# 3. Les xifres de flux

Ja hem vist en anteriors capítols que un criptosistema incondicionalment segur necessita tants bits de clau com bits de text a 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 a 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 dóna la paradoxa que si tenim un canal segur per intercanviar les claus, aleshores també podem utilitzar-lo per 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, obtenint-ne 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 xifrar el text en clar. Caldrà doncs, veure quines propietats hauran de complir les esmentades 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 critosistemes 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. En el Capítol 4 es tracten amb més detall les xifres de bloc.

Com veurem en aquest capítol, el xifrat de flux utilitza una clau diferent per cada bit d'informació. Aquesta clau depèn de l'estat inicial del generador, però també de l'estat del generador en el moment de xifrar un bit concret. Per tant, dos bits iguals es poden xifrar de maneres diferents

depenent de l'estat en què es trobi el generador. En el proper capítol veurem que 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.

## 3.1 Criptografia de clau simètrica o compartida

En aquest capítol introduirem les xifres de clau compartida.

> Les **xifres de clau simètrica o compartida** són aquelles en els quals l'emissor i el receptor comparteixen una mateixa clau, que és la que utilitzen per xifrar i desxifrar missatges.

És a dir, en les xifres de clau compartida, la clau que s'utilitza per xifrar és la mateixa que es fa servir per desxifrar i per tant, en qualsevol moment, l'emissor pot passar a fer de receptor i a l'inrevés, 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 a bit, és a dir, els bits es xifren de manera invididual, mentre que els criptosistemes de bloc xifren un bloc sencer de $n$ bits alhora.

## 3.2 Definició de les xifres de flux

D'una manera esquemàtica, un **criptosistema de flux** es pot expressar tal com mostra la Figura 3.1.

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

Per tal de 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 a 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 capítol les seqüències amb les que treballarem seran binàries i les operacions a les que ens referirem seran totes mòdul 2.[1]

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 següent bit de sortida. Idealment, el que es necessita per a la seguretat incondicional és que la clau, en aquest cas la seqüència xifrant, sigui

[1] De forma equivalent, podem pensar la suma mòdul 2 com una XOR.

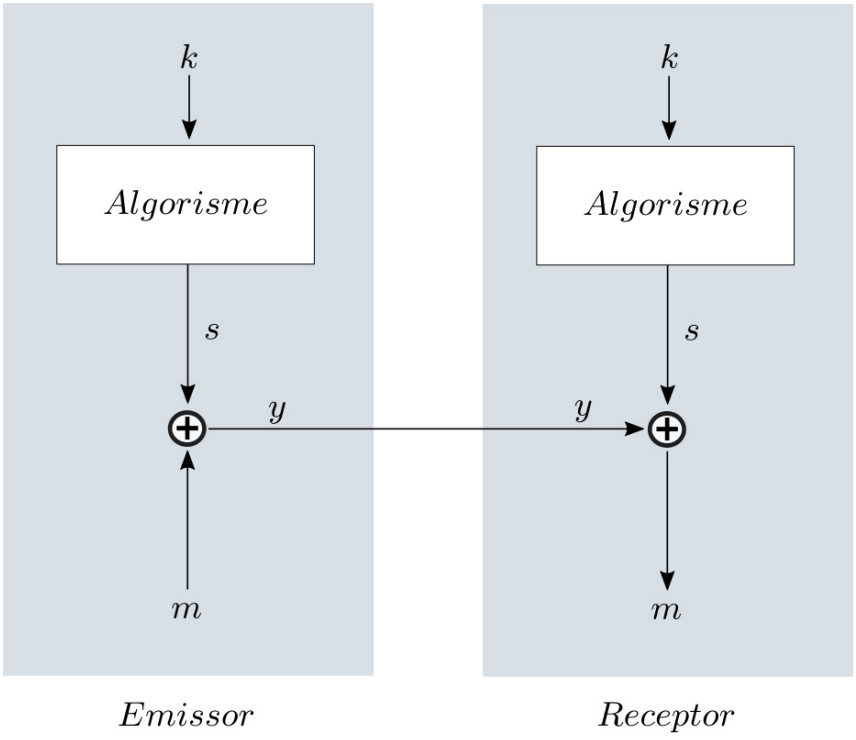



Figura 3.1: Esquema general de flux

completament aleatòria. En el nostre esquema no es pot donar aquesta condició ja que el generador que utilitzem ha de ser determinista per tal que 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 en els propers apartats.

No hem d'oblidar que els criptosistemes de clau compartida basen la seva seguretat en el fet que la clau utilitzada per 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 xifrar, cal igualment que no es faci pública ja que l'algorisme determinista es 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 3.1 veiem que per tal d'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 hardware o en software), 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 de l'esmentat xifrador.

### 3.2.1 Període

Hem vist que per tal d'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 tal que $s_{i+p} = s_i$ 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 Mbyte/seg. ja que a aquesta velocitat el període es repeteix només cada 8.5 minuts.

### 3.2.2 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 en els 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, per a que un PRNG sigui considerat un CSPRNG, cal que les seqüències que genera tinguin dues propietats. A partir de $k$ bits de la seqüència generada, $s_{i+1}, s_{i+2}, \cdots, s_{i+k}$:

1. No existeix un algorisme en temps polinomial que pugui predir el següent bit de la seqüència, $s_{i+k+1}$, amb probabilitat major al 50% i
2. No és computacionalment possible predir el bit anterior de la seqüència, $s_i$.

**PRNG i CSPRNG**

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

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 tal de 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 tal de 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: a través de hardware, explotant l'aleatorietat que es produeix en fenòmens físics, o a través de software, a partir d'observacions afectades pel comportament de l'usuari. Així, per exemple, es pot fer servir el so capturat per un micròfon, el soroll tèrmic[2] d'una resistència o d'un diode, les turbolències creades per l'aire en segons quins dispositius o el moviment del ratolí.

#### Tests d'aleatorietat del NIST

El NIST disposa d'un banc de proves estadístiques per avaluar l'aleatorietat de seqüències binàries generades per PRNG. El banc consta de 15 testos. En aquest apartat, descriurem els testos 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.

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 fer-ho, en primer lloc es transforma la seqüència binària d'entrada a una seqüència de $-1$ i 1:

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

és a dir, els zeros es converteixen en $-1$ i els uns segueixen representant-se amb 1.

Després, es calcula $s_{obs}$:

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

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 major a zero.

**Superació dels testos**

A partir del valor $s_{obs}$, els testos 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 major o igual a 0.01. El lector interessat en els detalls sobre els testos estadístics pot consultar la publicació original del NIST: A. Rukhin, J. Soto, J. Nechvatal, et al. (2010). *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*.

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

**Exemple 3.1 Càlcul de la prova de freqüència de bits individuals**

Donada la seqüència:
$S = \{0,1,0,1,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,1,0,1,0,1,0,1,0,0,0\}$
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:

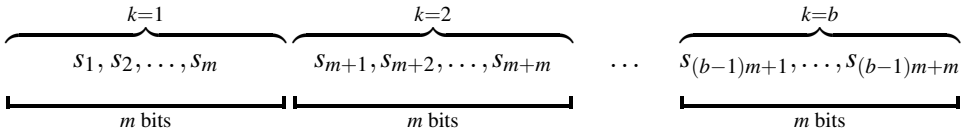
$$X = \{-1,1,-1,1,-1,-1,-1,-1,-1,-1,-1,1,1,-1,-1,1,1,1,1,-1,1,-1,1,-1,\\ 1,-1,-1,-1\}$$

Seguidament, calculem el valor $s_{obs}$:

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

| **Superació del test** | En aquest cas, el nivell de significació per a $s_{obs} = 1.1339$ és de 0.256, de manera que el test es considera 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 a avaluar en $b = \lfloor n/m \rfloor$ blocs de $m$ bits, descartant els bits sobrants.




Aleshores, per 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 3.2 Càlcul de la prova de freqüència en un bloc**

Donada la mateixa seqüència que en l'exemple anterior:
$S = \{0,1,0,1,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,1,0,1,0,1,0,1,0,0,0\}$
amb $n = 28$, procedim a calcular $\pi_k$ 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

$$\overbrace{0,1,0,1,0,0}^{k=1}\ \overbrace{0,0,0,0,0,1}^{k=2}\ \overbrace{1,0,0,1,1,1}^{k=3}\ \overbrace{1,0,1,0,1,0}^{k=4}\ 1,0,0,0$$
$$\underbrace{\qquad}_{6\text{ bits}}\ \underbrace{\qquad}_{6\text{ bits}}\ \underbrace{\qquad}_{6\text{ bits}}\ \underbrace{\qquad}_{6\text{ bits}}$$

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

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

$$\pi_1 = \frac{\sum_{j=1}^{m} s_{(k-1)m+j}}{m} = \frac{\sum_{j=1}^{6} s_j}{6} = \frac{2}{6} = 1/3$$
$$\pi_2 = \frac{\sum_{j=1}^{6} s_{6+j}}{6} = 1/6$$
$$\pi_3 = \frac{\sum_{j=1}^{6} s_{2\cdot 6+j}}{6} = \frac{4}{6} = 2/3$$
$$\pi_4 = \frac{\sum_{j=1}^{6} s_{3\cdot 6+j}}{6} = \frac{3}{6} = 1/2$$

I finalment estem en disposició de calcular $\chi^2_{obs}$:

$$\begin{aligned}\chi^2_{obs} &= 4m \sum_{k=1}^{b} (\pi_k - 1/2)^2 \\ &= 4 \cdot 6 \sum_{k=1}^{4} (\pi_k - 1/2)^2 \\ &= 24 \cdot \left((1/3 - 1/2)^2 + (1/6 - 1/2)^2 + (2/3 - 1/2)^2 + (1/2 - 1/2)^2\right) \\ &= 24 \cdot (1/36 + 1/9 + 1/36 + 0) = 4\end{aligned}$$

| **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, fent 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 avaluar la prova de ràfegues, es calcula:

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

on $r(i)$ és la funció:

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

**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.

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

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 3.3 Càlcul de la prova de ràfegues**

Seguint amb l'avaluació de la mateixa seqüència que en els exemples anteriors:
$S = \{0,1,0,1,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,1,0,1,0,1,0,1,0,0,0\}$
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 $V_{28}(obs)$.

$$\begin{aligned} V_n(obs) &= \left(\sum_{i=1}^{n-1} r(i)\right) + 1 \\ &= (1+1+1+1+0+0+0+0+0+0+1+0+1+0+1+0+ \\ &\quad +0+0+1+1+1+1+1+1+1+0+0)+1 = 15 \end{aligned}$$

**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, fent que el test es consideri superat ja que $0.5151 \geq 0.01$.

Com s'ha comentat, el banc de proves del NIST recull 15 testos diferents. A més dels tres comentats, els altres testos 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.

## 3.3 Generadors lineals de seqüència xifrant

En l'apartat anterior, 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 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.

### 3.3.1 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 hi 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 3.4** La funció *rand*() del sistema UNIX BSD utilitza el següent generador congruencial afí:

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

on la llavor especifica el valor inicial.

### 3.3.2 Registres de desplaçament realimentats linealment (LFSR)

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 3.2.

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:

$$s_{n+1} = c_1 s_n + \cdots + c_n s_1 \tag{3.1}$$

on els $c_i \in \{0, 1\}$ corresponen als valors de les portes lògiques de l'esquema. És a dir, els coeficients

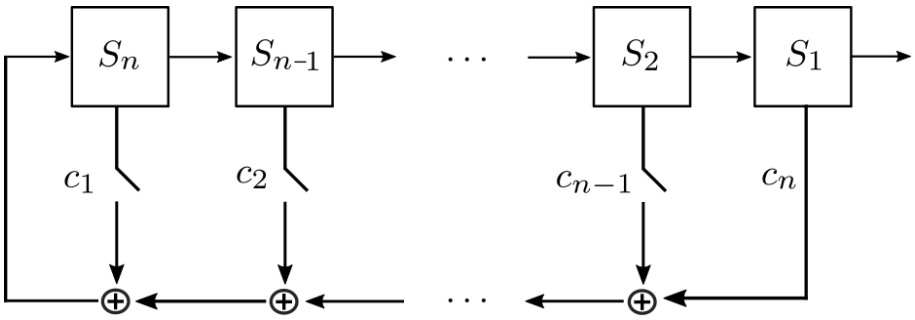


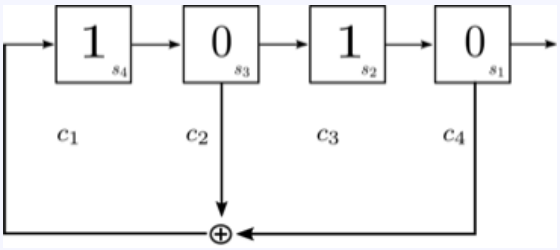

Figura 3.2: Esquema general d'un LFSR

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 3.5 Funcionament d'un LFSR**

Aquest exemple segueix el funcionament de l'LFSR de 4 cel·les representat en la següent figura:




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

($t = 1$). Noteu com 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:

$$
\begin{aligned}
s_1(t = i) &= s_2(t = i-1) \\
s_2(t = i) &= s_3(t = i-1) \\
s_3(t = i) &= s_4(t = i-1) \\
s_4(t = i) &= s_1(t = i-1) \oplus s_3(t = i-1)
\end{aligned}
$$

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 seves 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 3.1 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

$$
s_{n+k} = \sum_{i=1}^{n} c_i s_{n+k-i} \quad \text{per } k \geq 0 \tag{3.2}
$$

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

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

$$
C(x) = 1 + c_1 x^1 + c_2 x^2 + \cdots + x^n
$$

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

**Exemple 3.6 Polinomi de connexions d'un LFSR**

El polinomi de connexions corresponent a l'LFSR de l'exemple 3.5 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:

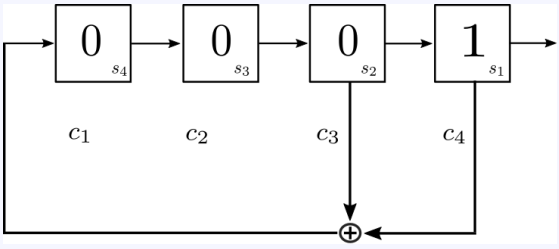
1. Polinomi de connexions **factoritzable**: els LFSR's que tenen polinomis de connexions factoritzables generen seqüències que depenen de l'estat inicial. A més, el període de les esmentades seqüències és sempre més petit que el període màxim que pot tenir un LFSR, que és $2^n - 1$.
2. Polinomi de connexions **irreductible**: igual que en el cas anterior, un LFSR amb un polinomi de connexions irreductible (però que no sigui primitiu) genera seqüències que depenen de l'estat inicial de l'LFSR i, en aquest cas, el seu període és 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 és també irreductible. Per tant, les seqüències generades per 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 3.7 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 al fer servir els valors 0001 com a estat inicial.




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 com la seqüència $L_2$ és la seqüència $L_1$ desplaçada 9 posicions a l'esquerra:

$$\begin{array}{lll} L_1 = & 100010011 & 0101111000100110101 \\ L_2 = & & 0101111000100110101 \end{array}$$

Per últim, aprofitarem l'exemple per mostrar perquè no podem generar una seqüència de període superior amb un LFSR de 4 cel·les. La taula següent mostra tots els estats per els quals passa l'LFSR per generar la seqüència $L_1$:

| $t$ | Estat | Sortida | $t$ | Estat | Sortida |
|---|---|---|---|---|---|
| 0 | 0001 | 1 | 8 | 0101 | 1 |
| 1 | 1000 | 0 | 9 | 1010 | 0 |
| 2 | 0100 | 0 | 10 | 1101 | 1 |
| 3 | 0010 | 0 | 11 | 1110 | 0 |
| 4 | 1001 | 1 | 12 | 1111 | 1 |
| 5 | 1100 | 0 | 13 | 0111 | 1 |
| 6 | 0110 | 0 | 14 | 0011 | 1 |
| 7 | 1011 | 1 | 15 | 0001 | 1 |

Fixeu-vos que l'estat a $t = 15$ correspon a l'estat inicial ($t = 0$), motiu per el qual la seqüència comença a repetir-se. Noteu també com l'LFSR passa per 15 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 $L_2$.

**Exercici 3.1** 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 $0,0,0,1,1$.

### 3.3.3 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 $\phi$ 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 3.2 per plantejar el següent sistema d'equacions:

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

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.[3]

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[4] 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.

**Exercici 3.2** Quin és el període i la complexitat lineal màxima de les seqüències generades per l'LFSR amb polinomi de connexions $1 + x^2 + x^5$?

**Exercici 3.3** Donada la seqüència $s = 00010011010111100010$ , sintetitzeu l'LFSR que l'ha generada, sabent que el polinomi de connexions té grau 4.

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.

## 3.4 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.

[3] 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.

[4] 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*.

### 3.4.1 A5

L'A5 és un algorisme de xifrat en flux que s'utilitza per al xifrat de dades en les transmissions de la xarxa GSM[5].

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 propi 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 poder complir amb les regulacions per exportar criptografia (i feta servir als Estats Units) i l'A5/3 és un algorisme de xifratge totalment diferent (afegida amb posterioritat). En aquest apartat, descriurem el funcionament de l'algorisme A5/1.

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'quest paradigma està totalment desconsellat pels experts ja que viola el principi de Kerckhoffs.[6] 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.

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 (el que en el llenguatge de telefonia mòbil es coneix com 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-ne 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 propi 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 3 LFSR tal com es mostra en la Figura 3.3.

La Taula 3.1 detalla les longituds de cadascun dels LFSR de l'A5/1 així com els seus polinomis de connexions.

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 amb sombrejat, 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

[5]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.

[6]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.

irregular segons la funció majoritària.

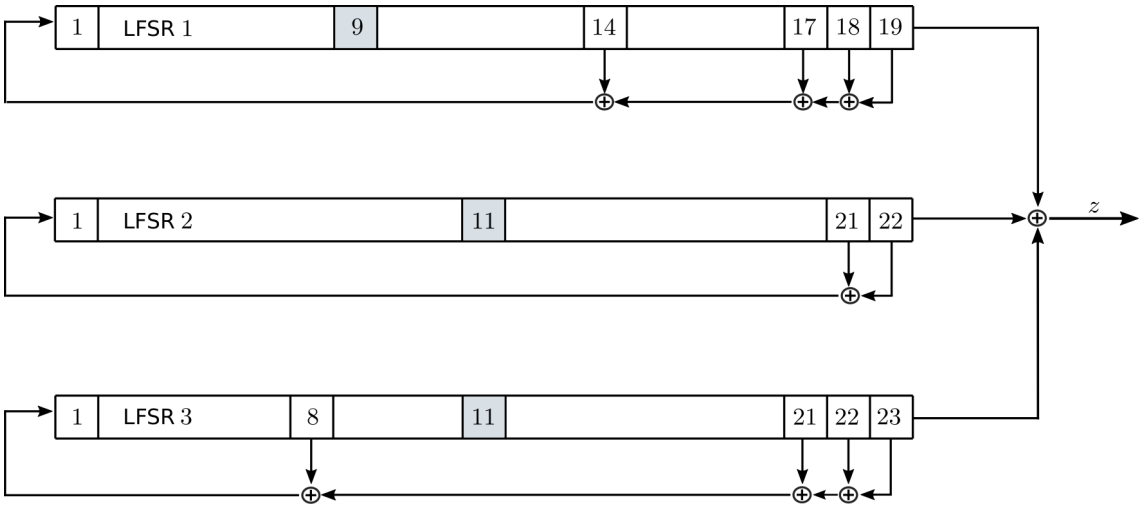



Figura 3.3: Esquema de l'A5.

| 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 |

Taula 3.1: Descripció dels LFSR de l'A5.

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 l'XOR que acabarà proporcionant cada bit de la seqüència de xifratge.

**Exemple 3.8 Iteració de l'A5**

Si en l'instant $t$ tenim els següents estats interns[a] en els LFSR:
LFSR1: `1011100011 011000010`
LFSR2: `1011011011 1101000010 01`
LFSR3: `1110111110 0111001000 001`

La sortida en aquest mateix instant de temps $t$ serà doncs:
$z = 0 \oplus 1 \oplus 1 = 0$ i 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 en el següent 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 en $t+1$ és:
LFSR1: `1101110001 101100001`

LFSR2: 1101101101 1110100001 00
LFSR3: 1110111110 0111001000 001

[a] 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 polsacions 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 3.4 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 $64+22=86$ polsacions de rellotge fent una xor amb cadascun dels bits de la clau de sessió seguida del número de trama.

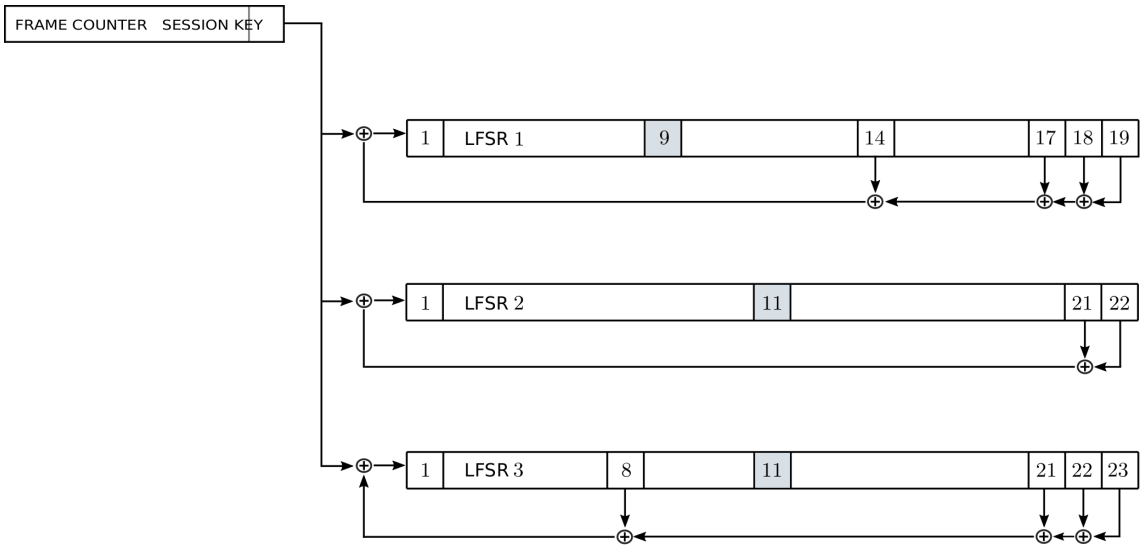



Figura 3.4: Esquema dels passos 2 i 3 de la inicialització de l'A5.

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.

Per tal de xifrar la següent trama 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 a xifrar, sinó que, en el context de telefonia mòbil en el que s'utilitza aquest sistema, la clau de sessió s'actualitza quan la xarxa decideix tornar a autenticar el dispostiu mòbil.

### 3.4.2 Trivium

El Trivium és un generador pseudoaleatori dissenyat pels criptògrafs Christophe De Cannière i Bart Preneel que aprofita una implementació hardware 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 en 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 3 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. En la Figura 3.5 podem veure l'esquema del Trivium.

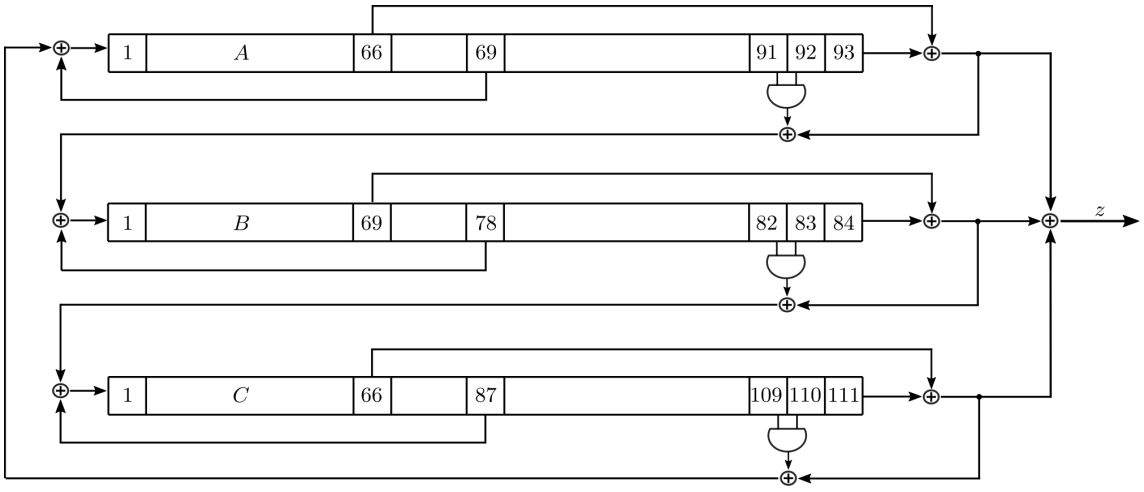



Figura 3.5: Esquema del Trivium.

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 ($t^a$, $t^b$, $t^c$):

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

on 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 següent manera:

$$t^a = s^a_{93} + s^a_{66}$$
$$t^b = s^b_{84} + s^b_{69}$$
$$t^c = s^c_{111} + s^c_{66}$$

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 $c$; i 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:

$$s^a_{new} = t^c + (s^c_{109} \cdot s^c_{110}) + s^a_{69}$$
$$s^b_{new} = t^a + (s^a_{91} \cdot s^a_{92}) + s^b_{78}$$
$$s^c_{new} = t^b + (s^b_{82} \cdot s^b_{83}) + s^c_{87}$$

on, de nou, tots els operands són bits i tant la suma com el producte[7] d'aquesta expressió es realitzen mòdul 2.

La taula següent resumeix les accions que realitza cada posició específica de cada un dels registres:

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

Taula 3.2: Taula 3.2. Posicions destacades dels registres del Trivium.

**Exemple 3.9 Iteració del Trivium**

Si en l'instant $t$ tenim els següents estats interns$^a$ en els registres:

A: `0111111100 1101111010 1111100101 1101010001 0111100010 0110110001 1100110111 1111100110 0101011100 011`

B: `0100001010 1111011011 0110101000 1100010001 1111011000 1011110001 1101110100 0111100001 1011`

C: `1111110100 0111011101 0101111100 1010111100 0100011100 0001111011 1000011010 0111000011 1101010011 0101001000 0100000011 0`

[7] De forma equivalent, també podem pensar el producte mòdul 2 com un AND.

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 el seu 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`

[a] 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.

### 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 en 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 3 cel·les de més a la dreta del registre C, en les que s'hi inclou un 1 en cada un d'elles. La Figura 3.6 mostra gràficament la inicialització del Trivium.

Una vegada s'han situat aquests valors en els estats dels 3 registres, s'executen 1152 cicles de rellotge descartant els bits de sortida d'aquestes 1152 iteracions.

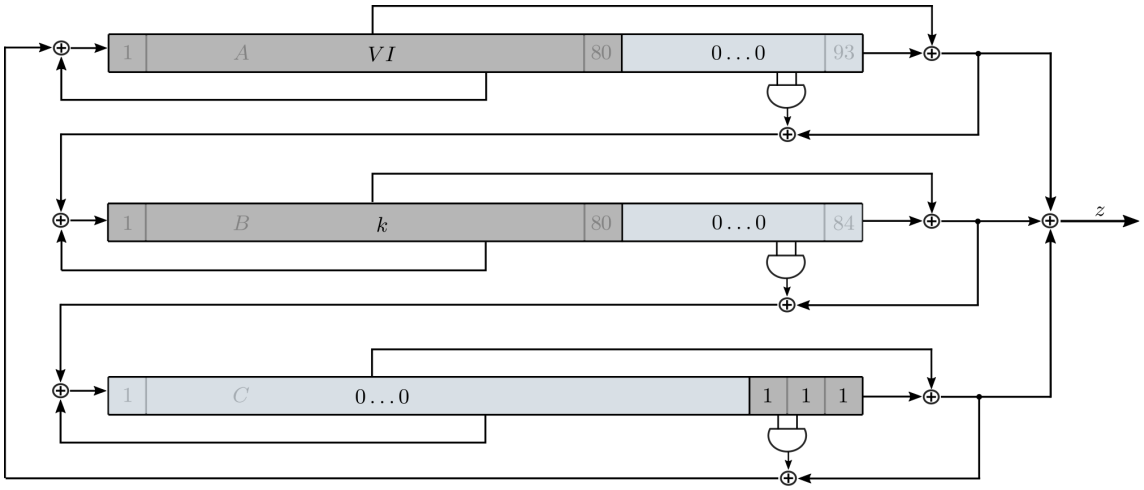



Figura 3.6: Fase d'inicialització del Trivium.

Finalment, una vegada s'ha inicialitzat el generador ja es pot utilitzar la seqüència de sortida per 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 a xifrar.

Per desxifrar un missatge utilitzant el Trivium, caldrà realitzar exactament el mateix procés 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.

## 3.5 Resum

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

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

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

## 3.6 Solucions dels exercicis

**Exercici 3.1:**

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) en l'instant de temps anterior:

| Impuls de rellotge ($t$) | Estat | Sortida |
|---|---|---|
| 0 | 0 $\underline{0}$ 0 1 $\underline{1}$ | 1 |
| 1 | 1 $\underline{0}$ 0 0 $\underline{1}$ | 1 |
| 2 | 1 $\underline{1}$ 0 0 $\underline{0}$ | 0 |
| 3 | 1 $\underline{1}$ 1 0 $\underline{0}$ | 0 |
| 4 | 1 $\underline{1}$ 1 1 $\underline{0}$ | 0 |
| 5 | 1 $\underline{1}$ 1 1 $\underline{1}$ | 1 |
| 6 | 0 $\underline{1}$ 1 1 $\underline{1}$ | 1 |
| 7 | 0 $\underline{0}$ 1 1 $\underline{1}$ | 1 |
| 8 | 1 $\underline{0}$ 0 1 $\underline{1}$ | 1 |
| 9 | 1 $\underline{1}$ 0 0 $\underline{1}$ | 1 |
| 10 | 0 $\underline{1}$ 1 0 $\underline{0}$ | 0 |
| 11 | 1 $\underline{0}$ 1 1 $\underline{0}$ | 0 |
| 12 | 0 $\underline{1}$ 0 1 $\underline{1}$ | 1 |
| 13 | 0 $\underline{0}$ 1 0 $\underline{1}$ | 1 |
| 14 | 1 $\underline{0}$ 0 1 $\underline{0}$ | 0 |

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

**Exercici 3.2:**

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 $2^n-1=2^5-1=31$.:

**Exercici 3.3:**

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 següent sistema d'equacions:

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

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 $x^4+x^3+1$.

## 3.7 Bibliografia

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

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

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

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

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

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

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

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

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