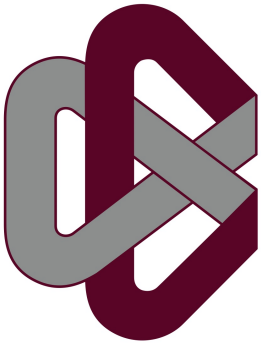

CIMAT

Centro de Investigación en Matemáticas A. C.

ALGORITMOS DE CIFRADO DE DATOS

TESIS

que para obtener el grado de

Maestría en Ciencias con Especialidad en Computación y Matemáticas Industriales

presenta

Marcos Daniel Calderón Calderón

Director de Tesis

Rogelio Hasimoto Beltrán

Dedicado a: *Mi familia.*

Índice general

Índice de figuras

Índice de cuadros

Agradecimientos

Capítulo 1

Introducción al caos

1.1. Origen

El término "caos" tiene su origen en la antigua cultura Griega y se deriva del vocablo $\chi\alpha o\zeta$, data de alrededor de 800 A. C. y significa "ausencia total de orden".

Capítulo 2

Mapa caótico Renyi

2.1. Proceso de discretización

El mapa caótico Renyi $\phi_\beta \colon [0,1) \longrightarrow [0,1)$ se define de la siguiente manera:

$$\phi(x) = (\beta \cdot x) \mod 1 \tag{2.1}$$

donde $1 < \beta \in \mathbb{R}$; además, si a y b son números reales no negativos, se cumple lo siguiente: $b = a - \lfloor a/b \rfloor b$. La función suelo $\lfloor \cdot \rfloor$ regresa el mayor número entero menor o igual que el argumento.

De aquí en adelante, $\tilde{x}$ será una representación en punto fijo de n bits donde

$$\tilde{x} = \mathbb{Q} : \ \tilde{x} = \frac{k}{2^n}, \qquad k \in \mathbb{N}, \qquad k < 2^n. \tag{2.2}$$

De acuerdo a la ecuación 2.2, toma valores racionales en el intervalo $[0,1)$ y se tiene una cardinalidad de 2^n.

Para lograr una evaluación de precisión finita de la ecuación 2.1, se utiliza una aproximación por truncamiento de la siguiente manera:

$$\tilde{\phi}_\beta(\tilde{x}) = (\beta \cdot \tilde{x})_{tr} \mod 1 = \lfloor (2^n \cdot \beta \cdot \tilde{x}) \mod 2^n \rfloor \cdot 2^{-n}. \tag{2.3}$$

La ecuación 2.3 debe ser implementada en máquinas de estado finito; por lo tanto, β puede ser, en general, cualquier número mayor que 1 cuya representación binaria requiera un número finito de dígitos. Por medio del isomorfismo $k = \tilde{x} \cdot 2^n$, el conjunto de posibles valores obtenidos en 2.2 puede ser relacionado con el conjunto de números naturales

$$\Lambda_n = \{k \in \mathbb{N}, \ 0 \leq k < 2^n\} \tag{2.4}$$

y una función $f \colon \Lambda_n \longrightarrow \Lambda_n$ puede ser definida como

$$\begin{aligned} f(k) &= 2^n \tilde{\phi}_\beta(2^{-n}k) \qquad = \lfloor \beta \cdot k \mod 2^n \rfloor \\ &= \lfloor \beta \cdot k \rfloor \mod 2^n. \end{aligned} \tag{2.5}$$

La función f es equivalente a 2.3. La ecuación 2.5 representa al Generador de Números Pseudoaleatorios que se utiliza en el trabajo descrito en este documento.

@@@@@@@@@@@@@@@@@@@@@@2Aqui voy jajaj

Capítulo 3

Caos Discretizado para la generación de números pseudoaleatorios en Criptografía.

3.1. Introducción

Los números aleatorios tienen un papel fundamental en Cryptorafía, se utilizan en muchas cosas, por ejemplo, para definir claves de cifrado o contraseñas. Actualemnte, se pueden generar números aleatorios de dos maneras: True Random Number Generators (TGNGs) y Pseudo Random Number Generators (PRNGs). Los primeros son dispositivos que explotan fenómenos físicos un un comportamiento verdaderamente estocástico; por ejemplo, el ruido electróncio de sistemas dinámicos caóticos no lineales, la secuencia obtenida de estos dispositivos tiene un grado intrínseco de impredecibilidad, que es medida mediante las herramientas te[oricas que se han desarrollado el área de Teoría de la Información. Del otro modo, PRNGs son máquinas de estado finito determinístas cuyo propósito es imitar el comportamiento aleatorio de una secuencia de números verdaderamente aleatoria. Desde un punto de vista teórico, debido a su naturaliza determinista, PRNGs son potencialmente predecibles al observar sus secuencias generadas. Sin embargo, en la literatura existente, algunas familias de PRNGs son clasificados como "seguros", esto significa que su estructura algorítmica implica cálculos que en promedio, requieren una cantidad de tiempo de cálculo que es asintóticamente inviable con el tamaño del problema. Cabe señalar que dado un generador, incluso si éste pertenece a una familia de PRNGs asintóticamente segura, puede generar secuencias de un periodo corto para varios valores tomados como semilla inicial. Por lo tanto, adermás de la robustez criptográrfica de su estructura algorítmica, un PRNG criptográfico debe generar secuencias que sean aceptables desde un punto de vista estadístico, ésto se logra cuando la secuencia generada se somente a un conjunto de pruebas estadísticas.

En este trabajo, proponemos utilizar algunos sistemas caóticos como un punto de partida para diseñar PRNGs basadosz en congruencias no lineales. En la sección 2 se reporta una breve comparación entre PRNGs lineales y no lineales,. Ya que nuestro propósito es proponer generadores congruenciales no lineales de ciertos mapas caóticos. En la sección 3 revisamos algúns funtamentos teóricos sobre los TRNGs basasos en una mezcla de sistema dinámicos estadísticamente estables, se hará un énfasis en los mapas Renyi. En la sección 4 discutimos el enlace que existe entre la dinámica de los sistemas caóticos y sistemas pseudo caóticos: para explicar cómo las dos dinámicas son relacionadas es necesario utilizar algunos resultados logrados con la Teoría Ergódica (esto es válido para los sistemas caóticos) sobre el múndo del pseudo caos discretizado. Hemos propuesto una interpretación más general y débil de la Teoría Shadowing propuesta por Coomes et al., enfocándonos en las medida de probabilidad, en vez de sólo una simple

trayectoria caótica. En la sección 5 estudiamos cómo discretizar mapas Renyis, se discute cómo encontrar un periodo de mínima longitud para las trayectorias discretizadas. En la sección 6 presentamos dos métodos alternativos para el diseño de un PRNG basado en recurrencias no lineales derivadas del mapa Renyi.

3.2. Generadores congruenciales lineales vs. no lineales

Los sistemas criptográficos convencionales están basados en máquinas de estado finito, y el problema de generar números aleatorios puede ser analizado al hacer referencia a subconjuntos finitos de enteros. De acuerdo a lo anterior, sea $\Lambda_M = (0, ..., M)$ el conjunto de los primeros $(M + 1)$ enteros no negativos. Definimos la j-tupla $k_0, ..., k_{j-1} \in \Lambda_M$ como la semilla inicial de el generador, y definimos un generador congruencial como un método iterativo que genera la secuencia $\{k_i \in \Lambda_M, i \in \mathbb{N}\}$, donde

$$k_n = G(k_{n-1}, ..., k_{n-j}) \mod M, \quad n > j, \tag{3.1}$$

para una cierta función $G : \Lambda_M^J \longrightarrow \mathbb{N}$. El generador congruencial es llamado lineal si la función G es una combinación lineal de los j números previos de la secuencia (con coeficientes en Λ_M), de otro modo, se dice que el generador tiene un comportamiento no lineal. El ejemplo más simple de un generador lineal de la forma (3.1) es el generador congruencial lineal (LCG) $k_n = ak_{n-1} + c \mod M$, mientras que para el Linear Feedback Shift Register (LFSR) con el polinomio $x^3 + x + 1$ se tiene que $M = 2$, $\Lambda_2 = 0, 1$ y $G(k_{n-1}, k_{n-2}, k_{n-3}) = k_{n-1} + k_{n-3}$.Ejemplos de generadores congruenciales no lineales son el Nonlinear Feedback Shift Registers (NLFSRs) y el Generador Polinomial Congruencial en el cual $G(k_{n-1}) = a_p k_{n-1}^p + a_{p-1} k_{n-1}^{p-1} + ... + a_0$. Se mostrará de una manera alternativa qu la función G puede ser obtenida al discretizar un mapa caótico.

Independientemente de la linealidad de G, un generador con memoria finita como en el caso de (3.1), puede ser implementado en una máquina de estado finita, siendo el estado de la máquina en un tiempo n la j-tupla $\sigma_n = (k_{n-1}, ..., k_{n-j}) \in \Sigma = \Lambda_n^j$. Como la cardinalidad de Σ es finita y debido a la naturaleza determinista de la evolución de la máquina, para cualquier semilla inicial $\sigma_0 \in \Sigma$ la secuencia $\{\sigma_i, i \in \mathbb{N}\}$, se tiene un periodo $\mu(\sigma_0)$, y se llega al periodo después de $\eta(\sigma_0)$ pasos. Un generador que para algún σ_0 genera una secuencia con un periodo igual a la cardinalidad de Σ es conocimo como un generador de ciclo máximo.

ller el trece, falta leer mas de eesta pppppppppppppppppppppppparteeeeeeeeeeeeeeeeeee

3.3. Statistically Stable Mixing Systems

3.3.1. Generador verdadero de números aleatorios con Mapas Renyi

COnsideremos un caso especial de mapas PWAE: las transformaciones de la familia Renyi

$$S_\beta(x) = \beta x \mod 1, \quad \beta > 1, \quad \beta \in \mathbb{R}, \tag{3.2}$$

donse se asume que el operador módulo se extiende hacia los números reales: $\beta x \mod 1 = \beta x - \lfloor \beta x \rfloor$. A partir de ahora, será $b = \lfloor \beta \rfloor$ la parte entera de β, mientras que $\gamma = \beta \mod 1$ será la parte fraccionaria. Si el parámetro β asume valores enteros $b \in \mathbb{N}$ (esto significa que $\gamma = 0$) el mapa Renyi tiene de acuerto a [1]

Capítulo 4

Transmisión de datos

4.1. Sistema de Empaquetado RTP

RTP es la abreviación de *Real-Time Transport Protocol*, por su denominación en inglés. Es un protocolo de nivel de sesión utilizado para la transmisión de información en timpo real, como por ejemplo, audio y video en una video conferencia. En aplicaciones de **Voz sobre IP**, **RTP** es el protocolo responsable de la transmisión de los datos. Está desarrollado por el grupo de trabajo de transporte de audio y video del **IETF**.

4.1.1. Características generales y estructura

Aunque RTP tiene algunas características de protocolo de nivel de transporte (Según el modelo OSI), es transportado usando UDP, UDP no maneja sesiones ni mecanismos que garanticen la recepción de los paquentes, pero es usado por RTP en lugar de TCP debido a que reduce el tiempo de envío de los paquetes a través de la red. En aplicaciones de voz y video es más importante una transmisión rápida que la pérdida de algunos paquetes durante el recorrido.

La figura 4.1 muestra la estructura del encabezado de un paquete de tipo RTP.

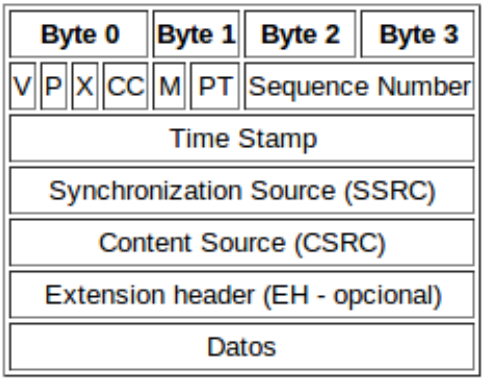

Figura 4.1: Encabezado de paquete RTP.

- Número de versión de RTP (V - versión number): 2 bits. La versión definida por la especificación actual es 2.
- Relleno (P - Padding): 1 bit. Si el bit del relleno está activado, hay uno o más bytes al final del

paquete que no es parte de la carga útil. El último byte del paquete indica el número de bytes de relleno. El relleno es usado por algunos algoritmos de cifrado.

- La extensión (X - Extensión): 1 bit. Si el bit de extensión está activado, entonces el encabezado fijo es seguido por una extensión del encabezado. Este mecanismo de la extensión posibilita implementaciones para añadir información al encabezado RTP.
- Conteo CSRC (CC): 4 bits. El número de identificadores CSRC que sigue el encabezado fijo. Si la cuenta CSRC es cero, entonces la fuente de sincronización es la fuente de la carga útil.
- El marcador (M - Marker): 1 bit. Un bit de marcador definido por el perfil particular de media.
- Tipo de Carga útil (PT - Payload Type): 7 bits. Un índice en una tabla del perfiles de media que describe el formato de carga útil. Los mapeos de carga útil para audio y vídeo están especificados en el RFC 1890.
- El número de Secuencia: 16 bits. Un único número de paquete que identifica la posición de este en la secuencia de paquetes. El número del paquete es incrementado en uno para cada paquete enviado.
- Sellado de tiempo: 32 bits. Refleja el instante de muestreo del primer byte en la carga útil. Varios paquetes consecutivos pueden tener el mismo sellado si son lógicamente generados en el mismo tiempo - por ejemplo, si son todo parte del mismo frame de vídeo.
- SSRC: 32 bits. Identifica la fuente de sincronización. Si la cuenta CSRC es cero, entonces la fuente de carga útil es la fuente de sincronización. Si la cuenta CSRC es distinta a cero, entonces el SSRC identifica el mixer(mezclador).
- CSRC: 32 bits cada uno. Identifica las fuentes contribuyentes para la carga útil. El número de fuentes contribuyentes está indicado por el campo de la cuenta CSRC; Allí puede haber más de 16 fuentes contribuyentes. Si hay fuentes contribuyentes múltiples, entonces la carga útil son los datos mezclados de esas fuentes.
- EH: El tamaño de este dato debe ser CC*32 en bits.
- Datos: El tamaño de los datos debe ser de X *((EHL+1)*32) donde EHL es la longitud de la extensión del la cabecera en unidades de 32 bits.

`http://en.wikipedia.org/wiki/Real-time_Transport_Protocol`

`http://es.wikipedia.org/wiki/User_Datagram_Protocol`

4.2. Códigos de longitud variable

Capítulo 5

Resultados

5.1. Sistema de cifrado

5.1.1. Inversión de Bits

Una vez que el sistema de cifrado recibe un paquete RTP, se dealiza uno de los pasos que conforman el algoritmo propuesto: diffuse and destroy the meaning of compressed codeword sequence and meke it impractial to predict the original codeword sequence. Para lograr esto, se propone la inversión da al menos un bit cada $BF = F \cdot Av \cdot MEPS$ bits, donde Av es el tamaño promedio de los códigos de Huffman, MEPL is the calculated Mean Average Propagation Length (MEPL) in codeword units, $Av \cdot MEPL$ es el promedio del error de propagación en bits, y f is a tunable security factor with values $\frac{1}{MEPL} \leq f \leq \frac{3}{4}$. Para este rango de f, $Av \leq BF \leq Av \cdot MEPL$ esto es, al menos un bit es invertido per average Huffman codeword size (Av) up to $\frac{3}{4}(Av \cdot MEPL)$ bits. Esto es lo que hace nuestro system scalable secured, as f gets smaller more bits are flipoed per BF-bits units increasing de system security. Depending of the specific needs of the user, f provides a tradeoff between security and performance.

The actual locariton B_i del bit a ser invertido en el flujo es calculado como sigue:

$$B_i = (rand\lceil \log_2(f \cdot Av \cdot MEPL) \rceil \mod BF) + 1 \tag{5.1}$$

donde $rand(Numero_b its)$ is our own designed PRNG function describes in the previous section. EL argumento $\lceil \log_2(f \cdot Av \cdot MEPL) \rceil$ representa la longitud de bits of the requested random number with límites $1 \leq B_i \leq BF$.

The algoritm for random bit flipping in the payload P of size P_s bytes es outlined in the following:

PONGO ALGORITMO CON LA LIBRERIA ALGORITM2e

Ahora, para lograr una mayor robustez en este paso, es importante tomar en cuenta las siguientes consideraciones:

1. Primer esquema. Nueva posición con referencia al inicio del bloque (gap can be considerable).

   La figura 4.1 muestra este esquema.

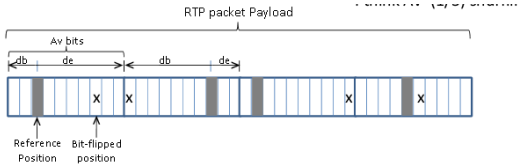

Figura 5.1: Esquema 1.

2. Segundo esquema. En esta alternativa, la nueva posici[on es elegida con respecto a el x previo. Esto asegura que cada uno de los bits invertidos no tienen una separación mayor de Av bits. El total de bits invertidos puede ser más alto que el promedio Av.

   La figura 4.1 muestra el segundo esquema.

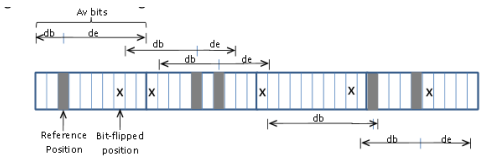

Figura 5.2: Esquema 2.

A continuación se hacen algunas especificaciones para la comprensión del proceso de inversión de bits.

a) **db** Es el número de bits que están antes del bit de referencia.

b) **de** Es el número de bits que están después del bit de referencia.

c) **Av** El el tamaño promedio de los códigos de HUffman. Este valor se calcula por medio de la siguiente expresión:

$$Av = \sum_{i=1}^{N} P(C_I) * L(C_i) \tag{5.2}$$

5.1.2. Segment shuffling

Después de que ha crecido el proceso de inversión de bits, ahora, es necesario permutar los RTP-packet payload by performing an L-way shuffling. Here the payload is divided into L segmentes which are schuffled using the algorithm described below and ilustrate in figure 5.3

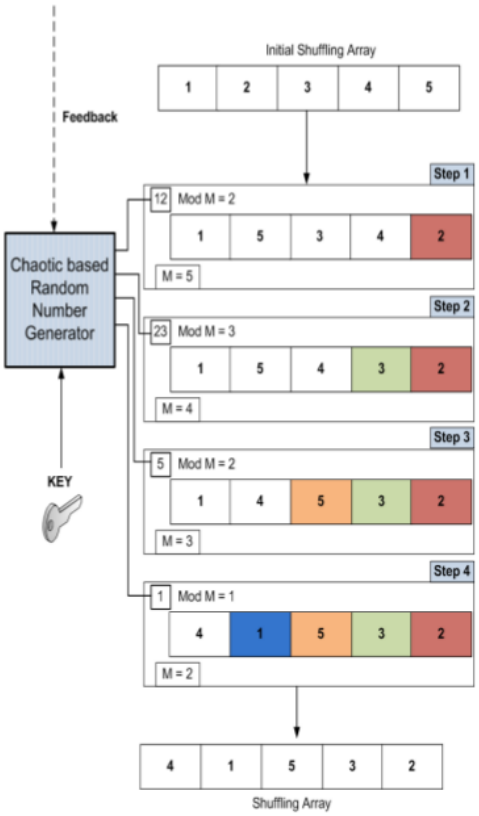

Figura 5.3: Esquema de permutacion.

5.1.3. Esquema robusto contra un ataque diferencial

EL problema con el esquema de inversión de bits es que el punto de referencia del cual el número aleatorio es utilizado para invertir el bit es conocido (de el primer bit invertido la referencia es el inicio del paquete y se incrementa en Av bits), así, la idea es la siguiente: (cambiar el bit cada $f * Av$ bits, $0 < f \leq 1$):

1. Elegimos una posición aleatoria ($Pos = randomChaoticNumber$ mód Av) de los primeros Av bits in el paquete y calcular la distancia de este punto al inicio y final del bloque (en bits), "db" y "de" respectivamente.

2. Si el número aleatorio es par: $flipbitPos = (randomNumber$ mód $db)$; por lo tanto $flipbitPos = Av - (randomNumber$ mód $de)$. Esto es, de la posición aleatoria lanzamos una moneda (par o impar) para decidir acerca de la dirección de la inversión de bits.

3. El siguiente punto puede tomar dos direcciones, o puede ocurrir una combinación de ambos.

   a) Ir hacia el siguiente block y repetir los pasos 1 y 2 (el inicio del siguiente blok es el nuevo punto globar de la referencia para el siguiente cálculo para la inversión del nuevo bit). EL inconveniente de este método es el espacio dejado entre bits invertidos, este puede ser más bien que Av (pero como un promedio la tasa de bits invertidos es Av).

b) La posición del último bit invertido es el nuevo punto de referencia para el siguiente bit invertido, y se repiten los pasos 1 y 2. Para considerar la difusión del punto de referencia en la iteración actual se realiza el siguiente cálculo:

$$Pos = (Prev_randomChaoticNumber + new_randomChaoticNUmber) \quad \text{mód } Av \tag{5.3}$$

el siguiente proceso es el mismo. En este esquema, los bits invertidos se encuentran aparte por no más que Av bits, esto puede necesitar iteraciones adicionals al final

Anexos

Prueba de la Frecuencia

La prueba consiste en medir la proporción de ceros y unos de la secuencia analizada. Se busca determinar si el número de ceros y unos en una secuencia es aproximadamente el mismo como debería ser para una secuencia verdaderamente aleatoria. La prueba evalúa que la fracción de unos se acerce a $\frac{1}{2}$. Es importante recordar que todas las pruebas subsecuentes dependen del resultado de esta prueba.

Prueba de la Frecuencia por Bloques

Esta prueba mide la proporción de unos en un bloque de M bits, se busca determinar si la frecuencia de unos en el bloque es aproximadamente de $\frac{M}{2}$, que es el resultado esperado bajo la suposición de aleatoriedad.

Prueba de las Corridas

La proueba mide el número total de corridas en la secuencia, donde una corrida es una secuencia nininterrumpida de bits idénticos. Una corrida de longitud k consiste de k bits idénticos que está reada al inicio y al final con un bit del valor opuesto. Con esta prueba se busca demostrar si el número de coridad de ceros y unos de diversas longitudes es el que se espera para una secuencia aleatoria.

Prueba de la Corrida más Larga en un Bloque

EL propósito de esta prueba es determinar si la longitud de la corrida más grande de unos en la secuencia probada es consistente con la longitud de la corrida más grande de unos que debería ser esperada en una secuencia aleatoria.

Prueba Binary Matrix Rank Test

Esta prueba consiste en revisar la dependencia lineal entre subcadenas de un tamaño fijo de la secuencia original, esto se logra mediante el siguiente procedimiento: se construyen matrices de fragmentos sucesivos de la secuencia generada, y se revida la dependencia lineal de las filas o columnas de las matrices creadas. La desviación del rango, o deficiencia del rango, de las matrices de un valor esperado teóricamente es lo que genera el estadístico de interés.

Prueba (espectral) de la Transformada Discreta de Fourier

Esta prueba está basada en la transformada Discreta de Fourier. Se detectan posibles caracterísitcas de la serie de bits que deberían indicar una desviación de la suposición de aleatoriedad.

Sea x_k el $k-$ésimo bit, donde $k = 0, ..., n-1$. Supongamos una codificación para los bits de -1 para el bit con valor cero y de 1 para el bit uno. Utilizamos la siguiente expresión:

$$f_j = \sum_{k=0}^{n-1} x_k \exp\left(2\pi kj/n\right) \tag{5.4}$$

donde $\exp(2\pi kj/n) = \cos(2\pi kj/n) + \imath \sin(2\pi kj/n)$, $j = 0, ..., n-1$, además $i = \sqrt{-1}$. Debido a la simetría de los valores de la transformada, únicamente los valores de 0 a $(n/2-1)$ son considerados. Ahora tomamos el módulo del número complejo f_j y lo denotaremos por mód j; baso la suposición de aleatoriedad de la serie x_j, se puede dar un intervalo de confianza para los valores de mód j. Más específicamente, 95 % de los valores de mód j deberían ser menores que

$$h = \sqrt{\left(\log \frac{1}{0{,}05} n\right)}. \tag{5.5}$$

Un $p - valor$ basado en este umbral surge de la distribución binomial. Se N_l el número de puntos más altos menores que h, únicamente los primeros $\frac{n}{2}$ son considerados. Sea $N_0 = {,}95N/2$ Y $d = (N_1 - N_0)/\sqrt{n(0{,}95)(0{,}05)/4}$, el $p - valor$ es

$$2\left(1 - \phi\left(|d|\ \right)\right) = erfc\left(\frac{|d|}{\sqrt{2}}\right) \tag{5.6}$$

Prueba de la Frecuencia

Prueba de la Frecuencia

Bibliografía

[1] A. Fort A. Pasini S. Rocchi T. Addabbo, M. Alioto and V. Vignoli. A class of maximum-periodo nonlinear congruential generators derived from the rényi chaotic map. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS*, 54(4):816–828, 2007.