CONSTRUCCIÓN DE UN CIFRADOR BASADO EN UNA PERMUTACIÓN PSEUDO-ALEATORIA

Hernando Castañeda Marín
Universidad de Pamplona, Estudiante Doctorado Universidad de los Andes,
Mérida, Venezuela, 0058
hcastaneda@ula.ve
y
Wladimir Rodríguez Graterol
Universidad de los Andes, Doctorado en Ciencias Aplicadas,
Mérida, Venezuela, 0058
wladimir@ula.ve

Abstract

The purpose of this paper is the analysis and development of as practical implementation of a cipher based on the schema proposed by Even and Mansour[1]. The schema corresponds to a block cipher based in the use of a permutation in the encryption operation and in the corresponding inverse function for the decryption process. The key K that consist in two sub keys K_1 and K_2, which are produce by means of a pseudorandom numbers generator with unpredictable inputs and pseudorandom output sequences. To measure the reliability of the permutation of the resulting sequence of the previous XOR operation, hypothesis test were done using the statistic χ^2 for different combinations of the { 0,1 } patterns referenced in those sequences. In order to reach this purpose the Yarrow-160, a visual C++ software from Counterpane System, computational tool was used; extended with the necessary encryption and decryption routines for the proposed cryptosystem based on the Even and Mansour schema. Also a routine was added, that allows doing statistical test by determining the computed values of χ^2 for the different combinatorial patterns of the sequences. This work is an academicals practice to test the strengths and debilities of the Counterpane System software and allows to extend it by means of the construction of a cipher based on a simple permutation that, as practical implementation allows the test of its computational complexity and security. The realization of a cryptanalysis and the realization of different cryptographic attacks will made possible future works in the field of cryptographic and computational security.
Keywords: Cryptographic, Computational Security.

1. INTRODUCCION

Para diseñar un sistema de seguridad se tiene que seleccionar los servicios de seguridad que hayan sido previstos y luego seleccionar los mecanismos que se implementarán por ejemplo: La encriptación, las firmas digitales, el control de acceso, la integridad de los datos y la autenticación.

La encriptación es una función matemática, los algoritmos de encriptación deben tener la propiedad de que los datos originales puedan ser recuperados a partir de los datos encriptados, si el valor de la clave utilizada es bien conocido. Las entradas al algoritmo de encriptación son referidas como texto plano, la salida del algoritmo es denominada texto cifrado o criptograma.

Las claves deben ser mantenidas en secreto, el criptograma estará comprometido si un intruso puede deducir la clave analizando la información pública disponible. La fortaleza de un criptosistema es medida por la dificultad, usualmente medido en el número de operaciones elementales, para determinar dicha clave.

Hay dos clases de criptosistemas, simétricos y asimétricos, llamados también de encriptación pública. Un criptosistema simétrico utiliza la misma clave en ambos extremos de un canal de comunicación, o una clave que es fácilmente derivada de otra, en caso de hacer uso de dos claves. En los criptosistemas asimétricos cada usuario tiene dos claves, una pública y una privada, la cual no es revelada.

La encriptación presenta algunas limitaciones prácticas visibles como son: Los mensajes de entrada a los algoritmos no cuentan con protección, el intercambio de claves, se realiza sobre los mismos canales en que se transmiten los datos, la encriptación no provee protección contra las modificaciones, esta puede ser detectada incluyendo como parte de los datos encriptados un número de bits de chequeo.

La metodología empleada durante la investigación cumplió con los siguientes pasos: Una revisión bibliografica exhaustiva que terminó con una sustentación publica del anteproyecto, una revisión bibliográfica del artículo de Even y Mansour [1] y una revisión computacional de criptosistemas y generadores de números seudo-aleatorios como requisitos previos para una sustentación pública de un avance de tesis; en la fase final se desarrollo el criptosistema enriqueciendo el software del PRNG Yarrow-160 y la aplicación de prueba estadística con el afinamiento del presente informe final.

2. EL CIFRADOR DE EVEN Y MANSOUR

En [1] Even y Mansour proponen la construcción de un cifrador de bloques usando únicamente una permutación seleccionada en forma aleatoria.

El propósito del presente trabajo es desarrollar y analizar una implementación práctica del cifrador de Even y Mansour. Adicionalmente los autores asumen que la permutación se supone que es de acceso público (Como una caja negra), y que cualquiera que desee atacar el cifrador tiene acceso a ella. Los autores "prueban" la seguridad del cifrador propuesto bajo la suposición de que la permutación es aleatoria, o al menos seudo- aleatoria, y que la única vía de accesar a través de una caja negra.

La figura 1 ilustra el esquema de Even y Mansour donde M =mensaje, F =permutación $\oplus$ = XOR (bit a bit) $K = \{k_1, k_2\}$ =Clave, C = criptograma.

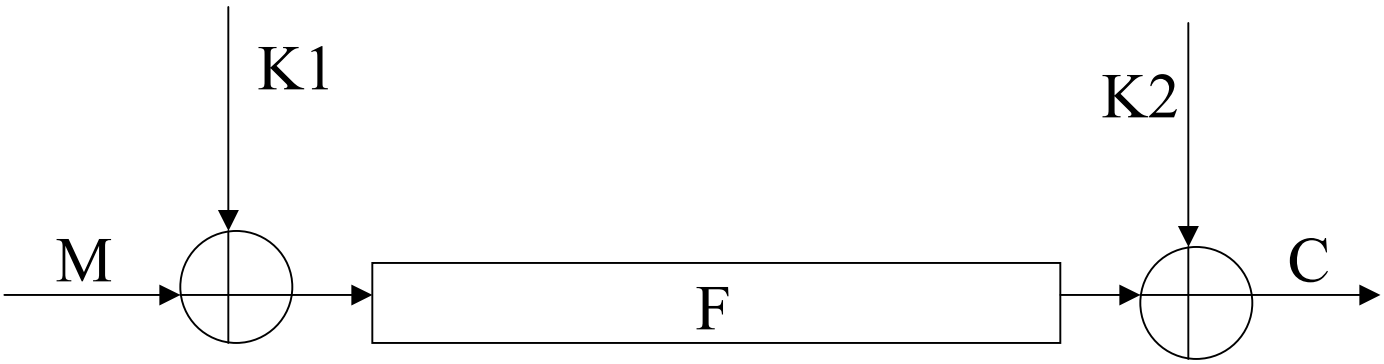

Figura 1. Esquema de Even Y Mansour.

Se asume que M y C son palabras binarias de longitud n, esto es que M y C son elementos del conjunto
También se asume que $F(x)$ es una permutación de $\{0,1\}^n$, y se denota su inversa por $F^{-1}(x)$, y que es "fácil" computar $F(x)$ o, $F^{-1}(x)$, ya sea directamente o usando una caja negra, la cual se denominara "oráculo".

La clave K consiste en dos subclaves k_1 y, k_2 cada una seleccionada en forma aleatoria del conjunto $\{0,1\}^n$.

Se asume que K es conocida solamente por las partes autorizadas, esto es el emisor y el receptor, y que esta es usada para encriptar mensajes por un largo periodo de tiempo.
El proceso de encriptación puede ser descrito por la siguiente ecuación:

$C = E_K(M) = F(M \oplus K_1) \oplus K_2$, Donde C es el criptograma correspondiente a $\{0,1\}^n$, y $\oplus$ denota la operación XOR (OR exclusivo), bit a bit. El proceso de descifrado puede ser descrito por la siguiente ecuación:

$$M = D_K(C) = F^{-1}(C \oplus K_2) \oplus K_1$$

2.1 DEFINICIONES DE SEGURIDAD Y SUS RELACIONES

En su artículo [1], Even y Mansour modelan al "adversario" como alguien que puede apoderarse por un tiempo determinado del cifrador para cifrar y descifrar mensajes escogidos por él puede usar las permutaciones F y F^{-1}, como cajas negras pero que no tiene acceso a la clave K. Even y Mansour modelan la situación anterior usando oráculos, los cuales responden preguntas de naturaleza particular.
El "problema del rompimiento "CP, es definido como un intento (por un adversario) de descifrar un criptograma dado $C_0 = E_k(M_0)$, sin un conocimiento de la clave K. Se asume que el algoritmo usado por el adversario tiene acceso a los siguientes oráculos:

1. El $F - oráculo$: Dado $x \in \{0,1\}^n$, el oráculo suministra $F(x)$.
2. El $F^{-1} - oráculo$: Dado $x \in \{0,1\}^n$, el oráculo suministra $F^{-1}(x)$.
3. El $E - oráculo$: Dado $M \in \{0,1\}^n$, el oráculo suministra $E_k(M)$.
4. El $D - oráculo \;\; Co - restringuido$: Dado $C \in \{0,1\}^n$, tal que $C \neq C_0$, el oráculo suministra $D_K(C)$.

En otras palabras se supone que el adversario tiene acceso a una colección arbitrariamente grande de pares (M,C), donde C es el criptograma correspondiente a M, generados usando la clave K, la cual no es conocida por el adversario y además, que la permutación F es públicamente accesible como una caja negra.
El algoritmo del adversario es exitoso si éste produce $M_0 = D_k(C_0)$. La probabilidad de éxito del algoritmo es la probabilidad de que sea escogida en forma aleatoria C, (con distribución uniforme) y el algoritmo produzca M_0.

El problema del rompimiento es también conocido como $ataquepormensaje \big/ criptogramaescogidos.$

Ahora se considera el problema de la falsificación existencial, EFP.
El adversario tiene acceso a cuatro oráculos: el $F - oráculo$, $F^{-1} - oráculo$, $E - oráculo$, $D - oráculo$ no restringido. Este último es definido como sigue: dado cualquier $C \in \{0,1\}^n$, el oráculo suministra $D_k(C)$.

El objetivo del adversario, en el problema EFP, es encontrar un nuevo par (M', C'), con $C^? = E_K(M')$., De esta forma el adversario puede producir un criptograma C' valido para mensajes con "sentido" $M' \neq M_j$ sin conocer K. Así el adversario puede enviar mensajes cifrados como si fuera un usuario autorizado.

Definición: sea f una función de los enteros positivos hacia $[0,1]$.

Se dice que f es de forma polinomial despreciable si para todo polinomio hay un n_0, tal que si $n \succ n_0$ entonces $f(n) \prec \frac{1}{p(n)}$. [1]

[1] Comparación asintótica entre dos funciones en el intervalo $[0,1]$ [5]

Se asume que el adversario emplea un "algoritmo randómico",[2] cuyo tiempo de corrida es de grado polinomial acotado en n, para resolver los problemas CP y EFP.

Se dice que un problema es duro, si, para todo algoritmo randómico, la probabilidad de éxito es polinomialmente despreciable. La probabilidad es tomada sobre todas las escogencias hechas en el diseño del cifrador (o sistema) , esto es las escogencias de los F , las claves escogidas por el usuario, y los lanzamientos de moneda realizados en el algoritmo del adversario.

Even y Mansour reducen EFP a CP, esto es, ellos muestran que la existencia de un ataque exitoso para el problema CP implica la existencia de un ataque exitoso para el problema EFP.

Teorema [Corolario 3.2] Si todo algoritmo, con tiempo de corrida polinomial, para el problema EFP tiene una probabilidad de éxito despreciable, entonces todo algoritmo, con tiempo de corrida polinomial, para el problema CP tiene una probabilidad de éxito despreciable.

El resultado principal del artículo de Even y Mansour es el siguiente:

Teorema $[1, Teorema\ 4.4]$ La probabilidad de que un algoritmo randómico A resuelva el problema EFP, cuando F y K son escogidas aleatoria y uniformemente, está acotada por:

$O(\frac{lm}{2^n})$ Donde l es él número de preguntas a los oráculos $\frac{E}{D}$, y m es el número de preguntas a los oráculos $\frac{F}{F^{-1}}$.

Como corolario del resultado anterior se obtiene lo siguiente:

Corolario Si F es escogida en forma aleatoria, entonces todo algoritmo randómico con tiempo de corrida polinomial, tiene una probabilidad de éxito despreciable. Así desde un punto de vista teórico, la probabilidad de "romper" el cifrador de Even y Mansour es despreciable, si F es escogida aleatoriamente.

Si La Familia de permutaciones F son seleccionadas seudo-aleatoriamente y aunque cualquier adversario tiene acceso a los cuatro oráculos, se garantiza que éste no llegará a las entrañas de la caja que implementa F.

La anterior aserción justifica el siguiente teorema $[2, Teorema\ 5.1]$ Si F es seleccionado en forma aleatoria y k es seleccionado uniformemente, entonces para todo algoritmo limitado en grado polinomial para solucionar el problema EFP, la probabilidad del evento es de grado polinomial limitado

2.3 VENTAJAS EN SU CONSTRUCCIÓN

Los análisis realizados por Even y Manssour [1] y confirmadas por Daemen [4] acerca de las características sobresalientes en la construcción de su cifrador de bloque y que específicamente son: la consideración establecida para el operador de encriptación interno F el cual es fijo y público; y la formalización de su esquema de encriptación. Adicionalmente la demostración sobre la efectiva longitud de la clave $n - lg\ l - lg\ m$ en donde al adversario se le permite realizar l llamadas al oráculo de encriptación y desencriptación y m llamadas al oráculo de la permutación $F\ o\ F^{-1}$.

Como ejemplo de motivación de esta implementación está la construcción de DESX introducida por Rivest y que permite proveer información económica acerca de la clave en DES.

[2] Los algoritmos aleatorios se clasifican de acuerdo a la probabilidad de que retornen una respuesta correcta Pág. 62 [5]

3. CONSTRUCCIÓN DEL CIFRADOR

Inicialmente se explicarán los mecanismos del generador de números seudo-aleatorios PRNG pero el propósito de utilizar Yarrow-160 para generar las claves se justifican por ser una utilidad para la acumulación de entropía.

Un número aleatorio es un número que no puede ser predecible para un observador antes que éste sea generado. Si un número está en el rango $0...2^{n-1}$, un observador no puede predecir el número con probabilidad superior a $1/2^n$.

Un PRNG contiene un estado interno que es usado en determinados instantes para generar salidas seudo-aleatorios. Este estado guarda confidencia y controla significativamente los procedimientos.

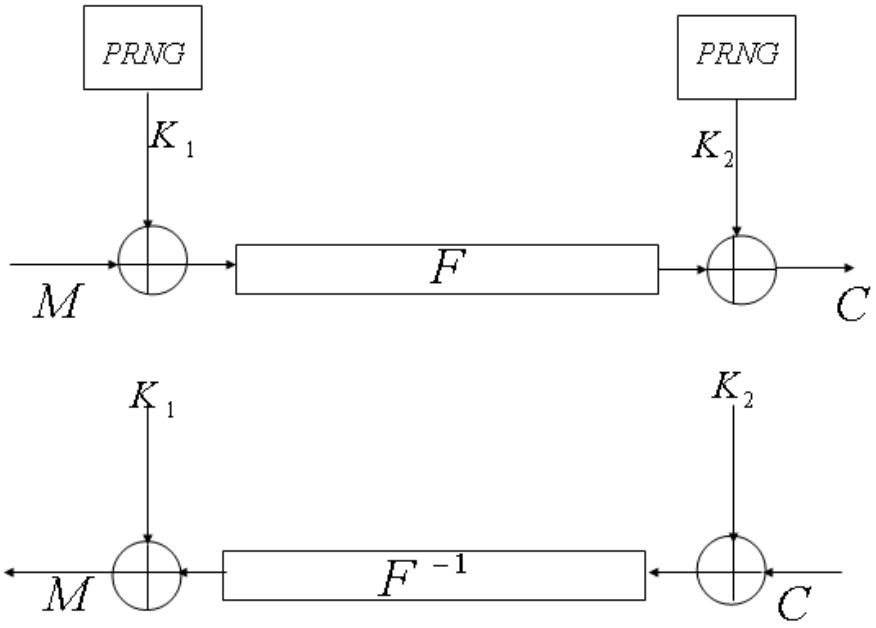

Figura 2. Esquema de implementación

Donde M =mensaje, F =permutación $\oplus$ = XOR (bit a bit),PRNG generador de números pseudo-aleatorios, $K = \{k_1, k_2\}$ =Clave, C = criptograma.

3.1 HERRAMIENTA CRIPTOGRÁFICA

Un generador de números seudo-aleatorios es un mecanismo criptográfico para procesar entradas impredecibles y generar salidas seudo-aleatorias. Si se diseña, implementa y utiliza apropiadamente, cualquier adversario con enormes recursos computacionales no será capaz de predecir una secuencia de la salida del PRNG.

Un adversario puede comprometer el estado interno mediante el uso de un modelo muy bueno que produzca valores impredecibles, utilizando las muestras de entrada del PRNG y con un gran manejo de poder computacional, trate de adivinar el estado interno del PRNG.

El PRNG posee cuatro componentes principales:

1. Un acumulador de entropía que colecciona muestras desde las fuentes de entropía y las almacena en dos piscinas.

2. Un mecanismo de resiembra el cual sirve periódicamente de semilla en la generación de la clave, con nueva entropía en las piscinas. El proceso de combinar la clave existente y las nuevas muestras dentro de una nueva clave, recibe el nombre de resemillar.
3. El mecanismo de generación que genera salidas de claves del PRNG.
4. El control de resiembra, el cual se determina cuando una resiembra sé esta realizando.

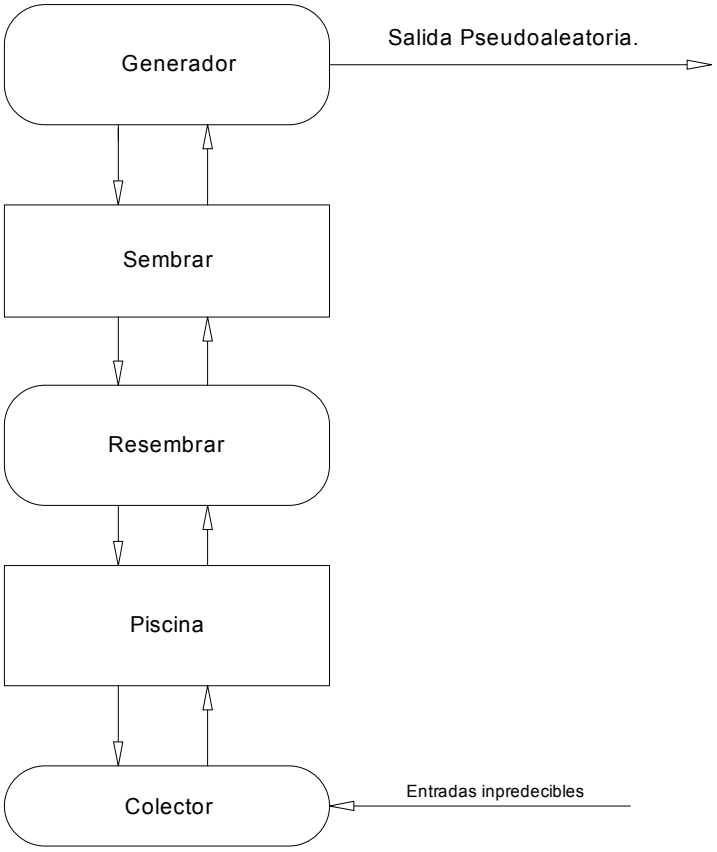

Figura 3 Generalización de un PRNG con resembrado periódico

A continuación se especifica el rol de cada componente en el diseño del PRNG y los requerimientos para cada componente en términos tanto de seguridad como de desempeño. También se especifica la forma como cada componente puede interactuar con los otros.

3.1.1 ACUMULADOR DE ENTROPÍA (COLECTOR).

La acumulación de entropía es el proceso de inicialización por el cual el PRNG adquiere un nuevo estado interno no adivinable. Durante la resiembra es indispensable que la entropía se acumule completamente desde las muestras, con el fin de evitar los ataques iterativos de adivinación [6]. Es importante también que se estime correctamente la cantidad de entropía que se tiene que coleccionar y ésta tiene que ser muy grande. El mecanismo de acumulación de entropía puede también resistir el ataque de selección de entrada [6],

En Yarrow, la entropía de las muestras es coleccionada en dos piscinas, cada una en contexto de mezcla. Las dos piscinas son una piscina rápida y una piscina lenta. La piscina rápida proporciona resiembras frecuentes de la clave, con el propósito de asegurar en lo posible que la clave comprometida tenga una corta duración y la medición de la precisión de los estimativos de la entropía de cada fuente. La piscina lenta proporciona pocas y conservativas resiembras.

3.1.2 MECANISMO DE RE-SIEMBRA.

El mecanismo de resiembra conecta al acumulador de entropía con el mecanismo de generación, esto es posible cuando el control de resembrado determina que esto se requiere, entonces el componente de resembrado modifica la clave y lo hace utilizando el mecanismo de generación, con información de una o ambas piscinas y que son actualizadas por el acumulador de entropía, de tal forma que el objetivo sea que la clave o la piscina sean desconocida por el adversario, después del resembrado.

El mecanismo de resiembra genera una nueva clave K para el generador desde la piscina del acumulador de entropía y la clave existente. El tiempo de ejecución del mecanismo de resiembra depende de un parámetro $P_t > 0$. Este parámetro puede estar fijado para la implementación o ser dinámicamente ajustable.

El proceso de resiembra consta de las siguientes etapas:

1. El acumulador de entropía computa el hash sobre la concatenación de todas las entradas dentro de la piscina rápida, El resultado se llama v_0
2. Se asigna $v_i := h(v_{i-1} \mid v_0 \mid i) para\ i = 1,...,t.$
3. Se asigna $K \longleftarrow h'(h(v_{P_t} \mid K), k).$
4. Se asigna $C \leftarrow \mathrm{E_k}(0)$.
5. Inicializa a cero todos los acumuladores de estimativos de entropía en el acumulador de entropía.
6. Limpia la memoria de todos los valores intermedios.
7. Si un archivo de semilla está en uso, los próximos $2k$ bits de salida del generador están escritos para el archivo de semilla, entonces sé sobre-escribe el valor presente.

En la etapa 1 se recolecta la salida del acumulador de entropía. En la etapa 2 se utiliza una fórmula iterativa de longitud P_t y se hace la resiembra, que es computacionalmente costosa. En la etapa 3 se utiliza una función hash h y una función h', la cual crea una nueva clave K a partir de la actualizada y un nuevo valor de entropía. La etapa 4 define un nuevo valor para el contador.
La función h' esta definida en términos de h. Para computar $h'(m,k)$ se construye así:

$$s_o = m$$
$$s_i = h(s_0 \mid ... \mid s_{i-1})\ i = 1,...$$
$$h'(m,k) := first\ \mathrm{k\ bits\ of}\ (\mathrm{s}_0 \mid s_1 \mid ...)$$

Esto es efectivamente una función conocida como "adaptador de tamaño" que convierte una entrada de cualquier longitud en una salida de longitud específica.

3.1.3 MECANISMO DE GENERACIÓN.

Se tiene un valor C de N-bits, para generar los próximos bloques se incrementa C y se cifra usando la clave, Para generar el próximo bloque de salida, se ejecutan las siguiente ecuaciones:

$$C \longleftarrow (C+1) \bmod 2^n$$
$$R \longleftarrow E_K(c)$$

Donde R es el próximo bloque de salida y K es la clave actualizada del PRNG.

Sí la clave está comprometida en cierto punto en el tiempo, el PRNG no debe producir lentamente muchos rendimientos de salida anteriores a las que fueron generadas antes del compromiso.

Es claro que este mecanismo de generación no es inherentemente resistente a una clase de ataque. Por esta razón se investiga cuántos bloques tienen rendimiento. Una vez que se alcanza algún límite P_g (parámetro de un sistema de seguridad $1 \le P_g \le 2^{n/3}$), se genera k bits de salida del PRNG y se usan estos como nueva clave.

$$k \longleftarrow \textit{proximo}\ \text{k bits de la salida del PRGN}$$

Se llama esta operación un generador de puerta. Nótese que todo no es una operación de resiembra y ninguna nueva entropía se introduce dentro de la clave.

Con el interés de guardar un diseño conservativo extremo, el máximo número de salidas del generador entre resiembras está limitado a $min\left(2^n, 2^{k/3} P_g\right)$ bloques de salida de n-bits. El primer término en el mínimo previene el valor de C en el ciclo.

El segundo término hace que sea extremadamente improbable que K repita el mismo valor dos veces. En la práctica P_g puede ser un conjunto más pequeño que éste, por ejemplo, en el orden de minimizar el número de salidas que pueden ser aprendidas por retro-alimentación.

3.1.4 CONTROL DE RESIEMBRA.

El mecanismo de control de sembrado se presenta teniendo en cuenta las siguientes consideraciones. Frecuentemente el resembrado es deseable, pero probablemente es vulnerable a un ataque de adivinación iterativo [6]. Un resembrado poco frecuente, permite actuar al adversario, siempre que tenga comprometida la clave con más información.

Existe control en los estimativos de entropía para cada fuente, así, como es que llegan las muestras a cada piscina. Cuando cualquier fuente en la piscina rápida ha pasado el valor del umbral, entonces se resiembra desde la piscina rápida. En muchos sistemas, se puede esperar que esto pase muchas veces en una hora.

Cuando cualquier K de las n fuentes tiene un alto umbral en la piscina lenta, entonces se resiembra desde la piscina lenta. Este es un proceso mucho más lento.

Para Yarrow –160, el umbral para la piscina rápida es de 100 bits y para la piscina lenta es de 160 bits. Al menos dos diferentes fuentes están sobre 160 bits en la piscina lenta antes de la resiembra en la piscina lenta[6]. (Esto se facilita para diferentes ambientes, los ambientes con tres fuentes de entropía rápida son buenos y razonables).

Él módulo de control de resiembra determina cuándo la resiembra está siendo realizada. Ocurre cuando alguna aplicación explícita pregunta por la operación de resiembra. Esto se usa raramente y sólo por aplicaciones que generan valores muy altos de aleatoriedad. El acceso a funciones de resiembra explícita puede ser restricto en muchos casos.

Las Re-siembras[3] periódicas ocurren automáticamente. La piscina rápida se usa para la resiembra, sin embargo cualquiera de estas fuentes tiene una entropía estimada sobre algún valor de umbral. La piscina lenta es usada para resembrar, sin embargo, al menos dos de las fuentes tienen estimativos de entropía antes que otros valores de umbral.

3.2 ESTIMACIÓN DE LA ENTROPÍA

La estimación de la entropía es el proceso de determinar, cuánto trabajo puede tomar un adversario para adivinar el contenido actual de las piscinas.
La implementación será costosa por la determinación de sus fuentes. Las fuentes no están cerradamente encadenadas, o exhiben cualquier correlación significativa.

La entropía de cada muestra es medida en tres formas:

[3]Entropía: magnitud que mide la información contenida en un flujo de datos, es decir, lo que nos aporta sobre un dato o hecho concreto [Wikipedia]

1. El programador suministra un estimativo de entropía en una muestra cuando escribe la rutina para coleccionar datos de la fuente. Así el programador puede enviar en una muestra, un estimativo de 20 bits de entropía.

2. Por cada fuente se utiliza un estimador estadístico de la muestra. Esta prueba se hace para descubrir situaciones anormales debido a que la muestra tiene una entropía muy baja.

3. Para determinar la máxima amplitud muestral, "densidad" se opera con la longitud de la muestra en bits y multiplicando por algún factor constante menor de uno, esto con el propósito de obtener un estimativo máximo de entropía en la muestra. En Yarrow-160 se usa un multiplicador de 0.5.

Se usa el valor más pequeño de estos tres estimativos como entropía de la muestra en cuestión.

3.3 GENERANDO SALIDAS SEUDO-ALEATORIAS

El mecanismo de generación puede tener las siguientes propiedades:

1. Resistencia al ataque cripto analítico.
2. Eficiente.
3. Resistente a la retroalimentación después de una clave comprometida
4. Capacidad de generar secuencias muy largas de rendimiento firme sin resembrar.

Hay dos puntos del plan principal que deben ser tenidos en cuenta con este generador seudo-aleatorio. El primero tiene en cuenta tanto la criptografía, como la calidad de sus entradas por su seguridad y calidad de los datos. El segundo consiste en que el mecanismo del rendimiento y piscinas de entropía son tan distintos como es posible.

4. FUNCIONAMIENTO DEL CRIPTOSISTEMA

4.1 Ejemplo que ilustra el proceso de cifrado y descifrado.

En el próximo párrafo se quiere realizar un ejemplo hipotético que demuestra como funcionan los procesos de cifrar y descifrar bloques del archivo de texto o del texto cifrado respectivamente. Esto es simplemente la abstracción del esquema propuesto por Even y Mansour [1] que es la base de la implementación pero enriquecido con la Funcionalidad PRNG para generar las claves.

4.1.1. PROCESO DE CIFRADO.

Se tiene en cuenta la expresión matemática que formaliza este proceso

y teniendo en cuenta lo anterior se establecen tamaños hipotéticos para la clave y el bloque y se generan secuencias de dicho tamaño para las posibles claves o sea k_1 y k_2.

Se considera un tamaño de bloque por ejemplo de 8 bits sea $b_1 = 10110111$ correspondiente al bloque texto y el PRNG genera una clave también de 8 bits por ejemplo $k_1 = 11101011$.

Sea $b_1 \oplus k_1 = 10110111 \oplus 11101011 = 01011100$. A partir de esta operación se seleccionan en forma aleatoria dos números diferentes, en este caso entre 1 y 8 por ejemplo que sean 2 y 7, se intercambian los bits de las posiciones 2 y 7, de la secuencia 01011100 Esto permite producir una de sus permutaciones, que en este caso es $P_1 = 00011110$. Ahora, utilizando PRNG, generamos una nueva clave, por ejemplo, $k_2 = 10101010$ .Sea $p_1 \oplus k_2 = 00011110 \oplus 10101010 = 10110100$ que corresponde al bloque cifrado. La clave compuesta para el proceso de descifrar es $k_1 = 11101011$ y $k_2 = 10101010$ y los índices de la permutación, buscando como medio el uso de un canal seguro.

4.1.2 PROCESO DESCIFRADO.

La descripción del proceso de descifrado del algoritmo criptográfico es sencilla. Consiste en sustituir las transformaciones utilizadas en el cifrado por sus inversas e invertir el orden de aplicación de dichas transformaciones o funciones matemáticas.

Teniendo en cuenta esto el proceso de descifrado es descrita, por la siguiente ecuación: El bloque cifrado es $Q_1 = 10110100$ ahora con la clave del proceso anterior $k_2 = 10101010$ se opera $Q_1 \oplus k_2 = 10110100 \oplus 10101010 = 00011110$, ahora se opera con la permutación inversa intercambiando los bits de la posición 2 y 7 de la secuencia 00011110 se produce $p_2 = 01011100$, ahora operando en el ámbito de bit se llega a $p_2 \oplus k_1 = 01011100 \oplus 11101011 = 10110111$.

Se puede observar que se llega a la misma secuencia con la que se inicio el proceso de cifrar el bloque del archivo de texto.

El proceso experimental se realiza con bloques y claves de 120 bits.

4. RESULTADOS

El proceso investigativo del presente trabajo se inicia con el conocimiento del esquema propuesto por Even y Mansour, y fue necesario entender qué es un cifrador de bloque y cómo una sola permutación es seleccionada en forma aleatoria y utilizada, tanto en el proceso de cifrado como en el descifrado. Con base en los resultados, se puede ver inmediatamente, que la clave que modifica la permutación es muy sencilla y fácil de implementar.

Dentro del procedimiento de aplicación, se realiza a continuación la ejecución del programa que cifrara el archivo de entrada.
Para comprobar la terminación de este trabajo de grado, también se implementó el programa que permite descifrar el archivo presentado anteriormente, produciendo el siguiente resultado.

```
// File choreographer.kPp  (from Karel++ textbook
//  by Bergin,et.Al. see p. 56(
// Run this on file "harvest.Wld"

class CHoreographer : Ur_Robot
{
  □ ur_Robot Lisa(4,2, East, 0);  // the 1st heLper robot
   □ur_Robot Tony(6,2, East, 0);  // the wnd helper robot

    void harvEst();
    void harvestARow()□
    void harvestCorner();
    void move()□
    void pickBeeper();
   □void turnLeft(            ;
    void tuRnOff();
};

void ChoreograPher::harvest()
{
    harvestARow();
    tUrnLeft();
   □move();
    tUrnLeft();
   □harvestARow((;
}
```

```
task
{   ChoReographer KareL (2,2, East, 0        ;
    Karel.hArvest();
    Karel.turnOff()
}
```

Se puede observar que los archivos presentan características similares para un formato dado, pero no podemos afirmar que son dos copias exactamente iguales para todo tipo de formato de edición.

Finalmente, para demostrar la hipótesis sobre el uso de una sola permutación en el esquema propuesto y el uso de un generador de números seudo-aleatorios se realizó un programa que implementa cinco pruebas estadísticas que son utilizadas generalmente para determinar si una secuencia binaria, como son las permutaciones generadas en esta implementación posee características específicas de secuencias seudo-aleatorias verdaderas, como probablemente se ha declarado.

Estos valores para el estadístico χ^2 reciben el nombre de valores calculados; y comparándolos contra los valores críticos de χ^2 en la tabla 5.1 y 5.2 en [5] para secuencias de tamaño n=120, nos permitirá comprobar o rechazar la hipótesis de esta investigación.

En la tabla a continuación para cada prueba estadística muestra la formula del estadístico χ^2 con los respectivos grados de libertad y los valores calculados con el programa de test estadístico[ver apéndice D] y los respectivos valores críticos con un nivel de significancia $\alpha = 0.05$.

La tabla nos permite establecer que para la secuencia mostrada anteriormente las pruebas de frecuencia y la prueba serial y se quedan las pruebas poker, test y correlación.

Prueba estadística	Formula para chi-cuadrado	G.L	χ^2 calculada	χ^2 Estimada
Frecuencia	$X_1 = \frac{(n_0 - n_1)^2}{n} \quad n_0 = 59 \text{ y } n_1 = 61$	1	0.033	3.8415
Serial	$X_2 = \frac{4}{n-1}\left(n_{00}{}^2 + n_{01}{}^2 + n_{10}{}^2 + n_{11}{}^2\right) - \frac{2}{n-1}\left(n_0{}^2 + n_1{}^2\right) - 1$	2	1	5.9915
Poker	$X_3 = \frac{2^m}{k}\left(\sum_{i=1}^{2^m} n_i{}^2\right) - k$	$2^m - 2$ =6	326	12.5916
Run	$X_4 = \sum_{i=1}^{k} \frac{(B_i - e_i)^2}{e_i} + \sum_{i=1}^{k} \frac{(G_i - e_i)^2}{e_i}$	$2k - 2$ =6	27	12.5916
Correlación	$X_5 = 2\frac{\left(A(d) - \frac{n-d}{2}\right)}{\sqrt{n-d}}$	$N(0,1)$	148.822	1.96

5. CONCLUSIONES

A lo largo de este trabajo se ha profundizado en el estudio de la estructura interna de un criptosistema en el orden de la construcción de un cifrador basado en una sola permutación y su respectivo PRNG, un generador de secuencias seudo-aleatorias, para generar las respectivas claves. Con las dificultades que obedecen al análisis y definición de un conjunto de los posibles ataques realizados por los adversarios.

Even y Mansour [1] establecieron que la construcción de un cifrador mediante una permutación no requería la necesidad de almacenar o generar una multitud de permutaciones y con esta construcción evidentemente no se requirió almacenamiento de permutaciones que confirma dicha afirmación.

Muchas opiniones corroboran que el diseño y análisis de algoritmos convencionales están basados en las propiedades de difusión y confusión. Desde el punto de vista de permutaciones no seleccionadas éstas se construyen usando primitivas realizables. Para la seguridad de los algoritmos, es mejor evaluarlos a la luz de los métodos cripto-analíticos y los principios existentes.

La afirmación de que los PRNG son de diferentes clases de primitivas criptográficas, hay aplicaciones donde el desempeño de PRNG es un problema serio que amerita un nuevo algoritmo. Yarrow-160 en la práctica tiene debilidades en la estimación de la entropía y no en el criptoanálisis. Es necesario continuar probando los mecanismos de la entropía y ésta estará sujeta a futuras investigaciones.

Las reglas de control de re-sembrado son aún un diseño ad-hoc; el estudio extenso podría rendir una mejora en el control de reglas en el proceso re-sembrar.

No está establecido cómo se resiste el compromiso de estado del sistema propuesto. Esto constituye un enorme problema práctico que ha recibido poca atención en la literatura.

BIBLIOGRAFIA

[1]. S. Even and Y. Mansour ,A Construction of a Cipher from a Single Pseudorandom Permutation, Journal of cryptology,Volume 10, Number 3, 1997, Pages 151-162

[2] C.E. SHANNON, Comunication theory of secresy system , Bell system Teech. J, Vol. 28,1949,pp, 656-715.

[3] O. GOLDREICH, S Goldwaser and S. Micali , "How To Construct Random Function " Proceeding of the –25th Annual Symposium of Foundation of computer science, 24-26, 1984.

[4] J. DAEMEN, "Limitation of the even-Mansour Construction " , Katholieke universiteit Leuven Laboratorium Esat , B-3001 heverloe Belgium.

[5] A. MENEZES, P. Van Oorchot , Scott a: castone , " Hand Book Applied Cryptography " , CRC press , 1997

[6] J. KELSEY, B. Schnneir , D. Wagner and C. Hall , "Cryptanalytic Attack on Pseudorandom Number Generators " , fast software Encryption , 5th international Workshop Proceedings Springer-Verlag , 1998 , pp 168-188.

[7] J. KELSEY, B. SCHNNEIR, N FERGUSON, " Yarrow –160 Notes on the design an analysis of the Yarrow Criptographic Pseudorandom Number generator", Counterpane System , 101 E Minnehaha Parkway , Minneapolis , MN 55419, USA. [8] J. VILLALOBOS, "Tipo abstracto de datos en lenguaje c ", Mc Graw Hill , 1995

[9] M. LUBY and C. RACKOFF, "How to construct pseudorandom permutation from pseudorandom function , SIAM Journal computing , 373-386, april 1988.

[10] N. HOLLAND" Stream Cipher and Number Theory " ,North_holland mathematical Library , volume 55 , Elsevier , 1998

[11] SHIMON Even, MANSOUR , Yishay: A Construction of a Cipher From a Single Pseudorandom Permutation, in ASIACRYPT 1991, pages 210-224