UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD DE CIENCIAS PURAS Y NATURALES

CARRERA DE INFORMÁTICA

TESIS DE GRADO

ENCRIPTACIÓN DE IMÁGENES APLICANDO MAPAS CAÓTICOS

PARA OPTAR AL TÍTULO DE LICENCIATURA EN INFORMÁTICA

MENCIÓN: INGENIERÍA DE SISTEMAS INFORMÁTICOS

POSTULANTE: MIGUEL ÁNGEL PATZI MAMANI

TUTOR METODOLÓGICO: M. SC. EDGAR PALMIRO CLAVIJO CARDENAS

ASESOR: LIC. JAVIER REYES PACHECO PH. D.

LA PAZ – BOLIVIA

2017

UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

LA CARRERA DE INFORMÁTICA DE LA FACULTAD DE CIENCIAS PURAS Y NATURALES PERTENECIENTE A LA UNIVERSIDAD MAYOR DE SAN ANDRÉS AUTORIZA EL USO DE LA INFORMACIÓN CONTENIDA EN ESTE DOCUMENTO SI LOS PROPÓSITOS SON ESTRICTAMENTE ACADÉMICOS.

LICENCIA DE USO

El usuario está autorizado a:

a) visualizar el documento mediante el uso de un ordenador o dispositivo móvil.
b) copiar, almacenar o imprimir si ha de ser de uso exclusivamente personal y privado.
c) copiar textualmente parte(s) de su contenido mencionando la fuente y/o haciendo la referencia correspondiente respetando normas de redacción e investigación.

El usuario no puede publicar, distribuir o realizar emisión o exhibición alguna de este material, sin la autorización correspondiente.

Dedicatoria

A mi madre y mis hermanos, que nunca dejaron de creer en mí.

AGRADECIMIENTO

A la Universidad Mayor de San Andrés por haberme acogido a lo largo de esta carrera brindándome oportunidades para mejorar académica y personalmente.

Agradezco además a todos los docentes de la carrera de Informática, de manera especial a M. Sc. Edgar Palmiro Clavijo Cárdenas y Lic. Javier Reyes Pacheco, sin su ayuda nada de esto hubiera sido posible.

Finalmente agradezco a mi madre Clara y mis hermanos Silvia, Ximena y Junior por ser ese faro de luz que ha iluminado mi camino durante esta hermosa travesía.

Tabla de contenido

Tabla de ilustraciones

Resumen

El presente trabajo titulado “Encriptación de imágenes aplicando mapas caóticos” es claramente un trabajo en el cual se desarrolla un algoritmo de encriptación y otro algoritmo de desencriptación, ambos algoritmos siguen las pautas de cualquier otro algoritmo de encriptación, básicamente el espacio en el cual trabaja el mensaje de entrada es transformado mediante procesos matemáticos, algo así como la transformación de una función que está representada en coordenadas cartesianas a coordenadas polares.

Pero este trabajo tiene una gran diferencia con los algoritmos de encriptación tradicionales, esta diferencia reside en la aplicación de la teoría del caos, en la teoría del caos se considera que “el aleteo de una mariposa puede provocar un tornado”, esta alegoría claramente hace referencia a la sensibilidad extrema a las condiciones iniciales, dicha propiedad es fundamental en la criptografía.

El caos nos introduce a los sistemas dinámicos que son conjuntos que representan los estados posibles del sistema, estos sistemas pueden ser continuos o discretos en el tiempo, pero para el desarrollo del presente trabajo se prefiere el uso de sistemas de tiempo discreto que son denominados mapas, existe una gran diversidad de mapas pero se prefiere para este trabajo mapas unidimensionales, los cuales son pieza fundamental para realizar la transformación del mensaje a ser encriptado.

El simple hecho de aplicar esta transformación a la imagen no garantiza que los tiempos de procesamiento mejoren y es debido a que es necesario abstraer la imagen para que resulte más fácil de manejar, esta abstracción permite ver a la imagen más como una matriz, al tener una matriz abstraída a partir de una imagen es más fácil aplicar la teoría matricial que nos permite utilizar generadores basados en mapas caóticos.

De acuerdo a todo lo expuesto con anterioridad, el rendimiento de la encriptación mejora considerablemente respecto a los algoritmos tradicionales, y la seguridad de la encriptación se incrementa por las características mismas de los mapas caóticos que tienen un comportamiento complejamente inestable pero a la vez determinístico.

Palabras clave: Caos, mapas caóticos, encriptación, cifrado, imagen.

1. Introducción

El acelerado desarrollo de las redes de comunicación de datos provoca un crecimiento en la difusión de datos, que eran comúnmente archivos de texto plano o imágenes de baja resolución, pero en la actualidad estos pueden ser archivos multimedia, como video, imagen y sonido. En muchos casos, las redes de comunicación de datos utilizan señales abiertas susceptibles de ser escudriñadas por intrusos y por ello estas redes requieren de mecanismos que permita tener la certeza de brindar seguridad y confidencialidad a nuestros archivos y para esto es necesario encriptarlos.

Los archivos multimedia contienen volúmenes considerables de datos debido a la relación directamente proporcional que existe entre resolución y espacio de almacenamiento, este aumento en el tamaño del archivo multimedia acrecienta el tiempo de procesamiento para la encriptación, como una de las soluciones a esta problemática se propone la encriptación basada en la teoría del caos para reducir los tiempos de procesamiento y mejorar los niveles de seguridad.

1.1. Antecedentes

La información procesada electrónicamente es transmitida, en la mayoría de los casos, a través de canales de comunicación públicos. La criptografía moderna coadyuva a lograr la confidencialidad de esta información.

La criptografía aplicada a las comunicaciones se estaba desarrollando de forma progresiva en la década de los 60 y culmina aproximadamente una década después con la publicación del DES (Data Encryption Standard). Después, a mediados de los 70 nace la criptografía pública, con lo cual pasa a ser una disciplina académica.

El descubrimiento de la sincronización del caos ayuda a combinar la teoría del caos con la encriptación, debido al carácter aleatorio de sus propiedades estadísticas con el que aportan los sistemas caóticos.

Basados en los descubrimientos, descritos anteriormente, se ha desarrollado una variedad de algoritmos que a continuación serán presentados de manera resumida, además de investigaciones sobre criptografía caótica.

- El artículo titulado *"Use of chaotic dynamical systems in cryptography"* (Schmitz, 2001), trata sobre las propiedades matemáticas relevantes para el uso de sistemas dinámicos caóticos en criptografía. Se evalúan estas propiedades para algunos de los sistemas propuestos en la literatura y se explica las consecuencias para el nivel de seguridad ofrecido por estos sistemas.
- En el artículo, "Criptografía caótica con reinyección de la información" (Millérioux, Hernández, & Amigó, 2006), se hace un estudio de los métodos de encriptación relacionados con la teoría del caos, que se han propuesto desde finales de los años 80. Algunos de ellos consisten en mezclar o enmascarar la información en el transmisor con ruido caótico determinista. Posteriormente se hace una comparación de la criptografía convencional con la criptografía caótica.
- El esquema de encriptación de una imagen propuesto por Pareek, Patidar y Sud; utilizan una clave secreta externa de 80 bits y son empleados dos mapas logísticos caóticos. Las condiciones iniciales para ambos mapas logísticos son derivados de la clave secreta externa para proveer diferentes valores a todos los bits. En el proceso de encriptación que proponen Pareek, Patidar y Sud, ocho diferentes tipos de operaciones son usados para encriptar los pixeles de una imagen, después uno de ellos es usado para un pixel particular, el cual es el resultado del mapa logístico. Para hacer que el algoritmo de cifrado sea más robusto contra cualquier ataque, la clave secreta externa es modificada después de encriptar cada bloque de dieciséis pixeles de la imagen (Pareek, Patidar, & Sud, 2006).
- Este otro algoritmo de encriptación de imágenes basado en difusión y mapas caóticos múltiples creado por Sathishkumar, Bhoopathy bagan y Sriraam; proveen una técnica de encriptación de imagen asegurada usando múltiple mapeo circular basado en caos. Primero, un par de subclaves es dada para usarlos en mapas logísticos caóticos. Segundo, la imagen es encriptada usando la subclave del mapa logístico y en su transformación pasa al proceso de difusión. Tercero, las subclaves son generadas por cuatro diferentes mapas caóticos. Basado en las condiciones iniciales, cada mapa puede generar varios números aleatorios desde varias orbitas de los mapas. De todos estos números aleatorios, un número en y

orbita en particular son seleccionados como clave para el algoritmo de encriptación (Sathishkumar, Bhoopathy bagan, & Sriraam, 2011).

- Por último en “Modelo de encriptación simétrica basada en atractores caóticos” (Moreno, Parra, Huérfano, Suárez, & Amaya, 2016), presenta un modelo de encriptación simétrico extensible para comunicaciones digitales, aprovechando el caos generado por sistemas dinámicos lineales. El modelo desarrollado es capaz de encriptar mensajes en tiempos de sincronización, encriptación y desencriptación pequeñas con una entropía superior a seis usando el atractor de Rössler.

1.2. Identificación del problema

Las imágenes de dimensiones grandes o alta resolución, y los algoritmos de encriptación cada vez más complejos elevan el tiempo de encriptación, lo que a su vez provoca que las imágenes, muchas veces, no estén disponibles. Esta baja disponibilidad hace que la confianza, en los sistemas de encriptación, baje o que el tiempo dedicado a obtener las imágenes aumente y por consiguiente los costos por el servicio, o en otros casos esta baja disponibilidad incita al uso de imágenes sin encriptar, poniendo en riesgo la privacidad.

1.3. Planteamiento de la hipótesis

El algoritmo de encriptación de imágenes ICT (Imágenes Caóticamente Transformadas) basado en sistemas dinámicos no lineales que tienden al caos incrementa la velocidad de procesamiento de encriptación de una imagen con respecto a métodos de encriptación tradicionales.

1.4. Objetivos

1.4.1. Objetivo general

Desarrollar un algoritmo de encriptación de imágenes basado en un sistema dinámico no lineal que pretende disminuir el tiempo de procesamiento respecto a métodos de encriptación tradicionales.

1.4.2. Objetivos específicos

- Analizar los fundamentos teóricos de una imagen.
- Analizar los sistemas dinámicos y como estos se comportan de manera caótica.
- Estudiar la encriptación y como esta se beneficia con la introducción del caos, para desarrollar el algoritmo en base a estos estudios.
- Realizar pruebas sobre formatos y dimensiones de imágenes para determinar en qué condiciones mejora el rendimiento del algoritmo.
- Demostrar el comportamiento caótico del algoritmo.
- Realizar pruebas de tiempos de procesamiento de encriptación de imágenes con métodos tradicionales.
- Realizar pruebas de tiempos de procesamiento con el algoritmo de encriptación de imágenes.

1.5. Justificación

1.5.1. Científica

Con el uso de la teoría del caos, considerada como una rama derivada de los sistemas dinámicos no lineales, se pretende aplicar modelos matemáticos que se vuelven caóticos en ciertos intervalos, ya que la encriptación basada en tales teorías proporciona dos propiedades básicas, propios de los sistemas basados en la teoría del caos, los cuales son aleatoriedad y sensibilidad a las condiciones iniciales.

1.5.2. Tecnológica

El algoritmo basado en criptografía caótica ayudara a mejorar la seguridad en las comunicaciones gracias a la naturaleza aleatoria del caos.

1.5.3. Económica

La seguridad en un sistema de comunicación implica mayores ganancias debido a la confianza que genera una buena encriptación.

1.5.4. Social

El algoritmo de encriptación será puesto a disposición de la comunidad investigadora para generar más conciencia respecto a la seguridad con la que debemos manejar nuestros datos, en este caso las imágenes, y también para que el algoritmo sea sujeto a mejoras.

1.6. Límites y alcances

1.6.1. Límites

Para el desarrollo del algoritmo se tomará en cuenta imágenes de dos dimensiones, excluyendo imágenes de tres dimensiones que son utilizadas con mucha frecuencia en ramas afines a la medicina.

1.6.2. Alcances

El tipo de imágenes que será tomado en cuenta para la elaboración del algoritmo serán imágenes a color que son compuestos por los colores básicos *RGB*, cada elemento oscila entre 0 y 255; además el formato de imagen que se utilizará es el estándar de compresión y codificación de archivos e imágenes creada por el Grupo Conjunto de Expertos en Fotografía *JPEG*, además se utilizará el formato de imágenes sin compresión *BPM*.

1.7. Metodología

Para el desarrollo del presente trabajo se utilizará; el método hipotético deductivo para desarrollar el algoritmo de encriptación y para la comprobación se experimentará con los resultados comparándolos con algoritmos que son usados en la actualidad.

1.8. Técnicas y herramientas

Para el desarrollo del algoritmo será necesario contar con la herramienta siguiente:

OpenCV, la cual es una librería para manipular imágenes de diversos formatos, su estructura está compuesta básicamente de cinco componentes.

2. Marco teórico

2.1. Imagen digital

La imagen digital es la representación de imágenes analógicas, estas imágenes digitales están construidas sobre bases matemáticas y probabilísticas.

La imagen digital es el resultado del proceso de digitalización de una imagen analógica.

2.1.1. Clasificación

Las imágenes digitales se clasifican de diversas formas, una de ellas es clasificarlas tomando en cuenta la dimensión:

- Dos dimensiones: Son todas las imágenes representadas en un plano, básicamente hay dos tipos de imágenes en dos dimensiones; imágenes de mapa de bits (véase Ilustración 1) e imágenes vectoriales (véase Ilustración 2) originalmente en formato Windows Metafile, WMF (convertido a PNG.) Se puede comprobar que a la imagen le falta realismo fotográfico en comparación con su equivalente en formato matricial o rasterizado (véase Ilustración 3).

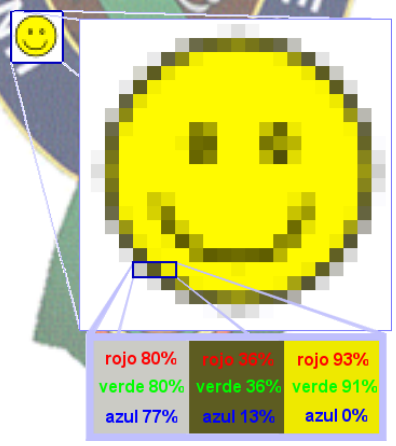

Ilustración 1: Imagen en mapa de bits

Ilustración 2: Locomotora a vapor en formato de imagen vectorial

Ilustración 3: La foto original que fue tomada en un formato matricial JPEG.

- Tres dimensiones: Son imágenes representadas en un sistema de coordenadas espaciales, estas son comúnmente utilizadas en la tecnología médica o en la representación gráfica de figuras volumétricas.

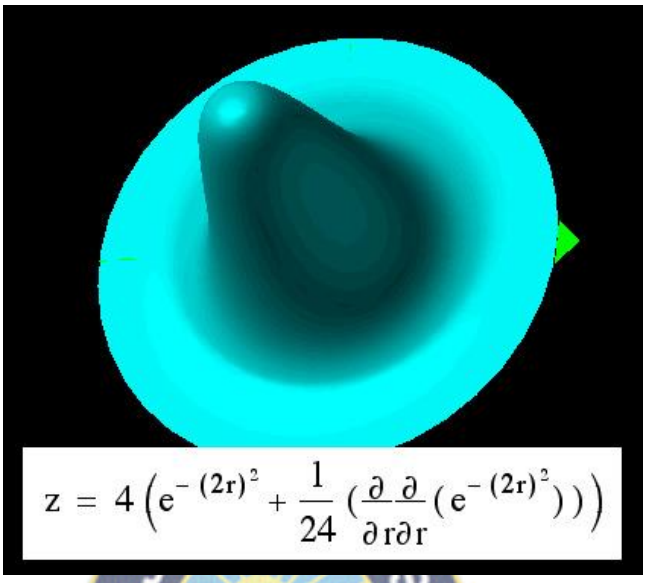

Ilustración 4: Forma tridimensional de una campana de Gauss.

También se clasifican según su origen:

- Imágenes reales: A esta corresponden las imágenes digitalizadas obtenidas de scanners o cualquier otra máquina digital.
- Imágenes sintéticas: Estas imágenes son creadas a partir de programas de edición, tal es el caso de los interferómetros de apertura sintética usan la rotación de la Tierra para incrementar el número de orientaciones de base incluidas en una observación. Aquí, con la Tierra representada como una esfera gris, las línea de base entre el telescopio A y el B cambia el ángulo con el tiempo, como se ve de la fuente de radio al rotar la Tierra. Tomando datos a diferentes tiempos se obtienen mediciones con diferentes separaciones de telescopio.

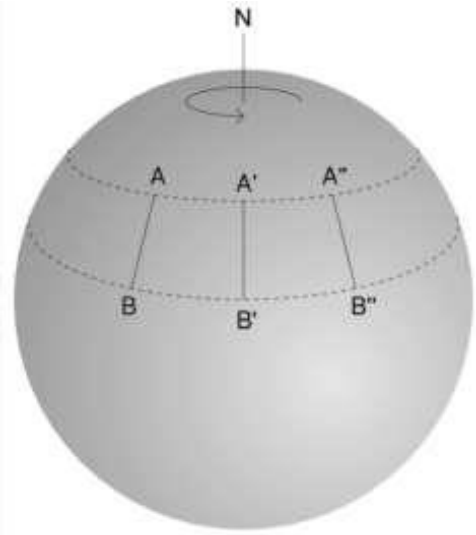

Ilustración 5: Interferómetros de apertura sintética

Según la paleta de colores que utilizan, las imágenes digitales se clasifican en:

- Imágenes binarias: Imágenes cuyos valores de pixel son 0 o 1, estas imágenes también son llamadas monocromas, donde se representan los colores sin escala intermedia.

Ilustración 6: Imagen monocroma.

- Imágenes en escala de grises: Son aquellas imágenes en las que el valor tonal puede oscilar entre 0 y 255 (El valor 0 representa al color negro y el 255 al blanco), es decir, todos los colores contenidos son negros, blancos o una graduación entre los dos.

Ilustración 7: Imagen en escala de grises.

- Imágenes a color: Los pixeles de este tipo de imágenes está compuesto por una paleta de colores denominada RGB (R rojo, G verde, B azul), cada elemento tiene un valor entre 0 y 255, la variación del valor de estos tres elementos produce los diferentes colores para cada pixel.

Ilustración 8: Imagen a color.

2.2. Formato JPEG (Joint Photographic Experts Group)

Grupo Conjunto de Expertos es el nombre de un comité de expertos, conocidos por su sigla en inglés *JPEG*, creadores del estándar de compresión y codificación de archivos e imágenes fijas. Este comité fue integrado desde sus inicios por la fusión de varias agrupaciones en un intento de compartir y desarrollar su experiencia en la digitalización de imágenes.

Además de ser un formato de compresión, es a menudo considerado como un formato de archivo. *JPEG/Exif* es el formato de imagen más común, utilizado por las cámaras

fotográficas digitales y otros dispositivos de captura de imagen, junto con *JPG/JFIF*, que también es otro formato para el almacenamiento y la transmisión de imágenes fotográficas en la World Wide Web. Estas variaciones de formatos a menudo no se distinguen, y se llaman "*JPEG*". Los archivos de este tipo se suelen nombrar con la extensión *.jpg*.

2.2.1. Codificación

Muchas de las opciones del estándar JPEG se usan poco. A continuación se hará una descripción breve de uno de los muchos métodos usados comúnmente para comprimir imágenes cuando se aplican a una imagen de entrada con 24 bits por pixel (ocho por cada uno de los colores: rojo, verde, y azul, o también dicho 8 bits por canal).

2.2.1.1. Transformación del espacio del color

Para transformar el espacio del color, primero se debe convertir la imagen desde su modelo de color RGB a otro llamado YUV o YCbCr. Este espacio de color es similar al que usan los sistemas de color para televisión PAL y NTSC, pero es mucho más parecido al sistema de televisión MAC (Componentes Analógicos Multiplexados).

A continuación se muestran los dos modelos de color:

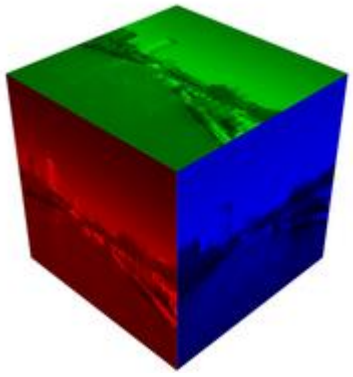

Ilustración 9: Esquema del modelo RGB.

Ilustración 10: Esquema del modelo YUV.

Este espacio de color (YUV) tiene tres componentes:

- La componente Y, o luminancia (información de brillo); es decir, la imagen en escala de grises.
- Las componentes U o Cb y V o Cr, respectivamente diferencia del azul (relativiza la imagen entre azul y rojo) y diferencia del rojo (relativiza la imagen entre verde y rojo); ambas señales son conocidas como crominancia (información de color).

Las ecuaciones que realizan este cambio de base de RGB a YUV son las siguientes:

$$Y = 0{,}257 * R + 0{,}504 * G + 0{,}098 * B + 16$$

$$Cb = U = -0{,}148 * R - 0{,}291 * G + 0{,}439 * B + 128$$

$$Cr = V = 0{,}439 * R - 0{,}368 * G - 0{,}071 * B + 128$$

Estas ecuaciones están en continuo desarrollo, por tal razón los coeficientes varían mas no con mucha variación, otra característica destacable de estas ecuaciones es que los valores individuales resultantes son representados en un *byte*. Durante esta fase no hay pérdida significativa de información, pero debido al redondeo que se realiza existe un pequeño margen de error no perceptible para el ojo humano.

2.2.1.2. Submuestreo

El submuestreo consiste en reducir la información del color, existen varios métodos al respecto. La información cromática puede reducirse a la mitad en dirección horizontal (4:2:2), de esta forma la imagen tiene la mitad de resolución en color y el brillo continua intacto. También puede reducirse el color a la cuarta parte en dirección horizontal y vertical (4:2:0). Si la imagen estuviera en escala de grises puede eliminarse por completo la información del color (4:0:0). Cabe resaltar que una imagen sin submuestreo es (4:4:4).

Las técnicas algorítmicas usadas para su reconstrucción suelen ser interpolación bilineal, vecino más próximo, convolución cúbica, Bezier, b-spline y Catmun-Roll.

2.2.1.3. Transformada discreta de coseno

La transformada discreta de coseno expresa una secuencia finita de varios puntos como resultado de la suma de distintas señales sinusoidales (con distintas frecuencias y amplitudes). Como la transformada discreta de Fourier, la transformada discreta de coseno trabaja con una serie de números finitos, pero solo trabaja con cosenos a diferencia de la transformada discreta de Fourier que lo hace con exponenciales complejos.

Algunas de las características útiles para la compresión de imágenes, son las siguientes:

- La transformada discreta de coseno tiene una buena capacidad de compactación de la energía de dominio transformado, es decir, que la transformada de coseno discreta consigue concentrar la mayor parte de la información en pocos coeficientes transformados.
- La transformación es independiente de los datos. El algoritmo aplicado no varía con los datos que recibe, como si sucede en otros algoritmos de compresión.
- Existen algoritmos equivalentes a la transformada rápida de Fourier.

- Produce pocos errores en los límites de los bloques imagen. La minimización de los errores a los bloques imagen permite reducir el efecto de bloque en las imágenes reconstruidas.
- Tiene una interpretación frecuencial de los componentes transformados. La capacidad de interpretar los coeficientes en el punto de vista frecuencial permite aprovechar al máximo la capacidad de compresión.

Formalmente, la transformada discreta de coseno es una función lineal invertible $f: \mathbb{R}^N \rightarrow \mathbb{R}^N$, o en forma equivalente a una matriz cuadrada de $N \times N$. Esta transformada tiene muchas variaciones debidas a la periodicidad y al tipo de simetría, a continuación se detalla una de las variaciones.

$$f_j = \sum_{k=0}^{n-1} x_k \cos\left[\frac{\pi}{n} j \left(k + \frac{1}{2}\right)\right]$$

Cada componente de la imagen se divide en pequeños bloques de $8x8$ pixeles, que se procesan de forma casi independiente, lo que disminuye notablemente el tiempo de cálculo. De esto resulta la típica formación cuadriculada, que se vuelve visible en las imágenes guardadas con alta compresión. Si la imagen es sometida a un submuestreo del color, los colores quedarían en la imagen final en bloques de 8×16 y 16×16 píxeles, según fuese el caso 4:2:2 o 4:2:0.

Después, cada pequeño bloque se convierte al dominio de la frecuencia a través de la transformación discreta de coseno, mediante la ecuación detallada con anterioridad.

2.3. Formato BMP

Windows bitmap (.BMP) es un formato de imagen de mapa de bits, propio del sistema operativo Microsoft Windows. Puede guardar imágenes de 24 bits (16,7 millones de colores), 8 bits (256 colores) y menos. Puede darse a estos archivos una compresión sin pérdida de calidad: la compresión *RLE (Run-length encoding)*.

Los archivos de mapas de bits se componen de direcciones asociadas a códigos de color, uno para cada cuadro en una matriz de píxeles. Normalmente, se caracterizan por ser muy

poco eficientes en su uso de espacio en disco, pero pueden mostrar un buen nivel de calidad. A diferencia de los gráficos vectoriales al ser reescalados a un tamaño mayor, pierden calidad. Otra desventaja de los archivos BMP es que no son utilizables en páginas web debido a su gran tamaño en relación a su resolución.

Dependiendo de la profundidad de color que tenga la imagen cada píxel puede ocupar 1 o varios bytes. Generalmente se suelen transformar en otros formatos, como JPEG, GIF o PNG, los cuales utilizan otros algoritmos para conseguir una mayor compresión.

Los archivos comienzan con las letras 'BM' (0x42 0x4D), que lo identifica con el programa de visualización o edición. En la cabecera también se indica el tamaño de la imagen y con cuántos bytes se representa el color de cada píxel.

2.3.1. Almacenamiento de pixeles

Los bits representan los pixeles del *bitmap* que son empaquetados en filas. El tamaño de cada fila se consigue redondeando a múltiplos de cuatro bytes para relleno.

Para imágenes con tamaño mayor a uno, las múltiples filas rellenadas son almacenadas consecutivamente, formando una matriz de pixeles.

$$Tamaño\ de\ fila = \left\lfloor \frac{bits\ por\ pixel * ancho\ de\ imagen + 31}{32} \right\rfloor * 4$$

La cantidad total de bytes necesarios para almacenar una matriz de pixeles en una imagen de *n* bits por pixel (bpp), con 2^n colores, pueden ser calculados mediante la siguiente ecuación:

$$Tamaño\ de\ la\ matriz\ de\ pixeles = Tamaño\ de\ fila * |Altura\ de\ imagen|$$

2.3.1.1. Matriz de pixeles

La matriz de pixeles es un bloque de 32 bits, conocido también como palabra doble, que describe la imagen pixel por pixel. Usualmente los pixeles son almacenados al revés respecto al orden normal de exploración de imagen, comenzando en la esquina izquierda inferior, yendo de izquierda a derecha, y luego fila por fila de abajo hacia arriba de la imagen. Una imagen descomprimida

también puede ser almacenada desde arriba hacia abajo, cuando el valor de la altura de la imagen es negativa.

En el original *bitmap* independiente del dispositivo, los únicos cuatro valores legales de profundidad de color eran 1, 4, 8 y 24 bits por pixel. Las cabeceras de *bitmaps* independientes del dispositivo contemporáneos permiten formatos de pixel con 1, 2, 4, 8, 16, 24 y 32 bits por pixel. *GDI+* también permite 64 bits por pixel.

Para rellenar *bytes* se debe añadir al final de las filas en orden a la longitud de las filas que deben ser múltiplos de cuatro *bytes*. Cuando la matriz de pixeles es cargada en la memoria, cada fila debe comenzar en una dirección de memoria que es un múltiplo de 4. Esta restricción de dirección es obligatoria solo para una matriz de pixeles cargadas en memoria. Para propósitos de almacenamiento de archivos, solo el tamaño de cada fila debe ser un múltiplo de 4 *bytes* mientras que el desplazamiento del archivo puede ser arbitrario. Un *bitmap* con ancho igual a 1, tendría 3 *bytes* de datos por fila (rojo, verde y azul) y un *byte* de relleno; mientras que con un ancho igual a 2, tendría dos *bytes* de relleno; con un ancho de 3, tendría tres *bytes* de relleno; finalmente con un ancho de 4, no tendría ningún relleno.

2.3.1.2. Compresión

- El indexado de imágenes a color puede ser comprimido con el algoritmo de Huffman o *RLE* de 4 u 8 bits.
- Las imágenes *OS/2 BITMAPCOREHEADER2* de 24 bits por pixel pueden ser comprimidas con el algoritmo *RLE* de 24 bits.
- Las imágenes de 16 y 32 bits por pixel son siempre almacenadas sin compresión.
- Las imágenes en todos los matices de colores pueden ser guardados sin compresión si se desea.

2.3.1.3. Formato del pixel

En un archivo de imagen *bitmap*, ya sea en disco o en memoria, los pixeles pueden ser definidos por un número variable de bits, como se detalla a continuación:

- El formato de 1 bit por pixel soporta dos distintos colores, por ejemplo blanco y negro. Los valores de los pixeles son almacenados en cada bit, con el primer pixel en el bit más significante del primer *byte*. Cada bit es un índice dentro de una tabla de dos colores. Un bit sin asignar se referirá al primer elemento de la tabla de colores y un bit asignado se referirá al último elemento de la tabla de colores.
- El formato de 2 bits por pixel soporta 4 distintos colores y almacena 4 pixeles por un *byte*, estando el pixel de más a la izquierda en los dos bits más significantes. Cada valor de pixel es un índice de 2 bits dentro de una tabla 4 colores.
- El formato de 4 bits por pixel soporta 16 distintos colores y almacena 2 pixeles por un *byte*, estando el pixel de más a la izquierda en el *nibble* más significante. Cada valor de pixel es un índice de 4 bits dentro de una tabla de hasta 16 colores.
- El formato de 8 bits por pixel soporta 256 distintos colores y almacena 1 pixel por un *byte*, Cada byte es un índice dentro de una tabla de hasta 256 colores.
- El formato de 16 bits por pixel soporta 65536 distintos colores y almacena 1 pixel por un palabra. Cada palabra puede definir las muestras rojo, verde, azul y alfa de un pixel.
- El formato de 24 bits por pixel soporta 16777216 distintos colores y almacena el valor de 1 pixel en 3 *bytes*. Cada valor de pixel define las muestras rojo, verde y azul de un pixel (8.8.8.0.0 en notación *RGBAX*). Específicamente en el siguiente orden: azul, verde y rojo (8 bits por cada muestra).

- El formato de 32 bits por pixel soporta 4294967296 distintos colores y almacena un pixel por cada palabra doble. Cada palabra doble puede definir las muestras alfa, rojo, verde y azul del pixel.

2.4. Caos

El caos es considerado como: una misteriosa forma primigenia del cosmos, forma superior del orden, inestabilidad dinámica, oscilaciones irregulares de apariencia aleatoria pero de naturaleza determinista, etc. En resumen son complejos e imprecisos movimientos por las influencias externas o internas, desconocidas, quizás pseudoaleatorias, en sistemas dinámicos; y si estas pudiesen ser medidas, eliminadas, aisladas o controladas; entonces, los comportamientos pueden ser tratados como deterministas y predecibles (Colina, 1999).

Según la teoría de la relatividad del orden, el caos ontológico y el orden total del mundo no existen. Algunas de las propiedades del caos son:

- Sensibilidad a las condiciones iniciales.
- Idea de mezcla o topológicamente transitivo.
- Puntos densos o tan próximos como se quiera.
- No linealidad.
- Efecto mariposa.
- No aleatoriedad, determinismo.
- Curva parábola logística.
- Inestabilidad, probabilidad e irreversibilidad.
- Fluctuaciones y tendencias gráficas.
- Atractores extraños.
- Fractales.
- Órbitas periódicas.
- Complejidad infinita.
- Incompletitud.
- Naturaleza impredecible.
- Incertidumbre, indeterminismo y desorden.

- Proceso oscilatorio irregular.

Estas propiedades son una acumulación de conocimiento, que llevan intrínsecamente a la física y matemática (Colina, 1999).

2.4.1. Historia de la teoría del caos

Esta teoría se origina a principios del siglo XX, los físicos de ese entonces creían que solo quedaba por descubrir: la explicación de la órbita irregular del planeta Mercurio, la discrepancia entre la teoría y la cantidad de energía que libera un agujero negro y el efecto de un cuerpo en el movimiento de otros dos. A medida que los científicos intentaban dar respuestas a estas interrogantes, comienzan a nacer distintas teorías: del primer problema surgió la teoría de la relatividad, del segundo la teoría cuántica y del último la teoría del caos (Montero, 2010).

El caos es aquella tendencia general al desorden en la naturaleza, lo cual es evidente en la caída de un vaso o cuando un vidrio se rompe, pero esto no significa que el caos implique confusión, más al contrario, una de las características de los sistemas caóticos es la adaptabilidad que presenta ante los cambios. Este comportamiento puede ser observado al arrojar una piedra a un río, una vez efectuada dicha acción el curso del río no se ve modificada. Si el río fuera un sistema ordenado, en el que cada partícula tuviera una trayectoria fija, la piedra hubiera afectado este orden (Prigogine, 1993).

Al simular algún sistema mediante modelos finitos, inevitablemente se excluirá información acerca del sistema y/o de sus componentes relacionados, error que será magnificado en cada unidad de tiempo simulada, hasta que el error sobrepase los límites tolerables. Es decir: aunque se conozca el modelo, la simulación con el pasar del tiempo tendrá una divergencia importante con la realidad. De hecho, eso fue lo que le paso a Edward Lorenz, científico norteamericano que efectuó trascendentes aportes a la teoría del caos, introduciendo los conceptos de efecto mariposa y atractor extraño.

Lorenz era físico, matemático y meteorólogo del Instituto de Tecnología de Massachusetts. Aprovechando la capacidad de cálculo de una gran computadora de

la institución, en 1961, Lorenz construyó un modelo para predecir el tiempo atmosférico mediante ecuaciones que expresaban las relaciones entre temperatura, presión, velocidad del viento, humedad y otros datos, empleando un programa simulador que representaba gráficamente las distintas variables, mediante curvas. Un día introdujo en la computadora los datos de las series del día anterior, expresados con seis decimales, aunque para ahorrar tiempo los escribió con tres decimales, esperando un resultado igual o similar al anterior. Grande fue su sorpresa cuando comprobó que los resultados eran semejantes en el corto plazo, pero se tornaban totalmente diferentes en el mediano y largo plazo. Observó que a partir del primer valle de la curva, ésta comenzaba a separarse y rápidamente ambas curvas perdían cualquier similitud. El motivo era que su programa trabajaba con millonésimas y Lorenz había redondeado los datos a milésimas. Pese a ser un pequeño cambio numérico, esa entrada ligeramente distinta tuvo efectos sorprendentes en la salida. Este fenómeno es conocido como dependencia sensitiva de las condiciones iniciales: una pequeña variación puede cambiar drásticamente el comportamiento a largo plazo de un sistema (efecto mariposa) (Gutzwiller, 1990).

Se afirma que el efecto mariposa no es accidental, sino necesario. De no producirse, los ciclos meteorológicos obtenidos con estos modelos informáticos serían iguales, rígidos, sin reproducir la variación climática real. El tiempo atmosférico, además de ser un sistema dinámico, es muy sensible a los cambios en las variables iniciales, por lo cual su estudio debe abordarse desde la matemática caótica.

2.5. Teoría del caos

La teoría del caos es un modelo teórico que intenta explicar el comportamiento de los sistemas dinámicos que aparentemente se desarrolla aleatoriamente, es decir la teoría del caos intenta explicar modelos físicos y matemáticos que representan gran parte de los fenómenos naturales, como la propagación de un incendio o la evolución de la sociedad. Por tal razón las leyes de la teoría del caos son aplicadas a diversas áreas científicas, donde el caos se haga presente, como lo es el movimiento de las partículas de los fluidos o la homeostasis del medio interno. Los sistemas semejantes a los descritos brevemente

son deterministas, o sea su comportamiento está determinado por sus condiciones iniciales (Skinner, Molnar, Vybirar, & Mitra, 1992).

El caos es considerado la ciencia de la naturaleza global de los sistemas. El caos no implica inestabilidad, un sistema caótico puede ser estable si un tipo particular de irregularidad persiste frente a pequeñas perturbaciones, tomando en cuenta la dependencia sensitiva a las condiciones iniciales.

La teoría del caos puede ser definida como el estudio de la conducta aperiódica en sistemas determinísticos no lineales. Un sistema determinístico no lineal consiste en elementos que tienen influencias no lineales entre sí. El sistema es dinámico si cambia en el tiempo. El estado de un sistema dinámico en un cierto momento puede ser descrito por cierto número de variables y puede ser representado por un punto en un espacio de fase de determinadas dimensiones. La evolución en el tiempo consiste en una serie de puntos que forman una trayectoria en el espacio de estados. Cuando el tiempo tiende a infinito, la trayectoria sólo ocupará un sub espacio del espacio de estados, denominado atractor.

En síntesis el caos sería un punto intermedio entre lo determinístico y lo aleatorio. Es decir fluctuaciones irregulares, las cuales son descritas por ecuaciones determinísticas, diferentes de las fluctuaciones que obedecen a causas aleatorias. Las principales características de los sistemas caóticos es que son impredecibles en períodos amplios de tiempo y muy sensibles a las condiciones iniciales del sistema. Una vez iniciado con valores específicos, el futuro sistema podría desarrollarse de forma totalmente diferente si comenzase bajo condiciones ligeramente diferentes. La importancia de este hecho para los sistemas biológicos es que desórdenes muy similares podrían estar determinados por muy pocos y simples factores que podrían llegar a ser conocidos (Coppo, 2010).

2.5.1. Atractores

Los sistemas dinámicos determinísticos se clasifican básicamente en estables, inestables y caóticos. A lo largo del tiempo, un sistema estable tiende a ser atraído o repelido hacia o desde un punto u órbita. Un sistema inestable se escapa de los atractores. Un sistema caótico manifiesta los dos comportamientos: por un lado, existe un atractor por el cual el sistema es atraído, pero a la vez se aleja de éste. Una de las mayores características de un sistema inestable es que tiene una gran

dependencia de las condiciones iniciales. Si estas condiciones iniciales son fijas, conociendo las ecuaciones características de un sistema inestable se puede predecir su evolución en el tiempo. En los sistemas caóticos, una mínima diferencia en esas condiciones causaría que el sistema evolucione de manera totalmente distinta, por ejemplo: el comportamiento de las placas tectónicas, de los fluidos en régimen turbulento, los crecimientos de población, los movimientos en el sistema solar, etcétera.

El atractor es la representación geométrica de la dinámica del sistema en el tiempo; los atractores pueden ser caracterizados por sus dimensiones. Un atractor de dimensión 0 corresponde a un sistema estático: el sistema no cambia en el tiempo. Un atractor de dimensión 1 corresponde a un sistema periódico, en el cual un número finito de estados se repiten indefinidamente. Un atractor de dimensión 2 y mayores corresponde a un sistema cuasi periódico; en el caso de un péndulo oscilante, el atractor sería el punto de equilibrio central. Un atractor periódico puede guiar el movimiento de un péndulo en oscilaciones periódicas; sin embargo, el péndulo puede registrar trayectorias erráticas alrededor de estas oscilaciones debido a otros factores (Moon, 1990).

En resumen, un atractor es el comportamiento que adopta un sistema ante el influjo de un determinado estímulo. La cuenca de un atractor es el conjunto de condiciones iniciales a partir de las cuales el sistema sigue un determinado comportamiento. La trayectoria es una descripción matemática de la secuencia de valores que toma la variable desde la condición inicial. Los atractores pueden ser periódicos, cuasi periódicos y caóticos (o también denominados atractores extraños).

2.5.1.1. Definición matemática de un atractor

En un sistema dinámico con dinámica $f(t,\bullet)$, el atractor Λ es un subconjunto del espacio de fases tal que:

- Existe un entorno de Λ, llamado cuenca de atracción, al que converge cualquier sistema abierto que contenga Λ.
- $f(t,\Lambda) \supset \Lambda$ para t suficientemente grande.

Comúnmente se considera al atractor como un conjunto cerrado formado por los puntos de acumulación o convergencia de las órbitas, así el atractor propiamente dicho puede definirse como:

$$\Lambda = \bigcap_{t>t_0}^{\infty} f(t, \Lambda_{t_0}) = \bigcap_{t>t_0}^{\infty} \Lambda_t$$

Siendo Λ_{t_0} cualquier conjunto invariante tal que:

$$\Lambda_{t_0} \supseteq f(t, \Lambda_{t_0})$$

2.6. Mapas caóticos

La teoría del caos describe el comportamiento de ciertos sistemas dinámicos no lineales que bajo condiciones específicas exhiben un comportamiento dinámico, los cuales son sensitivos a las condiciones iniciales. Las dos propiedades básicas de los sistemas caóticos son la sensibilidad a las condiciones iniciales y la propiedad de mezcla. Estos mapas caóticos son usados para producir secuencias caóticas y también para controlar procesos de encriptación. Los flujos de caos son generados usando varios mapas caóticos. Existen varios tipos de mapas, a continuación serán analizados cuatro tipos de mapas.

2.6.1. Mapa cuadrático

El mapa cuadrático es analíticamente uno de los más complicados, cuya función tiene una forma mónica y centrada:

$$z_{n+1} = f_c(z_n) = {z_n}^2 + c$$

El conjunto de Mandelbrot está formado por valores del parámetro c para los cuales la condición inicial $z_0 = 0$ no causa divergencia al infinito.

Un punto crítico de f_c debe cumplir la siguiente condición:

$$f'_c(z) = \frac{d}{dx} f_c(z) = 2z = 0$$

Además un punto fijo es estable, súper estable, repelente, indiferente de acuerdo a como sus multiplicadores satisfagan $|m| < 1$, $|m| = 0$, $|m| > 1, |m| = 1$. El segundo punto fijo es siempre repelente. Si el valor absoluto de la función es mayor

al segundo punto fijo, las iteraciones van al infinito. En cambio, si es menor al segundo punto fijo, las iteraciones serán atraídas al primer punto fijo. Este intervalo es la base de la atracción del punto.

2.6.2. Mapa tienda de campaña

En matemáticas, este mapa es una función iterativa, con la forma de una tienda de campaña, formando un sistema dinámico discreto en el tiempo.

$$x_{n+1} = f_\mu(x_n) = \begin{cases} \mu x_n, & x_n < \frac{1}{2} \\ \mu(1 - x_n), & x_n \geq \frac{1}{2} \end{cases}$$

Donde μ es una constante real positiva.

Dependiendo del valor de μ, el mapa demostrará un comportamiento dinámico desde predecible hasta caótico.

- Si μ es menor a 1 el punto $x = 0$ es un punto fijo atractivo del sistema para todos los valores iniciales de x, tal que el sistema converja en $x = 0$ a partir de cualquier valor inicial de x.
- Si μ es igual a 1, todos los valores de x menores o iguales a $1/2$ son puntos fijos del sistema.
- Si μ es mayor a 1, el sistema tiene dos puntos fijos, uno en 0 y el otro en $\mu/(\mu + 1)$. Ambos puntos fijos son inestables.
- Si μ esta entre 1 y $\sqrt{2}$, da lugar a un conjunto de Julia.
- Si μ está entre 1 y 2, el intervalo contiene puntos periódicos y no periódicos, aunque todas las orbitas son inestables.
- Si μ es igual a 2, el mapa es caótico debido a los puntos periódicos densos.
- Si μ es más grande que 2, el conjunto de Julia del mapa se convierte en un conjunto de Cantor. El conjunto canónico de Cantor es el conjunto de Julia del mapa tienda de campaña para $\mu = 3$.

2.6.3. Mapa de Bernoulli

El mapa de Bernoulli es también conocido como mapa binario, mapa desplazamiento de bit, mapa diente de sierra, tal mapeo es producido por la siguiente regla:

$$x_0 = x$$

$$\forall n \geq 0, x_{n+1} = (2x_n) mod\ 1$$

Igualmente, la transformación anterior puede ser definida como un mapa de función iterativa:

$$f(x) = \begin{cases} 2x, & 0 \leq x < 0.5 \\ 2x - 1, & 0.5 \leq x < 1 \end{cases}$$

La transformación se basa en un proceso Bernoulli, el cual es un proceso estocástico discreto, el cual consiste en una secuencia finita o infinita de variables aleatorias independientes x_i, el valor de cada variable es 0 o 1.

El mapa de Bernoulli es un modelo solucionable dentro de la teoría del caos determinístico. Las autofunciones integrables cuadradas del operador asociado del mapa de Bernoulli son los polinomios de Bernoulli. Estas autofunciones forman un espectro discreto con autovalores 2^{-n} para enteros no negativos. Hay más autovectores que no son cuadrados integrables, asociados a un espectro continuo. Estos están dados por la función zeta de Hurwitz; equivalentemente, las combinaciones lineales de la zeta de Hurwitz producen fractales, autofunciones no derivables, incluyen la función Takagi.

2.6.4. Mapa logístico

El mapa logístico es un polinomio de grado dos, a menudo citado como un ejemplo arquetípico de cuan complejo puede llegar a ser el comportamiento caótico desde ecuaciones dinámicas no lineales muy simples. El mapa fue popularizado en un artículo altamente influyente de 1976 realizado por el biólogo Robert May, para el estudio de la evolución de la población de insectos en un sistema cerrado, dicho modelo es análogo a la ecuación logística creada por Pierre François Verhulst. Matemáticamente el mapa logístico es:

$$x_{n+1} = r x_n (1 - x_n)$$

Donde x_n es un número entre cero y uno, este representa la relación entre la población existente y la población máxima posible. Los valores de interés para el parámetro r (a veces denotado como μ) están en dentro del intervalo $[0,4]$.

El comportamiento depende del parámetro r, para generar comportamientos caóticos es necesario calcular los puntos fijos y periódicos, los cuales deben cumplir ciertas condiciones, a continuación se detallan dichos puntos:

$$p_0 = 0$$

$$p_r = \frac{r-1}{r}$$

Variando el parámetro r, el comportamiento será el siguiente:

- Con r entre 0 y 1, la población morirá finalmente, independientemente de la población inicial.
- Con r entre 1 y 2, la población rápidamente se aproximara a un valor independiente de la población inicial.
- Con r entre 2 y 3, la población al final también se aproximara al mismo valor, pero primero fluctuara alrededor de ese valor por algún tiempo. La tasa de convergencia es lineal, excepto en $r = 3$, cuando es dramáticamente lento, menor a la lineal.
- Con r entre 3 y 3,44949 (aproximadamente), de casi todas las condiciones iniciales, la población se acercará a permanentes oscilaciones entre dos valores. Estos dos valores son dependientes de r.
- Con r entre 3,44949 y 3,54409 (aproximadamente), de la mayoría de condiciones iniciales, la población se acercará a oscilaciones permanentes entre cuatro valores. El último número es una raíz de un polinomio de grado doce.
- Con r incrementándose más allá de 3,54409, para casi todas las condiciones iniciales, la población se acercará a oscilaciones entre ocho valores, luego dieciséis, treinta y dos, etc. Las longitudes de los intervalos de los parámetros

producen como resultado oscilaciones de una longitud dada que decrece rápidamente; la relación entre las longitudes de los dos intervalos sucesivos de bifurcación se aproximan a la constante de Feigenbaum. Este comportamiento es un ejemplo de una cascada de periodo doble.

- En r aproximadamente 3,56995 es la fase inicial del caos, al final de la cascada de doble periodo. De casi todas las condiciones iniciales, no se ven más largas oscilaciones de periodo finito. Leves variaciones en la población inicial dan diferentes resultados en el tiempo, una primera característica del caos.
- La mayoría de valores de r mayores a 3,56995 muestran un comportamiento caótico, pero aún hay ciertos rangos aislados de r que no muestran comportamiento caótico; estos son, a veces, llamados islas de estabilidad. Por ejemplo, comenzando en 3,82843 (aproximadamente) hay un rango de parámetros r que muestran oscilación entre tres valores, y para valores levemente más altos de r la oscilación va entre seis valores, luego doce, etc.
- El desarrollo del comportamiento caótico de la secuencia logística, con respecto al parámetro r, varia de aproximadamente 3,56995 a aproximadamente 3,82843 es conocido como escenario de Pomeau-Manneville, caracterizado por una fase periódica interrumpida por ráfagas de comportamiento aperiódico. Como un escenario tiene aplicación en dispositivos semiconductores. Hay otros rangos que dan como resultado oscilaciones entre cinco valores, etc.; toda oscilación ocurre en algunos valores de r.
- Más allá de $r = 4$, casi todos los valores iniciales eventualmente dejan el intervalo entre cero y uno y divergen.

Para un valor de r hay un ciclo estable. Si un ciclo estable existe, es globalmente estable, atrayendo a casi todos los puntos. Algunos valores de r con un ciclo estable de algún periodo tienen infinitos ciclos inestables de varios periodos.

El diagrama de bifurcación resume este hecho. El eje horizontal muestra los valores posibles del parámetro r mientras el eje vertical muestra el conjunto de valores de x

formados asintóticamente de casi todas las condiciones iniciales por las iteraciones de la ecuación logística con el respectivo valor r.

2.7. Criptografía

La palabra criptografía proviene del griego κρύπτος que significa oculto y γραφη que significa escritura, literalmente criptografía significaría escritura oculta, tradicionalmente se ha definido como las técnicas de cifrado o codificado destinadas a alterar las representaciones lingüísticas de mensajes con el fin de hacerlos ininteligibles a receptores no autorizados. Estas técnicas se utilizan tanto en el arte como en la ciencia y en la tecnología. Por tanto, el único objetivo de la criptografía era conseguir la confidencialidad de los mensajes, para lo cual se diseñaban sistemas de cifrado y códigos, y la única criptografía existente era la llamada criptografía clásica.

La aparición de la informática y el uso masivo de las comunicaciones digitales, han producido un número creciente de problemas de seguridad. Las transacciones que se realizan a través de la red pueden ser interceptadas, por lo que es necesario que la seguridad de esta información se garantice. Este desafío ha generalizado los objetivos de la criptografía para ser la parte de la criptología que se encarga del estudio de los algoritmos, protocolos criptográficos y sistemas que se utilizan para proteger la información y dotar de seguridad a las comunicaciones y a las entidades que se comunican.

Para ello los criptógrafos investigan, desarrollan y aprovechan técnicas matemáticas que les sirven como herramientas para conseguir sus objetivos. Los grandes avances que se han producido en el mundo de la criptografía, han sido posibles gracias a los grandes avances que se han producido en el campo de la matemática y la informática.

En criptografía, el cifrado es un procedimiento que utiliza un algoritmo de cifrado con cierta clave (clave de cifrado) para transformar un mensaje, sin atender a su estructura lingüística o significado, de tal forma que sea incomprensible o, al menos, difícil de comprender a toda persona que no tenga la clave secreta (clave de descifrado) del algoritmo. Las claves de cifrado y de descifrado pueden ser iguales (criptografía simétrica), distintas (criptografía asimétrica) o de ambos tipos (criptografía híbrida).

El juego de caracteres (alfabeto) usado en el mensaje sin cifrar puede no ser el mismo que el juego de caracteres que se usa en el mensaje cifrado.

A veces el texto cifrado se escribe en bloques de igual longitud. A estos bloques se les denomina grupos. Estos grupos proporcionaban una forma de verificación adicional, ya que el texto cifrado obtenido debía tener un número entero de grupos. Si al cifrar el texto plano no se tiene ese número entero de grupos, entonces se suele rellenar al final con ceros o con caracteres sin sentido.

Aunque el cifrado pueda volver secreto el contenido de un documento, es necesario complementarlo con otras técnicas criptográficas para poder comunicarse de manera segura. Puede ser necesario garantizar la integridad la autenticación de las partes, etcétera.

2.7.1. Objetivos de la criptografía

La criptografía se encarga del estudio de los algoritmos, protocolos y sistemas que se utilizan para dotar de seguridad a las comunicaciones, a la información y a las entidades que se comunican (Pastor, Sarasa, & Salazar, 1998). El objetivo de la criptografía es diseñar, implementar, implantar, y hacer uso de sistemas criptográficos para dotar de alguna forma de seguridad. Por tanto el tipo de propiedades de las que se ocupa la criptografía son, por ejemplo:

- La confidencialidad garantiza que la información sea accesible únicamente a personal autorizado. Para conseguirlo utiliza códigos y técnicas de cifrado.
- La integridad garantiza la corrección y completitud de la información. Para conseguirlo puede usar por ejemplo funciones hash criptográficas MDC, protocolos de compromiso de bit, o protocolos de notarización electrónica.
- La vinculación permite vincular un documento o transacción a una persona o un sistema de gestión criptográfico automatizado. Cuando se trata de una persona, se trata de asegurar su conformidad respecto a esta vinculación de forma que pueda entenderse que la vinculación gestionada incluye el entendimiento de sus implicaciones por la persona. Para conseguirlo se puede usar por ejemplo la firma digital. En algunos contextos lo que se intenta es justo lo contrario: Poder negar que se ha intervenido en la comunicación. Por

ejemplo cuando se usa un servicio de mensajería instantánea y no queremos que se pueda demostrar esa comunicación. Para ello se usan técnicas como el cifrado negable.

- La autenticación proporciona mecanismos que permiten verificar la identidad del comunicador. Para conseguirlo puede usar por ejemplo función hash criptográfica MAC o protocolo de conocimiento cero.
- Soluciones a problemas de la falta de simultaneidad en la telefirma digital de contratos. Para conseguirlo puede usar por ejemplo protocolos de transferencia inconsciente.

Un sistema criptográfico es seguro respecto a una tarea si un adversario con capacidades especiales no puede romper esa seguridad, es decir, el atacante no puede realizar esa tarea específica.

2.7.2. Métodos criptográficos según sus claves

Un sistema de encriptación se denomina:

- Simétrico cuando utiliza la misma clave para cifrar y descifrar. Los métodos más conocidos de este tipo de cifrado son el *DES*, el Triple *DES* y el *AES*.
- Asimétrico al usar claves diferentes: una pareja compuesta por una clave pública, que sirve para cifrar, y por una clave privada, que sirve para descifrar. El punto fundamental sobre el que se sostiene esta descomposición pública/privada es la imposibilidad práctica de deducir la clave privada a partir de la clave pública. A este tipo de cifrado también se le llama criptografía de clave pública o *PKE (Public-Key Encryption).* Los métodos más conocidos de este tipo de cifrado son el *RSA* y *ElGamal*.

La utilización de un sistema simétrico o asimétrico depende de las tareas a cumplir. La criptografía asimétrica presenta dos ventajas principales: suprime el problema de transmisión segura de la clave y permite la firma electrónica. No reemplaza sin embargo los sistemas simétricos, ya que los tiempos de cálculo son evidentemente más cortos con los sistemas simétricos que con los asimétricos.

2.7.3. Métodos criptográficos según sus algoritmos

Según la forma en la que operan los algoritmos de encriptación o des encriptación, es posible distinguir varios tipos:

- Cifrado en flujo: En estos algoritmos la encriptación se realiza bit a bit. Están basados en la utilización de claves muy largas que son utilizadas tanto para encriptar como para des encriptar. Estas claves pueden estar predeterminadas (libreta de un solo uso) o generarse usando un generador de claves pseudoaleatorias o *RKG (Random Key Generator)*, que genera una secuencia binaria pseudoaleatoria a partir de una clave de inicialización *K*. A veces, en el cálculo de la clave pseudoaleatoria también interviene el mensaje cifrado hasta ese momento. Por otra parte, el encriptador propiamente dicho: habitualmente en este tipo de algoritmos hay que mantener en secreto tanto la clave como el encriptador.
- Cifrado por bloques: En este tipo de algoritmos, el cifrado se realiza bloque a bloque. En primera instancia, se descompone el mensaje en bloques de la misma longitud. A continuación, cada bloque se va convirtiendo en un bloque del mensaje cifrado mediante una secuencia de operaciones. Ejemplos típicos de operaciones realizadas para conseguir cada mensaje cifrado son la sustitución y la permutación (cifrado por transposición) de elementos.

Este tipo de algoritmos pueden ser tanto de clave simétrica como de clave asimétrica. Sin embargo, en la bibliografía suele haber confusión y es frecuente ver casos en que se refieren sólo a algoritmos de clave simétrica.

2.7.4. Métodos criptográficos según sus propiedades

Muchas veces se agrupan los algoritmos de cifrado en función de sus propiedades o características. Algunos ejemplos:

- Cifrado seguro hacia adelante
- Cifrado con umbral
- Cifrado basado en identidad
- Cifrado negable

- Cifrado con clave aislada
- Cifrado maleable

2.7.5. Criptografía simétrica

La criptografía simétrica también llamada criptografía de clave simétrica o criptografía de clave secreta o criptografía de una clave es un método criptográfico en el cual se usa una misma clave para cifrar y descifrar mensajes en el emisor y el receptor. Las dos partes que se comunican han de ponerse de acuerdo de antemano sobre la clave a usar. Una vez que ambas partes tienen acceso a esta clave, el remitente cifra un mensaje usando la clave, lo envía al destinatario, y éste lo descifra con la misma clave.

2.7.5.1. Clave criptográfica

Una clave, palabra clave o clave criptográfica es una pieza de información que controla la operación de un algoritmo de criptografía. Habitualmente, esta información es una secuencia de números o letras mediante la cual, en criptografía, se especifica la transformación del texto plano en texto cifrado, o viceversa. En sistemas informáticos, la clave sirve para verificar que alguien está autorizado para acceder a un servicio o un sistema. Las claves también se utilizan en otros algoritmos criptográficos, como los sistemas de firma digital y las funciones de hash con clave (asimismo llamadas códigos de autentificación de mensajes).

Un algoritmo bien diseñado debe producir, a partir del mismo texto plano, dos textos cifrados completamente diferentes si se usa una clave distinta. Similarmente, descifrar un texto cifrado con una clave errónea debería producir un galimatías aparentemente caótico. (En la criptografía denegable, dos claves pueden producir dos textos planos muy diferentes pero aparentemente normales.) Si la clave se pierde, los datos cifrados deberían ser irrecuperables en la práctica.

Los sistemas de cifrado que emplean la misma clave para el cifrado y el descifrado son conocidos como algoritmos de clave simétrica. En los años 70 se descubrieron nuevos métodos que usan un par de claves relacionadas, una para cifrar y otra

para descifrar información. Estos métodos, llamados de criptografía asimétrica, permiten que una de las dos claves sea hecha pública, posibilitando así que cualquiera pueda mandar al poseedor de la clave privada un mensaje cifrado que sólo esta persona puede descifrar.

2.7.5.2. Criterios para escoger una clave

La clave puede ser una palabra con o sin sentido o también una frase completa, siendo esta última más segura debido a su mayor longitud.

La seguridad de una clave viene determinada por:

- La longitud: cuanto mayor sea el número de bits de información en la clave, mayor es el número de combinaciones que debe probar un atacante que use la pura fuerza bruta. Por ejemplo, una clave compuesta de dos cifras necesita 100 intentos como máximo para ser descubierta, mientras que una clave de cinco dígitos requiere 100.000 intentos como máximo.
- La aleatoriedad: si una clave es elegida basándose en palabras que existen en una lengua natural, es susceptible de ser rota mediante un ataque de diccionario. Por ello, una clave segura debe ser generada de una manera aleatoria, usando letras y números, así como símbolos ortográficos si es posible. Desde un punto de vista matemático, una clave es más segura cuanta más entropía contenga. Algunos sistemas informáticos contienen herramientas para "obtener entropía" de procesos impredecibles, como los movimientos de la cabeza lectora del disco duro. Sin embargo, la verdadera aleatoriedad sólo puede conseguirse mediante procesos físicos, como el ruido producido por el viento captado por un micrófono o, simplemente, un dado.
- El periodo de uso: una clave se vuelve más insegura cuanto mayor sea el tiempo que ha estado en uso. Por eso es importante asegurarse de que se renuevan con suficiente regularidad, aunque hayan sido generadas con la mayor aleatoriedad posible.

2.7.5.3. Longitud de la clave

Para una libreta de un solo uso, la clave debe ser al menos igual de larga que el mensaje a cifrar. En sistemas de cifrado que utilizan un algoritmo de cifrado, la clave puede ser mucho más corta que el mensaje. La clave puede ser una contraseña o una frase completa.

Cuando una contraseña se emplea como clave de cifrado, un sistema bien diseñado la pasa primero por un algoritmo de derivación de claves que añade sal y reduce o expande el resultado hasta la longitud deseada, por ejemplo, reduciendo una frase larga a un valor de 128 bits utilizable por algoritmo de cifrado por bloques. Así obtenemos una clave que ocupa menos sin reducir la seguridad, ya que costaría más ir probando combinaciones sobre una clave corta pero muy aleatoria, que probar palabras y sus variaciones con un diccionario sobre la frase original.

Como medida adicional, se debe usar un sistema de cifrado de contraseña que siempre produzca un resultado con la misma longitud, independientemente de la longitud de la frase original, para que no se pueda inferir la longitud de la frase a partir de la longitud de la clave. Esto se consigue utilizando las llamadas funciones hash criptográficas.

En general, las funciones de hash no son biyectivas, por lo que la inversa, el descifrado, se complica, y la probabilidad de que dos entradas distintas retornen un mismo resultado es muy baja.

Matemáticamente hablando:

$$A <> B \rightarrow P\big(MD5(A) = MD5(B)\big) = 0$$

Un tamaño de clave de 80 bits se considera como el mínimo necesario para tener buena seguridad con algoritmos de clave simétrica. Comúnmente se usan claves de 128 bits, consideradas muy seguras.

En los sistemas de clave pública se utilizan claves que tienen una cierta estructura matemática, ya que deben estar relacionadas entre sí y no ser completamente aleatorias. Por ejemplo, las claves públicas usadas en el sistema RSA son el

producto de dos números primos. Por tanto, los sistemas de clave pública requieren mayores longitudes de clave que los sistemas simétricos para ofrecer un nivel de seguridad equivalente. La longitud de clave sugerida para sistemas basados en factorización y logaritmos discretos es de 3072 bits para obtener una seguridad equivalente a un sistema de cifrado simétrico de 128 bits.

La criptografía de curva elíptica podría permitir claves de menor longitud para una seguridad equivalente, pero estos algoritmos son conocidos desde hace relativamente poco tiempo y las estimaciones actuales para la dificultad de romper sus claves podrían no resultar acertadas. Recientemente, un mensaje cifrado mediante una clave de 109 bits y un algoritmo de curva elíptica fue descifrado usando tan sólo la fuerza bruta. La regla para estimar las longitudes de clave para estos algoritmos es utilizar el doble de bits que para un algoritmo de clave simétrica y conjeturar que ambos ofrecen entonces una seguridad equivalente.

De todas maneras, el único algoritmo que ha sido matemáticamente probado seguro es el método de la libreta de un solo uso, de manera que algún avance matemático futuro (como la computación cuántica) podría convertir todos nuestros datos secretos actuales en un libro abierto. Una razón más para preferir claves largas sobre las cortas.

2.7.6. Criptosistema

El sistema conformado por los procesos de confusión y difusión es conocido como Criptosistema, estos procesos se repiten por un número determinado de veces para lograr un nivel de seguridad satisfactorio. Cabe resaltar la propiedad aleatoria de los mapas caóticos, dicha propiedad es muy útil para la encriptación de imágenes.

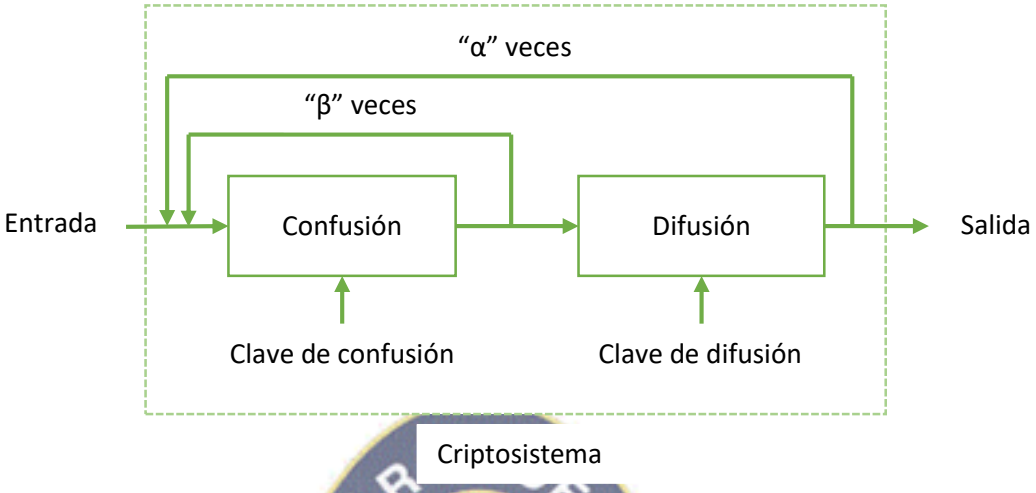

Ilustración 11. Esquema general de un sistema criptográfico

Matemáticamente el sistema puede ser representado de la siguiente manera:

$$\boldsymbol{R} = D^{\alpha}(C^{\beta}(\boldsymbol{P}, K_C), K_D)$$

Donde **P** representa la entrada que en el caso particular resulta ser una imagen y **R** es la salida del sistema que particularmente corresponde a la imagen cifrada; el bloque de confusión del sistema es representado por la función D y el bloque de difusión representado por la función D; α y β son variables que representan el número de iteraciones que se realizan en cada etapa. K_C y K_D son las claves de confusión y difusión respectivamente.

Mientras mayor sea la sensibilidad a las claves de confusión y difusión de las funciones de confusión y difusión mayor será el nivel de seguridad, de la misma manera el tamaño de las claves es un factor a considerar en la seguridad. El tamaño de la clave secreta se define a continuación:

$$S = ({S_C}^{\beta} S_D)^{\alpha}$$

Donde S_C y S_D son el tamaño de las claves de confusión y difusión respectivamente y están determinados por el tamaño de las claves de las condiciones iniciales y los parámetros de las funciones de difusión y confusión. Mientras más grandes sean los valores de las variables α y β mayor será el tamaño de la clave del sistema, esto por el carácter directamente proporcional de la ecuación S. También existe otro parámetro llamado EDT (Tiempo de encriptación/desencriptación) el cual también es directamente proporcional. Todo esto incrementa el tiempo de procesamiento en gran

manera, por lo tanto el diseño de un sistema criptográfico debe equilibrar estos dos aspectos: la seguridad y el tiempo de procesamiento.

2.8. Algoritmo de encriptación de datos *DES*

DES es un algoritmo de encriptación, es decir, un método para cifrar información, escogido como un estándar *FIPS (Federal Information Processing Standard)* en los Estados Unidos en 1976, y cuyo uso se ha propagado ampliamente por todo el mundo. El algoritmo fue controvertido al principio, con algunos elementos de diseño clasificados, una longitud de clave relativamente corta, y las continuas sospechas sobre la existencia de alguna puerta trasera para la *NSA (National Security Agency)*. Posteriormente *DES* fue sometido a un intenso análisis académico y motivó el concepto moderno de la encriptación por bloques y su criptoanálisis.

Hoy en día, DES se considera inseguro para muchas aplicaciones. Esto se debe principalmente a que el tamaño de clave de 56 bits es corto; las claves de DES se han roto en menos de 24 horas. Existen también resultados analíticos que demuestran debilidades teóricas en su cifrado, aunque son inviables en la práctica. Se cree que el algoritmo es seguro en la práctica en su variante de Triple DES, aunque existan ataques teóricos.

2.8.1. Historia del algoritmo *DES*

Los orígenes de *DES* se remontan a principios de la década de los 70. Específicamente en 1972, tras terminar un estudio sobre las necesidades del gobierno en materia de seguridad informática, la por entonces autoridad de estándares estadounidense *NBS (National Bureau of Standards)* concluyó en la necesidad de un estándar a nivel gubernamental para cifrar información confidencial. En consecuencia, el 15 de mayo de 1973, tras consultar con la *NSA*, el *NBS* solicitó propuestas para un algoritmo que cumpliera rigurosos criterios de diseño. A pesar de todo, ninguna de ellas parecía ser adecuada. Una segunda petición fue realizada el 27 de agosto de 1974. En aquella ocasión, *IBM* presentó un candidato que fue considerado aceptable, un algoritmo desarrollado durante el periodo de 1973 a 1974 basado en otro anterior, el algoritmo Lucifer de Horst Feistel. El equipo de *IBM* dedicado al diseño y análisis del algoritmo

estaba formado por Feistel, Walter Tuchman, Don Coppersmith, Alan Conheim, Carl Meyer, Mike Matyas, Roy Adler, Edna Grossman, Bill Notz, Lynn Smith, y Bryant Tuckerman.

El 17 de marzo de 1975, la propuesta de DES fue publicada en el Registro Federal. Se solicitaron comentarios por parte del público, y el año siguiente se abrieron dos talleres libres para discutir el estándar propuesto. Hubo algunas críticas desde ciertos sectores, incluyendo a los pioneros de la criptografía asimétrica Martin Hellman y Whitfield Diffie, mencionando la corta longitud de la clave y las misteriosas S cajas como una evidencia de la inadecuada interferencia de la *NSA*. La sospecha era que el algoritmo había sido debilitado de manera secreta por la agencia de inteligencia de forma que solo ellos pudiesen leer mensajes cifrados fácilmente. Alan Conheim comentó en una ocasión "enviaron las S cajas a Washington. Cuando volvieron eran totalmente diferentes". El Comité de Inteligencia del Senado de los Estados Unidos revisó las acciones de la NSA para determinar si había existido algún comportamiento inadecuado. En el resumen desclasificado sobre sus conclusiones, publicado en 1978, el Comité escribía: "En el desarrollo de *DES*, la *NSA* convenció a *IBM* de que un tamaño de clave reducido era suficiente; participó de forma indirecta en el desarrollo de las estructuras de las S cajas; y certificó que hasta donde ellos conocían estaban libres de cualquier punto débil matemático o estadístico". De todas formas, también concluyó que "La *NSA* no ejerció presión en el diseño del algoritmo en modo alguno. *IBM* inventó y diseñó el algoritmo, tomó todas las decisiones respecto a él, y coincidió en que el tamaño de la clave era más que apropiado para todas las aplicaciones comerciales para las que estaba pensado *DES*". Otro miembro del equipo de *DES*, Walter Tuchman, decía también: "Desarrollamos todo el algoritmo *DES* en *IBM* y con gente de *IBM*. ¡La *NSA* no dictó ni un sólo paso!".

Algunas de las sospechas sobre puntos débiles ocultos en las *S* cajas fueron descartadas en 1990, con el descubrimiento independiente y la publicación libre por Eli Biham y Adi Shamir del criptoanálisis diferencial, un método general para romper cifrados de bloque. Las S cajas de *DES* eran mucho más resistentes al ataque que si hubiesen sido escogidas al azar, lo que sugería que *IBM* conocía la técnica allá en los 70. De hecho, este era el caso en 1994, Don Coppersmith publicó los criterios de

diseño originales para las S cajas. *IBM* había descubierto el criptoanálisis diferencial en los 70 y tras asegurar *DES*, la *NSA* les ordenó mantener en secreto la técnica. Coppersmith explica: "Esto era así porque el criptoanálisis diferencial puede ser una herramienta muy potente, contra muchos esquemas diferentes, y había la preocupación de que aquella información en dominio público podía afectar negativamente a la seguridad nacional". Shamir también comentó "Yo diría, al contrario de lo que algunos creen, que no hay evidencias de influencia alguna en el diseño de *DES* para que su estructura básica esté debilitada".

Las otras críticas sobre la longitud de la clave se fundaban en el hecho de que la razón dada por la *NSA* para reducir la longitud de la clave de 64 bits a 56 era que los 8 bits restantes podían servir como bits de paridad, lo que en cierto modo resultaba sospechoso. Es ampliamente aceptado que la decisión de la *NSA* estaba motivada por la posibilidad de que ellos podrían llevar a cabo un ataque por fuerza bruta contra una clave de 56 bits varios años antes que el resto del mundo.

A pesar de la polémica, *DES* fué aprobado como estándar federal en noviembre de 1976, y publicado el 15 de enero de 1977 como *FIPS PUB 46*, autorizado para el uso no clasificado de datos. Fue posteriormente confirmado como estándar en 1983, 1988 (revisado como *FIPS-46-1*), 1993 (*FIPS-46-2*) y de nuevo en 1998 (*FIPS-46-3*), este último definiendo "Triple *DES*". El 26 de mayo de 2002, *DES* fue finalmente reemplazado por *AES (Advanced Encryption Standard)*, tras una competición pública.

Otro ataque teórico, el criptoanálisis lineal, fue publicado en 1994, pero fue un ataque por fuerza bruta en 1998 el que demostró que *DES* podría ser atacado en la práctica, y se destacó la necesidad de un algoritmo de repuesto. Estos y otros métodos de criptoanálisis se comentan con más detalle posteriormente en este artículo.

La introducción de *DES* es considerada como un desencadenador del estudio académico de la criptografía, en particular de los métodos para romper cifrados de bloque. Bruce Schneier escribe:

"De puertas hacia dentro, la *NSA* ha visto a *DES* como uno de sus grandes errores. Si hubiesen sabido que los detalles serían publicados para que la gente pudiese escribir software, nunca hubieran estado de acuerdo. *DES* hizo más para galvanizar el campo

de la criptografía que nunca nada antes. Ahora había un algoritmo que estudiar: uno que la *NSA* decía que era seguro".

2.8.2. Algoritmos de reemplazo

Muchos de los anteriores usuarios de *DES* ahora utilizan *Triple DES (3DES)* que fue descrito y analizado en una de las patentes de *DES (FIPS PUB 46-3). 3DES* ha sido ampliamente reconocido como seguro por ahora, aunque es bastante lento. Una alternativa más económica en términos computacionales es *DES-X*, que incrementa el tamaño de clave haciendo un *XOR* lógico sobre los elementos extra de la clave antes y después de *DES*. *GDES* fue una variante de *DES* propuesta para acelerar el proceso de cifrado, pero se demostró que era susceptible de ser sometido al criptoanálisis diferencial.

En 2001, tras un concurso internacional, el *NIST* escogió un nuevo algoritmo: el *AES (Advanced Encryption Standard)*, para reemplazar a *DES*. El algoritmo elegido para ser el reemplazo fue propuesto por sus diseñadores bajo el nombre de *Rijndael*. Otros finalistas en la competición *AES* del *NIST* fueron *RC6*, *Serpent*, *MARS*, y *Twofish*.

En general, no hay ningún algoritmo que se adapte perfectamente a todos los usos. Un algoritmo para uso en máquinas de uso general (por ejemplo, *SSH*, o algunos tipos de cifrado de correo electrónico), no siempre funciona bien en sistemas embebidos o tarjetas inteligentes, y viceversa.

2.8.3. Descripción del algoritmo *DES*

DES es el algoritmo prototipo del cifrado por bloque, un algoritmo que toma un texto en claro de una longitud fija de bits y lo transforma mediante una serie de operaciones básicas en otro texto cifrado de la misma longitud. En el caso de *DES* el tamaño del bloque es de 64 bits. *DES* utiliza también una clave criptográfica para modificar la transformación, de modo que el descifrado sólo puede ser realizado por aquellos que conozcan la clave concreta utilizada en la encriptación. La clave mide 64 bits, aunque en realidad sólo 56 de ellos son empleados por el algoritmo. Los ocho bits restantes se utilizan únicamente para comprobar la paridad, y después son descartados. Por

tanto, la longitud de clave efectiva en *DES* es de 56 bits, y así es como se suele especificar.

La estructura básica del algoritmo es representada en la siguiente figura:

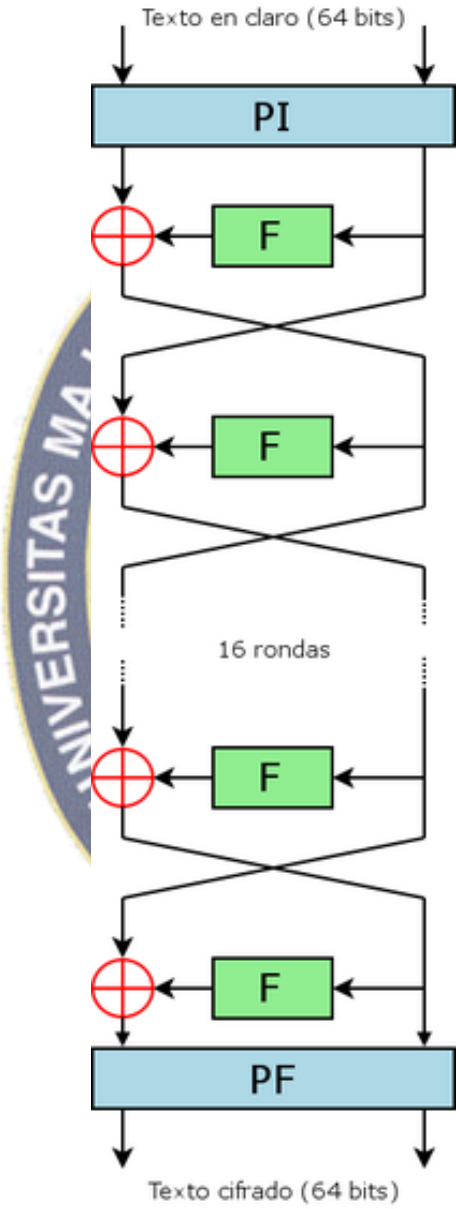

Ilustración 12: Estructura general de Feistel.

Hay 16 fases idénticas de proceso, denominadas rondas. También hay una permutación inicial y final denominada *PI* y *PF*, que son funciones inversas entre sí (*PI* deshace la acción de *PF* y viceversa). *PI* y *PF* no son criptográficamente significativas, pero se incluyeron presuntamente para facilitar la carga y descarga de

bloques sobre el hardware de mediados de los 70. Antes de las rondas, el bloque es dividido en dos mitades de 32 bits y procesadas alternativamente. Este entrecruzamiento se conoce como esquema Feistel.

La estructura de Feistel asegura que el cifrado y el descifrado sean procesos muy similares, la única diferencia es que las sub claves se aplican en orden inverso cuando desciframos. El resto del algoritmo es idéntico. Esto simplifica enormemente la implementación, en especial sobre hardware, al no haber necesidad de algoritmos distintos para el cifrado y el descifrado.

La función *F* mezcla la mitad del bloque con parte de la clave. La salida de la función *F* se combina entonces con la otra mitad del bloque, y los bloques son intercambiados antes de la siguiente ronda. Tras la última ronda, las mitades no se intercambian; ésta es una característica de la estructura de Feistel que hace que el cifrado y el descifrado sean procesos parecidos.

La función *F* opera sobre medio bloque de 32 bits cada vez y consta de cuatro pasos, descritos a continuación:

a) Expansión, la mitad del bloque de 32 bits se expande a 48 bits mediante la permutación de expansión, denominada *E*, duplicando algunos de los bits.
b) Mezcla, el resultado se combina con una sub clave utilizando una operación *XOR*. Dieciséis sub claves, una para cada ronda, se derivan de la clave inicial mediante la generación de sub claves descrita más abajo.
c) Sustitución, tras mezclarlo con la sub clave, el bloque es dividido en ocho trozos de 6 bits antes de ser procesados por las S cajas, o cajas de sustitución. Cada una de las ocho S cajas reemplaza sus seis bits de entrada con cuatro bits de salida, de acuerdo con una trasformación no lineal, especificada por una tabla de búsqueda. Las S cajas constituyen el núcleo de la seguridad de *DES*, sin ellas el cifrado sería lineal y fácil de romper.
d) Permutación, finalmente las 32 salidas de las S cajas se reordenan de acuerdo a una permutación fija dando como salida la caja *P*.

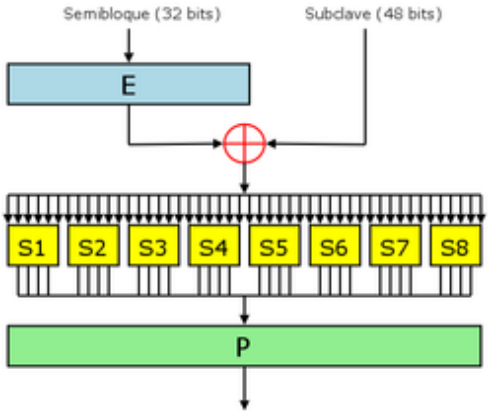

Ilustración 13: Estructura de la función de Feistel.

Alternando la sustitución de las S cajas, y la permutación de bits de la caja *P* y la expansión *E* proporcionan las llamadas confusión y difusión respectivamente, un concepto identificado por Claude Shannon en los 40 como una condición necesaria para un cifrado seguro y práctico.

La generación de claves que se representa a continuación, se encarga de proporcionar las sub claves:

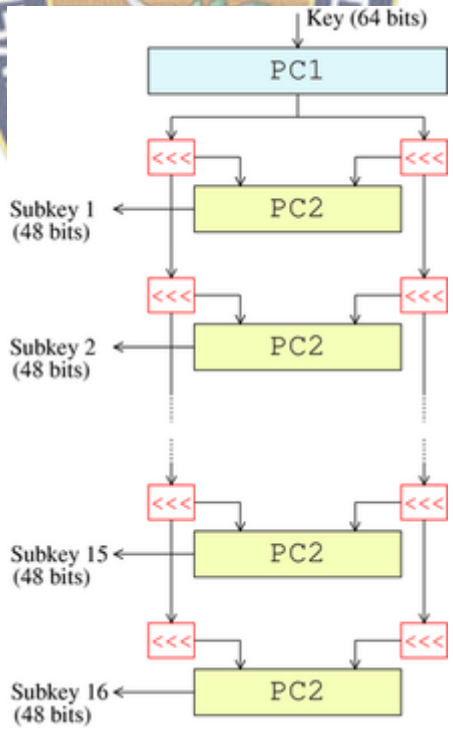

Ilustración 14: Esquema para la generación de claves.

Primero, se seleccionan 56 bits de la clave de los 64 iniciales mediante la elección permutada 1 denominada *PC1*, los ocho bits restantes pueden descartarse o utilizarse como bits de comprobación de paridad. Los 56 bits se dividen entonces en dos mitades de 28 bits; a continuación cada mitad se trata independientemente. En rondas sucesivas, ambas mitades se desplazan hacia la izquierda uno o dos bits dependiendo de cada ronda y entonces se seleccionan 48 bits de sub clave mediante la elección permutada 2 denominada *PC2*, 24 bits de la mitad izquierda y 24 de la derecha. Los desplazamientos implican que se utiliza un conjunto diferente de bits en cada sub clave; cada bit se usa aproximadamente en 14 de las 16 sub claves.

La generación de claves para descifrado sigue un proceso similar con la diferencia de generar las claves en orden inverso.

2.8.4. Seguridad y criptoanálisis del algoritmo *DES*

Aunque se ha publicado más información sobre el criptoanálisis de *DES* que de ningún otro algoritmo de encriptación por bloque, el ataque más práctico hoy en día sigue siendo por fuerza bruta. Se conocen varias propiedades criptoanalíticas menores, y son posibles tres tipos de ataques teóricos que, aunque requieren de una complejidad teórica menor que un ataque por fuerza bruta, requieren una cantidad irreal de textos planos conocidos o escogidos para llevarse a cabo, y no se tienen en cuenta en la práctica.

Para cualquier tipo de encriptación o cifrado, el método de ataque más simple es el ataque por fuerza bruta probando una por una cada posible clave. La longitud de clave determina el número posible de claves, y por tanto la factibilidad del ataque. En el caso de *DES*, ya en sus comienzos se plantearon cuestiones sobre su longitud de clave, incluso antes de ser adoptado como estándar, fue su reducido tamaño de clave, más que el criptoanálisis teórico, el que provocó la necesidad de reemplazarlo. Se sabe que la *NSA* animó, o incluso persuadió a *IBM* para que redujera el tamaño de clave de 128 bits a 64 bits y de ahí a 56 bits; con frecuencia esto se ha interpretado como una evidencia de que la *NSA* poseía suficiente capacidad de computación para romper claves de este tamaño incluso a mediados de los 70.

Académicamente, se adelantaron varias propuestas de una máquina para romper *DES*. En 1977, Diffie y Hellman propusieron una máquina con un coste estimado de 20 millones de dólares que podría encontrar una clave *DES* en un sólo día. Hacia 1993, Wiener propuso una máquina de búsqueda de claves con un coste de un millón de dólares que encontraría una clave en 7 horas. La vulnerabilidad de *DES* fue demostrada en la práctica en 1998 cuando la *EFF (Electronic Frontier Foundation)*, un grupo dedicado a los derechos civiles en el ciberespacio, construyó una máquina a medida para romper *DES*, con un coste aproximado de doscientos cincuenta mil dólares. Su motivación era demostrar que se podía romper *DES* tanto en la teoría como en la práctica: "Hay mucha gente que no creerá una verdad hasta que puedan verla con sus propios ojos. Mostrarles una máquina física que pueda romper *DES* en unos pocos días es la única manera de convencer a algunas personas de que realmente no pueden confiar su seguridad a *DES*". La máquina rompió una clave por fuerza bruta en una búsqueda que duró poco más de 2 días; más o menos al mismo tiempo, un abogado del Departamento de Justicia de los Estados Unidos proclamaba que *DES* era irrompible.

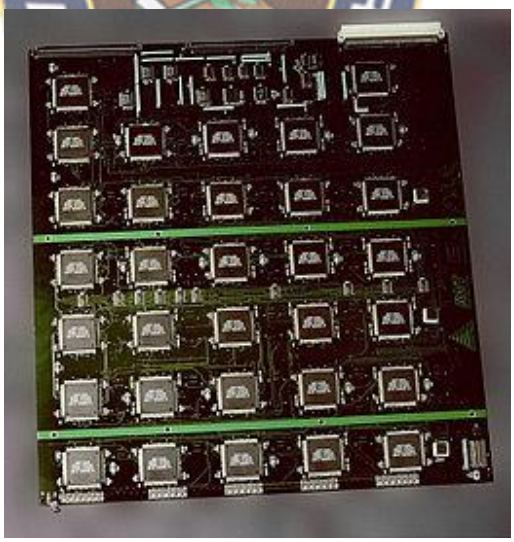

Ilustración 15: Circuito construido con chips Deep Crack.

La máquina que rompe la seguridad del algoritmo *DES* de la *Electronic Frontier Foundation* contenía mil quinientos treinta y seis chips y podía romper una clave *DES* por fuerza bruta en días.

3. Diseño del algoritmo

El diseño del algoritmo *ICT* se basa en dos propiedades identificadas por *Claude Shannon* en su reporte clasificado de 1945 *A Mathematical Theory of Cryptography*. Estas propiedades sirven para frustrar la aplicación de estadísticas y otros métodos de criptoanálisis.

Estas dos propiedades son la difusión y la confusión. La confusión en la encriptación significa que los datos de entrada cambian drásticamente con respecto a los datos de salida. La difusión significa que el cambio de un solo dato de la entrada cambiará muchos datos de la salida.

3.1. Algoritmo *ICT* de encriptación

En primer se toma algunas consideraciones antes del diseño del algoritmo *ICT*, las entradas serán la imagen y la contraseña, la contraseña utilizará una clave secreta de 80 bits, representadas de la siguiente manera:

$$K = k_0 k_1 k_2 \dots k_{80}$$

Esta clave de 80 bits consiste en diez caracteres alfanuméricos, con estos bits se formarán los parámetros de entrada del mapa logístico:

$$x_{n+1} = r x_n (1 - x_n)$$

De acuerdo a los conceptos mencionados con anterioridad tenemos dos parámetros de entrada, el primero es el parámetro *r* que debe estar contenido en el siguiente intervalo:

$$r = [3.56995, 3.82843]$$

Estos valores son determinados por el comportamiento de naturaleza caótica de ese intervalo en el mapa logístico.

El segundo parámetro es la condición inicial del mapa logístico que inicialmente representaba la tasa de población existente entre la máxima población posible, para los motivos del diseño de algoritmo solo interesa conocer el intervalo de acción del parámetro:

$$x_0 = [0, 1]$$

Ahora que tenemos definidos los intervalos de los valores que pueden tomar los parámetros *r* y x_0, para el diseño se dividirá la clave de 80 bits en dos, los primeros 40

bits generarán el valor del parámetro x_0 y los siguientes 40 bits generarán el valor del parámetro *r*.

Para el parámetro x_0 aplicamos una interpolación de valores para la correspondencia con los 40 bits designados, como se detalla a continuación:

$$x_0 = \frac{\sum_{i=0}^{39} k_i 2^i}{\sum_{i=0}^{39} 2^i}$$

Para generar el segundo parámetro se aplica la misma técnica para la correspondencia de valores, para obtener el valor del parámetro *r*, dando como resultado la siguiente ecuación:

$$r = \frac{\sum_{i=40}^{79} k_i 2^i}{\sum_{i=0}^{39} 2^i} 0.25848 + 3.56995$$

Cabe mencionar que el 0.25848 es el resultado de restar la cota superior menos la cota inferior del intervalo definido para el parámetro *r*, y el 3.56995 es la cota inferior de dicho intervalo.

Pasando al diseño del algoritmo de encriptación, primeramente definimos una imagen a colores de 24 bits, es decir que el formato de la imagen de entrada no es restringido ya que las imágenes serán convertidas a un formato de colores de 24 bits, para ser más precisos se obtendrá la imagen con sus componentes *RGB*, no siendo tomados en cuenta para el diseño los otros esquemas de color.

La imagen será representada de la siguiente manera para detallar de mejor manera el diseño:

$$IN = \begin{bmatrix} in_{00} & \cdots & in_{0m} \\ \vdots & \ddots & \vdots \\ in_{n1} & \cdots & in_{nm} \end{bmatrix}$$

Cada componente viene a representar el pixel de la imagen en la posición especificada por sus subíndices. Las dimensiones de la matriz representan el ancho y alto de la imagen, es decir la imagen no necesariamente será cuadrada, las dimensiones tanto de la matriz como de la imagen serán *n* para representar la altura de la imagen y m para representar el ancho de la imagen.

Cada elemento de la matriz que representa un pixel a su vez representa una terna de datos, los cuales son los componentes de cada pixel, esta terna de datos son los colores rojo, verde y azul, y serán representados de la siguiente manera:

$$in_{ij} = (r_{ij}, g_{ij}, b_{ij})$$

Donde el elemento *in* representa al (i,j)-ésimo pixel de la imagen, los subíndices *i* y *j* o pares ordenados *(i,j)* representan la posición del pixel, pudiendo *i* tomar el valor mínimo 0 y como valor máximo *n*, y el subíndice *j* toma valores desde 0 hasta *m*, otro aspecto importante de mencionar es el valor que toman los datos de la terna que varía desde 0 hasta 255.

Una vez definida la imagen se procede definir el mapa logístico con los parámetros previamente calculados mediante las entradas:

$$x_{n+1} = rx_n(1 - x_n)$$

Este mapa logístico nos ayudara a conseguir la aleatoriedad necesaria para el algoritmo, esta aleatoriedad es más evidente gracias al siguiente diagrama:

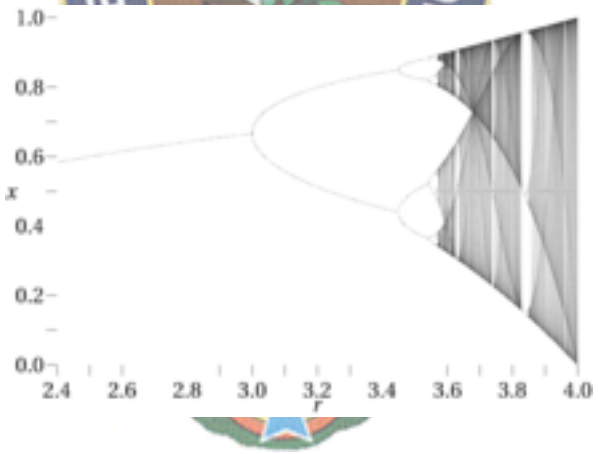

Ilustración 16: Diagrama de bifurcación para el mapa logístico.

Pero antes es necesario definir una imagen para la salida la cual será representada de la siguiente manera:

$$ENC = \begin{bmatrix} enc_{00} & \cdots & enc_{0m} \\ \vdots & \ddots & \vdots \\ enc_{n1} & \cdots & enc_{nm} \end{bmatrix}$$

Si bien las dimensiones de la imagen de salida serán las mismas los subíndices tendrán un comportamiento caótico, esto se consigue aplicando el mapa caótico a filas y columnas

mediante una interpolación, la manera de representar matemáticamente dicho comportamiento resulta complejo, por lo que se procederá a describir este paso con un ejemplo, basado en una imagen de 7x7.

Primeramente se definen los parámetros de entrada $x_0 = 0.653$ y $r = 3.99$, ambos parámetros son aplicados al mapa logístico dando como resultado:

Iteración	Valor de x
0	0,653
1	0,90409809
2	0,34595189
3	0,90281402
4	0,35008604
5	0,90782796
6	0,33386866

Ilustración 17: Tabla con seis iteraciones del mapa logístico.

Los valores de esta tabla deben ser normalizados, dando como resultado la siguiente tabla, en la que se muestra la correspondencia con las nuevas posiciones de la imagen encriptada:

Valor de x	Posición	Valor nuevo
0,653	3	0
0,90409809	5	1
0,34595189	1	2
0,90281402	4	3
0,35008604	2	4
0,90782796	6	5
0,33386866	0	6

Ilustración 18: Tabla con los nuevos valores de la imagen encriptada.

La matriz de la imagen original es la siguiente:

$$IN = \begin{bmatrix} in_{00} & in_{01} & in_{02} & in_{03} & in_{04} & in_{05} & in_{06} \\ in_{10} & in_{11} & in_{12} & in_{13} & in_{14} & in_{15} & in_{16} \\ in_{20} & in_{21} & in_{22} & in_{23} & in_{24} & in_{25} & in_{26} \\ in_{30} & in_{31} & in_{32} & in_{33} & in_{34} & in_{35} & in_{36} \\ in_{40} & in_{41} & in_{42} & in_{43} & in_{44} & in_{45} & in_{46} \\ in_{50} & in_{51} & in_{52} & in_{53} & in_{54} & in_{55} & in_{56} \\ in_{60} & in_{61} & in_{62} & in_{63} & in_{64} & in_{65} & in_{66} \end{bmatrix}$$

Con ayuda de la tabla de transformación de índices la matriz de salida seria la siguiente:

$$ENC = \begin{bmatrix} enc_{33} & enc_{35} & enc_{31} & enc_{34} & enc_{32} & enc_{36} & enc_{30} \\ enc_{53} & enc_{55} & enc_{51} & enc_{54} & enc_{52} & enc_{56} & enc_{50} \\ enc_{13} & enc_{15} & enc_{11} & enc_{14} & enc_{12} & enc_{16} & enc_{10} \\ enc_{43} & enc_{45} & enc_{41} & enc_{44} & enc_{42} & enc_{46} & enc_{40} \\ enc_{23} & enc_{25} & enc_{21} & enc_{24} & enc_{22} & enc_{26} & enc_{20} \\ enc_{63} & enc_{65} & enc_{61} & enc_{64} & enc_{62} & enc_{66} & enc_{60} \\ enc_{03} & enc_{05} & enc_{01} & enc_{04} & enc_{02} & enc_{06} & enc_{00} \end{bmatrix}$$

Cabe destacar que esta matriz es un ejemplo un tanto didáctico porque el proceso se realizara con tres iteraciones modificando los componentes de los pixeles mediante su complemento, al final de las iteraciones cada elemento de la matriz depende de muchos más datos, es más parecido a una lista múltiple o a un tensor.

3.2. Algoritmo *ICT* de desencriptación

Para el diseño del algoritmo de desencriptación se procederá inicialmente a tomar las condiciones iniciales, en otras palabras las entradas, la imagen encriptada es una de las entradas, la otra entrada es la misma clave usada para la encriptación, esto por tratarse de un algoritmo basado en encriptación simétrica.

Al estar basado el algoritmo en un tipo de encriptación de clave simétrica se realiza el mismo procedimiento para obtener las condiciones iniciales a partir de las entradas, es decir se utilizaran las mismas ecuaciones para generar x_0 y r:

$$x_0 = \frac{\sum_{i=0}^{39} k_i 2^i}{\sum_{i=0}^{39} 2^i}$$

$$r = \frac{\sum_{i=40}^{79} k_i 2^i}{\sum_{i=0}^{39} 2^i} 0.25848 + 3.56995$$

Además se consideran los mismos rangos.

$$x_0 = [0, 1]$$

$$r = [3.56995, 3.82843]$$

Pasando a la creación del algoritmo de transformación para la desencriptación, y por tratarse de una encriptación simétrica, el proceso es semejante al detallado para la encriptación, se usa el mismo mapa logístico con las mismas condiciones iniciales proveídas por las entradas, con estas condiciones iniciales se genera un proceso inverso de transformación.

4. Análisis de complejidad de los algoritmos *DES* e *ICT*

Para analizar la complejidad de los algoritmos se analizará ambos algoritmos mediante un análisis asintótico, estudiando de esta manera el comportamiento de ambos algoritmos cuando el tamaño de las entradas es lo suficientemente grande y obviando factores constantes y el comportamiento ante entradas pequeñas.

La notación asintótica que utilizamos para ambos algoritmos es la denominada notación o función O cuya definición es:

$$T(n)\ es\ O\big(f(n)\big) \Leftrightarrow \exists c \in \mathbb{R}, \exists n_0 \in \mathbb{N}\ tal\ que\ \forall n > n_0 \in \mathbb{N}, T(n) \leq cf(n)$$

En la hipótesis se señala al algoritmo *DES* como base de comparación en tiempo de procesamiento, por lo tanto es necesario dar a conocer la función O grande para el mencionado algoritmo.

$$O\big(f(n)\big) = O(n^4)$$

La función *f(n)* es calculada a partir del algoritmo *DES* que está basado en tablas de permutación las cuales transforman los datos secuencialmente, y es que el algoritmo *DES* realiza tantas permutaciones como bits contenga el archivo.

Para analizar la complejidad del algoritmo *ICT* se recurrirá al cálculo de complejidad basada en instrucciones del lenguaje de alto nivel, en la que cada instrucción del lenguaje se considera de costo unitario.

El resultado conseguido aplicando el método anteriormente mencionado es el siguiente:

$$t(n) = \sum_{i=0}^{x} (x+4)i + \sum_{j=0}^{y} (y+4)j + \sum_{i=0}^{x} 6yi + 12$$

Donde las variables *x* e *y* dependen de la siguiente ecuación:

$$n = 3xy$$

Reordenando y factorizando la ecuación *t(n)*, tenemos la siguiente expresión:

$$t(n) = (x + 6y + 4) \sum_{i=0}^{x} i + (y+4) \sum_{j=0}^{y} j + 12$$

Resolviendo las sumatorias se obtiene lo siguiente:

$$t(n) = (x + 6y + 4) \frac{(x+1)x}{2} + (y+4) \frac{(y+1)y}{2} + 12$$

Multiplicando y sumando la ecuación *t(n)* se obtiene:

$$t(n) = \frac{1}{2}(x^3 + y^3 + 5x^2 + 5y^2 + 4x + 4y + 6xy + 6x^2y) + 12$$

Para que la ecuación quede en función de *n*, asumimos lo siguiente:

$$x = cy$$

Donde *c* es una constante de proporcionalidad. Y la reemplazamos en las ecuaciones de *n* y *t(n)*, las cuales quedarían de la siguiente manera:

$$n = 3cy^3$$

$$t(n) = \frac{1}{2}(c^3y^3 + y^3 + 5c^2y^2 + 5y^2 + 4cy + 4y + 6cy^2 + 6c^2y^3) + 12$$

Simplificando la ecuación resulta de la siguiente forma:

$$t(n) = \frac{1}{2}(c^3 + 6c^2 + 1)y^3 + \frac{1}{2}(5c^2 + 6c + 5)y^2 + 2(c+1)y + 12$$

Ahora que toda la ecuación está en función de *y* realizamos el cambio de variable:

$$y = \left(\frac{n}{3c}\right)^{\frac{1}{3}}$$

Aplicando logaritmos y simplificando nos queda la siguiente expresión final:

$$t(n) = \frac{(c^3 + 6c^2 + 1)(5c^2 + 6c + 5)(c + 1)}{c^2} n^2$$

Claramente queda establecido que el orden de la función es $O(n^2)$, además se debe notar que existe una clara dependencia de la variable *c* que se había definido como la constante de proporcionalidad entre *x* e *y*.

5. Implementación del algoritmo

Para la implementación del algoritmo *ICT* se ha escogido el lenguaje de programación *Java*, este prototipo pretende servir como base para las pruebas que se realizarán más adelante, con el objetivo principal de demostrar la hipótesis, para tal efecto es necesario mencionar el algoritmo *DES*, es evidente que el propósito del presente trabajo no es desarrollar una implementación del algoritmo *DES* pero a su vez es necesario contar con una implementación para la comparación de tiempos de procesamiento de ambos algoritmos *DES* y *ICT*.

5.1. Algoritmo *DES*

Para la implementación del algoritmo *DES* se hace uso de una librería de *Java* denominada *crypto* esta librería nos permite realizar el cifrado y descifrado de distintos algoritmos que son estándares. Para este caso en particular se utiliza la instancia *DES* y se realiza la encriptación por bloques según el estándar *ISO-8859-1* especificado en el algoritmo.

A continuación se detalla una parte del código implementado para el algoritmo *DES*:

```
Cipher cifrado = Cipher.getInstance("DES");
if (cifrar.equals(args[0])) {
	cifrado.init(Cipher.ENCRYPT_MODE, sk);
}
if (descifrar.equals(args[0])) {
	cifrado.init(Cipher.DECRYPT_MODE, sk);
}
InputStream archivoEn = new FileInputStream(args[1]);
OutputStream archivoSa = new FileOutputStream(args[2]);
byte[] buff = new byte[1024];
byte[] bloque;
String textoCifrado = new String();
int finArch = -1;
int leidos;
leidos = archivoEn.read(buff);
while (leidos != finArch) {
	bloque = cifrado.update(buff, 0, leidos);
	textoCifrado = textoCifrado
		+ new String(bloque, "ISO-8859-1");
	leidos = archivoEn.read(buff);
}
archivoEn.close();
bloque = cifrado.doFinal();
textoCifrado = textoCifrado
	+ new String(bloque, "ISO-8859-1");
archivoSa.write(textoCifrado.getBytes("ISO-8859-1"));
```

5.2. Algoritmo *ICT*

Ahora se detallará la implementación del algoritmo *ICT*, para almacenar la imagen hacemos el uso de un espacio de memoria intermedia, también definimos un espacio de memoria intermedia para la imagen de salida, Java cuenta con una librería que nos ayuda a definir estos espacios de memoria y a la vez podemos definir qué tipo de imagen se almacenará en este espacio, para propósitos del algoritmo que se desea implementar, se utiliza el formato de imagen de 24 bits, que en se define como *TYPE_3BYTE_BGR*, resulta entonces el formato de 3 Bytes con el modelo de colores *RGB*.

Después de definir la imagen, pasamos a definir la transformación de la imagen mediante el mapa logístico de la siguiente forma:

```
x = mu * x0 * (1 - x0);
```

Pero es necesario que el valor generado por el mapa logístico se encuentre en el rango de las dimensiones de la imagen, para lo cual se realiza la siguiente operación:

```
y = (int) (x * (w - 1));
```

Es necesario cambiar el tipo de variable de *y* debido a la naturaleza discreta de la imagen.

Para la asignación de *y* se realiza mediante el uso de un bucle condicional, como se detalla a continuación:

```
while (fila[y] != 0) {
      if (y < w - 1)
            y++;
      else
            y = 0;
      }
fila[y] = i;
```

Con este bucle se genera la tabla de la Ilustración 18, básicamente asigna los valores convertidos de un dominio continuo a un dominio discreto, y como se observa puede ocurrir que la posición esté ocupada por lo que se intenta asignar a la siguiente posición y así hasta conseguir una posición vacía.

La misma operación se realiza para ambas dimensiones, y además es recursivo.

Una vez que obtenemos las tablas de transformación se procede a obtener los componentes de los pixeles.

Para esto se hace uso de un método que obtiene el color RGB, ya que los colores también deben ser modificados pixel por pixel. El método nos da como resultado un valor entero que representa el color RGB, para obtener los colores de manera individual se realizan las siguientes operaciones:

```
int r = (rgb & 16711680) >> 16;
int g = (rgb & 65280) >> 8;
int b = rgb & 255;
```

Los colores se obtienes como se observa con operaciones de desplazamiento y disyunción, aunque el concepto es más sencillo aun, el color está compuesto por tres bytes, el primer byte corresponde al color rojo, el segundo byte corresponde al color verde y el último byte corresponde a azul.

Después de realizar la transformación a los componentes del pixel, reconstruimos el pixel y lo almacenamos en su nueva posición:

```
encriptado.setRGB(fila[i], columna[j], rgb1);
```

En la anterior línea de código rgb1 es el pixel reconstruido a partir del anterior pixel y almacenado en la nueva posición obtenida a partir de la transformación de los mapas logísticos.

6. Análisis de resultados

6.1. Pruebas

Para analizar el funcionamiento del algoritmo *ICT*, se harán diversas pruebas considerando las dimensiones de la imagen, el formato, el comportamiento caótico y lo más importante el tiempo de encriptación, para la última prueba mencionada también se realizará dicha prueba con el algoritmo *DES*.

6.1.1. Pruebas de comprobación de eficiencia según el formato y las dimensiones

Para las primeras pruebas de tiempo de procesamiento se tomará en cuenta las dimensiones de la imagen y el formato, dos tipos de formato y cuatro tipos de dimensiones, en base a la siguiente tabla que muestra las dimensiones que serán tomadas en cuenta y el formato:

Formato de imagen	Dimensiones
JPG	1920 x 1080
JPG	1632 x 1224
JPG	3104 x 1746
JPG	3264 x 2448
BMP	1920 x 1080
BMP	1632 x 1224
BMP	3104 x 1746
BMP	3264 x 2448

Ilustración 19: Tabla de formatos y dimensiones de imagen.

Según la tabla anterior se pretende comprobar que el algoritmo trabaja con mayor eficiencia con imágenes con formatos en los que las imágenes no se comprimen.

6.1.2. Pruebas de calidad de imagen desencriptada según el formato

El algoritmo contempla en su diseño la ejecución por fases, es decir que la encriptación se la puede realizar con múltiples fases. Para realizar estas pruebas se tomarán en cuenta tres fases, en estas pruebas se mostrará los resultados por cada fase para comprobar la calidad de la imagen desencriptada o de salida por cada fase en relación con la original, los formatos consisten en dos uno de los cuales es a la vez un formato de compresión y el otro no.

La compresión de las imágenes juega un papel importante a la hora de realizar la encriptación, es por eso que surge la necesidad de realizar pruebas que muestren cuanto afecta este factor en el momento de recuperar la imagen original.

6.1.3. Pruebas de comportamiento caótico

Después de las anteriores pruebas se continuará con las pruebas de comportamiento caótico, es decir se variará el parámetro *r* del mapa logístico para analizar su comportamiento, para estas pruebas se utilizarán distintos valores para el parámetro *r*, como se detalla a continuación:

Tipo de imagen	**Parámetro *r***
BMP, 1920 x 1080	Entre 0 y 1
BMP, 1920 x 1080	Entre 1 y 2
BMP, 1920 x 1080	Entre 2 y 3
BMP, 1920 x 1080	Entre 3 y 3.44949
BMP, 1920 x 1080	Entre 3.44949 y 3.54409
BMP, 1920 x 1080	Entre 3.54409 y 3.56995
BMP, 1920 x 1080	Entre 3.56995 y 3.82843
BMP, 1920 x 1080	Más de 4

Ilustración 20: Tabla de comportamiento del parámetro r.

De acuerdo a la teoría el mapa caótico solo tiene comportamiento caótico en un intervalo, mediante las pruebas determinaremos este intervalo de manera práctica.

6.1.4. Pruebas de eficiencia del algoritmo *ICT* respecto al *DES*

Por último se realizarán pruebas a ambos algoritmos DES y *ICT* para comprobar los tiempos de procesamiento de ambos. Las imágenes para estas pruebas serán las mismas especificadas en la tabla de formatos y dimensiones (véase Ilustración 19), los tiempos serán promedios y estarán en milisegundos.

6.2. Resultados

6.2.1. Comprobación de eficiencia según el formato y las dimensiones

Después de realizar las primeras pruebas se han obtenido los siguientes resultados, detallados en la tabla, para el proceso de encriptación:

Formato de imagen	Dimensiones	Tiempo promedio
JPG	1920 x 1080	3411 ms
JPG	1632 x 1224	3348 ms
JPG	3104 x 1746	8066 ms
JPG	3264 x 2448	11586 ms
BMP	1920 x 1080	2921 ms
BMP	1632 x 1224	2797 ms
BMP	3104 x 1746	6505 ms
BMP	3264 x 2448	9440 ms

Ilustración 21: Tabla de tiempos de procesamiento según el tipo de imagen y dimensiones, utilizando el algoritmo ICT de encriptación.

Las mismas pruebas se realizaron para el proceso de desencriptación, dando como resultado los siguientes tiempos, que son promedios:

Formato de imagen	Dimensiones	Tiempo promedio
JPG	1920 x 1080	2668 ms
JPG	1632 x 1224	2789 ms
JPG	3104 x 1746	6546 ms
JPG	3264 x 2448	9172 ms
BMP	1920 x 1080	2287 ms
BMP	1632 x 1224	2138 ms
BMP	3104 x 1746	4650 ms
BMP	3264 x 2448	6604 ms

Ilustración 22: Tabla de tiempos de procesamiento según el tipo de imagen y dimensiones, utilizando el algoritmo ICT de desencriptación.

Se puede observar por los resultados de las tablas tanto al realizar la encriptación y la desencriptación que los tiempos están relacionados con las dimensiones, son proporcionales. Pero si comparamos las mismas dimensiones entre los dos formatos se ve claramente que el formato de imagen que utiliza compresión demora más que el formato de imagen sin compresión, esta demora es causada por el tiempo extra generado por la compresión y la descompresión.

6.2.2. Calidad de la imagen desencriptada según el formato

Continuando con las pruebas, se procedió con las pruebas de fases para determinar la calidad de la imagen después de los procesos de encriptación y desencriptación, en estas se utilizaron dos formatos *BMP* y *JPG*, antes es necesario mostrar en la siguiente ilustración la imagen original.

Ilustración 23: Imagen BMP original, antes de la encriptación.

A continuación se ilustra las imágenes encriptadas en tres fases del formato *BMP*.

Ilustración 24: Imagen BMP encriptada en fase 1.

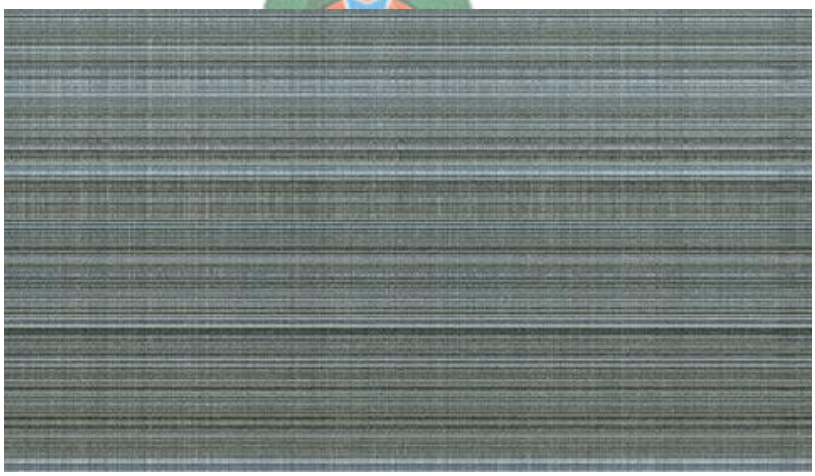

Ilustración 25: Imagen BMP encriptada en fase 2.

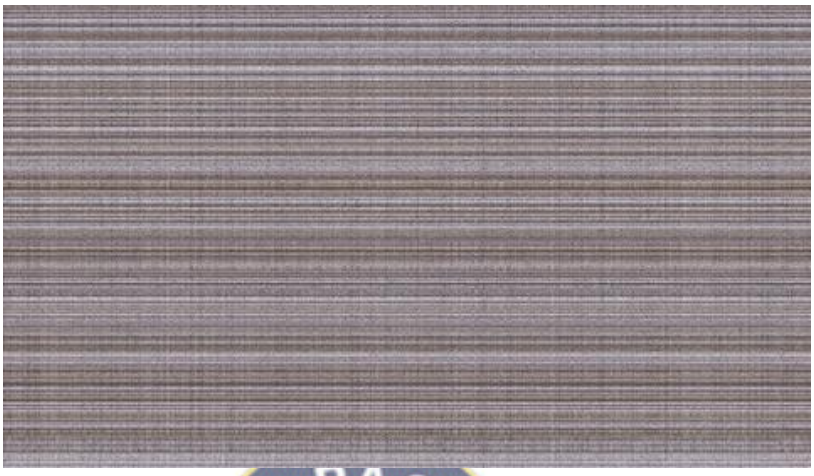

Ilustración 26: Imagen BMP encriptada en fase 3.

Las siguientes imágenes corresponden la imagen desencriptada, también en tres fases:

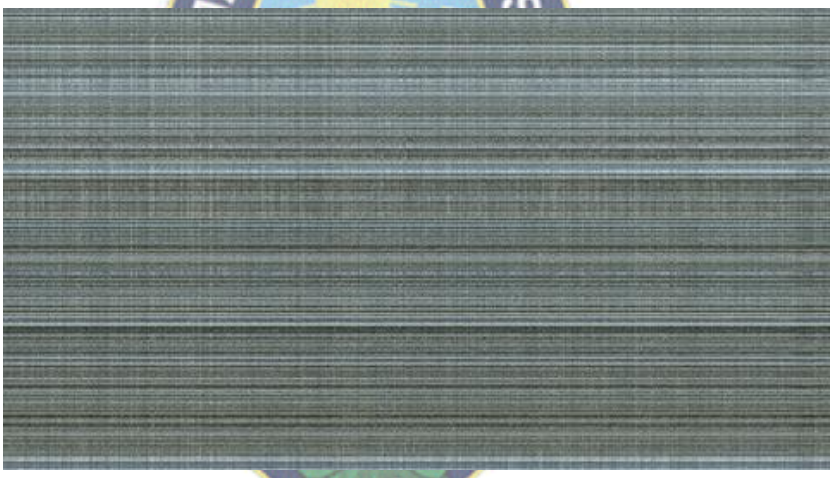

Ilustración 27: Imagen BMP desencriptada en fase 1.

Ilustración 28: Imagen BMP desencriptada en fase 2.

Ilustración 29: Imagen BMP desencriptada en fase 3.

A continuación se muestran los resultados de las pruebas realizadas a una imagen en formato *JPG*, utilizando el algoritmo de encriptación, también en tres fases, es además resultado de la imagen original (véase Ilustración 23) con excepción del formato, que en este caso es *JPG*:

Ilustración 30: Imagen JPG encriptada en fase 1.

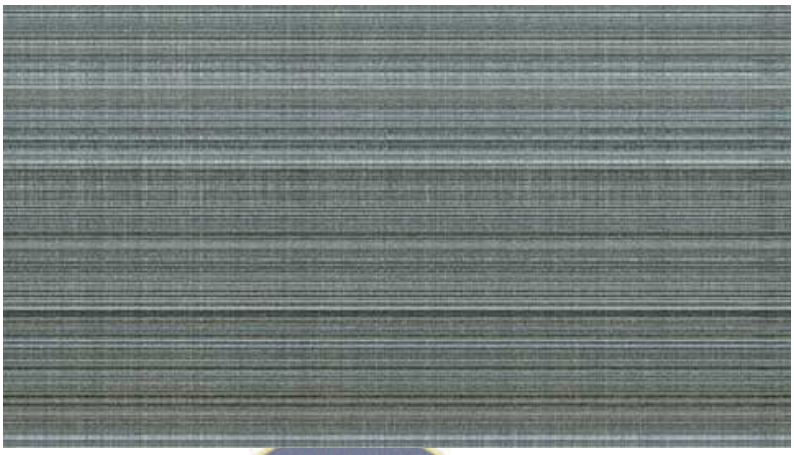

Ilustración 31: Imagen JPG encriptada en fase 2.

Ilustración 32: Imagen JPG encriptada en fase 3.

Ahora se detallan las imágenes desencriptadas con resultados que se analizaran en las conclusiones:

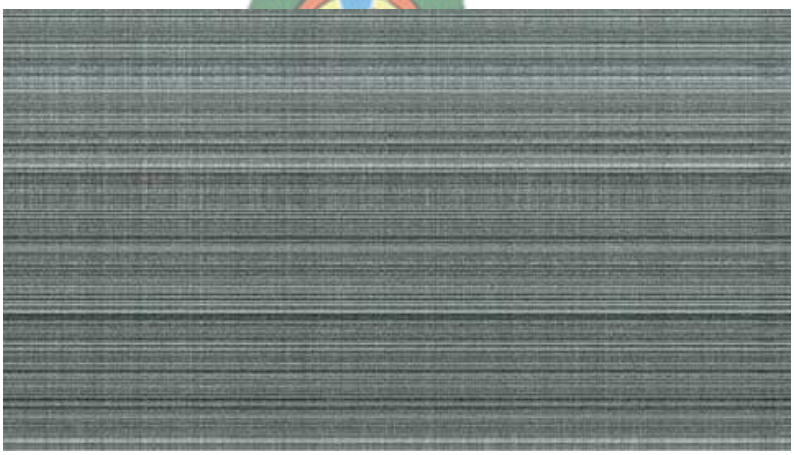

Ilustración 33: Imagen JPG desencriptada fase 1.

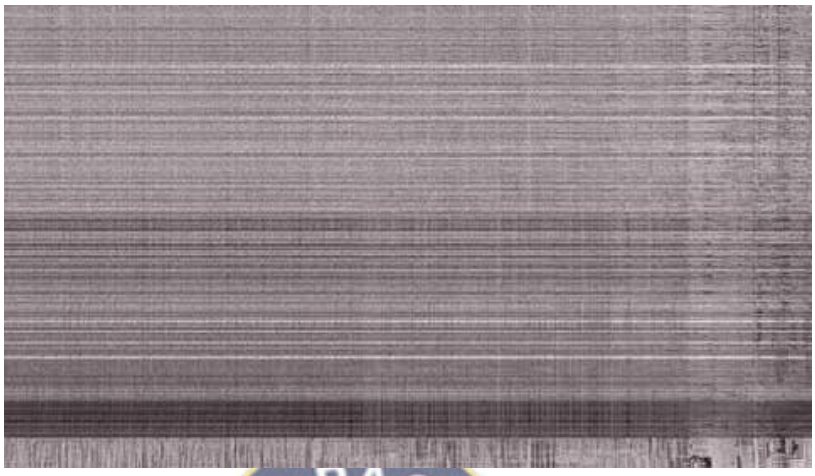

Ilustración 34: Imagen JPG desencriptada fase 2.

Ilustración 35: Imagen JPG desencriptada fase 3.

Después de las pruebas realizadas se determina que las imágenes *JPG* sufren una notable pérdida de calidad respecto a la imagen original, esto es ocasionado nuevamente por la compresión de la imagen. Por otro lado la imagen *BMP* que resulta después del proceso de desencriptación no muestra ninguna pérdida de calidad.

6.2.3. Comportamiento caótico

Después de realizar pruebas por formatos, se realizan las pruebas para comprobar el comportamiento caótico, para dichas pruebas se ha tomado en cuenta la tabla de comportamiento del parámetro r (véase Ilustración 20).

Considerando el valor del parámetro *r* entre cero y uno, además de considerar solo una fase de encriptación para hacer más evidente el comportamiento caótico, se

obtienen los siguientes resultados, demostrando en este intervalo la tendencia del mapa caótico a un valor estático.

Ilustración 36: Imagen encriptada con un valor de r igual a 0,5.

A continuación se muestra la imagen encriptada con un parámetro r con un valor entre uno y dos, el mapa caótico rápidamente tiende a un valor constante denotado por $(r-1)/r$.

Ilustración 37: Imagen encriptada con un valor de r igual a 1,5.

En el siguiente intervalo comprendido entre dos y tres el comportamiento del mapa caótico es el mismo que el anterior $(r-1)/r$ con la diferencia que en este caso primero fluctúa alrededor de ese valor.

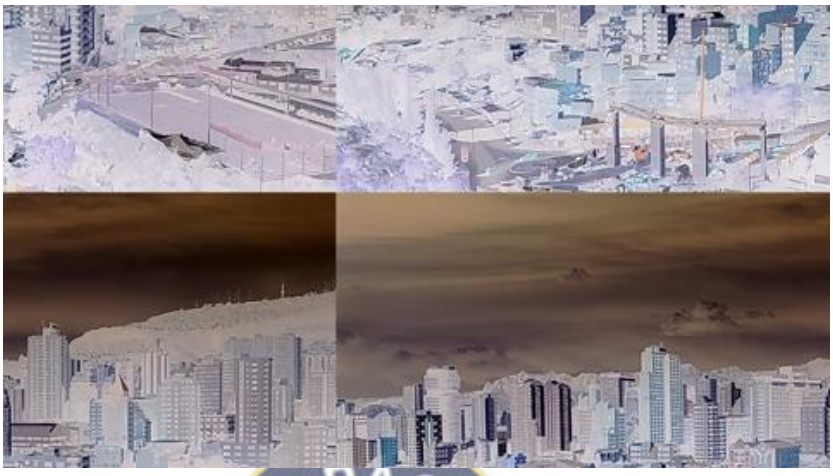

Ilustración 38: Imagen encriptada con un valor de r igual a 2,5.

Los valores comprendidos entre 3 y 3.44949 provocan oscilaciones en el comportamiento del mapa caótico, estas oscilaciones están comprendidas entre dos valores asintóticos dependientes del valor del parámetro *r*.

Ilustración 39: Imagen encriptada con un valor de r igual a 3,2.

El comportamiento del mapa caótico para los valores de *r* comprendidos entre 3.44949 y 3.54409 aproximadamente, hacen que el mapa logístico se comporte como un oscilador alrededor de cuatro valores.

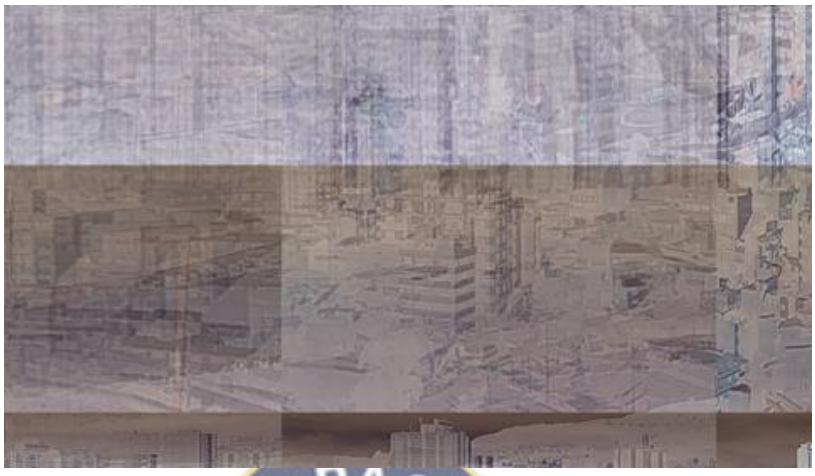

Ilustración 40: Imagen encriptada con un valor de r igual a 3,49.

El mapa logístico con *r* entre 3.54409 y 3.56995 empieza a tener oscilaciones entre 8 valores, luego 16, luego 32, y así sucesivamente, este comportamiento es comparable a una cascada de periodo doble.

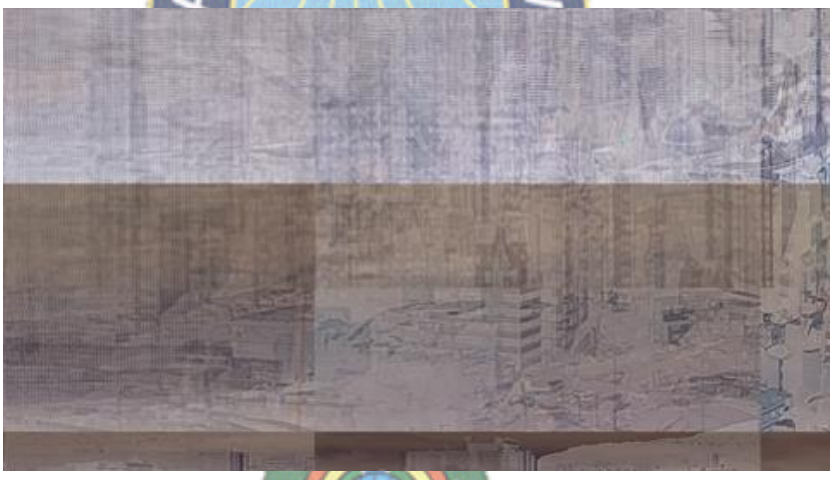

Ilustración 41: Imagen encriptada con un valor de r igual a 3,55.

A continuación se muestra la imagen encriptada con un mapa logístico operando con un valor de *r* entre 3.56995 y 3.82843, cuyo comportamiento es aperiódico, este espacio es claramente el esperado para generar caos.

Ilustración 42: Imagen encriptada con un valor de r igual a 3,82.

Para los valores posteriores a los analizados hasta ahora el comportamiento del mapa caótico es nuevamente oscilatorio y los periodos se dividen en sub-rangos, esta secuencia de sub-rangos es llamada cascada de harmónicos.

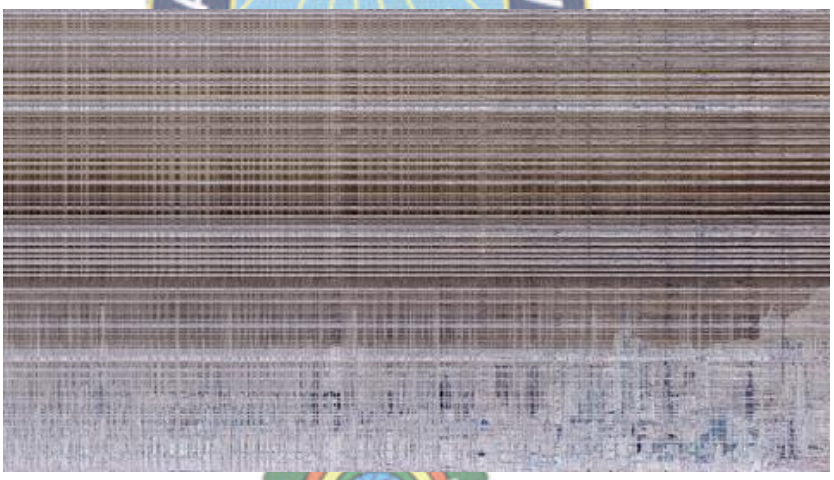

Ilustración 43: Imagen encriptada con un valor de r igual a 4.

Con los resultados se demuestra el comportamiento caótico, además de validar los intervalos teóricos de comportamiento caótico.

6.2.4. Eficiencia del algoritmo *ICT* respecto al *DES*

Finalmente se presentan los resultados de la prueba de eficiencia del algoritmo de encriptación *ICT* respecto al *DES*, a continuación se detalla la tabla de resultados que consiste en formatos, dimensiones y tamaño para ambos procesos, encriptación y desencriptación:

Formato de imagen	Dimensiones	Tamaño [KB]	Tiempo promedio [ms]	
			Algoritmo *DES*	Algoritmo *ICT*
JPG	1920 x 1080	982	13937	3372
JPG	1632 x 1224	993	13286	3997
JPG	3104 x 1746	2495	91913	8210
JPG	3264 x 2448	3307	170885	12169
BMP	1920 x 1080	6076	566846	2755
BMP	1632 x 1224	5853	475726	2860
BMP	3104 x 1746	15878	1026918966	6576
BMP	3264 x 2448	23410	209148442384	9310

Ilustración 44: Tabla de comparación de tiempos promedio entre ICT y DES para el proceso de encriptación.

Formato de imagen	Dimensiones	Tamaño [KB]	Tiempo promedio [ms]	
			Algoritmo *DES*	Algoritmo *ICT*
JPG	1920 x 1080	982	14916	2693
JPG	1632 x 1224	993	14955	2802
JPG	3104 x 1746	2495	98005	6229
JPG	3264 x 2448	3307	175237	8485
BMP	1920 x 1080	6076	517585	2101
BMP	1632 x 1224	5853	493091	2302
BMP	3104 x 1746	15878	1099054552	5016
BMP	3264 x 2448	23410	223840006214	6495

Ilustración 45: Tabla de comparación de tiempos promedio entre ICT y DES para el proceso de desencriptación.

Como se puede observar en las tablas anteriores ambos procesos, encriptación y desencriptación, tienen tiempos parecidos. En las tablas también se puede observar que el tiempo del algoritmo *DES* es directamente proporcional al tamaño de la imagen, mientras que también se puede observar que el algoritmo *ICT* depende de las dimensiones de la imagen, esta dependencia le da una ventaja en tiempos de procesamiento.

En las tablas se observa además que el algoritmo *ICT* no depende del tamaño de la imagen ya que este opera con conceptos abstractos que reducen la dependencia de la imagen con el tamaño físico.

Cabe aclarar que los resultados obtenidos por el algoritmo *DES* en los casos de imágenes con tamaños superiores a 10 MB son generados con la ayuda de la siguiente función empírica:

$$t_{proceso} = t_{referencia} * 2^{(\frac{T_{proceso}}{T_{referencia}})}$$

Donde $t_{proceso}$ es el tiempo que dura el proceso para una determinada imagen, $t_{referencia}$ es el tiempo promedio de una imagen de referencia con un tamaño menor a 10 MB, $T_{proceso}$ es el tamaño de la imagen de referencia y $T_{referncia}$ es el tamaño de la imagen de la cual se desea calcular el tiempo.

Con ayuda de la función se calculan los últimos dos tiempos de la tabla para el algoritmo *DES* dando como resultado cifras exorbitantes, la primera de ellas resulta en más de once días y la segunda en aproximadamente seis años.

Quedando demostrado de esta manera la superioridad del algoritmo *ICT* sobre el algoritmo *DES*, tomando en cuenta que el algoritmo *ICT* está diseñado exclusivamente para realizar la encriptación y desencriptación de imágenes, estando en desventaja el algoritmo *DES* que no está diseñado de forma exclusiva para el tratamiento de imágenes.

7. Conclusiones y recomendaciones

7.1. Conclusiones

Durante el desarrollo del presente trabajo, que propone un algoritmo de encriptación aplicando mapas caóticos, denominado algoritmo de encriptación de imágenes *ICT*, se hacía presente la incertidumbre sobre la eficiencia del mencionado algoritmo en comparación del algoritmo *DES*, tal incertidumbre iba desapareciendo a medida que el trabajo tomaba forma y finalmente desapareció tras las pruebas realizadas, estas pruebas afianzan toda la teoría en la que se sustenta el algoritmo *ICT*.

Las primeras conclusiones sobre el algoritmo surgen con los resultados de la comprobación de eficiencia del algoritmo según el formato y las dimensiones, estos resultados demuestran que el algoritmo es dependiente de las dimensiones de la imagen más que de su tamaño físico, el algoritmo está diseñado de manera tal que las imágenes siempre son transformadas a imágenes de 24 bits de resolución de color, esto hace que la abstracción de la imagen juegue un papel importante. Otro punto importante al respecto es el tiempo adicional que requieren las imágenes cuyos formatos incluyen compresión, este tiempo adicional no influye de manera drástica en los tiempos totales, por tanto la eficiencia del algoritmo radica en las dimensiones de la imagen, lo que hace que el algoritmo sea tarde menos en los procesos de encriptación y desencriptación que un algoritmo de texto plano.

La calidad de imagen también es analizada y se realizaron pruebas al respecto, resulta por demás decir que la calidad de la imagen a la salida del proceso de encriptación no es necesaria por tratarse de una imagen cifrada, pero es de suma importancia la imagen que resulta del proceso de desencriptación porque solo ahí es evidente si el algoritmo cumple con una propiedad muy importante de la criptografía la cual es denominada integridad, esta propiedad garantiza la corrección y completitud de la información. Los resultados producen dos conclusiones importantes; la primera correspondiente a las imágenes sin compresión, en este tipo de imágenes la imagen de salida es la misma que la imagen de la entrada; la segunda conclusión está relacionada con las imágenes comprimidas, si bien las formas de la imagen como tal no sufren cambios no se puede decir lo mismo de los

colores, es decir los pixeles de la imagen de salida se ubican en los mismos sitios que los pixeles de imagen original, pero los colores sufren una grave pérdida relacionada con la compresión. Uniendo ambas conclusiones se puede decir que el algoritmo si bien es eficiente en tiempos en cualquier formato, la integridad se ve afectada en imágenes que utilizan compresión.

Uno de los objetivos específicos del presente trabajo es demostrar el comportamiento caótico de dicho algoritmo, para esto se tomaron valores teóricos de las condiciones iniciales y fueron comprobadas de forma práctica, estos resultados como era de esperarse nos dan los límites inferior y superior correspondientes a la teoría, en conclusión el algoritmo *ICT* se comportará de manera caótica si y solo si el valor del parámetro de entrada *r* se encuentra entre 3.56995 y 3.82843.

Todos estos resultados nos sirven como referencia y justificación a la hora de comparar el algoritmo *DES* con el algoritmo *ICT*, los resultados demuestran la supremacía del algoritmo *ICT* sobre el algoritmo *DES*, esto es debido al diseño del algoritmo *ICT* que lleva de manera inherente a la imagen, por otro lado el algoritmo *DES* tiene la desventaja de no estar diseñado específicamente para la encriptación de imágenes, esto está demostrado con los resultados de la pruebas en las que el formato de la imagen y las dimensiones no afectaban de manera directa a su desempeño, por otro lado el tamaño físico de la imagen si era un factor importante, los resultados expuestos en las tablas de comparación (véase Ilustración 44 e Ilustración 45) son elocuentes. En cuanto a el algoritmo *ICT* se ve una clara dependencia en las dimensiones y el formato, los cuales son factores que ayudan a lograr un tiempo de procesamiento mucho menor.

En conclusión el algoritmo *ICT* resulta con tiempos menores por teorías y conceptos introducidos en su diseño, estos conceptos son la abstracción y la teoría de sistemas dinámicos no lineales. Y resulta ideal para imágenes sin compresión.

7.2. Recomendaciones

En el presente trabajo se establece la funcionalidad del algoritmo para imágenes con distintas resoluciones y también con compresión y sin compresión, pero sería recomendable hacer un análisis en profundidad del algoritmo de compresión cuyo

corazón o esencia es la Transformada Discreta de Coseno, resultaría interesante analizar y corregir los problemas del algoritmo *ICT* frente a las imágenes que utilizan compresión.

Se ha comprobado que el algoritmo *ICT* tiene un buen rendimiento en cuanto a tiempo de ejecución, pero eso no significa ni asegura que sea inquebrantable, para lo cual sería interesante desarrollar métodos teóricos para comprobar la robustez del algoritmo frente a ataques, existen varios métodos y teorías, como la estadística o la teoría de la información.

En la generación de clave también se podría utilizar algoritmos más complejos para elevar la seguridad, la seguridad en un algoritmo de esta índole es vital, por lo que producir métodos para analizar este tipo de algoritmos serían valiosos aportes para el campo de la criptografía.

El presente trabajo sirve como base para investigaciones futuras, como el diseño de un algoritmo aplicando la teoría del caos para texto plano, o la forma de documentos en imágenes, siempre y cuando transmitir la imagen resulte más eficiente que la transmisión de datos en general.

8. Bibliografía

Colina, J. (1999). Teoría del caos, el tercer paradigma: una aplicación de las series pseudoaleatorias para modelizar la dinámica no lineal.

Coppo, J. A. (2010). Teoría del caos y metodo científico.

Gutzwiller, M. (1990). *Chaos in classical and quantum mechanics.* New York: Springer-Verlag.

Matthews, R. (1989). *On the derivation of chaotic encryption algorithm.*

Millérioux, G., Hernández, A., & Amigó, J. M. (2006). Criptografía caótica con reinyección de la información. *HAL*.

Montero, M. (2010). *Teoría del caos*. Obtenido de http://marialuisamontero.blogspot.com

Moon, F. (1990). *Chaotic and fractal dynamics.* New York: Springer-Verlag.

Moreno, J., Parra, F., Huérfano, R., Suárez, C., & Amaya, I. (2016). Modelo de encriptación simétrica basada en atractores caóticos. *Revista Ingeniería*, 378-390.

MSDN. (12 de Marzo de 2014). *Windows Metafile Format*. Obtenido de https://msdn.microsoft.com

Pareek, N., Patidar, V., & Sud, K. (2006). Image encryption using chaotic logistic map. *Elsevier*, 926-934.

Pastor, J., Sarasa, M., & Salazar, J. (1998). *Criptografía digital: fundamentos y aplicaciones.* Prensas Universitarias de Zaragoza.

Prigogine, I. (1993). *¿Tan sólo una ilusión? Una exploración del caos al orden.* Barcelona: Tusquets.

Sathishkumar, G., Bhoopathy bagan, K., & Sriraam, N. (2011). Image encryption based on difussion and multiple chaotic maps. *International Journal of Network Security & Its Applications*, 181-194.

Schmitz, R. (2001). Use of chaotic dynamical systems in cryptography. *Journal of the Franklin Institute*, 429-441.

Skinner, J., Molnar, M., Vybirar, T., & Mitra, M. (1992). Application of chaos theory to biology and medicine. *Integral Physiological Behaviour Science*.

Support, M. H. (14 de Mayo de 2015). *DIBs and Their Uses*. Obtenido de https://msdn.microdoft.com