UNIVERSIDAD AUTÓNOMA DE SAN LUIS POTOSÍ

FACULTAD DE CIENCIAS

IMPLEMENTACIÓN DE UN SISTEMA DE ENCRIPTACIÓN CON UN ENFOQUE MATRICIAL

TESIS PROFESIONAL

PARA OBTENER EL TITULO DE:

DOCTOR EN CIENCIAS APLICADAS

presenta:

M.C. JESÚS GUSTAVO FLORES ERAÑA

asesor de tesis:

DRA. MARCELA MEJÍA CARLOS

DR. JOSÉ SALOMÉ MURGUÍA IBARRA

San Luis Potosí, S.L.P., Marzo de 2015

Índice general

Índice de figuras

1

Introducción

El hombre desde sus principios ha tenido la necesidad de comunicarse ya sea de forma oral o escrita y muchas veces la información trasmitida requiere ser tratada solo por ciertas personas y que nadie más tenga acceso a esta información. Para eso se ha desarrollado la disciplina de la *Criptología*, que es la ciencia encargada de estudiar mensajes secretos los cuales procesados de cierta manera se hacen difíciles o imposibles de predecir.

Actualmente mantener la información de manera secreta es prioridad para la mayor parte de los procesos y para las personas por lo que existen diversos sistemas de encriptación. Muchos de esto sistemas sacrifican con tiempo, el procesamiento para obtener una encriptación más confiable o viceversa, tener un procesamiento con menor tiempo pero un proceso de encriptación menos confiable.

Existen muchos sistemas de encriptación basados en diferentes enfoques que están disponibles en la literatura, tales como el de datos de encriptación estándar (DES), el de encriptación avanzada estándar (AES), el algoritmo internacional de encriptación de datos (IDEA), entre otros [19, 20, 21]. Sin embargo, un proceso de encriptación eficiente sigue sin ser una cuestión fácil, debido a que un sistema de encriptación completo implica más tiempo de procesamiento, mientras que un sistema de encriptación simple presenta problemas de seguridad con resultados

predecibles o descifrables. Por lo tanto, se requiere tener un sistema de encriptación que tenga un balance óptimo entre las dos situaciones.

Las técnicas de criptografía que están basadas en la aplicación autómata celular (AC) pueden ser una posibilidad de equilibrio. Algunos sistemas de encriptación ya han usado AC como un generador pseudo-aleatorio o combinaciones de estas en el algoritmo de encriptación [42, 44, 39, 27, 40, 29, 30]. El uso de AC en criptografía ha resultado ser atractivo desde que puede ser adaptado a un algoritmo de cálculos masivos en paralelo en arquitecturas de la computación [41], así como en realizaciones físicas VLSI [43].

Como se señala en la Referencia [29], una de las primeras aplicaciones de AC en criptografía fue llevada a cabo por Urías y colaboradores [27], donde usaron una clase de bloques de criptosistemas que comprenden dos familias indizadas de permutaciones y un generador de números pseudo-aleatorios asintóticamente perfecto. Tal sistema de encriptación está basado en el fenómeno de sincronización de AC, al cual denominaremos sistema ESAC, donde se considera la regla 90. Una de las principales ventajas del sistema ESAC, es que su implementación se puede llevar a cabo con un simple arreglo bidimensional conformado de funciones lógicas XOR.

Con la finalidad de que el sistema de encriptación ESAC tenga una mayor robustez de tal forma que sea más eficiente, adecuado y equiparable con estándares ya establecidos, en este trabajo se propone considerar un enfoque matricial para implementar todos los elementos del sistema. Para lograrlo, en principio se presenta una forma matricial para implementar el generador de números aleatorios, y establecer los parámetros para que las secuencias de números generadas tengan caracter aleatorio. Lo anterior se apoyará al evaluar las secuencias utilizando el conjunto de pruebas estadísticas de la NIST. Además, una estructura de carácter multifractal de la densidad de unos de la matriz que implementa al generador de secuencias se revela y cuantifica de acuerdo a un formalismo multifractal. Con base a lo anterior, se extiende el enfoque matricial que nos permite implementar no sólo al generador de secuencias, si no

a todos los elementos del sistema ESAC. Para lograrlo, se muestra que una matriz puede ser empleada como base para implementar de forma efectiva la mayoría de las matrices que se consideran en el sistema de encriptación. Considerando este enfoque matricial del sistema, se realiza la respectiva implementación numérica utilizando el software de Labview, marca registrada de National Instruments. Por último, y como aplicación, se realiza la implementación numérica de las estapas de compresión y encriptación a señales de audio. En la etapa de compresión se considera utilizar un esquema de energía basado en la transformada discreta wavelet, herramienta que ha resultado ser eficiente en varias aplicaciones, particularmente en cuestiones de compresión. De hecho, las tasas de compresión obtenidas fueron muy buenas, ya que una gran concentración de energía se presento en una pequeña cantidad de coeficientes en las señales de audio transformadas. Lo anterior nos permitirá manejar de manera más flexible grandes cantidades de información, para posteriormente encriptarlas y mantener seguridad de la misma. Creemos que con esta adecuación se tendrá una herramienta útil y flexible para las aplicaciones actuales de multimedia.

La estructura del trabajo es de la siguiente manera. En el Capítulo 2 se explican las bases fundamentales de la criptografía, las partes fundamentales de un criptosistema, autómatas celulares, el fenómeno de sincronización con la regla 90 y la estructura del sistema de encriptación ESAC. En el Capítulo 3 se describe la implementación de los principales componentes que conforman el sistema ESAC mediante un esquema o enfoque matricial. Además se realiza un análisis estadístico de las secuencias pseudo-aleatorias del generador de números aleatorios, así como las propiedad multifractales de la densidad de unos de las matrices que implementan el generador para diferente tamaño. Asimismo, en el Capítulo 4 se presenta la implementación numérica de un sistema que conjunta las estapas de encriptación y compresión utilizando el software de programación Labview. Tal sistema considera la encriptación de señales de audio comprimidas, donde la herramienta de la transformada wavelet es considerada para realizar la última etapa. Por último, el Capítulo 5 presenta las conclusiones formuladas de este trabajo, así como el trabajo a futuro que podría realizarse.

2

Sistema de encriptación ESAC

Existe un gran número de sistemas de encriptación cuyo principal objetivo es proteger la información por medio de un algoritmo que haga uso de una o más claves. En este Capítulo se describen las bases fundamentales de un sistema de encriptación haciendo énfasis en el sistema de encriptación ESAC.

2.1 Criptografía

La disciplina de la *criptología*, es la ciencia encargada de estudiar mensajes secretos, que al ser procesados de cierta manera se hacen difíciles o imposibles de predecir. En la Figura 2.1 se muestra una vista general del campo de la criptología.

Como se observa en la Figura 2.1 la criptología se divide en las ramas de criptografía y criptoanálisis [1].

La criptografía hace referencia a las técnicas para ocultar, transformar o enmascarar algún tipo de información, tal como video, audio, imágenes, entre otros, mientras que el criptoanálisis utiliza técnicas para desenmascarar o romper los códigos generados por la criptografía. Debido

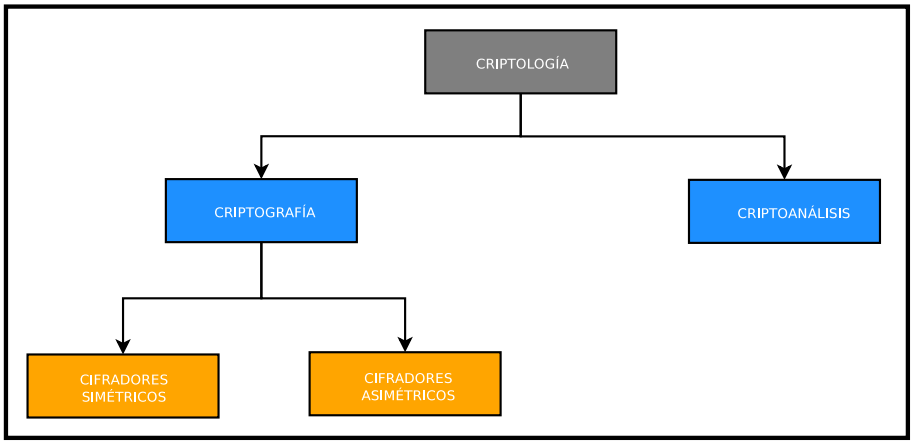

Figura 2.1: Vista general de la Criptografía

a que el criptoanálisis es la única manera de asegurar que un criptosistema es seguro, es una parte integral de la criptología. De hecho, se puede establecer que el criptoanálisis es la importancia central de los criptosistemas modernos: sin gente que trate de romper los métodos de encriptación, no sabremos si son seguros o no [1]. Sin embargo en este trabajo nos centramos en la *criptografía*.

Las palabras encriptar, cifrar y encriptación hacen referencia al hecho de transformar o enmascarar la información original, la cual se define como *texto plano*, en *texto encriptado* o *texto cifrado*. Mientras que las palabras desencriptar o descifrar hacen referencia al hecho de transformar el *texto encriptado* en *texto plano*.

Así mismo, la criptografía se puede clasificar en criptografía clásica y criptografía moderna. La criptografía clásica hace referencia a las operaciones de sustitución y trasposición de caracteres. En esta criptografía encontramos los cifradores monoalfabéticos, polialfabéticos, así como cifradores por sustitución homofónica y trasposición [1].

Por otra parte, la criptografía moderna, además de utilizar la sustitución y trasposición de sus caracteres, utiliza ciertas propiedades matemáticas, y adicionalmente se puede clasificar según el tipo de clave y según el tipo de mensaje [1].

Para el caso del tipo de claves podemos distinguir dos principales clases de criptosistemas: los de *clave privada* y los de *clave pública*. En los sistemas de *clave privada* los usuarios del sistema comparten o guardan en secreto la clave para encriptar y desencriptar, y se conocen también como criptosistemas simétricos. Por otra parte, en los sistemas de clave pública se consideran dos llaves, una para encriptar, llamada clave pública, y otra para desencriptar llamada clave privada. Estos sistemas son conocidos como criptosistemas asimétricos [1].

Para el del caso de tratamiento del mensaje se consideran cifradores de bloques, ver Figura 2.2, en donde el texto plano se divide en N bloques de igual longitud, la cual será previamente definida por el algoritmo de encriptación.

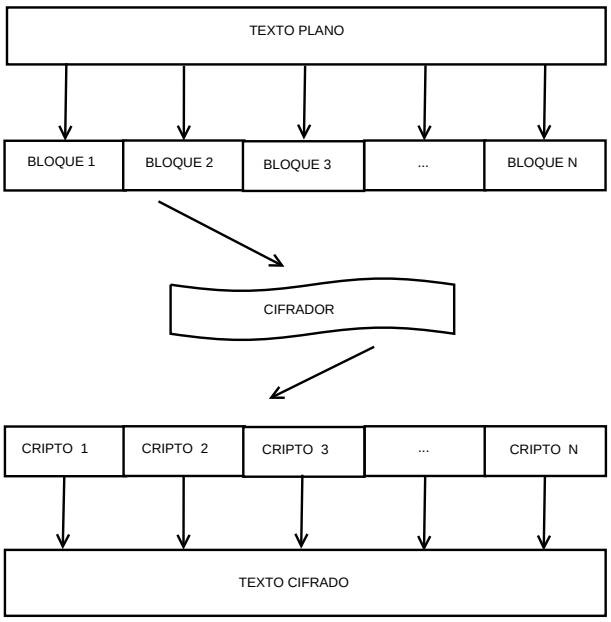

Figura 2.2: Diagrama ilustrativo de un cifrador de bloque.

También se tiene los cifradores de flujo, los cuales encriptan bit a bit. Normalmente la encriptación se realiza con la función Booleana XOR, resultando que este cifrador seá mas rápido que el cifrador de bloques. En la Figura 2.3 se muestra un diagrama a bloques de un cifrador de flujo.

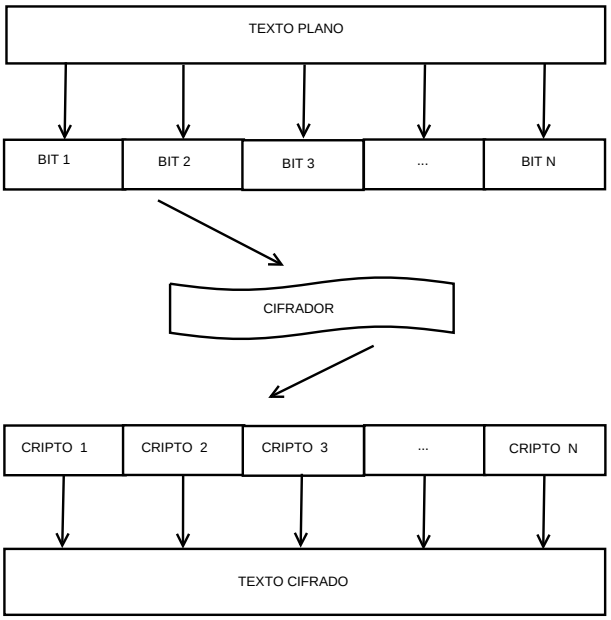

Figura 2.3: Digrama a bloques de un cifrador de flujo

2.2 Criptosistema

Un criptosistema o sistema criptográfico se conforma por los siguientes elementos (M,C,K, E y D), con las siguientes propiedades:

- M representa el conjunto de todos los mensajes sin encriptar, lo que se denomina texto plano.
- C representa el conjunto de elementos encriptados o criptogramas, lo que denominamos texto encriptado.
- K representa el conjunto de llaves o claves que se emplean dentro del criptosistema.
- E es el conjunto de transformaciones de encriptación o familias de funciones, que se le aplican a M para obtener C, en otras palabras, $E = \{E_k : k \in K\}$, donde $E_k : M \to C$. Con lo cual se tiene una transformación diferente E_K para cada valor posible de llave o clave

del sistema.

- D es el conjunto de transformaciones de desencriptación.

Por lo tanto todo sistema critográfico o criptosistema debe de cumplir la siguiente condición:

Para cada llave $e \in K$, tenemos una llave $d \in K$ tal que

$$D_d(E_e(m)) = m, \tag{2.1}$$

para todo texto plano $m \in M$.

Es decir que si tenemos un mensaje m y lo encriptamos utilizando una llave k obtenemos un mensaje encriptado c y si el mensaje c lo desencriptamos con la misma llave k obtendremos nuevamente el mensaje original m.

En la Figura 2.4 se muestra un criptosistema con sus respectivos elementos.

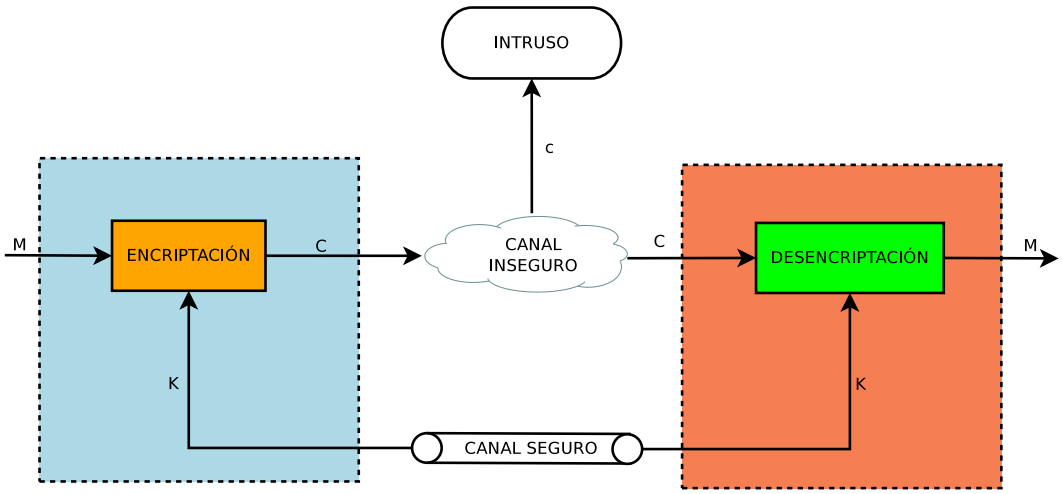

Figura 2.4: Sistema de encriptación o Criptosistema de llave simétrica

.

2.3 Criptografía Basada en Autómatas Celulares

2.3.1 Autómatas Celulares AC

Un autómata celular (AC), en su versión mas simple, es un un vector unidimensional de celdas o casillas, donde cada una tienen un estado de 0 o 1. El estado de la celda puede cambiar conforme al tiempo, es decir, en cada paso discreto (finito) de tiempo, las celdas actualizan su estado en función de su estado anterior y al de las celdas vecinas.

Los autómatas celulares han sido considerados para su aplicación en la criptografía desde que aparecieron por dos principales razones. La primera es que los autómatas celulares es un modelo simple que puede generar patrones pseudoaleatorios, y la segunda es que los autómatas celulares pueden ser implementado de manera eficiente en software y hardware [6].

2.3.2 Autómatas Celulares elementales

Los autómatas celulares elementales (ECA) pueden ser considerados como son sistemas dinámicos discretos que evolucionan en pasos de tiempo discretos. El espacio de estados de un ECA de tamaño N es el conjunto $\Omega = \mathbb{Z}_k^{\mathbb{N}}$ donde todas las secuencias de N toman valores de $\mathbb{Z}_k = \{0, 1, ..., k-1\}$, donde su evolución esta definida por la repetida iteración de un operador de evolución $\mathscr{A} : \mathbb{Z}_K^{\mathbb{Z}} \to \mathbb{Z}_K^{\mathbb{Z}}$. En este trabajo, se considera $k = 2$ donde $\mathbb{Z}$, es el conjunto de enteros. Un estado del autómata $\underline{x} \in \mathbb{Z}_2^{\mathbb{Z}}$, tiene coordenadas $(\underline{x})_i = x_i \in \mathbb{Z}_2$ donde $i \in \mathbb{Z}$ y el estado del autómata el tiempo $t \geq 0$ es denotado por $\underline{x}^t \in \mathbb{Z}_2^{\mathbb{Z}}$ y su evolución es definida iterativamente por la regla $\underline{x}^{t+1} = \mathscr{A}(\underline{x}^t)$. Empezando por el estado inicial $\underline{x}^0$ el autómata genera el siguiente patrón espacio-tiempo $x \in \mathbb{Z}_2^{\mathbb{Z}\times\mathbb{N}}$ con el estado $(\underline{x})^t = \underline{x}^t = \mathscr{A}(\underline{x}^0)$ alcanzado desde $\underline{x}^0$ después de $t \in \mathbb{N}$ pasos de tiempo. $\mathbb{N}$ denota el conjunto de enteros positivos.

Uno puede ver que el tiempo, el espacio y los estado de este sistema toman sólo valores discretos. El ECA considerado evoluciona de acuerdo con la regla local $\underline{x}_i^{t+1} =$

$\mathscr{A}_{\mathscr{L}}(x_{i-1}^t, x_i^t, x_{i+1}^t) = [x_{i-1}^t + x_i^t]$ mód 2 que corresponde a la regla 90. La siguiente es la tabla de consulta de la regla 90.

Número	7	6	5	4	3	2	1	0
Vecindad	111	110	101	100	011	010	001	000
Resultado de Regla	0	1	0	1	1	0	1	0

La tercera fila muestra el futuro estado de la celda si ella misma y sus vecinos se encuentran en el arreglo mostrado arriba en la segunda fila. De hecho, la regla es nombrada por el equivalente decimal de la expresión binaria en la tercera fila. Cuando la misma regla es aplicada para actualizar celdas de los ECA, estos ECA son llamados ECA uniformes; de otra forma los ECA son llamados "no uniformes" o híbridos. Es importante observar que las reglas de evolución de los ECA son determinadas por dos factores principales, la regla y las condiciones iniciales.

Los algoritmos de encriptación utilizados en este trabajo se basan en el autómata celular con la regla local

$$\mathscr{A}_{\mathscr{L}}(x_{i-1}, x_i, x_{i+1}) = x_{i-1} + x_{i+1} \in \mathbb{Z}_2.$$

La Figura 2.5 muestra la regla local donde las coordenadas son representadas por círculos y el valor de cada coordenada es la suma módulo 2 de los valores presentados por las flechas entrantes al círculo. Y las flechas salientes representan el valor tomado por la coordenada, en la Figura 2.5 el valor de la coordenada $(i, t+1)$ es $x_i^{t+1} = x_{i-1}^t + x_{i+1}^t$.

2.3.3 Sincronización en Automatas celulares

Dos sistemas dinámicos acoplados sincronizan si, después de un largo período de tiempo, sus comportamientos consiguen estar arbitrariamente cerca. Esto es, en cada paso de tiempo ambos sistemas evolucionan de acuerdo a la misma regla.

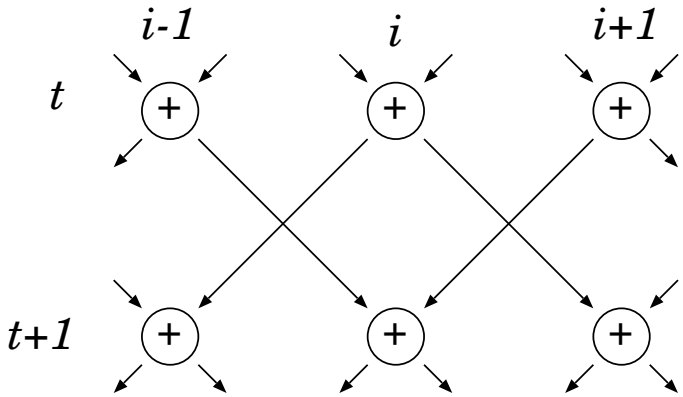

Figura 2.5: Regla local $\mathscr{A}_{\mathscr{L}}(x_{i-1}, x_i, x_{i+1}) = x_{i-1} + x_{i+1} \in Z_2$, $Z_2 = \{0,1\}$.

En el caso de autómatas celulares existe la sincronización como el resultado de un acoplamiento no trivial. El acoplamiento en autómatas celulares sucede cuando un conjunto determinado de coordenadas (*coordenadas acopladas*) es copiada de uno de los sistemas el cual es el *autómata celular manejador*, al sistema de respuesta que será el *autómata celular de respuesta*. Esto es, en cada paso de tiempo ambos sistemas evolucionan de acuerdo a la misma regla, y las coordenadas acopladas del autómata celular manejador son copiadas a las correspondientes coordenadas del autómata celular de respuesta. En la Figura 2.6 se muestra un ejemplo que ilustra el acoplamiento unidireccional en autómatas celulares.

Considerando esta sincronización con autómatas celulares se implemento una unidad encriptadora de la cual se construyen 2 familas de permutaciones, Ψ y Φ, de palabras binarias de longitud finita de $2^k - 1$, $k \in \mathbb{Z}$. Estas permutaciones son usadas para encriptar y desencriptar N bloques de longitud $2^k - 1$, $k \in \mathbb{Z}$.

Este fenómeno de sincronización también permite la construcción de una función llamada *función h* la cual se utiliza para la generación de llaves.

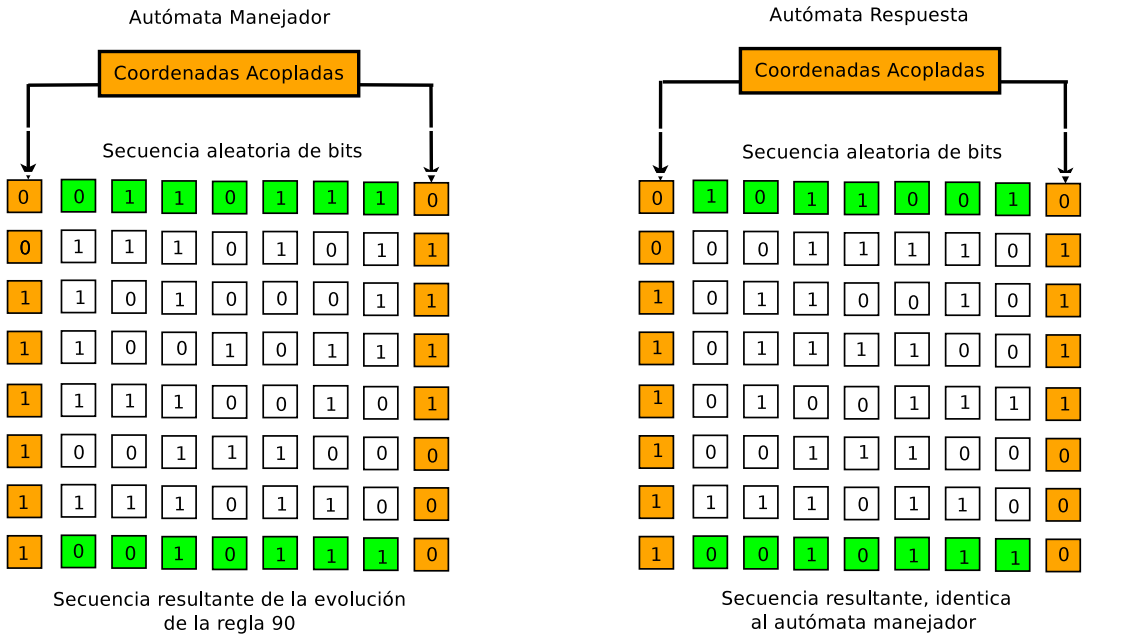

Figura 2.6: Ejemplo de acoplamiento unidireccional

2.4 Sistema de Encriptación ESAC

En la Figura 2.7 se muestra la unidad encriptadora básica del criptosistema ESAC (cuadro, formado con lineas punteadas), mediante la cual se definen las permutaciones ψ y ϕ y la función h. Se puede observar la evolución de la unidad a partir de las coordenadas acopladas x_0^0 y x_{N+1}^0 identificando las palabras $\mathbf{x}, \mathbf{y}, \mathbf{m}, \mathbf{c}$ y $\mathbf{t}$.

- **Permutación $\mathbf{m} = \phi_{\mathbf{x}}(\mathbf{c})$**

  La palabra $\mathbf{m}$, que se identifica como un bloque de texto franco, es la palabra $\mathbf{m} = (x_1^N, x_2^N, ..., x_i^N, ..., x_N^N)$, la cual se encuentra en la parte inferior del bloque básico en la Figura 2.7.

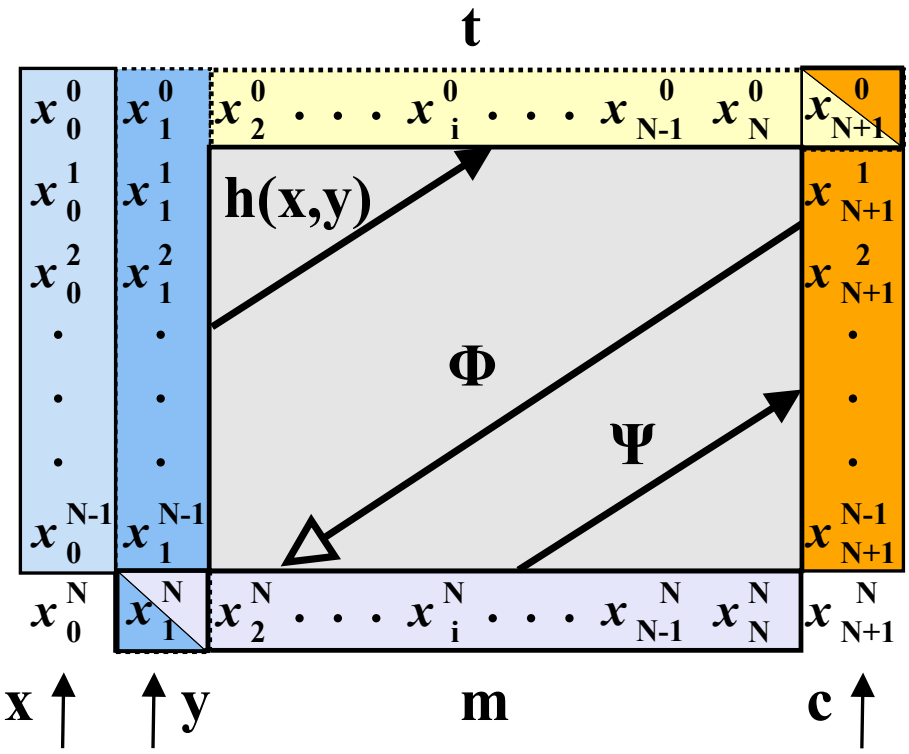

Figura 2.7: Evolución de una secuencia infinita donde las coordenadas $i = 0$ e $i = N+1$ están dadas externamente.

La forma evidente de generar la permutación $\mathbf{m} = \phi_{\mathbf{x}}(\mathbf{c})$, es haciendo evolucionar el autómata hacia adelante en el tiempo, usando como entradas las palabras **c** y **x**, ver Figura 2.5. En [3, 4, 5] se demuestra que **m** solo depende de **x** y de **c**. La independencia en **t** es resultado de la sincronización.

- **Permutación $\mathbf{c} = \psi_{\mathbf{x}}(\mathbf{m})$**

  La palabra **c**, la cual es identificada como el bloque encriptado, es la palabra $\mathbf{c} = (x^0_{N+1}, x^1_{N+1}, ..., x^n_{N+1}, ..., x^{N-1}_{N+1})$ situada al lado derecho en el bloque básico mostrado en la Figura 2.7. Para desencriptar este bloque se utiliza la permutación inversa ϕ de manera que se tiene $\mathbf{m} = \phi_{\mathbf{x}}(\mathbf{c}) = \phi_{\mathbf{x}}(\psi_{\mathbf{x}}(\mathbf{m}))$.

  Para generar la permutación $\mathbf{c} = \psi_{\mathbf{x}}(\mathbf{m})$, se hace evolucionar el autómata hacia atrás en el

tiempo, ver Figura 2.8, utilizando como entradas las palabras **x** y **m**. Operativamente es necesario proporcionar algún valor para **y** pero no afecta el valor de $\mathbf{c} = \psi_{\mathbf{x}}(\mathbf{m})$. De igual manera esta independencia es resultado de la sincronización, ver [3, 4, 5].

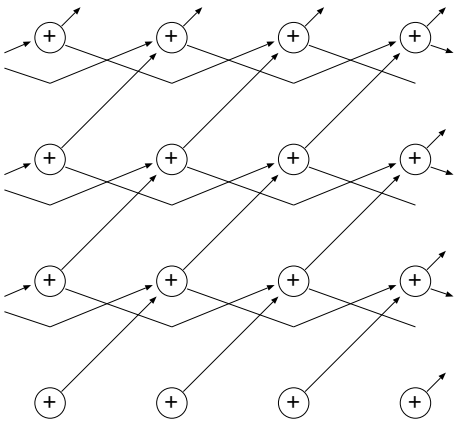

Figura 2.8: Ejecución del autómata hacia atrás

- **Función $\mathbf{t} = h(\mathbf{x}, \mathbf{y})$**

En la parte izquierda de la unidad encriptadora mostrada en la Figura 2.7 se encuentran las dos partes de la semilla **x** y **y**, utilizadas por la función $\mathbf{t} = h(\mathbf{x}, \mathbf{y})$. Estas palabras son $\mathbf{x} = (x_0^0, x_0^1, ..., x_0^n, ...x_0^{N-1})$ y $\mathbf{y} = (x_1^0, x_1^1, ..., x_1^n, ...x_1^{N-1}, x_1^N)$. En la parte superior está la palabra **t** que es el resultado de la función h, esta palabra es identificada como $\mathbf{t} = (x_2^0, x_3^0, x_4^0, ..., x_i^0, ..., x_{N+1}^0)$.

Para generar esta función h, se hace evolucionar el autómata hacia atrás, ver Figura 2.9, utilizando como entradas las palabras **x** y **y**.

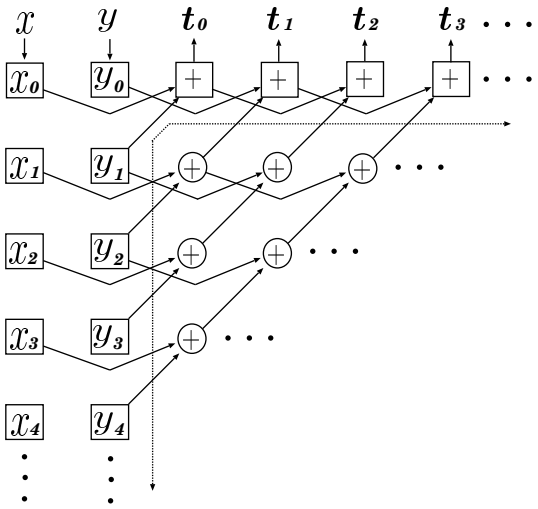

Figura 2.9: Esquina superior izquierda de la unidad básica

2.4.1 Funciones de un solo tiempo

Para las permutaciones ϕ, ψ y la *función h* se hace uso de las leyes y teoremas del álgebra booleanas para reducir estas funciones a un algoritmo de un solo tiempo de manera que cada ciclo de reloj se ejecute una función completa, y no una vez la regla del automata.

La palabra $\mathbf{m}$ corresponde a la permutación $\mathbf{m} = \phi_{\mathbf{x}}(\mathbf{c})$, la cual se genera haciendo evolucionar el autómata hacia adelante. En la Figura 2.10 se muestra como se obtiene el bit $m1$ de la palabra $\mathbf{m}$ de 7 bits.

En la Ecuación (2.2) se muestra el valor del bit $m1$ al pasar por todas la compuertas xor y su reducción utilizando las reglas del algebra booleana.

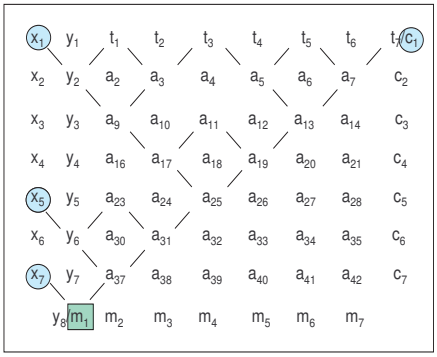

Figura 2.10: Reducción del bit $m1$ de M.

$$
\begin{aligned}
m1 &= x7 \oplus a37 = x7 \oplus y6 \oplus a31 = x7 \oplus x5 \oplus a23 \oplus a23 \oplus a25 \\
&= x7 \oplus x5 \oplus a25 = x7 \oplus x5 \oplus a17 \oplus a19 = x7 \oplus x5 \oplus a9 \oplus a13 \\
&= x7 \oplus x5 \oplus y2 \oplus a3 \oplus a5 \oplus a7 \\
&= x7 \oplus x5 \oplus x1 \oplus t1 \oplus t3 \oplus t3 \oplus t5 \oplus t5 \oplus c1 \\
m1 &= x7 \oplus x5 \oplus x1 \oplus c1
\end{aligned}
\tag{2.2}
$$

Este mismo procedimiento se aplicaría para cada uno de los bits de palabras de tamaño $2^k - 1$. En el apéndice A se muestran las ecuaciones reducidas de los bits de las palabras **m** y **c** de tamaño de 31 bits.

En la Figura 2.11 se muestra como obtener el bit $t3$ de la palabra **t** de tamaño de 15 bits. Como se comentó en la sección anterior el valor de la palabra **t** se obtiene haciendo evolucionar el autómata hacia atras, utilizando los valores de las palabras **x** y **y**.

En (2.3) se muestra el valor del bit t_3 al pasar por todas las compuertas xor, y su reducción utilizando las reglas de algebra booleana.

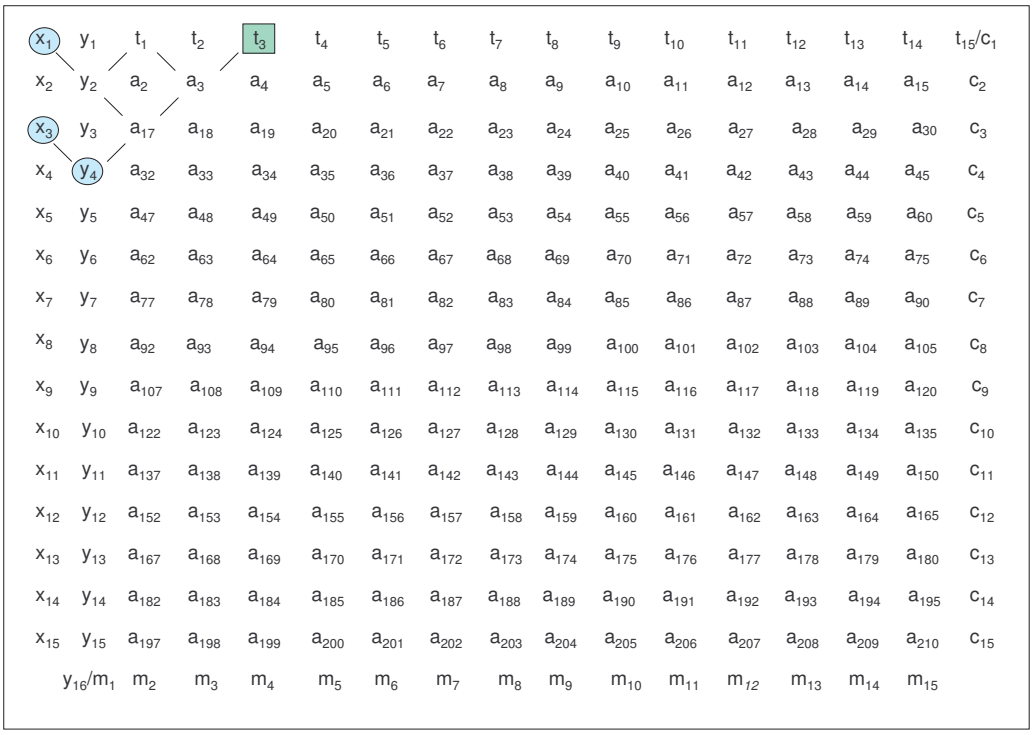

Figura 2.11: Se hace evolucionar el automata para obtener $t3$ de la palabra **t**.

$$\begin{aligned} t3 &= t1 \oplus a3 \\ &= x1 \oplus y2 \oplus y2 \oplus a17 \\ &= x1 \oplus a17 \\ &= x1 \oplus x3 \oplus y4 \end{aligned} \tag{2.3}$$

Haciendo el mismo procedimiento para cada bit de la palabra **t**, se obtienen las ecuaciones de un solo tiempo de la función h las cuales se muestran en el apéndice A.

3

Sistema ESAC con un enfoque matricial

En la actualidad, la búsqueda de algoritmos o procedimientos para implementar de manera más eficiente cualquier sistema se ha incrementado, donde el sistema ESAC no es la excepción. En este Capítulo se describe la implementación de los principales componentes que conforman al sistema ESAC mediante un esquema matricial. Se observará que la implementación se fundamenta principalmente en la matriz de secuencias denotada como $\mathbf{Q}_N$. En particular, mediante la aplicación de ciertas operaciones básicas de matrices o transformaciones en $\mathbf{Q}_N$, podemos ser capaces de implementar la mayoría de las etapas involucradas en el sistema de encriptación. Por ejemplo las familias de permutación ϕ y ψ y el generador de números pseudo-aleatorios.

3.1 Enfoque matricial de las familias de permutación

3.1.1 Proceso de encriptación

En el Capítulo 2 se describió la forma de desarrollar la etapa de encriptación, donde se requería calcular $\mathbf{c} = \Psi_{\mathbf{x}}(\mathbf{m})$. Con base a las ecuaciones de encriptación obtenidas en este proceso, se

proponen las matrices, $\mathbf{P}_N$ y $\mathbf{Q}_N$, tal que

$$\mathbf{c} = \Psi_{\mathbf{x}}(\mathbf{m}) = [(\mathbf{P}_N \times \mathbf{x}) + (\mathbf{Q}_N \times \mathbf{m})] \text{ mód } 2. \tag{3.1}$$

Ambas matrices son cuadradas de orden N, es decir, tienen dimensiones $N \times N = (2^n - 1) \times (2^n - 1)$, para $n = 1, 2, 3, \ldots$. El valor de N representa y/o se asocia con el número de bits a considerar en el proceso de encriptación. La matriz $\mathbf{P}_N$ se inicializa con un vector $\mathbf{p} = [p_1, p_2, \ldots p_N]$, el cual corresponde a la primer fila. Las componentes con índice de posición $j = (2^n + 1) - 2^{i+1}$, para $i = 0, 1, 2, ..(n-1)$, tienen el valor de 1, mientras que para otros valores es 0. Las siguientes $(N-1)$ filas se generan al aplicar un registro de corrimiento de una posición hacia la derecha de la fila anterior considerando que en cada fila después de realizar el corrimiento el primer valor sería 0. Por ejemplo, $n = 3$ implica $N = 7$, por lo que

$$\mathbf{P}_7 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3.2}$$

Por otra parte, la matriz de secuencias $\mathbf{Q}_N$ se puede generar de manera inicial por el vector $\boldsymbol{a} = [a_1, 0, 0, \ldots 0]$, donde la componente a_1 tiene el valor de 1. Este vector constituye la primer fila de la matriz $\mathbf{Q}_N$ y para generar las siguientes $(N-1)$ filas se aplica la regla local 90 del AC a la fila anterior utilizando condiciones de frontera cero. Lo anterior se ilustra para el valor de $N = 7$, donde la matriz $\mathbf{Q}_7$ tiene la siguiente estructura:

$$\mathbf{Q}_7 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}. \tag{3.3}$$

La encriptación del texto plano de 7 bits $\mathbf{m} = [m_1, m_2, m_3, m_4, m_5, m_6, m_7]$, con secuencia pseudo-aleatoria $\mathbf{x} = [x_1, x_2, x_3, x_4, x_5, x_6, x_7]$, y considerando las expresiones (3.2) y (3.3) en (3.1), es

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} \oplus \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \\ m_6 \\ m_7 \end{pmatrix}, \tag{3.4}$$

donde el símbolo $\oplus$ representa la operación XOR. Se puede observar que el resultado de (3.4) es el siguiente conjunto de ecuaciones:

$$
\begin{aligned}
c_1 &= x_1 + x_5 + x_7 + m_1, \\
c_2 &= x_2 + x_6 + m_2, \\
c_3 &= x_3 + x_7 + m_1 + m_3, \\
c_4 &= x_4 + m_4, \\
c_5 &= x_5 + m_3 + m_5, \\
c_6 &= x_6 + m_2 + m_6, \\
c_7 &= x_7 + m_1 + m_3 + m_5 + m_7,
\end{aligned}
$$

donde al resultado final se le aplica la operación de módulo 2.

3.1.2 Proceso de desencriptación

Para el proceso desencriptación tenemos que el texto plano $\mathbf{m} = \Phi_{\mathbf{x}}(\mathbf{c})$ se puede obtener al aplicar la inversa a la expresión (3.1). Lo anterior es posible ya que las matrices son triangulares, y por ende no son singulares. Así la implementación matricial de la permutación inversa, con estructura similar a (3.1), es:

$$
\mathbf{m} = \Phi_{\mathbf{x}}(\mathbf{c}) = [(\mathbf{R}_N \times \mathbf{x}) + (\mathbf{T}_N \times \mathbf{c})] \text{ mód } 2, \tag{3.5}
$$

donde las matrices son cuadradas con dimensiones $N \times N = (2^n - 1) \times (2^n - 1)$, para $n = 1, 2, 3, \ldots$. Las matrices en (3.5) tienen las expresiones $\mathbf{R}_N = [-\mathbf{Q}_N^{-1}\mathbf{P}_N] \text{ mód } 2$ y $\mathbf{T}_N = \mathbf{Q}_N^{-1} \text{ mód } 2$. Para el valor de $N = 7$ tenemos que las matrices $\mathbf{R}_7$ y $\mathbf{T}_7$ son:

$$\mathbf{R}_7 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \qquad \mathbf{T}_7 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}. \tag{3.6}$$

Como ilustración de la desencriptación del texto encriptado de 7 bits $\mathbf{c} = [c_1, c_2, c_3, c_4, c_5, c_6, c_7]$, con secuencia pseudo-aleatoria $\mathbf{x} = [x_1, x_2, x_3, x_4, x_5, x_6, x_7]$, y considerando las expresiones de las matrices de (3.6), se tiene

$$\begin{pmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \\ m_6 \\ m_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} \oplus \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{pmatrix}, \tag{3.7}$$

El resultado de (3.7) es el siguiente conjunto de ecuaciones:

$$
\begin{aligned}
m_1 &= x_1 + x_5 + x_7 + c_1,\\
m_2 &= x_2 + x_6 + c_2,\\
m_3 &= x_1 + x_3 + x_5 + c_1 + c_3,\\
m_4 &= x_4 + c_4,\\
m_5 &= x_1 + x_3 + c_1 + c_3 + c_5,\\
m_6 &= x_2 + c_2 + c_6,\\
m_7 &= x_1 + c_1 + c_5 + c_7.
\end{aligned}
$$

3.1.3 Generador de llaves pseudo-aleatorias

Existen varias formas de llevar a cabo la implementación de las familias indexadas y del generador de llaves. Con la finalidad de implementar y establecer la dependencia de la matriz $\mathbf{Q}_N$ para el generador, consideramos el enfoque de la Referencia [7], tratando de conjuntar la forma descrita en [8].

Para calcular secuencias pseudo-aleatorias de N bits, se utiliza la matriz de secuencias de $\mathbf{H}_N$, la cual tiene dimensiones $(2N+1)\times(2N+1)$ y está formada por dos matrices, $\mathbf{H}_{N_t}$ y $\mathbf{H}_{N_b}$, que constituyen las partes superior e inferior de $\mathbf{H}_N$, es decir, $\mathbf{H}_N = (\mathbf{H}_{N_t};\mathbf{H}_{N_b})$. La matriz $\mathbf{H}_{N_t}$ tiene dimensiones $N\times(2N+1)$, mientras que la matriz $\mathbf{H}_{N_b}$ tiene dimensiones $(N+1)\times(2N+1)$. Su estructura es la siguiente

$$
\mathbf{H}_N = \begin{pmatrix} \mathbf{H}_{N_t} \\ \mathbf{H}_{N_b} \end{pmatrix} = \left(\begin{array}{c|c} \mathbf{Q}_N^{-1} & \widehat{\mathbf{Q}} \\ \hline \mathbf{I} & \mathbf{0} \end{array}\right) = \left(\begin{array}{c|c} \mathbf{T}_N & \widehat{\mathbf{Q}} \\ \hline \mathbf{I} & \mathbf{0} \end{array}\right), \tag{3.8}
$$

donde $\mathbf{I}$ es la matriz identidad con dimensiones $(N+1)\times(N+1)$ y $\mathbf{0}$ es la matriz cero con dimensiones $N\times(N+1)$. La matriz $\widehat{\mathbf{Q}}$ tiene dimensiones $N\times(N+1)$, donde sus primeras $(N-1)$ filas y N columnas son las mismas de la matriz $\mathbf{Q}_N^{-1}$, pero sin la primer fila. Mientras

que las componentes de la N-ésima fila y la $(N+1)$-ésima columna de $\widehat{\mathbf{Q}}$ tienen el valor de 0, y en la intersección de ambos tiene el valor de 1. Por ejemplo, para el valor $N = 7$ y considerando (3.3) tenemos que

$$\mathbf{H}_7 = \left(\begin{array}{ccccccc|ccccccc|c} 1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&1&0&1&0&0&0&0&0 \\ 1&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&1&0&0&0&0&1&0&1&0&1&0&0&0 \\ 1&0&1&0&1&0&0&0&0&1&0&0&0&1&0&0 \\ 0&1&0&0&0&1&0&0&1&0&0&0&1&0&1&0 \\ 1&0&0&0&1&0&1&0&0&0&0&0&0&0&0&1 \\ \hline 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0 \end{array}\right), \tag{3.9}$$

donde podemos observar el rol de la matriz $\mathbf{Q}_7$ en la implementación de la matriz $\mathbf{H}_7$.

Para complementar, y de acuerdo al algoritmo establecido en el Capítulo 2, se debe entender que la matriz $\mathbf{H}_{N_t}$ calcula la secuencia pseudo-aleatoria de llaves, mientras que la matriz $\mathbf{H}_{N_b}$ se encarga de calcular la secuencia de retro-alimentación. Por lo tanto, una vez seleccionado el número de bits N de secuencias, se pueden generar las secuencias pseudo-aleatorias de N bits con ayuda de la matriz $\mathbf{H}_N$

$$\mathbf{U}_{k+1} = \mathbf{H}_N \mathbf{U}_k, \qquad k = 1, 2, \ldots \tag{3.10}$$

donde $\mathbf{U}_k = [\mathbf{x}\ \mathbf{y}]^t$ corresponde a las primeras entradas o semillas de la función h y $\mathbf{U}_{k+1}$ está conformada por las siguientes entradas de la función h, es decir, $\mathbf{U}_{k+1}$ está formada por la llave pseudo-aleatoria generada y la secuencia de retroalimentación.

Generador de llaves pseudo-aleatorias modificado

Con el fin de mantener al generador lo más impredecible posible bajo un ataque de búsqueda aleatoria, Mejía y Urías [15] proponen un esquema generador que acopla tres transformaciones h.

Tal propuesta se muestra en la Figura 3.1, donde el generador de manera interna tiene dos copias de la transformación básica h que se iteran de forma autónoma, que a partir de las secuencias de llaves iniciales generan las secuencias $\{\mathbf{p}_k\}_{k\geq 0}$ y $\{\mathbf{q}_k\}_{k\geq 0}$. La tercera copia, la cual denominamos mapeo-**x**, es iterada un poco diferente; la función h en el mapeo-**x** genera secuencias pseudo-aleatorias de llaves donde las secuencias de entrada son $\{\mathbf{p}_k\}_{k\geq 0}$ y $\{\mathbf{q}_k\}_{k\geq 0}$, es decir, $\mathbf{x}_k = h(\mathbf{p}_k, \mathbf{q}_k)$. A pesar de que las tres funciones acopladas generan secuencias pseudo-aleatorias, solo la secuencia del mapeo-**x** se libera.

Se debe observar que para prevenir predictibilidad las primeras dos palabras que son generadas en las dos primeras funciones h son usadas y destruidas dentro de este generador de llaves, y por lo tanto no se encuentran disponibles de manera externa. Por otra parte las secuencias $\mathbf{p}_k$ y $\mathbf{q}_k$ tienen una longitud de N bits, pero las secuencias de entrada que requiere la transformación h deben ser de N bits y de $(N+1)$ bits. Para resolver la situación, el bit faltante se obtiene al aplicar la suma módulo 2 a los dos bits menos significativos de las respectivas previas entradas, los cuales se convierten en los bits más significativos de las nuevas entradas de las dos funciones h. Cabe mencionar que existen diferentes maneras de generar este bit faltante, pero en este trabajo se considero de esta manera. Con este esquema, y en términos matriciales, las nuevas llaves pseudo-aleatorias son calculadas mediante la expresión

$$\mathbf{X}_N = \mathbf{H}_{N_t}\mathbf{V}_N \tag{3.11}$$

donde $\mathbf{X}_N = [x_1, x_2, \ldots, x_N]^T$, $\mathbf{H}_{N_t}$ es la parte superior de la matriz de $\mathbf{H}_N$, y $\mathbf{V}_N = [p_1, \ldots, p_N, q_1, \ldots, q_{N+1}]^T$. Por ejemplo, para el valor de $N = 7$, tenemos que la se-

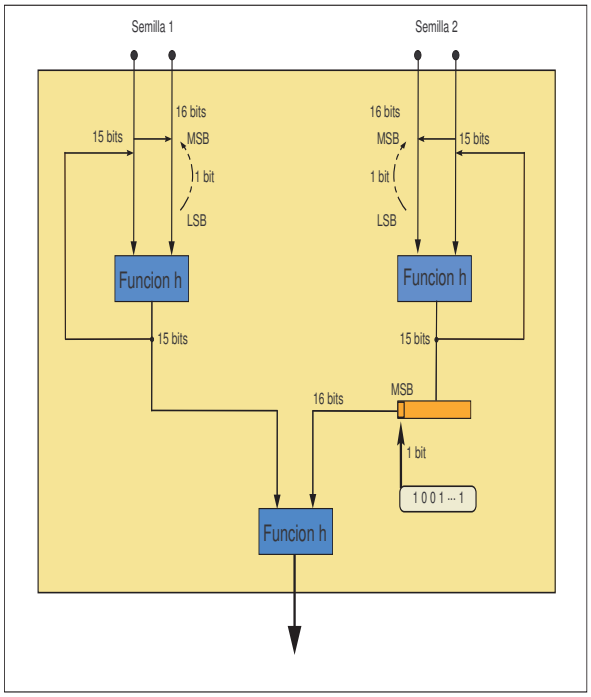

Figura 3.1: Esquema del generador de secuencias pseudo-aleatorias conformado de tres transformaciones acopladas.

cuencia pseudo-aleatoria generada $\mathbf{X}_7$, a partir de las secuencias pseudo-aleatorias $\mathbf{V}_7 = [p_1, \ldots, p_7, q_1, \ldots, q_8]^T$, se calcula

$$\mathbf{X}_7 = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{V}_7 \qquad (3.12)$$

donde los valores de p_i y q_i se calculan como se explicó anteriormente.

3.2 Análisis estadístico del generador de secuencias pseudo-aleatorias

Se sabe que en un sistema de encriptación el generador de secuencias o llaves pseudo-aleatorias es quizás la parte medular en el sistema. Con la finalidad de analizar las propiedades de aleatoriedad, existen diferentes opciones para evaluar la aleatoriedad, tales como los conjuntos de pruebas estadísticas de la suite NIST, de la suite DIEHARD o la de Crypt-XS, así como el conjunto de pruebas de Donald Knuth, entre otras [17].

En este trabajo se le aplican al generador un conjunto simple de pruebas estadísticas, y para complementar su evaluación se considera el conjunto de pruebas estadísticas de la suite NIST [16]. La razón principal de escoger tal conjunto de pruebas estadísticas es que esta suite cuenta con propiedades que la hacen atractiva [16, 18]. Por ejemplo, es uniforme, se conforma por una serie de pruebas bien establecidas, aunado a la disponibilidad de un tratamiento matemático exhaustivo. Más aún, el código fuente de todas las pruebas de la suite está disponible al público y se actualiza de manera constante [16]. En particular, se menciona que tal suite puede ser considerado como un primer paso para determinar si el generador es o no adecuado para una particular aplicación de criptografía.

En principio las pruebas que se aplicaron al generador fueron la determinación del histograma en una dimensión para observar la distribución de los datos, la correlación entre los datos y la transformada de Fourier para ver sus propiedades espectrales.

La Figura 3.2 muestra el histograma en el rango [10, 60] y su respectivo perfil del histograma de una secuencia de un millón de enteros de 15 bits. Se puede apreciar que se tiene una distribución de los datos con una dispersión del tipo gaussiana, lo cual podemos considerar, para nuestros propósitos, como característica de un buen generador de números pseudoaleatorios.

Por otra parte, la Figura 3.3 nos muestra la relación entre los números en el instante n y los números un instante anterior $n-1$. El número de datos a considerar fue de 100, 000, pero para propósitos de visualización se muestran solo 35,000. Se puede apreciar que no se tiene

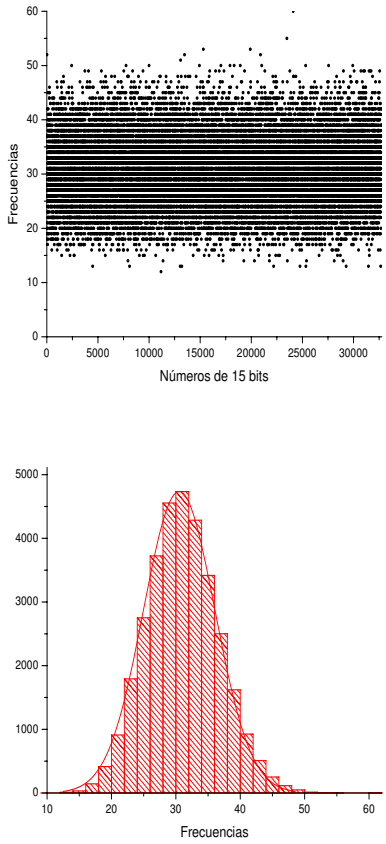

Figura 3.2: Histograma de una secuencia de 1 millón de enteros y su respectivo perfil.

correlación entre las muestras debido a que se tiene una distribución uniforme. En caso de existir una correlación, se tendría una agrupación de los datos asemejándose a una recta. De hecho, tienen un comportamiento muy parecido al que presenta el ruido blanco.

Por último, para la simple evaluación, se calculó la transformada de Fourier a una secuencia de 100,000 muestras. La Figura 3.4 muestra la transformada de Fourier normalizada de la secuencia considerada y su respectivo perfil. Se puede observar que los datos tienen un espectro extendido, es decir, están distribuidos en todas las frecuencias de manera uniforme y no se repiten en alguna frecuencia en particular. Los mismos cálculos se realizaron para un número

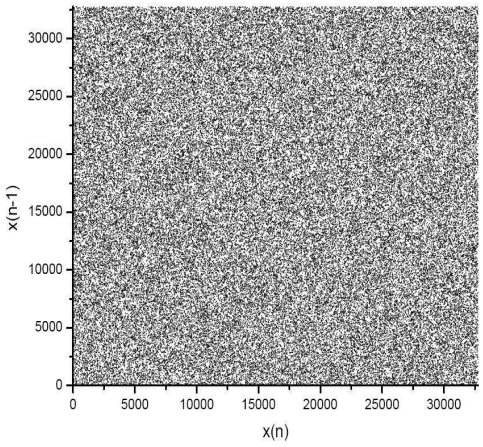

Figura 3.3: Distribución de los números en los instantes n y $n-1$.

mayor de muestras obteniendo resultados similares. Con las previas pruebas se puede decir que las secuencias generadas tienen un buen comportamiento aleatorio, su distribución de datos es uniforme, sin correlación y no se repiten en alguna frecuencia en particular.

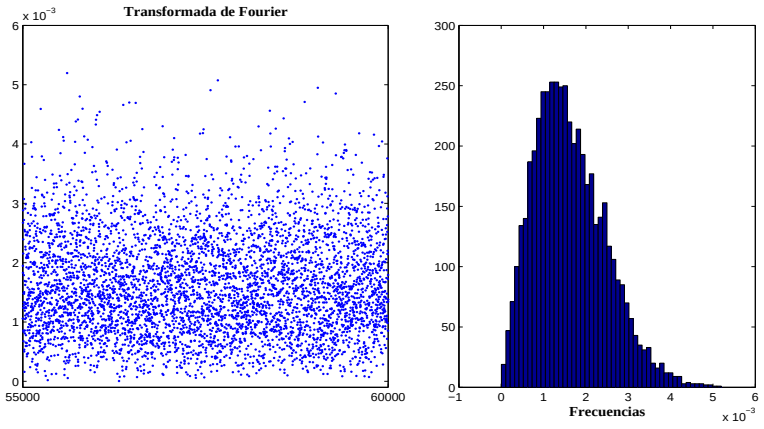

Figura 3.4: Transformada de Fourier y su respectivo perfil de una muestra de 5,000 datos en el intervalo 55,000 – 60,000.

Por otra parte, el conjunto de pruebas estadísticas de la NIST es un paquete estadístico que consiste de 15 pruebas que se enfocan en una variedad de diferentes tipos de no-aleatoriedad

que pudieran existir en una secuencia. Las pruebas son listadas en la Tabla 3.1 y para mayores detalles de las mismas, se recomienda al lector consultar la Referencia [16].

Cuadro 3.1: Lista de pruebas estadísticas de la suite NIST.

Number	Test name
1	The Frequency (Monobit) Test
2	Frequency Test within a Block
3	The Runs Test
4	Tests for the Longest-Run-of-Ones in a Block
5	The Binary Matrix Rank Test
6	The Discrete Fourier Transform (Spectral) Test
7	The Non-overlapping Template Matching Test
8	The Overlapping Template Matching Test
9	Maurer's "Universal Statistical" Test
10	The Linear Complexity Test
11	The Serial Test
12	The Approximate Entropy Test
13	The Cumulative Sums (Cusums) Test
14	The Random Excursions Test
15	The Random Excursions Variant Test

En el análisis del desempeño del generador consideramos $m = 100$ muestras de 10^6 secuencias de bits, donde cada secuencia de $N = 15$ bits se generaron a partir de una semilla escogida aleatoriamente para los casos del generador con una y tres transformaciones h.

Durante el proceso se calcularon los respectivos valores-P de cada secuencia para todas las pruebas contenidas en la suite del NIST. De acuerdo a la Referencia [16], un valor-P corresponde a la probabilidad (bajo la hipótesis nula de aleatoriedad) de que la prueba

estadística elegida asumirá valores que serán iguales o peores al valor estadístico de la prueba observado cuando se considera la hipótesis nula. Para el análisis de valores-P obtenidos para varias pruebas estadísticas se ha fijado el nivel significativo a $\alpha = 0.01$, lo que significa que se espera que alrededor del $1\,\%$ de las secuencias falle. Una secuencia pasa una prueba estadística cuando el valor-$P \geq \alpha$, y en caso contrario se dice que falla. Para cada prueba estadística la proporción de secuencias que pasan se calcula y analiza, respectivamente. Sin embargo, no es suficiente solo mirar las tasas de aceptación y declarar que el generador es aleatorio si estas lucen bien. Si las secuencias de prueba son realmente aleatorias, se espera que los valores-P calculados aparezcan uniformemente en el intervalo $[0,1]$. Para la interpretación de los resultados, la suite de la NIST ha adoptado dos enfoques, (I) la examinación de la proporción de secuencias que pasan una prueba estadística y (II) la distribución de los valores-P para verificar uniformidad.

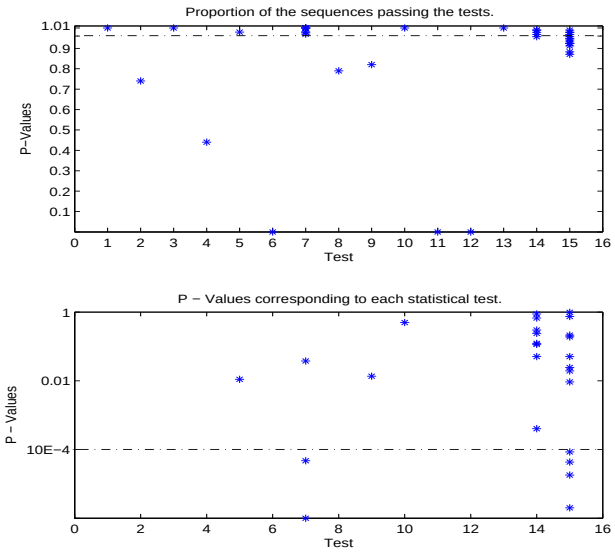

Figura 3.5: (a) Proporciones (ARRIBA), y (b) valores-P (ABAJO), correspondientes a $N = 7$ bits y una transformación h. Las líneas discontinuas separan las regiones de éxito y fallo.

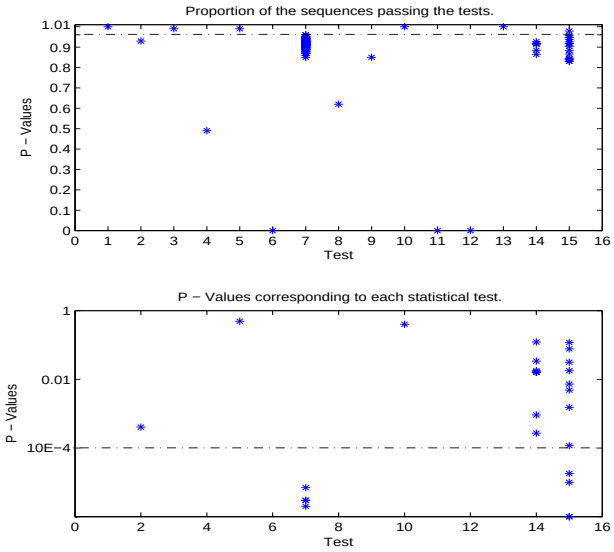

Figura 3.6: (a) Proporciones (ARRIBA), y (b) valores-P (ABAJO), correspondientes a $N = 7$ bits y tres transformaciones h. Las líneas discontinuas separan las regiones de éxito y fallo.

3.2.1 Resultados del paquete estadístico de la NIST

Considerando el paquete estadístico de la NIST, los dos análisis descritos anteriormente son aplicados y evaluados con el fin de determinar si las secuencias generadas son de carácter aleatorio o no. Para el análisis se han considerado $m = 100$ muestras de 10^6 secuencias de bits, donde cada secuencia ha sido generada a partir de una semilla escogida de manera aleatoria, donde la proporción debe ser mayor que $0.960150(\alpha = 0.01)$, mientras que $P - value_T \geq 0.0001$. Con la finalidad de investigar el desempeño del generador, se analizaron secuencias aleatorias para el caso de $N = 7$, $N = 15$ y $N = 31$ bits, considerando una y tres transformaciones.

Caso: $N = 7$ **bits**

En las Figuras 3.5 y 3.6 se muestran los resultados del paquete de la NIST para el caso de $N = 7$ bits, usando una y tres transformaciones, de manera respectiva. Podemos observar que para este caso se presenta un pobre desempeño; las secuencias aleatorias generadas sólo pasan algunas pruebas, y más aún no están distribuidas uniformemente.

Caso: $N = 15$ **bits**

Las Figuras 3.7 y 3.8 ilustran los resultados para $N = 15$ bits para una y tres transformaciones, respectivamente. Para este caso se presenta un mejor desempeño comparado con el caso anterior. Se puede observar que usando una transformación, el generador no pasa todas las pruebas, ver Figura 3.7 (a), a pesar de que se encuentra uniformemente distribuida, Figura 3.7(b).

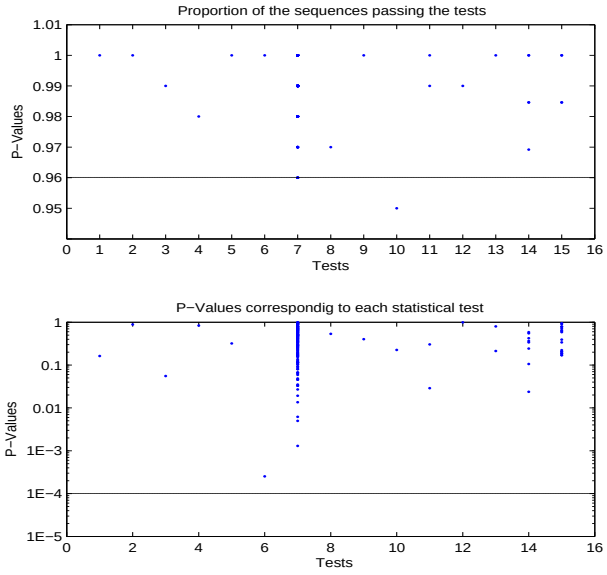

Figura 3.7: (a) Proporciones (ARRIBA), y (b) valores-P (ABAJO), correspondientes a $N = 15$ bits y una transformación h. Las líneas discontinuas separan las regiones de éxito y de fallo.

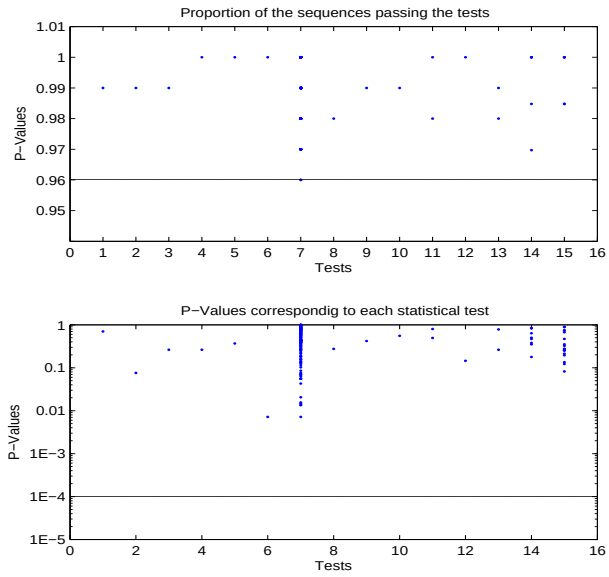

Figura 3.8: (a) Proporciones (ARRIBA), y (b) valores-P (ABAJO), correspondientes a $N = 15$ bits y tres transformaciones h. Las líneas discontinuas separan las regiones de éxito y de fallo.

Caso: $N = 31$ **bits**

Para este caso, las Figuras 3.9 y 3.10, muestran los resultados para $N = 31$ bits para una y tres transformaciones, respectivamente. Como se puede observar, no se presentan fallas en todas las pruebas independientemente del número de transformaciones.

Como resultado principal, se pudo observar que este generador puede generar números aleatorios de calidad alta usando una o tres transformaciones h cuando el tamaño de las llaves incrementa, es decir, entre más larga la longitud de los números generados, mejor la calidad de los números aleatorios que se obtienen [8].

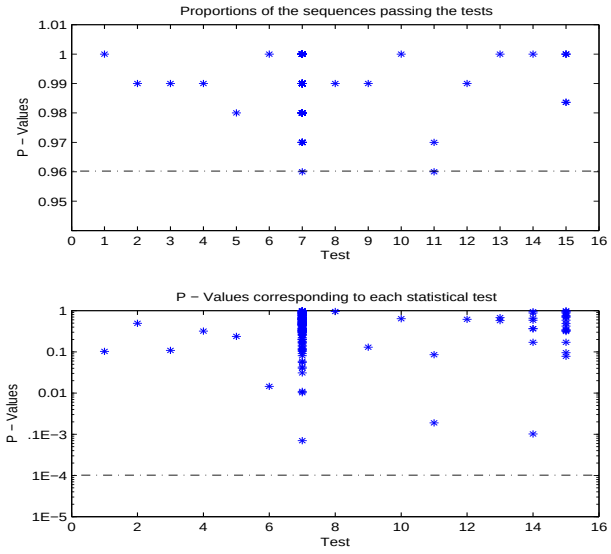

Figura 3.9: (a) Proporciones (ARRIBA), y (b) valores-P (ABAJO), correspondientes a $N = 31$ bits y una transformación h. Las líneas discontinuas separan las regiones de éxito y fallo.

3.3 Propiedades multifractales de las matrices del generador de secuencias pseudo-aleatorias

En estudios recientes se han podido determinar propiedades multifractales de autómatas celulares para un cierto conjunto de reglas, ver [11,12,13,14]. En particular, en la Referencia [9] se usó el método conocido como análisis multifractal de fluctuaciones sin tendencia basado en la transformada discreta de wavelet (WT-MFDFA, por sus siglas en inglés). Tal método les permitió cuantificar el comportamiento intrínseco multifractal de los ECAs para las reglas 90, 105 y 150. El procedimiento del WT-MFDFA se fundamenta en el cálculo de la función de fluctuación de q-ésimo orden, la cual se define como

$$F_q(s;m) = \left\{ \frac{1}{2M_s} \sum_{\nu=1}^{2M_s} |F^2(\nu,s;m)|^{q/2} \right\}^{1/q} \tag{3.13}$$

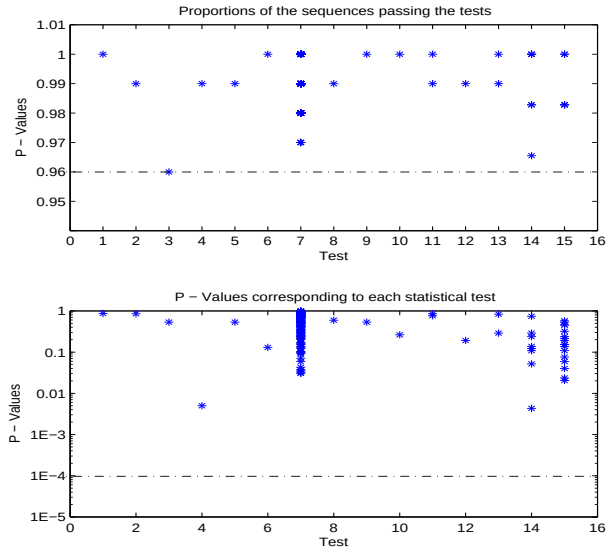

Figura 3.10: (a) Proporciones (ARRIBA), y (b) valores-P (ABAJO), correspondientes a $N = 31$ bits y tres transformaciones h. Las líneas discontinuas separan las regiones de éxito y fallo.

donde $q \in \mathbb{Z}$ con $q \neq 0$. El comportamiento divergente que se presenta cuando $q \to 0$ se puede evitar al utilizar un promedio logarítmico $F_0(s;m) = \exp\left\{ \dfrac{1}{2M_s} \displaystyle\sum_{\nu=1}^{2M_s} \ln|F^2(\nu,s;m)| \right\}$, tal y como se hizó en [9, 10, 11]. Con el objetivo de determinar el comportamiento de escala fractal de la serie de tiempo, la función de fluctuación debe presentar la ley de escala

$$F_q(s;m) \sim s^{h(q)}. \tag{3.14}$$

De la expresión anterior, el exponente $h(q)$ es considerado como el exponente de Hurst generalizado debido a que depende de q, donde el valor de $h(q=2)$ representa al exponente de Hurst. Asimismo, si $h(q)$ presenta un comportamiento constante para todo valor de q, entonces la serie de tiempo que se analiza es de carácter monofractal, mientras que para un valor no constante de $h(q)$ se tiene que la serie presenta un comportamiento multifractal. En el caso anterior, se podrían calcular otras cantidades que auxilian a describir las características

multifractales que presenta la información analizada. Por ejemplo, se puede calcular el espectro de singularidades de Hölder $f(\alpha)$ correspondiente a una señal o distribución $g(t)$, así como el exponente de escala $\tau(q)$, cantidades que se relacionan mediante una transformada de Legendre[12], la cual se expresa como

$$\alpha = \frac{d\tau(q)}{dq}, \qquad \text{y} \qquad f(\alpha) = q\alpha - \tau(q), \tag{3.15}$$

donde α caracteriza la "fuerza" de las singularidades, y el espectro de dimensiones Hölder $f(\alpha)$ es una función convexa no negativa que es soportada sobre un intervalo cerrado $[\alpha_{\min}, \alpha_{\max}]$. De hecho, el espectro $f(\alpha)$ puede ser interpretado como la dimensión fractal de Hausdorff del subconjunto de información que se caracteriza mediante el exponente de Hölder α[13, 14]. La singularidad más "frecuente", que corresponde al valor máximo de $f(\alpha)$, ocurre en el valor de $\alpha(q = 0)$, mientras que los valores de los extremos del soporte, $\alpha_{\min}$ para $q > 0$ y $\alpha_{\max}$ para $q < 0$, corresponden a las singularidades más fuerte y más débil, de manera respectiva. Por otra parte, si $\tau(q)$ presenta un comportamiento lineal es un indicativo que indica monofractalidad, mientras que un comportamiento no lineal indicaría que la señal analizada es de carácter multifractal. Más aún, se cuenta con una relación entre el exponente de escala $\tau(q)$ y los exponentes generalizado de Hurst $h(q)$, la cual está dada por $\tau(q) = qh(q) - 1$[10].

Dado que la evolución de la secuencia de la matriz $\mathbf{H}_N$ se encuentra basada en la evolución de la regla 90 de la CA, la estructura de los patrones de bits de esta última están reflejados directamente en la estructura de las entradas de $\mathbf{H}_N$. Por lo que al considerar el algoritmo WT-MFDFA, se analizan la suma de la densidad de unos en las secuencias de las filas de la matriz $\mathbf{H}_N$, donde la función wavelet db-4, perteneciente a la familia Daubechies, fue la que se utiliza en los cálculos de dicho algoritmo.

En la Figura 3.11 se ilustran los resultados de la densidad de unos por fila de la matriz $\mathbf{H}_{1023}$. En (a) se muestran sólo los primeros 2^8 puntos son mostrados del conjunto completo de $2^{10} - 1$ puntos de datos. En (b) y (c) se muestran los perfiles obtenidos con el método de WT-MFDFA de

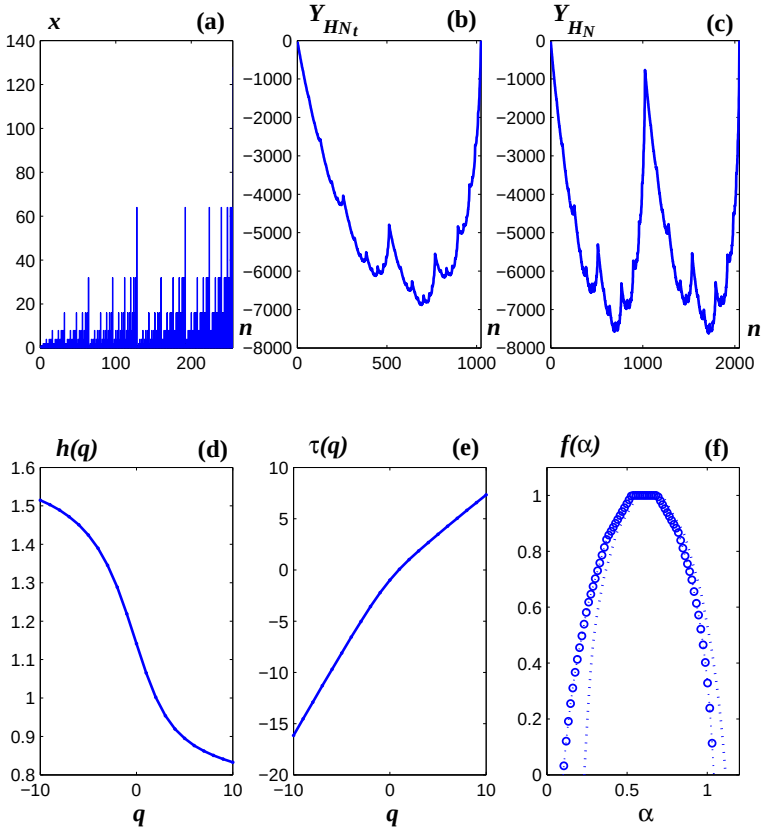

Figura 3.11: (a) Serie de tiempo correspondiente a la densidad de unos por fila de la matriz $\mathbf{H}_{1023}$, donde sólo los primeros 2^8 puntos son mostrados del conjunto completo de $2^{10}-1$ puntos de datos. Perfiles correspondiente de (b) $\mathbf{H}_{N_t}$ y (c) $\mathbf{H}_N$. (d) Exponente generalizado de Hurst $h_(q)$, (e) exponente de escala τ y (f) el espectro de singularidad $f(\alpha)$. Las gráficas punteadas corresponden al análisis MFDFA sin usar wavelets.

$\mathbf{H}_{N_t}$ y $\mathbf{H}_N$, respectivamente. Asimismo se muestran las cantidades multifractales (d) $h(q)$, (e) τ, y (f) $f(\alpha)$ realizados con el WT-MFDFA, donde las gráficas punteadas corresponden al análisis MFDFA sin usar wavelets. En las tres últimas gráficas se pueden confirmar las propiedades multifractales de la serie de tiempo debido a que $h(q)$ no es constante, obtenemos un espectro τ con dos pendientes, y se cuenta con un soporte de singularidades de $\Delta\alpha = \alpha_{\max} - \alpha_{\min}$ del espectro de singularidad parabólico $f(\alpha)$ sobre el eje α. En la literatura se ha podido observar que un indicativo o "fuerza" de multifractalidad en $f(\alpha)$ se indica con $\Delta(\alpha)$, donde

entre más grande sea Δ, más propiedades multifractal tiene la señal analizada. Para este caso $\Delta(\alpha_{H_{1023}}) = 1.16 - 0.212 = 0.948$, y la singularidad más frecuente ocurre en $\alpha_{\mathrm{mf}_{\mathrm{H}_{1023}}} = 0.694$.

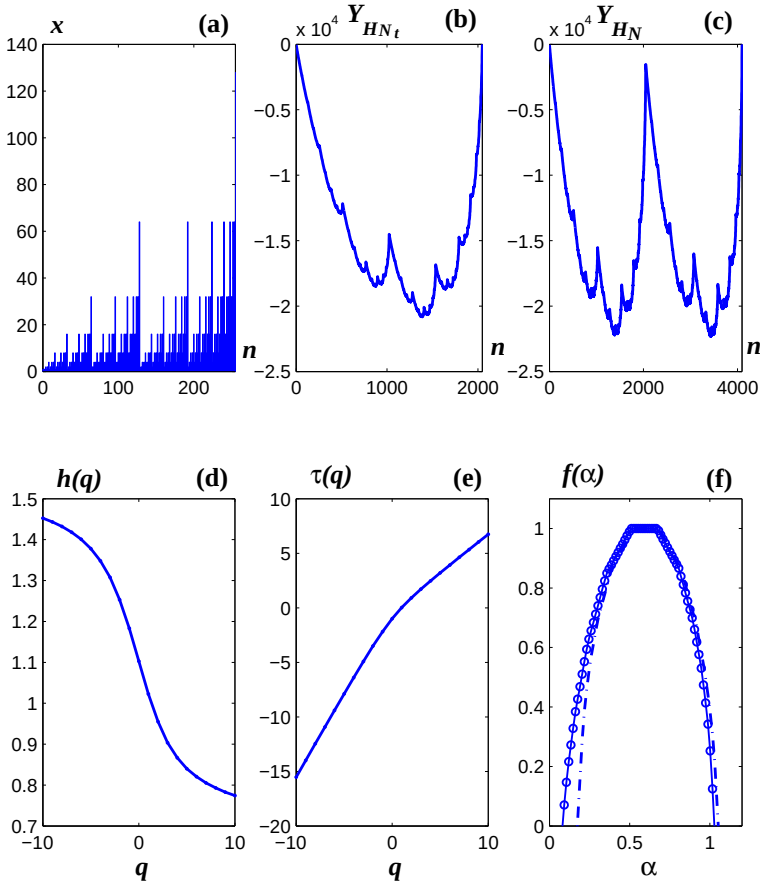

Figura 3.12: Misma leyenda que en la Figura 3.11, pero para caso de la matriz $\mathbf{H}_{2047}$.

De manera similar, los resultados para la matriz $\mathbf{H}_{2047}$ se muestran en la Figura 3.12. Podemos observar que los valores de la singularidad mas fuerte, $\alpha_{\min}$, y la más débil, $\alpha_{\max}$, son muy similares al caso anterior, así como el de la singularidad más frecuente. Estos resultados están en concordancia con los resultados obtenidos en la Referencia [9] para la regla 90, a pesar de que los resultados de la matriz superior presenta un ligero desplazamiento a la derecha. De hecho, este comportamiento es más evidente en los espectros correspondientes (línea punteada) de las señales de densidad de unos por fila de $\mathbf{H}_N$, los cuales son mostrados en las Figuras 3.11-3.12(f), mientras que en las Figuras 3.11-3.12 (d)y (e) no se percibe tal situación.

4

Implementación numérica y aplicación del ESAC

En este Capítulo se presenta la implementacion numérica del sistema de encriptación, aplicado a señales de audio que se comprimen por medio de la transformada wavelet. Esta implementación y aplicación nos permitirá tener un mejor panorama para el manejo apropiado de grandes cantidades de información de manera más segura.

4.1 Procedimiento de implementación numérica del ESAC

El procedimiento para llevar a cabo la implementación numérica del sistema de encriptación se fundamenta al considerar la matriz $\mathbf{Q}_N$ como principal elemento, la cual fue discutida en el Capitulo 3. Dicho procedimiento comprende los siguientes pasos:

1. Proporcionar el valor de n para considerar secuencias a encriptar de longitud $N = (2^n - 1)$ bits.
2. Generar la matriz $\mathbf{Q}_N$.
3. Calcular las semillas iniciales de tamaño N y $(N+1)$ bits.

4. Proporcionar las semillas iniciales para generar las secuencias pseudoaleatorias con la matriz correspondiente para el generador de números pseudoaleatorios en términos de la matriz $\mathbf{Q}_N$.

5. Calcular la matriz $\mathbf{P}_N$ y escoger la tarea a realizar. para el caso de la encriptación, se utiliza la matriz $\mathbf{P}_N$ y la matriz $\mathbf{Q}_N$ del paso dos para calcular (3.1). Para el otro caso, las matrices $\mathbf{R}_N$ y $\mathbf{T}_N$ son generadas para realizar el proceso de desencriptación según la relación (3.5) .

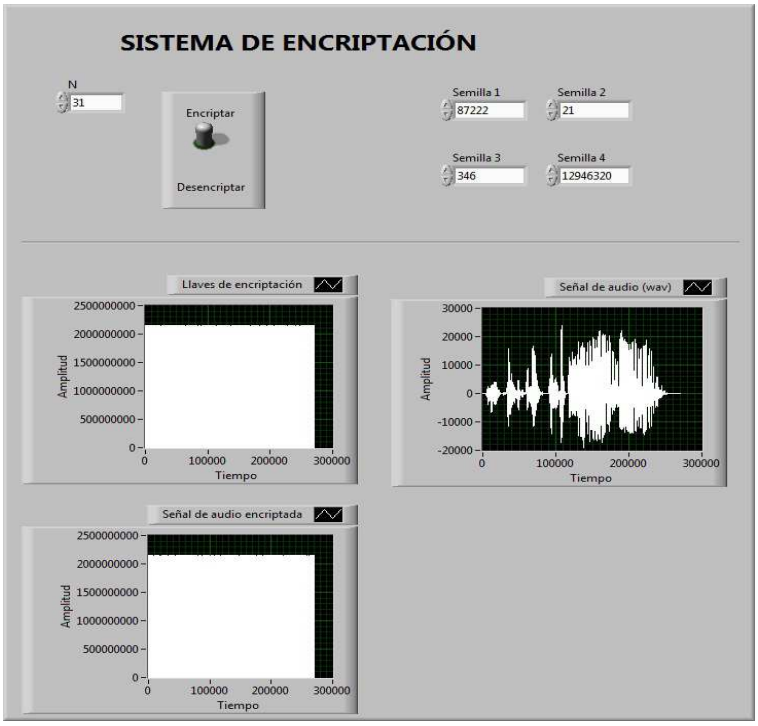

Figura 4.1: Implementación numérica del sistema de encriptación utilizando el lenguaje de programación gráfico LabVIEW de la compañia National Instruments.

Para llevar a cabo la implementación numérica del sistema ESAC se utilizo el lenguaje gráfico de programación LabVIEW de la compañía National Instruments. En la Figura 4.1 se presenta el instrumento virtual del sistema de encriptación, con los resultados obtenidos cuando el proceso

de encriptación es aplicado a una señal de audio. En esta ilustración se considera una longitud de $N = 31$ bits, donde se utiliza un generador pseudoaleatorio de tres trasformaciones.

Por otro lado, en la Figura 4.2 se muestran las familias de permutaciones indexadas para secuencias de tamaño de $N = 7$ bits, Si se quiere llevar a cabo el proceso de encriptación, se considera la Figura de la parte superior, y si es el proceso de desencriptación se tomará la parte inferior. En la parte derecha se pueden observar las respectivas expresiones booleanas de las palabras **c** y **m** con los resultados obtenidos, los cuales pueden ser comparadas con los resultados anteriores del Capítulo 2.

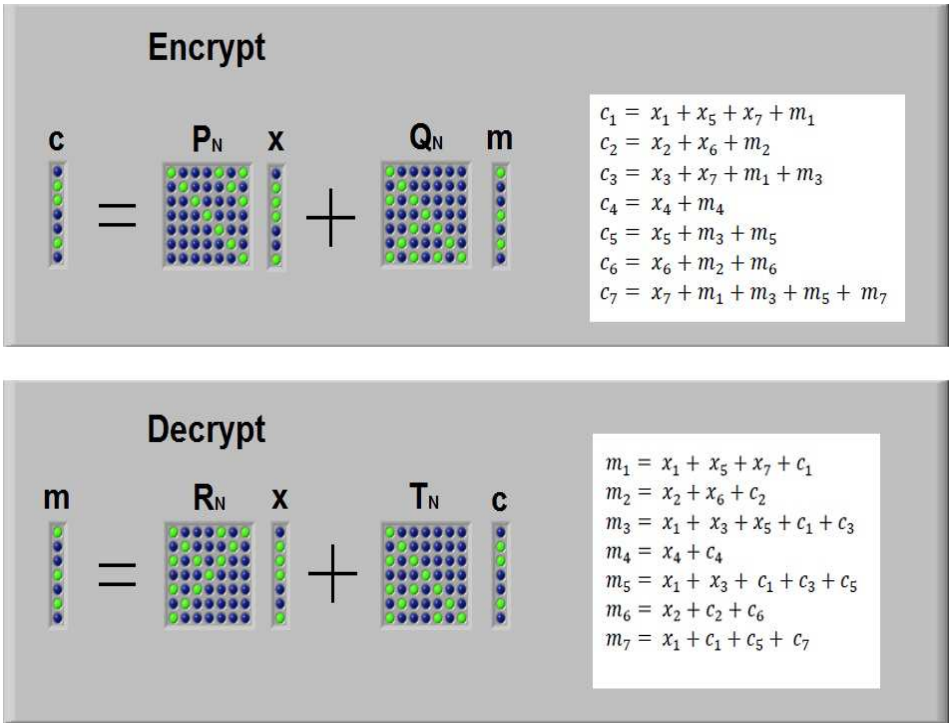

Figura 4.2: Implementación numérica de las familias de permutaciones indexadas $\mathbf{c} = \Psi_{\mathbf{x}}(\mathbf{m})$ (parte superior), and $\mathbf{m} = \Phi_{\mathbf{x}}(\mathbf{c})$ (parte inferior), considerando $N = 7$ bits.

Así mismo, en la Figura 4.3 se ilustran las evoluciones en la unidad básica encriptadora del generador de secuencias o llaves pseudoaleatorias con una y tres transformaciones para secuencias de tamaño de $N = 31$ bits. En Figura 4.3(a) se puede apreciar la evolución

en la unidad básica encriptadora de un generador con una transformación, donde las dos columnas izquierdas muestran las dos semillas $(\mathbf{x}, \mathbf{y})$, y en la parte superior se muestra la llave pseudoaleatoria generada $\mathbf{t}$, así como también se muestran los cálculos intermedios. Los puntos más claros representan un bit con valor de 1, mientras que los puntos obscuros corresponderían a un bit con valor de 0.

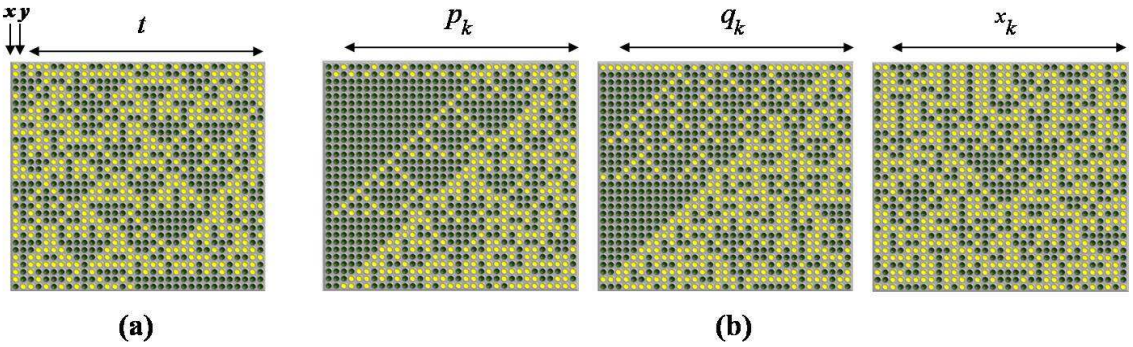

Figura 4.3: Evolución completa del ESAC para generar una secuencia aleatoria de 31 bits con (a) una transformación, y (b) tres transformaciones.

De manera similar, la Figura 4.3 (b) muestra la evolución completa del generador modificado con tres transformaciones del mismo tamaño que el caso de una transformación, donde la evolución completa para obtener la secuencia pseudoaleatoria $\mathbf{p}_k$ se encuentra en el lado izquierdo, mientras que para la secuencia $\mathbf{q}_k$ está en la parte media y para la secuencia de salida $\mathbf{x}_k$ en el lado derecho. Para un mayor detalle del funcionamiento de este generador en la Figura 4.4 se muestra la estructura de la evolución de las tres transformaciones h para generar la secuencia $\mathbf{x}_k$. En este caso, las secuencias $\mathbf{x}_i$ y $\mathbf{y}_i$, $i \in \{1,2\}$, son las palabras iniciales para cada transformación interna, h_1 y h_2. Con tales secuencias se generan las secuencias aleatorias $\mathbf{p}_k$ y $\mathbf{q}_k$ de longitud de N bits, que vienen siendo las secuencias iniciales de la tercer transformación h_3. Debido a que dichas transformaciones requieren dos palabras, una de N bits y otra de $(N+1)$ bits, el bit faltante es obtenido aplicando la operación módulo 2 entre los dos respectivos bits menos significativos, que se convierten en los bits respectivos más significativos de las secuencias previas de entrada. Es claro que $\mathbf{p}_k$, $\mathbf{q}_k$, y $\mathbf{x}_k$ son secuencias aleatorias, pero solo la

secuencia $\mathbf{x}_k$ es liberada.

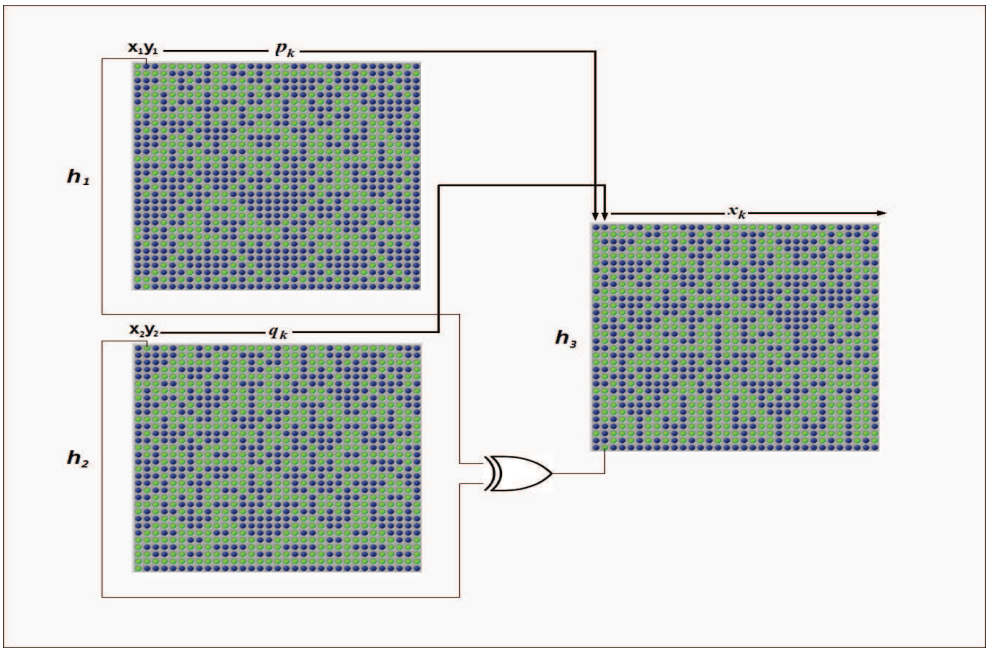

Figura 4.4: Evolución en el tiempo del generador pseudoaleatorio que hace uso de tres transformaciones acopladas para $N = 31$ bits.

4.2 Implemetacion numérica y aplicación del ESAC

Una vez implementado el sistema de encriptación, se procede a la aplicación del sistema ESAC a señales comprimidas de voz, que a su vez también fue implementada utilizando el lenguaje de programación gráfico LabVIEW. Antes de llevar a cabo tal aplicación, se describirá primero el procedimiento de comprimir señales de voz con la transformada wavelet, cuyas bases generales se pueden revisar en el Apéndice A.

4.2.1 Esquema de compresión

El método completo que se emplea en este trabajo es mostrado en la Figura 4.5. Al principio, y debido a que se consideran señales de voz, cuya longitud generalmente es grande, la señal original es segmentada en bloques de igual tamaño. Posteriormente se aplica la transformada wavelet a cada bloque de una longitud de 2^m muestras. Posteriormente, cada bloque transformado es sometido a un proceso de eliminación de los coeficientes transformados que se encuentran por abajo de un valor umbral.

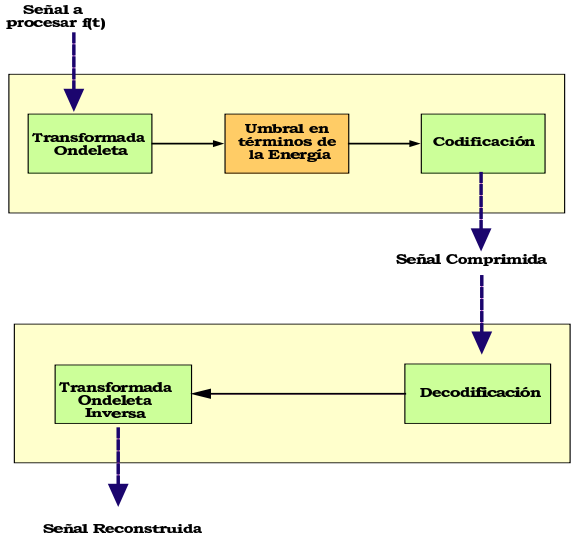

Figura 4.5: Esquema de compresión basado en la energía wavelet.

Para seleccionar o escoger un valor de umbral, se considera un criterio de energía de los coeficientes transformados. Al principio en se ordenan los coeficientes transformados en orden decreciente, esto es,

$$|y_1| \geq |y_2| \geq \cdots \geq |y_N|$$

donde $|y_1|$ denota el coeficiente más grande de la señal transformada, $|y_2|$ es el siguiente más

grande, y así sucesivamente. La cantidad de interés es la energía acumulada normalizada, la cual se define como

$$\mathscr{E}_{y,n} = \frac{\sum_{k=1}^{n} |y_k|^2}{\sum_{k=1}^{N} |y_k|^2}, \qquad n = 1, \ldots, N, \tag{4.1}$$

donde el denominador corresponde a la energía de la señal transformada Y, $\mathscr{E}_Y$. Una gráfica de $\mathscr{E}_{y,n}$ nos auxiliaría para indicar el número de coeficientes diferentes de cero, cuanto mejor sea la representación de la señal en términos de los coeficientes, más rápido la curva alcanzará el valor de uno. Teniendo lo anterior, el valor del umbral sería la magitud del coeficiente para el cual un porcentaje de la energía fue considerada en (4.1). Con el valor de umbral establecido ε cualquier coeficiente en los datos de la transformada wavelet cuyas magnitudes son menores que ε tendrán un valor de cero. Después del proceso de umbralización, los valores que no son cero de la señal transformada son agrupados en un vector, el cual debe tener un número menor de coeficientes que la original señal transformada. La cantidad de compresión obtenida puede ser controlada variando el parámetro de umbral ε.

Posteriormente, se calcula un mapa significativo el cual guarda la información de posición de los coeficientes de la señal transformada que "sobrevivieron" al valor del umbral. Si el coeficiente fue mayor que ε se acordó poner un valor de "1", y "0" en caso contrario. El mapa significativo nos permite agrupar los coeficiente significativos (coeficientes con valor diferente de cero) en un archivo separado y pueden ser comprimidos eficientemente usando, por ejemplo, un código de longitud variable basados en codificación run-length. Con lo cual se tendría una señal comprimida en términos de un criterio de energía de los coeficientes wavelet. Para el proceso inverso se realizan las tareas correspondientes.

4.2.2 Implementación numérica

La Figura 4.6 muestra un diagrama de bloques que describe la implementación numérica del sistema compresión-encriptación (SCE), que consiste de dos etapas. En la primera, los bloques de la parte superior llevan a cabo las tareas de compresión y encriptación de información. como se describe a continuación. Note que la señal resultante después de la etapa de compresión esta formada por dos señales, una señal sera los coeficientes que sobrevivieron a la umbralización, los cuáles serán la información que será únicamente encriptada, y la otra señal que es el vector binario que indica las posiciones originales de los coeficientes anteriores. La señal encriptada y el vector binario de posiciones estarían disponibles para ser transmitidas a través de un canal.

En la segunda etapa, los bloques inferiores realizan los preceson inversos: se realiza la desencriptación de la información encriptada, y posteriormente la etapa de descompresión toma lugar en la señal desencriptada adaptando el vector binario de posiciones para reconstruir la señal original.

En está implementación numérica, se analizan dos diferentes tipos de señales de voz, las cuales se denotan como $s1$ y $s2$. Ambas señales fueron grabadas como archivos wav con una resolución de 16 bits y una frecuencia de muestreo de 8 kHz y 44.1 kHz, respectivamente. Lo anterior nos da un total de $65,536$ muestras de $s1$ (aproximadamente 8 segundos), y $262,144$ muestras de $s2$ (aproximadamente 6 segundos) que fueron analizadas.

La Figura 4.7 muestra el instrumento virtual (VI) con los resultados obtenidos para la señal $s1$ considerando un criterio de energía de $90\,\%$ y utilizando la función wavelet de Haar en los cálculos de la transformada wavelet. En la parte (a) las señal $s1$ es mostrada, y la señal transformada se muestra en la parte (b). En la parte (c) se muestra los coeficientes wavelet encriptados que sobrevivieron en la etapa de compresión, con un pequeño número de muestras que la señal original, y finalmente en la parte (d) la señal recuperada es mostrada.

De manera similar, la Figura 4.8 muestra el VI con los resultados obtenidos para la señal $s2$

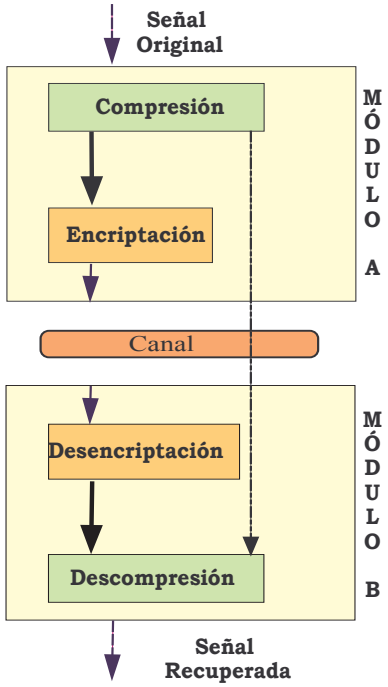

Figura 4.6: Esquema de sistema compresión-encriptación.

considerando el mismo criterio de energía de 90 %, pero ahora considerando la función wavelet de Daubechies db2 en los cálculos de la transformada wavelet.

Por completez, las tablas 4.1 y 4.2 muestran el rendimiento de compresión para diferentes porcentajes de energía considerados para las señales $s1$ y $s2$ respectivamente. Basados en estos resultados, se ha logrado buenas tasas de compresión para ambas señales con la función wavelet de Haar. Al escuchar la señal reconstruida, no hay mucha degradación en la calidad del sonido y el habla es comprensible. Se puede observar que a más altos porcentajes de energía una mejor calidad de sonido es obtenida.

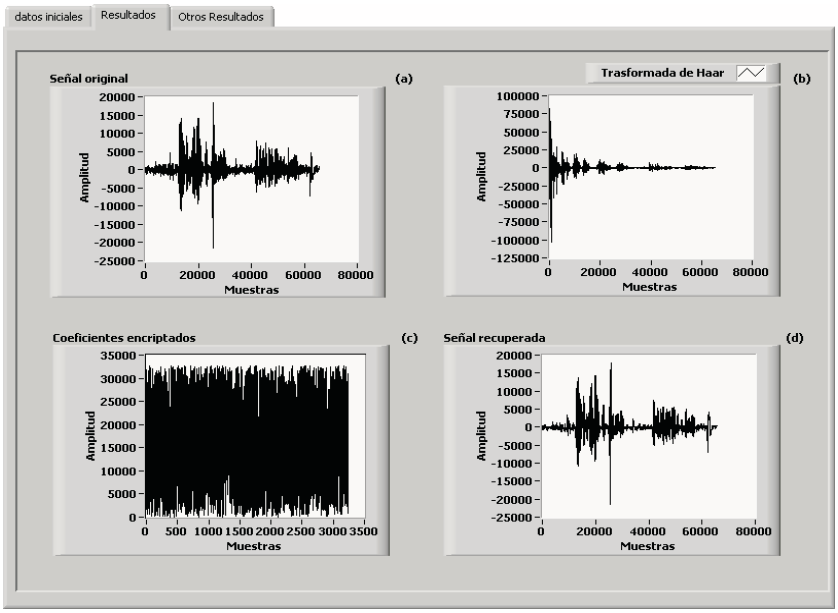

Figura 4.7: Análisis de la señal de voz $s1$ considerando un criterio de enrgía de 90 %. (a) Señal original $s1$, (b) transformada wavelet de Haar de la señal $s1$, (c) representación de los coeficientes encriptados, y (d) la señal recuperada.

Porcentaje de energía	Número de coeficientes	Relación de compresión
99 %	13913	4.71:1
95 %	5431	12:1
90 %	3231	20.28:1
85 %	2246	29.18:1
80 %	1627	40.28:1

Cuadro 4.1: Tasas de compresión para la señal $s1$.

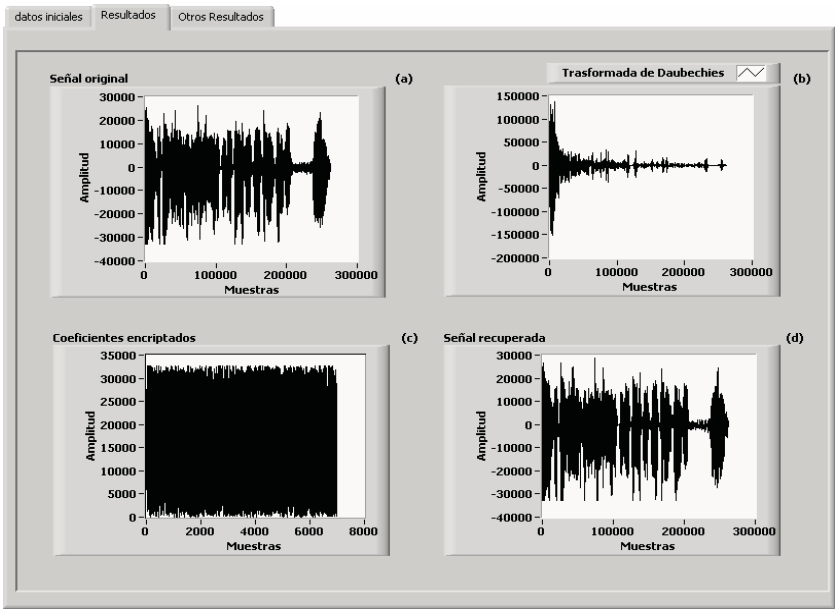

Figura 4.8: Análisis de la señal de voz $s2$ considerando un criterio de enrgía de 90 %. (a) Señal original $s2$, (b) transformada wavelet de Daubechies db2 de la señal $s2$, (c) representación de los coeficientes encriptados, y (d) la señal recuperada.

Porcentaje de energía	Número de coeficientes	Relación de compresión
99 %	48580	5.4:1
95 %	18769	13.97:1
90 %	10143	25.84:1
85 %	6448	40.65:1
80 %	4458	58.8:1

Cuadro 4.2: Tasas de compresión para la señal $s2$.

5

Conclusiones

Se presenta un enfoque matricial para la implementación flexible de un sistema de encriptación que se basa en el fenómeno de sincronización de autómatas celulares. Como primer paso se implementó y revisó la componente del generador de números aleatorios del sistema basado en la evolución hacia atrás en el tiempo de la regla 90 de los autómatas celulares. Este generador en su forma básica, usando una transformación, y su versión modificada, con tres transformaciones, son implementadas por medio de una matriz de secuencias $\mathbf{H}_N$, cuyas propiedades multifractales intrínsecas de la matriz para las dos versiones del generador se discuten, teniendo en mente su posible aplicación en el cripto-análisis. Además, la calidad de las secuencias aleatorias generadas se evaluaron usando un conjunto de típicas pruebas estadísticas, así como las del paquete estadístico de la NIST. De acuerdo a los resultados de las pruebas del conjunto de la NIST, se pudo establecer que la longitud de secuencias aleatorias es clave para tener secuencias aleatorias de alta calidad usando una o tres transformaciones. Entre más grande la longitud de los números generados es mejor la calidad de los mismos. En particular, para el valor de $N = 7$ bits el generador con una o tres transformaciones no pasa todas las pruebas de la NIST, mientras que el generador con una transformación falla en algunas pruebas cuando $N = 15$ bits, a pesar de que se pudo obtener secuencias aleatorias de 15 bits sin repetir con una longitud de periodo 2^{27} y 2^{31} considerando una y tres transformaciones, respectivamente. Estos

resultados son suficientes para muchas aplicaciones criptográficas, y se debe tener en cuenta que para longitudes de $N \geq 31$ bits ya no se presenta ningún problema con las pruebas del paquete estadístico de la NIST al usar una o tres transformaciones.

Este procedimiento con un enfoque matriz parece ser una forma atractiva y flexible para aplicar un sistema de encriptación ya sea completa o parcialmente. Creemos que esta propuesta puede ser una solución accesible y costeable para cuestiones de cifrado, ya que el criptosistema es un sistema flexible y reconfigurable que encaja con la tecnología digital de hoy en día con una amplia gama de aplicaciones. Además, el sistema es simple, rápido, y puede ser implementado fácilmente en un sistema de comunicación existente con mínimos requerimientos.

Se implemento sistema numérico que integra las etapas de compresión y encriptación de señales de voz. El esquema de compresión se encontró basado en la transformada wavelet de Haar, mientras que para el proceso de encriptación se consideró una versión del sistema encriptado propuesta en [49]. Por otro lado, la compresión de la transformada wavelet de Haar discreta tiene buenas tasas debido a que la alta concentración de energía se presentó en pocos coeficientes de la transformada de la señal de voz. Al mismo tiempo, el sistema de encriptación muestra una seguridad remarcable y flexibilidad para encriptar la información que contiene. Creemos que este sistema puede ser una herramienta útil en aplicaciones multimedia actuales. Además, el sistema implementado es simple, rápido, y puede ser implementado fácilmente en un sistema de comunicación existente con mínimos requerimientos.

A

Teoría Básica Wavelet

Para el procesamiento de señales existen diferentes técnicas para obtener "información" de las señales y en muchas ocasiones son complejas, pero que dependiendo del enfoque resultan ser muy útiles. La teoría de Fourier ha resultado ser muy eficiente para el procesamiento de señales. En particular, es una herramienta adecuada para varios tipos de señales cuyas propiedades estadísticas no varíen en el tiempo.

Sin embargo, la Transformada de Fourier (TF) para cierta clase de señales, como las no estacionarias, carece de localización en el tiempo y no proporciona una información adecuada. Una alternativa para el análisis de este tipo de señales es la Transformada Wavelet (TW), la cual es ideal para trabajar con señales no estacionarias y transitorias, debido a que brinda información en tiempo y frecuencia.

Además, surge como una herramienta poderosa para procesar señales que involucran grandes cantidades de información, ya que ha sido eficiente en la compresión de datos, permitiéndonos manipular y almacenar dicha información de manera más adecuada.

A.1 Teoría Wavelet

Una función wavelet se define como una onda "pequeña"que oscila en un pequeño intervalo de tiempo. La wavelet es una función $\psi(t) \in L^2$ real o compleja que dependiendo de la aplicación satisface ciertas condiciones:

1. ψ debe tener promedio cero
$$\int_{-\infty}^{\infty} \psi(t)dt = 0, \tag{A.1}$$
lo cual implica que es una función oscilatoria.

2. La función debe decaer con respecto al tiempo,
$$\lim_{t\to\infty} |\psi(t)| = 0. \tag{A.2}$$

Al aplicar la operaciones de escalamiento y traslación a la función $\psi(t)$ se puede generar una familia de funciones definidas como

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right), \tag{A.3}$$

donde a es el parámetro real de escala que debe cumplir con $a > 0$, mientras que b corresponde al parámetro real de traslación.

Dada una función de energía finita $f(t) \in L^2(\mathbb{R})$, podemos definir la **Transformada Wavelet Continua**(TWC) de f como

$$W_\psi f(a,b) = \int_{-\infty}^{\infty} f(t)\psi^*_{a,b}(t)dt, \tag{A.4}$$

donde el símbolo * significa complejo conjugado de la función. Además la TWC es una transformación reversible, por lo que para reconstruir la función f, se considera la siguiente expresión

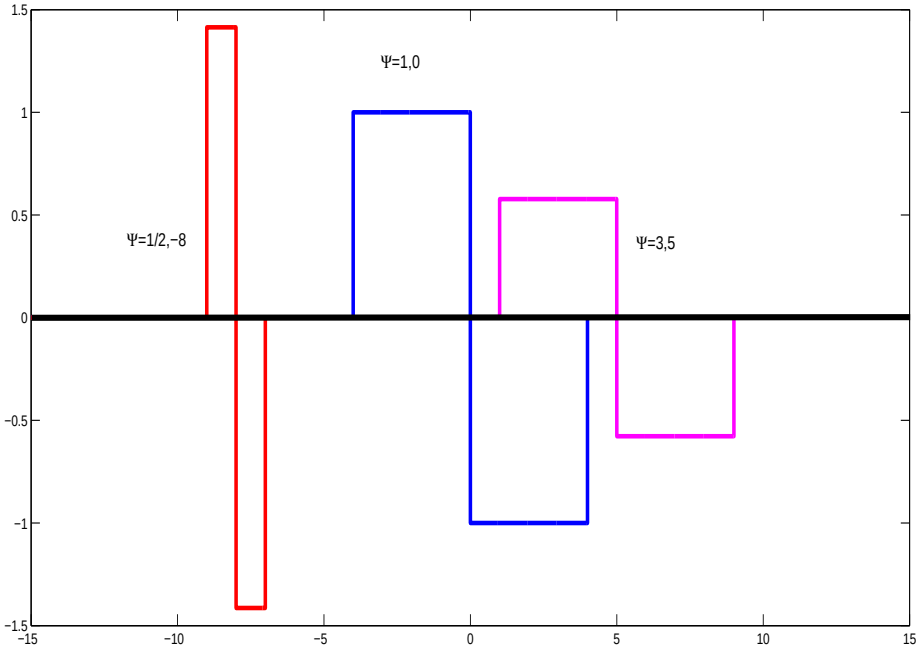

Figura A.1: Representación de algunas funciones wavelet escaladas y trasladadas de la familia de wavelets de Haar.

$$f(t) = \frac{1}{C_\psi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} W_\psi f(a,b) \psi_{a,b}(t) \frac{dadb}{a^2}. \tag{A.5}$$

En la actualidad, la mayoría de los cálculos se realizan mediante computador. Por tanto, resulta útil discretizar la TWC. Una de las maneras o formas de lograr una eficiencia en el cálculo de la TWC es la de discretizar los parámetros de escala y traslación (a,b) de la ecuación de la siguiente manera,

$$a = 2^{-j} \qquad \text{y} \qquad b = k2^{-j}, \tag{A.6}$$

donde $j,k \in \mathbb{Z}$. Así la familia de funciones wavelets queda de la siguiente manera,

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k), \tag{A.7}$$

que al sustituirla en la ecuación (A.4), tenemos que ahora los coeficientes son dados como

$$d_{j,k} = \int_{-\infty}^{\infty} f(t)\psi_{j,k}(t)dt. \tag{A.8}$$

Para la reconstrucción de la señal $f(t)$ a través de los coeficientes $d_{j,k}$ se tiene

$$f(t) = \sum_{j=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} d_{j,k}\psi_{j,k}(t). \tag{A.9}$$

A.2 Análisis Multi-Resolución

De manera general, el análisis multi-resolución (AMR) utiliza un algoritmo para descomponer una señal $f(t)$ en elementos más simples, los cuales son llamados promedios y detalles, siendo los promedios donde se concentra la mayor información de la señal original $f(t)$

En base al AMR la descomposición de una señal finita $f(t) \in L^2(\mathbb{R})$ puede ser escrita mediante:

$$f(t) = \sum_{k=0}^{2^{j_0}-1} c_{j_0}[k]\varphi_{j_0,k}(t) + \sum_{j=j_0}^{J-1}\sum_{k=0}^{2^j-1} d_j[k]\psi_{j,k}(t), \tag{A.10}$$

donde los coeficientes son

$$c_j[k] = \int_{-\infty}^{\infty} f(t)\varphi_{j_0,k}(t)dt, \tag{A.11}$$

$$d_j[k] = \int_{-\infty}^{\infty} f(t)\psi_{k,j}(t)dt, \tag{A.12}$$

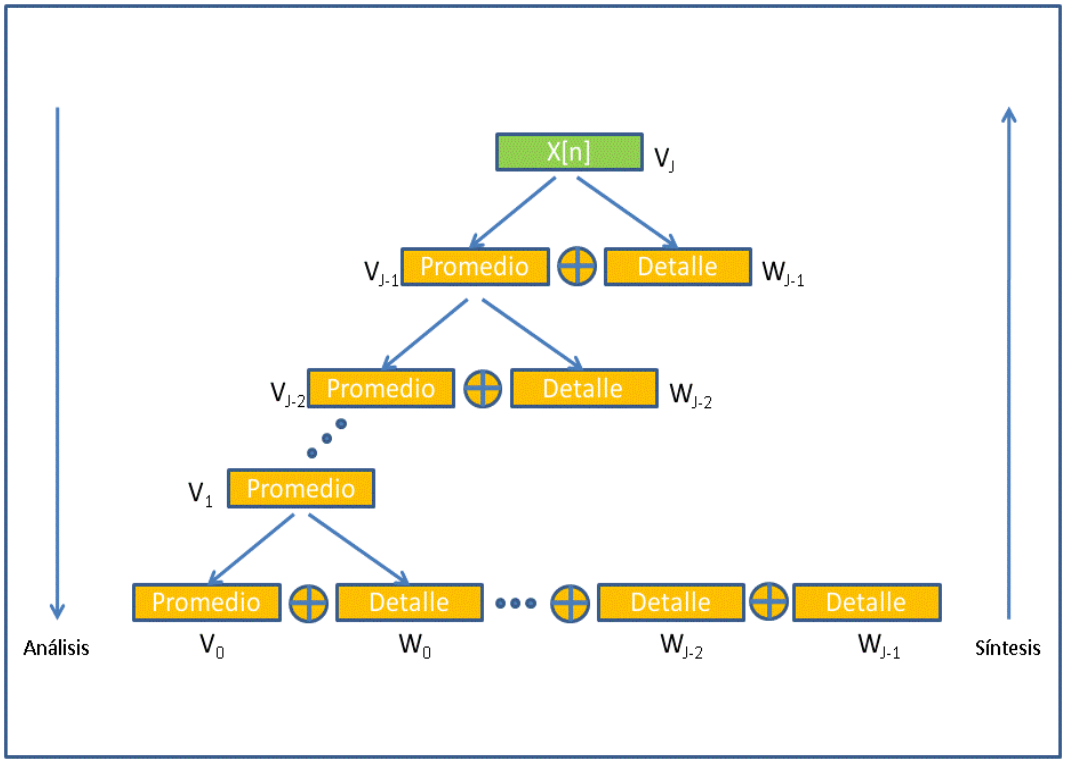

Figura A.2: Representación del Análisis Multi-Resolución.

A la ecuación (A.10) se le conoce como la **Transformada Wavelet Discreta (TWD)** de la función $f(t)$.

Para una implementación numérica eficiente de (A.11) y (A.12), se considera el Algoritmo de Mallat, el cual se basa en

$$c_j[k] = \sum_{l=0}^{m-1} h[l-2k]c_{j+1}[l], \quad d_j[k] = \sum_{l=0}^{m-1} g[l-2k]c_{j+1}[l] \tag{A.13}$$

para el análisis de la señal a tratar, mientras que para la síntesis se considera

$$c_{j+1}[k] = \sum_{l\in\mathbb{Z}} (h[k-2l]c_j[l] + g[k-2l]d_j[l]). \tag{A.14}$$

Dicho algoritmo es conocido también como la **Transformada Rápida Wavelet (TRW)**.

Por otra parte, la Transformada Wavelet en sus versiones continua y discreta, tienen una relación de conservación de energía.

$$E_f = \sum_n |f[n]|^2. \tag{A.15}$$

Y la energía acumulativa de una señal se define como

$$Ec_f = \left[\frac{f[1]^2}{E_f}, \frac{f[1]^2 + f[2]^2}{E_f}, \frac{f[1]^2 + f[2]^2 + f[3]^2}{E_f}, \ldots, 1 \right], \tag{A.16}$$

donde $E_f \neq 0$

Para el caso discreto en el que las funciones $\varphi_{k,j}(t)$ y $\psi_{k,j}(t)$ formen bases ortonormales, la energía de f en (A.10) está dada como

$$E_f = \sum_{k=1}^{2^{j_0}} |c_{j_0}[k]|^2 + \sum_{j=j_0}^{J-1} \sum_{k=1}^{2^j} |d_j[k]|^2 \tag{A.17}$$

Bibliografía

[1] Christof Paar and Jan Pelzl, *Understandig Criptography*, 1^a ed., Springer, 2010.

[2] Manuel José Lucena López l. *Criptografía y Seguridad en Computadores*, 3^a ed., Springer, 2002.

[3] M. Mejía and J. Urías, *An Asymptotically Perfect Pseudorandom Generator*, Discrete and Continuos Dynamical Systems, 7, 115-126 (2001).

[4] J. Urías, *An Cryptography primitives based on a cellular automaton*, appeared in Coding Theory, cryptography and related areas (J. Buchmann *et al*.; eds.), Springer, NY (2000).

[5] J. Urías, G. Salazar and E. Ugalde, *Synchronization of cellular automaton pairs*, Chaos. 8, 814-818 (1998)

[6] Feng Bao. *Cryptanalysis of a Partially Known Cellular Automata Cryptosystem*, IEEE TRANSACTIONS ON COMPUTERS, VOL. 53, NO. 11, NOVEMBER 2004.

[7] J. S. Murguía, G. Flores-Eraña, M. Mejía Carlos and H. C. Rosu, *Int. J. Mod. Phys. C* **23**, 1250078 (2012).

[8] J. S. Murguía, M. Mejía Carlos, H. C. Rosu, and G. Flores-Eraña, *Int. J. Mod. Phys. C* **21**, 741 (2010).

[9] J. S. Murguia, J. E. Perez-Terrazas, and H. C. Rosu, *Europhysics Letters*, **87**, 28003 (2009).

[10] J. W. Kantelhardt, S. A.Zschinegner, E. Koscielny-Bunde, S. Havlin, A. Bunde, and H. E. Stanley, *Physica A*, **316**, 87 (2002).

[11] L. Telesca, G. Colangelo, V. Lapenna, and M. Macchiato, *Phys. Lett. A*, **332** 398 (2004).

[12] T.C. Halsey, M.H. Jensen, L.P. Kadanoff, I. Procaccia, B.I. Shraiman, *Phys. Rev. A* **33**, 1141 (1986).

[13] J.F. Muzy, E.Bacry, and A. Arneodo, *Int. J. of Bifurcation and Chaos* **4**, 245 (1994).

[14] J.S. Murguía, J. Urías, *Chaos* **11**, 858 (2001).

[15] M. Mejía, J. Urías, *Discrete Cont. Dynamical Sys.* **7**, 115 (2001).

[16] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray and S. Vo, *NIST Special Pub. 800-22 Rev. 1*, http://csrc.nist.gov/rng/ (2008).

[17] V. Patidar and K. K. Sud, *Electronic Journal of Theoretical Physics*, **6**, 327 (2009).

[18] C. Kenny, *Random Number Generators: An Evaluation and Comparison of Random.org and Some Commonly Used Generators*, Trinity Collage Dublin, Management Sciencie and Information Systems Studies Project Report (2005).

[19] C. Paar, and J. Pelzl, *Understanding Cryptography: A Textbook for Students and Practitioners* (Springer-Verlag Berlin Heidelberg, 2010).

[20] T. W. Cusick, and P. Stanica, *Cryptographic Boolean Functions and Applications* (Academic Press, 2009).

[21] C. K. Koc, *Cryptographic engineering* (Springer Science+Business Media, LLC, 2009).

[22] L. Kocarev, *Circuits and Systems Magazine, IEEE* **1**, 6 (2001).

[23] N. Masuda, and K. Aihara, *IEEE Transactions on Circuits and Systems-I* **49**, 28 (2002).

[24] S. Wolfram, *Lecture Notes Comput. Sci.* **218**, 429 (1986).

[25] S. Nandi, B. K. Kar, and P. P. Chaudhuri, *IEEE Transactions on Computer* **43**, 1346 (1994).

[26] M. Sipper, and M. Tomassini, *Int. J. Mod. Phys. C* **7**, 181 (1996).

[27] J. Urías, E. Ugalde, and G. Salazar, *Chaos* **8**, 819 (1998).

[28] F. Seredynski, P. Bouvry, and A. Y. Zomaya, *Parallel Computing* **30**, 753 (2004).

[29] A. Fúster-Sabater, and P. Caballero-Gil, *Applied Soft Computing* **11**, 1876 (2011).

[30] S. Das, and D. R. Chowdhury, *Lecture Notes in Comput. Sci.* **6584**, 77 (2011).

[31] G. Zied, M. Mohsen, Z. Medien, and T. Rached, *Int. J. Comput. Sci. Eng. Sys.* **2**, 185 (2008).

[32] P. D. Hortensius, R. D. McLeod, and H. C. Card, *IEEE Transactions on Computer* **38**, 1466 (1989).

[33] C. Swenson, *Modern Cryptanalysis, Techniques for advanced code breaking* (Wiley Publishing, Inc., Indianapolis, 2008).

[34] O. Martin, A. M. Odlyzko, and S. Wolfram, *Communications in Mathematical Physics* **93**, 219 (1984).

[35] B. K. Kar, A. Gupta, and P. P. Chaudhuri, *Information Sciences* **72**, 83 (1993).

[36] J. Urías, G. Salazar, and E. Ugalde, *Chaos* **8**, 814 (1998).

[37] M. Mejía Carlos, Ph. D. Thesis, Universidad Autónoma de San Luis Potosí, SLP (2001).

[38] J. Urías, E. Ugalde and G. Salazar, *Chaos*, **8**, 819 (1998).

[39] M. Sipper and M. Tomassini, *International Journal of Modern Physics C*, **7**, 181 (1996).

[40] F. Seredynski, P. Bouvry and A. Y. Zomaya, *Parallel Computing*, **30**, 753 (2004).

[41] G. Zied, M. Mohsen, Z. Medien and T. Rached, *International Journal of Computer Sciences and Engineering Systems*, **2**, 185 (2008).

[42] S. Wolfram, *Lecture Notes in Computer Science*, **218**, 429 (1986).

[43] P. D. Hortensius, R. D. McLeod, D. M. Miller, and H. C. Card, *IEEE Transactions on Computer*, **38**, 1466 (1989).

[44] S. Nandi, B.K. Kar and P.P. Chaudhuri, *IEEE Transactions on Computer*, **43**, 1346 (1994).

[45] J. R. Sanchez, *International Journal of Modern Physics C*, **14**, 491 (2003).

[46] J.R. Sanchez and R. Alonso-Sanz, *International Journal of Modern Physics C*, **15**, 1461 (2004).

[47] J. Nagler and J.C. Claussen, *Phys. Rev. E*, **71**, 067103 (2005).

[48] S. Mallat, *A Wavelet Tour of Signal Processing*, (Academic Press, 2nd. Ed., 1999).

[49] Mario Alberto Almazán Montelongo, Master. Thesis, Universidad Autónoma de San Luis Potosí, SLP (2007).