Criptografía

ETSInf - UPV

Práctica 4

Generación y prueba de secuencias binarias pseudoaleatorias

---

1. Introducción

El objeto de esta práctica es generar secuencias pseudoaleatorias mediante simulación de sistemas de registros lineales retroalimentados (LFSR por sus siglas en inglés). Estos sistemas se utilizan fundamentalmente en sistemas de transmisión de información, bien para el cifrado o la modulación de las secuencias a transmitir. Por ejemplo, se utilizan para la el cifrado de la señal en comunicaciones entre dispositivos (E0 en Bluetooth) o en telefonía (A5 en GSM), y para el modulado de la señal en determinados estandares de difusión de video (NICAM,DVB-T) o de posicionamiento global (GPS),

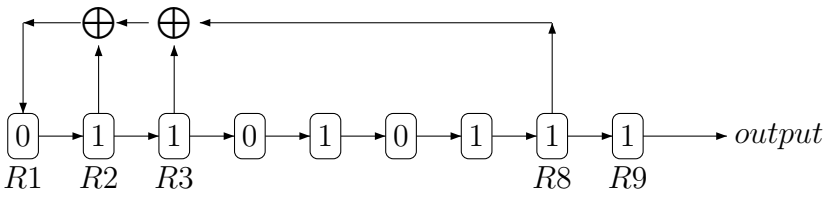
En esta práctica analizaremos las secuencias generadas, viendo en que forma se ajustan a las características deseables de las secuencias pseudoaleatorias fijadas por los postulados de Golomb.

2. LFSR

En *Mathematica* podemos representar un LFSR mediante dos listas de modo que se represente el estado del sistema (valor de cada uno de los registros) como el conjunto de registros que se consideran en el retroalimentado del sistema. Por ejemplo, la lista:

```
lfsr={{0,1,1,0,1,0,1,1,1},{2,3,8}}
```

representaría el LFSR de la siguiente figura.



Ejercicio 1:

Implementar un módulo *Mathematica* que, dado un determinado LFSR recibido como parámetro, devuelva el estado en el que se encuentra el LFSR después de un paso de generación, así como la salida obtenida.

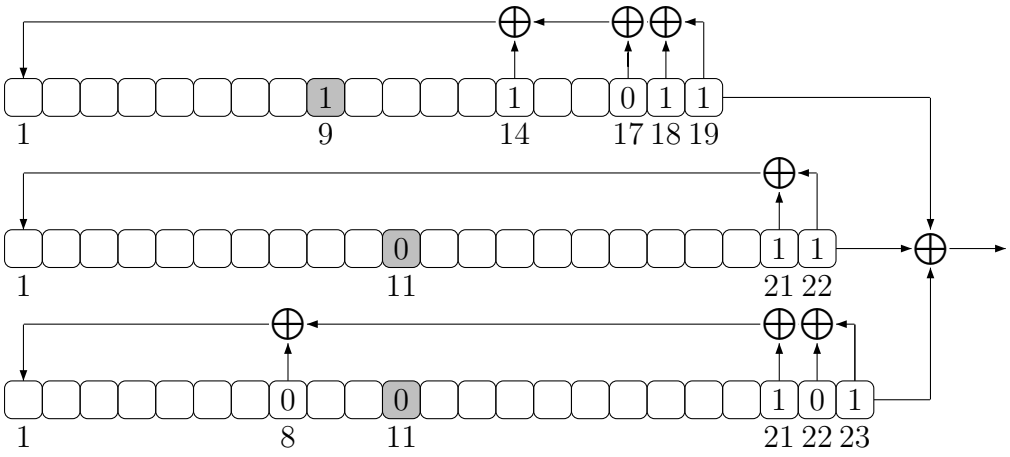


Figura 1: Esquema de cifrado en flujo A5/1. En gris se indican los registros que se tienen en cuenta para desplazar los respectivos LFSR.

3. Cifrado A5/1

El algoritmo A5/1 es un algoritmo de cifrado en flujo utilizado en teléfonía móvil (GSM). Su construcción considera tres LFSR de 19, 22 y 23 estados. Su arquitectura esquemática se muestra en la Figura 1.

Como puede verse en la figura, el primer LFSR calcula el bit de realimentación utilizando los registros 14, 17, 18 y 19 (*tapped bits*), el segundo LFSR utiliza los registros 21 y 22, y el tercero los registros 8, 21, 22, y 23. El sistema no desplaza todos los LFSR en cada paso de generación, considerando un registro de cada uno de los LFSR (los registros 9, 11 y 11 respectivamente) para determinar, mediante un criterio de mayoría, los LFSR que se desplazan. En el diagrama puede verse que, considerando el contenido de los bits a considerar (*clocking bits*), el bit mayoritario es el 0, por lo tanto, únicamente se desplazan los LFSR 2 y 3, (realimentandose en el ejemplo de la Figura 1 ambos con un bit 0). En este ejemplo, el bit que genera el sistema en el instante representado es 1.

El cifrado A5/1 comienza inicializando el sistema, para posteriormente generar secuencias pseudoaleatorias de 114 bits. En esta práctica nos centraremos en simular el comportamiento del sistema partiendo de una configuración inicial aleatoria. La representación *Mathematica* de este sistema

utillizará una lista que contenga el estado de cada uno de los LFSR, así como un lista que contenga que registros utilizar como clocking bits. Por ejemplo:

```
A5={
 (* primer LFSR *)
 {{0,1,1,0,1,0,1,0,1,0,1,1,0,1,0,1,0,1,1},{14,17,18,19}},
 (* segundo LFSR *)
 {{0,1,1,0,1,0,1,1,1,0,0,1,0,1,0,1,1,0,1,0,1,0},{21,22}},
 (* tercer LFSR *)
 {{0,0,0,0,1,0,1,0,1,0,1,0,1,1,0,1,0,1,1,0,0,1,1},{8,21,22,23}},
 (* clocking *)
 {9,11,11}
}
```

Ejercicio 2:

Implementar un módulo *Mathematica* que genere un sistema A5 de cifrado en flujo con una configuración inicial aleatoria.

Ejercicio 3:

Implementar un módulo *Mathematica* que simule un paso de generación de un sistema A5 de cifrado en flujo. El módulo deberá devolver el valor de salida y la configuración del sistema después del paso de generación.

4. Pruebas estadísticas

Consideramos en esta sección una secuencia binaria $s = s_1 s_2 \dots s_{n-1}$ de longitud n. A continuación se describen distintos test que, de modo probabilístico, establecen si una secuencia binaria posee las propiedades que se esperan de una secuencia completamente aleatoria.

Enfatizamos el hecho de que el cumplimiento de estos test no garantiza que la secuencia se haya generado mediante un generador de bits aleatorio. Únicamente proporcionan valores que se ajustan a determinada distribución de probabilidad, donde los valores obtenidos pueden compararse con los esperados para el tamaño de secuencia analizado.

4.1. Test de frecuencia (monobits test)

Este test sigue la filosofía del primer postulado de Golomb buscando cuantificar que el número de bits de cada tipo es aproximadamente el mismo. El estadístico utilizado es:

$$X_1 = \frac{(n_0 - n_1)^2}{n}$$

donde n_0 y n_1 indican respectivamente el número de bits 0 y de bits 1.

4.2. Test serie (two-bit test)

Este test pretende determinar si el número de ocurrencias de cada segmento de longitud 2 es aproximadamente el mismo. Denotamos con n_0 y n_1 el número de bits 0 y 1 y con n_{00}, n_{01}, n_{10} y n_{11} el número de ocurrencias de cada uno de los segmentos de longitud 2 posibles.

Es necesario tener en cuenta que los segmentos pueden solaparse, y que por lo tanto $n_{00} + n_{01} + n_{10} + n_{11} = n - 1$. El estadístico se obtine como sigue:

$$X_2 = \frac{4}{n-1}(n_{00}^2 + n_{01}^2 + n_{10}^2 + n_{11}^2) - \frac{2}{n}(n_0^2 + n_1^2) + 1$$

4.3. Poker test

Este test es una generalización del que valora la frecuencia de aparición de bits de cada tipo.

Sea m un valor tal que $\lfloor \frac{n}{m} \rfloor \geq 5(2^m)$, y sea $k = \lfloor \frac{n}{m} \rfloor$ (recordamos que n denota la longitud de la secuencia).

El test divide la secuencia s en k segmentos de longitud m que no solapen y denota con n_i el número de ocurrencias de secuencias de bits de tipo i en los segmentos (de todas y cada una de las secuencias posibles). El estadístico se obtiene como:

$$X_3 = \frac{2^m}{k}\left(\sum_{i=1}^{2^m} n_i^2\right) - k$$

4.4. Runs test

Este test tiene como objeto determinar si el número de rachas (de bits de cualquier valor) es el esperado para una secuencia aleatoria.

Considerando una secuencia aleatoria, el número esperado de rachas de longitud i puede calcularse como:

$$e_i = \frac{(n-i+3)}{2^{i+2}}$$

Tomando k como el máximo valor de i tal que $e_i \geq 5$, el test considera las rachas de bits 0 y 1 de longitud i, identificados por B_i y G_i para $1 \leq i \leq k$. Estos valores se utilizan en el cómputo del estadístico:

$$X_4 = \sum_{i=1}^{k} \frac{(B_i - e_i)^2}{e_i} + \sum_{i=1}^{k} \frac{(G_i - e_i)^2}{e_i}$$

Ejemplo 1 *Consideremos la secuencia binaria:*

```
11100 01100 01000 10100 11101 11100 10010 01001 11100
01100 01000 10100 11101 11100 10010 01001 11100 01100
01000 10100 11101 11100 10010 01001 11100 01100 01000
10100 11101 11100 10010 01001
```

Teniendo en cuenta que:

- $n_0 = 84$ *y* $n_1 = 76$
- $n_{00} = 44$, $n_{01} = 40$, $n_{10} = 40$ *y* $n_{11} = 35$

El test de frecuencia da como resultado $X_1 = 0{,}4$ *y el test serie* $X_2 = 0{,}6252$.

Teniendo en cuenta que $n = 160$, *consideramos valores sucesivos de* m *en la formula* $\lfloor \frac{n}{m} \rfloor \geq 5(2^m)$ *y obtenemos:*

m	$\lfloor \frac{n}{m} \rfloor \geq 5(2^m)$
2	$\lfloor \frac{160}{2} \rfloor = 80 \geq 5(2^2) = 20$
3	$\lfloor \frac{160}{3} \rfloor = 53 \geq 5(2^3) = 40$
4	$\lfloor \frac{160}{4} \rfloor = 40 \geq 5(2^4) = 80$

Tomando el mayor valor de m *que cumple la formula, consideramos las secuencias (no solapantes) de longitud* 3 *y el número de ocurrencias de cada una:*

$secuencia_i$	n_i
000	*5*
001	*10*
010	*6*
011	*4*
100	*12*
101	*3*
110	*6*
111	*7*

con lo que el poker test obtiene $X_3 = 9{,}6415$.

Respecto el runs test, los distintos valores de e_i *que se obtienen son* $e_1 = 20{,}25$, $e_2 = 10{,}0625$ *y* $e_3 = 5$ *por lo que* $k = 3$. *Considerando el número de rachas de* 0 *(gaps) y* 1 *(blocks) de longitud* i *tenemos:*

i	B_i	G_i
1	*25*	*8*
2	*4*	*20*
3	*5*	*12*

con lo que $X_4 = 31{,}7913$.

Por último, considerando $d = 3$, *el valor de* $A(d)$ *para la secuencia es* 35 *y el desl test de autocorrelación* $X_5 = -6{,}9434$.

4.5. FIPS 140 -1

En lugar de dejar los parámetros que establecen garantía probabilística a elección del usuario, el estandar publicado por la NIST estabece cotas específicas que deben cumplir las secuencias sometidas a distintos test probabilisticos.

Para valorar un generador pseudoaleatorio de acuerdo con este estándar, se evalúa una única secuencia generada de 20000 bits. Si uno sólo de los test falla, el generador no pasa el test.

i) *monobit test:* n_1 tiene que ser tal que $9654 < n_1 < 10346$

ii) *poker test:* Se calcula X_3 para $m = 4$. El test se cumple si $1{,}03 < X_3 < 57{,}4$.

iii) *runs test:* En este test se consideran todas las rachas, incluyendo cualquier racha de longitud mayor a 6 como de longitud 6. Cada uno de los 12 valores obtenidos (número de rachas de bits 0 o 1 de longitudes de 1 a 6) han de estar dentro del intervalo fijado en la siguiente tabla:

tamaño de racha	intervalo
1	2267 - 2733
2	1079 - 1421
3	502 -748
4	223 - 402
5	90 - 223
6	90 - 223

iv) *long run test:* El test se cumple si no existen rachas de longitud 34 o superior.

El estandar determina que los cuatro test deben realizarse cada vez que el generador se poner en funcionamiento. El estandar permite sustituir estos test en caso de se disponga de otros estadísticos más confiables.

Ejercicio 4:

> Implementar módulos *Mathematica* que, dada una secuencia pseudoaleatoria de bits representada mediante una lista, permitan analizar la secuencia de acuerdo a los test que conforman el estándar descrito.