# Congruencias lineales y sucesiones de números seudoaleatorios de período máximo

S.Sánchez[1],R.Criado[1]y C.Vega[2]

**Resumen**

En este trabajo mostramos cómo escoger los parámetros a, b y m del generador lineal congruente $x_{i+1} = (ax_i + b)modm$ de manera que la longitud del período de la serie de números seudoaleatorios generada se aproxime a la cota teórica de m!.

## 1 Introducción

En muchas áreas de las matemáticas surge la necesidad de utilizar las sucesiones de números seudoaleatorios para desarrollar técnicas tan alejadas a priori como la localización por sondeo primario mediante radares y la criptografía. Una sucesión de números seudoaleatorios se puede obtener por iteración de una función de una o varias variables de manera que la periodicidad de la sucesión no resulte evidente. La sucesión obtenida a partir del generador lineal congruente

$$x_{n+1} = (a \cdot x_n + b)modm,$$

introducido por Lehmer (1949), posee características y propiedades que la hacen especialmente atractiva para su utilización de forma masiva, tanto desde el punto de vista estadístico como por la posibilidad de obtener períodos suficientemente largos si los parámetros $a, b, m$ se eligen convenientemente [1]. Su principal virtud consiste en que el número de operaciones por bit generado es un número pequeo. Por otra parte, su mayor desventaja radica en que las sucesiones que genera son predecibles ([2],[3],[4]). Esta debilidad pone de manifiesto que este generador no es adecuado para la generación de sucesiones cifradas pero, puede ser útil para el desarrollo de ciertas herramientas criptográficas en aplicaciones en las que se modeliza un comportamiento aleatorio como la búsqueda de números primos grandes donde es necesario comprobar si un número es compuesto o "probablemente" primo. El generador presentado permite corregir algunos defectos, pues su configuración permite romper el orden natural en la generación de números aleatorios y nos permite aproximarnos a una longitud de período importante, superior al de los generadores combinados [5], que se acerca a $m!$. Al ser un generador no lineal dificulta los ataques analíticos.

[1]Dpto. de CC. Experimentales e Ingeniería. E-mail: {ssanchezg,rcriado@escet.urjc.es

[2]Dpto. de Matemática Aplicada (DMATI). E-mail: cvega@mat.upm.es

## 2 Preliminares

Recordemos que si $F$ es un conjunto con un número finito de elementos, un generador de $F$es un algoritmo que obtiene una sucesión $\{x_n\}_{n\geq 0}$ de elementos de $F$, y que una sucesión $\{x_n\}_{n\geq 0}$ es periódica si $\exists k, k \geq 1$ tal que para cualquier $n \in \mathbb{N}$ se verifique que $x_{n+k} = x_n$, denominándose período al menor número entero $\lambda$ que satisfaga la condición de que para cualquier $n \in \mathbb{N}$, $x_{n+\lambda} = x_n$. En lo sucesivo nos vamos a centrar en los generadores de un solo paso, es decir, aquellos que se pueden expresar de la forma $x_{n+1} = f(x_n)$. donde $f$ es una función, $f : F \to F$. Utilizaremos la notación $\{F, f, x_0\}$ para referirnos a un generador de este tipo. Se dice que $\{F, f, x_0\}$ es de período máximo si el período de la sucesión que genera coincide con el número de elementos del conjunto $F$. No es difícil probar el siguiente resultado:

**Lema 1** *Siendo $\{F, f, x_0\}$ un generador de un paso, son equivalentes:*

a) *$\{F, f, x_0\}$ es un generador de período máximo para $x_0 \in F$.*

b) *La aplicación $f$ es una permutación cíclica.*

c) $\forall x \in F$, $\{F, f, x\}$ *es un generador de período máximo.*

d) *Para cualesquiera* $x, y \in F$ *existe* $r \in \mathbb{N}$, *tal que* $f^r(x) = y$.

Si nos centramos en los generadores que utilizan congruencias lineales del estilo de

$$x_{n+1} = (a \cdot x_n + b) mod m,$$

como consecuencia del lema 2 que, seguidamente, enunciamos, veremos que es posible obtener sucesiones con períodos suficientemente largos si los parámetros $a, b$ y $m$ se eligen convenientemente:

**Lema 2** ([1]) *El generador* $x_{n+1} = (a \cdot x_n + b) mod m$ *tiene un período de longitud máxima si y solo si se cumplen las siguientes condiciones:*

a) $b$ *es inversible módulo* $m$.

b) $a \equiv 1$ $(mod\ p)$ *para cualquier primo* $p$ *que sea divisor de* $m$

c) *Si* 4 *divide a* $m$, *entonces* $a \equiv 1$ $(mod\ 4)$.

En nuestro caso, el período máximo será $\lambda = m$ por lo que, para centrar ideas, supondremos en lo sucesivo que $F = \mathbb{Z}_m = \{0, 1, ..., m-1\}$. Por otra parte, es inmediato comprobar que, siendo $x_{n+1} = (a \cdot x_n + b) mod m$, para cualquier $k \in \mathbb{N}$ se satisface la relación $x_k = a^k x_0 + (1 + a + a^2 + ... + a^{k-1}) \cdot b mod m$, con lo que, si utilizamos la notación $S_k(a) = (1 + a + a^2 + ... + a^{k-1}) \cdot b mod m$, resulta que para cualquier $k \in \mathbb{N}$ se verifica que $x_k = a^k x_0 + S_k(a)$, con lo que, considerando el conjunto $\Phi$ de las funciones afines invertibles $f_{a,b} : \mathbb{Z}_m \to \mathbb{Z}_m$ definidas por la expresión $f_{a,b}(x) = \overline{a \cdot x + b}$, donde $\overline{a \cdot x + b}$ es la clase de equivalencia módulo $m$ correspondiente al número $a \cdot x + b$, no es difícil verificar que $\Phi$ es un subgrupo del grupo simétrico de $\mathbb{Z}_m$, y que $H = \{e, f, f^2, ..., f^{m-1}\}$ es un subgrupo cíclico de $\Phi$, donde $f^k : \mathbb{Z}_m \to \mathbb{Z}_m$ está determinada por la expresión $f^k(x) = a^k x + S_k(a)$. Ahora bien, si $f$ satisface las condiciones del lema 1, $f$ es una permutación cíclica y la igualdad $f^m = e$ es equivalente a que para cualquier $x \in \mathbb{Z}_m$, $f^m(x) = x$ o, lo que es lo mismo, a que $a^m = 1$ y $b \cdot S_m(a) = 1$. Si, por simplicidad, ponemos $b = 1$, obtenemos que $S_m(a) = 0$ y, en consecuencia, $a^m \equiv 1 (mod\ m)$, con lo que, para el caso en el que $m$ sea un número primo, resultará que $f$ tiene un punto fijo y que una condición necesaria para que $f$ tenga la longitud máxima del período es que $a \equiv 1 (mod\ m)$.

**Lema 3** *Si* $m$ *es un numero primo,* $a \equiv 1 (mod\ m)$ *y* $b$ *no es congruente con* $0 (mod\ m)$, *la función* $f : \mathbb{Z}_m \to \mathbb{Z}_m$ *definida por la expresión* $f(x) = \overline{a \cdot x + b}$, *donde* $\overline{a \cdot x + b}$ *es la clase de equivalencia módulo* $m$ *correspondiente al número* $a \cdot x + b$, *es una permutación cíclica de orden* $m$ *en* $\mathbb{Z}_m$ *(y, en consecuencia, es un generador de período completo).*

Nuestro objetivo es diseñar un generador que se acerque a la cota teórica de las $m!$ posibles permutaciones. Es conocido que, siendo $|\langle f \rangle| = s$, y dado $x \in \mathbb{Z}_m$, se verifica que, o bien $x$ es un punto fijo de un elemento de $\langle f \rangle$ (en cuyo caso a $x$ le denotaremos por $x_F$), o bien el conjunto $H_s = \{x, f(x), ..., f^{s-1}(x)\}$ tiene exactamente $s$ elementos. Además, es posible encontrar $x_1, x_2, ..., x_l \in \mathbb{Z}_m$ tales que

$$\mathbb{Z}_m = H_0 \cup H^1_{x_1} \cup ... \cup H^l_{x_l}$$

donde $H_0 = \{x_F\}$, $card(H^1_{x_1}) = ... = card(H^l_{x_l}) = s$, y $l \cdot s = m - 1$.

# 3 Generación de números seudoaletorios y congruencias lineales.

La generación basada en la expresión

$$x_{n+1} = (a \cdot x_n + b) mod m, \qquad (1)$$

depende de cuatro parámetros $a$, $b$, $m$ y $x_0$. Si consideramos los números presentes en la expresión (1), los casos de interés se pueden dividir en dos categorías:

1. Coeficientes $a, b, m$ que facilitan la generación de la longitud del período completo.

2. Coeficientes que no facilitan esta generación. El primer caso está considerado exhaustivamente en [1], donde se estudian, principalmente, las condiciones en las que se obtiene el período máximo y algunos aspectos relacionados con la eficiencia de este método. En nuestro caso, vamos a realizar el análisis de la segunda opción, centrándonos en el caso en el que $m$ es un número primo. Para ello

nos apoyaremos en los siguientes resultados conocidos, que resumen parte de lo tratado en el apartado anterior:

**Teorema 1.** *Si gcd(a-1,m)=1 , entonces para cualquier sucesión generada mediante la expresión (1) existe un punto fijo $x_F$ tal que*

$$x_F = (a \cdot x_F + b) mod m.$$

**Teorema 2.** *Si $m$ es primo, se cumple el Teorema1 y existe un punto fijo $x_F$, el conjunto formado por los $m-1$ números restantes se divide en $l$ tramos de $s$ elementos cada uno, de manera que*

$$l \cdot s = m - 1 \tag{2}$$

*El número de valores posibles de $s$ se puede obtener mediante la expresión*

$$\tau(\varphi(m)) - 1, \tag{3}$$

*dónde si $\varphi(m) = p_1^{\alpha_1} p_2^{\alpha_2} ... p_n^{\alpha_n}$, $\tau(p_1^{\alpha_1} p_2^{\alpha_2} ... p_n^{\alpha_n}) = (\alpha_1 + 1)(\alpha_2 + 1)...(\alpha_n + 1)$. Cualquier valor posible de $s$ es un divisor de $m-1$,excluído el 1. El número de tramos $l$ es*

$$l = \frac{m-1}{s}.$$

**NOTA**. El parámetro $a$ y un valor determinado del parámetro $s$, para el caso en el que $m$ sea primo, se determinan a partir de la ecuación

$$a^s \equiv 1 (mod m) \tag{4}$$

El número de soluciones de esta ecuación es $\varphi(s)$, de manera que si $s = \varphi(m)$ tenermos el teorema de Euler. Para poder elegir el coeficiente $a$ de entre los muchos posibles, se pueden tener en cuenta las posibilidades mínimas de los lenguajes de programación como Fortran o C. En el caso del lenguaje Fortan para que los resultados intermedios de calculo con la expresión (1) no sobrepasen la barrera de 31 bits se puede utilizar la descomposición de Schrage [8] que se basa en el hecho que cualquier numero de 31 bits se puede expresar como

$$\alpha \cdot 2^{16} + \beta$$

donde $\alpha$ y $\beta$ son números de 15 y 16 bits respectivamente. Si se utiliza el lenguaje C con posibilidad de representación mínima de 32 bits se pueden elegir los parámetros $a$, $b$ y $m$ de manera que satisfagan la condición

$$a(m-1) + b \leq 2^{32} - 1 \tag{5}$$

Para imponer condiciones adicionales se pueden seguir las recomendaciones de Knuth [1] para cumplir los criterios teóricos. Así, en el caso del criterio de correlación serial, se utilizan la suma generalizada de Dedekind $\sigma(a,b,m)$ que es pequea si las *relaciones parciales* $\frac{a}{m}$ son pequeas. En el caso general su evaluación resulta compleja. Unas condiciones menos precisas pero más sencillas ([1]) consisten en que $a$ y $m$ deben satisfacer la condición

$$\sqrt{m} < a < m - \sqrt{m} \tag{6}$$

y en que el parámetro $b$ ($k = b/m$) satisfaga la condición

$$\lfloor k \cdot m \rfloor \leq b \leq \lceil k \cdot m \rceil \tag{7}$$

donde

$$k = \left( \frac{1}{2} - \frac{1}{6}\sqrt{3} \right) \approx 0.211324865405187117745 4$$

Sustituyendo en (5) primero las cotas inferiores de (6) y (7) obtenemos, por ejemplo, para $m$ la ecuación

$$m^{\frac{3}{2}} - \sqrt{m} + \lfloor k \cdot m \rfloor - 4294967295 \leq 0 \tag{8}$$

resolviendo esta ecuación en números enteros obtenemos $m = 2642018$. De forma análoga se obtiene la ecuación para las cotas superiores y para los parámetros $a$ y $b$. Luego obtenemos la tabla siguiente :

Para las cotas inferiores

$a = 1626$, $b = 558324$, $m = 2642018$.

Para las cotas superiores

$a = 65665$ , $b = 13877$ , $m = 65665$

Luego, utilizando la aritmética entera de 32 bits y un generador de período completo, se puede obtener una serie de números de longitud de período del orden $\approx 2.65 \cdot 10^6$. En un generador de período completo la única forma de aumentar la longitud del período es aumentar el valor del parámetro $m$ (o recurrir a una combinación de generadores).Para este caso entenderemos por longitud de período la longitud del tramo de la serie hasta que dos elementos de la serie coincidan. Para poder generalizar esta definición a series distintas pero que puedan contener dos elementos iguales se puede observar que un generador de período completo genera $m$ series distintas (no esencialmente), por lo que su longitud de período se puede definir

como la longitud de la serie hasta que dos tramos de la serie de longitud $m$ coincidan. Se obtiene la misma longitud de período que en el caso de definición que hemos dado. Nuestro propósito es definir de forma análoga la longitud de período de la serie para el caso del generador de período incompleto o para el caso de series distintas pero que puedan contener dos elementos iguales. En este caso entenderemos por longitud del período de la serie a la longitud del tramo de la serie entre dos tramos iguales. Se puede observar que en el caso del generador de período completo solo hay un tramo, luego esta definición coincide con la anterior. En nuestro caso, al no ser un generador de período completo, se pueden seguir los siguientes procedimientos para lograr una longitud de período importante. Como disponemos de $l$ tramos la forma mas simple de funcionamiento seria la siguiente: Introducir un valor inicial $x_0$ del *i-ésimo* tramo, generar todos los números $X$ pertenecientes a este *i-ésimo* tramo, tomar un $x_0$ del tramo siguiente (el *(i+1)-ésimo*), y así sucesivamente hasta que se hayan utilizado todos los $x_0$ de todos los $l$ tramos. De esta forma se obtendrán todos los $m$ valores de $X$ en el intervalo desde 0 hasta $m-1$(contando el valor $x_F$). La generación de la serie depende de: 1. El orden en que se recorren los tramos. El número de variaciones que puede haber es $l!$ , luego, la longitud del período será $l! \cdot m$. 2. La elección del valor inicial $x_0$ en cada tramo. El número de combinaciones para la elección del valor inicial es $s^l$, luego a longitud del período será $l! \cdot m \cdot s^l$. 3. La elección de intervalo puede ser aleatoria, como también la elección del numero de elementos dentro del intervalo escogido. Este proceso se realiza de forma que se cumpla la ley uniforme de distribución de los $x$ en el intervalo entre 0 y $m-1$, o, en otras palabras cada valor $x$ del intervalo entre 0 y $m-1$ debe aparecer sólo una vez. Luego, la longitud del período será $l! \cdot m \cdot (s!)^l$. De esta forma para los parámetros $a$ ,$b$ ,$m$ elegidos se pueden generar no solo una serie (como en el caso del generador de período completo) sino muchas series distintas y por tanto acercarse a la cota teóricamente posible de $m!$ elementos. Así de acuerdo a las formas 1 y 3 si tomamos $m = 2641153$ (el primo mas próximo a 2642018 de la tabla) y, por ejemplo, $l = 48$, determinamos $s = 55024$. La longitud del período en cada caso será: 1. $48! \cdot 2.641153 \cdot 10^6 \approx 10^{61} \cdot 2.641153 \cdot 10^6 \approx 10^{67}$;

2. $48! \cdot 2.641153 \cdot 10^6 \cdot (55024)^{48} \approx 10^{61} \cdot 2.641153 \cdot 10^6 \cdot 10^{227} \approx 10^{294}$;

3. $48! \cdot 2.641153 \cdot 10^6 \cdot (55024!)^{48} \approx 10^{61} \cdot 2.641153 \cdot 10^6 \cdot 10^{11.373.648} \approx 10^{11.373.715}$. Para ver e grado de aproximación que se consigue podemos aplicar la formula de Stirling para aproximar el valor de 2641153!.

$$\ln(m!) \approx (m + \frac{1}{2}) \cdot \ln(m) - m + \ln(\sqrt{2\pi})$$

de donde

$$2641153! \approx 10^{15.813.905}$$

Veamos algunos ejemplos concretos en los cuales $m$ es un número primo. Si queremos un generador con $m$ del orden de $6 \cdot 10^5$, el número primo siguiente al elegido es $m = 600011$, y el número primo anterior es $m = 599999$, $m-1 = 599998 = 2 \cdot 7 \cdot 17 \cdot 2521$. Los valores posibles de $l$ son: 2 ,7 ,14 ,17 ,34 ,...Tomamos, por ejemplo, $l = 17$. Entonces $s = \frac{599999-1}{17} = 35294$. De la misma forma para el número primo $m = 600011$. De (4) y utilizando (5), (6) y (7) determinamos los parámetros $a$ y $b$. Los resultados de los cálculos para este $m$ para tres ejemplos los reunimos en la siguiente tabla:

| Ejemplo 1 | Ejemplo 2 | Ejemplo 3 | Ejemplo 4 |
|---|---|---|---|
| a=7557 | a=7557 | a=7648 | a=7648 |
| m=599999 | m=599999 | m=600011 | m=600011 |
| l=17 | l=14 | l=10 | l=29 |
| $X_F$=111550 | $X_F$=111550 | $X_F$=575043 | $X_F$=575043 |
| s=352294 | s=42857 | s=44884 | s=20690 |
| $X_0[i]$ | $X_0[i]$ | $X_0[i]$ | $X_0[i]$ |
| 26006 | 26006 | 26007 | 26007 |
| 26010 | 26010 | 26013 | 26013 |
| 26017 | 26017 | 26017 | 26017 |
| 26030 | 26030 | 26022 | 26022 |
| 26035 | 26035 | 26031 | 26031 |
| 26047 | 26047 | 26037 | 26037 |
| 26065 | 26065 | 27039 | 27039 |
| 26080 | 26080 | 27050 | 27050 |
| 26096 | 26096 | 27054 | 27054 |
| 26115 | 26115 | 27059 | 27059 |
| 26138 | 26138 | | 27066 |
| 26214 | 26214 | | 27071 |
| 26233 | 26233 | | 27078 |
| 26561 | 26561 | | 27093 |
| 26809 | | | 27102 |
| 27026 | | | 27111 |
| 27046 | | | 27113 |
| | | | 27128 |
| | | | 27169 |
| | | | 27181 |
| | | | 27185 |
| | | | 27189 |
| | | | 27204 |

El generador ha sido sometido a una serie de criterios estadísticos. No son suficientes para juzgar sus propiedades estadísticas, por lo que estamos preparando una batería de 15 test más significativa ([10]). Cada criterio se aplica a la secuencia de números reales

$$x_0, x_1, ... \tag{9}$$

que se suponen estadísticamente independientes y equiprobables entre 0 y 1. Algunos criterios están indicados para números enteros y no para la secuencia (8) . En este caso en vez de esta se utiliza la secuencia

$$y_0, y_1, \cdots \tag{10}$$

definida por

$$y_n = \lfloor d \cdot x_n \rfloor \tag{11}$$

que es una secuencia de números enteros igualmente probables ente 0 y $d-1$. El numero $d$ se elige de forma que sea cómodo usarlo en un sentido o en otro.

Para cada ejemplo se han generado 3 ficheros de acuerdo a las estrategias de generación $1-3$. Cada valor $x$ fue reducido al intervalo $(0,1)$ y multiplicado por un múltiplo $d = 2^8$ según se indica en [1]. Los ficheros obtenidos fueron sometidos a una serie de test estadísticos para un nivel de significación de $p = 0.99$.

| Ejemplos | | Test | |
|---|---|---|---|
| | $\chi^2$ | series | inversiones |
| Ejemplo 1 | | | |
| 1. | 0.0034 | 2.51 | 1.85 |
| 2. | 0.0034 | 3.06 | 2.20 |
| 3. | 0.0102 | 2.59 | 2.00 |
| Ejemplo 2 | | | |
| 1. | 0.0034 | 2.03 | 2.13 |
| 2. | 0.0034 | 3.23 | 2.15 |
| 3. | 0.0102 | 2.30 | 2.01 |
| Ejemplo 3 | | | |
| 1. | 0.0228 | 2.74 | 1.85 |
| 2. | 0.0228 | 4.18 | 2.25 |
| 3. | 0.0684 | 2.62 | 1.94 |
| Ejemplo 4 | | | |
| 1. | 0.0228 | 2.43 | 1.70 |
| 2. | 0.0228 | 5.18 | 2.37 |
| 3. | 0.0684 | 2.50 | 1.94 |

# 4 Generación de números seudoaletorios en criptología.

Los generadores que se utilizan en aplicaciones criptográficas se pueden dividir, en el caso general, en dos clases: generadores basados en elementos hardware y generadores programables. El generador programable es cierto procedimiento( o algoritmo) que se caracteriza por cierta secuencia de salida $x_0, x_1, ...x_n$. Los requisitos básicos que debe satisfacer esta secuencia es que los elementos que la componen sean equiprobables y que sean estadísticamente independientes. Un generador programable, en principio, no puede satisfacer este último requisito. Por consiguiente, el requisito de independencia estadística debe ser sustituido por otro que tenga sentido algorítmico. Se puede utilizar el concepto de complejidad algorítmica de recuperación del elemento $x_i$ conociendo el elemento $x_j$ cuando $i \neq j$ y viceversa. La justificación de este planteamiento consiste en que si existiera una dependencia analítica o de otra naturaleza entre los elementos de la salida, entonces utilizando ésta se podría definir un criterio de cálculo eficiente(en el sentido de complejidad algorítmica) de dependencia estadística. Si se garantiza que el esfuerzo computacional de este criterio es alto, entonces se puede hablar de complejidad (algorítmica) y, por tanto, de independencia estadística. Un generador programable, como cualquier autómata finito, se define por un estado $S(t)$, donde $t$ es el ciclo de funcionamiento, una función de transición entre los estados $\Phi(S(t)) = S(t+1)$ y una función de salida $\Psi(S(t)) = X(t)$. En nuestro caso la función de transición $\Phi$ es lineal y la función de salida $\Psi$ no es lineal, por lo que el generador presentado es no lineal. Se pueden plantear, para un generador programable los siguientes problemas :

- Alta complejidad para la recuperación de $X(t)$ conociendo $S(t-1)$ (ataque desde el pasado).

- Alta complejidad para la recuperación de $S(t)$ conociendo $X(t)$ (ataque desde el presente).

-Alta complejidad para la recuperación de $S(t)$ ó $X(t)$ conociendo $S(t+1)$(ataque desde el futuro).

De acuerdo con estos problemas se pueden plantear una serie de requisitos que deben satisfacer los generadores programables en aplicaciones criptograficas. El requisito 1, para los generadores programables, se puede formular como la imposibil-

idad de ataques eficientes desde el pasado, presente y futuro. El requisito 2 que la longitud de período del generador debe ser garantizada como alta . El requisito 3 es que cierta información (clave), como $S(t)$ o cualquier información que conduzca a su calculo, se debe ocultar.

## 5 Conclusiones

El generador presentado es en cierto sentido un modelo de prueba de ciertos planteamientos. Así las restricciones para los parámetros $a$, $b$, $m$ son algo artificiales y pueden ser sustituidos por otros o simplemente eliminados para futuras plataformas informáticas. Luego, por ejemplo, el parámetro $m$ puede ser un numero primo grande y entonces la factorización del número $m-1$ resulta compleja. Por consiguiente, todas las estimaciones y reconstrucciones de otros parámetros en los que interviene el parámetro $m$ no resultan eficientes. La clave la constituye el vector de semillas de cada intervalo

$$\overline{x_0} = \{x_0^1, x_0^2, ..., x_0^l\} \qquad (12)$$

Estas son raíces primitivas o generadores de los grupos cíclicos. Si la factorización de $m-1$ no se conoce, el cálculo de estas raíces no resulta eficiente existiendo, además, $\varphi(m-1)$ raíces primitivas. En principio, teniendo una salida extensa del generador, se pueden determinar los parámetros $a$, $b$ y $m$. Sin embargo la determinación de los intervalos no resulta eficiente si no se conoce la factorizacion de $m-1$, resultando especialmente complejo determinar el número de elementos por intervalo y el parámetro $s$ de la ecuación $a^s \equiv 1(modm)$ .

## Referencias


[1] D. E. Knuth. *The Art of Computer Programming.* V.2.Addison-Wesley.1981.

[2] Joan Boyar *Inferring sequences produced by pseudo-random number generators..* J.A.C.M. vol.36. pp.129-141. 1989.

[3] H.Krawczyk. *How to predict congruential generators.* Journal of Algorithms vol.13. pp.527-545. 1992.

[4] Frize A.M.,Hastad R.,Lagarias J.C., and Shamir *Trans Reconstruction truncated integer variables satisfing linear congruences.* Journal on Computing,,April 1988, vol.17(2). pp.262-280.

[5] Ramanujachary Kumanduri,Cristina Romero *Number Theory.* Prentice Hall,1998

[6] P.Bratley,B.L.Fox, L.E.Schrange *A Guide To Simulation.* Springer-Verlang ,1983

[7] Kostrikin A.I. *Introducción al Álgebra.* Mir ,1983

[8] Bruce Schneier *Applied Cryptography.* John Willey&Sons, Inc. 1996.

[9] L.Schrage *A more portable Fortran random number generator.* ACM.Trans. Math.Software vol.5. pp.132-138.

[10] G.Marsaglia *The Marsaglia Random Number CDROM, including the DIEHARD Battery of tests of Randomness.* Departament of Statistics,Florida State University, Tallahassee, Florida(1995).