Extensión y parametrización de un generador pseudoaleatorio matricial

Rafael Álvarez
Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante
Email: ralvarez@dccia.ua.es

Francisco-Miguel Martínez
Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante
Email: fmartine@dccia.ua.es

José-Francisco Vicent
Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante
Email: jvicent@dccia.ua.es

Antonio Zamora
Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante
Email: zamora@dccia.ua.es

***Resumen*—En este trabajo se describe un nuevo generador pseudoleatorio basado en técnicas matriciales, extensión de otro desarrollado previamente por algunos de los autores de este artículo. Los buenos resultados estadísticos obtenidos indican que puede hacerse un uso criptográfico del mismo para generar claves de sesión, valores de desafío, etc. Se trata de un algoritmo muy flexible, que puede adaptarse con facilidad para satisfacer diversos requerimientos de hardware y software, capaz de producir secuencias de períodos grandes tomando primos y tamaños de matrices muy pequeños.**

I. Introducción

Es un hecho conocido que las empresas, las administraciones e incluso los particulares han incrementando la demanda de sistemas y servicios cada vez más seguros que hacen uso de herramientas criptográficas, generalmente combinando criptosistemas de clave pública (para acordar una clave de sesión) y criptosistemas de clave privada para transmitir datos de forma confidencial.

Las claves, los números primos, los valores de desafío o las secuencias cifrantes de muchos criptosistemas (véanse [14], [19], [21]) necesitan ser lo suficientemente impredecibles como para que las probabilidades de todos los valores posibles estén razonablemente equilibradas; evitando, de esta manera ataques que supongan reducciones del espacio de búsqueda. Estos valores se obtienen de secuencias aleatorias que pueden ser realmente aleatorias o, simplemente, comportarse como tal.

La generación de secuencias aleatorias es relevante en esquemas de cifrado en flujo, esquemas de firma digital, la generación de claves de los sistemas de cifrado, sistemas de identificación de desafío-respuesta y en muchos otros protocolos criptográficos.

Dentro de los generadores aleatorios podemos distinguir entre los no deterministas y los deterministas (véanse [23], [24]). En los primeros, se producen secuencias de bits que no se pueden reproducir y que, por lo general, se obtienen al explotar fuentes naturales de azar, como la desintegración radiactiva, el ruido térmico, la turbulencia del aire dentro de un sistema físico, etc., así como elementos software como el contenido de los buffers, el movimiento del ratón, etc. En los segundos, las secuencias de bits se generan mediante un algoritmo determinista cuya salida es perfectamente reproducible en función de la entrada o semilla; se les conoce como generadores pseudoaleatorios (véase [8], [10], [11], [13]) por que, aunque, las secuencias generadas no son genuinamente aleatorias se comportan como tal.

En las aplicaciones de los generadores pseudoaleatorios a la seguridad, necesitamos producir secuencias de grandes períodos, complejidades lineales altas y una buena distribución estadística que las haga impredecibles. Si no es posible algebráicamente describir las propiedades de las secuencias generadas por un generador pseudoaleatorio (hecho bastante común) se recurre a diversas pruebas estadísticas (batería de tests) que pueden ayudar a detectar las posibles debilidades del generador. Mediante estas pruebas, se analiza la frecuencia de bits individuales, de parejas y otros patrones de bits; así como la autocorrelación y la complejidad lineal (véase [9], [20], [22]).

Un generador muy conocido es Blum Blum Shub (BBS) [5]. Se basa en dos números primos de gran tamaño p y q, que satisfacen $p \equiv 3 \pmod 4$ y $q \equiv 3 \pmod 4$, y un número s,escogido aleatoriamente,que es primo con $n = pq$. Definiendo $X_0 = s^2 \mod n$ y $X_i = (X_i - 1)^2 \mod n$, la secuencia de salida se obtiene tomando el bit menos significativo de cada X_i. La seguridad del generador BBS se fundamenta en el coste computacional que supone factorizar n.

Otros generadores pseudoaleatorios están basados en registros de desplazamiento con retroalimentación lineal (LFSR, véanse [7]) que pueden implementarse fácilmente en hardware y analizarse su estructura y propiedades con relativa facilidad. También los hay que combinan un cifrador en bloque de diferentes maneras para obtener una secuencia pseudoaleatoria con seguridad criptográfica (véase [25]) así como autores que proponen una familia de generadores pseudoaleatorios con propiedades criptográficas [12].

En este trabajo se describe un nuevo generador pseudoleatorio basado en técnicas matriciales, extensión de otro desarrollado previamente por algunos de los autores [1] de este artículo. El nuevo generador se puede utilizar para generar claves de sesión, valores de desafío, números de secuencia, etc. Se trata de un algoritmo muy flexible, que se puede ajustar para satisfacer diversos requerimientos de memoria o velocidad y que presenta muy buenas características estadísticas.

El resto del artículo está organizado de la siguiente manera:

en la sección II se presentan algunos resultados básicos que fundamentan la descripción del generador; seguidamente, en la sección III, se realiza un estudio sobre el orden de las matrices del conjunto Θ, descrito en la sección II . En la sección IV se hace una descripción del generador; la sección V incluye los resultados experimentales obtenidos para nuestra propuesta; finalmente, exponemos algunas conclusiones en la sección VI.

II. Preliminares

Dado p un número primo y r, s, t, números naturales, denotamos por $\mathrm{Mat}_r(\mathbb{Z}_p)$, $\mathrm{Mat}_s(\mathbb{Z}_p)$, $\mathrm{Mat}_t(\mathbb{Z}_p)$, $\mathrm{Mat}_{r\times s}(\mathbb{Z}_p)$, $\mathrm{Mat}_{r\times t}(\mathbb{Z}_p)$ y $\mathrm{Mat}_{s\times t}(\mathbb{Z}_p)$ a las matrices de tamaño $r\times r$, $s\times s$, $t\times t$, $r\times s$, $r\times t$ y $s\times t$, respectivamente, con elementos en $\mathbb{Z}_p$ y por $\mathrm{GL}_r(\mathbb{Z}_p)$, $\mathrm{GL}_s(\mathbb{Z}_p)$ y $\mathrm{GL}_t(\mathbb{Z}_p)$ a las matrices invertibles de tamaño $r\times r$, $s\times s$ y $t\times t$, también con elementos en $\mathbb{Z}_p$.

Consideramos el conjunto Θ de matrices

$$M = \begin{bmatrix} A & Y & X \\ \boldsymbol{0} & B & Z \\ \boldsymbol{0} & \boldsymbol{0} & C \end{bmatrix}, \tag{1}$$

donde $A \in \mathrm{GL}_r(\mathbb{Z}_p)$, $B \in \mathrm{GL}_s(\mathbb{Z}_p)$, $C \in \mathrm{GL}_t(\mathbb{Z}_p)$, $Y \in \mathrm{Mat}_{r\times s}(\mathbb{Z}_p)$, $X \in \mathrm{Mat}_{r\times t}(\mathbb{Z}_p)$ y $Z \in \mathrm{Mat}_{s\times t}(\mathbb{Z}_p)$.

Teorema 1. *Dado $M \in \Theta$, consideramos las diferentes potencias de M.*

Tomando h como un entero no negativo se tiene que

$$M^h = \begin{bmatrix} A^h & Y^{(h)} & X^{(h)} \\ \boldsymbol{0} & B^h & Z^{(h)} \\ \boldsymbol{0} & \boldsymbol{0} & C^h \end{bmatrix}, \tag{2}$$

donde

$$Y^{(h)} = \begin{cases} \boldsymbol{0}, & \text{si } h = 0, \\ \displaystyle\sum_{i=1}^{h} A^{h-i} Y B^{i-1}, & \text{si } h \geq 1, \end{cases} \tag{3}$$

$$X^{(h)} = \begin{cases} \boldsymbol{0}, & \text{si } h = 0, \\ \displaystyle\sum_{i=1}^{h} (A^{h-i} X C^{i-1} + \Sigma_i), & \text{si } h \geq 1, \end{cases} \tag{4}$$

$$\text{con } \Sigma_i = \sum_{j=1}^{h-i} A^{h-i-j} Y B^{j-1} Z C^{i-1},$$

$$Z^{(h)} = \begin{cases} \boldsymbol{0}, & \text{si } h = 0, \\ \displaystyle\sum_{i=1}^{h} B^{h-i} Z C^{i-1}, & \text{si } h \geq 1. \end{cases} \tag{5}$$

Además, si $0 \leq q \leq h$, se tiene

$$Y^{(h)} = A^q Y^{(h-q)} + Y^{(q)} B^{h-q}, \tag{6}$$

$$X^{(h)} = A^q X^{(h-q)} + Y^{(q)} Z^{(h-q)} + X^{(q)} C^{h-q}, \tag{7}$$

$$Z^{(h)} = B^q Z^{(h-q)} + Z^{(q)} C^{h-q}. \tag{8}$$

Proof: Lo demostraremos usando inducción sobre h.

Para $h = 0$ y $h = 1$, el resultado es obvio.

Supondremos ciertas las expresiones 3), (4), (5) para $h-1$ y demostraremos que también lo son para h.

$$\begin{aligned} M^h &= M M^{h-1} \\ &= \begin{bmatrix} A & Y & X \\ \boldsymbol{0} & B & Z \\ \boldsymbol{0} & \boldsymbol{0} & C \end{bmatrix} \begin{bmatrix} A^{h-1} & Y^{(h-1)} & X^{(h-1)} \\ \boldsymbol{0} & B^{h-1} & Z^{(h-1)} \\ \boldsymbol{0} & \boldsymbol{0} & C^{h-1} \end{bmatrix} \\ &= \begin{bmatrix} A^h & Y^{(h)} & X^{(h)} \\ \boldsymbol{0} & B^h & Z^{(h)} \\ \boldsymbol{0} & \boldsymbol{0} & C^h \end{bmatrix}. \end{aligned}$$

Por hipótesis de inducción, aplicando (3), tenemos que

$$\begin{aligned} Y^{(h)} &= A Y^{(h-1)} + Y B^{h-1} \\ &= A \sum_{i=1}^{h-1} A^{h-i-1} Y B^{i-1} + Y B^{h-1} \\ &= \sum_{i=1}^{h-1} A^{h-i} Y B^{i-1} + A^0 Y B^{h-1} \\ &= \sum_{i=1}^{h} A^{h-i} Y B^{i-1}, \end{aligned}$$

de manera que la expresión (3) se verifica también para h.

Finalmente, para demostrar las expresiones (4) y (5), consideramos que

$$M = \begin{bmatrix} A & Y & X \\ \boldsymbol{0} & B & Z \\ \boldsymbol{0} & \boldsymbol{0} & C \end{bmatrix} = \begin{bmatrix} A_1 & X_1 \\ \boldsymbol{0} & C_1 \end{bmatrix}, \tag{9}$$

donde $A_1 = \begin{bmatrix} A & Y \\ \boldsymbol{0} & B \end{bmatrix}$, $X_1 = \begin{bmatrix} X \\ Z \end{bmatrix}$ y $C_1 = C$.

Basándonos en las expresiones para matrices triangulares superiores con 2×2 bloques (véanse [1], [2]) en las que para una matriz

$$M = \begin{bmatrix} A & X \\ O & B \end{bmatrix},$$

la potencia M^h viene dada por

$$M^h = \begin{bmatrix} A^h & X^{(h)} \\ O & B^h \end{bmatrix},$$

con

$$X^{(h)} = \begin{cases} \boldsymbol{0} & if \quad h = 0, \\ \displaystyle\sum_{i=1}^{h} A^{h-i} X B^{i-1} & if \quad h \geq 1; \end{cases}$$

tenemos, para la matriz M considerada en (9),

$$\begin{aligned} M^h &= M M^{h-1} \\ &= \begin{bmatrix} A_1 & X_1 \\ \boldsymbol{0} & C_1 \end{bmatrix} \begin{bmatrix} A_1^{h-1} & X_1^{(h-1)} \\ \boldsymbol{0} & C_1^{h-1} \end{bmatrix} \\ &= \begin{bmatrix} A_1^h & X_1^{(h)} \\ \boldsymbol{0} & C_1^h \end{bmatrix}, \end{aligned}$$

donde

$$\begin{aligned} X_1^{(h)} &= \sum_{i=1}^{h} A_1^{h-i} X_1 C_1^{i-1} \\ &= \sum_{i=1}^{h} \left[\begin{array}{cc} A^{h-i} & Y^{(h-i)} \\ \mathbf{0} & B^{h-i} \end{array} \right] \left[\begin{array}{c} X \\ Z \end{array} \right] C^{i-1} \\ &= \sum_{i=1}^{h} \left[\begin{array}{c} A^{h-i}X + Y^{(h-i)}Z \\ B^{h-i}Z \end{array} \right] C^{i-1} \\ &= \sum_{i=1}^{h} \left[\begin{array}{c} A^{h-i}XC^{i-1} + Y^{(h-i)}ZC^{i-1} \\ B^{h-i}ZC^{i-1} \end{array} \right] \\ &= \left[\begin{array}{c} X^{(h)} \\ Z^{(h)} \end{array} \right]. \end{aligned}$$

En consecuencia,

$$X^{(h)} = \sum_{i=1}^{h} (A^{h-i}XC^{i-1} + Y^{(h-i)}ZC^{i-1}),$$

y, como,

$$Y^{(h-i)} = \sum_{j=1}^{h-i} A^{h-i-j}YB^{j-1},$$

resulta que

$$X^{(h)} = \sum_{i=1}^{h} (A^{h-i}XC^{i-1} + \sum_{j=1}^{h-i} A^{h-i-j}YB^{j-1}ZC^{i-1}),$$

$$Z^{(h)} = \sum_{i=1}^{h} B^{h-i}ZC^{i-1},$$

y las expresiones (4) y (5) también son ciertas para h.

Si $0 \leq q \leq h$, tenemos

$$M^h = M^q M^{h-q} =$$

$$= \left[\begin{array}{ccc} A^q & Y^{(q)} & X^{(q)} \\ \mathbf{0} & B^q & Z^{(q)} \\ \mathbf{0} & \mathbf{0} & C^q \end{array} \right] \left[\begin{array}{ccc} A^{h-q} & Y^{(h-q)} & X^{(h-q)} \\ \mathbf{0} & B^{h-q} & Z^{(h-q)} \\ \mathbf{0} & \mathbf{0} & C^{h-q} \end{array} \right]$$

$$= \left[\begin{array}{ccc} A^h & Y_h^* & X_h^* \\ \mathbf{0} & B^h & Z_h^* \\ \mathbf{0} & \mathbf{0} & C^h \end{array} \right],$$

donde

$$\begin{aligned} Y_h^* &= A^q Y^{(h-q)} + Y^{(q)} B^{h-q}, \\ X_h^* &= A^q X^{(h-q)} + Y^{(q)} Z^{(h-q)} + X^{(q)} C^{h-q}, \\ Z_h^* &= B^q Z^{(h-q)} + Z^{(q)} C^{h-q}. \end{aligned}$$

Comparando este resultado con la expresión (2) obtenemos

$$\begin{aligned} Y^{(h)} &= A^q Y^{(h-q)} + Y^{(q)} B^{h-q}, \\ X^{(h)} &= A^q X^{(h-q)} + Y^{(q)} Z^{(h-q)} + X^{(q)} C^{h-q}, \\ Z^{(h)} &= B^q Z^{(h-q)} + Z^{(q)} C^{h-q}, \end{aligned}$$

que se corresponde con las expresiones (6), (7) y (8). ■

Como consecuencia, si a y b son dos enteros tales que $a + b \geq 0$, tenemos

$$\begin{aligned} Y^{(a+b)} &= A^a Y^{(b)} + Y^{(a)} B^b, \\ X^{(a+b)} &= A^a X^{(b)} + Y^{(a)} Z^{(b)} + X^{(a)} C^b, \\ Z^{(a+b)} &= B^a Z^{(b)} + Z^{(a)} C^b. \end{aligned}$$

En el caso $a = h - 1$, $b = 1$, tenemos

$$Y^{(h)} = A^{h-1}Y + Y^{(h-1)}B, \quad (10)$$

$$X^{(h)} = A^{h-1}X + Y^{(h-1)}Z + X^{(h-1)}C, \quad (11)$$

$$Z^{(h)} = B^{h-1}Z + Z^{(h-1)}C. \quad (12)$$

III. Orden de $M \in \Theta$

Como ya se ha comentado con anterioridad, en las aplicaciones de los generadores pseudoaleatorios a la seguridad necesitamos producir secuencias de grandes períodos. En el generador propuesto, se genera un bit por cada una de las potencias de la matriz inicial

$$M = \left[\begin{array}{ccc} A & Y & X \\ \boldsymbol{0} & B & Z \\ \boldsymbol{0} & \boldsymbol{0} & C \end{array} \right] \in \Theta,$$

por lo que es deseable que el orden de M sea lo más grande posible.

Vamos a analizar, a continuación, la forma de obtener ese orden, lo que se traducirá en períodos grandes para las secuencias generadas (no necesariamente deben coincidir el orden de M y el periodo de la secuencia).

Sea $f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1}x^{n-1} + x^n$ un polinomio mónico en $\mathbb{Z}_p[x]$, cuya matriz $n \times n$ asociada es

$$\overline{A} = \left[\begin{array}{cccccc} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-2} & -a_{n-1} \end{array} \right];$$

si f es un polinomio irreducible en $\mathbb{Z}_p[x]$, entonces el orden de la matriz $\overline{A}$ es igual al orden de una raíz cualquiera de f en $\mathbb{F}_{p^n}$ y el orden de $\overline{A}$ divide $p^n - 1$ (véase [15]). Además, en el caso de que f sea un polinomio primitivo en $\mathbb{Z}_p[x]$, el orden de $\overline{A}$ es exactamente $p^n - 1$.

Odoni, Varadharajan y Sanders [18] proponen un esquema extendido, basado en la construcción de matrices por bloques de la forma

$$\overline{A} = \left[\begin{array}{cccc} \overline{A}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \overline{A}_2 & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \overline{A}_k \end{array} \right],$$

donde $\overline{A}_i$ es la matriz asociada a f_i, siendo f_i polinomios primitivos diferentes en $\mathbb{Z}_p[x]$ de grado n_i, para $i = 1, 2, \ldots, k$. El orden de $\overline{A}_i$, $i = 1, 2, \ldots, k$, es $p^{n_i} - 1$ y, por tanto, el orden de $\overline{A}$ es $mcm(p^{n_1} - 1, p^{n_2} - 1, \ldots, p^{n_k} - 1)$. Con el objetivo de

p	r	s	t	$o(M)$	p	r	s	t	$o(M)$
3	21	22	23	31	19	11	12	13	44
	33	34	35	49		25	26	27	98
5	20	21	22	43	29	19	21	22	88
	31	32	33	66		23	24	25	103
7	14	15	17	38	229	5	6	7	38
	30	31	32	77		25	26	27	180
11	13	14	15	42	251	7	8	9	53
	23	25	27	77		27	28	29	197
13	15	17	19	55	257	8	9	10	68
	25	26	27	85		30	31	32	217

Tabla I
ORDEN DE M EN FUNCIÓN DE p Y DEL TAMAÑO DE LOS BLOQUES

usar este tipo de matrices en criptosistemas de clave pública, los citados autores, conjugan la matriz $\overline{A}$ con una matriz invertible P de tamaño $n \times n$, con $n = n_1 + n_2 + \ldots + n_k$, obteniendo una nueva matriz $A = P\overline{A}P^{-1}$ que tiene el mismo orden que $\overline{A}$.

En nuestro caso, si construimos los bloques A, B y C de

$$M = \begin{bmatrix} A & Y & X \\ \boldsymbol{0} & B & Z \\ \boldsymbol{0} & \boldsymbol{0} & C \end{bmatrix} \in \Theta,$$

usando polinomios primitivos, podemos garantizar un orden muy alto.

Sean

$$\begin{aligned} f(x) &= a_0 + a_1 x + a_2 x^2 + \ldots + a_{r-1} x^{r-1} + x^r, \\ g(x) &= b_0 + b_1 x + b_2 x^2 + \ldots + b_{s-1} x^{s-1} + x^s, \\ h(x) &= c_0 + c_1 x + c_2 x^2 + \ldots + c_{t-1} x^{t-1} + x^t, \end{aligned}$$

tres polinomios primitivos en $\mathbb{Z}_p[x]$ y $\overline{A}, \overline{B}$ y $\overline{C}$ las correspondientes matrices asociadas. Dadas P, Q y R, tres matrices invertibles, definimos $A = P\overline{A}P^{-1}$, $B = Q\overline{B}Q^{-1}$ y $C = R\overline{C}R^{-1}$. Con esta construcción el orden de la matriz M es $mcm(p^r - 1, p^s - 1, p^t - 1)$ (véanse [15] y [18]).

En la tabla I mostramos el orden de M dependiendo de p, r, s y t, donde r, s y t son los tamaños de los bloques A, B y C, respectivamente. Como podemos observar, es posible alcanzar grandes ordenes de M con pequeños valores de p y valores no muy grandes de r, s y t. En la columna $o(M)$ el orden de M está expresado en número de cifras decimales. Como referencia, indicaremos que el entero 2^{128} necesita 39 cifras decimales para ser expresado.

IV. DESCRIPCIÓN DEL GENERADOR

Como se ha indicado con anterioridad, en esta sección describiremos un nuevo generador pseudoleatorio basado en potencias de matrices triangulares superiores con 3×3 bloques definidas en $\mathbb{Z}_p$, con p primo; extensión de otro desarrollado previamente por algunos de los autores [1] de este artículo.

Consideramos la matriz

$$M = \begin{bmatrix} A & Y & X \\ \boldsymbol{0} & B & Z \\ \boldsymbol{0} & \boldsymbol{0} & C \end{bmatrix},$$

con las características definidas en (1).

Para generar la secuencia pseudoaleatoria de bits, se fijan las matrices A, B y C, y se eligen de forma aleatoria las matrices Y, X, Z; que constituyen la semilla de la secuencia. A continuación se aplican las expresiones (10), (11) y (12) para obtener las secuencias de matrices

$$\begin{aligned} &Y^{(2)},\ Y^{(3)},\ Y^{(4)},\ \ldots \\ &X^{(2)},\ X^{(3)},\ X^{(4)},\ \ldots \\ &Z^{(2)},\ Z^{(3)},\ Z^{(4)},\ \ldots \end{aligned}$$

Notemos que, para el cálculo de $Y^{(h)}$, $X^{(h)}$ y $Z^{(h)}$ sólo se necesita almacenar A^{h-1}, B^{h-1}, $Y^{(h-1)}$, $X^{(h-1)}$, $Z^{(h-1)}$ y los valores iniciales A, B, C, Y, X, Z; no siendo necesario calcular las sucesivas potencias de C. Este hecho, además de mejorar la eficiencia en el cálculo, puede influenciar en la elección del tamaño de los bloques A, B y C que, por otra parte, determinan el tamaño de la semilla.

Para cada iteración $h = 2, 3, \ldots$ se extrae un bit, obtenido como suma $\oplus$ de los bits menos significativos de cada uno de los elementos de $Y^{(h)}$, $X^{(h)}$ y $Z^{(h)}$; de esta forma, se obtiene la secuencia de bits

$$b^{(2)},\ b^{(3)},\ b^{(4)},\ \ldots$$

A medida que se va generando esta secuencia, se le aplica un filtrado que elimina el sesgo de la misma, mediante el siguiente proceso:

$$\begin{aligned} c^{(1)} &= 0, \\ c^{(h)} &= b^{(h)} \oplus c^{(h-1)},\ h = 2, 3, \ldots \end{aligned}$$

obteniendo la secuencia de bits

$$c^{(2)},\ c^{(3)},\ c^{(4)},\ \ldots$$

Notemos que cada bit de esta secuencia depende de la semilla inicial Y, X, Z. La forma de obtener períodos grandes para la misma consiste en utilizar para M una construcción como la descrita en la parte final de la sección III.

La autocorrelación obtenida en el análisis empírico de las secuencias generadas (sección V) indica la no presencia de subperiodos, en concordancia con el hecho de que si el orden de M vale w se tiene que $M^w = I_{r+s+t}$ y, por tanto, $A^w = I_r$, $B^w = I_s$, $C^w = I_t$, $Y^{(w)} = \boldsymbol{0}_{r \times s}$, $X^{(w)} = \boldsymbol{0}_{r \times t}$, $Z^{(w)} = \boldsymbol{0}_{s \times t}$. Si para algún valor $w_0 < w$ se anulasen los bloques superdiagonales ($Y^{(w_0)} = \boldsymbol{0}_{r \times s}$, $X^{(w_0)} = \boldsymbol{0}_{r \times t}$, $Z^{(w_0)} = \boldsymbol{0}_{s \times t}$), M^{w_0+1} no tendría por que ser igual a M ya que los bloques diagonales de M^{w_0} (A^{w_0}, B^{w_0} y C^{w_0}) no pueden ser al tiempo la identidad, por ir en contra de que el orden sea w; y, consecuentemente, el bit (o los bits) generado a partir de M^{w_0+1} no tiene por que coincidir con el generado a partir de M. De igual manera, el bit (o los bits) generado a partir de M^{w_0+2} no tiene por que coincidir con el generado a partir de M^2 y así sucesivamente. Aún cuando no se hayan detectado subperiodos en las secuencias analizadas, no se puede asegurar que el periodo de la secuencia generada coincida con el orden de M; notemos, observando las expresiones (10), (11) y (12), que eligiendo los bloques A

y B de manera que sus ordenes sean primos entre sí se puede garantizar, al menos, un periodo igual al producto de ambos ordenes. El estudio de la relación entre los ordenes de los bloques y del periodo de la secuencia así como la probabilidad de que se anulen los tres bloques superdiagonales (claves débiles, semidébiles, ...) no ha estado entre los objetivos de este trabajo en el que se presenta una primera versión del generador con resultados prometedores.

V. Resultados

El generador ha sido analizado con cinco estadísticas diferentes (tests de frecuencia y autocorrelación, véanse [4], [17] para más información) y con el cálculo de la complejidad lineal de la secuencia mediante el algoritmo de Berlekamp-Massey [3], [16].

La complejidad lineal esperada para una secuencia aleatoria de tamaño n es $\frac{n}{2}$. Para el resto de los tests, los resultados se comparan con los valores umbrales que se dan en la tabla II, pasando el test aquellos por debajo del umbral. El nivel de significancia, α, es la probabilidad de que una secuencia falle el test incluso cuando debería haberlo pasado; de esta forma cuanto mayor es el nivel de significancia más restrictivo se es con los resultados.

Durante la extensa experimentación realizada, se han considerado diferentes valores para p, r, s y t, combinados con un gran número de secuencias de distintas longitudes; obteniendo en todos los casos resultados muy positivos para los tests mencionados en la parte inicial de esta esta sección. A modo de ejemplo, en la tabla III se reflejan los resultados obtenidos para $p = 229$, $r = 5$, $s = 6$ y $t = 7$. Los polinomios primitivos empleados para obtener las matrices A, B, C, tal y como se describe en la sección IV, son $f(x) = x^5 + 226x^2 + 152$, $g(x) = x^6 + 131x^5 + 28$ y $h(x) = x^7 + 122x^6 + 6$, respectivamente. Para P, Q y R se han utilizado matrices identidad, de forma que A, B y C sean, directamente, las matrices asociadas a estos polinomios.

Las semillas tomadas son matrices Y, X, Z con tamaños 5×6, 5×7, 6×7, respectivamente; obtenidas usando un generador aleatorio (http://www.random.org/). Se ha realizado cada test con gran número de secuencias de longitudes $2 \cdot 10^3$, $2 \cdot 10^4$, $2 \cdot 10^5$ y $2 \cdot 10^6$ bits.

Como se puede apreciar en la tabla III, los resultados obtenidos son particularmente buenos, mejores que los valores umbrales para el nivel de significancia más exigente ($\alpha = 0{,}1$) de la tabla II. Habría que destacar que en los resultados de los tests realizados sobre las distintas secuencias hay uniformidad, en el sentido de que no hay valores extraños y que todos superan los valores umbrales. La regularidad en los resultados es independiente de las semillas escogidas y de la longitud de las secuencias generadas, como se puede apreciar en la tabla III, no apreciándose ninguna relación directa entre la longitud de la secuencia y los valores obtenidos de los tests.

Los tests de complejidad lineal también ofrecen muy buenos resultados, obteniendo valores próximos a $\frac{n}{2}$ en todos los casos. Esto significa que la secuencia generada presenta la misma no linealidad que una secuencia realmente aleatoria, evitando que sea predecible a causa de una alta linealidad.

Los resultados estadísticos obtenidos para esta primera versión del generador pseudoaleatorio son prometedores, por lo que cabe estudiar otros esquemas de extracción de bits que mejoren la eficiencia, así como otros filtrados que mejoren la seguridad y otras pruebas estadísticas más exigentes como el sistema TESTU01 de L'Ecuyer y Simard [6].

VI. Conclusiones

Se ha presentado un nuevo generador pseudoaleatorio, con aplicaciones criptográficas, basado en potencias de matrices triangulares superiores con 3×3 bloques definidas en $\mathbb{Z}_p$, con p primo; extensión de otro desarrollado previamente por algunos de los autores [1] de este artículo. Es de destacar que, usando polinomios primitivos para generar los bloques diagonales y tomando primos y tamaños de matrices muy pequeños, podemos producir secuencias de períodos muy grandes, buenas propiedades estadísticas y alta complejidad lineal; de esta forma, la secuencia se genera de forma muy eficiente ya que no requiere el coste computacional que sería necesario con primos grandes o matrices enormes. Incluso con tamaños tan pequeños, el algoritmo asegura un espacio de claves muy grande hecho que; unido a que la semilla participe en la generación de cada bit, al gran periodo y a los resultados estadísticos obtenidos; hace suponer un buen nivel de seguridad para esta primera versión del generador.

Es de destacar la flexibilidad del algoritmo que permite, si se tiene en cuenta el tamaño de las matrices y el número de bits que se puedan tomar por iteración[1], adaptarse de forma precisa para cada requerimiento de software y hardware; pudiendo utilizar muy poca memoria o conseguir velocidades muy altas.

Agradecimientos

Este trabajo ha sido realizado bajo el marco de los proyectos GRE09-02 y GRE10-34 de la Universidad de Alicante y GV/2011/001 y GV/2012/111 de la Generalitat Valenciana.

Referencias

[1] R. Álvarez, J.J. Climent, L. Tortosa, A. Zamora. "An efficient binary sequence generator with cryptographic applications", en Applied Mathematics and Computation, vol. 167, pp. 16–27, 2005.

[2] R. Alvarez, F. Ferrández, J.F. Vicent, A. Zamora. "Applying quick exponentiation for block upper triangular matrices", en Applied Mathematics and Computation, vol. 183, pp. 729–737, 2006.

[3] E. R. Berlekamp. "Algebraic Coding Theory", en McGraw Hill, New York, 1968.

[4] H. Beker, F. Piper. "Cipher Systems: The Protection of Communications", en John Wiley and Sons, New York, 1982.

[5] L. Blum, M. Blum, M. Shub. "A Simple Unpredictable Pseudorandom Number Generator" en SIAM J. Comput., vol. 15, pp. 364–383, 1986.

[6] P. L'Ecuyer, R. Simard."TestU01: A C Library for Empirical Testing of Random Number Generators" en ACM Transactions on Mathematical Software, vol. 33, artículo 22, 2007.

[7] A. Fuster, J. García. "An Efficient Algorithm to Generate Binary Sequences for Cryptographic Purposes" en Theoretical Computer Science, vol. 259, pp. 679–688, 2001.

[1]Se pueden utilizar otros esquemas de extracción de bits además del propuesto inicialmente.

Tabla II
VALORES UMBRALES PARA LOS TESTS ESTADÍSTICOS

α	Monobit	Serial	Poker	Rachas				Autocorrelación
				$2 \cdot 10^3$	$2 \cdot 10^4$	$2 \cdot 10^5$	$2 \cdot 10^6$	
0.001	10.830	13.820	330.5	29.59	39.25	51.18	59.70	3.090
0.005	7.870	10.590	316.9	25.19	34.27	45.56	53.67	2.576
0.010	6.635	9.210	310.5	23.21	32.00	42.98	50.89	2.326
0.025	5.024	7.378	301.1	20.48	28.85	39.36	46.98	1.960
0.050	3.842	5.992	293.2	18.31	26.30	36.42	43.77	1.645
0.100	2.706	4.605	284.3	15.99	23.54	33.20	40.26	1.282

Tabla III
RESULTADOS DE LOS TESTS ESTADÍSTICOS PARA $p = 229$, $r = 5$, $s = 6$ Y $t = 7$

Longitud	Monobit	Serial	Poker	Rachas	Autocorrelación	Complejidad Lineal
$2 \cdot 10^3$	1.057	2.542	244.0	11.02	0.811	1000
$2 \cdot 10^4$	0.386	0.889	240.3	14.19	0.793	10001
$2 \cdot 10^5$	1.588	2,444	245.5	27.72	0.798	100001
$2 \cdot 10^6$	1.732	3,087	249.4	32.34	0.798	1000001

[8] J. García, M. Rodríguez. "A Family of Keystream Generators with Large Linear Complexity" en Applied Mathemathics Letters, vol 14, pp. 545–547

[9] C.M. González, H.A. Larrondo, O.A. Rosso. "Statistical complexity measure of pseudorandom bit generators" en Physica A., vol 354, pp. 281–300, 2001.

[10] K. Hosseini, M.P. Kennedy. "Architectures for maximum sequence length digital delta-sigma modulators" en IEEE Trans. Circuits Syst. II, Exp. Briefs, vol 55(11), pp. 1104–1108, 2008.

[11] A. Kanso, N. Smaoui. "Logistic chaotic maps for binary numbers generations" en Chaos, Solitons and Fractals, vol. 40, pp. 2557–2568, 2009.

[12] J. Kelsey, B. Schneier, B., N. Ferguson. "Yarrow-160: Notes on the Design and Analysis of the Yarrow Cryptographic Pseudorandom Number Generator" en Lecture Notes in Computer Science, vol. 1758, pp. 13–33, 2000.

[13] S. Kwok, E. Lam. "Effective uses of FPGAs for Brute-Force Attack on RC4 Ciphers" en IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 16, pp. 1096–1100, 2008.

[14] W. Lempken, S.S. Magliveras, T. van Trung, W. Wei. "A public key cryptosystem based on non-abelian finite groups" en Journal of Cryptology, vol. 22, pp. 62–74, 2009.

[15] R. Lidl, H. Niederreiter. "Introduction to Finite Fields and their Applications" en Cambridge University Press, Cambridge, 1994.

[16] J. L. Massey- "Shift-Register Synthesis and BCH Decoding" en IEEE Transactions on Information Theory, vol. 15, pp. 122–127, 1969.

[17] A. Menezes, P. van Oorschot, S. Vanstone. "Handbook of Applied Cryptography" en CRC Press, Florida, 2001.

[18] R. W. K. Odoni, V. Varadharajan, P.W. Sanders. "Public Key Distribution in Matrix Rings" en Electronic Letters, vol. 20, pp. 386–387, 1984.

[19] F. Pareschi, R. Rovatti, G. Setti. "A fast chaos-based true random number generator for cryptographic applications" en Proceedings of 26th Eur. Solid-State Circuit Conf., Montreux, Switzerland, pp. 130–133, 2006.

[20] F. Pareschi, R. Rovatti, g. Setti. "Second-level NIST randomness tests for improving test reliability" en Proceedings of IEEE ISCAS, pp. 1437–1440, 2007.

[21] W. Stallings. "Cryptography and Network Security: Principles and Practice. Third Edition" en Prentice Hall, New Jersey, 2003.

[22] Statistical test suite for random and pseudorandom number generators for cryptographic applications. National Institute of Standards and Technology (NIST) Special Publication 800-22, Revised April 2010, National Institute of Standards and Technology, 2010.

[23] D.B. Thomas, L. Howes, W. Luk. "A comparison of CPUs, GPUs, FPGAs, and massively parallel processor arrays for random number generation" en Proceedings of ACM/SIGDA Int. Symp. Field Programmable Gate Arrays, pp. 63–72, 2009.

[24] T. Xiang, K. Benkrid. "Mersenne Twister random number generation on FPGA, CPU and GPU" en Proceedings of NASA/ESA Conf. AHS, pp. 460–464, 2009.

[25] Accredited Standards Committee, X9 - Financial Services: American National Standard, Financial Institution Key Management. X9 - Secretariat, American Bankers Association, ANSI X9.17, 1995.