Bol. Soc. Esp. Mat. Apl. nº 21 (2002), 65–87

Aplicaciones de los autómatas celulares a la generación de bits *

L. Hernández Encinas†, A. Martín del Rey‡
y G. Rodríguez Sánchez‡

† Dpt. Tratamiento de la Información y Codificación,
Instituto de Física Aplicada, C.S.I.C.
‡ Dpt. Matemática Aplicada, E.T.S.I.I.,
Universidad de Salamanca

luis@iec.csic.es, delrey@usal.es, gerardo@usal.es

Resumen

Se presentan en este artículo la principales propiedades de los autómatas celulares lineales, y en particular de los llamados de Wolfram. Se definen, además, algunas nociones de criptografía con el fin de señalar las principales aplicaciones de los autómatas celulares a la misma, como la generación de números pseudoaleatorios para los cifrados en flujo o la generación de claves.

Palabras clave: *Autómatas celulares, criptografía, números pseudoaleatorios, generadores de bits.*

Clasificación por materias AMS: *68Q80, 94A60, 11K45, 65C10, 68W20.*

1 Introducción

Uno de los objetivos de las Matemáticas ha sido el de proporcionar herramientas que expliquen algunos de los fenómenos naturales que nos rodean. En general, este proceso se lleva a cabo buscando modelos matemáticos que den respuesta a dichos fenómenos. Así, se puede mencionar el gran desarrollo que ha tenido desde hace unos años el estudio del caos ([6, 10]) y de los sistemas dinámicos ([9, 21]). En el estudio de estos últimos cabe destacar el de los llamados sistemas dinámicos discretos, en particular de los autómatas celulares.

*Los autores agradecen al Dr. J. Muñoz Masqué, del Instituto de Física Aplicada del C.S.I.C., sus sugerencias en la redacción final de este artículo. Este trabajo ha sido parcialmente subvencionado por la Fundación "Samuel Solórzano Barruso". El trabajo de A. M. R. se ha desarrollado durante una estancia en el Dpto. de Tratamiento de la Información y Codificación del C.S.I.C., cuya hospitalidad agradece.

Un *autómata celular* es un modelo formal compuesto por un conjunto de células que toman determinados valores. Estos valores van evolucionando con el paso discreto del tiempo según una determinada expresión matemática, que es sensible a los estados de las células vecinas (para un estudio más extenso, véase [22]).

Uno de los ejemplos más extendidos y populares de los autómatas celulares posiblemente sea el conocido como *juego de la vida de Conway* ([5]). El juego consiste en un conjunto de células dispuestas en el plano, cada una de las cuales puede adoptar dos estados: viva (representada por ■ o por un 1) o muerta (representada mediante □ o por un 0). Cada célula está rodeada por otras 8 vecinas, de modo que la distribución de estas 9 células es la de un cuadrado 3×3 (véase la Figura 1), situándose en el centro la considerada en primer lugar.

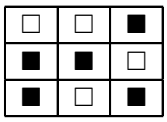

Figura 1. Ejemplo de una célula y su vecindad

A medida que discurre el tiempo, las células nacen y mueren según la siguiente regla: una célula muerta en un momento dado, cambia de estado en el instante siguiente si tiene exactamente 3 células vivas a su alrededor, en caso contrario permanece muerta en el siguiente instante. Por su parte, una célula viva en un instante dado, permanece viva en el instante siguiente si hay 2 ó 3 células vivas a su alrededor; en caso contrario pasa a estar muerta en el instante siguiente. En este caso particular, los autómatas celulares son bidimensionales, dado que la distribución y evolución de las células se lleva a cabo en el plano. Resulta sorprendente que con una regla de evolución tan sencilla como la anterior y partiendo de diferentes configuraciones iniciales se llegue, después de un número determinado de iteraciones, a resultados o configuraciones que puedan ser catalogados de forma tan precisa como la siguiente:

- *Configuraciones estables*: son aquellas disposiciones de células que permanecen inalterables con el paso del tiempo, una vez que se ha llegado a ellas. Algunas de estas diposiciones se pueden observar en la Figura 2.

Figura 2. Configuraciones estables

- *Configuraciones cíclicas*: son disposiciones, como algunas de las mostradas en la Figura 3, que se van obteniendo una detrás de otra cíclicamente.

Figura 3. Configuraciones cíclicas

- *Configuraciones móviles*: son las disposiciones de células que, sin modificar su estructura, se van desplazando en el plano a medida que pasa el tiempo (ver Figura 4).

Figura 4. Configuraciones móviles

A lo largo de este artículo se considerarán autómatas celulares unidimensionales, es decir, aquéllos cuyas células se pueden representar linealmente. Existen otros tipos de autómatas de dimensiones mayores (dependiendo de la disposición del conjunto de sus células), como el del juego de la vida de Conway (que, como ya se ha dicho, es de dimensión 2), pero que no serán tratados aquí. Para un estudio más detallado y extenso de estos autómatas puede consultarse [17].

Las aplicaciones de la teoría de autómatas celulares abarcan aspectos de la ciencia tan diversos como el comportamiento de las moléculas de un gas, el de las personas votando en un sufragio, el estudio de las bacterias eliminadoras de manchas petrolíferas, la evolución de la población, del tráfico, etc. (véanse, por ejemplo [13, 16]). No obstante, las aplicaciones de los autómatas celulares lineales son lo suficientemente interesantes como para no extendernos en los de complejidad mayor. En particular, en este artículo nos vamos a detener en algunas de las aplicaciones que los autómatas celulares tienen en criptología (otras aplicaciones pueden verse en [8, 26]).

Entre otras aplicaciones a la criptología cabe hacer mención a la propuesta en [14], donde se utilizan determinados autómatas celulares para sistemas de cifrado; sin embargo, ésta se demostró insegura en [1]. Por otra parte, el uso de autómatas híbridos o no uniformes (es decir, aquellos autómatas celulares en los que la evolución de los estados de las distintas células viene determinada por más de una expresión matemática) ha sido propuesto en [2, 23, 24, 25].

El resto del artículo se distribuye como sigue. En primer lugar se presentan algunas nociones elementales de criptología en la Sección 2, para pasar en la 3 a la definición de forma más rigurosa de los autómatas celulares unidimensionales. Se presentarán también algunas de sus propiedades y se incluirán algunos ejemplos. En la Sección 4 se analizará su aplicación como generadores de números pseudoaleatorios para cifrados en flujo y en la Sección 5 se verá cómo pueden ser aplicados para generar claves a utilizar en diferentes criptosistemas, en particular a la generación de números primos. Finalmente, se expondrán las conclusiones de este trabajo en la Sección 6.

2 Nociones básicas de criptología

En la actualidad, la gran cantidad de información transmitida mediante redes de ordenadores (Internet, bases de datos remotas, email, etc.) y, en la mayoría de los casos, la necesidad de su confidencialidad (datos personales, cuentas bancarias, números de tarjetas de crédito, etc.) hace necesario que esta información se transmita de manera fiable y segura. Esta seguridad requiere del diseño e implementación de protocolos que garanticen el secreto de los datos enviados. De aquí el auge de la *criptología* (del griego *cripto* –oculto– y *logos* –tratado, ciencia–), cuyo estudio tiene dos ramas claramente diferenciadas: la *criptografía*, que se ocupa del cifrado seguro de la información a enviar, y el *criptoanálisis*, cuya tarea es la de analizar técnicas y métodos para obtener la información cifrada ([4, 12]).

El proceso para cifrar un mensaje consiste en transformarlo mediante un algoritmo de modo que sólo quien esté autorizado podrá invertir el proceso de cifrado (descifrado) para recuperar el texto original. En el algoritmo se utilizan determinados parámetros que se conocen como *claves*, mientras que el mensaje cifrado se denomina *criptograma* y todo este proceso se conoce como *criptosistema*. Si la clave es única y sólo es conocida por las dos personas que se intercambian el mensaje, el criptosistema se llama de *clave simétrica* (o secreta); en otro caso, es decir, si la clave que permite cifrar mensajes es conocida públicamente, mientras que la que descifra es mantenida en secreto, el criptosistema se denomina de *clave asimétrica* (o pública). Dentro de los criptosistemas de clave simétrica existen dos categorías: *cifrados en flujo* y *cifrados en bloque*. De entre los segundos cabe mencionar los criptosistemas DES, IDEA, Triple-DES y Rijndael (para mayor información, ver [4, 12, 19]).

Por su parte, los cifrados en flujo modifican un mensaje mediante una secuencia pseudoaleatoria de bits que se genera a partir de una clave secreta y un algoritmo determinístico. Una vez que el remitente ha expresado el mensaje que se desea transmitir mediante ceros y unos (utilizando, por ejemplo, la equivalencia en ASCII entre las letras y bytes de 8 bits), suma, bit a bit, dicho mensaje con la secuencia pseudoaleatoria, obteniendo el criptograma. Para descifrar el criptograma y obtener el texto claro, el destinatario genera la misma secuencia pseudoaleatoria que el remitente, utilizando el mismo algoritmo determinístico y la misma clave, y suma esta secuencia con el criptograma (la operación de suma bit a bit es una involución). Según este protocolo, para utilizar los cifrados en flujo se debe ser capaz de generar la misma secuencia de bits, tanto en origen como en destino. De ahí que dicha secuencia tenga que ser pseudoaleatoria y dependa de una clave secreta para que nadie, que no sea el remitente o el destinatario, pueda recuperar el mensaje que se está transmitiendo.

La seguridad de los criptosistemas de clave simétrica suele basarse en la dificultad de los ataques llamados de *fuerza bruta*, que consisten en probar todas y cada una de las posibles claves que se pueden emplear en el algoritmo a utilizar. Por su parte, la seguridad de los de clave pública se basa en la dificultad de resolver un problema matemático, aperentemente difícil computacionalmente

hablando. No obstante, no siempre es necesario calcular la clave para poder descifrar mensajes. En ocasiones, si se dispone de información adicional es posible llevar ataques diferentes de los de fuerza bruta o de resolver el problema matemático subyacente. Así, existen principalmente cuatro tipos de ataques a un criptosistema:

1. *Ataque al texto cifrado*: en este caso sólo se conoce un trozo del criptograma correspondiente a un texto claro.

2. *Ataque al texto claro conocido*: en este ataque se utiliza el conocimiento de un trozo del texto claro y su correspondiente criptograma.

3. *Ataque al texto claro elegido*: en este caso, el atacante elige un texto claro y consigue el criptograma correspondiente (no es imprescindible conocer la clave para este ataque, bastaría con que el atacante tuviera acceso temporal a la máquina donde se encuentra implementado el criptosistema).

4. *Ataque al texto cifrado elegido*: en éste, el atacante puede conseguir el texto claro a partir de un texto cifrado que elija (tampoco en este caso se tiene por qué conocer la clave, pues el atacante podría tener temporal acceso a la máquina que lleva a cabo el descifrado de los mensajes).

3 Autómatas celulares

Se denomina *autómata celular d-dimensional* a una colección finita o infinita de células idénticas dispuestas uniformemente según un espacio de d dimensiones y que poseen un estado determinado, que va cambiando con el paso discreto del tiempo según una determinada regla. Esta regla está influida por los estados de las células vecinas ([28]). Consecuentemente, los cuatro elementos que determinan un autómata celular son los siguientes:

1. Su *dimensión* d. Los autómatas celulares más utilizados suelen ser los unidimensionales o lineales (es decir $d = 1$ y las células se disponen según una linea recta), y los bidimensionales (en cuyo caso $d = 2$ y las células se distribuyen según un plano).

2. El *conjunto de estados* S. Normalmente se considera que el número de estados que puede adoptar una célula es finito, es decir $\#S = k$, por lo que el conjunto que se suele tomar es $S = \mathbb{Z}_k$.

3. La *vecindad* de cada célula del autómata. Es el conjunto de células dispuestas alrededor de una dada y cuyos estados influyen en el estado de la célula considerada en el instante siguiente.

4. La *regla de transición* f. Esta regla rige la evolución de los estados de las células teniendo en cuenta la vecindad de las mismas.

Los primeros autómatas celulares rigurosamente establecidos se debieron a von Neumann ([15]) y a Ulam ([27]). Si bien es verdad que durante cerca de treinta años los autómatas celulares han sido considerados como una especie de curiosidad matemática sin apenas aplicaciones, en la actualidad y gracias fundamentalmente a los trabajos de Wolfram ([30, 31]), se están convirtiendo en una de las herramientas imprescindibles en el estudio de múltiples fenómemos naturales.

3.1 Autómatas celulares lineales

Salvo que se diga lo contrario, consideraremos sólo autómatas celulares unidimensionales, es decir, aquellos para los que $d = 1$, de modo que sus células están dispuestas una a continuación de otra a modo de una cadena. Estos autómatas celulares serán denotados sencillamente por AC. Si el AC lineal consta de n células, cada una de ellas se nombrará por $\langle i \rangle$ con $0 \leq i \leq n-1$. Un ejemplo de un AC lineal con 5 células es el siguiente:

$\langle 0 \rangle$	$\langle 1 \rangle$	$\langle 2 \rangle$	$\langle 3 \rangle$	$\langle 4 \rangle$

Si, además, S_k es el conjunto de k estados y $a_i^{(t)} \in S_k$, $0 \leq i \leq n-1$, es el estado de la célula $\langle i \rangle$ en el instante t, entonces se denomina *configuración del* AC *en el instante* t y se denota por $C^{(t)}$ al siguiente vector:

$$C^{(t)} = \left(a_0^{(t)}, a_1^{(t)}, \ldots, a_{n-1}^{(t)}\right) \in S_k \times \overset{(n}{\ldots} \times S_k.$$

La evolución de un AC a lo largo del tiempo se representa de forma sencilla sin más que escribir las sucesivas configuraciones de sus células, una debajo de otra (*diagrama de evolución* del AC). A continuación se muestra un ejemplo del diagrama de evolución de un AC de 4 células.

$a_0^{(0)}$	$a_1^{(0)}$	$a_2^{(0)}$	$a_3^{(0)}$	$\rightsquigarrow$	$C^{(0)}$
$a_0^{(1)}$	$a_1^{(1)}$	$a_2^{(1)}$	$a_3^{(1)}$	$\rightsquigarrow$	$C^{(1)}$
$a_0^{(2)}$	$a_1^{(2)}$	$a_2^{(2)}$	$a_3^{(2)}$	$\rightsquigarrow$	$C^{(2)}$
		$\ldots$			$\ldots$

Denotaremos por V_i a la vecindad de la célula $\langle i \rangle$, es decir, al conjunto de células cuyo estado va a influir en el de $\langle i \rangle$ según la regla de transición que se considere. Las vecindades más comunes en los AC son de caracter *simétrico*, de modo que la célula $\langle i \rangle$ es la célula central. Estas vecindades pueden escribirse de la siguiente manera:

$$V_i(r) = \{\langle i-r \rangle, \ldots, \langle i-1 \rangle, \langle i \rangle, \langle i+1 \rangle, \ldots, \langle i+r \rangle\}, \tag{1}$$

donde r recibe el nombre de *radio de la vecindad*. Existen otros tipos de vecindades no simétricas como por ejemplo las definidas por

$$V_i = \{\langle i-1\rangle, \langle i\rangle, \langle i+1\rangle, \langle i+2\rangle\},$$

o vecindades arbitrarias como la siguiente

$$V_i = \{\langle i-3\rangle, \langle i\rangle, \langle i+1\rangle\}.$$

Dada la célula $\langle i\rangle$, con $i < r$ ó $i > n-r$, la determinación de la vecindad $V_i(r)$ queda restringida a determinadas condiciones de contorno del AC. Usualmente estas condiciones son de dos tipos:

- *Condiciones de contorno periódicas*: en este caso se supone que las n células del AC están dispuestas uniformemente según una circunferencia (ver Figura 5):

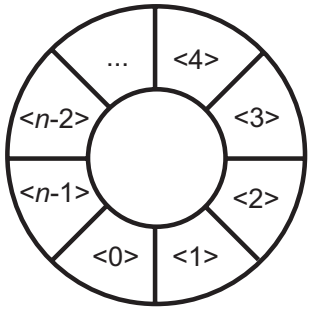

Figura 5. Disposición periódica

  de tal forma que a la izquierda de la célula $\langle 0\rangle$ están las células $\langle n-1\rangle$, $\langle n-2\rangle, \ldots$, mientras que a la derecha de la célula $\langle n-1\rangle$ estarían las células $\langle 0\rangle, \langle 1\rangle, \ldots$

- *Condiciones de contorno reflexivas*: en este otro caso se supone que a la izquierda de la célula $\langle 0\rangle$ se encuentran las células $\langle 1\rangle, \langle 2\rangle, \ldots$, mientras que a la derecha de la célula $\langle n-1\rangle$ se encuentran las células $\langle n-2\rangle, \langle n-3\rangle, \ldots$:

$$\langle n-1\rangle \quad \cdots \quad \langle 1\rangle \; \boxed{\langle 0\rangle \mid \langle 1\rangle \mid \cdots \mid \langle n-1\rangle} \; \langle n-2\rangle \quad \cdots \quad \langle 0\rangle$$

De aquí en adelante, si no se hace referencia expresa a lo contrario, se supondrá que las vecindades usadas son las simétricas y las condiciones de contorno son las periódicas.

Finalmente, la evolución de los estados de las distintas células del AC viene determinada por la denominada regla de transición $f : S_k^{2r+1} \to S_k$. Así, si

$V_i(r)$ es la vecindad definida en el AC según (1), entonces el estado de la célula $\langle i \rangle$ en el instante $t+1$ vendrá dado por una expresión matemática que depende de los estados de los elementos de $V_i(r)$ en el instante t:

$$a_i^{(t+1)} = f\left(a_{i-r}^{(t)}, \ldots, a_{i-1}^{(t)}, a_i^{(t)}, a_{i+1}^{(t)}, \ldots, a_{i+r}^{(t)}\right).$$

Si el AC tiene k estados y el radio de la vecindad es r, el número de posibles reglas que se pueden definir es $k^{k^{2r+1}}$.

Ejemplo. Consideremos un AC de $n = 11$ células, en el que $k = 2$ y $r = 1$ (obviamente suponemos condiciones de contorno periódicas, de tal forma que a efectos de notación, los subíndices se toman módulo n, es decir, en nuestro ejemplo, $a_{-1}^{(t)} = a_{10}^{(t)}$ y $a_{11}^{(t)} = a_0^{(t)}$). El conjunto de estados de este AC es $\mathbb{Z}_2$ y la vecindad de cada célula depende exclusivamente de las dos que la rodean. Supongamos, además, que la regla de transición que rige su evolución viene dada por la siguiente expresión:

$$a_i^{(t+1)} = \left(a_{i-1}^{(t)} + a_i^{(t)} + a_{i+1}^{(t)}\right) \bmod 2, \quad i \geq 0. \tag{2}$$

Si el estado inicial de este AC es el definido por la siguiente configuración

$$C^{(0)} = (0,0,0,0,0,1,0,0,0,0,0),$$

calculando 8 iteraciones sucesivas según la función de transición considerada, se obtiene la tabla de evolución siguiente:

$a_{10}^{(t)}$	$a_0^{(t)}$	$a_1^{(t)}$	$a_2^{(t)}$	$a_3^{(t)}$	$a_4^{(t)}$	$a_5^{(t)}$	$a_6^{(t)}$	$a_7^{(t)}$	$a_8^{(t)}$	$a_9^{(t)}$	$a_{10}^{(t)}$	$a_0^{(t)}$
0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	1	1	0	0	0	0	0
0	0	0	0	1	0	1	0	1	0	0	0	0
0	0	0	1	1	0	1	0	1	1	0	0	0
0	0	1	0	0	0	1	0	0	0	1	0	0
1	1	1	1	0	1	1	1	0	1	1	1	1
0	0	1	0	0	0	1	0	0	0	1	0	0
0	0	1	1	0	1	1	1	0	1	1	0	0

Si ahora se sustituye cada uno de los bits 1 por un cuadrado negro (■) y cada uno de los bits 0 por un cuadrado blanco (□), tal y como se hizo para el caso del AC del juego de la vida de Conway, se obtiene una tabla en la que se muestra el diagrama de evolución de este AC para las 8 iteraciones mostradas en la tabla anterior:

$a_{10}^{(t)}$	$a_0^{(t)}$	$a_1^{(t)}$	$a_2^{(t)}$	$a_3^{(t)}$	$a_4^{(t)}$	$a_5^{(t)}$	$a_6^{(t)}$	$a_7^{(t)}$	$a_8^{(t)}$	$a_9^{(t)}$	$a_{10}^{(t)}$	$a_0^{(t)}$
	□	□	□	□	□	■	□	□	□	□	□	
	□	□	□	□	■	■	■	□	□	□	□	
	□	□	□	■	□	■	□	■	□	□	□	
	□	□	■	■	□	■	□	■	■	□	□	
	□	■	□	□	□	■	□	□	□	■	□	
	■	■	■	□	■	■	■	□	■	■	■	
	□	■	□	□	□	■	□	□	□	■	□	
	□	■	■	□	■	■	■	□	■	■	□	

Para observar de forma más detallada la evolución de este AC con la configuración inicial dada, se puede proceder a determinar, por ejemplo, las 50 primeras iteraciones. Llevando a cabo un proceso similar al anterior, de modo que sólo se sustituyen los bits 1 por ■, dejando en blanco los lugares correspondientes a los bits 0, se obtiene una representación de su diagrama de evolución, que se puede observar en la Figura 6.

Figura 6. Ejemplo del diagrama de evolución del AC definido en (2)

3.2 Autómatas celulares de Wolfram

Consideremos el caso particular de los AC definidos de modo que su conjunto de estados es $S = \mathbb{Z}_2$ y el radio de la vecindad es $r = 1$. Para este caso particular de AC, el número de reglas de transición existentes es, según se mencionó anteriormente, $2^{2^{2r+1}} = 2^8 = 256$. Wolfram ([28]) ideó una notación consistente en asignar a cada una de las 256 reglas de transición un número comprendido entre 0 y 255. Dado que $r = 1$ la vecindad de una célula está formada por ella misma, la célula situada a su izquierda y la célula situada a su derecha. Como el conjunto de estados es $\mathbb{Z}_2$, los dos estados en que puede encontrarse una célula son 0 ó 1. Consecuentemente existen $2^3 = 8$ posibles configuraciones de la vecindad de una célula dada, a saber:

$$111 \quad 110 \quad 101 \quad 100 \quad 011 \quad 010 \quad 001 \quad 000$$

El primer dígito de cada configuración representa el estado de la célula de la izquierda, el dígito central el estado de la célula de cuya vecindad hablamos

y el último dígito hace referencia al estado de la célula de la derecha. Es claro que toda regla de transición consiste en asignar a cada una de estas posibles configuraciones de la vecindad un elemento de $\mathbb{Z}_2$. El número a asignar a una regla de transición se calculará de la siguiente forma:

1. Se aplica la regla de transición a cada una de las 8 configuraciones anteriores,
2. Se concatenan los bits obtenidos,
3. Se interpreta dicha concatenación como un número en base 2 y
4. Se considera la expresión decimal de dicho número.

El número así obtenido se denomina el *número de Wolfram* de la regla de transición considerada. En el caso concreto de la regla dada en (2) este proceso sería el siguiente:

$$\left.\begin{array}{llll} 111 \to 1, & 110 \to 0, & 101 \to 0, & 100 \to 1, \\ 011 \to 0, & 010 \to 1, & 001 \to 1, & 000 \to 0, \end{array}\right\} \Rightarrow 10010110_2 = 150,$$

de modo que el número de Wolfram de la regla de transición mencionada es el 150.

El propio Wolfram, tras múltiples simulaciones de los diferentes AC, estableció la siguiente clasificación de los mismos en virtud del comportamiento manifestado en los diagramas de evolución ([29]):

- AC *de clase* 1: son aquellos que evolucionan hacia estados homogéneos o constantes —o todo ceros, o todo unos—. Además, dicha evolución es independiente de la configuración inicial considerada.
- AC *de clase* 2: son los autómatas que dan lugar a conjuntos de estructuras periódicas y estables. En ellos la evolución del estado de una determinada célula a lo largo del tiempo estará influida por los estados de un grupo fijo de células de la configuración inicial.
- AC *de clase* 3: son todos aquellos autómatas celulares cuyo comportamiento se vuelve caótico con el paso del tiempo, de tal forma que el cambio de estado de una célula va a depender cada vez más de un mayor número de estados iniciales. Se ha conjeturado que el cálculo de dicho estado se puede hacer mediante un simple algoritmo.
- AC *de clase* 4: son aquellos que dan lugar a estructuras complejas, las cuales pueden permanecer localizadas en el espacio o bien moverse a lo largo del mismo. En esta clase, la evolución del estado de una célula en particular dependerá de una gran cantidad de estados iniciales, de manera que la determinación exacta de dicho estado es un problema cuya complejidad es equivalente a la propia simulación explícita del AC.

4 Los AC como generadores pseudoaleatorios

Como ya se mencionó (ver §2), para utilizar un cifrado en flujo es necesario disponer de un generador de bits pseudoaleatorio que genere la misma secuencia de bits, tanto en origen como en destino, a partir de una misma clave secreta. A continuación presentaremos algunos de los generadores de bits más utilizados en este tipo de cifrados, para pasar posteriormente a comentar el uso de los autómatas celulares como generadores pseudoaleatorios de secuencias cifrantes.

4.1 Generación de bits pseudoaleatorios

Un *generador de bits (o de números) aleatorio* es un dispositivo que proporciona como salida una secuencia de dígitos binarios (o de números) que son estadísticamente independientes. Algunos de los generadores de bits aleatorios diseñados por hardware están basados en la aleatoriedad de fenómenos físicos, como por ejemplo: La emisión de partículas durante un proceso radiactivo entre dos instantes de tiempo, el ruido producido por un diodo semiconductor en una corriente, etc. Debido a la imposibilidad práctica de que una secuencia obtenida por uno de estos generadores se pueda volver a obtener exactamente de la misma forma, en diferentes momentos, estos generadores de bits no suelen ser utilizados en criptología, al menos en los cifrados en flujo. Es posible utilizarlos en otras aplicaciones criptográficas como para generar determinados tipos de números (primos, de determinada longitud, etc.)

Por su parte, un *generador de bits (o de números) pseudoaleatorio* es un algoritmo determinístico que al darle como entrada una secuencia auténticamente aleatoria de longitud pequeña, proporciona una secuencia de bits (o de números) de longitud mucho mayor y que parece ser aleatoria. Las salidas de estos generadores no son aleatorias en el sentido de que cada vez que se ejecute el algoritmo con los mismos parámetros se van a obtener las mismas salidas. No obstante, lo que se desea conseguir con estos generadores es que las salidas obtenidas parezcan aleatorias, es decir, que resulte imposible poder distinguir entre la secuencia pseudoaleatoria obtenida y una realmente aleatoria en un tiempo polinómico ([4, Apéndice B]).

Estos generadores de números requieren verificar determinadas características en función de las aplicaciones prácticas a las que estén destinados. Una de ellas es la *aleatoriedad*, que debe analizarse para cada una de las secuencias que se generen y antes de ser utilizada en la práctica, de modo que la serie de números a emplear cumpla las propiedades que se supone verifican las series de números aleatorios (por ejemplo, los póstulados de Golomb [7]). Para llevar a cabo este análisis se recurre a determinados tests estadísticos diseñados ad hoc (véanse [4, Apéndice A], [12, Chapter 5]).

Una característica adicional que se requiere en criptografía es la *seguridad*, es decir, se trata de conseguir alguna garantía de que estos generadores son seguros en el sentido de que las salidas son imprevisibles. Para ello se recurre a analizar las operaciones matemáticas en las que se basa la definición del generador y la dificultad de los problemas matemáticos subyacentes.

Algunos de los generadores de bits pseudoaleatorios más utilizados son los siguientes:

1. *Generadores de bits lineales en congruencias*: son los generadores que utilizan la siguiente expresión recursiva:

$$x_{i+1} = (a \cdot x_i + c) \bmod m, \quad i \geq 0,$$

a partir de una semilla aleatoria dada x_0, considerando como secuencia de bits, el bit de paridad (es decir, el bit menos significativo) de cada uno de los números x_i generados.

2. *Generadores cuadráticos*: son los generadores definidos por

$$x_{i+1} = (a \cdot x_i^2 + b \cdot x_i + c) \bmod m, \quad i \geq 0.$$

También en este caso se considera como secuencia la definida por paridad(x_i).

3. *Generadores de registro de desplazamiento realimentados linealmente (LFSR)*: son generadores cuya regla de recurrencia está definida por:

$$x_i = a_1 \cdot x_{i-1} \oplus a_2 \cdot x_{i-2} \oplus \ldots \oplus a_r \cdot x_{i-r}, \quad i > r,$$

siendo $\oplus$ la operación XOR.

4.2 Los AC de Wolfram como cifradores en flujo

Como ya hemos mencionado, una de las principales aplicaciones de los AC a la criptografía es su uso como generadores de bits para los cifrados en flujo. En esta sección veremos cómo los AC lineales, en particular los AC de Wolfram, pueden generar números pseudoaleatorios.

Consideraremos los AC de Wolfram definidos por las reglas de transición números 30 y 45, respectivamente, que parecen ser los que tienen mejores propiedades como generadores de bits pseudoaleatorios ([31]):

$$\begin{aligned} a_i^{(t+1)} &= \left(a_{i-1}^{(t)} + a_i^{(t)} + a_{i+1}^{(t)} + a_i^{(t)} \cdot a_{i+1}^{(t)}\right) \bmod 2 \\ &= a_{i-1}^{(t)} \text{ XOR } \left(a_i^{(t)} \text{ OR } a_{i+1}^{(t)}\right), \qquad (3) \\ a_i^{(t+1)} &= \left(1 + a_{i-1}^{(t)} + a_{i+1}^{(t)} + a_i^{(t)} \cdot a_{i+1}^{(t)}\right) \bmod 2 \\ &= a_{i-1}^{(t)} \text{ XOR } \left(a_i^{(t)} \text{ OR } \left(\text{NOT } a_{i+1}^{(t)}\right)\right), \qquad (4) \end{aligned}$$

Los diagramas de evolución de cada uno de los dos AC anteriores pueden verse en las Figuras 7 y 8, respectivamente.

Figura 7. Diagrama de evolución del AC de regla 30

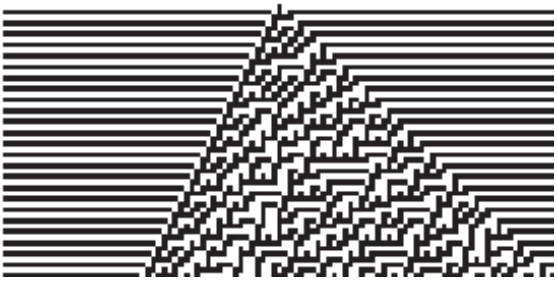

Figura 8. Diagrama de evolución del AC de regla 45

La obtención de tales representaciones se ha hecho de forma análoga a como se hizo para el AC definido por la regla dada en (2) (ver Figura 6). En este caso, se han considerado 50 iteraciones para cada uno de los dos AC definidos en (3) y (4) con la misma configuración inicial, a saber:

$$\left(0, \overset{(50}{\ldots}, 0, 1, 0, \overset{(50}{\ldots}, 0\right).$$

Se podría pensar en utilizar estos dos AC como generadores pseudoaleatorios sin más que considerar como semilla a la configuración inicial y tomar como salida la lista de ceros y unos definida por la última configuración. Sin embargo, hacerlo así supondría que se generaría una lista de tantos bits como bits de partida, dado que el número de células de todas las configuraciones es el mismo, por lo que este procedimiento no parece adecuado. Si lo que se desea es obtener una secuencia de bits cuya longitud sea mucho mayor que la longitud de la semilla o configuración inicial, es conveniente considerar como secuencia de salida la evolución de una de las células del AC. Con ello sólo haría falta conocer el estado inicial de unas pocas células e iterar el AC tantas veces como bits se requieran.

Si se considera como salida del AC la evolución de la célula central (sin considerar su estado inicial), a lo largo de 100 iteraciones, para la configuración inicial o semilla dada por

$$(0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0)\,, \tag{5}$$

se tiene la siguiente salida de 100 bits:

$$1,0,1,1,1,0,0,1,1,0,0,0,1,0,1,1,0,0,1,0,0,1,1,1,1,0,1,1,1,1,0,1,0,1,$$
$$1,1,1,0,1,0,1,0,0,1,0,0,0,1,0,1,1,1,0,0,0,0,0,1,0,0,1,1,0,1,1,1,1,1,$$
$$0,0,1,0,0,1,0,0,1,0,0,1,0,1,1,1,1,0,1,0,1,1,1,0,0,1,0,1,0,1,0,0. \qquad (6)$$

El diagrama de evolución en este caso se presenta en la Figura 9 (la evolución de la célula central se ha girado 90 grados y sigue un recorrido evolutivo de izquierda a derecha).

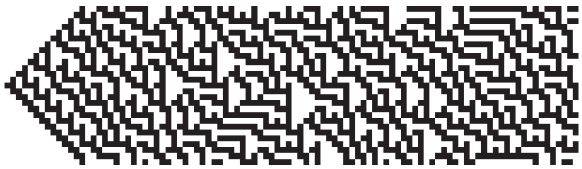

Figura 9. Diagrama de evolución del AC de regla 30 con 100 iteraciones

Se puede observar que con la configuración inicial de 25 células dada en (5), se ha obtenido la secuencia de 100 bits presentada en (6), si bien esta longitud podría ser mucho mayor sin más que iterar más veces la evolución del AC considerado.

La decisión de la secuencia de bits a utilizar como salida dependerá del tipo de AC de que se trate. Téngase en cuenta que exiten AC cuya evolución es muy simétrica, lo que dificulta su uso como generadores de números pseudoaleatorios. Como ejemplo puede verse el AC definido por la regla 22, cuya expresión es:

$$a_i^{(t+1)} = \left(a_{i-1}^{(t)} + a_i^{(t)} + a_{i+1}^{(t)} + a_{i-1}^{(t)} \cdot a_i^{(t)} \cdot a_{i+1}^{(t)}\right) \bmod 2$$

y cuya representación se muestra en la Figura 10.

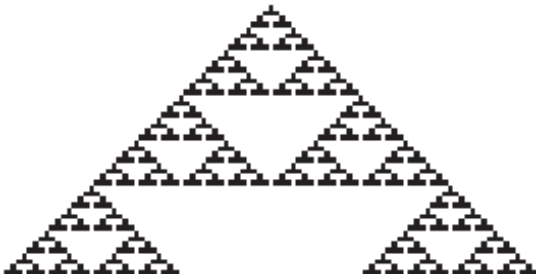

Figura 10. Diagrama de evolución del AC de regla 22

Una vez que se ha generado una secuencia pseudoaleatoria de bits (formada por los valores de los estados de las células que ocupan la posición $\langle i \rangle$ a lo largo del tiempo), se puede considerar el cifrado en flujo cuya clave secreta (que comparten tanto el remitente como el destinatario del mensaje) es la configuración inicial considerada.

Ejemplo. Supongamos que se tiene el AC definido en (3), para el que se utiliza la semilla (5) y como salida de bits pseudoaleatorios la dada en (6). Para obtener el criptograma, C, correspondiente al mensaje

SECRETO

basta con concatenar el valor en binario de cada una de las letras del mensaje, según el código ASCII, y luego sumar, bit a bit, el mensaje obtenido, M, con la secuencia de bits o clave, K:

$$\begin{array}{rl} & \overbrace{}^{\text{S}}\overbrace{}^{\text{E}}\overbrace{}^{\text{C}}\overbrace{}^{\text{R}}\overbrace{}^{\text{E}}\overbrace{}^{\text{T}}\overbrace{}^{\text{O}} \\ M: & 01010011010001010100001101010010010001010101010001001111 \\ K: & 10111001100010110010011110111101011110101001000101110000 \\ C: & \underbrace{11101010110011100110010011101111001111111100010100111111}_{\hat{\text{e}}\qquad\hat{\text{I}}\qquad\text{d}\qquad\ddot{\text{i}}\qquad?\qquad\mathring{\text{A}}\qquad?} \end{array}$$

De este modo, el criptograma que se envía es el siguiente:

êÎdï?Å?

El destinatario recupera el mensaje original sin más que concatenar el valor en binario de cada una de las letras o símbolos del criptograma recibido y sumar, bit a bit, el criptograma, C, con la clave, K, dado que esta operación es una involución.

La seguridad de este criptosistema está basada en la impredecibilidad de la clave K, es decir, en la dificultad de poder obtener la secuencia pseudoaleatoria generada por el AC. De ahí la importancia de que los AC elegidos como generadores de bits pseudoaleatorios tengan buenas propiedades estadísticas.

En [11] se estudia la seguridad del criptosistema definido anteriormente con el AC dado en (3) mediante el ataque al texto claro conocido. Los autores desarrollan un algoritmo de criptoanálisis para este AC, de modo que el ataque tiene éxito sobre un PC si el tamaño de la clave (es decir, el número de células de la configuración inicial del AC) está comprendido entre 300 y 500 bits. Para adversarios con mayores potencias de cálculo (por ejemplo, computación en paralelo), se recomienda que el tamaño de la clave sea superior a 1000 bits. El ataque se fundamenta en el hecho de que a partir de la expresión dada en (3), es posible obtener la siguiente fórmula lineal en a_{i-1} para el instante t:

$$a_{i-1}^{(t)} = \left(a_i^{(t+1)} + a_i^{(t)} + a_{i+1}^{(t)} + a_i^{(t)} \cdot a_{i+1}^{(t)}\right) \bmod 2, \tag{7}$$

que permite calcular el valor del estado de la célula $\langle i-1\rangle$ en un instante dado, en función de los valores de los estados de las células $\langle i\rangle$ e $\langle i+1\rangle$ en el mismo instante y de la célula $\langle i\rangle$ en el instante siguiente. Es decir, si se conocen los valores de n estados consecutivos de la célula $\langle i\rangle$ y de $n-1$ estados de la célula adyacente $\langle i+1\rangle$, es posible deteminar $n-1$ estados de la otra célula adyacente a la célula $\langle i\rangle$, esto es de la $\langle i-1\rangle$. En efecto, consideremos el siguiente

Ejemplo. Supongamos que se conocen los valores de $n = 5$ estados de la célula $\langle 5\rangle$ para los instantes $t, \ldots, t+4$: $(0, 0, 1, 1, 0)$ y de $n-1 = 4$ estados de la célula adyacente $\langle 6\rangle$ para los instantes $t, \ldots, t+3$: $(1, 1, 1, 0)$. Entonces, aplicando la expresión (7) para el instante $t+3$ y para el estado $i-1 = 4$, se tiene

$$\begin{aligned} a_4^{(t+3)} &= \left(a_5^{(t+4)} + a_5^{(t+3)} + a_6^{(t+3)} + a_5^{(t+3)} \cdot a_6^{(t+3)}\right) \bmod 2 \\ &= (0 + 1 + 0 + 1 \cdot 0) \bmod 2 = 1, \end{aligned}$$

ahora, reiterando esta misma expresión, se calculan los siguientes valores:

$$\begin{aligned} a_4^{(t+2)} &= \left(a_5^{(t+3)} + a_5^{(t+2)} + a_6^{(t+2)} + a_5^{(t+2)} \cdot a_6^{(t+2)}\right) \bmod 2 \\ &= (1 + 1 + 1 + 1 \cdot 1) \bmod 2 = 0, \\ a_4^{(t+1)} &= \left(a_5^{(t+2)} + a_5^{(t+1)} + a_6^{(t+1)} + a_5^{(t+1)} \cdot a_6^{(t+1)}\right) \bmod 2 \\ &= (1 + 0 + 1 + 0 \cdot 1) \bmod 2 = 0, \\ a_4^{(t)} &= \left(a_5^{(t+1)} + a_5^{(t)} + a_6^{(t)} + a_5^{(t)} \cdot a_6^{(t)}\right) \bmod 2 \\ &= (0 + 0 + 1 + 0 \cdot 1) \bmod 2 = 1, \end{aligned}$$

con lo que se obtienen los valores de los estados de la célula $\langle 4\rangle$ para los instantes $t, \ldots, t+3$: $(1, 0, 0, 1)$.

A partir de la expresión (7) y con el criptosistema descrito anteriormente, si se considera el ataque al texto claro conocido, se dispone de la pareja formada por un trozo de texto claro y su correspondiente criptograma. Sumando bit a bit ambas secuencias, se obtiene la evolución de la célula que ocupa la posición i-ésima, es decir, una parte de la secuencia cifrante. Ahora bien, por la fórmula (7) si se conocieran además los estados de la célula $\langle i+1\rangle$, adyacente a la célula $\langle i\rangle$, en cada instante de tiempo, se podría recuperar el diagrama de evolución completo del AC utilizado y, en particular, la configuración inicial, esto es, la clave del criptosistema que permitiría descifrar cualquier otro criptograma. Así pues, este ataque prueba que el conocimiento de la evolución de la célula $\langle i+1\rangle$ es equivalente al conocimiento de la clave. A partir de este hecho, el criptoanálisis sólo busca cómo determinar dicha evolución.

Concretamente, el algoritmo dado en [11] permite obtener la clave de $2n+1$ bits del criptosistema a partir del conocimiento de n bits de la secuencia cifrante, que es la evolución de una de las células del AC. Para ello se supone que la célula cuya evolución se conoce es la central, es decir, se conocen los n valores $a_i^{(t+k)}$, con $k = 0, \ldots, n-1$. A continuación se generan de forma aleatoria los $n-1$ valores de las células que están a su derecha para el instante t, es decir, $a_{i+j}^{(t)}$, con $j = 1, \ldots, n-1$. A partir de aquí, se determinan los valores de las células que aparecen en el triángulo derecho de la configuración del AC, es decir, los estados de las células $a_{i+j}^{(t+k)}$, siendo $j = 1, \ldots, n-k$, con $k = 1, \ldots, n-1$. Posteriormente se deduce el valor de las células del triángulo izquierdo, es decir,

de $a_{i-j}^{(t+k)}$, siendo $k = 0, \ldots, n-j$, con $j = 1, \ldots, n-1$. Con ello se obtiene la configuración inicial completa que corresponde a la secuencia cifrante del AC dado. Se puede comprobar que con la configuración inicial obtenida se genera un AC que tiene como evolución de la célula $\langle i \rangle$ la dada originalmente. El algoritmo escrito en pseudocódigo es el siguiente:

1. For j from 1 to $n-1$ do $a_{i+j}^{t} := \text{rand}(0..1)$

2. For k from 1 to $n-1$ do
   For j from 1 to $n-k$ do
   $a_{i+j}^{k} := \left(a_{n+j-1}^{k-1} + a_{n+j}^{k-1} + a_{n+j+1}^{k-1} + a_{n+j}^{k-1} \cdot a_{n+j+1}^{k-1}\right) \bmod 2$

3. For j from 1 to $n-1$ do
   For k from $n-j$ by -1 to 0 do
   $a_{n-j}^{k} := \left(a_{n-j+1}^{k+1} + a_{n-j+1}^{k} + a_{n-j+2}^{k} + a_{n-j+1}^{k} \cdot a_{n-j+2}^{k}\right) \bmod 2$

A modo de ejemplo, se presenta el siguiente caso.

Ejemplo. Supongamos que la secuencia cifrante viene dada por los siguientes $n = 11$ valores: $s = (1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1)$. Aplicando el primer paso del algoritmo obtenemos los siguientes $n - 1 = 10$ valores aleatorios: $1, 1, 0, 0, 1, 0, 1, 1, 1, 0$. Mediante el segundo paso del algoritmo se construye el triángulo de la derecha:

1	1	0	0	1	0	1	1	1	0
0	0	1	1	1	0	1	0	0	
0	1	1	0	0	0	1	1		
0	1	0	1	0	1	1			
0	1	0	1	0	1				
1	1	0	1	0					
1	0	0	1						
0	1	1							
1	1								
0									

Con el siguiente paso se determina el triángulo de la izquierda:

1	0	0	1	0	1	1	0	0	1
	1	1	1	0	1	0	1	1	1
		0	0	0	1	0	1	0	0
			0	1	1	0	1	1	1
				1	0	0	1	0	0
					1	1	1	1	0
						0	0	0	1
							0	1	1
								1	0
									0

Finalmente, se puede comprobar que la configuración inicial (esto es, la

clave) de $2n-1=21$ bits obtenida

$$C^{(0)}=(1,0,0,1,0,1,1,0,0,1,1,1,1,0,0,1,0,1,1,1,0)$$

genera, mediante el AC, la secuencia cifrante original. En efecto, la evolución del AC con dicha configuración es la siguiente:

1	0	0	1	0	1	1	0	0	1	1	1	1	0	0	1	0	1	1	1	0
1	1	1	1	0	1	0	1	1	1	0	0	0	1	1	1	0	1	0	0	0
1	0	0	0	0	1	0	1	0	0	1	0	1	1	0	0	0	1	1	0	1
0	1	0	0	1	1	0	1	1	1	1	0	1	0	1	0	1	1	0	0	1
0	1	1	1	1	0	0	1	0	0	0	0	1	0	1	0	1	0	1	1	1
0	1	0	0	0	1	1	1	1	0	0	1	1	0	1	0	1	0	1	0	0
1	1	1	0	1	1	0	0	0	1	1	1	0	0	1	0	1	0	1	1	0
1	0	0	0	1	0	1	0	1	1	0	0	1	1	1	0	1	0	1	0	0
1	1	0	1	1	0	1	0	1	0	1	1	1	0	0	0	1	0	1	1	1
0	0	0	1	0	0	1	0	1	0	1	0	0	1	0	1	1	0	1	0	0
0	0	1	1	1	1	1	0	1	0	1	1	1	1	0	1	0	0	1	1	0

Obsérvese que el algoritmo requiere del conocimiento de n bits de la secuencia cifrante, es decir, de n valores de la evolución de una célula para poder determinar una clave de $2n-1$ bits (esto es, una configuración de $2n-1$ células). Así pues, si se conocen menos de n valores de la secuencia cifrante, no es posible recuperar la clave original, supuesto que ésta tenga $2n-1$ bits. En efecto, basta con observar cómo en el siguiente ejemplo, no se recupera la clave original, lo que lleva a que la evolución de la célula central del AC obtenido con el algoritmo anterior no conduzca a la secuencia cifrante de partida.

Ejemplo. Supongamos ahora que la clave tiene la misma longitud que antes, es decir, $2n-1=21$ bits, pero sólo se conocen 6 valores de la secuencia cifrante: $(1,0,1,1,0,0)$. En este caso, el algoritmo anterior daría la siguiente evolución y configuración inicial:

1	1	0	0	1	1	1	1	0	0	1
	0	1	1	1	0	0	0	1	1	
		1	0	0	1	0	1	1		
			1	1	1	0	1			
				0	0	0				
					0					

mientras que la evolución del AC con la clave obtenida proporciona los siguientes

resultados:

1	1	0	0	1	1	1	1	0	0	1
0	0	1	1	1	0	0	0	1	1	1
1	1	1	0	0	1	0	1	1	0	0
1	0	0	1	1	1	0	1	0	1	1
0	1	1	1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	1	0	1	1
0	0	1	1	1	0	1	0	0	1	0
1	0	0	0	1	0	1	0	1	1	0
1	1	0	1	1	0	1	0	1	0	1
0	0	0	1	0	0	1	0	1	0	1
0	0	1	1	1	1	1	0	1	0	1

Como se puede ver, no todos los valores de la evolución de la célula central de este caso $(1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1)$ coinciden con los valores de la secuencia dada en el primer ejemplo: $s = (1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1)$.

Para el AC definido en (4), el criptoanálisis es aún más efectivo, es decir, se deben considerar claves de longitudes mayores para obtener la misma seguridad que con el AC definido en (3).

5 Los AC como generadores de claves

En numerosos criptosistemas es necesario generar claves numéricas de forma aleatoria. Por ejemplo, en el criptosistema RSA ([18, 20]) hace falta obtener dos números primos grandes, de modo que a partir de ellos se genere el módulo para la clave pública. También hace falta generar una clave de sesión a la hora de cifrar mediante otros muchos criptosistemas, como, por ejemplo, en el propuesto por ElGamal ([3]), basado en el logaritmo discreto. Por tanto, y dado que los AC de Wolfram pueden generar números aleatorios, cabe la posibilidad de utilizarlos con este fin.

En general, las claves numéricas que se utilizan vienen caracterizadas por su longitud, de ahí que para generar una clave de k bits sea necesario partir de una determinada configuración del AC de Wolfram definido en (3) e iterarlo k veces. Si la única restricción es que la clave sea aleatoria y de k bits, para garantizar que el número a obtener tenga exactamente k bits se puede generar una secuencia pseudoaleatoria de $k - 1$ bits y añadir un bit 1 a la izquierda. Posteriormente, si se desea que la clave generada sea un número en base 10, bastará con transformar la colección de bits, considerada como un número en binario, a base decimal.

Ejemplo. Si se desea generar una clave de 256 bits, bastará con utilizar una configuración inicial de, por ejemplo, 25 células e iterar el AC 255 veces. Así, si la configuración inicial es

$$(0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0),$$

después de 255 iteraciones se obtiene la siguiente secuencia pseudoaleatoria de bits:

$$0,1,0,1,0,1,0,1,1,1,1,0,1,1,1,1,0,1,1,0,1,1,0,1,0,0,1,0,1,1,1,0,0,0,1,$$
$$0,1,1,0,1,1,0,0,1,0,1,1,1,1,0,0,1,0,0,0,1,0,0,1,0,1,0,1,1,0,1,0,1,0,1,$$
$$0,1,1,0,0,1,0,1,1,1,1,0,1,0,1,0,0,1,0,1,1,1,1,1,1,1,1,0,0,1,0,0,1,1,0,$$
$$0,0,0,0,0,0,1,0,0,0,1,1,0,0,1,0,1,1,1,1,1,1,0,1,0,0,0,1,0,1,1,0,0,0,0,$$
$$1,0,0,0,1,1,0,1,1,0,0,0,1,1,1,0,1,0,1,0,0,0,0,1,0,0,1,0,1,1,1,1,0,0,0,$$
$$0,1,1,1,1,1,0,1,0,0,1,0,1,1,0,0,0,0,1,0,0,0,1,0,1,1,0,0,0,0,1,1,0,0,1,$$
$$1,0,1,1,1,0,1,0,0,1,1,1,1,1,1,0,1,1,1,1,0,0,1,0,0,0,0,0,1,1,1,0,0,1,1,$$
$$0,1,1,0,1,0,1,0,0,1.$$

Como dicha secuencia comienza por 0, al añadir el bit 1 delante y convertir el número obtenido en base decimal, se obtiene que el valor de la clave es un número de 77 dígitos:

$$77330\ 85567\ 53433\ 11675\ 30533\ 45272\ 09632\ 25354\ 28229\ 51380\ 28346\ 06129$$
$$11250\ 49192\ 67608\ 73.$$

En el caso de que se deseen generar números con otras condiciones añadidas, como ser primos, por ejemplo, se puede realizar el proceso anterior de forma parecida, si bien se deberá utilizar algún criterio o test que asegure (aunque sea de forma probabilística) que el número obtenido sea primo. Una forma de ahorrar algún tiempo de computación para generar un número primo de exactamente k bits, consiste en generar una secuencia pseudoaleatoria de $k-2$ bits y luego añadir sendos bits 1 al inicio y al final de dicha secuencia. De esta forma queda garantizado que el número decimal en el que se convierte dicha secuencia tiene exactamente k bits y que es impar (hecho que se deduce porque el último bit del número —el bit de paridad— es un 1). En el caso del número generado anteriormente, se puede afirmar que es compuesto dado que su factorización como producto de primos es la siguiente:

$$19 \cdot 29 \cdot 313 \cdot 2663 \cdot 125093$$
$$\cdot\, 1346023694316866997551227374112895407476083323419667212185295469.$$

Uno de los criterios de primalidad más extendidos es el debido a Miller-Rabin ([4, 12]). Éste es un test probabilístico para decidir si un número dado es o no primo y está basado en los pseudoprimos de Euler (dados dos números enteros, a y n, se dice que n es un *pseudoprimo de Euler de base* a si son primos entre sí y si $a^{(n-1)/2} \equiv \left(\frac{a}{n}\right) \bmod n$, donde $\left(\frac{a}{n}\right)$ representa el símbolo de Legendre). Si la salida del test afirma que el número es *compuesto*, entonces el número evaluado es, en efecto, compuesto; mientras que si la salida del test indica que es *primo*, dicho número es primo con una probabilidad $1-(1/4)^t$, siendo t el número de veces que se lleva a cabo el test. Por tanto, para obtener una probabilidad

del $99,99\%$ de que un número dado sea primo, basta con ejecutar el test de Miller-Rabin $t = 7$ veces para diferentes valores de la base. Como ejemplo de lo que acabamos de señalar presentamos el siguiente

Ejemplo. Si se desea generar un número primo de 100 bits, consideramos la siguiente configuración inicial para el AC:

$$(1,0,0,1,0,1,0,1,0,1,1,1,1,1,0,1,0,0,1,1,0,1,0,0,0)$$

y lo iteramos 98 veces, obteniendo la siguiente secuencia de bits:

$$1,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,0,1,1,1,0,1,0,1,0,1,0,1,1,1,0,0,1,1,0,$$
$$0,0,0,1,0,1,0,0,0,1,0,1,1,1,1,1,1,1,0,1,1,0,1,0,1,0,0,1,0,1,1,1,1,1,0,$$
$$1,0,0,1,1,0,0,0,0,1,0,1,0,1,1,0,0,1,0,0,1,1,1,1,1,1,0,1,0.$$

Añadiendo ahora un bit 1 al inicio y otro al final de la secuencia anterior, se obtienen 100 bits. La expresión de esta secuencia de bits en base 10 es

$$950984169159075177987037645301.$$

Ejecutando el test de Miller-Rabin 7 veces a dicho número se obtiene como salida que dicho número es primo, por lo que se puede asegurar con una probabilidad del $99,99\%$ de que lo es.

6 Conclusiones

En el presente artículo hemos definido y señalado algunas características y propiedades de los autómatas celulares, en particular, de los AC lineales y, de forma más concreta, de los llamados autómatas celulares de Wolfram. De entre todos ellos, nos hemos detenido con mayor detalle en el definido por la regla 30. También hemos presentado algunas nociones básicas de criptografía con el fin de comentar algunas aplicaciones de los AC a esta ciencia. A modo de resumen podemos señalar que los autómatas celulares lineales, y principalmente el definido por la regla 30, que es el que presenta mejores propiedades pseudoaleatorias, pueden ser utilizados como generadores de bits pseudoaleatorios con diferentes propósitos. Uno de ellos es el de ser utilizados en el algoritmo que genera la secuencia cifrante en los cifrados en flujo, siempre que se tengan en cuenta determinadas precauciones. Estas precauciones están relacionadas con temas referidos a su seguridad o impredecibilidad, dado que existen procedimientos y algoritmos que permiten determinar la secuencia generada por un AC, si la longitud de la configuración inicial, que es utilizada como clave, no es lo suficientemente grande. Por otra parte, también hemos señalado la posibilidad de utilizar las secuencias de bits generadas por un AC de Wolfram como claves de sesión, o para generar determinadas claves parciales, verificando propiedades particulares, como ser números primos, por ejemplo.

Referencias

[1] S. R. Blackburn, S. Murphy and K. G. Paterson. *Comments on: Theory and applications of cellular automata in cryptography.* IEEE Trans. Comput. **46**, 5 (1997), 637–639.

[2] K. Cattell, S. Zhang, M. Serra and J. C. Muzio. *2-by-n hybrid cellular automata with regular configuration: theory and application.* IEEE Trans. Comput. **48**, 3 (1999), 285–295.

[3] T. ElGamal. *A public-key cryptosystem and a signature scheme based on discrete logarithm*, IEEE Trans. Inform. Theory **31** (1985), 469–472.

[4] A. Fúster, D. de la Guía, L. Hernández, F. Montoya y J. Muñoz. *Técnicas criptográficas de protección de datos.* RA-MA, 2^a ed., Madrid, 2000.

[5] M. Gardner. *The fantastic combinations of John Conway's new game of 'life'.* Scientific American, Octubre (1970), 100.

[6] J. Gleick. *Caos: La creación de una ciencia.* Seix Barral, Barcelona, 1988.

[7] S. W. Golomb. *Shift register sequences.* Holden-Day, San Francisco, 1967.

[8] D. de la Guía y A. Fúster. *Estudio de autómatas celulares para criptografía.* Actas de la IV Reunión Española de Criptología, 167–174, J. Tena y M. F. Blanco (ed.), Universidad de Valladolid, 1996.

[9] M. W. Hirsch y S. Smale. *Ecuaciones diferenciales, sistemas dinámicos y álgebra lineal.* Alianza Universidad Textos, Madrid, 1983.

[10] E. N. Lorenz. *La esencia del caos.* Debate, Pensamiento, Barcelona, 1995.

[11] W. Meier and O. Staffelbach. *Analysis of pseudo random sequences generated by cellular automata.* Advances in Cryptology-Proceedings of EUROCRYPT'91, LNCS **547** (1991), 186–199, Springer-Verlag, Berlín.

[12] A. Menezes, P. van Oorschot and S. Vanstone. *Handbook of applied cryptography.* CRC Press, Boca Ratón, FL., 1997.

[13] K. Nagel and M. Schreckenber, *A cellular automaton model for freeway traffic.* J. Physique I, **2** (1992), 2221.

[14] S. Nandi, B. K. Karr and P. Pal Chaudhuri. *Theory and applications of cellular automata in cryptography.* IEEE Trans. Comput. **43**, 12 (1994), 1346–1357.

[15] J. von Neumann. *Theory of selft-reproducing automata.* A. W. Burks (ed.), University of Illinois Press, 1966.

[16] D. Ostrov and R. Rucker. *Continuous-valued cellular automata for nonlinear wave equations.* Complex Systems **10** (1996), 91–119.

[17] N. H. Packard and S. Wolfram. *Two-dimensional cellular automata.* J. Statist. Phys. **38** (1985), 901–946.

[18] A. Quirós Gracián. *Números primos y criptografía.* Bol. Soc. Esp. Mat. Apl. **17** (2001), 13–21.

[19] Rijndael: accesible en http://csrc.nist.gov/encryption/aes/

[20] R. L. Rivest, A. Shamir and L. Adleman. *A method for obtaining digital signatures and public-key cryptosystems.* Commun. ACM **21** (1978), 120–126.

[21] R. Schmitz. *Use of chaotic dynamical systems in cryptography.* J. Franklin Inst. **338** (2001), 429–441.

[22] M. Sipper. *The evolution of parallel cellular machine: The cellular programming approach.* Springer-Verlag, Berlin, 1997.

[23] M. Sipper and M. Tomassini. *Generating parallel random number generators by cellular programming.* Internat. J. Modern Phys. C **7**, 2 (1996), 181-190.

[24] —. *Computation in artificially evolved, non-uniform cellular automata.* Theoret. Comput. Sci. **217**, 1 (1999), 81-98.

[25] S. Tezuka and M. Fushimi. *A method of designing cellular automata as pseudorandom number generator for built-in-self-test for VLSI.* Finite fileds: Theory, applications and algorithms, 363–367, Contemp. Math. **168**, Amer. Math. Soc. Providence, RI, 1994.

[26] A. J. Tomeu Hardasmal y R. Ruíz Rentero. *Cifrado de cadenas mediante autómatas celulares.* Actas de la IV Reunión Española de Criptología, 175–182, J. Tena y M. F. Blanco (ed.), Universidad de Valladolid, 1996.

[27] S. Ulam. *On some mathematical problems connected with patterns of growth of figures.* A. W. Burks (ed.), Essays on Cellular Automata, University of Illinois Press, 1970.

[28] S. Wolfram. *Cellular automata.* Los Alamos Science **9** (1983), 2–21.

[29] —. *Universality and complexity in cellular automata.* Physica D **10** (1984), 1–35.

[30] —. *Cryptography with cellular automata.* Advances in Cryptology-Proceedings of CRYPTO'85, LNCS **218** (1986), 429–432, Springer-Verlag, Berlín.

[31] —. *Random sequence generation by cellular automata.* Adv. in Appl. Math. **7** (1986), 123–169.