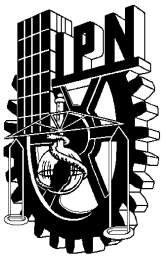


INSTITUTO POLITECNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

CAMPOS FINITOS JV EN EL CONJUNTO DE LAS PERMUTACIONES: UNA APLICACIÓN A LA CRIPTOGRAFÍA

T E S I S

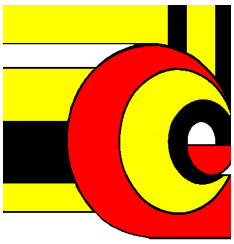
QUE PARA OBTENER EL GRADO DE
DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

ROLANDO FLORES CARAPIA

DIRECTORES DE TESIS:

DR. VÍCTOR MANUEL SILVA GARCÍA
DR. CORNELIO YÁÑEZ MÁRQUEZ



MÉXICO, D.F. ENERO DE 2011

Índice General

Resumen

Esta tesis trata sobre criptografía. Específicamente, en este trabajo de tesis se propone, enuncia y demuestra el teorema JV, y se toma como base teórica para construir campos finitos en el conjunto de las permutaciones, los cuales se aplican en la construcción de criptosistemas; además, se propone, enuncia y demuestra el Teorema Factorial. Los teoremas JV y Factorial constituyen la base teórica para implementar un nuevo algoritmo de cifrado: el Criptosistema Factorial.

Los criptosistemas diseñados, implementados y aplicados en este trabajo de tesis son competitivos con la norma internacional AES y una de sus aplicaciones principales es el reforzamiento de algoritmos criptográficos conocidos, como el DES y el TripleDES.

Primeramente se propone, enuncia y demuestra el Teorema JV. Acto seguido, se definen dos operaciones binarias y una operación unaria a fin de construir campos finitos en el conjunto de las permutaciones, tomando como base teórica el Teorema JV.

Luego, se propone, enuncia y demuestra el Teorema Factorial, el cual sirve como base teórica, juto con el Teorema JV, para proponer e implementar el algoritmo del Criptosistema Factorial.

Se realiza un análisis de la forma en que se fortalecen los criptosistemas simétricos DES y Triple-DES al aplicar los resultados de esta tesis y se implementa este fortalecimiento; además, se usa como base el Teorema Factorial en el cifrado de cadenas de 192 bits.

Finalmente, se diseña una aplicación donde se usan los resultados originales de este trabajo de tesis en el envío de cheques electrónicos cifrados mediante el algoritmo TripleDES con permutación variable.

Abstract

This thesis deals specifically with cryptography. This work proposes, formulates and demonstrates the JV theorem taken as a theoretical base to construct finite fields in a group of permutations which are applied in the construction of crypto-systems. It is also proposes formulating and demonstrating the factorial theorem. Factorial and JV theorem constitute the theoretical base to implement a new coded algorithm: the factorial crypto-system.

The crypto-systems designed, implemented and applied in this thesis are competitive with the international AES norm, and one of their principal applications is the re-enforcement of known cryptographic algorithms such as DES and triple DES.

First, the formulation and demonstration of the JV theorem is proposed. Then two binary and one unary operations are defined with the aim to construct finite fields in a group of permutations taken the JV theorem as theoretical base.

Next, formulating and demonstrating the factorial theorem is proposed, which serves as a theoretical base, together with the JV theorem, to propose and implement the factorial crypto-system algorithm.

An analysis is made of the form to strengthen the DES and triple DES symmetrical crypto-system, apply the thesis results and implement this strengthening. Also the factorial theorem is used as a base in decoding chains of 192 bits.

Finally, an application to use the original results of this thesis is designed for sending coded electronic checks through the triple DES algorithm with a variable permutation.

CAPÍTULO 1

Introducción

En esta tesis se usan los teoremas JV y Factorial, después de ser demostrados, para realizar dos tareas relevantes: la construcción de campos finitos en el conjunto de las permutaciones, y la implementación de nuevos algoritmos de cifrado de información.

Los criptosistemas diseñados, implementados y aplicados en este trabajo de tesis son competitivos con la norma internacional AES y una de sus aplicaciones principales es el reforzamiento de algoritmos criptográficos conocidos, como el DES y el TripleDES [1].

1.1 Antecedentes

El ser humano, desde tiempos remotos, ha buscado maneras de proteger la información que considera valiosa; para ello, se han inventado algoritmos que tienden a esconder la información de posibles intrusos. Se puede decir de manera sintética que intervienen tres personajes en esta actividad: el que envía la información, el que recibe el mensaje y el que desea averiguar el mensaje entre los dos primeros [2].

La palabra Criptología tiene raíces griegas (criptos = oculto y logos = tratado, ciencia) y se puede considerar que es el nombre genérico de dos disciplinas que son opuestas y a su vez se complementan, a saber: Criptografía y Criptoanálisis.

La Criptografía se encarga de cifrar o encriptar un mensaje de texto claro, esto es, de ocultar o enmascarar la información que se considera confidencial y, en contraparte, el Criptoanálisis se encarga de romper el proceso de cifrado para recuperar la información original o el texto claro [3].

Es cuantioso el material que se ha ido acumulando, a través de los años, en relación con el tema de la Criptología. Se ha evolucionado desde los métodos como el de Julio César, hasta los modernos estándares como el DES y el AES [4].

Cuando se encuentran debilidades en un algoritmo de cifrado, se intenta reforzarlo mediante la mejora de las técnicas matemáticas que intervienen en el diseño, y un claro ejemplo es la aparición del TripleDES, después de que alguien logró romper el DES [14].

Actualmente, el estándar internacional vigente es el AES [29].

1.2 Justificación

Es conocido que los algoritmos relacionados con sistemas tales como DES, Triple-DES, SPN y AES emplean básicamente tres tipos de operaciones: permutaciones, sustituciones y la función lógica OR exclusiva [2]. Las permutaciones se definen mediante tablas y se consideran fijas, y hasta el momento la posibilidad de representar una permutación mediante un entero no negativo no ha sido explorada.

De manera natural surge la necesidad de construir tal algoritmo que permita asignar una permutación a un número natural: esa es precisamente la justificación de esta tesis.

De hecho, la construcción de tal algoritmo debe definir una función biyectiva, la cual propicia que la permutación sea considerada como una clave, porque ahora podrá ser variable. Así, se pueden construir criptosistemas rápidos y de gran complejidad por medio de esta idea novedosa.

1.3 Objetivos

Objetivo General

Proponer, enunciar y demostrar el Teorema JV y, con base en este teorema, construir campos finitos en el conjunto de las permutaciones a fin de aplicarlos en la construcción de criptosistemas. Proponer, enunciar y demostrar el Teorema Factorial, e implementar el nuevo Criptosistema Factorial. Estos criptosistemas serán competitivos con la norma internacional AES y reforzarán algunos algoritmos conocidos de cifrado.

Objetivos Particulares

1. Proponer, enunciar y demostrar el Teorema JV.
2. Definir dos operaciones binarias y una operación unaria a fin de construir campos finitos en el conjunto de las permutaciones, tomando como base teórica el Teorema JV.
3. Proponer, enunciar y demostrar el Teorema Factorial.
4. Tomando como base teórica el Teorema JV y el Teorema Factorial, proponer e implementar el algoritmo del Criptosistema Factorial.
5. Elaborar un análisis de la forma en que se fortalecen los criptosistemas simétricos DES y Triple-DES al aplicar los resultados de esta tesis. Implementar este fortalecimiento.
6. Usar como base el Teorema Factorial en el cifrado de cadenas de 192 bits.
7. Diseñar una aplicación donde se usen los resultados originales de este trabajo de tesis en el envío de cheques electrónicos cifrados mediante el algoritmo TripleDES con permutación variable.

1.4 Contribuciones

Las contribuciones de este trabajo de tesis, tanto teóricas como prácticas, son:

1. Dos operaciones binarias y una operación unaria que permiten construir campos finitos en el conjunto de las permutaciones, tomando como base teórica el Teorema JV.
2. El Teorema Factorial.
3. El algoritmo del Criptosistema Factorial, el cual es más robusto que AES.
4. Implementación del fortalecimiento de los criptosistemas simétricos DES y Triple-DES.
5. Aplicación del Teorema Factorial en el cifrado de cadenas de 192 bits.
6. Uso del algoritmo TripleDES con permutación variable en el envío de cheques electrónicos cifrados.

1.5 Organización del documento

En este capítulo se han presentado: los antecedentes, la justificación, los objetivos del trabajo de tesis y las contribuciones. El resto del documento de tesis está organizado de la siguiente manera:

En el capítulo 2 se describen brevemente algunos conceptos del estado del arte en criptografía, mientras que en capítulo 3 se presentan las herramientas matemáticas que se usan en el desarrollo de la tesis.

El capítulo 4 es la parte más relevante de este documento, dado que ahí se definen las dos nuevas operaciones binarias y la operación unaria que permitirán la construcción de un campo finito basado en el Teorema JV; se enuncia y se demuestra el Teorema Factorial y además se propone un nuevo criptosistema: el Criptosistema Factorial, el cual permitirá reforzar algunos algoritmos conocidos de cifrado.

Los resultados experimentales están contenidos en el capítulo 5: una aplicación a la criptografía de los campos finitos JV, en la que se analiza la forma en que se fortalecen los criptosistemas simétricos DES y Triple-DES al aplicar los resultados de esta tesis, el Criptosistema Factorial, el uso del Teorema Factorial en el cifrado de cadenas de 192 bits, y finalmente se presenta una interesante aplicación de los resultados originales de este trabajo de tesis: el envío de cheques electrónicos cifrados mediante el algoritmo TripleDES con permutación variable.

En el capítulo 6 se presentan las conclusiones y recomendaciones para trabajo a futuro. Se anexa un apéndice donde se incluye una implementación en hardware del Criptosistema Factorial y, finalmente, las referencias bibliográficas.

CAPÍTULO 2

Estado del Arte

Este capítulo consta de cuatro secciones; en la sección 2.1 se presentan someramente y se ejemplifican dos algoritmos de cifrado de datos que se usaron en los primeros tiempos de la criptografía, mientras que en la sección 2.2 se describe brevemente el estándar DES.

Las secciones 2.3 y 2.4 tratan, respectivamente, acerca de criptoanálisis y de criptosistemas, conceptos muy importantes en el desarrollo de esta tesis.

2.1 Primeros tiempos

Como una consecuencia de la necesidad que se tienen en los círculos de poder de conservar para sí la información delicada, la Criptografía permaneció durante siglos vinculada a la clase política y militar. Es famoso el sistema que usaba Julio César durante el siglo I AC para ocultar información.

El alfabeto latino que se usaba en aquel entonces constaba de estas 21 letras: A B C D E F G I J L M N O P Q R S T U V X, y el procedimiento para ocultar información de aquellos tiempos era sustituir a la primera letra A por la cuarta D, la segunda B por la quinta E y así sucesivamente [32].

Si se llama Y_i a las letras encriptadas, X_i a las letras del texto claro y Z_i a la clave, entonces el cifrado anterior se puede expresar utilizando la aritmética modular [6] mediante la siguiente fórmula:

$$Y_i = X_i + Z_i \pmod{21}$$, en este caso Z_i es siempre D o 3.

Un procedimiento también famoso de cifrar es el "Cifrado Vigenere" (1586), [2] y [32]. Este procedimiento es una generalización del caso anterior y con el objeto de ilustrarlo consideremos el siguiente ejemplo, en el que se trabaja con el alfabeto del idioma español, el cual se escribe a continuación:

A	B	C	D	E	F	G	H	I	J	K	L	M	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13

Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
14	15	16	17	18	19	20	21	22	23	24	25	26

Como puede observarse las letras son numeradas del 0 al 26, con la finalidad de aplicar la aritmética modular.

Si el mensaje es "TORTAS DE JAMON" y la clave es "ROLANDO", entonces el procedimiento de cifrado se puede expresar de la siguiente manera:

Mensaje	T	O	R	T	A	S	D	E	J	A	M	O	N	
Clave	R	O	L	A	N	D	O	R	O	L	A	N	D	
Mensaje Cifrado	L	D	C	T	N	V	R	V	X	L	M	B	P	

Como puede verificarse fácilmente se aplicó la fórmula $Y_i = X_i + Z_i \pmod{27}$, en donde de la misma manera que en el caso anterior Y_i representa las letras del texto cifrado, X_i las letras del texto claro y Z_i las de la clave.

2.2 DES

En la actualidad la situación ha cambiado drásticamente, porque el desarrollo de las comunicaciones y el uso masivo de las computadoras hace posible la transmisión y almacenamiento de grandes volúmenes de información confidencial que es necesario proteger. Entonces la Criptografía pasa de ser una necesidad de élite, para convertirse en una exigencia del ser humano [32]. En 1973 el NBS (National Bureau of Standard, USA) organizó un concurso solicitando un "algoritmo de encripción para la protección de datos de computador durante su transmisión y almacenaje". En 1974, la corporación IBM presentó, entre otros, una propuesta inspirada en su sistema propietario LUCIFER, que, convenientemente modificado dio lugar a "Data Encryption Standard" (DES) [32].

Antes de iniciar la descripción del algoritmo DES será necesario precisar la simbología que será utilizada. Las letras mayúsculas como X,Y,L,R,K representarán cadenas, entendiéndose por cadena a la concatenación de ceros y unos de longitud finita, en el entendido que para el caso de DES no será mayor a 64. Para la operación binaria "XOR", definida a continuación, se utiliza el símbolo $\oplus$:

Si x_i, y_i son elementos de las cadenas X, Y entonces

$$x_i \oplus y_i = \begin{cases} 0 & \text{si } x_i = y_i \\ 1 & \text{si } x_i \neq y_i \end{cases}$$

Note que los elementos de una cadena dada se escribirán con minúsculas.

Para los símbolos del texto claro y del texto cifrado serán utilizados caracteres de 8 bits, lo que corresponde a los símbolos del código ASCII. A continuación se hará una descripción a grandes rasgos del algoritmo DES, el cual consta de tres pasos:

a) Dada una cadena de texto claro de 64 bits X, se construye otra cadena X_0 aplicando una permutación inicial fija IP a X, la que se denotará como IP(X) = X_0. La cadena X_0 se separa en dos subcadenas de 32 bits cada una, se escribirá como $X_0 = L_0 R_0$. L_0 serán los primeros 32 bits y R_0 los 32 bits restantes. La permutación inicial fija se ilustra a continuación:

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

b) Se aplican 16 iteraciones o rondas, en cada una de ellas se calculan L_i, R_i de acuerdo a las siguientes expresiones:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \text{ para } i = 1,2,..,16.$$

f es una función y por el momento se dirá que K_i es una cadena de 48 bits de longitud, la que a su vez se obtiene de una llave K que es una cadena de 64 bits de longitud, el algoritmo para obtener las K_i se describirá más adelante también.

c) Se aplica la permutación inversa IP^{-1} a la cadena $R_{16} L_{16}$ (note que primero aparece R_{16} y después L_{16}) para obtener el texto cifrado Y. La tabla de IP^{-1} se presenta a continuación:

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

2.3 Criptoanálisis

La disciplina del Criptoanálisis se encarga de romper los sistemas de cifrados; esto es, de buscar los puntos débiles de un algoritmo de cifrado con el objeto de encontrar la clave o llave utilizada, ya que con el conocimiento de la clave o llave se puede recuperar la información cifrada. Otro aspecto importante que se debe mencionar, es el hecho de que la robustez de un algoritmo de cifrado no depende de su desconocimiento; al contrario, éste debe ser público para que la comunidad criptográfica lo analice y de esta forma se logren fortalecer sus puntos débiles. En este orden de ideas, se puede afirmar que otro de los objetivos del criptoanálisis es descubrir los puntos débiles de un algoritmo con la finalidad de fortalecerlo, y este es otro de los aspectos que serán presentados en este trabajo de tesis.

No se puede hablar de un procedimiento general de Criptoanálisis, pues cada algoritmo de cifrado ha de ser atacado mediante una técnica adecuada, dependiendo de su estructura. Es claro que el Criptoanálisis depende también de cuanta información se posea del procedimiento de cifrado; a continuación se dará una lista en orden de facilidad creciente de situaciones frecuentes [1].

a) Sólo se conoce el texto cifrado.- En realidad esta es la situación más difícil si se considera que los sistemas de cifrado actuales son bastante robustos. De hecho, si este fuese el caso, aún podría considerarse al DES simple como un criptosistema seguro, ya que en todos los ataques que se le han hecho se incluye, además del texto cifrado, una porción de texto claro cuando menos.
b) Se conoce una parte del texto claro y su correspondiente texto cifrado.- Es una situación común en todos los ataques que se le han hecho a DES.
c) Se conoce una parte del texto claro y su correspondiente texto cifrado, además, de una pequeña parte del protocolo de cifrado.- Esta situación es la que se presentará en este trabajo.
d) Se conoce parte de la clave o llave, o se puede limitar el espacio de los posibles valores que pudiera tomar la clave o llave.

2.3 Criptosistemas

Definición.- Un criptosistema es una tupla de cinco elementos (**P, C, K, E, D)** que cumplen con lo siguiente:

a) **P** es el conjunto finito de todos los posibles textos claros.
b) **C** es el conjunto finito de todos los posibles textos cifrados.

Es claro que ambos conjuntos **P, C** descansan sobre algún alfabeto; por ejemplo, para textos claros podrían utilizarse el alfabeto del idioma español o inglés.

c) **K** es el conjunto – llave, que tiene un número finito de posibles llaves. El número de llaves puede ser grande pero debe de ser finito.

d) Para cada K $\in$ **K**, se definen dos funciones inyectivas , $e_K \in$ **E** y $d_k \in$ **D** tales que e_K: **P** $\rightarrow$ **C** y d_k: **C** $\rightarrow$ **P** y además, $d_k(e_K(X)) = X$ para cualquier X $\in$ **P;** esto es, d_k es la función inversa de e_K.

El término cadena es un concepto que se citará con frecuencia en este trabajo, por lo que es conveniente definirlo, pero antes será necesario mencionar qué se entiende por un alfabeto.

Definición.- Un conjunto no vacío finito de símbolos se llamará alfabeto [17].

Definición.- Una secuencia finita de elementos de un alfabeto se llamará cadena.

Si se supone que el alfabeto es el conjunto $\{0,1\}$, entonces una cadena podría ser 001111010011, que como se observa es una secuencia de 12 elementos del alfabeto $\{0,1\}$. De hecho, el texto claro y el texto cifrado se pueden considerar cadenas de símbolos, de tal forma que **P y C** son conjuntos cuyos elementos son cadenas. Es claro que no son de interés la cadena vacía ni cadenas de longitud infinita.

Otro concepto muy utilizado en este trabajo es el de permutación, que se define así:

Definición.- Dado un conjunto **S** con un número finito de elementos, una permutación P definida sobre **S** es una función biyectiva que va de **S** a **S**, lo cual se representa de la siguiente manera: P: **S** $\rightarrow$ **S**.

Como ejemplo para aclarar la definición anterior, supongamos que **S**= {1,2,3,4,5}, entonces un caso particular para P: **S** $\rightarrow$ **S** sería: P(1)=2, P(2)=5, P(3)=4, P(4)=1, P(5)=3. En este trabajo se usarán permutaciones sobre conjuntos, en donde los elementos de estos conjuntos representarán las posiciones de los caracteres de una cadena

Aunque previamente fue descrita la operación binaria $\oplus$ la cual está definida en cadenas de la misma longitud sobre el alfabeto $\{0,1\}$, no fueron mencionadas algunas propiedades importantes de esta operación.

Sea $\mathbf{H}_m$ el conjunto de las cadenas X de longitud m, sobre el alfabeto $\{0,1\}$; entonces la operación binaria $\oplus$ cumple con las siguientes propiedades:

a) *Si* $X,Y \in \mathbf{H}_m$ *entonces* $X \oplus Y = Y \oplus X$
b) *Si* X,Y *y* $Z \in \mathbf{H}_m$ *entonces* $(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z)$
c) $\forall X \in \mathbf{H}_m$ *se cumple que* $X \oplus X =$ ***0*** *donde* ***0*** *es una cadena de ceros de longitud m.*
d) $\forall X \in \mathbf{H}_m$ *se cumple que* $X \oplus$ ***0*** $= X$.
e) $\forall X \in \mathbf{H}_m$, *si* $\overline{X}$ *es el negado de la cadena X se cumple que* $X \oplus \overline{X} =$***1,*** *donde* ***1*** *es una cadena de unos de longitud m..*

De hecho, esta operación define un grupo abeliano [6], en el conjunto $\{0,1\}$.

CAPÍTULO 3

Materiales y Métodos

Este capítulo contiene las herramientas matemáticas (los materiales y los métodos) que se usarán de manera intensiva en el capítulo 4 a fin de describir adecuadamente las aportaciones originales de esta tesis.

El capítulo consta de tres secciones. En la sección 3.1 se presentan los elementos de la aritmética modular, incluyendo definiciones y algunos teoremas, y las secciones 3.2 y 3.3 tratan, respectivamente, acerca de los campos finitos y las permutaciones, dos temas fundamentales para el desarrollo de los conceptos y las aportaciones de esta tesis.

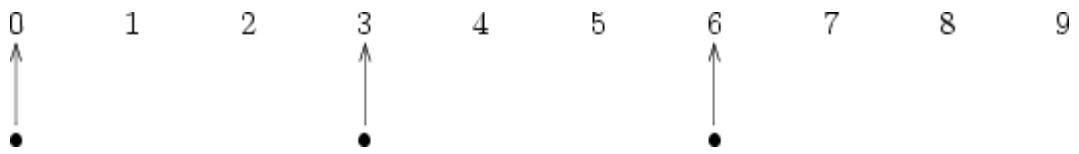
Todos los conceptos, definiciones, teoremas y ejemplos de las secciones de este capítulo están basados en las referencias [1], [2], [5] y [6].

3.1 Aritmética modular

La aritmética modular es un sistema aritmético para clases de equivalencia de números enteros llamadas clases de congruencia. Algunas veces se le llama “aritmética del reloj”, ya que los números “dan la vuelta” tras alcanzar cierto valor (el módulo). Por ejemplo, cuando el módulo es 12, entonces cualesquiera dos números que divididos por doce den el mismo resto son equivalentes (o "congruentes") uno con otro. Los números

$$..., -34, -22, -10, 2, 14, 26, ...$$

son todos "congruentes módulo 12" unos con otros, ya que cada uno deja el mismo resto, 2, cuando los dividimos por 12. La colección de todos esos números es una clase de congruencia. Se puede pensar en un "peine" (finito si queremos ver sólo unos cuantos números equivalentes alrededor del cero, o infinito si se quiere imaginarlos todos de una vez; se puede pensar también en una flecha con muchas barbas igualmente espaciadas y que apunta hacia un objeto, el de los números enteros, que se representa abajo)



No aparecen los negativos, que también están, ni por supuesto toda la ristra de enteros ni de barbas de la flecha. Además, se ha dibujado el caso de la aritmética 3-modular, por lo que

el peine tiene una "distancia" de 3 entre sus púas, y está apuntando a la clase de equivalencia del 0. Si desplazamos el peine un espacio, encontramos la clase de equivalencia del 1; cuando se llega al 3 y como el peine es infinito, no podemos distinguir esa situación de la situación inicial: hemos llegado otra vez a la clase del 0 = [0]; así que [1] + [1] + [1] = [0]. Por lo anterior, la operación "+" tiene el sentido geométrico relacionado con el verbo activo "desplazar".

El ejemplo anterior ilustra el hecho de que al sumar clases de equivalencia se obtiene otra clase; se puede, incluso, restarlas y multiplicarlas. Cuando el módulo es un número primo, siempre se puede dividir por una clase que no contenga el 0. Las siguientes definiciones y teoremas son de la referencia [6].

Definición:
Si *a* y *b* son enteros y *m* un entero positivo, entonces *a* es congruente con *b* módulo *m* si *m* divide *a* – *b*. Usamos la notación $a \equiv b (\mathrm{mod}\, m)$ para indicar que *a* es congruente con *b* módulo *m*. Si *a* y *b* no son congruentes módulo *m*, usamos la notación $a \not\equiv b (\mathrm{mod}\, m)$.

Teorema:
Sean *a* y *b* enteros y sea *m* un entero positivo, entonces $a \equiv b (\mathrm{mod}\, m)$ si y sólo si $a \bmod m = b \bmod m$.

Teorema:
Sea *m* un entero positivo, los enteros *a* y *b* son congruentes módulo *m* si y sólo si existe un entero *k* tal que *a=b+km*.

Demostración:
Si $a \equiv b (\mathrm{mod}\, m)$, entonces $m | (a - b)$. Esto quiere decir que existe un entero *k* tal que $a - b = km$, de donde tenemos $a = b + km$. Si existe un entero k tal que $a = b + km$, entonces $km = a - b$. Aquí, m divide $a - b$, por lo tanto tenemos que $a \equiv b (\mathrm{mod}\, m)$.

Teorema:
Sea *m* un entero positivo. Si $a \equiv b (\mathrm{mod}\, m)$ y $c \equiv d (\mathrm{mod}\, m)$, entonces

$$a + c \equiv (b + d) (\mathrm{mod}\, m) \quad \text{y} \quad ac \equiv (bd) (\mathrm{mod}\, m)$$

Demostración:
Como $a \equiv b (\mathrm{mod}\, m)$ y $c \equiv d (\mathrm{mod}\, m)$, existen dos enteros s y t con $b = a + sm$ y $d = c + tm$. Se tiene,

$$b + d = (a + sm) + (c + tm) = (a + c) + m(s + t)$$

es decir,

$$a + c \equiv (b + d) (\mathrm{mod}\, m)$$

$$bd = (a + sm)(c + tm) = (ac) + m(at + cs + stm)$$

es decir,

$$ac \equiv (bd) (\mathrm{mod}\, m)$$

3.2 Campos finitos

Los campos finitos son una herramienta matemática muy importante para el control de errores, al codificar códigos y, desde luego, para la Criptografía.

Un campo finito o campo de Galois (llamado así por Évariste Galois) es un campo que contiene un número finito de elementos [6], [11].

Dado que todo campo de característica 0 contiene a los racionales y es por lo tanto infinito, todos los campos finitos tienen característica prima y, por lo tanto, su tamaño (o cardinalidad) es de la forma p^n, para p primo y n > 0 entero.

Para un primo p, los enteros módulo p forman un cuerpo de p elementos, denotado por Z/pZ, Fp, o GF(p).

Definición:
Un campo $(F,+,\cdot)$ consiste en un conjunto F junto con dos operaciones binarias + y ·, tal que

- $(F,+,\cdot)$ es un anillo.
- $(F - \{0\},\cdot)$ es un grupo conmutativo.

Un subcampo F del campo $(K,+,\cdot)$ es un subconjunto de K que es un campo usando las mismas operaciones que K.

Si F es un subcampo de K, entonces K es llamado un campo extendido de F denotado por K/F.

Para K/F, el grado de K sobre F es $[K:F]$ la dimensión de K como un vector en el espacio sobre F.

Definición:
Un campo finito es un campo F que contiene un número finito de elementos. El orden de F es el número de elementos de F.

Propiedades (existencia y unicidad de los campos finitos)
Si F es un campo finito, entonces F contiene p^m elementos para un primo dado p y un entero dado $m \geq 1$.

Para cada p^m, existe un campo finito de orden p^m y es único. Este campo es denotado por F_{p^m} o por GF(p^m).

Si F_q es un campo finito de orden $q = p^m$, con p primo, entonces la característica de F_q es p. Por otra parte, F_q contiene una copia de Z_p como un subcampo. Aquí F_q puede ser visto como una extensión del campo de Z_p de grado m.

Sea F_q un campo finito de orden $q = p^m$. Entonces cada subcampo de F_q tiene orden p^n, donde n es un divisor positivo de m. Si n es un divisor positivo de m, entonces hay exactamente un subcampo de F_q de orden p^n.

Definición:
El campo finito F_q con uno o más elementos forma un grupo dentro de la multiplicación llamado el grupo multiplicativo de F_q, denotado por F_q^*.

F_q^* es un grupo cíclico de orden q-1. Aquí $a^q = a$ para todo $a \in F_q$.

Definición:
Un generador del grupo cíclico F_q^* es llamado un elemento primitivo o generador de F_q.

Si $a, b \in F_q$, un campo finito de característica p, entonces
$(a + b)^{p^t} = a^{p^t} + b^{p^t}$ para todo $t \geq 0$.

3.3 Permutaciones y combinaciones

Una permutación es un reordenamiento de una colección de objetos. Por ejemplo, si se tienen tres personas, Pedro, Luis y Carlos, cada una de las diferentes formas de ordenarse en fila:

Pedro-Luis-Carlos, Pedro-Carlos-Luis, Luis-Pedro-Carlos, Luis-Carlos-Pedro, Carlos-Pedro-Luis, Carlos-Luis-Pedro

es una permutación de ellos.
También se usa el término permutaciones (o variaciones) para referirse al número de diferentes ordenamientos. Si consideramos una colección de objetos sin importar el orden se habla de una combinación.

Dado un conjunto X, una permutación es una función biyectiva $f : X \to X$. Cuando el conjunto es finito, cada permutación corresponde a un reordenamiento de los elementos sin repetición de las "combinaciones primarias" sobre el reordenamiento.

Por ejemplo, si X = {a,b,c} entonces una función biyectiva de X en sí mismo podría ser

$$f(a) = c$$
$$f(b) = b$$
$$f(c) = a$$

la cual corresponde al reordenamiento de a b c dado por c b a.

Permutaciones de conjuntos finitos
Dado un conjunto de n elementos, el número de formas de disponerlos en forma ordenada es

$$n! = n(n-1)(n-2)\cdots 2\cdot 1$$

Dado que hay n formas de escoger el primer elemento, luego n-1 formas de escoger el segundo n-2 formas de escoger el tercero, y así sucesivamente. A cada arreglo ordenado de los elementos se le conoce como una permutación del conjunto.

Para el ejemplo del enunciado: $3! = 3\cdot 2\cdot 1 = 6$

Si del conjunto con n elementos deseamos escoger únicamente k de ellos para ordenarlos, el mismo argumento muestra que hay

$$P(n,k) = n(n-1)(n-2)\cdots(n-k+1)$$
$$P(n,k) = \frac{n!}{(n-k)!}$$

formas de realizar la tarea.

El número $P(n,k)$ se conoce como permutaciones de n en k. Se tienen las siguientes propiedades

$$P(n,n) = n!$$
$$P(n,1) = n \quad .$$

El número de combinaciones de k elementos en una colección de n elementos se simboliza por $C(n,k)$ y su valor puede calcularse a partir de $P(n,k)$ así:

$$C(n,k) = \frac{P(n,k)}{n!}$$

Permutaciones circulares
Son permutaciones donde no existe un primer y último lugar. Por ejemplo, n personas sentadas alrededor de una mesa circular, el número de permutaciones en las que pueden estar distribuidos. Se calcula fijando un elemento y permutando los demás, por tanto:

$$PC_n = P_{n-1} = (n-1)! \quad .$$

CAPÍTULO 4

Modelo Propuesto

Este capítulo es el más relevante del presente documento de tesis. Aquí se enuncia y demuestra el Teorema JV, se definen las dos nuevas operaciones binarias que permitirán la construcción de un campo finito basado en el Teorema JV, se enuncia y se demuestra el Teorema Factorial y además se propone un nuevo criptosistema: el Criptosistema Factorial, el cual permitirá reforzar algunos algoritmos conocidos de cifrado.

El capítulo consta de tres secciones. En la sección 4.1 se enuncia, se demuestra y se ejemplifica el sustento teórico de este trabajo de tesis: el Teorema JV; y además, se presentan detalladamente dos ejemplos, uno ilustrativo y otro exhaustivo.

La sección 4.2 contiene las definiciones de las dos operaciones binarias $\oplus$ y $|$, además de una operación unaria en el conjunto $\mathbf{\Pi}_m$; con base en estas definiciones y en el Teorema JV se construye un campo finito en el conjunto de las permutaciones, lo cual constituye la principal aportación de esta tesis.

En la sección 4.3 se propone, enuncia y demuestra el Teorema Factorial, el cual sirve de base teórica, junto con el Teorema JV, para proponer un criptosistema original del presente trabajo de tesis: el Criptosistema Factorial.

4.1 Teorema JV

Esta sección es especial, porque aquí se enuncia, se demuestra y se ejemplifica el sustento teórico de este trabajo de tesis: el Teorema JV.

4.1.1 Enunciado y demostración del Teorema JV

Sean dos números n, m no negativos tales que $0 \leq n \leq m!-1$; al usar el algoritmo de la división de Euclides [1], el número n se puede escribir de manera única de la siguiente forma:

$$n = C_0(m-1)! + C_1(m-2)! + C_2(m-3)! + \ldots + C_{m-2}\,1!$$

Nótese que $n \in \mathbf{N}_m = \{ n \in \mathbf{N} \mid 0 \leq n \leq m!\text{-}1\}$ y que $(m\text{-}1)!,(m\text{-}2)!,\ldots,1!$ se consideran fijos. Además, es conocido que se cumplen las siguientes desigualdades:

$$0 \leq C_i < (m\text{-}i), \text{ con } 0 \leq i \leq (m-2)$$

Una vez calculados los valores de $C_0, C_1,\ldots, C_{m\text{-}2}$ se puede construir el siguiente algoritmo:

Paso 0. Se define un arreglo creciente de la siguiente forma:

$$X[0] = 0, X[1] = 1, X[2]=2,..., X[m\text{-}1] = m\text{-}1$$

Paso 1. Al considerar que $C_0 < m$, es claro que $X[C_0]$ es alguno de los elementos del arreglo del paso 0. Se elimina a $X[C_0]$ del arreglo del paso 0 y se crea un nuevo arreglo con los valores restantes, desde X[0] hasta X[m-2].

Paso 2. Nuevamente se tiene que $C_1 < m\text{-}1$, y por ello $X[C_1]$ es alguno de los elementos del arreglo del paso 1. De la misma forma que en el paso anterior, se elimina a $X[C_1]$ del arreglo del paso 1 y, se reordena nuevamente a partir de X[0] hasta X[m-3].

.....
.....
.....

Paso *m*-1. Si se continúa trabajando de esta forma, al final se tendrá el siguiente arreglo: $X[C_{m\text{-}2}]$ y X[0].

Finalmente, el resultado de los números eliminados $X[C_0]$, $X[C_1]$, …, $X[C_{m\text{-}2}]$ y X[0] es una permutación del arreglo 0,1,2,…, m-1. Se puede afirmar que a cualquier $n \in \mathbf{N}_m$ se le puede asociar una permutación en m-1 pasos.

En este punto surge la pregunta: dados dos números diferentes del conjunto $\mathbf{N}_m$ ¿tendrán éstos asociadas dos permutaciones diferentes?. A esta última pregunta le da respuesta el Teorema JV, el cual se enuncia a continuación:

Teorema JV. Dados los dos conjuntos:

$\mathbf{N}_m = \{ n \in \mathbf{N} \mid 0 \leq n \leq m!\text{-}1\}$ y $\mathbf{\Pi}_m = \{ \pi \mid \pi$ es una permutación del arreglo 0,1,2,..,m-1$\}$,

el algoritmo descrito anteriormente define una función uno a uno y sobre que va del conjunto $\mathbf{N}_m$ al conjunto $\mathbf{\Pi}_m$; es decir, $I_m : \mathbf{N}_m \rightarrow \mathbf{\Pi}_m$ es biyectiva.

Demostración. La demostración se realizará por reducción al absurdo.
Suponga que para $n_1 \neq n_2$ con $n_1, n_2 \in \mathbf{N}_m \Rightarrow I_m(n_1) = I_m(n_2)$. Por el algoritmo de Euclides, se sabe que n_1, n_2 se pueden escribir como:

$$n_1 = C_{0,1}(m\text{-}1)! + C_{1,1}(m\text{-}2)! + C_{2,1}(m\text{-}3)! + \ldots + C_{m\text{-}2,1}\ 1!$$
$$n_2 = C_{0,2}(m\text{-}1)! + C_{1,2}(m\text{-}2)! + C_{2,2}(m\text{-}3)! + \ldots + C_{m\text{-}2,2}\ 1!$$

Ahora bien, si $I_m(n_1) = I_m(n_2)$ significa que: $C_{0,1} = C_{0,2}$, $C_{1,1} = C_{1,2}$, …, $C_{m\text{-}2,1} = C_{m\text{-}2,2}$. Por lo que $n_1 = n_2$; lo que contradice la hipótesis inicial. Entonces se concluye que si $n_1 \neq n_2$ con $n_1, n_2 \in N_m \Rightarrow I_m(n_1) \neq I_m(n_2)$. Lo anterior demuestra que la función I_m es uno a uno.
Dado que el número de elementos del conjunto $\mathbf{N}_m$ es igual al número de elementos del conjunto $\mathbf{\Pi}_m$, podemos afirmar que la función I_m es sobre.
Por lo tanto, la función $I_m : \mathbf{N}_m \rightarrow \mathbf{\Pi}_m$ es biyectiva.

4.1.2 Primer ejemplo

Trabajemos con cadenas de tamaño 8. Una permutación de las 8 posiciones es un arreglo particular de los números 0, 1, 2, 3, 4, 5, 6 y 7; por ejemplo 5,7,6,4,2,0,1 y 3. Ahora, sea un número entero no negativo tal que $0 \leq n \leq 8! - 1$; digamos n = 24637. Este número natural puede ser expresado como sigue:

$$24637 = 4(7!) + 6(6!) + 1(5!) + 1(4!) + 2(3!) + 0(2!) + 1(1!)$$

De hecho, cualquier entero n en el intervalo $0 \leq n \leq 8! - 1$ puede ser expresado de manera única, en términos de 7!,…1!, usando el algoritmo de Euclides. Nótese que la base aritmética que se usa al expresar n es 7!, 6!, 5!, 4!, 3!, 2! y 1!.

Denotemos los coeficientes de 7!, 6!, 5!, 4!, 3!, 2! y 1!, respectivamente, por C_0, C_1, C_2, C_3, C_4, C_5, C_6; en este ejemplo, los coeficientes tienen los valores $C_0 = 4$, $C_1 = 6$, $C_2 = 1$, $C_3 = 1$, $C_4 = 2$, $C_5 = 0$ y $C_6 = 1$. Los valores de C_i son los coeficientes que se obtienen de dividir n por 7!, …1!; además, por el algoritmo de Euclides, se cumplen todas las siguientes desigualdades [1]: $C_0 < 8$, $C_1 < 7$,…, $C_6 < 2$.

En este escenario se puede construir el siguiente algoritmo:

Paso 0. Se define el arreglo creciente:
X[0] = 0, X[1] = 1, X[2] = 2, X[3] = 3, X[4] = 4, X[5] = 5, X[6] = 6 y X[7] = 7

Paso 1. Se guarda el valor de $X[C_0 = 4] = 4$ y se elimina del arreglo definido en el paso 0. Se reordena el arreglo sin el valor de $X[C_0]$, y el resultado es:
X[0] = 0, X[1] = 1, X[2] = 2, X[3] = 3, X[4] = 5, X[5] = 6 y X[6] = 7

Paso 2. Se guarda el valor de $X[C_1 = 6] = 7$ y se elimina del arreglo definido en el paso 1. Se reordena el arreglo sin el valor de $X[C_1]$, y el resultado es:
X[0] = 0, X[1] = 1, X[2] = 2, X[3] = 3, X[4] = 5 y X[5] = 6

Paso 3. Se guarda el valor de $X[C_2 = 1] = 1$ y se elimina del arreglo definido en el paso 1. Se reordena el arreglo sin el valor de $X[C_2]$, y el resultado es:

$$X[0] = 0, X[1] = 2, X[2] = 3, X[3] = 5, \text{ y } X[4]=6$$

Paso 4. Se continúa de la misma forma y el valor $X[C_3 = 1] = 2$ se salva, mientras que el nuevo arreglo es:

$$X[0] = 0, X[1] = 3, X[2] = 5 \text{ y } X[3] = 6$$

Paso 5. Ahora se salva $X[C_4 = 2] = 5$ y el nuevo arreglo es:

$$X[0] = 0, X[1] = 3, X[2] = 6$$

Paso 6. Se salva $X[C_5 = 0] = 0$ y el arreglo queda así:

$$X[0] = 3, X[1] = 6$$

Paso 7. Finalmente, se salva $X[C_6 = 1] = 6$ y queda $X[0] = 3$.

Al escribir en orden $X[C_0]$, $X[C_1]$, $X[C_2]$, $X[C_3]$, $X[C_4]$, $X[C_5]$, $X[C_6]$ y $X[0]$ se obtiene, en 7 pasos, la permutación: 4,7,1,2,5,0,6 y 3. Esta es la permutación 24637 de la cadena 0,1,2,3,4,5,6 y 7.

4.1.3 Segundo ejemplo

Para *m*=4 resulta que el número de permutaciones que obtenemos de 4 elementos ordenados de uno en uno es:

$$P(4,1) = 4! = 4(3)(2)(1) = 24$$

Es decir, podemos escoger los números que van desde cero hasta veintitrés. Si tomamos el 4, tendremos:

$4 = 0(3!) + 2(2!) + 0(1!)$

Paso1:

$X[0] = 0$
$X[1] = 1$
$X[2] = 2$
$X[3] = 3$

Paso 2

$X[0] = 1$
$X[1] = 2$
$X[2] = 3$

Paso 3

$X[0] = 1$
$X[1] = 2$

Paso 4

X[0] = 2

Primero se eliminó el 0, despues el 3, luego el 1 y por último nos quedó el 2. Es decir, la permutación que le corresponde al 4 es 0 3 1 2.

A continuación se muestra cómo se obtienen las permutaciones que le corresponden a cada uno de los números desde el cero hasta 4! – 1 (23).

Permutaciones correspondientes para el cero:

0 = 0(3!) + 0(2!) + 0(1!)

Paso 1	Paso 2	Paso 3	Paso 4
0	1	2	3
1	2	3	
2	3		
3			
Permutación: 0 1 2 3			

1 = 0(3!) + 0(2!) + 1(1!)

Paso 1	Paso 2	Paso 3	Paso 4
0	1	2	2
1	2	3	
2	3		
3			
Permutación: 0 1 3 2			

2 = 0(3!) + 1(2!) + 0(1!)

Paso 1	Paso 2	Paso 3	Paso 4
0	1	1	3
1	2	3	
2	3		
3			
Permutación: 0 2 1 3			

3 = 0(3!) + 1(2!) + 1(1!)

Paso 1	Paso 2	Paso 3	Paso 4
0	1	1	1
1	2	3	
2	3		
3			
Permutación: 0 2 3 1			

4 = 0(3!) + 2(2!) + 0(1!)

Paso 1	Paso 2	Paso 3	Paso 4
0	1	1	2
1	2	2	
2	3		
3			
Permutación: 0 3 1 2			

$5 = 0(3!) + 2(2!) + 1(1!)$

Paso 1	Paso 2	Paso 3	Paso 4
0	1	1	1
1	2	2	
2	3		
3			
Permutación: 0 3 2 1			

$6 = 1(3!) + 0(2!) + 0(1!)$

Paso 1	Paso 2	Paso 3	Paso 4
0	0	2	3
1	2	3	
2	3		
3			
Permutación: 1 0 2 3			

$7 = 1(3!) + 0(2!) + 1(1!)$

Paso 1	Paso 2	Paso 3	Paso 4
0	0	2	2
1	2	3	
2	3		
3			
Permutación: 1 0 3 2			

$8 = 1(3!) + 1(2!) + 0(1!)$

Paso 1	Paso 2	Paso 3	Paso 4
0	0	0	3
1	2	3	
2	3		
3			
Permutación: 1 2 0 3			

$9 = 1(3!) + 1(2!) + 1(1!)$

Paso 1	Paso 2	Paso 3	Paso 4
0	0	0	0

1	2	3	
2	3		
3			
Permutación: 1 2 3 0			

$10 = 1(3!) + 2(2!) + 0(1!)$

Paso 1	Paso 2	Paso 3	Paso 4
0	0	0	2
1	2	2	
2	3		
3			
Permutación: 1 3 0 2			

$11 = 1(3!) + 2(2!) + 1(1!)$

Paso 1	Paso 2	Paso 3	Paso 4
0	0	0	0
1	2	2	
2	3		
3			
Permutación: 1 3 2 0			

$12 = 2(3!) + 0(2!) + 0(1!)$

Paso 1	Paso 2	Paso 3	Paso 4
0	0	1	3
1	1	3	
2	3		
3			
Permutación: 2 0 1 3			

$13 = 2(3!) + 0(2!) + 1(1!)$

Paso 1	Paso 2	Paso 3	Paso 4
0	0	1	1
1	1	3	
2	3		
3			
Permutación: 2 0 3 1			

$14 = 2(3!) + 1(2!) + 0(1!)$

Paso 1	Paso 2	Paso 3	Paso 4
0	0	0	3
1	1	3	
2	3		
3			
Permutación: 2 1 0 3			

15 = 2(3!) + 1(2!) + 1(1!)

Paso 1	Paso 2	Paso 3	Paso 4
0	0	0	0
1	1	3	
2	3		
3			
Permutación: 2 1 3 0			

16 = 2(3!) + 2(2!) + 0(1!)

Paso 1	Paso 2	Paso 3	Paso 4
0	0	0	1
1	1	1	
2	3		
3			
Permutación: 2 3 0 1			

17 = 2(3!) + 2(2!) + 1(1!)

Paso 1	Paso 2	Paso 3	Paso 4
0	0	0	0
1	1	1	
2	3		
3			
Permutación: 2 3 1 0			

18 = 3(3!) + 0(2!) + 0(1!)

Paso 1	Paso 2	Paso 3	Paso 4
0	0	1	2
1	1	2	
2	2		
3			
Permutación: 3 0 1 2			

19 = 3(3!) + 0(2!) + 1(1!)

Paso 1	Paso 2	Paso 3	Paso 4
0	0	1	1
1	1	2	
2	2		
3			
Permutación: 3 0 2 1			

20 = 3(3!) + 1(2!) + 0(1!)

Paso 1	Paso 2	Paso 3	Paso 4
0	0	0	2
1	1	2	
2	2		
3			
Permutación: 3 1 0 2			

$21 = 3(3!) + 1(2!) + 1(1!)$

Paso 1	Paso 2	Paso 3	Paso 4
0	0	0	0
1	1	2	
2	2		
3			
Permutación: 3 1 2 0			

$22 = 3(3!) + 2(2!) + 0(1!)$

Paso 1	Paso 2	Paso 3	Paso 4
0	0	0	1
1	1	1	
2	2		
3			
Permutación: 3 2 0 1			

$23 = 3(3!) + 2(2!) + 1(1!)$

Paso 1	Paso 2	Paso 3	Paso 4
0	0	0	0
1	1	1	
2	2		
3			
Permutación: 3 2 1 0			

Si representamos todas estas permutaciones en la siguiente tabla, podemos observar que, dados dos diferentes valores de n, las permutaciones correspondientes también son diferentes; además, dadas dos permutaciones diferentes, los valores de n que les corresponden también son diferentes.

Es decir, para cada n existe una permutación y es única, y para cada permutación existe una n y es única. Por lo tanto, la función es biyectiva, que es lo que nos dice el teorema JV.

n	Permutación			
0	0	1	2	3
1	0	1	3	2
2	0	2	1	3
3	0	2	3	1

4	0	3	1	2
5	0	3	2	1
6	1	0	2	3
7	1	0	3	2
8	1	2	0	3
9	1	2	3	0
10	1	3	0	2
11	1	3	2	0
12	2	0	1	3
13	2	0	3	1
14	2	1	0	3
15	2	1	3	0
16	2	3	0	1
17	2	3	1	0
18	3	0	1	2
19	3	0	2	1
20	3	1	0	2
21	3	1	2	0
22	3	2	0	1
23	3	2	1	0

4.2 Un campo finito en el conjunto de las permutaciones

En esta sección se construirá un campo en un conjunto de permutaciones, utilizando como base teórica el Teorema JV, mediante el cual es posible construir una biyección, I_m, que va del conjunto $\mathbf{N}_m = \{ n \in \mathbf{N} \mid 0 \leq n \leq m!\text{-}1\}$ al conjunto $\mathbf{\Pi}_m = \{ \pi \mid \pi$ es una permutación del arreglo $0,1,2,..,m\text{-}1\}$. Esta biyección define un isomorfismo $I_m : \mathbf{N}_m \rightarrow \mathbf{\Pi}_m$.

Como uno de los aportes principales de eta tesis, se definirán dos operaciones binarias $\oplus$ y $\mid$ en el conjunto $\mathbf{\Pi}_m$, de tal forma que con estas dos operaciones se puede construir un campo finito en un subconjunto de $\mathbf{\Pi}_m$. Se define, también, una operación unaria en el conjunto $\mathbf{\Pi}_m$; esto es, para todo elemento $\boldsymbol{\pi} \in \mathbf{\Pi}_m$ es posible definir su permutación inversa, la cual denotamos como $\neg\boldsymbol{\pi}$.

4.2.1 Definición de las operaciones binarias $\oplus$ y $\mid$

En el Capítulo 3 se estudió en detalle la función $I_m : \mathbf{N}_m \rightarrow \mathbf{\Pi}_m$, donde el dominio y el codominio se definen, respectivamente, cmo los conjuntos $\mathbf{N}_m = \{ n \in \mathbf{N} \mid 0 \leq n \leq m!\text{-}1\}$ y $\mathbf{\Pi}_m = \{ \pi \mid \pi$ es una permutación del arreglo $0,1,2,..,m\text{-}1\}$. En el Capítulo 3 también se demostró el Teorema JV, el cual afirma que esta función I_m: $\mathbf{N}_m \rightarrow \mathbf{\Pi}_m$ es biyectiva.

Ahora construyamos la función inversa $(I_m)^{-1}$, la cual tiene como entrada una permutación dada $I_m(n)$, y a la salida se obtiene el valor de n; a continuación se presenta el algoritmo que realiza dicha operación:

Paso 0.- Se inicia con el arreglo en orden; esto es, 0, 1, 2, …., m-1.
Paso 1.- Se toma el primer elemento de la permutación $I_m(n)$, digamos π_1, y se localiza la posición del valor π_1, contado a partir de 0 dentro del arreglo 0, 1, 2, …., m -1. Llamemos C_0 a esta posición y eliminemos del arreglo del paso 0 el valor π_1, quedándonos uno nuevo que denotaremos como $A(C_0)$. De hecho, C_0 es el coeficiente de $(m\text{-}1)!$; esto es, $C_0(m\text{-}1)!$.
Paso 2.- Se toma el segundo elemento de la permutación $I_m(n)$, digamos π_2, y se localiza la posición del valor de π_2, contado a partir de 0 dentro del arreglo $A(C_0)$, dicha posición la denominaremos C_1. Posteriormente, se elimina este valor del arreglo $A(C_0)$, quedándonos como arreglo resultante $A(C_0,C_1)$, el número C_1 es el coeficiente de $(m\text{-}2)!$ y los dos primeros sumandos serían: $C_0(m\text{-}1)! + C_1(m\text{-}2)!$.

Después de m-1 pasos, el valor de n que estamos buscando swe expresa así:

$$n = C_0(m\text{-}1)! + C_1(m\text{-}2)! + C_2(m\text{-}3)! + \ldots + C_{m-2}\,1!$$

En términos de las funciones I_m y $(I_m)^{-1}$, las operaciones $\oplus$ y $|$ se definen como sigue:

Definición.- La suma de 2 elementos $\boldsymbol{\pi}_1, \boldsymbol{\pi}_2 \in \boldsymbol{\Pi}_m$ se expresa como sigue:

$$\boldsymbol{\pi}_1 \oplus \boldsymbol{\pi}_2 = I_m\left(\left((I_m)^{-1}(\boldsymbol{\pi}_1) + (I_m)^{-1}(\boldsymbol{\pi}_2)\right) \bmod m!\right)$$

Definición.- El producto de 2 elementos $\boldsymbol{\pi}_1, \boldsymbol{\pi}_2 \in \boldsymbol{\Pi}_m$ se define como:

$$\boldsymbol{\pi}_1 \mid \boldsymbol{\pi}_2 = I_m\left(\left((I_m)^{-1}(\boldsymbol{\pi}_1) * (I_m)^{-1}(\boldsymbol{\pi}_2)\right) \bmod m!\right)$$

Utilizando las 2 operaciones definidas anteriormente, se verá que la función I_m define un isomorfismo [10], que va del conjunto $\mathbf{N}_m$ al conjunto de las permutaciones $\boldsymbol{\Pi}_m$.

Teorema.- La función $I_m : \mathbf{N}_m \rightarrow \boldsymbol{\Pi}_m$ define un isomorfismo con las operaciones $\oplus$ y $|$. Esto significa que se cumplen las siguientes propiedades, si $a, b \in \mathbf{N}_m$:

a) $I_m(a + b \bmod m!) = I_m(a) \oplus I_m(b)$

b) $I_m(a * b \bmod m!) = I_m(a) \mid I_m(b)$

Demostración.- Se iniciará con el inciso a, esto es, $I_m(a + b \bmod m!) = I_m(a) \oplus I_m(b)$.
$I_m(a) \oplus I_m(b) = \boldsymbol{\pi}_a \oplus \boldsymbol{\pi}_b$ donde $\boldsymbol{\pi}_a$, $\boldsymbol{\pi}_b$ son las permutaciones asociadas a los números a, b respectivamente. Sin embargo, por la definición de suma,$\oplus$, se tiene que $\boldsymbol{\pi}_a \oplus \boldsymbol{\pi}_b = I_m((I_m)^{-1}(\boldsymbol{\pi}_a) + (I_m)^{-1}(\boldsymbol{\pi}_b) \bmod m!)$. Se sigue que $I_m((I_m)^{-1}(\boldsymbol{\pi}_a) + (I_m)^{-1}(\boldsymbol{\pi}_b) \bmod m!) = I_m(a + b \bmod m!)$, con lo cual queda demostrado el primer inciso.
Para el caso del inciso b se seguirá una estrategia similar, esto es, se iniciará por el lado derecho de la expresión.
$I_m(a) \mid I_m(b) = \boldsymbol{\pi}_a \mid \boldsymbol{\pi}_b$. Sin embargo, por la definición de producto, $|$, se tiene que $\boldsymbol{\pi}_a \mid \boldsymbol{\pi}_b = I_m((I_m)^{-1}(\boldsymbol{\pi}_a) * (I_m)^{-1}(\boldsymbol{\pi}_b) \bmod m!)$. Por último, se concluye que $I_m((I_m)^{-1}(\boldsymbol{\pi}_a) * (I_m)^{-1}(\boldsymbol{\pi}_b) \bmod m!) = I_m(a * b \bmod m!)$. ■

4.2.2 Construcción del campo finito

Previo a la construcción del campo finito, mostremos que con las operaciones binarias $\oplus$ y $\mid$ se puede construir un anillo en el conjunto $\mathbf{\Pi}_m$.

Teorema.- A partir de las dos operaciones binarias $\oplus$ y $\mid$ definidas anteriormente, se puede construir un anillo en el conjunto $\mathbf{\Pi}_m$.

Demostración.- El teorema queda demostrado al considerar que el resultado conocido de que en el conjunto $\mathbf{N}_m = \{ n \in \mathbf{N} \mid 0 \leq n \leq m!\text{-}1\}$ se puede construir un anillo utilizando las propiedades de aritmética modular; y además, tomando en cuenta que I_m es un isomorfismo con imagen en el conjunto $\mathbf{\Pi}_m$.

Antes de abordar el problema del inverso multiplicativo, se debe mencionar que cualquier elemento diferente de 0 del conjunto $\mathbf{N}_m$, digamos $a \in \mathbf{N}_m$, tiene inverso multiplicativo módulo $m!$, si y sólo si el máximo común divisor de a y $m!$ es 1 [11].

En este sentido, si se elige un subconjunto $\boldsymbol{N}_m^p \subset \mathbf{N}_m$, de tal forma que todos los elementos diferentes de 0 del subconjunto $\boldsymbol{N}_m^p$ tengan inverso multiplicativo, se sigue que, se debe escoger un subconjunto de tamaño primo; esto es, $|\boldsymbol{N}_m^p| = p$. Este número primo p, cumple con la condición $1 \leq p \leq m!\text{-}1$.

De hecho, existen algunos casos particulares en que $m!$-1 es primo, como por ejemplo cuando m = 3 ó 4. Entonces procedamos de la siguiente manera: para un entero positivo m dado, se elige un número primo p tal que $1 \leq p \leq m!\text{-}1$. Posteriormente, utilizando la función I_m , se puede conocer la imagen del conjunto $\boldsymbol{N}_m^p = \{0, 1, \ldots p\text{-}1\}$, la cual es:

$$\mathbf{\Pi}_m^p = \{ I_m(0), I_m(1), \ldots., I_m(p\text{-}1)\}.$$

Las operaciones $\oplus$ y $\mid$ en $\mathbf{\Pi}_m^p$ se definen de la misma manera presentada previamente, excepto que en lugar de módulo $m!$ es módulo p.

Teorema.- En el conjunto $\mathbf{\Pi}_m^p$ se puede construir un campo con las operaciones $\oplus$ y $\mid$.

Demostración.- Ya se ha demostrado antes que el conjunto $\mathbf{\Pi}_m^p$ forma un anillo con las operaciones $\oplus$ y $\mid$.

Queda pendiente por demostrar que, para cualquier permutación $\boldsymbol{\pi} \in \mathbf{\Pi}_m^p$, y que además cumple con $\boldsymbol{\pi} \neq \mathbf{0} = I_p(0)$, existe $\boldsymbol{\pi}^{-1}$ tal que $\boldsymbol{\pi} \mid \boldsymbol{\pi}^{-1} = \boldsymbol{\pi}^{-1} \mid \boldsymbol{\pi} = \mathbf{1} = I_m(1)$. Si $n \neq 0$ es el entero positivo asociado a $\boldsymbol{\pi}$, se sigue que existe n^{-1} tal que $n * n^{-1} = n^{-1} * n \equiv 1 \bmod p$ [11]. Por lo que $\boldsymbol{\pi}^{-1}$ es la permutación asociada a n^{-1}, tal que $\boldsymbol{\pi} \mid \boldsymbol{\pi}^{-1} = \boldsymbol{\pi}^{-1} \mid \boldsymbol{\pi} = \mathbf{1}$.

Ahora abordaremos otro problema. Suponga a la permutación $\boldsymbol{\pi}$ como una función que va del conjunto $\{0,1,2,\ldots, m\text{-}1\}$ en sí mismo, entonces la permutación inversa, la cual denotamos como $\neg\boldsymbol{\pi}$, es aquella que cumple con $\neg\boldsymbol{\pi}$ o $\boldsymbol{\pi} = \mathbf{0}$, donde el símbolo "o" corresponde a la función composición.

Ahora bien, si consideramos a la operación unaria $\mathbf{U}(\boldsymbol{\pi}) = \neg\boldsymbol{\pi}$, ésta no necesariamente es cerrada en el conjunto $\boldsymbol{\Pi}_m^p$; esto es, dado $\boldsymbol{\pi} \in \boldsymbol{\Pi}_m^p$, no necesariamente $\neg\boldsymbol{\pi} \in \boldsymbol{\Pi}_m^p$. Sin embargo, la operación $\mathbf{U}(\boldsymbol{\pi})$ si es cerrada en $\boldsymbol{\Pi}_m$ debido a que este último conjunto contiene todas las permutaciones del arreglo 0, 1, 2, …., m-1. La propiedad anterior es importante, ya que los criptosistemas simétricos como DES o Triple-DES inician con una permutación fija $\boldsymbol{\pi}$ [4, 3], y al final del ciclo de cifrado se aplica la permutación $\neg\boldsymbol{\pi}$.

4.3 Un criptosistema basado en el Teorema Factorial

Otra aportación relevante de esta tesis es el Teorema Factorial, el cual se enuncia y demuestra en esta sección; específicamente, se demuestra que para cualquier permutación π_L definida en el conjunto de los números $\{0,1, \ldots, L\text{-}1\}$, con L múltiplo de 3, ésta puede ser construida a partir de 3 permutaciones definidas en el conjunto $\{0,1,\ldots,\frac{2}{3}L\text{-}1\}$.

Lo anterior permite definir un criptosistema de bloques de cadenas de 96 bits de longitud, en el cual se trabaja con números de $64! - 1 \approx 10^{90}$ en lugar de $96! - 1 \approx 10^{150}$ con lo que se reduce el tiempo y recursos de cómputo. El nuevo criptosistema se ilustra con las cajas de AES (Advanced Encryption Standard) y un procedimiento de cifrado por bloques de 96 bits de texto claro; se usan las cajas de AES debido a su alta no linealidad.

Al igual que AES este es un criptosistema del tipo Sustitution Permutation Network, con la diferencia de que es posible cifrar no sólo cadenas de 128 bits como lo hace AES, sino que haciendo uso del teorema factorial es posible cifrar cadenas que sean múltiplos de 2 y 3, como es el caso de 192 bits.

Además, se muestra que el conjunto de las llaves crece de manera factorial, de tal forma que el número de elementos de este conjunto llega a ser del orden de $10^{150} \approx 2^{500}$ cuando se trabaja con cadenas de 96 bits.

4.3.1 Teorema Factorial

Teorema Factorial.- Dada una permutación π_L en las posiciones de una cadena de longitud L siendo L un entero múltiplo de 3, entonces π_L se puede construir mediante 3 permutaciones en cadenas de longitud $\frac{2}{3}L$.

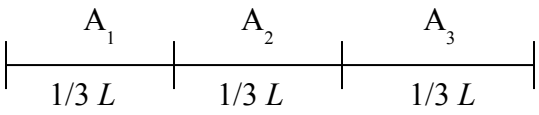
Demostración.- Sea una permutación de las posiciones de una cadena de longitud L así:

$$\pi_L = \sigma(0) = j_0, \sigma(1) = j_1, \ldots \sigma(L\text{-}1) = j_{L-1}$$

Se parte el conjunto de posiciones en dos conjuntos de la siguiente manera:

$$A=\{0,1,\ldots, \tfrac{2}{3}L\text{-}1\} \text{ y } B= \{\tfrac{2}{3}L, \tfrac{2}{3}L+1, \ldots, L\text{-}1\}$$

Se divide la permutación inicial en tres bloques como se ilustra a continuación:



La primera permutación $\pi_1(y)$ con $0 \leq y \leq \frac{2}{3}L\text{-}1$ se obtiene así:

1. Se asignan las posiciones que son elementos del conjunto A a los bloques A_1, A_2.
2. Las posiciones del conjunto B que estén en los bloques A_1, A_2, en caso de que existan, se asignan al azar usando los restantes elementos de A.

A fin de aplicar la permutación π_2, se usa la función de desplazamiento $g_1(y) = y\text{-}\frac{1}{3}L$ con la condición de que $\frac{1}{3}L \leq y \leq L\text{-}1$.

Permutación $\pi_2(g_1(y))$:

1. Si quedan posiciones en su lugar, no se modifican.
2. Si las hay, se asignan las posiciones de los bloques A_2 y A_3 que son elementos del conjunto B. También, si las hay, se asignan las posiciones de la forma $\pi_1(y)$ con la condición de que $\frac{1}{3}L \leq y \leq \frac{2}{3}L\text{-}1$ las cuales deben estar en el bloque A_3. Las posiciones de la forma $\pi_1(y)$ que cumplen la condición de que $0 \leq y \leq \frac{1}{3}L\text{-}1$ que deben estar en el bloque A_3 no se sustituyen en esta etapa; las posiciones restantes se asignana al azar. En este punto, las posiciones del bloque A_2 están en su lugar correcto.

Para aplicar la permutación π_3, se usa la función de desplazamiento $g_2(y)$:

$$g_2(y)= \begin{cases} \text{if} \leq y \leq \text{-}1 \\ \text{if } 0 \leq y \leq \text{-}1 \end{cases}$$

La permutación $\pi_3(g_2(y))$ se obtiene de la siguiente manera:

1. Si quedan posiciones en su lugar, no se modifican.
2. Si existen, se asignan las posiciones que son del bloque A_1 las cuales son elementos del conjunto B. También, si existen, se asignan las posiciones de las forma $\pi_1(y)$ con la condición de que $0 \leq y \leq \frac{1}{3}L\text{-}1$ que deben estar en el bloque A_3.

Con lo anterior queda establecido que al aplicar las tres permutaciones, se ha construido la permutación inicial

$$\pi_L = \sigma(0) = j_0,\ \sigma(1) = j_1, \ldots\ \sigma\,(L\text{-}1) = j_{L\text{-}1}$$

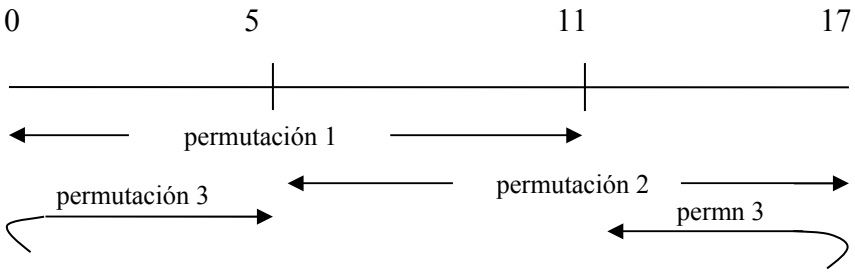
Ejemplo.-

Ejemplificaremos el Teorema Factorial con cadenas de tamaño 18, una de cuyas permutaciones es:

9 0 12 2 3 13 1 14 4 5 15 16 6 8 7 17 11 10

Aquí cabe una pregunta cuya respuesta es relevante: ¿Es posible aplicar permutaciones de tamaño menor que 18 de modo que resulte la permutación previa?

Afortunadamente la respuesta es afirmativa. Gráficamente, el procedimiento es así:



Ahora veamos cómo es posible expresar cualquier permutación de 18 posiciones, mediante la aplicación de las 3 permutaciones ilustradas en la figura previa. De hecho, el procedimiento es útil para cualquier longitud L que sea múltiplo de 3.

Se inicia con el arreglo ordenado 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15, 16,17 y se parte en 2 conjuntos, digamos: **A** = {0,1,2,3,4,5,6,7,8,9,10,11} y **B** = {12,13,14,15,16,17} de 12 y 6 posiciones, respectivamente.

Además, la permutación inicial 9 0 12 2 3 13 1 14 4 5 15 16 6 8 7 17 11 10 se parte en tres bloques de longitud igual, como se ilustra en la siguiente figura:

A_1	A_2	A_3
9 0 12 2 3 13	1 14 4 5 15 16	6 8 7 17 11 10

La primera permutación asigna las posiciones del conjunto A a los bloques A_1 y A_2; dejando fuera las posiciones que corresponden al conjunto B:

9 0 - 2 3 - 1 - 4 5 - -

Al comparar esta expresión con los elementos del conjunto A, se observa que están ausentes los números 6, 7, 8, 10 y 11; asignemos estos números a los hoyos, de manera aleatoria; por ejemplo: 10, 6, 7, 8 y 11. El resultado de aplicar la primera permutación es:

9 0 10 2 3 6 1 7 4 5 8 11 12 13 14 15 16 17

												$g_2(y)=0$	1	2	3	4	5
9	0	10	2	3	6	1	14	4	5	15	16	13	8	7	17	11	12
$g_2(y)=6$	7	8	9	10	11												

Se sigue que la primera permutación aplicada fue: $\pi_1(y)$ = 9 0 10 2 3 6 1 7 4 5 8 11 con la

						$g_1(y)=0$	1	2	3	4	5	6	7	8	9	10	11
9	0	10	2	3	6	1	7	4	5	8	11	12	13	14	15	16	17

condición $0 \leq y \leq 11$. La segunda permutación se aplica a las posiciones 6 a 17. Sin embargo, se debe de aplicar el siguiente desplazamiento para poder aplicar esta segunda permutación: $g_1(y) = 6 - y$ con $6 \leq y \leq 17$; esto se ilustra gráficamente abajo:

La permutación $\pi_2(g_1(y))$ se construye como sigue:

Las posiciones que están en su lugar no se modifican, por lo que $g_1(y)$ = 0, 2, 3, corresponden a los números 1,4 y 5.

Ahora se asignan las posiciones de los bloques A_2 y A_3 que son elementos del conjunto B, como es el caso de los números 14, 15, 16 y 17, los cuales son asignados a las posiciones $g_1(y)$ = 1, 4, 5 y 9.

Adicionalmente, se asignan los números de la forma $\pi_1(y)$ con $6 \leq y \leq 11$ que deben de estar en A_3, como es el caso de los números 7, 8, y 11,los cuales se escriben en las posiciones $g_1(y)$ = 8, 7 y 10. Los números de la forma $\pi_1(y)$ con $0 \leq y \leq 5$ deben estar en el bloque A_3, no se pueden asignar en esta segunda etapa. Las posiciones restantes son: $g_1(y)$ = 6, 7 las cuales corresponden a los números 12 y 13 y serán asignados al azar. Supongamos que se escribe primero el 13 seguido por el 12; entonces, el resultado es:

9 0 10 2 3 6 1 14 4 5 15 16 13 8 7 17 11 12

Para poder aplicar la tercera permutación, se requiere un desplazamiento $g_2(y)$, el cual se define de la siguiente manera:

$$g_2(y)= \begin{cases} y\text{-}12 \quad \text{para} \quad 12 \leq y \leq 17 \\ y\text{+}6 \quad \text{para} \quad 0 \leq y \leq 5 \end{cases}$$

La permutación $\pi_3(g_2(y))$ $g_2(y)$ se aplica así:

Las posiciones que están en su lugar no se modifican $g_2(y)$ = 1, 2, 3, 4, 6, 7, 9 y 10.

Se asignan los números que son elementos del conjunto B en el bloque A_1 ; así, el 12 y 13 quedan en las posiciones $g_2(y) = 8$ y $g_2(y) = 11$. También, se asignan las posiciones de la forma $\pi_1(y)$ with $0 \leq y \leq 5$ que deben estar en A_3; esto pone a 10, 6 en las posiciones definidas por $g_2(y) = 5$ y $g_2(y) = 0$. Se sigue que la tercera permutación es: $\pi_3(g_2(y))$= 11 1 2 3 4 8 6 7 5 9 10 0. Finalmente, el resultado es:

9 0 12 2 3 13 1 14 4 5 15 16 6 8 7 17 11 10

Al aplicar este procedimiento, se trabaja con números del orden de 10^{90} en lugar de 10^{150}, aproximadamente, cuando las cadenas son de tamaño 96. En general, se reduce la cantidad de cálculos.

Nota importante: Con el Teorema Factorial se demuestra que cualquier permutación sobre las posiciones de una cadena de longitud *L*, donde L es múltiplo de 3, puede construirse a partir de 3 cadenas de longitud $\frac{2}{3}L$. En este punto puede surgir la siguiente pregunta: ¿deben de ser necesariamente 3 o existe la posibilidad de que otro número de cadenas diferente de 3 cumpla con el requisito de construir cualquier permutación sobre las posiciones de unca cadena de longitud *L*?
La respuesta a esta pregunta parece ser que es NO, aunque aún no existe una demostración de esto. Se ha observado para algunos casos particulares, como por ejemplo utilizar 4 o 5 cadenas de longitudes menores que *L*, y para valores particulares de L, que siempre hay permutaciones sobre las posiciones de la cadena de longitud *L* que no pueden ser construidas a partir de estas cadenas y de esos valores.

4.3.2 Criptosistema Factorial

El criptosistema que se propone aquí es de naturaleza iterativa, y a pesar de que el tiempo de ejecución en software es similar en orden de magnitud a de DES, es mucho más resistente a los ataques de fuerza bruta. A continuación se describe el nuevo criptosistema:

1. Se inicia con una cadena *TC* de texto claro de 12 bytes, es decir, de 96 bits. Se escogen tres enteros positivos n_1, n_2 y n_3, que cumplan con la siguiente propiedad: $0 \leq n_i \leq 64!-1$ for $i = 1, 2, 3$.

2. Con base en el Teorema JV, es posible asociar estos tres enteros con tres permutaciones sobre cadenas de 64 posiciones. Entonces, con base en el Teorema Factorial, es posible contruir cualquier permutación de las posiciones de la cadena de texto claro de 96 bits, mediante el uso de los enteros n_i, con $i = 1, 2, 3$, permutación que llamaremos π_{96}. La aplicación de la permutación π_{96} al texto claro será designada como $\pi_{96}(TC)$.

3. En virtud de que la cadena $\pi_{96}(TC)$ tiene una longitud de 96 bits, se puede partir en dos porciones iguales, una cadena izquierda de 48 bits de longitud, y una cadena derecha de

48 bits de longitud in length, las cuales denotaremos por L^*_0 y R^*_0, respectivamente. Se ejecutan 16 rondas, de las cuales 15 siguen el procedimiento iterativo bosquejado aquí:

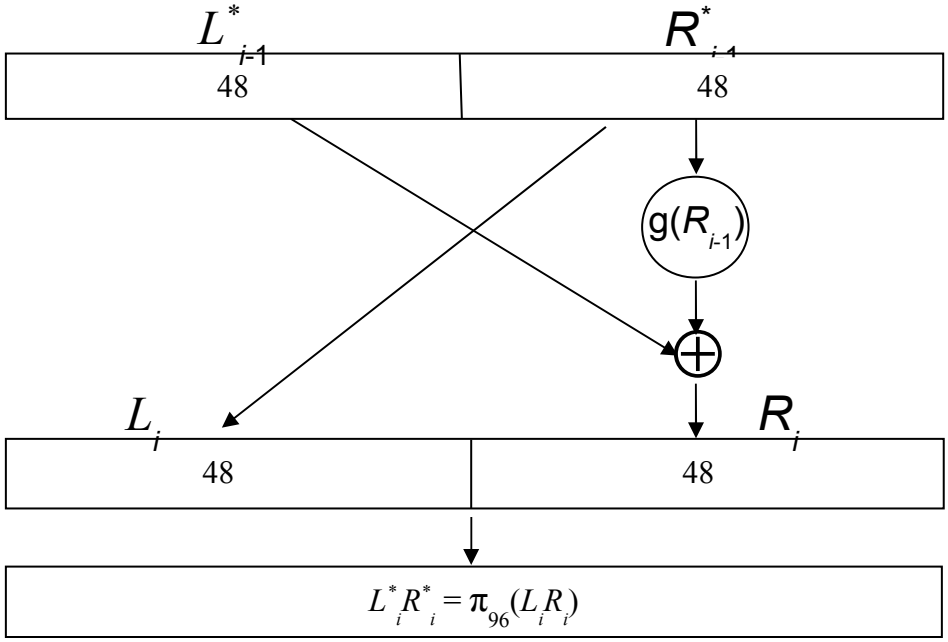
Las cadenas ${}_iR_i$ y ${}^*L_i{}^*R_i$ se obtienen como sigue:

Para $i = 1$ a 15 hacer

$L_i = R^*_{i-1}$; $R_i = L^*_{i-1} \oplus g(R^*_{i-1})$; ${}^*L_i{}^*R_i = \pi_{96}(L_iR_i)$

Nótese que la permutación π_{96} se aplica 16 veces.. La función g se ejecuta como sigue:

La cadena derecha R^*_{i-1} tiene 48 bits y puede ser partida en 6 bloques de 8 bits cada uno. Estos bloques constituyen las entradas a las cajas de AES, cuya salida es entonces una substitución. Por ejemplo, supongamos que la entrada es el bloque 01110011. Esta cadena se divide en dos partes, digamos 0111 y 0011. La primera parte direcciona el renglón de la caja, y la segunda parte direcciona la columna de la caja, que corresponde a 10001111; es decir, el bloque 01110011 se intercambia con el bloque 10001111. Si se sigue este procedimiento para cada uno de los bloques de R^*_{i-1}, resulta una cadena de 48 bits que será designada como $g(R^*_{i-1})$. El procedimiento se ilustra así:



Durante al ronda décimosexta de aplicación de la permutación $(\pi_{96})^{-1}(R_{16}\ L_{16})$ se obtiene el texto cifrado. Nótese que $(\pi_{96})^{-1}$ es la inversa de la permutación de π_{96} y que los bloques R_{16} y L_{16} aparecen en orden invertido. Algunas notas adicionales:

1 Los enteros n_1, n_2 y n_3 actúan como una clave, porque la permutación π_{96} puede modificarse al cambiar uno o varios de los números n_1, n_2 y n_3.

2 Al considerar que cada permutación es una clave, se ve claramente que la cantidad de posibles claves es aproximadamente 10^{150}.

3 Este criptosistema tiene la propiedad *whitening* [10].

4 Es importante mostrar que el ciclo de encriptación e : $\mathbf{Z}_2^{96} \rightarrow \mathbf{Z}_2^{96}$ es una función uno a uno.

Teorema.- Dada la permutación π_{96} definida por los teoremas JV y Factorial, entonces el algoritmo de encriptación e : $\mathbf{Z}_2^{96} \rightarrow \mathbf{Z}_2^{96}$ descrito arriba define una función uno a uno.

Demostración.- Por contradicción. Debemos mostrar que para cualesquiera dos textos claros diferentes, TC_1, TC_2 ε $\mathbf{Z}_2^{96}$ => e(TC_1) ≠ e(TC_2). Ahora, asumamos que existen al menos dos textos claros diferentes TC^{*1}, TC^{*2} ε $\mathbf{Z}_2^{96}$ tales que e(TC^{*1}) = e(TC^{*2}). Sean L_8^{*1} y R_8^{*1}, respectivamente, los bloques izquierdo y derecho de la octava ronda correspondiente al texto claro TC^{*1}, y, de manera similar, sean L_8^{*2} y R_8^{*2} los bloques izquierdo y derecho de la octava ronda correspondiente al texto claro TC^{*2}. Entonces, si se cumple la igualdad e(TC^{*1}) = e(TC^{*2}) => $L_8^{*1} = L_8^{*2}$ y $R_8^{*1} = R_8^{*2}$. Sin embargo, si esto es verdad, entonces debe suceder que $L_7^{*1} = L_7^{*2}$ y $R_7^{*1} = R_7^{*2}$, donde L_7^{*1}, L_7^{*2} y R_7^{*1}, R_7^{*2} son definidas similarmente como L_8^{*1}, L_8^{*2} y R_8^{*1}, R_8^{*2}. Si se continúa con este proceso, se concluye que $TC^{*1} = TC^{*2}$. Esto es una contradicción con el hecho de que $TC^{*1} \neq TC^{*2}$, y de aquí se sigue que para cualesquiera dos textos claros diferentes TC_1, TC_2 ε $\mathbf{Z}_2^{96}$ => e(TC_1) ≠ e(TC_2).

A continuación se demuestra el siguiente resultado importante: los algoritmos DES o Triple-DES definen funciones uno a uno.

Sin pérdida de generalidad se puede suponer que el texto claro $TCL = L_0R_0$ y el texto cifrado es $TCI = L_{16}R_{16}$. Por otro lado, si denotamos a un ciclo de encriptación DES como $e_k(TCL) = TCI$ donde k es una llave dada, entonces se enuncia el siguiente teorema:

Teorema.- Dados 2 textos claros diferentes $L_0R_0 \neq L_0^*R_0^*$, entonces se cumple que $e_k(L_0R_0) \neq e_k(L_0^*R_0^*)$ para una llave k dada.

Demostración.- La demostración se realizará por el método de reducción al absurdo. En este orden de ideas, supongamos que $e_k(L_0R_0) = e_k(L_0^*R_0^*)$. Pero si esto es así, entonces se cumple que $L_{16} = L_{16}^*$ y $R_{16} = R_{16}^*$.

Pero $L_{16} = L_{16}^* \Rightarrow R_{15} = R_{15}^*$ y por lo tanto se cumple que $f(R_{15}, k_{16}) = f(R_{15}^*, k_{16})$, lo cual implica que $L_{15} = L_{15}^*$ ya que $L_{15} = f(R_{15}, k_{16}) \oplus R_{16}$ y $L_{15}^* = f(R_{15}^*, k_{16}) \oplus R_{16}^*$.

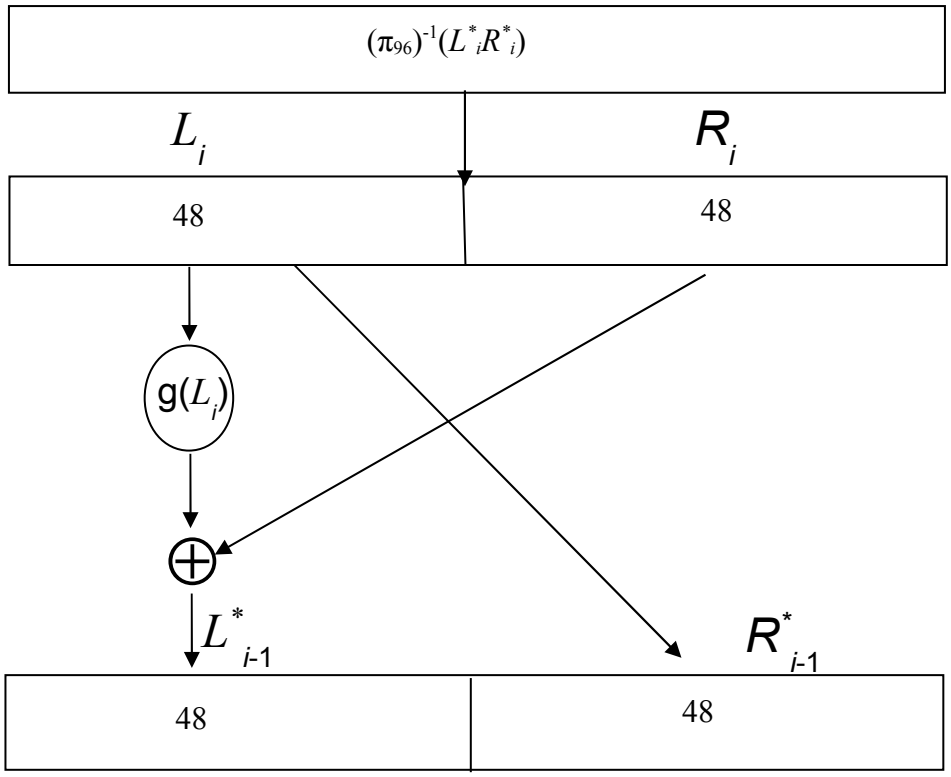
Pero si $L_{15} = L^*_{15}$ se sigue que $R_{14} = R^*_{14}$ y por lo tanto $L_{14} = L^*_{14}$ utilizando el mismo argumento de arriba. Si continuamos con este mismo procedimiento se llega a la conclusión de que $L_0 = L^*_0$ y $R_0 = R^*_0$.

Sin embargo, la conclusión anterior contradice la hipótesis de que $L_0R_0 \neq L^*_0R^*_0$.

Por lo tanto, se concluye que si $L_0R_0 \neq L^*_0R^*_0$, entonces se cumple que $e_k(L_0R_0) \neq e_k(L^*_0R^*_0)$ para una llave *k* dada.■

Después de la demostración del teorema anterior, no es complicado percatarse que el criptosistema Triple-DES también define una función uno a uno.

El proceso de descifrado se ilustra a continuación:



CAPÍTULO 5

Resultados Experimentales y Discusión

Este capítulo es de vital importancia en el presente trabajo de tesis, puesto que permite ilustrar, experimentalmente y a través de ejemplos, los resultados originales generados.

El capítulo consta de cuatro secciones. En la sección 5.1 se presenta una aplicación a la criptografía de los campos finitos JV, en la que se analiza la forma en que se fortalecen los criptosistemas simétricos DES y Triple-DES al aplicar los resultados de esta tesis.

Las secciones 5.2 y 5.3 se dedican a ejemplificar, por un lado, el Criptosistema Factorial, y por otro, el uso del Teorema Factorial en el cifrado de cadenas de 192 bits. Finalmente, en la sección 5.4 se presenta una interesante aplicación de los resultados originales de este trabajo de tesis: el envío de cheques electrónicos cifrados mediante el algoritmo TripleDES con permutación variable.

5.1 Fortalecimiento de los criptosistemas DES y Triple DES

En esta sección se presenta una aplicación a la criptografía de los campos finitos JV, los cuales fueron creados en el capítulo 4 y constituyen la principal aportación de este trabajo de tesis. Concretamente, se analiza cómo se fortalecen los criptosistemas simétricos DES y Triple-DES [10], tanto para ataques de fuerza bruta, como para los criptoanálisis diferencial y lineal en el caso DES [1, 8].

Como se sabe, los criptosistemas DES y Triple-DES se inician con una permutación fija [2, 13]; de hecho, podemos afirmar que en todos los criptosistemas del tipo "Substitution Permutation Network" interviene una permutación la cual se considera fija. Hasta el momento, en la literatura no aparece alguna propuesta de permutaciones variables.

Ahora bien, cuando se aplica una permutación fija se realiza mediante una tabla. Para los criptosistemas DES y triple-DES se aplica una permutación IP al inicio del proceso de cifrado y al final se usa la permutación inversa $\neg$(IP). La permutación IP y la descripción del algoritmo para DES se presenta en **FIPS PUB 46-3**, cabe aclarar que esta misma permutación se utiliza para el caso de Triple-DES.

Al utilizar el algoritmo descrito en la sección 4.1, se puede comprobar que el número asociado a la permutación IP de la norma internacional es:

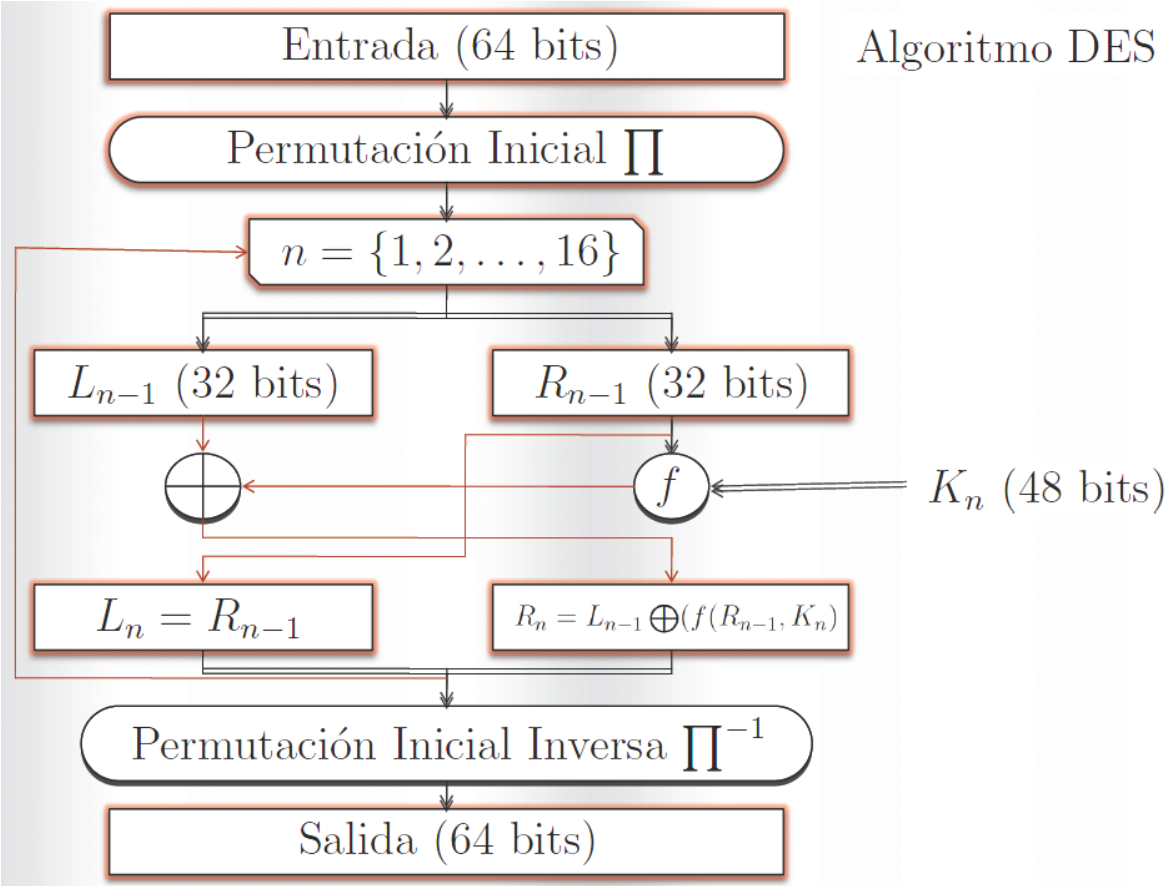
n_{IP}=
1145717915565593966585098047364889452612375674516926436332284482028493619
88738097376113279

Por otro lado, es importante señalar que un ataque de fuerza bruta no sería posible para el criptosistema DES con permutación variable, pues no sería suficiente conocer un bloque de texto claro y su correspondiente texto cifrado y probar las 2^{56} llaves, proceso que en este momento se realiza por completo, ya que no se conocería la permutación inicial [9].

En estricto, lo anterior obligaría al ataque de fuerza bruta probar para cada permutación las 2^{56} llaves binarias posibles y dar con el arreglo correcto de salida; es claro que lo anterior imposibilitaría, computacionalmente hablando, este tipo de criptoanálisis.

En este sentido, una de las propuestas de este trabajo es que la permutación inicial IP que se aplica al inicio de Triple-DES o DES, sea variable; es decir, que no se use la permutación fija de las cajas de DES, sino que cada vez que se inicie un proceso de cifrado, la permutación sea diferente. Queda claro que la permutación inversa, IP^{-1}, se puede obtener de manera sencilla a partir de IP.

El esquema de bloques siguiente ilustra este fortalecimiento que representa para el algoritmop DES el uso de una permutación variable inicial IP, con su correspondiente permutación final inversa IP^{-1}.



Otra cuestión relevante de destacar, es que al iniciar con una permutación variable se tendría una propiedad similar a la de "whitening" [10], que los algoritmos de encriptación más recientes, como AES **FIPS PUB 46-3**, lo tienen.

Para ilustrar lo anteriormente mencionado, se presenta como ejemplo una *n* diferente a n_{IP}, por ejemplo:
n=
9897671413654676849465468798798797777777777978946546546541321231564654987
9486541351590214

La permutación que corresponde al número anterior se presenta a continuación:

IP							
49	59	7	12	28	16	43	34
45	41	36	10	17	27	4	25
23	22	63	55	57	44	46	40
6	24	5	42	37	20	47	11
19	50	53	21	8	2	33	13
39	29	56	61	54	15	9	18
31	1	35	60	30	58	26	51
32	62	3	48	52	38	0	14

Ahora presentaremos la argumentación de cómo se incrementa la complejidad de los criptosistemas DES y Triple-DES cuando se propone una permutación variable.

Debido a que los criptosistemas DES y Triple-DES inician su proceso de encriptación con cadenas de 64 bits de longitud, se sigue que hay 2^{64} posibilidades de entrada y 2^{64} posibilidades de salida. Sin embargo, si observamos a las 2^{64} posibilidades de salida como un arreglo, entonces hay $(2^{64})!$ arreglos diferentes de salida, además, no se debe perder de vista que, como se demuestra en el Apéndice B, los algoritmos DES y Triple-DES definen una función uno a uno, por lo que se puede afirmar que las $64! \cong 2^{300}$ posibles permutaciones, definen 2^{300} arreglos diferentes de salida.

Lo anterior se demuestra de la siguiente manera. Suponga que las llaves de 64 bits: *k* para DES y k_1, k_2 para Triple-DES permanecen fijas [9]. Además, se dan 2 permutaciones $\boldsymbol{\pi}_1, \boldsymbol{\pi}_2 \in \boldsymbol{\Pi}_{64}$ con $\boldsymbol{\pi}_1 \neq \boldsymbol{\pi}_2$, entonces existe al menos una cadena de 64 bits "tc", tal que $\boldsymbol{\pi}_1(tc) \neq \boldsymbol{\pi}_2(tc)$. Sin embargo, lo anterior significa que las permutaciones $\boldsymbol{\pi}_1, \boldsymbol{\pi}_2$ usadas en un proceso DES o Triple-DES construyen 2 arreglos diferentes de salida, ya que al menos tienen una posición diferente.

Obsérvese que $2^{300} << (2^{64})!$, de tal forma que podemos decir que todas las permutaciones actúan como llaves diferentes, ya que darán como resultado arreglos de salida diferentes.

En este sentido, se puede decir que hay al menos 2^{300} llaves posibles, lo cual es muy superior a 2^{56} para DES o 2^{112} para Triple-DES.

5.2 Un ejemplo de aplicación del Criptosistema Factorial

En esta sección se describe, a través de un ejemplo, la aplicación del Criptosistema Factorial. Se parte del siguiente texto claro: **MiguelLindig**, y se asignan los siguientes valores a los enteros n_1, n_2 y n_3:

n_1=899154456457901212645790121264788888888888999995645790121899117999 9

n_2= 99999999999999998888888888887777777774641064598798888888888979797 9789877879

n_3= 14136546768494654687987987988888888889789465465465413212315646549 87948654135159

Estos números satisfacen la condición de que $2 \leq n_i \leq 64!-1$. Las permutaciones asociadas a estos números son:

π_1=0,1,2,3,4,5,6,7,8,9,10,11,17,53,44,57,48,41,16,51,32,34,35,38,19,26,37,52,39,58, 63,21,13,15,28,29,55,27,42,20,56,45,25,43,22,18,14,23,60,61,40,36,54,12,30,47, 31,33,50,62,59,49,46,24.

π_2=0,1,2,3,4,5,6,7,15,57,19,33,60,30,43,45,21,29,11,9,63,12,50,18,58,23,42,17,13, 44,27,26,46,25,55,62,59,37,20,22,10, 31,40,34,49,14,53,54,39,8,61,38,24,35,48,41, 52,32, 47,51,28,16,56,36.

π_3=0,1,2,3,4,5,40,57,20,59,53,7,14,18,27,50,34,13,28,54,51,44,24,49,19,23,36,52,8, 15,55,47,41,43,31,6,62,45,61,26,33, 30,60,58,42,21,38,22,35,11,25,48,29,63,37,56, 39,16,12,32,17,46,16.

La permutación π_{96} del algoritmo da como resultado:

π_{96}=9,11,68,54,63,53,58,79,1,88,39,37,10,67,6,80,3,66,64,48,60,21,24,5,19,7,71,81, 0,56,44,72,13,15,28,29,55,27,42,20, 23,89,36,65,92,46,75,77,12,49,43,45,95,22,82, 40,90,47,74,61,18,76,62,50,78,33,87,94,91,69,8,26,31,52,34,30,85,93,83, 16,2,14, 59,35,51,17,84,41,70,73,4,32,25,86,38,57.

Y la inversa de la permutación π_{96} se muestra abajo:

π_{96}^{-1} =28,8,80,16,90,23,14,25,70,0,12,1,48,32,81,33,79,85,60,24,39,21,53,40,22,92, 71,37,34,35,75,72,91,65,74,83,42,11, 94,10,55,87,38,50,30,51,45,57,19,49,63,84, 73,5,3,36,29,95,6,82,20,59,62,4,18,43,17,13,2,69,88,26,31,89,58,46,61,47,64,7,15, 27,54,78,86,76,93,66,9,41,56,68,44,77,67,52.

Finalmente, el resultado del proceso de cifrado, en formato hexadecimal, es como sigue:
523E903E05A1A6C492E991D9.

5.3 Uso del Teorema Factorial para cifrar cadenas de 192 bits

En esta sección se describe el proceso para cifrar cadenas de 192 bits, el cual es del tipo ”Substitution Permutation Network”-SPN [22]. Como cualquier algoritmo de este tipo consta de 2 partes, a saber: la primera de ellas define el proceso de mezclado de la información, en el cual intervienen 3 tipos diferentes de operaciones, a saber: la operación XOR [16], la operación de substitución [23] y la de permutación [4].

En la segunda parte se define el algoritmo que genera un programa de llaves, las cuales intervienen en cada una de las rondas. En este trabajo se considerarán 16 rondas [23]. Lo anterior, con la finalidad de evitar los ataques que se hacen a AES con pocas rondas [22]; además, como el algoritmo que se mostrará tiene la característica de ”whitening” [22] no son posibles los ataques diferencial y lineal [8, 3], tampoco puede realizarse un ataque de fuerza bruta debido a la complejidad computacional para resolverlo [25].

Aquí se presentará el algoritmo de mezclado de la información, para lo cual describiremos una función ronda **f** característica y la función ronda final **g**, la cual es ligeramente diferente a **f**.

La función **f** tiene 2 argumentos y se representa como $\mathbf{f}(w^{i-1},K^i)$ para $1 \leq i \leq 15$. El primer argumento es una cadena de 192 bits y que para $i = 1$; esto es w^0, es un bloque del texto claro.

El segundo argumento es una llave del programa de llaves que interviene en la ronda i; también es una cadena de 192 bits y más adelante se hará una descripción del algoritmo que genera las llaves. Ahora bien, con esta información la función **f** procede de la siguiente manera:

i) Se inicia con la aplicación de la operación x-or a las cadenas w^{i-1} y K^i, esto es, $w^{i-1} \oplus K^i = u^i$.

ii) Antes de mencionar el procedimiento de substitución es conveniente señalar que se utilizará la caja de AES, que como se sabe, dicha caja se construye a partir del campo finito que definen los polinomios de grado 7, con coeficientes en el conjunto $\{0,1\}$ y tomando como módulo el polinomio irreductible $x^8 + x^4 + x^3 + x + 1$ [1]. Vale la pena señalar que la caja es un arreglo de 16x16 elementos donde cada uno de ellos se representa mediante una cadena de 8 bits. Ahora, se divide la cadena u^i de 192 bits en 24 subcadenas de 8 bits cada una, y para cada una de estas subcadenas los primeros 4 bits definen el número de renglón de la caja y, los últimos 4 bits definen el número de columna, de tal manera que a cada subcadena de u^i se le puede asignar un elemento del arreglo. Entonces, lo que se esta haciendo es sustituir una subcadena de u^i por un elemento del arreglo. Este proceso se lleva

a cabo para cada una de las 24 subcadenas de u^i. Representemos a la cadena resultante de este proceso como v^i. La construcción de la caja de AES aparece en las referencias [2, 4].

iii) En este paso se aplica una permutación sobre la cadena v^i, la cual como se desprende del inciso anterior tiene una longitud de 192 bits. En este orden de ideas, como lo que se desea es que la permutación sea variable, entonces, es conveniente asociarla a un entero positivo [11] y no a una tabla [2]. Sin embargo, si se realiza de forma directa el tamaño del entero positivo puede ser de una magnitud de $10^{356} \cong 192! - 1$. Es en este punto donde interviene el teorema Factorial, debido a que cualquier permutación sobre una cadena de 192 bits, puede ser construida a partir de 3 permutaciones sobre cadenas de 128 bits. Lo anterior reduce el problema computacional de manera importante, ya que se trabajaría con números más pequeños; esto es, del orden $10^{215} \cong 128! - 1$. Resumiendo, lo que se propone en esta parte es aplicar una permutación a v^i, la cual se obtiene de 3 enteros positivos; digamos n_1, n_2 y n_3, del orden de 10^{215}. Llamemos a esta última permutación como Π_{192}^3.

Por último, el resultado de aplicar Π_{192}^3 a v^i es la cadena w^i, la cual es a su vez la cadena de entrada para la siguiente ronda.

En la ronda 16 se utiliza la función **g** que es ligeramente diferente a la función **f**, a continuación se muestra la función **g**:

iv) La cadena de entrada en la ronda 16 es w^{15} y de la misma forma que la función **f**, se iniciará el proceso ejecutando la operación $w^{15} \oplus K^{16} = u^{16}$. Posteriormente y de igual manera que la función **f** se realiza la operación de substitución usando la cadena u^{16}, siguiendo los pasos mostrados en el inciso ii, con lo cual se obtiene a v^{16}. A continuación y ya de forma diferente a **f**, se aplica a la cadena v^{16} la permutación $(\Pi_{192}^3)^{-1}$.

Nótese que esta última permutación cumple con la siguiente característica:

$:\Pi_{192}^3$ **o** $(\Pi_{192}^3)^{-1} = 012....m-1$ donde el símbolo **o** significa función composición.

Finalmente, se realiza la operación XOR con la llave K^{17}; esto es, $(\Pi_{192}^3)^{-1}(v^{16}) \oplus K^{17}$. El resultado del proceso anterior será el texto cifrado, el cual se representa de la siguiente manera: $(\Pi_{192}^3)^{-1}(v^{16}) \oplus K^{17} = y$.

Falta por presentar cuál es el procedimiento que genera las 17 llaves de 192 bits cada una. En este sentido, se propondrá una modificación al algoritmo que genera las llaves en AES-192 [1]. Como se sabe, el algoritmo mediante el cual se obtiene el programa de llaves para AES-192, genera un total de 1664 bits a partir de una cadena aleatoria de 192 bits y, que a la postre se convierten en 13 cadenas de 128 bits. Sin embargo, para este trabajo serán necesarias 17 cadenas de 192 bits, lo que nos da un requerimiento de 3264 bits. A continuación se muestran los pasos para producir un poco más de los bits requeridos:

i) Se inicia con una cadena aleatoria de 192 bits, digamos K, y los 3 enteros positivos n_1, n_2 y n_3 que intervinieron en el proceso de mezclado, los cuales están en un rango $2 \leq n_i \leq 128! - 1$. Llamemos a las permutaciones asociadas a los n_i como $\Pi_{128}^{n_i}$

ii) Como se sabe, a partir de estas 3 permutaciones y de acuerdo con el teorema factorial, se puede construir una permutación en las posiciones de una cadena de 192 bits, denotemos esta permutación como Π_{192}^{3}.

iii) Utilizando a Π_{192}^{3}, se puede construir otra cadena aplicando Π_{192}^{3} a K. Llamemos a esta última cadena como K^*, entonces se tienen 2 cadenas de 192 bits Ky K^*. Se sigue, que se puede aplicar el algoritmo de generación del programa de llaves de AES para ambas cadenas, lo cual da como resultado una cadena de 3328 bits, que a su vez es suficiente para obtener 17 cadenas de 192 bits, ya que se requieren 3264 de éstos.

Para terminar esta sección será conveniente presentar un ejemplo para ilustrar lo anteriormente expuesto:

Supongamos que n_1,n_2 y n_3 tienen los siguientes valores particulares, expresados en bloques de 8 bits y de forma hexadecimal n_1= 12 3C BB F8 28 1C 3C A6 18 70 10 D0 69 0C C2 D6 32 79 72 9D CD 7B CD F9 AA 63 E5 3E 87 7D FB 01 A3 3D 1C C4 7E 83 BE 86 A1 F1 C5 18 A3 7F A8 EF FA 09 72 0A 71 BB 92 A2 1B 87 AE D6 D0 51 C7 11 C6 D0 04 70 3F 53 AC 51 4A 98 B1 23 45 67 89 01 23 45 67 89 01 23 45 67 89 0

n_2 = 12 BF 82 81 C3 CA 61 87 01 0D 06 90 CC 2D 63 27 97 29 DC D7 BC DF 9A A6 3E 53 E8 77 DF B0 1A 33 D1 CC 47 E8 3B E8 6A 1F 1C 51 8A 37 FA 8E FF A0 97 20 A7 1B B9 2A 21 B8 7A ED 6D 05 1C 71 1C 6D 00 47 03 F5 3A C5 14 A9 8B 12 34 56 78 90 12 34 56 78 90 12 34 56 78 90

n_3 = 9C 28 1C 3C A6 18 70 10 D0 69 0C C2 D6 32 79 72 9D CD 7B CD F9 AA 63 E5 3E 87 7D FB 01 A3 3D 1C C4 7E 83 BE 86 A1 F1 C5 18 A3 7F A8 EF FA 09 72 0A 71 BB 92 A2 1B 87 AE D6 D0 51 C7 11 C6 D0 04 70 3F 53 AC 51 4A 98 B1 23 45 67 89 01 23 45 67 89 01 23 45 67 89 0

La llave que se propone es de la norma internacional [1], la cual se escribe de la misma forma que los n_i, en bloques de 8 bits.

K = 8E 73 B0 F7 DA 0E 64 52 C8 10 F3 2B 80 90 79 E5 62 F8 EA D2 52 2C 6B 7B

Dada la información anterior, a continuación se muestran las permutaciones asociadas a los n_i con $1 \leq i \leq 3$, Π_{192}^{3}, (Π_{192}^{3})^-1yK^*:

$\Pi_{128}^{n_1}$ = 8 20 34 108 76 93 95 86 67 11 116 47 65 79 68 0 114 26 28 35 111 101 92 54 52 123 41 118 70 16 63 42 125 33 2 3 102 12 90 105 57 19 36 59 96 94 31 75 84 21 126 113 17 55 1 15 49 44 32 14 127 27 91 115 60 73 39 81 122 5 106 117 99 29 40 88 23 4 56 22 77 103 45 72 50 74 104 124 46 24 13 37 119 9 58 10 64 7 25 107 66 98 38 110 109 112 30 87 82 53 85 100 83 78 18 71 69 120 61 43 6 48 51 62 97 121 80 89

$\Pi_{128}^{n_2}$ = 0 1 34 95 40 114 60 43 96 93 71 55 13 112 49 119 56 11 70 17 18 9 21 111 27 25 46 83 92 107 30 65 87 77 123 2 101 14 66 90 125 59 58 19 110 5 100 48 52 98 35 109 115 86 94 84 80 105 97 113 51 85 121 7 72 24 75 103 29 38 47 12 22 124 39 42 69 10 54

31 15 57 73 99 16 89 41 45 102 88 127 106 122 67 116 126 62 8 28 74 68 64 76 32 20 23 61 78 81 37 117 53 82 6 26 91 33 79 120 108 36 118 3 63 44 50 4 104

$\Pi_{128}{}^{n_3}$ = 0 1 2 11 43 114 63 99 62 79 58 103 47 56 35 7 95 117 55 18 5 71 78 100 61 126 111 28 27 80 76 30 96 77 81 64 8 59 31 72 113 116 104 24 10 38 13 70 106 121 14 41 90 6 82 124 118 93 115 46 125 12 101 66 4 22 73 68 85 119 75 107 60 44 26 16 17 98 32 39 53 109 33 19 65 112 105 40 94 83 110 54 97 88 45 9 91 89 29 51 25 67 102 49 52 84 122 123 3 87 127 37 21 34 23 15 36 20 86 57 42 120 69 74 92 108 48 50

A partir de las 3 permutaciones anteriores se puede obtener la permutación $\Pi_{192}{}^{3}$, la cual se presenta a continuación:

$\Pi_{192}{}^{3}$ = 9 112 11 76 101 15 47 59 82 145 191 22 48 2 80 64 143 94 99 163 20 84 19 50 63 35 31 184 33 52 98 188 118 123 131 155 152 108 90 106 7 111 32 14 167 54 115 128 153 119 53 10 132 77 92 182 121 49 93 116 70 96 146 13 60 73 25 159 109 178 97 87 160 157 135 43 4 176 78 183 6 88 134 103 45 29 74 175 37 24 85 147 156 171 58 129 151 141 187 39 165 56 130 154 189 62 51 72 174 5 164 83 69 162 107 173 179 150 158 148 144 169 161 177 71 149 185 117 136 46 139 30 142 126 168 3 122 0 81 105 120 66 138 23 42 55 172 137 38 86 68 102 18 91 75 186 170 114 65 180 125 79 26 8 104 89 190 67 21 17 57 166 110 140 40 95 36 44 61 124 41 100 28 127 1 16 113 181 27 133 12 34

La permutación inversa de la anterior; esto es, ($\Pi_{192}{}^{3}$)^-1*eslasiguiente* :

($\Pi_{192}{}^{3}$)^-1=
137184131357610980401630512190634351851691522201681114389661621881828513
126422819125176881489917418014475177841296125723106295045145101170947641 7
810524151581411671501126012410765861543537816114138811121901497181165381 5
354581717561703018181415183164139391143768172411186157465912732491405613 6
331791601331834795102345218982741281471421301739713216120962911191251179 6
364810335927311867721221131911010017144134121156931461151088771236911615
918755792712615598311041661 0

Cuando se aplica Π_{192} a la cadena *K* se obtiene K^* = A5 E3 53 69 1A 32 F3 D6 25 09 AC 41 22 45 37 B2 F0 EA 3E BF AA 35 C1 68

Aquí se analizará la pregunta cuánto ahorro hay en operaciones bit cuando se trabaja con números de tamaño 10^{215}, en lugar de números de tamaño 10^{356}?. Nosotros utilizaremos el concepto de big *O*, el cual establece un cálculo ”grosso modo” del número de operaciones bit [10]. En este orden de ideas, no es complicado saber que un número de 10^{356} dígitos requiere de 1200 bits para su representación y uno de 10^{215} dígitos de 750 bits aproximadamente. Ahora bien, como en realidad la inmensa mayoría de las operaciones bit que se realizan cuando se transforma un número en una permutación son las divisiones [11], entonces tomaremos en cuenta el número de operaciones bit que se requieren para ejecutar una división. Se sabe que si un número *m* se representa con *k* bits y otro número *n* se representa con *l* bits, además, con $m \geq n$. Entonces, el número de operaciones bit para encontrar el cociente y el residuo cuando se divide *m* entre *n* es $O(k * l)$ [10].

En este sentido, la peor situación que pudiera ocurrirnos cuando transformamos un número de 1200 bits a una permutación de un arreglo de 192 posiciones es $O(1200^2 * 191) = 275040000$ operaciones bit. Por otro lado, si planteamos el mismo escenario pero para un número de 750 bits, se tendría que el número de operaciones bit sería $O(750^2 * 127) = 71437500$. Se sigue que es casi 4 veces más la cantidad de operaciones bit para el primer caso que para el segundo.

Por otro lado, aclararemos porqué es importante asociarle a una permutación un entero positivo. Como se sabe, un esquema de comunicación segura en la Internet es ”Public Key Infrastructure”- PKI [6], y este tipo de protocolos utiliza frecuentemente el criptosistema RSA [26], el cual cifra llaves numéricas. En este sentido, es más conveniente enviar los enteros positivos cifrados y posteriormente convertirlos en una permutación que enviar permutaciones cifrados a través de Internet.

5.4 Aplicación: envío de cheques electrónicos

Una aplicación muy práctica para mostrar la encriptación de información utilizando a TripleDES con permutación variable es el envío de cheques electrónicos por la red. A continuación se presenta el desarrollo de esta aplicación.

Para utilizar TripleDES con permutación variable es necesario tener tres llaves n1, n2 y n3, donde n1 es un número que se encuentre entre 0 y 64!-1 y se utilizará para el cálculo de la permutación inicial variable. n1 y n2 son dos llaves de 64 bits. Para enviar las llaves n1, n2 y n3 es necesario utilizar un algoritmo de encriptación de llave pública para ser enviadas por un medio público como internet, para este caso se utilizara el criptosistema RSA.

El algoritmo para encriptar a n1, n2 y n3 es el siguiente:

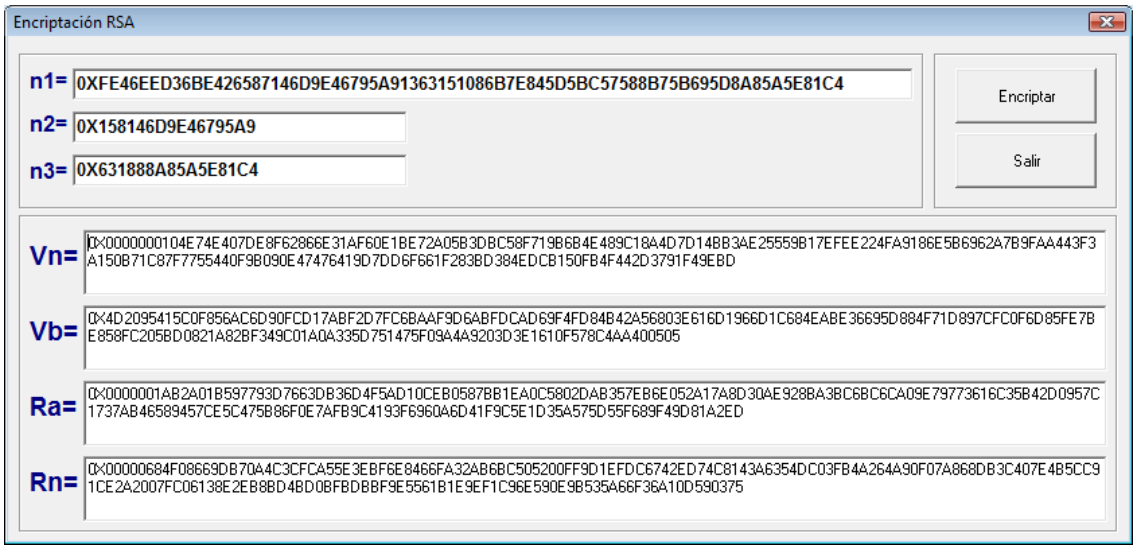
RSA_ENCRIPTA
Entrada:
n1, n2, n3,Vb, Vn números enteros.
Salida:
RSAn1, RSAn2, RSAn3 números enteros.
INICIO

$$RSAn1 \leftarrow n1^{Vb} \bmod Vn$$
$$RSAn2 \leftarrow n2^{Vb} \bmod Vn$$
$$RSAn3 \leftarrow n3^{Vb} \bmod Vn$$

FIN

donde n1,n2 y n3 son las llaves a enviar y Vb y Vn son las llaves publicas de V.

Este algoritmo se ha implementado en C++ builder y resultado se muestra en la figura.



En esta ventana se dan las llaves n2 y n3 para encriptar y desencriptar utilizando TripleDES y también n1 que es un número que se encuentra entre 0 y 64!-1 para el cálculo de la permutación variable que utilizará TripleDES. Vn y Vb son las llaves públicas del usuario V y Ra y Rb son las llaves privada y pública respectivamente del usuario R Al presionar el botón *Encriptar*, n1, n2 y n3 son encriptadas utilizando RSA con las llaves públicas de V y serán enviadas al usuario R.

Una vez encriptadas las llaves n1,n2 y n3 son enviadas por el usuario R (por cualquer medio electrónico) al usuario V quien podrá desencriptar a n1, n2 y n3 haciendo uso de RSA y sus llaves privada y pública (Ra y Rn) utilizando el siguiente algoritmo.

RSA_DESENCRIPTA
Entrada:
RSA(n1), RSA(n2), RSA(n3), Va,Vn números enteros.
Salida:
n1, n2, n3 números enteros.
INICIO

$$n1 \leftarrow RSAn1^{Va} \bmod Vn$$

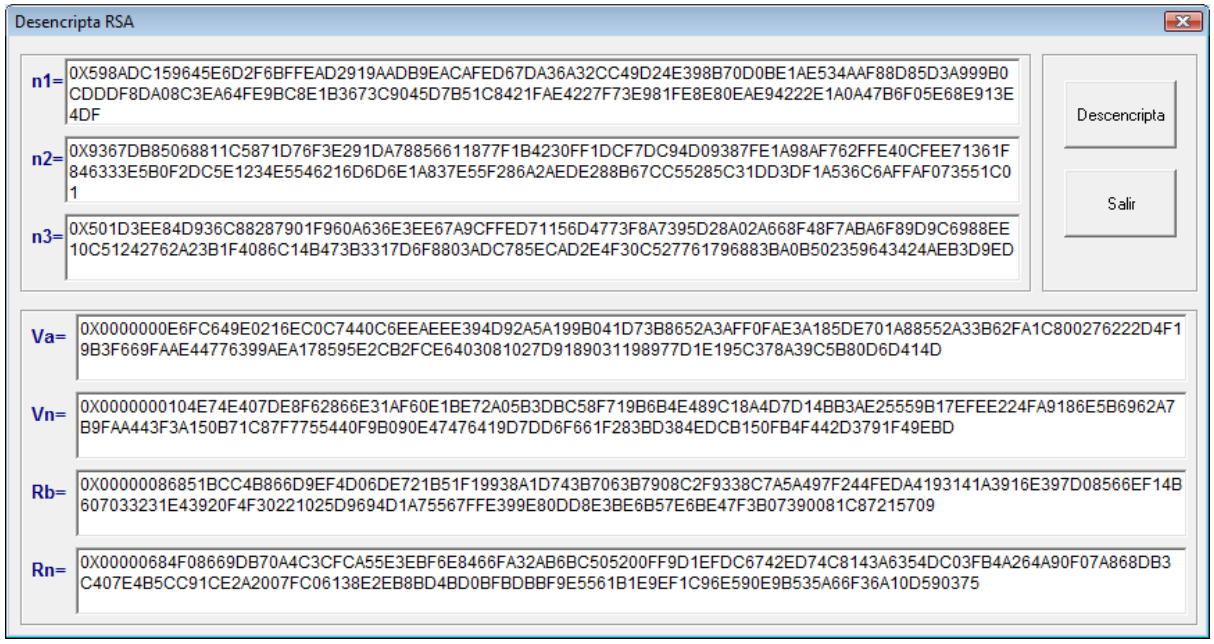
$$n2 \leftarrow RSAn2^{Va} \bmod Vn$$

$$n3 \leftarrow RSAn3^{Va} \bmod Vn$$

FIN

donde RSA(n1),RSA(n2) y RSA(n3) son las llaves encriptadas recibidas y Va y Vn son las llaves privada y pública de V.

La implementación de este algoritmo da como resultado la siguiente pantalla.



En esta ventana n1, n2 y n3 se encuentran encriptados con RSA y las llaves publicas de V.n1, n2 y n3 son números enteros de aproximadamente 230 dígitos decimales (en la ventana están representados en hexadecimal). Va y Vn son las llaves pública y privada de V (donde V es el usuario que recibe el mensaje y desea desencriptarlo). Al presionar el botón *Desencriptar* el programa hará uso del algoritmo RSA con las llaves Va y Vn para desencriptar a n1, n2 y n3.

Teniendo a n1, n2 y n3 es posible encriptar información electrónica utilizando 3DES con la permutación inicial variable, es decir, podemos por ejemplo encriptar un cheque electrónico. El algoritmo para la encriptación del cheque electrónico es el siguiente:

ENCRIPTA_TripleDES

Entrada:
Msg; cadena de caracteres
n1, n2, n3; números enteros
Salida:
E3DESMsg; cadena de caracteres
INICIO
$\prod_{I} \leftarrow Permutación(n1)$
$Msg \leftarrow Msg + Espacios(8 - (|Msg| \bmod 8))$
$E3DESMsg \leftarrow$ ""

Para $i \leftarrow 0$ hasta $|Msg|$ con incrementos de 8
Hacer

$$Msgx \leftarrow Subcadena(Msg, i, 8)$$
$$Msgx \leftarrow E3DES(Msgx, \prod_I, n2)$$
$$E3DESMsg \leftarrow E3DESMsg + E3DES(Msgx, \prod_I, n3)$$

Fin hacer

Regresar E3DESMsg

FIN

Algoritmo TripleDES

Entrada:

Msg; cadena de 8 caracteres o 64 bits

$\prod_I$ Permutación inicial variable

n2, n3 llaves de 64 bits

Salida:

E3DESMsg; cadena de 8 caracteres o 64 bits

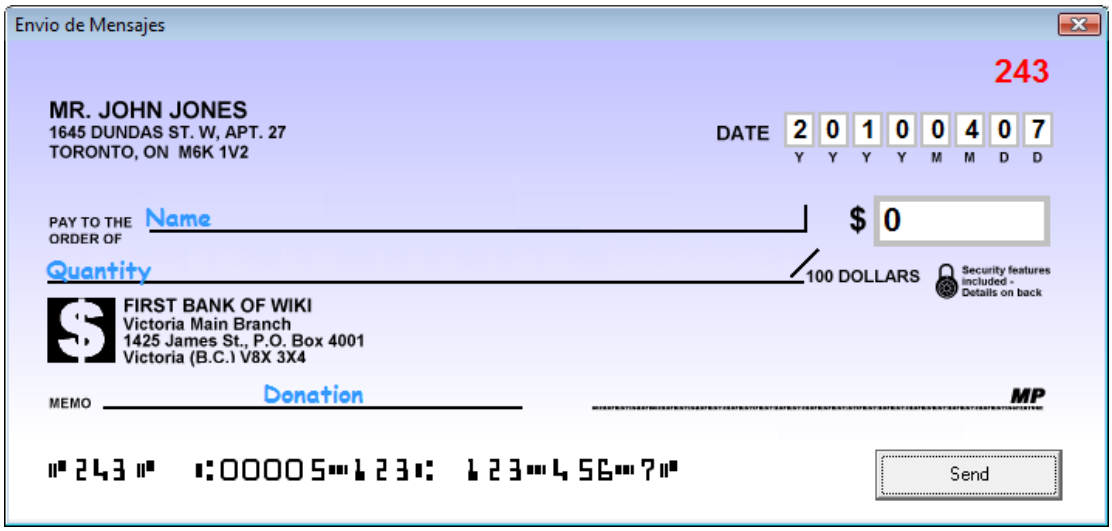
INICIO

$$E3DESMsg \leftarrow E3DES(Msg, \prod_I, n2)$$
$$E3DESMsg \leftarrow E3DES(E3DESMsg, \prod_I, n3)$$

Regresar E3DESMsg

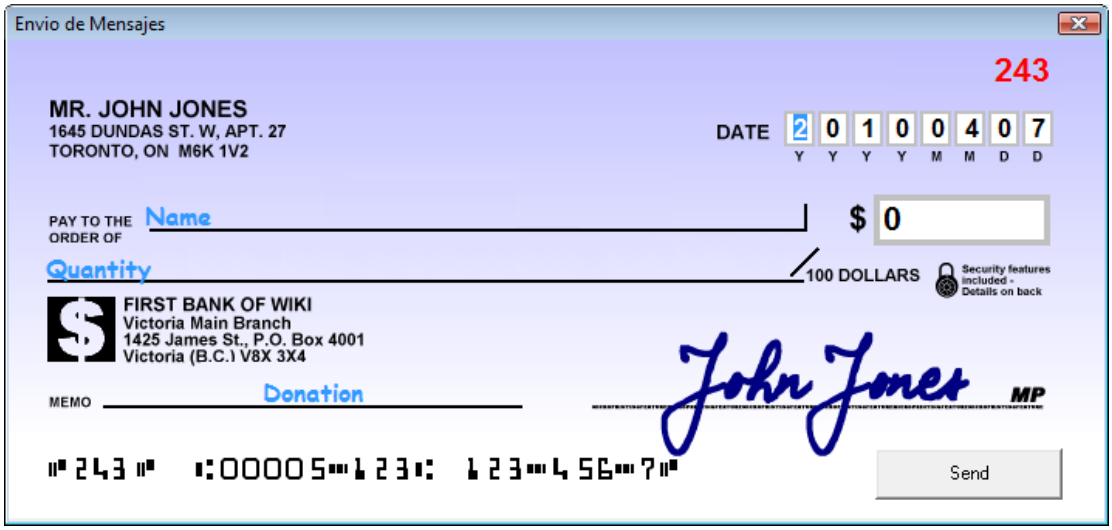
FIN

El resultado de la implementación de estos algoritmos en el lenguaje C++ builder se muestra a continuación.



En esta ventana se cargan los datos del cheque y se envía presionando el botón *Send*. Al enviarlo, se genera una cadena con toda la información del cheque la cual es dividida en palabras de 64 bits o en 8 bytes para ser encriptadas con el algoritmo 3DES haciendo uso de las llaves n1,n2 y n3. Para la garantizar la autenticidad de la información se utiliza la

función HASHA-384 y una vez encriptado y firmado electrónicamente el cheque, se mostrara la siguiente pantalla.



Finalmente el usuario V recibe el cheque y utiliza el siguiente algoritmo para desencriptar.

DESENCRIPTA_TripleDES
Entrada:
Msg; cadena de caracteres
n1, n2, n3; números enteros
Salida:
D3DESMsg; cadena de caracteres
INICIO
$\prod_I \leftarrow Permutación(n1)$
$D3DESMsg \leftarrow$ ""
Para $i \leftarrow$ 0 hasta $|Msg|$ con incrementos de 8
Hacer
$Msgx \leftarrow Subcadena(Msg, i, 8)$
$Msgx \leftarrow D3DES(Msgx, \prod_I, n2)$
$D3DESMsg \leftarrow D3DESMsg + D3DES(Msgx, \prod_I, n3)$
Fin hacer
Regresar D3DESMsg
FIN

Algoritmo DTripleDES
Entrada:
Msg; cadena de 8 caracteres o 64 bits
$\prod_I$ Permutación inicial variable
n2, n3 llaves de 64 bits
Salida:

D3DESMsg; cadena de 8 caracteres o 64 bits

INICIO

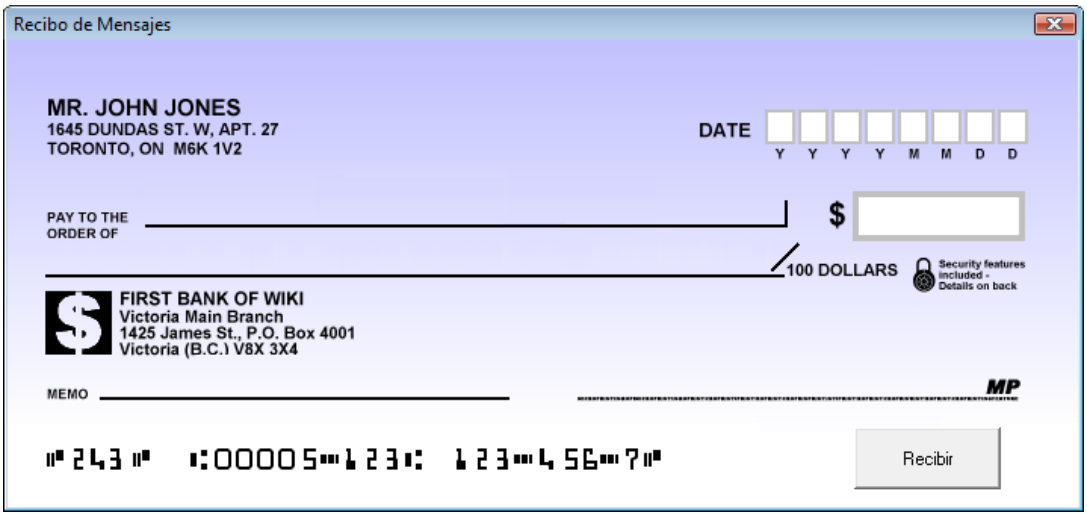
$D3DESMsg \leftarrow D3DES(Msg, \prod_{1}, n2)$

$D3DESMsg \leftarrow D3DES(D3DESMsg, \prod_{1}, n3)$

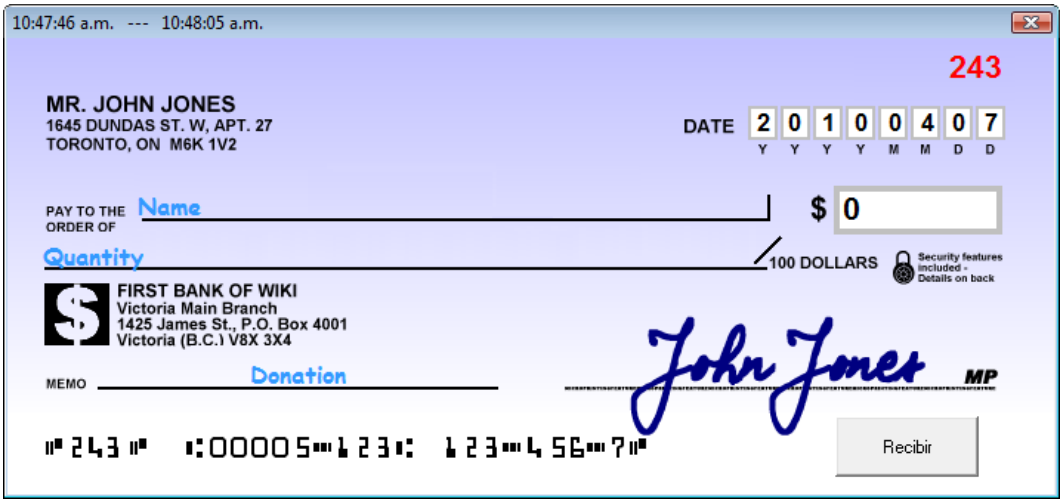
Regresar D3DESMsg

FIN

El resultado de la implementación de estos algoritmos en el lenguaje C++ builder se muestra a continuación.



En esta ventana al presionar Recibir se desencripta la información usando TripleDES y las llaves n1,n2 y n3, se calcula la huella con el algoritmo HASHA-384 y también se calcula la huella haciendo uso de la firma electrónica recibida. Si ambas huellas son correctas (quiere decir que el cheque es auténtico) se mostrará la información del cheque como se muestra a continuación.



CAPÍTULO 6

Conclusiones y Trabajos a Futuro

En este capitulo se presentan las conclusiones derivadas de los resultados obtenidos en el proceso de este trabajo de tesis; además, se proponen algunos de los posibles trabajos que se podrían realizar con objeto de continuar con las ideas propuestas aquí, dando la pauta a futuros investigadores sobre los puntos no cubiertos, pero que pudieran ser afrontados en otros trabajos de investigación.

6.1 Conclusiones

1. Una aplicación inmediata del isomorfismo I_m que va del conjunto $\mathbf{N}_m$ al conjunto $\mathbf{\Pi}_m$, es considerar a las permutaciones como llaves en los criptosistemas simétricos de tipo "Substitution Permutation Network". En este trabajo de tesis se ha mostrado de manera específica la aplicación en los criptosistemas DES y TripleDES.

2. Lo anterior permite incrementar la complejidad computacional de los criptosistemas mencionados, como es el caso de TripleDES que va de 2^{112} a 2^{300}.

3. Algo notable es que para ejecutar los ataques diferencial y lineal al criptosistema DES es necesario llegar a las cajas; sin embargo, si la permutación inicial es variable este tipo de procedimiento no puede llevarse a cabo.

4. En general, es posible afirmar que utilizando a las permutaciones como llaves, se puede incrementar la complejidad computacional de cualquier criptosistema "Substitution Permutation Network", ante un ataque de fuerza bruta y los criptoanálisis diferencial y lineal.

5. Es posible construir un campo finito en el conjunto de permutaciones $\mathbf{\Pi}_m^p$, donde $p < m!$ es un número primo.

6. Dado que la función factorial crece más rápido que la función exponencial, el número de claves en el Criptosistema Factorial crece a valores tan grandes como aproximadamente 2^{500} (10^{150}).

7. Si el algoritmo del Criptosistema Factorial se implementa en software, este sistema nos da la posibilidad de trabajar con cadenas de 64 bits en lugar de 96 bits o de 128 bits, con las evidentes ventajas que esto representa. En particular, este criptosistema es más robusto que AES debido a las permutaciones variables.

8. El teorema factorial implica un ahorro al trabajar con números de 10^{215} en lugar de 10^{356}, lo cula significa 4 veces menos operaciones bit a bit, aproximadamente.

6.2 Trabajos a Futuro

1. Creación de sistemas criptográficos de 192 bits con llaves de 192 bits del tipo Substitution Permutation Network.

2. Construcción de sistemas PKI utilizando permutaciones variables y el teorema factorial.

3. Encriptación de imágenes utilizando 3DES con permutación inicial variable. Este tipo de encriptación deberá considerar una permutación diferente de acuerdo con la posición del pixel de la imagen y con esa permutación se encriptará un bloque de 64 bits (8 pixeles en imágenes de 8 bits).

6.3 Publicaciones

[1] (2010) Una Evidencia Robusta de que el Algoritmo DES Fortalecido con una Permutación Inicial Variable es Eficiente, **Artículo aceptado** en la Revista Computación y Sistemas (Revista en el Padrón de Excelencia del CONACyT).

[2] (2009) ALGORITHM FOR STRENGTHENING SOME CRYPTOGRAPHY SYSTEMS, Journal of applied Mathematics and Decision Sciences, 967-976.

[3] (2009). CONSTRUCCIÓN DE UN CRIPTOSISTEMA USANDO LAS CAJAS DE AES Y UNA FUNCIÓN BIYECTIVA QUE VA DE LOS NÚMEROS NATURALES AL CONJUNTO DE LAS PERMUTACIONES, Revista Ciencias e Ingeniería Neogranadina, Vol. 19, No. 1, pp. 5-24, Universidad Militar Nueva Granada, Bogotá, Colombia.

[4] (2009). Detección automática de fracturas craneales en imágenes radiográficas, en Proc. de la Convención Internacional Tecnología y Salud 2009, organizada por el Instituto Superior de Ciencias Médicas de La Habana, La Habana, Cuba, 23 al 27 de marzo de 2009. ISBN: 978-959-7139-87-4.

[5] (2008). Handwritten Digit Classification Based on Alpha-Beta Associative Model, Lecture Notes in Computer Science (ISI Proceedings), LNCS 5197, Springer-Verlag Berlin Heidelberg, pp. 437-444. ISBN: 978-3-540-85919-2.
doi: 10.1007/978-3-540-85920-8_54

[6] (2007). An automatic color retrieval system based on artificial intelligence techniques, en Proc. 23rd. ISPE International Conference on CAD/CAM Robotics and Factories of the Future, Militar Nueva Granada University, Bogotá, Colombia, 16 al 18 de agosto de 2007.

Referencias

[1] A. Menezes, Hanbook of Applied Cryptography, CRC Press, New York, 1997.

[2] Douglas R. Stinson, 1995, CRYPTOGRAPHY: Theory and practice, CRC Press, pp. 70-113.

[3] Gustavus J. Simmons. ”Cryptanalysis and protocol failures”, Communications of the ACM, pp. 56-65, january 1988.

[4] J. Daemen and V. Rijmen, 1999, AES Proposal : Rijndael, AES algorithm Submission, FIPS 197.

[5] Herstein I.N., 1986, Álgebra Abstracta, Grupo Editorial Iberoamérica, pp. 22 y 11.

[6] Rosen K., 2003, Discrete Mathematics and its Applications, Mc. Graw Hill, fifth edition.

[7] Biham E. and Shamir A., 1993, “Differential cryptanalysis of the full 16-round DES”, *Lecturer Notes in computer Science*.

[8] Matsui M, 1994, “Linear Cryptanalysis for DES cipher”, *Lecture Notes in Computer Science*.

[9] Grabbe J. Orlin, 2003, “Data Encryption Standard: The DES algorithm illustrated”, *Laissez faire City time*, vol. 2, no 28.

[10] Douglas R. Stinson, 2002, *CRYPTOGRAPHY: Theory and practice,* CHAPMAN & HALL/ CRC Press, second edition, pp. 74-116.

[11] Koblitz M., 1987, *A Course in Number Theory and Cryptography*, Springer-Verlag, pp. 53-80, New York Inc.

[12] Sorking A., 1980, *LUCIFER: A cryptographic algorithm*, Cryptología 8, pp 22-35.

[13] Ritter T, 2006, *“Triple-DES is Proven to be Very Secure?”,* http://www.ciphersbyritter.com/NEWS5/PROVSEC.HTM.

[14] Stalling W, March 2006, *“Encryption Options Beyond DES”*, www.commsdesign.com.

[15] Silva García V.M. et al, 2007, “Bijective Funtion with Domain in **N** and Image in the Set of Permutations: An Application to Cryptography”, *International Journal of Computer Science and Network Security*, Vol. 7 No 4, pp. 117.

[16] Lindig Bos M., Silva García V.M., 2006, "Diseño de un dispositivo para encripción de datos en tiempo real", *CIDETEC-ESIQIE-IPN.*, vol. 2.

[17] R. Greenlaw and H. J. Hoover, 1998, *Fundamentals of the Theory of Computation,* Morgan-Kaufmann Publishers, Inc., pp. 241-257, San Francisco, California.

[18] H. Vollmer, 1999, *Introduction to Circuit Complexity: a Uniform Approach,* Springer Verlag, ISBN 3-540-64310-9.

[19] T. Leighton, 1992, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*, Morgan-Kaufmann Publishers, San Mateo, California, pp. 394.

[20] Shahana Pagen, 2005, "Actel SX-S and AX Signal and Power Integrity Considerations," NASA Goddard Space Flight Center, USA

[21] Bona Miklos, Combinatorics of Permutation, Chapman & Hall/CRC, USA, 2004.

[22] Diffie W. and Hellman M. E., Exhaustive Cryptanalysis of the NBS Data Encryption Standard, IEEE Computer Society Press, vol. 10 (1977), 74-84.

[23] FIPS 46-2, Data Encryption Standard Federal Information Processing Standards Publication 46-2, USA, 1977.

[24] Gómez A., Enciclopedia de la Seguridad Inform_atica, Alfaomega, M_exico, 2007.

[25] R.L. Rivest, A. Shamir and L. Adleman, A method for obtaining digital signatures and Public Key cryptosystems, Communications of the ACM (1978), 120-126.

[26] Silva V. Algorithm for Strengthening Some Cryptography Systems, Journal of appliedMathematics and Decision Sciences (2009), 967-976.

[27] Silva-García, V.M., Lindig-Bos, M.K.. Yáñez-Márquez, Cornelio, Flores-Carapia, R. & López-Yáñez, I. (2009). CONSTRUCCIÓN DE UN CRIPTOSISTEMA USANDO LAS CAJAS DE AES Y UNA FUNCIÓN BIYECTIVA QUE VA DE LOS NÚMEROS NATURALES AL CONJUNTO DE LAS PERMUTACIONES, Revista Ciencias e Ingeniería Neogranadina, Vol. 19, No. 1, pp. 5-24, Universidad Militar Nueva Granada, Bogotá, Colombia.

[28] Silva V. Algorithm for Strengthening Some Cryptography Systems, Journal of appliedMathematics and Decision Sciences (2009), 967-976.

[29] Douglas R. Stinson, 2006, CRYPTOGRAPHY: Theory and practice, CHAPMAN & HALL/ CRC Press, second edition, pp. 73-116.

[30] FIPS 197, 2001, Advanced Encryption Standard Federal Information Processing Standards Publication 197.

[31] Adams, C., Lloyd, S., 2002, Understanding PKI: Concepts, Standards, and Deployment Considerations, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA. ISBN: 0672323915.

[32] Fúster Sabater A. et al, *Técnicas Criptográficas de protección de datos*. Alfaomega 2ª. Edicion, capítulo, 2001.

[33] Ashkan Hosseinzadeh Namin , Huapeng Wu , Majid Ahmadi. "Comb Archi-tectures for Finite Field Multiplication in F(2^m)", IEEE Transactions on Computers, pp. 909-916, July 2007.

[34] Silva García Victor Manuel. "Criptoanálisis para la modificación de los estándares DES y Triple DES", Tesis de Doctorado, 2007.

[35] Youbo Wang , Zhiguang Tian , Xinyan Bi , Zhendong Niu. "Efficient Multiplier over Finite Field Represented in Type II Optimal Normal Basis", Sixth International Conference on Intelligent Systems Design and Applications (ISDA'06), pp. 1132-1128, october 2006.

[36] Raphael C.-W. Phan, Mohammad Umar Siddiqi. "A Framework for Describing Block Cipher Cryptanalysis", IEEE Transactions on Computers, pp. 1402-1409, november 2006.

[37] Ben Jmaa Ahmed Bassem, Jarboui Slaheddine, Bouallegue Ammar, "A PAPR Reduction Method for STBC MIMO-OFDM Systems Using SLM in Combination with Subband Permutation", Third International Conference on Wireless and Mobile Communications (ICWMC'07), pp. 88, march 2007.

Apéndice A

Una implementación en Hardware del Criptosistema Factorial

As opposed to the DES and Triple-DES systems where the permutations are fixed, the cryptosystem proposed here is based on variable permutations that, when implemented in hardware, require significant amounts of gates as well as execution times. In what follows, conventional circuit complexity concepts will be used [B8], [B9]. That is, gates (excluding inputs) possess either one or two inputs and unlimited fan-out. In order to simplify the analysis, only integer powers of 2, that is, numbers of the form $N = 2^n$ are considered.

It is clear that the propagation delay of the algorithm, excluding the permutation process, has a depth of $O(\log_2 m)$, where $m = 8$ is the length of the bit input string to the AES box [B4]. Hence, it is the permutations that contribute most significantly to the execution time, and the discussion will be limited to their implementation.

A1.1 Permutations by means of a crossbar switch.
Consider an implementation based on a crossbar switch as shown below. The input is applied to the columns, and the output is obtained from the rows. Here, the permutation is as follows:

$$0 \to 3,\ 1 \to 0,\ \ 2 \to 4,\ \ 3 \to 2 \text{ and } 4 \to 1$$

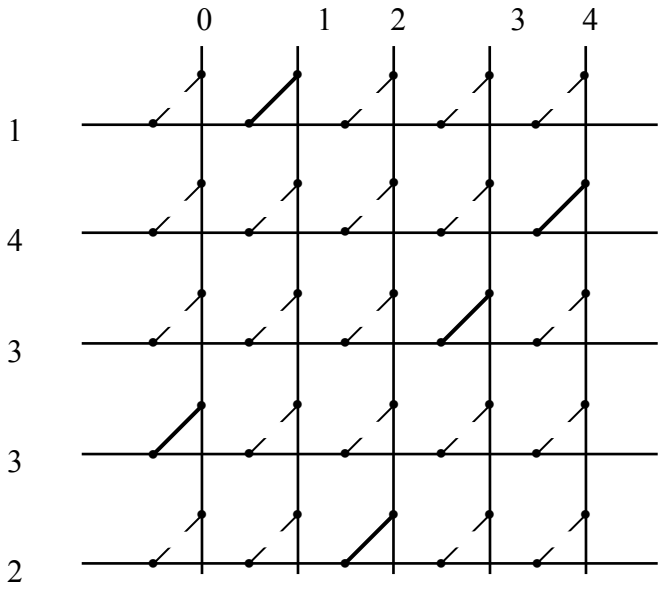


Figure 1 A permutation obtained by means of a crossbar switch

This solution requires $\mathbf{N}^2$ switches. Associated to each switch is a decoder of $\log_2 N$ inputs that closes the switch if so required. All the outputs of the switches of a given row are summed together by an OR – array of N inputs. The size and depth of the circuit are given as shown below:

Switches: N^2 AND gates, depth:1

Decoders: N NOT gates, plus $N^2(N-1)$ AND gates. The depth of the decoders is $(\log_2 N)+1$.

OR arrays: N $(N-1)$ gates, depth: $(\log_2 N)$

The total size of the circuit is $N^3 + N^2 - N$, and the total depth is $2[(\log_2 N)+1]$. For N = 64, a total of 266,144 gates are required and the total gate delay is 14. Even though this is clearly the fastest possible solution, it is also impractical because of the amounts of gates required. Recall that the algorithm requires the execution of 3 permutations on 64 bits for each of 8 rounds and, furthermore, the execution time of the displacement function has to be considered. A direct implementation on strings of 96 bits is not feasible by state-of –the art FPGA's due to the resulting circuit size.

A1.2 Permutations by means of multistage switches.

The circuit size may be considerably reduced if the permutations are executed by multistage switches. Here, we consider an implementation by means of a butterfly network.

It is well known that an N-input butterfly network may perform any permutation on N/2 inputs by means of two passes through the network [B10]. For an N – bit array, 2 sub-arrays are generated by means of even – odd separation. Each sub-array is then processed separately by the network by two successive passes. By virtue of the delta notation, the destination of each input bit defines the setting of the switches at each stage of the network, that is, no routing algorithm is required. Furthermore, there are no internal collisions possible and the total delay for any input is a function of the number of stages of the network. The figure below shows an example for N=8.

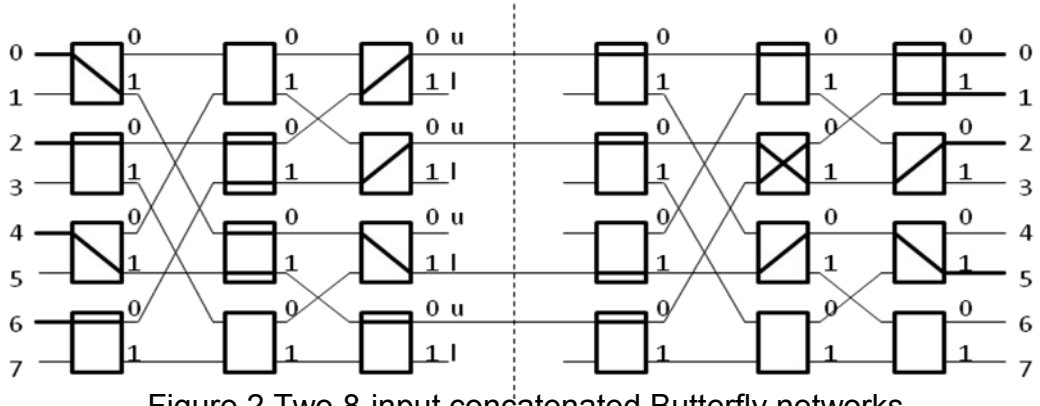


Figure 2 Two 8-input concatenated Butterfly networks

Consider the permutation:

0	1	2	3	4	5	6	7
5	4	0	6	1	7	2	5

Separating even and odd terms, the following permutations are obtained:

0	2	4	6	and	1	3	5	7	
5	0	1	2			4	6	7	5

Now, if inputs 1 to 3 constitute the lower group, and 4 to 7 the upper group, than a sufficient condition for a permutation to not cause any internal collision in the network is that the inputs are directed to outputs in an alternating fashion, that is, output 0 receives an input from the upper group, output 1 from an input in the lower group, and so on. For the even permutation, output 5 receives correctly an input from the lower group (input 0), as well as outputs 0 and 2 from inputs in the upper group. However, output 1 receives the input 4, which could cause a collision. Hence, the first butterfly redirects input 4 to output 6, and the second butterfly correctly routes input 6 to output 1. Note that this re-direction is obtained by negating the original address (001 is negated to obtain 110). Also, the switch settings are defined by the destination address, where the most significant bit sets the switch of the leftmost column, the second the intermediate column and the third the rightmost column. For example, the assignment 4 □ 6 produces a switch setting of 1, 1 and 0. The odd permutation is handled in similar fashion, except that the odd inputs of the network are now used.

This implementation requires $(N/2)\log_2 N$ switches, where each switch is a 2x2 crossbar. For $N=2^7$, 448 switches are thus necessary. Even though the size of the circuit is reasonable in terms of the resources found in FPGA's or ASIC's, it entails however a large amount of interconnections. Without considering inputs and outputs, $N[(\log_2 N)-1]$ busses are used to interconnect the stages of the network. Each bus consists of $\log_2 N$ wires that convey the destination address, plus 1 wire of data. This results in 6144 wires for $N=2^7$. We propose here an iterative, single stage implementation of the butterfly network that reduces this amount to 1024, as shown in figure 5.3.

In figure 5.3, the blocks designated as **m** are three-to-one multiplexors (AND-OR arrays), and the blocks **sw** are the switches of the butterfly network. Obviously, the outputs of the switches must be registered and the multiplexors are controlled by an iteration counter (a ring counter, control inputs I0, I1 and I2).

Given:

1. A destination address A_j, $0 \le j \le N\text{-}1$, coded in $\lceil \log_2 N \rceil$ bits
2. A group identification g, such that g = 0 for inputs $i < N/2$ and g = 1 otherwise
3. An iteration number I_k, coded as bit string of length $|\lceil \log_2 N \rceil|$ such that I_j=0 for any $j \ne k$, and $I_j = 1$ for $j = k$, where $0 \le j, k \le N\text{-}1$.

Then, the input to any switch as a function of I_k may be obtained by an AND-OR array of depth $\lceil \log_2 n \rceil + 1$, where $n = \log_2 N$. The switch setting as a function of I_k and destination address A_j may be similarly obtained and determined in parallel to the input selection. Given the switch setting, the propagation delay of the switch is constant and equal to 3 gate delays. Finally, the decision to negate, or not, the destination address in the first iteration is a function of the input group, g, and the least significant address bit, $A_j(0)$ and may be determined by an exclusive-NOR gate as shown in the truth table below:

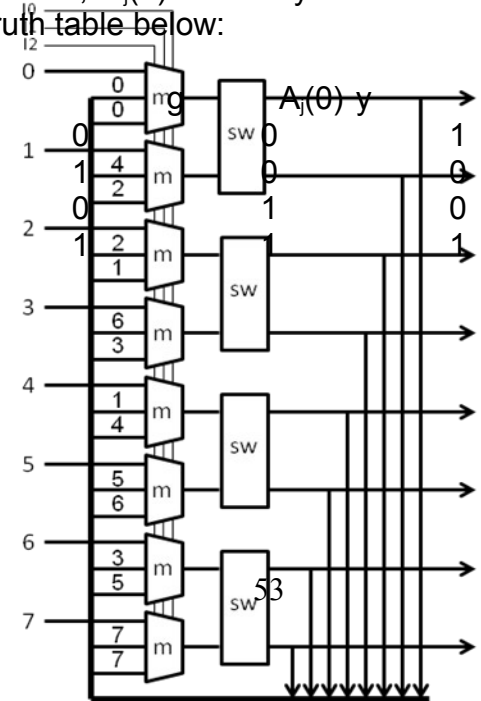


Figure 3 An 8-input, single stage Butterfly implementation

Where y = 1 stands for the negation of the address. Again, this decision delay is masked by the input determination. The negation of the address bits may be accomplished by an exclusive-OR gate for each address bit and introduces a gate delay of 1.

In view of the above, the total depth of the circuit is $\lceil \log_2 n \rceil + 5$. For N =128, n = 7, $\lceil \log_2 n \rceil = 3$ and the total depth is 8. Now, 7 iterations are required to route N/2 inputs to the outputs of the network, and this procedure has to be repeated twice for both even and odd inputs. Hence, a permutation of 128 bits requires 4(7)(8) = 224 gate delays. Note that this amount can be reduced to (224/2) +1=113 if two networks are used, one for even, and one for odd inputs.

Circuit size. In terms of the blocks shown in figure 6.3, we have:
N AND-OR arrays, of (n+1)(2n-1) = $2n^2$+n-1 gates each (input selection of n address and one data bit)
N/2 AND-OR arrays, of 2n-1 gates each (switch setting)
N arrays of 1 excl-NOR gate and n excl.-OR gates (address negation)
N/2 switches of 6(n+1) + 1 gates each

That is, a total of N[$2n^2$+6n+3] gates are required. For N =128, this amounts to 18304 gates, a value well within the resources available in an FPGA [B11].

As formerly mentioned, a permutation executed as described above requires 224 gate delays if one network is used. It is interesting to compare this value to the time required by a sequential implementation. Assume that at any given time a data bit and its corresponding destination address is an input of an N-bit register array, where the register array is composed of N D-type flip-flops. At any given time, a flip-flop either stores a data bit, or retains its former output value. The input

selection thus requires 3 gate delays and is a function of the output of an address decoder of depth $\lceil \log_2 n \rceil + 1$, where $n = \lceil \log_2 N \rceil$. Here, $N = 96$, $\lceil \log_2 N \rceil = 7$ and $\lceil \log_2 7 \rceil + 1 = 4$. Hence, the total circuit delay is 7. It follows that the sequential solution executes the permutation in 96(7) = 672 gate delays. That is, the here proposed solution is 3 times faster than the sequential solution, or 6 times if two networks are used.

Some final remarks. The former analysis is, of course, very simplistic in terms of actual circuit implementation. While the results given may accurately reflect order of magnitude values, a more precise analysis must take into consideration factors such as limited fan-out, fan-in values greater than two, the availability of building blocks that implement more complex Boolean functions than the functions here considered (NOT, AND, OR, excl. NOR and excl. OR) and, of course, wiring delays that vary with circuit geometry. The here proposed solution has been simulated in an FPGA from Actel [B11]. Clock frequencies in excess of 200 MHz were obtained. A complete encryption is computed in less than 300 cycles, or 1.5 us. That is, a bandwidth of 64 Mb/s is possible, that may be increased to 100 Mb/s with faster versions of the device we used. Of course, this value may be doubled using two butterfly networks, at the expense of greater circuit size.