux són xifres de clau compartida, ja que la llavor (que es fa servir per obtenir la seqüència xifrant) és utilitzada tant per a xifrar com per a desxifrar, i és, per tant, compartida entre l'emissor i el receptor.

Una alternativa al xifratge de flux és el que s'anomena **xifratge de bloc**. Aquest xifratge s'inclou 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 de bloc és la utilització de memòria en els algorismes de xifratge.

Com veurem en aquest mòdul, el xifratge de flux utilitza una clau diferent per a 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. 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 xifratge que en resulten són força vulnerables, perquè 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'Advanced Encryption Standard (AES), ja que està establert com a estàndard per l'Institut Nacional d'Estàndards i Tecnologia (NIST) des de l'any 2002.

Objectius

En aquest mòdul s'estudiaran les xifres de flux i les de bloc, que formen part de les xifres de clau compartida. A continuació es fixen els objectius que l'estudiant ha d'assolir:

1. Comprendre l'esquema general de les xifres de flux.
2. Assimilar les característiques necessàries que ha de complir una seqüència pseudoaleatòria per tal que sigui utilitzable en un esquema de xifratge de flux.
3. Entendre el funcionament de diferents generadors pseudoaleatoris.
4. Comprendre l'esquema general de les xifres de bloc i les característiques comunes de tots ells.
5. Entendre els diferents modes d'operació del xifratge de bloc.
6. Conèixer el funcionament del criptosistema de bloc AES.

1. Criptografia de clau simètrica o compartida

En aquest mòdul didàctic introduirem les xifres de clau compartida.

Els **criptosistemes de clau simètrica o compartida** són aquells en els quals l'emissor i el receptor comparteixen una mateixa clau, que és la que utilitzen per a xifrar i desxifrar missatges.

És a dir, en les xifres de clau compartida, la clau que s'utilitza per a xifrar és la mateixa que es fa servir per a desxifrar i, per tant, en qualsevol moment, l'emissor pot passar a fer de receptor i al revés, utilitzant sempre les mateixes claus.

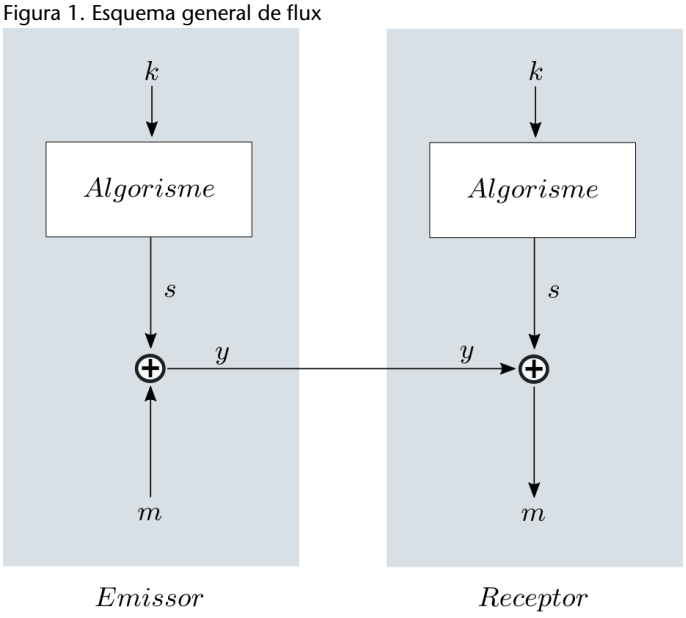
Per les seves característiques, les xifres de clau compartida no poden oferir la propietat de no-repudi. Com veurem més endavant, existeixen altres construccions que sí que ens permetran oferir aquesta propietat.

Els dos tipus de xifres de clau simètrica més utilitzats són les xifres de flux i les de bloc. La diferència entre els dos tipus de xifres radica en com es processa la informació: a les xifres de flux la informació es xifra bit per bit, és a dir, els bits es xifren de manera individual, mentre que els criptosistemes de bloc xifren un bloc sencer de n bits alhora.

2. Les xifres de flux

D'una manera esquemàtica, un **criptosistema de flux** es pot expressar tal com mostra la figura 1.

Figura 1. Esquema general de flux



Tan l'emissor com el receptor disposen d'una mateixa clau *k*, anomenada llavor del generado), i d'un mateix algorisme determinista *Alg*, anomenat **generador pseudoaleatori**. En proporcionar la clau *k* com a entrada a l'algorisme, aquest dona com a sortida una seqüència *s*, que s'anomena **seqüència xifrant**.

Per a xifrar el missatge, l'emissor va sumant cada bit del missatge *m* amb cada bit de la seqüència xifrant *s*, obtenint el missatge xifrat *y*. Quan el receptor rep el missatge xifrat *y*, utilitza el mateix algorisme determinista *Alg* i la clau *k*, que comparteix amb l'emissor, per tal d'obtenir la mateixa seqüència xifrant. Així, sumant bit per bit el missatge que li arriba *y* amb la seqüència resultant de l'algorisme *s*, obté el text en clar *m* enviat per l'emissor. Al llarg de tot aquest mòdul didàctic les seqüències amb què treballarem seran binàries i les operacions a què farem referència seran totes mòdul 2.

Suma mòdul 2

De manera equivalent, podem pensar la suma mòdul 2 com una XOR.

Per tal que aquest criptosistema sigui segur, és bàsic que la seqüència xifrant no sigui coneguda, és a dir, que en cap moment es pugui saber quin serà el bit de sortida següent. Idealment, el que es necessita per a la seguretat incondicional és que la clau, en aquest cas la seqüència xifrant, sigui completament aleatòria. En el nostre esquema no es pot donar aquesta condició, ja que el

generador que utilitzem ha de ser determinista, perquè emissor i receptor obtinguin la mateixa seqüència quan donen com a entrada la mateixa clau secreta. Així doncs, la seqüència xifrant tindrà propietats molt properes a les que té una seqüència completament aleatòria i, per tant, s'anomenarà **seqüència pseudoaleatòria**.

Concretament, si una seqüència no és aleatòria, vol dir que a partir d'un cert moment es repeteix. Aquesta subseqüència que es va repetint és el que s'anomena període. El que és important, doncs, és que aquesta subseqüència, el període, sigui indistingible d'una seqüència completament aleatòria d'igual longitud. Per això aquesta seqüència ha de complir certes propietats que veurem al llarg dels apartats d'aquest mòdul.

No hem d'oblidar que els criptosistemes de clau compartida basen la seva seguretat en el fet que la clau utilitzada per a xifrar i desxifrar només és coneguda per l'emissor i el receptor. En el xifratge en flux, si bé la clau no s'utilitza directament per a xifrar, cal igualment que no es faci pública, ja que l'algorisme determinista és conegut i, per tant, es podria obtenir la seqüència xifrant a partir d'ell i la clau.

Si ens fixem en l'esquema de xifratge en flux de la figura 1 veiem que, per a obtenir el text xifrat que enviem al receptor hem d'anar sumant el text en clar amb la seqüència xifrant que resulta del generador pseudoaleatori. Això vol dir que la velocitat de transmissió de les dades entre l'emissor i el receptor ve determinada pel mínim entre la velocitat de generació del missatge, m, i la velocitat de generació de la seqüència xifrant, s. Així doncs, cal tenir en compte aquest fet quan estudiem els possibles generadors pseudoaleatoris, ja que en funció de la seva implementació (ja sigui en maquinari o en programari), obtindrem una velocitat o una altra. Cal que l'algorisme que ens generi la seqüència sigui de fàcil implementació, tant des del punt de vista de complexitat com pel que fa al vessant econòmic.

No oblidem el món real

Els telèfons mòbils amb tecnologia GSM incorporen un xifrador de flux. Seria impensable que el cost econòmic del xifrador incrementés el preu del telèfon mòbil. Tampoc no seria admissible que la velocitat de la comunicació es veiés afectada per la velocitat d'aquest xifrador.

2.1. Període

Hem vist que per a implementar un criptosistema de xifratge de flux necessitem un algorisme que ens doni com a sortida la seqüència xifrant. Com ja hem dit anteriorment, el fet que aquest algorisme sigui determinista fa que la seqüència que en resulta no sigui completament aleatòria i, per tant, implica que a partir d'un cert moment es repeteix. Aquesta subseqüència que es va repetint és el que s'anomena **període**. Formalment, sigui $\{s_i\}_{i\geq 0}$ una seqüència periòdica, el període p és l'enter més petit:

$$s_{i+p} = s_i$$

Això passa per a tot $i \geq 0$.

Atès que el període es repeteix, una vegada es coneix, ja es pot determinar exactament tota la seqüència xifrant i, per tant, el criptosistema es pot trencar. Per això les seqüències que s'utilitzen per al xifratge en flux cal que tinguin un període molt gran, ja que d'aquesta manera triguen molt a repetir-se i, per tant, és més difícil predir-ne la sortida.

Període gran

El concepte de període gran és relatiu al xifrador i a l'aplicació. Un període de 2^{32} pot no ser prou llarg per a un xifrador que xifri a 1 megabyte/seg, ja que a aquesta velocitat el període es repeteix només cada 8,5 minuts.

2.2. Aleatorietat

Com hem comentat, les xifres de flux fan servir generadors pseudoaleatoris.

Un **generador pseudoaleatori** (o PRNG, de l'anglès *pseudo random number generator*) és un algorisme **determinista** que genera una seqüència a partir d'una entrada que anomenem **llavor**. La seqüència generada per un PRNG intenta reproduir les propietats que tindria una seqüència generada de manera aleatòria.

Determinisme dels PRNG

Noteu que els PRNG són algorismes deterministes, és a dir, donat un PRNG i una llavor, la seqüència generada serà sempre la mateixa.

Els **generadors pseudoaleatoris criptogràficament segurs** (o CSPRNG, de l'anglès *cryptographycally secure pseudo random number generator*) són un tipus especial de PRNG que generen seqüències no predictibles. En concret, perquè un PRNG sigui considerat un CSPRNG, cal que les seqüències que genera tinguin dues propietats. A partir de k bits de la seqüència generada, $s_{i+1}, s_{i+2}, \cdots, s_{i+k}$, podem afirmar:

PRNG i CSPRNG

Tots els CSPRNG són PRNG, però l'afirmació contrària no és certa, és a dir, un generador pseudoaleatori no té per què ser criptogràficament segur.

1) No existeix un algorisme en temps polinomial que pugui predir el següent bit de la seqüència, s_{i+k+1}, amb una probabilitat més alta del 50%.

2) No és computacionalment possible predir el bit anterior de la seqüència, s_i.

El concepte de complexitat lineal ens mesura el grau d'impredictibilitat d'una seqüència. En concret, la complexitat lineal ens diu quina part de la seqüència ens cal conèixer per tal de poder-la predir tota. Per a calcular la complexitat lineal utilitzarem l'algorisme de Massey. La definició formal de la complexitat lineal va lligada al concepte d'LFSR, que presentarem més envadant. Així doncs, detallarem la definició formal de complexitat lineal una vegada haguem introduït l'arquitectura dels LFSR.

Una configuració bastant habitual en criptografia és fer servir CSPRNG amb valors veritablement aleatoris com a llavors. Aconseguir nombres (veritablement) aleatoris no és una tasca senzilla. Per a generar-los cal disposar d'una font d'aleatorietat natural. Addicionalment, si aquesta font d'aleatorietat es vol fer servir en criptografia, caldrà assegurar també que un adversari no és capaç de manipular-la ni observar-la. Existeixen, principalment, dues maneres d'obtenir valors realment aleatoris: per mitjà de maquinari, explotant l'aleatorietat que es produeix en fenòmens físics, o per mitjà de programari, a partir

Soroll tèrmic

S'anomena soroll tèrmic o soroll de Johnson-Nyquist a les fluctuacions elèctriques generades per l'agitació tèrmica dels electrons.

d'observacions afectades pel comportament de l'usuari. Així, per exemple, es pot fer servir el so capturat per un micròfon, el soroll tèrmic d'una resistència o d'un diode, les turbulències creades per l'aire en segons quins dispositius o el moviment del ratolí.

2.2.1. Tests d'aleatorietat del NIST

El National Institute of Standards and Technology (NIST) dels Estats Units disposa d'un banc de proves estadístiques per avaluar l'aleatorietat de seqüències binàries generades per PRNG. El banc consta de quinze testos. En aquest subapartat, descriurem els tests més senzills, amb l'objectiu de donar una idea del que es busca en l'avaluació de l'aleatorietat de seqüències. Per tal de descriure els testos, suposarem que s'avalua la seqüència binària $S = \{s_1, s_2, \cdots, s_n\}$ de mida n bits.

Tests estadístics

El lector interessat en els detalls sobre els tests estadístics pot consultar la publicació original del NIST: **A. Rukhin; J. Soto; J. Nechvatal i altres** (2010). *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*.

El test de **freqüència de bits individuals** comprova que la proporció d'uns i zeros de la seqüència proporcionada s'aproxima a la que observaríem en una seqüència veritablement aleatòria, és a dir, que la proporció d'uns i zeros és similar i s'aproxima, per tant, a 0.5. Per a fer-ho, en primer lloc es transforma la seqüència binària d'entrada en una seqüència de –1 i 1:

$$X = \{x_i \mid x_i = 2 \cdot s_i - 1, \forall i \in [1,n]\}$$

Així doncs, els zeros es converteixen en –1 i els uns segueixen representant-se amb 1.

Després, es calcula s_{obs}:

$$s_{obs} = \frac{\mid \sum_{i=1}^{n} x_i \mid}{\sqrt{n}}$$

Superació dels testos

A partir del valor s_{obs}, els tests del NIST calculen el nivell de significació observat per decidir si la seqüència supera o no la comprovació. En concret, es considera que la seqüència supera la prova si el valor p és més elevat o igual que 0.01.

Si la seqüència és aleatòria, s_{obs} tendirà cap a 0, mentre que si hi ha massa zeros o massa uns en la seqüència, aleshores s_{obs} tendirà a ser més gran que zero.

Exemple de càlcul de la prova de freqüència de bits individuals

Donada la seqüència $S = \{0,1,0,1,0,0,0,0,0,0,0,1,1,0,0,1,1,1,1,0,1,0,1,0,1,0,0,0\}$ amb $n = 28$, procedim a calcular s_{obs}.

En primer lloc, transformem la seqüència de zeros i uns en una seqüència de –1 i 1:

$$X = \{-1,1,-1,1,-1,-1,-1,-1,-1,-1,1,1,-1,-1,1,1,1,1,-1,1,-1,1,-1, \quad 1,-1,-1,-1\}$$

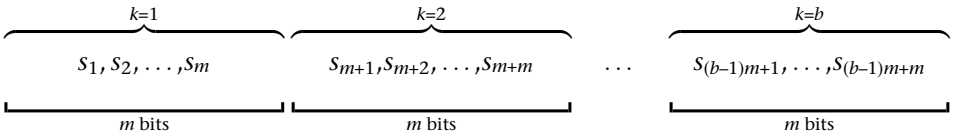
Seguidament, calculem el valor s_{obs}:

$$s_{obs} = \frac{\mid \sum_{i=1}^{28} x_i \mid}{\sqrt{28}} = \frac{\mid -6 \mid}{\sqrt{28}} \approx 1.1339$$

Superació del test

En aquest cas, el nivell de significació per a $s_{obs} = 1.1339$ és de 0.256, fent que el test es consideri superat, ja que $0.256 \geq 0.01$

El test de **freqüència en un bloc** comprova que el número de zeros i uns en un bloc de m bits sigui aproximadament $m/2$. Per fer-ho, es particiona la seqüència que es vol avaluar en $b = \lfloor n/m \rfloor$ blocs de m bits, descartant els bits sobrants.



Aleshores, per a cada bloc k (amb $k = 1, \cdots, b$), es calcula:

$$\pi_k = \frac{\sum_{j=1}^{m} s_{(k-1)m+j}}{m}$$

És a dir, es calcula la proporció d'uns que hi ha a cada bloc.

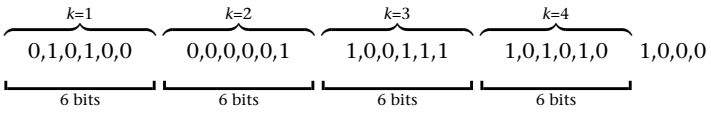
Finalment, es calcula:

$$\chi^2_{obs} = 4m \sum_{k=1}^{b} (\pi_k - 1/2)^2$$

Exemple de càlcul de la prova de freqüència en un bloc

Donada la mateixa seqüència que en l'exemple anterior:
$S = \{0,1,0,1,0,0,0,0,0,0,0,1,1,0,0,1,1,1,1,0,1,0,1,0,1,0,1,0,0,0\}$ amb $n = 28$, procedim a calcular π_k per a cada bloc amb $m = 6$.

En primer lloc, dividim la seqüència en $b = \lfloor n/m \rfloor = \lfloor 28/6 \rfloor = 4$ blocs de $m = 6$ bits.



Els últims 4 bits de la seqüència es descarten i no s'utilizen en el test.

Aleshores, per a cada bloc k (amb $k = 1, \cdots, 4$), calculem π_k:

$$\pi_1 = \frac{\sum_{j=1}^{m} s_{(k-1)m+j}}{m} = \frac{\sum_{j=1}^{6} s_j}{6} = \frac{2}{6} = 1/3$$

$$\pi_2 = \frac{\sum_{j=1}^{6} s_{6+j}}{6} = 1/6$$

$$\pi_3 = \frac{\sum_{j=1}^{6} s_{2\cdot 6+j}}{6} = \frac{4}{6} = 2/3$$

$$\pi_4 = \frac{\sum_{j=1}^{6} s_{3\cdot 6+j}}{6} = \frac{3}{6} = 1/2$$

I finalment estem en disposició de calcular χ^2_{obs}:

$$\chi^2_{obs} = 4m \sum_{k=1}^{b} (\pi_k - 1/2)^2$$

$$= 4 \cdot 6 \sum_{k=1}^{4} (\pi_k - 1/2)^2$$

$$= 24 \cdot \left((1/3 - 1/2)^2 + (1/6 - 1/2)^2 + (2/3 - 1/2)^2 + (1/2 - 1/2)^2 \right)$$

$$= 24 \cdot (1/36 + 1/9 + 1/36 + 0) = 4$$

Superació del test

En aquest cas, el nivell de significació per a $\chi^2_{obs} = 4$ (tenint en compte que tenim $b = 4$ blocs) és de 0.4060, fet que fa que el test es consideri superat, ja que $0.4060 \geq 0.01$.

El test de **ràfegues** comprova si el número de ràfegues tant d'uns com de zeros de la seqüència s'assembla al que trobaríem en una seqüència aleatòria.

Definirem una **ràfega** com un conjunt de bits consecutius iguals, és a dir, una ràfega de longitud k consta dels elements $s_t, \cdots, s_{t+k-1}$, tals que $s_{t-1} \neq s_t = s_{t+1} = \cdots = s_{t+k-1} \neq s_{t+k}$.

Per a avaluar el test de ràfegues, es calcula:

$$V_n(obs) = \left(\sum_{i=1}^{n-1} r(i) \right) + 1$$

en què $r(i)$ és la funció:

$$r(i) = \begin{cases} 0, & \text{si } s_i = s_{i+1} \\ 1, & \text{altrament} \end{cases}$$

Valors grans de V_{obs} indiquen que les oscil·lacions de valors en la seqüència avaluada (és a dir, els canvis d'u a zero o de zero a u) succeeixen ràpidament, mentre que valors petits indiquen que les oscil·lacions són lentes.

Oscil·lacions

La seqüència 10101010 oscil·la molt ràpidament, ja que cada bit canvia el valor respecte al bit anterior. En canvi, la seqüència 11111100 oscil·la molt lentament, ja que només es produeix un canvi de valor en tota la seqüència.

El NIST recomana que les seqüències avaluades amb aquest test tinguin com a mínim 100 bits (és a dir, $n \geq 100$). Addicionalment, aquest test té com a prerequisit que la seqüència passi el test de freqüència de bits individuals que hem descrit anteriorment. És a dir, si una seqüència no supera el test de bits individuals, aleshores ja no es realitza el test de ràfegues.

Exemple de càlcul de la prova de ràfegues

Seguint amb l'avaluació de la mateixa seqüència que en els exemples anteriors: $S = \{0,1,0,1,0,0,0,0,0,0,0,1,1,0,0,1,1,1,1,0,1,0,1,0,1,0,0,0\}$ amb $n = 28$. Noteu que si seguíssim les recomanacions del NIST, no efectuaríem el test de ràfegues sobre aquesta seqüència, ja que aquesta no té una longitud mínima de 100 bits. A tall d'exemple, però, realitzarem els càlculs per a aquesta seqüència.

En primer lloc, comprovem que la seqüència superi el test de freqüència de bits individuals. Com hem vist al primer exemple, la seqüència supera aquest test, així que procedim a calcular $V_{28}(obs)$.

$$V_n(obs) = \left(\sum_{i=1}^{n-1} r(i)\right) + 1$$

$$= (1+1+1+1+1+0+0+0+0+0+0+1+0+1+0+1+0+$$

$$+0+0+1+1+1+1+1+1+1+0+0) + 1 = 15$$

Superació del test

En aquest cas, el nivell de significació per a $V_n(obs) = 15$ (i tenint en compte que la seqüència té 28 bits i una proporció de 11/28 d'uns) és de 0.5151, de manera que el test es considera superat, ja que $0.5151 \geq 0.01$.

Com s'ha comentat, el banc de proves del NIST recull quinze tests diferents. A més dels tres comentats, els altres tests comproven l'aparició de les ràfegues d'uns més llargues, la repetició de subseqüències concretes dins la seqüència, la facilitat de comprimir-la, la seva complexitat lineal o les seves propietats espectrals, entre d'altres.

3. Generadors lineals de seqüència xifrant

A l'apartat 2, hem estudiat les propietats que han de tenir les seqüències xifrants per tal de poder-les utilitzar en criptosistemes de xifratge de flux. Tractem ara com han de ser els algorismes deterministes que generen aquests tipus de seqüències.

Des d'un punt de vista general tenim dos tipus de generadors: els lineals i els no lineals.

Els **generadors lineals** són aquells que només realitzen operacions lineals sobre els elements d'entrada per a obtenir la seqüència de sortida. Contràriament, els **generadors no lineals** són els que realitzen, a més a més, operacions no lineals, com podrien ser permutacions.

3.1. Generadors congruencials

Els **generadors congruencials** es basen en equacions modulars recurrents del tipus:

$$x_n = (ax_{n-1} + b) \bmod m$$

En aquest cas el valor x_0 seria la llavor de la seqüència xifrant. Un criptosistema que utilitzi un generador d'aquest tipus tindrà com a clau secreta els valors $\{x_0,a,b,m\}$ i, per tal que el període sigui màxim, caldrà que compleixi que $\text{mcd}(a,m) = 1$.

Cal dir, però, que aquests tipus de generadors pseudoaleatoris no són segurs des d'un punt de vista criptogràfic, ja que s'ha pogut demostrar que amb pocs valors x_i coneguts ja es poden esbrinar els paràmetres secrets $\{x_0,a,b,m\}$. Fins i tot només amb una part dels bits que formen els x_i, però, això sí, coneixent els paràmetres $\{a,b,m\}$, es pot arribar a determinar el valor de la llavor x_0.

Tot i això, aquests tipus de generadors són molt utilitzats en sistemes informàtics per a aplicacions no criptogràfiques.

Exemple 1

La funció *rand*() del sistema UNIX BSD utilitza el generador congruencial afí següent:

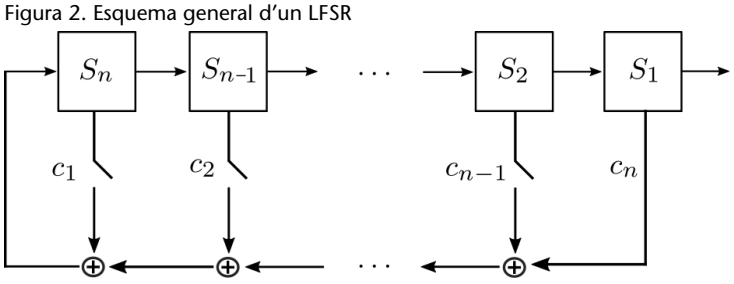
$$x_n = (1103515245x_{n-1} + 12345) \bmod 2^{31}$$

en què la llavor especifica el valor inicial.

3.2. Registres de desplaçament realimentats linealment

Un registre de desplaçament realimentat linealment (o LFSR, de l'anglès *linear feedback shift register*) de longitud n és un dispositiu físic o lògic format per n cel·les de memòria i n portes lògiques que té una estructura com es mostra a la figura 2.

Figura 2. Esquema general d'un LFSR



Inicialment, les cel·les contenen els valors d'entrada, i a cada impuls de rellotge el contingut de la cel·la s_i es desplaça a la cel·la s_{i-1} realitzant les operacions associades. D'aquesta manera, es genera un nou element, s_{n+1}, que és determinat per l'expressió:

$$s_{n+1} = c_1 s_n + \cdots + c_n s_1 \tag{1}$$

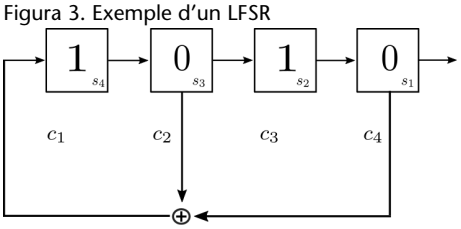
en què els $c_i \in \{0,1\}$ corresponen als valors de les portes lògiques de l'esquema. És a dir, els coeficients seran 1 si hi ha una connexió i 0 si no n'hi ha. Aquest nou element, s_{n+1}, se situa a la cel·la s_n, que ha quedat buida a causa del desplaçament.

El conjunt de valors continguts en cada cel·la en un instant de temps s'anomena **estat**. L'**estat inicial** és l'estat en què es troba l'LFRS en el moment de començar el procés.

Exemple del funcionament d'un LFSR

Aquest exemple segueix el funcionament de l'LFSR de quatre cel·les que trobareu representat a la figura 3.

Figura 3. Exemple d'un LFSR



Com es pot veure, l'estat inicial és 1010, que correspon a l'impuls de rellotge $t = 0$. La taula 1 mostra l'evolució de l'LFSR en els diferents instants de temps.

Taula 1. Evolució de l'LFSR en el temps

Impuls de rellotge (t)	s_4	s_3	s_2	s_1	Sortida
0	1	0	1	0	0
1	0	1	0	1	1
2	0	0	1	0	0
3	0	0	0	1	1
4	1	0	0	0	0
5	0	1	0	0	0
6	1	0	1	0	0
7	0	1	0	1	1
⋮		⋮			⋮

En $t = 0$, les cel·les s_1 i s_3 contenen un 0 i, per tant, el bit s_4 serà 0 al següent impuls de rellotge ($t = 1$). Noteu que la resta de valors es desplacen: a $t = 1$ la cel·la s_1 conté el valor que hi havia en $t = 0$ a la cel·la s_2, la cel·la s_2 conté el valor que hi havia a s_3, etc.

En general, per $i \geq 1$ tenim:

$$s_1(t = i) = s_2(t = i - 1)$$

$$s_2(t = i) = s_3(t = i - 1)$$

$$s_3(t = i) = s_4(t = i - 1)$$

$$s_4(t = i) = s_1(t = i - 1) \oplus s_3(t = i - 1)$$

Si ens fixem en l'impuls de rellotge $t = 6$, tornem a tenir l'estat inicial i, per tant, a partir d'aquí la seqüència es torna a repetir. Aquesta seqüència, doncs, té període 6.

Un cop definit el que és un LFSR podem passar a fer un estudi una mica més exhaustiu per tal de determinar-ne les característiques més importants. L'avantatge principal dels LFSR és que tenen una formulació matemàtica molt simple, com veurem a continuació i, per tant, es poden estudiar de manera força clara i completa. A més, com que es defineixen per mitjà de cel·les i portes lògiques, s'implementen fàcilment en el maquinari, fet que permet obtenir generadors de gran velocitat.

Primerament cal fer notar que l'estat inicial d'un LFSR no pot ser tot zeros. Si fos així, la seqüència que produiria seria també tota de zeros, ja que totes les operacions són lineals. Es diu que l'estat que tan sols té zeros és un **estat absorbent**. També convé destacar que el període màxim d'un LFSR és $2^n - 1$. Aquest valor s'obté de considerar tots els estats possibles 2^n i eliminar-ne l'estat absorvent.

Si ens fixem en l'expressió 1 ens adonarem que tota la seqüència de sortida d'un LFSR queda determinada per l'estat inicial $\{s_1, \cdots, s_n\}$ i per la relació

$$s_{n+k} = \sum_{i=1}^{n} c_i s_{n+k-i} \quad \text{per } k \geq 0 \tag{2}$$

en què $c_i \in \{0,1\}$ per $1 \leq i \leq n$.

El **polinomi de connexions** d'un LFSR de longitud n és el polinomi de grau n.

$$C(x) = 1 + c_1x^1 + c_2x^2 + \cdots + c_nx^n$$

en què els $c_i \in \{0,1\}$ corresponen als valors de les portes lògiques de la figura de l'esquema general d'un LFSR.

Polinomi de connexions de l'LFSR de l'exemple anterior

El polinomi de connexions corresponent a l'LFSR de l'exemple anterior és:

$$C(x) = 1 + 0 \cdot x^1 + 1 \cdot x^2 + 0 \cdot x^3 + 1 \cdot x^4 = 1 + x^2 + x^4$$

Un LFSR queda determinat pel seu polinomi de connexions. Una **seqüència** queda determinada pel polinomi de connexions i pel seu estat inicial.

Definit el polinomi de connexions, ja podem determinar les característiques de l'LFSR d'acord amb les del seu polinomi de connexions:

1) Polinomi de connexions **factoritzable**: els LFSR que tenen polinomis de connexions factoritzables generen seqüències que depenen de l'estat inicial. A més, el període d'aquestes seqüències és sempre més petit que el període màxim que pot tenir un LFSR, que és $2^n - 1$.

2) Polinomi de connexions **irreductible**: les seqüències generades pels LFSR que tenen polinomis de connexions irreductibles no depenen de l'estat inicial, sinó que simplement queden desplaçades. En aquest cas, el període serà un divisor de $2^n - 1$.

3) Polinomi de connexions **primitiu**: un LFSR amb polinomi de connexions primitiu té la seqüència de sortida de període màxim $2^n - 1$. Aquesta seqüència de període màxim s'obté per a qualsevol estat inicial, llevat de l'estat absorbent.

Polinomi primitiu

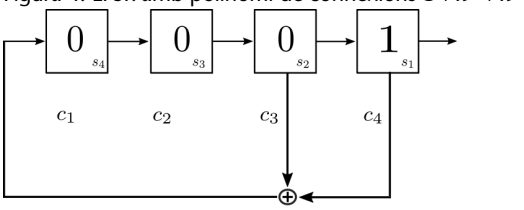
Un polinomi primitiu també és irreductible. Per tant, les seqüències generades pels LFSR amb polinomis de connexions primitius no dependran de l'estat inicial.

Considerant les propietats de les seqüències segons els seus polinomis de connexions, veiem que per a esquemes de xifratge de flux és aconsellable fer servir la que determina el període màxim.

Exemple d'LFSR amb polinomi de connexions primitiu

El polinomi $1 + x^3 + x^4$ (amb coeficients a $\mathbb{Z}_2$) és primitiu. Construïm un LFSR amb aquest polinomi de connexions i observem la seqüència de sortida de l'LFSR en fer servir els valors 0001 com a estat inicial. Vegeu aquest LFSR a la figura 4.

Figura 4. LFSR amb polinomi de connexions $1 + x^3 + x^4$



La seqüència generada serà:

$$L_1 = \underline{100010011010111}1000\ldots$$

Efectivament, el període de la seqüència generada és $2^n - 1 = 2^4 - 1 = 15$; a partir del bit 16, la seqüència torna a repetir-se.

Si generem una segona seqüència amb el mateix LFSR, però fent servir els valors 1010 com a estat inicial, la seqüència que s'obté és:

$$L_2 = \underline{010111100010011}0101\ldots$$

De nou, el període de la seqüència generada és 15.

El polinomi de connexions que hem fet servir és també irreductible. Per tant, les seqüències generades amb diferents estats inicials són les mateixes, però amb un desplaçament. En efecte, podem veure que la seqüència L_2 és la seqüència L_1 desplaçada 9 posicions a l'esquerra:

$$L_1 = \quad 100010011 \quad 010111100010011010 1$$
$$L_2 = \quad\quad\quad\quad\quad 010111100010011010 1$$

Per acabar, aprofitarem l'exemple per mostrar per què no podem generar una seqüència de període superior amb un LFSR de 4 cel·les. La taula 2 mostra tots els estats pels quals passa l'LFSR a l'hora de generar la seqüència L_1:

Taula 2. Estats de l'LFSR

t	Estat				Sortida	t	Estat				Sortida
0	0	0	0	1	1	8	0	1	0	1	1
1	1	0	0	0	0	9	1	0	1	0	0
2	0	1	0	0	0	10	1	1	0	1	1
3	0	0	1	0	0	11	1	1	1	0	0
4	1	0	0	1	1	12	1	1	1	1	1
5	1	1	0	0	0	13	0	1	1	1	1
6	0	1	1	0	0	14	0	0	1	1	1
7	1	0	1	1	1	15	0	0	0	1	1

Fixeu-vos que l'estat a $t = 15$ correspon a l'estat inicial ($t = 0$), motiu pel qual la seqüència comença a repetir-se. Noteu també que l'LFSR passa per quinze estats diferents, que són tots els possibles estats que es poden generar amb 4 bits, exceptuant l'estat absorvent (0000). El període és, doncs, màxim, i no hi ha manera de generar un període superior amb l'estructura d'un LFSR, ja que no hi ha més estats possibles.

Addicionalment, fixeu-vos que l'estat en $t = 9$ correspon a l'estat inicial amb el que generem la seqüència L_2.

3.3. Limitacions dels generadors lineals

Número de polinomis primitius

No hem d'oblidar que el nombre de polinomis primitius de grau n ve donat per l'expressió $\phi(2^n - 1)/n$, on ϕ és la funció totient d'Euler.

Ja hem posat en relleu que els LFSR es comporten molt bé en termes de facilitat d'anàlisi, d'implementació i de velocitat. Un dels punts negatius d'aquests generadors és que per tal que el període $2^n - 1$ sigui gran cal que la longitud

de l'LFSR, també ho sigui. Això pot representar un problema, ja que el cost de trobar polinomis primitius amb grau gran és força elevat.

Malgrat els avantatges i inconvenients presentats, la raó principal per la qual els LFSR no serveixen per si sols per a sistemes de xifratge en flux és que són fàcilment predictibles.

En efecte, suposem que coneixem $2n$ bits consecutius, $s_{k+1}, s_{k+2}, \cdots, s_{k+2n}$; aleshores, podem determinar els coeficients del polinomi de realimentació, c_i, i, per tant, tota la seqüència. Per fer-ho només cal basar-se en l'expressió 2 per a plantejar el sistema d'equacions següent:

$$\begin{pmatrix} s_{k+1} & s_{k+2} & \cdots & s_{k+n} \\ s_{k+2} & s_{k+3} & \cdots & s_{k+n+1} \\ \vdots & \vdots & \cdots & \vdots \\ s_{k+n} & s_{k+n+1} & \cdots & s_{k+2n-1} \end{pmatrix} \begin{pmatrix} c_n \\ c_{n-1} \\ \vdots \\ c_1 \end{pmatrix} = \begin{pmatrix} s_{k+n+1} \\ s_{k+n+2} \\ \vdots \\ s_{k+2n} \end{pmatrix}$$

Resolució del sistema d'equacions

Noteu que aquest sistema d'equacions es pot resoldre fàcilment amb qualsevol mètode de resolució de sistemes d'equacions lineals, per exemple, el mètode de Gauss. Per a una introducció als sistemes d'equacions lineals, podeu consultar el primer capítol del llibre *Elementary linear algebra*, d'H. Anton.

Per tant, tenim un sistema d'n equacions amb n incògnites, c_i per $1 \leq i \leq n$, amb la qual podem determinar tots els coeficients. Així doncs, a l'hora d'utilitzar un generador per a un procés de xifratge en flux, cal fixar-se també, com ja hem esmentat anteriorment, en la seva predictibilitat, és a dir, el que s'anomena la complexitat lineal.

Atès que qualsevol seqüència periòdica es pot generar amb un LFSR no singular, J. L. Massey va definir la complexitat lineal d'una seqüència de la següent manera:

La **complexitat lineal** d'una seqüència és el nombre de cel·les de l'LFSR més curt capaç de generar-la.

Per tant, una seqüència generada per un LFSR de longitud n té òbviament com a molt complexitat lineal n, molt baixa comparada amb el període, $2^n - 1$. El mateix Massey va proposar un algorisme que, a partir d'una seqüència, determina l'LFSR mínim que la genera amb l'estat inicial corresponent.

Algorisme de Massey

Massey va proposar un algorisme per sintetitzar l'LFSR més curt capaç de generar una seqüència l'any 1969 a l'article "Shift-Register Synthesis and BCH decoding".

Per a disminuir la predictibilitat de la seqüència de xifratge cal, doncs, augmentar la complexitat lineal de la seqüència de xifratge, que convindria que fos de llargada propera a la del període. Una manera de fer-ho és basant-se en operacions no lineals, tal com veurem més endavant.

4. Generadors no lineals

A continuació analitzarem alguns del generadors no lineals destinats a augmentar la complexitat lineal de les seqüències de flux que es fan servir en l'actualitat. En concret, descriurem l'A5 i el Trivium.

4.1. A5

L'A5 és un dels algorismes de xifratge de flux que més es fan servir actualment, en part per la seva utilització per al xifrat de dades en les transmissions de la xarxa GSM. L'A5 disposa de quatre variants, denotades amb els noms A5/0, A5/1, A5/2 i A5/3. L'A5/0 no fa servir xifrat (retorna el mateix text en clar), l'A5/1 correspon a la versió original de l'algorisme que es fa servir a Europa, l'A5/2 és un algorisme de xifrat més dèbil creada per a poder complir amb les regulacions per a exportar criptografia (i que es fa servir als Estats Units) i l'A5/3 és un algorisme de xifratge totalment diferent (afegit amb posterioritat). En aquest subapartat, descriurem el funcionament de l'algorisme A5/1.

GSM

GSM són les sigles de *global system for mobile communcation*, la xarxa que englobava més del 80% de les connexions mòbils el 2010. L'ús de la xarxa ha anat minvant amb l'aparició de xarxes amb més ample de banda com ara el 3G o 4G.

L'A5 va començar a utilitzar-se a la xarxa GSM sense fer pública la seva especificació, seguint el principi de seguretat per obscuritat (en anglès, *security through obscurity*). L'ús d'aquest paradigma està totalment desaconsellat pels experts, ja que viola el principi de Kerckhoffs. Tot i no fer-se públic oficialment, un primer esborrany de l'algorisme va ser publicat al 1994 i l'especificació completa va ser finalment obtinguda a través d'un procés d'enginyeria inversa del firmware d'un telèfon mòbil i donada a conèixer al públic el 1999.

Principi de Kerckhoffs

Recordem que el principi de Kerckhoffs postula que un criptosistema ha de ser segur encara que tota la informació sobre el criptosistema sigui pública, exeptuant la clau, que ha de romandre privada. És a dir, la seguretat d'un criptosistema ha de recaure únicament en el secret de la clau.

El criptosistema A5/1 és un criptosistema de flux que utilitza una combinació no lineal de la sortida de tres LFSR. Si pensem que l'A5/1 xifra cadenes de text en clar de 228 bits (trames de 228 bits), podem dividir el funcionament de l'A5/1 en tres etapes:

1) la inicialització dels LFSR,

2) l'obtenció dels 228 bits de la seqüència de xifrat a partir del moviment dels LFSR,

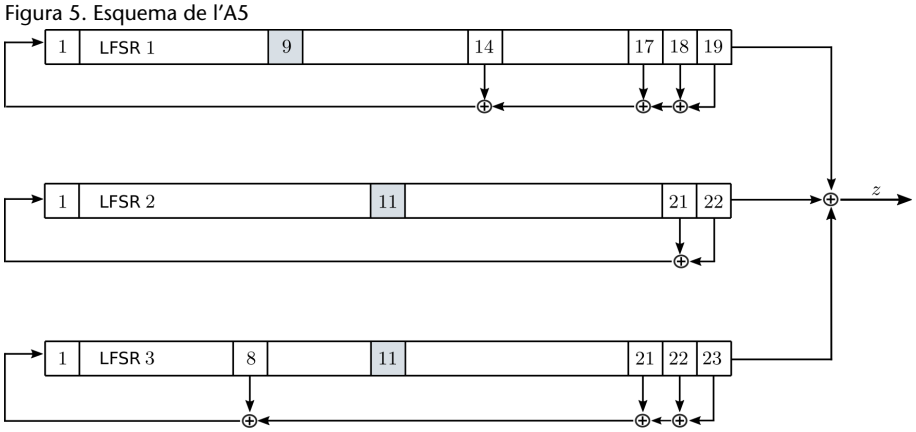
3) el xifrat del text en clar pròpiament dit, que segueix el procediment habitual dels xifrats de flux, realitzant una XOR de la seqüència de xifrat amb el text en clar.

Per xifrar 228 bits més caldrà tornar a reinicialitzar els LFSR, obtenir els nous 228 bits de la seqüència xifrant i fer l'XOR amb els nous bits de text en clar.

La inicialització dels tres LFSR que formen l'A5/1 no es limita a donar els seus valors inicials, sinó que el contingut inicial de les cel·les dels LFSR es calcula a partir d'unes claus d'entrada i d'unes transformacions que descriurem més endavant. Com que la inicialització dels LFSR es fa a partir de mateix funcionament del sistema, passem primer a descriure com s'obtenen els bits de la seqüència de xifrat.

L'A5/1 té una estructura formada per tres LFSR, tal com es mostra a la figura 5.

Figura 5. Esquema de l'A5



La taula 3 detalla les longituds de cadascun dels LFSR de l'A5/1, així com els seus polinomis de connexions.

Taula 3. Descripció dels LFSR de l'A5

LFSR	Longitud	Polinomi de connexions	*Clocking bit*
1	19	$x^{19} + x^{18} + x^{17} + x^{14} + 1$	9
2	22	$x^{22} + x^{21} + 1$	11
3	23	$x^{23} + x^{22} + x^{21} + x^{8} + 1$	11

La no linealitat del sistema ve donada perquè a cada impuls de rellotge no tots els LFSR avancen. Només ho fan aquells LFSR els bits dels quals són majoria en les cel·les anomenades *clocking bit* (en el cas de l'esquema, els clocking bits són les cel·les marcades en taronja, és a dir la cel·la 9 per al primer LFSR i les cel·les 11 per al segon i tercer). Aquest esquema es coneix com a clocking irregular segons la funció majoritària.

Així, per exemple, si en la cel·la 9 del primer LFSR hi ha un 1, i en les cel·les 11 del segon i tercer LFSR hi ha un 0, només avançaran el segon i el tercer LFSR, que tenen un 0. Si els tres són iguals, aleshores avancen tots. D'aquesta manera es van obtenint les sortides de cada un dels LFSR que formen la XOR, que acabarà proporcionant cada bit de la seqüència de xifratge.

Exemple d'una iteració de l'A5

A l'instant t tenim els estats interns següents als LFSR:

LFSR1: 1011100011 011000010
LFSR2: 1011011011 1101000010 01
LFSR3: 1110111110 0111001000 001

Així doncs, la sortida en aquest mateix instant de temps t serà:

$$z = 0 \oplus 1 \oplus 1 = 0$$

Es calcula a partir de la sortida dels tres LFSR (els bits subratllats en els estats interns).

Per tal de calcular l'estat dels LFSR a l'instant de temps $t+1$, observarem el bit de *clocking* de cada LFSR (indicat amb negreta). En aquest cas, els bits de *clocking* són 1, 1 i 0 per a l'LFSR 1, 2 i 3, respectivament. Per tant, el bit majoritari és 1, i avançaran, doncs, els LFSR 1 i 2. Així, l'estat intern dels LFSR a $t+1$ és:

LFSR1: 1101110001 101100001
LFSR2: 1101101101 1110100001 00
LFSR3: 1110111110 0111001000 001

Agrupació de bits

L'agrupació de bits de 10 en 10 respon únicament a qüestions estètiques: s'ha triat aquesta representació per tal que sigui més fàcil de llegir.

Passem ara a descriure el procés d'inicialització de l'A5. La inicialització requereix dos valors, una clau de sessió de 64 bits i un número de trama de 22 bits, i consta de quatre passos:

1) En primer lloc, s'omplen tots els registres dels tres LFSR amb el valor 0.

2) Seguidament, s'executen 64 impulsos de rellotge dels tres LFSR sense fer servir *clocking* irregular. És a dir, a cada impuls de rellotge, els tres LFSR avancen. La particularitat d'aquest pas és que el bit de retroalimentació de l'LFSR fa una XOR amb un bit de la clau de sessió abans de ser inserit a la primera cel·la de l'LFSR. Cadascuna de les 64 pulsacions fa servir un dels bits de la clau de sessió diferent, de manera seqüencial.

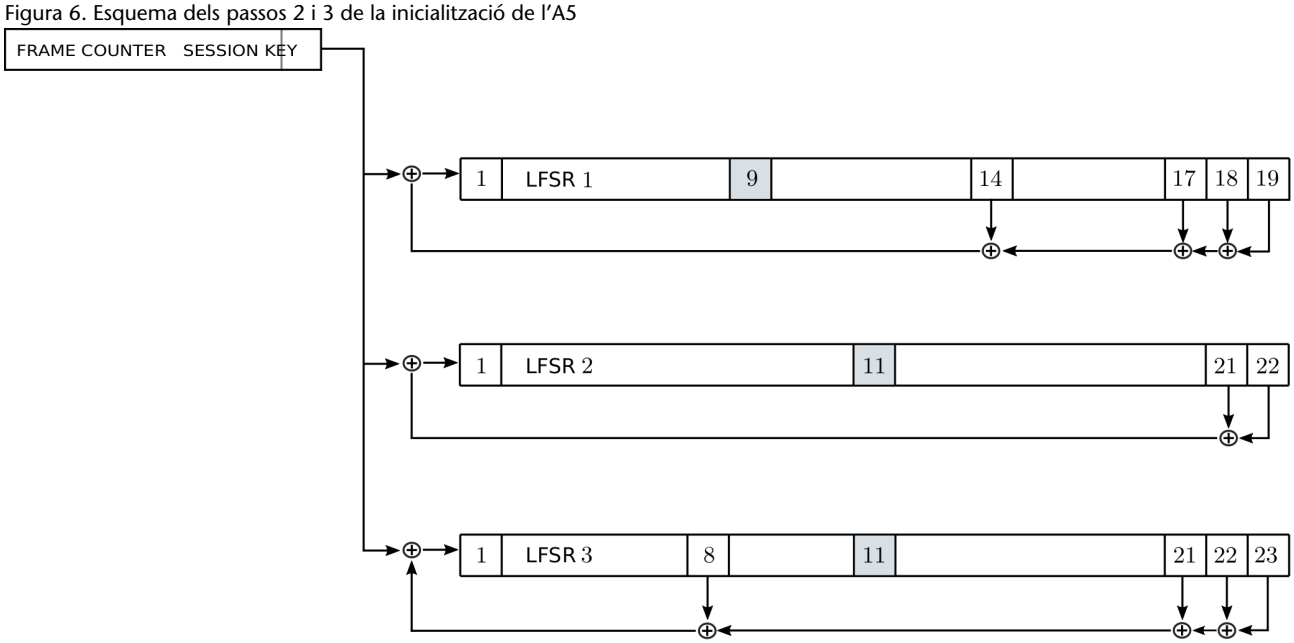
3) De manera similar al pas anterior, s'executen 22 impulsos de rellotge dels tres LFSR sense fer servir *clocking* irregular. Aquest cop, però, el bit de retroalimentació fa una XOR amb els bits del número de trama abans d'inserir-se de nou a la cel·la corresponent.

4) Finalment, es realitza una fase d'escalfament, on s'executen 100 impulsos de rellotge amb *clocking* irregular.

La figura 6 esquematitza el procés utilitzat per a realitzar els passos 2 i 3 de l'algorisme d'inicialització. Noteu que els passos 2 i 3 poden unir-se també amb un sol pas, on s'executen $64 + 22 = 86$ pulsacions de rellotge fent una XOR amb cadascun dels bits de la clau de sessió seguida del número de trama.

És important remarcar que en aquests passos d'inicialització el que interessa és el contingut que acabaran tenint les cel·les dels LFSR i, per tant, els bits de sortida dels LFSR en tots aquests passos es descarten. Un cop inicialitzats els LFSR es procedeix a obtenir els 228 bits de la seqüència de xifratge. Finalment, per tal de xifrar una trama, es farà una XOR amb els 228 bits obtinguts de la sortida de l'A5/1 i els 228 bits que representen el text en clar de la trama.

Figura 6. Esquema dels passos 2 i 3 de la inicialització de l'A5



Per a xifrar la trama següent de 228 bits, procedirem a incrementar el comptador de trama i tornarem a realitzar el procés d'inicialització amb la mateixa clau de sessió i el nou valor de comptador de trama.

Noteu que la clau de sessió no es canvia per cada nova trama que es vol xifrar, sinó que s'actualitza quan la xarxa decideix tornar a autenticar el dispositiu mòbil.

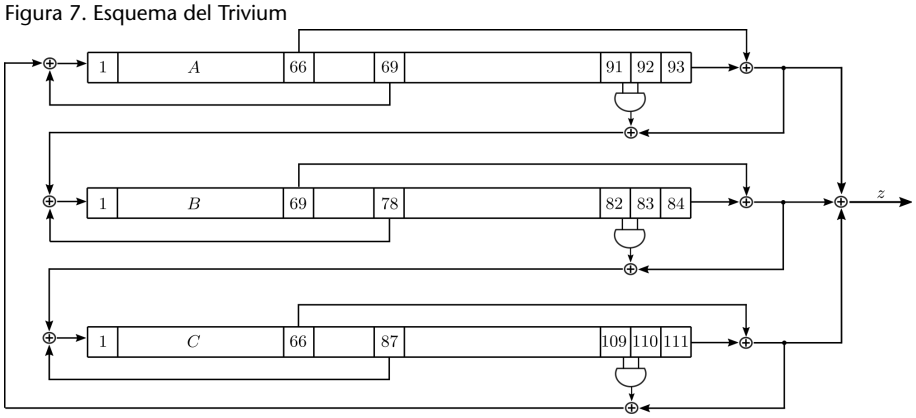
4.2. Trivium

El Trivium és un generador pseudoaleatori dissenyat pels criptògrafs Christophe de Cannière i Bart Preneel que aprofita una implementació de maquinari molt simple amb una velocitat elevada de generació de la seqüència, fet que el fa interessant en dispositius amb unes capacitats limitades de processat, com ara etiquetes RFID. El seu funcionament està descrit a l'estàndard ISO/IEC 29192-3.

El Trivium utilitza una clau de 80 bits i un vector d'inicialització també de 80 bits i permet generar seqüències de fins a 2^{64} bits.

A diferència de l'A5, el Trivium no es basa en LFSR, però sí que està format per tres registres de desplaçament, tot i que la seva realimentació no és lineal. És a dir, les cel·les que contenen els registres es desplacen a la dreta com en un LFSR, però la seva retroalimentació no està definida per una funció lineal. A la figura 7 podem veure l'esquema del Trivium.

Figura 7. Esquema del Trivium



Com es pot veure, el Trivium està format per tres registres de desplaçament, *A*, *B* i *C*, de 93, 84 i 111 cel·les, respectivament. La retroalimentació de cada registre no és lineal i, a més, la sortida de cada un dels registres retroalimenta un altre dels registres.

D'una banda, la sortida del Trivium (z) ve determinada en cada instant per les sortides dels tres registres (t^a, t^b, t^c):

$$z = t^a + t^b + t^c$$

en què cada un dels elements són bits i, per tant, la suma es realitza mòdul 2.

Cada una de les sortides t queden determinades per l'estat dels registres de la manera següent:

$$t^a = s^a_{93} + s^a_{66}$$

$$t^b = s^b_{84} + s^b_{69}$$

$$t^c = s^c_{111} + s^c_{66}$$

Per tal de calcular el valor de la cel·la en la retroalimentació, es fan servir les sortides t, de manera que la sortida del registre a, t^a s'utilitza en el càlcul de la retroalimentació del registre b; la sortida del registre b, t^b es fa servir en la retroalimentació de c; finalment, la sortida del registre c, t^c es fa servir en la retroalimentació del registre a. En concret, la retroalimentació de cada registre ve donada per les expressions:

$$s^a_{new} = t^c + (s^c_{109} \cdot s^c_{110}) + s^a_{69}$$

$$s^b_{new} = t^a + (s^a_{91} \cdot s^a_{92}) + s^b_{78}$$

$$s^c_{new} = t^b + (s^b_{82} \cdot s^b_{83}) + s^c_{87}$$

en què, de nou, tots els operands són bits i tant la suma com el producte d'aquesta expressió es realitzen mòdul 2.

La taula 4 resumeix les accions que realitza cada posició específica de cada un dels registres.

Taula 4. Posicions destacades dels registres del Trivium

Registre	*Feedback* bit	*Feedforward* bit	AND *inputs*
A	69	66	91,92
B	78	69	82,83
C	87	66	109,110

Producte mòdul 2

De manera equivalent, també podem pensar el producte mòdul 2 com un AND.

Exemple d'una iteració del Trivium

A l'instant t tenim els estats interns següents en els registres:

Agrupació de bits

De nou, l'agrupació de bits de 10 en 10 respon únicament a qüestions estètiques: s'ha triat aquesta representació per tal que sigui més fàcil de llegir. Noteu, però, que els 80 bits corresponen a l'estat del registre, sense cap mena de separació entre ells.

A : 0111111100 1101111010 1111100101 1101010001 0111100010 0110110001
1100110111 1111100110 0101011100 **01**1

B : 0100001010 1111011011 0110101000 1100010001 1111011000 1011110001
1101110100 0111100**0**01 1**01**1

C : 1111110100 0111011101 0101111100 1010111100 0100011100 0001111011
1000011010 0111000011 1101010011 0101001000 0100000**11** 0

Les sortides dels registres t corresponen als valors:

$$t^a = s^a_{93} + s^a_{66} = 1 + 1 = 0$$

$$t^b = s^b_{84} + s^b_{69} = 1 + 0 = 1$$

$$t^c = s^c_{111} + s^c_{66} = 0 + 1 = 1$$

Noteu que els bits involucrats en els càlculs dels valors t es troben subratllats en l'estat dels registres per facilitar la lectura.

Així, la sortida del Trivium en l'instant t correspon a:

$$z = t^a + t^b + t^c = 0 + 1 + 1 = 0$$

Podem calcular també els bits que es faran servir en la retroalimentació dels registres, per tal d'actualitzar-ne l'estat:

$$s^a_{new} = t^c + (s^c_{109} \cdot s^c_{110}) + s^a_{69} = 1 + (1 \cdot 1) + 1 = 1 + 1 + 1 = 1$$

$$s^b_{new} = t^a + (s^a_{91} \cdot s^a_{92}) + s^b_{78} = 0 + (0 \cdot 1) + 0 = 0 + 0 + 0 = 0$$

$$s^c_{new} = t^b + (s^b_{82} \cdot s^b_{83}) + s^c_{87} = 1 + (0 \cdot 1) + 0 = 1 + 0 + 0 = 1$$

Noteu que els bits involucrats en els càlculs dels valors s_{news} es troben indicats en negreta en l'estat dels registres per facilitar la lectura.

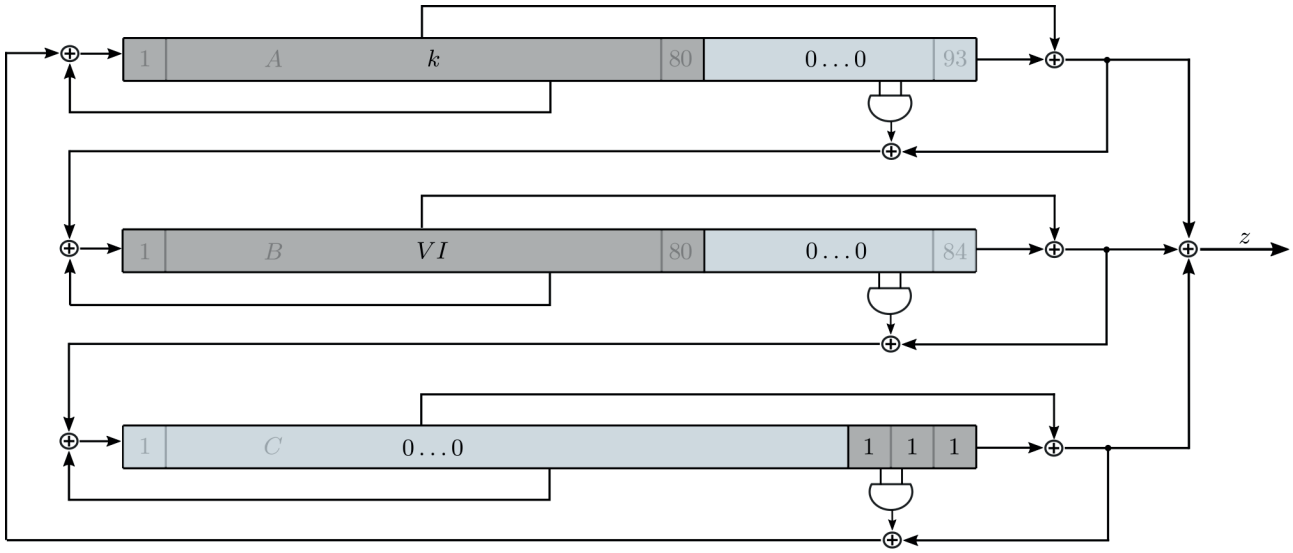
Els bits s_{new} serviran per actualitzar l'estat intern de cadascun dels registres. A tall d'exemple, veiem quin seria l'estat del registre A en l'instant $t + 1$:

A: 1011111110 0110111101 0111110010 1110101000 1011110001 0011011000
1110011011 1111110011 0010101110 001

4.2.1. Inicialització

A l'hora de xifrar un missatge, en primer lloc caldrà realitzar la **fase d'inicialització** del Trivium. Aquesta fase fa servir el vector inicial, *VI*, i la clau, *k*, ambdós valors de 80 bits. Aleshores, es prenen els 80 bits del vector inicial i es posen en les cel·les de més a l'esquerra del registre *B*. Seguidament, es prenen els 80 bits de la clau i es posen a les cel·les de més a l'esquerra del registre *A*. La resta de cel·les, de qualsevol dels tres registres, que no han quedat plenes s'omplen amb zeros, llevat de les tres cel·les de més a la dreta del registre *C*, en les quals s'inclou un 1. La figura 8 mostra gràficament la inicialització del Trivium.

Figura 8. Fase d'inicialització del Trivium



Una vegada s'han situat aquests valors als estats dels tres registres, s'executen 1152 cicles de rellotge descartant els bits de sortida d'aquestes 1152 iteracions.

Finalment, una vegada s'ha inicialitzat el generador ja es pot utilitzar la seqüència de sortida per a xifrar el missatge en clar. Així, cada bit de sortida del generador a partir de la iteració 1153 (un cop s'ha inicialitzat el generador) es combinarà amb una XOR amb el bit de text en clar que cal xifrar.

Per a desxifrar un missatge utilitzant el Trivium, caldrà realitzar exactament el mateix procés, però aquesta vegada sobre el missatge xifrat, procés que es pot dur a terme perquè emissor i receptor comparteixen tant el vector inicial com la clau, ja que estem davant d'un criptosistema de clau simètrica.

5. 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 xifratge 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'autentificació 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 de n bits de text en clar i retorna un text xifrat c també de 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 enal 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 desxifratge, que realitza el procés invers de la de xifratge:

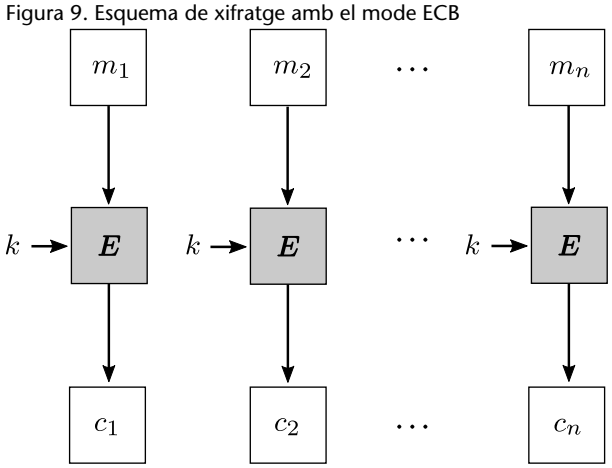
$$b = D_k(c)$$

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, ..., cada un dels quals té la llargada corresponent al bloc per a xifrar (n bits), i xifrar cadascun dels fragments. El procediment que cal seguir per a xifrar cadascun dels fragments queda determinat pel **mode d'operació**.

5.1. Modes d'operació

El mode d'operació més senzill es 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 a formar el text xifrat c. La figura 9 esquematitza el procés de xifratge en mode ECB.

Figura 9. Esquema de xifratge amb el mode ECB



Les propietats principals que ens ofereix el mode ECB són les següents:

1) Els blocs de text en clar idèntics resulten en blocs xifrats també idèntics (si es fa servir la mateixa clau).

2) Cada bloc es xifra de manera independent als altres blocs.

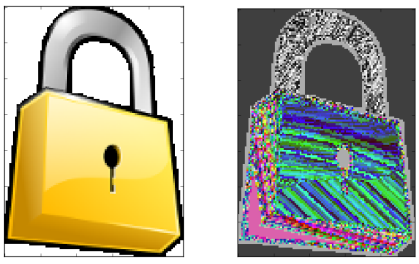
3) Permet accés aleatori al contingut, és a dir, és possible desxifrar un bloc sense haver de desxifrar els anteriors.

4) Els errors no es propaguen: un error en un bloc afecta només 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 en 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 desxifratge 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 (vegeu la figura 10).

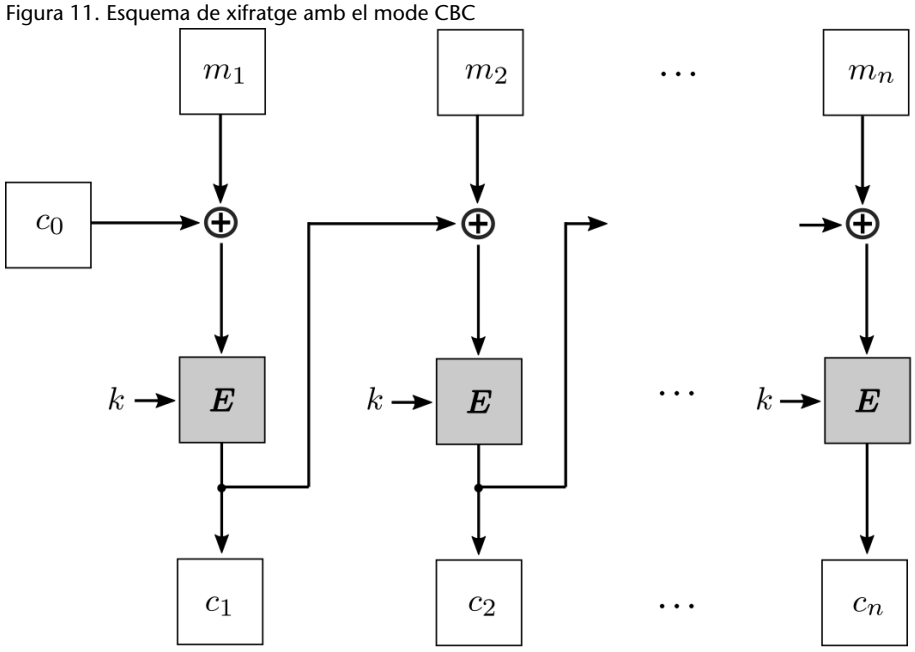
Figura 10. Exemple de xifratge d'una imatge amb ECB



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 revelen a la versió xifrada (per exemple, el color), la silueta de la imatge queda perfectement recognoscible.

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 11 esquematitza el funcionament del mode CBC.

Figura 11. Esquema de xifratge 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 al 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.

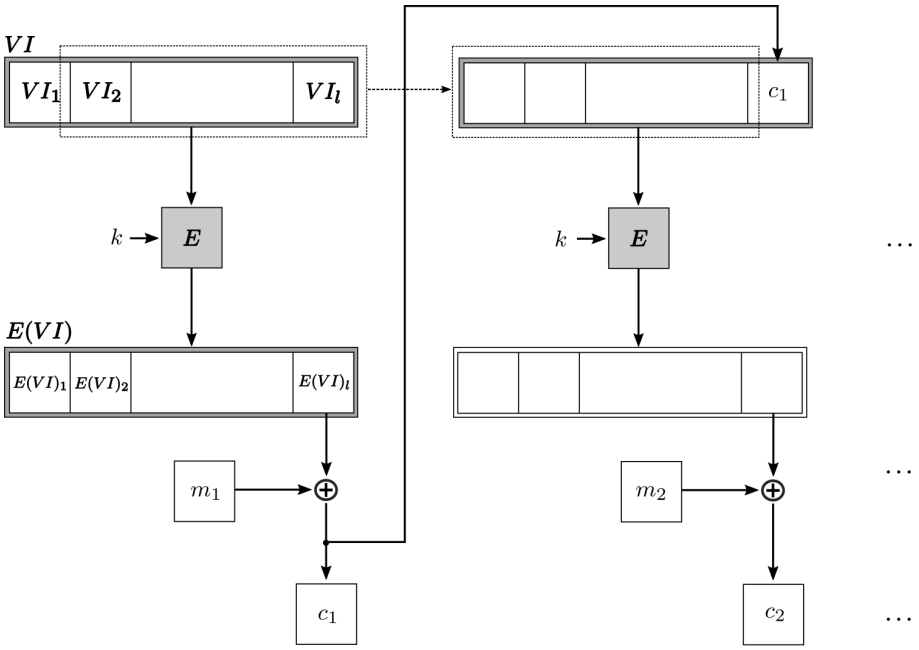
En contraposició amb el mode ECB, si un atacant canvia l'ordre dels blocs xifrats amb CBC, aleshores el procés de desxifratge no es realitza correctament. Addicionalment, un error en un bloc xifrat afecta el desxifratge d'aquell bloc, però també del següent. Noteu que els blocs successius ja es desxifren 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 candau fent servir ara el mode CBC, podrem observar que ara no podem intuir el perfil de la imatge a partir de la imatge xifrada (vegeu la figura 12).

Figura 12. Exemple de xifratge d'una imatge amb CBC

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 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 13.

Figura 13. Esquema de xifratge 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,... representen els blocs de longitud n que formen el missatge, si considerem el vector inicial VI com una concatenació de 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 per 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:

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

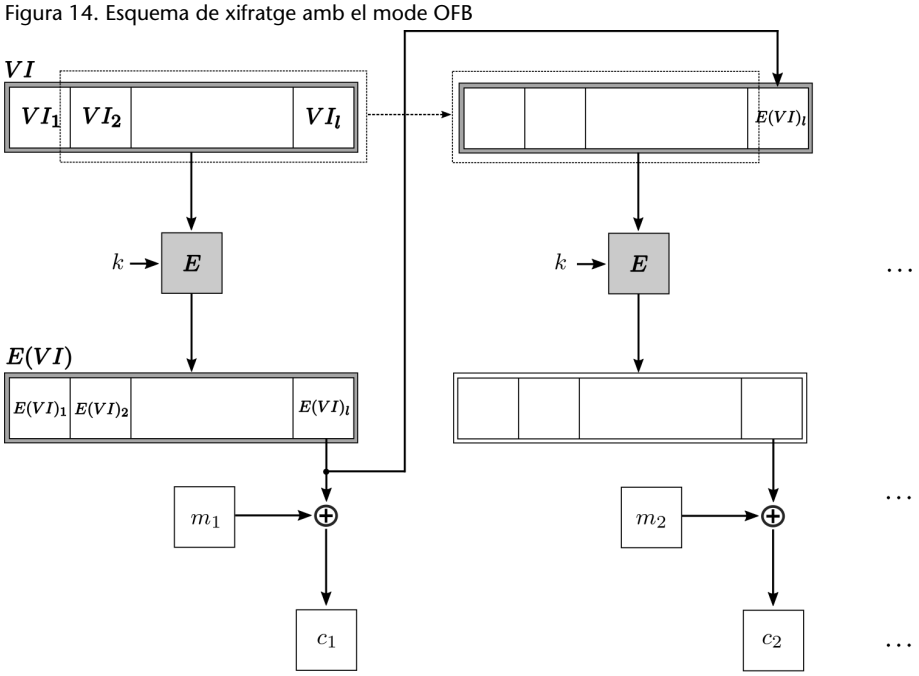
És a dir, hem desplaçat els blocs de n bits cap a l'esquerra per afegir-hi el bloc c_1 i descartar-ne el VI_1. D'aquesta manera, el segon bloc de text xifrat l'obtenim fent l'operació següent:

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

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.

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 per bit amb el bloc de text en clar, com es pot veure a la figura 14.

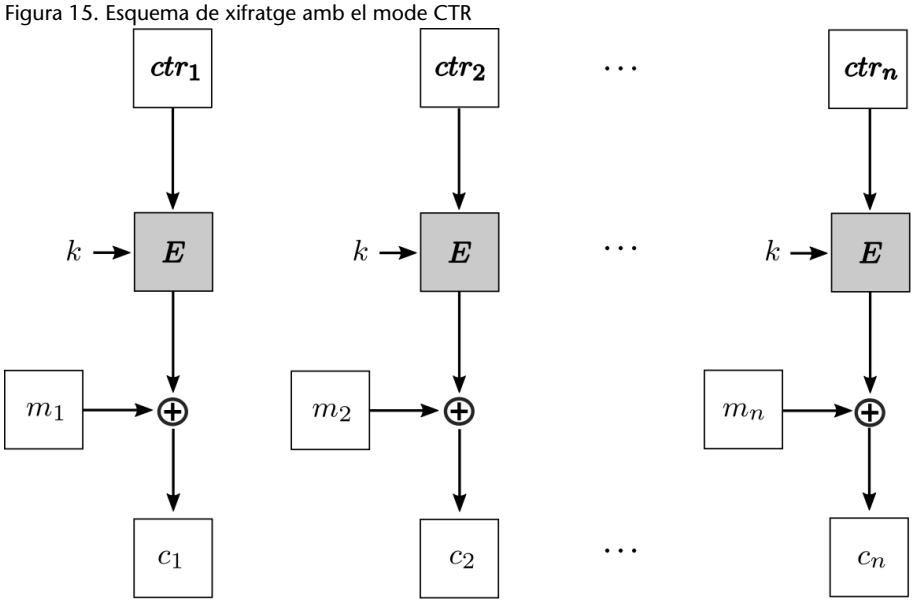
Figura 14. Esquema de xifratge amb el mode OFB



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.

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

Figura 15. Esquema de xifratge 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 que s'utilitza 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 xifratge com el de desxifratge, fet que el fa adient per a funcionar en dispositius amb més d'un processador. A més, permet accés aleatori (com el mode ECB).

6. 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 xifratge amb diferents mides de clau i de bloc. L'AES n'és només un subconjunt, amb mida de bloc fixada a 128 bits.

Mida de bloc i de clau

El Rijndael permet múltiples mides de bloc i de clau. En concret, el Rijndael defineix blocs i claus de mida mínima 128 i màxima de 256, acceptant múltiples de 32 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.

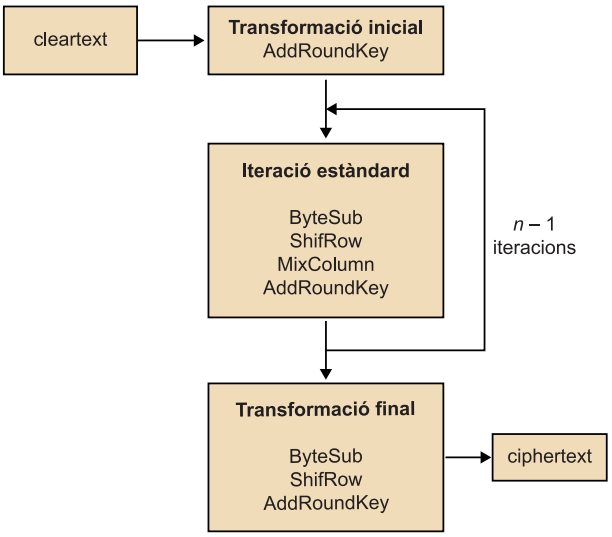
6.1. Descripció del funcionament

El funcionament de l'AES es mostra a la figura 16. Es basa en una transformació inicial seguida d'un nombre d'iteracions que varien entre 10 i 14, segons la longitud de la clau.

Nombre d'iteracions

El nombre d'iteracions que es mostren a la figura 16 és $n-1$ perquè la iteració final, tot i que es considera iteració, no conté la funció mixColumn.

Figura 16. 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 a 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, tenim 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 manera següent:

$$a_{0,0} = m_1 m_2 m_3 m_4 m_5 m_6 m_7 m_8$$
$$a_{1,0} = m_9 m_{10} m_{11} m_{12} m_{13} m_{14} m_{15} m_{16}$$
$$\cdots$$
$$a_{3,3} = m_{121} m_{122} m_{123} m_{124} m_{125} m_{126} m_{127} m_{128}$$

Aquests bytes es poden expressar de manera 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 de 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:

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

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

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

Exemple de representació polinòmica

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 $y = (y_7,y_6,y_5,y_4,y_3,y_2,y_1,y_0)$.

Operació XOR

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

D'una banda, definim l'operació suma:

$$x \oplus y = (x_7 \oplus y_7,x_6 \oplus y_6,x_5 \oplus y_5,x_4 \oplus y_4,x_3 \oplus y_3,x_2 \oplus y_2,x_1 \oplus y_1,x_0 \oplus y_0)$$

en què $\oplus$ denota l'operació XOR bit per bit.

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 de càlcul de suma i producte:

Donats els bytes x i y següents:

$$x = 57_{(16} = 01010111_{(2} = x^6 + x^4 + x^2 + x + 1$$
$$y = 83_{(16} = 10000011_{(2} = x^7 + x + 1$$

Calculem la suma i el producte de bytes:

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

Això és així perquè:

$$01010111_{(2} \oplus 10000011_{(2} = 11010100_{(2} = D4_{(16}$$

D'altra banda, pel producte tenim:

$$x \otimes y = 57_{(16} \otimes 83_{(16} = C1_{(16}$$

Això és així perquè:

$$(x^6 + x^4 + x^2 + x + 1) \cdot (x^7 + x + 1) \pmod{x^8 + x^4 + x^3 + x + 1} =$$
$$= (x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1) \pmod{x^8 + x^4 + x^3 + x + 1} =$$
$$= x^7 + x^6 + 1 = 11000001_{(2} = C1_{(16}$$

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

6.2. Detall d'una iteració

A la figura 16 el 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 de fer 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.

A continuació, passem a descriure cada una de les funcions que s'executen en cada iteració.

6.3. 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 $K(0)$.

Subclaus

L'índex i denota la subclau de 128 bits que es fa servir en la i-èssima 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 al subapartat 6.7. d'aquest mòdul.

Exemple de càlcul de la funció AddRoundKey

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

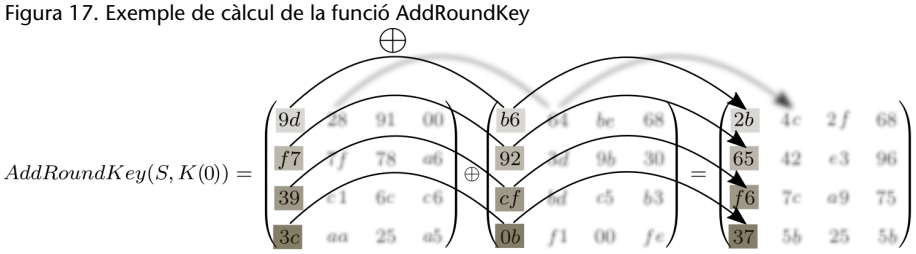
i la matriu d'estat $S = \begin{pmatrix} 9d & 28 & 91 & 00 \\ f7 & 7f & 78 & a6 \\ 39 & c1 & 6c & c6 \\ 3c & aa & 25 & a5 \end{pmatrix}$

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 $9D \oplus B6 = 10011101 \oplus 10110110 = 2B$.

Figura 17. Exemple de càlcul de la funció AddRoundKey



6.4. Funció ByteSub

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

Nom de la funció ByteSub

La funció **ByteSub** apareix amb aquesta denominació amb 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 dona, en tots dos casos és la mateixa funció.

La funció 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.

6.4.1. 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 representen com una matriu de 16 files i 16 columnes. Si representem cada byte que cal processar amb dos caràcters hexadecimals *x* i *y*, 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.

	*x*0	*x*1	*x*2	*x*3	*x*4	*x*5	*x*6	*x*7	*x*8	*x*9	*xa*	*xb*	*xc*	*xd*	*xe*	*xf*
0*y*	63	7*c*	77	7*b*	*f*2	6*b*	6*f*	*c*5	30	01	67	2*b*	*fe*	*d*7	*ab*	76
1*y*	*ca*	82	*c*9	7*d*	*fa*	59	47	*f*0	*ad*	*d*4	*a*2	*af*	9*c*	*a*4	72	*c*0
2*y*	*b*7	*fd*	93	26	36	3*f*	*f*7	*cc*	34	*a*5	*e*5	*f*1	71	*d*8	31	15
3*y*	04	*c*7	23	*c*3	18	96	05	9*a*	07	12	80	*e*2	*eb*	27	*b*2	75
4*y*	09	83	2*c*	1*a*	1*b*	6*e*	5*a*	*a*0	52	3*b*	*d*6	*b*3	29	*e*3	2*f*	84
5*y*	53	*d*1	00	*ed*	20	*fc*	*b*1	5*b*	6*a*	*cb*	*be*	39	4*a*	4*c*	58	*cf*
6*y*	*d*0	*ef*	*aa*	*fb*	43	4*d*	33	85	45	*f*9	02	7*f*	50	3*c*	9*f*	*a*8
7*y*	51	*a*3	40	8*f*	92	9*d*	38	*f*5	*bc*	*b*6	*da*	21	10	*ff*	*f*3	*d*2
8*y*	*cd*	0*c*	13	*ec*	5*f*	97	44	17	*c*4	*a*7	7*e*	3*d*	64	5*d*	19	73
9*y*	60	81	4*f*	*dc*	22	2*a*	90	88	46	*ee*	*b*8	14	*de*	5*e*	0*b*	*db*
ay	*e*0	32	3*a*	0*a*	49	06	24	5*c*	*c*2	*d*3	*ac*	62	91	95	*e*4	79
by	*e*7	*c*8	37	6*d*	8*d*	*d*5	4*e*	*a*9	6*c*	56	*f*4	*ea*	65	7*a*	*ae*	08
cy	*ba*	78	25	2*e*	1*c*	*a*6	*b*4	*c*6	*e*8	*dd*	74	1*f*	4*b*	*bd*	8*b*	8*a*
dy	70	3*e*	*b*5	66	48	03	*f*6	0*e*	61	35	57	*b*9	86	*c*1	1*d*	9*e*
ey	*e*1	*f*8	98	11	69	*d*9	8*e*	94	9*b*	1*e*	87	*e*9	*ce*	55	28	*df*
fy	8*c*	*a*1	89	0*d*	*bf*	*e*6	42	68	41	99	2*d*	0*f*	*b*0	54	*bb*	16

Exemple de 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:

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

6.5. 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 de càlcul de la funció ShiftRow

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 18, deixant la fila zero de la matriu sense modificar i desplaçant les files 1, 2 i 3, una, dues i tres posicions, respectivament:

Figura 18. Exemple de càlcul de la funció ShiftRow

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

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

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

6.6. 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 a polinomis de grau 3. Cada columna es multiplica pel polinomi $c(x) = \text{“}\texttt{03}\text{”}x^3 + \text{“}\texttt{01}\text{”}x^2 + \text{“}\texttt{01}\text{”}x + \text{“}\texttt{02}\text{”}$ 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 de suma i producte entre els elements de la matriu i els del vector columna són les operacions $\oplus$ i $\otimes$ definides al subapartat 6.1.

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 pel polinomi $d(x)$:

$$c(x) \otimes d(x) = \text{“}\texttt{01}\text{”}$$

El polinomi $d(x)$ és, doncs, $\text{“}\texttt{0B}\text{”}x^3 + \text{“}\texttt{0D}\text{”}x^2 + \text{“}\texttt{09}\text{”}x + \text{“}\texttt{0E}\text{”}$.

Exemple de 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 els valors de la taula 5.

Taula 5. Representació polinòmica de valors hexadecimals

Hexadecimal	Binari								Polinomi
01	0	0	0	0	0	0	0	1	1
d5	1	1	0	1	0	1	0	1	$x^7 + x^6 + x^4 + x^2 + 1$
02	0	0	0	0	0	0	1	0	x
4b	0	1	0	0	1	0	1	1	$x^6 + x^3 + x + 1$
03	0	0	0	0	0	0	1	1	$x + 1$
af	1	0	1	0	1	1	1	1	$x^7 + x^5 + x^3 + x^2 + x + 1$
5e	0	1	0	1	1	1	1	0	$x^6 + x^4 + x^3 + x^2 + x$

Si ara fem els càlculs, resulta:

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

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

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

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

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:

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

6.7. Generació de subclaus

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 xifratge farà servir quatre paraules de 32 bits i caldran quatre paraules addicionals per a la inicialització. Si denotem per $K(i)$ cada una de les subcadenes de W de quatre 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 de la manera següent:

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).

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

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, en què la clau de xifratge és una còpia íntegrament de les primeres posicions de la clau estesa. 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'estén. L'algorisme que implementa aquesta fase queda descrit pel pseudocodi següent:

```
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 [$a1$,$a2$,$a3$,$a0$]. Així mateix, també es fa servir la constant $Rcon[i]$, que val $Rcon[i] = [x^{i-1}, \text{“00”}, \text{“00”}, \text{“00”}]$. Recordeu que el polinomi x en hexadecimal val “02”, ja que correspon a la representació en binari de 00000010. La figura 19 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 $K = (K_0K_1K_2K_3)$, 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.

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

Exemple de càlcul de l'expansió de claus

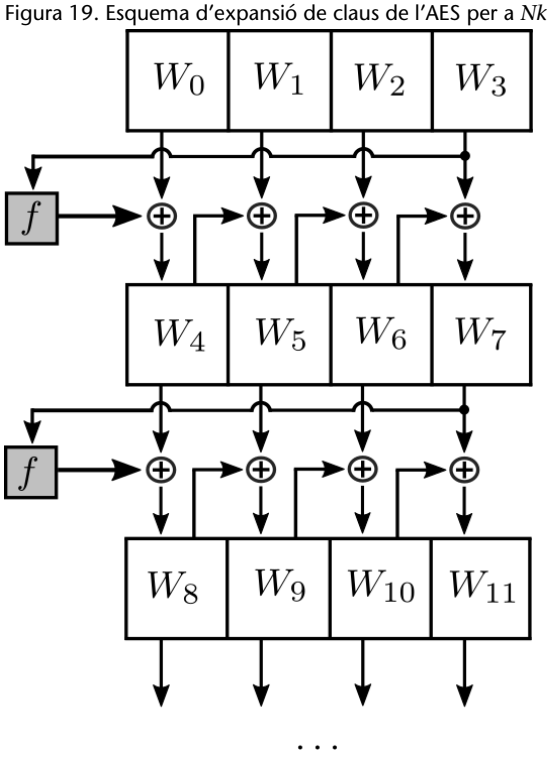
Suposem que la longitud de la clau és de 128 bits, és a dir, $Nk = 4$ paraules de 32 bits i que la clau de xifratge (representada en hexadecimal) correspon a:

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

Representació hexadecimal

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

Figura 19. Esquema d'expansió de claus de l'AES per a $Nk = 4$



Amb aquests paràmetres tenim que el nombre d'iteracions és $Nr = 10$. Això vol dir que la clau estesa 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 estesa 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:

$$W_4 = W_0 \oplus SubWord(RotWord(W_3)) \oplus Rcon[1]$$

$$SubWord(RotWord(W_3)) = RotWord(\texttt{0C 0D 0E 0F}) = \texttt{0D 0E 0F 0C}$$

$$SubWord(\texttt{0D 0E 0F 0C}) = (\texttt{D7 AB 76 FE})$$

$$W_4 = \texttt{00 01 02 03} \oplus \texttt{D7 AB 76 FE} \oplus \texttt{01 00 00 00} = \texttt{D6 AA 74 FD}$$

$$W_5 = W_1 \oplus W_4 = \texttt{04 05 06 07} \oplus \texttt{D6 AA 74 FD} = \texttt{D2 AF 72 FA}$$

$$W_6 = W_2 \oplus W_5 = \texttt{08 09 0A 0B} \oplus \texttt{D2 AF 72 FA} = \texttt{DA A6 78 F1}$$

$$W_7 = W_3 \oplus W_6 = \texttt{0C 0D 0E 0F} \oplus \texttt{DA A6 78 F1} = \texttt{D6 AB 76 FE}$$

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.

6.8. Desxifratge

En 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 funció inversa corresponent.

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.

Resum

En aquest mòdul didàctic 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) són 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 amb les xifres de bloc, en primer lloc n'hem descrit l'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.

Exercicis d'autoavaluació

1. Calculeu els primers 15 bits de la seqüència de sortida d'un LFSR de 5 cel·les que té com a polinomi de connexions $1 + x^2 + x^5$ i que s'inicialitza amb l'estat 0,0,0,1,1.

2. Quin és el període i la complexitat lineal màxima de les seqüències generades per l'LFSR amb polinomi de connexions $1 + x^2 + x^5$?

3. Donada la seqüència s = 00010011010111100010 , sintetitzeu l'LFSR que l'ha generada, sabent que el polinomi de connexions té grau 4.

4. Suposem un esquema de xifratge de bloc com el de la taula 6 amb mida de bloc de 2 bits i mida de clau també de 2 bits que implementa aquesta funció de xifratge E:

Taula 6. Funció de xifratge

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 xifratge E i el mode d'operació ECB.

5. Xifreu el mateix missatge m amb la funció E i la clau k = 10, fent servir ara el mode d'operació CBC amb el vector inicial 10.

6. Xifreu el mateix missatge m amb la funció E i la clau k = 10, fent servir ara el mode d'operació CFB amb el vector inicial 10.

7. Xifreu el mateix missatge m amb la funció E i la clau k = 10, fent servir ara el mode d'operació OFB amb el vector inicial 10.

8. Xifreu el mateix missatge m amb la funció E i la clau k = 10, fent servir ara el mode d'operació CTR amb el vector inicial 10.

9. 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 $K(1)$.

10. 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?

Solucionari

1. Tenint en compte el polinomi de connexions de l'LFSR, el nou bit es calcula fent una XOR entre els bits de les cel·les s_4 i s_1 (que es troben subratllats a la taula 7) en l'instant de temps anterior:

Taula 7. Evolució de l'LFSR

Impuls de rellotge (t)	Estat					Sortida
0	0	0	0	1	1	1
1	1	0	0	0	1	1
2	1	1	0	0	0	0
3	1	1	1	0	0	0
4	1	1	1	1	0	0
5	1	1	1	1	1	1
6	0	1	1	1	1	1
7	0	0	1	1	1	1
8	1	0	0	1	1	1
9	1	1	0	0	1	1
10	0	1	1	0	0	0
11	1	0	1	1	0	0
12	0	1	0	1	1	1
13	0	0	1	0	1	1
14	1	0	0	1	0	0

Així doncs, els 15 primers bits de la seqüència de sortida són: 110001111100110

2. El polinomi $1 + x^2 + x^5$ té grau $n = 5$ i és un polinomi primitiu. Per tant, la complexitat lineal màxima de les seqüències que genera és $n = 5$ i el període serà $2^n - 1 = 2^5 - 1 = 31$.

3. Per trobar el polinomi de connexions necessitem únicament $2n = 8$ bits consecutius de la seqüència de sortida. Si agafem, per exemple, els 8 primers bits, podem plantejar el sistema d'equacions següents:

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_4 \\ c_3 \\ c_2 \\ c_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

La solució del sistema és $c_4 = 1, c_3 = 1, c_2 = 0, c_1 = 0$ i, per tant, el polinomi de connexions és $x^4 + x^3 + 1$.

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

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

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

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

5. En primer lloc, procedim a separar el missatge en blocs de 2 bits. Després, per a 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 que cal seguir és, doncs, el que es mostra a la taula 8.

Taula 8. Porcés amb el mode CBC

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.

6. En aquest cas, la mida de bloc del criptosistema és de 2 bits, per la qual cosa els blocs de text que cal xifrar poden ser com a molt de 2 bits. Agafem, doncs, blocs de text que volem xifrar de 2 bits i procedim a realitzar el procés de xifratge. 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. Vegeu tot el procés a la taula 9.

Taula 9. Procés amb el mode CFB

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.

7. En aquest cas, la mida de bloc del criptosistema és de 2 bits, per la qual cosa els blocs de text que es volen xifrar poden ser com a molt de 2 bits. Agafem, doncs, blocs de text que es volen xifrar de 2 bits i procedim a realitzar el procés de xifratge. 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 $v_0 = VI$. Vegeu tot el procés a la taula 10.

Taula 10. Procés amb el mode OFB

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.

8. 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 quatre valors, per la qual cosa 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. Vegeu tot el procés a la taula 11.

Taula 11. Procés amb el mode CTR

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.

9. 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:

$$W_0 = \texttt{8E 73 B0 F7}$$

$$W_1 = \texttt{DA 0E 64 52}$$

$$W_2 = \texttt{C8 10 F3 2B}$$

$$W_3 = \texttt{80 90 79 E5}$$

$$W_4 = \texttt{62 F8 EA D2}$$

$$W_5 = \texttt{52 2C 6B 7B}$$

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:

$W_6 = W_0 \oplus SubWord(RotWord(W_5)) \oplus Rcon[1]$

$RotWord(W_5) = RotWord(\texttt{52 2C 6B 7B}) = \texttt{2C 6B 7B 52}$

$SubWord(\texttt{2C 6B 7B 52}) = (\texttt{71 7F 21 00})$

$W_6 = \texttt{8E 73 B0 F7} \oplus \texttt{71 7F 21 00} \oplus \texttt{01 00 00 00} = \texttt{8E 73 B0 F7} \oplus \texttt{70 7F 21 00} = \texttt{FE 0C 91 F7}$

$W_7 = W_1 \oplus W_6 = \texttt{DA 0E 64 52} \oplus \texttt{FE 0C 91 F7} = \texttt{24 02 F5 A5}$

$W_8 = W_2 \oplus W_7 = \texttt{C8 10 F3 2B} \oplus \texttt{24 02 F5 A5} = \texttt{EC 12 06 8E}$

$W_9 = W_3 \oplus W_8 = \texttt{80 90 79 E5} \oplus \texttt{EC 12 06 8E} = \texttt{6C 82 7F 6B}$

$W_{10} = W_4 \oplus W_9 = \texttt{62 F8 EA D2} \oplus \texttt{6C 82 7F 6B} = \texttt{0E 7A 95 B9}$

$W_{11} = W_5 \oplus W_{10} = \texttt{52 2C 6B 7B} \oplus \texttt{0E 7A 95 B9} = \texttt{5C 56 FE C2}$

Per tant, la subclau:

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

10. 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:

$$AddRoundKey(S,K(0)) = \begin{pmatrix} 32 & 88 & 31 & e0 \\ 43 & 5a & 31 & 37 \\ f6 & 30 & 98 & 07 \\ a8 & 8d & a2 & 34 \end{pmatrix} \oplus \begin{pmatrix} 2b & 28 & ab & 09 \\ 7e & ae & f7 & cf \\ 15 & d2 & 15 & 4f \\ 16 & a6 & 88 & 3c \end{pmatrix} = \begin{pmatrix} 19 & a0 & 9a & e9 \\ 3d & f4 & c6 & f8 \\ e3 & e2 & 8d & 48 \\ b3 & 2b & 2a & 08 \end{pmatrix} = S_1$$

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:

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

Calculem el valor de la clau de la segona iteració:

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

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:

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

Glossari

A5 *m* Família d'algorismes de xifratge de flux utilitzada a la xarxa GSM.

advanced encryption standard *m* Subconjunt del criptosistema de bloc Rijndael escollit com a estàndard de xifratge pel NIST.
sigla **AES**

AES *m* Vegeu ***advanced encryption standard***.

CBC *m* Vegeu ***cipher block chaining***.

CFB *m* Vegeu ***cipher feedback***.

cipher block chaining *m* Mode d'operació de les xifres de bloc en què no es xifra directament cada bloc de text en clar, sinó que primer s'aplica una XOR amb el bloc xifrat anterior.
sigla **CBC**

cipher feedback *m* Mode d'operació de les xifres de bloc que fa servir el xifrador de bloc com a generador pseudoaleatori utilitzant cada bloc xifrat per a realimentar el xifratge del bloc següent.
sigla **CFB**

complexitat lineal d'una seqüència *f* Nombre de cel·les de l'LFSR més curt que és capaç de generar-la.

counter *m* Mode d'operació de les xifres de bloc que fa servir el xifrador de bloc com a generador pseudoaleatori, generant la seqüència xifrant a partir de xifrar un comptador.
sigla **CTR**

criptosistema de bloc *m* Sistema de xifratge de clau simètrica que processa els bits d'entrada en blocs d'una mida predeterminada de bits.

criptosistema de clau compartida *m* Criptosistema en què tant l'emissor com el receptor comparteixen una sola clau que fan servir tant per a xifrar com per a desxifrar.

criptosistema de flux *m* Sistema de xifratge que utilitza un generador pseudoaleatori per a xifrar un missatge, sumant bit per bit el text en clar amb la seqüència pseudoaleatòria que resulta del generador.

CTR *m* Vegeu ***counter***.

ECB *m* Vegeu ***electronic code book***.

electronic code book *m* Mode d'operació de les xifres de bloc en què cada bloc d'entrada es xifra de manera individual, fent servir la mateixa clau.

estat d'un LFSR *m* Conjunt de valors continguts en cada cel·la d'un LFSR en un instant de temps.

generador congruencial *m* Procés determinista que genera seqüències en base a equacions modulars recurrents.

generador lineal *m* Generador de seqüències de bits que només executa operacions lineals sobre els elements d'entrada per a obtenir la seqüència de sortida.

generador no lineal *m* Generador de seqüències de bits que executa operacions no lineals, com ara permutacions, sobre els elements d'entrada per a obtenir la seqüència de sortida; a més, pot fer servir també operacions lineals.

generador pseudoaleatori *m* Procés determinista capaç de generar una seqüència pseudoaleatòria.
sigla **PRNG**
en pseudo random number generator

global system for mobile communcation *m* Xarxa que englobava més del 80% de les connexions mòbils el 2010.

GSM *m* Vegeu ***global system for mobile communcation***.

LFSR Vegeu **registre de desplaçament realimentat linealment**.

linear feedback shift register *m* Vegeu **registre de desplaçament realimentat linealment**.

OFB Vegeu ***output feedback***.

output feedback *m* Mode d'operació de les xifres de bloc que fa servir el xifrador de bloc com a generador pseudoaleatori, fent servir la sortida del xifrador a cada iteració per a realimentar el xifratge del bloc següent.

període *m* En una seqüència $\{s_i\}_{i\geq 0}$ periòdica, enter més petit p tal que $s_{i+p} = s_i$ per a tot $i \geq 0$.

polinomi de connexions d'un LFSR *m* Polinomi que determina o que és determinat per la funció lineal de realimentació de l'LFSR.

PRNG *m* Vegeu **generador pseudoaleatori**.

pseudo random number generator *m* Vegeu **generador pseudoaleatori**.

ràfega *f* Conjunt de bits consecutius iguals dins una seqüència.

registre de desplaçament realimentat linealment *m* Dispositiu físic o lògic format per *n* cel·les de memòria i una funció de realimentació lineal.
sigla **LFSR**
en linear feedback shift register

seqüència de xifratge *f* Seqüència que resulta del generador pseudoaleatori en un criptosistema de flux.

seqüència pseudoaleatòria *f* Seqüència generada per un algorisme determinista que exhibeix propietats similars a les d'una seqüència veritablement aleatòria.

Trivium *m* Generador pseudoaleatori lleuger a partir de registres amb realimentació no lineal.

Bibliografia

Cannière, C. de; Preneel, B. (2005). *Trivium Specifications*. Lovaina: Universitat Catòlica de Lovaina.

Daemen, J.; Rijmen, V. (2002). *The Design of Rijndael, AES - The Advanced Encryption Standard* (238 pàg.). Springer-Verlag.

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

Institut Nacional d'Estàndards i Tecnologia (2001). *Advanced Encryption Standard (AES)*. (núm. 197). <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf>

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

Menezes, A.; Oorschot, P. van; Vanstone, S. (1996). *Handbook of Applied Cryptography*. Florida: CRC Press. <http://cacr.uwaterloo.ca/hac/>

Paar, C.; Pelzl, J. (2009). *Understanding Cryptography: A Textbook for Students and Practitioners*. Nova York: Springer.

Rukhin, A.; Soto, J.; Nechvatal, J. i altres (2010). *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications* . Maryland: Institut Nacional d'Estàndards i Tecnologia.
<http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf>

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