Ciencias Matemáticas, Vol. 32, No. 1, Pag. 61-69, 2018
Recibido 03-2018

Optimización del test LIL-débil para la evaluación de PRGN

Y. Matos[1], C. M. Legón[1,2], E. J. Madarro[3]

Resumen
La evaluación de aleatoriedad en generadores de números pseudoaleatorios (PRGN) es de gran importancia en criptografía, existen numerosos tests de comprobación de aleatoriedad, muchos de ellos han sido diseñados basados en leyes importantes de aleatoriedad como el *Teorema Central del Límite* y la *Ley de los Grandes Números*. Recientemente fue propuesto por Wang un novedoso test de tres variantes para la evaluación de generadores de números pseudoaleatorios, basado en la *Ley del Logaritmo Iterado* (LIL). Las probabilidades teóricas que deben calcularse para aplicar el test poseen una dependencia entre ellas, por lo cual se deben calcular secuencialmente. En este trabajo se propone una nueva expresión, más simple que las expresiones dadas por Wang para el cálculo de estas probabilidades teóricas. La nueva expresión elimina la dependencia entre estas probabilidades y permite la optimización de las implementaciones del cálculo de las mismas utilizando los algoritmos secuencial y paralelos dados aquí, lo cual facilita la realización de experimentaciones más amplias.

Palabras claves
Evaluación de PRGN — Test de aleatoriedad — Optimización del test LIL-débil

Abstract
The randomness evaluation in pseudorandom generators numbers (PRGN) is of great importance in cryptography, there are numerous tests of randomness testing, many of them have been designed based on important laws of randomness such as the *Central Limit Theorem* and the *Law of Large Numbers*. Recently a new three variant test for the evaluation of pseudorandom generators number was proposed by Wang based on the *Iterated Logarithm Law* (LIL) . The theoretical probabilities that must be calculated to apply the test have a realization of dependence between them so they must be calculated sequentially. In this paper we propose a new simpler expression of the expressions given by Wang for the calculation of these theoretical probabilities. The new expression eliminates the dependence between these probabilities and allows the optimization of computational implementations using the algorithms sequential and parallel given here which facilitates the realization of larger experiments.

Keywords
PRGN evaluation — Randomness test — LIL-weak test optimization

[1] *Universidad de la Habana, Facultad de Matemática y Computación, Instituto de Criptografía, La Habana, Cuba*
[2] *CUJAE, Facultad de Informática, La Habana, Cuba [clegon@ceis.cujae.edu.cu]*
[3] *Universidad Central de las Villas,Villa Clara, Cuba*

1. Introducción

Los números aleatorios siempre han tenido una gran aplicación y utilidad en distintas ramas de la ciencia, como en la estadística, la ciencia de la computación, la modelación, la simulación, la criptografía y otras aplicaciones [34]. Los números aleatorios pueden ser generados desde un dispositivo físico, llamado generador de números aleatorios (RNG) y el mismo puede por ejemplo utilizar ruido térmico proveniente de diodos electrónicos, pero con más frecuencia son generados desde algún programa de computadora llamado generador de números pseudoaleatorios (PRNG), el cual dado un valor inicial llamado semilla produce una secuencia de números determinísticos que deben tener una distribución estadística correspondiente a una variable aleatoria independiente e igualmente distribuida. De aquí que los PRNG tienen una salida periódica y determinística, por lo que está claro a priori que ellos no producen variables aleatorias independientes en el sentido matemático y que ellos no pueden pasar todos los posibles tests estadísticos de uniformidad e independencia. Pero algunos de ellos tienen períodos de longitud larga y los fragmentos de salidas de longitud menor que el período pueden tener un comportamiento similar a la aleatoriedad, por lo cual se usa en ocasiones el término de "aleatoriedad local", estas salidas pueden presentar un comportamiento bastante bueno al aplicar tests estadísticos en un tiempo razonable. Sin embargo, numerosos PRNG populares en software comercial, fallan muchos de los tests simples (ver [11, 34]).

El diseño de un PRNG es una tarea de alta complejidad,

pues se debe de garantizar que se cumplan las mayorías de las propiedades exigidas para una sucesión verdaderamente aleatoria. Este diseño consta de dos fases, la fase de fundamentación teórica y diseño e implementación del generador y la fase de evaluación mediante la aplicación de diversos tests de seguridad (ver [10, 12]). La fase dos es de gran importancia para la detección de posibles vulnerabilidades en la seguridad de los sistemas criptográficos, por ejemplo en algunos artículos (ver [5, 28, 32]) muestran que se han incluido puertas traseras en los generadores de números pseudoaleatorios incluyendo los del NIST SP800-90A para conseguir capacidad de criptoanálisis, incluso en línea. Los tests estadísticos son también requeridos para RNG basados (parcial o totalmente) en dispositivos físicos. Los RNG son usados para generar llaves en criptosistemas y para otras aplicaciones ([34]).

Por un largo tiempo se estuvieron aplicando como estándar para PRNG los tests descritos en las ediciones del libro de Knuth (ej. en [10]). Otros tests más potentes habían sido propuestos para detectar regularidades en generadores lineales por Marsaglia en 1985 y 1996 y por Marsaglia y Tsang en el 2002 (ver [19, 20, 21]). Algunos de estos tests y otros nuevos fueron estudiados más extensivamente por Erdmann en 1992 , Marsaglia y Zaman en 1993, Vattulainen en 1995, L'Ecuyer y Simard en 1999, 2000, 2001, 2002, Rukhin en 2001 (ver [6, 22, 31, 13, 14, 15, 17, 25]). Por ejemplo los tests clásicos de Knuth [1997] más otros pocos están disponibles en SPRNG [23], y la mayoría de los que se han mencionados hasta aquí están disponibles en TestU01 (ver [16]). Un estado del arte más actualizado de los tests de aleatoriedad se puede ver en [4], donde se expone el problema de múltiples tests y para revisar más sobre este problema en la actualidad, se puede consultar [29, 30].

Algunos de los paquetes de pruebas estadísticas más conocidos en el dominio público son TestU01, el paquete de pruebas estadísticas implementado por Nacional Institute of Standard and Technology NIST SP800-22 ([26]) y DIEHARD, (ver [20, 26]). La librería TetsU01 provee un conjunto de pruebas estadísticas para PRNG, es flexible, las implementaciones son eficientes y puede trabajar con tamaño de muestras largas y tiene un amplio rango de tests paramétricos mayor que cualquier otra librería. El paquete del NIST SP80022 contiene 15 tests, orientados ante todo para la evaluación y certificación de RNG y PRNG usados en aplicaciones criptográficas [26] y el mismo es ampliamente utilizado. DHIEHARD contiene numerosos tests estadísticos pero tiene inconvenientes y limitaciones. La selección de los tests así como los parámetros de estos (tamaño de muestras, etc.) son fijados en el paquete. Además los tamaños de muestras no son muy largos y el paquete completo de tests corre en unos pocos segundos de tiempo de CPU en una computadora estándar de escritorio. El resultado de esto, es que ellos no son muy exigentes y el usuario tiene poca flexibilidad para realizar experimentaciones con cambios de parámetros [16].

Wang en [18, 33, 34] señala que el NIST800-22 no cubre algunas de las importantes leyes de aleatoriedad. Existen dos teoremas fundamentales de límites sobre sucesiones binarias aleatorias que son el *Teorema Central del Límite* y la *Ley del Logaritmo Iterado*. Numerosos tests del NIST SP800-22 cubren el *Teorema Central del Límite* mientras que ninguno de ellos tiene en cuenta la *Ley del Logaritmo Iterado*, ni tampoco las de las otras baterías mencionadas anteriormente. Actualmente existe un interés en encontrar nuevos métodos para comprobar la *Ley del Logaritmo Iterado* en sucesiones aleatorias y para otras aplicaciones, como se puede apreciar en [1, 2, 8, 24, 27, 37].

Wang propone técnicas de pruebas estadísticas basadas en LIL y pruebas de distancias estadísticas en generadores pseudoaleatorios para reducir el error de Tipo II presente en el paquete de pruebas del NIST SP800-22. Además realiza tres diseños del test LIL para la evaluación de generadores de números pseudoaleatorios: débil, débil II y fuerte, de los cuales solo aplica la variante débil (ver [18, 33, 34, 35]). En las evaluaciones de los PRNG se toman 1000 y 10000 sucesiones de 2^{34} bits (2GB) y se conforman conjuntos de sucesiones de 2^{26} a 2^{34} bits, lo cual podría ser una limitación (desde el punto de vista de disponibilidad de recursos) para poder aplicar la prueba.

Este artículo, propone una forma más simplificada que la dada por Wang en [33] para calcular las probabilidades teóricas utilizadas en el test LIL-débil. Esta simplificación permite mejores implementaciones del test en cuanto a eficiencia y posibilidad de paralelización porque se obtuvo una independencia en el cálculo estas probabilidades lo cual resulta ventajoso para implementaciones más eficientes del test.

Este artículo se organiza como sigue. En la sección 2 se describe la *Ley del Logaritmo Iterado* y se da la aproximación normal para S_{lil}. En la Sección 3 se presenta el test LLL-débil dado por Wang en [33] y algunas consideraciones de importancia así como los pasos para la evaluación de generadores pseudoaleatorios dada por Wang, donde intervienen el cálculo de las probabilidades teóricas y que ahora pueden ser calculadas con los resultados de este trabajo. En la sección 4 se presentan dos corolarios que optimizan el cálculo de las probabilidades teóricas, y tres algoritmos para su implementación más eficiente. En la sección 5 se exponen los resultados experimentales asociados a los mismos. Finalmente en la sección 6 se realizan las conclusiones con algunos comentarios y observaciones.

2. Ley del logaritmo iterado

La Ley del Logaritmo Iterado , fue primeramente descubierta por Khintchine (ver [9]) que da una cota óptima superior $\sqrt{2\ln(\ln(n))}$ para las fluctuaciones de un camino aleatorio. Esta ley es también descrita por Feller en [7], y por Wang en [36] que muestra que la ley del logaritmo iterado también se mantiene para "sucesiones p-aleatorias" (aleatorias en tiempo polinomial).

La *Ley del Logaritmo Iterado* describe las escalas de fluctuaciones de $S^*(\xi[0,\cdots,n-1])$. Para una cadena no vacía $\delta \in \sum^*$, sea

$$S(\delta)=\sum_{i=0}^{|\delta|-1}\delta[i] \text{ y } S^*(\delta)=\frac{2\cdot S(\delta)-|\delta|}{\sqrt{|\delta|}}$$

donde $S(\delta)$ denota el número de unos en δ y $S^*(\delta)$ denota el número reducido de unos en δ. $S^*(\delta)$ mide las desviaciones $S(\delta)$ de $\frac{|\delta|}{2}$ en unidades de $\frac{1}{2}\sqrt{|\delta|}$.

La *Ley de los Grandes Números* establece que, para una sucesión aleatoria ξ, el límite de $\frac{S(\xi[0,\cdots,n-1])}{n}$ es $\frac{1}{2}$, lo cual se corresponde con el test de frecuencia (Monobit) del NIST SP800-22 [26]. Pero no se dice nada acerca de la desviación reducida $S^*(\xi[0,\cdots,n-1])$. Es intuitivamente claro que para una sucesión pseudoaleatoria ξ, $S^*(\xi[0,\cdots,n-1])$ podría tomar valores arbitrarios grandes (a pesar de la lentitud).

Teorema 2.1 *(Wang [33])Para una secesión $\xi\in\sum^{\infty}$, sea*

$$S_{lil}(\xi|n)=\frac{2\sum_{i=0}^{n-1}\xi[i]-n}{\sqrt{2\ln(\ln(n))}}, \quad (1)$$

entonces se tiene que

$$\lim_{n\to\infty}\sup S_{lil}(\xi|n)=1 \text{ y } \lim_{n\to\infty}\inf S_{lil}(\xi|n)=-1,$$

para cada sucesión p-aleatoria $\xi\in\sum^{\infty}$.

2.1 Aproximación normal para S_{lil}

En esta sección se da la aproximación normal de la función $S_{lil}(\cdot)$ que se usará en lo adelante. El teorema de Moivre-Laplace da una aproximación normal para la distribución binomial, la cual establece que el número de "éxitos" de n lanzamientos independientes de una moneda con probabilidad de $1/2$ es aproximadamente una distribución normal con media $n/2$ y desviación estándar $\sqrt{n}/2$.

Definición 1 *La función de densidad normal con media μ y varianza σ es definida como*

$$f(\delta)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(\delta-\mu)^2}{2\sigma^2}} \quad (2)$$

Para $\mu=0$ y $\sigma=1$, se tiene a la función de densidad normal estándar

$$\varphi(\delta)=\frac{1}{\sqrt{2\pi}}e^{-\frac{\delta^2}{2}} \quad (3)$$

su integral

$$\Phi=\int_{-\infty}^{\delta}\varphi(\lambda)d\lambda \quad (4)$$

es la función de distribución normal estándar.

Teorema 2.2 *(Wang [33]) Para δ_1,δ_2 fijos, se tiene que*

$$\lim_{n\to\infty}Prob[\delta_1\le S^*(\xi|n)\le\delta_2]=\Phi(\delta_2)-\Phi(\delta_1). \quad (5)$$

El crecimiento de la velocidad para la aproximación de arriba está delimitada por $\max k^2/n^2, k^4/n^3$ donde $k=S(\xi|n)-n/2$.

El siguiente lema es útil para la interpretación de S^* basado en resultados de aproximaciones en términos de S_{lil}. Esto es obtenido por el hecho de que $\sqrt{2\ln(\ln(n))}\cdot S_{lil}(\xi|n)=S^*(\xi|n)$.

Lema 1 *(Wang [33]) Para cualquier δ_1,δ_2, se tiene que*

$$Prob[\delta_1<S_{lil}(\xi|n)<\delta_2]=$$
$$Prob[\delta_1\sqrt{2\ln(\ln(n))}<S^*(\xi|n)<\delta_2\sqrt{2\ln(\ln(n))}].$$

3. Descripción del test LIL-débil

El teorema 2.1 muestra que una sucesión pseudoaleatoria debería satisfacer la ley del logaritmo iterado (LIL). En [33] se propone el siguiente test:

Test LIL-débil. Sea $\alpha\in(0,0.25]$ y $\aleph\subset N$ un subconjunto de números naturales, se dice que una sucesión ξ no pasa el test $(\alpha,\aleph)$-LIL débil si $-1+\alpha<S_{lil}(\xi|n)<1-\alpha, \forall n\in\aleph$.

Por definición, una sucesión ξ pasa el test $(\alpha,\aleph)$-LIL débil si S_{lil} está entre $1-\alpha$ o $-1+\alpha$ de algunos puntos en $\aleph$. En la práctica, es importante hacer una selección apropiada para el test de puntos del conjunto $\aleph$ y calcular la probabilidad para que una sucesión ξ pase el test $(\alpha,\aleph)$-LIL débil. En esta sección se presenta el cálculo de las probabilidades para que una sucesión pase el test $(\alpha,\aleph)$-LIL débil con la siguiente selección de $\aleph$:

$$\aleph_0=\{2^0n_1\},\cdots,\aleph_t=\{2^tn_1\}, \text{ y } \bigcup\aleph_i$$

dados n_1 y t. Específicamente se podría considerar los casos para $t=8$ y $n_1=2^{26}$.

Donde las probabilidades $P_{(\alpha,\aleph_i)}$ de que una sucesión aleatoria pasa el test $(\alpha,\aleph_i)-$LIL débil viene dada por la siguiente expresión que se obtiene a partir del Teorema 2.2 y del Lema 1.

$$Prob[|S_{lil}(\xi|n)|\ge\theta]\simeq 2(1-\Phi(\theta\sqrt{2\ln(\ln(n))})). \quad (6)$$

Y para las probabilidades $P_{(\alpha,\aleph)}$ de que una sucesión aleatoria pase el test $(\alpha,\aleph)$-LIL débil con $\aleph$ como la unión de dos $\aleph_i$, se da el siguiente teorema:

Teorema 3.1 *(Wang [33]) Para $0<\alpha<1$ fijo y $t\ge 2$, sea $\theta=1-\alpha$, $\aleph=\{n,tn\}$, $\aleph_a=\{n\}$, $\aleph_b=\{tn\}$. Se tiene que*

$$P_{(\alpha,\aleph)}\simeq P_{(\alpha,\aleph_a)}$$
$$+\int_{-\theta\sqrt{2\ln(\ln(n))}}^{\theta\sqrt{2\ln(\ln(n))}}\frac{1}{\pi}\int_{\sqrt{\frac{1}{t-1}}(\theta\sqrt{2t\ln(\ln(tn))}-\lambda)}^{\infty}e^{-\frac{\delta^2+\lambda^2}{2}}d\delta d\lambda \quad (7)$$

Alternativamente se tiene que:

$$P_{(\alpha,\aleph)}\simeq P_{(\alpha,\aleph_a)}+P_{(\alpha,\aleph_b)}$$

$$-\int_{\theta\sqrt{2\ln(\ln(n))}}^{\infty}\frac{1}{\pi}\int_{\sqrt{\frac{1}{t-1}}(\theta\sqrt{2t\ln(\ln(tn))}-\lambda)}^{\infty}e^{-\frac{\delta^2+\lambda^2}{2}}d\delta d\lambda \quad (8)$$

En [33] se propone el teorema anterior para el cálculo de $P_{(\alpha,\aleph)}$, sin embargo, no se dá un algoritmo para la obtención de estas probabilidades.

El siguiente algoritmo se deduce de la expresión 6 y el teorema 3.1 para el cómputo de $P_{(\alpha,\aleph)}$.

Algoritmo 1: Cálculo de $P_{(\alpha,\aleph_a)}, P_{(\alpha,\aleph)}$.

Data: n, r, α.
Result: $\{P_{(\alpha,\aleph_a)}, P_{(\alpha,\aleph)}\}$.

$\theta \leftarrow 1-\alpha$; $c_1 \leftarrow \log_2(n)-1$; $c_2 \leftarrow \ln(2)$;
$f_0 \leftarrow e^{-\lambda^2/2}/(2\pi)^{1/2}$; $f_1 \leftarrow e^{-\lambda^2/2-\delta^2/2}$;
for $j \leftarrow 1,2,\cdots,r$ **do**
  $\eta_j \leftarrow c_1 + j$;
  $\beta_j \leftarrow \theta(2\ln(c_2\eta_j))^{1/2}$;
  $\rho_{(j-1)} \leftarrow 2-2\mathrm{Int}(f_0, \lambda \leftarrow -\infty \cdot\cdot \beta_j)$;
  for $\tau \leftarrow 1,2,\cdots,r-j$ **do**
    $a_{j,\tau} \leftarrow (\theta(2^{\tau+1}\ln(c_2(\eta_j+\tau)))^{1/2}-\lambda)/(2^{\tau}-1)^{1/2}$;
    $f_{2_{j,\tau}} \leftarrow \mathrm{Int}(f_1, \lambda \leftarrow a_{j,\tau}\cdot\cdot\infty)$
    $p_{j,\tau} \leftarrow \rho_{(j-1)} + 1/(\pi)\mathrm{Int}(f_{2_{j,\tau}}, \lambda \leftarrow -\beta_j \cdot\cdot \beta_j)$;
    $P_j \leftarrow [\rho_{(j-1)}, p_{j,\tau}]$;
  if $j \leftarrow r$ **then**
    $P_j \leftarrow [\rho_{(j-1)}]$;
return $\{P_{(\alpha,\aleph_a)}, P_{(\alpha,\aleph)}\} \leftarrow P_j$;

3.1 Evaluación de PRGN mediante el test LIL-débil

Para evaluar la calidad de un generador de números pseudoaleatorio G, Wang en [33] propone la siguiente metodología, primeramente se selecciona una sucesión de longitud fija n, un valor $0 < \alpha \leq 0{,}1$ y recíprocamente los subconjuntos distintos $\aleph_0, \cdots, \aleph_t$ de $\{1,\cdots,n\}$. Entonces para una selección de conjuntos $\aleph_i$ se pueden realizar los siguientes pasos.

1. Sea $P^+_{(\alpha,\aleph)} = P^-_{(\alpha,\aleph)} = \frac{1}{2}P_{(\alpha,\aleph)}, \forall\, \aleph$.
2. Usar G para construir el conjunto de $m \geq 100$ secuencias binarias de longitud n.
3. Para cada $\aleph$, se calcula la probabilidad $p^+_{(\alpha,\aleph)}$ de que estos conjuntos pasan el test LIL-$(\alpha, \aleph_i)$ débil mediante $S_{lil} \geq 1-\alpha$ ($p^-_{(\alpha,\aleph)}$ para $s_{lil} \leq -1+\alpha$).
4. Calcular el promedio de la distancia de la probabilidad absoluta

$$\Delta_w lil = \frac{1}{t+1}\sum_{i=0}^{t} P^{-1}_{(\alpha,\aleph_i)}$$

$$\left(|p^+_{(\alpha,\aleph_i)} - P^+_{(\alpha,\aleph_i)}| + |p^-_{(\alpha,\aleph_i)} - P^-_{(\alpha,\aleph)}|\right)$$

y la desviación de la raíz-cuadrática media

$$\mathrm{RMSD}_{wlil} = \sqrt{\frac{\sum_{0\geq i\geq j\geq t}(p^2_{i,j,1}+p^2_{i,j,2})}{(t+1)(t+2)}}$$

donde

$$p^+_{i,j,1} = p^+_{(\alpha,\aleph_i\cup\aleph_j)} - P^+_{(\alpha,\aleph_i\cup\aleph_j)}$$

y

$$p^+_{i,j,2} = p^-_{(\alpha,\aleph_i\cup\aleph_j)} - P^-_{(\alpha,\aleph_i\cup\aleph_j)}$$

5. Criterio de decisión: Mientras más pequeños sean Δ_{wlil} y RMSD_{wlil} mejor será el generador.

3.2 Algunas consideraciones para la evaluación aplicando el test LIL-débil

La distribución inducida por la función S_{lil} define una medida de probabilidad en la recta real R. Sea $\Re \subset \Sigma^n$ un conjunto de m sucesiones con probabilidad de definición estándar, o sea que para cada sucesión $\delta_0 \in \Re$, $Prob[\delta = \delta_0] = 1/m$. Entonces cada conjunto $\Re \subset \Sigma^n$, induce una medida de probabilidad $\mu^{\Re}_n$ en la recta R dada por

$$\mu^{\Re}_n(I) = Prob[Slil(\delta) \in I, \delta \in \Re].$$

Para $U = \Sigma^n$, se denota μ^U_n como la correspondiente medida de probabilidad inducida por la distribución uniforme. Por definición, si $\Re$ es la colección de todas las sucesiones generadas por un generador pseudoaleatorio, entonces la diferencia entre μ^U_n y $\mu^{\Re_n}_n$ es despreciable.

Por el teorema 2.2, la distribución inducida por la función $S^*(\xi|n)$ de una sucesión ξ seleccionada uniformemente, puede ser aproximada a la distribución normal con media 0 y varianza 1. Por consiguiente la medida μ^U_n puede ser calculada como

$$\mu^U_n((-\infty,\delta)) \simeq \Phi(\delta\sqrt{2\ln(\ln(n))})\int_{-\infty}^{\delta}\phi(\lambda\sqrt{2\ln(\ln(n))})d\lambda.$$

Entonces para evaluar un generador pseudoaleatorio G, primeramente se selecciona las longitudes de las sucesiones $n_0, \cdots, n_t$ (Ej., $n_0 = 2^{26+t}$). Segundamente se usa el generador G para generar el conjunto $\Re \subseteq \Sigma^{n_t}$ de m sucesiones. Por último se compara las distancias entre las dos medidas de probabilidad μ^R_n y μ^U_n para $n = n_0, \cdots, n_t$.

Un generador G es considerado "bueno", si para una suficiente cantidad de sucesiones m (ej., $m1 \geq 10000$), la distancia entre $\mu^{\Re}_n$ y μ^U_n es despreciable (o más pequeña que el valor esperado). Existen varias definiciones de distancias estadísticas para medidas de probabilidad, como las siguientes.

La variación total de distancia

$$d(\mu^{\Re}_n, \mu^U_n) = sup_{A\subseteq B}|\mu^{\Re}_n(A) - \mu^U_n(A)|$$

Distancia de Hellinger

$$H(\mu^{\Re}_n||\mu^U_n) = \frac{1}{\sqrt{2}}\sqrt{\sum_{A\in B}\left(\sqrt{\mu^{\Re}_n(A)} - \sqrt{\mu^U_n(A)}\right)^2}$$

Desviación de la raíz-cuadrática-media

$$RMSD(\mu_n^{\Re},\mu_n^U)=\sqrt{\frac{\sum_{A\in B}(\mu_n^{\Re}(A)-\mu_n^U(A))^2}{|B|}}$$

donde B es una partición de la recta real R que se define como $\{(\infty,1),[1,\infty)\}\cup\{0.05\lambda-1,0.05\lambda-0.95)\}:0\leq\lambda\leq 39$.

En la sección 5 se presenta un ejemplo de aplicación de estas distancias en la evaluación de generadores de números pseudoaleatorios.

4. Optimización de las probabilidades teóricas del Test LIL-débil

En esta sección se exponen los resultados obtenidos a partir de una simplificación de las probabilidades teóricas del test LIL-débil.

A partir de los resultados obtenidos por Wang [33] en la ecuación 6 y el Teorema 3.1 se pueden conformar los siguientes corolarios para el cálculo de las probabilidades teóricas del test LIL débil:

Corolario 1 *Para* $0<\alpha<1$ *fijo, sea* $\theta=1-\alpha$ *y* $\aleph_a=\{n\}$. *Se tiene que*

$$P_{(\alpha,\aleph_a)}=Prob[|S_{lil}(\xi|n)|\geq\theta]\simeq \operatorname{erfc}\left(\theta\sqrt{\ln(\ln(n))}\right) \quad (9)$$

Demostración.

Por la ecuación dada en 6 tenemos la probabilidad representada en términos de la distribución normal y ahora se procede a representarla en término de la función error de Gauss de la siguiente manera.

$$Prob[|S_{lil}(\xi|n)|\geq\theta]\simeq 2\left(1-\Phi\left(\theta\sqrt{2\ln(\ln(n))}\right)\right)$$

$$=2(1-\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\theta\sqrt{2\ln(\ln(n))}}e^{-\frac{\delta^2}{2}}d\delta) \quad (10)$$

donde, $\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\theta\sqrt{2\ln(\ln(n))}}e^{-\frac{\delta^2}{2}}d\delta$

$$=\frac{1}{2}(1+\operatorname{erf}(\theta\sqrt{\ln(\ln(n))})) \quad (11)$$

Sustituyendo el resultado de la expresión 11 en la 10 y simplificando se obtiene que:

$$P_{(\alpha,\aleph_a)}\simeq 2\left(1-\Phi\left(\theta\sqrt{2\ln(\ln(n))}\right)\right)=$$
$$1-\operatorname{erf}(\theta\sqrt{\ln(\ln(n))})=\operatorname{erfc}(\theta\sqrt{\ln(\ln(n))}).\diamond$$

Corolario 2 *Para* $0<\alpha<1$ *fijo y* $t\geq 2$, *sea* $\theta=1-\alpha$, $\aleph=\{n,tn\}$. *Se tiene que*

$$i)\ P_{(\alpha,\aleph)}\simeq 1-\frac{1}{\sqrt{2\pi}}\int_{-\theta\sqrt{2\ln(\ln(n))}}^{\theta\sqrt{2\ln(\ln(n))}}e^{-\frac{\lambda^2}{2}}$$

$$\operatorname{erf}\left(\frac{2\theta\sqrt{t\ln(\ln(tn))}-\sqrt{2}\lambda}{2\sqrt{t-1}}\right)d\lambda. \quad (12)$$

Alternativamente se tiene que:

$$ii)\ P_{(\alpha,\aleph)}\simeq\frac{1}{2}P_{(\alpha,\aleph_a)}+P_{(\alpha,\aleph_b)}-\frac{1}{\sqrt{2\pi}}\int_{\theta\sqrt{2\ln(\ln(n))}}^{\infty}e^{-\frac{\lambda^2}{2}}$$

$$\operatorname{erf}\left(\frac{2\theta\sqrt{t\ln(\ln(tn))}-\sqrt{2}\lambda}{2\sqrt{t-1}}\right)d\lambda. \quad (13)$$

Demostración.

i) A partir de la expresión 7 del Teorema 3.1 se tiene que:

$$\int_{\sqrt{\frac{1}{t-1}}(\theta\sqrt{2t\ln(\ln(tn))}-\lambda)}^{\infty}e^{-\frac{\delta^2+\lambda^2}{2}}d\delta=$$

$$e^{-\frac{\lambda^2}{2}}\int_{\sqrt{\frac{1}{t-1}}(\theta\sqrt{2t\ln(\ln(tn))}-\lambda)}^{\infty}e^{-\frac{\delta^2}{2}}d\delta=$$

$$-\frac{\sqrt{2\pi}}{2}e^{-\frac{\lambda^2}{2}}\left(\operatorname{erf}\left(\frac{2\theta\sqrt{t\ln(\ln(tn))}-\sqrt{2}\lambda}{2\sqrt{t-1}}\right)-1\right),$$

de aquí que:

$$\frac{1}{\pi}\int_{-\theta\sqrt{2\ln(\ln(n))}}^{\theta\sqrt{2\ln(\ln(n))}}\int_{\sqrt{\frac{1}{t-1}}(\theta\sqrt{2t\ln(\ln(tn))}-\lambda)}^{\infty}e^{-\frac{\delta^2+\lambda^2}{2}}d\delta d\lambda$$

$$=-\frac{1}{\sqrt{2\pi}}\int_{-\theta\sqrt{2\ln(\ln(n))}}^{\theta\sqrt{2\ln(\ln(n))}}e^{-\frac{\lambda^2}{2}}$$

$$\left(\operatorname{erf}\left(\frac{2\theta\sqrt{t\ln(\ln(tn))}-\sqrt{2}\lambda}{2\sqrt{t-1}}\right)-1\right)d\lambda=$$

$$-\frac{1}{\sqrt{2\pi}}\int_{-\theta\sqrt{2\ln(\ln(n))}}^{\theta\sqrt{2\ln(\ln(n))}}e^{-\frac{\lambda^2}{2}}\operatorname{erf}\left(\frac{2\theta\sqrt{t\ln(\ln(tn))}-\sqrt{2}\lambda}{2\sqrt{t-1}}\right)d\lambda$$

$$+\frac{1}{\sqrt{2\pi}}\int_{-\theta\sqrt{2\ln(\ln(n))}}^{\theta\sqrt{2\ln(\ln(n))}}e^{-\frac{\lambda^2}{2}}d\lambda$$

donde,

$$\frac{1}{\sqrt{2\pi}}\int_{-\theta\sqrt{2\ln(\ln(n))}}^{\theta\sqrt{2\ln(\ln(n))}}e^{-\frac{\lambda^2}{2}}d\lambda=$$

$$\frac{2}{\sqrt{2\pi}}\int_{0}^{\theta\sqrt{2\ln(\ln(n))}}e^{-\frac{\lambda^2}{2}}d\lambda=\operatorname{erf}\left(\theta\sqrt{\ln(\ln(n))}\right),$$

teniendo en cuenta que

$$\operatorname{erf}(\delta)=\frac{2}{\sqrt{\pi}}\int_0^{\delta}e^{-t^2}dt \text{ y } \operatorname{erfc}(\delta)=\frac{2}{\sqrt{\pi}}\int_{\delta}^{\infty}e^{-t^2}dt$$

son las funciones error y error complementario respectivamente.

Por consiguiente, sustituyendo este resultado y el de la expresión 9 en 7, se obtiene 12. ⋄

ii) De manera análoga a partir del teorema 3.1 se puede demostrar la expresión alternativa 13 dado que:

$$-\frac{1}{\sqrt{2\pi}}\int_{\theta\sqrt{2\ln(\ln(n))}}^{\infty} e^{-\frac{\lambda^2}{2}}$$

$$\left(\operatorname{erf}\left(\frac{2\theta\sqrt{t\ln(\ln(tn))}-\sqrt{2}\lambda}{2\sqrt{t-1}}\right)-1\right)d\lambda =$$

$$-\frac{1}{\sqrt{2\pi}}\int_{\theta\sqrt{2\ln(\ln(n))}}^{\infty} e^{-\frac{\lambda^2}{2}}\operatorname{erf}\left(\frac{2\theta\sqrt{t\ln(\ln(tn))}-\sqrt{2}\lambda}{2\sqrt{t-1}}\right)d\lambda$$

$$+\frac{1}{\sqrt{2\pi}}\int_{\theta\sqrt{2\ln(\ln(n))}}^{\infty} e^{-\frac{\lambda^2}{2}}d\lambda$$

donde

$$\frac{1}{\sqrt{2\pi}}\int_{\theta\sqrt{2\ln(\ln(n))}}^{\infty} e^{-\frac{\lambda^2}{2}}d\lambda$$

$$=\frac{\sqrt{2}}{2\sqrt{\pi}}\left(\frac{\sqrt{2\pi}}{2}-\frac{\sqrt{2\pi}}{2}\operatorname{erf}\left(\theta\sqrt{\ln(\ln(n))}\right)\right)$$

simplificando se obtiene que

$$\frac{1}{\sqrt{2\pi}}\int_{\theta\sqrt{2\ln(\ln(n))}}^{\infty} e^{-\frac{\lambda^2}{2}}d\lambda=\frac{1}{2}-\frac{1}{2}\operatorname{erf}\left(\theta\sqrt{\ln(\ln(n))}\right)$$

$$=\frac{1}{2}\operatorname{erfc}\left(\theta\sqrt{\ln(\ln(n))}\right)$$

Con este resultado y sustituyendo adecuadamente se obtiene 13. ⋄

Como se puede apreciar la fórmula obtenida en 12 no depende de $P_{(\alpha,\aleph_a)}$ por consiguiente la probabilidad $P_{(\alpha,\aleph)}$ se puede calcular de manera independiente.

De esta manera, se propone el Algoritmo 2 más optimizado que el Algoritmo 1, para el cálculo secuencial de $P_{(\alpha,\aleph_a)}$ y $P_{(\alpha,\aleph)}$, y los Algoritmos 3 y 4 para la implementación paralela de las mismas.

La Tabla I muestra el cálculo de $P_{(\alpha,\aleph_a)}$ y $P_{(\alpha,\aleph)}$ con $r=9$ y $n=2^{26}\cdots 2^{34}$, al aplicar el Algoritmo 2 o el 3 y 4 implementado en Maple, las cuales coinciden con las dadas por Wang en [33].

5. Resultados experimentales

En esta sección se exponen los resultados experimentales de la implementación de los algoritmos propuestos para el cálculo de las probabilidades teóricas del test LIL.

Desde el punto de vista de implementación el Algoritmo 2 es de mayor eficiencia respecto al Algoritmo 1, y aprovechando la independencia entre el cálculo de las probabilidades

Algoritmo 2: Cálculo de $P_{(\alpha,\aleph_a)}, P_{(\alpha,\aleph)}$.

Data: n, r, α.
Result: $\{P_{(\alpha,\aleph_a)}, P_{(\alpha,\aleph)}\}$.

$\theta \leftarrow 1-\alpha$; $c_1 \leftarrow \log_2(n)-1$; $c_2 \leftarrow \ln(2)$; $f_1 \leftarrow e^{(\frac{-\lambda^2}{2})}$;
for $j \leftarrow 1,2,\cdots,r$ **do**
  $\varepsilon_j \leftarrow c1+j$; $\beta_j \leftarrow \theta(2\ln(c_2\varepsilon_j))^{1/2}$;
  $\rho_{(j-1)} \leftarrow \operatorname{erfc}(\beta_j/2^{1/2})$;
  for $\tau \leftarrow 1,2,\cdots,r-j$ **do**
    $a_{j,\tau} \leftarrow 1/2(2\theta(2^\tau \ln(c_2(\tau+\varepsilon_j)))^{1/2} - 2^{1/2}\lambda)/(2^\tau-1)^{1/2}$;
    $f_{2_{j,\tau}} \leftarrow \operatorname{erf}(a_{j,\tau})$;
    $p_{j,\tau} \leftarrow 1-1/(2\pi)^{1/2}\mathrm{Int}(f_1 f_{2_{j,\tau}}, \lambda \leftarrow -\beta_j \cdot\cdot \beta_j)$;
    $P_j \leftarrow [\rho_{(j-1)}, p_{j,\tau}]$;
  if $j \leftarrow r$ **then**
    $P_j \leftarrow [\rho_{(j-1)}]$;
return $\{P_{(\alpha,\aleph_a)}, P_{(\alpha,\aleph)}\} \leftarrow P_j$;

Algoritmo 3: Cálculo de $P_{(\alpha,\aleph_a)}$.

Data: n, r, α.
Result: $\{P_{(\alpha,\aleph_a)}\}$.

$\theta \leftarrow 1-\alpha$; $c_1 \leftarrow \log_2(n)-1$; $c_2 \leftarrow \ln(2)$;
for $j \leftarrow 1,2,\cdots,r$ **do**
  $\varepsilon_j \leftarrow c1+j$; $\beta_j \leftarrow \theta(\ln(c_2\varepsilon_j))^{1/2}$;
  $\rho_{(j-1)} \leftarrow \operatorname{erfc}(\beta_j)$;
return $\{P_{(\alpha,\aleph_a)}\} \leftarrow \rho_{(j-1)}$;

Algoritmo 4: Cálculo de $P_{(\alpha,\aleph)}$.

Data: n, r, α.
Result: $\{P_{(\alpha,\aleph)}\}$.

$\theta \leftarrow 1-\alpha$; $c_1 \leftarrow \log_2(n)-1$; $c_2 \leftarrow \ln(2)$; $f_1 \leftarrow e^{(\frac{-\lambda^2}{2})}$;
for $j \leftarrow 1,2,\cdots,r$ **do**
  $\varepsilon_j \leftarrow c1+j$; $\beta_j \leftarrow \theta(2\ln(c_2\varepsilon_j))^{1/2}$;
  for $\tau \leftarrow 1,2,\cdots,r-j$ **do**
    $a_{j,\tau} \leftarrow 1/2(2\theta(2^\tau \ln(c_2(\tau+\varepsilon_j)))^{1/2} - 2^{1/2}\lambda)/(2^\tau-1)^{1/2}$;
    $f_{2_{j,\tau}} \leftarrow \operatorname{erf}(a_{j,\tau})$;
    $p_{j,\tau} \leftarrow 1-1/(2\pi)^{1/2}\mathrm{Int}(f_1 f_{2_{j,\tau}}, \lambda \leftarrow -\beta_j \cdot\cdot \beta_j)$;
return $\{P_{(\alpha,\aleph)}\} \leftarrow p_{j,\tau}$;

$P_{(\alpha,\aleph_a)}$ y $P_{(\alpha,\aleph)}$ los Algoritmos 3 y 4 se pueden implementar paralelamente disminuyendo el tiempo de sus cómputos, como se ilustran los resultados en la Figura 1.

La primera barra del gráfico, correspondiente al Algoritmo 1 indica que este demora poco menos de 3 segundos para el cómputo $P_{(\alpha,\aleph_a)}$ y $P_{(\alpha,\aleph)}$ con $n=2^{26}\cdots 2^{34}$, y el tiempo

va aumentando como muestran las demás barras cuando se incrementa el valor de n hasta $n = 2^{40}$ que demora poco más de 8 segundos, tomando como valor inicial $n = 2^{26}$. En el caso del Algoritmo 2 para el cálculo secuencial, y el 3 y 4 para su implementación paralela, dados aquí, el tiempo de cálculo disminuye considerablemente a centésimas de segundos como ilustra la Figura 1, con $n = 2^{26} \cdots 2^{45}$. Por estos resultados es que se plantea la posibilidad de poder realizar experimentaciones más amplias en la evaluación de PRGN con el test LIL. Tomar los valores distintos de n significan el análisis de muestras con estos tamaños en una corrida del test LIL-débil.

A continuación se muestra mediante un ejemplo los resultados de evaluación utilizando la metodología vista en la sección anterior, aplicando para la comparación del tiempo de cómputo los algoritmos 1 y 2. Para las experimentaciones de evaluación se tomó como muestra un fichero cifrado con el algoritmo AES en modo CTR. Se analizaron 1000 sucesiones binarias en el conjunto $\aleph = \{2^{12}, \cdots, 2^{20}\}$. Según la Figura 2 podemos decir que las distribuciones observadas para los distintos tamaños de muestras ($2^{12} \ldots 2^{20}$) oscilan alrededor de las distribuciones teóricas. Para cada una de estas distribuciones observadas se puede ir chequeando la distancia estadística entre ella y la distribución teórica. Como se puede observar en la figura hay algunos puntos o picos en el que la distribución observada se separa de la teórica, como por ejemplo la distribución $\mu_n^{AES-CTR}$ representada en color rojo con $n = 2^{13}$, esta separación afecta el promedio de la técnica de distancia que se utilice, de maneara que mientras mejor sea el promedio de las distancias utilizadas, mejor será el generador. Estas distancias de las distribuciones se pueden analizar de manera independiente o de manera conjunta como lo realiza el test. El valor del promedio de la distancia obtenido es comparado con un valor téorico esperado, lo que permite decidir sobre el comportamiento de aleatoriedad del generador, por ejemplo, si es recomendable para aplicaciones criptográficas.

En cuanto a la Figura 3 se representan las distribuciones $S_{lil}^{AES-CTR}$ de 100 sucesiones en el rango de 1000 a 10000 bits, como se puede observar varias de estas sucesiones se expanden fuera del intervalo [-1, 1] en el eje vertical. El test LIL débil se encarga de comprobar la probabilidad de que las distribuciones de S_{lil} caigan en los rangos $1-\alpha$ y $-1+\alpha$, representados en el gráfico mediante las líneas rojas y verdes.

Los resultados al implementar los algoritmos analizados se muestran a continuación.

Alg. 1: $\Delta_w lil = 0{,}1824$ y $\text{RMSD}_{wlil} = 0{,}00134$, en un tiempo de ejecución de 28.25 segundos.

Alg. 2: $\Delta_w lil = 0{,}1824$ y $\text{RMSD}_{wlil} = 0{,}00134$, en un tiempo de ejecución de 19.75 segundos.

Las experimentaciones se realizaron en una PC Intel(R) Core(TM)i3 CPU: 3.10GHz con 2G de RAM.

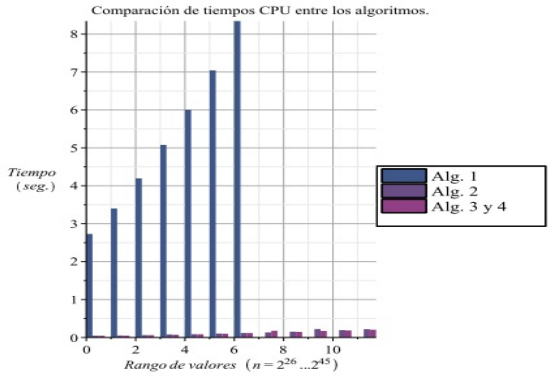

Figura 1. Comparación de tiempos de la implementación de los algoritmos 1,2,3 y 4.

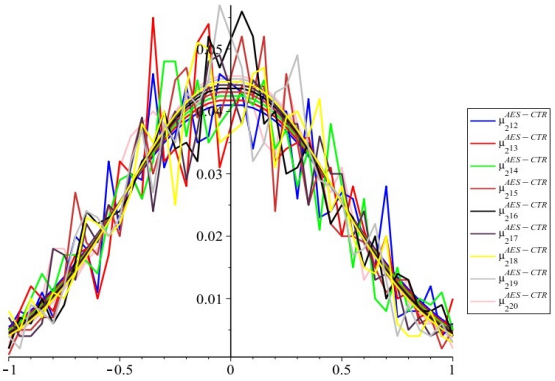

Figura 2. Distribuciones de μ_n^U y $\mu_n^{AES-CRT}$, con $n = 2^{12}, \cdots, 2^{20}$.

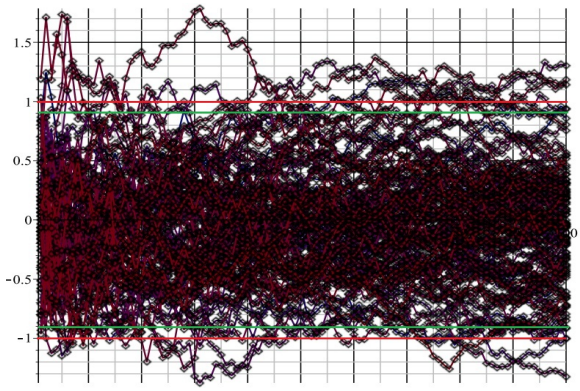

Figura 3. Resultados del test LIL para sucesiones generadas por AES-CTR.

$\alpha = 0.1$	$\aleph_0(2^{26})$	$\aleph_1(2^{27})$	$\aleph_2(2^{28})$	$\aleph_3(2^{29})$
$\aleph_0(2^{26})$	0.03044	0.05085	0.05441	0.05540
$\aleph_1(2^{27})$		0.02938	0.04918	0.05263
$\aleph_2(2^{28})$			0.02838	0.04762
$\aleph_3(2^{29})$				0.02746
$\alpha = 0.1$	$\aleph_5(2^{31})$	$\aleph_6(2^{32})$	$\aleph_7(2^{33})$	$\aleph_8(2^{34})$
$\aleph_5(2^{31})$	0.02580	0.04351	0.04660	0.04750
$\aleph_6(2^{32})$		0.02505	0.04230	0.04531
$\aleph_7(2^{33})$			0.02434	0.04116
$\aleph_8(2^{34})$				0.02367

Tabla I. Probabilidades del test LIL-débil con $n = 2^{26} \cdots 2^{34}$.

6. Conclusiones

En este artículo se revisó el diseño del test LIL-débil introducido en trabajos recientes para la evaluación de la calidad de generadores de números peseudoaleatorios. Se realizó una optimización de las fórmulas dadas en [33] para el cálculo de las probabilidades teóricas ($P_{(\alpha,\aleph)}$ y $P_{(\alpha,\aleph_a)}$) para que una sucesión pase el test sobre la *ley del logaritmo iterado* y se presentaron dos corolarios y 3 algoritmos para el computo de las mismas.

La forma que se propone para el cálculo de estas probabilidades permite una independencia entre ellas, lo cual facilita la paralelización del algoritmo y por consiguiente implementaciones más eficientes del test LIL-débil. Esta forma de representación en términos de la función error, también podría ser utilizada en los otros diseños de los test LIL (diseño del test LIL débil-II y diseño del test LIL-fuerte) con el objetivo de facilitar el cálculo de las probabilidades teóricas para que una sucesión pase el test y de esta manera poder realizar experimentaciones y evaluaciones de PRGN más amplias y eficientes.

Referencias

[1] E. Azmoodeh, G. Peccati y G. Poly. *The law of iterated logarithm for subordinated Gaussian sequences: uniform Wasserstein bounds.* ALEA, Lat. Am. Probab. Math. Stat. 13, 659-686, 2016.

[2] A. Balsubramani, A. Ramdas. *Sequential nonparametric testing with the law of the iterated logarithm.* Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence. AUAI Press, p. 42-51, 2016.

[3] E. Barker y J. Kelsey. NIST SP 800-90A: *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*. NIST, 2012.

[4] H. Demirhan y N. Bitirim. *Statistical Testing of Cryptographic Randomness*. Journal of Statisticians: Statistics and Actuarial Sciences IDIA 9, 1, 1-11, 2016.

[5] Y. Dodis, C. Ganesh, A. Golovnev, A. Juels y T. Ristenpart. *A Formal Treatment of Backdoored Pseudorandom Generators*. Advances in Cryptology-EUROCRYPT 2015. 34th Annual Conference on the theory Applications of Cryptographic techniques, Sofia, Bulgaria, p. 102-128, 2015.

[6] E. D. Erdmann. *Empirical tests of binary keystreams*. M.S. thesis, Department of Mathematics, Royal Holloway and Bedford New College, University of London, 1992.

[7] W. Feller. *Introduction to Probability Theory and Its Applications*. vol. I, Wiely, Berlin, 1968.

[8] K. G. Jamieson, M. Malloy, R. D. Nowak, S. Bubeck. *lil'ucb: An optimal exploration algorithm for multi-armed bandits*. COLT, vol. 35, p. 423-439, 2014.

[9] A. Khintchine. *Uber einen satz der wahrscheinlichkeitsrechnung*. Fund. Math, 6: p. 9-20, 1924.

[10] D. E. Knuth. *The Art of Computer Programming, Seminumerical Algorithms*. Vol. 2, 3 rd ed., Addison-Wesley, 1998.

[11] P. L'ecuyer. *Software for uniform random number generation: Distinguishing the good and the bad*. Proceedings of the Winter Simulation Conference. IEEE Press, p. 95-105, 2001.

[12] P. L'ecuyer. *Random number generation*. Handbook of Computational Statistics, J.E. Gentle, W. Haerdle, and Y. Mori, Eds. Springer-Verlag, Berlin, Germany. p. 35-70. Chapter II.2, 2004.

[13] P. L'ecuyer y R. Simard. *Beware of linear congruential generators with multipliers of the form* $a = \pm 2^q \pm 2^r$. ACM Trans. Math. Soft. 25, 3, p. 367-374, 1999.

[14] P. L'ecuyer y R. Simard. *On the performance of birthday spacings tests for certain families of random number generators*. Mathem. Comput. Simul. 55, p. 1-3, p. 131-137, 2001.

[15] P. L'ecuyer, R. Simard y S. Wegenkittl. *Sparse serial tests of uniformity for random number generators*. SIAM J. Scient. Comput. 24, 2, p. 652-668, 2002.

[16] P. L'ecuyer y Richard Simard. *A C Library for Empirical Testing of Random Number Generators*. ACM Trans. Math. Softw. 33, 4, Article 22, 2007.

[17] P. L'ecuyer y R. Touzin. *Fast combined multiple recursive generators with multipliers of the form* $a = \pm 2^q \pm 2^r$. Proceedings of the Winter Simulation Conference. Fishwick, Eds. IEEE Press, p. 683-689, 2000.

[18] P. Li, X. Yi, X. Liu, Yuncai Wang y Yongee Wang. *Brownian motion properties of optoelectronic random bit generators based on laser chaos*. OPTICS EXPRESS, Vol. 24, No. 14, 2016.

[19] G. Marsaglia. *A current view of random number generators*. Computer Science and Statistics, Sixteenth Symposium on the Interface. Elsevier Science Publishers, North-Holland, Amsterdam, The Netherlands. p. 3-10, 1985.

[20] G. Marsaglia. *DIEHARD: A battery of tests of randomness*. http://stat.fsu.edu/ geo/diehard.html, 1996.

[21] G. Marsaglia y W.Tsang. *Some difficult-to-pass tests of randomness*. J. Statist. Soft. 7, 3, p. 1-9, 2002.

[22] G. Marsaglia y A. Zaman. *Monkey tests for random number generators*. Comput. Math. Applic. 26, 9, p. 1-10, 1993.

[23] M. Mascagni y A. Srinivasan. *Algorithm 806: SPRNG: A scalable library for pseudorandom number generation*. ACM Trans. Mathem. Soft. 26, p. 436-461, 2000.

[24] K. Miyabe, A. Takemura. *The law of the iterated logarithm in game-theoretic probability with quadratic and stronger hedges*. Stochastic Process. Appl., 123(8), p. 3132-3152, 2013.

[25] A. L. Rukhin. *Testing randomness: A suite of statistical procedures*. Theo. Probab. Applic. 45, 1, p. 111-132, 2001.

[26] A. L. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel,D. Banks, A. Heckert, J. Dray y S. Vo. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*. NIST SP 800-22, 2010.

[27] T.Sasai, K.Miyabe, A. Takemura,. *A game-theoretic proof of Erdos-Feller-Kolmogorov-Petrowsky law of the iterated logarithm for fair-coin tossing* arXiv:1408.1790v2,[math.PR], 2014.

[28] D. Shumow y N. Ferguson. *On the possibility of a back door in the NIST SP800-90 Dual Ec Prng*. Proc. Crypto'07, 2007.

[29] Y. Song, G. Fellouris. *Sequential multiple testing with generalized error control: an asymptotic optimality theory*. arXiv:1608.07014v1 [math.ST], 2016.

[30] Y. Song, G. Fellouris. *Asymptotically optimal, sequential, multiple testing procedures with prior information on the number of signals*. Electronic Journal of Statistics, ISSN: 1935-7524, Vol.11, p. 338-363, 2017.

[31] I. Vattulainen, T. Ala-Nissila y K. Kankaala. *Physical models as tests of randomness*. Physic. Rev. E 52, 3, p. 3205-3213, 1995.

[32] U. V. Vazirani y V. V. Vazirani. *Trapdoor pseudo-random number generators, with applications to protocol design*. FOCS. vol. 83, p. 23-30, 1983.

[33] Y. Wang. *On the Design of LIL Test for (Pseudo) Random Generators and Some Experimental Result*. arXiv:1401.3307v1 [cs.CR], 2014.

[34] Y. Wang y T. Nicol. *On statistical distance based testing of pseudo random sequences and experiments with PHP and Debian Open SSL*. Elsevier, Computers and Security, 53, p. 44-64, 2015.

[35] Y. Wang. *On Stochastic Security of Pseudorandom Sequences*. http://webpages.uncc.edu/yonwang/papers/lilprfV2.pdf, 2014.

[36] Y. Wang. *Resource bounded randomness and computational complexity*. Theoret. Comput. Sci., 237: p. 33-55, 2000.

[37] F. Yang, A. Ramdas, K. Jamieson, M. Wainwright. *A framework for Multi-A(rmed)/B(andit) testing with online FDR control*. arXiv:1706.05378v1 [stat.ML], 2017.