INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INNOVACIÓN Y DESARROLLO
TECNOLÓGICO EN CÓMPUTO

Triple DES-96

Tésis

Que para obtener el grado de
Maestría en Tecnología de Cómputo

Presenta:
Israel Murillo Hernández

Directores:
Dr. Víctor Manuel Silva García
M. en C. Eduardo Rodríguez Escobar

Noviembre 2014.

SIP-14-BIS

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 12:20 horas del día 8 del mes de agosto del 2014 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del CIDETEC para examinar la tesis titulada:

"TRIPLE DES-96"

Presentada por el alumno:

MURILLO	HERNÁNDEZ	ISRAEL
Apellido paterno	Apellido materno	Nombre(s)

Con registro:

B	1	2	1	2	8	4

aspirante de:

Maestría en Tecnología de Cómputo

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

LA COMISIÓN REVISORA

Directores de tesis

DR. VÍCTOR MANUEL SILVA GARCÍA
Primer Vocal

M. EN C. EDUARDO RODRÍGUEZ ESCOBAR
Segundo Vocal

DR. ROLANDO FLORES CARAPIA
Presidente

M. EN C. JUAN CARLOS GONZÁLEZ ROBLES
Secretario

M. EN C. MARLON DAVID GONZÁLEZ RAMÍREZ
Tercer Vocal

M. EN C. EDUARDO VEGA ALVARADO
Suplente

PRESIDENTE DEL COLEGIO DE PROFESORES

DR. OSCAR CAMACHO NIETO

S.E.P.
INSTITUTO POLITECNICO NACIONAL
CENTRO DE INNOVACION Y DESARROLLO TECNOLOGICO EN COMPUTO

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México el día 24 del mes noviembre del año 2014, el que suscribe Israel Murillo Hernández alumno del Programa de Maestría en Tecnología de Cómputo con número de registro B121284, adscrito al Centro de Innovación y Desarrollo Tecnológico en Cómputo, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección de Dr. Víctor Manuel Silva García y del M. en C. Eduardo Rodríguez Escobar y cede los derechos del trabajo intitulado "Triple DES-96" al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

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

Israel Murillo Hernández

Resumen

Este trabajo estudia y describe el desarrollo de un criptosistema simétrico que mejora la norma ANSI X9.52 y su implementación con relación a la FIPS 46-3 que describe el estándar DES y TripleDES. En dicho criptosistema se incrementa la velocidad de ejecución en más del doble gracias a que se elimina una permutación en cada ronda de cifrado y así mismo incrementando su robustez haciendo uso de técnicas como permutaciones variables utilizando el Teorema JV, números aleatorios generados a partir de la constante matemática e y aritmética modular.

Abstract

This thesis examines and describes the development of a symmetric cryptosystem that improves ANSI X9.52 and implementation with respect to that described in FIPS 46-3 DES and TripleDES standard. In this cryptosystem speed increases executing more than doubled thanks to removed a permutation in each round encryption and likewise increasing its robustness using techniques such as permutations variables using Theorem JV, random numbers generated from the mathematical constant e and modular arithmetic.

Agradecimientos

A mis padres, que gracias a su apoyo he podido concluir esta maestría y sinceramente cualquier proyecto que me he planteado, siempre conté con su apoyo incondicional, es una dicha ser parte de esta familia. Manuel, gracias por siempre entenderme, procurarme y por ser mi inspiración al momento tomar este camino y a seguirlo en la vida. Silvia, gracias por quererme tanto y ser la súper mujer, madre y amiga que siempre estuvo conmigo. Los amo.

A Roció, gracias por impulsarme siempre, pensando en un mejor futuro juntos. Te amo.

A mi familia que son los cimientos que tengo, la unión y el cariño que nos hacen mas fuertes.

A mis amigos y profesores, gracias por todo su apoyo, por compartir sus valiosos conocimientos, por confiar en mi y por siempre darme esa palabra de aliento o a seguir esa disciplina, que se que siempre esta encaminado a mi bien, pueden contar conmigo siempre.

Índice general

Índice de figuras

Índice de cuadros

1 Introducción

La criptografía se puede entender como el conjunto de técnicas que resuelven principalmente los siguientes problemas de seguridad de la información: la autenticidad, la integridad, la confidencialidad y el no rechazo. Desde este punto de vista, la criptografía se divide en dos grandes ramas: La criptografía simétrica y la asimétrica. Esencialmente, con la primera se resuelven los problemas de confidencialidad e integridad, mientras que con la segunda se resuelven los de autenticidad y no rechazo. En general, el proceso criptográfico se aplica a un mensaje de entrada (al que se le puede llamar mensaje original), y da como resultado el mensaje cifrado. Este mensaje cifrado sólo se puede descifrar (para conocer su contenido) con la clave correspondiente. La principal diferencia entre la criptografía simétrica y asimétrica, es que en la simétrica la clave de cifrar y descifrar es la misma, mientras que en la asimétrica se tiene una clave para cifrar y otra diferente para descifrar. Otra importante diferencia es que la criptografía simétrica es muy rápida respecto a la asimétrica, por lo tanto la primera se usa para cifrar grandes cantidades de información, mientras que la segunda se usa para intercambiar información secreta pero muy corta. Sin embargo ambos tipos de criptografía se usan conjuntamente en casi todas las aplicaciones. A la criptografía simétrica pertenecen los cifradores de bloques, los cifradores de flujo y las funciones hash. De los cifradores de bloques (se llaman así porque cifran de bloque en bloque de, digamos, 64 bits), por ejemplo DES (Data Encryption Standard); o una versión más robusta, denominada TripleDES (consistente en aplicar tres veces DES). En este trabajo se desarrolla un criptosistema simétrico llamado TripleDES-96, que se basa en criptosistemas anteriormente desarrollados y que forman parte de una norma como lo es TripleDES. Este nuevo criptosistema cifra bloques de 96 bits y hace uso de herramientas como son permutaciones variables generadas con el teorema JV, la aritmética modular y la constante matemática e, lo que lo hace un criptosistema mas robusto y se elimina una permutación dentro de las rondas de DES, lo cual le da una velocidad mucho mayor.

1.1. Antecedentes

Triple DESestá descrito por la norma ANSI X.9.52[1] y la 800-67[6] las cuales describen un criptosistemas que cifra bloques de 64 bit aplicando tres veces sucesivas el algoritmo DES, el cual es un estándar FIPS que se desarrolló en los Estados Unidos en 1976

descrito en la FIPS 46-3[2]. Hoy en día, DES se considera inseguro para muchas aplicaciones. Esto se debe principalmente a que el tamaño de clave de 56 bits es corto; las claves de DES se han roto en menos de 24 horas. Existen también resultados analíticos que demuestran debilidades teóricas en su cifrado, aunque son inviables en la práctica. Para Triple DES se mejora respecto a la longitud de claves la cual incrementa su complejidad al utilizar 2 o 3 llaves siendo así de 2^{112} y de 2^{168} respectivamente.

1.2. Objetivos

1.2.1. Objetivo General

El principal objetivo de esta investigación es desarrollar un criptosistema que mejora en velocidad y robustez de la norma ANSI X9.52 Triple-DES, dicho criptosistema implementa bloques de 96 bits y es llamado Triple-DES 96[29].

1.2.2. Objetivos Particulares

- Desarrollar el criptosistema en lenguaje de programación Java.
- Que el criptosistema TripleDES-96, proponga una alternativa que mejore las prestaciones de la norma ANSI X9.52.

1.3. Justificación

Los criptosistemas tienen un ciclo de vida determinado por su complejidad y el poder de cómputo del hardware, con el cual es posible hacer una elevada cantidad de cálculos por segundo. La norma ANSI lo define a Triple-DES en ANSI x.9.52[1] así como también lo describe la NIST 800-67[6], las cuales han estado vigentes, siendo así una solución confiable y probada por ya muchos años. Incrementar su robustez y mejorar su velocidad es una solución que puede ser implementada en donde haya sido implementado Triple-DES. De esta manera se mejora un recurso probado y confiable con mejoras que demuestran que puede estar aún por más tiempo vigente.

1.4. Organización de la tesis

En este capítulo se presentan: los antecedentes, la justificación, los objetivos del trabajo de tesis. 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 presenta el criptosistema desarrollado TripleDES-96, el uso del Teorema JV en el cifrado de cadenas de 96 bits.

En el capítulo 5 se presentan la implementación en software del criptosistema, conclusiones y recomendaciones para trabajo a futuro y, finalmente, las referencias bibliográficas.

2 Estado del arte

La criptografía es la técnica que permite codificar un bloque de forma que su significado no sea conocido. Es un medio para mantener datos seguros en un entorno inseguro, en el cual un objeto original (O) se puede convertir en un objeto cifrado(C) aplicando una función de cifrado (E). Obviamente, es necesario que se pueda descifrar el bloque cifrado, para volver a obtener el bloque original aplicando una función de descifrado (D), que puede ser, o no, la inversa de E. Esta técnica permite que la información no sea inteligible para los usuarios que no conocen la forma de descifrar la información. $C = E(O)$ $O = D(C)$ Existen dos conceptos básicos en criptografía: clave y algoritmos de cifrado. La Criptografía simétrica, llamada así debido a que solo se tiene una clave tanto para cifrar como para descifrar un mensaje, ha sido la más usada a lo largo de toda la historia de la humanidad. La criptografía simétrica es usada principalmente para resolver el problema de confidencialidad de la información, uno de los problemas fundamentales de controlar en la seguridad de la información.

2.1. Claves

La clave es el patrón que usan los algoritmos de cifrado y descifrado para manipular los mensajes en uno u otro sentido. El uso de claves tiene varias ventajas, ya que permite que las funciones de cifrado y descifrado puedan ser públicas, que se puedan usar las mismas funciones para generar distintos cifrados y que el resultado cifrado no dependa únicamente del diseñador del algoritmo, sino también de una clave fija por el dueño del objeto cifrado.

2.1.1. Sistemas de Claves privadas y sistemas de clave pública o asimétrica

Los sistemas de claves privadas el emisor y el receptor comparten una clave de codificación que conocen solo ellos. El problema generado es que se basa en la ocultación de claves, por lo que si se quiere que alguien descifré el objeto cifrado debe conocer la clave con que está cifrado, podemos dar dos soluciones: propagar la clave o usar claves nuevas. Pero cuando existen muchas claves es muy difícil mantener claves confidenciales y con buenas características. El problema se puede resolver usando sistemas de

cifrado con clave publica, en el que cada usuario tiene una clave de cifrado para que cualquiera pueda enviar un mensaje cifrado de dicha clave. Sin embargo, cada usuario tiene también una clave de descifrado pública, que es secreta.

La clave pública y la privada cumple con dos propiedades:

- Se puede descodificar con la clave privada algo codificado con la clave pública.
- Se puede descodificar algo codificado con la clave privada solo si se dispone de la clave pública. Esto implica que ambas claves se pueden aplicar en cualquier orden, lo que significa que se pueden descifrar aplicando la clave privada y luego la publica. Cualquier proceso A puede enviar mensajes a B cifrados con la clave pública de B. Sin embargo, el receptor sólo los podrá descifrar si tiene la clave privada de B. Este método asegura la privacidad.
- Dando solución a que:
- No es necesario intercambiar claves para poder comunicarse con un servidor de forma segura.
- Muestran lo más posible al exterior, evitando que los intrusos tengan curiosidad por conocer las claves de los servidores.

2.2. Algoritmo de cifrado Data Encryption Standard (DES)

DES (Data Encryption Standard) es un esquema de cifrado simétrico desarrollado en 1977 por el Departamento de Comercio y la Oficina Nacional de Estándares de EEUU en colaboración con la empresa IBM, que se creó con objeto de proporcionar al público en general un algoritmo de cifrado normalizado para redes de cómputo. Estaba basado en la aplicación de todas las teorías criptográficas existentes hasta el momento, y fue sometido a las leyes de USA. Posteriormente se sacó una versión de DES implementada por hardware, que entró a formar parte de los estándares de la ISO con el nombre de DEA.

Se basa en un sistema mono alfabético, con un algoritmo de cifrado consistente en la aplicación sucesiva de varias permutaciones y sustituciones. Inicialmente el texto en claro a cifrar se somete a una permutación, con bloque de entrada de 64 bits (o múltiplo de 64), para posteriormente ser sometido a la acción de dos funciones principales, una función de permutación con entrada de 8 bits y otra de sustitución con entrada de 5 bits, en un proceso que consta de 16 etapas de cifrado.

En general, DES utiliza una clave simétrica de 64 bits, de los cuales 56 son usados para el cifrado, mientras que los 8 restantes son de paridad, y se usan para la detección de errores en el proceso. Como la clave efectiva es de 56 bits, son posible un total de

2^{56} = 72057594037927936 claves posibles, es decir, unos 72000 billones de claves, por lo que la ruptura del sistema por fuerza bruta o diccionario aún es un proceso laborioso, aunque no imposible si se dispone de suerte y una gran potencia de cálculo.

Los principales inconvenientes que presenta DES son:

- Se considera un secreto nacional de EEUU, por lo que está protegido por leyes específicas, y no se puede comercializar ni en hardware ni en software fuera de ese país sin permiso específico del Departamento de Estado.
- La clave es corta, tanto que no asegura una fortaleza adecuada. Hasta ahora había resultado suficiente, y nunca había sido roto el sistema. Pero con la potencia de cálculo actual y con el trabajo en equipo por Internet se cree que se puede violar el algoritmo, como ya ha ocurrido una vez, aunque eso sí, en un plazo de tiempo que no resultó peligroso para la información cifrada.
- No permite longitud de clave variable, con lo que sus posibilidades de configuración son muy limitadas, además de permitirse con ello la creación de limitaciones legales.
- La seguridad del sistema se ve reducida considerablemente si se conoce un número suficiente textos elegidos, ya que existe un sistema matemático, llamado Criptoanálisis Diferencial, que puede en ese caso romper el sistema en 2^{47} iteraciones.

Entre sus ventajas se puede citar:

- Es el sistema más extendido del mundo, el que más máquinas usan, el más barato y el más probado.
- Es muy rápido y fácil de implementar. DES trabajaba sobre bloques de 64 bits, teniendo la clave igual longitud. Se basa en operaciones lógicas booleanas y podía ser implementado fácilmente, tanto en software como en hardware.

2.2.1. Esquema general del algoritmo DES

La primera etapa es una permutación inicial (IP) del texto plano de 64 bits, independientemente de la clave. La última etapa es otra transposición (IP-1), exactamente la inversa de la primera. La penúltima etapa intercambia los 32 bits de la izquierda y los 32 de la derecha. Las 16 etapas restantes son una Red de Feistel de 16 rondas. En cada una de las 16 iteraciones se emplea un valor, Ki, obtenido a partir de la clave de 56 bits y distinto en cada iteración.

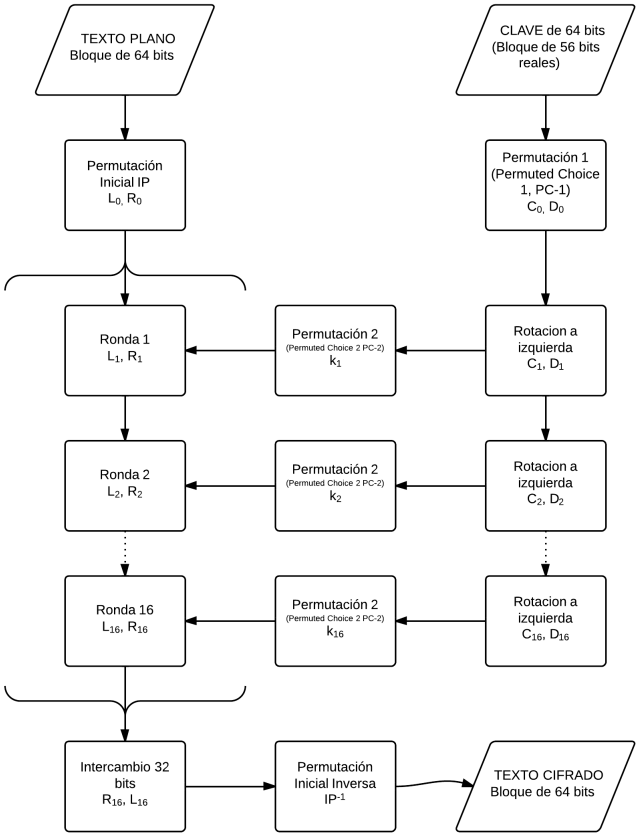

Figura 2.1: Esquema general del algoritmo DES

2.2.2. Cálculo de las subclaves

Se realiza una permutación inicial (PC-1) sobre la clave, y luego la clave obtenida se divide en dos mitades de 28 bits, cada una de las cuales se rota a izquierda un número de bits determinado que no siempre es el mismo. Ki se deriva de la elección permutada (PC-2) de 48 de los 56 bits de estas dos mitades rotadas. La función f de la red de Feistel se compone de una permutación de expansión (E), que convierte el bloque correspondiente de 32 bits en uno de 48. Después realiza una or-exclusiva con el valor Ki, también de 48 bits, aplica ocho S-Cajas de 6*4 bits, y efectúa una nueva permutación (P).

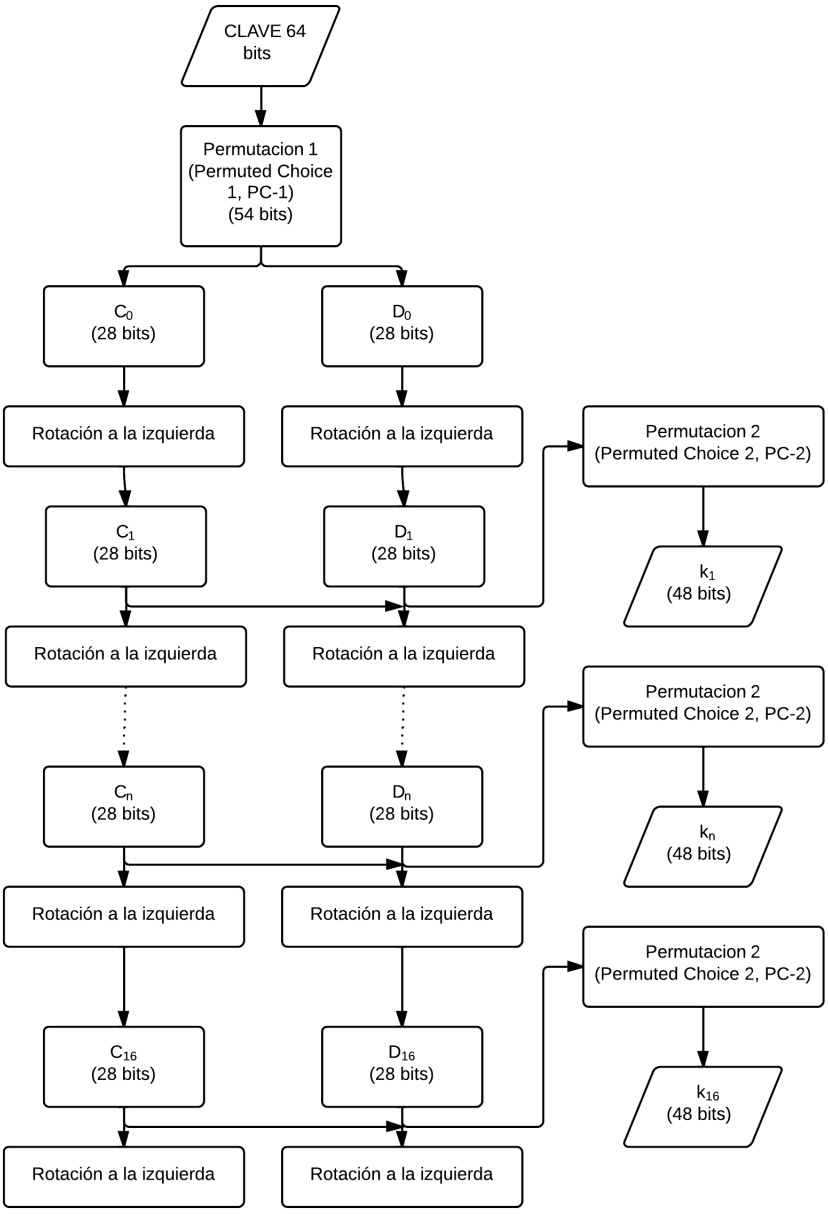

Figura 2.2: Cálculo de subclaves

2.2.3. Ronda del algoritmo DES

La forma para descifrar es el mismo algoritmo empleando las Ki en orden inverso. Está descrito oficialmente en FIPS PUB 46. A continuación se describen los pasos necesarios para implementar este algoritmo.

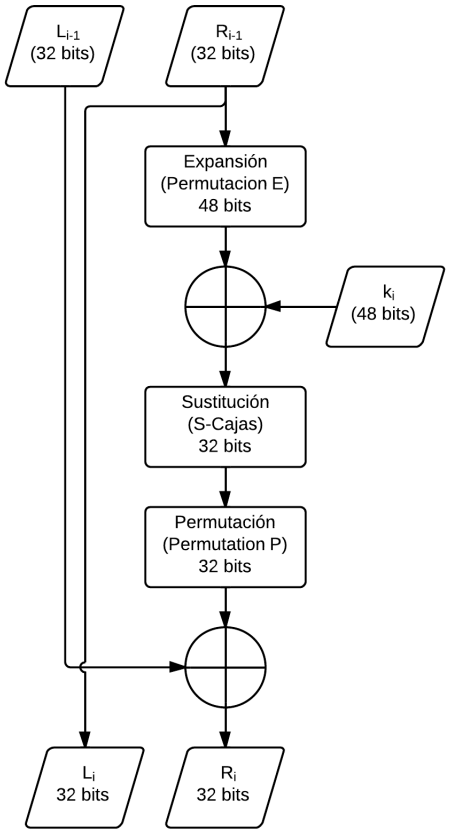

Figura 2.3: Ronda del Algoritmo DES

2.2.4. Cifrado

1. Procesar la clave.

   a) Solicitar una clave de 64 bits al usuario. La clave se puede introducir directamente o puede ser el resultado de alguna operación anterior, ya que no hay ninguna especificación al respecto. De cada uno de los ocho bytes se elimina el octavo bit (el menos significativo).

   b) Calcular las subclaves.

      1) Realizar la siguiente permutación en la clave de 64 bits reduciéndose la misma a 56 bits (El bit 1, el más significativo, de la clave transformada es el bit 57 de la clave original, el bit 2 pasa a ser el bit 49, etc.)

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

Cuadro 2.1: Permutacion PC-1

2) 1.2.2.- Dividir la clave permutada en dos mitades de 28 bits cada una. C(0) el bloque que contiene los 28 bits de mayor peso y D(0) los 28 bits restantes.

3) Calcular las 16 subclaves (Empezar con i=1)

   a' Rotar uno o dos bits a la izquierda C(i-1) y D(i-1) para obtener C(i) y D(i), respectivamente. El número de bits de desplazamiento está dado por la tabla siguiente.

Desplazamiento																
Ronda	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits despl.	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Cuadro 2.2: Desplazamiento

   b' Concatenar C(i) y D(i) y permutar como se indica a continuación. Así se obtiene K(i), que tiene una longitud de 48 bits.

Permuted Choise 2 (PC-2)							
14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

Cuadro 2.3: Permutacion PC-2

   c' Ir a 1-b-1 hasta que se haya calculado K(16).

2. Procesar el bloque de datos de 64 bits.

   a) Obtener un bloque de datos de 64 bits. Si el bloque contiene menos de 64 bits debe ser completado para poder continuar con el siguiente paso.

   b) Realizar la siguiente permutación del bloque.

Initial Permutation (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

Cuadro 2.4: Permutación Inicial

   c) Dividir el bloque resultante en dos mitades de 32 bits cada una. L(0) el bloque que contiene los 32 bits de mayor peso y R(0) el resto.

   d) Aplicar las 16 subclaves obtenidas en el paso 1.

      1) E(R(i)). Expandir R(i) de 32 a 48 bits de acuerdo con la tabla.

Expansion (E)							
32	1	2	3	4	5	4	5
6	7	8	9	8	9	10	11
12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21
22	23	24	25	24	25	26	27
28	29	28	29	30	31	32	1

Cuadro 2.5: Expansión

      2) E(R(i-1)) Xor K(i). Or-exclusiva del resultado del paso 2-d-1 con K(i)

      3) B(1), B(2),..., B(8). Partir E(R(i-1)) Xor K(i) en ocho bloques de seis bits. B(1) representa a los bits 1-6, B(2) representa a los bits 7-12,..., B(8) representa a los bits 43-48.

      4) S(1)(B(1)), S(2)(B(2)),..., S(8)(B(8)). Sustituir todos los B(j) por los valores correspondientes de las S-Cajas o tablas de sustitución (Substitution Boxes, S-Boxes) de 6*4 bits, según se indica en los sub-apartados

que siguen. Todos los valores de las S-Cajas se consideran de 4 bits de longitud. (Ver S-cajas del algoritmo DES, Cuadro 2.7).

a′ Tomar los bits 1º y 6º de B(j) y formar un número de 2 bits que llamaremos m. Este valor nos indicará la fila en la tabla de sustitución correspondiente. S(j). Obsérvese que m=0 representa la 1ª fila y m=3 la última.

b′ Con los bits 2º a 5º de B(j) formar otro número, n, de cuatro bits que indicará la columna de S(j) en la que buscar el valor de sustitución. En esta ocasión n=0 representa la 1ª columna y n=15 la última columna.

c′ Reemplazar B(j) con S(j)(m,n), m fila y n columna.

d′ Ejemplo. Sea B(3)=42, en binario B(3)=101010. Buscaremos el nuevo valor de B(3) en S(3). Fila m y columna n, según lo expuesto anteriormente m=10, n=0101, y en decimal m=2 y n=5. Por tanto, B(3) será S(3)(2,5)=15

e′ Volver a 2.4.4.1. hasta que todos los bloques B(j) hayan sido reemplazados por el valor de S(j) adecuado.

5) P[S(1)(B(1))... S(2)(B(8))]. Concatenar los bloques B(1) a B(8) y permutar los 32 bits (cuatro bits cada B(j)) en función de la Tabla:

Permutation P			
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Cuadro 2.7: Permutación P

6) R(i). Realizar una or-exclusiva entre el valor resultante y L(i-1). Este valor será R(i). Por tanto, R(i)=L(i-1) Xor P[S(1)(B(1))... S(2)(B(8))]

7) L(i). L(i)=R(i-1)

8) Repetir desde 2.4.1. hasta que se hayan aplicado las 16 subclaves.

e) Hacer la siguiente permutación del bloque R(16)L(16). Obsérvese que esta vez R(16) precede a L(16).

Final Permutation (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

Cuadro 2.8: Permutacion Final

2.2.5. Descifrado

Descifrado: Usar el mismo proceso descrito con anterioridad pero empleando las subclaves en orden inverso, esto es, en lugar de aplicar K(1) para la primera iteración aplicar K(16), K(15) para la segunda y así hasta K(1).

2.3. TripleDES

El algoritmo Triple-DES se llama así ya que consiste en aplicar en repetidas veces el algoritmo DES con diferentes llaves al mensaje original. Se puede hacer ya que el algoritmo DES no presenta estructura de grupo. El algoritmo Triple-DES responde a la siguiente estructura:

- M: Mensaje a cifrar
- C: Texto cifrado
- E_k: Cifrado usando la clave k
- D_k: Descifrado usando la clave k

De esta manera la expresión para el cifrado es:

$C = E_{K1}(D_{K2}(E_{K1}(M)))$

La expresión para el descifrado siguiendo el orden inverso de la anterior es:

$M = D_{K1}(E_{K2}(D_{K1}(C)))$

Es decir, para cifrar se codifica con la llave K1, se decodifica con K2 y volvemos a codificar con K1, para descifrar se decodifica con la llave K1, se codifica con K2 y volvemos a decodificar con K1

2.3.1. Esquema de cifrado TripleDES

Como se ha visto, el sistema DES se considera en la actualidad poco práctico, debido a la corta longitud de su clave. Para solventar este problema y continuar utilizando DES se creó el sistema Triple DES, basado en tres iteraciones sucesivas del algoritmo DES, con lo que se consigue una longitud de clave de 112 bits y 168 bits usando dos y tres llaves respectivamente, y que es compatible con DES simple. Este hecho se basa en que DES tiene la característica matemática de no ser un grupo, lo que implica que si se cifra el mismo bloque dos veces con dos llaves diferentes se aumenta el tamaño efectivo de la llave. Para este ejemplo se utilizaron dos claves de 64 bits (56 efectivos) aplicándose el siguiente proceso al documento en claro:

1. Se le aplica al documento a cifrar un primer cifrado mediante la primera clave, C1.
2. Al resultado se le aplica un descifrado con la segunda clave, C2.
3. Y al resultado se le vuelve a aplicar un tercer cifrado con la primera clave, C1.

Fila	Columna																S-Caja
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	S_1
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	S_2
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	S_3
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	S_4
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	S_5
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	S_6
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
3	4	2	3	12	9	5	15	10	11	14	1	7	6	0	8	13	
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	S_7
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	S_8
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

Cuadro 2.6: S-Cajas del Algoritmo DES

3 Materiales y Métodos

Este capítulo contiene las herramientas matemáticas que determinarán el funcionamiento del algoritmo propuesto que se usarán de manera intensiva en el capítulo 4 a fin de describir adecuadamente las aportaciones originales de esta tesis.

3.1. Fundamentos Matemáticos

3.1.1. Permutaciones

Una permutación es la variación del orden o de la disposición de los elementos de un conjunto. Por ejemplo, en el conjunto {1,2,3}, cada ordenación posible de sus elementos, sin repetirlos, es una permutación. Existe un total de 6 permutaciones para estos elementos: "1,2,3", "1,3,2", "2,1,3", "2,3,1", "3,1,2" y "3,2,1".

Una permutación de un conjunto X es una función biyectiva de dicho conjunto en sí mismo. Ejemplo de permutación considerada como función biyectiva. En el ejemplo, X={1, 2, 3}. Entonces, cada correspondencia uno a uno entre el conjunto {1, 2, 3} a sí mismo equivale a una forma de ordenar los elementos. Por ejemplo, la asignación biyectiva dada por $1 \rightarrow 1$ $2 \rightarrow 2$ $3 \rightarrow 3$ Puede hacerse corresponder al ordenamiento "1, 2, 3".

3.1.2. Permutación Variable

Una permutación de un conjunto de objetos distintos es una ordenación de esos objetos. Es bien sabido que el número de permutaciones de un conjunto de n elementos distintos se calcula a través del factorial de dicho número de elementos, (n!). Por tanto, si tuviésemos un conjunto de 8 elementos, se podría calcular el número de permutaciones existentes, el cual sería $8! = 40320$. Ahora trabajar con cadenas de 96 posiciones, se tendrían 96! permutaciones posibles de ser utilizadas, esto es $9.91\text{x}10^{149}$ aproximadamente, pero, calcularlas todas y asignarles un número para utilizar posteriormente una de ellas es imposible en este momento, dado que para cadenas de 8 posiciones, habrá 40320 permutaciones; para cadenas de 10 posiciones habrá 3628800 permutaciones; para cadenas de 20 posiciones se tienen 2432902008176640000 permutaciones, y crecen

más rápido que cualquier función del tipo f(n)=a^n. El reto es diseñar un algoritmo que sea capaz de asignar a un número entero positivo una de las n! permutaciones en un número razonable de pasos, aún más, cada vez que se desee esconder información (cifrar), Este algoritmo tiene que utilizar una permutación diferente, esto es, variable y no fija. En el caso del algoritmo DES, siempre se utiliza una permutación fija (conocida), la cual permite a los criptoanalistas tratar de romper el algoritmo realizando un análisis diferencial, dado que dicha permutación ha quedado publicada en el documento que define el funcionamiento del criptosistema.

La innovación en el algoritmo de TripleDES-96 además de tomar bloques de 96 bits con lo cual se convierte en un algoritmo más rápido, aplica una permutación que se genera en el momento en el que se corre el algoritmo, motivo por el cual incrementa su robustez.

La permutación variable se logra a partir de las llaves dadas por el usuario las cuales son concatenadas y multiplicadas por los decimales del número e el cual es un numero trascendental y al mismo tiempo presenta las características de numero normal el cual es es un número real cuyas cifras en cualquier base están distribuidas siguiendo una distribución uniforme, siendo todas las cifras igualmente probables, así como todos los pares, tríos, etc. Las cifras de ese número son tanto los de su parte entera como la sucesión infinita de dígitos que hay detrás de la coma o parte fraccionaria. Estas propiedades junto con el teorema JV y la aritmética modular nos ayudan a construir una permutación aleatoria, la cual es la base de mejoras de este algoritmo ya que agrega robustez al cifrado en general, ya que la permutación aplicada a al bloque de 96 bits no es conocida ni está definida.

3.1.3. La Constante Matemática e

Es un número irracional, no expresable por la razón de dos enteros, en otras palabras, no puede ser expresado con un número finito de cifras decimales o con decimales periódicos. Además, es un número trascendente, es decir, que no puede ser obtenido mediante la resolución de una ecuación algebraica con coeficientes racionales, también cumple con las propiedades de un número normal.

Es considerado el número por excelencia del cálculo, El simple hecho de que la función, coincida con su derivada hace que la función exponencial se encuentre frecuentemente en el resultado de ecuaciones diferenciales sencillas. Como consecuencia de esto, describe el comportamiento de acontecimientos físicos regidos por leyes sencillas, como pueden ser la velocidad de vaciado de un depósito de agua, el giro de una veleta frente a una ráfaga de viento, el movimiento del sistema de amortiguación de un automóvil o el cimbreo de un edificio metálico en caso de terremoto. De la misma manera, aparece en muchos otros campos de la ciencia y la técnica, describiendo fenómenos eléctricos y electrónicos (descarga de un condensador, amplificación de corrientes en transistores

BJT, etc.), biológicos (crecimiento de células, etc.), químicos (concentración de iones, periodos de semi-desintegración, etc.).

El número e representa una valiosa herramienta para generar permutaciones variables ya que cumple con las características de un número normal el cual es un número real cuyas cifras en cualquier base están distribuidas siguiendo una distribución uniforme, siendo todas las cifras igualmente probables, así como todos los pares, tríos, etc. Las cifras de ese número son tanto los de su parte entera como la sucesión infinita de dígitos que hay detrás de la coma o parte fraccionaria. De esta manera resulta útil al momento de generar números aleatorios a partir de sus cifras decimales.

El número e es un número irracional, y se obtiene a partir de la expresión $\left(1+\frac{1}{n}\right)^n$ haciendo n cada vez más grande[31].

Lo anterior supone que, aumentando suficientemente el valor que sustituyamos por n en la fórmula, más decimales del número e obtendremos:

$\left(1+\frac{1}{100}\right)^{100} = 2.704813...$

$\left(1+\frac{1}{1000}\right)^{1000} = 2.716023...$

$\left(1+\frac{1}{100000}\right)^{1000000} = 2.718280...$

De manera que cuando n tiende a infinito la expresión queda definida así:

$$\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n$$

Tomando un valor que actualmente sigue aumentando en su complejidad con más de 1000000000000 cifras decimales calculadas al día de hoy.

3.1.4. 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, sugerente, aritmética del reloj, ya que es un ejemplo muy utilizado de lo que es una aritmética en modulo 12.

El criptosistema TripleDES-96 utiliza la aritmética modular para la generación de la permutación variable, ya que a partir de un byte tomado de un número aleatorio generado a partir del producto de la concatenación de las claves dadas por el usuario y decimales del número e, se toman las constantes para la aplicación del teorema JV, el cual genera una permutación de 96 bits. Siendo así la aritmética modular la herramienta para crear números entre 0 y 95.

3.1.5. Teorema JV

Antes de enunciar el teorema JV, se darán ciertas condiciones matemáticas que servirán para dar formalidad al enunciado. De acuerdo con el autor del teorema[1], se tiene que: Dados dos números enteros positivos n, m tales que, $0 \leq$ n $\leq$ m!-1, utilizando el algoritmo para la División Euclidiana, el número n puede escribirse como sigue:

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

Lo anterior significa que, los valores de los C_i son cocientes cuando se realizan divisiones entre (m-1)!, (m-2)!,...,1!; a su vez cada C_i es menor que su respectiva (m-i) constante. Una vez calculados los valores de C_0,C_1,C_2,...,C_{m-2}, se está en condiciones de construir el siguiente algoritmo:

1. Se define un arreglo en orden creciente de la siguiente manera: $X[0] = 0, X[1] = 1, X[2] = 2, X[3] = 3, ..., X[m-1] = m-1$.

2. Considerando que $C_0 < m$; por tanto, $X[C_0]$ es uno de los elementos del arreglo dado en el paso 1. $X[C_0]$ se elimina del arreglo y un nuevo arreglo se define desde $X[0]$ hasta $X[m-2]$.

3. De nuevo se tiene que $C1 < m-1$; por tanto, $X[C_1]$ es uno de los elementos del arreglo definido en el paso 2. Continuando con el orden de ideas del paso anterior, $X[C_1]$ se elimina del arreglo definido en el paso 2 y un nuevo arreglo se construye desde $X[0]$ hasta $X[m-3]$.

4. Paso m-1. Procediendo de forma repetitiva como en los pasos anteriores, se llega al final del proceso, debiendo tener el arreglo: $X[C_{m-2}]$ y $X[0]$. Finalmente, el resultado que se obtiene cuando se fueron eliminando los números $X[C_0]$, $X[C_1]$, $X[C_2], ..., X[C_{m-2}]$ y $X[0]$ representa una permutación del arreglo 0,1,2,...,m-1. Entonces se puede afirmar que a cualquier $n \in N_m$, a una permutación en $m-1$ pasos.

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

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

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

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

[1]Dr. Víctor Manuel Silva Garcia, profesor investigador del CIDETC, IPN.

Paso 1	Paso 2	Paso 3	Paso 4
0	3	2	1
1	1	1	
2	2		
3			
Permutación: 0321			

Cuadro 3.1: permutación 0

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

Paso 1	Paso 2	Paso 3	Paso 4
0	3	2	2
1	1	1	
2	2		
3			
Permutación: 0312			

Cuadro 3.2: permutación 1

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

Paso 1	Paso 2	Paso 3	Paso 4
0	3	3	2
1	1	2	
2	2		
3			
Permutación: 0132			

Cuadro 3.3: permutación 2

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

Paso 1	Paso 2	Paso 3	Paso 4
0	3	3	3
1	1	2	
2	2		
3			
Permutación: 0123			

Cuadro 3.4: permutación 3

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

Paso 1	Paso 2	Paso 3	Paso 4
0	3	3	1
1	1	1	
2	2		
3			
Permutación: 0231			

Cuadro 3.5: permutación 4

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

Paso 1	Paso 2	Paso 3	Paso 4
0	3	3	3
1	1	1	
2	2		
3			
Permutación: 0213			

Cuadro 3.6: permutación 5

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

Paso 1	Paso 2	Paso 3	Paso 4
0	0	2	3
1	3	3	
2	2		
3			
Permutación: 1023			

Cuadro 3.7: permutación 6

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

Paso 1	Paso 2	Paso 3	Paso 4
0	0	2	2
1	3	3	
2	2		
3			
Permutación: 1032			

Cuadro 3.8: permutación 7

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

Paso 1	Paso 2	Paso 3	Paso 4
0	0	0	2
1	3	2	
2	2		
3			
Permutación: 1302			

Cuadro 3.9: permutación 8

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

Paso 1	Paso 2	Paso 3	Paso 4
0	0	0	0
1	3	2	
2	2		
3			
Permutación: 1320			

Cuadro 3.10: permutación 9

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

Paso 1	Paso 2	Paso 3	Paso 4
0	0	0	3
1	3	3	
2	2		
3			
Permutación: 1203			

Cuadro 3.11: permutación 10

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

Paso 1	Paso 2	Paso 3	Paso 4
0	0	0	0
1	3	3	
2	2		
3			
Permutación: 1230			

Cuadro 3.12: permutación 11

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

Paso 1	Paso 2	Paso 3	Paso 4
0	0	3	1
1	1	1	
2	3		
3			
Permutación: 2031			

Cuadro 3.13: permutación 12

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

Paso 1	Paso 2	Paso 3	Paso 4
0	0	3	3
1	1	1	
2	3		
3			
Permutación: 2013			

Cuadro 3.14: permutación 13

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

Paso 1	Paso 2	Paso 3	Paso 4
0	0	0	3
1	1	3	
2	3		
3			
Permutación: 2103			

Cuadro 3.15: permutación 14

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

Paso 1	Paso 2	Paso 3	Paso 4
0	0	0	0
1	1	3	
2	3		
3			
Permutación: 2130			

Cuadro 3.16: permutación 15

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

Paso 1	Paso 2	Paso 3	Paso 4
0	0	0	1
1	1	1	
2	3		
3			
Permutación: 2301			

Cuadro 3.17: permutación 16

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

Paso 1	Paso 2	Paso 3	Paso 4
0	0	0	0
1	1	1	
2	3		
3			
Permutación: 2310			

Cuadro 3.18: permutación 17

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

Paso 1	Paso 2	Paso 3	Paso 4
0	0	2	1
1	1	1	
2	2		
3			
Permutación: 3021			

Cuadro 3.19: permutación 18

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

Paso 1	Paso 2	Paso 3	Paso 4
0	0	2	2
1	1	1	
2	2		
3			
Permutación: 3012			

Cuadro 3.20: permutación 19

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

Paso 1	Paso 2	Paso 3	Paso 4
0	0	0	2
1	1	2	
2	2		
3			
Permutación: 3102			

Cuadro 3.21: permutación 20

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

Paso 1	Paso 2	Paso 3	Paso 4
0	0	0	0
1	1	2	
2	2		
3			
Permutación: 3120			

Cuadro 3.22: permutación 21

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

Paso 1	Paso 2	Paso 3	Paso 4
0	0	0	1
1	1	1	
2	2		
3			
Permutación: 3201			

Cuadro 3.23: permutación 22

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

Paso 1	Paso 2	Paso 3	Paso 4
0	0	0	0
1	1	1	
2	2		
3			
Permutación: 3210			

Cuadro 3.24: permutación 23

Si representamos todas estas permutaciones en una tabla, se puede 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.

Resulta más atractivo aún pensar que, este procedimiento se puede realizar en m-1 pasos. Por tanto, al hablar de cadenas de longitud m, (m=96 bits, por ejemplo), se podrá construir una permutación de entre m! (96!) permutaciones posibles, asociándole un número entero positivo $n(0 \leq n \leq 96! - 1)$ en m-1 (95) pasos.

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

Cuadro 3.25: Tabla de permutaciones 4!

3.1.6. Ejemplo

Suponiendo que se trabaja con cadenas de una longitud 8. Una permutación de dichas posiciones es un arreglo particular de los números 0, 1, 2, 3, 4, 5, 6 y 7; por ejemplo 8,5,3,2,6,1,7,4. Ahora, sea un número entero no negativo tal que $0 \leq n \leq 8!–1$; digamos

n = 20111. Este número natural puede ser expresado como sigue:

$0 \leq n \leq 8!–1$

$8! - 1 = 40319$

$20111 = 3(7!) + 6(6!) + 5(5!) + 2(4!) + 3(3!) + 2(2!) + 1(1!)$

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.

Posición	Paso 1	Paso 2	Paso 3	Paso 4	Paso 5	Paso 6	Paso 7
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	
2	2	2	2	4	4		
3	7	7	7	7			
4	4	4	4				
5	5	5					
6	6						
7							
Permutación: 36527410							

Cuadro 3.26: Permutación número 20111 asignada a 8!-1

como se puede observar el número que es sacado de la posición correspondiente es remplazado por el que ocupa la última posición del arreglo, esto con el fin de hacer el algoritmo más rápido, ya que podría recorrerse todo el arreglo una posición pero en términos de rendimiento implica un mayor tiempo de procesamiento.

4 Modelo Propuesto TripleDES-96

Dentro de este apartado se plasma el diseño del algoritmo de cifrado empleando el sistema criptográfico TripleDES-96 de permutación variable.

TripleDES-96 es una implementación de TripleDES[1] con mejoras que hacen este criptosistema más robusto y aún muy seguro, dicho criptosistema se basa en cifrar bloques de 96 bits, quitar una permutación (P) usada en TripleDES, además de realizar una permutación variable en el primer y tercer ciclo de DES-96, tal permutación es aplicada gracias a una variación del Teorema JV. También el hecho de cifrar bloques de 96 bits nos da la ventaja de ser más rápido que cifrar bloques de 64 bits, asimismo de que al separar en bloques L y R de 48 bits cada uno, estos pueden pasar directamente a hacer una operación ex-or con la llave que también es de 48 bits, lo cual nos permite reducir el número de operaciones en cada ronda, proporcionando así una mayor velocidad al algoritmo.

4.1. Análisis

En esta etapa del proyecto se describen los requerimientos definidos para el cifrado y descifrado, para ello es necesario realizar las siguientes funciones:

4.1.1. Cifrado

Se segmenta el archivo en bloques de 96 bits y se aplican tres rondas sucesivas del algoritmo DES-96. En la primera ronda del algoritmo se realiza una permutación variable utilizando una variante del Teorema JV utilizando un cifrado con K1, en seguida se aplica la segunda ronda del algoritmo en la que se realiza una etapa de descifrado con K2 y por último una tercer ronda del algoritmo en la cual se utiliza la permutación variable inversa de la permutación generada en la primer ronda además de un cifrado con K1, obteniendo el bloque de bit's cifrados.

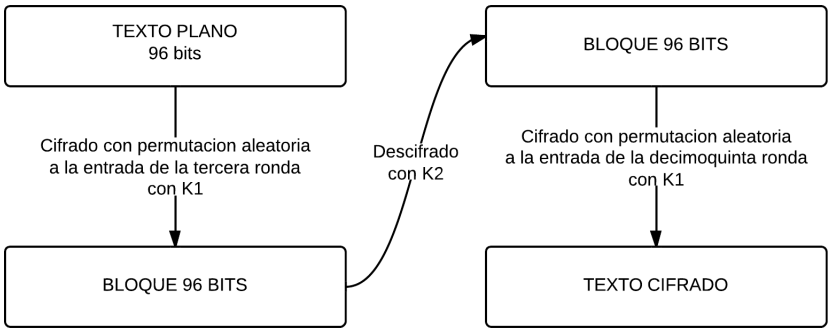

Figura 4.1: Cifrado TripleDES-96

4.1.2. Descifrado

Se segmenta el archivo en bloques de 96 bits, en la primera ronda del algoritmo se realiza una permutación variable utilizando una variante del Teorema JV y se aplica un descifrado con K1. Aplicar la segunda ronda del algoritmo en la que se realiza una etapa de cifrado con K2. En la tercer ronda del algoritmo se utiliza la inversa de la permutación variable que se generó en la primer ronda y se aplica un descifrado con K1 Obteniendo el bloque de bit's original.

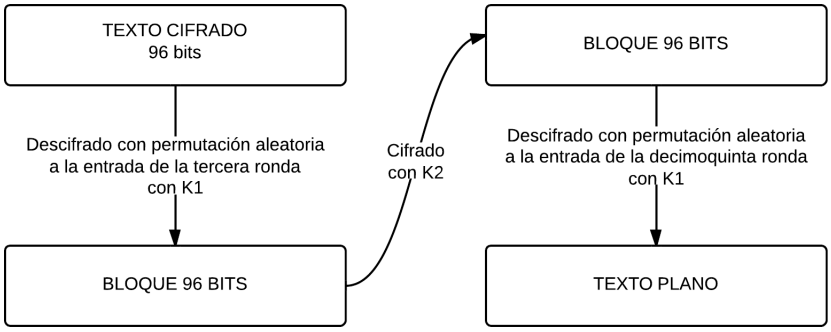

Figura 4.2: Descifrado TripleDES-96

4.2. Funcionamiento

Debido a que el algoritmo de DES-96 divide el bloque de entrada en dos bloques para procesar únicamente uno por ronda (feistel), es necesario que dos rondas se procesen

para que el bloque completo sea cambiado. De acuerdo con este razonamiento para poder aplicar una permutación efectiva que involucre todo el bloque con el que se trabaja, ésta (IP) debe de aplicarse después de la segunda ronda del algoritmo DES-96, de la misma manera que a la hora de aplicar la permutación inversa al final del ciclo, ésta se aplique cuando aun restan dos rondas del algoritmo, lo que significa que la IP^{-1}debe ser aplicada a la salida de la decimocuarta ronda, para así terminar el ciclo de 16 rondas del algoritmo DES-96.

Ya que el funcionamiento de TripleDES-96 involucra tres ciclos de DES-96 los cuales emplean una secuencia de Cifrado-Descifrado-Cifrado para el caso del cifrado y de Descifrado-Cifrado-Descifrado para el caso de Descifrar, resulta innecesario que se aplique en las tres etapas la permutación, ya que si vemos a TripleDES-96 como un todo, basta con tener una permutación al inicio (IP) y una al final (IP^{-1}). Ya que aplicar una serie de permutaciones consecutivas el efecto es el mismo al de aplicar una sola. Por tal motivo de los tres ciclos de DES-96 aplicados solo en el primero y el tercero se utilizan las permutaciones. Estas permutaciones no son fijas ya que se generan al momento y utilizan como fuente las Llaves proporcionadas por el usuario, K1 y K2, a través del Teorema JV y la constante matemática e.

El número de llaves utilizadas por TripleDES-96 pueden ser dos o tres, dependiendo del nivel de complejidad que se quiera alcanzar, estas llaves son proporcionadas por el usuario y pueden verse de la siguiente manera:

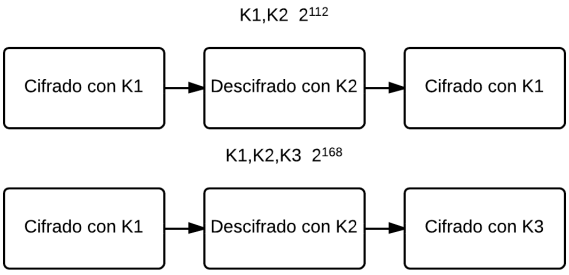

Figura 4.3: Llaves en el Cifrado de TripleDES-96

De la misma manera el proceso inverso por el cual se recupera la información final puede y debe coincidir con el número de llaves utilizadas, sean dos o tres según el caso, lo cual se ve de la siguiente manera:

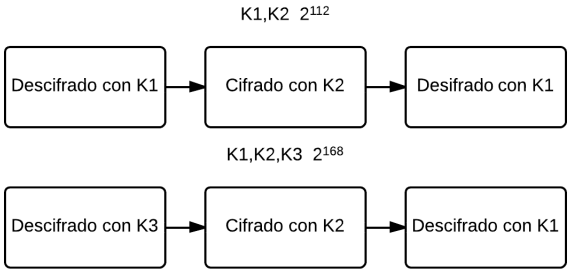

Figura 4.4: Llaves en el Descifrado de TripleDES-96

El algoritmo contempla tres ciclos los cuales son Cifrado-Descifrado-Cifrado para Cifrar y Descifrado-Cifrado-Descifrado, para descifrar. Cada uno de los ciclos opera con llaves dadas por el usuario, de manera que la complejidad del algoritmo esta dado también por el número de llaves utilizadas, de esta manera se pueden utilizar dos o tres siendo su complejidad de 2^{112} y 2^{168} respectivamente.

4.3. DES-96

La principal mejora con respecto a DES es que se elimina una permutación (P) en cada una de sus 16 rondas en la cual al ser cifrado tipo Feistel se tienen que aplicar dos veces para cifrar el bloque completo, lo que significa que para cifrar un bloque completo es necesarias 32 rondas, eliminando una permutación que se efectúa en cada ronda incrementa su velocidad de ejecución. Además una permutación variable es agregada en el ciclo de cifrado.

El algoritmo puede dividirse en cuatro formas de aplicarlo, esto depende de en que ronda se encuentre, ya que solo a la entrada de la tercera ronda y a la entrada de la decimoquinta ronda de los ciclos uno y tres, es aplicada la permutación variable. por lo tanto nos genera las siguientes posibilidades:

1. Cifrado sin permutación
2. Cifrado con Permutación Variable
3. Descifrado sin permutación
4. Descifrado con Permutación Variable

- Para el caso 1, el diagrama queda de la siguiente manera:

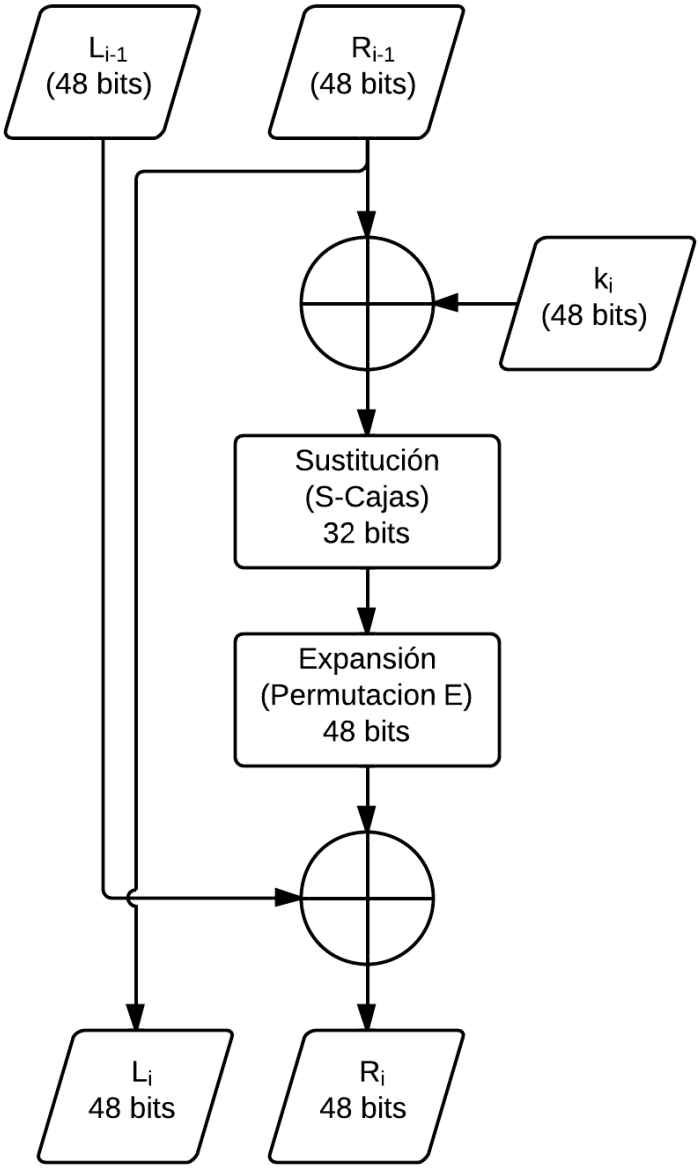

Figura 4.5: Ronda DES-96

- El caso 3 es el mismo que el 1 solo que el orden de las llaves es decreciente, ya que se trata de un descifrado.
- El caso número 2 se aplica la permutación variable, la cual se aplica en el primer

y tercer ciclo DES-96. Se hace uso de ella a la entrada de la tercer y decimoquinta rondas del algoritmo DES-96, quedando de esta manera:

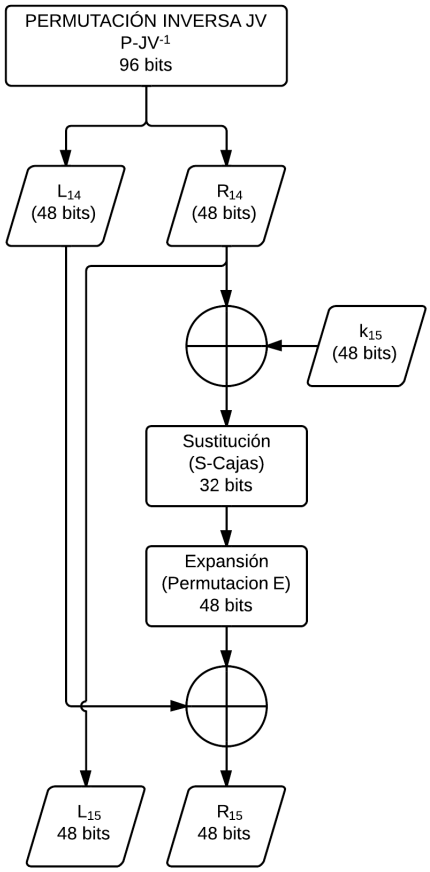

Figura 4.6: Ronda DES-96 con Permutación Variable

- Por último el caso número 4 es el mismo que el caso 2 donde de nuevo la única diferencia es el orden de aplicación de las llaves, que en este caso es en un orden decreciente.

La generación de las llaves es realizada de la misma manera que en DES, la cual esta especificada en la norma FIPS 46-3.

4.3.1. Permutación Variable en TripleDES-96

La permutación variable en TripleDES-96 es una herramienta que ayuda a mejorar la robustez del algoritmo incrementando las prestaciones en la norma FIPS 46-3. Esto

se logra generando una permutación única no definida en ninguna norma ya que es generada con las llaves proporcionadas por el usuario y haciendo uso de las propiedades de la constante matemática e.

El cifrado de bloques basado en Feistel divide al bloque a la mitad ejecutándose solo en una mitad por vez haciendo que sean necesarios dos rondas de cifrado para abarcar el bloque completo, a la entrada de la tercera y de la decimoquinta ronda, es donde es aplicada la permutación variable y la permutación variable inversa, respectivamente.

4.3.2. Cifrado

1. Procesar la clave.

   a) Solicitar una clave de 64 bits al usuario. De cada uno de los ocho bytes se elimina el octavo bit (el menos significativo).

   b) Calcular las subclaves.

      1) Realizar la siguiente permutación en la clave de 64 bits reduciéndose la misma a 56 bits (El bit 1, el más significativo, de la clave transformada es el bit 57 de la clave original, el bit 2 pasa a ser el bit 49, etc.)

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

Cuadro 4.1: Permutación PC-1

      2) Dividir la clave permutada en dos mitades de 28 bits cada una. C(0) el bloque que contiene los 28 bits de mayor peso y D(0) los 28 bits restantes.

      3) Calcular las 16 subclaves (Empezar con i=1)

         a' Rotar uno o dos bits a la izquierda C(i-1) y D(i-1) para obtener C(i) y D(i), respectivamente. El número de bits de desplazamiento está dado por la tabla siguiente.

Desplazamiento																
Ronda	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits despl.	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Cuadro 4.2: Desplazamiento

b′ Concatenar C(i) y D(i) y permutar como se indica a continuación. Así se obtiene K(i), que tiene una longitud de 48 bits.

(PC-2)							
14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

Cuadro 4.3: Permutación PC-2

c′ Ir a 1-b-1 hasta que se haya calculado K(16).

2. Procesar el bloque de datos de 96 bits.

   a) Obtener un bloque de datos de 96 bits. Si el bloque contiene menos de 96 bits debe ser completado para poder continuar con el siguiente paso.

   b) Dividir el bloque resultante en dos mitades de 48 bits cada una. L(0) el bloque que contiene los 48 bits de mayor peso y R(0) el resto.

   c) Aplicar las 16 subclaves obtenidas en el paso 1.

   d) Si es la ronda 3 aplicar permutación variable

   e) si es la ronda 15 aplicar permutación variable inversa

      1) E(R(i-1)) Xor K(i). Or-exclusiva del resultado del paso 2-d-1 con K(i)

      2) B(1), B(2),..., B(8). Partir E(R(i-1)) Xor K(i) en ocho bloques de seis bits. B(1) representa a los bits 1-6, B(2) representa a los bits 7-12,..., B(8) representa a los bits 43-48.

      3) S(1)(B(1)), S(2)(B(2)),..., S(8)(B(8)). Sustituir todos los B(j) por los valores correspondientes de las S-Cajas o tablas de sustitución (Substitution Boxes, S-Boxes) de 6*4 bits, según se indica en los subapartados que siguen. Todos los valores de las S-Cajas se consideran de 4 bits de longitud. (Ver S-cajas del algoritmo DES, Cuadro 4.5).

a′ Tomar los bits 1º y 6º de B(j) y formar un número de 2 bits que llamaremos m. Este valor nos indicará la fila en la tabla de sustitución correspondiente. S(j). Obsérvese que m=0 representa la 1ª fila y m=3 la última.

b′ Con los bits 2º a 5º de B(j) formar otro número, n, de cuatro bits que indicará la columna de S(j) en la que buscar el valor de sustitución. En esta ocasión n=0 representa la 1ª columna y n=15 la última columna.

c′ Reemplazar B(j) con S(j)(m,n), m fila y n columna.

d′ Ejemplo. Sea B(3)=42, en binario B(3)=101010. Buscaremos el nuevo valor de B(3) en S(3). Fila m y columna n, según lo expuesto anteriormente m=10, n=0101, y en decimal m=2 y n=5. Por tanto, B(3) será S(3)(2,5)=15

e′ Volver a 2.4.4.1. hasta que todos los bloques B(j) hayan sido reemplazados por el valor de S(j) adecuado.

f′ E[S(1)(B(1))... S(2)(B(8))]. Concatenar los bloques B(1) a B(8) y E(R(i)). Expandir R(i) de 32 a 48 bits de acuerdo con la tabla 4.4.

Expansión (E)							
32	1	2	3	4	5	4	5
6	7	8	9	8	9	10	11
12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21
22	23	24	25	24	25	26	27
28	29	28	29	30	31	32	1

Cuadro 4.4: Expansión

g′ Volver a 2.4.4.1. hasta que todos los bloques B(j) hayan sido reemplazados por el valor de S(j) adecuado.

Fila	Columna																S-Caja
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	S_1
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	S_2
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	S_3
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	S_4
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	S_5
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	S_6
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
3	4	2	3	12	9	5	15	10	11	14	1	7	6	0	8	13	
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	S_7
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	S_8
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

Cuadro 4.5: S-Cajas del Algoritmo DES

4) R(i). Realizar una or-exclusiva entre el valor resultante y L(i-1). Este valor será R(i). Por tanto, R(i)=L(i-1) Xor E[S(1)(B(1))... S(2)(B(8))]

5) L(i)=R(i-1)

6) Repetir desde 2.4.1. hasta que se hayan aplicado las 16 subclaves.

4.3.3. Descifrado

Descifrado: Usar el mismo proceso descrito con anterioridad pero empleando las subclaves en orden inverso, esto es, en lugar de aplicar K(1) para la primera iteración aplicar K(16), K(15) para la segunda y así hasta K(1).

5 Implementación

Una vez expuesto el diseño del algoritmo TripleDES-96, se procedió a su implementación por software, usando el lenguaje de programación Java en un entorno de desarrollo Netbeans 7.4.

5.1. Metodología

Previo a la planificación, es necesario decidir qué modelo de ciclo de vida se va a utilizar en el desarrollo de la aplicación. De entre los diferentes modelos (lineal, cascada, sashimi, etc.) se ha elegido el modelo evolutivo.

Este modelo es ideal para el proyecto ya que ofrece gran flexibilidad para la inclusión de nuevos requerimientos, ya que es muy complicado obtenerlos todos al inicio, permitiendo añadir o mejorar la funcionalidad de los diferentes módulos del proyecto en cualquier momento. Como muestra la figura 5.1 este modelo se basa en la iteración del ciclo requerimientos-desarrollo-evaluación

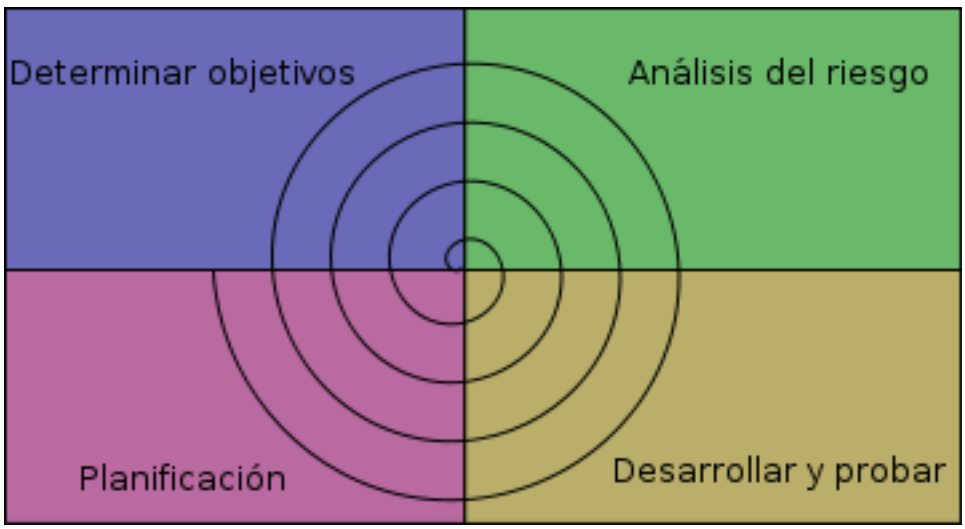

Figura 5.1: Modelo Evolutivo

Finalizada la fase de desarrollo se tiene una versión del producto que es evaluada, tras la cual se pueden añadir requerimientos nuevos o revisar los actuales si la evaluación no es satisfactoria, consiguiendo que el producto se pueda ir puliendo y mejorando continuamente.

En los diferentes módulos o componentes del proyecto también se ha aplicado este modelo de manera que ha sido posible depurarlos por separado hasta obtener la versión final. Los componentes son:

- Formación inicial: Debe familiarizarse con los elementos necesarios para el desarrollo del proyecto y estudiar tanto el entorno de desarrollo como con la plataforma para la que se implementa la aplicación.

- Memoria y documentación: A medida que se avanza en el desarrollo del proyecto es necesario recoger toda la documentación necesaria.

- Aspectos y estructura básica: Este módulo aporta la funcionalidad básica para que la aplicación funcione.

- Interfaz gráfico: Se diseñan las apariencias de las diferentes pantallas y se crean los diferentes menús, diálogos e iconos.

- Evaluación y pruebas: Una vez obtenida la versión completa se debe evaluar y comprobar que todo funciona según lo esperado.

5.2. Funcionamiento

Además del bloque de 96 bits a cifrar, se le pide al usuario dos llaves de 64 bits y se define previamente una sucesión de números aleatorios que forman parte de la constante matemática e.

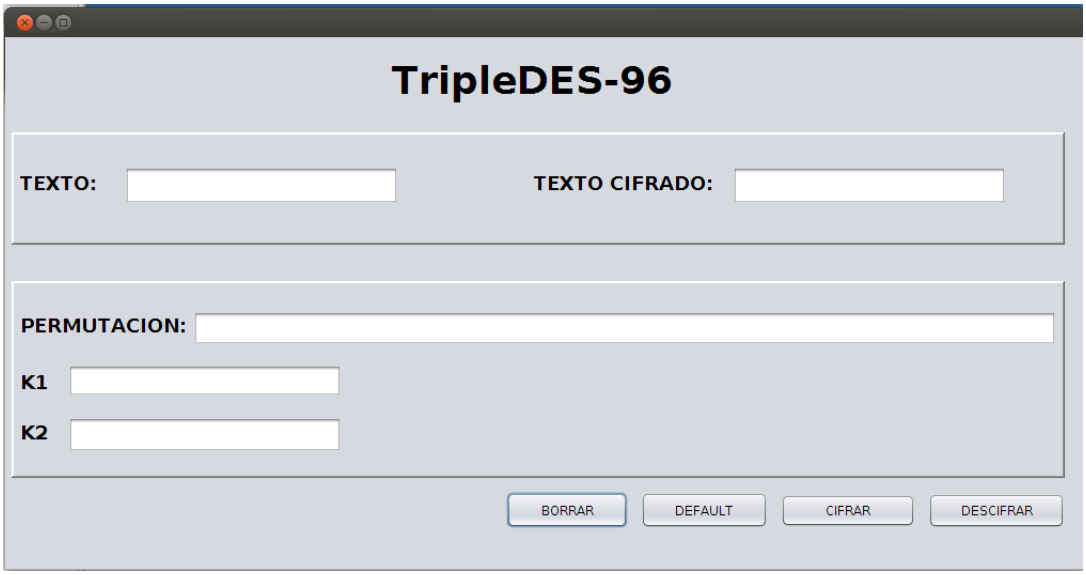

Figura 5.2: Pantalla Inicial

El bloque de 96 bits a cifrar es introducido en hexadecimal al igual que las llaves de 64 (56 utilizables) lo que nos da una longitud de 24 y 16 caracteres en hexadecimal respectivamente.

Figura 5.3: Datos de Entrada

Para esto es necesario proporcionar los datos de entrada que son Texto, K1, K2 y por default se proporciona un conjunto de decimales de e.

Una vez con los datos de entrada requeridos el primer paso para cifrar el bloque es generar las llaves con las que se llevaran a cabo las rondas del algoritmo DES96, debido a que TripleDES-96 utiliza tres ciclos de cifrado con dos llaves distintas, se generan por separado la llaves resultantes para K1 y K2 estas llaves son generadas de acuerdo a la norma establecida en FIPS 46-3.

```
Recibo k1:6516516550abcdef
La llave en binario es:011001010001011001010001011001010101000010101011110011011101111

El conjunto de llaves resultantes para Key1 es:
llave[0]:001010011000101001101111000100110100100111110100
llave[1]:111010011001011011011000110001000110101001010110
llave[2]:010101001101001011001110101101101101000101011000
llave[3]:001100101101100101010010101101100011001011001000011
llave[4]:001011000110100101100111000111101010011000100110
llave[5]:101000110110010100011101001111000110110111000100
llave[6]:010011010000011110110001001010001110000011010011
llave[7]:110101111001100010111001111001111111001000000011
llave[8]:100010001100100110101011001000111001101100011011
llave[9]:101100010010101100011111010101110001010100110010
llave[10]:001001010011111101000000101001101000010010110 1100
llave[11]:010100110011110011110100010000001111100011011100
llave[12]:110111001110010011010000011000011001010010111101
llave[13]:010101101100011100101110100010110001110010101011
llave[14]:111010101001000100000111000011100101101100110101
llave[15]:110110110011011000100011101111000100100000100010
```

Figura 5.4: Generación de llaves K1

Del mismo modo se hace para K2

```
Recibo k2:abcdef6546154915
La llave en binario es:101010111100110111101111011001010100011000010101010010010001 0101

El conjunto de llaves resultantes para Key2 es:
llave[0]:101101000110101010010110101000011110011000011100
llave[1]:001001000101111101000110100100110001111010001001
llave[2]:011000100111100101000101000110100011001100110101
llave[3]:000010011110010101110001001100110110100110100100
llave[4]:110001010100010110111011011000000010100110010011
llave[5]:111101111000001110000001111001110010000000011111
llave[6]:000110111001101010000011011001110001001111001010
llave[7]:001110010011000011011110000101001001000101101111
llave[8]:000000010111111010011100011111000100010101001100
llave[9]:010101000011110111010000010010001111000011001010
llave[10]:010101101110110001100001111001001111010000101001
llave[11]:110010111110010100000110101010100001111001101010
llave[12]:011010001000011110001111100111001101101100110010
llave[13]:011100011001000000101011000101010100111001110000
llave[14]:101000011000100011110010110110011010100001010000
llave[15]:100100001111001001011010110100011000001010111101
```

Figura 5.5: Generación de llaves K2

Una vez generadas las llaves que serán utilizadas en cada ronda a partir de las llaves del usuario utilizando el teorema JV y los decimales del número e junto con las claves dadas por el usuario K1 y K2 se genera una permutación variable.

Así mismo se genera una permutación inversa a esta.

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

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

Figura 5.6: Generación de permutaciones variables

Las permutaciones son procesadas y se genera un equivalente a nivel de bits.

```
La permutacion en bit es:
 |0|  |1|  |10|  |11|  |100|  |101|  |110|  |111|  |1000|
 |1001|  |1010|  |1011|  |1100|  |1101|  |1110|  |1111|  |10000|
 |10001|  |10010|  |10011|  |10100|  |10101|  |10110|  |10111|  |11000|
 |11001|  |11110|  |101000|  |1000110|  |1000000|  |110001|  |11101|  |101111|
 |1001110|  |100100|  |100111|  |1010100|  |111111|  |110011|  |1011101|  |101101|
 |1001011|  |111010|  |1011010|  |1010011|  |1011111|  |111011|  |100110|  |111110|
 |1000101|  |1001010|  |110110|  |100001|  |1000011|  |1001111|  |1010001|  |1010010|
 |1000001|  |111100|  |111001|  |11111|  |1010111|  |1011100|  |100010|  |1010000|
 |101010|  |111101|  |101100|  |1000010|  |110010|  |110111|  |1001000|  |110100|
 |1011000|  |100000|  |1010110|  |101011|  |110101|  |1000100|  |1001101|  |100011|
 |110000|  |11011|  |1011110|  |1001001|  |101110|  |1011001|  |111000|  |1000111|
 |100101|  |1010101|  |11100|  |11010|  |1001100|  |1011011|  |101001|

La permutacion inversa en bit es:
 |0|  |1|  |10|  |11|  |100|  |101|  |110|  |111|  |1000|
 |1001|  |1010|  |1011|  |1100|  |1101|  |1110|  |1111|  |10000|
 |10001|  |10010|  |10011|  |10100|  |10101|  |10110|  |10111|  |11000|
 |11001|  |1011100|  |1010010|  |1011011|  |11111|  |11010|  |111100|  |1001010|
 |110100|  |111111|  |1010000|  |100010|  |1011001|  |101111|  |100011|  |11011|
 |1011111|  |1000001|  |1001100|  |1000011|  |101000|  |1010101|  |100000|  |1010001|
 |11110|  |1000101|  |100110|  |1001000|  |1001101|  |110011|  |1000110|  |1010111|
 |111011|  |101010|  |101110|  |111010|  |1000010|  |110000|  |100101|  |11101|
 |111001|  |1000100|  |110101|  |1001110|  |110001|  |11100|  |1011000|  |1000111|
 |1010100|  |110010|  |101001|  |1011101|  |1001111|  |100001|  |110110|  |1000000|
 |110111|  |111000|  |101100|  |100100|  |1011010|  |1001011|  |111101|  |1001001|
 |1010110|  |101011|  |1011110|  |111110|  |100111|  |1010011|  |101101|
```

Figura 5.7: Permutación a nivel de bits

La clase principal es la encargada de cifrar y y descifrar adquiriendo los datos de entrada y utilizando tres ciclos de DES96

```
Los datos recibidos en principal son:

Texto:abcdef012345678912345678
Texto:1010101111001101111101111100000001001000110100010101100111100010010001001000110100010101100111100 0
Key1:6516516550abcdef
Key2:abcdef6546154915
llaveconcatenada:6516516550abcdefabcdef6546154915
```

Figura 5.8: Proceso Principal de cifrado

En el primer ciclo Cifra con K1 en el segundo Descifra con K2 y en el tercer y último Cifra con K1

```
Primera etapa con cifrado con permutacion
Se recibio este texto intermedio:1111010000011001000100100111100111001101010000100110010000001100100110011001001010111110010110
El texto cifrado intermedio es:f4191279cd42640c99995f2e

segunda etapa con cifrado sin permutacion
Se recibio este texto cifrado:0000111010001111111011011011001110100110111011010011101101001111011010110010001111100011100000111
El texto cifrado es:0e8fedb3a6ed3b4f6b23e387

Tercera etapa con cifrado con permutacion
Se recibio este texto final:0100010011001110101111110000101100110101110111001010100000000111000100111001110101010000000110000
El texto cifrado final es:44cebf0b35dca807139d5030
```

Figura 5.9: TripleDES-96

Finalmente se genera un bloque de salida de 24 caracteres en hexadecimal (96 bits) el cual corresponde al bloque cifrado de entrada.

Figura 5.10: Cifrado Triple DES-96

El descifrado es el proceso inverso del cifrado en el cual el orden de las claves es inverso y la permutación variable inversa (IP^{-1})es aplicada primero que la permutación variable (IP). Necesariamente las llaves del usuario tienen que ser la mismas, de lo contrario el bloque generado seria completamente diferente.

```
Recibo k1:6516516550abcdef
La llave en binario es:0110010100010110010100010110010101010000101010111100110111101111

El conjunto de llaves resultantes para Key1 es:
llave[0]:0010100110001010011011110001001101001001111110100
llave[1]:1110100110010110110110001100010001101010010110
llave[2]:0101010011010010110011101011010110100010110111000
llave[3]:0011001011011001010100101011000110010110010000011
llave[4]:0010110001101001011001110001111010100110001000110
llave[5]:1010001101100101000111010011111000110110111000100
llave[6]:0100110100000111101100010010100011100000011010011
llave[7]:1101011110011000101110011111001111111001000000011
llave[8]:1000100011001001101010110010001110011011000011011
llave[9]:1011000100101011000111110101011100010101001100110
llave[10]:0010010100111110100000010100110100000100101101100
llave[11]:0101001100111100111101000100000011111100011011100
llave[12]:1101110011100100110100000110000110010100101111101
llave[13]:0101011011000111001011101000101100011100101010111
llave[14]:1110101010010001000001110000111001011011001101101
llave[15]:1101101100110110001000111011110001001000001000010


Recibo k2:abcdef6546154915
La llave en binario es:1010101111001101111011110110010101000110000101010100100100010101

El conjunto de llaves resultantes para Key2 es:
llave[0]:1011010001101010100101101010000111100110000111100
llave[1]:0010010000101111101000110100100110001111010001001
llave[2]:0110001001111001010001010001101000110011001101101
llave[3]:0000100111100101011100010011001101101001101001000
llave[4]:1100010101000101101110110110000000010100110010011
llave[5]:1111011110000011100000011110011100100000000011111
llave[6]:0001101110011010100000110110011100010011111001010
llave[7]:0011100100110000110111100001010010010001011011111
llave[8]:0000000101111110100111000111110001000101010011100
llave[9]:0101010000111101110100000100100011110000011001010
llave[10]:0101011011101100011000011110010011110100001010001
llave[11]:1100101111100101000001101010101000011110011010010
llave[12]:0110100010000111100011111001110011011011001100010
llave[13]:0111000110010000001010110001010101001110011100000
llave[14]:1010000110001000111100101101100110101000010100000
llave[15]:1001000011110010010110101101000110000010101111101
```

Figura 5.11: Llaves de Descifrado

El proceso de descifrado de TripleDES-96 consta de tres ciclos en los cuales, en el primero se descifra con K1 en el segundo se cifra con K2 y el tercero y ultimo se descifra con K1.

```
Primera etapa con descifrado con permutacion
Se recibio este texto cifrado:0000111010001111111011011001110100111011101100111011010011110110111001000111111000111000011 1
El texto generado es:0e8fedb3a6ed3b4f6b23e387
Segunda etapa con cifrado sin permutacion
Se recibio este texto intermedio:1111010000011001000100100111100111001101010000100110010000001100100110011001100101011111001011 10
El texto intermedio es:f4191279cd42640c99995f2e
Tercera etapa con descifrado con permutacion
Se recibio este texto final:1010101111001101111011111000000010010001101000101011001111000100100010010001101000101011001111000
El texto cifrado final es:abcdef0123456789123456 78
```

Figura 5.12: Descifrado TripleDES-96

Finalmente el Bloque obtenido al aplicar el proceso de descifrado del bloque anteriormente cifrado con TripleDES-96 es el mismo, u original, cerrando así el ciclo de cifrado y descifrado.

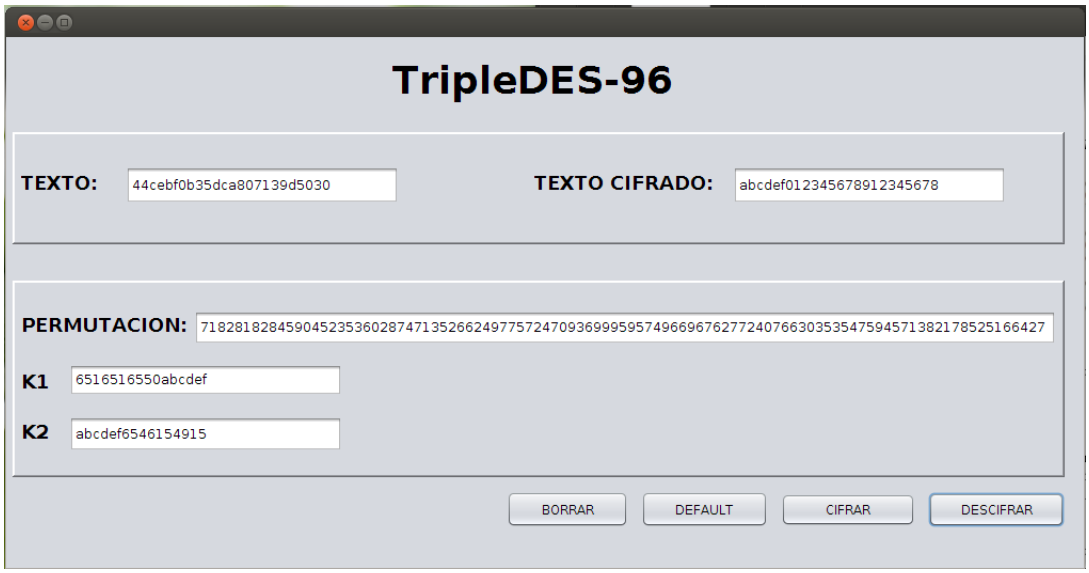

Figura 5.13: Pantalla de Descifrado TripleDES-96

6 Resultados

A partir de los programas desarrollados se procedió a hacer pruebas con el fin de constatar su funcionamiento y su velocidad.

Como se puede ver en la figura 6.1, la implementación de la norma ANSI X9.52 de TripleDES se realizo utilizando dos llaves proporcionadas por el usuario K1 y K2.

Figura 6.2: Datos de Entrada ANSI X9.52

De esta manera son ingresados los datos que son:

- Texto de entrada de 64 bits representados en 16 caracteres en hexadecimal
- Llave 1 de 64 bits representados en 16 caracteres en hexadecimal
- Llave 2 de 64 bits representados en 16 caracteres en hexadecimal

Figura 6.1: Pantalla principal ANSI X9.52

```
Los datos recibidos en principal son:
Texto:cabef12458736def
Texto:1100101010111110111100010010010001011000011100110110110111101111
Key1:6516516550abcdef
Key2:abcdef6546154915
```

Figura 6.3: Datos de entrada ANSI X9.52

Una vez con los datos ingresados en el sistema se procede a generar las llaves para las rondas de DES

```
Recibo k1:6516516550abcdef
La llave en binario es:011001010001011001010001011001010101000010101011110011011101111
El conjunto de llaves resultantes para Key1 es:
llave[0]:001010011000101001101111000100110100100111110100
llave[1]:111010011001011011011000110001000110101001010110
llave[2]:010101001101001011001110101101011010001011011000
llave[3]:001100101101100101010010101100011001011001000011
llave[4]:001011000110100101100111000111101010011000100110
llave[5]:101000110110010100011101001111000110110111000100
llave[6]:010011010000011110110001001010001110000011010011
llave[7]:110101111001100010111001111001111110010000000011
llave[8]:100010001100100110101011001000111001101100011011
llave[9]:101100010010101100011111010101110001010100110010
llave[10]:001001010011111010000001010011010000100101101100
llave[11]:010100110011110011110100010000001111100011011100
llave[12]:110111001110010011010000011000011001010010111101
llave[13]:010101101100011100101110100010110001110010101011
llave[14]:111010101001000100000111000011100101101100110101
llave[15]:110110110011011000100011101111000100100000100010
```

Figura 6.4: Generación de llaves K1 ANSI X9.52

```
Recibo k2:abcdef6546154915
La llave en binario es:101010111100110111101111011001010100011000010101010010010010101
El conjunto de llaves resultantes para Key2 es:
llave[0]:101101000110101010010110101000011110011000011100
llave[1]:001001000101111101000110100100110001111010001001
llave[2]:011000100111100101000101000110100011001100110101
llave[3]:000010011110010101110001001100110110100110100100
llave[4]:110001010100010110111011011000000010100110010011
llave[5]:111101111000001110000001111001110010000000011111
llave[6]:000110111001101010000011011001110001001111001010
llave[7]:001110010011000011011110000101001001000101101111
llave[8]:000000010111111010011100011111000100010101001100
llave[9]:010101000011110111010000010010001111000011001010
llave[10]:010101101110110001100001111001001111010000101001
llave[11]:110010111110010100000110101010100001111001101010
llave[12]:011010001000011110001111100111001101101100110010
llave[13]:011100011001000000101011000101010100111001110000
llave[14]:101000011000100011110010110110011010100001010000
llave[15]:100100001111001001011010110100011000001010111101
```

Figura 6.5: Generación de llaves K2 ANSI X9.52

Una vez que se a cifrado el bloque se procede a descifrar el bloque resultante, con el objeto de comprobar que el descifrado del bloque generado nos genere el bloque original

Figura 6.6: Cifrado en ANSI X9.52

Figura 6.7: Descifrado ANSI X9.52

Finalmente se puede mostrar el tiempo de ejecución tanto de la norma ANSI X9.52 como el de TripleDES-96

```
Se recibio este texto final:1110011000010001000110001100111111000000010101100011001101001101O
El texto cifrado final es:e61118cf80ac669a
Tiempo de ejecucion: 29
milisegundos
```

Figura 6.8: Tiempo de ejecución ANSI X9.52

```
Tercera etapa con cifrado con permutacion
Se recibio este texto final:010001001100111010111111000010110011010111011100101010000000011100010011100111010101000000110000
El texto cifrado final es:44cebf0b35dca807139d5030
Tiempo de ejecucion: 20
milisegundos
```

Figura 6.9: Tiempo de ejecución TripleDES-96

Con el fin de comprobar la mejoras en el criptosistema TripleDES-96 se ha desarrollado TripleDES de la norma ANSI X9.52 con las mismas funciones que fue desarrollado TripleDES-96, dado que TripleDES-96 trabaja con bloques de 96 bits y la norma ANSI X9.52 trabaja con 64 bits se generó una serie de archivos binarios, cuyo contenido son bits completamente aleatorios, dichos archivos tienen una propiedad, dado que los dos criptosistemas manejan tamaños de bloque diferente los archivos generados cumplen con las características de la siguiente tabla 6.1.

Archivo	Tanaño	Ciclos TripleDES-96	Ciclos ANSI X9.52
M768.dat	768kB	64000	96000
7M68.dat	7.68MB	640000	960000
76M8.dat	76.8MB	6400000	9600000

Cuadro 6.1: Archivos aleatorios

El tamaño de los archivos se calculó utilizando un común múltiplo entre los dos tamaños de bloque, que resulta del producto de 96*64, lo que nos da un tamaño de 6144 bits, representado en bytes es 768 bytes.

Este tamaño resulta muy pequeño para obtener resultados significativos así que se utilizaron múltiplos de 1000, 10000 y 100000 veces dándonos como resultado los archivos de la tabla 5.1.

Estos archivos se cifraron con los dos criptosistemas, con el fin de tomar el tiempo de ejecución de cada uno de los archivos.

El contexto en que se efectuaron las pruebas es el siguiente:

Computadora Laptop SAMSUNG modelo NP880Z5E con procesador Intel® Core™ i7 3635QM, memoria RAM 8GB, disco duro de 1TB y tarjeta gráfica AMD Radeon™ HD 8770M.

Las pruebas se realizaron en dos sistemas operativos distintos, ya que el sistema fue desarrollado en Java, no fue necesaria otra cosa mas que tener instalada la máquina virtual de Java en los distintos sistemas operativos.

Los sistemas operativos son los siguientes:

- Linux (ubuntu 13.10)
- Windows Pro 8.1

Los resultados después de realizar 10 pruebas en cada archivo y en cada sistema operativo pueden verse en la siguiente tabla:

Criptosistema	Tiempo de ejecución		
	M768.dat	7M68.dat	76M8.dat
TripleDES-96	1368	14860	132634
ANSI X9.52	2230	27641	284270

Cuadro 6.2: Comparación TripleDES-96 vs ANSI X9.52

Finalmente tenemos una tabla con las sumas vectoriales de los archivos generados al cifrar y descifrar, para poder así comprobar que el archivo regresa a su estado original, lo cual se puede ver en la tabla 6.3

Archivo	tamaño	Tamaño md5sum original	md5sum cifrado	md5sum Descifrado
M768.dat	768kB	724e92455c753954b626edbe9b849601	55140c9a5fb3234ec9d7f53d19a9484b	724e92455c753954b626edbe9b849601
7M68.dat	7.68MB	677926dca9938ce7ac513deecceaeab0	b21bdc74451c1b290d67c17cb9e29ab7	677926dca9938ce7ac513deecceaeab0
76M8.dat	76.8MB	f18bd71dd97b9d94a06ee9d2b797b2a7	d56bba56e92043769a479cf555b2aed7	f18bd71dd97b9d94a06ee9d2b797b2a7

Cuadro 6.3: Sumatoria Vectorial TripleDES-96

Archivo	tamaño	Tamaño md5sum original	md5sum cifrado	md5sum Descifrado
M768.dat	768kB	724e92455c753954b626edbe9b849601	8b277fb3b317eb2e3a73c7ee4f5cab58	724e92455c753954b626edbe9b849601
7M68.dat	7.68MB	677926dca9938ce7ac513deecceaeab0	754ffaa7e2e8b4265b7482d75e994005	677926dca9938ce7ac513deecceaeab0
76M8.dat	76.8MB	f18bd71dd97b9d94a06ee9d2b797b2a7	58d557f6d0ca6dfa9419ce142be349b4	f18bd71dd97b9d94a06ee9d2b797b2a7

Sumatoria Vectorial ANSI X9.52

6.1. Conclusión

En este trabajo de tesis se ha mostrado de manera específica la aplicación en los criptosistemas basados en DES, como lo es TripleDES, TripleDES-96.

El criptosistema TripleDES[1] fue mejorado ya 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 y al desconocerla, este tipo de procedimiento no puede llevarse a cabo. En general, es posible afirmar que utilizando a las permutaciones variables y un tamaño de bloque mayor se puede incrementar la complejidad computacional al mismo tiempo de que se hace mas rápido el cifrado, además de quitar la permutación P en cada ronda de DES. todo esto compatible y comparado con el de la norma aún vigente ANSI X9.52[1]. Esto nos demuestra que en promedio el criptosistema desarrollado TripleDES-96 es aproximadamente el doble de rápido que el de la norma ANSI X9.52, lo cual nos resulta en un criptosistema seguro ya probado por muchos años y ahora fortalecido con una permutación variable y más rápido.

6.2. Trabajo a futuro

1. Creación de sistemas criptográficos implementados en hardware de 96 bits con llaves que sigan la norma FIPS 46-3 de 64 bits Sistema de comunicaciones basado en cifrado de archivos utilizando TripleDES-96.
2. Implementación del criptosistema TripleDES-96 en hardware.

Referencias

[1] Ansi x9.52-1998, triple data encryption algorithm modes of operation.

[2] Fips pub 46-3, data encryption standard (des).

[3] Nist/fips-197, advanced encryption standard (aes).

[4] *Proceedings of the 2009 International Conference on Advances in Recent Technologies in Communication and Computing 27-28 October 2009, Kottayam, India*. IEEE Computer Society, Los Alamitos, Calif, 2009.

[5] Julia Allen. *The CERT guide to system and network security practices*. Addison-Wesley, Boston, 2001.

[6] William C. Barker and Elaine B. Barker. Sp 800-67 rev. 1. recommendation for the triple data encryption algorithm (tdea) block cipher. Technical report, Gaithersburg, MD, United States, 2012.

[7] Eli Biham. *Differential cryptanalysis of the data encryption standard*. Springer-Verlag, New York, 1993.

[8] Ernest Brickell. *Advances in cryptology : proceedings*. Springer, Berlin u.a, 1993.

[9] Aiden Bruen. *Cryptography, information theory, and error-correction : a handbook for the 21st century*. Wiley-Interscience, Hoboken, N.J, 2005.

[10] Tom Denis. *Cryptography for developers*. Syngress Pub, Rockland, MA, 2007.

[11] Dorothy Denning. *Proceedings of the 1st ACM conference on Computer and communications security*. ACM, New York, NY, 1993.

[12] Alexander Dent. *User's guide to cryptography and standards*. Artech House, Boston, MA, 2004.

[13] H. Feistel. Cryptography and computer privacy. *Scientific American*, 1973.

[14] Nick Goots. *Modern cryptography protect your data with fast block ciphers*. A-LIST, Wayne, Pa, 2003.

[15] Steven Holzner. *La biblia de Java 2*. Anaya Multimedia, Madrid, 2000.

[16] Jonathan Katz. *Introduction to modern cryptography*. Chapman & Hall/CRC, Boca Raton, 2008.

[17] Alan Konheim. *Computer security and cryptography*. Wiley-Interscience, Hoboken, N.J, 2007.

[18] Ariel Maiorano. *Criptografia : tecnicas de desarrollo para profesionales*. Alfaomega, Buenos Aires Mexico, D.F, 2009.

[19] Wenbo Mao. *Modern cryptography : theory and practice*. Prentice Hall PTR, Upper Saddle River, NJ, 2004.

[20] M. Matsui. Linear cryptanalysis method for des cipher. *In Advances in Cryptology*, 765:386–397, 1993.

[21] A. J. Menezes. *Handbook of applied cryptography*. CRC Press, Boca Raton, 1997.

[22] Richard Mollin. *An introduction to cryptography*. Chapman & Hall/CRC, Boca Raton, 2007.

[23] O Reilly. Cracking des. *Electronic Frontier Fondation*, 1998.

[24] Kenneth Rosen. *Discrete mathematics and its applications*. McGraw-Hill, Boston, 2007.

[25] Amparo Sabater. *Criptografia, proteccion de datos y aplicaciones : una guia para estudiantes y profesionales*. Ra-Ma, Paracuellos de Jarama, Madrid, 2012.

[26] Herbert Schildt. *Java 2 : manual de referencia*. Osborne/McGraw-Hill, Madrid, 2001.

[27] Herbert Schildt. *Java manual de referencia*. McGraw Hill, Mexico D.F, 2011.

[28] Bruce Schneier. *Applied cryptography : protocols, algorithms, and source code in C*. Wiley, New York, 1996.

[29] Victor Manuel Silva Garcia. Algorithm for strengthening some cryptographic systems. *Applied Mathematical Sciences, Vol. 4*, 2010.

[30] Victor Manuel Silva Garcia. Reducing computational complexity for 192 bits spn encryption algorithms with variable permutation. *JP Journal of Algebra, Numer Theory and Applications; PUSHPA Publishing House. ISSN 0972- 5555.*, 2010.

[31] Michael Spivak. *Calculo infinitesimal*. Reverte, Mexico Barcelona, 1996.

[32] Douglas Stinson. *Cryptography : theory and practice*. CRC Press, Boca Raton, 1995.

[33] Douglas Stinson. *Cryptography : theory and practice*. Chapman & Hall/CRC, Boca Raton, 2006.

[34] Thomas. *Introduccion a la programacion orientada a objetos con Java*. McGraw Hill, Madrid, 2001.

[35] Henk Tilborg. *Encyclopedia of cryptography and security*. Springer, New York, 2005.

Nomenclatura

Algoritmo	Detalle de pasos específicos, ordenados y finitos para la solución de un problema.
ANSI	(American National Standards Institute): Instituto Nacional de Estándares Americano. Organización o asociación privada y sin fines de lucro de los E.E.U.U. que produce estándares industriales. Es un miembro de la ISO y del IEC.
Bit	Digito binario; los valores posibles son 0 y 1.
Biyectiva	Asignación biyectiva, es un tipo de número irracional que no es raíz de ninguna ecuación algebraica con coeficientes enteros no todos nulos. En este sentido, número trascendente es antónimo de número algebraico. La definición no proviene de una simple relación algebraica, sino que se define como una propiedad fundamental de las matemáticas.
Bloque	Secuencia de bits de una longitud determinada. Por lo general, secuencias de mayor longitud son divididas en bloques de tamaño particular para la aplicación de diferentes procesos.
Byte	Conjunto de 8 bits. También llamado octeto. Un byte puede representar 256 valores diferentes o símbolos.
Criptoanálisis	Disciplina que estudia el quiebre o ruptura de un cifrador para recuperar información.
Criptografía	Ciencia que estudia el diseño de algoritmos para el cifrado y descifrado de información.
Diferencial	Criptoanalisis diferencial, técnica criptoanalitica de tipo estadístico,consiste en cifrar parejas de texto en claro escogidas con la condición de que su producto or-exclusivo obedezca a un patrón definido previamente. Los patrones de los correspondientes textos cifrados suministran información con la que se puede conjeturar la clave criptográfica.

Feistel	Clase especial de cifrado por bloques donde el texto-cifrado es computado a partir de texto plano, repitiendo la aplicación de la misma transformación.
FIPS	(Federal information Processing Standards): Estándares Federales de Procesamiento de la Información. Se trata de estándares anunciados públicamente desarrollados por el gobierno de los E.E.U.U. para que sean utilizados por las agencias de gobierno no militares y por los contratistas de gobierno. Los emite la NIST.
ISO	Organización Internacional de Normalización. Organismo encargado de promover el desarrollo de normas internacionales de fabricación (tanto de productos como de servicios), comercio y comunicación para todas las ramas industriales a excepción de la eléctrica y la electrónica.
NIST	El Instituto Nacional de Normas y Tecnología (NIST por sus siglas en inglés, National Institute of Standards and Technology), llamada entre 1901 y 1988 Oficina Nacional de Normas (NBS por sus siglas del inglés National Bureau of Standards), es una agencia de la Administración de Tecnología del Departamento de Comercio de los Estados Unidos. La misión de este instituto es promover la innovación y la competencia industrial en Estados Unidos mediante avances en metrología, normas y tecnología de forma que mejoren la estabilidad económica y la calidad de vida.
Normal	Número normal, es un número real cuyas cifras en cualquier base están distribuidas siguiendo una distribución uniforme, siendo todas las cifras igualmente probables, así como todos los pares, tríos, etc. Las cifras de ese número son tanto los de su parte entera como la sucesión infinita de dígitos que hay detrás de la coma o parte fraccionaria.
Teorema JV	Juan, Víctor. Iniciales de los autores del teorema.
Trascendental	Es un tipo de número irracional que no es raíz de ninguna ecuación algebraica con coeficientes enteros no todos nulos. En este sentido, número trascendente es antónimo de número algebraico. La definición no proviene de una simple relación algebraica, sino que se define como una propiedad fundamental de las matemáticas.
Triple DES	Algoritmo de cifrado simétrico por bloques. Basado en DES, en su implementación aplica tres veces el algoritmo original.
Xor	Operador binario. Dados los dos bits operados, la operación resultara en 1 si los valores son diferentes y en 0 si son iguales. Es una abreviación de la OR exclusiva.