Instituto Politécnico Nacional
Centro de Investigación en
Computación

Modelado de criptosistemas usando autómatas celulares y transformaciones no afines

Tesis que presenta:
Joel Noyola Bautista

Que para obtener el Grado de:
Maestría en Ciencias de la Computación

Director:
M. en C. Germán Téllez Castillo

México, D.F. **Diciembre de 2014**

SIP-14

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 12:00 horas del día 14 del mes de noviembre de 2014 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"Modelado de criptosistemas usando autómatas celulares y transformaciones no afines"

Presentada por el alumno:

NOYOLA	**BAUTISTA**	**JOEL**
Apellido paterno	Apellido materno	Nombre(s)

Con registro:

B	1	2	1	0	7	1

aspirante de: **MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Director de Tesis

M. en C. Germán Téllez Castillo

Dr. Grigori Sidorov

Dr. Luis Pastor Sánchez Fernández

Dr. Carlos Fernando Aguilar Ibáñez

Dra. Nareli Cruz Cortés

Dr. Moisés Salinas Rosales

PRESIDENTE DEL COLEGIO DE PROFESORES

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION
EN COMPUTACION
DIRECCION

Dr. Luis Alfonso Villa Vargas

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México el día 26 del mes de noviembre del año 2014, el (la) que suscribe Noyola Bautista Joel alumno (a) del Programa de Maestría en Ciencias de la Computación con número de registro B121071, adscrito al Centro de Investigación en Computación, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección del M. en C. Germán Téllez Castillo y cede los derechos del trabajo intitulado Modelado de criptosistemas usando autómatas celulares y transformaciones no afines, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección joelnb88@gmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Noyola Bautista Joel

Nombre y firma

Resumen

En este trabajo presentamos un modelo basado en Autómatas Celulares de una y dos dimensiones para simular un criptosistema de llave privada con un cifrado por bloques de 128 bits. El modelo emplea una serie de transformaciones, incorporando un nivel de cifrado por cada una de ellas. El autómata celular de una dimensión trabaja sobre la extensión del campo GF(2^8) y toma una llave privada K como estado inicial, después de iterar el autómata, se genera una secuencia de cifrado la cual controlará las transformaciones. El autómata de dos dimensiones sirve para añadir la instancia de un problema de la clase NP e incrementar la seguridad del criptosistema. Los resultados del modelo propuesto nos permiten concluir que el sistema es capaz de eliminar periodicidad en el texto cifrado cuando el texto plano contiene caracteres con alto índice de periodicidad; es decir, el criptosistema genera una función de distribución casi uniforme en los textos cifrados.

Abstract

In this thesis we have developed a private key cryptosystem model based on one-dimensional and two-dimensional cellular automata with a 128 bit cipher block. The model employs a series of transforms, adding an encryption level for each. The one-dimensional cellular automaton works over GF(2^8) extension field. An encryption stream is generated by this cellular automaton, using the private key as initial state. This stream helps control the transforms. The two-dimensional cellular automaton is used to add an instance of a NP problem in order to increase cryptosystem security level. Our results allow us to conclude that our model can remove periodicity on ciphertext when plaintext contains periodicity in its symbols.

Índice general

Índice de figuras

Índice de tablas

Lista de algoritmos

Capítulo 1

Introducción

1.1. Planteamiento del problema

La seguridad de la información desde la antigüedad ha sido un problema a resolver, los primeros intentos de ocultar la información fueron hechos por Julio César, él usó una substitución de letras en sus cartas que enviaba a sus legiones. A mediados del siglo veinte hubo un gran avance cuando este problema atrajo la atención de los matemáticos y computólogos quienes hicieron una teoría formal de los sistemas criptográficos. Hoy en día, debido al incremento en tecnología; las redes de computadoras juegan un papel importante en la comunicación, por tal razón el problema de seguridad de la información sigue vigente. Las grandes industrias necesitan mantener su información segura, los bancos nacionales e internacionales deben asegurar que los canales de comunicación al efectuar transacciones sean seguros de terceros que intentan robar o alterar la información, la milicia debe asegurar la integridad de la información, y la comunicación entre dos personas a través de disposivos móviles o a través de redes sociales debe ser segura.

En computación existe la clase de complejidad P y NP. Los problemas que son tratables o solubles eficientemente; es decir, los problemas que se pueden resolver en tiempo polinomial, pertenecen a la clase P, mientras que los problemas que no se pueden resolver eficientemente en tiempo polinomial pertenecen a la clase NP.

El modelado y simulación bajo el enfoque de los Autómatas Celulares es una de las metodologías usadas para el estudio de sistemas complejos. Para nuestro problema de seguridad de la información vamos a modelar y simular un sistema criptográfico usando el enfoque de autómatas celulares, añadiendo operaciones sobre estructuras algebraicas, además de incorporar la instancia de un problema que pertence a la clase NP.

1.2. Justificación

Si utilizamos un modelo para simular un sistema criptográfico basado en autómatas celulares, podremos ver a cada símbolo contenido en la información a cifrar como una célula que pertene al autómata celular. Los autómatas celulares evolucionan en etapas de tiempo discreto, permitiendo en cada evolución incrementar el nivel de seguridad de la información. Si, además incorporamos la estructura algebraica de campo finito y distintos niveles de transformación para que los autómatas trabajen sobre ellos, esto implicará que el modelo basado en autómatas celulares, aumentará la seguridad de la información para evitar que sea alterada o leída.

Invertir un autómata celular reversible de dos dimensiones es un problema que se encuentra en la clase de problemas NP [17], por lo tanto si nosotros integramos en alguno de los niveles de transformación un autómata celular reversible de dos dimensiones, entonces en orden de desvelar la información, sin conocimiento alguno del funcionamiento del sistema criptográfico, se deberá resolver la instancia de este problema, por lo cual el tiempo requerido para romper el criptosistema será exponencial, incrementando así la seguridad de la información.

1.3. Hipótesis

Una combinación adecuada de herramientas matemáticas permitirá diseñar un criptosistema robusto para cifrar y descifrar texto, donde a pesar de que el texto plano a cifrar presente periodicidad, el resultado será un texto cifrado donde sus caracteres presentan una función de distribución casi uniforme.

1.4. Objetivos

El objetivo general así como los objetivos particulares de la tesis son los siguientes:

1.4.1. Objetivo general

Diseñar un criptosistema de llave privada con cifrado por bloques de 128 bits, usando Autómatas Celulares de una dimensión y dos dimensiones, incorporando cuatro niveles de transformación a cada bloque.

1.4.2. Objetivos particulares

◇ Diseñar un autómata celular de una dimensión grupo sobre GF(2^8).

◇ Diseñar un autómata celular de dos dimensiones reversible.

◇ Diseñar los cuatro niveles de transformación.

◇ Diseñar un isomorfimo entre elementos del autómata celular de una dimensión y elementos del autótama celular reversible de dos dimensiones.

◇ Diseñar los algoritmos para realizar el proceso de cifrado y descifrado de texto.

◇ Diseñar una interfaz gŕafica que permita al usuario la manipulación del sistema.

1.5. Contribuciones

◇ Un criptosistema de llave privada para cifrar y descifrar texto de manera eficiente y robusta.

◇ Una metodología basada en autómatas celulares que conjunta la extensión de campos finitos, álgebra lineal y transformaciones afines.

1.6. Organización de la tesis

La tesis está formada por seis capítulos y cada uno de ellos está organizado como sigue:

El capítulo uno consiste del planteamiento del problema de seguridad de la información, además de los objetivos general y particulares que buscamos con la elaboración de este trabajo de investigación y finalmente la justificación del problema.

El capítulo dos contiene el marco teoríco de criptología, autómatas celulares, aritmética de estructuras algebraicas y finalmente un apartado donde hablamos sobre la teoría de extensión de campos sobre autómatas celulares.

El capítulo tres presenta el estado del arte; es decir, mostramos el panorama general de las investigaciones sobre criptología usando autómatas celulares. Particularmente hablamos de tres modelos propuestos para resolver el problema de seguridad bajo el enfoque de autómatas celulares. También incluye una sección que habla de la caracterización de autómatas celulares usando álgebra matricial sobre la extensión de campos.

El capítulo cuatro es el capítulo más importante debido a que muestra el modelo propuesto para resolver el problema planteando en el capítulo uno, explicando las mejoras que implementan al modelo propuesto en [11]. Detalla el diseño de cada uno de los autómatas, los cuatro niveles de transformación así como el proceso de cifrado y descifrado de texto.

El capítulo cinco muestra los resultados obtenidos a través de la simulación del modelo propuesto en el capítulo cuatro, comparando con los modelos [11, 13, 31]. También hacemos algunas pruebas al modelo en orden de verificar la seguridad al sistema criptográfico.

El capítulo seis contiene las conslusiones que se obtuvieron una vez finalizado el trabajo de investigación. También muestra los trabajos futuros implicados a través del desarrollo de la investigación que pueden ayudar ampliando el modelo prupuesto en el capítulo cuatro.

Con el fin de que el lector entienda todos los términos que se emplean en la tesis -donde la gran mayoría de los términos están definidos en el transcurso de los capítulos- al final del trabajo se encuentra un glosario de términos. El lector a través de los capítulos encontrará palabras que finalizan con el símbolo $^{(\clubsuit)}$; por ejemplo: *word*$^{\clubsuit}$, indica que este término puede ser consultado al final del escrito, en el glosario de términos en el **Apéndice B**.

Capítulo 2

Antecedentes

Hay dos periodos en los cuales la criptología se ha desarrollado. En el primer periodo resultados publicados permiten concluir que la criptología de este periodo es más un arte que una ciencia. El segundo periodo se da, cerca del año 1940, cuando se empezó a desarrollar una teoría formal de sistemas criptográficos. En este capítulo se resvisa los conceptos fundamentales para entender el problema que se plantea en esta tesis, esto es, se revisa los con conceptos necesarios de criptología, criptografía y criptoanálisis. Abordamos también el concepto de Autómatas Celulares, partiendo de una idea básica y posteriormente definirlos formalmente. Finalmente, en este capítulo se revisará un apartado de matemática afín, en el cual se revisará los conceptos de tranformaciones afines y campos finitos.

2.1. Criptología

Es necesario hacer mención de la importancia que juega la información y más aún la información privada; es decir, información la cual queremos no sea vista o interceptada por algún tercero en la comunicación entre un transmisor y un receptor. Por información privada (secreta) nos referimos a información que, por acuerdo común entre un transmisor y un receptor, no está destinada a un tercero. En referencia al término "tercero" usaremos el término "*adversario*". La historia de este término se remonta a las aplicaciones militares a mediados del siglo veinte [20].

La Criptología se ocupada de estudiar y desarrollar métodos matemáticos para proteger información privada transmitida por canales de comunicación pública. La Criptología consiste de dos partes: *criptosíntesis* (o *criptografía*) y *criptoanálisis* [20] [29].

La Criptografía desarrolla técnicas para proteger información. La información privada está contenida en el *texto plano* compuesto por el transmisor, el cual es una secuencia de letras que pertenecen a un *alfabeto*♣.

Un intento de *transformación*♣ del texto plano, complicando la extracción de la información privada contenida en el texto, es llamado *cifrado*. El procedimiento usado para este propósito es llamado *sistéma criptográfico*. Los sistemas criptográficos incluyen un elemento especial, conocido sólo por el transmisor y o el receptor, el cual es llamado *llave privada (secreta)*. Un *texto cifrado* o *criptograma* es el resultado de crifrar un texto plano. El cifrado debe ser reversible, es decir; debe permitir que el texto plano se recupere de forma única apartir del criptograma (*descifrado*).

El Criptoanálisis, en contraste con la criptografía, estudia las posibilidades de romper el criptosistema; es decir, la extracción del texto plano apartir de un criptograma por un adversario, quien no conoce la llave privada.

Definición 2.1 *Un* criptosistema *es una* ***5****-tupla* $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ [29] *tal que:*

1. $\mathcal{P}, \mathcal{C}$ *y* $\mathcal{K}$ *son conjuntos finitos, donde*

- $\mathcal{P}$ *es el espacio de textos planos*
- $\mathcal{C}$ *es el espacio de textos cifrados*
- $\mathcal{K}$ *es el espacio de llaves.*

Elementos que pertenecen a $\mathcal{P}$ *se conocen como texto plano y elementos que pertenecen a* $\mathcal{C}$ *se conocen como texto cifrado.*

2. $\mathcal{E} = \{E_k \mid k \in \mathcal{K}\}$ *es un conjunto de funciones tal que* $E_k : \mathcal{P} \longrightarrow \mathcal{C}$*, que son usadas para el cifrado, y* $\mathcal{D} = \{D_k \mid k \in \mathcal{K}\}$ *es un conjunto de funciones tal que* $D_k : \mathcal{C} \longrightarrow \mathcal{P}$*, que son usadas para el descifrado.*

3. Para cada llave $e \in \mathcal{K}$*, existe una llave* $d \in \mathcal{K}$ *tal que para cada* $p \in \mathcal{P}$*:*

$$D_d(E_e(p)) = p.$$

Un criptosistema es *simétrico* o de *llave privada* si $d = e$, o si d puede ser calculada en tiempo polinomial a partir de e. Un criptosistema es *asimétrico* o de *llave pública* si $d \neq e$ y calcular d a partir de e es computacionalmente [20]. Aquí d es la llave privada y e es la llave pública.

Definición 2.2 *Un* cifrado por bloques *es un sistema criptográfico en el cual el texto plano se divide en bloques de tamaño fijo y la misma función de cifrado es aplicada a cada bloque* [29].

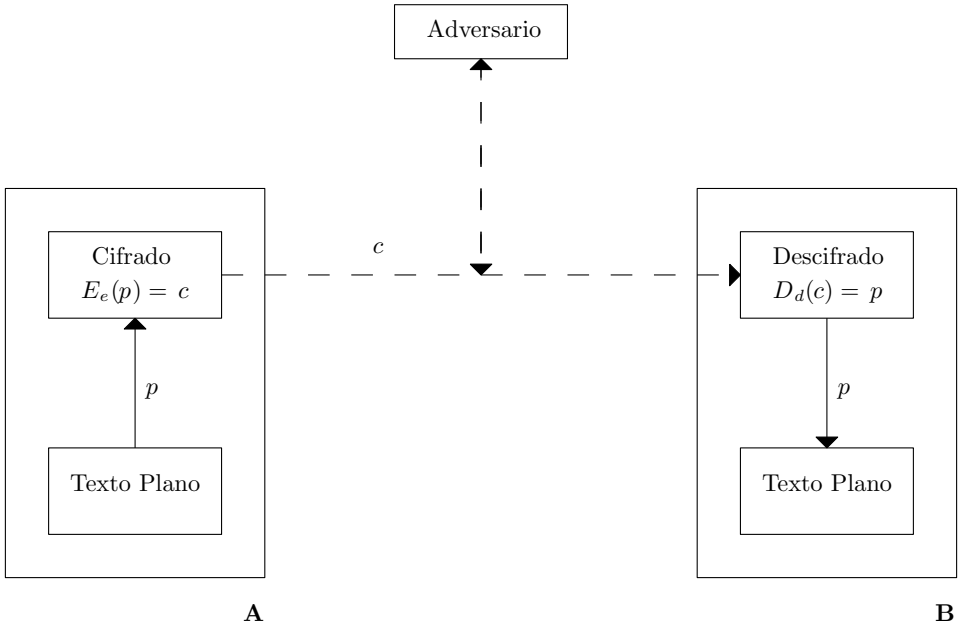

Figura 2.1: Comunicación entre dos partes usando un sistema criptográfico, $e = d$.

La Figura 2.1 es una representación gráfica de la comunicación entre un tramisor y un receptor. El transmisor **A** cifra el texto plano p utilizando la función de cifrado E, llave privada e y envía el texto cifrado c. El receptor **B** recibe el texto cifrado c, lo descifra utilizando la función de descifrado D, con llave privada d; obteniendo el texto plano p. Obsérvese que en la comunicación puede intervenir un adversario quien desea descifrar el texto cifrado.

La medida de fiabilidad criptográfica es llamada *seguridad*. La seguridad del criptosistema depende de las fuentes y las suposiciones de la información que están a la disposición del adversario. Como se indicó anteriormente, la suposición de que el adversario no conoce únicamente a el texto cifrado sino que también el criptosistema incluyendo la llave privada es común en criptología.

Romper un criptosistema es usualmente un procedimiento de recuperación del texto plano o parte de él mediante un criptograma (no necesarimente utilizando la llave privada). Esto supone que el texto plano se obtiene solamente analizando las debilidades del criptosistema.

Un intento por el adversario de romper el criptosistema es llamado *ataque* sobre el criptosistema. A continuación se enlistan los diferentes tipos de ataque:

- *Ataque por elección de texto plano.* Se supone que el adversario tiene la oportunidad de elegir el número necesario de textos planos y obtener sus criptogramas. La elección del siguiente texto plano está basada sobre todos los criptogramas vistos previamente.

- *Ataque por elección de texto cifrado.* El adversario tiene la oportunidad de elegir el número necesario de criptogramas y obtener sus correspondientes textos planos. La elección del siguiente criptograma está basada en el análisis de todos los textos planos obtenidos previamente.

- *Ataque por elección de texto.* El adversario tiene la oportunidad de elegir ambos, criptogramas (y descrifarlos) y textos planos (y cifrarlos) basado sobre el análisis de los criptogramas y textos planos obtenidos previamente.

La seguridad del criptosistema con respecto a un tipo de ataque en particular es proporcional al tiempo requerido por el adversario para romper el criptosistema con un ataque de este tipo.

Usualmente un criptosistema se considera seguro si no existe un *algoritmo*♣ que rompa el criptosistema en tiempo polinomial [20].

2.2. Matemática afín

En esta sección se revisan algunos conceptos relacionados con la teoría de campos y funciones de transformación que sirven como base para entender el modelo planteado, que se propone como posible solución de nuestro problema de seguridad. Particularmente haremos un énfasis con la *estructura algebraica*♣ de *campo* y las *transformaciones afines.*

2.2.1. Campos finitos

Definición 2.3 *La estructura algebraica* (F, +, ·) *-con operaciones de adición y multiplicación respectivamente- es llamado* campo *si satisface lo siguiente* [14]*:*

1. (F, +) *es un* grupo abeliano♣
2. $a \cdot b \in \mathrm{F}$
3. $a \cdot (b \cdot c) = (a \cdot b) \cdot c$
4. $a \cdot (b + c) = a \cdot b + a \cdot c$
5. $a \cdot e = e \cdot a = a$
6. $a \cdot b = b \cdot a$
7. $\forall\ a \neq 0,\ \exists a^{-1} \in \mathrm{F}$: $\quad a \cdot a^{-1} = a^{-1} \cdot a = e$

donde $a, b, c \in \mathrm{F}$ *y* e *representa el elemento unidad de* F *sobre la multiplicación del campo.*

Si la estructura algebraica (F, ·) cumple la propiedad número 3, entonces es un *semigrupo.* Si la estructura algebraica es un semigrupo y además con cumple las propiedades número 5 y 6, entonces es un *monoide.* Si la estructura algebraica es un monoide y además cumple la propiedad número 7, entonces es un *grupo.* Si la estructura algebraica

(F, +, ·) donde (F, +) es un grupo abeliano y (F, ∗) cumple con las propiedades 3 y 4, entonces es un *anillo*. Todo lo anterior se puede revisar en [9].

Si el número de elementos en el campo es finito, entonces el campo es llamado *campo finito*. También un campo finito es conocido como un campo de *Galois*.

Un campo *primo*, denotado como GF(q), es un campo de Galois de q elementos donde q es un número primo. De esta manera GF(q) es un conjunto {0,1,...,q-1} finito de elementos el cual es un campo sobre las operaciones adición módulo q y multiplicación módulo q. Aquí q es llamada la característica del campo. En general GF(q) no será un campo a menos que q sea un número primo. En los libros de álgebra [14] se muestra que para algún número primo q existe un campo finito de q elementos. De hecho es posible extender el campo primo GF(q) a un campo de q^m elementos el cual es llamado *la extensión del campo* y se denota como GF(q^m).

El campo *binario* $\{0, 1\}$, denotado por GF(2), tiene las operaciones mostradas en la Tabla 2.1.

Tabla 2.1: Adición + y multiplicación ×, que $\in GF(2)$.

Operadores en $GF(2)$					
+	0	1	×	0	1
0	0	1	0	0	0
1	1	0	1	0	1

2.2.2. Extensión de campos

Una extensión del campo GF(q) de grado p, se denota como GF(q^p), cosiste de un número finito de elementos q^p . El campo GF(q) y todas sus extensiones GF(q^p) tienen la misma característica q. Para el propósito de la tesis, nosostros básicamente estaremos interesados con $\mathbf{GF(2^p)}$; es decir, la característica para trabajar en la tesis será 2.

Para GF(2^p) existe un elemento α que genera el campo [14]; de tal manera que si nosotros calculamos $\{\alpha, \alpha^2, ..., \alpha^{2^{p-1}}\}$, obtendremos todos los elementos diferentes de cero de GF(2^p). También α es llamado el *generador*. El *polinomio irreducible*♣ del cual α es una raíz es llamado *polinomio generador*. Este polinomio generador también es conocido como *base polinomial*.

Así, GF(2^p) es una extensión de campo de GF(2) y consiste de 2^p elementos designados por el conjunto $\{0, 1, 2, ..., (2^p - 1)\}$. Los elementos de la extensión del campo

$\mathrm{GF}(2^p)$ pueden ser representados usando una representación polinomial como $(a_{p-1}x^{p-1}+a_{p-2}x^{p-2}+...+a_0) \in \mathrm{GF}(2^p)$, donde $a_i \in \mathrm{GF}(2)$, $i = 0, ..., (p-1)$. Ahora $< a_{p-1}, a_{p-2}, ..., a_0 >$ puede ser visto como un vector que corresponde a un único polinomio (biyección); es decir, para un valor específico de p, $< a_{p-1}, a_{p-2}, ..., a_0 >$ representa un elemento único y, donde $0 \leq y \leq (2^p - 1)$. De esta forma el polinomio $a_{p-1}x^{p-1} + a_{p-2}x^{p-2} + ... + a_0$ corresponde a y.

Definición 2.4 *Un* polinomio sobre un campo $\mathrm{GF}(q)$ *es una expresión* $f(x) = f_0 + f_1(x) + f_2(x)\cdots$, *donde los coeficientes* $f_0, f_1, ...$ *están en* $\mathrm{GF}(q)$ [14].

Si A denota a el conjunto de todos los polinomios sobre un campo $\mathrm{GF}(q)$, entonces se puede mostrar que A es la estructura algebraica de anillo bajo las operaciones de adición y multiplicación de polinomios. Por otro lado, sea $r(x)$ un polinomio irreducible de grado p sobre el campo $\mathrm{GF}(q)$. Consideremos el conjunto S de polinomios módulo $p(x)$ en A como $\{b(x) \mid grado[b(x)] < m\}$. S bajo las operaciones de adición y multiplicación módulo $r(x)$ cumple con los axiomas de campo de la Definición 2.3, la demostración se puede consultar en [14].

Por ejemplo, si consideramos a el polinomio irreducible $r(x) = x^2+x+1$ sobre $\mathrm{GF}(2)$. El conjunto de polinomios módulo $r(x)$ es un campo y se denota por $\mathrm{GF}(2^2) = \{0, 1, x, x+1\}$. Las Tablas 2.2 y 2.3 muestra las operaciones de adición y multiplicación, respectivamente.

Tabla 2.2: Adición en $\mathrm{GF}(2^2)$.

+	0	1	x	$x+1$
0	0	1	x	$x+1$
1	1	0	$x+1$	x
x	x	$x+1$	0	1
$x+1$	$x+1$	x	1	0

Tabla 2.3: Multiplicación en $\mathrm{GF}(2^2)$.

·	0	1	x	$x+1$
0	0	0	0	0
1	0	1	x	$x+1$
x	0	x	$x+1$	1
$x+1$	0	$x+1$	1	x

Existe un método alternativo para la representación de elementos de la extensión de campos donde, α -el cero del polinomio irreducible $r(x)$- es usado para generar los elementos. Por lo tanto, siguiendo el ejemplo anterior, sea α la raíz del polinomio irreducible $r(x) = x^2 + x + 1$ sobre $\mathrm{GF}(2)$, entonces:

$$r(\alpha) = \alpha^2 + \alpha + 1 \tag{2.1}$$

luego entonces,

$$\alpha^2 = \alpha + 1 \tag{2.2}$$

Esta ecuación es llamada *ecuación característica* y es la base para la estructura de la extensión del campo. Multiplicando α por ambos lados en la ecuación anterior, obtenemos

$$\alpha^3 = \alpha \cdot \alpha^2 = \alpha(\alpha + 1) = \alpha^2 + \alpha = \alpha + 1 + \alpha = 1 \tag{2.3}$$

El conjunto $\{0, \alpha, \alpha^2, \alpha^3\}$ es un *grupo cíclico*♣ generado por α. El conjunto también puede ser representado como $\{0, 1, \alpha, \alpha^2\}$. Las tablas que corresponden a las operaciones de adición y multiplicación para la estructura algebraica pueden ser generadas sustituyendo x por α y $x + 1$ por $\alpha^2 = \alpha + 1$.

2.2.3. Transformaciones afines

Definición 2.5 *Sea* V *un conjunto no vacío y sea* (F, +, ·) *un campo. Se dice que* V *es un* espacio vectorial *sobre el campo* F *si satisface lo siguiente* [14]*:*

1. (V, +) *es un grupo abeliano*
2. $a(u + v) = au + av$
3. $(a + b)u = au + bu$
4. $(ab)u = a(bu)$
5. $1u = u$

donde $a, b \in$ F *y* $u, v \in$ V *y donde* 1 *representa el elemento unidad de* F *sobre la multiplicación del campo.*

La Definición 2.5 funciona para cualquier campo [14]. Por ejemplo, el espacio vectorial $\mathbb{R}^n$ sobre el campo $\mathbb{R}$, el espacio vectorial $\mathbb{C}^n$ sobre el campo $\mathbb{C}$. En particular nosotros trabajaremos con campos de Galois, por lo tanto todo campo finito de la forma GF(2^p) es un espacio vectorial sobre GF(2).

Definición 2.6 *Sean* V *y* W *espacios vectoriales sobre el campo campo* F*. Una función* $f : \mathrm{V} \longrightarrow \mathrm{W}$ *es una* transformación lineal [14] *si para todo* $x, y \in \mathrm{V}$ *y para todo* $\alpha \in \mathrm{F}$*, cumple las siguientes condiciones:*

$$f(x + y) = f(x) + f(y)$$

$$f(\alpha x) = \alpha f(x)$$

si f *no cuple con las condiciones de la Definición 2.6, entonces* f *es una* transformación no lineal*.*

Definición 2.7 *Sean* V *y* W *espacios vectoriales sobre el campo* F*. Una función* $T : \mathrm{V} \longrightarrow \mathrm{W}$ *es una* transformación afín [14] *si para todo* $u \in \mathrm{V}$*, cumple lo siguiente:*

$$T(u) = f(u) + t$$

donde f *es una transformación lineal que poseé inversa y* $t \in \mathrm{V}$ *es fijo. Si* T *no es afín, entonces* T *es llamada* **transformación no afín***.*

Notar que si dada una función, la función es no lineal, entonces la función es no afín.

2.3. Autómatas Celulares

2.3.1. Idea básica

¿Qué son los Autómatas Celulares (AC)? La palabra "*celular*" significa "*que consiste de células*", entonces un autómata celular está hecho de células. Cada una de esas células contiene un "*autómata*", una máquina de estados finitos. La construcción que resulta es un espacio lleno de células, cada una de esas células contiene una máquina de estados finitos. Usualmente el espacio celular está dividido en una malla o matriz, también conocido como "*lattice*". Existen varios tipos de lattice y estructuras de *vecindad*. Ejemplo: para un AC 2-dimensional, la Figura 2.2 b) muestra una lattice cuadrada. Un espacio celular 2-dimensional se ve como una pieza de papel cuadriculado -una pieza de papel cuadriculado infinitamente grande-. Los espacios celulares puden tener cualquier dimensión. En la Figura 2.2 podemos observar 1, 2 ó 3 dimensiones.

La idea básica de los AC hace referencia a los sistemas dinámicos discretos y sirve para simular sistemas complejos por la interacción de células siguiendo funciones simples. El comportamiento complejo emerge de la iteración de componentes simples siguiendo reglas simples.

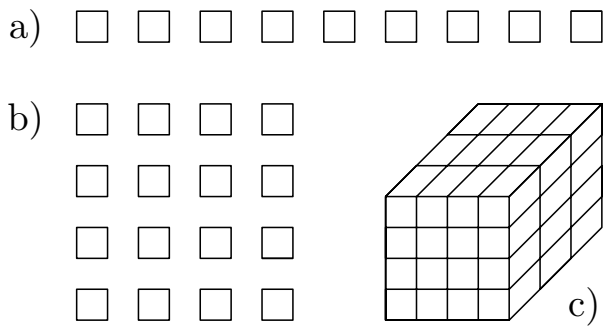

Figura 2.2: Lattice cuadrada en: a) uno, b) dos y c) tres dimensiones.

2.3.2. Historia

La historia de los AC gira alrededor de tres eventos o periodos de interés. El primero de esos fue la "*auto-reproducción*" de John von Neumann, el segundo "*el juego de la vida*" de Conway, el tercero, la "*clasificación de AC*" de Stephen Wolfram. El concepto de AC está asociado con Jhon von Neumann, los detalles de la construcción de un dispotivo autoreproductor permanecieron sin publicar hasta su muerte en 1957. Pero, después, fueron publicados por A. W. Burks. Stanislaw Marcin Ulam sugirió a von Neumann usar un AC para su investigación. El conocimiento público de los AC es atribuído al británico Jhon Horton Conway, él desarrolló un espacio celular que se llama *juego de la vida*. El juego de la vida exhíbe la computación o construcción universal. En la decada de los 80's Wolfram publicó varios artículos sobre el comportamiento de los AC lineales y propusó una clasificación en base al comportamiento de los AC.

2.3.3. Características de un AC

Una célula es una región del espacio que puede tomar diferentes valores llamados *estados*. Los estados evolucionan dependiendo del valor de un cierto número de vecinos (la vecindad), siguiendo una regla de evolución [19]. La *regla de evolución* o *transición* nos indica a que estado pasa una célula de una generación (iteración) a la siguiente dependiendo de su estado actual y de su vecindad.

Las células tienen el mismo número de estados. Si puede tener un valor entre dos posibles se llama binario, entre tres se llama ternario y así sucesivamente [19]. Cada célula puede tener una regla de evolución o bien todas las células pueden tener la misma regla de evolución.

La *vecindad simple* consiste de una célula y sus dos células vecinas más cercanas, esto es, la iteración se realiza a primeros vecinos [15]. En general, vamos a considerar una vecindad con r (*radio*) vecinos en cada lado. La Figura 2.3 muestra la vecindad simple para un AC 1-dimensional, se observa la célula i junto con sus vecinos izquierdo $i-1$ y derecho $i+1$.

Figura 2.3: Células involucradas en la vecindad simple de un AC 1-dimensional.

La Figura 2.4 muestra las vecindades de radio igual a uno utilizadas más comunmente en un AC 2-dimensional. Incluye la vecindad de von Neumann a), la cual consiste de cuatro células que están horizontalmente y verticalmente adycentes a la célula central de color negro y la vecindad de Moore b), la cual consiste de 8 células [22].

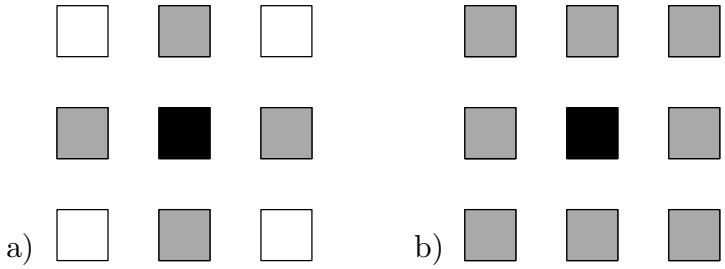

Figura 2.4: Las vecindades de a) von Neumann y b) Moore con radio $r = 1$.

Wolfram clasificó AC lineales en cuatro clases [30]:

- Clase 1. Los AC envuelven estados finales homogeneos.
- Clase 2. Los AC producen comportamientos periódicos que se repiten.
- Clase 3. Los AC exhiben comportamietos caóticos.
- Clase 4. Los AC desarrollan patrónes de forma inestable.

La notación de Wolfram para un AC lineal es (k, r) donde k es el número de estados de cada célula, r es el número de células en cada lado de la vecindad.
Dado un AC (k, r), se tiene que:

- Cada vecindad tiene longitud $n = 2r + 1$.
- Existen k^{2r+1} posibles vecindades diferentes.
- Existen $k^{k^{2r+1}}$ posibles reglas de evolución.

Por lo que para un AC (2,1), se tiene una vecindad de 3 células, cada una de las cuales está en uno de dos estados. Existen 8 posibles vecindades y el número total de reglas posibles para ellas es de 256.

Para poder enumerar las vecindades de (2,1) se elige un número binario de 8 dígitos. También se puede hacer referencia al número de regla por su equivalente en decimal. Cada regla de evolución obtiene su propio número de serie en el proceso. La Tabla 2.4 muestra cómo las vecindades de 3 células evolucionan por la regla 22 de Wolfram.

Tabla 2.4: Regla 22 de Wolfram.

Vecindad	$\longrightarrow$	111	110	101	100	011	010	001	000
Regla 22	$\longrightarrow$	0	0	0	1	0	1	1	0

El trabajo de la regla de evolución puede observarse en la Figura 2.5, son diez generaciones de la regla de evolución 22, con elección de configuración inicial aleatoria. ¿Qué pasa con la primer y última célula de cada fila? La primer célula de cada fila es el vecino derecho de la última célula, al igual que la última célula es el vecino izquierdo de la primer célula. La evolución está representada por los nuevos valores que adquieren las células, lo cual es representado por los renglones de la matriz mostrada en la Figura 2.5.

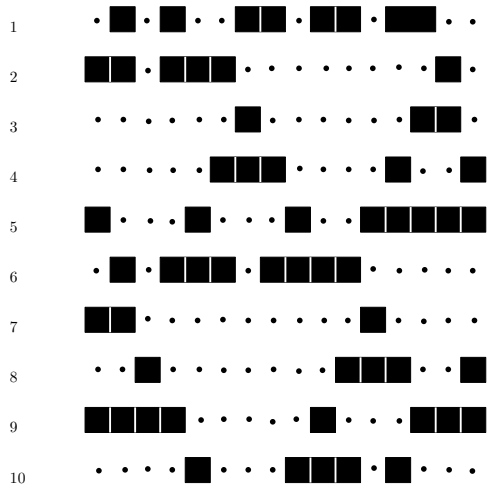

Figura 2.5: Evolución de la regla 22 con una configuración inicial aleatoria.

Todo lo anterior se puede formalizar mendiante la Definición 2.8.

Definición 2.8 Un AC es una 4-tupla $(d, \mathcal{S}, \mathcal{H}, \mathcal{F})$ [19], donde:

1. $d \in \mathbb{N}$ *es la dimensión.*

2. $\mathcal{S}$ *es un conjunto finito de elementos llamados estados y es denotado por:*

   $\mathcal{S} = \{s_k : k \in \{0, ..., |s| - 1\}\}$.

3. $\mathcal{H} \subset \mathcal{S}^{\mathbb{Z}^d}$*, es subconjunto finito llamado vecindad y es denotado por:*

   $\{V_j = \{X_{1,j}, ..., X_{d,j} : j \in \{1, ..., |\mathcal{H}|\}\}$,
   donde los elementos son llamados vectores vecindad.

4. $\mathcal{F} : \mathcal{S}^{|\mathcal{H}|} \longrightarrow \mathcal{S}$*, es llamada la función de evolución o transición.*

El juego de la vida de Conway [4] es un ejemplo de un autómata binario bidimensional cuya vecindad de nueve células, está formada por la célula central y las ocho células que le son adyacentes.

2.3.4. AC reversible

Un *autómata reversible* es aquel, para el cual existe otra regla de evolución que permite observar la evolución del autómata en reversa; es decir, se puede aplicar sobre la configuración del autómata y regresar los pasos de la evolución hasta el estado inicial y aún más atrás [8].

Sabemos que las células que pertenecen a un AC cambian sus estados síncronamente en etapas de tiempo discreto. El estado siguiente de cada célula depende del estado actual de las células vecinas según la regla de evolución. Se supone que todas las células usan la misma regla y la regla es aplicada a todas las células al mismo tiempo. Según la Definición 2.8 la regla de evolución local de un AC es una función $\mathcal{F} : \mathcal{S}^{|\mathcal{H}|} \longrightarrow \mathcal{S}$ donde $|\mathcal{H}| = n$ es el tamaño del vector vecindad. El estado $\mathcal{F}(a_1, a_2, ..., a_n)$ es un nuevo estado de células cuyos n vecinos fueron estados $(a_1, a_2, ..., a_n)$ en una etapa de tiempo anterior. Esta regla de actualización determina una dinámica global del AC.

Las células son direccionadas como elementos de $\mathbb{Z}^d$, donde d es la dimensión del AC. Una *configuración* del AC es una función $c : \mathbb{Z}^d \longrightarrow \mathcal{S}$. El conjunto de todas las configuraciones se denota por $\mathcal{C}$. Una configuración c después de una etapa de tiempo evolucionará a la configuración e. Decimos que $e = \mathcal{G}(c)$ y llamamos a $\mathcal{G} : \mathcal{C} \longrightarrow \mathcal{C}$ *la función de transición global* de un AC.

Generalmente identificamos a un AC con su función de evolución global $\mathcal{G}$ y hablamos de la función $\mathcal{G}$ del AC o simplemente autómata celular $\mathcal{G}$.

Jarkko Kari demostró en [18] que un AC es *inyectivo (sobreyectivo)* si y sólo si su función $\mathcal{G}$ es *uno a uno* (*sobre*, respectivamente). El AC es *biyectivo* si $\mathcal{G}$ es sobre y uno a uno. Por lo tanto un AC con función global $\mathcal{G}$ es llamado *reversible* o *invertible* si existe un AC con función global $\mathcal{F}$ tal que $\mathcal{G} \circ \mathcal{F} = id$, donde id es la función identidad. Entonces $\mathcal{F}$ y $\mathcal{G}$ son autómatas inversos uno del otro.

2.4. Complejidad computacional

Un problema de decisión, es un problema en el cual la solución del problema es la respuesta "sí" o "no". En la teoría formal de la computación un problema de decisión $S \subseteq \{0,1\}^n$ es soluble en tiempo polinomial si existe una máquina de Turing M, que termina en tiempo polinomial, tal que $M(x) = 1$ si y sólo si $x \in S$. La clase de problemas de decisión que son solubles en tiempo polinomial es denotada por la clase P [12].

Ahora, supóngase que se tiene el problema de saber si el número N es un número compuesto. Si N es compuesto, existe una "prueba corta" de este hecho. Tal prueba consiste de un factor de N y es fácil verificar que esta prueba es correcta. Es fácil hacer un algoritmo M que tome al par (N, K) de enteros positivos y verifique en tiempo polinomial si K es un factor de N. Si N es primo, entonces $M(N, K) = 0$ para todo K, mientas que si N es compuesto, siempre existirá un entero K tal que $M(N, K) = 1$. Más aun, la cadena que codifica a K será tan larga como la cadena que codifica a N.

Definición 2.9 *Un problema de decisión $S \subset \{0,1\}^n$ pertenece a la clase NP si existe un subconjunto $R \subset \{0,1\}^n \times \{0,1\}^n$, con las siguientes propiedades* [12]*:*

1. Existe una función p tal que $|y| \leq p(|x|)$ siempre que $(x, y) \in R$

2. $x \in S$ si y sólo si existe algún y tal que $(x, y) \in R$.

3. El problema de determinar si el par $(x, y) \in R$ está en P.

Cuando y existe, es llamado una "prueba" del hecho de que $x \in S$. El algoritmo que determina en tiempo polinomial si el par $(x, y) \in R$, es llamado procedimiento verificador.

Definición 2.10 *Un problema de decisión S es* NP-completo [12] *si*

1. $S \in NP$

2. Para todo problema que está en NP, es reducible a S en tiempo polinomial.

Para mostrar que S está en NP basta con probar que una solución de S se puede verificar en tiempo polinomial. Notar que si un problema S únicamente cumple con la condición número dos de la Definición 2.10, entonces se dice que S es un problema $NP - hard$.

Capítulo 3

Estado del Arte

En este capítulo se revisan algunos trabajos relacionados con el tema de esta tesis. A manera de resumen, se platica la metodología usada por los autores y las ideas tomadas de los trabajos para la construcción del modelo planteado. Uno de los pioneros en usar autómatas celulares para construir sistemas criptográficos fue Stephen Wolfram. Él usó una regla reversible para un autómata 1-dimensional para su criptosistema. Manuel Martínez en su tesis de maestría del CIC IPN, usa autómatas celulares reversibles 2-dimensional para construir un criptosistema de llave pública. Finalmente S. Nandi y P. Pal Chaudhuri usando ideas de autómatas sobre campos finitos construye un criptosistema de llave privada.

3.1. Criptografía con AC

Desde la aparición de los AC, han sido considerados para aplicaciones en criptografía. Dado que los AC son modelos muy simples que pueden generar patrónes pseudoaleatorios, ha habido grandes esfuerzos para usarlos en criptografía. Stephen Wolfram en [31] con un AC(2,1) periódico, uniforme de n células, donde la i-ésima célula está representada por a_i y función de transición:

$$a_i = a_{i-1}\ XOR\ (a_i\ OR\ a_{i+1}), \tag{3.1}$$

o también,

$$a_i = (a_{i-1} + a_i + a_{i+1} + a_i a_{i+1})\ mod\ 2, \tag{3.2}$$

desarrolló un criptosistema con *cifrado por flujo*♣.

El estado inicial del AC es usado como llave del criptosistema. El valor de una célula a en particular, obtenida a través del tiempo puede servir como una secuencia aleatoria. El texto cifrado $\mathcal{C}$ se obtiene a partir de un texto plano binario $\mathcal{P}$.

$$\mathcal{C}_i = \mathcal{P}_i \; XOR \; a^{(i)} \tag{3.3}$$

El texto plano $\mathcal{P}$ se puede recuperar, con la misma operación, pero sólo si la secuencia $a^{(i)}$ es conocida. Los AC, tales como el usado en este trabajo, han sido investigados en estudios sobre el origen de la aleatoriedad en sistemas físicos [32].

La seguridad del criptosistema depende de la dificultad de encontrar la semilla de los valores de las células en una secuencia de tiempo. Este problema pertenece a la *clase NP* [31], además si quisiéramos hacer estadística con los resultados del criptosistema, no se encontrarían regularidades más cortas que el número de células del AC.

3.2. Cifrado usando la regla 60 de AC aditivos

Anteriormente se mostró una tabla donde se describe una lista de reglas sobre los AC aditivos. En particular en [33] Stephen Wolfram usó la regla 60 de un AC aditivo; es decir, AC sobre GF(2) para cifrar. Wolfram usó una secuencia de células como la representación de la regla 60.

La Figura 3.1 muestra la manera de cifrar el texto plano, representado por secuencias de células (cuadros blancos y negros) y el método se basa en tener una secuencia de células de cifrado, columna b). El texto plano $\mathcal{P}$, representado por la secuencia a), se convertirá en el texto cifrado $\mathcal{C}$ representado por la secuencia c), usando la secuencia de cifrado.

Las células que representan el texto plano se invierten de color si su correspondiente célula de cifrado es una célula negra, en otras palabras si tengo una célula blanca y su célula de cifrado es negra, entonces la célula cifrada será de color negro, de lo contrario la célula permanecerá blanca.

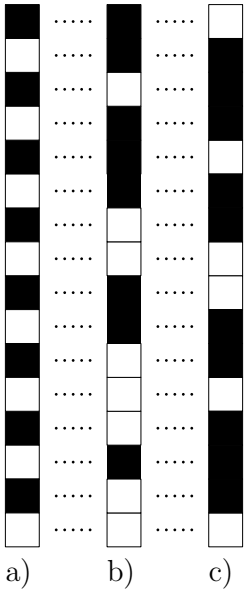

Figura 3.1: Esquema de cifrado: a) texto plano, b) secuencia de cifrado y c) texto cifrado.

Ahora bien, suponiendo que se recibe el texto cifrado c), ¿cómo podemos recuperar el texto plano a) correspondiente al texto cifrado? Si la secuencia de cifrado b) es conocida entonces descifrar el texto cifrado es fácil, es suficiente con repetir el proceso de cifrado e invertir el color de cada célula dependiendo su correspondiente célula en la secuencia de cifrado.

Por otro lado para poder cifrar necesitamos la secuencia de cifrado. La Figura 3.2 muestra la forma de generar las secuencias de cifrado. Usando un autómata celular aditivo de una dimensión, con la regla de evolución 60. Adicionalmente se toma una llave privada como su estado inicial y generamos las secuencias de cifrado.

El texto plano $\mathcal{P}$ se encuentra en la primer columna, mientras que su correspondiente texto cifrado $\mathcal{C}$ usando la secuencia de cifrado mostrada en la columna central, se muestra en la columna final. El AC itera 36 veces usando como estado inicial la llave privada, que para este ejemplo es igual a la secuencia 1000000000000.

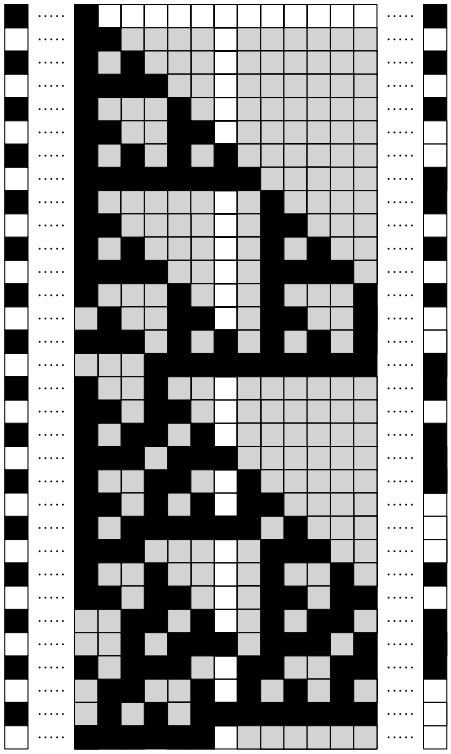

Figura 3.2: Cifrado usando regla 60 de un AC aditivo.

3.3. Criptosistema de llave publica con AC

Manuel Martínez en su tesis de maestría [13] del Centro de Investigación en Computación del I.P.N, hace uso de autómatas celulares 2-dimensional reversibles, para construir un criptosistema de llave pública basado en la composición de autómatas celulares reversibles. Debido a las propiedades matemáticas y su facilidad de implementación, este tipo de autómatas son candidatos para ser usados en el diseño de criptosistemas y ser altamente resistentes ante los distintos tipos de ataques por parte de criptoanalistas en su intento por romper la seguridad del criptosistema. La seguridad del criptosistema cae en la complejidad computacional de calcular la llave privada a partir de los autómatas celulares reversibles.

Manuel usó el resultado de Jarkko Kari en [17], el cual dice que revertir un AC 2-dimensional es un problema indecidible en tiempo polinomial, para construir la llave privada de su criptosistema. Más aún, su trabajo utiliza el resultado de [6], el cual dice que revertir un AC de dos dimensiones es un problema NP-*hard*. Formalmente el fundamento matemático se basa en los siguientes resultados:

Teorema 3.1 *Dado un AC reversible de dos dimensiones es indecidible invertir el AC* [17]*. Esto se cumple incluso cuando el AC tiene la restricción de usar la vecindad de Von Neumann.*

Corolario 3.1 *Para cada algoritmo que encuentre la inversa de todos los AC reversibles de dimensión dos, dados como entrada y para cada función computable f, existe un AC reversible $\mathcal{A}$ tal que el algoritmo no encuentra la inversa de $\mathcal{A}$ en tiempo f(s), donde s es el tamaño de $\mathcal{A}$* [17].

Ahora, considere el problema de invertir un AC 1-dimensional como INVERTIRAC$_1$ y también para el caso de un AC 2-dimensional como INVERTIRAC$_2$, los cuales calculan la función $f : \{0,1\}^N \longrightarrow \{0,1\}^N$ y la cadena $y \in \{0,1\}^N$, donde la salida debe ser la preimagen x de y bajo la función f.

Proposición 3.1 *El peor caso de intratabilidad de la inversión de un AC* (INVERTIRAC) *se resume como sigue* [6]*:*

1. INVERTIRAC$_1$ puede ser resuelto en tiempo polinomial.

2. INVERTIRAC$_2$ es NP-hard.

El proceso de cifrado y descifrado está basado en la composición de n AC reversibles. Por ejemplo, la composición de n AC reversibles mostrada en la Ecución 3.4 funciona para cifrar el texto plano $\mathcal{P}$ y obtener el texto cifrado $\mathcal{C}$. La Ecuación 3.5, a partir de la composición de n AC reversibles funciona para descifrar el texto cifrado $\mathcal{C}$ y obtener el texto plano $\mathcal{P}$.

$$\mathcal{C} = ACR_n(\cdots ACR_2(ACR_2(\mathcal{P}))) \tag{3.4}$$

$$\mathcal{P} = ACR_1(\cdots ACR_{n-1}(ACR_n(\mathcal{C}))) \tag{3.5}$$

3.4. Criptosistema con AC sobre campos finitos

Sen, Shaw, Chowdhuri, Ganguly y Chaudhuri en [11] usaron AC sobre campos finitos para construir un criptosistema de llave privada con cifrado por bloques. El nombre del sistema criptográfico es "Criptosistema basado en AC". Ellos emplean una serie de transformaciones para aumentar el nivel de seguridad del criptosistema.

Dos criptosistemas; uno basado en AC con cifrado por bloques; y otro con cifrado por flujo, fueron presentados en [26], pero ambos fueron rotos con facilidad dado que presentan una propiedad de afinidad en los Autómatas Celulares [27]. En [11] presentan un criptosistema donde los los AC son arreglados con tranformaciones no afines para eliminar la propiedad de afinidad y hacer un cifrado por bloques, con tamaño de bloque y tamaño de llaves de 128 bits..

El Criptosistema está basado en AC 1-dimensional de 16 células, donde el valor de cada célula pertenece a GF(2^8). Con función de evolución f_i, i =1,2,...,16, sobre $GF(2^8)$.

Hay dos autómatas celulares usados en el criptosistema; uno de ellos genera una secuencia de cifrado a partir de la llave privada $\mathcal{K}$. A partir de esta secuencia de cifrado el texto plano $\mathcal{P}$ es cifrado y se obtiene el texto cifrado $\mathcal{C}_i$. Adicionalmente el segundo AC es iterado usando como estado inicial el texto cifrado $\mathcal{C}_i$, donde el número de iteraciones de este AC está especificado por la secuencia de cifrado del primer AC.

Feng Bao en [7] hace criptoanálisis del criptosistema y logra romper el sistema en tiempo polinomial. La debilidad del criptosistema radíca en que los AC que emplean son 1-dimensional y según Jarkko Kari en [18] un AC reversible consta de ciclos. Debido a que los AC que usan en este trabajo tienen estructura cíclica, luego entonces se puden ver como AC reversibles. Por lo tanto según Jarkko Kari en [17] existe un algoritmo en tiempo polinomial para invertir un AC 1-dimensional.

Capítulo 4

Propuesta de Solución

En el presente capítulo proponemos un modelo para simular un criptosistema de llave privada con cifrado por bloques. El criptosistema está basado en dos clases de AC: un AC 1-dimensional sobre campos finitos y un AC 2-dimensional reversible. Incorporamos varias transformaciones para aumentar la seguridad del criptosistema.

4.1. Modelo

En el capítulo uno del presente trabajo, se propuso como objetivo el diseñar e implementar un criptosistema de llave privada con cifrado por bloques de 128 bits, usando autómatas celulares de una dimensión y dos dimensiones, incorporando cuatro niveles de transformación a cada bloque. Para ello se tomaron las ideas propuestas en los siguientes modelos:

- ◇ Del trabajo de Stephen Wolfram en [31] se considera la idea de tener un AC que genere estados pseudoaleatorios con el objetivo de que este AC tome la llave privada K como estado inicial y genere un estado S_N que servirá para controlar los cuatro niveles de transformación.

- ◇ De Manuel Martínez en [13] se toma la idea de tener un AC 2-dimensional reversible en un criptosistema. Revertir un AC 2-dimensional reversible es un problema que pertence a la clase NP, según los resultados de Jarkko Kari [17] y [6]. Por lo tanto, usamos un AC de dos dimensiones reversible para el segundo nivel de transformación.

- ◇ De Sen, Shaw, Chowdhuri, Ganguly y Chaudhuri en [11] empleamos la idea de implementar los AC sobre campos finitos. En particular los AC utilizados en este trabajo, tendrán como conjunto de estados al campo finito GF(2^8). Las funciones de evolución de los AC operan sobre la aritmética del campo GF(2^8).

El criptositema tiene cuatro niveles de transformación, se pueden observar en la Figura 4.1 como se elabora cada transformación. Se aplica una transformación *lineal* y una transformación *afín* en los primeros dos niveles, mientras que en el tercer nivel se aplica una transformación *no afín* para alcanzar un alto nivel de seguridad. El cuarto nivel se encarga de mezclar la llave.

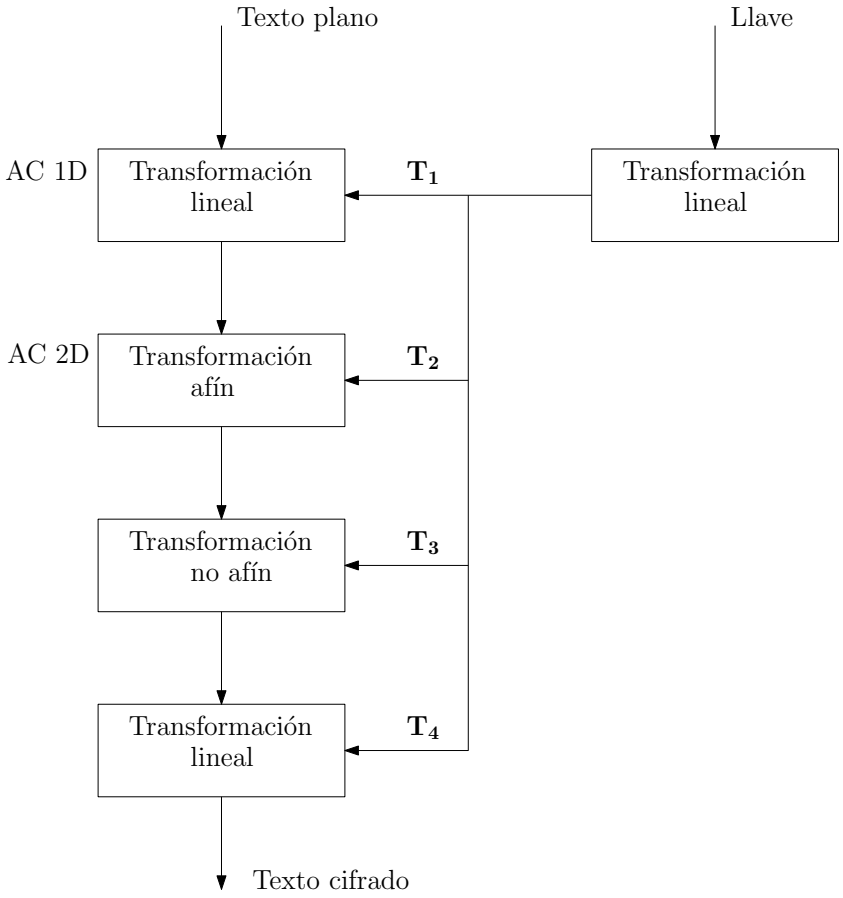

Figura 4.1: Diseño del sistema criptográfico.

La Figura 4.2 muestra el modelo para simular el criptosistema. Los cuatro niveles de transformación son representados con *Nivel 1, 2, 3* y *4*. En la Figura 4.2 muestra diferentes estados de computación que son representados con (I), (II), (III), (IV), (V) y (VI).

Como se puede observar en el modelo, la flecha punteada indica el proceso de descifrado, mientras que la flecha normal indica el proceso de cifrado. Los recuadros punteados representan los niveles de computación, el modelo tiene seis niveles de computación, según se observa en la Figura 4.2.

En el primer nivel de computación, el AC 1-dimensional toma como estado inicial a la llave privada K, después de que el AC es iterado d número de veces, se obtiene al estado S_N. En el segundo nivel, el estado S_N proporciona parámetros para controlar las cuatro transformaciones del criptosistema. A partir del estado S_N obtenemos un vector $< \delta_1, ...\delta_{|S_N|} >$, este vector servirá para rotar los bytes de los bloques del texto plano en la primera transformación, esto en el tercer nivel de computación. Más adelante describimos el algoritmo para calcular el vector de los valores δ_i.

En el cuarto nivel se transforman los bloques de texto plano usando una transformación afín, el resultado de transformar el bloque será el estado inicial del AC 2-dimensional. Éste autómata será iterado un número Δ de veces, donde Δ se obtiene del estado S_N en el segundo nivel de computación.

En el quinto nivel de computación los bloques del texto plano son transformados usando una transformación no afín, dependiendo del resultado de una función de evaluación. Finalmente en el sexto nivel de computación el estado S_N es mezclado usando la adición sobre el campo GF(2^8) con el bloque al efectuar la transformación no afín.

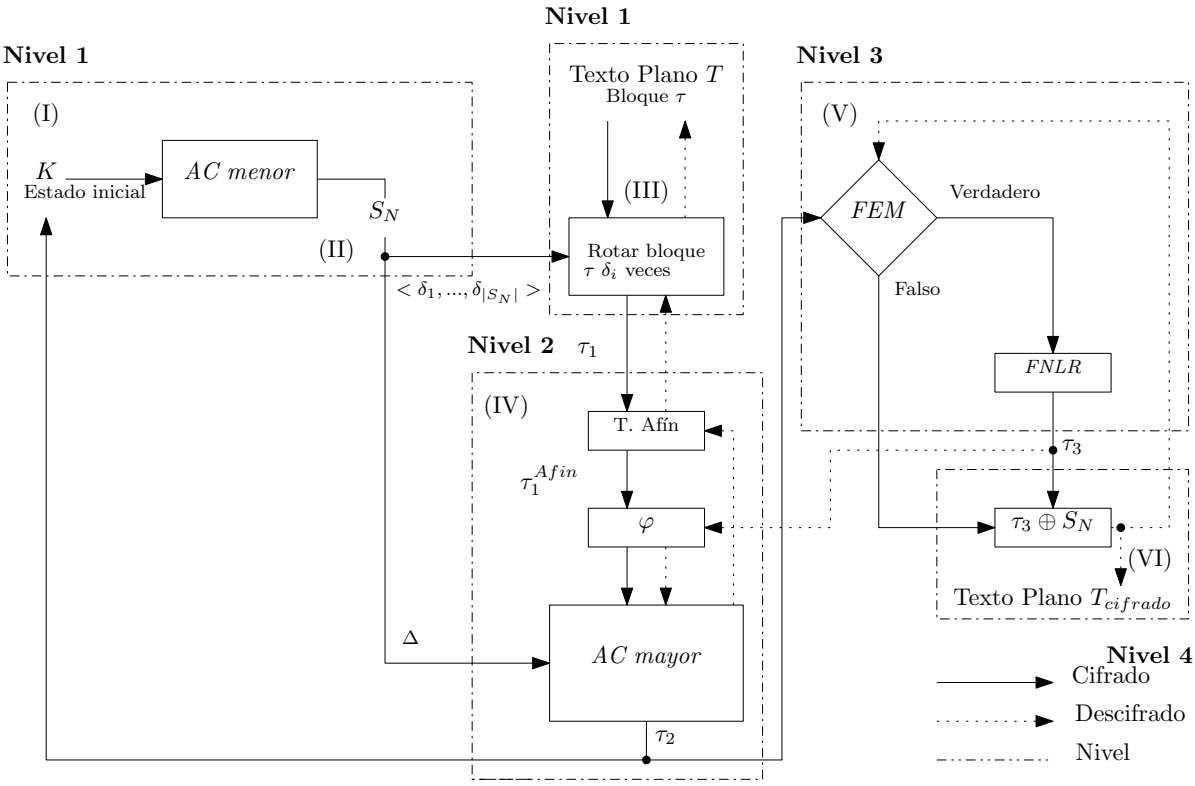

Figura 4.2: Cifrado y descifrado.

El criptosistema está basado en dos clases de AC. El primer AC es un AC 1-dimensional de 16 células sobre GF(2^8), este autómata está construído a partir de cuatro autómatas celulares sobre GF(2^8) de 4 células donde el AC es un AC grupo de tamaño máximo. El segundo AC es un AC 2-dimensional reversible con una retícula cuadrada de 4×4.

Debido a que cada una de las células es un polinomio de grado a lo más ocho y cada polinomio puede ser representado usando ocho bits o un byte, el criptosistema puede cifrar $16 \times 8 = 128$ bits a la vez. Por lo tanto el tamaño del bloque (τ) y de la llave privada K es tomado como 128 bits. Si quisiéramos cambiar el tamaño del bloque y de la llave, debemos ajustar el número de células del AC o el valor de p en GF(2^p).

Las operaciones de *cifrado* y *descifrado* se describen a continuación, cada paso se explica con la ayuda de la Figura 4.2, además describimos cada uno de los componentes del modelo más a detalle; es decir, explicamos a profundidad los componentes de los dos autómatas, así como los cuatro niveles de transformación.

4.1.1. AC 1-dimensional sobre GF(2^8)

El AC 1-dimensional interviene en el primer nivel de transformación, a continuación describimos la transformación que lleva acabo el AC, además de sus componentes. Los elementos del campo finito GF(2^8) son generados usando el polinomio generador $r(x)$ de la Ecuación 4.1. Los elementos de este campo forman al conjunto de estados $\mathcal{S}_1$ del autómata celular.

Nivel 1 - Transformación lineal sobre la llave. La llave (K) es usada como estado inicial del AC 1-dimensional que lleva por nombre **AC menor**. Este autómata es un AC lineal sobre $GF(2^8)$ de 16 células acotado, donde el autómata está formado por cuatro autómatas celulares de cuatro células. Cada uno de los cuatro AC tienen matriz característica T (Ecuación 4.2) y el polinomio irreducible sobre GF(2^8) está dado por $r(x)$ en la Ecuación 4.1.

La matriz companion asociada a el polinomio generador α, cuyo polinomio característico es el polinomio irreducible $r(x)$ que genera la extesión del campo GF(2^8), está dada por la Ecuación 4.3.

$$r(x) = x^8 + x^6 + x^3 + x^2 + 1 \quad (4.1)$$

$$T = \begin{pmatrix} 170 & 1 & 0 & 0 \\ 103 & 145 & 1 & 0 \\ 0 & 23 & 150 & 1 \\ 0 & 0 & 176 & 76 \end{pmatrix} \quad (4.2)$$

$$\alpha = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{4.3}$$

$$170 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{4.4}$$

$$103 = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \tag{4.5}$$

$$145 = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{4.6}$$

$$23 = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \end{pmatrix} \tag{4.7}$$

$$76 = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \end{pmatrix} \tag{4.8}$$

$$176 = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \tag{4.9}$$

$$150 = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{4.10}$$

Los AC grupo sobre GF(2^p) obedecen la regla de evolución 4.11, según se puede ver en el **Apéndice A**.

$$q_i(t+1) = \phi(w_{i-1}q_{i-1}(t), w_i q_i(t), w_{i+1}q_{i+1}(t)) \tag{4.11}$$

Por lo tanto, los cuatro AC obdecen a los siguientes vectores regla, siguiendo la Ecuación 4.11. El polinomio característico $p_T(x)$ de la matriz T está dado por Ecuación 4.13. El polinomio contiene información acerca de la estructura cíclica de los AC.

$$< 0, 170, 1 > \ < 103, 145, 1 > \ < 23, 150, 1 > \ < 176, 76, 0 > \tag{4.12}$$

$$p_T(x) = x^4 + 225x^3 + 27x^2 + 83x + 192. \tag{4.13}$$

Ahora bien, debido a que el AC está formado por 4 AC sobre GF(2^8) de 4 células, donde los AC poseen la misma matriz característica T al igual que el mismo polinomio característico $p_T(x)$ y además éste polinomio es primitivo, implica que la estructura cíclica está formada por cuatro cíclos de tamaño máximo 4.14:

$$\Big[4(1), 4(2^{32} - 1)\Big]. \tag{4.14}$$

La Figura 4.3 muestra la estructura de cada uno de los cuatro autómatas según las reglas de evolución dadas por la Ecuación 4.12.

El *AC menor* evoluciona un número fijo de iteraciones d para cada bloque. El estado inicial S_0 del AC es la llave K (ver I en la Figura 4.2). Para cada bloque sucesivo el AC genera un nuevo estado llamado *estado* S_N después de iterar el AC d veces desde su estado actual (ver II en la Figura 4.2). El estado S_N se utiliza para cuatro diferentes propósitos:

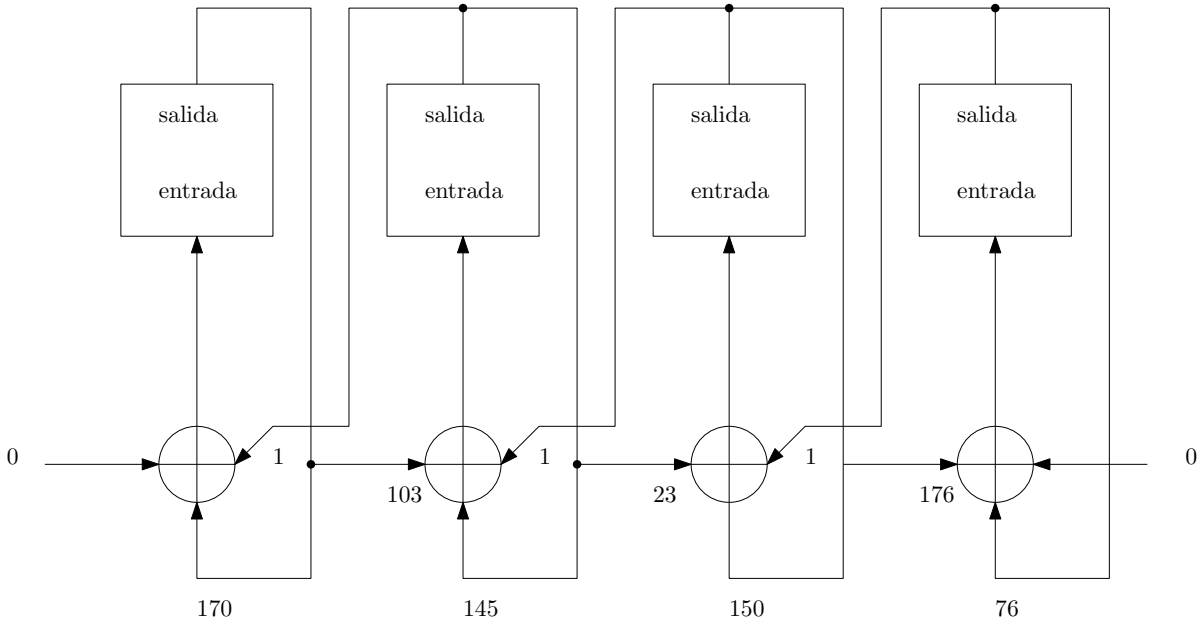

Figura 4.3: Estructura del AC de cuatro células sobre GF(2^8).

◇ Proporciona el vector de valores $< \delta_1, ..., \delta_{|S_N|} >$ con el cual cada uno de los bytes del bloque de entrada τ es rotado.

◇ Proporciona el estado que será usado en el segundo nivel de transformación.

◇ Proporciona el número Δ de iteraciones para el AC 2-dimensional reversible.

◇ Proporciona un operando para el cuarto nivel de transformación.

Nivel 1 - Transformación lineal sobre el bloque. La transformación lineal sobre el bloque τ se lleva acabo rotanto δ_i veces cada byte b_i del bloque τ, generando el bloque τ_1 (ver III en la Figura 4.2). En la operación de descifrado el bloque generado por el AC 2-dimensional reversible está sujeto a la misma operación en sentido opuesto. El Algoritmo 1, muestra como obtener el vector $< \delta_1, ..., \delta_{|S_N|} >$ a partir del estado S_N. Cada uno de los valores del vector estará dentro del intervalo 0 a 7.

Algoritmo 1 Calcular vector $< \delta_1, ..., \delta_{|S_N|} >$

Entrada: S_N
Salida: $< \delta_1, ..., \delta_{|S_N|} >$
1: **para** $i = 1$ hasta $|S_N|$ **hacer**
2: Obtener la representación binaria del elemento S_{Ni}
3: Convertir el valor a decimal y obtener el módulo 8
4: **fin para**

4.1.2. AC 2-dimensional reversible

Nivel 2 - Transformación afín. Usamos una transformación afín para transformar el bloque τ_1. El resultado de la transformación será usado como estado inicial por el AC 2-dimensional reversible $\mathcal{A}$ que lleva por nombre **AC mayor**, donde el vector vecindad $\mathcal{H}$ podrá ser el vector vecindad de radio $r = 1$ de *Von Neumann* o el vector vecindad de *Moore*.

$$\mathcal{H} = (x_1, x_2, ..., \overline{x_n}) \tag{4.15}$$

El conjunto de estados $\mathcal{S}_2$ de $\mathcal{A}$ es un *producto cartesiano* de un conjunto de n estados. La función Φ es la función global biyectiva del AC. La función π_i donota la i-ésima proyección de $\mathcal{S}_2$ y la función $\mathcal{F}$ es la función de transición del AC. Por lo tanto el AC $\mathcal{A}$ es inyectivo, ésto sigue de la biyección de Φ.

$$\mathcal{S}_2 = s_1 \times s_2 \times \cdots \times s_n \tag{4.16}$$

$$\Phi : \mathcal{S}_2 \longrightarrow \mathcal{S}_2 \tag{4.17}$$

$$\pi_i : s_k = (a_1, a_2, a_3, ..., a_n) \longrightarrow a_i \tag{4.18}$$

$$\mathcal{F} : \mathcal{S}_2^n \longrightarrow \mathcal{S}_2 \tag{4.19}$$

$$\mathcal{F}(s_1, s_2, ..., s_n) = \Phi\Big(\pi_1(s_1), \pi_2(s_2), ..., \pi_n(s_n)\Big) \tag{4.20}$$

Sea $\mathcal{B}$ otro AC y sea $\mathcal{G}$ su función de transición y sea $\mathcal{H}^{-1}$ el vector vecindad obtenido cambiando los signos de todas las coordenadas del vector vecindad $\mathcal{H}$. Entonces el AC $\mathcal{B}$ es inyectivo y más aún el AC es inverso de $\mathcal{A}$.

$$\mathcal{G}(s_1, s_2, ..., s_n) = \left(\pi_1\left(\Phi^{-1}(s_1)\right), \pi_2\left(\Phi^{-1}(s_2)\right), ..., \pi_n\left(\Phi^{-1}(s_n)\right)\right) \tag{4.21}$$

$$\mathcal{B}_{\mathcal{G}} = {\mathcal{A}_{\mathcal{F}}}^{-1} \Longrightarrow \mathcal{B}_{\mathcal{G}} \circ \mathcal{A}_{\mathcal{F}} = Id \tag{4.22}$$

Para poder construir el conjunto de estados $\mathcal{S}_2$ de los AC reversibles $\mathcal{A}$, $\mathcal{B}$. Sea U el conjunto de 16 estados, dado por la Ecuación 4.23.

$$U = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p\} \tag{4.23}$$

El conjunto de estados de $\mathcal{S}_2$, se construye a partir de un conjunto de 16 estados, luego dependiendo del vector vecindad elegido, el conjunto será particionado en n subconjuntos dependiendo del número de células que forman el vector vecindad; es decir, si elegimos el vector vecindad de Von Neumann la partición será de 4 subconjuntos; por otro lado, si elegimos el vector vecindad de Moore la partición será de 8 subconjuntos. Independientemente del vector vecindad que sea elegido el conjunto de estados $\mathcal{S}_2$ de $\mathcal{A}$ y $\mathcal{B}$ tendrá una cardinalidad de 256 estados, esto es $|\mathcal{S}_1| = |\mathcal{S}_2|$.

El Algoritmo 2 muestra cómo construir el conjunto de estados $\mathcal{S}_2$ dependiendo del vector vecindad que será usado

Algoritmo 2 Construcción del conjunto de estados $\mathcal{S}_2$

Entrada: U, $\mathcal{H}$
Salida: $\mathcal{S}_2$

1: Crear permutacion aleatoria de U
2: **si** $\mathcal{H}$ es igual a Von Neumann **entonces**
3: Particionar U en 4 subjuntos
4: **devolver** $\mathcal{S}_2 = U_1 \times ... \times U_4$
5: **si no**
6: Particionar U en 8 subjuntos
7: **devolver** $\mathcal{S}_2 = U_1 \times ... \times U_8$
8: **fin si**

El vector vecindad de Von Neumann para el AC $\mathcal{A}$ está dado por $\mathcal{H}_{VN}$ Ecuación 4.24 y el vector vecindad del AC inverso $\mathcal{B}$ está dado por $\mathcal{H}_{NV}^{-1}$, Ecuación 4.25. La Figura 4.4 muestra las coordenadas del vector vecindad. Por otro lado el vector vecindad de Moore para el AC $\mathcal{A}$ está dado por $\mathcal{H}_M$, Ecuación 4.26 y el vector vecindad del AC inverso $\mathcal{B}$ está dado por $\mathcal{H}_M^{-1}$, Ecuación 4.27. La Figura 4.5 muestra las coordenadas de los vectores vecindad.

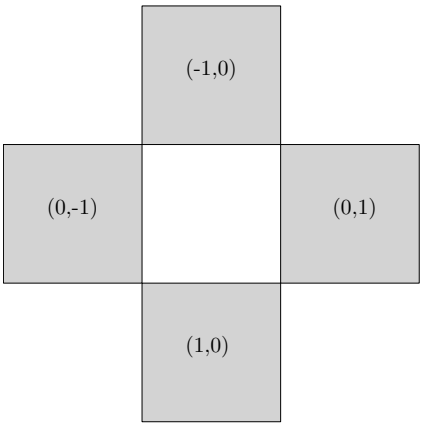

Figura 4.4: Vecindad de Von Neumann.

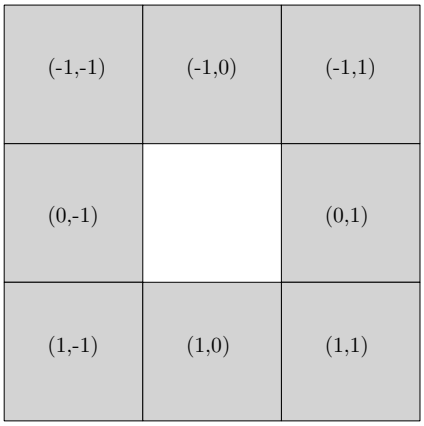

Figura 4.5: Vecindad de Moore.

$$\mathcal{H}_{VN} = \Big\{(-1,0),(1,0),(0,-1),(0,1)\Big\} \tag{4.24}$$

$$\mathcal{H}_{VN}^{-1} = \Big\{(1,0),(-1,0),(0,1),(0,-1)\Big\} \tag{4.25}$$

$$\mathcal{H}_M = \Big\{(-1,-1),(0,-1),(1,-1),(-1,0),(1,0),(-1,1),(0,1),(1,1)\Big\} \tag{4.26}$$

$$\mathcal{H}_M^{-1} = \Big\{(1,1),(0,1),(-1,1),(1,0),(-1,0),(1,-1),(0,-1),(-1,-1)\Big\} \tag{4.27}$$

La transformación afín es tomada de [2] donde construyen un criptosistema de llave privada que usa esta transformación afín. Cabe señalar que si quisiéramos aumentar el tamaño del bloque de cifrado se puede seguir utilizando esta transformación siempre que sólo incrementemos el número de células de los AC. La transformación está definida por la Ecuación 4.28. Esta transformación es invertible y la inversa está dada por la Ecuación 4.29.

$$A \cdot x + b = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \tag{4.28}$$

$$C \cdot y + d = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.29}$$

Cada bloque τ_1 contiene 16 elementos de GF(2^8) por lo tanto cada elemento será transformado. El resultado de transformar el bloque τ_1 será el bloque τ_1^{Afin}. El bloque τ_1^{Afin} se convertirá en el estado inicial del *AC mayor*. Por lo tanto el *AC mayor* tiene una *estructura reticular* cuadrada de 4×4.

Debido a que el conjunto de estados $\mathcal{S}_2$ del *AC mayor* es diferente al conjunto de elementos $\mathcal{S}_1$ en GF(2^8), debemos crear un *isomorfismo*♣ entre el conjunto $\mathcal{S}_2$ de elementos de GF(2^8) y el conjunto de estados $\mathcal{S}_2$; es decir, debemos crear una función biyectiva φ del conjunto de elementos $\mathcal{S}_1$ que forman el campo finito GF(2^8) al conjunto de estados $\mathcal{S}_2$, Ecuación 4.30. Este isomorfismo permite trabajar con cualquiera de los dos dominios, dicho de otra manera, si trabajamos en el dominio $\mathcal{S}_1$ del conjunto de GF(2^8) y realizamos operaciones sobre los AC, implicitamente las mismas operaciones serás realizadas en el dominio $\mathcal{S}_2$ y viceversa.

$$\varphi : \mathcal{S}_1 \longrightarrow \mathcal{S}_2 \tag{4.30}$$

El Algoritmo 3 muestra la forma para construir la funcion biyectiva φ. Este isomorfismo nos permitirá mapear los elementos del bloque τ_1^{Afin} a elementos de $\mathcal{S}_2$, posteriormente iterar el *AC mayor* y despúes regresar al dominio $\mathcal{S}_1$ de elementos del campo finito GF(2^8).

Algoritmo 3 Construcción del isomorfismo φ

Entrada: $\mathcal{S}_1$, $\mathcal{S}_2$

Salida: φ

1: **para** $i = 1$ hasta $|\mathcal{S}_1|$ **hacer**
2: Aleatoriamiente sacar elementos $a \in \mathcal{S}_1$ y $a' \in \mathcal{S}_2$
3: Asignar i como índice para referenciar los elementos a y a'
4: **fin para**

La Figura 4.6 muestra cómo trabaja el isomorfismo de un estado del AC sobre elementos de la extensión del campo GF(2^8) a su correspondiente estado sobre elementos del AC reversible, donde $a_i \in GF(2^8)$ y $(b_1, ..., b_n)_i \in \mathcal{S}_2$ de $\mathcal{A}$ y $\mathcal{B}$. El número k es el número de células de los AC y n es el número de células que forman el vector vecindad.

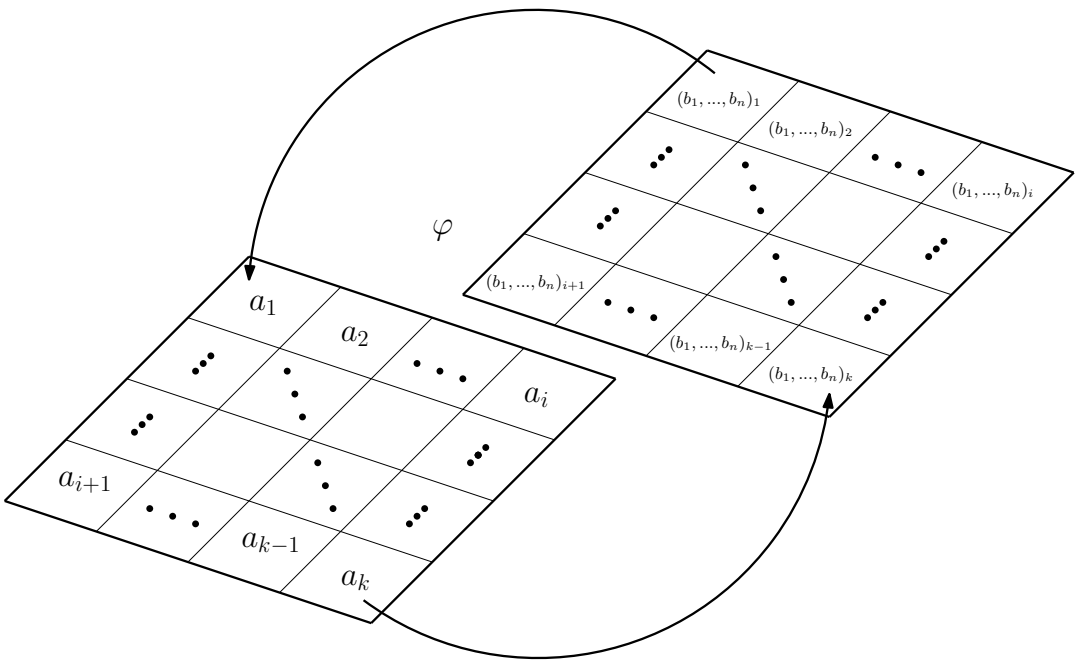

Figura 4.6: Isomorfismo φ

El *AC mayor* usa como estado inicial el bloque isomórfico τ_1^{Afin} y es iterado Δ veces generando el bloque cifrado τ_2 (ver IV en Figura 4.2). En la operación de descifrado usaremos el *AC mayor* $\mathcal{B}$ (inverso de $\mathcal{A}$) iterando Δ veces. El valor Δ se obtiene del estado S_N usando el Algoritmo 1, pero únicamente para el valor de entrada S_{N9}.

Una vez obtenido el bloque cifrado τ_2, será el nuevo estado del **AC menor**; esto es, el bloque τ_2 generará la nueva llave K (ver IV en Figura 4.2). El AC iterará d veces y obtendrá un nuevo estado S_N diferente al de la primer iteración. Asignar el bloque cifrado τ_2 como el nuevo estado del AC, permitirá que la llave privada K sea diferente para cada uno de los bloques que contiene el texto plano, agregando una pseudoaleatoriedad a la llave privada por cada bloque. Debido a que el AC tiene una estructura cíclica de acuerdo a la Ecuación 4.14, entonces la llave K podrá ser igual a un elemento que forma el ciclo.

Nivel 3 - Transformación no afín. Una transformación no afín es lograda por una función no lineal reversible (FNLR), según [16] este tipo de transformaciones aumenta la seguridad contra ataques por elección de texto cifrado. La función está definida en la Ecuación 4.31.

$$y = x \oplus \{(c_1 \cdot c_2) \oplus (c_2 \cdot c_3) \oplus (c_3 \cdot c_1)\} \tag{4.31}$$

Los los bits c_1, c_2 y c_3 forman el conjunto de control C. Usando el Algoritmo 4 obtendremos varios conjuntos de control con el propósito de usarlos en la transformación no afín. Los conjuntos son tomados del estado S_N. La operación $\oplus$ denota la operación XOR a nivel de bits y $\cdot$ denota la operación AND a nivel de bits.

Algoritmo 4 Obtener conjuntos de control C_i

Entrada: S_N
Salida: $C_1, ..., C_{|S_N|}$

1: **para** $i = 1$ hasta $|S_N|$ **hacer**
2: Obtener la representación binaria del elemento S_{Ni}
3: $C_i =$ Los bits b_1, b_5, b_7 de la representación binaria
4: **fin para**

El bloque τ_2 es transformado usando la Ecuación 4.31. Depenediendo del resultado de una *Función de Evaluación Mayoritaria (FEM)*. Esta función toma 5 bits fijos $p_1p_2p_3p_4p_5$ del bloque τ_2 y calcula la cantidad de 1's que hay en esos bits. Particularmente, los bits son tomados del byte número trece del bloque τ_2. El Algoritmo 5 muestra la forma para obtener el bloque cifrado τ_3 (ver V en la Figura 4.2), aplicando los conjuntos de control C_i, junto con la función FEM a el bloque τ_2.

Algoritmo 5 Tercer nivel de transformación

Entrada: τ_2, $p_1p_2p_3p_4p_5$, $C_1, ..., C_{|S_N|}$
Salida: τ_3

si $\sum_{i=1}^{5} B_{p_i}(\tau_2) > 2$ **entonces**
  para $i = 1$ hasta $\frac{|\tau_2|}{8}$ **hacer**
    Obtener representación binaria del $byte_i$ del bloque τ_2
    para $j = 1$ hasta 8 **hacer**
      Aplicar FNLR para el $byte_i$ del bloque τ_2 usando el conjunto de control C_i
    fin para
  fin para
si no
  τ_2 permanece igual
fin si

Los conjuntos de control C_i son tomados del estado S_N y son aplicados a la transformación no afín alternadamente. Los cinco bits $p_1p_2p_3p_4p_5$ permanecen sin transformar debido a que la operación de descifrado requiere de los mismos cinco bits para regresar al bloque τ_2.

Nivel 4 - Mezcla de la llave. Como último nivel de cifrado y lograr un mayor nivel de seguridad, generamos un bloque final de cifrado τ_c con la Ecuacíon 4.32 (ver VI la Figura 4.2). Nótese que la operación es la operación de adición sobre el campo GF(2^8).

$$\tau_c = \tau_3 \oplus S_N \tag{4.32}$$

Finalmente, utilizando los algoritmos listados anteriormente, mostramos el Algoritmo 6 que muestra la operación de cifrado del criptosistema, mientras que si hacemos el mismo procedimiento a la inversa, entonces obtenemos el Algoritmo 7 que muestra la operación de descifrado.

Algoritmo 6 Cifrado

Entrada: Texto plano T y llave K
Salida: Texto cifrado $T_{cifrado}$

1: Dividir T en bloques de tamaño 128
2: Asignar la llave K como estado inicial del AC menor $S_0 = K$
3: **para** cada bloque τ **hacer**
4: Iterar el AC menor d veces y obtener S_N
5: Obtener $< \delta_1, ..., \delta_{|S_N|} >$ de S_N
6: Rotar los bytes de τ usando $< \delta_1, ..., \delta_{|S_N|} >$ y obtener τ_1
7: Obtener τ_1^{Afin} de τ_1
8: Obtener Δ de S_N
9: Iterar el AC mayor $\mathcal{A}$, Δ pasos con $\varphi(\tau_1^{Afin})$ como estado inicial y obtener τ_2
10: Regresar al dominio de GF(2^8) con $\varphi^{-1}(\tau_2)$
11: $S_0 = \tau_2$
12: **si** los bits $(p_1p_2p_3p_4p_5)$ de τ_2 satisfacen a FEM **entonces**
13: Obtener los conjuntos de control C_i para FNLR
14: Aplicar FNLR a τ_2 usando los conjuntos de control alternadamente
15: Asignar el resultado a τ_3
16: **fin si**
17: Obtener $\tau_c = \tau_3 \oplus S_N$
18: Escribir τ_c en el archivo de salida $T_{cifrado}$
19: **fin para**

Algoritmo 7 Descifrado

Entrada: Texto cifrado $T_{cifrado}$ y llave K
Salida: Texto plano T

1: Dividir $T_{cifrado}$ en bloques de tamaño 128
2: Asignar la llave K como estado inicial del AC menor $S_0 = K$
3: **para** cada bloque τ_c **hacer**
4: Iterar el AC menor d veces y obtener S_N
5: Obtener $\tau_3 = \tau_c \oplus S_N$
6: **si** los bits $(p_1p_2p_3p_4p_5)$ de τ_3 satisfacen a FEM **entonces**
7: Obtener los conjuntos de control C_i de bits para FNLR
8: Aplicar FNLR a τ_3 usando los conjuntos de control alternadamente
9: Asignar el resultado a τ_2
10: **fin si**
11: $S_0 = \tau_2$
12: Obtener $\varphi(\tau_2)$
13: Obtener Δ de S_N
14: Iterar el AC mayor $\mathcal{B}$, Δ pasos con τ_2 como estado inicial y obtener τ_1^{Afin}
15: Regresar al dominio de GF(2^8) con $\varphi^{-1}(\tau_1^{Afin})$
16: Obtener τ_1 de τ_1^{Afin}
17: Obtener $< \delta_1, ..., \delta_{|S_N|} >$ de S_N
18: Rotar los bytes de τ_1 usando $< \delta_1, ..., \delta_{|S_N|} >$ a la inversa y obtener τ
19: Escribir τ en el archivo de salida T
20: **fin para**

Capítulo 5

Simulación y Resultados

En este capítulo simulamos el modelo propuesto en el capítulo cuatro, con varios ejemplos de texto plano y obtenemos su correspondiente texto cifrado. El criptosistema ha sido implementado usando el software *Mathematica* y *Wolfram Workbench*. La implementación permite cifrar texto plano usando las vecindades de Von Neumann o Moore, además este capítulo describe dos tipos de ataque hechos al criptosistema. Finalmente hacemos una comparación con los trabajos presentados en el capítulo tres.

5.1. Interfaz de aplicación

La Figura 5.1 es la interfaz de aplicación del criptosistema y además, sirve para ajustar parámetros de la simulación. Entre los ajustes que se pueden hacer al criptosistema, particularmente podemos elegir entre las operaciones de cifrado y descifrado, elegir vector vecindad para el AC 2-dimensional reversible (Von Neumann o Moore), seleccionar el archivo que deseamos cifrar (archivos de texto), número de iteraciones para el autómata celular de dimensión uno y finalmente la llave privada que será el estado incial del AC grupo sobre GF(2^8). Debido a que el tamaño de bloque a cifrar es de 128 bits, la llave privada también será del mismo tamaño.

Los archivos de texto están compuestos por caracteres, por tal razón debemos crear un mapeo de los caracteres que forman el texto y elementos de la extensión del campo GF(2^8). Ahora, debido a que el conjunto de estados de los autómatas es de 256, necesitamos una codificación de caracteres de 8 bits, de esta manera podemos tener 2^8 posibles caracteres. La codificación ASCII[1] usa 8 bits para representar caracteres, por tal razón, para las simulaciones que se harán se usará dicha codificación. Ahora, para realizar el mapeo de caracteres a elemetos de GF(2^8) usamos el Algoritmo 8.

[1]Acrónimo inglés de American Standard Code for Information Interchange – Código Estándar Estadounidense para el Intercambio de Información

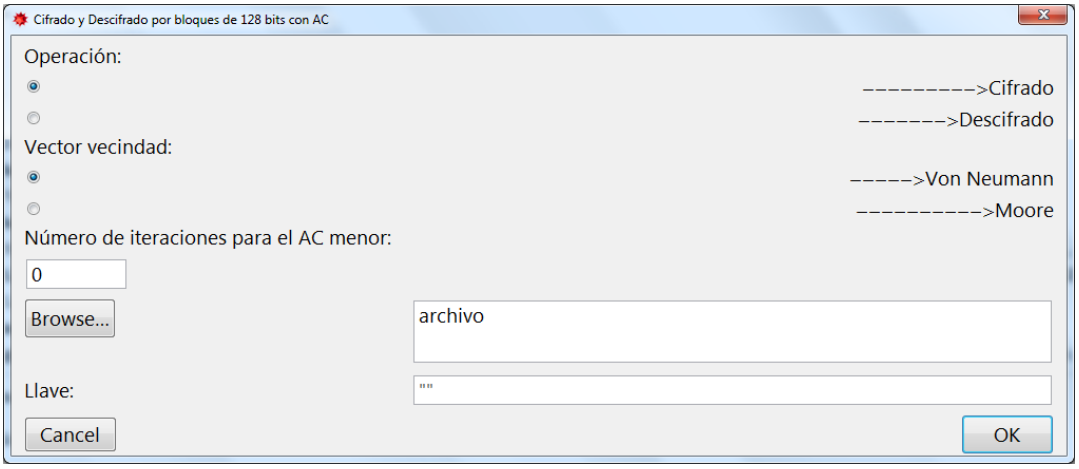

Figura 5.1: Interfaz de inicialización

Algoritmo 8 Mapeo de caracteres a elementos de GF(2^8)

Entrada: Caracter c
Salida: $a \in GF(2^8)$
1: Obtener la representación binaria de c
2: $a = \Sigma_{i=7}^{0} c_i t^i$

Por otro lado, si nosotros deseamos mapear elementos de la extensión del campo a caracteres de la codificación ASCII, lo podemos hacer obteniendo cada uno de los coeficientes del polinomio respectivo, de esta manera obtenemos un byte que representa a un caracter, esto de la Definición 2.4 en el capítulo 2.

El primer escenario de simulación está dado por el ajuste de parámetros mostrado en la Tabla 5.1. El archivo que se especifica en los parámetros puede ser consultado en el **Apéndice C**. Particularmente para esta simulación el texto que se usa para cifrar es la letra del "Himno Nacional Mexicano". El tamaño de este archivo es de 86 bloques, para cada caso se usa las vecindades de Moore y Von Neumman de radio $r = 1$, el número d de iteraciones del AC menor es de 10. La llave K está codificada en hexadecimal.

Las Figuras 5.2 y 5.3 muestran el texto cifrado como resultado del primer escenario de simulación, de acuerdo a las Tablas 5.1 y 5.2 respectivamente.

Tabla 5.1: Configuración de parámetros del criptosistema para cifrar el archivo HimnoNacionalMexicano.txt

Configuración de los parámetros del sistema		
Paramétro	**Descripción**	**Valor**
Operación	Transformación al texto	Cifrado
Vecindad	Tipo de vector vecindad que interviene en la evolución de los AC	VON NEUMANN $r = 1$
Iteraciones AC 1D	Número de iteraciones del AC menor	10
Archivo	Especificación del archivo de texto	HimnoNacionalMexicano.txt
Llave K	Estado inicial del AC menor	CCAC2AED591056BE4F90FD441C534766

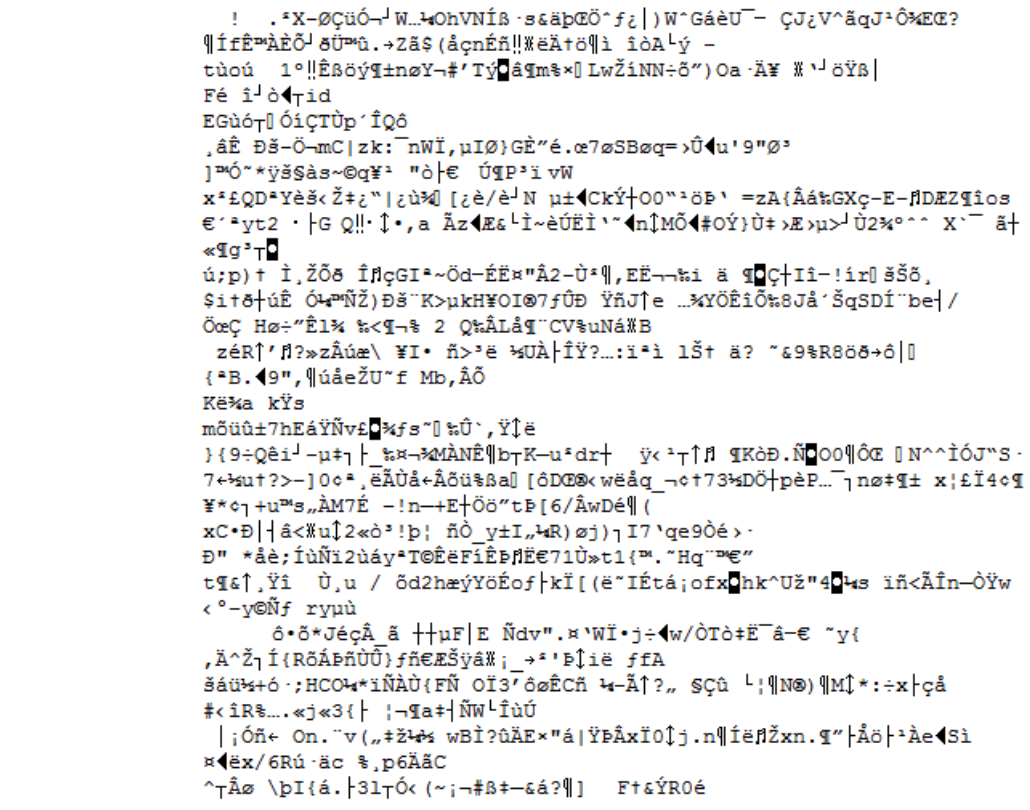

Figura 5.2: Himno Nacional Mexicano cifrado usando parámetros de la Tabla 5.1.

Tabla 5.2: Configuración de parámetros del criptosistema para cifrar el archivo HimnoNacionalMexicano.txt

Configuración de los parámetros del sistema		
Paramétro	**Descripción**	**Valor**
Operación	Transformación al texto	Cifrado
Vecindad	Tipo de vector vecindad que interviene en la evolución de los AC	MOORE $r = 1$
Iteraciones AC 1D	Número de iteraciones del AC menor	10
Archivo	Especificación del archivo de texto	HimnoNacionalMexicano.txt
Llave	Estado inicial del AC menor	CCAC2AED591056BE4F90FD441C534766

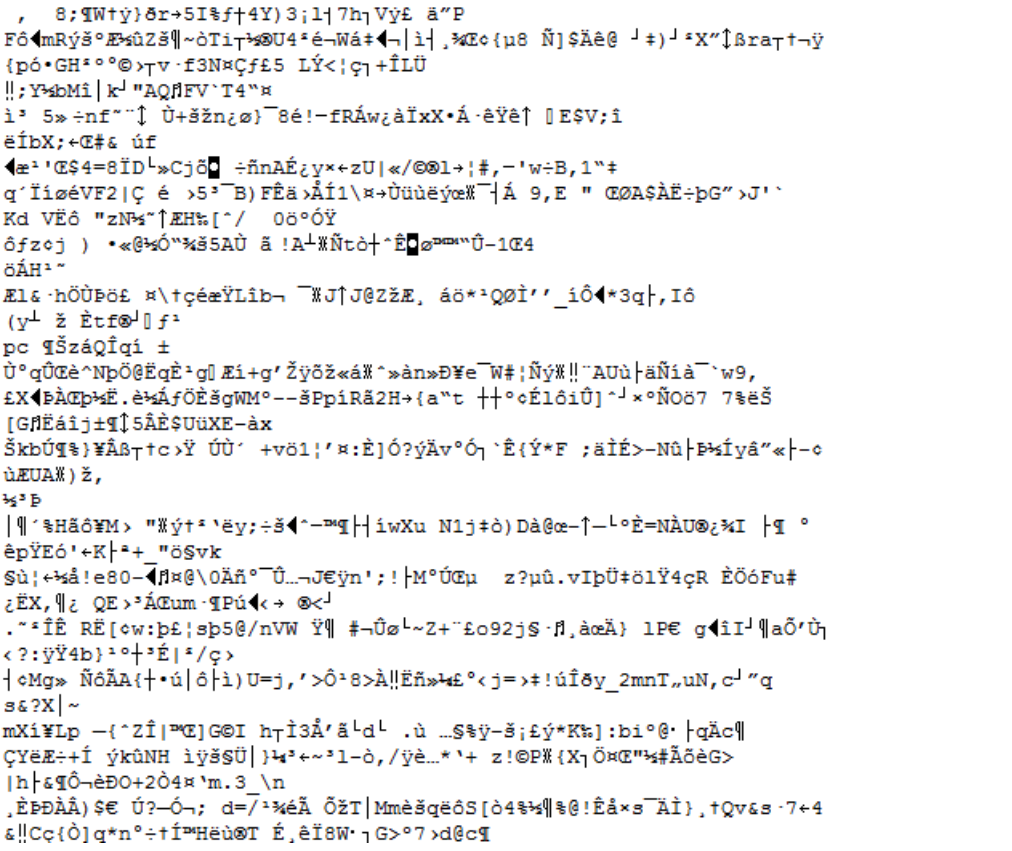

Figura 5.3: Himno Nacional Mexicano cifrado usando parámetros de la Tabla 5.2.

El segundo escenario de simulación está dado por el ajuste de parámetros mostrado en la Tabla 5.3. Para esta simulación se usará un archivo de texto formado por bloques que contienen al caracter 'a'.

Las Figuras 5.4 y 5.5 muestran el texto cifrado como resultado del primer escenario de simulación, de acuerdo a las Tablas 5.3 y 5.4 respectivamente.

Tabla 5.3: Configuración de parámetros del criptosistema para cifrar el archivo textoa.txt

Configuración de los parámetros del sistema		
Paramétro	**Descripción**	**Valor**
Operación	Transformación al texto	Cifrado
Vecindad	Tipo de vector vecindad que interviene en la evolución de los AC	VON NEUMANN $r = 1$
Iteraciones AC 1D	Número de iteraciones del AC menor	10
Archivo	Especificación del archivo de texto	textoa.txt
Llave	Estado inicial del AC menor	11111111111111111111111111111111

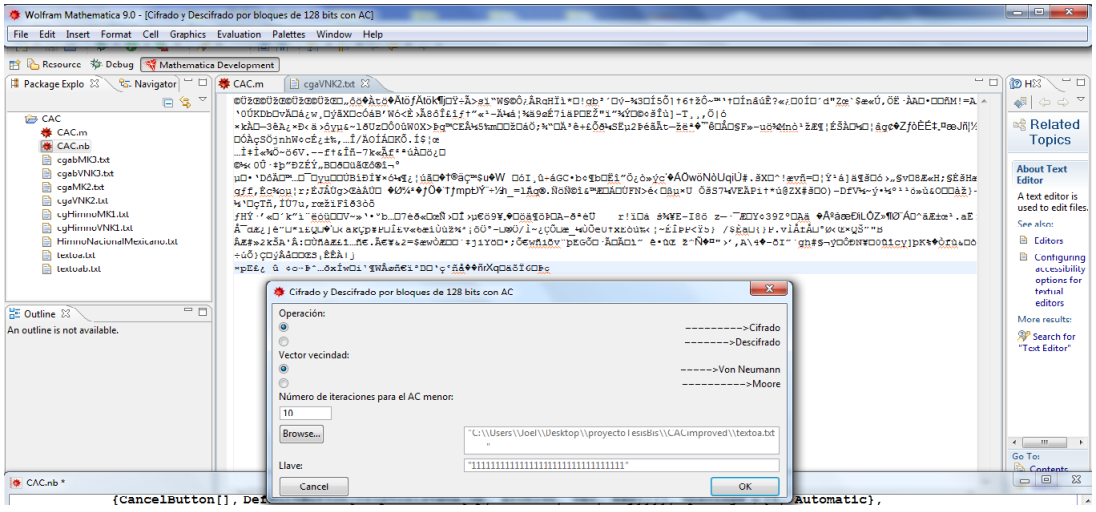

Figura 5.4: Texto cifrado usando parámetros de la Tabla 5.3. Notar que todos los bloques que contiene el texto están formados únicamente por el caracter 'a'.

Tabla 5.4: Configuración de parámetros del criptosistema para cifrar el archivo textoa.txt

Configuración de los parámetros del sistema		
Paramétro	**Descripción**	**Valor**
Operación	Transformación al texto	Cifrado
Vecindad	Tipo de vector vecindad que interviene en la evolución de los AC	MOORE $r = 1$
Iteraciones AC 1D	Número de iteraciones del AC menor	10
Archivo	Especificación del archivo de texto	textoa.txt
Llave	Estado inicial del AC menor	11111111111111111111111111111111

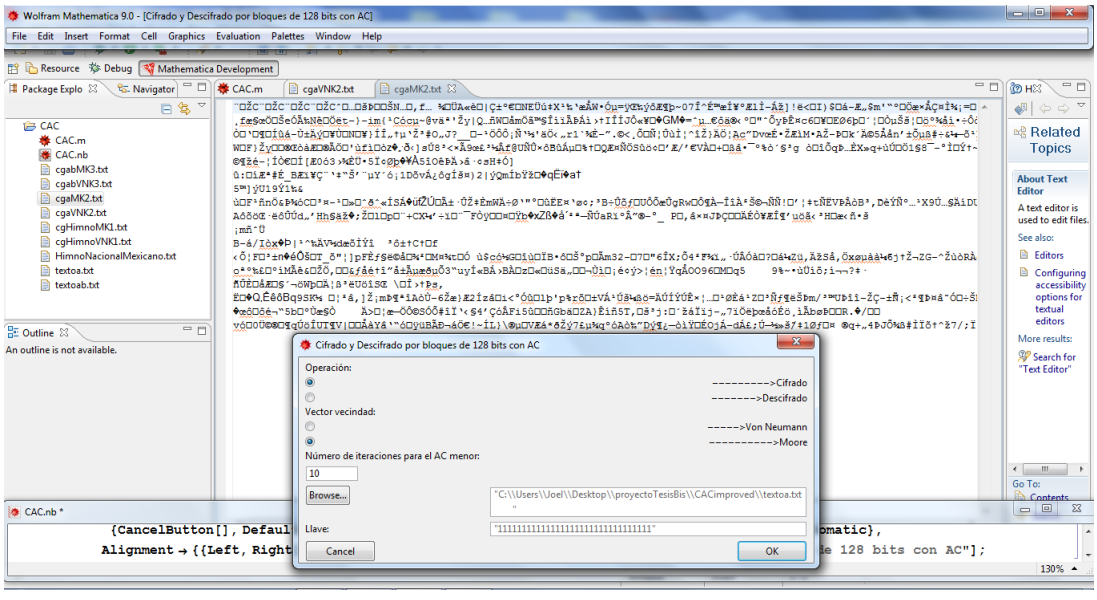

Figura 5.5: Texto cifrado usando parámetros de la Tabla 5.4. Notar que todos los bloques que contiene el texto están formados únicamente por el caracter 'a'.

El tercer escenario de simulación está dado por el ajuste de parámetros mostrado en la Tabla 5.5. Para esta simulación se usará un archivo de texto formado por dos bloques intercalados, uno de ellos contiene solamente al caracter 'a' y el otro de ellos solamente al caracter 'b'.

Las Figuras 5.6 y 5.7 muestran el texto cifrado como resultado del primer escenario de simulación, de acuerdo a las Tablas 5.5 y 5.6 respectivamente.

Tabla 5.5: Configuración de parámetros del criptosistema para cifrar el archivo textoab.txt

Configuración de los parámetros del sistema		
Paramétro	**Descripción**	**Valor**
Operación	Transformación al texto	Cifrado
Vecindad	Tipo de vector vecindad que interviene en la evolución de los AC	VON NEUMANN $r = 1$
Iteraciones AC 1D	Número de iteraciones del AC menor	10
Archivo	Especificación del archivo de texto	textoab.txt
Llave	Estado inicial del AC menor	0123456789ABCDEF0123456789ABCDEF

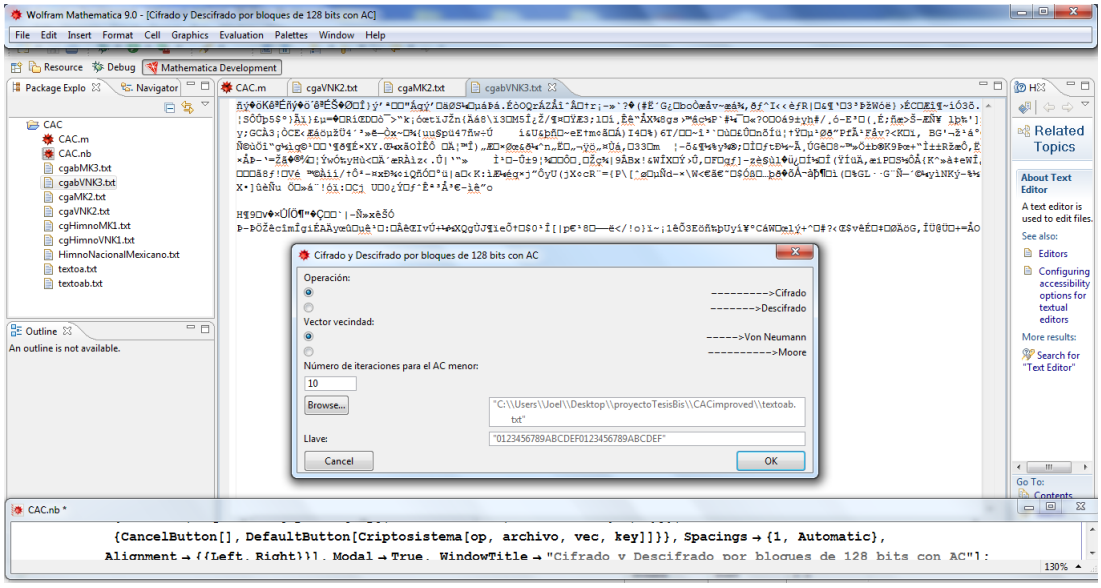

Figura 5.6: Texto cifrado usando parámetros de la Tabla 5.5. Notar que todos los bloques que contiene el texto están formados por dos bloques diferentes, uno de ellos formado por el caracter 'a' y el otro formado por el caracter 'b'.

Tabla 5.6: Configuración de parámetros del criptosistema para cifrar el archivo textoab.txt

Configuración de los parámetros del sistema		
Paramétro	**Descripción**	**Valor**
Operación	Transformación al texto	Cifrado
Vecindad	Tipo de vector vecindad que interviene en la evolución de los AC	MOORE $r = 1$
Iteraciones AC 1D	Número de iteraciones del AC menor	10
Archivo	Especificación del archivo de texto	textoab.txt
Llave	Estado inicial del AC menor	0123456789ABCDEF0123456789ABCDEF

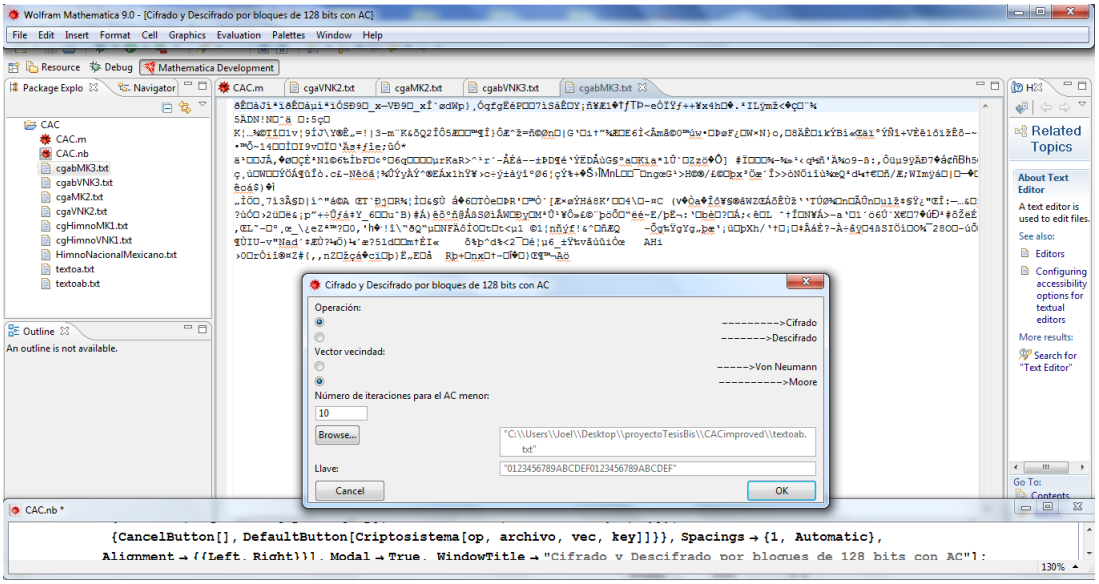

Figura 5.7: Texto cifrado usando parámetros de la Tabla 5.6. Notar que todos los bloques que contiene el texto están formados por dos bloques diferentes, uno de ellos formado por el caracter 'a' y el otro formado por el caracter 'b'.

5.2. Fuerza bruta

Debido a que el criptosistema es de llave privada y tiene sólo una llave con tamaño de llave de 128 bits, veremos si es computacionalmente seguro contra el ataque por *fuerza bruta*. El tamaño de la llave usado en el cifrado determina la factibilidad de romper el criptosistema con un ataque por fuerza bruta. Veremos que con llaves más grandes será exponencialmente más difícil romper el criptosistema que con llaves más cortas.

Utilizar la fuerza bruta sistemáticamente hace una revisión de todas las posibles combinaciones de la llave hasta encontrar la llave correcta. La Figura 5.8 muestra un ejemplo de un ataque por fuerza bruta sobre una llave de cuatro bits.

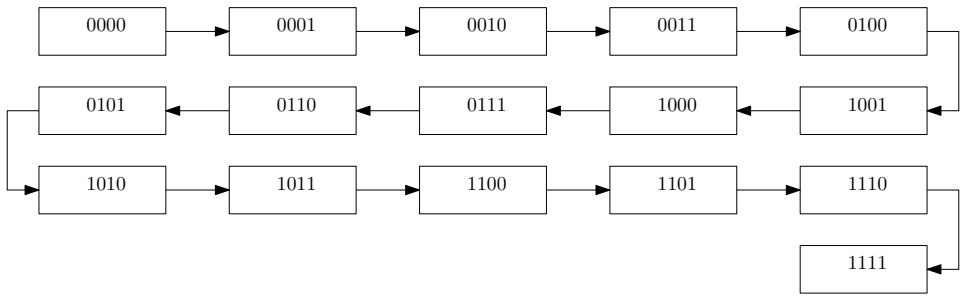

Figura 5.8: Fuerza bruta sobre una llave de cuatro bits.

Como se observa, tomará un máximo de 16 rondas para verificar todas las posibles combinaciones comenzando con 0000. Dado suficiente tiempo, un ataque por fuerza bruta es capaz de romper cualquier criptosistema. La Tabla 5.7 muestra el número de combinaciones de una llave con respecto a su tamaño.

Tabla 5.7: Tamaño de llave y el número de posibles combinaciones.

Tamaño de llave K	Posibles combinaciones
1 bit	2
2 bits	4
4 bits	16
8 bits	256
16 bits	65536
32 bits	$4.2\times\ 10^{9}$
64 bits	$7{,}2 \times 10^{16}$
64 bits	$1{,}8 \times 10^{19}$
128 bits	$3{,}4 \times 10^{38}$

Nótese el incremento exponencial en el número de combinaciones así como el tamaño de llave. Por ejemplo un criptosistema de llave privada en [3] con tamaño de llave de 56 bits ha sido roto usando un ataque de fuerza bruta.

Según [1] la computadora más poderosa puede calcular $33{,}86 \times 10^{15}$ *FLOPS*♣. Para fines ilustrativos si una instrucción de la computadora puede pobrar una posibe combinación. Entonces, el número de combinaciones que la super computadora puede verificar por segundo son:

$$\frac{33{,}86 \times 10^{15}}{1} = 33{,}86 \times 10^{15}.$$

El número de segundos en un año es igual a $365 \times 60 \times 60 \times 24 = 31536000$. Por lo tanto, si tomamos en cuenta lo anterior y si lo usamos para medir la resistencia del criptosistema, implementado en este trabajo, contra una ataque por fuerza bruta entonces, el número de años en romper el criptosistema, con un tamaño de llave de 128 bits, será de:

$$\frac{3{,}4 \times 10^{38}}{(33{,}86 \times 10^{15}) \times 31536000}$$

$$= \frac{1{,}004135 \times 10^{22}}{31536000}$$

$$= 3{,}184091 \times 10^{14}$$

$$= 318409100000000.$$

Como se muestra en la Tabla 5.3, incluso con la ayuda de una super computadora, tardaría aproximadamente $3{,}184091 \times 10^{14}$ años en romper el criptosistema con un ataque por fuerza bruta usando un tamaño de llave de 128 bits.

Tabla 5.8: Tiempo que tarda en romper el cripstosistema respecto al tamaño de llave.

Tamaño de llave K	Tiempo
56 bits	2.12 segundos
128 bits	$3{,}184091 \times 10^{14}$ años

5.3. Análisis de frecuencias

En esta sección analizaremos las frecuencias de aparición de los caracteres que conforman a los textos planos y sus correspondientes textos cifrados con el fin de observar el comportamiento de la distribución de las frecuencias de aparición de caracteres con respecto al número de estados. La Ecuación 5.1 nos ayuda a calcular las frecuencias de los caracteres contenidos en los textos.

$$frec_x = \frac{O_x}{|\mathcal{S}|} \tag{5.1}$$

donde:

◇ $x \in \mathcal{S}$.

◇ $frec_x$ es la frecuencia de aparición de x.

◇ O_x es el número de ocurrencias del elemento x en el mensaje.

La Tabla 5.9 muestra los archivos involucrados en este análisis, se observa que son los mismos archivos que usamos para la simulación (revisar **Apéndice C**) con sus respectivos textos cifrados tomados del resultado de la simulación. La elección de estos archivos es para observar lo que ocurre cuando ciframos archivos que contienen cadenas de caracteres con periodicidad. Además la tabla también contiene el número de bloques que se está analizando en cada caso.

Tabla 5.9: Archivos involucrados en el análisis de frecuencias y el número de bloques contenidos en cada caso.

Archivo	**Descripción**	**Bloques**
HimnoNacionalMexicano.txt y textos cifrados	Contiene la letra del Himno Nacional Mexicano	86
textoa.txt y textos cifrados	Todos los bloques están formados por el caracter 'a'	112
textab.txt y textos cifrados	Archivo formado por los bloques "aaaaaaaaaaaaaaaa" y "bbbbbbbbbbbbbbbb"	112

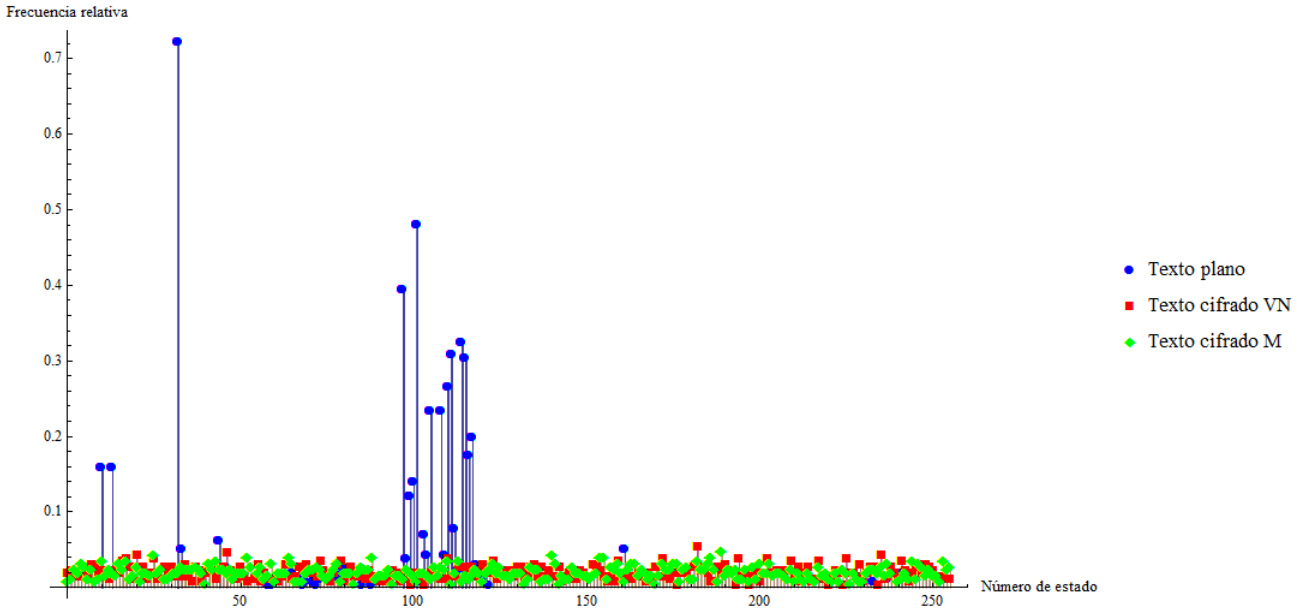

Figura 5.9: Distribución de la frecuencia de aparición de los caracteres, los textos cifrados corresponden al resultado de la simulación según las Tablas 5.1 y 5.2. En azul se observa la distribución del archivo $HimnoNacionalMexicano.txt$, en rojo la distribución del texto cifrado usando el vector vecindad de Von Neuman y en verde el texto cifrado usando el vector vecindad de Moore.

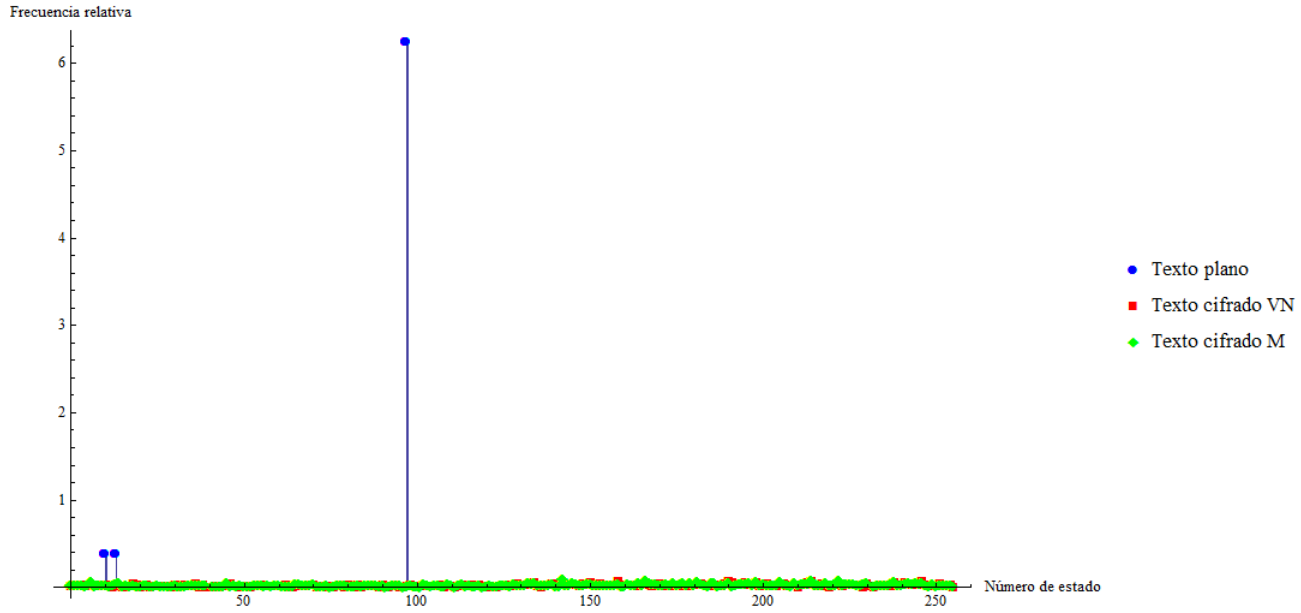

Figura 5.10: Distribución de la frecuencia de aparición de los caracteres, los textos cifrados corresponden al resultado de la simulación según las Tablas 5.3 y 5.4. En azul se observa la distribución del archivo *textoa.txt*, en rojo la distribución del texto cifrado usando el vector vecindad de Von Neuman y en verde el texto cifrado usando el vector vecindad de Moore.

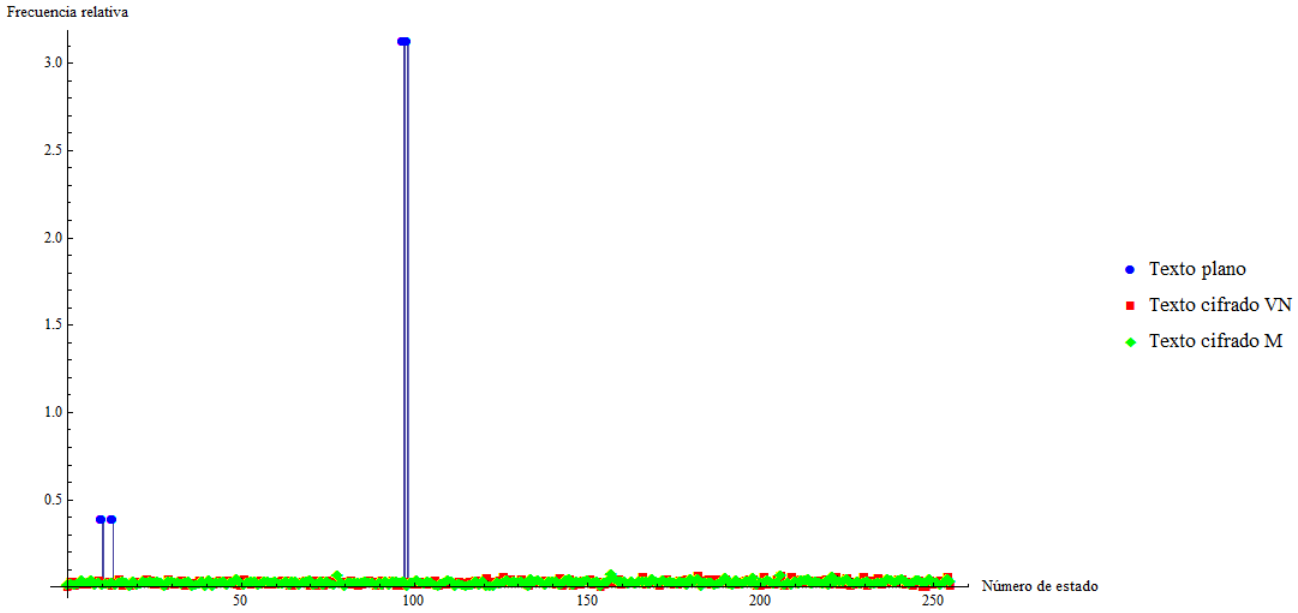

Figura 5.11: Distribución de la frecuencia de aparición de los caracteres, los textos cifrados corresponden al resultado de la simulación según las Tablas 5.5 y 5.6. En azul se observa la distribución del archivo *textoab.txt*, en rojo la distribución del texto cifrado usando el vector vecindad de Von Neuman y en verde el texto cifrado usando el vector vecindad de Moore.

5.4. Comparación con otros modelos

En el capítulo tres presentamos algunos trabajos relacionados con el problema planteado al inicio de la tesis. La regla 60, en [33] Stephen Wolfram usó un AC 1-dimensional, con la regla 60 aditiva. La regla 60 permite trabajar sobre el campo finito GF(2). Ahora, según Jarkko Kari en [18] un AC reversible está formado por ciclos, por lo tanto los AC sobre GF(2) que forman estructuras cíclicas pueden verse como AC reversibles. Por otro lado, uno podría pensar que usar un AC reversible y que su función de transición tiene dominio sobre la aritmética del campo, tendría como consecuencia una mayor dificultad para romper el cripstosistema, pero aun con todo esto en [5] S. Amoroso, desmuestra que existe un algoritmo para revertir un AC 1-dimensional en tiempo polinomial. Para efectos de seguridad, un cripstosistema que puede ser roto en tiempo polinomial no es un buen cripstosistema [20].

El trabajo en [11] utiliza AC sobre GF(2^8) para implementar un criptosistema (CAC), pero debido a que los AC son de una dimensión existe un algoritmo que invierte los AC en tiempo polinomial y por lo tanto el cripstosistema es roto en tiempo polinomial. Este criptosistema fue implementado por expertos en AC y criptografía del Instituto Tecnológico de la India. Ahora bien, el criptosistema modelado y simulado en esta tesis también trabaja con AC sobre GF(2^8), donde uno de los AC es de dos dimensiones reversible. Usar AC reversibles de dos dimensiones tiene como consecuencia de que invertir este tipo de AC es un problema $NP - hard$, según [6].

Por lo tanto, tratar de romper el criptosistema, implicaría resolver este problema, luego entonces el tiempo que taradaría en hacer esta tarea sería exponencial.

5.5. Ventajas y desventajas

De acuerdo a los resultados obtenidos por la simulación del sistema, consideramos lo siguiente,

Ventajas:

◇ Es un sistema criptográfico que puede cifrar y descifrar de manera eficiente y robusta.

◇ Es posible agregar o disminuir niveles de transformación al texto.

◇ El uso de autómatas permite un paralelismo inherente en el sistema.

◇ Es posible ajustar el tamaño de bloque.

◇ Se puede trabajar con un valor $p > 8$ para la extensión de campo GF(2^p).

Desventajas:

◇ El sistema sólo sirve para cifrar texto (hasta el momento).

◇ Un número mayor de evoluciónes para los autómatas implicará más tiempo de cómputo y por lo tanto el tiempo requerido para cifrar el texto se incrementará.

Capítulo 6

Conclusiones y trabajos futuros

En base a los objetivos planteados al inicio de este trabajo, llegamos a las siguientes conclusiones:

Utilizar AC implica una gran cantidad de cómputo debido a la iteración de cada una de las células que contiene, ya que se necesita aplicar la función de transición célula a célula junto con sus vecinos respectivos, multiplicado por el número de iteraciones. Por lo tanto, si nosotros incrementamos el número de células, el radio de la vecindad y además el número de iteraciones, entonces el tiempo de cálculo también se incrementará. Para efectos de cifrar información de manera más rápida, usando nuestro modelo, es recomendable especificar un número pequeño de evoluciones a los AC.

Los cuatro niveles de transformación que se aplican a cada uno de los bloques del texto plano agregan una mayor seguridad al sistema. Es posible incrementar más niveles de cifrado, pero esto perjudicaría en el tiempo requerido por el sistema para cifrar, debido a la gran cantidad de cálculos. Por otro lado si eliminamos uno o más niveles de cifrado seguimos obtiendo resultados satisfactorios, pero el sistema contiene más vulnerabilidades.

Al analizar las frecuencias de aparición de los caracteres que conforman a los textos planos y los caracteres de sus respectivos textos cifrados, especialmente los textos que contienen un alto índice de periodicidad en su contenido, nos permite concluir que el modelo propuesto elimina la periodicidad en el texto cifrado, incorporando una distribución casi uniforme en la frecuencia de aparición de los caracteres.

Los autómatas celulares tienen la propiedad de que soportan un paralelismo inherente. Por tal razón, un trabajo futuro, explotando esta propiedad, es implementar los AC del modelo propuesto usando técnicas de programación paralela y aumentar el rendimiento del criptosistema. Por otro lado, un trabajo parecido en ésta dirección, es implementar el criptosistema en su versión hardware usando dispositivos lógicos programables.

El resultado más importante de este trabajo es el software que permite cifrar y decifrar textos. Debido a que este criptosistema trabaja a nivel de bits, podemos incorporar la opción de cifrar imágenes para mejorar este trabajo.

Amoroso en [5] investigó posibles reglas reversibles para un AC de una dimensión. Por lo cual, una posibilidad para seguir este trabajo de investigación es usar los resultados obtenidos en [24] sobre la caracterización de AC sobre GF(2^p) 1-dimensional y encontrar reglas reversibles sobre ésta clase de autómatas.

En [10] ampliaron los resultados de la caracterización de AC sobre GF(2) de una dimensión a dos dimensines. Si tomamos un enfoque más teórico, entonces extender la caracterización para AC de dos dimensiones sobre GF(2^p) será una buena línea de investigación.

En en modelo del presente trabajo, usamos la extensión de campo GF(2^8). Se puede trabajar con una extensión de campo para p mayor que ocho y analizar los resultados del comportamiento del criptosistema.

Referencias

[1] Top 500 of supercomputer sites. `http://www.top500.org/lists/2013/11/`. Accessed: 2014-05-05.

[2] 197, F. I. P. S. P. Advanced encryption standard (aes).

[3] 46-3, F. I. P. S. P. Data encryption standard (des).

[4] Adamatzky, A. Game of life cellular automata. *Springer 1* (2010), 11–17.

[5] Amoroso, S., and Patt, N. Decision procedures for subjectivity and injectivity of parallel maps for tessellation structures. *Computer and Systems Sciences 6* (1972), 448–464.

[6] B. Applebaum, Y. Ishai, E. K. Cryptography by cellular automata or how fast can complexity emerge in nature? *Innovations in Computer Science* (2010), 420 – 287.

[7] Bao, F. Cryptanalysis of a partially known cellular automata cryptosystem. *IEEE TRANSACTIONS ON COMPUTERS 53* (2004), 1493–1497.

[8] Castillo, G. T. Los fractales un punto de convergencia de los autómatas celulares y los sistemas complejos. Master's thesis, Centro de Invectigación en Computación, IPN, 1999.

[9] Castillo, G. T. *Discrete Mathematics Notes*. Course notes. CIC, IPN, 2012.

[10] D. Roy Chowdhury, I. S., and Chaudhuri, P. P. Characterization of two-dimensional cellular automata using matrix algebra. *Information Sciences 76* (1993), 289–314.

[11] Ganguly, S. S. C. S. R. C. D. N., and P., P. C. Cellular automata based cryptosystem (cac). *Information and Communications Security 2513* (2002), 303–314.

[12] Gowers, T. *The Princeton Companion to Mathematics*, 1 ed. Princeton University Press, 2008.

[13] Hernández, J. M. M. Cifrado y descifrado de patrónes usando autómatas celulares reversibles. Master's thesis, Centro de Investigación en Computación, IPN, 2013.

[14] Herstein, I. N. *Topics in Algebra*. Vikas publishing House, Pvt. Ltd, 1987.

[15] Ilachinski, A. *Cellular Automata. A Discrete Universe*, 1 ed. World Scientific Publishing Co. Pte. Ltd, Singapore, 2001.

[16] Jaydeb Bhaumik, D. M., and Chowdhury, D. R. Reversible addition with increased nonlinearity. *International Journal of Network Security 15* (2013), 279 – 287.

[17] Kari, J. Reversibility of 2d cellular automata is undecidable. *Physica D 45* (1990), 379–385.

[18] Kari, J. Reversible cellular automata. *Springer-Verlag Berlin Heidelberg* (2004), 57–68.

[19] McIntosh, H. V. Linear cellular automata. 1–13.

[20] N. P. Varnovsky, A. I. V., and Primenko, E. A. Mathematical problems in cryptology. *Probability Theory and Mathematical Statistics 2* (1993), 3373–3406.

[21] Niloy Gangyly, B. S. P. P. C. Theory of additive cellular automata. *Fundamental Informatic 1* (2001), 1001–1021.

[22] Packard, N. H., and Wolfram, S. Two-dimensional cellular automata. *Statistical Physics 38* (1998), 901–908.

[23] Parimal Pal Chaudhuri, D. R. C. S. N. S. C. *Additive Cellular Automata: Theory and Applications, Volume 1*. Wiley-IEEE Computer Society Press, 1997.

[24] Paul, K. *Theory and Applications of GF(2^p) Cellular Automata*. PhD thesis, Bengal Engineering College (Deemed University), Calcutta , India., 2002.

[25] Rao, T. R. N., and Fujiwara, E. *Error-control Coding for Computer Systems*. Prentice-Hall, 1989.

[26] S. Nandi, B. K. K., and Chaudhuri, P. P. Theory and applications of cellular automata in cryptography. *IEEE TRANSACTIONS ON COMPUTERS 43* (1994), 1346–1357.

[27] S.R. Blackburn, S. M., and Paterson, K. Comments on theory and applications in cryptography. *IEEE TRANSACTIONS ON COMPUTERS 46* (1997), 637–638.

[28] Un-Sook Choi, S.-J. C., and Hwang, Y.-H. Analisys of linear group $gf(2^p)$ cellular automata. *Springer-Verlag Berlin Heidelberg* (2008), 136–143.

[29] van Oorschot. Scott A. Vanstone, A. J. M. P. C. *Handbook of Applied Cryptography*, 1 ed. CRC Press, United States of America, 1997.

[30] Wolfram, S. Computation theory of cellular automata. *Communications in Mathematical Physics* (1984).

[31] Wolfram, S. Cryptography with cellular automata. *The Institute for Advanced Study, Princeton NJ 08540.* (1985).

[32] Wolfram, S. Origins of randomness in physical systems. *Physical Review Letters 55* (1985), 449–452.

[33] Wolfram, S. *A New Kind of Science*, 2 ed. Wolfram Media, Champaign, IL, 2002.

Apéndice A

AC sobre campos finitos

Un AC (2,1) tiene una vecindad de 3 células, cada una de las cuales está en uno de dos estados. Existen 8 posibles vecindades y el número total de reglas posibles para ellas es de 256. La Tabla A.1 muestra algunas reglas del conjunto de 256 reglas y su función de transición, donde $\oplus$ denota al operador lógico XOR y $q_i(t)$ denota el estado de la i-ésima célula en el instante de tiempo t, q_{i-1} y q_{i+1} son los estados de los vecinos izquierdo y derecho.

Tabla A.1: Reglas de un AC aditivo.

Reglas con XOR (AC lineal)	Reglas con XNOR (reglas complemento)
R 60: $q_i(t+1) = q_{i-1}(t) \oplus q_i(t)$	R 195: $q_i(t+1) = \overline{q_{i-1}(t) \oplus q_i(t)}$
R 90: $q_i(t+1) = q_{i-1}(t) \oplus q_{i+1}(t)$	R 165: $q_i(t+1) = \overline{q_{i-1}(t) \oplus q_{i+1}(t)}$
R 102: $q_i(t+1) = q_i(t) \oplus q_{i+1}(t)$	R 153: $q_i(t+1) = \overline{q_i(t) \oplus q_{i+1}(t)}$
R 150: $q_i(t+1) = q_{i-1}(t) \oplus q_i(t) \oplus q_{i+1}(t)$	R 105: $q_i(t+1) = \overline{q_{i-1}(t) \oplus q_i(t) \oplus q_{i+1}(t)}$
R 170: $q_i(t+1) = q_{i+1}(t)$	R 85: $q_i(t+1) = \overline{q_{i+1}(t)}$
R 204: $q_i(t+1) = q_i(t)$	R 51: $q_i(t+1) = \overline{q_i(t)}$
R 240: $q_i(t+1) = q_{i-1}(t)$	R 15: $q_i(t+1) = \overline{q_{i-1}(t)}$

Si las células del AC obedecen a la misma función de transición, se dice que el AC es *uniforme*, de lo contrario es un AC *híbrido*. Si la función de transición que obtiene la siguiente iteración de células de un AC, únicamente usa funciones XOR, entonces el AC es llamado *lineal*. Si el AC usa reglas XOR y XNOR, entonces el AC es llamado *aditivo*.

Un *vector regla* es un vector unidimensional que representa las reglas que se aplican a las diferentes células de un AC. La i-ésima posición del vector corresponde a la regla que se aplica a la i-ésima célula del AC. Un AC es *periódico*, si las células extremas son adyacentes entre ellas. Un AC es *acotado*, si el vecino izquierdo (derecho) de la primer (última) célula, está conectado a un estado lógico 0.

Una característica importante sobre los AC aditivos es el hecho de que el estado de cada célula puede ser representado por un elemento del conjunto $\{0, 1\}$ y debido a que este conjunto también es el conjunto de elementos del campo binario GF(2), entonces esta clase de autómatas es llamada **AC sobre GF(2)**. En la siguiente sección se analizará la extensión de campos finitos sobre AC, además de mostrar su caracterización usando álgebra matricial. Todas las definiciones y los fundamentos matemáticos vistos en secciones anteriores sirven para entender el modelo planteado en la tesis.

A.1. Caracterización de AC Aditivos usando álgebra matricial

El estado de un AC de n células, en el instante de tiempo t, puede ser caracterizado utilizando álgebra matricial [21, 23]. Un AC de n células es caracterizado por una *matriz característica* con dimensiones $n \times n$, donde la matriz opera sobre GF(2). La matriz característica T es construída de la siguiente manera:

$$T = \{t_{i,j}\} = \begin{cases} 1, & \text{Si el estado de la } i\text{-ésima célula depende del estado actual} \\ & \text{de la } j\text{-ésima célula.} \\ 0, & \text{de otra manera.} \end{cases} \tag{A.1}$$

Por ejemplo, sea un AC de cuatro células donde las células obedecen al vector regla $< 150, 150, 90, 150 >$. Ahora bien, según la regla número 150 de la Tabla A.1, la función de transición para la i-ésima célula, representada por q_i en el instante de tiempo $t + 1$, depende de las céluas q_{i-1}, q_{i+1} y q_i. Para la regla número 90 la función de transición para la i-ésima célula en el instante de tiempo $t + 1$ depende de las células q_{i-1} y q_{i+1}. Siguiendo esto, la matriz característica del AC de cuatro células está dada por:

$$T = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

y el polinomio característico de la matriz T está dado por $p_T(x) = x^4 + x^3 + 1$.

Definición A.1 *Dada una una matriz T, la matriz tiene un* polinomio característico [21] *definido por*

$$p_T(x) = det(T - xI) \tag{A.2}$$

donde I es la matriz identidad.

Dado que la función XNOR no es lineal y no puede ser representado por una notación matricial, la transición de estados de un AC aditivo con reglas XNOR se construye de la siguiente manera.

Definición A.2 *Un* vector complemento *de n bits o un* vector de inversión *asociado a un AC aditivo de n células es un vector de n bits en el cual un 1 en la i-ésima posición indica que la regla en la i-ésima célula es una regla aditiva* [21].

Existe un AC lineal que corresponde a un AC aditivo cuyo comportamiento en la transición de estados puede ser utilizado para encontrar el comportamiento de un AC aditivo. Para un AC aditivo con el vector de inversión F, el estado siguiente $S(t+1)$ es representado por:

$$S(t+1) = T \cdot S(t) \oplus F, \tag{A.3}$$

donde la matriz T corresponde a la caracterización del AC lineal.

Dependiendo del comportamiento de su diagrama de transición, un AC puede ser clasificado en dos clases: **AC grupo** y **AC no grupo** [24].

Definición A.3 *Si cada uno de los estados en el diagrama de transición de estados de un AC tiene un único estado predecesor y un único estado sucesor, entonces el AC es llamado* AC grupo. *Un AC grupo es identificado por la condición de que la matriz T es no singular* [21]. *En otras palabras, debe existir algún entero positivo p tal que*

$$T^p = I, \tag{A.4}$$

$$S(t+p) = T^p \cdot S(t) = S(t) \tag{A.5}$$

donde I es la matriz identidad.

Un AC grupo es clasificado en: *AC grupo de tamaño máximo* Figura A.1 y *AC grupo de tamaño no máximo* Figura A.2. Un AC de n células de tamaño máximo tiene un polinomio característico *primitivo*$^{\clubsuit}$ y su diagrama de transición de estados tiene un ciclo de tamaño $(2^n - 1)$ que cubre todos los estados diferentes de cero.

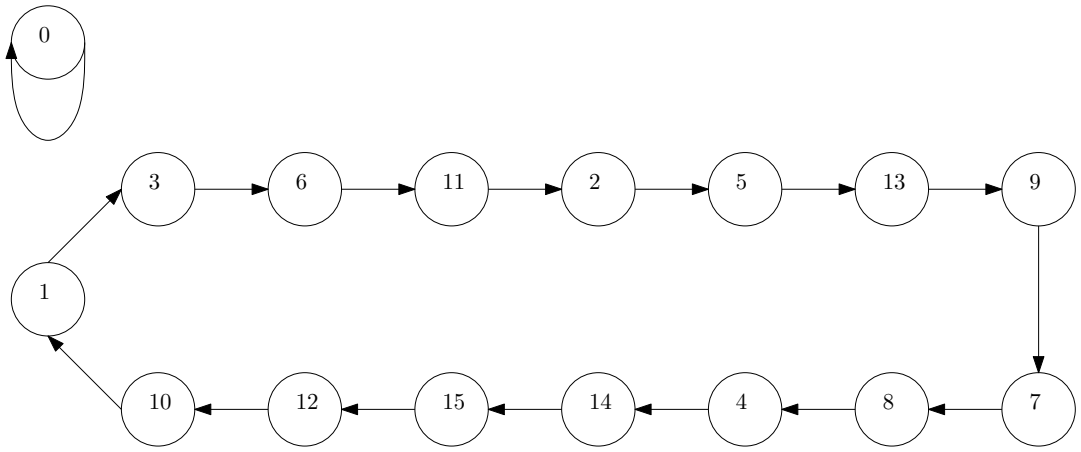

Figura A.1: Diagrama de transición de un AC grupo de cuatro células de tamaño máximo.

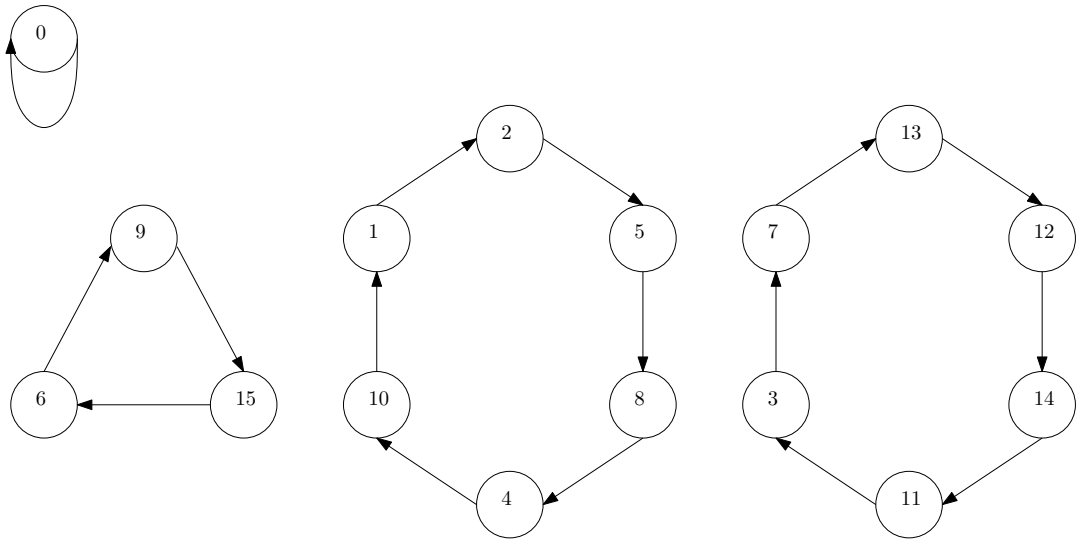

Figura A.2: Diagrama de transición de un AC grupo de cuatro células de tamaño no máximo.

A.2. Extensión de AC sobre GF(2) a AC sobre GF(2^p)

Un autómata celular sobre GF(2^p) puede ser visto como una extensión de un AC sobre GF(2). El AC de arreglos de células, espacialmente interconectadas en una manera regular, cada célula evoluciona en estados donde cada estado es un elemento de GF(2^p) (Figura A.3). Una célula de un AC sobre GF(2^p) puede verse como un conjunto de p células de un AC sobre GF(2) (Figura A.4). La caracterización completa de este tipo de autómatas celulares puede ser vista en [24, 28], aquí sólo presentamos lo necesario para nuestro trabajo.

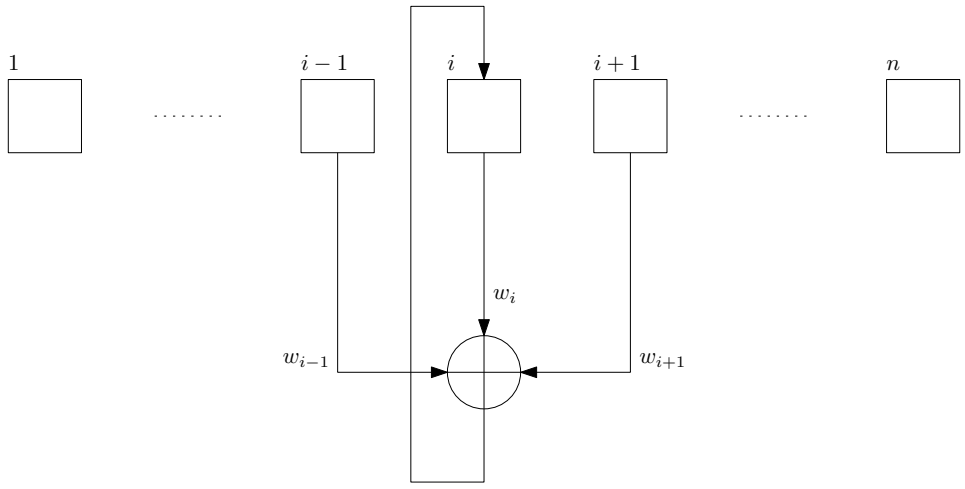

Figura A.3: Estructura de un AC sobre GF(2^p).

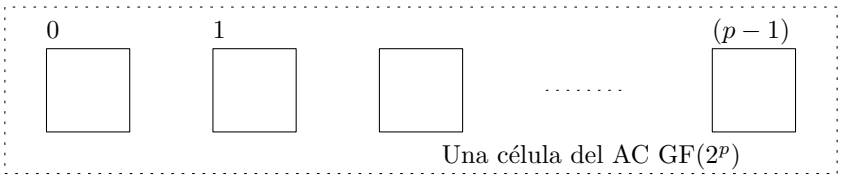

Figura A.4: Estructura de una célula.

Si tenemos un vector vecindad de tres células, el estado siguiente de la i-ésima célula es una función que hace una combinación ponderada entre los estados presentes de la i-ésima célula y sus vecinos $i-1$ y $i+1$, izquierdo y derecho respectivamente, y los pesos w_i, que son elementos del campo finito GF(2^p).

Por lo tanto si $q_i(t)$ representa la i-ésima célula de un AC sobre GF(2^p), en el instante de tiempo t, entonces

$$q_i(t+1) = \phi(w_{i-1}q_{i-1}(t), w_i q_i(t), w_{i+1}q_{i+1}(t)) \tag{A.6}$$

donde ϕ es la función de transición de la i-ésima célula que especifica la combinación de pesos w_{i-1}, w_i y $w_{i+1} \in \mathrm{GF}(2^p)$. Las operaciones de adición y multiplicación obedecen a las reglas del campo $\mathrm{GF}(2^p)$.

La Figura A.3 representa una célula de un AC sobre $\mathrm{GF}(2^p)$. Cada célula puede evolucionar en un estado del conjunto de estados $\{0, 1, 2, ..., (2^p-1)\}$. La combinación entre las células, como se muestra en la Figura A.3, es ponderada en el sentido de que evolucionar en el estado siguiente de la i-ésima célula, los estados presentes $(i-1)$, i y $(i+1)$ son multiplicados con w_{i-1}, w_i y w_{i+1}, respectivamente y después sumados. Esto da lugar a una dependencia de la i-ésima célula sobre la combinación ponderada de las células vecinas.

La regla para un vector vecindad de 3 células de un AC sobre $\mathrm{GF}(2^p)$ es representada por un vector regla de tamaño 3, $< w_{i-1}, w_i, w_{i+1} >$ para todo $w_i \in \mathrm{GF}(2^p)$. Así, w_{i-1} indica la dependencia del peso de la célula sobre su vecino izquierdo, mientras que w_i y w_{i+1} indican la dependencia sobre la misma célula y su vecino derecho. Si el mismo vector regla se aplica a todas las células de un AC sobre $\mathrm{GF}(2^p)$, entonces tenemos un AC uniforme, de otra manera un AC híbrido.

Al igual que para un AC sobre GF(2), un AC sobre $\mathrm{GF}(2^p)$ de n células también puede ser caracterizado por una matriz característica T de $n \times n$ de la siguiente forma:

$$T = \{t_{i,j}\} = \begin{cases} w_{i,j}, & \text{Si el estado de la } i\text{-ésima célula depende del estado actual} \\ & \text{de la } j\text{-ésima célula por un peso } w_{i,j} \in \mathrm{GF}(2^p). \\ \\ \\ 0, & \text{de otra manera.} \end{cases} \tag{A.7}$$

Si restringimos a una dependencia de un vector vecindad de 3 células, entonces la matriz $T[i, j]$ tiene elementos diferentes de cero para valores de $j = (i-1)$, i y $(i+1)$. Por lo tanto T es una matriz tridiagonal con elementos en $\mathrm{GF}(2^p)$.

Por ejemplo, sea un AC de tres células sobre $\mathrm{GF}(2^2)$, acotado con elementos $\{0, 1, \alpha, \alpha^2\}$ que pertenecen a $\mathrm{GF}(2^2)$ y vectores regla $< 0, 0, \alpha >$, $< \alpha, 0, \alpha >$ y $< \alpha^2, 1, 0 >$ para cada célula respectivamente. La matriz característica T está dada por:

$$T = \begin{pmatrix} 0 & \alpha & 0 \\ \alpha & 0 & \alpha \\ 0 & \alpha^2 & 1 \end{pmatrix}$$

A.3. Representación matricial de elementos que pertenecen a GF(2^p)

Sea Y un AC sobre GF(2^p) de n células, el estado siguiente del autómata está dado por

$$Y = T \cdot X \tag{A.8}$$

donde X representa el estado actual del AC. La matriz T, de $n \times n$, es la matriz característica del AC, X y Y son vectores de $n \times 1$. Cada elemento $t_{i,j} = \alpha^k$ de la matriz característica es un elemento de GF(2^p) y puede ser reemplazado por una matriz $T^k_{GF(2)}$ que es la matriz característica de un AC de tamaño máximo sobre GF(2), cuyo polinomio característico es el polinomio generador de GF(2^p).

Definición A.4 *Si el polinomio* $f(x) = f_0 + f_1 x + \cdots + f_{p-1}x^{p-1} + x^p$ *es un polinomio irreducible sobre el campo GF(*q^p*), entonces la matrix* $p \times p$

$$\begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & f_0 \\ 1 & 0 & 0 & \cdots & 0 & f_1 \\ 0 & 1 & 0 & \cdots & 0 & f_2 \\ 0 & 0 & 1 & \cdots & 0 & f_3 \\ \cdot & & & & & \cdot \\ \cdot & & & & & \cdot \\ \cdot & & & & & \cdot \\ 0 & 0 & 0 & \cdots & 1 & f_{p-1} \end{pmatrix}$$

es la matriz companion [14].

Los elementos de los vectores X y Y, también son elementos $\alpha^i \in$ GF(2^p), si quisiéramos hacer referencia a alguno de estos elementos, entonces necesitamos una representación para cada α^i. El último vector columna de la matriz $T^i_{GF(2)}$ es usado como la representación vectorial de α^i.

Por ejemplo, sea el campo finito GF(2^3), la representación vectorial de los elementos de GF(2^3) es:

$$0 :< 000 >= 0$$

$$1 :< 100 >= 1$$
$$\alpha :< 010 >= 2$$
$$\alpha^2 :< 001 >= 4$$
$$\alpha^3 :< 110 >= 3$$
$$\alpha^4 :< 011 >= 6$$
$$\alpha^5 :< 111 >= 7$$
$$\alpha^6 :< 101 >= 5$$

y la matriz companion asociada al polinomio irreducible de GF(2^3) es

$$T_{GF(2)} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

donde el polinomio irreducible es $p(x) = 1 + x + x^3$ y α puede ser representado como la matriz $T_{GF(2)}$ [25]. Por ejemplo, la representación vectorial de $\alpha^6 = \alpha^2 \cdot \alpha^4$ está dada por

$$\alpha^6 = \alpha^2 \cdot \alpha^4 = T^2_{GF(2)} \cdot \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

A manera de resumen, si α es el generador del campo GF(2^p) y $p(x)$ es su respectivo polinomio generador, entonces tenemos un *AC de tamaño máximo de p células sobre GF(2)* cuyo polinomio característico es $p(x)$.

Dado un AC sobre GF(2^2) de tres céluas la matriz característica T del AC está dada por la Ecuación A.10. El polinomio generador de la extensión del campo es $p(x) = x^2 + x + 1$. La representación matricial del elemento generador del campo α, está dada por la Ecuación A.9, donde la matriz representa a un AC sobre GF(2) y su polinomio característico de la matriz es el polinomio generador de la extensión del campo.

$$\alpha = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \tag{A.9}$$

$$T = \begin{pmatrix} 0 & \alpha & 0 \\ \alpha & 0 & \alpha \\ 0 & \alpha^2 & 1 \end{pmatrix} \tag{A.10}$$

Si calculamos las potencias sucesivas de α, entonces obtendremos los demás elementos del campo (α^2 y α^3), Ecuación A.11.

$$\alpha = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \quad \alpha^2 = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \quad \alpha^3 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{A.11}$$

Por otro lado si sustituimos cada elemento de la matriz característica del autómata celular por su correspondiente matriz binaria de 2×2, obtendremos una matriz binaria de 6×6, ver en Ecuación A.12.

$$T = \begin{pmatrix} 0 & \alpha & 0 \\ \alpha & 0 & \alpha \\ 0 & \alpha^2 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \tag{A.12}$$

A.4. Caracterización del grupo lineal del AC sobre GF(2^p)

Si $f_t(x)$ representa el estado de un AC sobre GF(2^p) en el instante de tiempo t, entonces el estado siguiente está dado por la ecuación:

$$f_{t+1}(x) = T \cdot f_t(x) \tag{A.13}$$

El estado del autómata después de m iteraciones está dado por

$$f_{t+m}(x) = T^m \cdot f_t(x) \tag{A.14}$$

Las operaciones matriciales son evaluadas con las operaciones de adición y multiplicación definidas en el campo GF(2^p).

Si el AC bajo la operación de transformación con T forma un grupo cíclico, entonces debe existir un entero m tal que,

$$T^m = I \tag{A.15}$$

donde I es la matriz identidad y

$$f_m(x) = T^m \cdot f_(x) = f(x) \tag{A.16}$$

Un autómata con esta propiedad se conoce como **AC grupo**. La matriz T opera sobre GF(2^p).

Los siguientes teoremas son resultado fundamental en la caracterización del comportamiento de la transición de estados del AC sobre GF(2^p). Las pruebas se pueden consultar en [24].

Teorema A.1 *Un autómata celular es un AC grupo si y sólo si el determinante de la matriz T es diferente de cero* [24].

Teorema A.2 *Un AC grupo tiene ciclos de tamaño p o factores de p con estado inicial diferente de cero si y sólo si det(T^p + I) = 0* [24].

Un AC de n células de *tamaño máximo* es caracterizado por la presencia de todos sus estado diferentes de cero en un sólo ciclo (Figura A.1). En el caso de un AC sobre GF(2^p) de n células, el tamaño de su ciclo es de $(2^{np} - 1)$. El polinomio característico del AC es un *polinomio primitivo*. La definición de polinomio primitivo es invariante en la extensión.

Por ejemplo, el AC sobre GF(2^2) de 3 células, tiene matriz característica T, es un AC de tamaño máximo y los elementos de la matriz son elementos de la extensión del campo con su valor decimal de su representación vectorial,

$$T = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 2 \\ 0 & 1 & 2 \end{pmatrix}$$

el polinomio característico es

$$p_T(x) = x^3 + 2x^2 + 3x + 2. \tag{A.17}$$

Debido a que el polinomio característico $p_T(x)$ es primitivo, entonces el AC contiene un ciclo de tamaño máximo, según el Teorema A.2 el $det(T^{63} + I) = 0$, por lo cual el entero k más pequeño para el cual este polinomio divide a $x^k + 1$ es $k = 2^{2 \cdot 3} - 1 = 63$. Por lo tanto es un AC con un ciclo de tamaño máximo.

Apéndice B

Glosario de términos

Alfabeto	Conjunto finito de letras o símbolos.
Algoritmo	Es una Máquina de Turing y su complejidad es el número de pasos que tarda en parar.
Cifrado por flujo	Es un cifrado por bloques donde el tamaño del bloque es igual a uno.
Estructura algebraica	Es un sistema $(X, *_1, ..., *_n)$, formado por un conjunto $X \neq \emptyset$ y una colección de n operaciones distintas $*_1, ..., *_n$ sobre el conjunto.
FLOPS	FLoating-point Operations Per Second. Operaciones de punto flotante por segundo.
Grupo abeliano	Grupo en el que la operación es conmutativa.
Grupo cíclico	Es un grupo el cual es generado por un elemento que pertenece al grupo.
Isomorfismo	Es una función biyectiva entre dos conjuntos, la cual respeta la estructura de los conjuntos.
Polinomio irreducible	Un polinomio $p(x)$ de grado m sobre GF(2) es irreducible sobre GF(2) si $p(x)$ no es divisible por algún polinomio $g(x)$ de grado n sobre GF(2), donde $0 < n < m$.
Polinomio primitivo	Es un polinomio irreducible $p(x)$ de grado m, donde el entero positivo más pequeño n para el cual $p(x)$ divide a $x^n + 1$ es $n = 2^m - 1$
Transformación	Es una función biyectiva de un conjunto en sí mismo.

Apéndice C

Textos planos usados en la simulación del criptosistema

Archivo: `HimnoNacionalMexicano.txt` (*Archivo de texto*)

```
HIMNO NACIONAL MEXICANO

Mexicanos, al grito de guerra
El acero aprestad y el bridón;
Y retiemble en sus centros la tierra
Al sonoro rugir del cañón.

Ciña ¡Oh Patria! tus sienes de oliva
de la paz el arcángel divino,
que en el cielo tu eterno destino
por el dedo de Dios se escribió.
Mas si osare un extraño enemigo
profanar con su planta tu suelo,
piensa ¡Oh Patria querida! que el cielo
un soldado en cada hijo te dio.

¡Guerra, guerra sin tregua al que intente
de la patria manchar los blasones!,
¡guerra, guerra! los patrios pendones
en las olas de sangre empapad.
¡Guerra, guerra! en el monte, en el valle,
los cañones horrísonos truenen
y los ecos sonoros resuenen
con las voces de ¡Unión! ¡Libertad!

Antes, Patria, que inermes tus hijos
bajo el yugo su cuello dobleguen,
```

```
tus campiñas con sangre se rieguen,
sobre sangre se estampe su pie.
Y tus templos, palacios y torres
se derrumben con hórrido estruendo,
y tus ruinas existan diciendo:
de mil héroes la Patria aquí fue.

¡Patria! ¡Patria! tus hijos te juran
exhalar en tus aras su aliento,
si el clarín con su bélico acento
los convoca a lidiar con valor.
¡Para ti las guirnaldas de oliva!
¡un recuerdo para ellos de gloria!
¡un laurel para ti de victoria!
¡un sepulcro para ellos de honor!
```

Archivo: `textoa.txt` (*Archivo de texto*)

```
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

Archivo: `textoab.txt` (*Archivo de texto*)

```
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
aaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb
```