BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

CIFRADO SIMÉTRICO BASADO EN LA TEORÍA DEL CAOS PARA IMÁGENES DIGITALES

TESIS

QUE PARA OBTENER EL TÍTULO DE:
LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:
ALDO RAFAEL MATUS ANGULO

ASESOR:
MC. LUIS RENÉ MARCIAL CASTILLO

H. Puebla de Zaragoza a 26 de Noviembre del 2015

Dedicado a

mi padre, siempre te tendré presente a cada paso que dé, dejaste una parte de tí en mi alma, en mi forma de ser y en mi forma de pensar.

Agradecimientos

A todos esas personas incondicionales que han mejorado mi vida, a ellos agradezco muchos de los éxitos en mi vida, a mi novia, a mis amigos, a mi familia.

A los que me aconsejaron y apoyaron en las correcciones a lo largo de la realización de este trabajo, al profesor Marcial, a la profesora Marcela y a la doctora Sandoval, además de que fué un honor haber tomado clases con ustedes.

Índice general

Índice de figuras

Índice de cuadros

INTRODUCCIÓN

La criptografía no es algo nuevo, a lo largo de la historia se han utilizado métodos primitivos de ocultación de información, sin embargo su edad y orígen exactos son de alguna forma inciertos así como su definición exacta. Buscando definiciones de varios autores de criptografía moderna se encontró que Goldreich, Oded[1] transcribe el planteamiento del problema de la criptografía como:

> "El problema de proveer *comunicación secreta sobre medios inseguros*",

mientras que Stinson, Douglas[2] lo plantea de manera más explicita afirmando que:

> "El objetivo fundamental de la criptografía es permitir a dos personas, usualmente referidas como Alice y Bob, comunicarse sobre un canal inseguro de tal forma que un oponente, Oscar, no pueda entender lo que se está diciendo."

El diccionario de la Real Academia de la lengua Española por su parte es más concreta pero algo inexacta y define a la criptografía como "El arte de escribir con clave secreta o de un modo enigmático".

Otra manera de definir a la criptografía es mediante su etimología, la cual proviene del griego *kryptos* —ocultar— y *graphos* —escribir— y ésta nos brindaría la definición: *"Estudio de la escritura oculta"*, o algo similar [3][4].

Como se puede ver buscando en varias fuentes se encuentran referencias muy variadas y aunque anteriormente haya sido mencionada como un arte (se le veía así en la antigüedad), la criptografía moderna se considera una ciencia por su proximidad con la probabilidad, la teoría de números y la teoría de la complejidad computacional[4].

Establecido todo lo anterior, se puede dar una definición propia y decir que la criptografía es una ciencia o disciplina que se encarga de estudiar métodos

nuevos y robustos para ocultar información y de esta manera permitir a las personas u organizaciones compartirla en medios de comunicación inseguros de manera segura.

El criptoanálisis, también llamado "criptología", "perlustración" o "descriptación", por su parte es la disciplina contraria a la criptografía[3], se encarga de analizar la información cifrada para revelar la información original sin necesidad de las claves secretas y de esta forma romper los procedimientos previamente establecidos por la criptografía[3][4], siendo el antagonista principal de los criptosistemas; esta práctica aplicada contra organizaciones oficiales se le considera ilegal por los derechos de privacidad en algunos países, aunque por el otro lado existen trabajos actuales de investigación que tratan el criptoanálisis como una forma de probar la fortaleza de los criptosistemas.

Es preciso mencionar que el paso del tiempo ha ayudado al desarrollo de estas disciplinas con las nuevas teorías en campos de matemáticas y ciencias y que además la tecnología actual ha dotado de nuevas herramientas tanto a la criptografía como al criptoanálisis por igual ya que los procesadores multicore, los GPU's y el supercómputo han hecho posible ejecutar en semanas lo que habría costado bastantes años de ejecución en un procesador de la tecnología de hace 20 años, esto ha impulsado la necesidad de buscar procedimientos criptográficos cada vez más sofisticados para asegurar la invulnerabilidad a ataques criptoanalíticos como los estadísticos, el diferencial y el de módulo n por nombrar algunos.

Adicionalmente, la criptografía en imágenes digitales, particularmente, es muy reciente y se han encontrado diversas aplicaciones como lo son la medicina, donde es conveniente cifrar imágenes radiológicas que protejan la privacidad de los pacientes, el plano militar, donde también es necesario mantener en secreto las fotografías o imágenes de posiciones y edificios estratégicos, y la telefonía móvil y el internet, donde en las redes sociales y las nuevas tecnologías de la nube han sucedido eventos que dejan en evidencia la vulnerabilidad de los sistemas, en los que en muchos casos estuvieron implicadas imágenes cuya publicación viola la privacidad de los propietarios. Bien es dicho que *"una imagen vale más que mil palabras"*.

En cuanto a la aplicación práctica de la criptografía en imágenes digitales varios sistemas como los mecanismos de seguridad en sistemas de imágenes digitales médicas se basan en los estándares de la criptografía simétrica como DES (Data Encryption Standard) y AES (Advanced Encryption Standard) los cuales son algoritmos muy seguros pero tienen la desventaja de ser lentos en su proceso de cifrado cuando se tienen grandes volúmenes de datos[5] lo cual los hace poco viables al trabajar con imágenes sin importar la resolución

ni el tipo de aplicación en que se ocupen.

Por otro lado, las funciones caóticas deterministas desarrolladas como modelos matemáticos de sistemas no lineales son completamente sensibles a cambios simples en las condiciones iniciales lo que les proporciona gran aplicabilidad en diversos campos de la ciencia y la tecnología como meteorología, física de partículas, electrónica y la criptografía simétrica por su comportamiento pseudo aleatorio que es clave para que se hayan adaptado en un esquema de rondas iterativas para obtener así un código rápido para cifrar imágenes digitales.

Para este proyecto de tesis se propone un criptosistema basado en el esquema propuesto en el trabajo titulado *"Un esquema eficiente y seguro de protección de imágenes médicas basado en mapas caóticos"* elaborado por Fu, Chong et. al.[5], no obstante, se ha reemplazado uno de los sistemas caóticos por otro cuyas características en sus parámetros no sean tan restrictivas y así dar un rango mayor de valores posibles en la clave secreta, pues esto es sumamente importante para la seguridad de los sistemas simétricos. En cuanto a los apartados de este trabajo se inicia con Criptografía Simétrica donde se habla de los tipos de métodos criptográficos: simétrico, asimétrico y sus caracteristicas clave para poder diferenciarlos, en Funciones Caóticas se hace un breve estudio sobre la teoría del caos y las funciones caóticas, así como las características de las que toma ventaja la criptografía, luego, en imágenes Digitales se proporciona una breve exposición sobre cómo se representan los datos y en particular las imágenes en los sistemas digitales, todo esto para establecer el marco teórico en el que se desarrolló este trabajo y posteriormente explicar el esquema propuesto en el que se basa[5] y dar los detalles del esquema propio de cifrado-descifrado y contrastarlos en Propuesta y, para finalizar, en Pruebas se explica todas las pruebas a las que se sometió el criptosistema y se exponen los resultados para finalmente concluir con una breve reflexión general sobre este estudio en Conclusiones.

CRIPTOGRAFÍA SIMÉTRICA

Se ha visto antes la definición de criptografía y como parte complementaria se presentan algunas de las características de los sistemas criptográficos. —Auguste Kerckhoffs en su trabajo *"La criptografía militar"*[6] en 1883— recomendó que todo sistema criptográfico cumpliera con los siguientes principios:

1. Si el sistema no es teóricamente irrompible, al menos debe serlo en la práctica.

2. La efectividad del sistema no debe depender de que su diseño permanezca en secreto.

3. La clave debe ser fácilmente memorizable de manera que no haya que recurrir a notas escritas.

4. Los criptogramas deberán dar resultados alfanuméricos.

5. El sistema debe ser operable por una única persona.

6. El sistema debe ser fácil de utilizar.

La criptografía moderna está basada en estos principios, aunque con la diversidad de algoritmos que han surgido a lo largo del tiempo se han podido llegar a clasificar de acuerdo con:

a. El tipo de operación utilizada para obtener el criptograma (información cifrada) a partir del texto en claro.

b. El número de claves utilizadas durante el proceso de cifrado/descifrado.

c. La manera en que el texto es procesado.[4]

Dentro de la clasificación **b** hay dos tipos de métodos criptográficos muy característicos, uno es la criptografía simétrica o de clave secreta y la otra es la criptografía asimétrica o de clave pública.

- *La criptografía simétrica* como su nombre lo dice usa llave secreta y es simétrica, o en otras palabras, se usa la misma llave tanto para encriptar como para desencriptar la información cifrada —y en todos los casos la clave debe permanecer secreta a todos excepto a las entidades autorizadas para encriptar o desencriptar datos—[7], otra característica es que por su uso de operaciones matemáticas simples los sistemas criptográficos simétricos son usualmente más rápidos que los asimétricos. En la *Figura 1* se representa el esquema de un sistema simétrico con la misma llave para cifrar y descifrar.

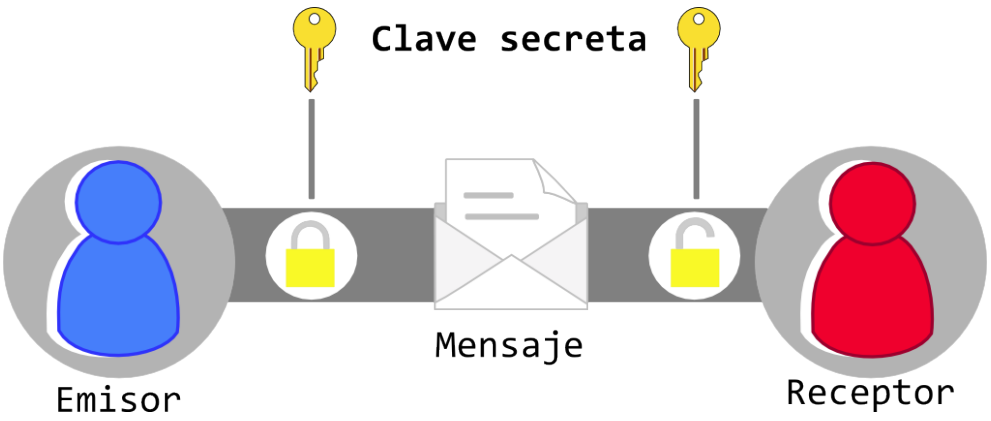

Figura 1: Esquema de la criptografía simétrica

- Mientras, las *funciones asimétricas* son las que permiten a los datos ser encriptados usando una clave para cifrar mientras se usa otra diferente para descifrar. Esto significa que no solo no es necesario mantener en privado una clave de encriptación, sino también es posible (y en muchos casos deseable) dar a conocer la clave de encriptación pública[7], este factor hace a los criptosistemas implementados de funciones asimétricas más seguras en comparación con las simétricas. En la *Figura 2* se puede visualizar un esquema de cifrado y descifrado con llaves diferentes.

En general, la característica principal de un buen criptosistema en esta clasificación es que su fortaleza recae en sus llaves y no en su algoritmo, esto quiere decir que para un criptoanalista de poco le debe servir conocer

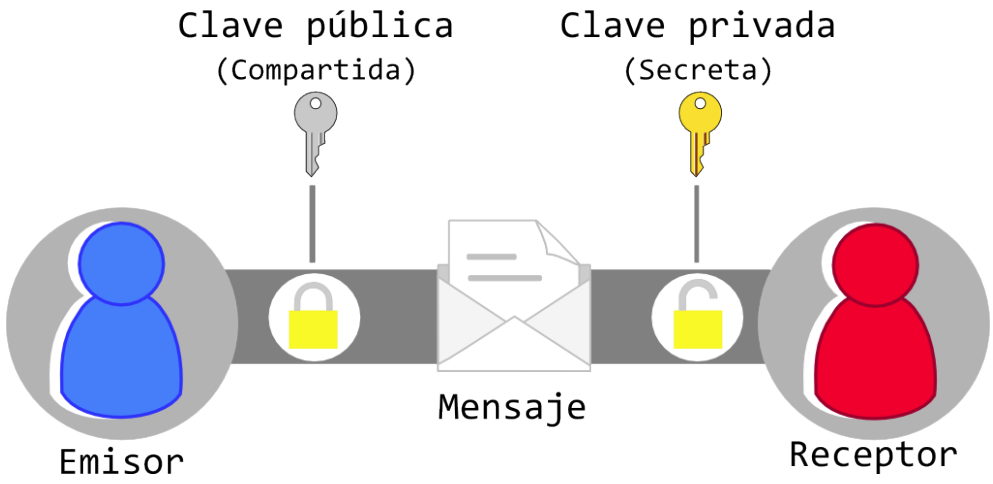

Figura 2: Esquema de la criptografía asimétrica

el algoritmo que se usó para cifrar el mensaje, así podemos ver que esta clasificación le da gran relevancia al principio no. 2 de Kerckhoffs.

En cuanto a la relación velocidad/seguridad con lo anteriormente dicho y comparando un método con el otro, se puede afirmar que las funciones simétricas son rápidas e inseguras mientras las asimétricas son lentas y seguras, aunque esto no signifique que con métodos simétricos no se pueda llegar a un nivel de seguridad aceptable y viceversa, con los asimétricos no se pueda llegar a un buen nivel de velocidad.

FUNCIONES CAÓTICAS

> "Por un clavo, se perdió una herradura, por una herradura, se perdió un caballo, por un caballo se perdió una batalla, por una batalla, se perdió el Reino. Y todo por un clavo de una herradura."
>
> — Canción popular Inglesa

Puede decirse que la dinámica caótica inició con el trabajo del matemático Francés Henri Poincaré apenas concluyendo el siglo XIX. La motivación de Poincaré fué promovida en parte por el problema de las orbitas de tres cuerpos celestiales experimentando atracción gravitacional mutua (por ejemplo, una estrella y dos planetas). Poincaré fué capaz de mostrar que orbitas muy complicadas (ahora llamadas caóticas) eran posibles. Subsequentes trabajos matemáticos iniciales dignos de notar en dinámica caótica incluyen a aquellos de G. Birkhoff en los años de 1920, M. L. Cartwright y J. E. Littlewood en los 40's, S. Smale en los años 60 y los matemáticos soviéticos, notablemente la de A. N. Kologorov y sus colaboradores. No obstante de estos trabajos, la posibilidad de caos en sistemas físicos reales no fué ampliamente apreciado hasta la actualidad. La razón de esto fué primero porque los artículos matemáticos son difíciles de leer para trabajadores de otros campos y segundo porque los teoremas prueban a menudo no ser lo suficientemente fuertes para convencer a los investigadores de estas otras áreas que este tipo de comportamiento sería importante para sus sistemas. La situación ahora ha cambiado drásticamente y mucho del crédito por esto es adscrito a la extensa solución numérica de sistemas dinámicos en computadoras digitales[8].

En los últimos 10 años, los hilos de caos y dinámica no lineal se han esparcido a través de disciplinas científicas como una intrincada red de araña. Caos y dinámica no lineal han provisto de nuevas herramientas teóricas y conceptuales que permiten capturar, entender, y enlazar los comportamientos complejos de sistemas simples —el tipo de comportamiento llamado *caos* en la ciencia contemporánea—[9].

10

Definiendo caos y funciones caóticas

Dado que la teoría del caos surge dentro del estudio de sistemas dinámicos, se debe analizar primero la definición de estos sistemas. Así, se encuentra que un *sistema dinámico* puede ser definido como un modelo matemático determinista para el estado evolutivo de un sistema a lo largo del tiempo. Aquí tiempo puede ser tanto una variable continua como una variable discreta de valor entero[8]. En otras palabras se puede decir que por lo general los sistemas dinámicos describen los fenómenos de elementos con diferentes características (parámetros) que evolucionan a través del tiempo y que un estado en tiempo t dependerá del estado previo en el instante *t-1*. Un ejemplo de estos sistemas es la trayectoria de un péndulo simple al ser impulsado con una fuerza inicial, se observa entonces que la trayectoria que sigue el elemento *péndulo* describe la manera en que evoluciona este elemento en el espacio a través del tiempo.

Por otra parte, los sistemas dinámicos pueden ser diferenciados entre lineales y no lineales. Los sistemas caóticos por su parte tienen la propiedad de ser no lineales y se debería entender esta característica para tener un panorama más completo de lo que las funciónes caóticas son, asi que principalmente se podría desarrollar una idea intuitiva de no linealidad caracterizando el comportamiento de un sistema en términos de estímulo y respuesta, o dicho de otra forma, causa y efecto: Si se diera una "patada" al sistema y se observara cierta respuesta a esa patada, entonces podría preguntarse ¿que es lo que pasaría si se pateara al sistema el doble de fuerte? Si la respuesta es el doble de grande, entonces el sistema se dice que es lineal (al menos para el rango de tamaños de patada que hemos usado). Si la respuesta no es el doble de grande (podría ser más grande o más chico), entonces se dirá que el comportamiento del sistema es no lineal. El estudio del comportamiento no lineal es llamado **dinámica no lineal**. Dando una definición un poco tosca, se encuentra que: "Un sistema no lineal es un sistema cuyas ecuaciones de evolución de tiempo son no lineales; esto es, las variables dinámicas describiendo las propiedades del sistema (por ejemplo, posición, velocidad, aceleración, presión, etc.) aparecen en la ecuación en una forma no lineal[9].

Ahora, se puede continuar con la definición de lo que se conoce como caos. *Caos* es un término usado para describir el comportamiento aparentemente complejo de lo que se consideran ser sistemas simples o bien portados. El comportamiento caótico, cuando es visto casualmente, luce errático y casi aleatorio[9]. De forma similiar a esta definición como se vió al principio del capítulo, Poincaré se encontró con un problema que bajo observación parece simple, sin embargo al intentar describir y predecir las tres órbitas de los

cuerpos con atracción gravitacional mútua, reveló ser realmente un problema bastante complejo.

Toda vez que se ha definido el caos, se puede decir que una función caótica es toda función matemática usada para describir un sistema dinámico no lineal complejo cuya evolución en el tiempo hace imposible la predicción a largo plazo, luciendo errático y casi aleatorio. Es de notar además que se suelen llamar también *mapas* a las funciones que describen trayectorias sobre el espacio, por lo tanto pueden haber mapas caóticos de distintas dimensiones y esto depende de la dimensión del espacio sobre el cual se mapean (unidimensional, bidimensional, tridimensional o multidimensional), esto último es importante para nuestro estudio ya que dentro de la criptografía caótica es popular el uso de mapas caóticos.

Características caóticas

Los ejemplos más notorios de las características del caos son el llamado *"efecto mariposa"* y la *impredecibilidad de órbitas pseudo-aleatorias*, generadas por ecuaciones deterministas. Estos fenómenos, así como otros relacionados con el caos han sido tradicionalemente asociados a mecanismos de *"confusión"* y *"difusión"* los cuales son la base principal de un buen sistema criptográfico. A continuación se enlistan ciertas popiedades interesantes que ayudan a generar las características antes mencionadas:

- **Ergodicidad:** propiedad por la cual una trayectoria en espacio de fase se vuelve arbitrariamente cercana a sus estados anteriores. Esto refleja, esencialmente, que el sistema eventualmente es confinado a un objeto espacial (un conjunto de puntos llamado *atractor*). La densidad de dichos puntos es invariante al tiempo. En otras palabras, las órbitas de la trayectoria del sistema caótico a pesar de ser diferente cada vez y no presentar repeticiónes cíclicas aparentes, también llegan a ser similares y a orbitar —todas— alrededor de puntos en el espacio llamados atractores. Ver *Figura 3*.

- **Sensibilidad a condiciones iniciales:** a diferencia de los sistemas deterministas en los cuales si se tiene un estado inicial, los estados futuros del sistema pueden predecirse, los sistemas caóticos hacen imposible la predicción a largo plazo. Para valores específicos de los parámetros, dos trayectorias inicialmente muy cercanas divergen exponencialmente en un corto periodo de tiempo. De esta manera, la información inicial sobre el sistema se pierde completamente.

- **Mezcla:** característica de un sistema en la cual un pequeño intervalo de condiciones iniciales se esparce sobre el epacio de fase completo en su evolución asintótica. En un sistema caótico, un intervalo arbitrario de condiciones iniciales extiende la parte del espacio de fases (*atractor*) hacia el cual se confina la trayectoria asintóticamente[10].

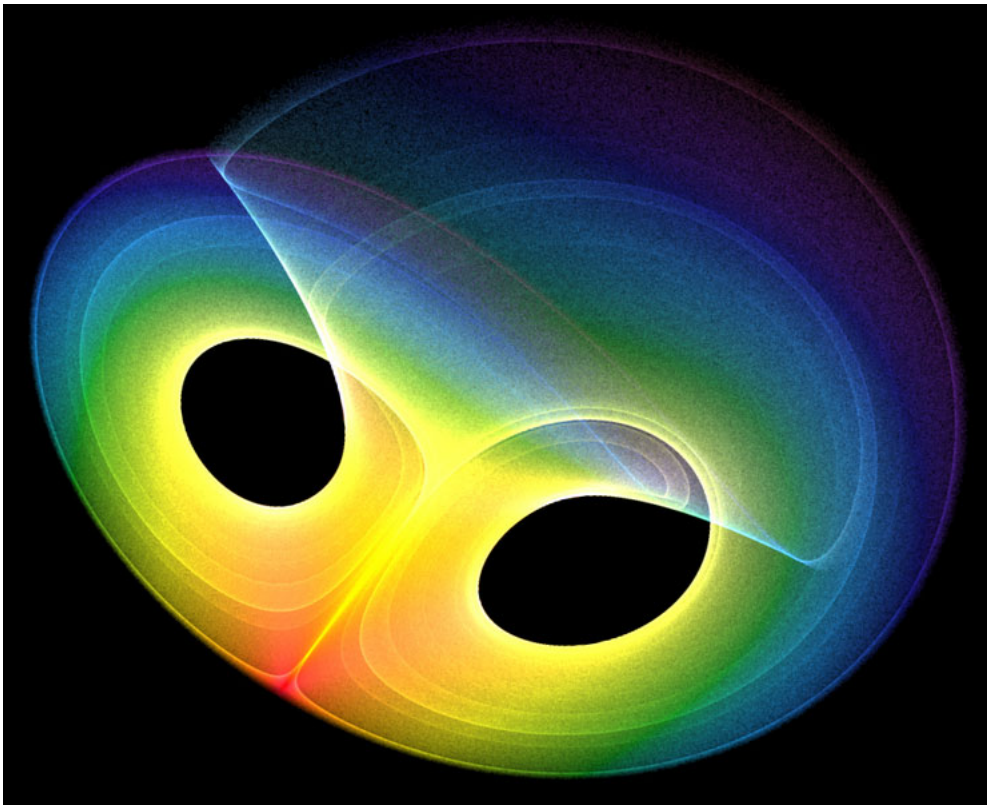

Figura 3: Ergodicidad y atractores del sistema de Lorenz

Algunos autores usan el fragmento de la canción que aparece al inicio de este capítulo para ejemplificar estas propiedades, principalmente *la sensibilidad a condiciones iniciales*, la cual es una propiedad notable para el marco teórico de este y otros trabajos similares pues además es aprovechada dentro de la criptografía ya que los parámetros y las condiciones iniciales forman un amplio espacio de claves que brinda gran seguridad al sistema.

Otros estudios

El estudio en este capítulo —como se puede intuír— es solo la punta del iceberg dentro del estudio de sistemas caóticos que engloban las aportaciones que ha tenido la teoría en diferentes trabajos en los últimos años y que se han agudizado gracias a las herramientas de cálculo y simulación que ha brindado la computación a las ciencias modernas, así que para mayor información y ejemplos más especificos sobre este tema es recomendable acudir a los siguientes libros dentro y fuera de la bibliografía aquí presentada:

- **Chaos and Nonlinear Dynamics** de Robert C. Hilborn donde explica a fondo las características particulares de los sistemas caóticos y dinámicos no lineales. [9]
- **Chaos in Dynamical Systems** de Edward Ott, donde además de caracterizar el caos por el estudio de sistemas particulares, también explica otras teorías y definiciones que giran en torno a la órbita del caos como los multifractales y el caos cuántico[8].
- **¿Juega Dios a los dados?** de Ian Stewart, donde el caos es resaltado en varios aspectos de la ciencia a lo largo de la historia y genera definiciones muy bien fundamentadas y entendibles. (Sin entrada en bibliografía).
- **Teoría del Caos: El efecto mariposa** de Pablo Mejía, Carlos Roncero, Saida Aguilar, Roberto Martinez y Victor Yataco, el cual es un estudio muy completo y bien explicado sobre la teoría de caos, sus características y sus efectos, recomendable para cualquier persona no iniciada en el tema y que quiera tener una idea clara sobre el caos y sistemas dinámicos no lineales. (Sin entrada en bibliografía).

IMÁGENES DIGITALES

Para el propósito de este trabajo resulta provechoso sumergirse un poco en el mundo digital y conocer cómo es que se representan y manipulan las imágenes en los dispositivos regidos por los principios de los sistemas digitales, ya que sin esta singular visión resultaría problemático entender los algoritmos empleados en los criptosistemas para imágenes así como lo sería el proponer e implementar cambios a dichos algoritmos.

Los datos y los sistemas digitales

Buscando una definición apropiada se encuentra en *"Sistemas Digitales: Principios y Aplicaciones"*[11] que un **sistema digital** es:

> *"Un conjunto de dispositivos destinados a la generación, transmisión, manejo, procesamiento o almacenamiento de señales digitales."*

Dicho de otra forma es una combinación de dispositivos diseñados para manipular cantidades físicas o información que estén representadas en forma digital; es decir, que sólo puedan tomar valores discretos. En el libro *"Sistemas Electrónicos Digitales"*[12], se encuentra también que:

> *"Los sistemas digitales actúan bajo el control de variables discretas, entendiéndose por éstas, las variables que pueden tomar un número finito de valores."*

Dando otra definición, se podría iniciar tomando en cuenta que los sistemas electrónicos se dividen en **sistemas analógicos** y **sistemas digitales** por sus formas características de representar a los datos. En los sistemas analógicos un dato es determinado por la tensión eléctrica variable en un canal, mientras en los sistemas digitales los datos son determinados por la presencia o ausencia de señales o pulsos eléctrónicos en un número finito de medios dispuestos en forma de arreglo. Así que bajo estas características, los datos

en los sistemas digitales se toman como un arreglo de variables que pueden tomar solo dos valores usualmente representados por los símbolos *[0, 1]* y por lo tanto, se dice que los números están codificados en un sistema binario o de base dos.

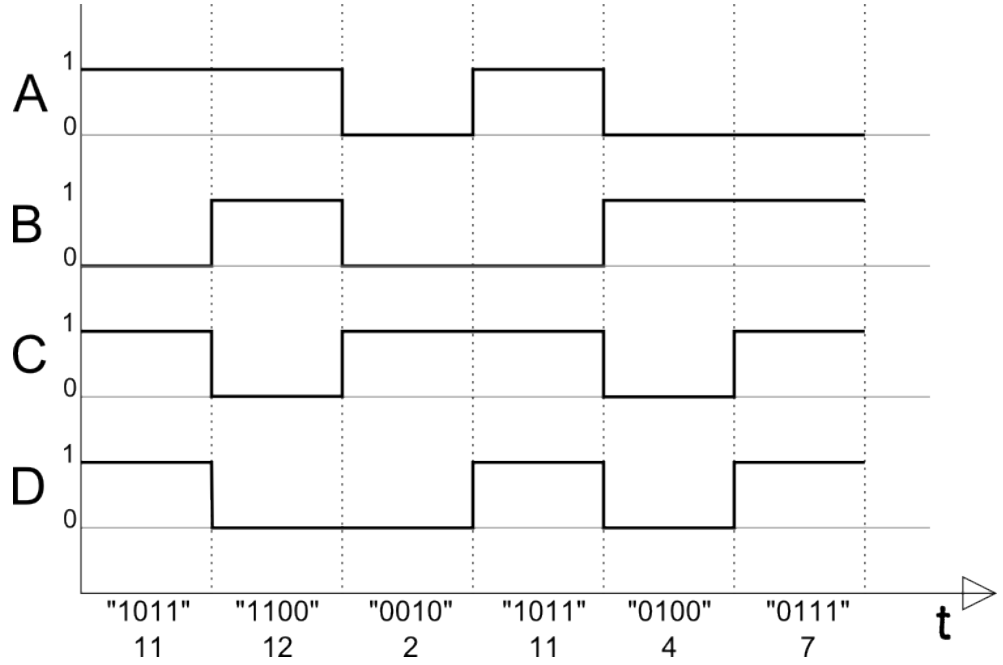

Figura 4: Esquema teórico, señales digitales

En las señales binarias con las que trabajan los sistemas digitales la unidad básica de información es el **BIT** por la contracción en inglés de "BInary digiT" o dígito binario. Dado que un bit puede contener una cantidad de información sumamente pequeña, los sistemas digitales como las computadoras suelen trabajar con agrupaciónes de bits a las que se llama **palabra**, normalmente compuesta por 8, 16, 32 o 64 bits. Esta longitud de palabra determina el número de bits con que el sistema puede trabajar a la vez. En la *Figura 4*, por ejemplo, podemos ver un esquema teórico de las señales binarias en un sistema que trabaja con 4 bits. Los grupos de 8 bit son utilizados con mucha frecuencia, por lo que se les conoce especialmente como **byte**.

Dentro de la computación fundamentalmente, resulta muy importante conocer la codificación de los sistemas binarios pues con estos números y usando estructuras se codifican además otros tipos de datos, como por ejemplo, los números naturales o los enteros, los números reales o de punto flotante, las letra y símbolos que usamos en los textos o caracteres, los números imagi-

narios, las imágenes propiamente, etc. Un tópico importante que viene de conocer el sistema binario y que por cierto se necesitará más adelante, es el del rango de números que se pueden representar con n digitos binarios y que se explica a continuación.

Como anteriormente se dijo, con un digito binario o bit se puede representar unicamente 2 estados diferentes, si a este bit, le agregamos otro, entonces podemos representar $2 \cdot 2 = 4$ diferentes estados (o números), si aumentamos a 3 nuestro tamaño de arreglo de bits entonces tendremos $2 \cdot 2 \cdot 2 = 8$ números diferentes que podremos representar, bueno, así susesivamente, si continuamos aumentando hasta n bits se tendría que:

$$\underbrace{2 \cdot 2 \cdot 2 \cdot ... \cdot 2}_{\text{n veces}} = 2^n \tag{1}$$

números diferentes, esto es porque básicamente lo que se tiene es una permutación de 2 elementos con repetición sobre n lugares posibles. En una *imagen* especificamente, el dato básico está codificado en una representación a 8 bits, con lo cual se pueden representar, $2^8 = 256$ números, que en este caso particular cubren el rango [0 – 255] y que se necesitará para explicar la representación general de las imágenes en medios digitales.

Imágenes en medios digitales

La forma en que se visualizan imágenes o gráficos en los sistemas digitales es por medio de pantallas, estas son arreglos de pequeños puntos (pixeles) cuya intensidad y frecuencia en la radiación de luz permiten percibir diferentes colores. Para poder almacenar, manipular y procesar estas imágenes, se requiere **una estructura de datos** y **un código para representar las diferentes tonalidades de colores** que se traducirán en brillantes imágenes en pantalla. Se iniciará con este último, en los pixeles de las pantallas se pueden encontrar tres colores fundamentales brillando a diferentes intensidades, el rojo, el verde y el azúl, la combinación de estos colores permiten percibir casi cualquier otro color cuando jugamos con las intensidades, análogamente, en la representación digital se encuentra un código para representar las intensidades de estos colores llamado *Código RGB*. El código RGB es un arreglo de tres valores y como se dijo antes, el dato básico está representado en código binario a 8 bits, así que cada uno de los tres valores del código RGB puede tomar valores del rango [0 – 255], el conjunto de estos tres valores es la representación de un simple *pixel*.

Además de esto necesitamos una estructura, puesto que una pantalla está compuesta por varios pixeles acomodados en un arreglo bidimiensio-

nal, encontraremos que una imagen digital por lo tanto es una matríz de M columnas por N renglones donde cada elemento es la representación (estructura) de un pixel, otra manera de ver esta representación es como una matriz tridimensional de $M \cdot N \cdot 3$, es decir, tres matrices o canales diferentes que le añaden una nueva dimensión al arreglo.

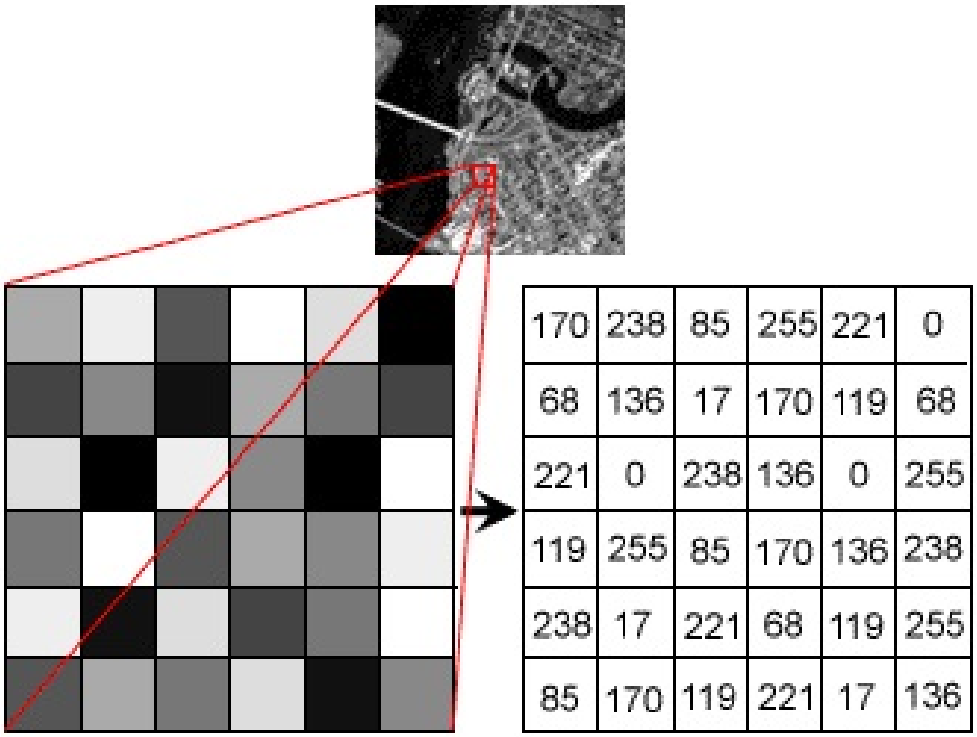

Figura 5: Representación de imagen a escala de grises

De esta forma se representa una amplia gamma de colores y cualquiér imagen puede ser codificada en estos arreglos. Para las imágenes en escala de grises, se usa el mismo tipo de representación pero cada valor del código RGB en un pixel es el mismo lo que nos da un color gris, la variación en ese valor nos dá una amplia intensidad de grises. En la *Figura 5* se observa como se representan con valores numéricos en una matriz, los pixeles de una imagen a escala de grises, los tres valores del código RGB tienen el mismo valor en cada pixel.

PROPUESTA

Este proyecto se centra en la conjunción de la criptografía simétrica, las funciones caóticas y las imágenes digitales. Ya se estudió cada uno de estos tópicos por separado pero hace falta conocer cómo se han adaptado estos conceptos para formar esquemas eficaces para fines de encriptación de imágenes.

Antecedentes

La teoría de funciones y mapas caóticos para protección de imágenes digitales se han presentado en varios trabajos, Fridrich en 1998[13] sugirió un esquema de encriptación de imagen basado en caos compuesto de dos módulos: confusión caótica (cambio de lugar de los pixeles) y difusión de pixeles (cambio de valor). Esta arquitectura conocida como confusión-difusión o permutación-sustitución, es la primera arquitectura general la cual guía el diseño de los cifrados de imágenes basados en caos, posteriormente Wong, Kwok y Law en el 2008 [14] proponen un esquema de encriptación basado en el mapa caótico estándar; Gao y Chen en el 2008[15] proponen un algoritmo nuevo de permutación de pixeles; Sun, Liu y Li en el 2008[16] proponen un esquema de encriptación basado en mapas caóticos espaciales; Xu, Wang y Yang en el 2008[17] proponen una mejora en el algoritmo de encriptación de imágenes que usan mapas caóticos agregando un pequeño estado de difusion en la fase de permutación; Ye[18] en el 2009 propone un cripto-sistema basado en las matrices de Töplitz y Hankel; Tong y Cui en el 2009[19] proponen un generador de cifrado de secuencias caóticas con componentes dinámicos; Huang y Nien en el 2009[20] proponen un sistema multi-caótico basado en el mismo principio de permutación; Patidar, Pareek y Sud en el 2009[21] proponen un cifrado tipo substitución-difusión basado en mapas logísticos y caóticos; Rhouma, Meherzi y Belghith en el 2009 [22] proponen un mapa reticular unidireccionalmente acoplado (UCML) para el cifrado de imágenes a color; Xiao y Xia en el 2009[23] proponen un esquema de encriptación usando mapas de permutación; en el 2013 Fuan, Meng, Zhan, Zhu, Lau, Tse y Ma[5] proponen un esquema de protección de imágenes médicas basado en el mapa del gato

de Arnold a nivel de bits y el mapa logístico.

Como lo muestran los trabajos mencionados, las funciones y los mapas caóticos han llamado la atención de varios investigadores y se han adaptado ya en esquemas de rondas iterativas para cifrar imágenes digitales y así se ha comprobado la velocidad y la seguridad de estos métodos comparados con otros algoritmos de la criptografía simétrica lo que ha inspirado este trabajo e impulsado el desarrollo de una propuesta propia.

Objetivos

El objetivo general de este proyecto es desarrollar una aplicación en OCTAVE/MATLAB que permita cifrar y descifrar de forma rápida y segura imágenes digitales en cualquier tipo de formato, de cualquier tamaño, a color o a escala de grises y dar evidencia de la rapidéz con experimentos y pruebas de ejecución. Para lo cual se deberán cubrir además los siguientes objetivos particulares:

1. Investigar y comprender la teoría de las funciones y los mapas caóticos
2. Diseñar un algoritmo o propuesta para cifrar y descifrar una imagen digital usando un esquema de rondas iterativas.
3. Implementar en OCTAVE/MATLAB el algoritmo para cifrar y descifrar.
4. Realizar las pruebas con imágenes de diversos tamaños y registrar los tiempos de ejecución.

Material utilizado

Para realizar el proyecto se hizo uso de una computadora portátil con las siguientes especificaciones de Hardware:

1. Procesador Intel Core 2 Duo de 64 bits a 2.10 Ghz.
2. 6GB de Memoria Ram DDR3.
3. Pantalla integrada LCD de 17".

En cuanto al Software se tienen las siguientes características y herramientas:

1. Sistema Operativo Windows 7 Professional.

2. Ambiente de desarrollo de cálculo matemático OCTAVE/MATLAB.

Para los que quizá no conozcan nada sobre MATLAB éste es un ambiente de desarrollo con un lenguaje de programación propio que integra varias bibliotecas de funciones e interfaces que facilitan el desarrollo de trabajos matemáticos y científicos gracias a que la sintáxis (parecido también al lenguaje C) se asemeja a las utilizadas en estos ámbitos. Además facilita la generación de gráficos y por esto es ampliamente utilizada para fines de investigación como en este caso.

Esquema general

Como anteriormente se mencionó, en 1998 Fridrich[13] propone una arquitectura de encriptación basada en caos de confusión-difusión, esta se convirtió en una guía de diseño para los subsecuentes trabajos de criptografía caótica para imágenes.

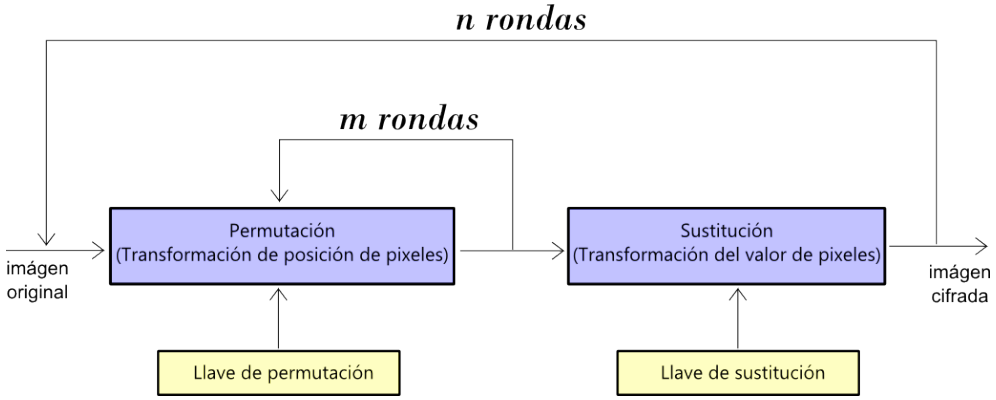

Figura 6: Arquitectura para encriptación de imágenes basado en caos

El esquema de Fridrich (ver *Figura 6*) se compone por dos fases iterativas, la fase de permutación y la fase de sustitución. En la fase de permutación la imagen completa es reorganizada en un orden diferente y usualmente muy complejo, sin embargo los pixeles por si solos permanecen sin cambio. Para esta fase se utilizan usualmente uno de tres mapas caóticos bidimensionales que no presentan pérdida de datos o preservan el aŕea de la imagen: el mapa del gato de Arnold (Arnold cat map), el mapa del pastelero (Baker map) y el mapa estándar (standard map). En la fase de sustitución los pixeles son alterados sequencialmente y la modificación en un pixel usualmente depende

del efecto acumulado de los valores de todos los pixeles anteriores y por lo tanto la influencia de un solo pixel puede ser efectivamente difuminado, para este fin se usan funciones caóticas como el mapa logístico, el mapa de Lorenz o el mapa de la tienda (tent map) para generar un flujo de claves pseudo-aleatorias y después combinarlas con los valores de los pixeles usando operaciones simples como el XOR, XNOR, suma, resta o rotación de bits o incluso una combinación de ellas y así, aplicando varias rondas generales de encriptación obtener un alto nivel de seguridad[5].

Fase de permutación

Esta propuesta está basado en el esquema general de Fridrich y como tal cuenta con una fase de permutación, para lo cual decidió basarse además en una nueva e interesante variante del mapa del gato de Arnold propuesta por Fu, Meng y sus colaboradores[5]. En aras de explicar el método de permutación de la propuesta deberemos entender primero el mapa del gato de Arnold y posteriormente la adaptación que le fué implementada.

Mapa del gato de Arnold

El mapa del gato de Arnold recibe este nombre en reconocimiento del matemático Ruso Vladimir I. Arnold quien lo descubrió usando la imagen de un gato. En este mapa una imagen, necesariamente cuadrada, es modificada con una transformación que aparentemente vuelve aleatoria a la organización original de los pixeles[24]. Otra forma de verlo es como una biyección de la unidad cuadrada $I \cdot I$ (imagen) en sí misma y es definida por:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & p \\ q & pq+1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} \bmod N \tag{2}$$

donde p y q son parámetros de control y $X_r \bmod N$ es el residuo de la división de la matríz resultante de la multiplicación de matrices previa —por nombrarla X_r— entre el tamaño del lado de la imagen N. En la *Figura 7* se puede observar el efecto de esta transformación en una hipotética imagen de tamaño 1, al aplicar a la imagen esta transformación varias veces el resultado es una imagen irreconocible. Este mapa no presenta pérdida de datos de la imagen original y es invertible ya que la matríz de transformación (la que contiene a los parámetros de control) tiene determinante 1[5].

Sin embargo, existe una característica interesante en este mapa, cuando la transformación del mapa es aplicada varias veces lo suficiente, la transformación vuelve a ser la imagen original habiendo así ciclos donde la imagen

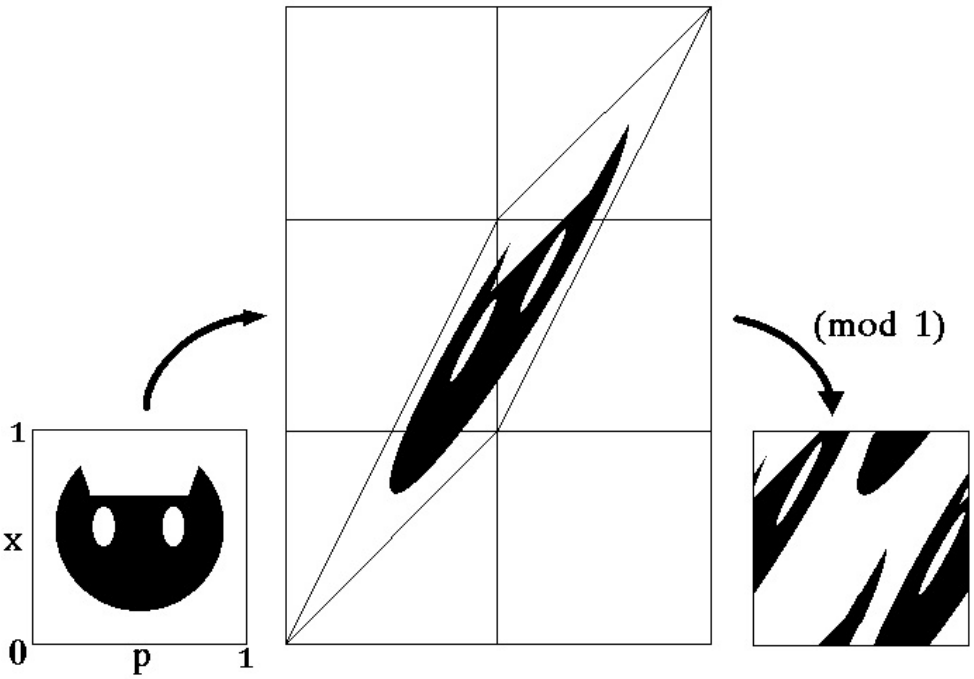

Figura 7: Efecto del mapa del gato de Arnold

pasa por varias fases hasta regresar al inicio, esto puede ser apreciado cuando los parámetros p y q son suficientemente pequeños, por ejemplo, cuando p y $q = 1$ tenemos que:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} \bmod N$$

y haciendo un experimento en la aplicación para calculo numérico MATLAB con este mapa de Arnold se encontró que iterando esta transformación 192 veces sobre una imagen de tamaño 256x256, la imagen pasa por varios estados y regresa al estado inicial de la imagen original como lo vemos presentado en la *Figura 8*. Sin embargo no siempre el ciclo es igual y depende tanto de los parámetros que se elijan como del tamaño de la imagen que se está transformando. Para el experimento se implementó una función la cual toma el mapa del gato anteriormente descrito y transforma una sola vez la imagen, luego en un script se itera la imagen varias veces (192) llamando a esta función cada vez. A continuación se presenta el código de esta función del mapa del gato de Arnold:

```
function [ img1 ] = cat_map( img )
```

```
%Obtenemos las dimensiones MXN de la imagen (donde m = n)
[ m, n ] = size( img( :, :, 1 ) );

%Matriz de transformacion del mapa del gato
M=[1 1;1 2];

%Inicializamos una matriz nueva (imagen).
img1 = uint8(zeros( m, n, 3));

for i = 1 : m
      for j = 1 : n

              %Calcula nueva posicion para el pixel actual
              pos = mod( M * [i-1;j-1], n );

              %Copia pixel en nueva posicion (3 canales)
              img1( i, j, 1 ) = img( pos(1)+1, pos(2)+1, 1 );
              img1( i, j, 2 ) = img( pos(1)+1, pos(2)+1, 2 );
              img1( i, j, 3 ) = img( pos(1)+1, pos(2)+1, 3 );
      end
end
end
```

La transformación inversa del mapa para deshacer los efectos en una imagen está definida por:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} pq+1 & -q \\ -p & 1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} \bmod N \tag{3}$$

permitiendo con esta crear un método para regresar a la imagen original aplicando el mismo número de iteraciones que el aplicado en la primera transformación.

Variante, mapa del gato a nivel de bits

No obstante, el mapa del gato de Arnold es un mapa caótico de dos dimensiones que por si mismo no genera cambios en los pixeles, solo altera el orden de estos en toda la imagen. Esta reiteración es principalmente por las conclusiones del trabajo de Sun F.Y., Liu S.T. y Li Z.Q. et. al.[16] las cuales afirman que un pequeño cambio en la etapa de permutación permite disminuir el número de iteraciones generales y llegar así a reducir el tiempo

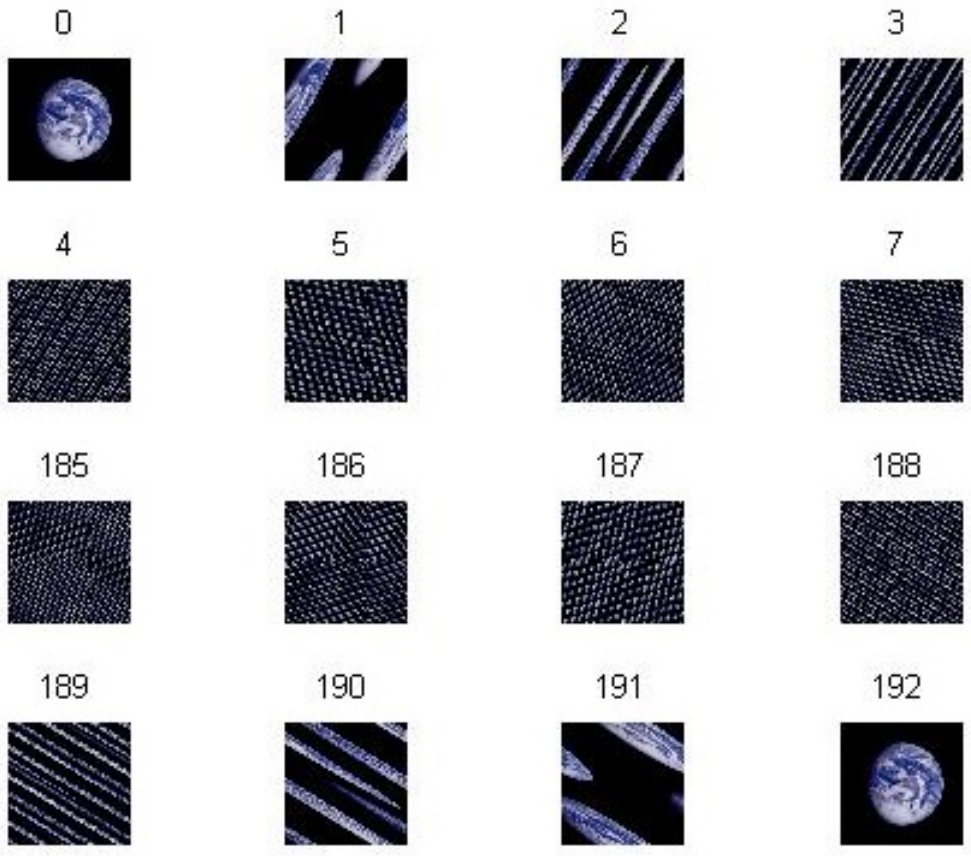

Figura 8: Iterando el mapa del gato de Arnold

global de encriptación sin afectar el nivel de seguridad del criptosistema. Por esto, Fu Chong, Meng Wei-hong, Zhan Yong-feng y sus colaboradores[5] proponen la variante del mapa del gato de Arnold aplicado a nivel de bits para implementar de esta forma un cambio de valor en los pixeles mientras estas son permutadas.

Como fué visto anteriormente en imágenes Digitales, la representación básica de un pixel es llevada a cabo con 3 datos distribuidos en tres canales diferentes y estos datos se encuentran codificados en código binario en una representación a 8 bits, por lo tanto, en esta variante se propone trabajar por separado cada uno de los 8 niveles de bits de los datos de la imagen con un mapa del gato de Arnold diferente teniendo así un efecto de difusión mientras se permutan los pixeles. En palabras simples, cada dato básico se descompone en bits y cada bit es mandado a un lugar diferente de la imagen. Siendo que con cada uno de los 8 niveles de bits se inicializa un mapa del

gato diferente se puede ver esto como 8 diferentes planos de bits influenciado por un mapa diferente lo único que habría que hacer para deshacer el trabajo de este método es compartir la llave, en este caso el conjunto de valores para los parámetros p y q de cada uno de los mapas usados, y luego inicializar 8 mapas inversos cada uno con sus parámetros y trabajar cada nivel con su respectivo mapa inverso.

De esta manera un método con esta variante del mapa del gato de Arnold ejecutaría lo siguiente:

1. Obtener las medidas de la imagen m, n y generar los parámetros *p(8)* y *q(8)*.

```
[ m, n ] = size( img( :, :, 1 ) );

%generamos las llaves p y q.
p = ceil( rand( 1, 8 ) * n );
q = ceil( rand( 1, 8 ) * n );
```

2. Inicializar las matrices *M(2)(2)(8)* y *planes(m)(n)(8)* donde se almacenarán los 8 mapas de transfromación del mapa del gato diferentes y los 8 planos de bits de la imagen respectivamente.

```
M = zeros( 2, 2, 8 );
planes = uint8(zeros( m, n, 8 ));

% a[ m ][ n ] = 128 = '10000000'
a = uint8( ones( m, n, 1 ) * 128 );
```

3. Calcular los 8 mapas y obtener los 8 planos de bits de la imagen (variable *planes(m)(n)*).

```
for k = 1 : 8
    %calculamos la matriz del mapa
    M(:,:,k)=[1 p(k);q(k) p(k)*q(k)+1];

          %planos[][][ k ] = ( a & img[][][ 1 ]) << (8 - k)
          planes(:,:,k)=bitshift(bitand(img(:,:,1),a),-(8- k));
          % a  = '10000000'>> 1 = '01000000'
          a = bitshift( a, -1 );
end
```

4. Inicializar una nueva imagen e iterar 3 veces las transformaciones de los mapas del gato.

```
%Inicializamos la imagen (dividida en planos)
%cifrada que obtendremos.
img1 = uint8(zeros( m, n, 8 ));

%iteramos 3 veces.
for r = 1:3
    for i=1:m
        for j=1:n
            for k=1:8
                pos = mod(M( :, :, k )*[i-1;j-1], n);
                img1(i,j,k) = planes(pos(1)+1, pos(2)+1, k);
            end
        end
    end
    planes = img1;
end
```

5. Por último, unir los planos de bits para convertirlos en una imagen completa y devolver las llaves (16).

```
cifra = uint8( zeros( m, n ) );
for k = 1:8
   img1( :, :, k ) = bitshift( img1( :, :, k ), 8-k );
    cifra( :, :, 1 ) = bitor( cifra, img1( :, :, k ) );
end
llave=[p,q];
```

Para implementar el método inverso de esta variante del mapa del gato (para descifrar), se deberían ejecutar prácticamente las mismas instrucciones con la excepción que las llaves p y q se toman de la llave obtenidas al cifrar y cuando se generan los mapas de transformación *M(2)(2)(8)*, estos se deben obtener con el mapa inverso.

Método de redimensión de imágenes

Aún entendiendo la variante del mapa del gato de Arnold faltaría un detalle para entrar de lleno a la implementación del algoritmo, y esto es por el hecho de que el mapa del gato solo puede ser aplicado a imágenes

cuadradas, si no se quiere limitar a este tipo de imágenes y poder trabajar con cualquiera, un método que convierta una imagen rectangular a una cuadrada es necesaria.

Para redimensionar una imagen rectangular convirtiéndola en una cuadrada se necesita saber el tamaño de lado L que minimamente debe tener la imagen nueva para poder contener el número de datos de la imagen, esto quiere decir que, la dimensión de la nueva imagen sería por ende $L \cdot L$, en el trabajo de Fu Chong [5] se muestra la fórmula para calcular este dato, la cual es:

$$L_s = \lceil \sqrt{M \cdot N} \rceil \tag{4}$$

donde el fragmento $M \cdot N$ es el tamaño de la imagen rectangular y aplicandole la raíz cuadrada se encuentra el tamaño exacto del lado para redimensionar la imagen, sin embargo, este no es un número entero por lo tanto se usa la operación $\lceil n \rceil$ (también llamada cielo) para obtener el siguiente número entero mayor (redondear), esto quiere decir que al final deben necesitarse más pixeles de los que se tienen, por lo tanto lo que una función de redimensión debería hacer es —una vez creada la nueva imagen de $L \cdot L$—

```
[ m,  n ] = size(  img( :, :, 1 ) );
 Ls = ceil( sqrt( m * n ) );
 rshp_img = uint8( zeros( Ls, Ls, 1 ) );
```

reacomodar la imagen pixel por pixel de forma secuencial en la imagen con nuevas dimensiones

```
k = 1; l = 1;
for i = 1 : m
    for j = 1 : n
        rshp_img( k,  l, 1 ) = img( i, j, 1 );
        l = l + 1;
        if l > Ls
            l = 1;
            k = k + 1;
        end
    end
end
```

y al final rellenar los pixeles faltantes con números aleatorios.

```
for i = k : Ls
 for j = l : Ls
     rshp_img( k, l, 1 ) = ceil( rand(1)*255 );
```

```
    end
  end
```

Como en el caso del mapa del gato, también este método de redimensión debe tener un método inverso que convierta la imagen cuadrada a la imagen rectangular original para ser usada en el método de descifrado cuando sea necesario. Para esto debe compartirse cierta información para saber las dimensiones originales de la imagen, en nuestra propuesta particular se decidió primero despejar la ecuación, de la siguiente forma. Se tiene la ecuación (4) y si se eleva al cuadrado ambos lados de la igualdad se tiene que:

$$L_s^2 \approx M \cdot N$$

como no se puede decir que ambas partes son iguales porque anteriormente para obtener L_s se aplicó la operación $\lceil x \rceil$ la diferencia de estas dos cifras resultan ser el número de pixeles faltantes o sobrantes, según el tipo de redimensión que se esté aplicando, aquí llamado R y definido por:

$$R = L_s^2 - M \cdot N$$

Ahora, una vez especificado esto, podemos compartir dos datos suficientes para redimensionar la imagen a su forma original, esto es el número R y cualquier lado de la imagen original, ya sea M o N, con esto se deberá obtener la medida del lado faltante de la siguiente forma:

$$\begin{aligned} M &= (L_s^2 - R)/N \quad \text{Se comparten} R \text{y} N \\ \text{ó} & \\ N &= (L_s^2 - R)/M \quad \text{Se comparten} R \text{y} M \end{aligned}$$

De esta manera el método de redimensión inverso debe inicializar una matriz de $M \cdot N \cdot 3$ y luego mover secuencialmente cada pixel de la imagen cuadrada a la nueva imagen deshaciendose al final de los pixeles extra.

```
k = 1; l = 1;
for i = 1:m
    for j = 1:n
        rshp_img( i, j, 1 ) = img( k, l, 1 );
        l = l+1;
        if l > Ls
            l=1;
            k=k+1;
        end
    end
end
```

En la *Figura 9* se observa una sencilla prueba de estas dos funciones, en la primera imagen está la imagen original, la siguiente es la imagen cuadrada construida con el método de redimensión y al final se muestra la imagen resultante de aplicarle el método inverso de redimensión a la imagen cuadrada.

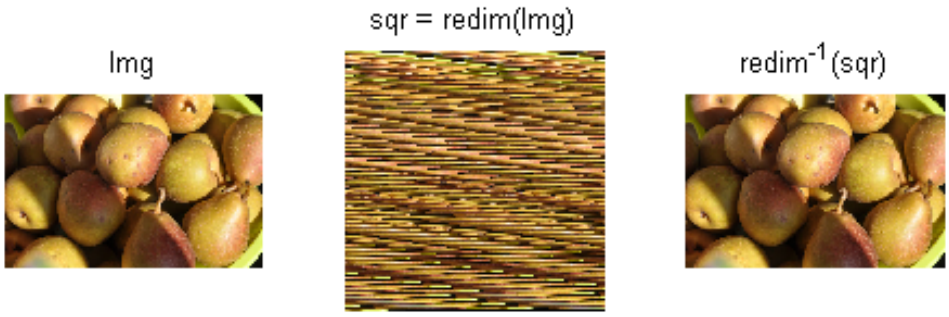

Figura 9: Prueba, métodos de redimensión de imágenes

Fase de sustitución

Para la fase de sustitución Fu Chong y sus colaboradores[5] implementaron el mapa logístico, sin embargo, teniendo en cuenta que para que el mapa se comporte de forma caótica es necesario que uno de los parámetros se encuentre dentro de un rango de valores específico, además es necesario comprobar la efectividad de este valor —es decir que hay valores que funcionan o dan un comportamiento más caótico que otros— y para esto se ocupa el *exponente de Lyapunov*, se decidió analizarlo mejor. Para evitarnos el uso de este exponente cada vez que tengamos que elegir una llave secreta y porque además el parámetro es restrictivo y resulta en un menor rango para el espacio de llaves posible —lo que debilita el criptosistema simétrico— se eligió un mapa caótico diferente que no presentara este detalle, por eso se implementó *el mapa de Lorenz* para generar un flujo de claves (números pseudo aleatorios) y combinarlos con los valores de los pixeles.

Mapa de Lorenz

En 1963 el matemático y meteorólogo estadounidense Edward Norton Lorenz construyó un modelo matemático muy simplificado que simulaba el fenómeno de convección en la atmósfera, como en esa época ya era común el

cálculo numérico en computadoras digitales Lorenz probó el sistema en éstas. Después de algunos experimentos notó que el mínimo cambio en las condiciones iniciales del sistema —como el de utilizar menos puntos decimales en los parámetros— hacen que el sistema a largo plazo sea totalmente diferente, así, con este sistema, Lorenz introdujo los primeros conceptos de *Atractor* y *Efecto mariposa*[26].

El mapa de Lorenz es descrito por una ecuación diferencial de tres variables en función del tiempo —teniendo una proyección sobre el plano de tres dimensiones y formando una trayectoria—, esta ecuación es descrita por:

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x) \\ \frac{dy}{dt} &= rx - rz - y \\ \frac{dz}{dt} &= xy - bz \end{aligned} \tag{5}$$

donde σ, r y b son parámetros y x, y y z son las coordenadas del sistema en el estado anterior, por lo cual son parte de las condiciones iniciales del sistema y deben ser inicializadas.

Para la implementación, Steffi y Sharma —quienes implementaron ya este sistema en su trabajo titulado *"Un Algoritmo de Encriptación de imágenes basado en el mapa 3D de Lorenz"*[25]— afirman que la trayectoria puede ser trazada usando el algoritmo de cuarto orden de Runge-Kutta —un algoritmo de análisis numérico el cual es un refinamiento del método de Euler para solución numérica de ecuaciones diferenciales—. Este algoritmo puede describirse como n iteraciones de las siguientes expresiones:

$$\begin{aligned} k_1 &= \phi(t_i, Y_i) \cdot h \\ k_2 &= \phi(t_i + \frac{h}{2}, Y_i + \frac{k_1}{2}) \cdot h \\ k_3 &= \phi(t_i + \frac{h}{2}, Y_i + \frac{k_2}{2}) \cdot h \\ k_4 &= \phi(t_i + h, Y_i + k_3) \cdot h \\ Y_{i+1} &= Y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned} \tag{6}$$

donde $\phi(t, Y)$ es la función que se desea resolver —en este caso será el sistema de Lorenz— t_i es el valor de tiempo en el instante i, Y_i es el valor

actual de los parámetros de nuestra función —en este caso como se trata con un sistema de tres ecuaciones, este valor es un vector o tripleta de valores— por ende Y_{i+1} es el nuevo valor de estos parámetros para el estado posterior y finalmente h es la magnitud de paso del parámetro tiempo (t) en cada iteración.

Al probar el sistema de Lorenz con este método y con los siguientes valores de parámetros:

$$x_0 = 10 \quad y_0 = 20 \quad z_0 = 30 \quad \sigma = 10$$

$$r = 28 \quad b = \frac{8}{3} \quad n = 60000 \quad h = 0{,}01$$

donde —para el método implementado de Runge-Kutta— n es el número de instantes que se quieren graficar y h es el tamaño de paso para ir de un instante a otro, se obtienen los valores necesarios para graficar la trayectoria mostrada en la *Figura 10*.

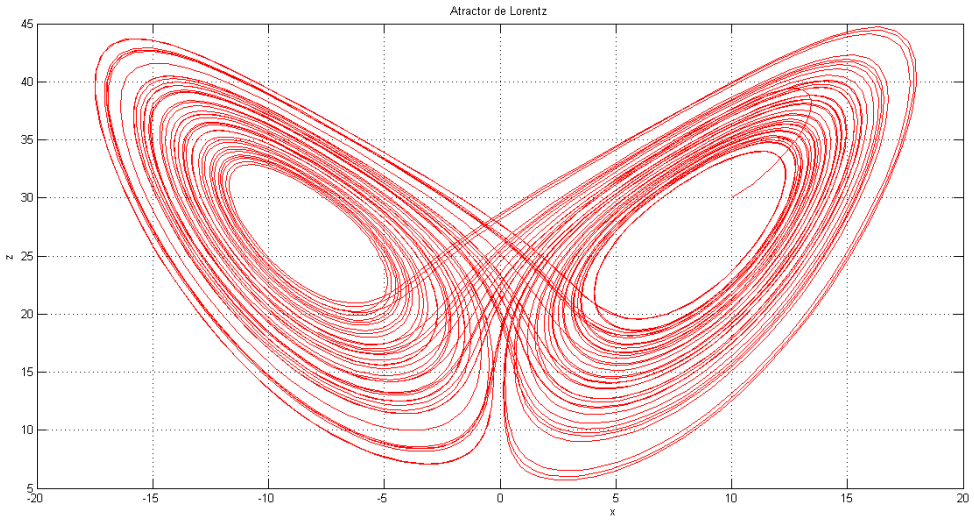

Figura 10: Sistema de Lorenz resuelto con el método Runge-Kutta de 4to. orden

No obstante esta trayectoria, Steffi y Sharma afirman que se obtienen mejores valores pseudo-aletorios cuando se cambia el valor de h a 0,1 ya que el sistema se extiende un poco más con un paso de tiempo más grande y los valores de las tres variables resultan más variadas, como puede observarse en

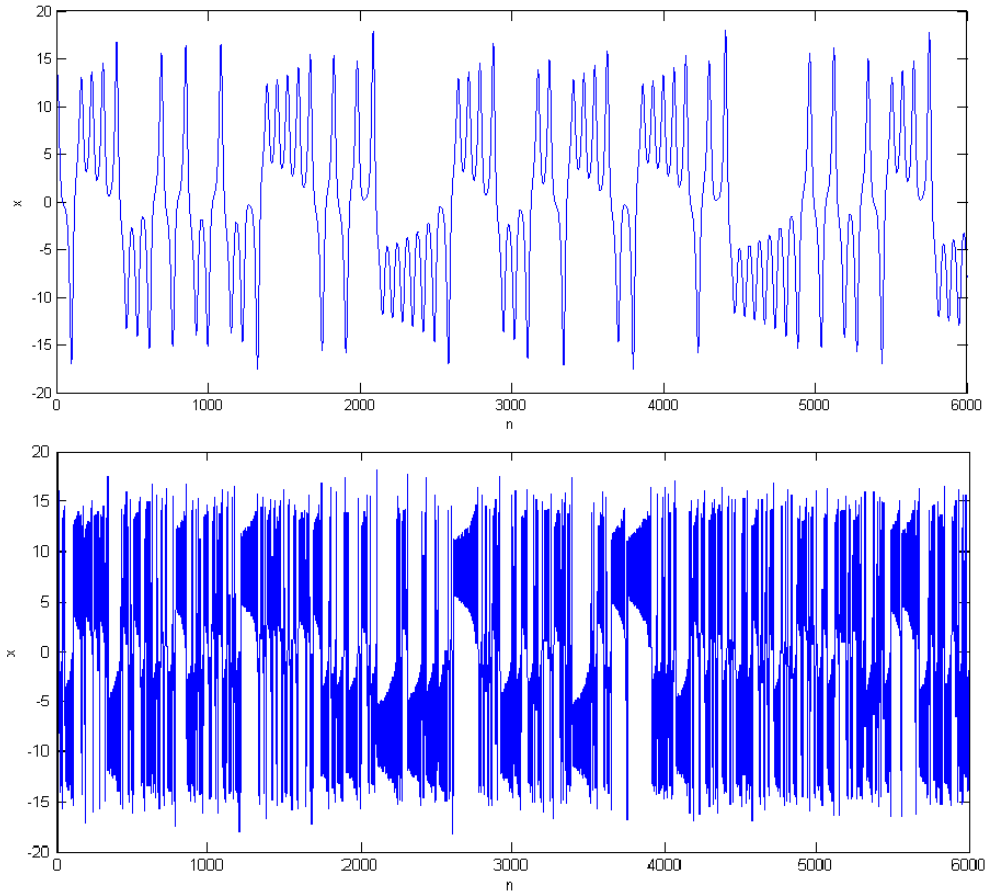

Figura 11: Variaciones de valores de x en el sistema de Lorenz, arriba con h=0,01, abajo con h=0,1

las gráficas de la *Figura 11*, cuando h=0,1 la gráfica presenta menos espacios blancos.

Con esta generación de valores pseudo-aleatorios se tiene un flujo de llaves que se usará para la sustitución de pixeles con operaciónes básicas de XOR, en esta propuesta, el método de sustitución sigue el siguiente algoritmo:

```
1. Inicializar todos los parametros y valores iniciales.
2. Obtener con el metodo de Runge-Kutta la secuencia de
valores del sistema de Lorenz X [n] [3].
3. Generar un numero k y un valor de pixel inicial (rango
entre 0 y 255) cn aleatorios (llaves de sustitucion).
4. Recorrer la matriz de la imagen de forma secuencial y para
cada pixel hacer:
```

```
    4.1. xn <-- X [k] [1]; //k-esimo valor, secuencia de la
    variable x.
    4.2. kn <-- floor( xn * 10000) mod 255.
    4.3. Cij <-- Pij XOR kn XOR cn.
    4.4. cn <-- Cij.
    4.5. Si k = 6000 entonces:
        4.5.1. k <-- 1.
    4.6. Sino:
        4.6.1. k <--  k + 1.
    4.7. Fin_Para.
5. Fin_ciclos.
6. Devolver las llaves y el criptograma.
7. Fin.
```

Donde *Pij*, *Cij*, *kn* y *cn* son el pixel actual de la imagen, el pixel actual de la imagen cifrada, el elemento del flujo de llaves y el valor del pixel sustituido anterior respectivamente. El procedimiento de descifrado es similar al descrito arriba y la instrucción inversa al de la *línea 4.3* está dada por: $Pij \leftarrow Cij \bigoplus kn \bigoplus cn$.

Implementación

Una vez que definidos y probados los módulos que se han de usar en esta propuesta, falta ensamblar cada una de las partes en el esquema general de encriptación-desencriptación y mostrar las llaves que se usaron cada vez, para describir la manera en que esto fué llevado a cabo se presenta el algoritmo y se muestra el mismo en un diagrama general.

Algoritmo: encriptación

El método de encriptación de nuestra propuesta tiene las siguientes características:

- **Interfaz**: cifra_img(String img[ ]).
- **Entrada**: La ruta de la imagen con nombre y extensión de la imagen a cifrar.
- **Salida**: Un vector conteniendo las llaves generadas y la imagen cifrada.

Algoritmo:

```
1. Lee la imagen de la ruta especificada con imread y guardala
en la variable cifrado[ ][ ][3].
2. Obtiene las medidas m y n de la imagen.
3. R <-- 0.
4. Si m <> n entonces:
    4.1 Redimensiona cifrado a una imagen cuadrada y guardala
    en cifrado.
    4.2 Obten las medidas Ls y b de la imagen cuadrada.
    4.3 R <-- Ls^2 - m * n.
    4.4 Fin_Si
5.Inicializa el vector llave[ 38 ].
6. j <-- 1.
7. Para i <-- 1 hasta 2 hacer:
    7.1 Cifra con el mapa del gato la imagen en la variable
    cifrado, resultados: key y cifrado. Integrada 3 iteraciones.
    7.2 Guarda key resultante en el vector llave[ j ] en adelan-
    te.
    7.3 j <-- j + 16.
    7.4 Cifra con el mapa de lorenz la imagen en la variable
    cifrado, resultados: devolviendo key y cifrado.
    7.5 Guarda key resultante en el vector llave[ j ] hasta j+15.
    7.6 j <-- j + 2.
    7.7 Fin_Para.
8. llave[ 37 ] <-- R.
9. llave[ 38 ] <-- n.
10. Solo para b/n. Si R > 0, entonces:
    10.1 Genera valores aleatorios entre 0 y 255  para el canal
     2 de la matriz cifrado[ Ls ][ Ls ][ 2 ].
    10.2 Genera valores aleatorios entre 0 y 255  para el canal
     3 de la matriz cifrado[ Ls ][ Ls ][ 3 ].
11. Si no entonces:
    11.1 Genera valores aleatorios entre 0 y 255  para el canal
    2 de la matriz cifrado[ m ][ n ][ 2 ].
    11.2 Genera valores aleatorios entre 0 y 255  para el canal
    3 de la matriz cifrado[ m ][ n ][ 3 ].
    11.3 Fin_Si.
12. Fin
```

En la *Figura 12* se observa un diagrama donde se describe el algoritmo del sistema de cifrado anterior.

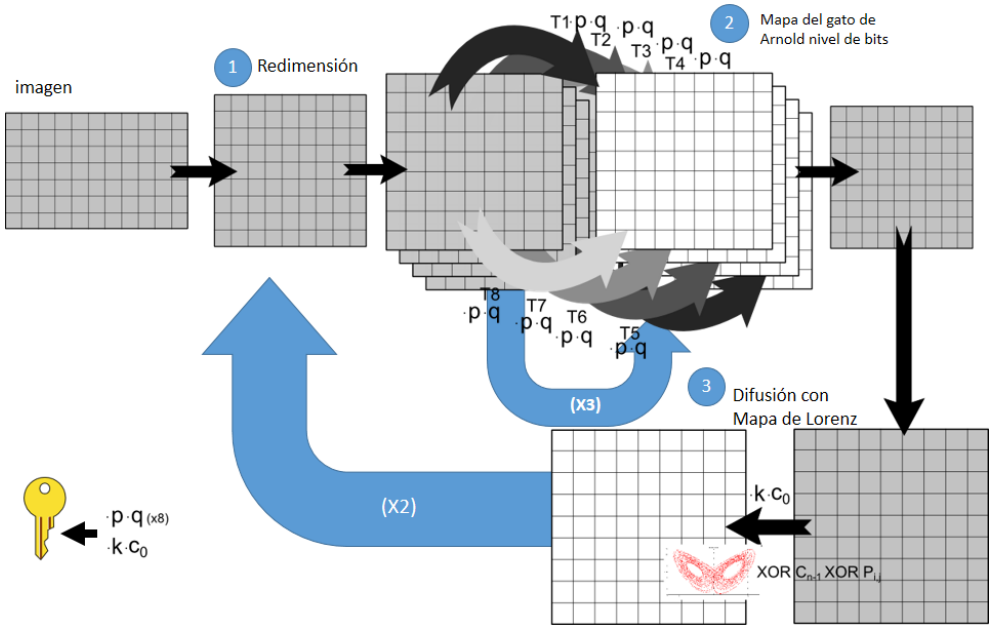

Figura 12: Diagrama general, método de encriptación

Algoritmo: desencriptación

El método de desencriptación de esta propuesta tiene las siguientes características:

- **Interfaz**: descifra_img(Float llave[ ], Uint8 cifrado[ ][ ][ ]).
- **Entrada**: Vector de llaves y la matriz de la imagen cifrada.
- **Salida**: La matriz de la imagen descifrada.

Algoritmo:

```
1. img <-- cifrado.
2. j <-- 36.
3. Para i <-- 2 hasta 1, paso -1, hacer.
    3.1 Obtiene key asignandole llave[ j-1 ] hasta j.
    3.2 j <-- j - 2.
    3.3 Descifra img con mapa de Lorenz usando key, resultado
    en img.
    3.4 Obtiene key asignandole llave[ j-15 ] hasta j.
    3.5 j <-- j - 16.
    3.6 Descifra img con mapa del gato usando key, resultado
```

```
    en img. Integradas 3 iteraciones.
    3.7 Fin_Para
4. R <-- llave[ 37 ].
5. n <-- llave[ 38 ].
6. Si R > 0 entonces:
    6.1 Redimension inversa a img con R y n, resultado en img
    6.2 Fin_si.
7. Solo b/n, img[ ][ ][ 2 ] <-- img[ ][ ][ 1 ].
8. Solo b/n, img[ ][ ][ 3 ] <-- img[ ][ ][ 1 ].
9. Fin
```

En la *Figura 13* se observa el diagrama donde se describe el algoritmo del sistema de descifrado anterior con los métodos inversos de las funciones.

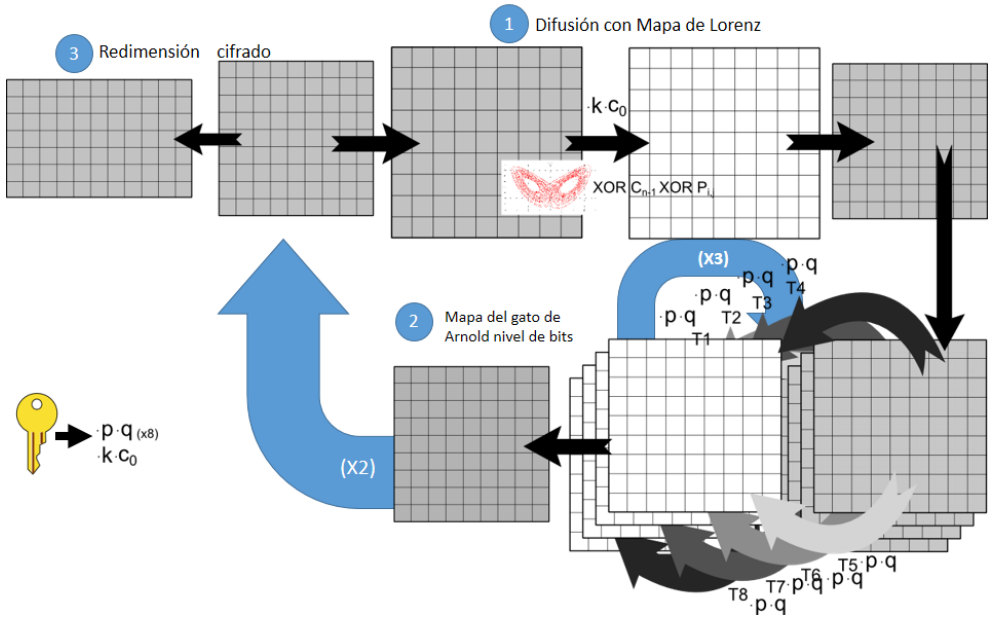

Figura 13: Diagrama general, método de desencriptación

PRUEBAS

Varias pruebas de cifrado-descifrado fueron aplicadas en un ordenador con las siguientes características:

- Procesador Intel Core i5 – 3.00 GHz.
- Memoria RAM DDR3 – 6GB.
- Sistema Operativo Windows 7 (x64).

Las imágenes seleccionadas para estas pruebas se pensaron que fueran de distintas dimensiones y formas (cuadradas y rectangulares) para mostrar el rendimiento de los algoritmos de cifrado/descifrado. Además se realizaron pruebas para medir la integridad de las imágenes al descifrar, por lo tanto se eligieron imágenes con grandes dimensiones y un alto nivel de detalle, por ejemplo, las imágenes médicas de leucocitos que hicieron posible detectar facilmente errores a simple vista al descifrarse, sin embargo se realizaron algunas pruebas alternas para estos fines, efectuando una resta de matrices entre la imágen cifrada y la original y revisando que el resultado sea siempre una matriz de "ceros".

La *Figura 14* muestra los resultados de las pruebas del cifrado de imágenes a escala de grises, los cuales son los esperados por la apariencia de las imágenes cifradas donde no se reconoce ningún patrón y tiene cierta aleatoriedad, además, cabe destacar que las imágenes son convertidas a color en el cifrado (eligiendo valores aleatorios en los otros canales) para hacer más complicado romper la seguridad, esto es, al observarse parece que la imágen original es a color, por lo tanto al descifrarse incluso teniendo las llaves apropiadas, el resultado podría ser una imágen sin sentido, de esta manera hay un plus en la seguridad de las imágenes a escala de grises. Mientras el *Cuadro 1* muestra sus respectivos tiempos de ejecución, estos serán comparados con el de otro método de cifrado clásico para medir la velocidad de la propuesta.

La *Figura 15* muestra las pruebas en imágenes a color y el *Cuadro 2* los tiempos de ejecución en esa prueba. Al igual que las pruebas en escala de grises, los resultados de las pruebas en imágenes a color fueron satisfactorios

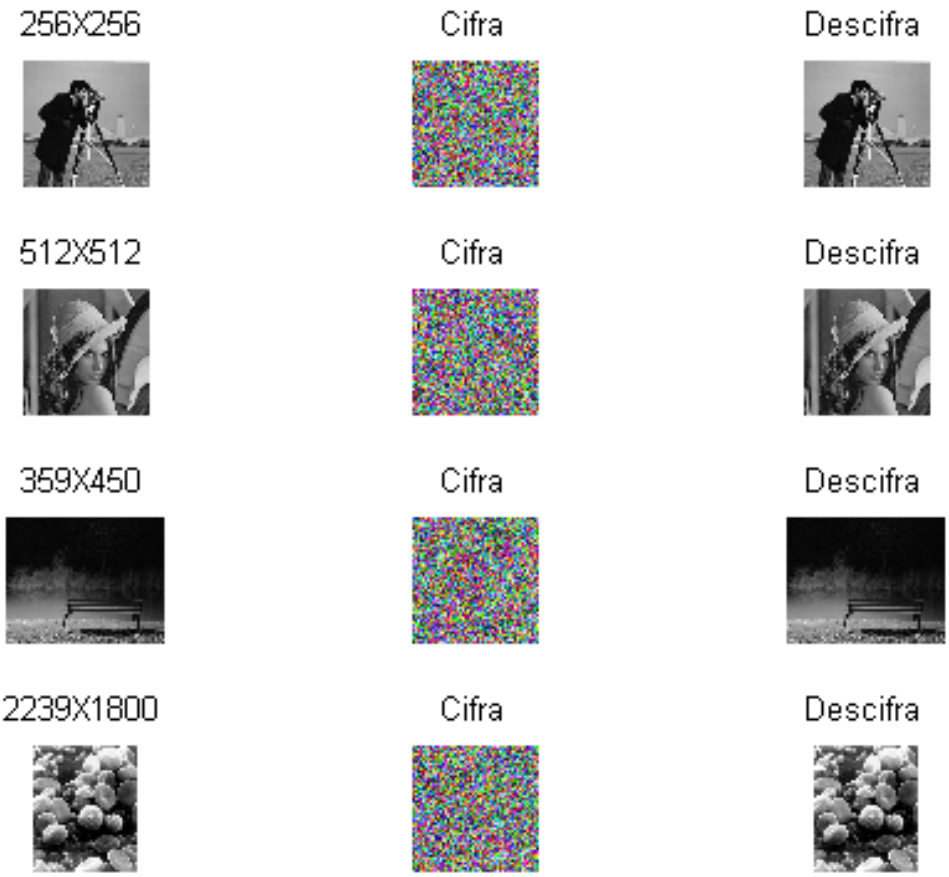

Figura 14: Pruebas en imágenes a escala de grises

y son congruentes con los esperados antes de la ejecución. Cabe mencionar, en cuanto al cuadro de tiempos, que al procesar y manipular imágenes digitales a color se trabajan con tres canales lo que tomará dos veces más del tiempo que con las de escala de grises, donde los datos trascendentes son obtenidos de un solo canal.

Finalmente se adaptó el algoritmo estándar de encriptación de datos DES para trabajar con imágenes digitales, el cual es implementación propia de uno de los cursos de la carrera de ciencias de la computación y con esto fueron realizadas pruebas sobre las mismas imágenes a escala de grises y de esta forma comparar los tiempos de ejecución de ambos algoritmos con el mismo volúmen de datos, en el *Cuadro 3* y la *Figura 16* se muestran los resultados de las pruebas con el algoritmo DES.

Resolución	Bytes	T. cifrado (seg.)	T. descifrado (seg.)
256 X 256	65536	11.502	11.434
512 X 512	262144	45.557	45.470
394 X 450	161550	28.033	28.024
2239 X 1800	4030200	692.863	697.911

Cuadro 1: Tiempos de ejecución de las imágenes a escala de grises

Resolución	Bytes	T. cifrado (seg.)	T. descifrado (seg.)
450X450	202500	44.524	43.212
287X432	123984	26.426	26.991
1024X1024	1048576	227.712	221.576
1080X1920	2073600	439.809	438.255

Cuadro 2: Tiempos de ejecución de las imágenes a color

Además de estos cuadros que muestran los tiempos, se realizó la graficación de los datos de ambos algoritmos donde se aprecia la abrumadora diferencia de los tiempos de las pruebas de los algoritmos de la propuesta y los del DES, Ver *Figura 17* y *18*.

Durante las ejecuciones de las pruebas fué notable la diferencia de los tiempos entre el algoritmo de nuestra propuesta y el DES y al analizar la información obtenida se llegó a concluír que el DES, un algoritmo estándar de la criptografía simétrica, tarda en promedio 12 veces más en ejecutarse que nuestra propuesta, esto comparando el *Cuadro 1* y *3*. Esto siendo prueba concluyente de que los algoritmos basados en caos proveen métodos rápidos, seguros y con mejor desempeño con los grandes volúmenes de datos de las imágenes digitales que lo que el estándar de la criptografía simétrica ofrece.

Resolución	Bytes	T. cifrado (seg.)	T. descifrado (seg.)
256 X 256	65536	135.604	137.216
512 X 512	262144	543.705	549.199
394 X 450	161550	335.028	457.473
2239 X 1800	4030200	8654.614	8669.690

Cuadro 3: Tiempos de ejecución de las imágenes a escala de grises con el algoritmo DES

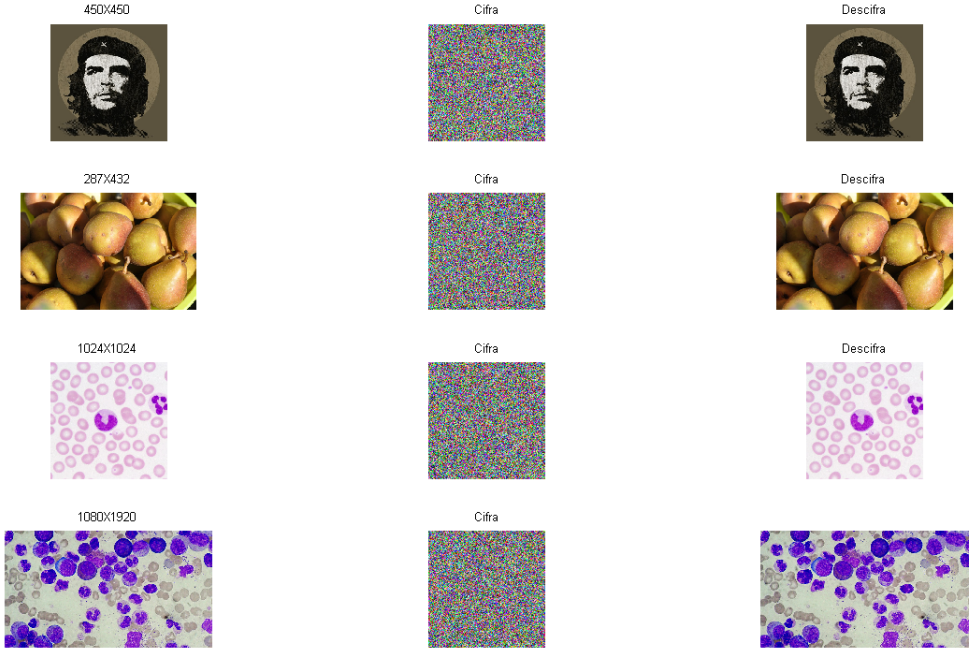

Figura 15: Pruebas en imagenes a color

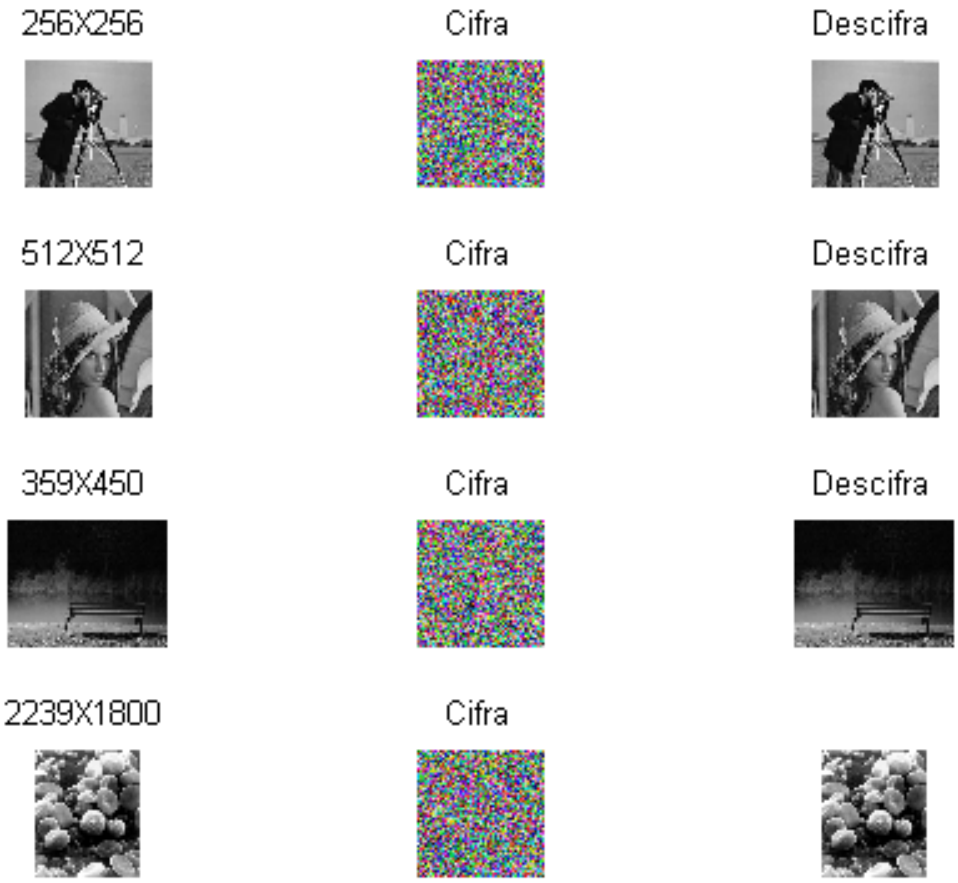

Figura 16: Pruebas del algoritmo DES sobre imágenes a escala de grises

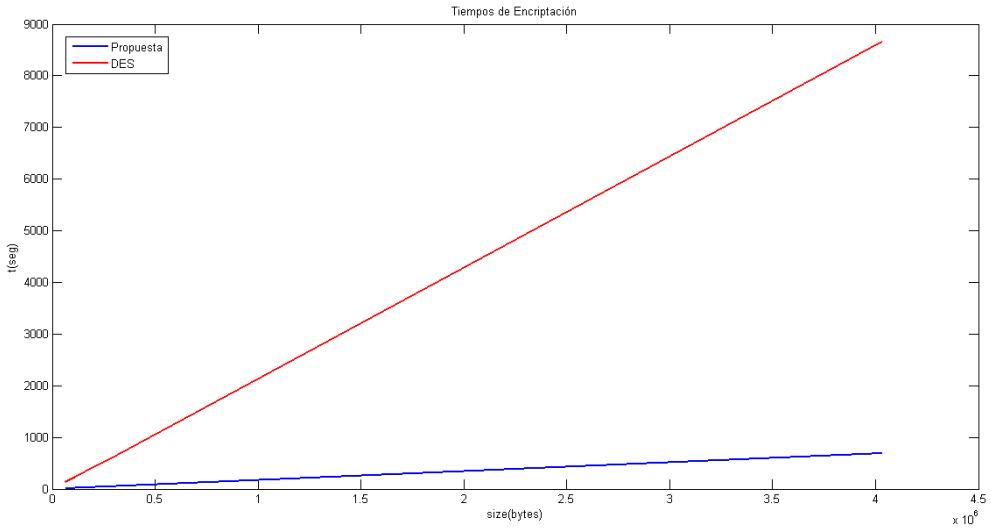

Figura 17: Gráfica comparativa, algoritmos de cifrado de DES y propuesta

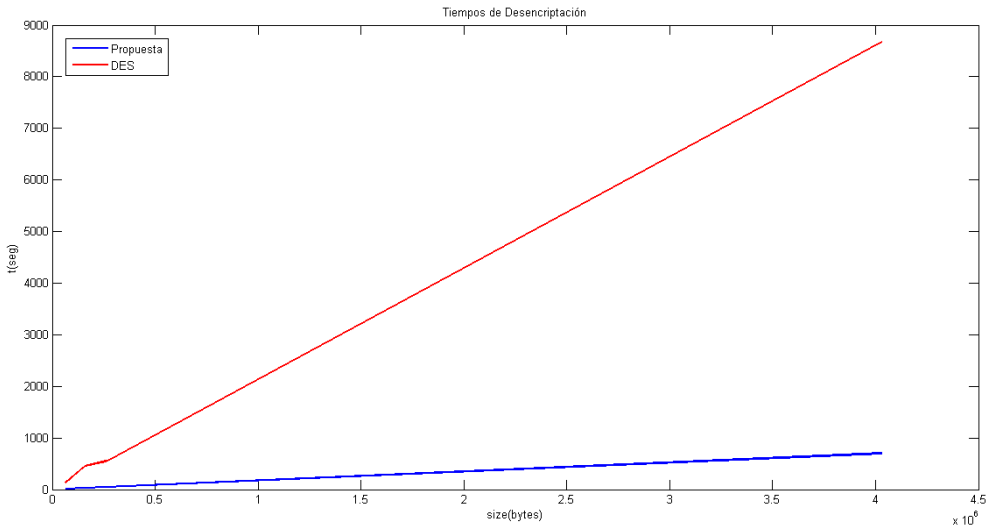

Figura 18: Gráfica comparativa, algoritmos de descifrado de DES y propuesta

CONCLUSIONES

Al finalizar este trabajo se concluye que se requirieron conocimientos en varias áreas de ciencia y tecnología y este es un trabajo donde estos conocimientos convergen para un fin específico. Áreas de las ciencias de la computación como la criptografía, el procesamiento digital de imágenes, las matemáticas discretas, la programación y los métodos numéricos, y otras que están fuertemente relacionadas con las matemáticas y la física como el álgebra lineal, las ecuaciones diferenciales y el estudio de sistemas dinámicos caóticos, permitiendo que el resultado sean sistemas de seguridad de datos cada vez más sofisticados y aportando a la demanda de nuevos esquemas de seguridad.

En cuanto a la propuesta, el utilizar dos diferentes mapas caóticos con la arquitectura de permutación-sustitución permite tener una colección de valores (llaves) requiriendo que cada valor sea exactamente el mismo al descifrar, lo que brinda seguridad al sistema de encriptación simétrico ya que un cambio de valor en las condiciones iniciales de los sistemas caóticos produce un conjunto de valores finales completamente diferente. Mientras el uso del mapa del gato de Arnold a nivel de bits introduce un cambio de valor a la fase de permutación beneficiando así a la aleatoriedad del producto final en cada iteración, pero también a que se pueda iterar menos veces el esquema general y así, disminuir el tiempo necesario para obtener el criptograma sin comprometer su seguridad.

Por otro lado, al comparar el esquema con aquel propuesto por Fu, Meng, Zhan y sus colaboradores [5] se puede afirmar que al usar el sistema de Lorenz se ha expandido el espacio de llaves, mejorando la seguridad del criptosistema simétrico sin comprometer la velocidad que los algoritmos caóticos prometen ya que se presume ambas propuestas tengan una velocidad muy parecida. Desafortunadamente no hubo forma de hacer una comparación de rendimiento objetiva entre estas dos propuestas ya que no se cuenta con la arquitectura con la cual se realizaron las pruebas de ese otro trabajo.

Y por último se pudo comparar uno de los algoritmos simétricos estándar

más seguros, el Data Encryption Standard (DES), con nuestra propuesta en cuestión de tiempos y al analizar los datos se llega a la conclusión que en promedio nuestra propuesta es 12 veces más rápida que el DES. De esta manera se hizo evidente la ventaja en la rapidez que los algoritmos basados en caos tienen sobre los estándares de la criptografía simétrica.

Con todo esto se concluye con que se ha cumplido con el objetivo del trabajo de tésis que se planteó en un principio, ya que estudiamos los algoritmos que incorporan mapas o funciones caóticos en sus métodos, se propuso una mejora a un esquema en específico y se probó que la velocidad es, efectivamente, mayor que los algoritmos estándar de la criptografía simétrica. Para trabajos posteriores, se propone dejar de lado el rendimiento y la rapidez y aplicar un estudio a fondo sobre la aleatoriedad del sistema y la seguridad concreta del sistema efectuando un estudio sobre el conjunto de llaves.

Bibliografía

[1] Goldreich, Oded. Modern Cryptography, Probabilistic Proofs and Pseudo-randomness. En *Algorithms and Combinatorics (17)*. Springer-Verlag 1999.

[2] Stinson, Douglas. Cryptography, Theory and Practice. En *Discrete Mathematics and its applications*. 3ra. Edición, Chapman & Hall 2006.

[3] Galende, Juan C. Criptografia, Historia de la Escritura Cifrada. 1ra. Edición, Editorial Complutense, 1995.

[4] Aguillón, Erika (2012). Fundamentos de Criptografía. Laboratorio de Redes y Seguridad, Universidad Nacional Autónoma de México. Recuperado de: http://redyseguridad.fi-p.unam.mx/proyectos/criptografia/criptografia/

[5] Fu Chong, Meng Wei-hong, Zhan Yong-feng, Zhu Zhi-liang, Lau Francis, Tse Chi y Ma Hong-feng. An efficient and secure medical image protection scheme based on chaotic maps. En *Computers in Biology and Medicine. Elsevier Ltd. vol. 43*, (2013), pp. 1000-1010.

[6] Kerckhoffs, Auguste. La Cryptographie Militaire. En *Journal des sciences militaires*, vol. IX, pp. 5–83, Enero 1883, pp 161–191, Febrero 1883.

[7] Loshin, Pete. Personal Encryption, Clearly Explained. Academic Press Professional, 1998.

[8] Ott, Edward. Chaos in Dynamical Systems. Cambridge University Press, 1993.

[9] Hilborn, Robert C. Chaos and Nonliear Dynamics. 2da Edición, Oxford Univesity Press, 2000.

[10] Jimenez, Mario y Palomar, Rafael. Criptografía Caótica. GoogleCode 12 de Abril, 2008. Recuperado de: https://cripcaos.googlecode.com/svn/trunk/cripcaos.pdf

[11] Tocci, R.J. Digital Systems: Principles and Applications. 10ma Edición, Prentice-Hall, 1980.

[12] Mandado, Enrique. Sitemas Electrónicos Digitales. 7ma. Edición, Alfaomega, 1998.

[13] Fridrich, J. Symmetric cyphers based on two-dimensional chaotic maps. En *Int. J. Bifurcation Chaos 8 (6)*, (1998) 1259-1284.

[14] Wong K.W., Kwok B.S.H. y Law W.S. A fast image encryption scheme based on chaotic standard map. En *Phys. Lett. A. 372 (15)*, (2008) 2645-2652.

[15] Gao T.G. y Chen Z.Q. Image encryption based on a new total shuffling algorithm. En *Chaos, Solitons & Fractals 38 (1)*, (2008) pp. 213-220

[16] Sun F.Y., Liu S.T. y Li Z.Q. et al. A novel image encryption scheme based on spatial chaos map. En *Chaos Solitons & Fractals 38 (3)*, (2008) 631-640.

[17] Xu S.J., Wang J.Z., Yang S.X. An improved image encryption algorithm based on chaotic maps. En *Chin. Phys. B 17 (11)*, (2008) 4027-4032.

[18] Ye G.D. A chaotic image cryptosystem based on Töplitz and Hankel matrices. En *Imaging Sci. J. 57 (5)*, (2009) 266-273.

[19] Tong X.J., Cui M.G. Image encryption scheme based on 3D baker with dynamical compound chaotic sequence cipher generator. En *Signal Processing 89 (4)*, (2009) 480-491.

[20] Huang C.K. y Nien H.H. Multi chaotic systems based pixel shuffle for image encryption. En *Opt. Commun. 282 (11)*, (2009) pp. 2123-2127.

[21] Patidar V., Pareek N.K. y Sud K.K. A new substitution-diffusion based image cipher using chaotic standard and logistic maps. En *Commun. Nonlinear Sci. Numer. Simulation 14 (7)*, (2009) pp. 3056-3075.

[22] Rhouma R., Meherzi S., Belghith S. OCML-based colour image encryption. En *Chaos, Solitons & Fractals 40 (1)*, (2009) pp. 309-318.

[23] Xiao Y.L. y Xia L.M. An Image Encryption Approach Using a Shuffling Map. En *Commun. Theor. Phys. 52 (5)*, (2009) 876-880.

[24] Peterson, Gabriel. Arnold's Cat Map. Cornell University Department of Physics, 1997. Recuperado de: http://pages.physics.cornell.edu/šethna/teaching/562_S03/HW/pset02_dir/catmap.pdf

[25] Steffi, Anto y Sharma, Dipesh. An Image Encryption Algorithm based on 3D Lorenz map. En *International Journal of Advanced Research in Computer Science 4 (2)*, (2013) 312-316.

[26] Camacho, Luis. Breve Introducción a los atractores caóticos. Grupo Docente UCO-050 Dpto. Quimica Fisica y Termodinámica Aplicada, Universidad de Córdoba. Recuperado de: http://www.uco.es/dptos/quimica-fisica/quimica-fisica/MC/QC2.htm