Casa abierta al tiempo

Universidad Autónoma Metropolitana-Unidad Iztapalapa

División de Ciencias Básicas e Ingeniería

Posgrado en Ciencias y Tecnologías de la Información

Estudio y adaptación del algoritmo criptográfico AES para radios cognitivos

Idónea Comunicación de Resultados para obtener el grado de

Maestra en Ciencias y Tecnologías de la Información

Presentado por:

Cuevas Papalotzin H. Cristina
2173802172

Asesores: Dr. Enrique Rodríguez de la Colina
Dr. Leonardo Palacios Luengas

Jurado Calificador
Presidente: Dra. Mariko Nakano Miyatake
Secretario: Dr. Ricardo Marcelín Jiménez
Vocal: Dr. Leonardo Palacios Luengas

Ciudad de México, noviembre de 2019

Agradecimientos

Al Dr. Enrique Rodríguez quien me animó a continuar aprendiendo sobre radios cognitivos y al Dr. Leonardo Palacios quien me mostro el maravilloso mundo de la criptografía. Ustedes facilitaron mi aprendizaje e investigación compartiéndome sus conocimientos, me guiaron en este proceso y creyeron en mí. Gracias a su ayuda, esto es posible y gran parte de este trabajo se lo debo a ustedes.

A mis sinodales Dra. Mariko Nakano y Dr. Ricardo Marcelín, por sus consejos, por su paciencia y por el tiempo que le dedicaron a la revisión de este trabajo.

A mis padres Rogelio Cuevas y Magdalena Papalotzin quienes con su amor y esfuerzo me han permitido llegar a cumplir hoy un sueño más, gracias por inculcar en mí el ejemplo de esfuerzo y valentía.

A mis hermanos Rogelio, Daniela y César, por enseñarme tolerancia y a ver las cosas desde otro punto de vista.

A José Espinoza y Oscar Huerta por apoyarme cuando más lo necesito, por extender su mano en momentos difíciles, por creer en mi aun cuando yo misma dude y por el amor brindado cada día.

Resumen

El espectro radioeléctrico es un recurso limitado, al que se le ha dado una mala administración ya que se asigna de forma estática y por largos periodos de tiempo a usuarios específicos. Estas regulaciones tienen el propósito de controlar el uso del espectro, pero resultan ineficientes. A esto le sumamos que en actualidad existe una sobrepoblación de dispositivos para las comunicaciones inalámbricas, y obtenemos problema de escasez y subutilización de dicho recurso.

El paradigma propuesto para mitigar estos problemas es la radio cognitiva ya que usa el espectro de manera dinámica y oportunista, lo que implica lapsos de transmisión de datos cortos y complica la transmisión de recursos multimedia como imágenes debido al alto volumen de información que contienen. Además, el uso del espectro de forma dinámica y oportunista hace que la transmisión sea vulnerable a ataques como el de emulación del usuario primario donde el atacante se hace pasar por usuario con licencia y el ataque de función objetiva donde el atacante trata de cambiar los parámetros del Radio Cognitivo (Cognitive Radio, CR).

Este trabajo propone adaptar el Estándar de Cifrado Avanzado (Advanced Encryption Standard, AES) para proteger imágenes que se transmiten por un canal de comunicaciones inalámbrico cognitivo. La propuesta fue implementada a partir de AES-Contador, se modificó una de las operaciones de AES-Estándar y se redujo el número de rondas para mejorar el cifrado esto debido a que en una red con características cognitivas durante la transmisión de información existen múltiples interrupciones y esto hace que los lapsos de tiempo de transmisión sean cortos. La disminución de las rondas fue compensada con la adición de la transformada caótica de Kent, ya que si se implementa de manera iterada puede ser utilizada como generador de secuencias de números pseudoaleatorios con propiedades caóticas.

Los resultados del algoritmo propuesto son comparados con AES-Estándar y AES-Contador en pruebas de rendimiento tales como: entropía, correlación y tiempo de ejecución en el proceso de cifrado; también se realizan pruebas estadísticas utilizando las baterías de prueba del NIST 800-22SP. Obteniendo en todas estas pruebas mejores resultados que los dos algoritmos de cifrado mencionados, por ejemplo con el algoritmo propuesto AES-Caos se obtiene una correlación 92% menor que la de AES-Estándar y 35% menor que AES-Contador y en las baterías de pruebas del NIST AES-Caos pasa el 86% de las pruebas, AES-Contador el 80% y AES-Estándar el 66% de las pruebas.

Contenido

Lista de Figuras

Acrónimos

Acrónimos	Inglés	Español
AES	Advanced Encryption Standard	Estándar de Cifrado Avanzado
AET	Approximate Entropy Test	Prueba de Entropía Aproximada
BMRT	Binary Matrix Rank Test	Prueba de Rango de Matriz Binaria
CBC	Cipher Block Chaining	Encadenamiento de Bloque Cifrado
CCR	Chaotic Cognitive Radio	Radio Cognitivo Caótico
CFB	Cipher Feedback	Realimentación de Cifrado
CR	Cognitive Radio	Radio Cognitivo
CRN	Cognitive Radio Network	Red de Radios Cognitivos
CTR	Counter	Contador
Cusums	Cumulative Sums Test	Prueba de Sumas Acumulativas
DES	Data Encryption Standard	Estándar de Cifrado de Datos
DFTT	Discrete Fourier Transform Test	Prueba de Transformada Discreta de Fourier
DoS	Denial of Service	Negación de Servicio
DSA	Dynamic Spectrum Access	Acceso Dinámico al Espectro
DTV	Digital Television	Televisión Digital
ECB	Electronic Codebook	Libro de Códigos Electrónico
FIPS	Federal Information Processing Standards	Estándares Federales de Procesamiento de Información
FMT	Frequency Monobit Test	Prueba de Frecuencia Monobit
FTB	Frequency Test within a Block	Prueba de Frecuencia Dentro de un Bloque
GRC	GNU Radio Companion	Compañero GNU Radio
JPEG	Joint Photographers Experts Group	Grupo Conjunto de Expertos en Fotógrafía
LCT	Linear Complexity Test	Prueba de complejidad lineal
LFSR	Linear Feedback Shift Register	Registro con Desplazamiento de Retroalimentación Lineal
MAC	Media Access Control	Control de Acceso al Medio
MT	Maurer's "Universal Statistical" Test	Prueba de "estadística universal" de Maurer
NIST	National Institute of Standards and Technology	Instituto Nacional de Estándares y Tecnología
NoTMT	Non-overlapping Template Matching Test	Prueba de Coincidencia de Plantillas no Superpuestas
OFB	Output Feedback	Realimentación de Salida
OTMT	Overlapping Template Matching Test	Prueba de coincidencia de plantillas superpuestas
PU	Primary Users	Usuario Primario
PUEA	Primary Users Emulation Attack	Ataque de Emulación del Usuario Primario

RET	Random Excursions Test	Prueba de excursiones aleatorias
REVT	Random Excursions Variant Test	Prueba Variante de Excursiones Aleatorias
RT	Runs Test	Prueba de Ejecuciones
ST	Serial Test	Prueba en Serie
SU	Secondary Users	Usuario Secundario
TLROB	Test for the Longest Run of Ones in a Block	Pruebas de Ejecuciones de Unos más Larga en Bloque
VSD	Vestigial Side Band	Banda Lateral Vestigial

1. Introducción

1.1 Motivación

En la actualidad el espectro radioeléctrico es regulado mediante políticas de asignación estáticas, es decir, el espectro está regulado por agencias gubernamentales; donde la mayor parte del espectro se asigna por largo plazo para grandes regiones geográficas a usuarios con licencia [1] [2]. Estas regulaciones tienen el propósito de controlar el uso del espectro, pero han derivado en un problema de escasez y subutilización [2] [3]. La radio cognitiva (Cognitive Radio, CR), es la tecnología propuesta para solucionar estos problemas ya que, en las redes de radios cognitivos (Cognitive Radio Network, CRN), los nodos inalámbricos cambian sus parámetros para comunicarse de manera eficiente, evitando la interferencia a los usuarios primarios (Primary Users, PU), y a otros Usuarios Secundarios (Secondary Users, SU) [4]. A pesar de que el CR se propuso como una solución prometedora para mejorar la utilización y la eficiencia del espectro, también conlleva riesgos y problemas de seguridad, ya que permiten el acceso dinámico al espectro (Dynamic Spectrum Access, DSA) [1].

En una CRN generalmente, los atacantes apuntan a tres capas durante la comunicación, es decir, física, enlace y red; pero la capa más vulnerable tiende a ser la física [5] en comparación con el sistema de comunicación inalámbrico convencional. Por lo que se desea proteger la información en el canal de comunicación de una CRN. Se considera que proteger la privacidad de la información que se envía por un canal de comunicaciones inalámbrico cognitivo, es igual de importante y necesario que en cualquier otro canal de comunicaciones. Asimismo es importante mencionar que por la forma en que opera una CRN el tiempo disponible para transmitir datos es corto por esto se propone que la información que se transmite en un canal de cognitivo sea protegida con el algoritmo AES ya que proporciona seguridad, mejor velocidad de cifrado y mayor rendimiento en comparación con otros algoritmos de cifrado simétrico [6].

Por otro lado, si se desea proteger imágenes, por ejemplo: imágenes médicas, educativas, sensoriales, etc., una opción es cifrar la imagen para ocultar su contenido, pero es sabido que las técnicas criptográficas tradicionales como el Estándar de Cifrado de Datos (Data Encryption Standard, DES) y el Estándar de Cifrado Avanzado (Advanced Encryption Standard, AES) no son aplicables a las imágenes debido a la redundancia y la alta correlación entre los píxeles [7].

Recapitulando existen dos importantes razones para hacer un estudio del algoritmo AES para radios cognitivos. La primera es la vulnerabilidad ocasionada por el acceso dinámico al espectro, esto se planea mitigar con AES, incrementando la velocidad de cifrado. La segunda, consiste en que AES no es aplicable a imágenes y se espera una transmisión segura de éstas, por lo que se modificó alguna de las operaciones del algoritmo estándar AES para una trasmisión segura de imágenes.

1.2 Objetivos

Objetivo general:

- Mejorar el algoritmo criptográfico AES para cifrar imágenes en un entorno con lapsos de tiempo de transmisión cortos similares a las que ocurren en Redes de Radios Cognitivos.

Objetivos particulares

- Diseñar una modificación del algoritmo AES, para su aplicación en imágenes.
- Comparar el desempeño del algoritmo propuesto con diferentes versiones del algoritmo AES.
- Implementar el algoritmo propuesto en GNU-Radio.

1.3 Metodología de la investigación

La modificación de AES fue implementada a partir de AES-Contador, se modificó una de las operaciones de AES-Estándar y se redujo el número de rondas para mejorar el tiempo de cifrado, debido a que en una red de radio cognitivo existen muchas interrupciones durante la transmisión, que limitan los tiempos efectivos de transferencia, la disminución de las rondas fue compensada con un generador de caos basado en el mapeo de Kent para fortalecer el cifrado.

AES-Estándar, AES-Contador y AES-Caos fueron programados en Python, para posteriormente implementar AES-Contador en un módulo de GNU-Radio.

Los resultados del algoritmo propuesto son comparados con AES-Estándar y AES-Contador empleando pruebas de rendimiento tales como: entropía, correlación y tiempo de ejecución en el proceso de cifrado; también se realizarán pruebas estadísticas utilizando las baterías de prueba del NIST 800-22SP.

Para llegar a esto se siguieron los siguientes pasos:

1. Revisión de la literatura correspondiente.
2. Simular el algoritmo de cifrado AES-Estándar, AES-Contador y AES-Caos en Python.
3. Pruebas de desempeño de AES-Estándar, AES-Contador y AES-Caos Presentación de avances al final del trimestre
4. Implementar el algoritmo AES-Caos en GNU-Radio
5. Primera versión de la idónea comunicación de resultados (ICR)

Estos pasos serán calendarizados de la siguiente manera:

Tabla I. Calendario de actividades

Actividad	Trimestre 18-P			Trimestre 18-O			Trimestre 19-I		
	1	2	3	4	5	6	7	8	9
Revisión de la literatura correspondiente									
Simular en Python el algoritmo de cifrado AES-Estándar, AES-Contador y AES-Caos									
Pruebas de desempeño de AES-Estándar, AES-Contador y AES-Caos									
Implementar el algoritmo AES-Caos en GNU-Radio									
Primera versión de la idónea comunicación de resultados (ICR)									

El resto de este documento se divide en los siguientes capítulos: en el capítulo 2 se presentan algunos aspectos importantes para entender este trabajo de investigación tales como: radios cognitivos, AES-Estándar, AES-Contador y por último caos determinista. En el capítulo 3 se desarrolla el estado del arte respecto a las modificaciones que se le han realizado a AES-Estándar y sobre el uso de AES en los radios cognitivos. En el capítulo 4 se describe de manera detallada el desarrollo de algoritmo criptográfico propuesto. En el capítulo 5 se hace una síntesis de los resultados alcanzados se analizan. El capítulo 6 contiene la conclusión del trabajo desarrollado. En el capítulo 7 se presenta una posible dirección para un trabajo a futuro.

2 Marco teórico

En este capítulo se describen algunos aspectos importantes para este tema de investigación. En primer lugar, se definen los conceptos de seguridad de la información y criptografía, así como algunos aspectos importantes relacionados con la misma, posteriormente se explican algunos ataques contra los radios cognitivos, así como a que capa están destinados, por último, se explica a detalle el Estándar de Cifrado Avanzado y los tipos de cifrado a bloques.

2.1 Radios Cognitivos

La seguridad de la información es el conjunto de medidas preventivas y reactivas de las organizaciones y sistemas tecnológicos que permiten resguardar y proteger la información [22].

La criptografía trata de proteger la información. A continuación, una definición más concreta de la criptografía:

"La criptografía es el estudio de técnicas matemáticas relacionadas con aspectos de seguridad de la información tales como confidencialidad, integridad de datos, autenticación de entidades y autenticación de origen de datos" [23].

Confidencialidad implica mantener la información en secreto para todos excepto para las personas autorizadas en tenerla. La integridad de datos se ocupa de detectar o prevenir la alteración no autorizada de la información. Para garantizar la integridad de los datos, se considera un mecanismo para detección de la manipulación de datos por terceros no autorizados. La autenticación se relaciona con la identificación de las partes que entran en una comunicación, es necesario identificarse entre sí. La información entregada a través de un canal debe ser autenticada en cuanto al origen, fecha de origen, contenido de datos, hora de envío, etc. Por estos motivos, este aspecto de la criptografía suele subdividirse en dos clases principales: autenticación de entidad y autenticación de origen de datos. El no repudio impide que una entidad niegue compromisos o acciones anteriores.

Respecto a la seguridad en CRN, generalmente, los atacantes apuntan a las tres capas más bajas de la comunicación, es decir, física, enlace de datos y red. Debido a la estrategia de acceso dinámico al espectro (Dynamic Spectrum Access, DSA) para CR en comparación con el sistema de comunicación inalámbrico convencional, la capa física de CR tiende a ser la más vulnerable. La Tabla II muestra algunos ataques en CRN.

Tabla II. Principales ataques en CRN [5]

Capa	*Ataque*	*Descripción*
Capa Física	Emulación de Usuario Primario (PUEA)	El atacante se hace pasar por usuario primario
	Aprendizaje	Los CR, cuentan con sistemas de aprendizaje, para aprender de eventos pasados. En este ataque el CR es alimentado con información falsa por lo que comienza a actuar de forma errónea.
	Interferencia	El atacante transmite señales de alta frecuencia en uno o varios canales para inundar la red y congelar la comunicación.
	Eavesdropping	El atacante escucha a la estación base y al UP para obtener información sobre los canales disponibles.
	Ubicación del PU	La ubicación del PU puede detectarse mediante la señal que emite. El atacante puede calcular la ubicación exacta de la PU e iniciar un ataque físico directo que podría deshabilitar el procesamiento.
	Ataque de función objetivo	El atacante intenta cambiar los parámetros del RC. Por ejemplo, frecuencia central, ancho de banda, etc.
Capa de enlace	Interferencia del canal	Este ataque impide que el PU reciba mensajes de control validos lo que da como resultado DoS[1]. Mientras el PU no puede usar los servicios el atacante usa el espectro.
	Negación de servicio (DoS)	Este ataque consiste en generar una cantidad masiva de peticiones al servidor o canales de comunicación de la red, provocando así una sobrecarga de este y, por consiguiente, altera del servicio a los PU.
	Colisión	El atacante envía por el canal paquetes de ruido que causan colisión con otros de la red. Estas colisiones causan la

[1] DoS: Ataque de negación del servicio (Denial of Service, DoS).

		corrupción de los datos.
	Ataque de agotamiento	El atacante envía continuamente solicitudes RTS[2] y recibe mensajes CTS[3] de los vecinos. Hace que el nodo transmita datos continuamente, lo que afecta su procedimiento de interrupción y agota su batería.
	Broadcast no autenticado	El atacante inicia una transmisión no autenticada que sigue todos los protocolos MAC para aumentar el consumo de energía de los nodos y agotar la batería.
	Sumidero	El atacante se hace pasar por la ruta más corta hacia la estación base. De esta forma los otros nodos le usan para reenviar sus paquetes y el nodo malicioso obtendrá buena reputación.
	Ataque Civil	Influye en la red al asumir múltiples identidades incógnitas. Se usa para alterar el proceso de toma de decisiones y puede llevar a restringir que los UP accedan al canal.
	Desbordamiento de mensajes de aviso	Un atacante lejano utiliza alta potencia para anunciar continuamente a los nodos de una red su supuesta cercanía y los hace creer que son vecinos.
	Agujero de gusano	Los mensajes son recibidos por el nodo malicioso desde un lado de la red y luego trasmitidos en un enlace de baja inactividad, después la información es transmitida de nuevo.
	Efecto Ripple	Se utiliza la capacidad de cambiar de canal en medio de la transmisión. Se van cambiando los canales para que la SU consuma algo de potencia y tiempo para las operaciones (como la detección de espectro, etc.), el atacante podría transferir información defectuosa y dejar la red en un estado desordenado.
Multicapa	Medusa	En este ataque se crean respuestas falsas de retraso y pérdida de paquetes durante las comunicaciones.
	León	Cuando el ataque de emulación del usuario primario (PUEA) se dirige a la

[2] RTS: Trama de solicitud de envió de datos (Request to Send, RTS).
[3] CTS: Trama de contestación: preparado para transmitir (Clear to Send, CTS)

		capa física, deriva en el ataque del león y causa la distorsión del TCP. Este TCP distorsionado crea conexiones lógicas y envía paquetes de datos continuamente. Cuando el tiempo de espera del paquete termina, la retransmisión tarda el doble con la desactivación. Este retroceso da como resultado el retraso y la pérdida de paquetes.
	Interferencia de información de enrutamiento	Cuando se intercambia la información de enrutamiento, un nodo interno malicioso apunta a otro nodo del sistema, incitándolo a la transferencia del servicio de una estación base a otra y crea señales para detener el intercambiar paquetes de datos.

En la Figura 4 se puede observar el porcentaje de ataques que realizan a la capa física, a la capa de enlace y multicapa.

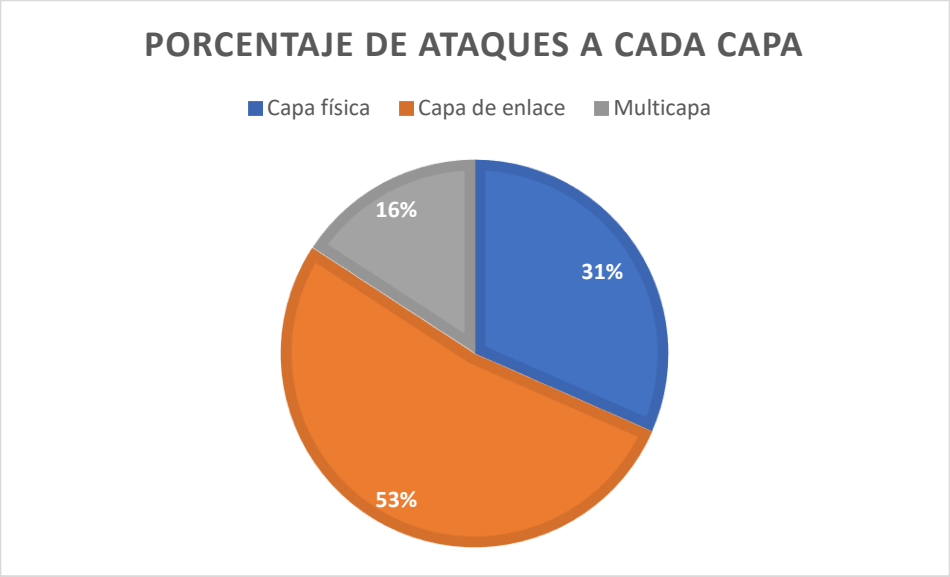

Figura 1. Porcentaje de ataques en cada capa en redes de radios cognitivos

2.2 GNU Radio

GNU Radio es una herramienta de desarrollo libre y abierta que proporciona bloques de procesamiento de señal para implementar sistemas de radio definida por software

(Software Defined Radios, SDR). Puede utilizarse con hardware de RF o sin hardware en un ambiente de simulación.

Las aplicaciones de GNU Radio se construyen mediante un entorno gráfico GNU Radio Companion o mediante lenguaje de programación Python directamente mientras que la parte que requiere alto rendimiento se implementa en lenguaje C++, como es el caso de sus librerías. Así, es posible desarrollar sistemas de radio en tiempo real y alto rendimiento mediante el uso simple y rápido de su entorno de desarrollo de aplicaciones. GNU Radio es un paquete de procesamiento de señales, que se distribuye bajo la licencia GNU GPL (General Public License, GPL). Todo el código tiene los derechos de autor de la Free Software Foundation.

Como todos los sistemas de radio definidos por software, la reconfigurabilidad es una característica clave. En vez de adquirir comercialmente varios tipos de radio, se puede adquirir una simple radio genérica la cual utiliza procesamiento de señal por software [24].

2.3 Estándar de cifrado avanzado

AES es un algoritmo criptográfico aprobado por la Federación de Estándares de Procesamiento de Información (Federal Information Processing Standards, FIPS). El algoritmo AES es un cifrado a bloques capaz de utilizar claves criptográficas de 128, 192 y 256 bits para cifrar y descifrar datos en bloques de 128 bits.

El algoritmo puede implementarse en software, firmware o hardware. La implementación específica puede depender de varios factores, como la aplicación, el entorno, la tecnología utilizada, etc. Este estándar se basa en el algoritmo de Rijndael [25]. La Figura 5 muestra el diagrama a bloques de AES.

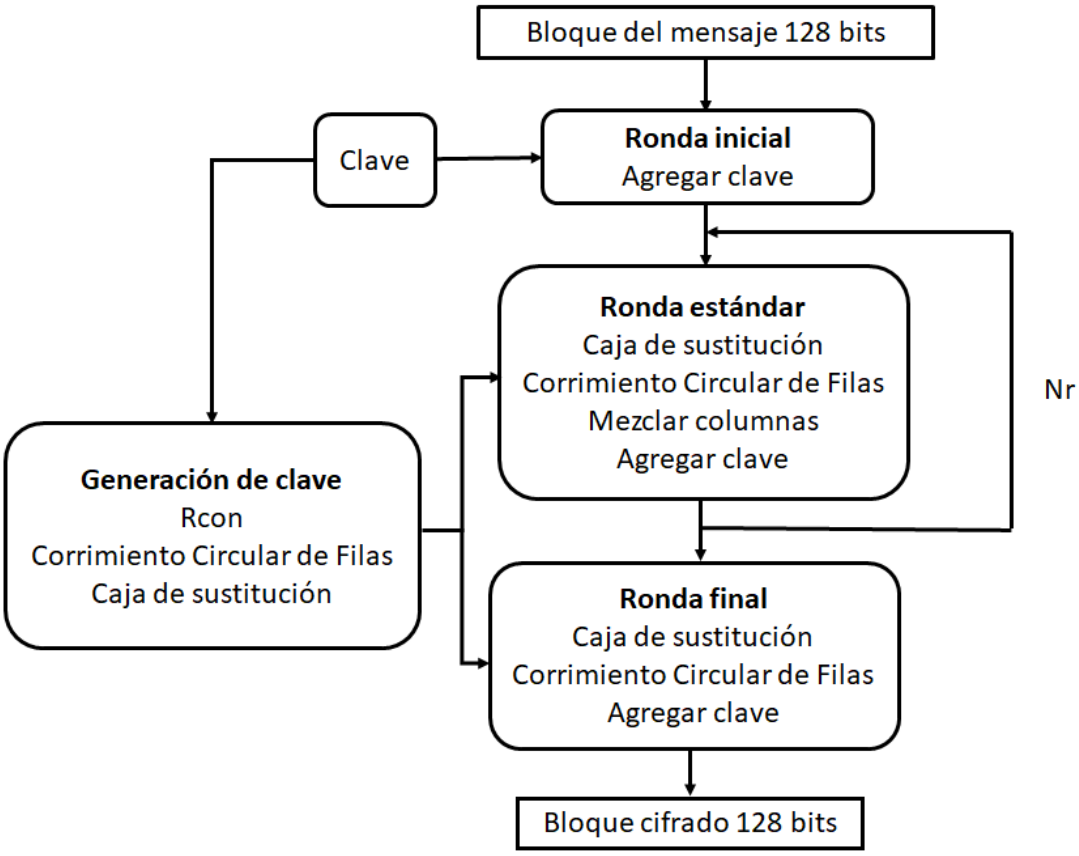

Figura 2. Diagrama a bloques del cifrado AES

Donde Nr es el número que indica las repeticiones de la ronda estándar y cambia dependiendo de la longitud de la clave, ver Tabla III.

Tabla III. Número de rondas según el tamaño de la clave

Tamaño de clave [bits]	Número de Rondas estándar (Nr)	Rondas Totales
128	9	11
192	11	13
256	13	15

La longitud del bloque de entrada, bloque de cifrado y de la clave es de 128bits. Internamente las operaciones de AES se realizan en un arreglo, por lo que se crea una matriz bidimensional de bytes llamada Estado.

La matriz del Estado consta de cuatro filas de bytes, cada una con Nb bytes, donde Nb es la longitud del bloque dividido entre 32.

La matriz de Estado es denotada con el símbolo s. Cada byte tiene dos índices, la fila se representa con r y está en el rango de $0 \leq r < 4$; la columna se representa con c y está dentro del rango $0 \leq c < Nb$. De esta forma, cada byte del Estado se denomina $s_{r,c}$, para AES-Estándar con una clave de 128 bits Nb=4. En la Ecuación 1 se puede observar la matriz de Estado, así como los bytes con sus coordenadas correspondientes.

$$s = \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} \quad (1)$$

Los cuatro bytes de cada columna de la matriz de estado forman una palabra de 32bits, donde el número de fila r proporciona un índice para los cuatro bytes dentro de cada palabra. Por lo tanto, el estado puede interpretarse como un vector de palabras de 32 bits (columnas), $w_0, \dots, w_3$, donde el número de columna c proporciona un índice en esta matriz. El estado se puede considerar como un conjunto de cuatro palabras que se observan en la Ecuación 2.

$$\begin{aligned} w_0 &= s_{0,0}, s_{1,0}, s_{2,0}, s_{3,0} \\ w_1 &= s_{01}, s_{1,1}, s_{2,1}, s_{3,1} \\ w_2 &= s_{0,2}, s_{1,2}, s_{2,2}, s_{3,2} \\ w_3 &= s_{0,3}, s_{1,3}, s_{2,3}, s_{3,3} \end{aligned} \quad (2)$$

AES ocupa 4 transformaciones para cifrar como para descifrar, estas se describen a continuación.

Sustitución de Byte (SubBytes)

Es una transformación no lineal que opera independientemente en cada byte del Estado utilizando una tabla de búsqueda denominada caja de sustitución. Esta tabla es invertible y se construye componiendo dos transformaciones:

1. Se toma el inverso multiplicativo en el campo finito GF (2^8).
2. Se aplica la transformación afín de la Ecuación 3 sobre GF (2^8).

$$b'_i = b_i \oplus b_{(i+4) mod 8} \oplus b_{(i+5) mod 8} \oplus b_{(i+6) mod 8} \oplus b_{(i+7) mod 8 \oplus c_i} \quad (3)$$

Para $0 \leq i < 8$ donde b_i es el bit i-ésimo del byte, c_i es el bit i-ésimo de un byte c con el valor {63} o {01100011} y $\oplus$ representa la operación XOR. La prima en la variable b'_i indica que el valor debe actualizarse con el valor que está a la derecha.

Los elementos de la trasformación lineal se pueden expresar de la siguiente forma (4):

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad (4)$$

La caja de sustitución utilizada en la transformación SubBytes se presenta en forma hexadecimal y se puede observar en la Figura 6. Para llevar a cabo esta transformación, se toma un byte y la parte más significativa representa las coordenadas 'x' y la menos significativa las coordenadas en 'y'. En la Figura 6 se puede observar el funcionamiento de la caja de sustitución.

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Figura 3. Sustitución de bytes, usando la caja de sustitución

Desplazamiento de filas (ShiftRows)

En la transformación ShiftRows, los bytes de las últimas tres filas del estado se desplazan cíclicamente un número diferente de bytes (offsets). La primera fila, $r = 0$, no se desplaza. Específicamente, la transformación ShiftRows procede como en la Ecuación 5.

$$s'_{r,c} = S_{r,(c+desplazamiento(r,Nb))modNb} \quad (5)$$

Donde $0 \leq r < 4$, $0 \leq c < Nb$ y $desplazamiento(r, Nb)$ depende del número de la fila r.

$$\begin{aligned} desplazamiento(1,4) &= 1 \\ desplazamiento(2,4) &= 2 \\ desplazamiento(3,4) &= 3 \end{aligned} \quad (6)$$

Mezclar columnas (MixColumns)

Esta transformación opera en el estado columna por columna, tratando cada columna como un polinomio de cuatro términos en el campo $GF(2^8)$ dado por la Ecuación 7 y cada columna se multiplican por $x^4 + 1$.

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \quad (7)$$

Esto se puede escribir en una matriz de multiplicación como se muestra en la Ecuación 8.

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad (8)$$

Con $0 \leq c < Nb$

Agregar clave (AddRoundKey)

En esta transformación únicamente se realiza una operación XOR bit a bit entre cada bit del estado actual y una clave tal como se muestra en (9). Cada clave consta de Nb palabras.

$$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] = [s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}] \oplus [w_{ronda \times Nb + c}] \; para \; 0 \leq c < Nb \quad (9)$$

Donde $[w_i]$ es la expansión de la palabra de la clave y $ronda$ es el valor dentro del intervalo $0 \leq ronda \leq Nr$. En la Figura 7, podemos observar esto.

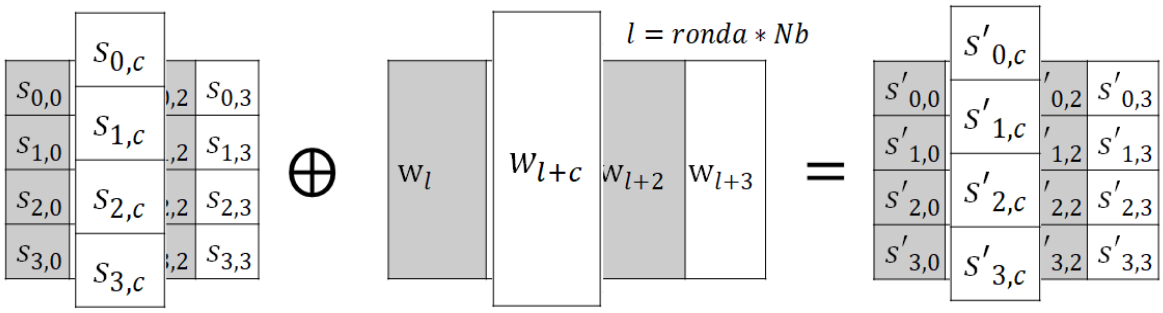

Figura 4. Agregar clave

Expansión de la clave

El algoritmo AES toma la clave de cifrado k y realiza una expansión de ésta. La expansión de la clave genera un total de $Nb(Nr+1)$ palabras. El algoritmo requiere un conjunto inicial de palabras y cada una de las rondas (Nr) requiere Nb palabras de clave. El algoritmo resultante consiste en una matriz lineal de palabras de 4 bytes, denotadas $[w_i]$, donde $0 \leq i < Nb(Nr+1)$.

Dentro de la rutina de expansión de la clave se usan las siguientes funciones:
Sustitución de Byte: Transformación que procesa el estado utilizando una tabla de sustitución de bytes no lineal, que opera en cada uno de los bytes de estado de forma independiente.

Rotar palabra: Se utiliza en la expansión de clave, toma una palabra de cuatro bytes (por ejemplo $[a_0, a_1, a_2, a_3]$) y se realiza una permutación cíclica (regresa $[a_1, a_2, a_3, a_0]$).

Rcon: Es un vector, se define como $rcon = [x^{i-1}, \{00\}, \{00\}, \{00\}]$ donde x^i es la $i-$ésima potencia y x es un elemento en el campo GF (2^8). Este vector cambia dependiendo de la ronda.
Donde x^{i-1} es un valor de 8 bits definido como muestra la Ecuación 10

$$x^{i-1} = \begin{cases} 1 & si \quad i = 1 \\ 2 \cdot x^{i-1} - 1 & si \quad i > 1 \; y \; x^{i-1} < 80_{16} \\ (2 \cdot x^{i-1} - 1) \oplus 1B_{16} & si \quad i > 1 \; y \; x^{i-1} > 80_{16} \end{cases} \tag{10}$$

Donde $1B_{16}$ el número 27 en hexadecimal y 80_{16} el número 128 en hexadecimal. Obteniendo como resultado la Tabla IV.

Tabla IV. Valores de x^{i-1} en hexadecimal

i	1	2	3	4	5	6	7	8	9	10
x^{i-1}	01	02	04	08	10	20	40	80	1B	36

2.3.1 Modos de operación de cifrados a bloques.

Es importante describir la forma de operación de los cifrados a bloques ya que AES es un cifrado a bloques y en este trabajo se basa en uno de estos modos de operación. Para este tipo de cifrados el Instituto Nacional de Estándares y Tecnología (National Institute of Standards and Technology, NIST) recomienda cinco modos de operación. Estos modos varían en la forma de interacción de los bloques durante el proceso de cifrado. Los modos de operación son: Libro Electrónico de Códigos (Electronic Codebook, ECB), Encadenamiento de Bloque de Cifrado (Cipher Block Chaining, CBC), Realimentación de Cifrado (Cipher Feedback, CFB), Realimentación de salida (Output Feedback, ECB) y Contador (Counter, CTR). A continuación, se describen los modos de operación.

1) El modo Libro Electrónico de Códigos es el modo de operación más sencillo. En este modo de operación los mensajes se dividen en bloques y cada uno de ellos es cifrado por separado utilizando la misma clave. Para una clave dada, cualquier bloque de texto siempre se cifra en el mismo bloque. El cifrado de un bloque es análogo a la asignación de palabras en un libro de códigos. Lo que lo convierte en un método fácil de romper con criptoanálisis estadístico. La Figura 8 muestra el diagrama a bloques de este modo de operación.

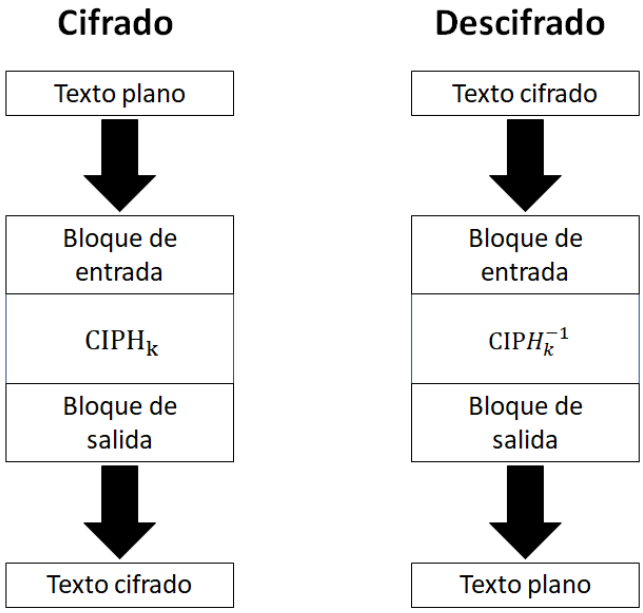

Figura 5. Cifrado en Modo Libro Electrónico de Códigos

2) El modo Encadenamiento de Bloque de Cifrado, es un método cuyo proceso de cifrado presenta la combinación ("encadenamiento") de los bloques de texto en claro con los bloques de texto cifrado. Antes de ser cifrado a cada bloque de texto plano se le aplica la operación XOR con el bloque anterior cifrado. De esta forma, cada bloque de texto cifrado depende de todo el texto en claro que se ha procesado hasta el momento. Para hacer cada mensaje único se utiliza un vector de inicialización. La Figura 9 muestra el diagrama a bloques de este modo de operación.

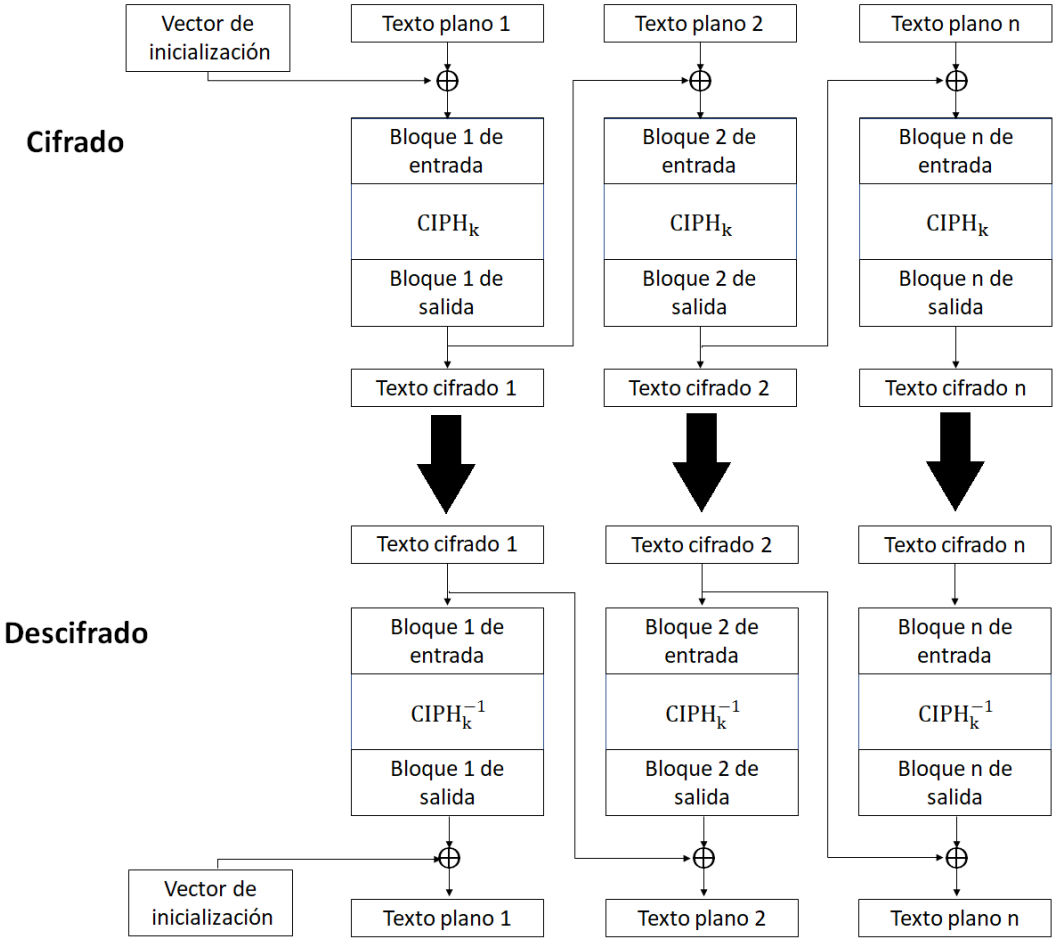

Figura 6. Cifrado en Modo Encadenamiento de Bloque de Cifrado

3) En el modo Realimentación de Cifrado, el texto cifrado resulta de una XOR del mensaje en claro y la realimentación de segmentos sucesivos del texto cifrado en los bloques de entrada. Se requiere un vector de inicialización como el bloque de entrada inicial, no necesita ser secreto. La Figura 10 muestra el diagrama a bloques de este modo de operación de cifrado.

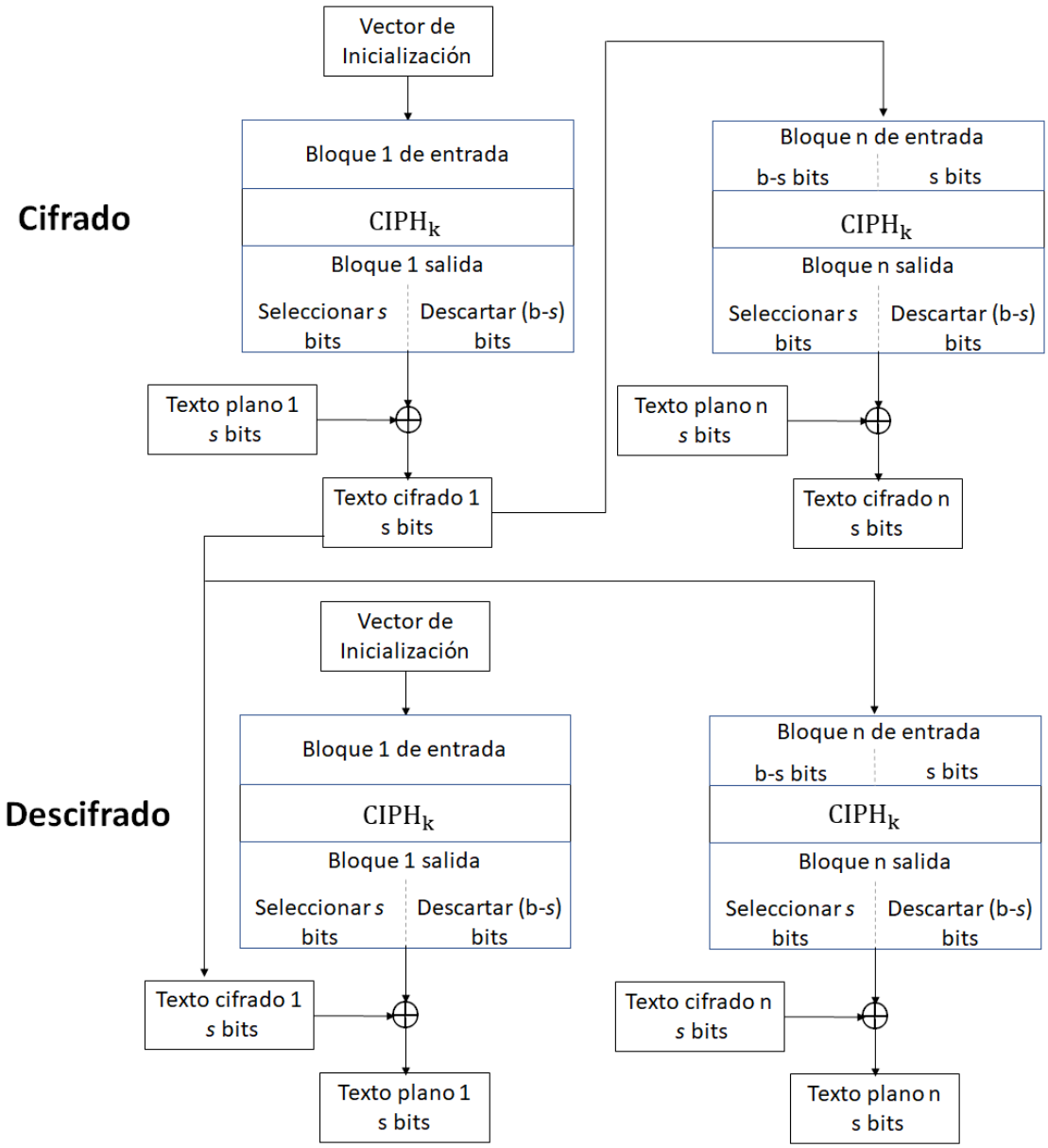

Figura 7. Cifrado en Modo Realimentación de Cifrado

4) El modo Realimentación de salida, presenta la iteración del cifrado directo con un vector de inicialización para generar una secuencia de bloques de salida que son XOR con el texto en claro para producir el texto cifrado y viceversa. El modo OFB requiere que el vector de inicialización sea único para cada ejecución del modo bajo la clave dada. La Figura 11, muestra el diagrama a bloques de este modo de operación para el cifrado.

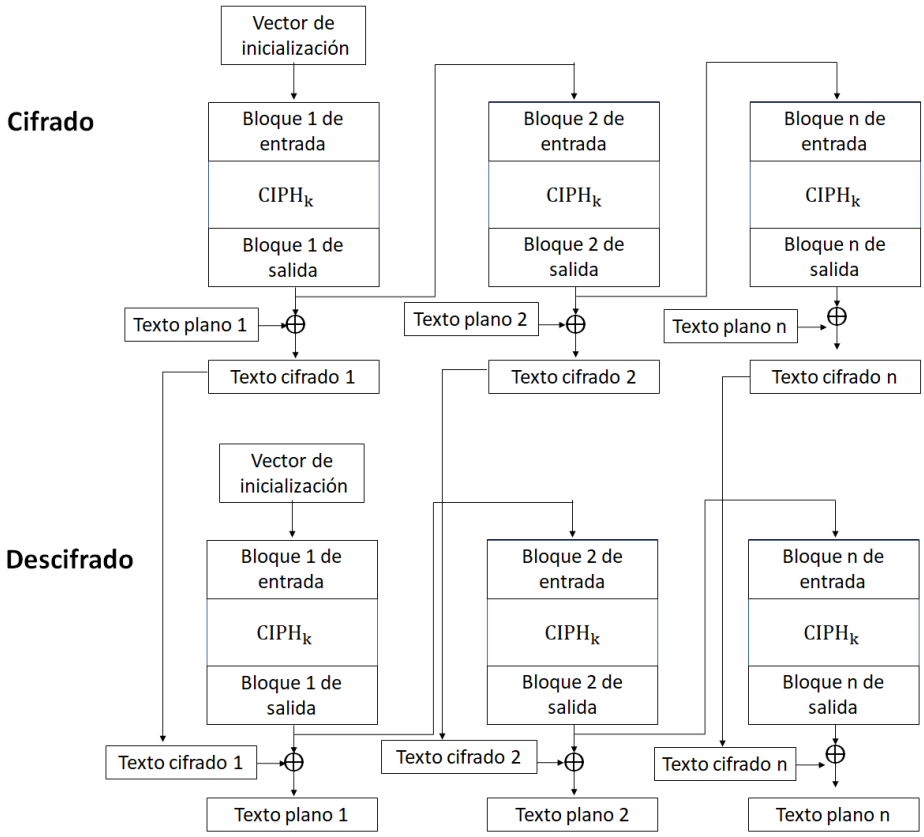

Figura 8. Modo realimentación de salida

5) Modo contador, en este modo de operación se cifra un conjunto de bloques de entrada llamados contadores, y se produce una secuencia de bloques de salida. A la secuencia de salida se le aplica una operación XOR con el texto plano para producir el texto el cifrado. La secuencia de contadores debe tener la propiedad de que cada bloque en la secuencia es diferente de cualquier otro bloque. Esta condición no está restringida a un solo mensaje: en todos los mensajes cifrados con la clave especificada, todos los contadores deben ser distintos [26]. La Figura 12 muestra el diagrama a bloques de este modo de operación para el cifrado.

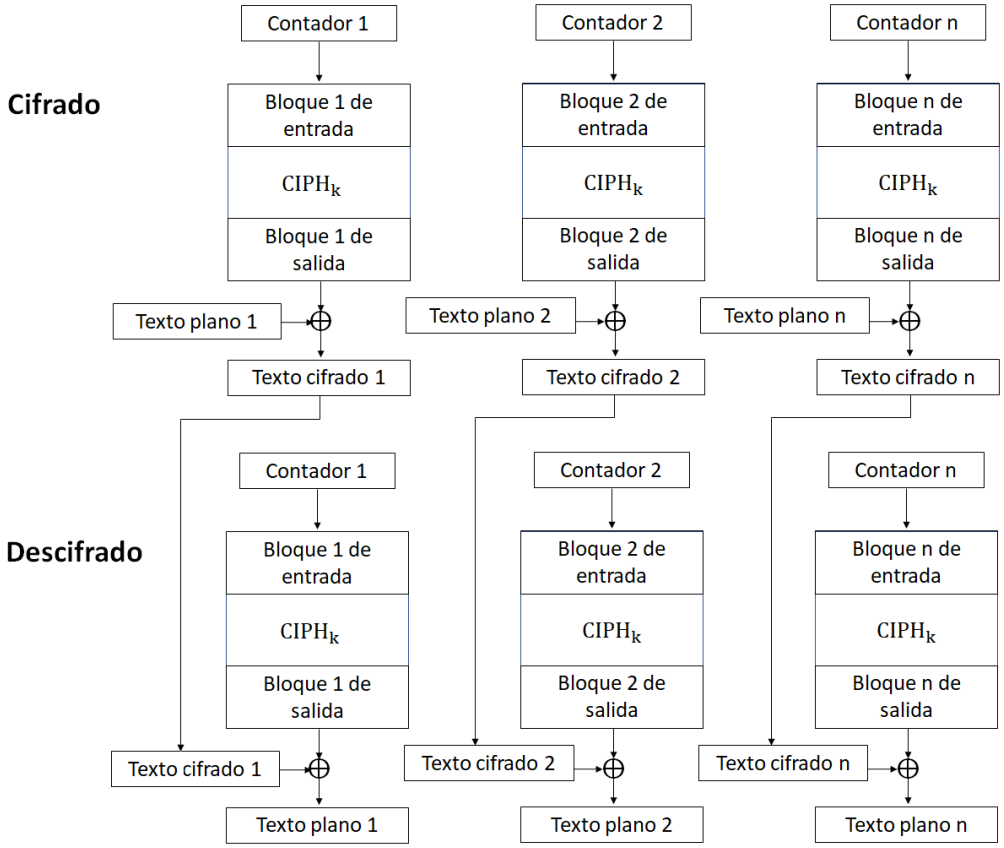

Figura 9. Modo Contador

2.4 AES en modo contador

Como anteriormente se había mencionado AES es un cifrado de bloques recomendado por el NIST. Cuando se utiliza en el modo de contador se cifra un bloque de contador y como resultado, se genera un XOR con un bloque de texto sin formato que creará el texto de cifrado.

La longitud del contador es de 128 bits y está conformado por dos partes, la primera se le llama “nonce” que son 64 bits aleatorios y la segunda CTR que son los 64 bits restantes y comienza en 0. El contador incrementa en uno después del cifrado de cada bloque.

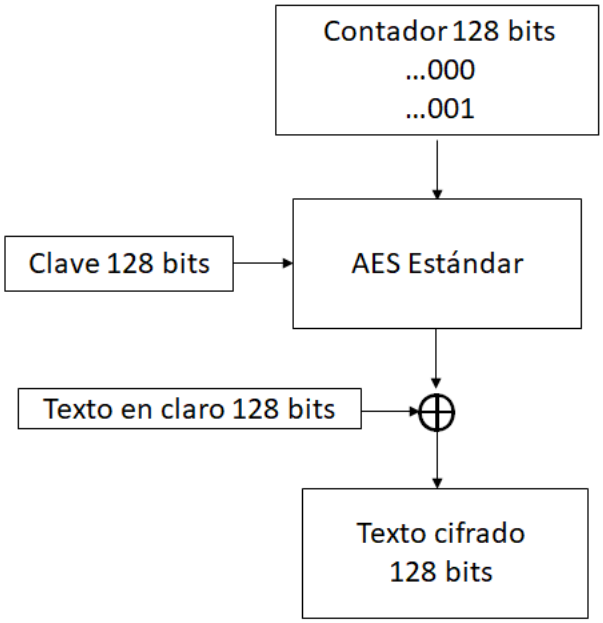

Figura 10. AES en modo contador

El modo de contador resulta útil ya que el cifrado o descifrado de un bloque no depende del cifrado o descifrado anterior de otro bloque.

El modo CTR tiene ventajas de eficiencia significativas sobre los modos de cifrado estándar sin debilitar la seguridad [27].

Además, se puede observar que el modo de operación funciona como un cifrado de flujo, donde el contador se cifra para generar una secuencia cifrante.

2.5 Caos determinista

Según la Real Academia de la lengua española, la palabra "caos" se deriva del griego "χαοσ" y originalmente significaba el espacio infinito vacío que existía antes de todas las cosas [28]. En el uso que adoptaremos aquí, el caos denota un estado de desorden e irregularidad. Por otro lado, el caos determinista denota el movimiento irregular o caótico generado por sistemas no lineales cuyas leyes dinámicas determinan de manera única la evolución temporal de un estado del sistema a partir de un conocimiento de su historia anterior [29]. En otras palabras, los sistemas caóticos deterministas evolucionan en el tiempo por el proceso de iteraciones, donde el siguiente estado del sistema es determinado por su estado actual. En general esto se expresa como $x_{n+w1} = f(x_n)$.

2.6 Mapa de Kent o mapa Skew Tent

El mapeo de Kent es un sistema caótico con buen rendimiento por lo que tiene alta sensibilidad a las condiciones iniciales, esto quiere decir que cuando los valores iniciales cambian ligeramente, se generarán dos secuencias pseudoaleatorias completamente diferentes. En la Ecuación 11, se muestra la transformación caótica de Skew Tent en su versión iterada [30].

$$f_n(x_0, \mu_0) = \begin{cases} \dfrac{x_{n-1}}{\mu} & si \quad 0 < x_{n-1} \leq \mu \\ \dfrac{1 - x_{n-1}}{1 - \mu} & si \quad \mu \leq x_{n-1} < 1 \end{cases} \tag{11}$$

Donde $n = 1, 2, 3, ...$ representa el paso de la iteración, $x_0 \ni (0,1)$ es la condición inicial, x_n es el número real producido por la transformación caótica en iteración de n, $\mu \ni (0,1)$ es el parámetro de control, $f_n(x_0, \mu_0)$ representa la iteración de la transformación caótica aplicada n veces en x_0 usando μ [30].

2.6.1 Diagrama de bifurcación

El diagrama de bifurcación es la forma más eficiente de dar una visión de conjunto de los diferentes comportamientos asintóticos según el valor de x.
El diagrama de bifurcación nos dice los puntos x_n que son visitados en un intervalo unidad al iterar un gran número de veces y tras descartar un transitorio (una cantidad de iteraciones iniciales).

En consecuencia, la distribución estadística de una secuencia producida por la transformación caótica Kent se puede determinar encontrando la frecuencia con la que se visitan las diferentes regiones en (0, 1), este proceso se puede apreciar en su diagrama de bifurcación que se muestra en la Figura 14.

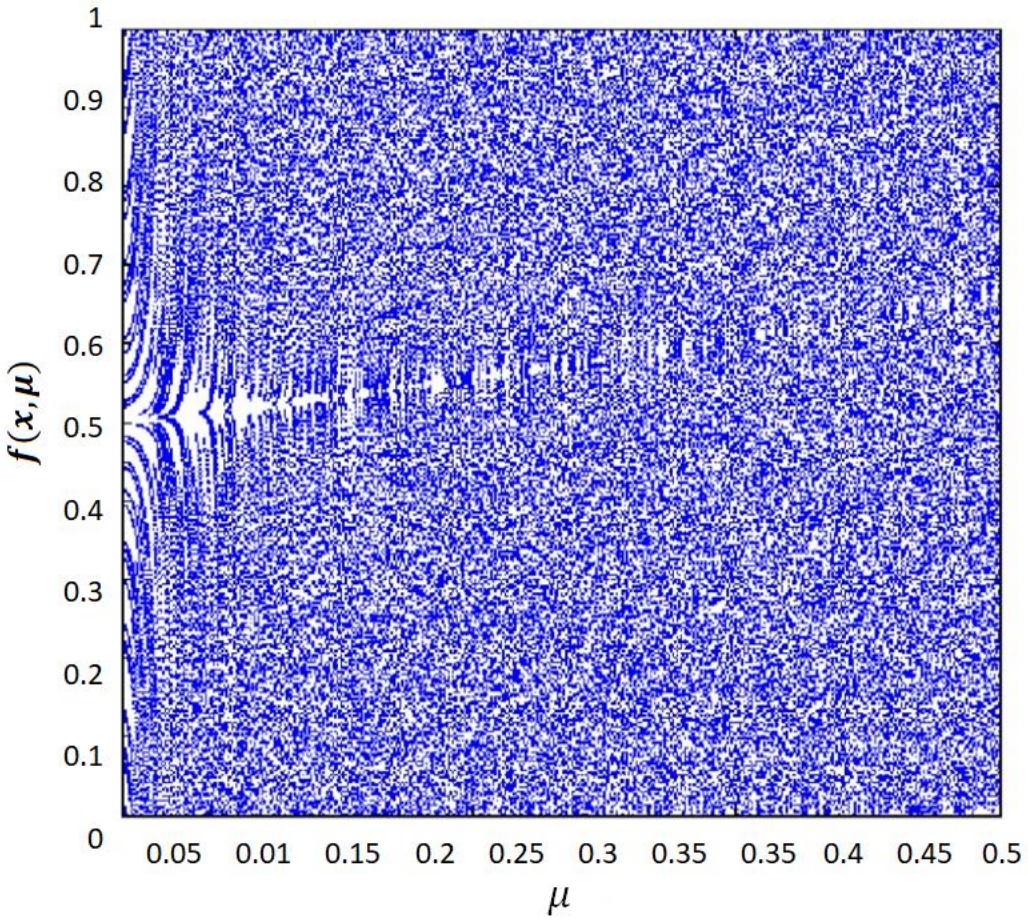

Figura 11. Distribución de $f(x,\mu)$ vs μ con 5000 iteraciones

3 Estado del arte

En este capítulo se muestran una serie de modificaciones que se han realizado al Estándar de Cifrado Avanzado (AES) ya sea para mejorar el tiempo de cifrado, mejorar la seguridad o para que sea posible aplicarlo a imágenes. Algunas de las modificaciones realizadas implican cambiar operaciones de AES-Estándar, por ejemplo, la caja de sustitución; algunas más aumentan el tamaño del bloque o el tamaño de la clave. Cabe mencionar que las siguientes propuestas son comparadas con AES-Estándar utilizando un tamaño de clave de 128 bits. Estos trabajos nos permiten identificar las mejoras y cambios que se le han hecho a AES, así como identificar cuáles son las operaciones que hacen el proceso de cifrado más lento.

En la sección 2.2 se exponen algunas adaptaciones AES para radios cognitivos. El estándar de cifrado AES es utilizado por [8] [9] para defensa de redes de radios cognitivos contra el ataque de emulación de usuario primario (Primary Users Emulation Attack, PUEA), en ambos casos se proponen para la banda de TV digital, para generar una señal de referencia segura. En [10] el algoritmo de cifrado AES es usado para proteger información en redes caóticas de CR, se transmite una señal sobre múltiples subportadoras. Seha-Birendra et al. proponen una modificación en la expansión de la clave con un desplazamiento de filas y una disminución de las rondas para disminuir el tiempo de cifrado [21].

3.1 Comparación de diferentes algoritmos modificados del estándar de cifrado avanzado

El algoritmo AES-Estándar proporciona buena seguridad, mejor velocidad de cifrado y rendimiento en comparación con otros cifrados simétricos [6]. Actualmente se están realizando modificaciones para reducir los recursos de hardware, aumentar la seguridad contra los ataques estadísticos, disminuir el tiempo de cifrado [11]. Sin embargo, las modificaciones mostradas no mejoran significativamente el tiempo y las que mejoran en cuanto a seguridad lo hacen a costa del tiempo. Así, el trabajo de tesis considera un mecanismo para aumentar la velocidad sin perder la seguridad del cifrado.

La Tabla V, muestra las diferentes modificaciones de AES encontrados en la literatura, donde se compara la longitud de la clave, mejoras en el tiempo de cifrado y descifrado, mejoras en la seguridad, mejoras en el cifrado de imágenes e indican si se hizo alguna modificación en la manera en que se generan las claves.

Tabla V. Análisis comparativo de diferentes algoritmos AES modificados encontrados en la literatura contra AES-Estándar con clave de 128 bits

Modificación	Longitud de clave	Tiempo de cifrado	Tiempo de descifrado	Seguridad	Imágenes
AES-A5/1 y AES-W7 [12]	128	Disminuye	No se indica	Mejora	Si
AES caja gris [13]	128	Aumenta	Aumenta	Mejora	No
AES mezcla de filas [14]	128	No cambia	No cambia	Mejora	Si
AES-200 [15]	200	Disminuye	Disminuye	Mejora	No
AES-DES [16]	128	Disminuye	Disminuye	Mejora	No
AES-Caos [17]	128	Disminuye	Disminuye	Mejora	Si
AES-512 [18]	512	Aumenta	Aumenta	Mejora	No
AES columna y s box [19]	128	Disminuye	Disminuye	Buena	No
AES con JEX [20]	128	Aumenta	Aumenta	Mejora	Si

En la Figura 1 se puede observar que el 56% de las modificaciones del cifrado AES no presentan mejoras con respecto al cifrado de imágenes y el 44% hace mejoras para el cifrado de imágenes con AES.

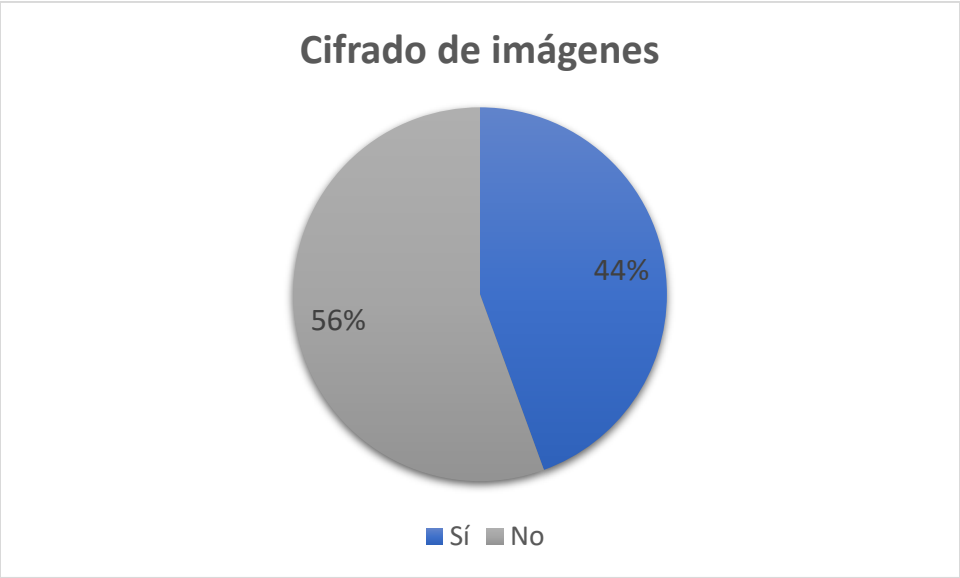

Figura 12. Porcentaje de algoritmos que implementan mejoras para cifrado AES para imágenes

A continuación, se describen los algoritmos desarrollados para el cifrado de imágenes usando una modificación de AES-Estándar con una clave 128 bits.

AES-A5/1 y AES-W7 es una modificación de AES propuesto por Zeghid et al. [12]. Considera un esquema de cifrado que combina la generación de claves de AES con un generador de flujo de clave usando A5/1 o W7. Donde A5/1 es un algoritmo de cifrado de flujo que se basa en una combinación de 3 registros de desplazamiento con realimentación (Linear Feedback Shift Register, LFSR) y el desplazamiento se realiza con una función mayoría que indica que LFSR se desplazará. Con respecto a W7 es un cifrado de flujo síncrono de un byte para una implementación a velocidades de datos altas. Con este algoritmo aumenta la seguridad, en particular se refiere a la confidencialidad y autenticación de una imagen cifrada y el tiempo de cifrado disminuye. Este algoritmo está pensado principalmente para imágenes caracterizadas por entropía[4] reducida. Para evaluar la eficiencia del algoritmo propuesto cifraron 5 imágenes en escala de grises y evalúan cuantitativamente la correlación y el tiempo de cifrado entre la imagen cifrada con los 3 algoritmos antes mencionados. La correlación vertical de AES-128 es 0.066, vertical de AES-A5/1 es de 0.077 y la de AES+w7 es de 0.03, obteniendo una menor correlación de AES+w7 [12]. Este algoritmo cumple con las características que buscamos en el algoritmo propuesto que son mejorar en el tiempo de cifrado, que sea aplicable a imágenes y mantener la confidencialidad de la información a cifrar.

En el algoritmo AES mezcla de filas, fue presentado por Abdulkarim Amer et al. [14]. Proponen una modificación en la fase de rotar filas, donde se examina si el valor de la primera fila y columna es par o impar (estado [0][0]). Si es impar, la primera y la tercera fila no cambian, pero la segunda fila se desplaza una posición a la izquierda y la cuarta fila

[4] La entropía es una medida estadística de la aleatoriedad de la información contenida en un mensaje.

se desplaza tres posiciones hacia la izquierda. Si es par, la primera y la cuarta filas no se modifican y cada byte de la segunda fila se desplaza tres posiciones a la derecha y la tercera fila se desplaza dos posiciones hacia la derecha. Para demostrar la eficiencia de su algoritmo compararon la correlación entre los pixeles de una imagen sin cifrar y una imagen cifrada con MAES y obtuvieron que la correlación horizontal de la imagen sin cifrar 0.9452 y de la imagen cifrada: 0.0112, la vertical 0.9471, AES mejorado: 0.0813. También obtuvieron la entropía de una imagen cifrada por AES 7.9989 y AES mejorado 7.9992 [14]. Aunque este algoritmo presenta una mejora significativa al cifrar imágenes, el tiempo de cifrado respecto AES-128 no cambia.

El algoritmo propuesto AES-DES fue desarrollado por Pravin Kawle et al. [16] consideran una modificación al proceso de cifrado y descifrado que se asemeja al de AES-Estándar, en función del número de rondas, datos y tamaño de clave. Omiten el paso de mezclar columnas esto debido a que este paso necesita muchos cálculos que hacen que el algoritmo de cifrado sea lento y agregan el paso de permutación de DES. Lo interesante de esta propuesta es la velocidad que se obtiene en los procesos de cifrado y descifrado con AES-estándar se tiene un tiempo de cifrado de aproximadamente 1.925 s. mientras que el algoritmo propuesto con DES tarda 1.874 s. Esto implica que el algoritmo AES con permutación DES sólo mejora en tiempo y no fue diseñado para el cifrado de imágenes.

AES-Caótico, es propuesto por Ahmed M. Atteya et al. [8], plantean un algoritmo de cifrado híbrido que resulta al combinar AES y una transformación Caótica. El algoritmo propuesto utiliza una clave de 128 bits para cifrar la cantidad de datos que el usuario necesita, beneficiándose del comportamiento pseudoaleatorio de las secuencias caóticas, reemplazando la S-box y agregando la etapa de clave circular mediante la operación XOR con la transformación caótica de Henon en 2-D. AES-Caótico utiliza una clave de 128 bits para cifrar los datos, por lo que utiliza como condición inicial cuatro transformaciones caóticas de Henon de 32 bits. Posteriormente se hace una XOR con los cuatro valores generados en el mapa de Henon de 32 bits y los datos de entrada de 128 bits, se realizan las operaciones rotar filas y mezclar columnas convencionales de AES con el resultado de la XOR. El algoritmo propuesto requiere sólo una ronda, obteniendo la correlación de una imagen cifrada de 0.0067 y en cada ciclo de reloj, se pueden cifrar 128 bits, lo que equivale a un rendimiento de 11.577 Gbps [17]. De acuerdo con el análisis realizado, en este trabajo se mejora el tiempo de cifrado comparado con AES-128 además de que este cifrado puede ser utilizado para el cifrado de imágenes.

T. S. Sneha-Birendra et al. [20] proponen un algorito criptografica AES-JEX, La técnica de cifrado utiliza codificación JEX con una modificación de AES. La codificación JEX es una mejora del algoritmo de compresión JPEG. La imagen se codifica con JEX y se genera un archivo de encabezado que luego se agrega al archivo de imagen, la imagen resultante pasa por compresión JPEG básica, para luego ser cifrada utilizando el algoritmo AES propuesto. Observe que al usar codificación JEX, cambia la entropía de la imagen con lo que se obtienen mejores resultados al cifrarla con AES. Con respecto a la modificación de AES, ésta se efectúa en el procedimiento de generación de claves de ronda, en el que se

realizan operaciones circulares de cambio e inverso en la matriz de clave de cifrado, utilizando una clave de 128 bits. El tiempo de cifrado fue evaluado comparando el algoritmo propuesto contra AES-Estándar con un clave de 128 bits, tanto en el cifrado como en el descifrado y utilizando diferentes tamaños de archivo o imagen. Para un archivo de 576kb el tiempo de cifrado y descifrado con AES convencional fue de 31ms, mientras el tiempo de cifrado con AES modificado fue de 33ms y el tiempo de descifrado fue de 39ms. La entropía al usar compresión JPEG es de 54.95 mientras que si usan Codificación JEX es de 49.92. Aunque este algoritmo permite el cifrado de imágenes y cambia la forma en que se generan las claves no mejora en tiempo de cifrado.

En contraste la Figura 2 muestra que sólo el 18% modificaron el proceso para generan las claves criptográficas en el algoritmo AES-Estándar con una clave de128 bits, el 82% de las propuestas mantuvo la generación de claves de AES-Estándar.

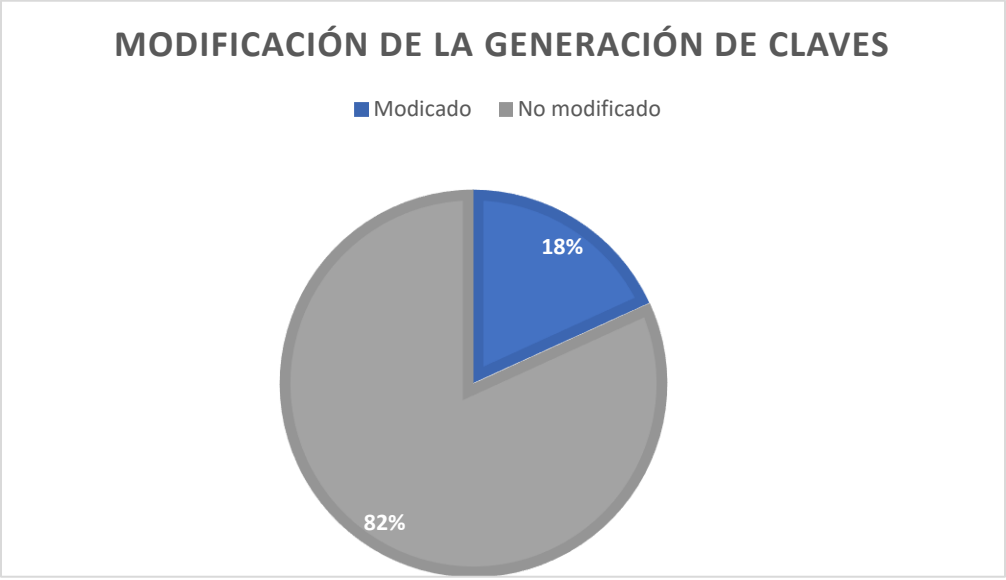

Figura 13. Porcentaje de artículos que presentan modificaciones en la generación de claves

En la propuesta presentada por Zeghid et al. [12] desarrollaron un esquema que utiliza como generador de claves criptográficas los algoritmos de A5/1 y W7. Sneha-Birendra et al. [20], proponen una modificación en la clave de cifrado, en el que se realizan operaciones de desplazamiento circular y un proceso reflexión. Estos dos procesos se realizan por cada ronda.

Con respecto a la Figura 3 se puede apreciar que el 67% de las modificaciones del algoritmo criptográfico AES, se ocupan de aumentar la velocidad del cifrado. El 22% restante, únicamente presentan una mejora en la seguridad de AES-128 sin importar la velocidad o el tiempo que tarda en cifrar la información.

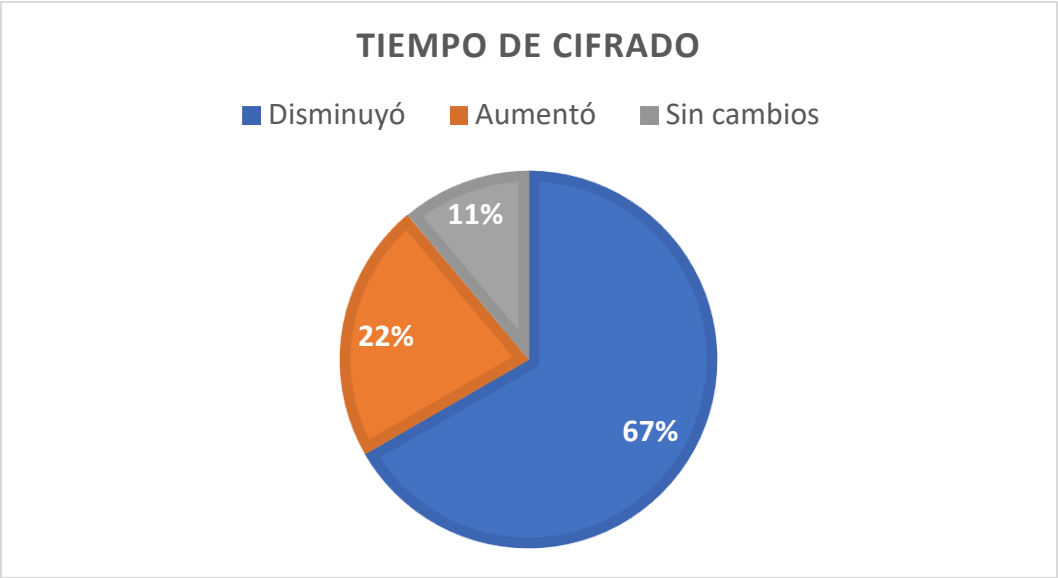

Figura 14. Porcentaje de artículos que presentan mejoras en tiempo de cifrado respecto al tiempo de cifrado de AES-Estándar

De acuerdo con esto los algoritmos que mejoran el tiempo de cifrado y descifrado se consideran las siguientes:

El algoritmo AES-200 fue presentado por Ritu y Vikas [15], quienes propusieron un AES que cuenta con bloques y clave de 200 bits, con una matriz 5 filas por 5 columnas a diferencia de AES-Estándar de 128 bits que utiliza una matriz de 4 filas por 4 columnas, el número de rondas permanece sin cambios. El proceso de la generación de la caja de expansión y sustitución de clave se realiza de la misma manera que en el cifrado que AES-Estándar, el proceso de transformación de mezclado de columnas es modificado para incrementar la seguridad del algoritmo. En consecuencia, los resultados muestran que la velocidad de cifrado y el rendimiento en el extremo de cifrado aumentan y la velocidad de descifrado, el rendimiento en el extremo de descifrado se reduce con respecto al algoritmo AES convencional. Después de la adición de la clave de la ronda inicial, se realizan diez rondas de cifrado. Las primeras nueve rondas son iguales, con una pequeña diferencia en la ronda final. Cada una de las primeras nueve rondas consiste en 4 transformaciones: sustitución de bytes, rotar filas, mezclar columnas y agregar clave. Pero en la ronda final, la transformación de mezclar columnas no se usa. Al evaluar el algoritmo observaron que el tiempo de cifrado AES-200 por bit se reduce hasta un 20% y el tiempo de descifrado por bit se incrementa hasta un 25%, también evaluaron el rendimiento y concluyen que el rendimiento de cifrado de AES-200 es un 15% mayor que AES-128 [15]. A pesar de las mejoras en el tiempo de cifrado y de rendimiento, este algoritmo no está diseñado para cifrar imágenes.

Rizky et al. [19], proponen una modificación al algoritmo criptográfico AES que combina “mezclar de columnas” y “caja de sustitución” (Substitution Box, S-Box). Esta propuesta implementa dos modificaciones: la primera es en la etapa de rotar filas y la segunda es en mezcla de columnas, así como caja de sustitución. En la fase de rotar filas se hace un mapeo directo de los bytes a la posición donde quedarían al rotar la columna, en

vez de rotar las columnas, de esta forma se ahorra el tiempo de la rotación. La segunda modificación se hace una combinación de las operaciones mezclar columnas y caja de sustitución, para esto se hace una multiplicación de la caja de sustitución por las constantes 2 y 3. Este proceso mejora la combinación de las etapas de cifrado de AES. Sin embargo, requiere de una mayor cantidad de memoria. Las evaluaciones se basan en medir el tiempo que tarda el algoritmo en cifrar y descifrar información comparándolo con AES-Estándar de 128 bits. Esta mejora aumenta la velocidad de cifrado y el tiempo se reduce tres milisegundos con porcentaje de optimización 86.1% promedio en cifrado. Mientras tanto, el proceso de descifrado tiene una mejora menos significativa. El tiempo de descifrado disminuye en dos milisegundos con un porcentaje de optimización del 13.0% en promedio. Como podemos observar en la descripción del algoritmo, sólo se mejora el tiempo de cifrado, debido a que combina dos pasos en uno mismo. También se observa que no adiciona algún mecanismo para mejorar el cifrado de imágenes.

Asimismo, hay algoritmos que mejoran la seguridad a costa del tiempo. Entre estos tenemos a [13], [14], [18] y [20].

AES caja gris es reportado por Minh-Triet Tran et al. [13], presentan un proceso de sustitución desarrollando una caja gris. Esta caja gris se construye agregando una transformación de código Gray binario como un paso de preprocesamiento a la caja de sustitución original de AES. El S-box gris corresponde a un polinomio con todos los 255 términos distintos de cero en comparación con el polinomio de 9 términos de la caja de sustitución original de AES. Esto aumenta la seguridad. La caja gris también logra importantes propiedades criptográficas de la caja de sustitución de AES, que incluyen el criterio estricto de avalancha[5], la no linealidad y la uniformidad diferencial [13]. Los autores no reportan mejoras respecto al cifrado de imágenes, ni respecto al tiempo de cifrado, ellos afirman que el número de términos en la expresión algebraica de cada caja de sustitución se usa como una métrica para evaluar la complejidad algebraica y al mejorar la complejidad algebraica mejora la complejidad criptográfica, teniendo una complejidad algebraica de 255 términos y AES-Estándar tiene una complejidad algebraica de 9, provocando un aumento en el tiempo de cifrado.

El algoritmo AES-512 fue desarrollado por Dandekar et al. [18], quienes propusieron un algoritmo modificando la longitud de la clave. Dicho algoritmo utiliza una longitud de clave de 512 bits debido a esto se aumenta el número de rondas. El algoritmo propuesto tiene cuatro principales transformaciones basadas en bytes. La primera transformación, es la sustitución de bytes de los 512 bits mediante s-boxes paralelas. La segunda transformación consiste en combinar las filas haciendo un desplazamiento igual al número de fila. La tercera transformación es la mezclar columnas donde cada columna se multiplica por un valor diferente. La transformación final de la ronda agrega una clave circular al resultado de esta ronda. Los resultados evalúan el tiempo que tarda AES-

[5] El criterio estricto de avalancha mide la cantidad de no linealidad de las cajas de sustitución.

Estándar de 128 bits y AES-512 en cifrar información, los resultados son: AES-128 considera un intervalo de 20s a 40s y AES-512 tiene un rango 30s a 50s.

De los algoritmos revisados en la literatura, únicamente [2], [7] y [10] consideran algunas de las características deseadas para ser implementadas en dispositivos de radio definido por software (Software Defined Radio, SDR) para aplicaciones de redes con capacidades cognitivas. Estas características son: la posibilidad de cifrar imágenes a color y mejora en el tiempo de cifrado.

Cabe mencionar que ninguno de estos algoritmos es utilizado para CRN. A continuación, se muestra la literatura respecto al uso del algoritmo de cifrado AES en CRN.

3.2 AES en Radios cognitivos

El estándar de cifrado AES ha sido utilizado por [8] y [9] como alternativa de protección para redes de radios cognitivos contra el ataque de emulación de usuario primario (Primary Users Emulation Attack, PUEA) en ambos casos se proponen para la banda de TV digital.

En [8] proponen un esquema de televisión digital (Digital Television, DTV) protegido con AES, en el que se genera una señal de referencia cifrada con AES-Estándar en la TV transmisora y la utilizan como los bits de sincronización la trama de datos DTV. El transmisor DTV obtiene la señal de referencia a través de dos pasos: primero, generar una secuencia pseudoaleatoria (Pseudorandom Number, PN,) usando un registro de desplazamiento con retroalimentación lineal (Linear Feedback Shift Register, LFSR) con un vector de inicialización seguro y luego lo cifra con AES- Estándar usando una clave de 256 bits. La salida de AES se usa como la señal de referencia, el transmisor coloca la señal de referencia en los bits de sincronización de los segmentos de datos de DTV. El receptor regenera la señal de referencia cifrada, con la clave que conocen el transmisor y el receptor. Se emplea un detector de correlación, para la detección del usuario primario, el receptor evalúa la correlación cruzada entre la señal recibida y la señal de referencia regenerada; para la detección de usuarios mal intencionados, el receptor evalúa aún más la autocorrelación de la señal recibida.

En [9] también consideran el ataque PUEA en la banda DTV, detección de espectro seguro, acceso dinámico al espectro, Banda Lateral Vestigial de ocho niveles (Vestigial Side Band, 8-VSB). En el sistema propuesto, el usuario primario genera una señal de referencia pseudoaleatoria cifrada con AES-Estándar usando una clave 256 bits que se utiliza como los bytes de sincronización de 624 segmentos. En el extremo de recepción, la señal de referencia se regenera para la detección del usuario primario. Se garantiza la sincronización de los bits de referencia que también se usan con este propósito.

Se muestra en [10] que el algoritmo de cifrado AES-Estándar es usado para proteger redes de radio cognitivo caótico. Más específicamente, se transmite la señal sobre múltiples subportadoras disponibles con un generador de pulsos no retorno a cero que forma un flujo de bits de entrada binaria y luego esto es cifrado con AES-128. La información cifrada es modulada por secuencias de caos donde la secuencia caótica se

utiliza debido a sus numerosas ventajas tales como la mitigación de desvanecimientos multitrayectoria, la alta seguridad, la baja densidad de potencia y las capacidades anti-interferencia. Dichas secuencias de caos son basadas en el dominio de la frecuencia y transmitidas a través de las subportadoras no continuas detectadas por el CR. Con esto sólo el receptor legítimo puede recuperar la imagen original transmitida. Incluso cuando los nodos maliciosos obtienen de alguna manera el valor inicial o los parámetros clave de los generadores caóticos, no pudieron obtener la imagen original porque no pueden conseguir la clave secreta del cifrado AES tan fácilmente.

Los trabajos anteriores usan AES-128 y AES-256, para cifrar la información, sin realizar modificaciones. Considerando lo anterior en este trabajo se propone un algoritmo para cifrar imágenes que se transmitan en un canal de acceso dinámico al espectro.

Maria Saher et al. [21] proponen un algoritmo basado en AES para Radios cognitivos (Cognitive Radio-Advanced Encryption Standard, CR-AES). Los cambios destacados en esta propuesta son los siguientes:

- En CR-AES, se presenta una expansión de clave que depende de los datos.
- Se introduce un nuevo método para desplazar filas utilizando información del byte de datos anterior. Esto mantiene la red a salvo de los atacantes.
- Se reducen las rondas de 10 a 6 para aumentar la eficiencia computacional.

El CR-AES respeta las transformaciones que se usan en AES-Estándar, pero utiliza un nuevo proceso de generación de claves.

El proceso de expansión de claves propuesto tiene los siguientes pasos:

- Se aplica la operación XOR entre la primera palabra del dato de entrada y los datos anteriores.
- Se ejecuta el desplazamiento circular izquierdo de un byte en la palabra de entrada.
- Se realiza la sustitución de bytes mediante el uso la caja de sustitución en cada byte.
- Se realiza una XOR entre el resultado del paso anterior con la ronda de rcon, los valores de rcon se pueden observar en la Tabla IV, capítulo 2.

Así en CR-AES, se propone una transformación de cambio de filas, donde las filas de datos de entrada se desplazan circularmente a la izquierda, de acuerdo con un vector de desplazamiento de 8 bits que contiene el último byte de datos anteriores. La primera fila se desplaza según el valor de los primeros 2 bits, la segunda fila según los siguientes 2 bits, y así sucesivamente.

Para evaluar el rendimiento de CR-AES se utilizan histogramas y correlación estadística. Evaluando con nueve imágenes (i) Cameraman, (ii) House, (iii) Pepper, (iv) Building, (v) House-2, (vi) Doll, (vii) Raccoon, (viii) Parrot y (ix) MRI-Skull. Se observa que las distribuciones de las imágenes cifradas tienden a una distribución uniforme.

Para la correlación se toman el promedio de la correlación entre los píxeles vecinos en direcciones horizontal, vertical y diagonal. Para las imágenes de prueba sin cifrar, los valores de correlación están en el rango de 0.80-0.95. Con respecto a los valores de correlación para las imágenes cifradas son inferiores a 0.0299, lo que implica poca correlación entre los pixeles vecinos en cualquier dirección.

4 Desarrollo del sistema criptográfico propuesto

El algoritmo AES-Estándar en modo contador fue modificado para obtener el algoritmo propuesto llamado AES-Caos. Dichas modificaciones se pueden dividir en dos secciones importantes.

La Sección 4.1 describe las modificaciones hechas a AES-Estándar tales como: la disminución del número de rondas, el uso del modo contador con AES como cifrado a bloques y la adición de un bloque generador de caos.

La Sección 4.2 describe el bloque generador de caos, que engloba varias funciones tales como el mapa de Kent, el bloque decimal a entero, la caja de sustitución dinámica y la función de rotación.

4.1 Modificaciones a AES

El algoritmo propuesto, considera AES en modo contador. Del algoritmo AES-Estándar, se tomaron todas las funciones incluyendo la expansión de la clave, pero se redujo a una ronda estándar. De esta manera, se obtiene un AES reducido a tres rondas que son la ronda inicial, una ronda estándar y la ronda final. Esto permitirá disminuir el tiempo de cifrado respecto a los otros dos algoritmos, incrementando la probabilidad de transmisión cuando se tenga una oportunidad en la red de radios cognitivos.

El número de rondas es reducido, por lo que el tiempo de cifrado decrementa significativamente y también la seguridad del cifrado disminuye. Para evitar la disminución en la seguridad del cifrado y mantener un cifrado rápido, se usa AES de manera análoga a modo contador sustituyendo el contador y la clave por un bloque generador de caos determinista basado en el mapa de Kent para todos los bloques de la imagen. Al usar el bloque generador de caos aseguramos que la secuencia cifrante, sea suficientemente fuerte. La Figura 15, muestra la modificación del cifrado de bloque.

Por último, se modificó la forma en que se selecciona el vector $rcon$ para generar una clave nueva, ya que al haber sólo dos expansiones de la clave se usaban únicamente los dos primeros elementos de los 11 que se tiene originalmente para AES-Estándar. Con la modificación estos dos vectores se seleccionan de manera pseudoaleatoria y uniforme entre los 10 posibles. $rcon$ se define como $rcon = [x^{i-1}, \{00\}, \{00\}, \{00\}]$ donde $i-1$ son potencias de x en el campo GF(2^8), se puede ver la Tabla IV.

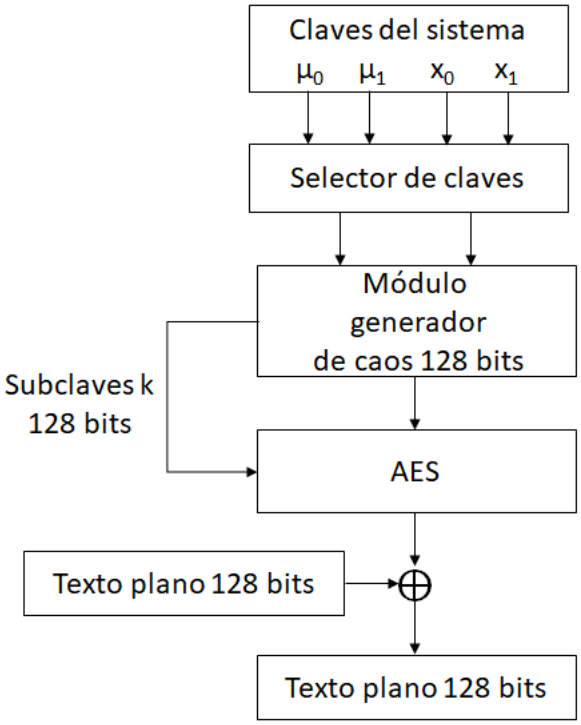

Figura 15. Diagrama a bloques AES-Caos

4.2 Bloque generador de caos

A continuación, se describirá el bloque generador de caos. La Figura 16, muestra las funciones que se usaron para el bloque generador de caos.

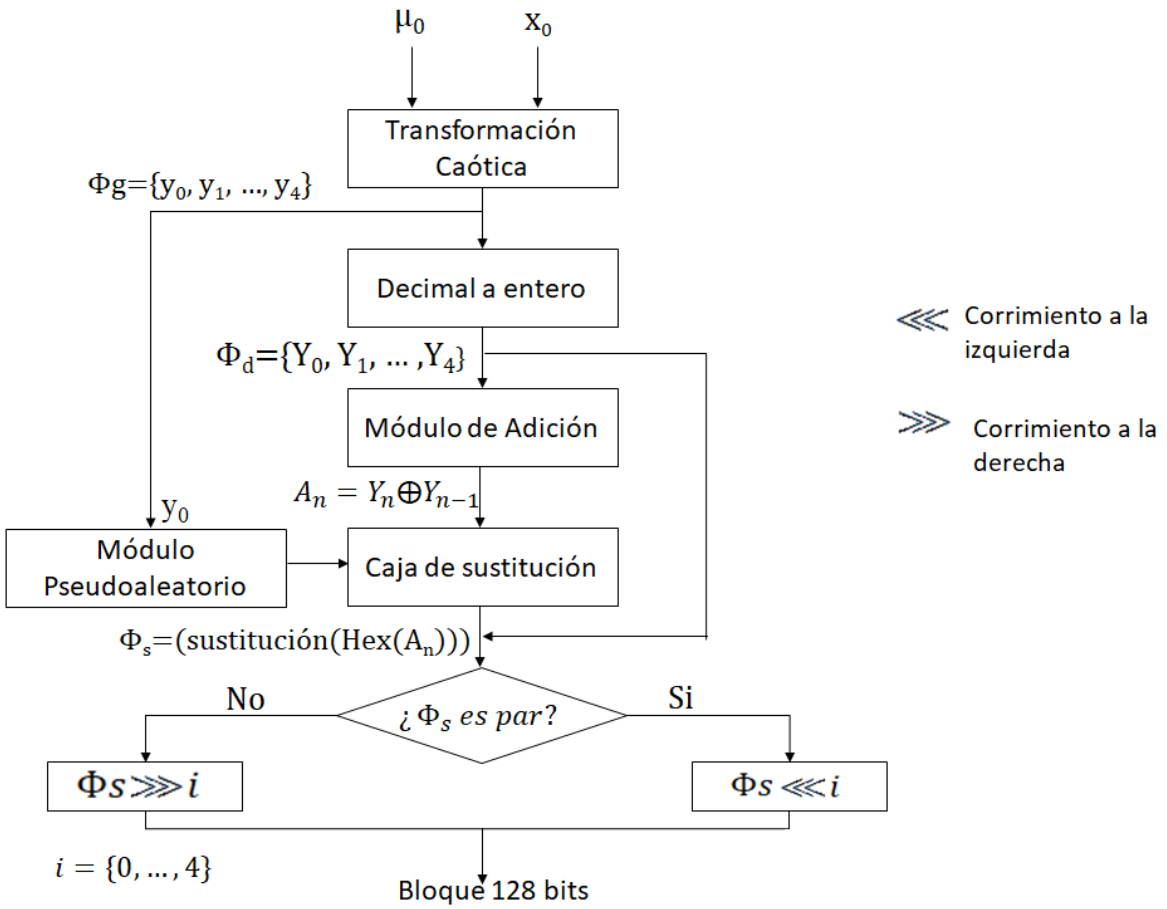

Figura 16. Diagrama a bloques del módulo generador de caos

El bloque generador de caos usa el mapeo de Kent, que fue descrito en la sección anterior en la Ecuación 11.

Se aplica un valor de x_0 y μ_0 en la Ecuación 11 se generan secuencias de números $\phi_g = \{y_0, y_1, \dots\} \epsilon (0,1)$, así el comportamiento determinista depende de la condición inicial x_0 y el parámetro de control μ. Por lo tanto, si x_0 y μ cambian los valores de ϕ_g serán otros.

El bloque Decimal a entero realiza un escalamiento de los valores de ϕ_g para generar $\phi_d = [Y_0, Y_1, \dots]$, y conserva la parte entera, esto se hace para trabajar en el intervalo de los pixeles [0,255]. Este proceso se realiza a partir de la Ecuación 12.

$$\phi_d = \phi_g \cdot 2^{16} \qquad (12)$$

El bloque de adición realiza la operación XOR, entre el valor actual y el anterior, para cada elemento del vector. Cuando se trata del primer valor del vector el anterior se considera cero.

La caja de sustitución funciona igual que la de AES-Estándar, pero no es una caja fija, para cada bloque de 128 bits se genera una caja de sustitución diferente. Para generar la caja de sustitución se llena un vector de 0x00 a 0xFF, y se genera un número pseudoaleatorio N que va de 0 a 255. La semilla del generador de números pseudoaleatorio es el primer número que se generó en el mapa Kent, de esta manera la caja cambiará para cada bloque de datos. El número pseudoaleatorio señala la posición del vector de donde se elige un número y se acomoda en la primera posición de la caja de sustitución, para la siguiente ronda, habrá 255-1 valores en el vector y así sucesivamente hasta que la caja ya este llena.

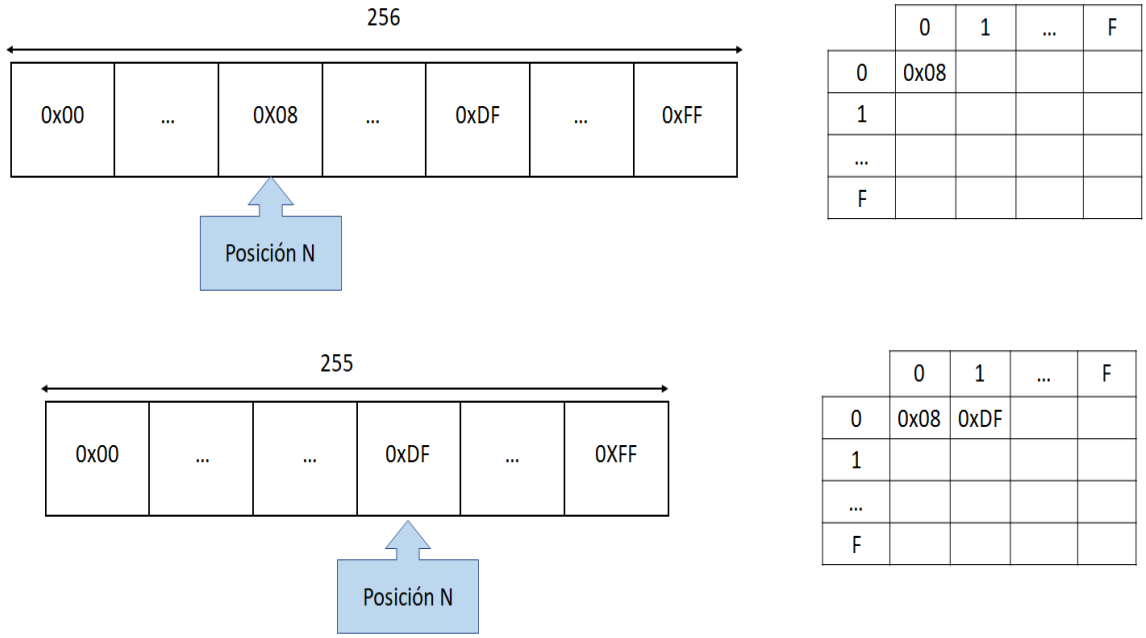

Figura 17. Formación de la caja de sustitución

El bloque de Rotación hace un corrimiento circular a nivel de bit, como lo muestra la Figura 12. La primera fila de la matriz se desplaza una vez, la segunda dos y así consecutivamente. Para decidir hacia donde se hará el corrimiento se llama al generador de caos, con la condición inicial y el parámetro de control diferentes al primero, si este número es par el corrimiento se hará a la izquierda y si es impar se hará a la derecha. A la salida se obtiene 128 bits, que son usados para la clave, y como bloque de entrada en AES para generar la secuencia cifrante con la que se hará una XOR al bloque de la imagen.

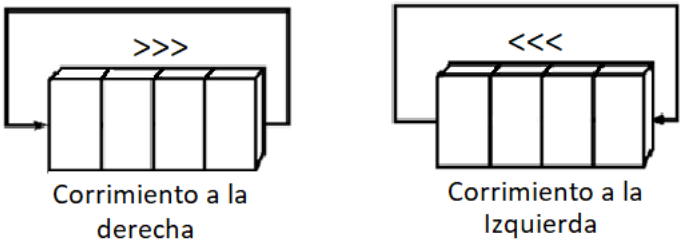

Figura 18. Corrimiento a nivel de bit

5 Análisis de Resultados

Para las pruebas del algoritmo criptográfico, se consideran imágenes a color BMP debido a que los mapas de bits conservan toda la información de imagen. Las imágenes que usan para las pruebas son de diferentes tamaños: 256×256, 1024×1024, 2048×2048 y 4096×4096 pixeles. Las pruebas se realizaron en Python 2.7, en una computadora, con procesador Intel Core i5-7200U a 2.5GHz, 8GB en RAM, y arquitectura de 64 bits. Las pruebas se realizan sobre los criptogramas generados y consideran las siguientes: histogramas, diagramas de dispersión, cálculo de la entropía y correlación. Debido a las características de la transmisión en las redes de radio cognitivo, la velocidad de cifrado debe ser alta, para atender los cortos lapsos de tiempo de transmisión debido a las constantes interrupciones de la comunicación. Entonces, se considera medir el tiempo de cifrado.

5.1 Análisis de Histograma

En el campo procesamiento de imágenes, un histograma de una imagen a color es una representación de la distribución del color de los pixeles de una imagen. En las imágenes digitales, un histograma representa el número de píxeles (eje y) que tiene cada matriz en el intervalo del color o saturación del color (eje x) se consideran valores de 0 a 255.

En los histogramas obtenidos de las imágenes cifradas es deseable una distribución uniforme de los pixeles, esto indica que el cifrado es robusto contra el ataque estadístico.

En la Figura 19, se observa el histograma de la imagen de Lena a color de 512×512 pixeles, note que, en las tres capas de color la mayoría de los pixeles se concentran en una región.

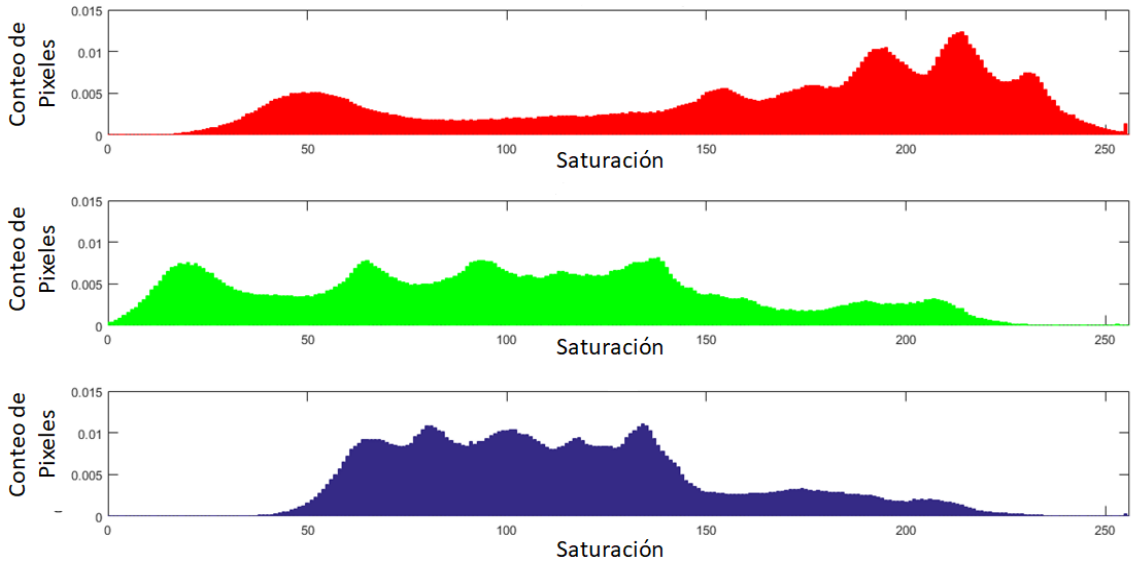

Figura 19. Histograma de la imagen Lena a color

Los histogramas de las imágenes cifradas de Lena se pueden observar en las figuras 20, 21 y 22. Se observa que los pixeles están distribuidos sobre todo el intervalo [0,255], en consecuencia, la distribución estadística tiende a una distribución uniforme.

En la Figura 20 se utiliza AES-Estándar para el cifrado, se puede ver que la distribución no es suficientemente plana.

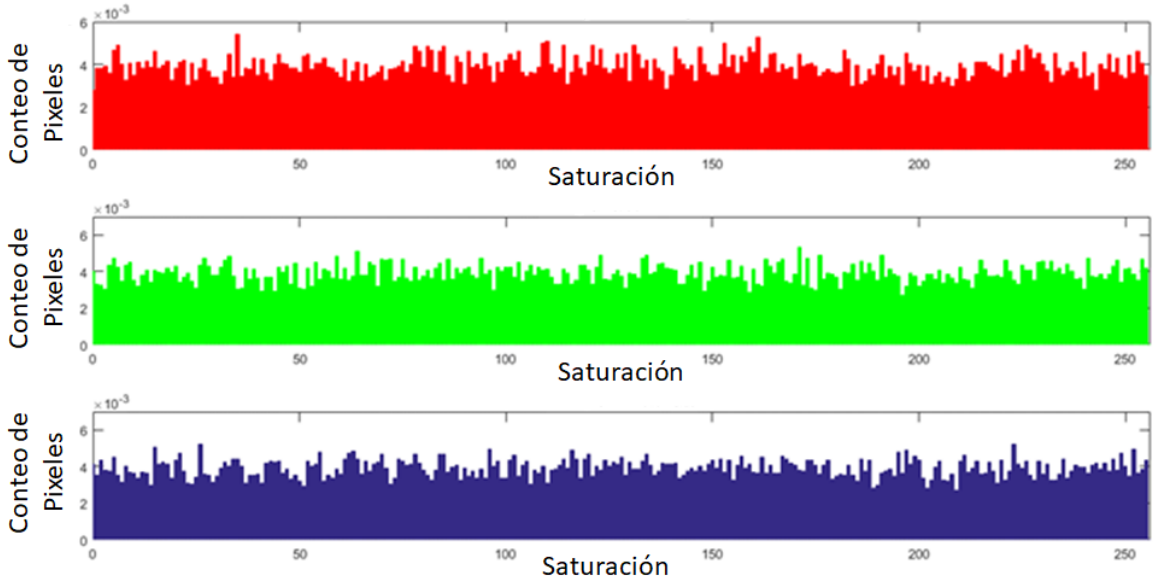

Figura 20. Histograma de la imagen cifrada con AES-Estándar

La Figura 21 muestra los histogramas obtenidos de la imagen cifrada con AES-Contador, se puede observar que los pixeles están distribuidos de mejor manera sobre el intervalo [0,255].

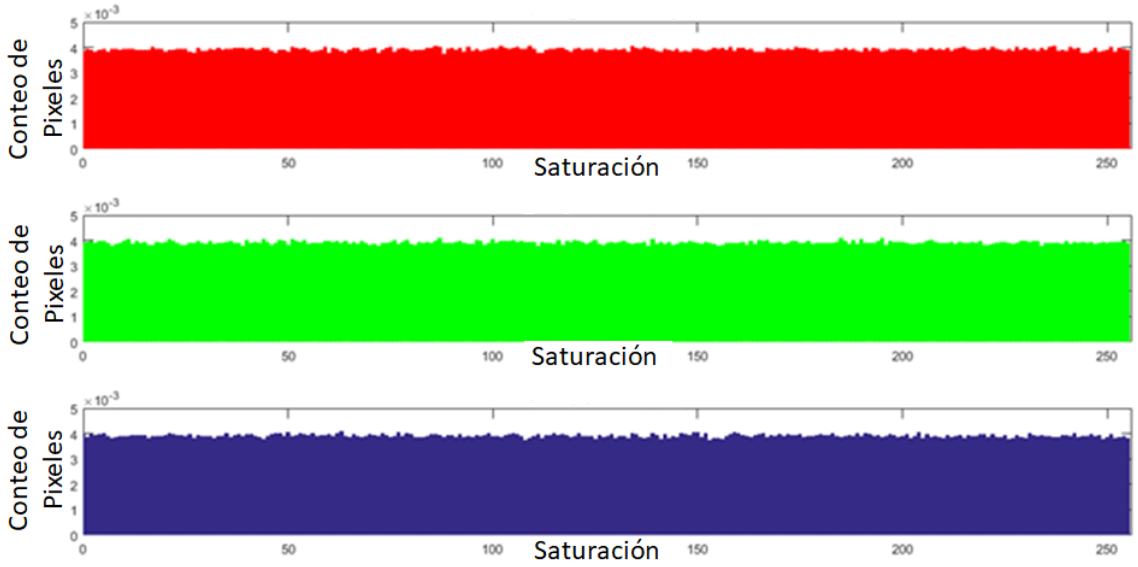

Figura 21. Histograma de la imagen cifrada con AES-Contador

Finalmente, la Figura 22 muestra los histogramas obtenidos del cifrado de Lena con el algoritmo propuesto AES-Caos, de igual manera que en AES-Contador, se puede observar que la distribución de los pixeles tiende a ser uniforme. En consecuencia, esto significa que el algoritmo propuesto no proporciona evidencia alguna relacionada con el tipo de imagen cifrada, lo cual indica que es robusto contra el ataque estadístico.

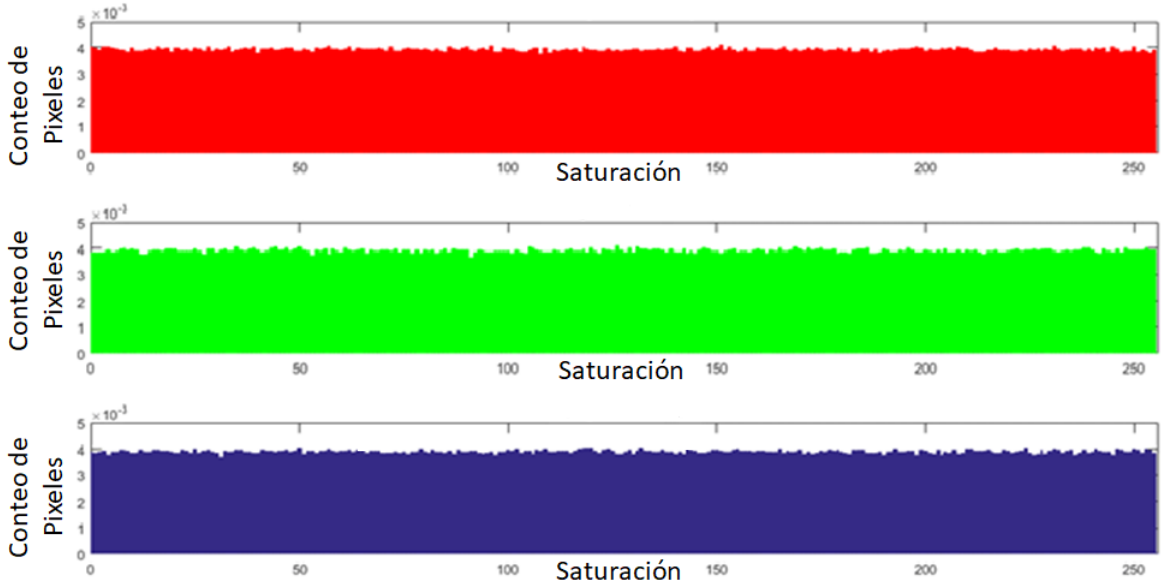

Figura 22. Histograma de la imagen cifrada con AES-Caos

5.2 Diagramas de dispersión

El diagrama de dispersión es un tipo de diagrama matemático que utiliza las coordenadas cartesianas para estudiar la correlación entre dos pixeles adyacentes. El diagrama muestra estos pares como una nube de puntos.

La alta redundancia de pixeles es una de las características naturales de las imágenes. En esta sección se considera, que el algoritmo de cifrado propuesto rompa la alta correlación de los pixeles vecinos en las direcciones horizontal vertical y diagonal.

Para poder observar la distribución de los pixeles, se realizó el diagrama de dispersión de la imagen sin cifrar, la imagen cifrada con AES-Estándar, AES-Contador y AES-Caos. Esto sólo se hizo para la imagen de 512×512 pixeles.

Los diagramas de dispersión de imagen se usan para examinar la similitud entre los pixeles de una imagen. En la Figura 23, se puede observar el diagrama de dispersión para la imagen sin cifrar. Se aprecia que la dispersión de los pixeles se concentra en una región específica.

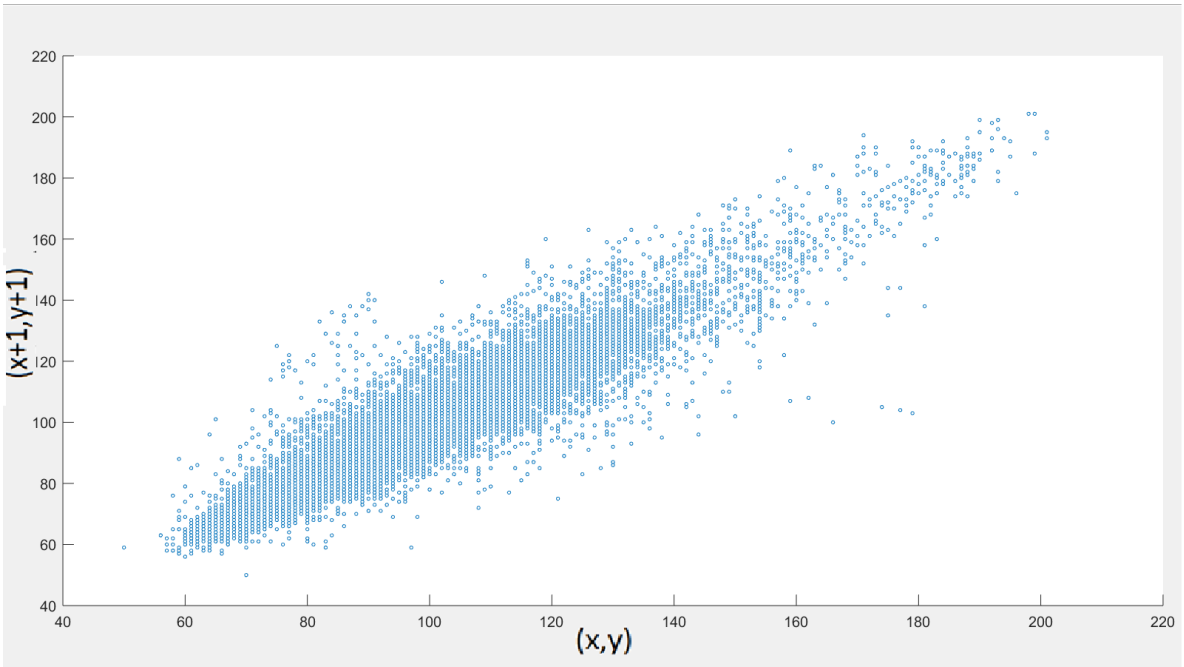

Figura 23. Diagrama de dispersión de la imagen sin cifrar

En la Figura 24, se observa el diagrama de dispersión de la imagen cifradas con AES-Estándar.

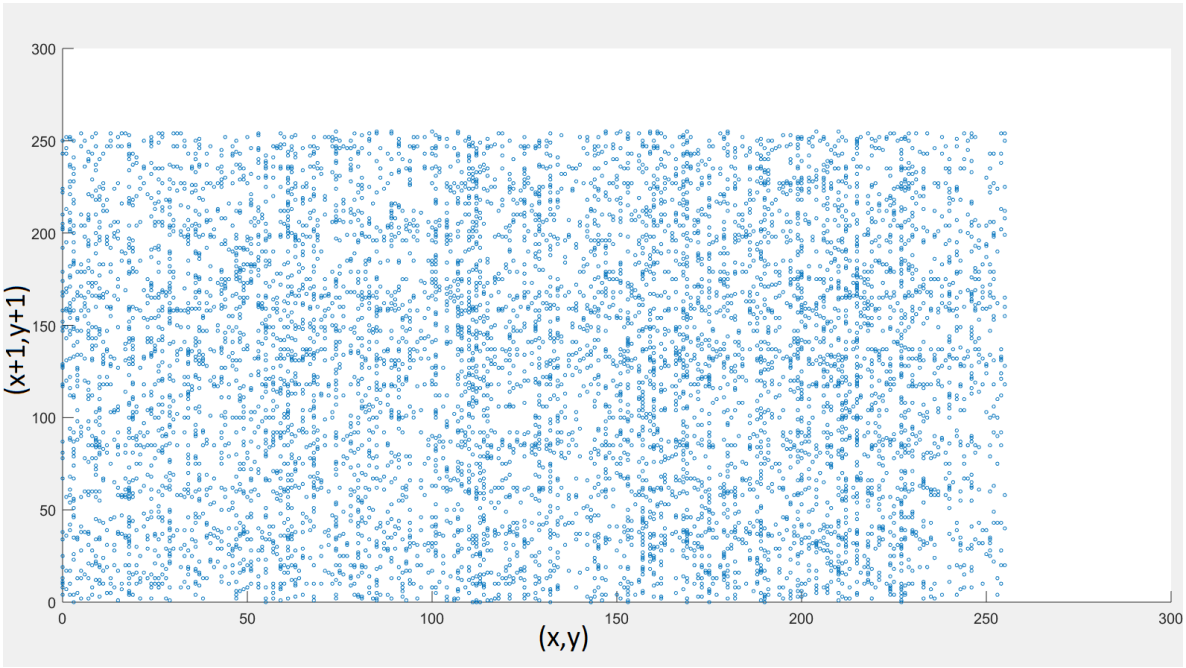

Figura 24. Diagrama de dispersión AES-Estándar

En la Figura 25, se observa el diagrama de dispersión de las de la imagen cifrada con AES-Contador.

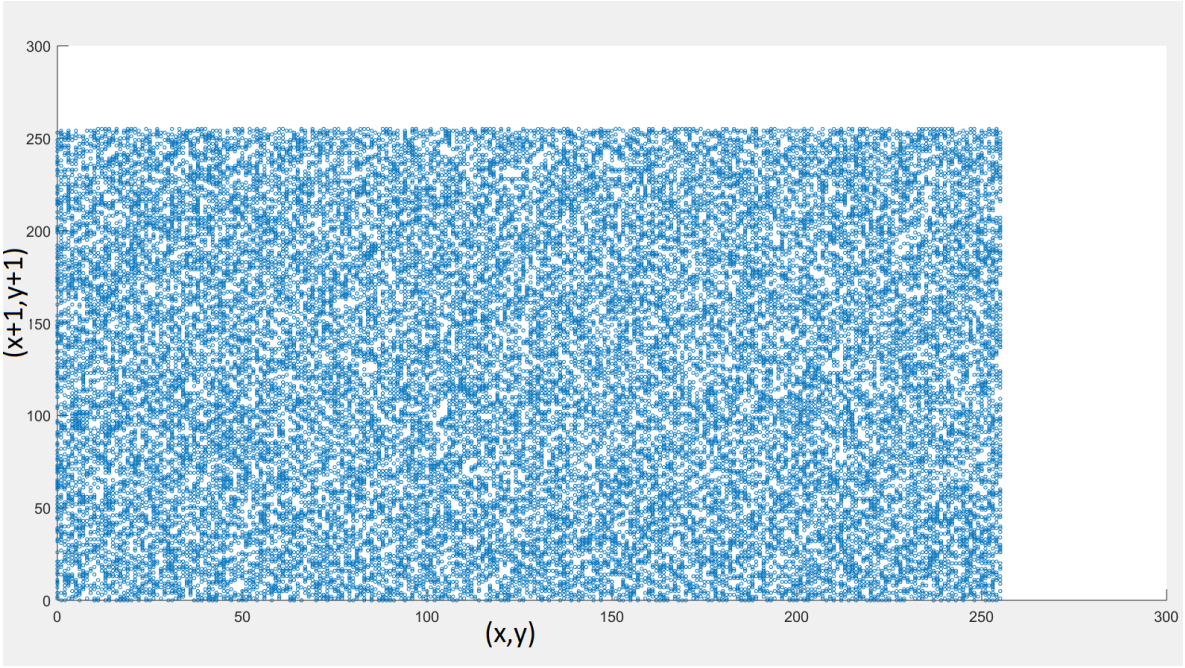

Figura 25. Diagrama de dispersión de AES-Contador

En la Figura 26, se observa el diagrama de dispersión de la imagen cifrada con AES-Caos.

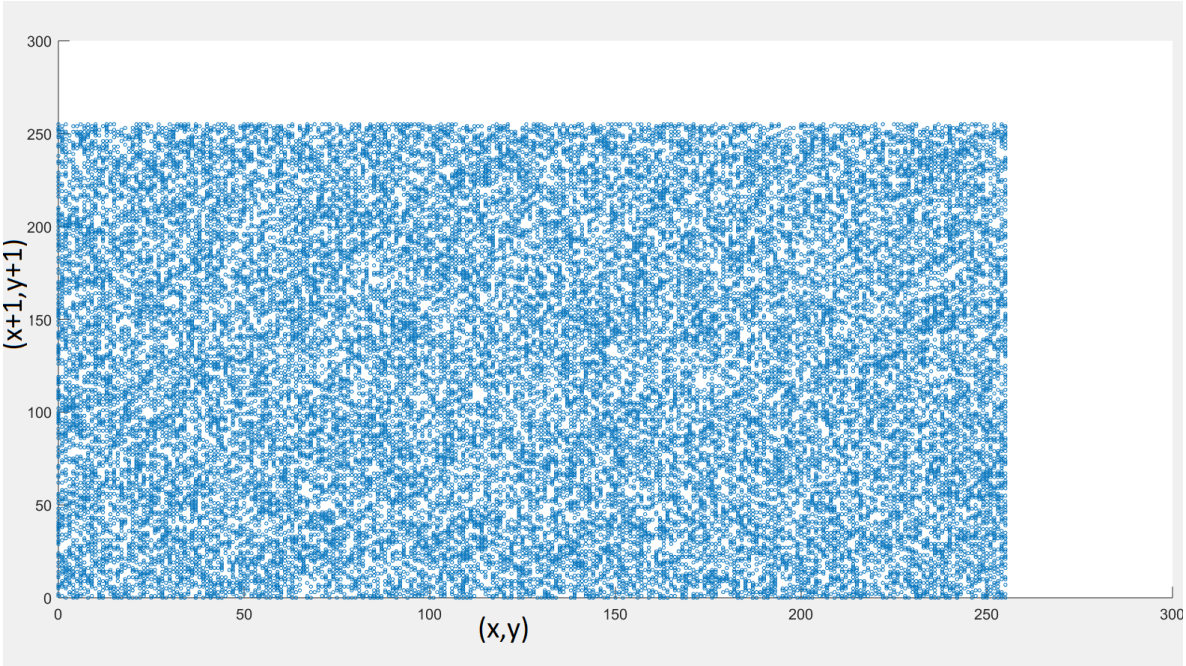

Figura 26. Diagrama de dispersión de AES-Caos

Los diagramas fueron graficados con la misma cantidad de puntos. Sin embargo, el diagrama de dispersión de la imagen cifrada con AES-Estándar parece que consta de menos puntos, pero los puntos se superponen debido a que son similares o iguales.

5.3 Análisis de Entropía

La entropía se define como la cantidad de información promedio que contienen los símbolos usados en un mensaje. Es decir, los símbolos con menor probabilidad son los que aportan más información; por ejemplo, si se considera como sistema de símbolos a las palabras en un texto, palabras frecuentes como “que”, “el”, “a” aportan poca información, mientras que palabras menos frecuentes como “corren”, “niño”, “perro” aportan más información.

La entropía de un mensaje X, es denotada por $H(x)$ [31], es el valor medio ponderado de la cantidad de información de los diversos estados del mensaje, la expresión usada para calcularla está dada por la Ecuación 13.

$$H(X) = -\sum_{i}^{k} P(x_i) \log_2 p(x_i) \tag{13}$$

Donde $P(x_i)$ representa la probabilidad de un símbolo de X, es decir $X = \{x_i : \mathrm{i} = 0, 1, 2, \ldots, \mathrm{k}\}$. Para un mensaje con alfabeto equiprobable, la probabilidad de que se dé

alguna combinación está dado por: $p(xi) = \frac{1}{k}$ donde k es el número de símbolos en el alfabeto [31].

La entropía de Shannon también puede utilizarse en imágenes. En este caso se utiliza la distribución de los valores de los pixeles en la imagen en vez de la probabilidad de ocurrencia de los símbolos.

La distribución de probabilidad de los valores del píxel se puede estimar contando el número de veces que aparece cada valor del píxel en la imagen y dividiendo el conteo por la cantidad total de pixeles [31].

De esta manera una imagen con una gran cantidad de sus pixeles con la misma intensidad va a tener una baja entropía, mientras que, con una distribución más uniforme de los valores, obtiene una alta entropía. Por lo que también puede verse la entropía de Shannon como una medida de la distribución de probabilidad de los valores de los pixeles.

Si $\mathrm{k} = 2(n-1)$ y la fuente genera $n = 8$ símbolos aproximadamente con igual probabilidad, entonces el valor de $H(x)$ es muy cercano a 8 [32]. Entonces, la entropía de las imágenes cifradas debe ser muy cercanas a 8. con lo cual se muestra que la información tiene un comportamiento estadístico muy cercano a la distribución estadística uniforme.

A continuación, se muestran los resultados de las mediciones de la entropía para diferente tamaño de imágenes de Lena.

La entropía obtenida de la imagen de Lena sin cifrar es 7.90535.

La Tabla VI, muestra el resultado de la medición de la entropía para Lena, en diferentes tamaños y cifrada con AES-Estándar, AES-Contador y AES-Caos.

Tabla VI. Resultados de entropía

Cifrado	Tamaño en pixeles			
	512	1024	2048	4096
AES-Estándar	7.98713	7.9932	7.9957	7.99802
AES-Contador	7.99976	7.99994	7.99999	8
AES-Caos	7.99978	7.99995	7.99999	8

En la Tabla VI se puede observar que para las imagen de 512×512 y 1024×1024 pixeles la entropía de la imagen cifrada con AES-Caos es mayor que la entropía de las imágenes cifradas con AES-Contador y AES-Estándar; para la imagen de 2048×2048 y 4096×4096 pixeles la entropía de la imagen cifrada con AES-Caos y AES-Contador es la misma, lo que implica que con tres rondas obtenemos la misma entropía que AES-Contador obtiene con 11.

5.4 Análisis de Correlación

La correlación de pixeles vecinos de la imagen cifrada se muestra en los diagramas de dispersión. Sin embargo, para cuantificar y comparar la correlación de dos pixeles

vecinos de una imagen, se usa el coeficiente de correlación r_{ij} para la imagen en claro y las imágenes cifradas.
La correlación se calcula horizontal, diagonal y vertical. La correlación cruzada discreta bidimensional. Se puede definir como:

$$r_{ij} = \frac{\sum_m \sum_n \left[f(m,nf) - \bar{f}\right][f(m+1,n+1) - \bar{f}]}{\sqrt{\sum_m \sum_n [f(m,n) - \bar{f}]^2 \sum_m \sum_n \left[f(m+1,n+1) - \bar{f}\right]^2}} \quad (14)$$

Aquí $f(m,n)$ es la intensidad del píxel o el valor de escala de grises en un punto (m,n) en la imagen, $\bar{f}$ es el valor medio de las matrices de intensidad f.
A continuación, se muestra la correlación para los pixeles en diferentes direcciones para las diferentes capas de color. La Tabla VII muestra la correlación de la imagen Lena de 512×512 pixeles sin cifrar.

Tabla VII Correlación de la imagen sin cifrar

	Rojo	Verde	Azul
Horizontal	0.9992	0.9999	0.9985
Vertical	0.9995	0.9995	0.9992
Diagonal	0.9987	0.9985	0.9977

Se puede observar la fuerte correlación entre los pixeles de la imagen sin cifrar. Entonces, la interpretación de los valores de la correlación considera que una baja correlación está relacionada con las imágenes cifradas y una alta correlación con las imágenes sin cifrar.

La correlación de la imagen sin cifrar es aproximadamente 1, que es la máxima correlación.

La Tabla VIII muestra la correlación de la imagen Lena de tamaño 512×512 pixeles cifrada con AES-Estándar, AES-Contador y AES-Caos. La Tabla IX muestra la correlación de la imagen Lena de 1024×1024 pixeles cifrada con AES-Estándar, AES-Contador, AES-Caos. La Tabla X muestra la correlación de la imagen Lena de 2048×2048 pixeles cifrada con AES-Estándar, AES-Contador, AES-Caos. La Tabla XI muestra la correlación de la imagen Lena de 4096×4096 pixeles cifrada con AES-Estándar, AES-Contador, AES-Caos.

En la Tabla VIII, se observa que la correlación de la imagen de 512×512, cifrada usando los métodos ya mencionados. En general AES-Estándar tiene una correlación mayor que AES-Contador y AES-Contador tiene una correlación mayor que la de AES-Caos, excepto en la matriz roja diagonal.

Tabla VIII. Correlación en Imágenes de 512 pixeles

	AES-Estándar	AES-Contador	AES-Caos
Rojo Horizontal	0.0089	0.0023	0.0013
Rojo Vertical	0.0163	0.0024	0.0020
Rojo Diagonal	0.0169	0.0001	0.0009
Verde Horizontal	0.0061	0.0017	0.0008
Verde Vertical	0.0082	0.0039	0.0001
Verde Diagonal	0.0203	0.0023	0.0008
Azul Horizontal	0.0021	0.0024	0.0007
Azul Vertical	0.0012	0.0047	0.0000
Azul Diagonal	0.0118	0.0017	0.0004

En la Tabla IX, la imagen cifrada con AES-Estándar tiene mayor correlación que las imágenes cifradas con los otros algoritmos, únicamente los valores de correlación de la imagen cifrada con AES-Contador son menores que los de AES-Caos para la matriz roja tanto vertical como horizontal y verde vertical.

Tabla IX. Correlación en Imágenes de 1024 pixeles

	AES-Estándar	AES-Contador	AES-Caos
Rojo Horizontal	0.003	0.0007	0.0017
Rojo Vertical	0.0125	0.0005	0.0000
Rojo Diagonal	0.0035	0.0011	0.0000
Verde Horizontal	0.0032	0.0014	0.0001
Verde Vertical	0.0083	0.0002	0.0008
Verde Diagonal	0.0128	0.0005	0.0007
Azul Horizontal	0.0194	0.0022	0.0003
Azul Vertical	0.0154	0.0006	0.0007
Azul Diagonal	0.0059	0.0015	0.0013

En la Tabla X, se puede observar que los valores de correlación de la imagen cifrada con AES-Caos están muy próximos a los valores de correlación obtenidos de la imagen cifrada con AES-Contador.

Tabla X. Correlación en imágenes de 2048

	AES-Estándar	AES-Contador	AES-Caos
Rojo Horizontal	0.0108	0.0005	0.0001
Rojo Vertical	0.0095	0.0003	0.0001
Rojo Diagonal	0.0042	0.0009	0.0002
Verde Horizontal	0.0138	0.0003	0.0002
Verde Vertical	0.0047	0.0000	0.0009
Verde Diagonal	0.0019	0.0004	0.0003
Azul Horizontal	0.0025	0.0002	0.0004
Azul Vertical	0.0044	0.0001	0.0002

Azul Diagonal	0.0149	0.0006	0.0001

En la Tabla XI, se puede observar que los valores de correlación de la imagen cifrada con AES-Caos están muy próximos a los valores de correlación obtenidos de la imagen cifrada con AES-Contador.

Tabla XI. Correlación en imágenes de 4094 pixeles

	AES-Estándar	AES-Contador	AES-Caos
Rojo Horizontal	0.0108	0.0005	0.0001
Rojo Vertical	0.0095	0.0003	0.0003
Rojo Diagonal	0.0042	0.0009	0.0002
Verde Horizontal	0.0138	0.0003	0.0001
Verde Vertical	0.0047	0.0000	0.0000
Verde Diagonal	0.0019	0.0004	0.0005
Azul Horizontal	0.0025	0.0002	0.0002
Azul Vertical	0.0044	0.0001	0.0000
Azul Diagonal	0.0149	0.0006	0.0000

5.5 Análisis de Tiempo

Para medir el tiempo de procesamiento de cada algoritmo, se utilizó el módulo time de Python. Este módulo proporciona varias funciones relacionadas con el tiempo.

La función time. time ( ) devuelve el tiempo en segundos como un número de punto flotante. Es importante tomar en cuenta que, aunque el tiempo siempre se devuelve como un número de punto flotante, no todos los sistemas proporcionan el tiempo con una precisión mejor que 1segundo [33].

Para medir el tiempo de cifrado de la imagen, el tiempo inicial se colocó al abrir la imagen, por lo que el tiempo se mide desde que se abre la imagen hasta que se reconstruye la imagen cifrada. La misma imagen se cifro 5 veces para cada algoritmo y obtener un promedio. Esto se hizo en los tres algoritmos.

En la Tabla XII, se puede observar el tiempo de cifrado en segundos para las diferentes imágenes y cifrados con un intervalo de confianza del 95%. El tiempo de cifrado de AES-Caos es un 90% menor que el de AES-Estándar y 60% menor que el de AES-Contador, debido a la reducción de las rondas.

Tabla XII. Tiempo de cifrado

	AES E	AES C	AES-Caos
512×512	158.8 s	48.8 s	17.9 s
1024×1024	635.6 s	196.5 s	70.4 s
2048×2048	2546.5 s	780.0s	278.6 s
4096×4096	10176.9 s	3096.2 s	1109.3 s

Las pruebas estadísticas son usadas para decidir si una determinada muestra o conjuntos de datos responde a un patrón o puede considerarse aleatoria. Para determinar la aleatoriedad de las imágenes cifradas se usará las pruebas estadísticas NIST.

5.6 NIST Test Suite

Las pruebas NIST son un conjunto de pruebas estadísticas para valoración de generadores de números aleatorios y pseudoaleatorios para aplicaciones criptográficas. Este software es de uso libre y fue desarrollado en el Instituto Nacional de Estándares y Tecnología de los Estados Unidos. El NIST Test Suite es un paquete estadístico que consta de 15 pruebas que se desarrollaron para probar la aleatoriedad de secuencias binarias, este caso la aleatoriedad de las imágenes cifradas. Las pruebas con las que cuenta son [34]:

i. La Prueba de frecuencia monobit (Frequency MonobitTest, FMT)
ii. Prueba de frecuencia dentro de un bloque (Frequency Test within a Block, FTB)
iii. La prueba de ejecuciones (Runs Test, RT)

iv. Pruebas de ejecuciones de unos más larga en un bloque (Test for the Longest Run of Ones in a Block, TLROB)
v. Prueba de rango de matriz binaria (Binary Matrix Rank Test, BMRT)
vi. Prueba de Transformada Discreta de Fourier (Discrete Fourier Transform Test, DFTT).
vii. Prueba de coincidencia de plantillas no superpuestas (Non-overlapping Template Matching Test, NoTMT)
viii. Prueba de coincidencia de plantillas superpuestas (Overlapping Template Matching Test, OTMT)
ix. Prueba de "estadística universal" de Maurer (Maurer's "Universal Statistical" Test, MT)
x. La prueba de complejidad lineal (Linear Complexity Test, LCT)
xi. La prueba en serie (Serial Test, ST)
xii. La Prueba de Entropía Aproximada (Approximate Entropy Test, AET)
xiii. La Prueba de Sumas Acumulativas (Cumulative Sums Test, Cusums)
xiv. Prueba de excursiones aleatorias (Random Excursions Test, RET)
xv. Prueba variante de excursiones aleatorias (Random Excursions Variant Test, REVT)

Las pruebas del NIST fueron aplicadas a la imagen Lena de 1024×1024 cifrada con AES-Estándar, AES-Caos y AES-Contador.

Para este conjunto de pruebas la tasa de aprobación mínima para cada prueba estadística con la excepción de la prueba de la excursión aleatoria (variante) es aproximadamente igual a 123 para un tamaño de muestra igual 128 secuencias binarias. La tasa de aprobación mínima para la prueba de excursión aleatoria (variante) es aproximadamente igual a 10 para un tamaño de muestra igual a 14 secuencias binarias.

Para estas pruebas, cada P-valor es la probabilidad de que un generador de números aleatorios perfecto hubiera producido una secuencia menos aleatoria que la secuencia que se probó. Si se determina que un valor P para una prueba es igual a 1, entonces la secuencia parece tener una aleatoriedad perfecta. Un valor P de cero indica que la secuencia parece ser completamente no aleatoria.

Respecto a los P-valores la tasa de aprobación de este conjunto de pruebas se encuentra dentro del rango 0.01 y 0.9999.

Tabla XIII. Resultados de las pruebas de aleatoriedad de AES-Estándar

Prueba	Puntaje	P-Valores	Pasa
Prueba de frecuencia monobit	128/128	0.000002	No Pasa
Prueba de frecuencia dentro de un bloque	128/128	0.010840	Pasa
La prueba de ejecuciones	128/128	0.0513	Pasa
Pruebas de ejecuciones de unos más larga en un bloque	126/128	0.0500	Pasa
Prueba de rango de matriz binaria	128/128	0.654467	Pasa
Prueba de Transformada Discreta de Fourier	127/128	0.756476	Pasa
Prueba de coincidencia de plantillas no superpuestas	126/128	0.000042	No pasa
Prueba de coincidencia de plantillas superpuestas	128/128	0.066882	Pasa
Prueba de "estadística universal" de Maurer	0/128	0	No pasa
Prueba de complejidad lineal	126/128	0.299251	Pasa
Prueba en serie	20/128	0	No pasa
Prueba de Entropía Aproximada	39/128	0	No pasa
La Prueba de Sumas Acumulativas	128/128	0.01399	Pasa

Prueba de excursiones aleatorias	14/14	0.213309	Pasa
Prueba de excursiones aleatorias variante	14/14	0.350485	Pasa

Tabla XIV. Resultados de las pruebas de aleatoriedad de AES-Contador

Prueba	Puntaje	P-Valores	Pasa
Prueba de frecuencia monobit	128/128	0.082177	Pasa
Prueba de frecuencia dentro de un bloque	128/128	0.045730	Pasa
La prueba de ejecuciones	126/128	0.723129	Pasa
Pruebas de ejecuciones de unos más larga en un bloque	125/128	0.834308	Pasa
Prueba de rango de matriz binaria	122/128	0.671779	Pasa
Prueba de Transformada Discreta de Fourier	126/128	0.063482	Pasa
Prueba de coincidencia de plantillas no superpuestas	127/128	0.090936	Pasa
Prueba de coincidencia de plantillas superpuestas	127/128	0.253551	Pasa
Prueba de "estadística universal" de Maurer	0/128	0	No Pasa
Prueba de complejidad lineal	128/128	0.392456	Pasa
Prueba en serie	20/128	0	No Pasa
Prueba de Entropía Aproximada	0/128	0	No Pasa
La Prueba de Sumas Acumulativas	128/128	0.324180	Pasa
Prueba de excursiones aleatorias	14/14	0.213309	Pasa
Prueba de excursiones aleatorias variante	14/14	0.122325	Pasa

Tabla XV. Resultados de las pruebas de aleatoriedad de AES-Caos

Prueba	Puntaje	P-Valores	Pasa
Prueba de frecuencia monobit	128/128	0.65703	Pasa
Prueba de frecuencia dentro de un bloque	128/128	0.43727	Pasa
La prueba de ejecuciones	125/128	0.04500	Pasa
Pruebas de ejecuciones de unos más larga en un bloque	125/128	0.02519	Pasa
Prueba de rango de matriz binaria	127/128	0.58393	Pasa
Prueba de Transformada Discreta de Fourier	127/128	0.08217	Pasa
Prueba de coincidencia de plantillas no superpuestas	127/128	0.77276	Pasa
Prueba de coincidencia de plantillas superpuestas	127/128	0.17029	Pasa
Prueba de "estadística universal" de Maurer	0/128	0	No Pasa
Prueba de complejidad lineal	126/128	0.23276	Pasa
Prueba en serie	125/128	0.03363	Pasa
Prueba de Entropía Aproximada	54/128	0	No Pasa
La Prueba de Sumas Acumulativas	125/128	0.422034	Pasa
Prueba de excursiones aleatorias	14/14	0.232760	Pasa
Prueba de excursiones aleatorias variante	14/14	0.293610	Pasa

De las Tablas XIII, XIV y XV, podemos concluir que AES-Estándar pasa el 66% de las pruebas incluidas en estas baterías, AES-Contador pasa el 80% de las pruebas y AES-Caos pasa el 86% de las pruebas.

5.7 Módulo de GNU Radio

GNU Radio trabaja con sus propias funciones y bloques por lo que si se desea agregar un módulo nuevo es necesario crear un módulo fuera del árbol (OOT, Out-of-tree modules). Esto implica crear un módulo de GNU Radio que no se anida dentro del árbol principal de GNU Radio. Esto nos permite mantener nuestro propio código y tener funcionalidades adicionales junto con el código principal.

Para agilizar el desarrollo de módulos GNU Radio cuenta con gr-modtool que es un sistema de construcción de módulos, gr-modtool es un script que tiene como objetivo minimizar la generación de código repetitivo editando automáticamente archivos "make". gr-modtool se basa en el uso de plantillas. De esta forma el desarrollador puede saltar directamente a la codificación.

Es importante considerar que gr-modtool hace muchas suposiciones sobre lo que se espera del código. Cuantos más cambios específicos tiene el módulo, gr-modtool menos útil será.

Los módulos generados se pueden observar en GNU Radio Companion es la interfaz gráfica de GNU Radio. GRC permite crear aplicaciones de procesamiento de señales con solo arrastrar y soltar módulos. Además, viene con un conjunto de herramientas listas para su uso y programas de utilidades [35].

Los módulos de GNU Radio pueden ser programados con Python o con C++, para que sea posible visualizarlos en GRC, se utiliza el Lenguaje de Marcado Extensible (Extensible Markup Language, XML), que permite a gr-modtool crear etiquetas compatibles con Python y GNU-Radio. En el Anexo B, se puede observar el código en Python del módulo AES-Caos para GNU Radio y en el Anexo C se puede observar el código XML para GNU Radio.

La Figura 27 muestra el módulo de AES-Caos en GNU radio.

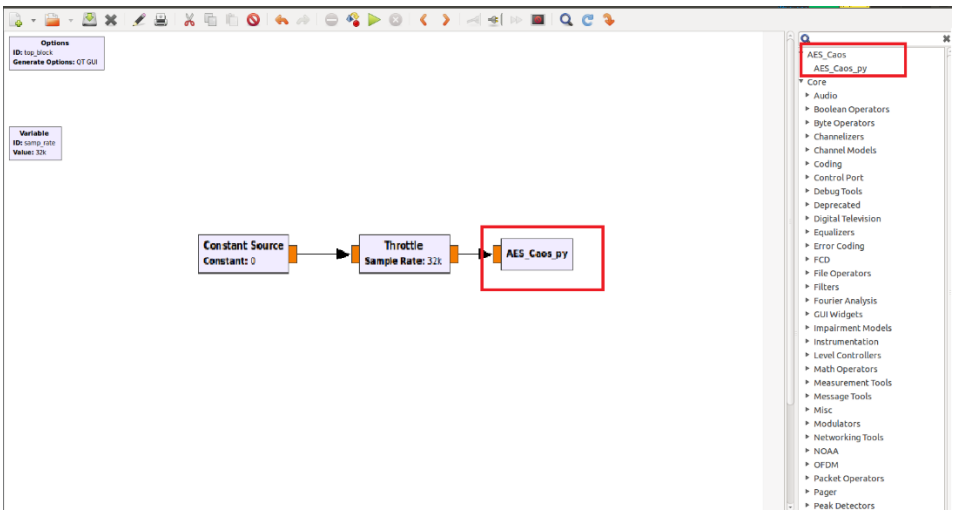

Figura 27. Módulo AES-Caos en GNU Radio

6 Conclusión

Este proyecto de investigación presenta una adaptación del algoritmo criptográfico AES para Radios Cognitivos (CR) debido a que estos dispositivos tienden a ser vulnerables a diversos ataques. Una de las características más importantes de los CR es que hacen uso del espectro de manera oportunista y dinámica, lo que implica lapsos de transmisión muy cortos, por lo que es importante tener cifrado rápido para agilizar la trasmisión en situaciones de este tipo. Al reducir el número de rondas a tres rondas en total y usar el cifrado a bloques en modo contador, se obtiene un cifrado más rápido en caso AES-Caos es 90% más rápido que AES-Estándar y 65% más rápido que AES-Contador.

Es importante mencionar que no sólo la velocidad de cifrado es importante, sino que también es transcendental que la información esté cifrada de manera segura, al reducir el número de rondas la seguridad del cifrado disminuye. Para compensar la seguridad perdida con la reducción de las rondas se usa un mapa caótico determinista (mapa de Kent), que genera tanto las claves usadas por AES-Caos como la secuencia a cifrar, obteniendo de esta manera una secuencia cifrante fuerte que se usa para cifrar un bloque de información y así obtener un cifrado seguro. Esto fue comprobado con la correlación que nos permite saber que tan uniformemente distribuida esta la información en una imagen cifrada, por lo tanto, una correlación alta implica una mejor distribución de la información. Con AES-Caos se obtiene una correlación 92% menor que la de AES-Estándar y 35% menor que AES-Contador, lo que implica que no sólo mejora en tiempo. También se hicieron otras pruebas estadísticas y en todas estas resultó ser mejor AES-Caos que AES-Estándar y AES-Contador.

7 Trabajo a futuro

Este trabajo de investigación es un buen nicho de oportunidad para continuar con el tema de algoritmos criptográficos para radios cognitivos, ya que es un tema poco investigado y dichos dispositivos tienden a ser atacados debido a algunas de sus características más importantes.

El siguiente paso es realizar una optimización del algoritmo propuesto y pruebas de funcionamiento en una red con características cognitivas.

Referencias

[1] M. Khasawneh y A. Agarwal, «A Collaborative Approach for Monitoring Nodes Behavior during Spectrum Sensing to Mitigate Multiple Attacks in Cognitive Radio Networks,» *Hindawi, Security and Communication Networks,* vol. 217, pp. 1-17, 2017.

[2] I. Akyildiz, W.-Y. Lee, M. Vuran y . S. Mohanty, «A Survey on Spectrum Management in Cognitive Radio Networks,» *IEEE Communications Magazine,* vol. 163, pp. 40-48, 2008.

[3] P. Crocioni, «Is allowing trading enough Making secondary markets in spectrum work,» *Telecommunications Policy,* vol. 33, pp. 441-458, 2009.

[4] Y. Moustafa, I. Mohamed y A. Mohamed, «Routing Metrics of Cognitive Radio Networks,» *IEEE Communications Surveys & Tutorials,* vol. 16, nº 1, pp. 92-109, 2014.

[5] W. A. A. Muhammad , S. Muhammad y A. Munam, «A Review: Security Challenges in Cognitive Radio Networks,» de *23rd International Conference on*, Huddersfield, 2017.

[6] V. C. Koradia, «Modification in Advanced Encription Standard,» *Journal of Information, Knowledge and Research In,* vol. 2, nº 2, pp. 356-358, 2013.

[7] B. Subramanyan, V. M. Chhabria y T. Sankar babu, «Image Encryption Based On AES Key Expansion,» de *Second International Conference on Emerging Applications of Information Technology*, Madurai, India, 2011.

[8] A. Ahmed, A. Mai, R. Jian y L. Tongtong, «Defense Against Primary User Emulation Attacks in Cognitive Radio Networks Using Advanced Encryption Standard,» *IEEE Transactions on Information Forensics and Security,* vol. 9, nº 5, pp. 772-781, 2014.

[9] A. Alahmadi, M. Abdelhakim, J. Ren y T. Li, «Mitigating Primary User Emulation Attacks in Cognitive Radio Networks Using Advanced Encryption Standard,» de *Signal Processing for Communications Symposium*, 2013.

[10] Z. Lin, Y. Jiaolong y W. Zhiqiang, «Secured Chaotic Cognitive Radio System Using Advanced Encryption Standard,» *IEEE 26th International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC): Fundamentals,* pp. 7-11, 2015.

[11] S. K. Kirti Prakash Choudhury, «Comparative Analysis of Different Modified Advanced Encryption Standard Algorithms over Conventional Advanced Encryption Standard,» *International Journal of Current Research and Review,* vol. 9, nº 22, pp. 31-34, 2017.

[12] M. Zeghid, M. Machhout, A. Baganne, R. Tourki y L. Khriji, «A Modified AES Based Algorithm for Image Encryption,» de *World Academy of Science, Engineering and Technology*, Tunisian/French, 2007.

[13] T. Minh-Triet , B. Doan-Khanh y D. Anh-Duc, «Gray S-box for Advanced Encryption Standard,» de *International Conference on Computational Intelligence and Security*, 2008.

[14] A. S. Abdulkarim, E. M. H. Bahaa y E. F. .. H. Abd, «An Efficient Modified Advanced Encryption Standard (MAES),» *IJCSNS International Journal of Computer Science and Network Security,* vol.

10, nº 2, pp. 226-232, 2010.

[15] P. Ritu y K. Vikas, «Efficient Implementation of AES,» *International Journal of Advanced Research in Computer Science and Software Enginnering*, vol. 3, nº 7, pp. 290-295, 2013.

[16] K. Pravin, H. Avinash, B. Gautam, T. Ekant y K. Rahul, «Modified Advanced Encryption Standard,» *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 4, nº 1, pp. 21-23, 2014.

[17] A. M. Atteya y A. H. Madian, «A Hybrid Chaos-AES Encryption Algorithm and Its Impelmention Based on FPGA,» *IEEE*, pp. 217-220, 2014.

[18] K. Ankit, S. Pradhan y S. Ghormade, «Design of AES-512 Algorithm for Communication Network,» *International Research Journal of Engineering and Technology (IRJET)*, vol. 9, nº 5, pp. 438-443, 2016.

[19] R. Riyaldhia, Rojalia y K. Aditya, «Improvement of Advanced Encryption Standard,» de *2nd International Conference on Computer Science and Computational Intelligence*, Bali, Indonesia, 2017.

[20] T. S. Sneha-Birendra y K. Manjeet, «Novel Approach to Image Encryption: Using a Combination of JEX Encoding–Decoding with the Modified AES,» de *Springer Nature Singapore Pte Ltd*, Singapore, 2018.

[21] M. Saher, A. Amin, I. A. Qureshi, M. A. Qureshi y M. M. Jawaid, «Efficient Advanced Encryption Standard for Securing Cognitive Radio Networks,» *Mehran University Research Journal of Engineering & Technology*, vol. 37, nº 4, pp. 645-654, 2019.

[22] M. Stamp, Information security: principles and practice, Hoboken, New Jersey: Wiley, 2006.

[23] A. J. Menezes, Handbook of applied cryptography, Boca Raton: CRC Press, 2001.

[24] GNU Radio, «GNU Radio,» [En línea]. Available: https://wiki.gnuradio.org/index.php/Main_Page. [Último acceso: 14 09 2019].

[25] Federal Information Processing Standards Publication 197, «Announcing the Advanced Encryption Standard (AES),» National Institute of Standards and Technology (NIST), noviembre 26,2001.

[26] National Institute of Standards and Technology, «Recommendation for Block Cipher Modes of Operation,» *Computer Security*, nº 800-3a, p. 66, 2001.

[27] H. Lipmaa y P. Rogaway, «Comments to NIST concerning AES Modes of Operations: CTR-Mode Encryption.,» 2000.

[28] Real Academia Española, [En línea]. Available: http://dle.rae.es/srv/search?m=30&w=caos. [Último acceso: 20 Noviembre 2018].

[29] H. G. Schuster y W. Just, de *Deterministic Chaos: An Introduction*, Weinheim, WILEY-VCH, 2005, p. 22.

[30] D. Yang y et al., «On the efficiency of chaos optimization algorithms for global optimization,» *Chaos Solitons Fractals*, vol. 34, nº 4, p. 1366, 2007.

[31] S. Moser y P.-N. Chen, «Entropy and Shannon's Source Coding Theorem,» de *A Student's Guide to Coding and Information Theory*, New York,, Cambridge University Press, 2012, pp. 81-96.

[32] D. Tsai y Y. Lee, «Information entropy measure for evaluation of image quality,» *Journal of Digital Imaging*, vol. 3, pp. 338-347, 2008.

[33] Python Sofware Foundation, «Python Documentation,» [En línea]. Available: https://docs.python.org/2/library/time.html. [Último acceso: Enero 2019].

[34] National Institute of Standards an Technology., «A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, Special Publication 800-22 Revision 1a,» Technology Administration U.S Department of Commerce, 2010.

[35] GNU-Radio, «Wiki.gnuradio,» [En línea]. Available: https://wiki.gnuradio.org/index.php/GNU_Radio_3.8_OOT_Module_Porting_Guide#Python_Blocks. [Último acceso: 01 10 2019].

[36] M. Gorski y S. Lucks, New Related-Key Boomerang Attacks on AES, Springer-Verlag Berlin Heidelberg : INDOCRYPT, 2008.

[37] R. A. Española, «http://dle.rae.es/srv/search?m=30&w=caos,» [En línea]. [Último acceso: 20 noviembre 2018].

[38] H.-G. Schuster y W. Just, de *Deterministic Chaos: An Introduction*, WILEY-VCH Verlag GmbH & Co. KGaA, 2005, p. 2.

[39] X. Huimin y K. Yilan, «Digital image correlation technique,» *Optics and Lasers in Engineering*, vol. 65, pp. 1-2, 2015.

[40] P. L'Ecuyyr y R. Simard , «TestU01: A C Library for Empirical Testing of Random Number Generators,» *ACM Transactions on Mathematical Software*, vol. 33, nº 4, pp. 1-40, 2007.

[41] Federal Information Processing Standards, «Announcing the Advanced Encryption Standard (AES),» National Institute of Standards and Technology (NIST), 2001.

ANEXO A. Código AES-Caos en Python

```
1   #-------------------------------------------------------------------------
2   # Nombre:    AES-Caos
3   # Autor:       Cuevas Papalotzin H. Cristina
4   # Created:    20/11/2018
5   #-------------------------------------------------------------------------
6
7   import cv2
8   import copy
9   import math
10  import random
11  import numpy as np
12  from time import time
13
14
15  """Funciones para el generador de caos"""
16  def SboxDinamica(semilla):
17      m_colision=np.zeros((16,16))
18      sbox=[]
19      aux=[]
20      aux2=[]
21      for i in range(256):
22          aux.append(i)
23      random.seed(semilla)
24      tam = len(aux)
25      while(tam > 0):
26          for j in range(16):
27              aux2 = []
28              for k in range(16):
29                  x = random.randint(0,(tam - 1))
30                  aux2.append(aux[x])
31                  aux.remove(aux[x])
32                      tam = tam - 1
33              sbox.append(aux2)
34      return sbox
35
36
37  def Mapa_Caotico(xi,miu):
38      xii = xi
39      if (xi < 0):
40          xi = xi * ( -1)
41      if ((xi > 1) or (xi<0.00001)):
42          xi = random.random()
43      if(0 < xi <= miu):
44          xi = xi/miu
45      elif(miu < xi < 1):
46          xi = (xi - 1) / (miu - 1)
```

```
    if ((xi > 1) or (xi<0.00001) or (xi == xii)):
        xi = random.random()
    return xi


def Ad_mod(fi):
    for i in range(len(fi)):
        if((i != 0)):
            fi[i] = fi[i] ^ fi[i - 1]
    return fi


def SubByteC(Sbox,val):
    Byte = val % 256
    columna = Byte >> 4
    fila = Byte & 0x0F
    n_Byte = Sbox[columna][fila] #sustutucion
    return n_Byte


def rot(fi):
    x = 0.000191291562
    miu = 0.000123
    n = 8
    y_aux = []
    for i in range(len(fi)):
        x = Mapa_Caotico(x, miu)
        miu = miu + 0.0005
        x = (x * (2 ** 16))
        if (int (x)% 2 == 0):
            aux1 = fi[i] << i % n
            aux2 = fi[i] >> n-(i % n)
            fi[i] = aux1 | aux2
            fi[i] = fi[i] | int(x)
            fi[i] = fi[i] & 0x00FF
        else:
            aux1 = fi[i] >> i % n
            aux2 = fi[i] << n-(i % n)
            fi[i] = aux1 | aux2
            fi[i] = fi[i]^int(x)
            fi[i] = fi[i] & 0x00FF
    return fi


def caos(x,miu,sbox):
    x_i = []
    fi=[]
    i = 0
    valores = 16
    while i < valores:
```

```
97          de = (x * (2 ** 16))
98          fi.append(int(de))
99          y = Mapa_Caotico(x,miu)
100         x = y
101         miu = miu+0.0025
102         i += 1
103       while i < valores:
104         fi = Ad_mod(fi)
105       for i in range(len(fi)):
106         fi[i] = SubByteC(sbox,fi[i])
107   return fi,x
108
109
110   """Funciones para AES"""
111
112
113   def SubBytes(Byte): #Sustitucion de bytes
114
115       [0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5, 0x30, 0x01, 0x67, 0x2B, 0xFE, 0xD7, 0xAB,
      0x76],
116       [0xCA, 0x82, 0xC9, 0x7D, 0xFA, 0x59, 0x47, 0xF0, 0xAD, 0xD4, 0xA2, 0xAF, 0x9C, 0xA4, 0x72,
      0xC0],
117       [0xB7, 0xFD, 0x93, 0x26, 0x36, 0x3F, 0xF7, 0xCC, 0x34, 0xA5, 0xE5, 0xF1, 0x71, 0xD8, 0x31,
      0x15],
118       [0x04, 0xC7, 0x23, 0xC3, 0x18, 0x96, 0x05, 0x9A, 0x07, 0x12, 0x80, 0xE2, 0xEB, 0x27, 0xB2,
      0x75],
119       [0x09, 0x83, 0x2C, 0x1A, 0x1B, 0x6E, 0x5A, 0xA0, 0x52, 0x3B, 0xD6, 0xB3, 0x29, 0xE3, 0x2F,
      0x84],
120       [0x53, 0xD1, 0x00, 0xED, 0x20, 0xFC, 0xB1, 0x5B, 0x6A, 0xCB, 0xBE, 0x39, 0x4A, 0x4C, 0x58,
      0xCF],
121       [0xD0, 0xEF, 0xAA, 0xFB, 0x43, 0x4D, 0x33, 0x85, 0x45, 0xF9, 0x02, 0x7F, 0x50, 0x3C, 0x9F,
      0xA8],
122       [0x51, 0xA3, 0x40, 0x8F, 0x92, 0x9D, 0x38, 0xF5, 0xBC, 0xB6, 0xDA, 0x21, 0x10, 0xFF, 0xF3,
      0xD2],
123        [0xCD, 0x0C, 0x13, 0xEC, 0x5F, 0x97, 0x44, 0x17, 0xC4, 0xA7, 0x7E, 0x3D, 0x64, 0x5D, 0x19,
      0x73],
124       [0x60, 0x81, 0x4F, 0xDC, 0x22, 0x2A, 0x90, 0x88, 0x46, 0xEE, 0xB8, 0x14, 0xDE, 0x5E, 0x0B,
      0xDB],
125       [0xE0, 0x32, 0x3A, 0x0A, 0x49, 0x06, 0x24, 0x5C, 0xC2, 0xD3, 0xAC, 0x62, 0x91, 0x95, 0xE4,
      0x79],
126       [0xE7, 0xC8, 0x37, 0x6D, 0x8D, 0xD5, 0x4E, 0xA9, 0x6C, 0x56, 0xF4, 0xEA, 0x65, 0x7A, 0xAE,
      0x08],
127       [0xBA, 0x78, 0x25, 0x2E, 0x1C, 0xA6, 0xB4, 0xC6, 0xE8, 0xDD, 0x74, 0x1F, 0x4B, 0xBD, 0x8B,
      0x8A],
128       [0x70, 0x3E, 0xB5, 0x66, 0x48, 0x03, 0xF6, 0x0E, 0x61, 0x35, 0x57, 0xB9, 0x86, 0xC1, 0x1D,
      0x9E],
129       [0xE1, 0xF8, 0x98, 0x11, 0x69, 0xD9, 0x8E, 0x94, 0x9B, 0x1E, 0x87, 0xE9, 0xCE, 0x55, 0x28,
      0xDF],
130       [0x8C, 0xA1, 0x89, 0x0D, 0xBF, 0xE6, 0x42, 0x68, 0x41, 0x99, 0x2D, 0x0F, 0xB0, 0x54, 0xBB,
      0x16]
```

```
131 ]
132     columna = Byte >> 4
133     fila = Byte & 0x0F
134     n_Byte = Sbox[columna][fila] #sustutucion
135     return n_Byte
136
137
138 def ShiftRows(bloque):#Rotar de filas
139     for i in range (4):
140         if (i == 1):
141             aux = bloque[i][0]
142             bloque[i].pop(0)
143             bloque[i].append(aux)
144         elif(i == 2):
145             for j in range(2):
146                 aux = bloque[i][0]
147                 bloque[i].pop(0)
148                 bloque[i].append(aux)
149         elif(i == 3):
150             aux2 = []
151             for j in range(3):
152                 aux = bloque[i][0]
153                 bloque[i].pop(0)
154                 bloque[i].append(aux)
155     return bloque
156
157
158 def galoisMult(a, b): #Multiplicacion en GF2^8
159     p = 0
160     hiBitSet = 0
161     for i in range(8):
162         if b & 1 == 1:
163             p ^= a
164         hiBitSet = a & 0x80
165         a <<= 1
166         if hiBitSet == 0x80:
167             a ^= 0x1b
168         b >>= 1
169     return p % 256
170
171
172 def Mix_column(bloque):
173     bloque = transpuesta(bloque)
174     blque_nuevo = []
175     for i in range(4):
176         temp = bloque[i]
177         column = []
178         column = copy.deepcopy(temp)
179         column[0] = galoisMult(temp[0],2) ^ galoisMult(temp[3],1) ^ \
180                 galoisMult(temp[2],1) ^ galoisMult(temp[1],3)
```

```
        column[1] = galoisMult(temp[1],2) ^ galoisMult(temp[0],1) ^ \
                galoisMult(temp[3],1) ^ galoisMult(temp[2],3)
        column[2] = galoisMult(temp[2],2) ^ galoisMult(temp[1],1) ^ \
                galoisMult(temp[0],1) ^ galoisMult(temp[3],3)
        column[3] = galoisMult(temp[3],2) ^ galoisMult(temp[2],1) ^ \
                galoisMult(temp[1],1) ^ galoisMult(temp[0],3)
        blque_nuevo.append(column)
    blque_nuevo = transpuesta(blque_nuevo)
    return blque_nuevo


def GeneraBloque(vector): #Acomoda el vector en una matriz para el procesamiento
    m_bloque = []
    temporal1 = []
    temporal2 = []
    temporal3 = []
    temporal4 = []
    for i in range(0,16,4):
        temporal1.append(vector[i])
        temporal2.append(vector[i + 1])
        temporal3.append(vector[i + 2])
        temporal4.append(vector[i + 3])
    m_bloque.append(temporal1)
    m_bloque.append(temporal2)
    m_bloque.append(temporal3)
    m_bloque.append(temporal4)
    return (m_bloque)


"""Funciones para generar claves"""


def RCon(Nr):
    Rcon = [[0x01 ,0 ,0 ,0],[0x02, 0, 0, 0],[0x04, 0, 0, 0],[0x08, 0, 0, 0],[0x10, 0, 0, 0],[0x20, 0, 0,
0],[0x40, 0, 0, 0],[0x80, 0, 0, 0], [0x1b, 0, 0, 0], [0x36, 0, 0, 0]]
    return Rcon[Nr - 1]


def Rotword(palabra):
    palabra_r = copy.deepcopy(palabra)
    aux = palabra_r[0]
    palabra_r.pop(0)
    palabra_r.append(aux)
    return palabra_r


def primera_expansion(palabra,Nr,palabra_i):
    s_palabra = [] #Siguiente palabra
    aux2 = []
    aux1 = Rotword(palabra)
```

```
    for i in range(4):  #ciclo de 4 para llamar a Subbyte ya que recibe byte por byte
        sub = SubBytes(aux1[i])
        aux2.append(sub)
    aux3 = RCon(Nr)
    for i in range(4):
        temp = aux2[i] ^ aux3[i] ^ palabra_i[i]
        s_palabra.append(temp)
    return (s_palabra)


def KeyExpansion(palabra,palabra_i):
    s_palabra = []#Siguiente plabra
    temp = []
    aux0 = palabra#La palabra que se va a modificar (la cuarta palabra)
    for i in range(4):
        temp = palabra[i] ^ palabra_i[i]
        s_palabra.append(temp)
    return(s_palabra)


def GeneraBloque_clave(bloque): #Acomoda el vector clave en una matriz
     bloque_m = [] # matriz clave
     for i in range(0, 16, 4):
        temporal = []
        for j in range(i, i + 4,1):
            temporal.append(bloque[j])
        bloque_m.append(temporal)
     return bloque_m


def Genera_clave(Nr,m_clave):
    for i in range(4):
        clave_2 = transpuesta(m_clave)
        if(i == 0):
            n_palabra0 = primera_expansion(m_clave[3], Nr,m_clave[0])
            m_clave.insert(i, n_palabra0)
            m_clave.pop(i + 1)
        else:
            n_palabra0 = KeyExpansion(n_palabra0, m_clave[i])
            m_clave.insert(i, n_palabra0)
            m_clave.pop(i + 1)
        clave_2 = transpuesta(m_clave)
    return clave_2


def transpuesta(matriz):
    M = [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0],[0, 0, 0 ,0]]
    for i in range(4):
        for j in range(4):
            M[j][i] = matriz[i][j]
```

```
280     return M
281
282
283 def imprime(matriz):
284     for i in range(4):
285         for j in range(4):
286             print hex(matriz[i][j])
287
288
289 def incremento(VI, pos):
290     aux = VI[pos]#cuando es matriz [3][3]
291     aux = (aux + 1) % 255
292     VI[pos] = aux
293     return VI
294
295
296 def vectoriza(mat):
297     vector = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
298     k = 0
299     for i in range(4):
300         for j in range(4):
301             vector[k] = mat[i][j]
302             k = k + 1
303     return vector
304
305
306 def main():
307     x1 = 0.024439156
308     x2 = 0.0004356
309     x3 = 0.000045674
310     xc = 0.00976433
311     miu = 0.00123
312     sbox = SboxDinamica(xc)
313     aux = caos(xc,miu,sbox)
314     clave= aux[0]
315     clave = GeneraBloque(clave)
316     tiempos=[]
317     repeticion=0
318     max=0
319     min=100000
320     for a in range (1,101):
321         nombre = "Img" + str(a) +"_512.bmp"
322         tiempo_inicial = time()
323         Nr = 0
324         l = 0
325         m = 0
326         img = cv2.imread(nombre,cv2.IMREAD_COLOR )
327         (h, w) = img.shape[:2]
328         #ajusta img
329         y = h/16
```

```
330         z = w/16
331         h = int(y)*16
332         w = int(z)*16
333         img = cv2.resize(img, (w,h))
334         img2 = np.zeros((img.shape), np.uint8)
335         entradaR = []
336         entradaG = []
337         entradaB = []
338         for i in range(16):
339            entradaR.append(1)
340            entradaG.append(1)
341            entradaB.append(1)
342         for j in range(h):
343            miu = 0.00123
344            sbox = SboxDinamica(x1)
345            for k in range(w):
346               entradaR[k % 16] = (img.item(j, k, 0))#R
347               entradaG[k % 16] = (img.item(j, k, 1))#G
348               entradaB[k % 16] = (img.item(j, k, 2))#B
349               if(k%16 == 15):
350                  ret1 = caos(x1,miu,sbox)
351                  entrada1 = ret1[0]
352                  x1=ret1[1]
353                  ret2 = caos(x2,miu,sbox)
354                  entrada2 = ret2[0]
355                  x2=ret2[1]
356                  ret3 = caos(x3,miu,sbox)
357                  entrada3 = ret3[0]
358                  x3=ret3[1]
359                  miu = miu+0.000005
360                  #AES
361                  m_bloque1 = GeneraBloque(entrada1)
362                  m_bloque2 = GeneraBloque(entrada2)
363                  m_bloque3 = GeneraBloque(entrada3)
364                  #Ronda inicial
365                  for i in range(4):
366                     for jj in range (4):
367                        m_bloque1[i][jj] = m_bloque1[i][jj]^clave[i][jj]
368                        m_bloque2[i][jj] = m_bloque2[i][jj]^clave[i][jj]
369                        m_bloque3[i][jj] = m_bloque3[i][jj]^clave[i][jj]
370                  clave_2 = transpuesta(clave)
371                  #Rondas estandar
372                  Nr = random.randrange(9)
373                  clave_2 =  Genera_clave(Nr,clave_2)
374                  for i in range(4):
375                     for jj in range(4):
376                        m_bloque1[i][jj] = SubBytes(m_bloque1[i][jj])
377                        m_bloque2[i][jj] = SubBytes(m_bloque2[i][jj])
378                        m_bloque3[i][jj] = SubBytes(m_bloque3[i][jj])
379                  m_bloque1 = ShiftRows(m_bloque1)
```

```
            m_bloque2 = ShiftRows(m_bloque2)
            m_bloque3 = ShiftRows(m_bloque3)
            m_bloque1 = Mix_column(m_bloque1)
            m_bloque2 = Mix_column(m_bloque2)
            m_bloque3 = Mix_column(m_bloque3)
            for i in range(4):
               for jj in range(4):
                  m_bloque1[i][jj] = m_bloque1[i][jj]^clave_2[i][jj]
                  m_bloque2[i][jj] = m_bloque2[i][jj]^clave_2[i][jj]
                  m_bloque3[i][jj] = m_bloque3[i][jj]^clave_2[i][jj]
            clave_2 = transpuesta(clave_2)
            #Ronda Final
            clave_2 =  Genera_clave(Nr, clave_2)
            for i in range(4):
               for jj in range(4):
                  m_bloque1[i][jj] = SubBytes(m_bloque1[i][jj])
                  m_bloque2[i][jj] = SubBytes(m_bloque2[i][jj])
                  m_bloque3[i][jj] = SubBytes(m_bloque3[i][jj])
            m_bloque1 = ShiftRows(m_bloque1)
            m_bloque2 = ShiftRows(m_bloque2)
            m_bloque3 = ShiftRows(m_bloque3)
            for i in range(4):
                  for jj in range(4):
                     m_bloque1[i][jj] = m_bloque1[i][jj] ^ clave_2[i][jj]
                     m_bloque2[i][jj] = m_bloque2[i][jj] ^ clave_2[i][jj]
                     m_bloque3[i][jj] = m_bloque3[i][jj] ^ clave_2[i][jj]
            k = 0
            for i in range(4):
               for jj in range(4):
                  #print entradaR[k]
                  entradaR[k] = entradaR[k] ^ m_bloque1[i][jj]
                  entradaG[k] = entradaG[k] ^ m_bloque2[i][jj]
                  entradaB[k] = entradaB[k] ^ m_bloque3[i][jj]
                  k = k + 1
            for i in range(len(entradaR)):
               if(l == h):
                  break
               img2.itemset((l,m,0),entradaR[i])
               img2.itemset((l,m,1),entradaG[i])
               img2.itemset((l,m,2),entradaB[i])
               m = m + 1
               if(m == w):
                  #w
                  l = l + 1
                  m = 0
      nombre2 = "Img" + str(a) + "_512_caos.bmp"
      cv2.imwrite(nombre2,img2)#Guarda una imagen
      tiempo_final = time()
      tiempo_ejecucion = tiempo_final - tiempo_inicial
      print nombre
```

```
430         print 'El tiempo de ejecucion fue:',tiempo_ejecucion
431         repeticion=repeticion+1
432     pass
433
434
435 if __name__ == '__main__':
436     main()
```

ANEXO B. Código de AES-Caos en Python para GNU Radio

```
1   #-------------------------------------------------------------------------
2   # Nombre:     AES-Caos para GNU Radio
3   # Autor:      Cuevas Papalotzin H. Cristina
4   # Created:    05/10/2019
5   #-------------------------------------------------------------------------
6
7
8   #!/usr/bin/env python
9   # -*- coding: utf-8 -*-
10
11
12  import copy
13  import math
14  import random
15  import numpy as np
16  from gnuradio import gr
17
18
19  class AES_Caos_py(gr.sync_block):
20      """docstring for block AES_Caos_py"""
21      def __init__(self):
22          gr.sync_block.__init__(self,
23              name="AES_Caos_py",
24              in_sig=[np.float32],
25              out_sig=None)
26      self.x1 = 0.024439156
27      self.x2 = 0.0004356
28      self.x3 = 0.000045674
28      self.xc = 0.00976433
30      self.miu = 0.00123
31      self.sbox = self.SboxDinamica(self.xc)
32      self.aux = self.caos(self.xc,self.miu,self.sbox)
33      self.clave= self.aux[0]
34      self.clave = self.GeneraBloque(self.clave)
35
36
37      def work(self, input_items, output_items):
38          in0 = input_items[0]
39          # <+signal processing here+>
40          nombre = "/home/ubuntu/Pictures/Entrada.jpeg" #Entrada
41          nombre2 = "/home/ubuntu/Pictures/Cifrado.jpeg" # Salida
42          Nr = 0
43          l = 0
44          m = 0
45          archivo2 = open(nombre2,'wb')
46          archivo = open(nombre,'rb')
```

```
47                  self.miu = 0.00123
48                  self.sbox = self.SboxDinamica(self.x1)
49                 img = archivo.read()
50                 archivo2 = open(nombre2,'wb')
51                 for i in range(16):
52                         entradaR.append(1)
53                 for k in range(len(img)):
54                         if (k%160 == 0):
55                                 self.miu = 0.00123
56                                 self.sbox = self.SboxDinamica(self.x1)
57                          entradaR[k % 16] = (ord(img[k]))
58                          if(k%16 == 15):
59                                  ret1 = self.caos(self.x1,self.miu,self.sbox)
60                                  entrada1 = ret1[0]
61                                  self.x1=ret1[1]
62                                  self.miu = self.miu+0.000005
63                                  #AES
63                                  m_bloque1 = self.GeneraBloque(entrada1)
65                                 #Ronda inicial
66                                 for i in range(4):
67                                         for jj in range (4):
68                                                 m_bloque1[i][jj] = m_bloque1[i][jj]^self.clave[i][jj]
69                                 self.clave_2 = self.transpuesta(self.clave)
70                                 #Rondas estandar
71                                 Nr = random.randrange(9)
72                                 self.clave_2 =  self.Genera_clave(Nr,self.clave_2)
73                                 for i in range(4):
74                                         for jj in range(4):
75                                                 m_bloque1[i][jj] = self.SubBytes(m_bloque1[i][jj])
76                                  m_bloque1 = self.ShiftRows(m_bloque1)
77                                  m_bloque1 = self.Mix_column(m_bloque1)
78                                  for i in range(4):
79                                         for jj in range(4):
80                                                 m_bloque1[i][jj] = m_bloque1[i][jj]^self.clave_2[i][jj]
81                                  self.clave_2 = self.transpuesta(self.clave_2)
82                                  #Ronda Final
83                                  self.clave_2 =  self.Genera_clave(Nr, self.clave_2)
84                                  for i in range(4):
85                                         for jj in range(4):
86                                                 m_bloque1[i][jj] = self.SubBytes(m_bloque1[i][jj])
87                                  m_bloque1 = self.ShiftRows(m_bloque1)
88                                  for i in range(4):
89                                         for jj in range(4):
90                                                 m_bloque1[i][jj] = m_bloque1[i][jj] ^ self.clave_2[i][jj]
91                                    k = 0
92                                    for i in range(4):
93                                           for jj in range(4):
94                                                  entradaR[k] = entradaR[k] ^ m_bloque1[i][jj]
95                                                                 k = k + 1
96                                    for i in range (16):
```

```
97                              archivo2.write(chr(entradaR[i])
98                  archivo.close()
99                  archivo2.close()
100             return len(output_items[0])
101
102
103         def SboxDinamica(self,semilla):
104             m_colision = np.zeros((16,16))
105             sbox=[]
106             aux=[]
107             aux2=[]
108             for i in range(256):
109                 aux.append(i)
110             random.seed(semilla)
111             tam = len(aux)
112             while(tam > 0):
113                 for j in range(16):
114                     aux2 = []
115                     for k in range(16):
116                         x = random.randint(0,(tam - 1))
117                         aux2.append(aux[x])
118                         aux.remove(aux[x])
119                         tam = tam - 1
120                     sbox.append(aux2)
121             return sbox
122
123
124         def caos(self,x,miu,sbox):
125             x_i = []
126             fi=[]
127             i = 0
128             valores = 16
129             while i < valores:
130                 de = (x * (2 ** 16))
131                 fi.append(int(de))
132                 y  = self.Mapa_Caotico(x,miu)
133                 x = y
134                 miu = miu+0.0025
135                 i+= 1
136                 fi = self.Ad_mod(fi)
137                 for i in range(len(fi)):
138                     fi[i] = self.SubByteC(sbox,fi[i])
139                     fi = self.rot(fi)
140             return fi,x
141
142
143         def rot(self, fi):
144             x = 0.000191291562
145             miu = 0.000123
146             n = 8
```

```
147             y_aux = []
148             for i in range(len(fi)):
149                 x = self.Mapa_Caotico(x, miu)
150                 miu = miu + 0.0005
151                 x = (x * (2 ** 16))
152             if(int (x)% 2 == 0):
153                 aux1 = fi[i] << i % n
154                 aux2 = fi[i] >> n-(i % n)
155                 fi[i] = aux1 | aux2
156                 fi[i] = fi[i] | int(x)
157                 fi[i] = fi[i] & 0x00FF
158             else:
158                 aux1 = fi[i] >> i % n
160                 aux2 = fi[i] << n-(i % n)
161                 fi[i] = aux1 | aux2
162                 fi[i] = fi[i]^int(x)
163                  fi[i] = fi[i] & 0x00FF
164             return fi
165
166
167         def SubByteC(self,Sbox,val):
168             Byte = val % 256
169             columna = Byte >> 4
170             fila = Byte & 0x0F
171             n_Byte = Sbox[columna][fila]
172             return n_Byte
173
174
175         def Ad_mod(self,fi):
176             for i in range(len(fi)):
177                 if((i != 0)):
178                     fi[i] = fi[i] ^ fi[i - 1]
179         return fi
180
181
182         def Mapa_Caotico(self,xi,miu):
183             xii = xi
184             if (xi < 0):
185                 xi = xi * ( -1)
186             if ((xi > 1) or (xi<0.00001)):
187                 xi = random.random()
188             if(0 < xi <= miu):
189                 xi = xi/miu
190             elif(miu < xi < 1):
191                  xi = (xi - 1) / (miu - 1)
192             if ((xi > 1) or (xi<0.00001) or (xi == xii)):
193                 xi = random.random()
194         return xi
195
196
```

```
197     def GeneraBloque(self,vector):
198         m_bloque = []
199         temporal1 = []
200         temporal2 = []
201         temporal3 = []
202         temporal4 = []
203         for i in range (0,16,4):
204             temporal1.append(vector[i])
205             temporal2.append(vector[i + 1])
206             temporal3.append(vector[i + 2])
207             temporal4.append(vector[i + 3])
208         m_bloque.append(temporal1)
209         m_bloque.append(temporal2)
210         m_bloque.append(temporal3)
211         m_bloque.append(temporal4)
212         return (m_bloque)
213
214
215     def transpuesta(self,matriz):
216         M = [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0],[0, 0, 0 ,0]]
217         for i in range(4):
218             for j in range(4):
219             M[j][i] = matriz[i][j]
220     return M
221
222
223     def Genera_clave(self,Nr,m_clave):
224         for i in range(4):
225             clave_2 = self.transpuesta(m_clave)
226             if(i == 0):
227                 n_palabra0 = self.primera_expansion(m_clave[3], Nr,m_clave[0])
228                 m_clave.insert(i, n_palabra0)
229                 m_clave.pop(i + 1)
230             else:
231                 n_palabra0 = self.KeyExpansion(n_palabra0, m_clave[i])
232                 m_clave.insert(i, n_palabra0)
233                 m_clave.pop(i + 1)
234             clave_2 = self.transpuesta(m_clave)
235     return clave_2
236
237
238     def KeyExpansion(self,palabra,palabra_i):
239         s_palabra = []#Siguiente plabra
240         temp = []
241         aux0 = palabra#La palabra que se va a modificar (la cuarta palabra)
242         for i in range(4):
243             temp = palabra[i] ^ palabra_i[i]
244             s_palabra.append(temp)
245         return(s_palabra)
246
```

```
247
248         def primera_expansion(self,palabra,Nr,palabra_i):
249             s_palabra = [] #Siguiente palabra
250             aux2 = []
251             aux1 = self.Rotword(palabra)
252             for i in range(4):
253                 sub =self.SubBytes(aux1[i])
254                 aux2.append(sub)
255             aux3 = self.RCon(Nr)
256             for i in range(4):
257                 temp = aux2[i] ^ aux3[i] ^ palabra_i[i]
258                 s_palabra.append(temp)
259             return (s_palabra)
260
261
262         def Rotword(self,palabra):
263             palabra_r = copy.deepcopy(palabra)
264             aux = palabra_r[0]
265             palabra_r.pop(0)
266             palabra_r.append(aux)
267             return palabra_r
268
269
270         def SubBytes(self,Byte): #Sustitucion de bytes
271             Sbox = [
272                 [0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5, 0x30, 0x01, 0x67, 0x2B, 0xFE, 0xD7,
    0xAB, 0x76],
273                 [0xCA, 0x82, 0xC9, 0x7D, 0xFA, 0x59, 0x47, 0xF0, 0xAD, 0xD4, 0xA2, 0xAF, 0x9C,
    0xA4, 0x72, 0xC0],
274                 [0xB7, 0xFD, 0x93, 0x26, 0x36, 0x3F, 0xF7, 0xCC, 0x34, 0xA5, 0xE5, 0xF1, 0x71, 0xD8,
    0x31, 0x15],
275                 [0x04, 0xC7, 0x23, 0xC3, 0x18, 0x96, 0x05, 0x9A, 0x07, 0x12, 0x80, 0xE2, 0xEB, 0x27,
    0xB2, 0x75],
276                 [0x09, 0x83, 0x2C, 0x1A, 0x1B, 0x6E, 0x5A, 0xA0, 0x52, 0x3B, 0xD6, 0xB3, 0x29, 0xE3,
    0x2F, 0x84],
277                 [0x53, 0xD1, 0x00, 0xED, 0x20, 0xFC, 0xB1, 0x5B, 0x6A, 0xCB, 0xBE, 0x39, 0x4A, 0x4C,
    0x58, 0xCF],
278                 [0xD0, 0xEF, 0xAA, 0xFB, 0x43, 0x4D, 0x33, 0x85, 0x45, 0xF9, 0x02, 0x7F, 0x50, 0x3C,
    0x9F, 0xA8],
279                 [0x51, 0xA3, 0x40, 0x8F, 0x92, 0x9D, 0x38, 0xF5, 0xBC, 0xB6, 0xDA, 0x21, 0x10, 0xFF,
    0xF3, 0xD2],
280                 [0xCD, 0x0C, 0x13, 0xEC, 0x5F, 0x97, 0x44, 0x17, 0xC4, 0xA7, 0x7E, 0x3D, 0x64, 0x5D,
    0x19, 0x73],
281                 [0x60, 0x81, 0x4F, 0xDC, 0x22, 0x2A, 0x90, 0x88, 0x46, 0xEE, 0xB8, 0x14, 0xDE, 0x5E,
    0x0B, 0xDB],
282                 [0xE0, 0x32, 0x3A, 0x0A, 0x49, 0x06, 0x24, 0x5C, 0xC2, 0xD3, 0xAC, 0x62, 0x91, 0x95,
    0xE4, 0x79],
283                 [0xE7, 0xC8, 0x37, 0x6D, 0x8D, 0xD5, 0x4E, 0xA9, 0x6C, 0x56, 0xF4, 0xEA, 0x65, 0x7A,
    0xAE, 0x08],
284                 [0xBA, 0x78, 0x25, 0x2E, 0x1C, 0xA6, 0xB4, 0xC6, 0xE8, 0xDD, 0x74, 0x1F, 0x4B, 0xBD,
```

```
0x8B, 0x8A],
285                 [0x70, 0x3E, 0xB5, 0x66, 0x48, 0x03, 0xF6, 0x0E, 0x61, 0x35, 0x57, 0xB9, 0x86, 0xC1,
0x1D, 0x9E],
286                 [0xE1, 0xF8, 0x98, 0x11, 0x69, 0xD9, 0x8E, 0x94, 0x9B, 0x1E, 0x87, 0xE9, 0xCE, 0x55,
0x28, 0xDF],
287                 [0x8C, 0xA1, 0x89, 0x0D, 0xBF, 0xE6, 0x42, 0x68, 0x41, 0x99, 0x2D, 0x0F, 0xB0, 0x54,
0xBB, 0x16]
288              ]
289             columna = Byte >> 4
290             fila = Byte & 0x0F
291             n_Byte = Sbox[columna][fila] #sustutucion
292             return n_Byte
293
294     def RCon(self,Nr):
295         Rcon = [[0x01 ,0 ,0 ,0],[0x02, 0, 0, 0],[0x04, 0, 0, 0],[0x08, 0, 0, 0],[0x10, 0, 0, 0],[0x20, 0, 0,
0],[0x40, 0, 0, 0],[0x80, 0, 0, 0], [0x1b, 0, 0, 0], [0x36, 0, 0, 0]]
296             return Rcon[Nr - 1]
297
298
299     def ShiftRows(self,bloque):#Rotar de filas
300         for i in range (4):
301             if (i == 1):
302                 aux = bloque[i][0]
303                 bloque[i].pop(0)
304                  bloque[i].append(aux)
305             elif(i == 2):
306                 for j in range(2):
307                     aux = bloque[i][0]
308                     bloque[i].pop(0)
309                     bloque[i].append(aux)
310             elif(i == 3):
311                 aux2 = []
312                 for j in range(3):
313                     aux = bloque[i][0]
314                     bloque[i].pop(0)
315                     bloque[i].append(aux)
316         return bloque
317
318
319     def Mix_column(self,bloque):
320         bloque = self.transpuesta(bloque)
321         blque_nuevo = []
322         for i in range(4):
323             temp = bloque[i]
324              column = []
325              column = copy.deepcopy(temp)
326              column[0] = self.galoisMult(temp[0],2) ^ self.galoisMult(temp[3],1) ^ \
327                          self.galoisMult(temp[2],1) ^ self.galoisMult(temp[1],3)
328              column[1] = self.galoisMult(temp[1],2) ^ self.galoisMult(temp[0],1) ^ \
329                          self.galoisMult(temp[3],1) ^ self.galoisMult(temp[2],3)
```

```
330                 column[2] = self.galoisMult(temp[2],2) ^ self.galoisMult(temp[1],1) ^ \
331                             self.galoisMult(temp[0],1) ^ self.galoisMult(temp[3],3)
332                 column[3] = self.galoisMult(temp[3],2) ^ self.galoisMult(temp[2],1) ^ \
333                             self.galoisMult(temp[1],1) ^ self.galoisMult(temp[0],3)
334                 blque_nuevo.append(column)
335         blque_nuevo = self.transpuesta(blque_nuevo)
336         return blque_nuevo
337
338
339     def galoisMult(self,a, b): #Multiplicacion en GF2^8
340         p = 0
341         hiBitSet = 0
342         for i in range(8):
343             if b & 1 == 1:
344             p ^= a
345             hiBitSet = a & 0x80
346             a <<= 1
347             if hiBitSet == 0x80:
348                 a ^= 0x1b
348                 b >>= 1
350         return p % 256
```

APENDICE C. Código del archivo XML para GNU Radio

```
1  <?xml version="1.0"?>
2  <block>
3    <name>AES_Caos_py</name>
4    <key>AES_Caos_AES_Caos_py</key>
5    <category>[AES_Caos]</category>
6    <import>import AES_Caos</import>
7    <make>AES_Caos.AES_Caos_py()</make>
8    <!-- Make one 'param' node for every Parameter you want settable from the GUI.
9         Sub-nodes:
10        * name
11        * key (makes the value accessible as $keyname, e.g. in the make node)
12        * type -->
13     <!-- Make one 'sink' node per input. Sub-nodes:
14        * name (an identifier for the GUI)
15        * type
16        * vlen
17        * optional (set to 1 for optional inputs) -->
18   <sink>
19     <name>in</name>
20     <type>float</type>
21   </sink></block>
```

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA

ACTA DE EXAMEN DE GRADO

No. 00080
Matrícula: 2173802172

Estudio y adaptación del algoritmo criptográfico AES para radios cognitivos.

HORTENSIA CRISTINA CUEVAS PAPALOTZIN

ALUMNA

REVISÓ

MTRA. ROSALIA SERRANO DE LA PAZ
DIRECTORA DE SISTEMAS ESCOLARES

En la Ciudad de México, se presentaron a las 12:30 horas del día 8 del mes de noviembre del año 2019 en la Unidad Iztapalapa de la Universidad Autónoma Metropolitana, los suscritos miembros del jurado:

DRA. MARIKO NAKANO MIYATAKE
DR. LEONARDO PALACIOS LUENGAS
DR. RICARDO MARCELIN JIMENEZ

Bajo la Presidencia de la primera y con carácter de Secretario el último, se reunieron para proceder al Examen de Grado cuya denominación aparece al margen, para la obtención del grado de:

MAESTRA EN CIENCIAS (CIENCIAS Y TECNOLOGIAS DE LA INFORMACION)

DE: HORTENSIA CRISTINA CUEVAS PAPALOTZIN

y de acuerdo con el artículo 78 fracción III del Reglamento de Estudios Superiores de la Universidad Autónoma Metropolitana, los miembros del jurado resolvieron:

APROBAR

Acto continuo, la presidenta del jurado comunicó a la interesada el resultado de la evaluación y, en caso aprobatorio, le fue tomada la protesta.

DIRECTOR DE LA DIVISIÓN DE CBI

DR. JESUS ALBERTO OCHOA TAPIA

PRESIDENTA

DRA. MARIKO NAKANO MIYATAKE

VOCAL

DR. LEONARDO PALACIOS LUENGAS

SECRETARIO

DR. RICARDO MARCELIN JIMENEZ