



Casa abierta al tiempo

**UNIVERSIDAD AUTÓNOMA METROPOLITANA**  
**Unidad Iztapalapa**

División de Ciencias Básicas e Ingeniería

Tesis:

CIFRADO DE IMÁGENES Y DATOS  
UTILIZANDO EL MODELO CAÓTICO DE LORENZ

elaborada por:

**ERIKA NANCY LEOS RODRÍGUEZ**

Maestría en Ciencias Matemáticas  
Aplicadas e Industriales

Asesor:

Dr José Noé Gutiérrez Herrera

Sinodales:

Presidente: Dr. Horacio Tapia Recillas

Secretario: Dr. José Noé Gutiérrez Herrera

Vocal: Dra. Gina Gallegos García



# Agradecimientos

Quiero comenzar agradeciendo al Consejo Nacional de Ciencia y Tecnología (CONACYT), No. de beca 398908, por el apoyo brindado que hizo posible concluir mis estudios de maestría. A mi asesor el **Dr. José Noé Gutiérrez Herrera** por su guía y atención brindada a mi trabajo y al interés mostrado para poder concluirla. También agradezco a mi profesor el **Dr. Horacio Tapia Recillas** por su inspiración y pasión por lo que hace. A mis profesores de la UAM Iztapalapa por la formación brindada y el nivel de su enseñanza. A mi madre **Hortencia** por su formación y valores y a mi padre **Antonio** por ser mi guía académica y de vida. A mi amigo y compañero de la maestría **Alejandro** por sus consejos tanto en ésta tesis como fuera de ella.

A **José Manuel** por motivarme a terminar este ciclo y porque en el tiempo compartido aprendí más de mi misma y de la persona que quiero ser.

A mis sinodales por la cuidadosa revisión y comentarios acerca de mi trabajo para mejorarlo y por aceptar dedicarle el tiempo necesario. Agradezco al Departamento de Matemáticas de la UAM-I y al Laboratorio de códigos y criptografía por los recursos y herramientas brindados.

A mi prima y amiga **Nadya** y a mi segunda familia **Grecia** y **Pavel** por estar siempre a mi lado en todo proceso a pesar de todo.

*A mis Padres Antonio y Hortencia, esto es por y para ustedes.*

# Índice general

<b>Introducción</b>	<b>1</b>
<b>1. Sistemas Dinámicos</b>	<b>3</b>
1.1. Conceptos básicos . . . . .	3
1.2. Sistemas dinámicos discretos . . . . .	5
1.2.1. Puntos fijos . . . . .	6
1.2.2. Estabilidad de puntos fijos . . . . .	6
1.2.3. Estabilidad de órbitas periódicas . . . . .	8
1.3. Conjuntos invariantes y variedades . . . . .	9
1.3.1. Variedad estable e inestable . . . . .	10
<b>2. Caos</b>	<b>13</b>
2.1. Caos: Definición matemática . . . . .	13
2.2. Principales Características de un Sistema Caótico . . . . .	15
2.3. Exponente de Lyapunov . . . . .	21
2.3.1. Dimensiones superiores . . . . .	23
2.4. Órbitas periódicas y asintóticamente periódicas . . . . .	25
2.5. Órbitas caóticas . . . . .	27
<b>3. El Sistema Caótico de Lorenz</b>	<b>31</b>
3.1. El atractor extraño . . . . .	35
3.2. Cálculo de los Exponentes de Lyapunov . . . . .	37
<b>4. Criptografía basada en sistemas caóticos</b>	<b>41</b>
4.1. Definición y Terminología . . . . .	42
4.2. Historia . . . . .	43
4.3. Criptografía basada en Caos . . . . .	53

<b>5. Cifrado utilizando el modelo caótico de Lorenz</b>	<b>57</b>
5.1. Algoritmo para la generación de subllaves . . . . .	65
5.2. Etapas y esquema del cifrado de imágenes . . . . .	67
5.3. Modos de operación . . . . .	68
5.4. Descripción del descifrado . . . . .	71
5.5. Cifrado de datos . . . . .	72
5.6. Descifrado de datos . . . . .	76
5.7. Pseudocódigos . . . . .	76
5.7.1. Cifrado de datos . . . . .	79
5.7.2. Descifrado de datos . . . . .	79
<b>6. Criptoanálisis del cifrado</b>	<b>83</b>
6.1. Criptoanálisis Diferencial . . . . .	83
<b>7. Robustez del cifrado de imágenes y datos</b>	<b>89</b>
7.1. Análisis estadístico . . . . .	89
7.1.1. Histogramas . . . . .	90
7.1.2. Correlación de píxeles . . . . .	91
7.2. Sensibilidad de la llave . . . . .	92
7.3. Sensibilidad del cifrado . . . . .	93
7.4. Comprobación del descifrado . . . . .	96
7.5. Tiempos de ejecución . . . . .	96
<b>Apéndice</b>	<b>100</b>
<b>A. Cifrado de imágenes</b>	<b>101</b>
<b>B. Cifrado de datos</b>	<b>107</b>
<b>C. Código para calcular Lyapunov</b>	<b>113</b>
<b>D. Código para calcular correlación</b>	<b>115</b>
<b>Bibliografía</b>	<b>119</b>

# Introducción

La transmisión de imágenes se ha vuelto popular en el campo de la comunicación moderna. Una imagen puede contener gran cantidad de información privada, restringida y confidencial, por lo que es vulnerable a la divulgación no autorizada o modificaciones durante el almacenamiento o transmisión. La información puede ser adquirida por personal no autorizado con fines de desacreditación, robo o suplantación de identidad, por lo que es importante cifrar una imagen antes de su transmisión y garantizar la seguridad del cifrado.

Los sistemas dinámicos tienen cada vez más importancia en diversas disciplinas como la biología, la medicina, la computación y en la misma matemática para representar modelos que nos ayuden a entender o simular mejor un problema. Cuando la dinámica de un sistema presenta cambios cualitativos de forma desordenada que a pesar de ser provocados por funciones deterministas produzcan cambios impredecibles y además el sistema exhibe sensibilidad a condiciones iniciales decimos que el sistema es caótico.

Los sistemas caóticos por su naturaleza aleatoria y su sensibilidad a condiciones iniciales se volvieron atractivos para la creación de cifrados de datos e imágenes.

Los ataques a los sistemas criptográficos se han vuelto más eficaces y sofisticados las últimas décadas lo que ocasiona un problema para la seguridad de la información y obliga a la comunidad criptográfica a desarrollar nuevas técnicas para garantizar la seguridad en los cifrados propuestos. Analizar las características de los sistemas caóticos ha permitido cifrar datos explotando las propiedades de sistemas caóticos conocidos. Uno de los más conocidos por su amplia literatura es el sistema caótico de Lorenz, cuya fama aumentó durante las décadas que siguieron a 1960 y ha motivado desde entonces nume-

rosas aplicaciones en el ámbito computacional y de la criptografía.

El principal objetivo de este trabajo es el estudio del sistema caótico de Lorenz como una herramienta para diseñar un cifrado en bloques para datos e imágenes que resulte seguro y computacionalmente sencillo. Se propone un algoritmo en python que otorga seguridad suficiente en datos y en imágenes de acuerdo a las pruebas de robustez realizadas y logra mostrar que los sistemas dinámicos caóticos pueden ser una buena alternativa para la criptografía.

La seguridad en la tecnología de la información sigue siendo amenazada mientras la misma tecnología avanza por lo que es importante seguir en búsqueda de más herramientas matemáticas que junto con las técnicas criptográficas ayuden a hacer más segura la comunicación moderna.

La seguridad en la tecnología de la información adquiere cada vez más interés desde que el envío de datos, mensajes e imágenes se ha vuelto vulnerable a la divulgación no autorizada o modificaciones durante el almacenamiento o transmisión de las mismas con fines de desacreditación, robo, suplantación de identidad, entre otros, por lo que es importante seguir en búsqueda de más herramientas matemáticas que junto con las técnicas criptográficas ayuden a hacer más segura la comunicación moderna.

# Capítulo 1

## Sistemas Dinámicos

En este capítulo se recuerdan algunos conceptos básicos sobre sistemas dinámicos que serán útiles en el desarrollo del presente trabajo. Un estudio más detallado de estos temas pueden consultarse en [2] y en [28]. Se presenta la definición de un sistema dinámico, puntos fijos y órbitas periódicas así como la estabilidad de los mismos. También se abordan algunos ejemplos de sistemas caóticos discretos para estudiar sus cambios cualitativos a través del tiempo y los conceptos de conjunto invariante, variedades y bifurcación, necesarios para comprender la dinámica compleja de un sistema que nos interesa para este trabajo. Se elaboran algunas gráficas y diagramas para comprender dichos cambios que dan lugar lo que llamamos un sistema caótico.

### 1.1. Conceptos básicos

En la literatura hay distintas formas de definir un sistema dinámico, la idea intuitiva es que un sistema dinámico describe el recorrido de todos los puntos en un espacio dado que varían a través del tiempo bajo ciertas reglas establecidas.

Pensemos en la función  $f : \mathbb{R} \rightarrow (0, \infty)$  definida por:

$$f(x) = e^{\lambda x} \tag{1.1}$$

Para  $x = 1, 2, \dots$ ,  $f(1) = e^\lambda$ ,  $f(2) = e^{2\lambda}$ . Es fácil comprobar que podemos relacionar la función con la siguiente ecuación diferencial,

$$\dot{x} = \lambda x$$

Podemos resolver la ecuación anterior con ayuda del método de separación de variables, la solución está dada por

$$x(t) = ce^{\lambda t}, \quad c = x(0).$$

Ahora, consideremos el sistema

$$\dot{x} = Ax \tag{1.2}$$

donde

$$A = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$$

con  $a, b \in \mathbb{R}$ . Resolviendo con separación de variables este sistema, obtenemos que la solución es de la forma:

$$\begin{aligned} x_1(t) &= c_1 e^{at} \\ x_2(t) &= c_2 e^{bt}, \quad c = x(0) \end{aligned} \tag{1.3}$$

Un *sistema dinámico* es una función  $\phi_t(x)$  definida para todo  $t \in \mathbb{R}$  y  $x \in E \subset \mathbb{R}^n$  que describe la manera en que los puntos  $x \in E$  se mueven con respecto al tiempo.

Demos una definición formal de sistema dinámico. (Ver [23], Pág. 181)

**Definición 1.1.1.** *Suponga que  $E$  es un subconjunto abierto de  $\mathbb{R}^n$ . Un sistema dinámico en  $E$  es una función de clase  $C^1$* <sup>1</sup>

$$\phi : \mathbb{R} \times E \rightarrow E \tag{1.4}$$

donde  $\phi_t(x)$  satisface

1.  $\phi_0(x) = x$  para todo  $x \in E$
2. Es aditiva con respecto al primer argumento, es decir,  $\phi_t \circ \phi_s(x) = \phi_{t+s}(x)$  para todo  $s, t \in \mathbb{R}$  y  $x \in E$

---

<sup>1</sup>Una función es de clase  $C^1$  si sus derivadas parciales son continuas [23].

El sistema 1.2 define un sistema dinámico  $\phi : \mathbb{R} \times \mathbb{R}^2 \rightarrow \mathbb{R}^2$  que está dado por

$$\phi(t, c) = \begin{bmatrix} e^{at} & 0 \\ 0 & e^{bt} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$$

Además, si  $\phi_t(x)$  es un sistema dinámico en  $E \subset \mathbb{R}^n$  entonces la función

$$f(x) = \frac{d}{dt}\phi_t(x) \quad (1.5)$$

define un campo vectorial <sup>1</sup> en  $E$  y para cada  $x_0 \in E$ ,  $\phi_t(x_0)$  es la solución del problema de valor inicial

$$\begin{aligned} \dot{x} &= f(x) \\ x(0) &= x_0 \end{aligned} \quad (1.6)$$

Por otro lado, si  $\phi_t$  es un sistema dinámico en un conjunto abierto  $E \subset \mathbb{R}^n$ , entonces la función  $\mathbf{F} : E \rightarrow E$  definida por  $\mathbf{F}(x) = \phi_t(x)$  donde  $t$  representa lapsos fijos de tiempo en  $\mathbb{R}$ , define un *sistema dinámico discreto* que consiste en iteraciones de  $\mathbf{F}$ , es decir, para cada punto de  $E$  obtenemos una secuencia de puntos  $\mathbf{F}(x), \mathbf{F}^2(x), \mathbf{F}^3(x), \dots$  [23] llamados *estados* del sistema y al espacio determinado por  $x$  es llamado *espacio fase*.

Además, para  $x \in E$  la función  $\phi(\cdot, x) : \mathbb{R} \rightarrow E$  define una trayectoria u *órbita* de la Ec.(1.6) a través de  $x_0 \in E$  que se define como:

$$\Gamma_{x_0} = \{x \in E \mid x = \phi(t, x_0), t \in \mathbb{R}\}$$

Se debe precisar que un sistema dinámico es determinista, es decir, el conocimiento del estado del sistema en algún momento permite conocer su pasado y futuro de forma totalmente determinada.

## 1.2. Sistemas dinámicos discretos

Volvamos a la función (1.5), donde  $f : E \rightarrow E$ ,  $E \subseteq \mathbb{R}^n$ . Podemos definir el siguiente sistema dinámico discreto:

---

<sup>1</sup>Un campo vectorial en  $\mathbb{R}^n$  es una función  $\mathbf{F} : U \subset \mathbb{R}^n \rightarrow \mathbb{R}^n$  que asigna a cada punto  $x$  en su dominio  $U$  un vector  $\mathbf{F}(x)$ .

$$x_{t+1} = f(x_t) \quad (1.7)$$

$x_0$  el estado inicial y los estados futuros se obtienen de iterar la función  $f$ , es decir,  $x_1 = f(x_0)$ ,  $x_2 = f(x_1) = f(f(x_0))$ ,  $\dots$ ,  $x_n = f^n(x_0)$ , donde  $f^n$  es la  $n$ -ésima iteración de la función  $f$  que equivale a la composición:

$$f^n = \underbrace{f \circ f \circ \dots \circ f}_{n \text{ veces}}$$

Podemos definir la órbita  $\mathbb{O}(x)$  de  $x$  como el conjunto de puntos,  $\mathbb{O}(x) = \{x, f(x), f^2(x), \dots\} = \{f^n(x)\}$ , donde  $n \in \mathbb{Z}^+$ . En el caso del sistema (1.7),  $x \in \mathbb{R}^n$  por lo tanto la órbita  $\mathbb{O}(x)$  es un subconjunto de  $\mathbb{R}^n$  también.

### 1.2.1. Puntos fijos

Un *punto fijo* del sistema (1.7) es un  $x^* \in \mathbb{R}^n$  tal que  $f(x^*) = x^*$ . Además si existe  $n > 0$  para el cual  $f^n(x^*) = x^*$  y  $f^k(x^*) \neq x^*$  para cada  $k \leq n$ , con  $k > 0$ , entonces se dice que el punto  $x^*$  es un *punto periódico de periodo  $n$* . El periodo más pequeño para el cual un punto es periódico se llama *primer periodo*.

### 1.2.2. Estabilidad de puntos fijos

Consideremos el sistema  $x_{t+1} = f(x_t)$  donde  $f : E \rightarrow E$ ,  $E \subset \mathbb{R}$  y  $f$  es de clase  $\mathbb{C}^1$ . La estabilidad de un punto fijo de  $f$  en general depende del valor de la derivada de  $f$  en el punto fijo. Para analizar esto introducimos una pequeña perturbación en el punto fijo  $x^* = f(x^*)$ , tan pequeña como deseemos, que nombraremos  $\delta_0$  para obtener el punto inicial,  $x^* + \delta_0$ . La intención ahora es calcular la perturbación en la siguiente iteración, llamémosle  $\delta_1$ . Así tenemos que:

$$f(x^* + \delta_0) = x^* + \delta_1$$

Usando la expansión de Taylor se tiene

$$f(x^* + \delta_0) = f(x^*) + f'(x^*)\delta_0 + O(\delta_0^2)$$

como  $\delta_0$  es suficientemente pequeño, el término  $O(\delta_0^2)$  no tiene influencia sobre la estabilidad y puede ser despreciado. Entonces

$$f(x^* + \delta_0) = f(x^*) + f'(x^*)\delta_0$$

como  $x^*$  es punto fijo,

$$f(x^* + \delta_0) = x^* + \mu \delta_0$$

con  $\mu = f'(x^*)$ . Así, tenemos la perturbación buscada  $\delta_1$  en términos de la perturbación inicial  $\delta_0$ :

$$\delta_1 = \mu \delta_0$$

$\mu$  determina la estabilidad del punto fijo y es llamado *factor de estabilidad*. Ahora, si queremos calcular la perturbación después de  $n$  iteraciones, un análisis análogo al anterior nos lleva a que ésta es:

$$\delta_n \simeq (\mu)^n \delta_0$$

Para  $|\mu| < 1$  (resp.  $|\mu| > 1$ ) decimos que el punto fijo  $x^*$  es *estable* (resp. *inestable*).

Si  $|\mu| = 1$  no se puede asegurar algo al respecto de la estabilidad del punto fijo y para este caso particular, el sistema sufre cambios peculiares de los que se hablará más adelante.

**Ejemplo 1.2.1.** La función logística [2], se define como:

$$f(x) = rx(1 - x)$$

donde  $x \in [0, 1]$  y  $r$  toma valores en  $(0, 4]$ .

Cabe mencionar que si  $r$  tomara el valor 0 la función se reduce a la constante  $f(x) = 0$  y para hacer la función biológicamente admisible se toman valores de  $r < 4$ . Encontrar los puntos fijos de la ecuación logística es una tarea sencilla, sólo debemos resolver la ecuación

$$x = rx(1 - x)$$

Las soluciones a ésta ecuación son:

$$x_1 = 0 \text{ y } x_2 = \frac{r-1}{r}$$

Si  $0 < r < 1$  tenemos que  $x_1 = 0$  es un punto fijo estable, ya que  $f'(0) = r$  y  $x_2$  se vuelve inestable ya que  $f'(x_2) = -r + 2 > 1$ . Por otra parte si  $1 < r < 3$ ,  $x_1 = 0$  se vuelve inestable y  $x_2$  se vuelve estable, esto debido a que  $f'(x_2) = -r + 2 < 1$ . Si  $r > 3$  ambos puntos son inestables. Para los valores de  $r = 1$  y  $r = 3$  el criterio de estabilidad no proporciona información sobre los puntos fijos.

### 1.2.3. Estabilidad de órbitas periódicas

Consideremos el sistema  $x_{t+1} = f(x_t)$  donde  $f : E \rightarrow E$ ,  $E \subset \mathbb{R}$  y  $f$  es de clase  $\mathbb{C}^1$ . Una *órbita periódica de periodo  $n$*  se compone de  $n$  puntos periódicos,  $\mathbb{O}(x_n) = \{x_0, x_1, \dots, x_{n-1}\}$ , tales que  $x_i$  es punto periódico de periodo  $n$ , es decir que  $x_i = f^n(x_i)$  para  $i = 0, 1, \dots, n-1$ . Notemos que si un punto del sistema pertenece a la órbita periódica, éste irá alternando entre los puntos de  $\mathbb{O}(x_n)$ .

Ahora, consideremos la órbita de periodo  $p$ ,  $\mathbb{O}(x_p)$ . Si deseamos analizar la estabilidad de la órbita, como en el caso anterior, tomamos un punto  $x_i \in \mathbb{O}(x_p)$  e introducimos una pequeña perturbación, tan pequeña como deseemos, y tomamos este nuevo valor como estado inicial del sistema. Tenemos entonces que

$$x_{ini} = x_i + \delta_0$$

después de  $p$  iteraciones la perturbación que habrá sufrido el valor  $x_i$  será  $x_i + \delta_p$ , para algún  $\delta_p$ . Para calcular dicho valor tenemos que:

$$x_i + \delta_p = f^p(x_i + \delta_0)$$

y de nuevo por Taylor, podemos aproximar este valor como:

$$f^p(x_i + \delta_0) = f^p(x_i) + (f^p)'(x_i)\delta_0 + \mathbb{O}(\delta_0^2)$$

usando que  $x_i = f^p(x_i)$  y despreciando los órdenes mayores  $\mathbb{O}(\delta_0^2)$  ya que  $\delta_0$  es suficientemente pequeño,

$$f^p(x_i + \delta_0) = x_i + (f^p)'(x_i)\delta_0$$

definiendo  $\lambda_p = (f^p)'(x_i)$  tenemos que  $\delta_p = \lambda_p\delta_0$ . Además

$$\begin{aligned} \lambda_p = (f^p)'(x_i) &= \overbrace{(f \circ \dots \circ f)'(x_i)}^{p \text{ veces}} \\ &= f'(\overbrace{f \circ \dots \circ f}(x_i)) \cdot f'(\overbrace{(f \circ \dots \circ f)}(x_i)) \cdots f'(f(x_i)) \cdot f'(x_i) \\ &= f'(f^{p-1}(x_i)) \cdot f'(f^{p-2}(x_i)) \cdots f'(f(x_i)) \cdot f'(x_i) \\ &= f'(x_i) \cdot f'(f^1(x_i)) \cdot f'(f^2(x_i)) \cdots f'(f^{p-1}(x_i)) \end{aligned} \tag{1.8}$$

notemos que en la última expresión tenemos los puntos de la órbita, ya que

$$\mathbb{O}(x_p) = \{x_i, f(x_i), f^2(x_i), \dots, f^{p-1}(x_i)\}$$

Así podemos reescribir la igualdad 1.8 como:

$$\lambda_p = f'(x_0) \cdot f'(x_1) \cdots f'(x_{p-1})$$

Si aplicamos este valor otras  $p$  iteraciones alrededor de la órbita, se puede calcular con el análisis anterior, que el resultado es:

$$x_i + \delta_{2p} = x_i + \lambda_p \delta_p = x_i + \lambda_p (\lambda_p \delta_0) = x_i + (\lambda_p)^2 \delta_0.$$

En general después de  $np$  iteraciones el valor obtenido es:

$$x_i + \delta_{np}$$

donde

$$\delta_{np} = \lambda_p^n \delta_0$$

Le llamaremos a  $\lambda_p$  *factor de estabilidad* para una órbita periódica. Para  $|\lambda_p| > 1$  la perturbación que sufre el punto inicial cada  $p$  iteraciones aumenta por un factor de  $\lambda_p$ , y la órbita periódica actúa como una órbita inestable.

Cuando  $|\lambda_p| < 1$  cada vez que pasamos alrededor de la órbita periódica las perturbaciones disminuyen, es decir, la órbita es estable y todas las condiciones iniciales en su vecindad serán atraídas a dicha órbita asintóticamente.

Existen conjuntos especiales donde ocurren dinámicas especiales en los sistemas, en la siguiente sección se abordan dos muy importantes.

### 1.3. Conjuntos invariantes y variedades

Pensemos de nuevo en el sistema dinámico (1.4). Podemos definir un *conjunto invariante* bajo la función  $f$  como un subconjunto  $S$  del espacio fase  $X$  tal que para  $x_0 \in S$ ,  $f^n(x_0) \in S$ , para todo  $n$ . Con ésta formulación decimos de hecho que el subconjunto  $S$  se transforma en sí mismo bajo un número arbitrario de iteraciones de  $f$ , es decir,  $f^n(S) \subseteq S$ .

Notemos que  $\mathbb{O}(x_0)$  también es un conjunto invariante, el ejemplo más simple de conjuntos invariantes son los puntos fijos.

**Ejemplo 1.3.1.** *Pensemos un sistema de la forma (1.5), donde  $f$  se define en coordenadas polares de la forma  $f : (\mathbb{R}^+, [0, 2\pi]) \rightarrow (\mathbb{R}, [0, 2\pi])$*

$$f(r, \theta) = (r^2, \theta + q)$$

donde  $q \in [0, 2\pi]$  es un ángulo de rotación. El círculo unitario es invariante bajo  $f$  ya que cualquier punto  $(r, \theta)$  tal que  $r = 1$  y  $\theta \in [0, 2\pi]$  implica que  $f(r, \theta) \in \{1, [0, 2\pi]\}$ , es decir,  $(r, \theta)$  gira a través del ángulo  $\theta$  dado.

### 1.3.1. Variedad estable e inestable

El concepto de variedad involucra una gran cantidad de detalles técnicos por lo que se vuelve complejo dar una definición clara. El lector puede revisar una definición formal completa en [18]. Para fines de este trabajo se aborda el concepto desde uno de los teoremas más importantes en la teoría de ecuaciones diferenciales ordinarias, el teorema de la variedad estable [23].

El teorema afirma que cerca de un punto de equilibrio hiperbólico <sup>2</sup>  $x_0$ , el sistema no lineal

$$\dot{x} = f(x)$$

tiene variedades estable e inestable  $S$  y  $U$  tangentes en  $x_0$  a los subespacios estable e inestable  $E^s$  y  $E^u$  del sistema linealizado

$$\dot{x} = Ax$$

donde  $A = Df(x_0)$ . Además si  $\phi_t$  es el flujo del sistema no lineal  $\dot{x} = f(x)$ , entonces  $S$  y  $U$  son positiva y negativamente invariantes bajo  $\phi_t$ , respectivamente y satisfacen

$$\lim_{t \rightarrow \infty} \phi_t(c) = x_0 \text{ para todo } c \in S$$

y

$$\lim_{t \rightarrow -\infty} \phi_t(c) = x_0 \text{ para todo } c \in U.$$

---

<sup>2</sup>Un punto  $x_0 \in \mathbb{R}^n$  es un punto de equilibrio de la ecuación diferencial  $\dot{x} = f(x)$ , si  $f(x_0) = 0$ . Un punto de equilibrio se dice que es hiperbólico de la ecuación diferencial si ningún valor propio de la matriz jacobiana tiene parte real nula.

Si consideramos al sistema dinámico (1.4), podemos determinar la estabilidad de una variedad con respecto a  $x_0$  bajo  $f$  dependiendo del comportamiento del conjunto de trayectorias de puntos vecinos a él.

**Ejemplo 1.3.2.** Consideremos el siguiente sistema de ecuaciones [28],

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} a & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

multiplicando las matrices obtenemos,

$$\begin{aligned} \dot{x} &= ax \\ \dot{y} &= -y. \end{aligned}$$

Éste es un ejemplo de sistema **desacoplado**, es decir, la variable  $x$  no aparece en la ecuación de  $y$  y viceversa. Como vimos en la sección 1.1, la solución de este sistema de ecuaciones define un sistema dinámico. Al resolver este sistema obtenemos la siguiente solución:

$$\begin{aligned} x(t) &= x_0 e^{at} \\ y(t) &= y_0 e^{-t} \end{aligned}$$

Si  $a > 0$ , la figura 1.1 muestra las trayectorias del sistema en el plano  $XY$ , donde cada curva representa una condición inicial distinta, a esta representación gráfica se le conoce como **retrato fase**.

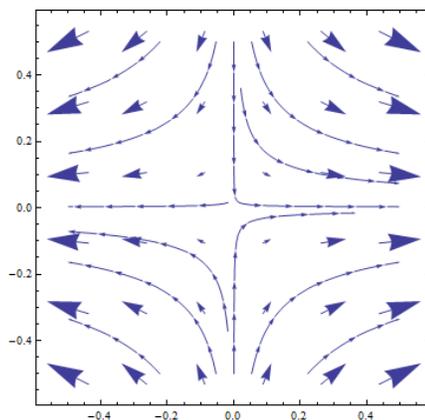


Figura 1.1: Retrato fase  $a > 0$

*Se puede verificar fácilmente que el punto  $x^* = 0$  es un punto fijo inestable. Localmente, cerca del punto de equilibrio las trayectorias se alejan de  $x^*$  y se dirigen al  $\infty$ , pero si la trayectoria comienza en el eje  $Y$ , ésta se va acercando a  $x^*$ . El eje  $Y$  es una variedad estable de  $x^*$ , es decir, es el conjunto de condiciones iniciales tales que  $x(t) \rightarrow x^* = 0$ , cuando  $t \rightarrow \infty$ . Análogamente, la variedad inestable de  $x^*$  es el conjunto de condiciones iniciales tales que  $x(t) \rightarrow x^* = 0$  cuando  $t \rightarrow -\infty$ , en este caso es el eje  $X$ . Un hecho curioso es que si elegimos una trayectoria del retrato fase, comúnmente ésta se aleja de la variedad estable y después se acerca asintóticamente a la variedad inestable. Esto parece contradictorio.*

Si la intersección de una variedad estable y una variedad inestable no es vacía, a los puntos de la intersección se les conoce como *puntos homoclínicos* [17]. En el ejemplo 1.3.2 el punto  $x^* = 0$  es un punto homoclínico.

La existencia de puntos homoclínicos da a lugar a dinámicas caóticas en un sistema, en el siguiente capítulo se aborda el concepto de Caos y las propiedades que se utilizan en el cifrado propuesto en este trabajo.

# Capítulo 2

## Caos

El concepto de *Caos* puede definirse como el comportamiento errático e impredecible de algunos sistemas dinámicos deterministas que se caracterizan por ser sensibles a pequeños cambios en las condiciones iniciales. En este capítulo se estudian brevemente las principales características del caos debido a que los sistemas caóticos han sido utilizados para crear algoritmos de cifrado. Se comenzará por dar una definición matemática de caos.

### 2.1. Caos: Definición matemática

Es importante abordar una definición formal de caos. Una de las más importantes y enunciadas en la literatura es la definición del matemático R. Devaney [10]. Antes de abordar dicha definición vamos a revisar dos conceptos importantes para el estudio del caos, la *transitividad topológica* y la *sensibilidad a condiciones iniciales*.

La idea intuitiva de transitividad topológica es la siguiente: una función se dice que es transitiva si dados dos subconjuntos cualesquiera del espacio donde está definida la función, existe un punto en uno de los subconjuntos cuya órbita en algún momento se encontrará en el otro. Esto quiere decir, que dado un sistema dinámico, la función que lo define es transitiva si existen puntos cuya órbita viaja de una parte arbitraria del espacio en donde está definido el sistema, a otra.

**Definición 2.1.1.** *Sea  $f : X \rightarrow X$  una función continua, donde  $X$  es un*

espacio métrico. Decimos que  $f$  es topológicamente transitiva en  $X$  si para cualesquiera  $A$  y  $B$  subconjuntos abiertos de  $X$ , distintos del vacío, existen  $a \in A$  y  $n \geq 1$  tales que  $f^n(a) \in B$ .

Para entender la sensibilidad a condiciones iniciales de un sistema dinámico, pensemos en pequeños cambios en los estados iniciales de una órbita o trayectoria de un sistema que producen diferencias impredecibles a través del tiempo. Para dar una primera noción de esto, imaginemos el camino que sigue una pelota que en su estado inicial está balanceada en la cima de una colina empinada. Si soltamos la pelota en un instante  $t_0$  desde una posición  $x_0$  la pelota recorre un camino aleatorio que podemos seguir perfectamente en el tiempo. Si posteriormente soltamos la misma pelota desde una posición mínimamente desplazada con respecto a  $x_0$ ,  $x_0 + \delta$ , la pelota generará un camino que bien puede coincidir en algunos puntos con el primer camino recorrido, pero en esencia será totalmente distinto e impredecible. Una definición formal de sensibilidad a condiciones iniciales se da a continuación.

**Definición 2.1.2.** Sea  $f : \mathbb{R} \rightarrow \mathbb{R}$  la función de un sistema dinámico. Un punto  $x_0 \in \mathbb{R}$  es sensible a condiciones iniciales si existe  $d > 0$  tal que cualquier vecindad  $V$  de  $x_0$  contiene al menos un punto  $x$  que satisface  $|f^k(x_0) - f^k(x)| \geq d$  para algún  $k$  entero no negativo. A  $x_0$  se le conoce como **punto sensible**.

Esto significa que para un punto dado  $x_0$  existe al menos un punto arbitrariamente cerca de él tal que su imagen después de  $k$  iteraciones difiere de la imagen de  $x_0$  por una cantidad mayor que la cantidad  $d$ . Dos ejemplos se dan en la siguiente sección.

Ahora podemos presentar la definición de *función caótica* propuesta por R. Devaney. [10].

**Definición 2.1.3.** Sea  $f : X \rightarrow X$  continua, donde  $X$  es un espacio topológico. Decimos que  $f$  es caótica en  $X$  si se cumplen las siguientes tres condiciones:

1. El conjunto de puntos periódicos de  $f$  forma un conjunto denso en  $X$ .<sup>1</sup>

---

<sup>1</sup>Sea  $A \subset B$ .  $A$  es denso en  $B$  si arbitrariamente cerca de cada punto de  $B$  existe un punto de  $A$ . Es decir, si para cada  $x \in B$  y cada  $\epsilon > 0$  la vecindad  $\mathcal{N}_\epsilon(x)$  contiene un punto de  $A$ .

## 2.2. PRINCIPALES CARACTERÍSTICAS DE UN SISTEMA CAÓTICO<sup>15</sup>

2.  $f$  es topológicamente transitiva en  $X$ .
3.  $f$  es sensible a las condiciones iniciales en  $X$ .

**Ejemplo 2.1.4.** La función  $T : [0, 1] \rightarrow [0, 1]$  conocida como “casa de campaña” o “Tent”, se define como:

$$T(x) = \begin{cases} 2x & , \text{ si } 0 \leq x < \frac{1}{2} \\ -2x + 2 & , \text{ si } \frac{1}{2} \leq x \leq 1 \end{cases}$$

La gráfica de la función  $T$  consta de dos líneas que asemejan una casa de campaña (Ver Fig. 2.1), de ahí su nombre. La función  $T$  es caótica en  $[0, 1]$ , [28]. Para probar esto, analizamos los puntos de la definición 2.1.3, pero primero se dan algunas características de un sistema caótico.

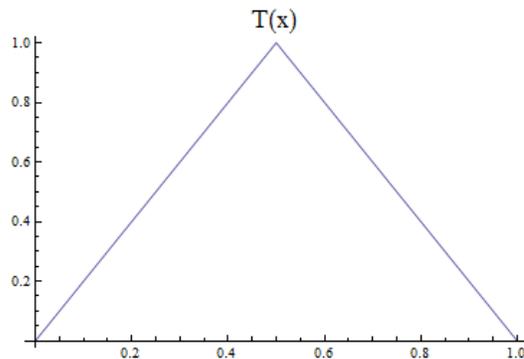


Figura 2.1: Función *Casa de campaña*

## 2.2. Principales Características de un Sistema Caótico

En esta sección se discutirán las principales propiedades que determinan caos en un sistema desde el punto de vista de un sistema dinámico no lineal que exhibe caos determinista cuando los parámetros toman valores especiales. Se mencionan las características de un sistema caótico que son de interés para el presente trabajo [30].

### Sensibilidad a cambios en las condiciones iniciales

Volvamos al ejemplo de la Ecuación Logística (Ec.1.2.1). En la figura 2.2 podemos observar la evolución de la función con dos valores iniciales distintos pero muy cercanos entre ellos,  $x_0 = 0.1$  y  $x_0 = 0.101$ . Podemos ver que desde muy tempranas iteraciones los caminos trazados se vuelven diferentes. Al cabo de 150 iteraciones las rutas son totalmente distintas.

Ahora tenemos la siguiente función:

$$f(x) = kx \pmod{1}, \quad k \in \mathbb{Z} \quad (2.1)$$

definida en el intervalo unitario, donde la notación  $y \pmod{1}$  representa al número  $y - \lfloor y \rfloor$ , y  $\lfloor y \rfloor$  representa la conocida “función piso” de  $y$ . Para  $y > 0$  el valor de  $y \pmod{1}$  es simplemente la parte fraccionaria de  $y$ . Cabe mencionar que la función 2.1 no es continua en los puntos  $x = \frac{1}{3}, \frac{2}{3}$ .

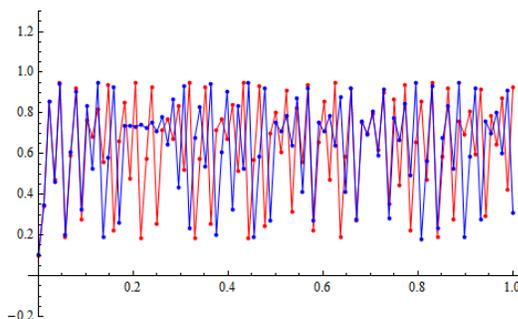


Figura 2.2: Función Logística con  $r = 3.85$ , gráfica azul  $x_0 = 0.1$ , gráfica roja  $x_0 = 0.101$

Introducimos ahora la siguiente definición.

**Definición 2.2.1.** Sea  $f : E \rightarrow E$ ,  $E \subset \mathbb{R}$ , decimos que un punto  $x$  es **periódico de periodo  $p$**  para  $f$  si para algún entero  $p > 0$ ,  $f^{n+p}(x) = f^n(x) \quad \forall n \geq N$ , donde  $N > 1$  y el periodo es el menor  $p$  que cumple dicha propiedad.

Esto nos dice que eventualmente la órbita de  $x$  coincidirá exactamente con una órbita periódica. En el ejemplo 2.1, con  $k = 3$  y  $x = \frac{1}{3}$  tenemos una órbita de periodo 0. Si partimos del punto  $x = 0.25$  obtenemos una órbita

## 2.2. PRINCIPALES CARACTERÍSTICAS DE UN SISTEMA CAÓTICO 17

de periodo dos, pero si perturbamos dicho valor inicial e iniciamos el sistema con  $x = 0.2501$  se genera un camino distinto sin alguna órbita periódica aparente, las primeras 15 iteraciones se muestran en la siguiente tabla.

$x_0$	0.25	0.2501	$x_5$	0.75	0.7743	$x_{10}$	0.25	0.1549
$x_1$	0.75	0.7503	$x_6$	0.25	0.3229	$x_{11}$	0.75	0.4647
$x_2$	0.25	0.2509	$x_7$	0.75	0.9687	$x_{12}$	0.25	0.3941
$x_3$	0.75	0.7527	$x_8$	0.25	0.9061	$x_{13}$	0.75	0.1823
$x_4$	0.25	0.2581	$x_9$	0.75	0.7183	$x_{14}$	0.25	0.5969

**Ejemplo 2.2.2.** La función  $f(x) = 3x \pmod{1}$  también puede escribirse como:

$$f(x) = \begin{cases} 3x, & 0 \leq x < \frac{1}{3} \\ 3(x - \frac{1}{3}), & \frac{1}{3} \leq x < \frac{2}{3} \\ 3(x - \frac{2}{3}), & \frac{2}{3} \leq x < 1 \\ 0, & x = 1 \end{cases}$$

Claramente esta función no es continua en  $x = \frac{1}{3}, \frac{2}{3}, 1$ .

Sea  $x \in [0, \frac{1}{3})$ , suponga que para un  $\varepsilon > 0$ ,  $x + \varepsilon \in [0, \frac{1}{3})$ , entonces  $f(x) = 3x$  y  $f(x + \varepsilon) = 3(x + \varepsilon)$ . Luego

$$|f(x) - f(x + \varepsilon)| = 3\varepsilon$$

Tome  $d > 3\varepsilon$  y  $k = 1$ . Se cumple entonces que  $x$  en el intervalo  $[0, \frac{1}{3})$  para la función dada, es punto sensible. De forma análoga se puede verificar fácilmente que la definición se cumple para  $x \in [\frac{1}{3}, \frac{2}{3})$  y  $x \in [\frac{2}{3}, 1)$ . Así podemos decir que cualquier elemento del dominio de  $f$  es punto sensible.

**Ejemplo 2.2.3.** Recordemos la función “Tent” (Ver Ej.2.1.4).

Analizando la estabilidad de la función “Tent” como se hizo en la Sección 1.2.2, al tomar dos puntos arbitrariamente cercanos con una pequeña perturbación entre ellos,  $\delta_0 > 0$ , dicha perturbación después de  $n$  iteraciones se convierte en  $\delta_n = \mu^n \delta_0$  donde el factor de estabilidad  $\mu$  se obtiene simplemente calculando la derivada de la función en el punto  $x$ . De modo que  $|\mu| = 2$ . Luego, si elegimos dos puntos  $x, y \in A = (0, 1)$  tales que

$y = x + \delta_0$  con  $\delta_0 > 0$ , después de  $n$  iteraciones, la separación  $\delta_0$  se convierte en  $\delta_n = |\mu^n|\delta_0$ . Además  $\delta_n > \delta_0$ , y se cumple que para la  $n$ -ésima iteración,  $|f^n(x) - f^n(y)| = |2^n|\delta_0 > \delta_0$ .

De este modo, decimos que cualquier punto en el intervalo  $(0, 1)$  es sensible a condiciones iniciales en la función “Tent”.

Probemos ahora que la función  $T$  es caótica en  $[0, 1]$ .

1. *El conjunto de puntos periódicos es denso en  $[0, 1]$ .* Es fácil observar que  $T^n$  mapea cada intervalo  $[\frac{k}{2^n}, \frac{k+1}{2^n}]$  en  $[0, 1]$  para  $k = 0, 1, \dots, 2^n - 1$ . Entonces,  $T^n$  intersecta la línea  $y = x$  una ocasión en cada intervalo, como resultado cada intervalo contiene un punto periódico de periodo  $n$ , de  $T$ . Así, los puntos periódicos de  $T$  son densos en  $[0, 1]$ .
2. *Transitividad Topológica.* Sean  $U_1$  y  $U_2$  subconjuntos abiertos de  $[0, 1]$ . Para algún  $k$ ,  $U_1$  contiene un intervalo de la forma  $[\frac{k}{2^n}, \frac{k+1}{2^n}]$ . Pero como dijimos en 1), para  $n$  suficientemente grande,  $T^n$  transforma a  $U_1$  en  $[0, 1]$  que contiene a  $U_2$ .
3. *Sensibilidad a condiciones iniciales.* Sea  $x_0 \in [0, 1]$  y  $\delta = 1/2$ . Como vimos en 2), algún intervalo abierto  $U$  de  $x_0$ , es transformado por  $T^n$  en  $[0, 1]$  para algún  $n$  suficientemente grande. Entonces existe  $y_0 \in U$  tal que  $|f^n(x_0) - f^n(y_0)| \geq \frac{1}{2} = \delta$ .

La noción de que bajo la acción de una función un punto es sensible a condiciones iniciales es muy importante para el comportamiento caótico de un sistema dinámico, ya que aunque tengamos la certeza de que un punto  $x_i$  inicialmente pertenece a un pequeño subintervalo del espacio donde está definida  $f$ , después de cierto número de iteraciones el estado  $x_i$  puede estar en cualquier parte del espacio.

En el estudio del caos es primordial poder verificar la sensibilidad a condiciones iniciales de una función, no sólo para darnos certeza de que tenemos un sistema con comportamiento caótico, sino para saber qué tanto lo es con respecto a otro, una herramienta para cuantificar esto es el *exponente de Lyapunov* del que se hablará en la sección 2.3.

## 2.2. PRINCIPALES CARACTERÍSTICAS DE UN SISTEMA CAÓTICO 19

Volvamos a la ecuación logística del ejemplo 1.2.1. Sabemos que tiene dos puntos fijos que son  $x_1 = 0$ ,  $x_2 = 1 - \frac{1}{r}$ . Los correspondientes factores de estabilidad de éstos puntos fijos son,  $\mu_1 = r$ ,  $\mu_2 = 2 - r$  ya que  $f'(x_1) = r$  y  $f'(x_2) = 2 - r$ .

Usando lo anterior, podemos ver que  $x_1$  es estable siempre y cuando  $r > 1$ . Por otra parte, si  $r$  toma valores  $1 < r < 3$ ,  $x_1$  pierde la estabilidad y  $x_2$  se vuelve estable ya que  $|2 - r| < 1$ . Así que para este rango de  $r$  podemos decir que  $x_2$  es un punto fijo atractor. Además podemos mostrar fácilmente que para valores de  $r \in (1, 3)$  no existen órbitas de periodo  $p \geq 2$ , ya que cualquier condición inicial que se elija  $0 \leq x_0 \leq 1$  converge al atractor  $x_2$ . (ver figura 2.3).

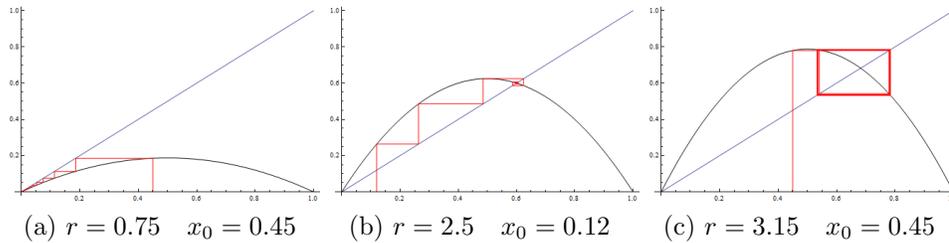


Figura 2.3: Cambios de estabilidad en función logística

También podemos analizar al punto fijo  $x_1 = 0$ . Para  $r \in (0, 1)$  es estable, ya que  $\mu_1 = r$  y  $|\mu_1| < 1$ , y de nuevo, cualquier condición inicial arbitraria converge a él. Sin embargo para estos valores de  $r$ , el punto fijo  $x_2 = 1 - \frac{1}{r}$  es inestable, ya que  $\mu_2 \in (1, 2)$  por lo tanto  $\mu_2 > 1$ .

Una de las características más interesantes del sistema dinámico logístico es que conforme incrementamos el parámetro  $r$ , el comportamiento del sistema sufre cambios peculiares. La estructura cualitativa del sistema puede cambiar conforme los parámetros varían, puntos fijos pueden aparecer o desaparecer y la estabilidad de los mismos puede variar. Estos cambios en la dinámica del sistema se denominan *bifurcaciones*. Una *bifurcación* surge cuando una variación de los parámetros de un sistema causa un cambio cualitativo o topológico en su comportamiento.

En el ejemplo de la ecuación logística hemos encontrado que para valores del parámetro  $0 < r < 2$  existe un punto fijo atractor. Para valores mayores aparecen cambios periódicos o caóticos y para valores  $r > 4$  la ecuación logística tiene infinidad de puntos periódicos pero no atractores, así, si existen puntos fijos o periódicos al variar los valores del parámetro llamamos a estos valores que ocasionan los cambios cualitativos, *valores de bifurcación del parámetro*.

Existen varios tipos de bifurcación dependiendo del cambio específico de la estabilidad de los puntos fijos del sistema o nacimiento de los mismos, un tipo de bifurcación especial es la *bifurcación de duplicado de periodo*, esto ocurre cuando un punto fijo pierde estabilidad y simultáneamente aparece una nueva órbita de periodo duplicado.

El periodo de las órbitas que van apareciendo en la ecuación logística, se dividen de una forma que pareciera tener un orden. La órbita de periodo 1 o punto fijo  $x_1 = 0$ , bifurca en cierto valor de  $r$  en una órbita de periodo 2, después en otro valor de  $r$  aparecen órbitas de periodo 4, y así sucesivamente.

La representación gráfica en donde podemos observar los cambios de carácter del sistema son llamados *diagramas de bifurcación*.

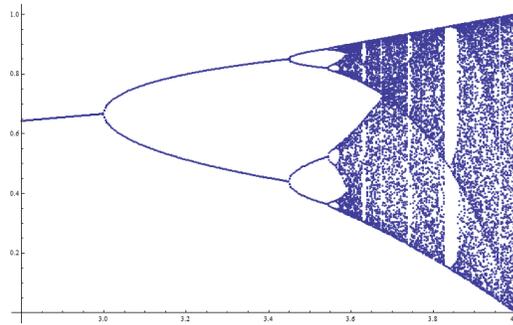


Figura 2.4: Diagrama de bifurcación: Función Logística

La Figura 2.4 muestra que conforme incrementamos el parámetro  $r$  se van generando divisiones en el periodo de las órbitas y que dichas divisiones continúan indefinidamente, *i.e.* órbitas de periodo  $2^n$  son creadas cuando  $n \rightarrow \infty$ ,  $n \in \mathbb{N}$ . A este tipo de bifurcación se le conoce como *cascada de*

*duplicación de periodo* y estas cascadas son de cierta manera las culpables de la emergencia del caos en la función logística.

Existen otros tipos de bifurcación que se pueden clasificar como locales, aquellas que se pueden analizar mediante cambios en la estabilidad local, y globales, que son más complejas e implican fenómenos globales, su estudio no es fácil de determinar. Entre las bifurcaciones locales están: la bifurcación transcítica y la bifurcación de Hopf y un ejemplo de bifurcación global es la bifurcación homoclínica. Los tipos de bifurcaciones no se estudian en este trabajo pero una descripción más detallada se puede consultar en ([28], pág. 44).

## 2.3. Exponente de Lyapunov

El comportamiento errático de un sistema dinámico puede “perseguirse” y analizarse a través de las órbitas del sistema. Pensemos en un sistema discreto, si deseamos medir la caoticidad del sistema nos será útil medir la tasa promedio de separación por iteración de puntos cercanos a lo largo de una órbita. Para este trabajo, nos interesa no sólo que el caos aparezca en un sistema, sino también que se conserve. Una manera de cuantificar lo caótico de un sistema dinámico está dada por el *Número de Lyapunov* [16]. El logaritmo natural del número de Lyapunov es conocido como *Exponente de Lyapunov*.

Pueden caracterizarse los sistemas dinámicos caóticos como aquellos con exponente de Lyapunov positivo. Esto nos motiva a analizar dicho concepto. En esta sección se da la definición de número y exponente de Lyapunov en una dimensión y se extiende a funciones en  $\mathbb{R}^m$  para  $m \geq 1$  con el fin de hacer un análisis del caos a partir de ellas.

En la Sección 1.2.2 vimos que la estabilidad de un punto fijo en el sistema está fuertemente influenciada por la derivada de la función en dicho punto. Por ejemplo, sea  $f : \mathbb{R} \rightarrow \mathbb{R}$ , y  $x^*$  punto fijo de  $f$ . Sea  $x \in \mathbb{R}$ , si  $f'(x^*) = a > 1$  en cada iteración los puntos de la órbita de  $x$  se alejan de  $x^*$  a una tasa multiplicativa de aproximadamente  $a$  hasta que quedan significativamente lejos del punto fijo. Esto es, la distancia entre  $f^n(x)$  y  $f^n(x^*)$  se magnifica por un valor aproximado de  $a > 1$  cada vez que  $n$  aumenta.

Ahora, sea  $x_1$  un punto periódico de “periodo  $k$ ”, según la ecuación (1.8) el factor de estabilidad está dado por la derivada en la iteración  $k$ -ésima de la función en dicho punto, ahora supongamos que dicho valor es  $A > 1$ . Al igual que el caso anterior, esto implica que la órbita de cada punto cercano a  $x_1$  se irá separando de  $x_1$  después de  $k$  iteraciones a una tasa de aproximadamente  $A$ . Además, esta separación se acumula y la tasa multiplicativa de separación será de  $A^k$  por iteración. Introducimos la siguiente definición [2].

**Definición 2.3.1.** Sea  $f : \mathbb{R} \rightarrow \mathbb{R}$  una función  $\mathbb{C}^1$ , el **número de Lyapunov**  $L(x_1)$  de la órbita  $\{x_1, x_2, \dots\}$  es definido como

$$L(x_1) = \lim_{n \rightarrow \infty} (|f'(x_1)| \cdots |f'(x_n)|)^{\frac{1}{n}}$$

si este límite existe. El **exponente de Lyapunov**  $h(x_1)$  se define como

$$h(x_1) = \lim_{n \rightarrow \infty} \left( \frac{1}{n} \right) [ \ln |f'(x_1)| + \dots + \ln |f'(x_n)| ]$$

Notemos que  $h(x_1)$  existe si y sólo si  $L(x_1)$  existe y es diferente de 0 y que  $h(x_1) = \ln L(x_1)$ . Además,  $h(x_1)$  y  $L(x_1)$  pueden no estar definidos para algunas órbitas. En particular, si una órbita contiene un punto  $x_i$  tal que  $f'(x_i) = 0$  el exponente de Lyapunov no está definido para dicho punto.

Supongamos que  $f$  es una función definida en  $\mathbb{R}$  y  $x^*$  punto fijo de  $f$ , el número de Lyapunov es simplemente  $L(x^*) = |f'(x^*)|$  o equivalentemente  $h(x^*) = \ln |f'(x^*)|$ . Si  $x_k$  es un punto periódico de periodo  $k$ , el exponente de Lyapunov se calcula como:

$$h(x_k) = \frac{\ln |f'(x_1)| + \dots + \ln |f'(x_k)|}{k}$$

Esto se debe a que, sin pérdida de generalidad, si  $x_1$  es punto periódico de periodo  $k$ , la órbita de  $x_1$  se puede representar como:

$$\mathbb{O}(x_1) = \{x_1, \dots, x_k, x_1, \dots, x_k, \dots\}$$

De este modo al calcular el número de Lyapunov tenemos

$$L(x_1) = \lim_{n \rightarrow \infty} ((|f'(x_1)| \cdots |f'(x_k)|)^{\frac{n}{k}})^{\frac{1}{n}}$$

y entonces

$$L(x_1) = \lim_{n \rightarrow \infty} (|f'(x_1)| \cdots |f'(x_k)|)^{\frac{1}{k}}$$

y de esto se sigue que

$$h(x_k) = \frac{\ln|f'(x_1)| + \dots + \ln|f'(x_k)|}{k}$$

**Ejemplo 2.3.2.** *Pensemos en la función  $f : \mathbb{R} \rightarrow [0, 1]$   $f(x) = 2x \pmod{1}$ . Claramente  $f$  no es continua en el punto  $x = \frac{1}{2}$  por lo tanto en ese punto no es diferenciable. Así, para órbitas  $\{x_1, x_2, \dots\}$  que no contengan  $x = \frac{1}{2}$  el exponente de Lyapunov se calcula como sigue:*

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n \ln|f'(x_i)| = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n \ln 2 = \ln 2$$

Notemos que en este ejemplo las derivadas en los puntos de la órbita son constantes, es decir,  $f'(x_i) = 2$  para todo  $x_i \in [0, 1] \setminus \{\frac{1}{2}\}$ . Entonces, cada órbita que no contenga  $x = \frac{1}{2}$  tiene exponente de Lyapunov igual a  $\ln 2$

### 2.3.1. Dimensiones superiores

Extendamos ahora los conceptos de número y exponente de Lyapunov a funciones en  $\mathbb{R}^m$ ,  $m \geq 1$ . La idea es que en dimensiones más altas, el comportamiento de la dinámica local puede cambiar de acuerdo a la dirección que se tome, es decir, dos puntos cercanos pueden permanecer “cerca”, en un sentido euclidiano a través de una dirección y separarse entre ellos a través de otra.

En una función en  $\mathbb{R}^m$  cada órbita posee  $m$  números de Lyapunov. Como en el caso de una dimensión, los números de Lyapunov miden la tasa de separación de puntos cercanos en dicha órbita a través de las  $m$  direcciones ortogonales de  $\mathbb{R}^m$ . Dichas direcciones se ordenan dependiendo de la dinámica del sistema, así tenemos que, la primera de éstas direcciones será por la cual la separación sea mayor, es decir, la dirección por la cual existe menor expansión. Para elegir la segunda dirección se repite el procedimiento pero esta vez entre todas las direcciones perpendiculares a la primera. Para la tercera se elige de la misma manera entre las direcciones perpendiculares a las primeras dos y así sucesivamente. Los factores de “estiramiento” o de “contracción” de éstas direcciones serán los números de Lyapunov de la órbita.

Para dar una definición formal, consideremos la esfera unitaria  $S$  en  $\mathbb{R}^n$  centrada en el primer punto  $v_0$  de la órbita bajo la función  $f$ . Como lo que nos interesa es el comportamiento infinitesimal cercano a  $v_0$  fijamos la atención en la matriz jacobiana de la función,  $J = Df(v_0)$ , ([2], pág. 69). Luego, sea  $J_n = Df^n(v_0)$  la matriz jacobiana de la  $n$ -ésima iteración de  $f$ . Si  $f$  está definida en  $\mathbb{R}^2$ , tenemos simplemente el disco unitario  $D$ .

Se puede mostrar que para cada matriz  $M$  la imagen  $M(D)$  es necesariamente un elipsoide (Ver [2]), cuyo eje longitudinal más largo está sobre la dirección de expansión de  $f$ , y el eje corto sobre la dirección de contracción. Si en lugar de  $D$  tenemos la esfera unitaria  $S$ ,  $J_n S$  será un elipsoide con  $m$  ejes ortogonales y las tasas de expansión promedio de los  $m$  ejes ortogonales son los números de Lyapunov.

**Definición 2.3.3.** *Suponga que  $f$  es una función suave tal que  $f : \mathbb{R}^m \rightarrow E \subseteq \mathbb{R}^m$  y  $N$  la esfera unitaria en  $\mathbb{R}^m$ . Sea  $J_n = Df^n(v_0)$  y que para  $k = 1, \dots, m$  denotamos como  $r_k^n$  a la longitud del  $k$ -ésimo eje ortogonal más grande del elipsoide  $J_n N$  para una órbita con punto inicial  $v_0$ . Entonces  $r_k^n$  mide la expansión o contracción de la órbita cerca de  $v_0$  durante las primeras  $n$  iteraciones. **El  $k$ -ésimo número de Lyapunov** de  $v_0$  se define como*

$$L_k = \lim_{n \rightarrow \infty} (r_k^n)^{1/n}$$

si este límite existe. **El  $k$ -ésimo exponente de Lyapunov** de  $v_0$  es  $h_k = \ln(L_k)$ .

Cabe mencionar que se cumple la propiedad de que  $L_1 \geq L_2 \geq \dots \geq L_m$  y por ende  $h_1 \geq h_2 \geq \dots \geq h_m$ . Para el cálculo de los ejes ortogonales del elipsoide  $M(S)$  donde  $S$  es la esfera unitaria en  $\mathbb{R}^m$  y  $M$  es una matriz de tamaño  $m \times m$ , necesitamos del siguiente teorema.

**Teorema 2.3.4.** *Sea  $N$  la esfera unitaria en  $\mathbb{R}^m$ , y sea  $A$  una matriz de tamaño  $m \times m$  con entradas reales. Sean  $s_1^2, \dots, s_m^2$  y  $u_1, \dots, u_m$  los eigenvalores y eigenvectores unitarios, respectivamente, de la matriz  $AA^T$ . Entonces se cumplen:*

1.  $u_1, \dots, u_m$  son vectores unitarios mutuamente ortogonales.

## 2.4. ÓRBITAS PERIÓDICAS Y ASINTÓTICAMENTE PERIÓDICAS 25

2. Los ejes ortogonales del elipsoide  $A(N)$  son  $s_i u_i$  para  $1 \leq i \leq m$ .

Para consultar el teorema 2.3.4 y su demostración (Ver [2], p.87).

De acuerdo a 2. del teorema 2.3.4, la longitud de los ejes ortogonales del elipsoide son las raíces cuadradas de los  $m$  eigenvalores de la matriz  $AA^T$  y las direcciones de los ejes están dadas por los  $m$  eigenvectores ortonormales correspondientes. De este modo podemos extender la definición de órbita caótica a dimensiones más altas [2].

## 2.4. Órbitas periódicas y asintóticamente periódicas

Como se hizo mención en la sección anterior, los puntos de una órbita pueden alejarse entre sí a cada iteración y podemos cuantificar este comportamiento. Pero no es la única forma en la que el sistema dinámico puede ser influenciado por una órbita periódica a través del tiempo, si se cumplen ciertas condiciones, la órbita de un punto puede converger a una órbita periódica. En esta sección se darán dichas condiciones y un teorema con un resultado importante para el estudio de una órbita caótica.

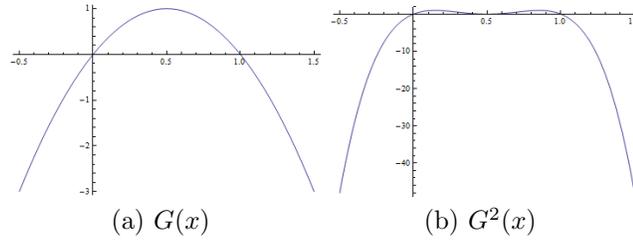
Demos la siguiente definición:

**Definición 2.4.1.** Sea  $f : \mathbb{R}^n \rightarrow E \subseteq \mathbb{R}^m$ , una función suave. Una órbita  $\{x_1, x_2, \dots, x_n, \dots\}$  es llamada **asintóticamente periódica** si converge a una órbita periódica cuando  $n \rightarrow \infty$ , esto es, existe una órbita periódica  $\{y_1, y_2, \dots, y_k\}$  tal que

$$\lim_{n \rightarrow \infty} |x_n - y_n| = 0$$

**Ejemplo 2.4.2.** La órbita de  $x = \frac{1}{2}$  de la función  $G : \mathbb{R} \rightarrow (-\infty, 1]$  definida por  $G(x) = 4x(1-x)$  es asintóticamente periódica pues luego de dos iteraciones coincide con el punto fijo  $x^* = 0$ .

La órbita de  $x = \frac{1}{2}$  es  $\{\frac{1}{2}, 1, 0, 0, \dots\}$ . Las gráficas de  $G(x)$  y  $G^2(x)$  se muestran en la figura 2.5.

Figura 2.5: Gráficas de  $G$  y  $G^2$ 

A continuación se da un teorema importante acerca de la periodicidad asintótica, pero antes se enuncia el siguiente teorema que será útil.

**Teorema 2.4.3.** *Sea  $S_n$  una sucesión infinita que converge a  $S$ , i.e.  $\lim_{n \rightarrow \infty} S_n = S$ , entonces*

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n S_i = S$$

*es decir, si una sucesión infinita de números converge, el promedio de la sucesión converge al mismo límite.*

La demostración del teorema 2.4.3 se puede consultar en [25]. Ahora podemos enunciar el siguiente teorema.

**Teorema 2.4.4.** *Sea  $f : \mathbb{R} \rightarrow E \subseteq \mathbb{R}$  una función suave. Si la órbita  $\{x_1, x_2, \dots\}$  de  $f$  satisface  $f'(x_i) \neq 0 \quad \forall i$  y es asintóticamente periódica a la órbita periódica  $\{y_1, y_2, \dots\}$ , las dos órbitas tienen exponentes de Lyapunov idénticos, asumiendo que ambos existen [2].*

*Demostración.* Supongamos que  $y_1$  es un punto fijo, entonces  $\lim_{n \rightarrow \infty} x_n = y_1$ , y por continuidad de la derivada se tiene que

$$\lim_{n \rightarrow \infty} f'(x_n) = f'(\lim_{n \rightarrow \infty} x_n) = f'(y_1)$$

Además  $\ln|x|$  es una función continua para  $x > 0$  entonces

$$\lim_{n \rightarrow \infty} \ln|f'(x_n)| = \ln \left| \lim_{n \rightarrow \infty} f'(x_n) \right| = \ln|f'(y_1)|$$

Esta ecuación nos plantea el límite de una sucesión infinita, así que usaremos el resultado del teorema 2.4.4 para obtener la siguiente ecuación.

$$\lim_{n \rightarrow \infty} \ln|f'(x_n)| = \ln|f'(y_1)|$$

y podemos calcular los exponentes de Lyapunov de la siguiente manera:

$$h(x_i) = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n \ln|f'(y_1)| = \ln|f'(y_1)| = h(y_1).$$

Ahora pensemos que  $y_1$  no es necesariamente un punto fijo. Entonces para algún  $k > 0$   $y_1$  es punto fijo para  $f^k$  y la órbita de  $x_1$  es asintóticamente periódica a la de  $y_1$ , entonces el exponente de Lyapunov de  $x_1$  bajo  $f^k$  es  $\ln|(f^k)'(y_1)|$  y esto implica que el exponente de Lyapunov de  $x_1$  bajo  $f$  es

$$h(x_1) = \frac{1}{k} \ln|(f^k)'(y_1)| = h(y_1).$$

■

## 2.5. Órbitas caóticas

El interés de la sección anterior es estudiar las condiciones en las que un sistema dado tiene o puede llegar a tener órbitas periódicas, y esto es importante porque nuestro interés está puesto en un comportamiento contrario, el de las órbitas caóticas. En esta sección se dará la definición de una órbita caótica y se trabajará con algunos ejemplos.

**Definición 2.5.1.** Sea  $f$  una función que  $f : \mathbb{R}^m \rightarrow E \subseteq \mathbb{R}^m$ ,  $m \geq 1$ , y sea  $\{v_1, v_2, \dots\}$  una órbita acotada de  $f$ . La órbita es caótica si

1. No es asintóticamente periódica.
2. Ninguno de los números de Lyapunov es exactamente uno.
3.  $L_1(v_0) > 1$ .

Equivalente a 3. de la definición 2.5.1, se puede pedir que  $h_1(v_0) > 0$ . El concepto de exponente y número de Lyapunov están fuertemente ligados a garantizar un comportamiento caótico en su dinámica.

Veamos la función  $f(x) = ax \pmod{1}$ ,  $a \in \mathbb{R} \setminus \{0\}$ ,  $x \in \mathbb{R}$ . Claramente la función no es continua y por lo tanto tampoco diferenciable en  $x = \frac{1}{a}$ . Sin

embargo nos podemos restringir a órbitas que no contengan el punto  $x = \frac{1}{a}$  y calcular el exponente de Lyapunov. Éste es

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n \ln |f'(x_i)| = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n \ln(a) = \ln(a)$$

por lo tanto, cada órbita que evita el punto  $x = \frac{1}{a}$  y no es asintóticamente periódica es caótica.

El comportamiento de  $f$  sobre el intervalo  $[0, 1]$  puede ilustrarse con el siguiente ejemplo. Sea  $a = 2$ . Tomaremos el punto inicial  $x = \frac{1}{5}$  y la órbita de periodo 4, se usa la representación binaria porque es visible que los puntos periódicos en  $[0, 1]$ , estos son los que tengan una expresión binaria repetitiva, como  $x_0 = \frac{1}{5}$ .

$x$	Expresión binaria
$f^0(\frac{1}{5}) = 0.2$	0.0011 $\overline{0011}$
$f^1(\frac{1}{5}) = 0.4$	0.011 $\overline{0011}$
$f^2(\frac{1}{5}) = 0.8$	0.11 $\overline{0011}$
$f^3(\frac{1}{5}) = 0.6$	0.1 $\overline{0011}$
$f^4(\frac{1}{5}) = 0.2$	0. $\overline{0011}$

Tabla 2.1: Representación binaria de las primeras iteraciones de  $f$ .

Como es el caso de  $\frac{1}{5}$ , los puntos periódicos o eventualmente periódicos bajo  $f$  en  $[0, 1]$  son los puntos con una representación binaria que se repite a partir de algún punto. Podemos concluir que cualquier punto en el intervalo cuya expresión binaria no es eventualmente periódica representa una órbita caótica. Estos puntos son no racionales.

**Teorema 2.5.2.** *La función Tent,  $T : [0, 1] \rightarrow [0, 1]$  tiene un número infinito de órbitas periódicas.*

*Demostración.* Verificamos los dos puntos de la definición 2.5.1. Para 2) tenemos que el exponente de Lyapunov de cada órbita para la cual está definido es  $\ln 2$ , lo que implica que el caos se reduzca a la comprobación de 1), es decir que no existen órbitas asintóticamente periódicas.

Ahora veamos,  $|T'(x)| = 2$  para  $x \neq \frac{1}{2}$ , entonces cualquier órbita que evite dicho punto y no sea asintóticamente periódica es una órbita caótica. Más aún, como la derivada de  $T^k$  en una órbita de periodo  $k$  es  $2^k$ , todas las órbitas periódicas son órbitas repulsoras y no atractoras.

■

Consideremos la siguiente función en el cuadrado unitario en  $\mathbb{R}^2$

**Ejemplo 2.5.3.** La función **Gato de Arnold**,  $f : \mathbb{R}^2 \rightarrow (0, 1) \times (0, 1)$ , se define como

$$f(v) = \mathbf{A}(v)(\text{mod } 1)$$

donde  $\mathbf{A} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$  y  $v = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ , nombrada así por Vladimir Arnold quien demostrara sus efectos en 1960 usando la imagen de un gato, de donde adquirió su nombre (Ver [2], pág. 95).

En este ejemplo, la matriz  $\mathbf{J} = \mathbf{D}f(v)$  es constante y es igual a la matriz  $\mathbf{A}$ . De acuerdo al teorema 2.3.4 los ejes de la elipse formada por  $f^n$  serán las raíces cuadradas de los eigenvalores de  $A^n(A^n)^T$ . Usando el hecho de que  $A$  es simétrica,  $A = A^T$ , basta con calcular los eigenvalores de  $A^n$ , ya que también es simétrica. Es fácil ver que para obtener dichos valores es suficiente con calcular los eigenvalores de  $A$ , éstos son  $e_1 = \frac{3+\sqrt{5}}{2}$  y  $e_2 = \frac{3-\sqrt{5}}{2}$  y claramente  $e_1 > e_2$ .

Así, para  $n$  iteraciones de la función Gato de Arnold, tenemos que un disco de radio  $r$  se transforma en un elipse con ejes  $re_1^n$  y  $re_2^n$ . De acuerdo a la definición, los números de Lyapunov para cualquier órbita de nuestra función son  $L_1 = e_1$  y  $L_2 = e_2$  y los exponentes de Lyapunov son  $h_1 = \ln e_1 \approx 0.962$  y  $h_2 = \ln e_2 \approx -0.962$ .

Cabe mencionar que en este ejemplo la matriz jacobiana  $\mathbf{D}f$  no depende del punto  $v$ , pero esto es inusual. Por lo general el cálculo de los eigenvalores de  $\mathbf{D}f$  no nos da la información necesaria acerca de los números de Lyapunov. Nótese además, que en la función Gato de Arnold ningún exponente de Lyapunov es cero y que cada órbita tiene un exponente de Lyapunov positivo. Podemos concluir que cualquier órbita que no sea asintóticamente periódica

en la función Gato, será caótica.

Uno de los sistemas caóticos con más fama en la literatura es el Sistema Caótico de Lorenz. Por la gran cantidad de trabajos acerca de este sistema se elige como alternativa para cifrar imágenes y datos en este trabajo. En el siguiente capítulo se abordan las características caóticas del sistema de Lorenz así como un concepto que nos provee de una herramienta para medir la caoticidad en un sistema, los exponentes de Lyapunov.

# Capítulo 3

## El Sistema Caótico de Lorenz

En la sección 1.1 vimos que la ecuación (1.2).

$$\dot{x} = Ax$$

define un sistema dinámico que describe el movimiento de los puntos en el espacio fase a través de las soluciones definidas para el sistema de ecuaciones diferenciales [23].

En este capítulo mostraremos como se define un sistema dinámico  $\phi_t$  a partir del sistema

$$\dot{x} = f(x) \tag{3.1}$$

**Definición 3.0.4.** *Sea  $E$  un subconjunto abierto de  $\mathbb{R}^n$  y sea  $f \in C^1$ . Para  $x_0 \in E$ , sea  $\phi_t(x_0)$  con  $t \in \mathbb{R}$ , solución del problema de valor inicial*

$$\begin{aligned} \dot{x} &= f(x) \\ x(0) &= x_0 \end{aligned} \tag{3.2}$$

*el conjunto de funciones  $\phi_t(x_0)$  es llamado el sistema dinámico asociado a la ecuación diferencial 3.1.*

Consideremos el sistema

$$\dot{x} = Ax$$

donde  $x \in \mathbb{R}^n$ ,  $A$  es una matriz  $n \times n$  y

$$\dot{x} = \frac{dx}{dt} = \begin{pmatrix} \frac{dx_1}{dt} \\ \vdots \\ \frac{dx_n}{dt} \end{pmatrix}$$

los sistemas de esta forma son llamados *sistemas lineales*.

Por ejemplo, si

$$A = \begin{pmatrix} a & 0 \\ 0 & -2 \end{pmatrix}$$

las ecuaciones del sistema son:

$$\begin{aligned} \dot{x}_1 &= ax_1 \\ \dot{x}_2 &= -x_2 \end{aligned}$$

Notemos que en el sistema las  $x_i$  aparecen del lado derecho solamente a la primera potencia, de lo contrario sería un *sistema no lineal*. Los típicos términos no lineales son potencias mayores a uno o funciones tales como:  $x_1x_2$ ,  $x_1^3$  o  $\cos(x_2)$ .

**Ejemplo 3.0.5.** *El balanceo de un péndulo se rige por la ecuación*

$$\ddot{x} + \frac{g}{L} \text{sen}(x) = 0$$

donde  $x$  es el ángulo del péndulo desde la vertical,  $g$  es la aceleración de la gravedad y  $L$  es la longitud del péndulo, (Ver [28] p.6).

El sistema equivalente no lineal del péndulo es:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{g}{L} \text{sen}(x_1) \end{aligned}$$

Hasta el momento, hemos discutido las propiedades y dinámica de sistemas simples y el caos no ha sido complejo. En este capítulo nos enfocaremos en un sistema no lineal de dimensiones superiores que es sin duda uno de las más importantes sistemas caóticos, el sistema de Lorenz.

El sistema de Lorenz surge como un modelo meteorológico y fue formulado por primera vez en 1963 por Edward N. Lorenz en su obra "*Deterministic Nonperiodic Flow*" [19]. A finales de 1950 se encontraba trabajando con soluciones aproximadas de un sistema de 12 ecuaciones diferenciales que modelan una atmósfera en miniatura, contaba con una computadora del tamaño de un refrigerador con una memoria interna de 16KB que en su momento era

un poder computacional asombroso. Mientras la comunidad científica estaba desarrollando sofisticados métodos lineales para pronosticar el tiempo, Lorenz tuvo una idea innovadora, predijo que existía un factor desconocido que limitaba el éxito de los modelos predictivos del tiempo. Con ayuda de su computadora exploró espacios de parámetros que produjeron trayectorias que parecían aperiódicas y que confundían las técnicas de predicción lineal.

Lorenz modificó el programa para imprimir sólo tres dígitos significativos de las trayectorias aproximadas de la solución y después de repetir los cálculos muchas veces descubrió que una trayectoria que había reanudado con condiciones iniciales que no habían cambiado significativamente se había ido muy lejos de la original. Lorenz concluyó que estaba viendo una sensibilidad a condiciones iniciales. Lorenz intentó reducir la complejidad del sistema de doce ecuaciones a sólo tres, sin embargo no tuvo éxito en la reducción de este modelo en particular hasta veinte años más tarde.

En 1962 Lorenz visita a Barry Saltzman y con su ayuda encuentra un modelo de siete ecuaciones diferenciales que más tarde derivaría en un conjunto de ecuaciones diferenciales ordinarias no lineales expandiendo las funciones de solución en una serie de Fourier fijando todos los coeficientes de Fourier a cero excepto tres de ellos obteniendo un sistema de tres ecuaciones diferenciales ordinarias:

$$\begin{aligned}\dot{x} &= -\sigma x + \sigma y \\ \dot{y} &= -xz + rx - y \\ \dot{z} &= xy - bz\end{aligned}\tag{3.3}$$

Este es un modelo altamente idealizado de un fluido donde el número de Prandtl  $\sigma$ , el número de Rayleigh  $r$  y  $b$  son parámetros del sistema. Cabe mencionar que las variables  $x$ ,  $y$  y  $z$  no son espaciales y representan características de la evolución del fluido que no se tocarán en este trabajo de tesis. Si se desea conocer un análisis más completo sobre el sistema de Lorenz ver ([2], Págs. 362-365).

Lo que compete a este trabajo es analizar las condiciones para las cuales el sistema de Lorenz se vuelve caótico y qué parámetros podemos usar para contar con trayectorias útiles para el cifrado propuesto.

Una característica importante del modelo de Lorenz, es que antes de su fama, los únicos tipos de atractores estables conocidos en ecuaciones diferenciales fueron los equilibrios y las órbitas cerradas. El modelo de Lorenz vino a introducir un nuevo tipo de atractor que es conocido como *atractor extraño* del que se hablará en la siguiente sección. A continuación se mencionan algunas de sus propiedades importantes.

## No linealidad

El sistema de Lorenz es no lineal debido a los dos términos  $xy$  y  $xz$ .

## Simetría

La simetría del sistema de Lorenz radica en que si reemplazamos las variables  $(x, y)$  por  $(-x, y)$  en 5.2 obtenemos exactamente el mismo sistema. Es decir, si  $(x(t), y(t), z(t))$  es solución de 5.2, también lo es  $(-x(t), -y(t), z(t))$ .

## Estabilidad inestable

Lorenz encontró numéricamente que el sistema se vuelve caótico con los valores  $\sigma = 10$ ,  $b = 8/3$  y cuando el número de Rayleigh excede el valor  $r \approx 24.74$ , encontró que las soluciones se vuelven aperiódicas y sensibles a cambios a condiciones iniciales. También halló comportamientos especiales si se fijan los parámetros  $\sigma = 10$ ,  $b = 8/3$  e incrementamos el valor de  $r > 0$ . El punto de equilibrio  $(0, 0, 0)$  siempre existe y es un atractor estable. Si  $r \geq 1$  aparecen dos nuevos puntos de equilibrio  $C_+ = (\sqrt{b(r-1)}, \sqrt{b(r-1)}, r-1)$  y  $C_- = (-\sqrt{b(r-1)}, -\sqrt{b(r-1)}, r-1)$  que son estables en su nacimiento cuando  $r = 1$  y permanecen así para  $r < r_* \approx 24.74$ . Para  $r > r_*$  tanto  $(0, 0, 0)$  como  $C_+$  y  $C_-$  se vuelven inestables. Cuando  $r > r_*$  un atractor caótico aparece. Si se desea conocer más detalle sobre este análisis consultar [2].

Lo que Lorenz notó numéricamente es que con casi cualquier condición inicial daba como resultado una forma similar. Una trayectoria de espiral aparece al rededor del punto de equilibrio  $C_+$  o de  $C_-$  hasta que la espiral rebasa cierta distancia crítica del punto de equilibrio la espiral gira al rededor del otro punto de equilibrio aumentando las oscilaciones hasta que de nuevo

rebasa la distancia del punto del equilibrio lo que hace que las soluciones estén atrapadas alrededor de los dos puntos de equilibrio sin poder escapar.

### 3.1. El atractor extraño

Aún existen discusiones sobre una definición formal sobre lo que es un atractor extraño. Para este trabajo se aborda la siguiente noción que será útil para comprender las características más importantes de un atractor.

La idea de un atractor es un conjunto al que convergen las trayectorias cercanas a él. Los ejemplos más simples de atractor son los puntos fijos estables y los ciclos límite.

**Definición 3.1.1.** *Considere el sistema*

$$\dot{x} = f(x) \tag{3.4}$$

con  $f \in C^1(E)$  donde  $E$  es un subconjunto abierto de  $\mathbb{R}^n$ . Como vimos en la sección 1.1, el sistema 3.4 define un sistema dinámico  $\phi_t(x)$  en  $E$ .

Un **conjunto atractor** es un conjunto invariante  $A \subset E$  de (3.4) si existe alguna vecindad  $U$  de  $A$  tal que para todo  $x \in U$ ,  $\phi_t(x) \in U$  para todo  $t \geq 0$  y  $\phi_t(x) \rightarrow A$  cuando  $t \rightarrow \infty$ . Un **atractor** de (3.4) es un conjunto atractor que contiene una órbita densa [23].

En el sistema de Lorenz, cuando  $r$  toma valores entre 24.06 y 24.74 un atractor caótico extraño y un punto de equilibrio atractor estable coexisten entre sí. Lo que definirá el rumbo de las órbitas será lo cerca que estén las condiciones iniciales del atractor o del punto de equilibrio.

Otro punto importante a recalcar es que aunque existen intervalos de  $r$  para los cuales el comportamiento del sistema puede estar definido y contar con trayectorias caóticas también existen muchos valores fuera y dentro de dichos intervalos para los cuales no se conoce con exactitud el comportamiento del sistema o no se tiene certeza de que existan órbitas estables o periódicas. A algunos de estos valores se les puede llamar *ventanas de incertidumbre*, por ejemplo, para valores de  $r$  mayores a 50 Lorenz encontró órbitas periódicas

estables y para algunos otros no halló atractor caótico. El análisis de Lorenz para distintos valores del parámetro  $r$  se puede resumir en la siguiente tabla que se puede encontrar en ([2], Pág.365).

$r$	Atractor
$[-\infty, 1.00]$	$(0, 0, 0)$ es un punto de equilibrio atractor
$[1.00, 13.93]$	$C_+$ y $C_-$ son puntos de equilibrios atractores y el origen es inestable
$[13.93, 24.06]$	existen órbitas caóticas pero no atractor caótico
$[24.06, 24.74]$	Un atractor caótico coexiste con los puntos de equilibrio $C_+$ y $C_-$
$[24.74, ?]$	Caos: Un atractor caótico existe pero $C_+$ y $C_-$ no son atractores.

Tabla 3.1: Para  $\sigma = 10$ ,  $b = 8/3$  y  $r$  variando.

El atractor extraño de Lorenz se muestra en la figura 3.1.

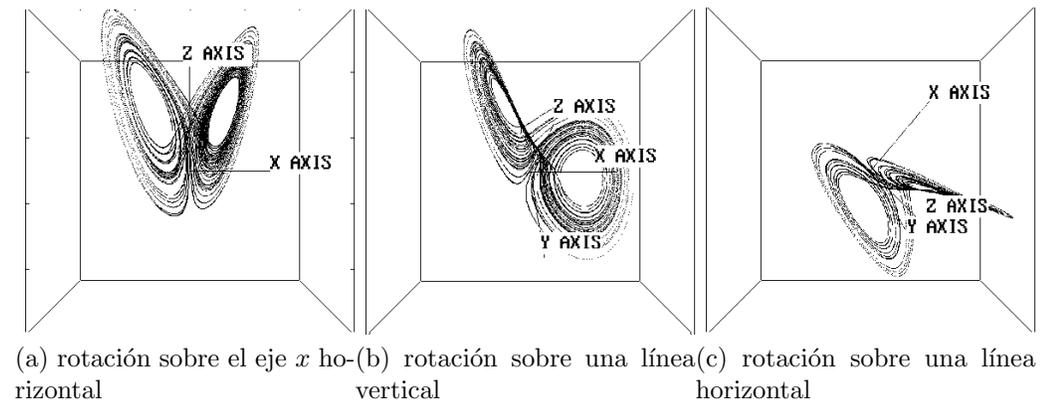


Figura 3.1: Vistas giradas del atractor extraño de Lorenz con  $r = 28$

## 3.2. Cálculo de los Exponentes de Lyapunov

En esta sección se introduce a grandes rasgos el cálculo del máximo exponente de Lyapunov del sistema de Lorenz como herramienta para probar caoticidad. Es importante esta sección porque en la mayoría de los artículos que se investigaron para este trabajo no se profundiza en la elección de ciertos parámetros en lugar de otros para generar órbitas caóticas. Sin embargo el análisis del cálculo de los exponentes de Lyapunov se torna complejo para funciones continuas por lo que solamente se mencionan las ideas más importantes sobre el cálculo del máximo exponente de Lyapunov. También se anexa el programa en Mathematica que sirve para encontrar el máximo exponente de Lyapunov de una órbita del sistema de Lorenz con ciertos parámetros fijos. Si al lector le interesa profundizar sobre este análisis en particular puede consultar ([2], Pags. 379-385).

Recordemos brevemente que el comportamiento local de la dinámica puede variar en las diferentes direcciones del espacio, dos puntos cercanos pueden moverse juntos a lo largo de un eje y alejarse a lo largo del otro. Podemos imaginar una esfera infinitesimal de condiciones iniciales que evoluciona en una elipse al iterarse el sistema. Como vimos anteriormente, la tasa de crecimiento promedio del eje ortogonal más largo se definió como el primer número de Lyapunov de la órbita y un exponente de Lyapunov positivo significa divergencia exponencial de trayectorias cercanas lo que nos indica existencia de caos.

Para funciones continuas podemos usar la misma idea pero necesitamos un nuevo concepto. Se define la *función discreta asociada a una función continua*  $f$ ,  $F_t(v_0)$ , como la función que representa al punto en el cual se encuentra  $f$  con condición inicial  $v_0$ , al tiempo  $t$ . Podemos entonces definir los números y exponentes de Lyapunov de un flujo continuo de acuerdo a la sección 2.3

**Definición 3.2.1.** *Sea*

$$\dot{v} = f(v) \tag{3.5}$$

*el sistema 3.4 antes definido, un sistema de  $n$  ecuaciones diferenciales autónomas, es decir, que no dependen explícitamente del tiempo con  $v = (x_1, x_2, \dots, x_n)$ . Definamos el exponente de Lyapunov de  $f$ .*

**Definición 3.2.2.** *Los **Números de Lyapunov (Exponentes de Lyapunov)** de la función discreta  $F_t(v)$  asociada a  $f$ , se definen como los números*

de Lyapunov (exponentes de Lyapunov) de su función discreta asociada al tiempo 1,  $F_1(v)$ .

La principal dificultad surge cuando queremos conocer  $DF_1(v)$  la derivada de  $F_1(v)$  con respecto al valor inicial  $v$ , ya que no necesariamente contamos con la forma explícita de  $F_1(v)$ . Para enfrentar esta situación se recurre al cálculo variacional. A continuación se da una idea de lo que se realiza para dicho cálculo. El código en Mathematica puede verse en el Apéndice C.

Si fijamos  $t$ , entonces  $DF_t(v)$  es una función lineal en  $\mathbb{R}^n$  y puede ser representada por una matriz de tamaño  $n \times n$ . Un problema será que no existe fórmula explícita de  $DF_t(v)$ , la alternativa entonces es hallar una ecuación diferencial que la involucre y que pueda resolverse en paralelo con (3.5). Por definición de  $F_t(v)$ ,  $\{F_t(v) : t \in \mathbb{R}\}$  es solución de 3.5 con valor inicial  $v$ , ya que por sí misma es una solución del sistema al tiempo  $t$ , entonces tenemos que

$$\frac{d}{dt}F_t(v) = f(F_t(v)).$$

Notemos que esta ecuación tiene dos variables, el tiempo  $t$  y el valor inicial  $v \in \mathbb{R}^n$ . Ahora derivemos con respecto a  $v$ , por la regla de la cadena tenemos que

$$\frac{d}{dt}DF_t(x) = Df(F_t(v)) \cdot DF_t(v) \quad (3.6)$$

A esta ecuación se le conoce como *ecuación variacional* de la ecuación diferencial. La idea es, que si podemos resolver la ecuación para  $DF_t(v)$  podríamos conocer la derivada de  $F_t$  y así, conocer cómo actúa  $F_t$  bajo pequeñas variaciones del valor inicial  $v_0$ .

Definamos  $J_t = DF_t(v)$  como el Jacobiano de la función discreta al tiempo  $t$  evaluada en el valor inicial  $v$ . Y sea  $A(t) = Df(F_t(v))$  la matriz de derivadas parciales del lado derecho de la ecuación (3.5) ya que conocemos explícitamente la ecuación diferencial original podemos calcular  $A(t)$ . Entonces podemos reescribir la ecuación (3.6) como

$$\frac{dJ_t}{dt} = A(t)J_t \quad (3.7)$$

En particular para el modelo de Lorenz tenemos que la función discreta  $F_t(v)$  se obtiene simplemente resolviendo las ecuaciones con la condición inicial  $v_0 = (x_0, y_0, z_0)$  y seguir su trayectoria al tiempo  $t$ . Entonces

$$F_t(x_0, y_0, z_0) = (x(t), y(t), z(t))$$

La matriz jacobiana  $J_t$  de  $F_t$  del sistema de Lorenz es:

$$A(t) = \begin{pmatrix} -\sigma & \sigma & 0 \\ r - z(t) & -1 & -x(t) \\ y(t) & x(t) & -b \end{pmatrix} \quad (3.8)$$

Calculamos cada columna de  $J_T$  de la ecuación variacional por ejemplo la primera columna se encuentra calculando la solución de la ecuación diferencial

$$\begin{pmatrix} \dot{J}_{11}(t) \\ \dot{J}_{21}(t) \\ \dot{J}_{31}(t) \end{pmatrix} = A(t) \begin{pmatrix} J_{11}(t) \\ J_{21}(t) \\ J_{31}(t) \end{pmatrix}$$

La ecuación se resuelve numéricamente con ayuda de Runge Kutta de orden 7 (Ver [11]). Se debe notar que  $A(t)$  involucra a las soluciones  $(x(T), y(T), z(T))$  es por esto que la ecuación variacional debe resolverse simultáneamente con una solución del sistema (3.5).

Finalmente se define una órbita caótica para un flujo continuo de la siguiente manera: (ver [2], Pag. 386), usando la definición de exponente de Lyapunov para un flujo continuo.

**Definición 3.2.3.** Sea  $F_t(v_0)$  solución de  $\dot{v} = f(v)$  donde  $v_0 \in \mathbb{R}^n$ . Decimos que la órbita  $F_t(v_0)$  es caótica si las siguientes condiciones se cumplen:

1.  $F_t(v_0), t \geq 0$ , es acotada.
2.  $F_t(v_0)$  tiene al menos un exponente de Lyapunov positivo, y
3. La órbita de  $v_0$ ,  $O(v_0)$  no es periódica y no consiste solamente de puntos de equilibrio.

Usando este análisis se resuelve la ecuación variacional y de manera simultánea la ecuación (5.2). La condición inicial será  $J_0 = I$ , la matriz identidad. Esto es porque se satisface que  $F_0(v) = v$  y así calcular el máximo exponente de Lyapunov para el sistema de Lorenz y constatar que es positivo. Podemos recalcar dos detalles: la ecuación (3.7) es una ecuación diferencial lineal mientras que la ecuación original (3.5) no lo es y además no es autónoma ya que depende del tiempo explícitamente. Para ver más detalle de consultar el código del Apéndice C en *Mathematica* para el cálculo del exponente de Lyapunov [11]).

Para los valores de los parámetros  $\sigma = 10$ ,  $b = 8/3$  y  $r = 28$  con ayuda del código en *Mathematica* el máximo exponente de Lyapunov que se obtiene es aproximadamente  $\lambda = 0.90037$ , lo cual indica que una órbita típica con los parámetros escogidos es caótica.

## Capítulo 4

# Criptografía basada en sistemas caóticos

En este capítulo se abordan las similitudes entre un sistema caótico y un sistema de cifrado. Se presentan las características que relacionan la difusión de un cifrado con la sensibilidad a condiciones iniciales en un sistema caótico, que es la motivación de este trabajo.

De la necesidad de compartir información de forma confidencial y segura, la humanidad se ha visto obligada a adquirir herramientas para proteger un mensaje que será enviado a través de un medio convencional. Cuando dicho medio es vulnerable al ataque de un intruso, éste podría obtenerla y hacer mal uso de ella. Para combatir el problema de que nuestra información sea robada, surgieron nuevas técnicas para disfrazarla, o bien, confundir al emisor no autorizado para que no se percate que es información valiosa.

Existen dos formas de resolver este problema, la primera se basa en esconder la existencia misma del mensaje, por ejemplo, una manera de esconder una cadena de bits que contenga cierta información valiosa es ocultar dicha cadena dentro de un archivo imagen que al ser percibida por un intruso parezca que no hay más que lo que ve, sin sospechar que existe algo escondido de otra naturaleza. A este método de esconder el mensaje se le llama *esteganografía*.

La esteganografía puede ser útil cuando la intención es no despertar la sospecha en algún receptor de que la información enviada es importante, sin

embargo en la práctica este método tiene sus debilidades e inconvenientes, por ello una segunda manera de mantener seguro un mensaje consiste en transformar de alguna manera la información para que en el caso que algún receptor no autorizado la obtenga, no tenga forma de entenderla. Esta segunda técnica se le conoce como *cifrar*.

## 4.1. Definición y Terminología

La *Criptografía* define el arte y la ciencia de la transformación de la información en una secuencia de símbolos que parece al azar y sin sentido para algún observador no autorizado. El *Criptoanálisis* es la ingeniería inversa de la Criptografía. Se le llama así a los intentos por identificar los puntos débiles de los distintos algoritmos criptográficos y sus implementaciones para la explotación de los mismos. ([24]).

Se le llama *Criptología* al estudio que abarca la criptografía y el criptoanálisis así como sus propiedades matemáticas.

El algoritmo usado para cifrar el mensaje se le conoce como *criptosistema* o *sistema de cifrado* y a su vez, el que describe la forma de recuperar el mensaje original a partir de un mensaje cifrado se le llama *sistema de descifrado*.

También cabe señalar que el criptoanálisis puede realizarse con la intención de analizar la *robustez* o fortaleza del sistema de cifrado y hacer mejoras en el mismo. Cuando existe un intento de criptoanálisis, éste se considera un ataque al sistema de cifrado en cuestión.

Podemos definir un criptosistema o sistema de cifrado como una quintupla  $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$  donde:

$\mathcal{P}$  es el conjunto finito de posibles mensajes a cifrar o *textos en claro*.

$\mathcal{C}$  es el conjunto finito de posibles mensajes cifrados.

$\mathcal{K}$  es el conjunto de posibles llaves, también llamado *espacio de llaves*.

$\mathcal{E}$  es el conjunto de posibles reglas de encriptación y  $\mathcal{D}$  el de posibles reglas

de descryptación, donde para cada  $k \in \mathcal{K}$ , los elementos de éstos conjuntos son funciones  $e_k \in \mathcal{E}$  y  $d_k \in \mathcal{D}$  tales que:

$e_k : \mathcal{P} \rightarrow \mathcal{C}$  y  $d_k : \mathcal{C} \rightarrow \mathcal{P}$  deben cumplir:

$$d_k(e_k(x)) = x \quad \text{para todo texto en claro } x \in \mathcal{P}.$$

A continuación se dan algunos antecedentes históricos de la criptografía así como la una descripción breve de los cifrados más importantes.

## 4.2. Historia

Si pensamos en criptografía inmediatamente la relacionamos con redes y medios electrónicos modernos. Sin embargo la necesidad de esconder información de los enemigos es tan antigua como la comunicación misma, algunas técnicas criptográficas clásicas datan de varios siglos atrás.

En este capítulo se describe parte de la historia más importante de la criptografía que da origen a los cifrados modernos que actualmente se utilizan para cifrar información bancaria, confidencial y gubernamental, entre otras. Se abordan también las ideas generales de algunos cifrados antiguos que fueron el nacimiento de las técnicas criptográficas, y los cifrados *DES* y *AES* se describen por ser algunos de los cifrados modernos más conocidos y por la complejidad de su estructura. Adicional se mencionan aunque sin detalle los cifrados de *RSA*, *Elgamal* y *Curvas Elípticas* que juegan un papel importante en la historia de la criptografía.

La palabra *criptografía* tiene su origen etimológico del griego **Kriptos** que significa *ocultar* y **Graphos** que significa *escritura* y es introducida a la formalidad de la ciencia por el matemático Claude Elwood Shannon a quien se le ha denominado “Padre de la Criptografía” después de publicar en 1949 su artículo *Communication Theory of Secrecy System* [26], en el cual Shannon establece una base sólida teórica para la criptografía y el criptoanálisis.

La criptografía se divide históricamente en dos partes: La criptografía clásica y la criptografía moderna, y ambas se clasifican de acuerdo a las técnicas o métodos que se utilizan para cifrar el mensaje.

La criptografía clásica puede entenderse como la criptografía no computarizada o digitalizada y los métodos de cifrado se mantenían en secreto a diferencia de los utilizados en la criptografía moderna. Dichos métodos eran variados, había algunos simples y otros más ingeniosos que incluían mecanismos para la transformación del mensaje en claro de forma más compleja como ejemplo conocido, la máquina Enigma que se utilizó en la segunda guerra mundial.

Dos técnicas de cifrado clásico importantes son el cifrado por permutación y el cifrado por sustitución. El primer tipo consiste en un reordenamiento de los caracteres del mensaje por medio de un algoritmo específico, y los cifrados por sustitución utilizan alguna técnica de modificación de cada caracter del texto de acuerdo a una relación entre el alfabeto original y un alfabeto propuesto para el cifrado.

En esta sección se incluyen dos de las técnicas más interesantes de cifrados clásicos, la escítala, el cifrado César y Vigenére. Para profundizar acerca de los métodos criptográficos siguientes consultar [13].

### La Escítala

La escítala es un ejemplo de cifrado por permutación, es una técnica para cifrar que se utilizó en el siglo V a.C. por un antiguo pueblo griego que consistía en escribir un mensaje en una cinta de forma longitudinal y después enrollarla en un bastón como se muestra en la figura 4.1 para posteriormente desenrollarla y entregarla al mensajero. La llave de este sistema reside en el diámetro del bastón de forma que solamente el receptor autorizado contaría con una réplica exacta de aquel bastón en el que enrollaba el mensaje y podía leer el mensaje original.



Figura 4.1: Mensaje enrollado en el bastón: Método de Escítala

### Cifrado de César

En el siglo I a.C. surge una técnica en honor al emperador Julio César y forma parte de los cifrados clásicos más conocidos en la actualidad. El método de cifrado de César es un cifrado por sustitución que consiste básicamente en aplicar una transformación al mensaje en claro de tipo monoalfabética. Se establece una relación entre el alfabeto del mensaje en claro y un alfabeto de cifrado que se genera aplicándole un corrimiento cíclico, tres espacios hacia la derecha al primer alfabeto. La transformación que realiza el cifrado de César de los caracteres para el alfabeto castellano de 27 letras se muestra en la figura 4.2.

De este modo, al cifrar cada letra del mensaje en claro se sustituye por la letra del alfabeto desplazado y se obtiene un mensaje en general ilegible. Una forma matemática de explicar este cifrado es la siguiente:

El alfabeto se enumera iniciando en 0, de modo que a cada letra le corresponde un número del 0 al 26 de la forma,  $A = 0, B = 1, \dots, Z = 26$ .

Luego, sea  $M = m_1, m_2, \dots, m_n$  el mensaje a cifrar. Para generar el mensaje cifrado  $C = c_1, c_2, \dots, c_n$  se tiene que

$$c_i = (m_i + 3) \text{ mod } 27^1$$

donde  $i = 1, 2, \dots, n$  y  $c_i$  es la letra cifrada y  $m_i$  es la letra a cifrar.

Para descifrar el mensaje realizamos

$$m_i = (c_i - 3) \text{ mod } 27 = (c_i + 24) \text{ mod } 27$$

con  $i = 1, 2, \dots, n$  con  $n$  el número de letras del mensaje.

Alfabeto	A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
Alfabeto	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Figura 4.2: Transformación del Alfabeto del cifrado de César

<sup>1</sup>Recuerde que  $a = r \text{ mód } m$  significa que  $a \equiv r \text{ (mód } m)$  y  $0 \leq r < m$ .

Otros cifrados importantes son los llamados polialfabéticos, la idea es utilizar varios alfabetos distintos, alternando entre ellos mediante algún esquema fijo, esto ayuda a disfrazar las frecuencias en las que aparecen las letras en el mensaje cifrado haciéndolo más seguro. Un cifrado polialfabético importante es el de **Vigenére** [12] que básicamente utiliza una tabla y una llave que puede ser una palabra o frase clave, y la idea es utilizar un monoalfabeto distinto para cada caracter del mensaje, en el orden especificado por la llave. Para conocer más detalle sobre Vigenère consultar ([12], Pág. 36).

La criptografía moderna se divide en dos tipos: la criptografía simétrica o de llave secreta, y la criptografía asimétrica o de llave pública.

La criptografía simétrica o de llave secreta es aquella en la que se usa la misma llave para cifrar y para descifrar el mensaje. Por obvias razones, dicha llave permanece en secreto. Como tanto emisor como receptor del mensaje deben contar con la misma llave secreta, el principal riesgo de la criptografía simétrica es la distribución de dicha llave a los usuarios que deban conocerla y el almacenamiento y protección de tantas llaves diferentes se requieran.

A su vez, la criptografía asimétrica o de llave pública utiliza dos llaves diferentes. Una es la llave pública que puede ser enviada a cualquier persona y la otra es una llave privada que debe ser guardada para que nadie tenga acceso a ella.

Dentro de la criptografía simétrica se han desarrollado diversos cifrados siendo el más usado el Cifrado de Datos Estándar, por sus sigas en inglés DES (Data Encrption Standard). Este surge tras una necesidad inminente de comunicaciones confidenciales entre equipos de cómputo a principio de la década de los setentas. Ante esta situación el *National Institute of Standards and Technology* o NIST emite una convocatoria pública para recibir propuestas para un algoritmo criptográfico estándar para ser utilizado por las dependencias gubernamentales. Al no tener éxito se emitió una nueva convocatoria el 27 de agosto de 1974, y en esta última se presentó un nuevo algoritmo presentado por un grupo de investigadores de IBM; el nombre del algoritmo era *Lucifer* [9]. Tras la controversia que existió después de que expertos en el área sospecharan de *puertas traseras* en el cifrado, el algoritmo se modificó en noviembre de 1976 y se adoptó como estándar para comunicaciones gubernamentales con el nombre de cifrado DES. Con esto, DES se

convirtió en el primer sistema de cifrado avalado por un gobierno.

### Descripción del cifrado DES

En seguida se da una breve descripción del cifrado DES, para propósitos de esta tesis solo se revisará los rasgos generales y ventajas del cifrado, si el lector desea una revisión más profunda puede consultarla en [12].

El cifrado DES es un algoritmo del tipo *redes de Feistel* ([12], Pags. 184-186). Consta de 16 rondas, cada una casi idéntica a las demás. DES usa una llave  $K$  que debe cumplir con ciertas pruebas de aleatoriedad, especificadas en el *Approved Random Bit Generator* (RBG), que detalla las reglas a seguir para que la llave sea lo suficientemente segura y resistente [4]. Tanto  $M$  el mensaje en claro, como  $K$  consisten de 64 bits originalmente pero en la llave ocho bits son considerados como bits de verificación de paridad, así que realmente la llave tiene  $64 - 8 = 56$  bits.

Al texto en claro se le aplica una permutación inicial  $PI$  que tiene el único objetivo de hacer más eficaz la implementación computacional del cifrado.

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Figura 4.3: Permutación inicial DES

Básicamente se separa el texto en claro en dos mitades  $R_0$  y  $L_0$ , luego de pasar por la permutación inicial, ambas son de 32 bits. La salida de esta ronda se constituye por dos mitades que forman la entrada de la siguiente ronda de la siguiente manera:

$$L_{i+1} = R_i$$

$$R_{i+1} = L_i \oplus f(R_i, k_i)$$

#### 48CAPÍTULO 4. CRIPTOGRAFÍA BASADA EN SISTEMAS CAÓTICOS

Las etapas del DES son las siguientes [12]:

1. Permutación de expansión. Luego de la permutación inicial, los bits de  $R_i$  pasan por una pseudo permutación expansión. Esta transformación tiene como entrada los 32 bits de  $R_i$  y da como salida 48 bits (de aquí que no sea una permutación formalmente hablando).
2. A la salida de la permutación de expansión se le aplica suma XOR con los 48 bits de la llave correspondiente a la ronda.
3. Al resultado de la salida del XOR se le aplica un procedimiento de sustitución que consiste en 8 cajas de sustitución o S-cajas.

En la figura 4.4 se muestra un esquema de la primera ronda del cifrado DES.

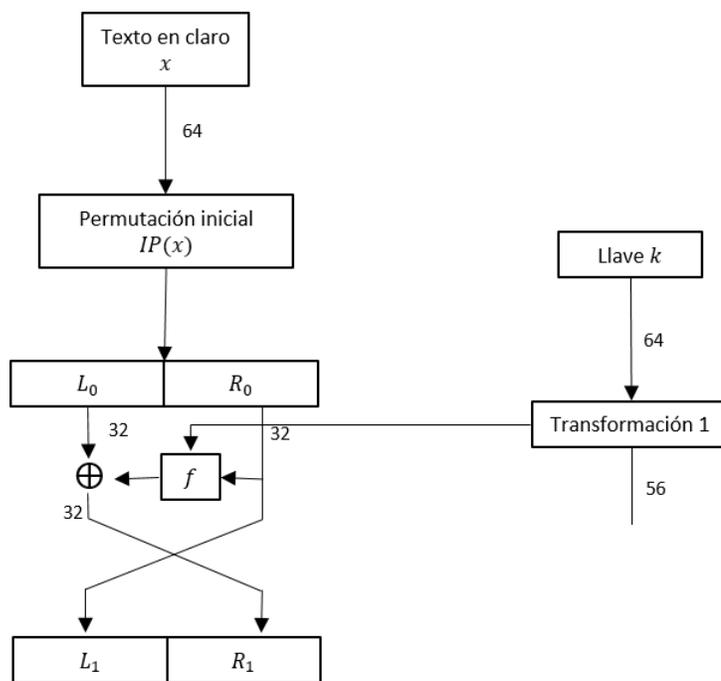


Figura 4.4: Primera ronda del DES

La razón principal de lo complejo del algoritmo del cifrado DES es lograr dos objetivos importantes en criptografía que hemos mencionado antes: la

difusión y la confusión. En el DES la relación entre el texto a cifrar y la llave es compleja y cada bit de la salida está en función de todos los bits de la entrada y los de la llave, es decir, el cambio de un sólo bit del mensaje en claro repercutirá fuertemente en al menos la mitad de los bits de la salida del cifrado. Esta es una característica que desde entonces se ha mantenido en los sistemas de cifrado.

El cifrado DES fue considerado seguro hasta el año de 1998 donde su debilidad fue probada, el tamaño de la clave era reducido (64 bits) y se necesitaban cerca de  $2^{47}$  textos elegidos para romper el cifrado por un ataque exhaustivo o fuerza bruta, lo que no era posible para las computadoras de los 90's. Cuando se demostró su vulnerabilidad se le hizo una modificación que llamaron triple DES (3DES) que era básicamente repetir el algoritmo 3 veces sobre un mismo bloque con 2 o 3 claves distintas cada vez, esto lo hizo más seguro pero 3 veces más lento [7].

### Descripción del cifrado AES

Después de haberse adoptado el DES como cifrado estándar, se hizo obligatoria la revisión periódica del mismo. Sin inconvenientes el DES aprueba las revisiones de 1983 y 1987, pero en septiembre de 1997 no corre con la misma suerte y el cifrado es roto. Debido a esto se anuncia la convocatoria para recibir algoritmos propuestos que reemplacen al DES. En octubre del 2000 se anuncia oficialmente el ganador, un proyecto propuesto por los belgas Joan Daemen y Vincent Rijmen y un año después se hace público el nuevo cifrado (Advanced Encryption Standard) AES y se convierte en el estándar hasta la fecha. [12].

Una de las razones principales para la aceptación del cifrado AES es la fácil implementación del algoritmo, a pesar de su complejidad matemática se sabe que el cifrado AES tiene una alta eficiencia computacional tanto en hardware como en software, véase [9].

El AES es un cifrado en bloques simétrico que procesa longitudes de bloques de 128 bits y longitudes de llave de 128, 192 y 256 bits. Con el fin de calificar como buen candidato para sustituir al DES, el AES tuvo que proveer una serie de pruebas conocidas y resistencia a ataques criptográficos. Es un cifrado iterativo, el número de rondas se denota por  $N_r$  y depende de la lon-

gitud del bloque y la longitud de la llave. La primera fase es una adición con la llave inicial denotada por *AddRoundKey* seguida de  $N_{r-1}$  aplicaciones de la ronda de transformación que es una sucesión de cuatro transformaciones que se muestran en la Tabla 4.1 y por último una ronda final llamada *FinalRound*. La llave usada para la ronda  $i$  se denota por *ExpandedKey[i]* y la derivación de la *ExpandedKey* de la llave del cifrado se denota por el proceso *KeyExpansion*.

1. SubBytes
2. ShiftRows
3. MixColumns
4. AddRoundKey

Tabla 4.1: Etapas de la ronda de transformación

Para consultar acerca del cifrado AES puede consultar ([9], pág. 33). Las técnicas actuales de criptoanálisis muestran que la resistencia de los cifrados de bloque a los ataques criptográficos aumenta con el número de rondas. La forma de determinar el número de rondas necesarias en el AES fue tomar el número máximo de rondas necesarias para resistir algunos ataques de atajo<sup>2</sup>(Ver [9], pág. 71). Además se añadieron cuatro rondas de seguridad y se ha mostrado que después de 6 rondas el cifrado por bloques de longitud de 128 bits es resistente a los ataques.

Parte de la robustez del AES se basa en la combinación de buena ingeniería y requisitos cumplidos que lo convierten en un cifrado único en su diseño y óptimo en su estructura hasta la fecha, sin embargo los intentos de ataques contra el AES continúan, poniendo en duda si el tamaño de la llave, 128 bits o 256 bits, será suficiente para resistir las técnicas computacionales modernas. Más información a detalle de la estructura del cifrado AES se puede encontrar consultando [9].

## RSA

---

<sup>2</sup>Los ataques de atajo “*shortcut attacks*” son aquellos en los que el adversario explota algunas de las propiedades del algoritmo de cifrado que le permita determinar la llave o el texto en claro en menos tiempo que realizando una búsqueda exhaustiva

El cifrado RSA fue propuesto en 1977 por Ronald Rivest, Adi Shamir y Leonard Adleman, de cuyas iniciales obtuvo su nombre. El RSA se convirtió en el cifrado asimétrico más utilizado hasta la fecha, patentado en USA aunque no en el resto del mundo hasta el 2000. Entre las aplicaciones más comunes del RSA están el cifrado de textos pequeños, especialmente el transporte seguro de llaves y firmas digitales o certificados digitales en Internet.

Una desventaja del RSA es que es varias veces más lento que el AES debido a la cantidad de cálculos involucrados en su implementación. La función de una sola dirección subyacente al RSA es el problema de factorización de enteros, multiplicar dos números grandes es tarea sencilla pero factorizar el producto resultante es bastante complejo. La función  $\phi$  de Euler y el teorema de Euler juegan un papel importante en la implementación de RSA, para consultar la descripción del cifrado RSA ver [22].

El RSA tiene otras vulnerabilidades importantes:

- El cifrado RSA es determinista, es decir, para una llave específica un texto en claro en particular genera siempre un texto cifrado en particular. Un atacante puede obtener información de un análisis estadístico y obtener así parte del texto en claro.
- Los textos en claro  $x = 0$ ,  $x = 1$  o  $x = -1$  producen los textos cifrados  $x = 0$ ,  $x = 1$  y  $x = -1$ .
- Los exponentes pequeños son vulnerables a ataques si no se usa un método de relleno *padding*

Otra debilidad del RSA es que es *maleable*, esto significa que un atacante puede transformar el texto cifrado en otro que conduzca a una transformación conocida de texto en claro (Ver [13] pág. 234).

El RSA se considera un cifrado seguro mientras no se encuentren formas rápidas de descomponer un número grande en producto de primos. En 1993 Peter Shor publicó un algoritmo mostrando que una computadora cuántica puede resolver la factorización en tiempo polinomial, sin embargo las computadoras cuánticas no se han desarrollado por completo en la actualidad [22].

## Elgamal

El cifrado Elgamal es propuesto por el criptógrafo egipcio Taher Elgamal en 1985, su seguridad se basa en la insolubilidad del problema del logaritmo discreto y Diffie-Hellman. Elgamal trabaja sobre el grupo  $\mathbb{Z}_p^*$  donde  $p$  es un número primo, pero puede aplicarse a otros grupos cíclicos, por ejemplo el grupo multiplicativo de  $GF(2^m)$  [22].

Un hecho importante es que Elgamal es un cifrado probabilístico, es decir, el cifrado de dos mensajes idénticos  $x_1$  y  $x_2$  donde  $x_1, x_2 \in \mathbb{Z}_p^*$  usando la misma llave pública genera dos mensajes cifrados distintos  $y_1 \neq y_2$ . Esto tiene que ver con que parte de la clave usada para cada ronda es elegida aleatoriamente, de ésta forma el ataque exhaustivo se evita. Tomando en cuenta que la seguridad del cifrado de Elgamal se basa en el problema del logaritmo discreto y en la actualidad el algoritmo computacional para calcular logaritmos discretos tiene la misma complejidad que la de factorizar dos números primos no es posible realizar esta tarea en tiempo polinomial, y el cifrado se considera un cifrado efectivo.

## Curvas Elípticas

Los Cifrados de Curvas Elípticas (CCE) se presentan desde mediados de la década de los 80's. Los CCE están basados en el problema del logaritmo discreto generalizado por lo que proporcionan un nivel de seguridad similar al del cifrado RSA. En general los CCE suelen tener ventajas de rendimiento sobre el RSA, sin embargo las operaciones que se involucran en el RSA son más rápidas que las que se usan en los CCE.

Para los CCE las curvas elípticas no se consideran sobre  $\mathbb{R}$  sino sobre un campo finito. La opción más común son los campos  $GF(p)$  y la aritmética se realiza módulo  $p$ , donde  $p$  es primo. La descripción general de curva elíptica es la siguiente:

**Definición 4.2.1.** *La curva elíptica sobre  $\mathbb{Z}_p$ ,  $p > 3$ , es el conjunto de todos los pares  $(x, y) \in \mathbb{Z}_p$  que satisfacen*

$$y^2 \equiv x^3 + ax + b \pmod{p}$$

junto con un punto imaginario llamado punto al infinito  $\mathcal{O}$  donde  $a, b \in \mathbb{Z}_p$  y además  $4a^3 + 27b^2 \neq 0 \pmod p$

En los elementos del conjunto anterior se definen operaciones de grupo de forma que podamos sumar y duplicar puntos que no se especifican en este trabajo pero pueden consultarse en ([22], pág. 242.) A su vez es posible encontrar el inverso a cada punto en la curva y proporcionar un elemento identidad.

En general los ataques potenciales contra el problema de logaritmo discreto módulo  $p$  no se aplican a curvas elípticas. Con la buena elección de curva elíptica el cifrado puede ser resistente a la mayoría de los ataques contra el PLD. En la práctica se utilizan curvas estandarizadas propuestas en el Instituto Nacional de Estándares y Tecnologías (NIST). Como se ha mencionado, el costo computacional de las operaciones para los CCE es alto, pero existen modos para resolver este problema, una curva especial que agiliza las operaciones, en específico hace casi instantánea la multiplicación es la curva de Koblitz, (consultar más detalles en [22], pág 254).

### 4.3. Criptografía basada en Caos

A la par del rápido desarrollo de las telecomunicaciones modernas, la telefonía móvil y el internet, los sistemas criptográficos requieren un progreso acelerado por lo cual nuevas técnicas han surgido estimulando diversas áreas de investigación referentes al cifrado de la información. Desde 1990 muchos de estos investigadores se han dado cuenta de que existe una relación entre caos y criptografía (cf. [3]).

De las similitudes entre Caos y Criptografía podemos encontrar sensibilidad a cambios en condiciones iniciales y parámetros que causan un comportamiento casi aleatorio, para seguir con órbitas periódicas, atractores extraños y conducirnos a las rondas en un algoritmo criptográfico, propiedades de difusión y confusión deseadas en un cifrado. En un sistema caótico podemos encontrar un camino que nos conecte a la criptografía y viceversa. De aquí que los parámetros de una función caótica pueden representar la llave o parte de ella para el algoritmo del cifrado, como es el caso del sistema propuesto en este trabajo.

Prop. Caótica	Prop. Criptográfica	Descripción
Ergodicidad	Confusión	Para cualquier entrada, la salida tiene la misma distribución
Sensibilidad a condiciones iniciales	Difusión	Un pequeño cambio en la entrada puede causar un cambio grande en la salida
Dinámica determinista	Pseudoaleatoriedad	Proceso determinista con comportamiento pseudo-aleatorio.
Estructura compleja	Complejidad de algoritmos	Procesos simples con alta complejidad

Tabla 4.2: Similitudes entre Criptografía y Caos

A partir de estas similitudes una basta variedad de sistemas criptográficos basados en caos han sido diseñados. A continuación se listan para el lector interesado, algunos de las propuestas de cifrados basados en caos más importantes realizados en los últimos años. En la bibliografía se describen algunas referencias adicionales sobre el tema.

En la tabla 4.2, se muestran algunas similitudes entre las propiedades más importantes del caos y su contraparte en la criptografía.

- Cifrado de bloques basado en múltiples funciones caóticas unidimensionales [29].
- Cifrado en flujo utilizando atractor de Pickover como generador de corriente aleatoria para cifrar [1].
- Algoritmo caótico unidimensional para cifrar imágenes médicas [8].
- Técnica de cifrado de imágenes basado en un generador caótico neuronal [14].
- Cifrado de texto e imágenes utilizando algoritmo RSA y Caos [21].

A pesar de las similitudes descritas antes, una relación formal entre el caos y la criptografía aún no ha sido establecida. Más aún, el impacto que las investigaciones antes mencionadas han tenido en la criptografía convencional es marginal. Principalmente esto es por dos razones: la mayoría de los

algoritmos criptográficos basados en caos que se han utilizado hasta ahora usan sistemas dinámicos definidos en el conjunto de los números reales, por lo que la implementación en la práctica se vuelve difícil y compleja.

El segundo punto es en cuanto a la seguridad y rendimiento de los métodos basados en caos, en la mayoría de los casos no han sido analizados en términos de las técnicas convencionales ya descubiertas en la criptografía. Además, la mayoría de los métodos propuestos tienen implementaciones computacionales lentas o poco eficaces en la práctica.

En este trabajo se abordan ambos puntos de este dilema. Para la seguridad y rendimiento del sistema se analiza el cifrado propuesto bajo el criptoanálisis diferencial, este último se aborda en un capítulo y se adapta para hacer pruebas de robustez al cifrado propuesto. Además de la realización de histogramas para las imágenes cifradas. También se optó por utilizar el modelo caótico de Lorenz discretizado que nos ofrece una posibilidad de trabajar en el mundo discreto que requiere un algoritmo criptográfico sin modificar de forma abrupta las propiedades del sistema original de Lorenz.

Otra aportación de este trabajo es que, a diferencia de la mayoría de los trabajos que se encuentran en la literatura acerca de sistemas criptográficos basados en sistemas caóticos, el cifrado que se propone en este trabajo no carece de detalles importantes como la generación de las llaves a utilizar, las descripción de las rondas en el cifrado, así como la forma de elegir los parámetros y condiciones iniciales con el fin de que cualquiera que esté interesado en su funcionamiento pueda poner en práctica el sistema.

De acuerdo a [3] un punto importante para una propuesta de cifrado basado en caos debe ser el bajo costo computacional y la velocidad del mismo debe ser aceptable, para esto se prueba el tiempo de ejecución de cifrado para texto e imágenes.



# Capítulo 5

## Cifrado utilizando el modelo caótico de Lorenz

En este capítulo se explica en qué consiste el cifrado propuesto en el presente trabajo, así como el algoritmo para la generación de subllaves y algunos aspectos que debe cumplir dicho algoritmo para que las subllaves sean lo más seguras posibles. El cifrado que se realiza en este trabajo es un sistema por bloques de llave privada que utiliza el modelo caótico de Lorenz [16]. Se explicará la metodología del cifrado de imágenes y al final de capítulo se hará hincapié en la forma en que se puede utilizar para cifrar datos con alguna modificación simple.

Antes de comenzar hay que decir que una imagen digital a color puede verse como una matriz  $A \in M_{n_1, n_2}(\mathbb{Z}_{256}^3)$ , y  $\mathbb{Z}_{256}^3 = \{(a_1, a_2, a_3), a_i \in \{0, 1, \dots, 255\}\}$ , los arreglos  $(a_1, a_2, a_3)$  son los píxeles de la imagen. El píxel  $j$ -ésimo se puede ver de la siguiente forma:

$$p_j = (x_{j,1}, x_{j,2}, x_{j,3})$$

donde cada entrada de  $p_j$  es un número entero entre 0 y 255, es decir, un elemento de  $\mathbb{Z}_{256}$ , éstos son llamados subpíxeles y definen la proporción de los colores rojo, verde y azul (RGB) de cada píxel, respectivamente.

## Descripción del cifrado

Un bloque consta de 24 pixeles, es decir 72 bytes (576 bits), dicha longitud se eligió para que el número de pixeles a procesar en cada iteración del cifrado fuera sustancial y reducir el tiempo de ejecución del algoritmo. Sean  $B_0$  y  $B_5$  el bloque en claro de la imagen y el bloque cifrado respectivamente.

El número de rondas se determinó, de acuerdo al libro de Rijndael [9], pág. 41) considerando el número máximo de rondas para las cuales se ha visto que los ataques de atajo (ver sección 4.2) son significativamente más eficientes que los ataques de búsqueda exhaustiva de llave, luego se añade un número de rondas de seguridad. Se puede considerar que una ronda completa de cifrado propuesto provee de difusión completa, esto significa, que el cambio de un bit afecta al menos a la mitad de los bits del texto o imagen a cifrar en la siguiente ronda, se piensa que una buena difusión se logra debido a que la primera etapa de una ronda cifra todos los bloques de izquierda a derecha y la segunda lo hace de derecha a izquierda, (ver sección 5.2). Se piensa que de este modo 5 rondas de cifrado proveerían de buen margen de seguridad, ya que cada ronda cifra de derecha a izquierda y después de izquierda a derecha una ronda es prácticamente dos rondas en una. Se realizaron pruebas de robustez con este número de rondas.

Las 5 rondas idénticas se aplican a los elementos de  $B_0$  para obtener de salida el bloque  $B_5$ . Cada ronda  $i$  es controlada por una subllave  $K_i$  que consta de 72 bytes cuya generación se describe a detalle más adelante.

Un bloque del cifrado se forma con 24 pixeles concatenados de tres en tres de forma que se tengan 8 grupos de tres pixeles a los que se les llamará “*sub-bloques*”. Esto es con el fin de procesar una cantidad grande de pixeles en cada iteración del cifrado sin que éste deje de ser eficiente. Los bloques a procesar se permutan tomando los elementos del pixel 0, los del 8 y los del 16 de la imagen, estos forman el sub-bloque  $X_0$ .

$$X_0 = (x_{0,1}, x_{0,2}, x_{0,3}, x_{8,1}, x_{8,2}, x_{8,3}, x_{16,1}, x_{16,2}, x_{16,3}) = (p_0, p_8, p_{16}).$$

Para el siguiente bloque tenemos  $X_1 = (p_1, p_9, p_{17})$  y para el siguiente tenemos  $X_2 = (p_2, p_{10}, p_{18})$ . En general tenemos que las partes de cada bloque

se ven de la forma:

$$X_i = (p_i, p_{i+8}, p_{i+16}), \text{ donde } 0 \leq i \leq 7.$$

El objetivo de elegir los pixeles de esta forma es evitar que en un ataque al cifrado sea posible “perseguir” los elementos del primer pixel a través de todo el cifrado, es decir, si el primer pixel continua en la misma posición después de una ronda del cifrado, un atacante puede obtener información del algoritmo que se utilizó para cifrar.

Antes de comenzar a cifrar, se determina el número de pixeles de ancho de la imagen, esta cantidad se divide entre 24 y la parte entera de dicho cálculo es el número de bloques que se procesarán por renglón en la primera etapa del cifrado. Por ejemplo, si tomamos la imagen estándar de Lena de 256 pixeles de ancho por 256 pixeles de largo, dividimos  $\frac{256}{24} = 10.666\dots$ , entonces se cifrarían 10 bloques en cada renglón en la primera etapa. Las etapas del cifrado se explican con más detalle en la sección de **Esquema del cifrado**.

El cifrado de este trabajo está basado en el propuesto por Kocarev en su publicación *Chaos and cryptography: Block encryption ciphers based on chaotic maps* [16], la principal razón de elegir este modo de cifrado es la comodidad de la implementación computacional, ya que las operaciones necesarias son solo sumas X-OR y Tablas de sustitución, sin embargo la parte de la función caótica le provee la complejidad necesaria para ser un cifrado seguro. Kocarev propone un cifrado de datos que consta de  $r$  rondas de transformaciones idénticas aplicadas a los elementos del bloque en claro de la forma:

$$Y_{k+1} = X_k \oplus f_{k-1}[X_1, \dots, X_{k-1}, K_{k-1}], \text{ con } k = 1, \dots, 8 \quad (5.1)$$

donde  $X_0, \dots, X_7$  son los 8 subbloques que componen  $B_0$ ,  $Y_0, \dots, Y_7$  son los subbloques de  $B_1$ , que es la salida de la primera ronda y  $\oplus$  es la suma módulo 256. Los índices se toman mod 8, es decir  $Y_8 \equiv Y_0$  y  $Y_9 \equiv Y_1$ , y las funciones  $f_0, \dots, f_7$  tienen la siguiente forma:

$$f_0 = f(k_0)$$

$$f_j = f(X_1, \dots, X_j, K_j)$$

donde  $j = 0, \dots, 7$ ,  $k_0, \dots, k_7$  son los 8 elementos respectivos de la subllave  $K_i$  y  $f : M \rightarrow M$ ,  $M = \{0, \dots, 255\}$  se deriva de alguna función caótica. En los ejemplos estudiados por los autores de [16],  $f$  tiene la forma  $f = g(x_1 \oplus x_2 \oplus \dots, x_j \oplus k_j)$  donde  $g$  se obtiene de alguna discretización de una función no lineal con propiedades caóticas.

La función 5.1 se itera 5 veces y los últimos subbloques que se obtienen son los que componen el bloque cifrado  $B_5$ . Para visualizar dichas transformaciones se muestra un esquema en la figura 5.1.

Una de las principales diferencias entre el cifrado que se propone en este trabajo y el de Kocarev radica en la función  $f$ . En este cifrado, la función caótica que juega el papel de  $f$  es la función  $\mathcal{L}$  que se deriva del sistema caótico de Lorenz y se explica a detalle más adelante. En el cifrado de Kocarev los bloques  $B_i$  se componen de 64 bits (8 bytes) mientras que en el cifrado propuesto en este trabajo los bloques constan de 72 bytes, lo que es 9 veces más, esto se propuso así para procesar mayor cantidad de información en cada ronda. Para definir la función  $\mathcal{L}$  se definen antes algunas funciones necesarias para el cifrado y una versión discretizada del sistema de Lorenz.

Recordemos que el sistema de Lorenz es de la forma:

$$\begin{aligned}\dot{x} &= -\sigma x + \sigma y \\ \dot{y} &= -xz + rx - y \\ \dot{z} &= xy - bz\end{aligned}\tag{5.2}$$

el número de Rayleigh  $\sigma$ ,  $r$  y  $b$  son parámetros positivos del sistema. La versión de aproximación de la ecuación 5.2 se escribe:

$$\begin{aligned}\frac{x_{k+1} - x_k}{t_{k+1} - t_k} &= \sigma(x_k - y_k) \\ \frac{y_{k+1} - y_k}{t_{k+1} - t_k} &= -x_k z_k + r x_k - y_k \\ \frac{z_{k+1} - z_k}{t_{k+1} - t_k} &= x_k y_k - b z_k\end{aligned}\tag{5.3}$$

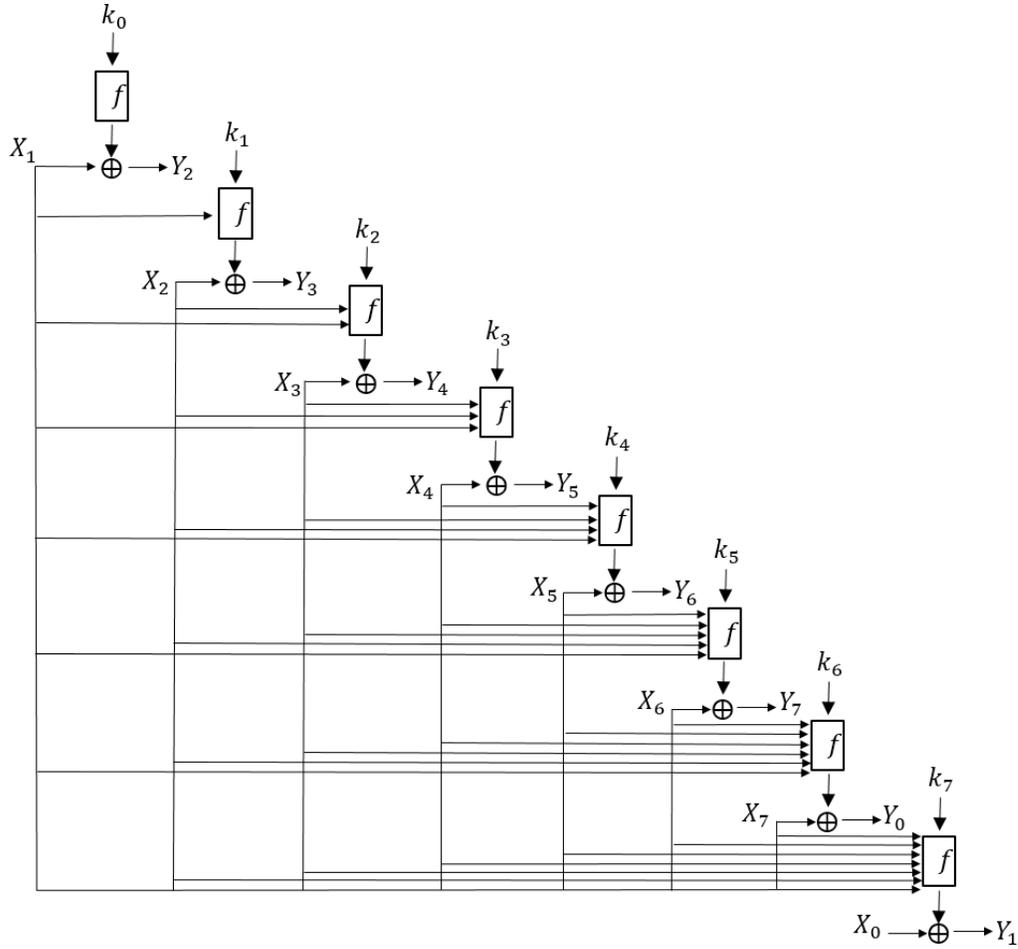


Figura 5.1: Esquema de las transformaciones del cifrado de Kocarev

La versión discreta o iterativa del sistema de Lorenz [27] se escribe:

$$\begin{aligned}
 x_{k+1} &= x_k + \sigma(x_k - y_k)(t_{k+1} - t_k) \\
 y_{k+1} &= y_k + (rx_k - x_k z_k - y_k)(t_{k+1} - t_k) \\
 z_{k+1} &= z_k + (x_k y_k - bz_k)(t_{k+1} - t_k)
 \end{aligned}
 \tag{5.4}$$

La evolución de la versión discreta de Lorenz se puede ver en la Figura 5.2.

**Una función Expansión**  $E : (\mathbb{Z}_{256})^3 \rightarrow \{0, 1, \dots, 6, 040\}^3$

$$E(w_1, w_2, w_3) = (8 * w_1 + 4, 000, 8 * w_2 + 4, 000, 8 * w_3 + 4, 000)$$

con  $0 \leq w_i \leq 255$ . La función lineal  $E$  tiene el fin de devolver una triada de números entre 4,000 y 6,040 que serán el número de iteraciones del modelo de Lorenz. Cabe mencionar que la función  $E$  es inyectiva y que no tiene impacto directo sobre la seguridad del cifrado, la utilidad de dicha función reside en expandir el número de iteraciones para evitar el ataque exhaustivo o de *fuerza bruta* ya que sin esta función las iteraciones se reducirían a 256, se eligió como tope de inicio 4,000 iteraciones porque es son suficientes para asegurar caoticidad en el sistema de Lorenz discreto.

**Una función Compresión**  $C : (\mathbb{Z}_{256})^9 \rightarrow (\mathbb{Z}_{256})^3$ .

$$C(x_0, \dots, x_8) = (x_0 + x_3 + x_6, x_1 + x_4 + x_7, x_2 + x_5 + x_8).$$

Esta función reduce al conjunto de tres pixeles concatenados a un sólo pixel, o una triada de elementos de  $\mathbb{Z}_{256}$ . Cabe mencionar que el orden en el que se concatenan los pixeles podría ser distinto pero se eligió así porque facilita su programación y la implementación. Será necesaria al realizar la composición  $E \circ C$ . Otro aspecto importante de esta función es que es lineal, en general las funciones lineales no son buenas para cifrar pero la seguridad del cifrado no recae en esta función, si no en la tabla de Lorenz.

**La función**  $\mathcal{L} : M = \{4,000, \dots, 6,040\}^3 \rightarrow (\mathbb{Z}_{256})^9$ .

La función  $\mathcal{L}$  tiene como entrada tres números  $(n_1, n_2, n_3)$ , con  $n_i \in M$ , dicho conjunto se elige así porque nos da un margen considerable para que el sistema de Lorenz sea caótico. Así  $M$  representa el número de iteraciones para el sistema de Lorenz discreto (Ec. 5.4), es decir, cada entero representa el número de iteraciones que “correrá” Lorenz con condiciones iniciales y parámetros fijos y el estado en el que se encuentre el sistema después de dichas iteraciones se considera como la salida de la función. Antes de definir la regla de correspondencia de  $\mathcal{L}$  hay que definir la tabla de Lorenz.

La Tabla de Lorenz se forma iterando la discretización del sistema de Lorenz (Ec. 5.4) usando parámetros y condiciones iniciales fijos que son parte de la llave del cifrado. Sea  $L : \{1, \dots, 6,040\} \rightarrow (\mathbb{Z}_{256})^3$ , para cada valor de  $i$  se almacena el valor  $L(i)$  en la Tabla 5.1.

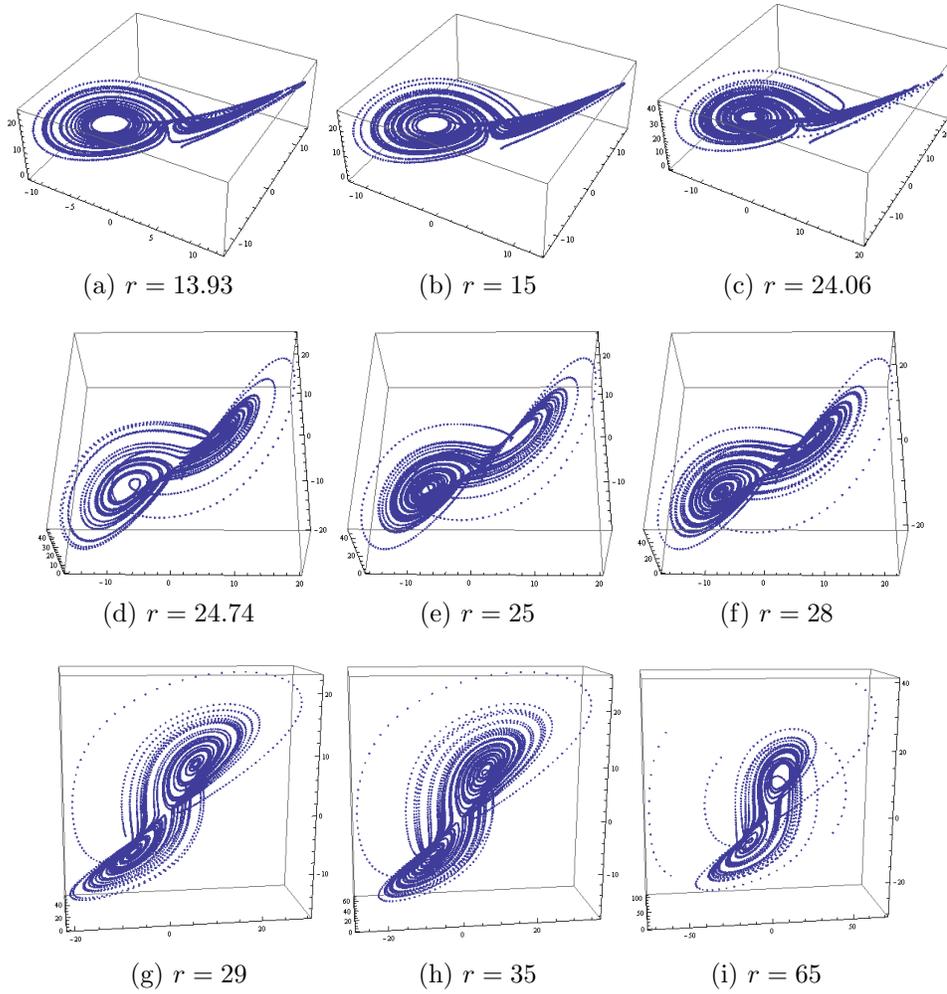


Figura 5.2: Atractor de Lorenz con  $\sigma = 10, b = 8/3$ , variando  $r$

Así, fijando  $(x_0, y_0, z_0, a, r, b, dt) \in \mathbb{R}^7$ ,  $\mathcal{L}$  se define:

$$\mathcal{L}(n_1, n_2, n_3) = (L(n_1), L(n_2), L(n_3)) \quad (5.5)$$

Podemos notar que  $(\lfloor 256 * x_i \pmod{1} \rfloor, \lfloor 256 * y_i \pmod{1} \rfloor, \lfloor 256 * z_i \pmod{1} \rfloor) \in (\mathbb{Z}_{256})^3$ . De este modo, la salida de  $\mathcal{L}$  puede verse como una concatenación de 3 pixeles, esto se requiere así porque dicha salida será sumada con algún subbloque  $X_i$ .

Iteración	$L(i)$
0	$(\lfloor 256 * x_0 \pmod{1} \rfloor, \lfloor 256 * y_0 \pmod{1} \rfloor, \lfloor 256 * z_0 \pmod{1} \rfloor)$
$\vdots$	$\vdots$
$i$	$(\lfloor 256 * x_i \pmod{1} \rfloor, \lfloor 256 * y_i \pmod{1} \rfloor, \lfloor 256 * z_i \pmod{1} \rfloor)$
$\vdots$	$\vdots$
6040	$(\lfloor 256 * x_{6040} \pmod{1} \rfloor, \lfloor 256 * y_{6040} \pmod{1} \rfloor, \lfloor 256 * z_{6040} \pmod{1} \rfloor)$

Tabla 5.1: Tabla de Lorenz

Ahora podemos definir formalmente la función  $f_j : M = \{4,000, \dots, 6,040\}^3 \rightarrow (\mathbb{Z}_{256})^9$ .

$$\begin{aligned} f_0 &:= \mathcal{L}(E(C(k_0)) \oplus C(k_0) \pmod{8}) \\ f_j &:= \mathcal{L}(E(C(X_1 + \dots + X_j + k_j)) \oplus C(X_j + k_j) \pmod{8}) \end{aligned} \quad (5.6)$$

donde  $k_j$  es parte de la llave utilizada en la ronda en que se encuentre el cifrado  $K = (k_0, k_1, \dots, k_7)$  por medio del algoritmo que se describe en la siguiente sección. Las sumas entre pixeles se realizan en módulo 256, la suma “ $\oplus$ ” se realiza entre elementos de  $\mathbb{Z}_{256}^3$  y la suma “+” entre elementos de  $(\mathbb{Z}_{256})^9$ . La operación  $(\pmod{8})$  afecta solamente a  $C(X_j + K_j)$  y su objetivo es variar el número de iteraciones que se usarán para  $\mathcal{L}$ .

Es importante mencionar que el cifrado propuesto es un cifrado por bloques simétrico similar al DES o al AES, (cifrado por rondas), y una de las diferencias importantes es la generación de las subllaves que se explica en la siguiente sección. Gran parte de la robustez del cifrado recae en la tabla de Lorenz por la caoticidad del sistema. Cabe mencionar que a pesar de que  $f$  no es inyectiva, es posible construir un algoritmo de descifrado. La confusión también se logra, ya que el cifrado de un bloque involucra los anteriores, esto ocasiona que al modificar un sólo pixel del texto o imagen en claro la mayoría de los pixeles del cifrado se modifiquen también, este análisis se lleva a cabo en el capítulo **Pruebas de robustez**, con la correlación de pixeles. Para probar la robustez del cifrado se realizaron pruebas de aleatoriedad y correlación que se abordan en el capítulo 7, **Seguridad del cifrado**. La subllave  $k_0$  puede elegirse por medio de generadores de números aleatorios [4], algunos buenos algoritmos de pueden encontrar en <https://www.nist.gov/itl>, las partes  $k_j$  con

$j = 1, \dots, 7$  se generan a partir de un algoritmo de generación de subllaves que se presenta en la siguiente sección.

## 5.1. Algoritmo para la generación de subllaves

Un papel primordial para la seguridad y robustez de un sistema criptográfico, sin duda lo juega la llave que se use para cifrar. Por esta razón, el algoritmo para generar las subllaves que se usarán en cada ronda debe cumplir ciertas características. Para el cifrado propuesto en esta tesis se consideraron algunos criterios que debe cumplir el algoritmo para generar las subllaves [9].

### 1. Eficiencia computacional

- a) **Espacio de memoria.** El algoritmo de generación de subllaves debe ejecutarse usando un espacio de memoria pequeño.
- b) **Ejecución.** El algoritmo debe poder ejecutarse en una alta gama de procesadores.

2. **Eliminación de Simetría.** Deben usarse varias rondas para eliminar la simetría.

3. **Difusión.** Debe tener una difusión eficiente en la generación de cada subllave.

4. **No linealidad.** El algoritmo debe involucrar no linealidad en la generación de la subllave de la ronda siguiente.

### Algoritmo generador de subllaves

Sea  $r$  el número de rondas del cifrado basado en caos. Sea  $K^1 = (k_0^1, k_1^1, \dots, k_7^1)$  la llave usada para la ronda 1 y la información inicial para el algoritmo. Cabe mencionar que  $K^1$  es parte de llave del cifrado y se obtiene con algún método de generación aleatoria. La llave  $K^j$ , usada para la ronda  $j$ -ésima se forma con el siguiente algoritmo:

$$k_0^j = \mathcal{L} (E(C(k_0^{j-1})) + C(k_0^{j-1}) \pmod{8})$$

$$k_i^j = k_i^{j-1} + k_{i-1}^j \pmod{256}$$

para  $i = 2, \dots, 7$ , y  $j = 1, \dots, r - 1$  representa el número de ronda.

La generación de las  $k_i^j$  de la subllave, para  $i = 2, 3, \dots, 7$  consta de una suma mod 256, la no linealidad del algoritmo se basa en que la parte  $k_0$  de cada subllave involucra a la función  $\mathcal{L}$  y esta a su vez involucra la función  $L : \{1, \dots, 6040\} \rightarrow (\mathbb{Z}_{256})^3$  que se especifica con la Tabla de Lorenz 5.1 y claramente es no lineal.

Si iteramos el algoritmo anterior, se genera la subllave que se usa en la segunda ronda

$$K^2 = (k_0^2, k_1^2, \dots, k_7^2)$$

Después la subllave para la tercera ronda

$$K^3 = (k_0^3, k_1^3, \dots, k_7^3)$$

y así sucesivamente las 7 subllaves para las  $r$  rondas del cifrado. Para el cifrado de este trabajo se crean 5 subllaves que constan cada una de 24 pixeles al igual que los bloques del cifrado.

El algoritmo de generación de subllaves cumple con las características mencionadas antes; es de rápida implementación computacional ya que el cálculo de las subllaves solo requiere la operación suma modular y una evaluación de la Tabla de Lorenz por cada ronda, que previamente se ha generado desde el inicio del cifrado. Además, el uso de la Tabla de Lorenz en el algoritmo le brinda no linealidad por estar basada en un sistema no lineal, y más aún, el hecho de usarse dicha tabla solamente en el primer elemento de cada subllave,  $k_0^j$ , le aporta asimetría al algoritmo, ya que podemos notar que los elementos siguientes de éste,  $(k_1^j, \dots, k_7^j)$ , se generan usando solo sumas de elementos de la subllave anterior y elementos calculados un paso anterior.

De este modo se tienen las propiedades necesarias para ser un algoritmo de generación de subllaves eficaz [9].

## 5.2. Etapas y esquema del cifrado de imágenes

El cifrado propuesto se compone básicamente por dos etapas principales que se explican a continuación.

### ■ Primera etapa

Sea  $l$  el número de píxeles de ancho de la imagen. Se calcula  $Nb = \lfloor \frac{l}{24} \rfloor$ . Dicha cantidad será el número de bloques a cifrar por cada renglón en esta etapa, siguiendo el orden de izquierda a derecha y de arriba hacia abajo hasta recorrer por completo la imagen. Los píxeles restantes de cifran en la segunda etapa. A los bloques correspondientes a esta etapa se les aplican las 5 rondas del proceso de cifrado que se explicó anteriormente. El resultado de esta etapa serán  $Nb$  bloques de píxeles cifrados en cada uno de los renglones de la imagen y una franja a la derecha que consta de  $l - Nb$  píxeles sin cifrar.

### ■ Segunda etapa

En la segunda etapa se cifra la imagen obtenida de la primera etapa, se cifrarán  $Nb$  bloques de píxeles de forma análoga a la etapa anterior pero invirtiendo el orden horizontal y el orden vertical. Es decir, se cifrará de abajo hacia arriba y de derecha a izquierda. El resultado obtenido en esta etapa son  $Nb$  bloques de píxeles cifrados junto con una franja de  $l - Nb$  píxeles sin modificación. Se debe observar que dicha franja no procesada por la segunda etapa, lo fue en la primera de la misma manera que la franja no procesada en la primer etapa lo será con la segunda y con esto logramos que cada uno de los píxeles de la imagen haya tenido alguna transformación en el proceso de cifrado. Las dos etapas del cifrado logran dos cosas: Se ahorra tiempo de ejecución al fijar el número de píxeles a cifrar desde un principio en lugar de calcular durante el cifrado, y con la segunda etapa aseguramos que todos los píxeles queden procesados por completo.

A continuación se presenta un esquema que ilustre el cifrado propuesto.

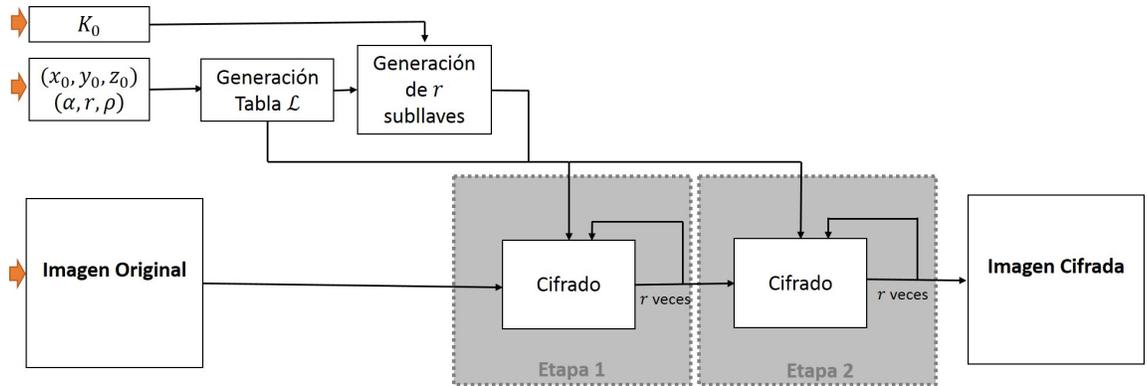


Figura 5.3: Esquema del cifrado de imágenes

### 5.3. Modos de operación

El cifrado propuesto es un cifrado por bloques es por eso que en esta sección introducimos varios modos populares de operación para cifrar por bloques ([22], Págs. 128-137). Dichos modos de operación logran dar mayor difusión al sistema de cifrado, ya que sin ellos, dos bloques en claro idénticos de la imagen o texto darían como salida bloques cifrados idénticos y esto puede dar información útil para un atacante del sistema de cifrado.

En esta sección se da una breve explicación de tres modos de operación comunes en cifrados por bloques incluyendo el que se utilizó para este trabajo de tesis.

#### Modo del libro de código electrónico. (Electronic Code Book mode ECB)

Este modo es la forma más sencilla de cifrar y la descripción formal del ECB es la siguiente:

**Definición 5.3.1.** Sea  $e_k()$  un cifrado por bloques de tamaño  $b$  usando la llave  $k$  y sea  $x_i$  y  $y_i$  cadenas de bits de longitud  $b$ .

Modo de cifrado:  $y_i = e_k(x_i)$ ,  $i \geq 1$

Modo de descifrado:  $x_i = e_k^{-1}(y_i) = e_k^{-1}(e_k(x_i))$ ,  $i \geq 1$

El principal problema con el ECB es que cifra de forma altamente determinista, es decir, dos bloques idénticos dan lugar a bloques de texto cifrado idéntico, siempre y cuando se use la misma llave para cifrar. De ahí el nombre del modo ya que el ECB puede verse como un enorme libro de código donde a cada entrada se le asigne una determinada salida. Esto tiene consecuencias indeseables obvias ya que si un atacante recibe dos veces el mismo mensaje, podría reconocer y deducir información de la llave y el mensaje mismo.

Otra desventaja importante es que con este modo dos bloques se cifran independientemente, el cifrado de un bloque no involucra al bloque anterior y el efecto de difusión puede disminuirse. Un modo que resuelve este problema es el siguiente.

### Modo de encadenamiento de bloques (Cipher Block Chaining Mode CBC)

El modo CBC tiene dos ventajas principales que lo hacen atractivo para el cifrado de este trabajo. La primera es que todos los bloques están de alguna manera “encadenados”, así que el bloque cifrado  $y_i$  no sólo depende del bloque  $x_i$  sino de todos los bloques sin cifrar anteriores. La segunda es el uso del vector de inicialización  $VI$ , este se añade al primer bloque en claro, así el primer bloque cifrado  $y_1$  depende del primer bloque en claro  $x_1$  y de  $VI$ , el segundo bloque cifrado  $y_2$  depende de  $VI$ ,  $x_1$  y  $x_2$  el tercer bloque  $y_3$  depende de  $VI$ ,  $x_1$ ,  $x_2$  y  $x_3$  y así sucesivamente. Añadir el vector de inicialización al cifrado no afecta de forma significativa el tiempo de ejecución ya que sólo involucra una suma  $XOR$ . El último bloque cifrado depende de  $VI$  y de todos los bloques anteriores lo que hace que la difusión obtenida en cada bloque se “sume” cada que se cifra uno nuevo. Además si elegimos un nuevo  $VI$  cada vez que ciframos el modo CBC se convierte en un algoritmo altamente probabilístico, dando como resultado mensajes cifrados completamente no relacionados entre sí a la vista de un atacante.

La descripción formal del CBC es la siguiente:

**Definición 5.3.2.** Sea  $e_k()$  un cifrado por bloques de tamaño  $b$  usando la llave  $k$  y sean  $x_i$  y  $y_i$  cadenas de bits de longitud  $b$  y sea  $IV$  el vector de inicialización elegido aleatoriamente (Ver generadores de números aleatorios [4]) y utilizado solamente una vez.

*Modo de cifrado primer bloque:*  $y_1 = e_k(x_1 \oplus VI)$

*Modo de cifrado general :*  $y_i = e_k(x_i \oplus y_{i-1}), \quad i \geq 2$

*Modo de descifrado primer bloque:*  $x_1 = e_k^{-1}(y_1) \oplus VI$

*Modo de descifrado general:*  $x_i = e_k^{-1}(y_i) \oplus y_{i-1}, \quad i \geq 2$

En el cifrado propuesto se utiliza el modo de operación CBC, debido a que aunque el uso del CBC no aumenta la seguridad del cifrado evita que bloques iguales tengan cifrados iguales, impidiendo que se otorgue información útil en un ataque. Además el uso del  $VI$  sirve para aumentar el espacio de llaves ya que este se conforma de 24 pixeles que serán parte de la llave  $K$  usada en la cifrado.

Con el fin de complementar la sección, se hará mención sobre un tercer modo de operación que también suele ser muy utilizado en los cifrados por bloques.

### **Modo de retroalimentación de salida (Output Feedback Mode OFB)**

El OFB es un modo de operación sumamente interesante ya que un cifrado por bloques es usado para construir una especie de cifrado en flujo. La idea del OFB es la siguiente: Comencemos con cifrar el  $VI$  con el cifrado por bloques, la salida del cifrado nos da la primera llave (secuencia de bits) para el cifrado. La siguiente llave se calcula simplemente cifrando la llave anterior, este proceso se repite como se muestra en la definición 5.3.3.

Lo interesante es que el OFB forma un cifrado en flujo que usa un cifrado por bloques para la elaboración de las llaves que no depende del mensaje en claro o cifrado.

El cifrado se realiza mediante la suma XOR y para el descifrado se utiliza simplemente otra suma XOR, a diferencia del modo ECB y del CBC donde se cifra y descifran los datos con el cifrado por bloques.

El esquema del OFB es el siguiente

**Definición 5.3.3.** *Sea  $e_k()$  un cifrado por bloques de tamaño  $b$  usando la llave  $k$  y sea  $x_i$  y  $y_i$  cadenas de bits de longitud  $b$  y sea  $VI$  una cadena de bit utilizada solamente una vez.*

*Modo de cifrado primer bloque:*  $s_1 = e_k(VI)$  y  $y_1 = s_1 \oplus x_1$

*Modo de cifrado general* :  $s_i = e_k(s_{i-1})$  y  $y_i = s_i \oplus x_i \quad i \geq 2$

*Modo de descifrado primer bloque*:  $s_1 = e_k(VI)$  y  $x_1 = s_1 \oplus y_1$

*Modo de descifrado general*:  $s_i = e_k(s_{i-1})$  y  $x_i = s_i \oplus y_i \quad i \geq 2$

## 5.4. Descripción del descifrado

La estructura del descifrado es muy parecida a la del cifrado, se aplican 5 rondas del sistema de descifrado a cada bloque cifrado para producir los bloques originales que forman la imagen o texto en claro.

Los bloques  $Y_0, Y_1, \dots, Y_7$  se obtiene aplicando un despeje a la ecuación 5.1, la transformación del descifrado es la siguiente:

$$X_k = Y_{k+1} \oplus f_{k-1}[X_1, \dots, X_{k-1}, K_{k-1}], \text{ con } k = 1, \dots, 8 \quad (5.7)$$

Los índices se toman mod 8, es decir  $Y_8 \equiv Y_0$  y  $Y_9 \equiv Y_1$ , y las funciones  $f_0, \dots, f_7$  tienen la forma ya definida en el cifrado. La función (5.7) se itera 5 veces y los últimos subbloques que se obtienen son los que componen el bloque descifrado  $B_0$ . La ventaja del cifrado propuesto es que como podemos notar, el algoritmo del descifrado es prácticamente el mismo que el del cifrado y así el programa que se usa para cifrar puede usarse para descifrar haciendo sólo pequeños cambios en el código.

Hay que considerar la parte del descifrado del modo de operación que se dió en la sección 5.3.

**Definición 5.4.1.** *Sea  $e_k()$  un cifrado por bloques de tamaño  $b$  usando la llave  $k$  y sea  $x_i$  y  $y_i$  cadenas de bits de longitud  $b$  y sea  $VI$  un número utilizado solamente una vez.*

*Modo de descifrado primer bloque*:  $x_1 = e_k^{-1}(y_1) \oplus VI$

*Modo de descifrado general*:  $x_i = e_k^{-1}(y_i) \oplus y_{i-1}, \quad i \geq 2$

*donde  $e_k^{-1}()$  es el descifrado que usa la llave  $k$ , y  $VI$  el vector de inicialización elegido aleatoriamente y utilizado solamente una vez.*

Además, las funciones  $f_j$  se definen igual que antes, la rondas del descifrado también se aplican como se vió en la sección 5.2. Y las subllaves de las rondas se aplican en orden descendente.

## 5.5. Cifrado de datos

De la misma manera en que se han cifrado imágenes, en el desarrollo presentado hasta ahora, es posible cifrar datos. Para esto cada caracter del texto en claro se representa como un número entre 0 y 255, que corresponde a su codificación en código ASCII. El texto en claro y el texto cifrado consistirán de 72 de estos números, los cuales se agruparán en bloques de tres enteros. De esta forma los espacios de texto en claro y texto cifrado serán ambos iguales al conjunto

$$((\mathbb{Z}_{256})^3)^8 \times ((\mathbb{Z}_{256})^3)^8 \times ((\mathbb{Z}_{256})^3)^8 .$$

Ahora podemos representar al mensaje como una imagen con un único renglón de bytes. El cifrado será como el utilizado para cifrar imágenes, sólo difiere de este en que no es necesario realizar un barrido de renglones.

Cada terna de enteros del texto cifrado que está entre 0 y 255, se concatena para formar un entero, que será menor que el número 256256256. El cifrado de una imagen resulta otra imagen, pero con la codificación empleada no podemos cifrar una cadena de caracteres como otra cadena de caracteres, pues en el código ASCII algunos enteros corresponden a caracteres no imprimibles, por lo tanto, el texto cifrado será el conjunto de números resultantes de la concatenación después de cifrar con el mismo algoritmo que se utilizó para cifrar imágenes (Ec. 5.1).

Para el cifrado de datos, cada bloque de 72 números entre 0 y 255 se agrupa de 24 subbloques de tres números que harán el papel de los 24 pixeles en el cifrado de imágenes y después se agrupan en conjuntos de 3 subbloques para formar los  $X_i$  del algoritmo.

Así, los 72 números del primer bloque son:

$$(x_{0,1}, x_{0,2}, x_{0,3}, x_{1,1}, x_{1,2}, x_{1,3}, \dots, x_{23,1}, x_{23,2}, x_{23,3})$$

La forma de concatenar los números es la misma que en el cifrado de imágenes:

$$X_0 = (x_{0,1}, x_{0,2}, x_{0,3}, x_{8,1}, x_{8,2}, x_{8,3}, x_{16,1}, x_{16,2}, x_{16,3}) = (n_0, n_8, n_{16})$$

para el siguiente bloque tenemos  $X_1 = (n_1, n_9, n_{17})$  y para el siguiente tenemos  $X_2 = (n_2, n_{10}, n_{18})$ . En general tenemos que las partes de cada bloque

se ven de la forma:

$$X_i = (n_i, n_{i+8}, n_{i+16}), \text{ donde } 0 \leq i \leq 7$$

Después se procede a cifrar usando la Tabla de Lorenz y la llave elegida de acuerdo al algoritmo para la generación de subllaves (Sección. 5.1).

Por ejemplo, el texto en claro <<MAFALDA MEDITA MIENTRAS ODI LA SOPA>> tiene la representación numérica:

[[77, 65, 70],	[65, 76, 68],	[65, 32, 77],	[69, 68, 73],	[84, 65, 32],
[77, 73, 69],	[78, 84, 82],	[65, 83, 32],	[79, 68, 73],	[65, 32, 76],
[65, 32, 83],	[79, 80, 65],	[32, 32, 32],	[32, 32, 32],	[32, 32, 32]
[32, 32, 32],	[32, 32, 32],	[32, 32, 32],	[32, 32, 32],	[32, 32, 32],
[32, 32, 32],	[32, 32, 32],	[32, 32, 32],	[32, 32, 32],	[32, 32, 32]]

Una vez codificado el mensaje se le han concatenado de uno a dos espacios en blanco, de forma que el número de caracteres resultante sea múltiplo de tres. A este proceso de relleno se le conoce como *padding*. Como el número de caracteres es menor que 72 se concatenan suficientes espacios en blanco para tener este número, pues 72 será el número de caracteres por bloque a cifrar. Cuando el número de caracteres es mayor o igual a 72 no es necesario agregar más espacios en blanco. En este ejemplo se tiene un único bloque a cifrar y no existen caracteres sin cifrar en la primera ronda. Al cifrar con la misma llave utilizada para cifrar las imágenes resulta el siguiente texto cifrado:

239151107	141178214	242154159	167084038	113222011	232136179
28039190	199217026	210120206	216104181	206217227	45078149
94235221	179051235	145124079	90190111	72208003	238180138
50126246	249132051	115168047	77158235	92244232	179115234

donde cada número representa una terna, en particular 239151107 representa a la terna [239,151,107]. Se hizo una prueba en una lap top Toshiba Satellite con un procesador AMD A6-4400M a 2.7 GHz y se obtuvieron los siguientes tiempos de ejecución.

Tiempo Cifrado 0.1074070930 seg

Tiempo Descifrado: 0.2338280678 seg

## 74CAPÍTULO 5. CIFRADO UTILIZANDO EL MODELO CAÓTICO DE LORENZ

El siguiente texto puede encontrarse en el libro <<Alicia en el país de las maravillas>>, escrito por Lewis Carrol, el cual puede obtenerse en la siguiente dirección:

<http://www.fundacionunam.org.mx/unam-al-dia/unam-te-ofrece-200-libros-para-descargar-gratis/>

--Me gustaría saber cuántas millas he descendido ya  
--dijo en voz alta--. Tengo que estar bastante cerca de el centro  
de la tierra. Veamos: creo que está a cuatro mil millas de  
profundidad...

Como veis, Alicia había aprendido algunas cosas de éstas en las  
clases  
de la escuela, y aunque no era un momento muy oportuno para presumir  
de sus conocimientos, ya que no había nadie allí que pudiera escucharla,  
le pareció que repetirlo le servía de repaso.

--Sí, está debe de ser la distancia... pero me pregunto a qué latitud  
o longitud habré llegado.

Alicia no tenía la menor idea de lo que era la latitud,  
ni tampoco la longitud, pero le pareció bien decir unas palabras  
tan  
bonitas e impresionantes.

Los tiempos para generar la tabla de iteraciones del modelo de Lorenz y  
de cifrado son los siguientes:

Tiempo Tabla: 0.0200519562 seg

Tiempo Cifrado 0.1389482021 seg

Para la decodificación el tiempo respectivo es:

Tiempo Descifrado: 0.4441480637 seg

El mensaje es entregado al receptor el cual cuenta con la llave para poder  
generar la Tabla de Lorenz y descifrar el mensaje para luego decodificarlo y  
convertirlo a sus respectivos caracteres. Si se cifra este texto con la misma  
llave que la empleada para el cifrado de imágenes obtenemos el texto cifrado  
que se muestra a continuación:

102232155	45071030	49109150	107014096	241095143	54215100
224088254	183171106	177224042	81059179	146222110	178116048
117195158	121044189	81206110	163167111	163179190	112074058
248193030	75185157	8011162	119237049	164162073	19169030
239012174	99048247	141134215	77091231	32190173	182039176
139166152	123180039	159020068	25102066	40131158	10174068
247159086	37242232	120029163	113191039	173202092	229084201
32104156	190158108	34212193	12165193	204046129	33215127
138003220	163105142	94154043	191074147	61171130	185239178
232171189	3054097	24204156	251093099	177144251	83205132
182078228	100009017	2099200	204191175	170010159	202086049
184046159	122145219	71166228	186033027	20125046	29235229
181127149	42235223	21186227	29232185	155194201	36199066
236054044	91228118	8181053	45127203	59139149	19241121
148162051	103224103	91197203	41124185	181175208	68039023
125160029	228087028	236036243	1146179	176055018	6183001
81069208	81137095	135221158	131065123	97042179	134193082
113205128	227017238	72096002	222094180	195109044	17086143
160087030	13217190	141091054	142036241	5137172	185051016
167151119	59129040	176156052	221062132	93017241	30243116
134108166	217194009	92205107	86121059	246123091	115124211
55096005	121169216	78237127	184069175	54176163	16208209
89101239	236157025	236072218	68098060	180129150	123181171
67188164	243002000	95184028	9174061	80219042	46041069
139100029	217194037	207097039	111026121	145165194	95109219
225148128	133221097	160094198	216161197	110191049	132082135
82220137	195249128	162138064	38037000	194133173	79253143
72031117	133050071	252098140	230219037	51158065	219002096
46202157	221089238	144079250	14248005	220095061	186016152
211220082	41228013	36122146	69054138	34071160	96069142
82065119	174152115	103159236	234039185	217021188	55178193
80090	185192010	44009115	122246235	168201147	43172056
73005228	155079250	154221125	207039159	98009187	25036087
72212057	46239163	169111246	19247238	222170049	230156049
16131118	68035226	50245038	92056246	42036214	108032103
60163162	158226238	72178096	192146161	164197008	198196071
35232052	55236204	52045004	131204095	178181174	20047019
208054062	94173197	194146112	37188163	87081034	152001202
224059180	191129081	241253040	6240225	59099198	105117140
206232095	245203181	248118194	111170128		

## 5.6. Descifrado de datos

Al cifrar texto lo que obtenemos son números concatenados que representan ternas, recordemos el ejemplo de la sección anterior, el número 239151107 representa a la terna [239,151,107]. De este modo, para proceder con el descifrado se debe separar los números del texto cifrado para obtener las ternas originales, luego dichas ternas juegan el papel de pixeles cifrados y se procede a descifrar como si fuera una imagen con el algoritmo 5.7 que vimos en la sección **Descripción del descifrado**. Se considera la llave que se utilizó para cifrar y se obtienen los números que representan el mensaje en claro.

## 5.7. Pseudocódigos

En esta sección se presentan los pseudocódigos del cifrado y descifrado de datos e imágenes junto con los resultados de cifrar cuatro imágenes distintas.

Algoritmo Lorenz

**Entrada:**  $x_0, y_0, z_0, a, r, \beta, dt$  reales,  $ite$  natural

**Salida:** un elemento de  $\mathbb{Z}_{256}^3$

**Para**  $n = 1$  **hasta**  $ite$

$$x_n = x_{n-1} + a(y_{n-1} - x_{n-1})dt$$

$$y_n = y_{n-1} + (rx_{n-1} - y_{n-1} - x_{n-1}z_{n-1})dt$$

$$z_n = z_{n-1} + (x_{n-1}y_{n-1} - \beta z_{n-1})dt$$

**Salida:**  $(\lfloor 256(x_{ite} \bmod 1) \rfloor, \lfloor 256(y_{ite} \bmod 1) \rfloor, \lfloor 256(z_{ite} \bmod 1) \rfloor)$

Denotamos como  $Lorenz(x_0, y_0, z_0, a, r, \beta, dt, ite)$  la salida del Algoritmo Lorenz. El algoritmo de cifrado, tanto de imágenes como de datos, fija los parámetros en el modelo de Lorenz de en los siguientes valores:

$$a = 10, r = 28, \beta = 8/3, dt = 0.005$$

$$sumpix : (\mathbb{Z}^3)^2 \rightarrow \mathbb{Z}^3, \quad sumpix((a, b, c), (d, e, f)) = (a + d, b + e, c + f)$$

$$sumpixm : (\mathbb{Z}_{256}^3)^2 \rightarrow \mathbb{Z}_{256}^3, \quad sumpixm(\mathbf{u}, \mathbf{v}) = \mathbf{u} + \mathbf{v}$$

$$SUMPIX : (\mathbb{Z}_{256}^8)^2 \rightarrow \mathbb{Z}_{256}^8, \quad SUMPIX(\mathbf{u}, \mathbf{v}) = \mathbf{u} + \mathbf{v}$$

$$prpix : \mathbb{Z}_{256}^3 \times \mathbb{N} \rightarrow \mathbb{Z}_{256}^3, \quad prpix(\mathbf{u}, n) = n\mathbf{u}$$

$$sumpixE : \mathbb{Z}_{256}^3 \times \mathbb{N} \rightarrow \mathbb{Z}_{256}^3, \quad sumpixE(\mathbf{u}, n) = \mathbf{u} + (n, n, n)$$

$$\begin{aligned} \text{comprime} : (\mathbb{Z}_{256}^3)^3 &\rightarrow \mathbb{Z}_{256}^3, & \text{comprime}(\mathbf{u}, \mathbf{v}, \mathbf{w}) &= \mathbf{u} + \mathbf{v} + \mathbf{w} \\ C : (\mathbb{Z}_{256}^3)^3 &\rightarrow \mathbb{Z}_{256}^3, & C(\mathbf{u}, \mathbf{v}, \mathbf{w}) &= \mathbf{u} + \mathbf{v} + \mathbf{w} \\ N : \mathbb{Z}_{256} &\rightarrow \{4000, 4001, \dots, 6048\}, & N(x) &= 8(x \bmod 8) + 4000 \end{aligned}$$

**Algoritmo Generación de subllaves****Entrada:**  $K$  llave de 24 pixeles #  $K \in (\mathbb{Z}_{256}^3)^{24}$ **Salida:** La imagen cifrada

# Generación de subllaves

 $R = 5$  #Número de rondas $K_0 = K = [k_0^{(0)}, k_1^{(0)}, \dots, k_7^{(0)}]$  #Llave para la ronda 0**Para**  $r = 0$  **hasta**  $R - 1$ 

$$k_0^{(r+1)} = TL[N(C(k_0^{(r)} + C(k_0^{(r)})) \bmod 8)]$$

**Para**  $j = 1$  **hasta** 7

$$k_j^{(r+1)} = k_j^{(r)} + k_{j-1}^{(r)}$$

$$K_{r+1} = [k_0^{(r+1)}, k_1^{(r+1)}, \dots, k_7^{(r+1)}] \text{ #Genera llave de la ronda } r+1$$

**Algoritmo Cifrado Imagen****Entrada:**  $x_0, y_0, z_0$  reales, una imagen de tamaño  $XX \times YY$ **Salida:** La imagen cifrada $R = 5$  #Número de rondas

#Generación de la Tabla de Lorenz

TL={Lorenz( $x_0, y_0, z_0, a, r, \beta, dt, ite$ ):  $4000 \leq ite \leq 6040$ }

VI = 0 # VI de 72 bits

#Num. de pixeles a cifrar por renglón

$$Nb = \lfloor XX/24 \rfloor$$

#Comienza cifrado\*\*\*\*\*

**Para**  $r = 0$  **hasta**  $R - 1$ **Para** etapa = 1 **hasta** 2**Si** etapa=2,

$$\text{pixel}(XX - 1 - i, YY - 1 - j) \leftarrow \text{pixel}(i, j), \text{ para cada } i$$

y cada  $j$ **Para**  $y = 1$  **hasta**  $YY$ ,**Para**  $\ell = 1$  **hasta**  $Nb$ 

$$B = [X_0, X_1, \dots, X_7] \text{ #bloque a cifrar}$$

$$B \leftarrow B + VI \text{ #Suma de bloque a cifrar con VI}$$

$$Y_2 = X_1 + TL \left( N \left[ C(k_0^{(r+1)}) + C(k_0^{(r+1)}) \bmod 8 \right] \right)$$

78CAPÍTULO 5. CIFRADO UTILIZANDO EL MODELO CAÓTICO DE LORENZ

**Para**  $i = (3, 4, 5, 6, 7, 0, 1)$ mód 8 #índices se toman mod 8

$$aux1 = C(X_1 + X_2 + \dots + X_{i-2} + k_{i-2}^{(r+1)})$$

$$aux2 = C(X_{i-2} + k_{i-2}^{(r+1)})\text{mód } 8$$

$$Y_i = X_{i-1} + TL(N[aux1 + aux2])$$

#Actualiza Xn

**Para**  $n = 0$  hasta 7

$$X_n \leftarrow Y_n$$

$$VI = X_n$$

Muestra imagen de la ronda R

#Termina cifrado\*\*\*\*\*

**Algoritmo Descifrado**

**Para**  $r = R$  hasta 1

Llave = Kr

**Para** etapa= 1 hasta 2

**Si** etapa=2,

$pixel(XX - 1 - i, YY - 1 - j) \leftarrow pixel(i, j)$ , para cada  $i$  y cada  $j$

da  $j$

**Para**  $y = 1$  hasta  $YY$ , #Renglones de la imagen

**Para**  $l = 1$  hasta  $Nb$

$$VI = Y_n$$

$$X_1 = Y_2 + TL\left[N\left(C(k_0^{(R)}) + C(k_0^{(R)})\text{mód } 8\right)\right]$$

**Para**  $i = (2, 3, 4, 5, 6, 7, 0)$ mód 8 #índices se toman mod 8

man mod 8

$$aux1 = C(Y_1 + Y_2 + \dots + Y_{i-2} + k_{i-2}^{(R)})$$

$$aux2 = C(Y_{i-2} + k_{i-2}^{(R)})\text{mód } 8$$

$$X_i = Y_{i-1} + TL[N(aux1 + aux2)]$$

#Actualiza Yn

**Para**  $n = 0$  hasta 7

$$Y_n \leftarrow X_n$$

Muestra imagen de la ronda R

#Termina descifrado\*\*\*\*\*

### 5.7.1. Cifrado de datos

Para el cifrado de datos, el código es el mismo considerando que antes de cifrar debemos modificar el texto de forma que se pueda representar como una imagen. El pseudocódigo es el siguiente:

#### Algoritmo Cifrado de datos

```
#Conversión de datos*****
mensaje = (Mafalda.txt)
Si longitud(mensaje) mód 3  $\neq$  0
    Hacer padding con 32
#Relleno hasta 72 caracteres
Si longitud(mensaje) < 72
    Hacer padding con 32
Para  $i = 1$  hasta  $l = \text{longitud}(\text{mensaje})/3$ 
    #Carga de caracteres de 3 en 3
    pix[i] = mensaje( $3_i$ ) hasta mensaje( $3_{i+1}$ )
XX = longitud(pix)
YY = 1
#Comienza cifrado
#Termina Conversión*****
```

### 5.7.2. Descifrado de datos

Al igual que en el cifrado de datos, el algoritmo de imagenes nos sirve pero antes de descifrar se necesita separar las ternas concatenadas de números en algo que pueda verse como un pixel, es decir elementos de  $(\mathbb{Z}_{256})^3$ . El espacio de textos en claro, y el espacio de textos cifrados ambos son el conjunto

$$\mathcal{P} = \mathcal{C} = (\mathbb{Z}_{256}^3)^8 \times (\mathbb{Z}_{256}^3)^8 \times (\mathbb{Z}_{256}^3)^8$$

El espacio de llaves es  $(\mathbb{Z}_{256}^9)^8 \times \mathbb{R}^3$

#### Algoritmo Descifrado de datos

```
#Conversión de datos*****
mensaje = (Mafalda cifrado.txt)
longitud(mensaje) = XX
#Carga ternas concatenadas
mensaje = [ $a_1, a_2, \dots$ ]
Para  $i = 1$  hasta XX
```

80 *CAPÍTULO 5. CIFRADO UTILIZANDO EL MODELO CAÓTICO DE LORENZ*

```
 $x_1 = a_i \bmod 1,000,000$  #Separa el primer pixel  
 $a_i = (a_i - x_1)/1,000$   
 $x_2 = a_i \bmod 1,000$  #Separa el segundo pixel  
 $a_i = (a_i - x_2)/1,000$   
 $x_3 = a_i$   
pix = ( $x_1, x_1, x_3$ )  
Agrega pix a pix  
#Termina Conversión*****
```

Los resultados de cifrar cuatro imágenes: Lena, tablero de ajedrez, libélula y cuadro negro se pueden observar en la Figura 5.4.

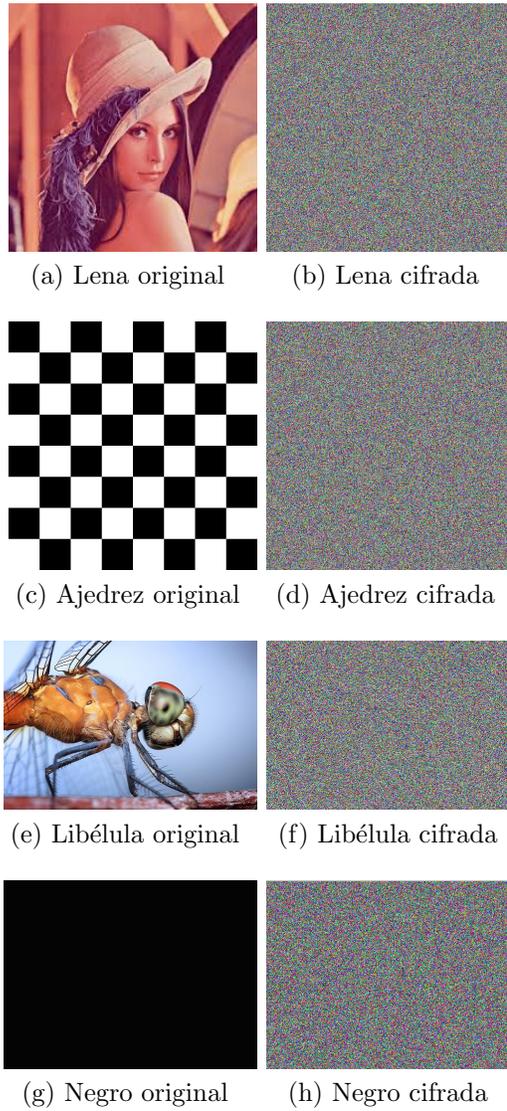


Figura 5.4: Imágenes originales y cifradas con el algoritmo propuesto



# Capítulo 6

## Criptoanálisis del cifrado

Como vimos en el Capítulo 4.2, la criptología se compone de dos disciplinas de naturaleza opuesta pero complementarias entre sí: la criptografía y el criptoanálisis. La Criptografía como se mencionó previamente, se encarga de las técnicas y procedimientos para cifrar determinada información que se desea tener en secreto o confidencia. El criptoanálisis, por su parte, se ocupa de buscar debilidades en dichos algoritmos de cifrado para recuperar ya sea la llave utilizada para el cifrado o parte de ella con el fin de conocer la información confidencial original. A pesar de lo opuestas que son, éstas disciplinas han sido desarrolladas de forma paralela a través de la historia ya que cualquier método de cifrado que se considere robusto debe tener en cuenta los posibles ataques que le puedan hacer, con el fin de evitar que sea “roto” o se pueda descubrir la información original cifrada.

Existen varios tipos de ataques a un cifrado, en este trabajo se analiza el criptoanálisis diferencial. A su vez se realizaron algunas pruebas probabilistas para mostrar que el cifrado propuesto es resistente a dicho ataque. Para ver con más detalle lo relacionado al criptoanálisis diferencial consultar [6].

### 6.1. Criptoanálisis Diferencial

Para generalizar la siguiente explicación, nos referiremos a un mensaje en claro, sea imagen o texto, como texto en claro, ya que como se mencionó antes un texto puede verse como imagen si representamos cada pixel como un elemento de  $(\mathbb{Z}_{256})^3$ .

El criptoanálisis diferencial es un ataque de texto en claro elegido, es decir, el atacante puede elegir un texto en claro y obtener su cifrado correspondiente, la idea entonces es que el atacante cuente con una gran cantidad de parejas de textos en claro y sus textos cifrados correspondientes [15].

La idea principal del ataque es analizar la relación entre la diferencia de un par de textos en claro y la diferencia de las salidas de rondas sucesivas en un cifrado de rondas iterativas.

Supongamos que  $X$  y  $X^*$  son dos textos en claro distintos y que  $Y$  y  $Y^*$  son sus textos cifrados correspondientes usando una llave  $K$  desconocida por el atacante. Sea  $X' = X \oplus X^*$  la diferencia módulo 256 entre los textos en claro,  $Y' = Y \oplus Y^*$  entre los textos cifrados y  $\oplus$  denota la suma XOR. Ahora denotemos como el **diferencial de la  $i$ -ésima ronda** como la pareja  $(X', Y')$  de la  $i$ -ésima ronda.

**La probabilidad del diferencial de una  $i$ -ésima ronda  $(X', Y')$**  es la probabilidad condicional de que  $Y'$  sea la diferencia  $\Delta Y_i$  del par de textos cifrados después de  $i$  rondas, dado que, el par de textos en claro tengan diferencia  $\Delta X_i = X'$  tomando en cuenta que los textos en claro y las subllaves usadas en las rondas del cifrado son independientes y tienen una distribución uniforme.

El procedimiento de un ataque diferencial a un cifrado iterativo que consta de  $r$  rondas se puede resumir en los siguientes pasos:

1. Encontrar un diferencial de la ronda  $(r - 1)$  tal que su probabilidad de ocurrir sea máxima o casi máxima, cabe señalar que esto no suele ser una tarea fácil.
2. Se eligen dos textos en claro  $X$  y  $X^*$  de tal forma que la diferencia  $\Delta X_i$  sea  $X'$  elegido previamente. Posteriormente se cifran  $X$  y  $X^*$  bajo la llave actual. De los textos cifrados resultantes  $Y$  y  $Y^*$  encontrar cada posible valor (si existe) de la subllave correspondiente a la última ronda,  $Z_r$ , con ayuda del anticipado valor  $Y'$ . Por último incrementar a uno el contador del número de apariciones de cada valor de la subllave de la última ronda.

3. Repetir los pasos 1 y 2 hasta que el contador dé algunos valores de  $Z_r$  sean significativamente más grandes que otros. Tomar este valor  $Z_r$  o el pequeño conjunto de  $Z_r$  con el contador más alto como candidatos para que el criptoanalista elija el valor de la subllave real.

Generalmente el paso más difícil del procedimiento del ataque diferencial es el primero, puede darse el caso en que encontrar el diferencial de la ronda  $(r - 1)$  con probabilidad alta sea muy difícil o imposible. Además, cuando el atacante busca diferenciales de rondas con probabilidades altas echa mano de debilidades de las transformaciones no lineales usadas en el algoritmo de cifrado. Esto debe tenerse en cuenta a la hora de proponer un cifrado y estudiar su fortaleza, las funciones lineales se emplean por su fácil y rápida implementación pero la robustez de un cifrado recae en la parte no lineal. Así, es necesario que un cifrado contenga una parte no lineal para resistir los ataques, aunque esto no garantiza por si solo la robustez del cifrado.

**La probabilidad de aproximación diferencial** de una función correspondiente a un cifrado,  $(DP_f)$ , es una medida de uniformidad diferencial y se define como sigue.

$$DP_f = \max_{\Delta x \neq 0, \Delta y} \left( \frac{\#\{x \in X | f(x) \oplus f(x \oplus \Delta x) = \Delta y\}}{2^n} \right) \quad (6.1)$$

$X$  es el conjunto de todos los posibles textos en claro,  $2^n$  es el número de sus elementos y  $f$  representa a la transformación del cifrado. Así,  $DP_f$  es la probabilidad máxima de tener diferencia de textos cifrados igual a  $\Delta y$  cuando la diferencia de sus correspondientes textos en claro es  $\Delta x$ .

Si deseamos analizar el cifrado propuesto, nos enfocamos en las funciones  $f_j$  de nuestro cifrado. Recordemos como se define la función  $\mathcal{L} : \{4,000, \dots, 6,040\}^3 \rightarrow \mathbb{Z}_{256}$  se define:

$$\mathcal{L}(n_1, n_2, n_3) = (T(n_1), T(n_2), T(n_3)) \quad (6.2)$$

Los valores de  $T(n_i)$  se obtienen con la Tabla de Lorenz (Ver Tabla 5.1).

Más aún, como  $\mathcal{L}$  es básicamente la función  $T(i)$  aplicada tres veces, enfocamos el análisis en dicha función. Además,  $T$  es la parte no lineal del

algoritmo y donde reside la seguridad del mismo. Si analizamos las transformaciones que sufre un sub-bloque de entrada de 3 pixeles concatenados, es decir un elemento  $X_i$  el procedimiento se puede representar por el esquema de la Figura 6.1. Podemos notar que hasta la función de la tabla de Lorenz,  $X_i$  sufre sólo transformaciones lineales; la función compresión y la función expansión son transformaciones lineales. La función compresión realiza la suma de sus elementos para transformar un bloque de nueve elementos a uno de tres, y ese nuevo bloque se va modificando por separado en sus tres elementos.

Así para simplificar el criptoanálisis y determinar que tomaremos como función  $f$ , veamos lo que le hace la función  $f_j$  a los bloques  $X_i$  del cifrado.

$$f_j(X_i) := \mathcal{L}(N(C(X_1 + \dots + X_j + K_j)) \oplus C(X_j + K_j) \bmod 8)$$

Si tomamos como el conjunto  $X = \mathbb{Z}_{256}^9$ , el número de elementos de este conjunto es  $256^9$ , y los textos en claro para el análisis serían los posibles *sub-bloques*  $X_i$ , la suma y resta en la fórmula deben ser módulo 256, y la probabilidad de aproximación diferencial del cifrado queda como sigue:

$$DP_f = \max_{\Delta x \neq 0, \Delta y} \left( \frac{\#\{x \in X | f(x) \oplus f(x \oplus \Delta x) = \Delta y\}}{256^9} \right) \quad (6.3)$$

donde  $f$  en este caso es la función  $f_j$  y el análisis de la transformación de un *sub-bloque* sería como sigue (Ver figura 6.1).

El problema con este análisis es que el equipo de cómputo con el que se cuenta no es capaz de calcular la probabilidad diferencial para cada  $\Delta X_i \in (\mathbb{Z}_{256})^9$ , ya que sería necesario calcular para cada  $X \in (\mathbb{Z}_{256})^9$ , la diferencia  $\Delta Y = f(X) \oplus f(X \oplus \Delta X)$  y compararlas entre ellas para incrementar el contador del paso 3 del ataque cada que se encuentren coincidencias. Sin embargo podemos realizar un criptoanálisis con mucho menos cálculos restringiendo  $f$  a una función más sencilla.

Para resolver este problema se tomó como conjunto  $X$  a  $\mathbb{Z}_{256}$  y como posibles entradas de la función  $f$  los posibles  $C_i \in \mathbb{Z}_{256}$  del esquema 6.1, y la función  $f$  se reduce a:

$$\mathcal{L}(N(C_i) \oplus C(K_i))$$

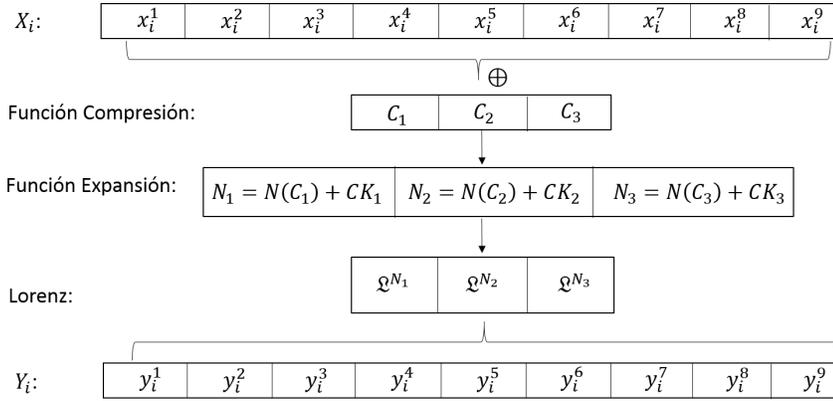


Figura 6.1: Transformaciones de un sub-bloque del cifrado de imágenes

Entonces calcular la probabilidad diferencial es posible y queda de la forma:

$$DP_f = \max_{\Delta x \neq 0, \Delta y} \left( \frac{\#\{x \in X | f(x) \ominus f(x \oplus \Delta x) = \Delta y\}}{256} \right) \quad (6.4)$$

El pseudocódigo para calcular  $DP_f$  es el siguiente:

**Algoritmo**  $DP_f$

**Define** Tabla de Lorenz  $T[i] = (x, y, z)$

$Z = z_i$

Lorenz[ $x_i$ ] :=  $T[N(x_i) + C(k_i)]$

$D_y = 0$

**Para**  $dx = 1$  hasta 256

$c = 0$

**Para**  $x = 1$  hasta 256

$c = c + 1$

$f_x = \text{Lorenz}[x]$

suma =  $(x + dx)$  mód 256

$f_{\text{suma}} = \text{Lorenz}[\text{suma}]$

$d_y = f_x - f_{\text{suma}}$

**Agrega**  $d_y$  a  $D_y$

cont = 0

**Para**  $i = 1$  hasta 256

**Para**  $j = 1$  hasta 256

**Si**  $D_y[i] = D_y[j]$   
 $\text{cont}[i] = \text{cont}[i] + 1$   
 $M[d_x] = \mathbf{m\acute{a}x}(\text{cont})$   
**Imprime** El maximo es:  $M[d_x]$

Se programo el codigo anterior en Python y cada vez que se probo se obtuvo el valor de  $DP_f = 0$ . Esto se explica de forma simple. Recordemos la forma en la que se define  $f$ . La entrada de esta funcion es un elemento de  $\mathbb{Z}_{256}$  mientras que la salida,  $f(x)$ , es un elemento de  $\mathbb{Z}_{256}^3$ , ası, para cada posible  $\Delta y \in \mathbb{Z}_{256}^3$  el numero de elementos  $x \in \mathbb{Z}_{256}$  tales que  $f(x) - f(x + \Delta x) = \Delta y$ , dado  $\Delta x$  es igual a cero.

Otras pruebas de robustez que se realizaron para la seguridad del cifrado propuesto son las pruebas de analisis estadıstico que se presentan en el siguiente capıtulo.

# Capítulo 7

## Robustez del cifrado de imágenes y datos

En este capítulo se abordan los resultados de algunas pruebas de robustez en el cifrado de imágenes. El análisis de los histogramas de los tres colores (RGB) de la imagen, su comparativo con el histograma de la imagen original, la sensibilidad de la llave al cifrar y descifrar y cálculo de la correlación entre píxeles adyacentes en la imagen cifrada son algunas de las pruebas que se realizaron a las imágenes resultantes del cifrado. Los resultados obtenidos en las pruebas nos dan evidencia de la seguridad del cifrado propuesto.

### 7.1. Análisis estadístico

Existen varios aspectos a considerar cuando se pretende realizar criptoanálisis a un cifrado en específico. Hay diferentes técnicas y herramientas que de acuerdo al algoritmo del cifrado, son útiles para obtener información sobre la llave que se utilizó para cifrar. Una técnica de gran importancia por su utilidad en una gran parte de sistemas criptográficos, es el análisis estadístico. Dentro del análisis estadístico nos interesan para este trabajo dos pruebas de robustez que son de importancia en el cifrado de imágenes: Los histogramas y la correlación de píxeles adyacentes. En las secciones siguientes se hacen las pruebas de histogramas de color (RGB) y correlación de píxeles adyacentes horizontal y vertical.

### 7.1.1. Histogramas

Un histograma de color (RGB) es una superposición de tres histogramas distintos que corresponden a cada uno de los canales que forman una imagen. A su vez, cada histograma representa la probabilidad de aparición de las distintas tonalidades de color en cada canal. Se espera que exista un alto contraste entre el histograma de la imagen original y la imagen cifrada correspondiente.

Se espera también que el histograma de la imagen cifrada tenga una distribución más uniforme a diferencia de la original. Para verificar esta propiedad se utilizaron cuatro imágenes para los resultados experimentales: La imagen estándar de Lena, la imagen de un tablero de ajedrez, la imagen de un insecto y un cuadro completamente en negro.

De la figura 7.1 podemos notar que la distribución de los colores de la imagen cifrada es más uniforme, más parecida a pixeles distribuidos aleatoriamente. Se eligieron imágenes distintas para hacer notar que la eficacia del cifrado no radica en los colores o distribución de los mismos en la imagen. De estos resultados podemos constatar que la función basada en el sistema de Lorenz tiene un efecto eficiente en la seguridad del cifrado.

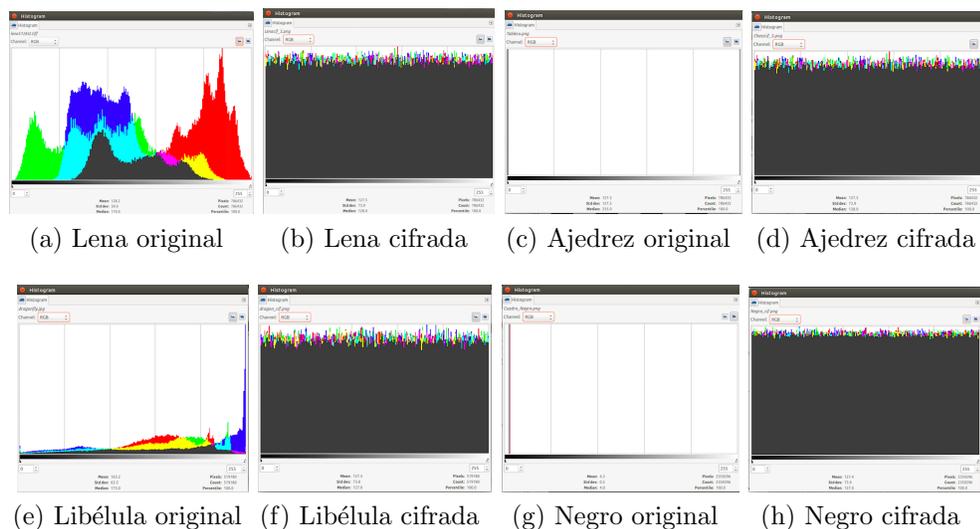


Figura 7.1: Histogramas

### 7.1.2. Correlación de píxeles

Generalmente si elegimos un píxel al azar en una imagen, dicho píxel está fuertemente correlacionado con píxeles adyacentes (vertical, horizontal y diagonal). Cuando una imagen es cifrada con un algoritmo fuerte, se espera que ésta tenga baja correlación entre píxeles adyacentes. Por esto la siguiente prueba consiste en elegir aleatoriamente  $N$  parejas de píxeles adyacentes  $(x_i, y_i)$  (vertical, horizontal) de la imagen original y posteriormente de la imagen cifrada y medir la correlación entre dichas parejas calculando el **coeficiente de correlación** [20], [1]. Entonces el coeficiente de correlación se calcula como

$$\gamma_{x,y} = \frac{cov(x,y)}{\sqrt{D(x)}\sqrt{D(y)}} \quad \text{con } D(x) \neq 0, D(y) \neq 0 \quad (7.1)$$

donde

$$cov(x,y) = \frac{1}{N} \sum_{i=1}^N (x_i - E(x))(y_i - E(y))$$

$$E(x) = \frac{1}{N} \sum_{i=1}^N x_i$$

$$D(x) = \frac{1}{N} \sum_{i=1}^N (x_i - E(x))^2$$

$x_i$  y  $y_i$  corresponden a una pareja de píxeles adyacentes, tomando en cuenta sólo uno de los colores del píxel,  $N$  es el número de parejas  $x_i, y_i$  y  $E(x)$  y  $E(y)$  son respectivamente los promedios de  $x_i$  y  $y_i$ . Cabe señalar que en el caso de imágenes a color se calculan tres coeficientes de correlación, uno por cada canal de color (RGB). Si dos píxeles adyacentes están altamente correlacionados, lo estarán en cada uno de los tres colores, análogamente, si dos píxeles adyacentes tienen baja correlación, se encontrará una baja correlación en cada uno de los colores.

En las imágenes originales los valores de coeficiente de correlación son altos (cerca de uno) esto se debe a que los píxeles en dichas imágenes están fuertemente correlacionados. Sin embargo, en las imágenes cifradas sus coeficientes de correlación son cercanos a cero. En la figura 7.2, se muestran las

Correlación	Tamaño (pix)	C.C. Horizontal	C.C. Vertical
Imagen Original: Lena	512x512	0.969065	0.982493
Imagen Cifrada: Lena		0.002630	0.001097
Imagen Original: Ajedrez	512x512	0.972602	0.972602
Imagen Cifrada: Ajedrez		0.001341	0.003599
Imagen Original: Libélula	509x340	0.917859	0.914721
Imagen Cifrada: Libélula		0.005053	0.000675
Imagen Original: Negro	1024x768	1.00	1.00
Imagen Cifrada: Negro		0.000541	0.001081

Tabla 7.1: Coeficiente de Correlación color verde (G)

distribuciones de correlación de pixeles adyacentes horizontales de las imágenes originales y de sus respectivas cifradas.

Podemos notar en la Tabla 7.1 que las imágenes originales tienen valores de coeficiente de correlación altos (cerca de uno), sin embargo esta correlación se pierde al ser cifradas las imágenes y sus coeficientes de correlación son cercanos a cero.

Esto es más claro en la figura 7.2, se muestran las distribuciones de correlación de pixeles adyacentes horizontales de las imágenes originales y de sus respectivas cifradas. Estos valores se calcularon usando el lenguaje Python que se encuentra en el Apéndice D.

## 7.2. Sensibilidad de la llave

Para notar la sensibilidad a un pequeño cambio en la llave que se usa para cifrar la imagen, se cifró cada una de las cuatro imágenes usadas en el trabajo con la llave  $k_1$  y después con la llave  $k_2$  que difiere de  $k_1$  en un solo bit. Después de cifrar la imagen con ambas llaves, se calcula la correlación adyacente entre dichas imágenes obtenidas para saber si difieren entre ellas.

En la figura 7.3 se muestra que las imágenes obtenidas de cifrar la misma imagen original con llaves distintas no están fuertemente correlacionadas entre sí, es decir, la imagen cifrada con la llave  $k_1$  y la cifrada con la llave  $k_2$  que difiere de  $k_1$  por un bit, son muy distintas entre sí, aún cuando  $k_1$  y

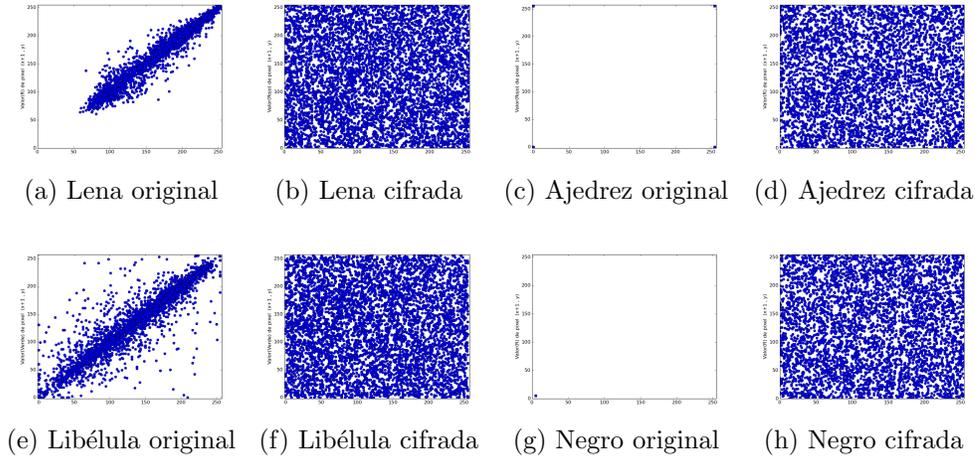


Figura 7.2: Correlación de pares de píxeles adyacentes (Rojo y Verde)

$k_2$  son muy similares. Las llaves  $k_1$  y  $k_2$  difieren del valor que aparece en la primera entrada del segundo arreglo, en  $k_1$  dicho valor es 12 y en  $k_2$  es 11.

$$k_1 = [(200, 145, 18, 2, 3, 4, 45, 66, 32), (12, 34, 77, 2, 178, 234, 12, 13, 14), (2, 178, 234, 12, 13, 14, 12, 45, 56), (67, 68, 89, 200, 145, 18, 2, 3, 4), (45, 66, 32, 2, 34, 77, 2, 178, 234), (12, 13, 14, 11, 45, 56, 67, 68, 89)],$$

$$k_2 = [(200, 145, 18, 2, 3, 4, 45, 66, 32), (12, 34, 77, 2, 178, 234, 12, 13, 14), (2, 178, 234, 12, 13, 14, 12, 45, 56), (67, 68, 89, 200, 145, 18, 2, 3, 4), (45, 66, 32, 2, 34, 77, 2, 178, 234), (12, 13, 14, 12, 45, 56, 67, 68, 89)]$$

Otra prueba de robustez consiste en medir la sensibilidad de la llave en el proceso de descifrado [5]. Se cifra la imagen usando  $k_1$  y después se procede a descifrarla usando la llave correcta  $k_1$ , luego se vuelve a descifrar pero con la llave incorrecta  $k_2$ . Como se espera, el descifrado de la imagen usando  $k_1$  nos devuelve la imagen original, mientras que al usar la incorrecta no. Los resultados se muestran en la figura 7.4.

### 7.3. Sensibilidad del cifrado

Ya que hemos probado la sensibilidad de la llave al cifrar las imágenes, una prueba interesante sería verificar la sensibilidad del cifrado mismo. Es decir,

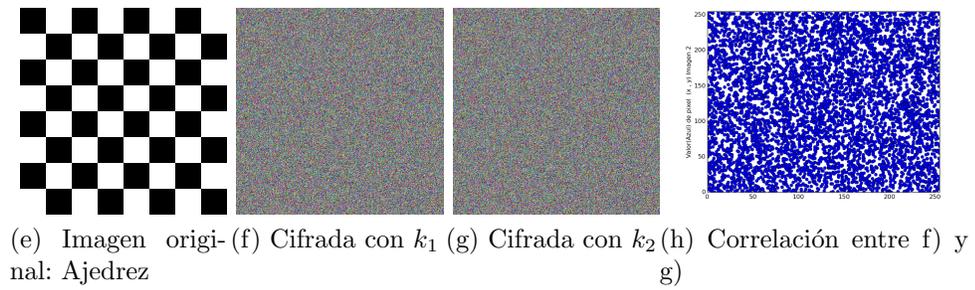
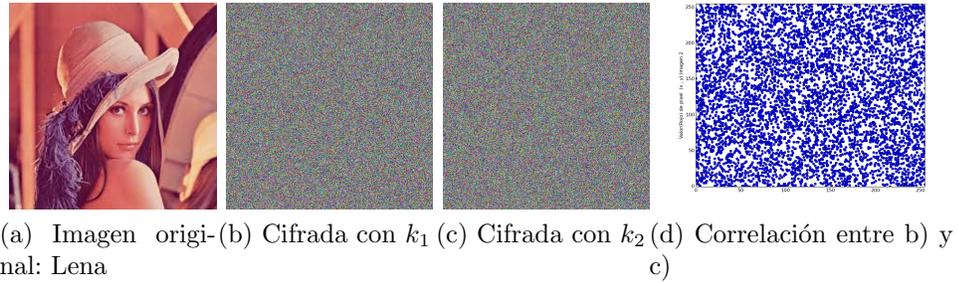


Figura 7.3: Sensibilidad de la llave para cifrar con Lorenz

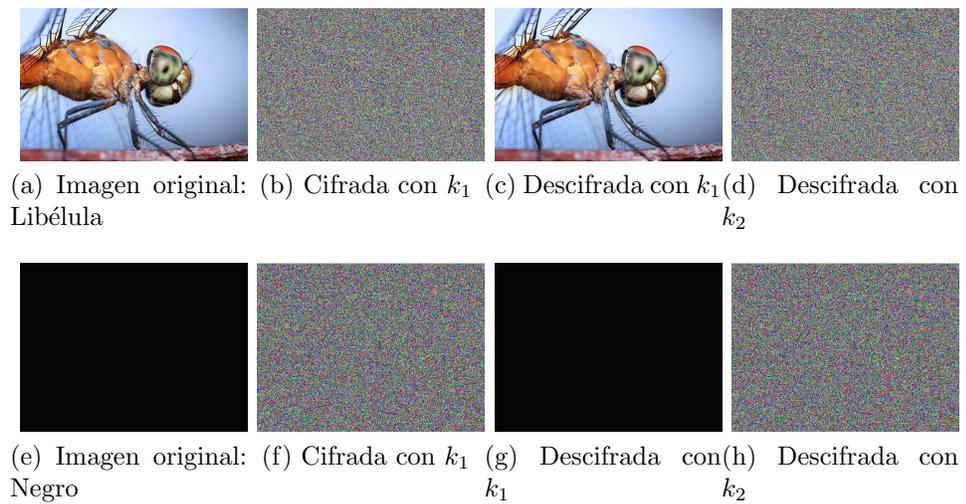


Figura 7.4: Sensibilidad de la llave para descifrar con Lorenz

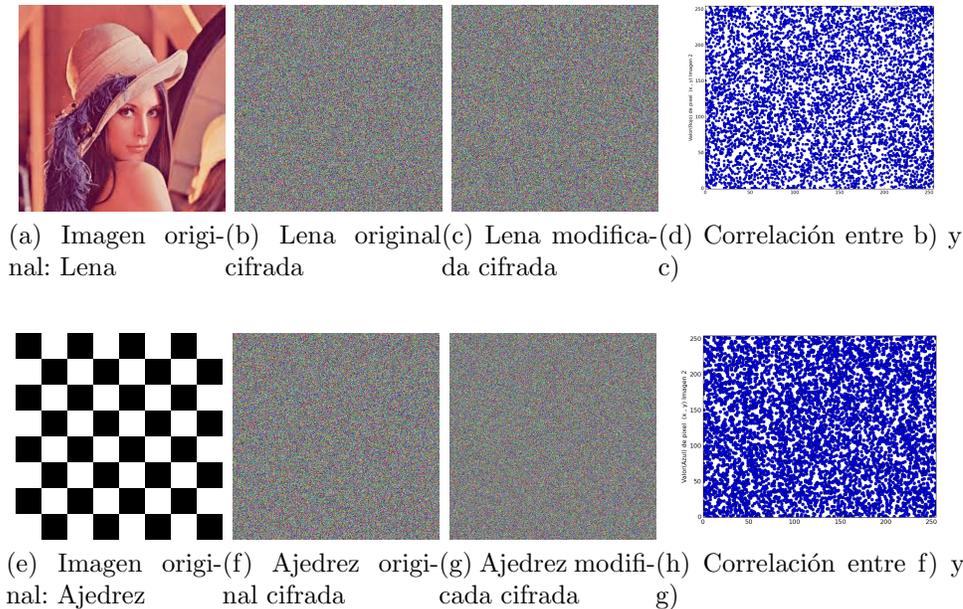


Figura 7.5: Sensibilidad del cifrado modificando un bit en imagen original

si cambiamos un bit de la imagen en claro y ciframos con la misma llave la imagen original y la del bit modificado esperamos obtener dos imágenes cifradas lo suficientemente distintas para no revelar información a un posible atacante. Se usaron las cuatro imágenes que hemos utilizado hasta ahora, se cifraron con  $k_1$  y cambiamos un bit del pixel  $(0, 10)$  en cada imagen, ciframos de nuevo usando  $k_1$  y analizamos la correlación de las imágenes obtenidas.

Cabe mencionar que la correlación que usamos solo toma en cuenta uno de los tres canales de color, pero el análisis se hizo para los tres canales, ya que si un pixel esta fuertemente correlacionado con otro, cada uno de sus colores también lo estará, así fue suficiente examinar la correlación de azul en las imágenes obtenidas. Los resultados de cifrar con la llave  $k_1$  las imágenes originales y después la imagen con un bit modificado se muestran en la figura 7.5, podemos notar que la correlación entre las imágenes cifradas es baja, esto nos dice que el cambio de un solo bit en la imagen original, aunque imperceptible a nuestra vista, afecta de manera considerable el resultado del cifrado.

## 7.4. Comprobación del descifrado

Para corroborar que después de descifrar una imagen obtenemos realmente la imagen original, usamos la función Hash SHA-512 en ambas imágenes. Podemos comprobar así, que la imagen después del proceso de descifrado no ha sufrido cambios que no puedan ser percibidos a simple vista y la imagen original sea recuperada efectivamente.

Después de aplicar la función Hash SHA-512 a la imagen original y la descifrada obtenemos la misma salida. Esto nos indica que el algoritmo de descifrado recupera la imagen original.



(a) Lena Original

(b) Lena Descifrada

**Original: hex:** f1f7466b6805f90a34ccd743f708360da698b6001ed2510eb1a46f57d466f0877aa8301a5e0344b4e1a4b4cf7b0a06582fe544148f5355aeede213a33de0b51

**Cifrada: hex:** f1f7466b6805f90a34ccd743f708360da698b6001ed2510eb1a46f57d466f0877aa8301a5e0344b4e1a4b4cf7b0a06582fe544148f5355aeede213a33de0b51

## 7.5. Tiempos de ejecución

Se midieron tiempos de ejecución del cifrado propuesto con una y cinco rondas usando varias imágenes. Los resultados son los que se muestran en la tabla 7.2.

<b>Cifrado de datos</b>	Texto Mafalda	0.0041310787 seg.
<b>Descifrado de datos</b>	Texto Mafalda	0.0062069893 seg.
<b>Cifrado de Imágenes</b>	Lena 5 rondas	48.1859748363 seg.
<b>Descifrado de Imágenes</b>	Len 5 rondas	58.9982681274 seg.
<b>Cifrado de Imágenes</b>	Chees 1 ronda	21.4956669807 seg.
<b>Descifrado de Imágenes</b>	Chees 1 ronda	38.2236268520 seg.

Tabla 7.2: Tiempos de ejecución



# Conclusiones y Perspectivas

En este trabajo se propone un método de cifrado de datos e imágenes usando el sistema caótico de Lorenz. Los puntos importantes son que las funciones caóticas pueden ser una buena alternativa para generar herramientas eficaces en el diseño de cifrados robustos. Además después de las pruebas de robustez realizadas como: histogramas de frecuencias, correlación entre pixeles adyacentes, sensibilidad de llave y cifrado y el análisis diferencial se concluye que el cifrado puede ser una buena alternativa para cifrar imágenes pequeñas y datos con buena seguridad, sin embargo cuando la imagen a cifrar es grande el cifrado se vuelve significativamente más perceptible.

Se realizó un análisis de caos con cierta profundidad para conocer las características del sistema de Lorenz que se pueden explotar en la forma de cifrar datos e imágenes. Se explota la aleatoriedad y no predictibilidad que provee la Tabla de Lorenz para la generación de subllaves lo que simplifica el algoritmo computacional, y en la cual recae la seguridad del cifrado.

Se propuso un diseño del cifrado basado en el trabajo de Kocarev y Jakimoski ([16]) con uso de las herramientas dadas por el atractor de Lorenz. Se realizaron varios cambios en el proceso con el fin siempre de producir un algoritmo complejo y a la vez rápido en su ejecución. Finalmente se concluye que el cifrado propuesto es seguro ya que el algoritmo puede implementarse fácilmente con operaciones sencillas de pixeles dando los resultados esperados en las pruebas del análisis estadístico. Sin embargo encontramos que el cifrado por bloques no resultó ser lo tan eficaz como se esperaba en el cifrado de imágenes, esto es debido a que el tamaño del bloque a cifrar que escogimos fue de 24 pixeles y una imagen promedio cuenta con aproximadamente  $512 \times 512$  pixeles, lo cual hizo que el programa aunque resultaba robusto en sus pruebas, también era lento. En cuanto al cifrado de datos la implementación muestra una velocidad que es apenas perceptible.

Dentro del diseño del cifrado se incluyó que se hiciera un barrido de izquierda a derecha y luego de derecha a izquierda, esto para lograr la difusión y confusión de Shannon, haciendo más difícil un ataque al cifrado. Otra parte importante para la seguridad del cifrado fue el tamaño del espacio de llaves, ya que el fijar los parámetros usados nos dio libertad de expandirlo incluyendo un rango considerable de condiciones iniciales para generar la Tabla de Lorenz impidiendo el ataque de búsqueda exhaustiva.

Este trabajo junto con muchos otros que lo anteceden sugieren que tal vez exista una relación más profunda de la que pensamos entre la criptografía y la teoría del caos, aún sin explotar, que pueden darnos buenas alternativas para cifrados robustos e interesantes.

Para concluir se tienen varios aspectos para continuar este trabajo de tesis con las que se podría mejorar el cifrado propuesto:

- Modificar el cifrado por un método de cifrado por flujo, para mejorar los tiempos de ejecución en las imágenes.
- Realizar Criptoanálisis Lineal para completar las pruebas de robustez realizadas, ya que junto con el Criptoanálisis Diferencial son de los ataques más importantes en criptografía.
- Realizar pruebas del NIST para probar la fortaleza del cifrado expandiendo el espacio de llaves a los parámetros usados en el sistema.
- Realizar una versión más eficiente en cuanto a tiempo de ejecución que en un futuro pueda programarse para una aplicación de dispositivos móviles.

# Apéndice A

## Cifrado de imágenes

```
import numpy as np
from PIL import Image, ImageOps
from time import time
from math import floor

import numpy as np
from PIL import Image
from time import time
from math import floor
#im = Image.open("dragonfly.jpg")
im.show()
pix = im.load()

XX, YY = im.size
print(XX,YY)
print "Píxeles iniciales \n"
for i in range(24):
print pix[i,0]
print "\n"

#Cambio de un píxel
pix[10,0] = (255,255,0)

#----División por bloques----#
N_bloques = floor(XX/24)
N_pix_sc = XX%24
```

```

print "Número de bloques por ancho de imagen: ", N_bloques
print "Número de pixeles sin cifrar primera ronda", N_pix_sc

def sumpix((a,b,c),(d,e,f)):
    return (a+d,b+e,c+f)
def sumpixm((a,b,c),(d,e,f)):
    return ((a+d)%256,(b+e)%256,(c+f)%256)
def SUMPIX((a,b,c,d,e,f,g,h,i),(a2,b2,c2,d2,e2,f2,g2,h2,i2)):
    return ((a+a2)%256,(b+b2)%256,(c+c2)%256,(d+d2)%256,
(e+e2)%256,(f+f2)%256,(g+g2)%256,(h+h2)%256,(i+i2)%256)
def prpix((a,b,c),n):
    return (n*a,n*b,n*c)
def mod((a,b,c),n):
    return (a%n,b%n,c%n)
def MOD((a,b,c,d,e,f,g,h,i),n):
    return (a%n,b%n,c%n,d%n,e%n,f%n,g%n,h%n,i%n)
def sumpixE((a,b,c),n):
    return (a+n,b+n,c+n)
def N((a,b,c),_cache = {}):
    if not _cache.has_key((a,b,c)):
        _cache[(a,b,c)] = (8*a,8*b,8*c)
    return (_cache[(a,b,c)])
def lorenz(x,y,z,a,r,b,dt,ite):
    for i in range(ite):
        x2 = x + a*(y-x)*dt
    y2 = y + (r*x - y - x*z)*dt
    z2 = z + (x*y - b*z)*dt
    #Reassigna
    x = x2
    y = y2
    z = z2
    a = int(256*(x%1))
    b = int(256*(y%1))
    c = int(256*(z%1))
    return (a,b,c) #Devuelve un pixel

def comprime((a,b,c,d,e,f,g,h,i),_cache = {}):
    if not _cache.has_key((a,b,c,d,e,f,g,h,i)):
    x = (a,b,c,d,e,f,g,h,i)
    y = [0,0,0]

```

```

for j in range(3):
y[j]= x[j]+x[j+3]+x[j+6]
_cache[(a,b,c,d,e,f,g,h,i)] =
(y[0]%256,y[1]%256,y[2]%256)
    return(_cache[(a,b,c,d,e,f,g,h,i)])

xini = 1.2
yini = 1.3
zini = 3.6
a = 10.0
r = 28.0
b = 8/3
dt = 0.005
#-----Genera Tabla de Lorenz-----
#Guarda los valores (x,y,z) generados por Lorenz
#para las iteraciones 4,000 a la 7,000.

print "Creando Tabla de Lorenz \n"
Tabla = [[0]*3 for i in range(3000)]
x1 = xini
y1 = yini
z1 = zini
t_tablai = time()

for i in range(4000):
    X2 = x1 + a*(y1-x1)*dt
    Y2 = y1 + (r*x1 - y1 - x1*z1)*dt
    Z2 = z1 + (x1*y1 - b*z1)*dt
    #Actualiza
    x1 = X2
    y1 = Y2
    z1 = Z2
for i in range(3000):
    X2 = x1 + a*(y1-x1)*dt
    Y2 = y1 + (r*x1 - y1 - x1*z1)*dt
    Z2 = z1 + (x1*y1 - b*z1)*dt
    #Actualiza
    x1 = X2
    y1 = Y2
    z1 = Z2

```

```

Tabla[i] = (int(256*(x1%1)),int(256*(y1%1)),int(256*(z1%1)))

t_tablaf = time()

#---Algoritmo para generar subllave---
rondas = 5
key = [(200,145,18,2,3,4,45,66,32), (12,34,77,2,178,234,12,13,14),
(12,45,56,67,68,89,200,145,18), (2,3,4,45,66,32,12,34,77),
(2,178,234,12,13,14,12,45,56), (67,68,89,200,145,18,2,3,4),
(45,66,32,2,34,77,2,178,234), (12,13,14,12,45,56,67,68,89)]
print "Generación de subLlaves"
print key[0],"\n"

for ron in range(rondas-1):
kw = comprime(key[8*ron])
kA = N(kw)
kB = sumpix( kA , mod(kw,8) )

lorc = ()
for j in range(3):
I = kB[j]
lorc = lorc + Tabla[I]
k1 = lorc
key.append(k1)
skey = k1
for i in range(1+8*ron,8+8*ron,1):
skey = SUMPIX(key[i],skey)
key.append(skey)

print key
#-----COMIENZA CIFRADO-----
tiempocif_i = time()
indices = (3,4,5,6,7,0,1)
for ron in range(rondas): #Se itera r veces.(No. rondas)
subkey = key[8*ron : 8*ron + 8] #Elige subllave
print "Ronda Número", ron+1 , " en ejecución \n"
print "Subllave", subkey
for z in range(2): #Número de ETAPAS
if ( z==1 ): #Voltea pixeles (ETAPA 2)
voltea_i = time()

```

```

im = ImageOps.mirror(im) #Voltea de izquierda a derecha
im = ImageOps.flip(im) #Voltea de arriba a abajo
pix = im.load()
voltea_f = time()
VI = [[0]*3 for i in range(24)] #Bloque de inicialización
tini_koc = time()
for y in np.arange(Y):
for l in np.arange(N_bloques): #Número de bloques por renglón
L = 24*l
for i in range(24): #Operación entre bloques
pix[L+i,y] = sumpixm(pix[L+i,y],VI[i])
suma = (0,0,0,0,0,0,0,0,0)
    #Píxeles iniciales
    Act = pix[2 + L,y]+pix[10 + L,y]+pix[18 + L,y]
    aux = pix[1 + L,y]+pix[9 + L,y]+pix[17 + L,y]

    w = comprime(subkey[0])
    A = N(w)
    B = sumpix(A , mod(w,8) )
    lor = ()
Asig_tabla_ini = time()
    for j in range(3):
I = B[j]
lor = lor + Tabla[I]
Asig_tabla_fin = time()
    pixeles = SUMPIX(aux , lor)
    pix[2 + L,y] = pixeles[0:3] #Actualiza
    pix[10 + L,y] = pixeles[3:6]
    pix[18 + L,y] = pixeles[6:9]
    for x in indices:
Actt = Act
    Act = pix[x + L,y]+pix[x+8 + L,y]+pix[x+16 + L,y]
suma = SUMPIX(suma,aux) #Acumula suma de X1+X2+...
suma2 = SUMPIX(suma,subkey[(x-2)%8])#Suma + Kj
w = comprime(suma2)
#auxk = SUMPIX(aux,subkey[(x-2)%8])
    k = comprime(SUMPIX(aux,subkey[(x-2)%8]))
    B = sumpix(N(w) , mod(k,8) )
lor = ()
Asig_tabla_ini = time()

```

```
        for j in range(3):
            I = B[j]
            #print "C vale: ", C[j], "I vale :", I
            lor = lor + Tabla[I]
            Asig_tabla_fin = time()
                pixeles = SUMPIX(Actt , lor)
                pix[x + L,y] = pixeles[0:3] #Actualiza
                pix[x+8 + L,y] = pixeles[3:6]
                pix[x+16 + L,y] = pixeles[6:9]
                aux = Actt
            for i in range(24): #Actualiza VI
                VI[i] = pix[L+i,y]
            #print VI, "\n"
            #im.show()
            #Se guarda la imagen cifrada
            im.show()
            print "Píxeles cifrados \n"
            for j in range(1):
                for i in range(24):
                    print pix[i,j]
                    print "\n"
            imcif = im
            imcif.save("Cifradas/Chess_cif_bc.png")
```

# Apéndice B

## Cifrado de datos

```
import numpy as np
from time import time

Msj = "MAFALDA MEDITA MIENTRAS ODI LA SOPA"
MsjANum = [ord(c) for c in Msj]

if (len(MsjANum) %9) != 0:
    reno = 9 - (len(MsjANum) %9)
    for w in range(reno):
        MsjANum.append(32)
pix = []
for i in range(len(MsjANum)/3):
    pix.append(MsjANum[3*i:3*(i+1)])
XX = len(pix)

#---División por bloques---#
N_bloques = (XX-(XX%24))/24
N_pix_sc = XX%24
print "Número de bloques por ancho de imagen: "

print "Número de pixeles sin cifrar primera ronda"

def sumpix((a,b,c),(d,e,f)):
    return (a+d,b+e,c+f)
def sumpixm((a,b,c),(d,e,f)):
    return ((a+d)%256,(b+e)%256,(c+f)%256)
```

```

def SUMPIX((a,b,c,d,e,f,g,h,i),
(a2,b2,c2,d2,e2,f2,g2,h2,i2)):
    return ((a+a2)%256,(b+b2)%256,
            (c+c2)%256,(d+d2)%256,
            (e+e2)%256,(f+f2)%256,
            (g+g2)%256,(h+h2)%256,
            (i+i2)%256)
def prpix((a,b,c),n):
    return (n*a,n*b,n*c)
def mod((a,b,c),n):
    return (a%n,b%n,c%n)
def MOD((a,b,c,d,e,f,g,h,i),n):
    return (a%n,b%n,c%n,d%n,
            e%n,f%n,g%n,h%n,i%n)
def sumpixE((a,b,c),n):
    return (a+n,b+n,c+n)
def N((a,b,c),n):
    return (n*a,n*b,n*c)
def lorenz(x,y,z,a,r,b,dt,ite):
    for i in range(ite):
        x2 = x + a*(y-x)*dt
        y2 = y + (r*x - y - x*z)*dt
        z2 = z + (x*y - b*z)*dt
        #Reassigna
        x = x2
        y = y2
        z = z2
    a = int(256*(x%1))
    b = int(256*(y%1))
    c = int(256*(z%1))
    return (a,b,c)

def comprime((a,b,c,d,e,f,g,h,i)):
    x = (a,b,c,d,e,f,g,h,i)
    y = [0,0,0]
    for j in range(3):
        y[j]= x[j]+x[j+3]+x[j+6]
    return(y[0]%256,y[1]%256,y[2]%256)
xini = 1.2
yini = 1.3

```

```

zini = 3.6
a = 10.0
r = 28.0
b = 8/3
dt = 0.005
#----Genera Tabla de Lorenz----
#Guarda los valores (x,y,z)
#para las iteraciones 4,000 a la 7,000.

print "Creando Tabla de Lorenz \n"
Tabla = [[0]*3 for i in range(3000)]
x1 = xini
y1 = yini
z1 = zini
t_tablai = time()

for i in range(4000):
    X2 = x1 + a*(y1-x1)*dt
    Y2 = y1 + (r*x1 - y1 - x1*z1)*dt
    Z2 = z1 + (x1*y1 - b*z1)*dt
    #Actualiza
    x1 = X2
    y1 = Y2
    z1 = Z2
for i in range(3000):
    X2 = x1 + a*(y1-x1)*dt
    Y2 = y1 + (r*x1 - y1 - x1*z1)*dt
    Z2 = z1 + (x1*y1 - b*z1)*dt
    #Actualiza
    x1 = X2
    y1 = Y2
    z1 = Z2
    Tabla[i] = (int(256*(x1%1)),
               int(256*(y1%1)),
               int(256*(z1%1)))

t_tablaf = time()

#-Algoritmo para generar subllaves-
rondas = 7

```

```

key = [(200,145,18,2,3,4,45,66,32),
(12,34,77,2,178,234,12,13,14),
(12,45,56,67,68,89,200,145,18),
(2,3,4,45,66,32,12,34,77),
(2,178,234,12,13,14,12,45,56),
(67,68,89,200,145,18,2,3,4),
(45,66,32,2,34,77,2,178,234),
(12,13,14,12,45,56,67,68,89)]
print "Generación de subLlaves"
print key[0],"\n"

for ron in range(rondas-1):
    kw = comprime(key[8*ron])
    kA = N(kw,8)
    kB = sumpix(kA , mod(kw,8))
    lork = ()
    for j in range(3):
        I = kB[j]
        lork = lork + Tabla[I]
    k1 = lork
    key.append(k1)
    skey = k1
    for i in range(1+8*ron,8+8*ron,1):
        skey = SUMPIX(key[i],skey)
        key.append(skey)

print key
#-----COMIENZA CIFRADO-----
indices = (3,4,5,6,7,0,1)
tini_koc = time()
for ron in range(rondas): #Se itera r veces.(rondas)
    subkey = key[8*ron : 8*ron + 8] #Elige subllave
    print "Ronda Número", ron+1 , " en ejecución \n"
    print "Subllave", subkey
    for z in range(2): #Número de ETAPAS
        if ( z==1 ): #Voltea pixeles (ETAPA 2)
            ancho = XX-1
            for i in range((XX-1)/2):
                aux_1 = pix[ancho-i]
                aux_2 = pix[i]

```

```

    pix[i] = aux_1
    pix[ancho-i]= aux_2

    for i in range(XX):
        for j in range((XX-1)/2):
            aux_1 = pix[i]
            aux_2 = pix[i]
            pix[i] = aux_1
            pix[i]= aux_2

VI = [[0]*3 for i in range(24)] #VI

for l in np.arange(N_bloques):
    L = 24*l
    for i in range(24): #_0p entre bloques
        pix[L+i] = sumpixm(pix[L+i],VI[i])

    suma = (0,0,0,0,0,0,0,0,0)
    #Píxeles iniciales
    Act = pix[2 + L]+pix[10 + L]+pix[18 + L]
    aux = pix[1 + L]+pix[9 + L]+pix[17 + L]
    w = comprime(subkey[0])
    A = N(w,8)
    B = sumpix(A , mod(w,8) )
    lor = ()
    for j in range(3):
        I = B[j]
        lor = lor + Tabla[I]
    pixeles = SUMPIX(aux , lor)
    pix[2 + L] = pixeles[0:3]
    pix[10 + L] = pixeles[3:6]
    pix[18 + L] = pixeles[6:9]
    for x in indices:
        Actt = Act
        Act = pix[x + L]+pix[x+8 + L]+
            pix[x+16 + L]

    suma = SUMPIX(suma,aux)
    suma2 = SUMPIX(suma,subkey[(x-2)%8])

```

```

w = comprime(suma2)
auxk = SUMPIX(aux,subkey[(x-2)%8])
k = comprime(auxk)

A = N(w,8)
B = sumpix(A , mod(k,8) )
lor = ()
for j in range(3):
    I = B[j]
    lor = lor + Tabla[I]
píxeles = SUMPIX(Actt , lor)
pix[x + L] = píxeles[0:3]
pix[x+8 + L] = píxeles[3:6]
pix[x+16 + L] = píxeles[6:9]
aux = Actt
for i in range(24): #Actualiza VI
    VI[i] = pix[L+i]

#print "Píxeles cifrados \n"
#for i in range(12):
#    print pix[i]
#    #print "\n"

print pix
# pix es un conjunto de ternas de números
# en cifrado cada terna de pix se representa
# como un número

b6=10**6; b3 = 10**3

cifrado = []
numpix=len(pix)
for i in range(numpix):
    terna = pix[i]
    cifrado.append( terna[0]*b6 +
    terna[1]*b3 + terna[2])
pix.pop(0)
del pix
print "Cifrado:"
print cifrado

```



# Apéndice C

## Código para calcular Lyapunov

```
(*PROGRAMA QUE CALCULA NUMÉRICAMENTE EL EXPONENTE DE LYAPUNOV UTILIZANDO RUNGE KUTTA ORDEN 7*)
```

```
(*Definimos Sistema dinámico y parámetros*)
```

```
F[{x_, y_, z_}] := {a (y - x), x (R - z) - y, x*y - b*z};
```

```
a = 10; b = 8/3; R = 28;
```

```
JacobianMatrix[f_List, v_List] := Outer[D, f, v];
```

```
J = JacobianMatrix[F[{y1[t], y2[t], y3[t]}], {y1[t], y2[t], y3[t]}];
```

```
(*Definimos nuevas variables y ecuación variacional*)
```

```
ϕ = Table[{yk[t], yk,1[t], yk,2[t]}, {k, 4, 12, 3}];
```

```
DPhi = Flatten[J.ϕ];
```

```
EQ3 = Table[D[yk[t], {t, 1}] = F[{y1[t], y2[t], y3[t]}][[k]], {k, 1, 3}];
```

```
EQ9 = Table[D[yk[t], {t, 1}] = DPhi[[k - 3]], {k, 4, 12}];
```

```
(*Definimos condiciones iniciales*)
```

```
YI9 = Table[yk[0] = If[Mod[k, 4] == 0, 1, 0], {k, 4, 12}];
```

```
YI3 = {y1[0] == 0.2, y2[0] == 0.3, y3[0] == 0.5};
```

```
(*RungeKutta 7-orden*)
```

```
T = 2000;
```

```
sol = NDSolve[Join[EQ3, EQ9, YI3, YI9], Table[yk[t], {k, 1, 12}], {t, 0, T}, StepMonitor -> Sow[x],  
Method -> {"ExplicitRungeKutta", "DifferenceOrder" -> 7}, MaxSteps -> Infinity, AccuracyGoal -> 15];
```

```
PhiT = Table[{yk[t], yk,1[t], yk,2[t]}, {k, 4, 12, 3}] /. sol /. t -> T;
```

```
Table[r[k] = Table[Random[], {3}], {k, 1, 100000}];
```

```
λ = Table[Log[Norm[PhiT.r[k]] / Norm[r[k]]] / T, {k, 1, 100000}];
```

```
Mean[λ]
```

```
0.9003768049408823051
```

# Apéndice D

## Código para calcular correlación

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
from time import time
from math import floor

#im = Image.open("dragonfly.jpg")
im.show()
pix = im.load()

XX, YY = im.size
print(XX,YY)

#----CORRELACIÓN HORIZONTAL----
print "CORRELACIÓN HORIZONTAL"
N = (XX-1)*YY #total de pixeles en imagen
print "N: ", N
for i in range(3):
#-----MEDIAS-----
suma = 0
for y in range(YY):
for x in range(XX-1):
suma = suma + pix[x,y][i]
EX = suma*1.0/N
```

```

print "E(X): ", EX

suma=0
for y in range(Y):
for x in range(1,XX):
suma = suma + pix[x,y][i]
EY = suma*1.0/N
print "E(Y): ", EY
#-----DESVIACIONES-----
suma=0
for y in range(Y):
for x in range(XX-1):
suma = suma
+ (pix[x,y][i]-EX)**2
DX = suma*1.0/N
print "D(X): ", DX

suma=0
for y in range(Y):
for x in range(1,XX):
suma = suma + (pix[x,y][i]-EX)**2
DY = suma*1.0/N
print "D(Y): ", DY
#-----COVARIANZA-----
suma = 0
for y in range(Y):
for x in range(XX-1):
suma = suma +
(pix[x,y][i]-EX)*(pix[x+1,y][i]-EY)
cov = suma*1.0/N
print "cov: ", cov

#-----COEFICIENTE DE CORRELACIÓN-----
den = (DX**0.5)*(DY**0.5)
Rxy = cov*1.0/den
print "Valor de Coeficiente: ", i+1, " es: ", Rxy

#---CORRELACIÓN VERTICAL---
print "CORRELACIÓN VERTICAL"
N = XX*(Y-1) #Total de pixeles en imagen

```

```

print "N: ", N
for i in range(3):
#-----MEDIAS-----
suma = 0
for x in range(XX):
for y in range(YY-1):
suma = suma + pix[x,y][i]
EX = suma*1.0/N
print "E(X): ", EX

suma=0
for x in range(XX):
for y in range(1,YY):
suma = suma + pix[x,y][i]
EY = suma*1.0/N
print "E(Y): ", EY
#-----DESVIACIONES-----
suma=0
for x in range(XX):
for y in range(YY-1):
suma = suma + (pix[x,y][i]-EX)**2
DX = suma*1.0/N
print "D(X): ", DX

suma=0
for x in range(XX):
for y in range(1,YY):
suma = suma + (pix[x,y][i]-EX)**2
DY = suma*1.0/N
print "D(Y): ", DY
#-----COVARIANZA-----
suma = 0
for x in range(XX):
for y in range(YY-1):
suma = suma +
(pix[x,y][i]-EX)*(pix[x,y+1][i]-EY)
cov = suma*1.0/N
print "cov: ", cov

#-----COEFICIENTE DE CORRELACIÓN-----

```

```
den = (DX**0.5)*(DY**0.5)
Rxy = cov*1.0/den
print "Valor de Coeficiente: ", i+1, " es: ", Rxy

#-GRÁFICAS DE CORRELACIÓN DE PÍXELES ADYACENTES-
X = []
Y = []
for y in np.random.choice(YY,80):
for x in np.random.choice(XX-1,80):
X.append(pix[x,y][1])
Y.append(pix[x+1,y][1])

print len(X), len(Y)
#print X_ran

plt.plot(X, Y, 'bo')
plt.xlim(-2, 257.5)
plt.ylim(-2, 257.5)
plt.ylabel("Valor(Verde) de pixel (x , y) ")
plt.ylabel("Valor(Verde) de pixel (x+1 , y) ")
#plt.title("Distribución")
plt.show()
```

# Bibliografía

- [1] Ali-Pacha A., Hadj-Said N., Ali-Pacha M. (2013). New Chaotic Cryptosystem Based on the Specific Generator and the Pickover's Attractor. *6th Chaotic Modeling and Simulation International Conference 11-14*, Po Box 1505 El M'Naouer Oran 31000 ALGERIA.
- [2] Alligood, K., Sauer, T. and Yorke, J. (1996). *Chaos: An introduction to dynamical systems*. Nueva York: Springer-Verlag, ISBN 0-387-94677-2.
- [3] Alvarez, G. and Li, S. (2006). Some basic cryptographic requirements for chaos based cryptosystems. *International Journal of Bifurcation and Chaos*, Vol. 16, 2129-2151.
- [4] Barker, E. and Roginsky, A. (2012). Recommendation for Cryptographic Key Generation. *NIST Special Publication 800-133*, <http://dx.doi.org/10.6028/NIST.SP.800-133>.
- [5] Bao, L. and Zhou, Y. (2012). A new chaotic system for image encryption. *International Conference on System Science and Engineering*. June 30-July 2, 78-1-4673-0945-5/12. 69-73.
- [6] Biham, E. and Shamir, A. (1991). Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1), 2-21.
- [7] Blanco G, R. (2010). *Análisis de algoritmos criptográficos y su aplicación al cifrado de archivos*. Tesis para obtener el título de ingeniero en informática. Instituto Politécnico Nacional. Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas, UPIICSA.
- [8] Cherif M. Malek B. and Mekhilef S. (2012). *Cryptography of the Medical Images*. Malasia: PIERS.
- [9] Daemen, J. and Rijmen, V. (2001). *The design of rijndael*. New York: Springer, ISBN 3-540-42580-2.

- [10] Devaney, R.L. (1989). *An Introduction to Chaotic Dynamical Systems, 2nd ed.*. Redwood City California: Addison- Wesley.
- [11] Dubeibe, F.L. (2013). Cálculo del máximo exponente de Lyapunov con Mathematica. *Revista Colombiana de Física*, Vol. 45, No. 152.155.
- [12] Galavis, J. y Magidin, A. (2003). Introducción a la criptografía. *UNAM, Vínculos Matemáticos*, Vol. 15.
- [13] Granados, Gibrán. (2006). Introducción a la Criptografía. *Revista Digital Universitaria*. Vol. 7 Núm 7, 1067-6079.
- [14] Hassan, H. Al Haj, Kassem A., Harkouss Y. and Assaf R. (2006). New chaotic image encryption technique. *Chaos, Solitons and Fractals*. Vol. 29, 393-399.
- [15] Howard M. (2012). A tutorial on linear and differential cryptoanalysis. *Journal Cryptologia*, Vol. 26, 189-221.
- [16] Jakimoski, G. y Kocarev, L. (2011). Chaos and cryptography: Block encryption ciphers based on chaotic maps. *IEEE Transactions on circuits and systems I: Fundamental theory and applications*, Vol. 48, 163-169.
- [17] Kocarev, L. and Lian, S. (2011). *Chaos-Based Cryptography: Theory, Algorithms and Applications*. Berlin: Springer-Verlaq, ISBN 978-3-642-20542-2.
- [18] Lee, John M. (2010). *Introduction to Topological Manifolds*. Second Edition. Nueva York: Springer.
- [19] Lorenz, Edward N. (1963). Deterministic Nonperiodic Flow. *Journal of the atmospheric sciences*. Vol. 20, 130-140.
- [20] Loukhaoukha, K., Chouinard, J. and Berdai, A. (2012). A secure image encryption algorithm based on Rubik's cube principle. *Journal of electrical an computer engineering*, Vol. 2012, 1-13.
- [21] Manjunath N, S.G. Hiremath. (2015). Image and text steganography based on RSA and chaos cryptography algorithm with Hash-LSB technique. *IJECS*. Vol. 3.
- [22] Paar, C. and Pelzl J. (2010). *Understanding Cryptography*. Berlin: Springer-Verlag, ISBN 978-3-642-04100-6.
- [23] Perko, Lawrence. (2000). *Differential Equations and Dynamical Systems*. Nueva York: Springer-Verlag, ISBN 0-387-95116-4.

- [24] Raphael, J. (2011). Cryptography and steganography - A Survey. *International Journal of Computer Technology and Applications*, Vol. 2(3), 626-630.
- [25] Salas, S., Hille, E., Etgen, G. (2007). *Calculus: One and several variables*. Westford: John Wiley & Sons, Inc. Tenth Edition, ISBN-13 978-0471-69804-3.
- [26] Shannon, C.E. (1949). Communication Theory of Secrey Systems. *Bell System Technical Journal*. Vol. 28, Issue 4, 601-753.
- [27] Song W., Liang J. (2013). Difference equation of Lorenz system. *International Journal of Pure and Applied Mathematics*. Vol. 83, 102-109.
- [28] Strogatz, Steven H. (1994). *Nonlinear Dynamics and Chaos*. Nueva York: Perseus Books.
- [29] Xingyuan W., Yang L. and Liu R. (2010). A chaotic image encryption algorithm based on perceptron model. *Nonlinear Dyn.* 62:615-621.
- [30] Zhang, Q., Guo, Y., Li, W. and Ding, Q. (2016). Image encryption method based on discrete Lorenz chaotic sequences. *Journal of Information Hiding and Multimedia Signal Processing*, Vol. 7(3), 576-586.



Casa abierta al tiempo

UNIVERSIDAD AUTÓNOMA METROPOLITANA

# ACTA DE EXAMEN DE GRADO

No. 00180

Matrícula: 2143805861

CIFRADO DE IMÁGENES Y DATOS  
UTILIZANDO EL MODELO CAÓTICO  
DE LORENZ

En la Ciudad de México, se presentaron a las 11:00 horas del día 23 del mes de julio del año 2018 en la Unidad Iztapalapa de la Universidad Autónoma Metropolitana, los suscritos miembros del jurado:

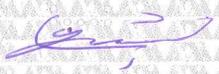
DR. HORACIO TAPIA RECILLAS  
DRA. GINA GALLEGOS GARCIA  
DR. JOSE NOE GUTIERREZ HERRERA



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

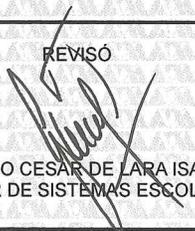
MAESTRA EN CIENCIAS (MATEMÁTICAS APLICADAS E INDUSTRIALES)

DE: ERIKA NANCY LEOS RODRIGUEZ

  
ERIKA NANCY LEOS RODRIGUEZ  
ALUMNA

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

Aprobar

REVISÓ  


LIC. JULIO CESAR DE LARA ISASSI  
DIRECTOR DE SISTEMAS ESCOLARES

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

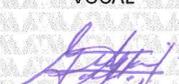
DIRECTOR DE LA DIVISIÓN DE CBI

  
DR. JESUS ALBERTO OCHOA TAPIA

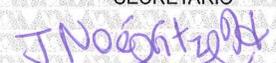
PRESIDENTE

  
DR. HORACIO TAPIA RECILLAS

VOCAL

  
DRA. GINA GALLEGOS GARCIA

SECRETARIO

  
DR. JOSE NOE GUTIERREZ HERRERA