Criptosistemas de clave compartida: cifrado en flujo

Jordi Herrera Joancomartí

PID_00214916
Módulo 3

UOC
Universitat Oberta de Catalunya

Índice

Introducción 5

Objetivos 6

1. Requisitos de las secuencias de cifrado en flujo 7
1.1. Periodo 8
1.2. Postulados de Golomb 9
1.3. Complejidad lineal 10
1.4. Implementabilidad 10

2. Generadores lineales 11
2.1. Generadores congruentes 11
2.2. Registros de desplazamiento realimentados linealmente 12
2.3. Limitaciones de los generadores lineales 14

3. Generadores no lineales 16
3.1. Registros de desplazamiento realimentados no linealmente 16
3.2. Filtrado no lineal 16
3.3. Combinadores no lineales 17
3.3.1. Generador de Geffe 18
3.3.2. Generador de Beth-Piper 19
3.3.3. Generador multivelocidad de Massey-Rueppel 19

Resumen 21

Actividades 23

Ejercicios de autoevaluación 23

Solucionario 24

Glosario 25

Bibliografía 26

Introducción

Hemos visto en otros módulos que un criptosistema incondicionalmente seguro necesita, como mínimo, tantos bits de clave como bits de texto para cifrar y que el cifrado de Vernam es lo único que consigue esta seguridad incondicional, pero el precio que paga por eso es la ineficiencia del cifrado. Esta ineficiencia recae justamente en el hecho de que la clave tiene que tener la misma longitud que el texto que se tiene que cifrar. Eso comporta que la longitud de las claves sea muy grande y, por lo tanto, sea más difícil mantenerlas en secreto. Además, se da la paradoja de que si tenemos un canal seguro para intercambiar las claves también lo podemos utilizar para intercambiar los mensajes, ya que tienen la misma longitud.

Ved el módulo "Fundamentos de criptografía" de esta asignatura.

Las **cifras en flujo** surgen como una aproximación optimizada del cifrado de Vernam. La idea es construir una clave lo bastante larga, cuando menos de la longitud del mensaje, a partir de una clave inicial corta mediante lo que se llama generador pseudoaleatorio. Este generador expande una clave corta, llamada semilla, para obtener una mucho más larga. Además, la operación de expansión tiene que tener unas características determinadas, ya que la secuencia que resulta se utiliza para cifrar el texto en claro.

Es necesario, pues, analizar las propiedades que tienen que cumplir las secuencias mencionadas y estudiar qué tipos de generadores hay para obtenerlas.

Objetivos

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

1. Comprender el esquema general de las cifras en flujo.

2. Asimilar las características que tiene que cumplir necesariamente una secuencia pseudoaleatoria para que sea utilizable en un esquema de cifrado en flujo.

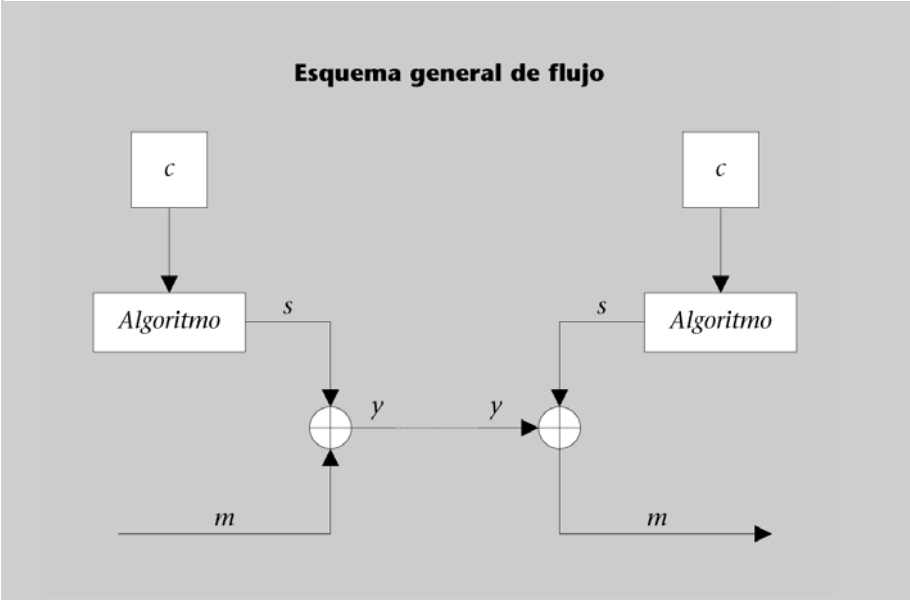
3. Entender el funcionamiento de diferentes generadores pseudoaleatorios.

1. Requisitos de las secuencias de cifrado en flujo

Las cifras en flujo se incluyen dentro de los llamados *criptosistemas de clave compartida*. Los **criptosistemas de clave compartida** son aquellos en los que el emisor y el receptor comparten una misma clave para cifrar y descifrar mensajes.

En estos criptosistemas, la clave que se utiliza para cifrar el mensaje es la misma que se emplea para descifrarlo; por lo tanto, al utilizar siempre la misma clave, en cualquier momento el emisor puede pasar a ser receptor y a la inversa.

De una manera esquemática, un criptosistema de flujo se puede representar como muestra la figura siguiente:



Los elementos que intervienen...

... en el cifrado de flujo son: c, la clave compartida; *Algoritmo*, el generador pseudoaleatorio; m, el mensaje en claro; s, la secuencia de cifrado, e y, la secuencia cifrada.

Tanto el emisor como el receptor disponen de una misma clave c, denominada **semilla del generador**, y de un mismo algoritmo determinista, llamado **generador pseudoaleatorio**. Al proporcionar la clave como entrada al algoritmo, éste genera a la salida una secuencia nombrada **secuencia de cifrado**.

Para cifrar el mensaje, el emisor va sumando cada bit del mensaje en claro con cada bit de la secuencia de cifrado. Cuando el receptor recibe el mensaje cifrado utiliza el mismo algoritmo determinista, que en la figura anterior hemos denotado como *Algoritmo*, y la clave c, que comparte con el emisor, para obtener la misma secuencia de cifrado. Así, sumando bit a bit el mensaje que le llega y con la secuencia resultante del algoritmo, s, el receptor obtiene el texto en claro m enviado por el emisor.

Todas las secuencias con que trabajaremos en este módulo serán binarias, y las operaciones a que nos referiremos serán todas módulo 2.

Para que este criptosistema resulte seguro, es fundamental que la secuencia de cifrado no sea conocida; es decir, que no se pueda saber en ningún momento cuál será el siguiente bit de salida. Idealmente, para garantizar la seguridad incondicional es necesario que la clave, en este caso la secuencia de cifrado, sea completamente aleatoria.

En nuestro esquema no se da esta condición, ya que el generador que utilizamos tiene que ser determinista para que el emisor y el receptor obtengan la misma secuencia al dar como entrada la misma clave secreta.

La secuencia de cifrado tendrá propiedades muy parecidas a las que tiene una secuencia completamente aleatoria y se denominará **secuencia pseudoaleatoria**.

En concreto, si una secuencia no es aleatoria, se repite a partir de un cierto momento. La subsecuencia que se repite se denomina *periodo*. Es importante que esta subsecuencia, el periodo, sea indistinguible de una secuencia completamente aleatoria de la misma longitud, y para conseguirlo, ésta tiene que cumplir unas propiedades determinadas, que veremos más adelante.

Ved los postulados de Golomb en el subapartado 1.2 de este módulo.

No debemos olvidar que la seguridad de los criptosistemas de clave compartida se basa en que la clave utilizada para cifrar y descifrar sólo es conocida por el emisor y el receptor. En el cifrado de flujo, aunque la clave no se utiliza directamente para cifrar, también es necesario que no se haga pública, ya que el algoritmo determinista es conocido y a partir de éste y de la clave se podría obtener la secuencia de cifrado.

1.1. Periodo

Hemos visto que para implementar un criptosistema de cifrado en flujo necesitamos un algoritmo que nos de como salida la secuencia de cifrado. El hecho de que este algoritmo sea determinista implica que la secuencia que resulta no sea completamente aleatoria, es decir, que se repita a partir de un cierto momento. Esta subsecuencia que se repite es el *periodo*.

Formalmente podemos definir el periodo de la manera siguiente: sea $\{s_i\}_{i\geq 0}$ una secuencia periódica, el **periodo** $\boldsymbol{p}$ es el entero más pequeño tal que $s_{i+p} = s_i$ para todo $i \geq 0$.

Una vez conocido el periodo, su característica periódica nos permite determinar la secuencia de cifrado exacta y romper el criptosistema. Por este motivo, interesa que las secuencias que se utilicen para el cifrado de flujo tengan un periodo largo, que tarden mucho en repetirse y, de este modo, sea más difícil predecir su salida.

El concepto de *periodo largo*...

... se refiere al cifrador y a la aplicación. Un periodo de 2^{32} puede no ser suficientemente largo para un cifrador que cifre a 1 megabyte por segundo, ya que a esta velocidad el periodo se repite sólo cada 8,5 minutos.

1.2. Postulados de Golomb

Para conseguir una secuencia que nos sirva para el cifrado flujo no basta con un periodo largo. También hace falta que la distribución de los ceros y los unos que la forman tenga una cierta uniformidad. En concreto, para que una secuencia se pueda considerar pseudoaleatoria tiene que cumplir los tres postulados de Golomb. Sin embargo, Antes de estudiar los postulados, tendremos que introducir algo de terminología.

Definimos una **ráfaga** como un conjunto de bits consecutivos iguales; es decir, una ráfaga de longitud k es el conjunto de los elementos $s_t, \ldots, s_{t+k-1}$ tales que $s_{t-1} \neq s_t = s_{t+1} = \ldots = s_{t+k-1} \neq s_{t+k}$.

La **función de autocorrelación**, $\boldsymbol{AC(k)}$, de una secuencia periódica $\{s_i\}_{i\geq 0}$ de periodo p se define como:

$$AC(k) = \frac{A-D}{p},$$

donde A y D son, respectivamente, el número de coincidencias y de no coincidencias de todo el periodo entre la sucesión $\{s_i\}_{i\geq 0}$ y la misma sucesión desplazada k posiciones, $\{s_{i+k}\}_{i\geq 0}$. Es decir:

- $A = |\{0 \leq i < p \text{ tal que } s_i = s_{i+k}\}|$
- $D = |\{0 \leq i < p \text{ tal que } s_i \neq s_{i+k}\}|$

Observad que si k es un múltiplo de p, entonces $AC(k) = 1$

Ejemplo de función de autocorrelación

Dada la secuencia de período p = 15 siguiente:

1101000110011101101...

La función de autocorrelación correspondiente es: $AC(12) = \frac{7-8}{15} = -\frac{1}{15}$, ya que las secuencias:

110100011001110 y 110110100011001

presentan 7 coincidencias y 8 no-coincidencias..

Los postulados de Golomb que ha de verificar cualquier secuencia pseudoaleatoria, establecen que:

1) En un periodo de secuencia pseudoaleatoria, el número de ceros y de unos tiene que ser el mismo o diferir como máximo en una unidad; es decir: $p/2$ si p es par y $(p \pm 1)/2$, si es impar.

2) El número total de ráfagas de longitud k en un periodo tiene que valer como mínimo $n/2^k$, donde n es el número total de ráfagas del periodo.

3) La función de autocorrelación $AC(k)$ es bivaluada, es decir, sólo toma dos valores: 1 si k es múltiplo de p, y otro valor constante, si p no es divisor de k.

Es preciso añadir que, aunque estos postulados dan información sobre la aleatoriedad de las secuencias, una vez verificados, es necesario recurrir además a varios tests estadísticos, como χ^2 o los tests espectrales.

1.3. Complejidad lineal

Para utilizar una secuencia pseudoaleatoria para el cifrado de flujo, cada bit ha de depender lo mínimo posible del anterior. Es decir, la secuencia de cifrado debe tener un grado elevado de impredictibilidad. La probabilidad de que salga un 0 o un 1 ha de ser próxima a 0,5 y su dependencia del del bit de salida anterior debe ser mínima.

El concepto de **complejidad lineal** mide esta impredecibilidad, es decir, establece qué parte de la secuencia es preciso conocer para poder obtener la secuencia entera.

Para calcular la complejidad lineal utilizaremos el algoritmo de Massey, que detallaremos más adelante.

Ved la referencia al algoritmo de Massey en el subapartado 2.3 de este módulo.

1.4. Implementabilidad

En el esquema de cifrado en flujo de la figura anterior se puede observar que para obtener el texto cifrado hay que sumar el texto en claro con la secuencia de cifrado que produce el generador pseudoaleatorio. Ello significa que la velocidad de transmisión de datos entre el emisor y el receptor viene determinada por el valor mínimo entre la velocidad de generación del mensaje y la velocidad de generación de la secuencia de cifrado. Es importante tener en cuenta este hecho al estudiar los posibles generadores pseudoaleatorios, ya que, según el tipo de implementación (*hardware* o *software*), la velocidad será una u otra.

Los teléfonos móviles...

... con tecnología GSM incorporan un cifrador de flujo. Sería impensable que el coste económico del cifrador incrementara el precio del teléfono móvil. Tampoco sería admisible que la velocidad de la comunicación se viera afectada por la velocidad del cifrador utilizado.

El algoritmo que genere la secuencia de cifrado ha de ser fácil de implementar tanto en complejidad como en coste económico.

2. Generadores lineales

Hemos estudiado las propiedades que han de tener las secuencias de cifrado para poderlas utilizar en criptosistemas de cifrado de flujo. Ahora analizamos cómo deben ser los algoritmos deterministas que generan estos tipos de secuencias. Desde un punto de vista global, hay dos tipos de generadores:

1) Los **generadores lineales** ejecutan sólo operaciones lineales sobre los elementos de entrada para obtener la secuencia de salida.

2) Los **generadores no lineales** ejecutan además operaciones no lineales, como permutaciones.

2.1. Generadores congruentes

Los generadores congruentes se basan en ecuaciones modulares recurrentes del tipo:

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

En este caso, el valor x_0 sería la semilla de la secuencia de cifrado. Un criptosistema que utilice un generador de este tipo ha de tener como clave secreta los valores $\{x_0, a, b, m\}$, y para que el periodo sea máximo, se tiene que cumplir que $\mathrm{mcd}(a, m) = 1$.

Estos generadores pseudoaleatorios no son, sin embargo, seguros desde un punto de vista criptográfico, ya que se ha demostrado que es posible obtener los parámetros secretos $\{x_0, a, b, m\}$ a partir de unos pocos valores x_i conocidos. Incluso se puede llegar a determinar el valor de la semilla x_0, si se conoce sólo una parte de los bits que forman los x_i (aunque conociendo los parámetros $\{a, b, m\}$). A pesar de ello, estos generadores son muy utilizados en aplicaciones no criptográficas de sistemas informáticos.

Ejemplos de generadores congruentes

La función *rand*() del sistema Unix utiliza el generador congruente afín siguiente:

$$x_n = (1.103.515.245x_{n-1} + 12.345) \bmod 2^{31}.$$

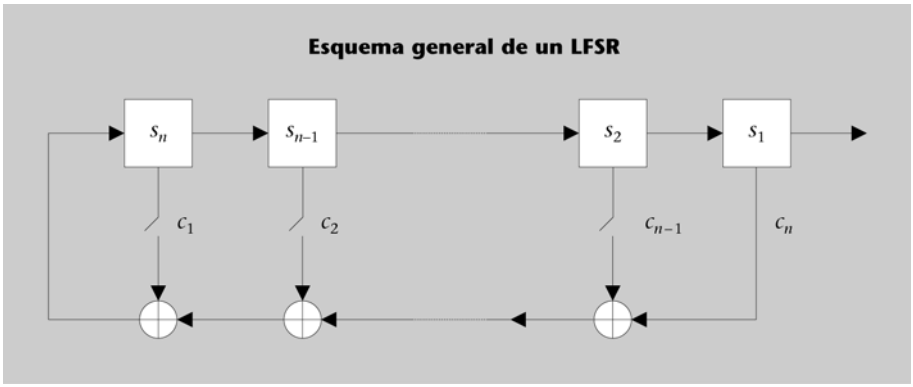
La misma función *rand*() de TurboPascal obtiene el resultado a partir de la congruencia siguiente:

$$x_n = (129x_{n-1} + 907.633.385) \bmod 2^{32}.$$

2.2. Registros de desplazamiento realimentados linealmente

Un registro de desplazamiento de realimentación lineal (LFSR), de longitud n, es un dispositivo físico o lógico formado por n celdas de memoria y n puertas lógicas, tal como muestra la figura siguiente:

LFSR son las siglas del término inglés *Lineal Feedback Shift Register*.



Al principio, las celdas contienen los valores de entrada, y a cada impulso de reloj el contenido de la celda s_i se desplaza a la celda s_{i-1}, y así se efectúan las operaciones asociadas. El elemento nuevo, s_{n+1}, que se genera de este modo viene determinado por:

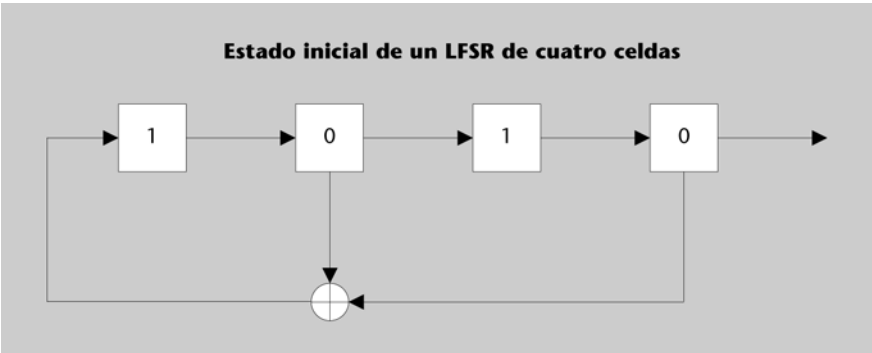
$$s_{n+1} = c_1 s_n + \ldots + c_n s_1, \tag{2.1}$$

donde los $c_i \in \{0, 1\}$ corresponden a los valores de las puertas lógicas del esquema. Es decir, los coeficientes serán 1 si hay una conexión, y 0 si no hay. Este nuevo elemento, s_{n+1}, se sitúa en la celda s_n, que ha quedado vacía a causa del desplazamiento.

El conjunto de valores contenidos en cada celda en un instante de tiempo se denomina **estado**. El **estado inicial** es el estado en que se halla el LFSR en el momento de empezar el proceso.

Funcionamiento de un LFSR de cuatro celdas

Veamos el funcionamiento de un LFSR de cuatro celdas:



Este registro de desplazamiento tiene como estado inicial 1010, que corresponde al impulso de reloj $t = 0$. La evolución del estado en cada ciclo del procesador se muestra en la tabla siguiente:

Impulso de reloj (t)	Estado				Salida
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 el impulso $t = 1$, el conjunto de celdas se desplaza pero no varía, ya que en las celdas s_1 y s_3 hay un 0 para $t = 0$. En cambio, para $t = 2$ ya ha habido modificaciones. En concreto, $s_4 = 0$, ya que $s_4(t = 2) = s_1(t = 1) \oplus s_3(t = 1) = 1 \oplus 1 = 0$. En general, para $i \geq 1$ tenemos:

- $s_4(t = i) = s_3(t = i - 1) \oplus s_1(t = i - 1)$,
- $s_3(t = i) = s_4(t = i - 1)$,
- $s_2(t = i) = s_3(t = i - 1)$,
- $s_1(t = i) = s_2(t = i - 1)$.

Observemos que en el impulso de reloj $t = 6$ volvemos a tener el estado inicial, y a partir de aquí la secuencia se repite. Luego, esta secuencia tiene periodo 6.

Una vez definido un LFSR, podemos estudiar sus características más importantes. La ventaja principal de los LFSR es que su formulación matemática es muy simple, de modo que su estudio resulta bastante claro y completo. Además, al definirse por medio de celdas y puertas lógicas, su *hardware* es fácil de implementar, lo cual permite obtener generadores de gran velocidad.

El estado inicial de un LFSR no puede ser una secuencia de sólo ceros, porque dicha secuencia generaría también una secuencia de sólo ceros, ya que todas las operaciones son lineales. Un estado de sólo ceros se denomina **estado absorbente**. Por lo tanto, el valor del periodo máximo de un LFSR es $2^n - 1$, que se obtiene de considerar todos los estados posibles, 2^n, y eliminar el estado nulo.

De la expresión 2.1 deducimos que cualquier secuencia generada por un LFSR está determinada por el estado inicial $\{s_1, ..., s_n\}$ y por la relación:

$$s_{n+k} = \sum_{i=1}^{n} c_i s_{n+k-i}, \qquad (2.2)$$

donde $c_i \in \{0, 1\}$ para $1 \leq i \leq n$.

El **polinomio de conexiones de un LFSR** de longitud n es un polinomio de grado n, que tiene la forma siguiente:

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

donde los $c_i \in \{0, 1\}$ corresponden a los valores de las puertas lógicas de la figura del esquema general de un LFSR.

Polinomio de conexiones del LFSR de cuatro celdas

El polinomio de conexiones correspondiente al LFSR del ejemplo anterior es el siguiente:
$C(x) = 1 + x^2 + x^4$.

Un LFSR queda determinado por su polinomio de conexiones, y una secuencia, por el polinomio de conexiones y por el estado inicial.

A continuación vamos a determinar las características del LFSR a partir de las de su polinomio de conexiones:

1) Polinomio de conexiones factorizable: los LFSR cuyos polinomios de conexiones son factorizables generan secuencias que dependen del estado inicial. Además, el periodo de estas secuencias es siempre más corto que el periodo máximo que puede tener un LFSR, que es $2^n - 1$.

2) Polinomio de conexiones irreductible: las secuencias generadas por los LFSR cuyos polinomios de conexiones son irreductibles no dependen del estado inicial, sino que simplemente quedan desplazadas. Su periodo será un divisor de $2^n - 1$.

3) Polinomio de conexiones primitivo: un LFSR cuyo polinomio de conexiones sea primitivo, tiene la secuencia de salida de periodo máximo, $2^n - 1$. Esta secuencia de periodo máximo se obtiene para cualquier estado inicial, salvo para el estado absorbente.

Para esquemas de cifrado de flujo es aconsejable utilizar la secuencia que determina el periodo máximo. Además, las secuencias generadas por un LFSR con un polinomio de conexión primitivo cumplen los tres postulados de Golomb, gracias a la longitud máxima de su periodo y a que tienen que pasar por todos los estados posibles.

2.3. Limitaciones de los generadores lineales

Hemos puesto de relieve que los LFSR se comportan muy bien en términos de facilidad de análisis, de implementación y de velocidad. Si, además, el polinomio de conexiones es primitivo, cumplen todos los postulados de Golomb. Ahora bien, para que el periodo $2^n - 1$ sea largo, es precisa una longitud del LFSR también larga. Esta particularidad puede representar un problema, ya que el coste de hallar polinomios primitivos de grado alto es bastante elevado.

Recordad

El número de polinomios primitivos de grado n lo da La expresión $\phi(2^n - 1)/n$, donde ϕ es la función phi de Euler.

Sin embargo, a pesar de sus ventajas, los generadores lineales no se emplean para implementar sistemas de cifrado de flujo porque suelen ser fáciles de resolver. En efecto, supongamos que conocemos $2n$ bits consecutivos de una secuencia de cifrado, $s_{k+1}, s_{k+2}, \ldots, s_{k+2n}$. A partir de la expresión 2.2 y del sistema de ecuaciones siguiente podemos determinar los coeficientes del polinomio de realimentación, c_i, y obtener toda la secuencia:

Sistemas lineales

Se trata de un sistema de n ecuaciones con n incógnitas, c_i, para $1 \leq i \leq n$, con el cual podemos determinar todos los coeficientes.

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

Así, pues, a la hora de utilizar un generador para un proceso de cifrado de flujo nos fijaremos también en su predictibilidad, es decir, en lo que se denomina *complejidad lineal*.

Dado que cualquier secuencia periódica se puede generar a partir de un LFSR no singular, J.L. Massey definió la **complejidad lineal de una secuencia** como el número de celdas del LFSR más corto que es capaz de generarla.

Una secuencia generada por un LFSR de longitud n tiene, obviamente, complejidad lineal n, una complejidad muy baja comparada con el periodo $2^n - 1$. El mismo Massey propuso un algoritmo que determina el LFSR mínimo que genera una secuencia y su correspondiente estado inicial.

El modo de reducir la predictibilidad de la secuencia de cifrado consiste, pues, en aumentar la complejidad lineal de la secuencia de cifrado, que convendría que fuera de longitud próxima a la del periodo. Más adelante veremos cómo lograrlo a partir de operaciones no lineales.

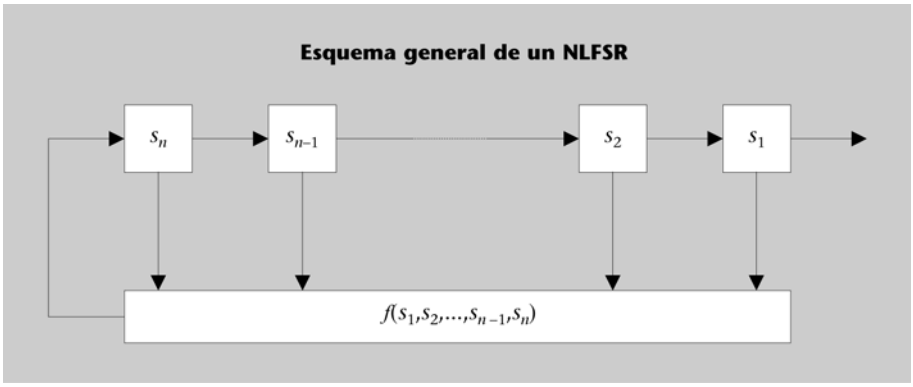
3. Generadores no lineales

En este apartado analizaremos algunos generadores no lineales que permiten aumentar la complejidad lineal de las secuencias en flujo.

3.1. Registros de desplazamiento realimentados no linealmente

Los registros de desplazamiento de realimentación no lineal (NLFSR) tienen un esquema muy parecido a los LFSR, con la variante de que la función que los realimenta no es lineal, tal como muestra la figura siguiente:

NLFSR es la sigla del término inglés *Non-Linear Feedback Shift Register*.



El hecho de que la función de realimentación no sea lineal presenta el problema de que se hace muy complicado estudiarla y especificarla. En particular, es difícil encontrar NLFSR con periodos largos*. Por esta razón, los NLFSR no se estudian exhaustivamente, aunque se intenta obtener propiedades a partir de algunas funciones de realimentación.

*** Un ejemplo lo tenemos en las secuencias de De Bruijin, generadas con NLFSR y de periodo máximo.**

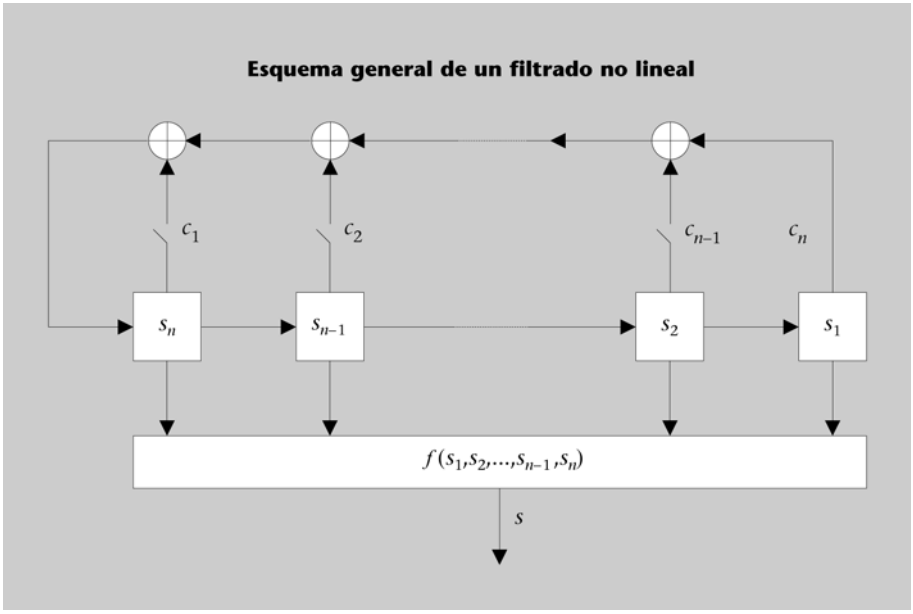
3.2. Filtrado no lineal

Una forma alternativa de generar secuencias es como salida de una función no lineal que tenga por entrada el estado de un LFSR. Una representación esquemática de la función de cifrado que utiliza un filtrado no lineal es la que podemos ver en la figura siguiente.

Este tipo de generadores son difíciles de analizar y también resulta difícil de calcular la complejidad lineal de la secuencia que generan. De hecho, no se intenta calcular su complejidad lineal, sino acotarla.

Como se ve en la figura siguiente, en general, una parte de los filtros no lineales suele ser un LFSR, cuyo estudio resulta sencillo y cuyas caracterís-

ticas son óptimas. Como sabemos, conviene que el polinomio de conexiones de este LFSR sea primitivo para que el periodo de la secuencia resultante sea máximo.



La parte más complicada de estos generadores consiste en elegir la función de filtrado f, pues ha de permitir obtener una complejidad elevada pero, a la vez, debe ser fácil de calcular.

3.3. Combinadores no lineales

Los combinadores no lineales, como su nombre indica, basan su funcionamiento en la combinación de varios LFSR.

Según cómo los utilicemos y combinemos, podemos distinguir tres tipos básicos:

1) Generadores basados en combinaciones no lineales de LFSR, que utilizan las salidas de diferentes LFSR como entrada de una función no lineal que generará la secuencia final.

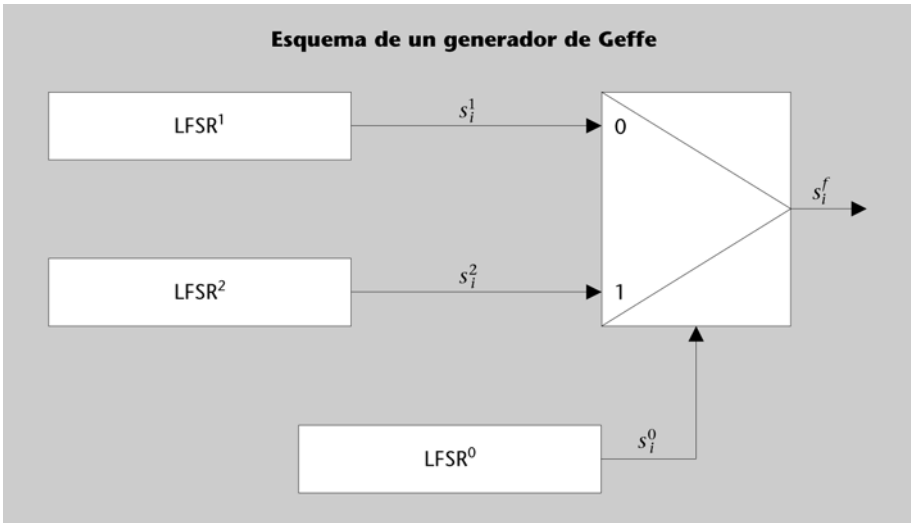
2) Generadores basados en un control paso a paso, que utilizan diferentes LFSR, algunos de los cuales regulan los impulsos de reloj de otros.

3) Generadores de secuencia multi-reloj, formados por diferentes LFSR regidos por ciclos de reloj de velocidades diferentes.

A continuación describimos el funcionamiento de diferentes combinadores no lineales para entender claramente la diferencia entre estos tres grupos.

3.3.1. Generador de Geffe

El generador de Geffe es un generador pseudoaleatorio formado por tres LFSR relacionados, tal como muestra la figura siguiente:



Notación

En este apartado los superíndice indican el LFSR al que hace referencia el valor.

Observamos que dos LFSR generan dos secuencias y un tercero determina la función de salida. De este modo, en cada impulso de reloj $t = i$ se obtienen dos valores de LFSR^1 y LFSR^2 y otro de LFSR^0, que es el que determina cuál de las dos salidas se toma. Si la salida del LFSR^0, s_0, es un 0, se toma la salida del LFSR^1, $s_i^f = s_i^1$, mientras que si $s_i^0 = 1$, se toma la salida del LFSR^2, $s_i^f = s_i^2$.

El valor de salida del generador para el impulso de reloj $t = i$ se puede expresar analíticamente de la manera siguiente:

$$s_i^f = (1 - s_i^0)\, s_i^1 \oplus s_i^0\, s_i^2$$

donde s_i^0, s_i^1 y s_i^2 son, respectivamente, los valores que los LFSR 0, 1 y 2 generan para el impulso de reloj $t = i$.

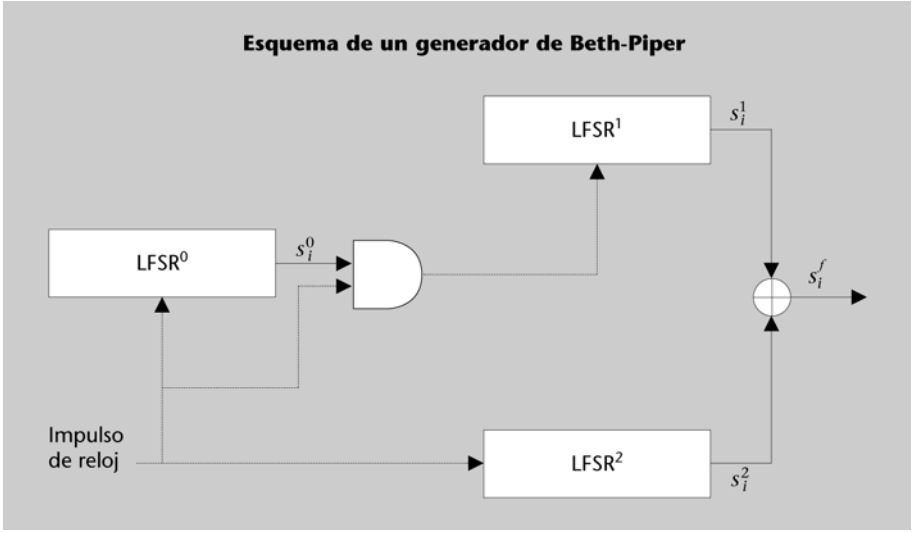
La complejidad lineal de este tipo de generadores es mayor con respecto a un generador formado por un simple LFSR. En concreto, sean n_1, n_2, n_0 los grados de los polinomios de conexiones de los tres LFSR; su complejidad lineal es $CL = n_1 + n_1 n_0 + n_2 n_0$. Por otra parte, el periodo resultante es $p = \mathrm{mcm}(2^{n_0} - 1,\ 2^{n_1} - 1,\ 2^{n_2} - 1)$.

A pesar de la elevada complejidad lineal que alcanza este generador, tampoco es útil para aplicaciones de cifrado de flujo porque presenta una gran probabilidad de coincidencia entre la secuencia final, s^f, y las secuencias de salida intermedias de los LFSR, s^1 y s^2:

- $P(s^f = s^1) = 3/4$,
- $P(s^f = s^2) = 3/4$.

3.3.2. Generador de Beth-Piper

Este generador está dentro del grupo de combinadores de control paso a paso y su funcionamiento se refleja en la figura siguiente:



El esquema muestra cómo el LFSR^0 controla el ritmo del reloj del LFSR^1. Así, LFSR^1 sólo cambia el estado en el instante t si $s_{t-1}^0 = 1$; es decir, si en el instante $t - 1$ LFSR^0 da como salida un 1.

La secuencia de salida final tiene una complejidad lineal $CL = (2^{n_0} - 1)\, n_1 + n_2$ y el periodo periodo $p = (2^{n_0} - 1)\,(2^{n_1} - 1)\,(2^{n_2} - 1)$, donde, como antes, n_0, n_1 y n_2 son los grados de los polinomios de conexiones de LFSR^0, LFSR^1 y LFSR^2, respectivamente.

Igual que en el generador de Geffe, a pesar de su elevada complejidad, el generador de Beth-Piper no ofrece una seguridad óptima, ya que la probabilidad de que la secuencia final, s^f, y la secuencia generada por el LFSR^2 coincidan es muy elevada:

$$P(s_t^f \oplus s_{t+1}^f = s_t^2 \oplus s_{t+1}^2) = \frac{3}{4}$$

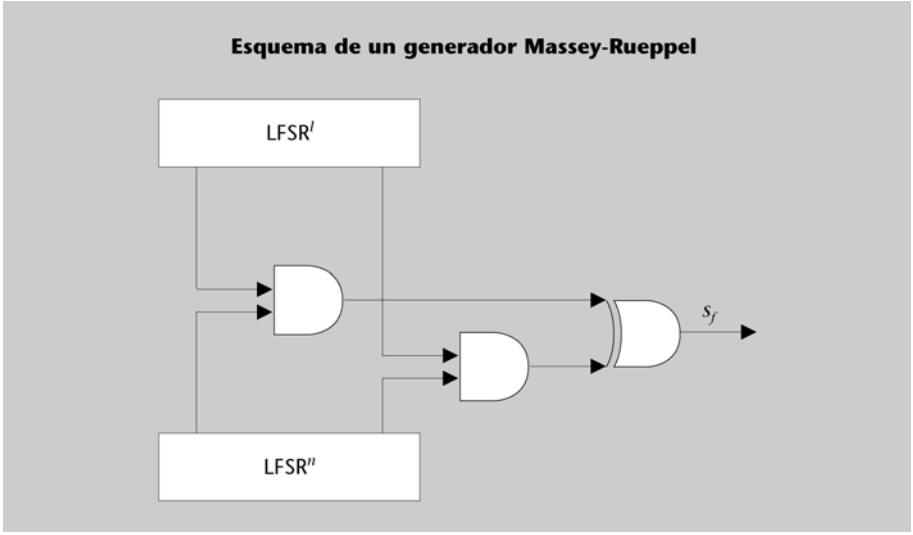
Una versión más segura...

... del generador de Beth-Piper es el generador en cascada de Gollmann, que también se basa en un combinador de control paso a paso.

3.3.3. Generador multivelocidad de Massey-Rueppel

En la figura siguiente se puede ver el funcionamiento del generador multivelocidad de Massey-Rueppel. Este generador utiliza dos LFSR que funcionan a velocidades diferentes. El LFSR^n, que tiene n celdas, va d veces ($d \geq 2$) más deprisa que el LFSR^l, de l celdas (donde $n \geq l$). La expresión algebraica de la secuencia de salida es la siguiente:

$$s_t^f = \sum_{i=0}^{l-1} s_{t+i}^l \cdot s_{dt+i}^n$$



El factor de velocidad d es variable y se utiliza como una parte de la clave de cifrado.

Con respecto a la complejidad lineal de las secuencias generadas, vale $CL = nl$ siempre que los dos generadores lineales tengan polinomios de conexiones primitivos, $\text{mcd}(l, n) = 1$ y $\text{mcd}(d, 2^n - 1) = 1$. El periodo de la secuencia generada vale: $p = (2^n - 1)(2^l - 1)$.

Resumen

En este módulo hemos descrito el funcionamiento del cifrado en flujo y hemos estudiado las propiedades que ha de tener una secuencia aleatoria para que se pueda utilizar como secuencia de cifrado.

Hemos presentado diferentes tipos de generadores para obtener secuencias pseudoaleatorias. Hemos señalado que los registros de desplazamiento de realimentación lineal (LFSR) son los más interesantes porque resultan fáciles de estudiar, aunque no aconsejamos su aplicación en criptografía por la sencillez de su criptoanálisis.

Finalmente, hemos estudiado diferentes tipos de generadores que utilizan como base los registros de desplazamiento de realimentación lineal, por ejemplo los filtros y los combinadores, aunque no todos cumplen los requisitos necesarios para generar secuencias seguras para la criptografía.

Actividades

1. Generad una secuencia pseudoaleatoria de 64 bits con el generador de números aleatorios de vuestra calculadora. Para generar un bit, pulsad la tecla RAN# y redondead el resultado para obtener un 0 o un 1. A partir de la secuencia obtenida, efectuad las operaciones siguientes:
   a) Calculad su periodo.
   b) Comprobad si se cumplen los tres postulados de Golomb.
   c) Evaluad su imprevisibilidad.

2. Simulad el generador de Geffe con los LFSR de cuatro celdas con polinomios de conexiones de los LFSR primitivos y comprobad que se cumple la relación:

$$P(s^f = s^1) = P(s^f = s^2) = \frac{3}{4}.$$

Ejercicios de autoevaluación

1. Dada la secuencia de cifrado $s = 000111101011001000111$ con un periodo $p = 15$, verificad si cumple los tres postulados de Golomb.

2. A partir de un LFSR de siete celdas con un polinomio de conexiones primitivo, calculad:
   a) el periodo de la secuencia resultante;
   b) su complejidad lineal.

3. Hallad el polinomio de conexiones de un LFSR de seis celdas sabiendo que los primeros doce dígitos de la secuencia de salida son 110011011101.

4. Indicad los parámetros deben tener:
   a) un generador de Beth-Piper, para generar una secuencia de periodo 32.426.527.
   b) un generador de Massey-Rueppel, para generar una secuencia de periodo 1.046.017.

Solucionario

Ejercicios de autoevaluación

1. Dado que el periodo vale 15, sólo tomaremos los primeros 15 bits: 000111101011001.
Comprobamos que la secuencia s cumple los tres postulados de Golomb:
- El primer postulado se cumple, ya que en el periodo p hay 7 ceros y 8 unos.
- Hay cuatro ráfagas de longitud 1, dos ráfagas de longitud 2, una ráfaga de longitud 3 y una ráfaga de longitud 4, por lo cual se cumple el segundo postulado.
- La condición necesaria para que se cumpla el tercer postulado se da; la función de autocorrelación de la secuencia para todos los valores de k no múltiples de 15 vale:

$$AC(k) = \frac{7-8}{15} = \frac{-1}{15}.$$

Verificamos para algunas de las secuencias desplazadas k bits, s_k, el número de bits coincidentes y no coincidentes con la secuencia inicial s = 000111101011001:

- $k = 1$; s_1 = 001111010110010. Para hallar el número de bits que coinciden con la secuencia inicial calculamos: $s \oplus s_1$ = 001000111101011.
- $k = 2$; s_2 = 011110101100100, $s \oplus s_2$ = 011001000111101.
- $k = 4$; s_4 = 111010110010001, $s \oplus s_4$ = 111101011001000.
- $k = 7$; s_7 = 010110010001111, $s \oplus s_7$ = 010001111010110.

2.
a) Teniendo en cuenta que el polinomio de conexiones del LFSR es primitivo, el valor del periodo es máximo y vale $2^7 - 1$.
b) La complejidad lineal es el número de celdas del LFSR más corto capaz de generarlo: $CL = 7$.

3. Para obtener el polinomio de conexiones que corresponde al LFSR a partir de los $2n$ datos de que disponemos, planteamos el sistema de ecuaciones correspondiente:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} c_6 \\ c_5 \\ c_4 \\ c_3 \\ c_2 \\ c_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}.$$

Una vez resuelto el sistema, vemos que el polinomio de conexiones vale:

$$f(x) = 1 + x + x^6.$$

4.
a) En el caso de Beth-Piper, el periodo quedaba determinado por la expresión:

$$p = (2^{n_0} - 1)(2^{n_1} - 1)(2^{n_2} - 1),$$

y así tenemos:

$$32.426.527 = (2^{n_0} - 1)(2^{n_1} - 1)(2^{n_2} - 1).$$

Si calculamos qué periodos dan los LFSR con un polinomio primitivo de grado pequeño tenemos:

n	$2^n - 1$
3	7
4	15
5	31
6	63
7	127
8	255
9	511
10	1.023
11	2.047

Si descomponemos 32.426.527 en factores primos tenemos que 32.426.527 = 7 · 23 · 31 · 73 · · 89. Observad que los podemos agrupar de la manera siguiente: 31 · (7 · 73) · (23 · 89) = 31 · · 511 · 2.047. Por lo tanto, si tomamos un generador de Beth-Piper con polinomios de grado 5, 9 y 11 obtendremos el periodo que nos piden.

b)En el caso de un generador de Massey-Rueppel ocurre lo mismo, pero en este caso la expresión es:

$$p = (2^n - 1)(2^l - 1),$$

y sustituyendo los datos conocidos obtenemos:

$$1.046.017 = (2^n - 1)(2^l - 1).$$

Descomponiendo p y agrupando convenientemente los factores, hallamos:

$$1.046.017 = 7 \cdot 23 \cdot 73 \cdot 89 = (7 \cdot 73) \cdot (23 \cdot 89) = 511 \cdot 2.047.$$

Luego: $n = 11$ y $l = 9$.

Recordemos, sin embargo, que la expresión anterior sólo es válida si mcd(l, n) = 1 y mcd(d, $2^n - 1$) = 1. En este caso se cumple que mcd(9, 11) = 1, y sólo hay que tomar $d \geq 2$ tal que mcd(d, 2.047) = 1. Así, por ejemplo, para $n = 11$, $l = 9$ y $d = 15$ obtenemos el generador que se nos pedía.

Glosario

complejidad lineal de una secuencia *f* Número de celdas del LFSR más corto que es capaz de generar una secuencia.

criptosistema de clave compartida *m* Criptosistema en que tanto el emisor como el receptor comparten una sola clave que utilizan tanto para cifrar como para descifrar.

criptosistema en flujo *m* Sistema de cifrado que utiliza un generador pseudoaleatorio para cifrar un mensaje, que suma bit a bit el texto en claro con la secuencia pseudoaleatoria que resulta del generador.

estado de un LFSR *m* Conjunto de valores contenidos en cada celda de un LFSR en un instante de tiempo.

función de autocorrelación de una secuencia periódica *f* Número de coincidencias menos número de no coincidencias entre la sucesión original y la misma sucesión desplazada k posiciones dividido por el periodo de la secuencia original.

generador lineal *m* Generador de secuencias de bits que sólo ejecuta operaciones lineales sobre los elementos de entrada para obtener la secuencia de salida.

generador no lineal *m* Generador de secuencias de bits que ejecuta operaciones no lineales, como permutaciones, sobre los elementos de entrada para obtener la secuencia de salida; además, puede utilizar también operaciones lineales.

generador pseudoaleatorio *m* Proceso determinista capaz de generar una secuencia pseudoaleatoria.

LFSR Registro de desplazamiento de realimentación lineal.

NLFSR Registro de desplazamiento de realimentación no lineal.

periodo *m* Entero más pequeño p tal que $s_{i+p} = s_i$ para todo $i \geq 0$, donde $\{s_i\}_{i \geq 0}$ es una secuencia periódica.

polinomio de conexiones de un LFSR *m* Polinomio que determina o que es determinado por la función lineal de realimentación del LFSR.

ráfaga *f* Conjunto de bits consecutivos iguales dentro de una secuencia.

registro de desplazamiento realimentado linealmente *m* Dispositivo físico o lógico formado por n celdas de memoria y una función de realimentación lineal.

secuencia de cifrado *f* Secuencia que resulta del generador pseudoaleatorio en un criptosistema de flujo.

secuencia pseudoaleatoria *f* Secuencia que cumple los tres postulados de Golomb.

Bibliografía

Bibliografía básica

Beker, H.; Piper, F. (1982). *Cipher systems the protection of the communications*. Londres: Northwood Books.

Beth, T.; Piper, F. (1984). "The stop-and-go generator". *Advances in Cryptology - Eurocrypt'84* (pág. 88-92). Berlín: Springer.

Fuster, A.; de la Guía, D.; Hernández, L.; Montoya, F.; Muñoz, J. (1997). *Técnicas criptográficas de protección de datos*. Madrid: Ra-ma.

Geffe, P. R. (1973). "How to protect data with ciphers that are really hard to break". *Electronics* (pág. 99-101).

Golomb, S. W. (1967). *Shift register sequences*. San Francisco: Holden-Day.

Massey, J. L. (1969). "Shift-register synthesis and BCH decoding". *IEEE transactions on information theory* (IT-15, pág.122-127).

Rueppel, R. A. (1986). *Analysis and design of streamciphers*. Berlín: Springer.

Bibliografía complementaria

Coveyou, R. R.; Mcpherson, R. D. (1967). "Fourier analysis of uniform random number generators". *J. Assoc. Comput. Mach* (núm. 14, pág. 100-119).

Fredricksen, H. (1982). "A survey of full length nonlinear shift register cycle algorithms". *SIAM journal on applied mathematics* (núm. 24(2), pág. 195-221).

Gollmann, G., Chambers, W. G. (1989). "Clock-controlled shift registers: a review". *IEEE journal on selected areas in communications* (núm. 7(4), pág. 525-533).

Knuth, D. (1981). *The art of computer programming*. Reading: Addison-Wesley.

Maurer, U. M. (1991). "En universal statistical test for random bit generators". *Advances in cryptology - Crypto '90* (pág. 409-420). Nueva York: Springer.

Ronce, C. A. (1984). "Feedback Shift Registers". *LNCS 169*. Berlín: Springer.