UNIVERSIDAD AUTÓNOMA METROPOLITANA

Casa abierta al tiempo

DESCRIPCIÓN POLINOMIAL DE LOS SISTEMAS DE CIFRADO DES Y AES

Tesis que para obtener el grado de Maestro en Ciencias
presenta:

Paulo Sergio García Méndez

Director de tesis: Dr. Horacio Tapia Recillas

Diciembre de 2011, México, D.F.

Contenido

Introducción **1**

1 Fundamentos de Criptografía **7**
1.1 Perspectiva histórica 7
1.2 Terminología 13
1.3 Criptografía de llave privada 17
1.3.1 Esquemas de cifrado por flujo 18
1.3.2 Esquemas de cifrado por bloque 20

2 Sistema de cifrado DES: Data Encryption Standard **25**
2.1 Antecedentes 25
2.2 Descripción original del sistema **DES** 27
2.2.1 Permutación inicial **IP** y permutación final $\mathbf{IP^{-1}}$ 28
2.2.2 Permutación **E** 30
2.2.3 Permutación **P** 31
2.2.4 Las cajas de substitución de DES 31
2.2.5 La función tipo Feistel de DES 33
2.2.6 Las rondas del estándar DES 33
2.2.7 Permutación $\mathbf{PC_1}$ y permutación $\mathbf{PC_2}$ 34
2.2.8 Generación de las subllaves 35
2.3 Cifrado y descifrado con DES 36
2.3.1 Ejemplo 37

3 Descripción polinomial del sistema de cifrado DES **45**
3.1 Sistema de cifrado DES 45
3.1.1 Permutación inicial **IP** y permutación final $\mathbf{IP^{-1}}$ 47
3.1.2 Permutación **E** 48
3.1.3 Permutación **P** 49

3.1.4 Las cajas de substitución de DES 53
3.1.5 La función tipo Feistel de DES 55
3.1.6 Las rondas del estándar DES 57
3.1.7 Permutación $\mathbf{PC_1}$ 58
3.1.8 Permutación $\mathbf{PC_2}$ 63
3.1.9 Generación de las subllaves 69
3.2 Cifrado y descifrado con DES 73
3.2.1 Ejemplo 74

4 Sistema de cifrado AES: Advanced Encryption Standard 77
4.1 Antecedentes 77
4.2 Descripción original del sistema AES 79
4.2.1 Transformación SubBytes 80
4.2.2 Transformación ShiftRows 82
4.2.3 Transformación MixColumns 83
4.2.4 Transformación AddRoundKey 84
4.2.5 Las rondas de AES 84
4.2.6 Expansión y selección de la llave 84
4.3 Cifrado y descifrado con AES 87
4.3.1 Ejemplo 90

5 Descripción polinomial del sistema de cifrado AES 93
5.1 Sistema de cifrado AES 93
5.1.1 Transformación SubBytes 95
5.1.2 Transformación ShiftRows 99
5.1.3 Transformación MixColumns 99
5.1.4 Transformación AddRoundKey 99
5.1.5 Las rondas de AES 100
5.1.6 Expansión y selección de la llave 100
5.2 Cifrado y descrifrado con AES 100
5.2.1 Ejemplo 101

Conclusiones 106

Apéndice A 113

Bibliografía 126

Introducción

Actualmente se vive en una época donde el intercambio de información por medios electrónicos como internet, correo electrónico, telefonía celular, comercio electrónico, tarjetas inteligentes, entre otros, resulta cotidiano.

Cuando se almacena o transmite información valiosa o secreta por medios convencionales, a menudo resulta insuficiente, inaplicable y costoso protegerla sólo de manera física. Por lo que se deben emplear otras técnicas más apropiadas y eficientes; la *Criptografía* se ha desarrollado como un conjunto de técnicas para solucionar este tipo de situaciones.

La palabra Criptografía proviene de las raíces griegas *kryptos* que significa esconder y *graphein* que quiere decir escritura. Sus inicios datan de las civilizaciones más antiguas como los egipcios, los hindúes, los mesopotámicos, entre otros. Cada una de estas civilizaciones ha desarrollado algún tipo de criptografía en su momento. También, las guerras han servido como fuente de inspiración para desarrollar sistemas de cifrado, en particular, la primera y segunda guerra mundial son momentos históricos llenos de anécdotas que tienen que ver con la comunicación secreta.

La Criptografía estudia y aplica técnicas y principios con los que la información se vuelve incomprensible para cualquiera excepto para aquella entidad a quién va dirigida. Las formas más sencillas y más antiguas de transformar un mensaje son: la *substitución* que consiste en usar un conjunto de símbolos fijo en vez de otro y la *permutación* que implica intercambiar la posición de un conjunto de símbolos. Estas técnicas se siguen aplicando al diseñar sistemas de cifrado modernos, en particular, los llamados de *llave privada*.

Los sistemas de cifrado se clasifican en sistemas de *llave pública* y en sistemas de llave privada. Los sistemas de llave pública requieren de dos piezas de informa-

ción, una pública y una privada, para lograr una comunicación segura. Algunos de los esquemas de cifrado de llave pública más famosos son RSA, Logaritmo Discreto, McEliece, ElGamal y Curvas Elípticas. Los esquemas de cifrado de llave privada usan la misma llave secreta tanto para el cifrado como para el descifrado. Dentro de los sistemas de cifrado modernos de este tipo, destacan IDEA, Serpent, DES, AES, entre otros.

El sistema de cifrado conocido como DES (Data Encryption Standard) ha resultado de enorme importancia para el desarrollo y difusión de la Criptografía moderna. En 1973 la Oficina Nacional de Estándares (National Bureau of Standards: NBS) de los Estados Unidos publicó una solicitud para recibir propuestas de algoritmos de cifrado para proteger información durante su transmisión y almacenamiento [27]. El sistema de cifrado DES (Data Encryption Standard), desarrollado por un equipo de la corporación IBM alrededor de 1974, respondió a este llamado y fue adoptado como estándar en 1977. El sistema DES fue uno de los primeros en desarrollarse comercialmente cuya estructura fue completamente publicada. Después de ser sometido a intensos escrutinios por una amplia comunidad de investigadores y sobrevivir por varios años se hizo evidente la necesidad de mejorarlo e incluso reemplazarlo.

En enero de 1997 el Instituto Nacional de Estándares y Tecnología (National Institute of Standards and Technology NIST antes NBS) de los Estados Unidos convocó a un concurso internacional para elegir un nuevo estándar de cifrado y en el año 2001 anunció que el sistema de cifrado Rijndael se convertiría en el nuevo estándar de cifrado (Advanced Encryption Standard, AES), diseñado por Vincent Rijmen y Joan Daemen, una de sus características más notables es que las operaciones criptográficas pueden describirse en términos del campo finito $GF(2^8)$.

Antecedentes

El desarrollo de DES y de la Criptografía de llave pública a mediados de los años 70 atrajo el interés de la comunidad científica en la Criptografía. Algo que, previamente, sólo había sido del dominio de las agencias de inteligencia. Desde el momento del anuncio de DES como el estándar de cifrado hubo muestras de preocupación respecto al tamaño de la llave y otras con respecto a las cajas de substitución. Esto provocó el estudio tanto de los componentes del sistema como de la estructura de DES.

Don Coppersmith y Edna Grossman, en [8], al igual que Shimon Even y Oded Goldreich, en [13], obtuvieron algunos resultados sobre las funciones tipo DES. Demostraron que el grupo de permutaciones generado por las funciones tipo DES es el grupo alternante. En 1988, Burton Kaliski, Ronald Rivest y Alan Sherman se preguntaron, en [16], si el conjunto de permutaciones que se podía definir para una función de encripción de DES y una llave dada, $E_k : \{0,1\}^{64} \to \{0,1\}^{64}$, era cerrado bajo la composición. Usando un par de estrategias para buscar ciclos obtuvieron evidencia estadística de que DES no es un cerrado bajo la composición. También, Keith Campbell y Michael Wiener, en [6], proporcionaron evidencia para afirmar con alto grado de certidumbre que DES no es un grupo pequeño. Si la respuesta hubiese sido afirmativa, esto habría implicado dos debilidades en el sistema. La primera habría sido que la encripción múltiple era equivalente a una encripción simple. Mientras que la segunda habría exhibido que DES era vulnerable ante un ataque por texto plano conocido que se ejecutaría en 2^{28} pasos en promedio. Cerca de los años 90, Eli Biham y Adi Shamir introdujeron el criptoanálisis diferencial [2] y, en 1993, Mitsuru Matsui usó el criptoanálisis lineal sobre DES [18]. Aunque estos dos tipos de criptoanálisis no resultaban prácticos en aquel momento para romper DES, sí fueron tomados en cuenta al momento de diseñar nuevos criptosistemas.

Hacia finales de los años 90, se hizo evidente que DES no permanecería como una solución efectiva a los problemas de seguridad en el almacenamiento y transmisión de información en el largo plazo. Esto se debió principalmente al incremento en la velocidad de los chips y a la construcción de máquinas destinadas a encontrar la llave de cifrado de DES que cada vez lo hacían en tiempos más cortos. Por ello, el NIST anunció el proceso para escoger el nuevo estándar de cifrado AES en 1997.

Daemen y Rijmen diseñaron Rijndael combinando ideas de criptosistemas previos y pensando en que este nuevo esquema fuera resistente a los ataques conocidos, exhibiera buen desempeño en una variedad de plataformas y fuera sencillo. Rijndael posee una estructura algebraica en el campo $GF(2^8)$ y operaciones criptográficas sencillas de describir desde el punto de vista algebraico. Sin embargo, al ser elegido como el nuevo estándar de cifrado AES, esta simplicidad algebraica generó preocupación por parte de algunos investigadores. A finales de 2001, Niels Ferguson, Richard Schroeppel y Doug Whiting proporcionaron, en [14], una descripción de este sistema mediante fracciones continuas en $GF(2^8)$ concluyendo que sería computacionalmente difícil tratar de solucionar dichas ecuaciones. Poco des-

pués, Nicolas Courtois y Josef Pieprzyk, en [10], describieron Rijndael como un sistemas de ecuaciones polinomiales multivariado. Ellos observaron que la S-caja de AES puede expresarse mediante ecuaciones cuadráticas booleanas. Posteriormente, Sean Murphy y Matt Robshaw encontraron una manera de incrustar AES en un sistema más grande en el que, excepto por la función inversa de la caja de substitución, el resto eran operaciones lineales en $GF(2^8)$ [7]. El sistema se conoce como BES (Big Encryption System) y lo que resulta de describir AES como un caso particular de este sistema es que recuperar una llave de AES equivale a resolver un sistema de ecuaciones cuadráticas multivariado extremadamente ralo. Sin embargo, el sistema generado resulta bastante complicado de resolver.

De entre varias descripciones que existen en la literatura de AES, Joachim Rosenthal ofrece con gran detalle una versión polinomial en [26]. Una duda natural que surge es si se puede intentar una representación polinomial como la de AES con otros esquemas de cifrado. Por la importancia histórica, un primer candidato a examinar es DES. En la mayoría de la bibliografía revisada se encontraron trabajos que apuntan hacia la estructura de grupo de DES, esta es otra razón por la que sería interesante explorar la idea de una representación polinomial con otra estructura, por ejemplo, usando campos finitos y anillos de polinomios tal y como se hace en [26] para el caso del AES. En [30], Takeshi Shimoyama y Toshinobu Kaneko presentan expresiones cuadráticas de las cajas de subtitución de DES y las usan para mejorar un ataque lineal de la versión completa de DES. Más recientemente, Nicolas Courtois y Gregory Bard en [11] describen métodos para expresar DES como un sistema de ecuaciones afirmando que es mejor intentar resolver un sistema de ecuaciones de grado lo más bajo posible, a pesar de que el número de ecuaciones crezca. Entonces surge la pregunta de cómo será la forma de las cajas de subtitución usando, por ejemplo, polinomios de permutación.

Objetivo

El objetivo general de este trabajo es dar una descripción algebraica del esquema de cifrado DES para apreciar una posible estructura algebraica del mismo más amplia que la estructura de grupo que otros autores han presentado [8, 13, 16, 6]. Para lograrlo, se aplicarán resultados y conceptos básicos del Álgebra Conmutativa y la Teoría de Campos Finitos así como polinomios de permutación.

Objetivos Particulares

En particular, nos preguntamos si es posible encontrar una descripción polinomial de las tablas de permutación así como de las cajas de substitución. También, se llevará a cabo la implementación de los algoritmos de cifrado y de descifrado en el sistema de cómputo simbólico *Mathematica v5.0* con la intención de apreciar las operaciones entre polinomios.

Organización

El presente trabajo se encuentra organizado de la siguiente manera: En el capítulo I, se proporcionan las definiciones básicas de la Criptografía. El objetivo es presentar de manera detallada los conceptos de la Criptografía de llave simétrica. El capítulo II se centra en presentar los detalles de la especificación original del sistema de cifrado DES. Se describen las tablas de permutación, las cajas de substitución, la función de Feistel para, finalmente, ilustrar su uso mediante un ejemplo.

La descripción algebraica del sistema de cifrado DES se explica en el capítulo III. Este capítulo representa la principal aportación del trabajo, en éste se exponen las descripciones polinomiales de las cajas de substitución y tablas de permutación, así como la descripción del algoritmo de cifrado en términos de polinomios. Estos resultados se obtuvieron al aplicar algunas de las técnicas usadas en la despcripción polinomial que Joachim Rosenthal [26] hace del sitema de cifrado AES.

La especificación original y la descripción polinomial del sistema de cifrado AES se revisan con mayor detalle en los capítulos IV y V, respectivamente. Se repasan brevemente algunos antecedentes históricos del sistema, para dar paso a una inspección de su estructura básica, las operaciones criptográficas que utiliza y la programación de sus llaves. Además, su uso se ilustra mediante un ejemplo. Después, en el capítulo V se vuelven a revisar los puntos anteriores pero mediante un enfoque completamente polinomial. Se introducen las estructuras algebraicas necesarias y las operaciones polinomiales que sirven para representar las operaciones criptográficas del capítulo IV. Finalmente, se presentan las conclusiones del trabajo y las rutinas desarrolladas en *Mathematica v5.0* que implementan los algoritmos de cifrado y descifrado en términos polinomiales tanto del DES como del AES.

Capítulo 1
Fundamentos de Criptografía

En este capítulo se introducen los conceptos básicos de la Criptografía con el objetivo de revisar con mayor detalle los conceptos de la Criptografía de llave privada. Primero, se incluye una breve semblanza histórica de lo que ha sido la evolución de la Criptografía a través del tiempo pues la historia de la Criptografía es tan extensa que resultaría imposible cubrir todos los hechos. Aquí sólo se mencionarán algunos acontecimientos recopilados de [15] y de [32]. En la siguiente sección, se proporciona la terminología que permitirá definir lo que se conoce como criptosistema. Finalmente, se presentan los conceptos que se manejan en los sistemas de cifrado de llave privada modernos.

1.1 Perspectiva histórica

Con la invención de la escritura, se originó el deseo de transmitir información de manera rápida y segura. Fue así como surgieron los primeros mecanismos para esconder o disfrazar la información. Por miles de años, reyes, reinas y generales se han apoyado en una comunicación secreta para gobernar sus países y dirigir sus ejércitos. Pero al mismo tiempo, han tenido que estar al tanto de las consecuencias de que sus mensajes caigan en manos equivocadas, revelando secretos valiosos a naciones rivales o fuerzas opositoras. La amenaza de una posible intercepción enemiga ha motivado el desarrollo de técnicas para disfrazar o esconder los mensajes con la intención de que sólo el receptor pueda entenderlos.

Uno de los primeros registros que se tienen sobre el uso de la comunicación secreta data de los tiempos de Herodoto, historiador romano que relató los conflictos entre Grecia y Persia en el siglo V a.c. De acuerdo a él, fué el arte de la escritura secreta lo que salvó a Grecia de ser conquistada por Xerxes, lider de los persas.

Cuando éste último decidió construir Persépolis como la nueva gran capital de su reino, recibió regalos y tributos de todas partes del imperio y estados vecinos con la notable excepción de Atenas y Esparta. Esto provocó el enojo de Xerxes y, como respuesta a tal insolencia, comenzó a formar un gran ejército con la intención de invadir Grecia. Sin embargo, Demarato, un griego expulsado de su tierra natal y quién vivía en la ciudad persa de Susa, utilizó la estrategia de esconder el mensaje en placas de madera plegables cubiertas de cera para alertar a los griegos de las intenciones de Xerxes. Con esto, el ataque de Xerxes perdió sorpresa y, al final, las extraordinarias fuerzas persas fueron derrotadas y humilladas.

La comunicación en secreto que se logra al esconder la existencia de un mensaje se conoce como *Esteganografía*. Dicha palabra proviene del griego *steganos* que significa cubrir y *graphein* que quiere decir escritura. En paralelo al desarrollo de la esteganografía, se encuentra la evolución de la *Criptografía*, palabra derivada del griego *kryptos* que significa esconder. El objetivo de la Criptografía no es esconder la existencia de un mensaje, sino esconder su significado. La ventaja de la Criptografía sobre la esteganografía es que, con esta última, la intercepción del mensajero puede comprometer inmediatamente la seguridad de la comunicación; en cambio, si un mensaje esconde su verdadero significado y cae en manos enemigas, el mensaje permanece incomprensible. En contraparte, un mensaje encriptado podría suscitar sospecha de un enemigo, mientras que si el mensaje se escondiese mediante métodos esteganográficos éste podría pasar desapercibido. Estas dos formas de comunicación secreta pueden combinarse para mejorar la seguridad de la comunicación. Por ejemplo, es muy común que la gente envíe imágenes por correo electrónico, entonces se puede insertar un mensaje de texto dentro de una imagen usándola como cubierta. En [3], se describe un método con estas características.

Con el fin de que la comunicación sea efectiva, se requiere de un sistema que tanto el que envía el mensaje (*emisor*) como el que lo recibe (*receptor*) conozcan pero que permanezca desconocido ante cualquier posible enemigo. Este sistema, denominado *sistema o esquema de cifrado*, o bien, *criptosistema*, consiste generalmente de un método para esconder (*cifrar*) la información, conocido como *algoritmo de cifrado*, y una pieza de información adicional, llamada *llave*, que proporciona detalles de un cifrado particular. Además, el alfabeto (*alfabeto plano*) usado para escribir el mensaje original no necesariamente es el mismo utilizado para escribir el mensaje cifrado *alfabeto del cifrado*.

Una primera clasificación de la Criptografía es la división en Criptografía de *substi-*

tución y Criptografía de *transposición*. En la Criptografía de transposición, las letras de un mensaje se reacomodan generando un anagrama que aumenta el número de posibles combinaciones a medida que se incrementa el número de letras en el mensaje. La Criptografía de substitución cambia cada letra en el texto plano por una letra diferente. Esto significa que cada letra cambia su significado como tal pero mantiene su posición.

Un ejemplo de Criptografía de substitución se da con el esquema de cifrado usado por el emperador romano Julio César, quién solía reemplazar cada letra en el mensaje con la letra que aparece a tres lugares de distancia de la primera. Por ejemplo, después de recorrer las letras del alfabeto español 3 posiciones hacia la izquierda, la asignación queda como sigue:

mensaje	A	B	C	D	E	F	G	H	I	J	K	L	M	N
criptograma	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P

mensaje	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
criptograma	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Con lo que el mensaje: VINE VI Y VENCI queda cifrado como: YLPH YL B YHPFL.

El *cifrado de Vigenère*, atribuido de manera equivocada a Blaise de Vigenère, es una generalización del esquema de cifrado tipo César. Su fortaleza radica en que no usa uno sino varios alfabetos; en el caso del alfabeto español, se usarían 27 alfabetos distintos para cifrar. Estos alfabetos se generan al hacer el desplazmiento a la izquierda de una letra del alfabeto anterior, el alfabeto inicial es el alfabeto español. Aunque originalmente este sistema fue descrito por Giovan Batista Bellaso en 1553, fue Blaise de Vigenère quién publicó una descripción más segura del sistema en 1586. Los cifrados de substitución tradicionales, aquellos que existieron antes del de Vigenère se conocen como cifrados de substitución monoalfabéticos. En contraste, el cifrado tipo Vigenère pertenece a la clase de los polialfabéticos porque emplea varios alfabetos para cifrar un sólo mensaje.

Otra de las actividades del hombre donde la Criptografía se hace presente es el desciframiento de los lenguajes usados por antiguas civilizaciones. Por siglos, los jeroglíficos epipcios permanecieron como un misterio y los arqueólogos sólo podían especular su significado. A finales del siglo XVIII, uno de los primeros hombres interesados en descifrar los jerogíficos egipcios fue un notable inglés de nombre Thomas Young quién a la edad de 14 años ya había estudiado griego, latín, francés,

italiano, hebreo, siriaco, samaritano, árabe, persa y turco. Young sentó las bases para el desciframiento de los jeroglíficos egipcios al identificar un buen número de éstos, algunos tan importantes como el símbolo usado para denotar la terminación femenína que se ponía después de los nombres de reinas y diosas.

México no es ajeno al uso de la Criptografía. Algunas de las anécdotas más conocidas datan del porfiriato y de la revolución mexicana [1]. El General Porfirio Díaz usó métodos criptográficos para comunicarse con gobernadores y jefes militares importantes. Estos esquemas criptográficos eran básicamente de substitución, es decir, se reemplazaba una letra por un número entero. Al parecer una particularidad añadida al esquema de Díaz era la posibilidad de asignar tres enteros de dos dígitos a cada letra. Con esto intentaba dificultar algún análisis de frecuencias. Por otro lado, los hermanos Flores Magón tuvieron que hacer uso de métodos criptográficos de substitución al ser perseguidos por promover sus ideas en contra del régimen de Porfirio Díaz y ser considerados como una amenaza a la estabilidad política. También, Francisco I. Madero usaba sistemas criptográficos de substitución simple de diferentes tipos. Algunos eran muy similares a los de Díaz, pero en otros solía reemplazar las letras por otras en lugar de hacerlo con números. El sistema criptográfico mexicano más conocido es el llamado *Mexican Army Cipher Disk* [15]. Fue usado en la época de la revolución y consistía de 4 alfaberos numéricos dispuestos en 4 discos. Desafortunadamente, el ejército estadounidense interceptó y descifró muchos mensajes como producto de la creciente fricción entre ambos países.

Finalmente, una de las historias que involucra a México en la Primera Guerra Mundial es la del famoso Telegrama de Zimmerman, ver [4] y [15]. En aquella época, el presidente de los Estados Unidos Woodrow Wilson se rehusaba a participar en la guerra, pero en 1915, un submarino alemán atacó el transatlántico Lusitania. Esto hubiese significado la entrada de los estadounidenses a la guerra, pero el nuevo ministro de relaciones exteriores alemán, Arthur Zimmerman, convenció a los Estados Unidos de no hacerlo asegurándoles que a partir de entonces sus submarinos saldrían a la superficie. Sin embargo, esto sólo fue una táctica para distraer a los Estados Unidos. Zimmerman propuso una alianza con México intentando persuadir al presidente mexicano, Venustiano Carranza, de invadir a los Estados Unidos con el objetivo de reclamar territorios perdidos como Texas, Nuevo México y Arizona. Al mismo tiempo, el presidente de México actuaría como mediador para convencer a Japón de atacar a los estadounidenses por el pacífico. Así, los norteamericanos no tendrían forma de enviar tropas a Europa. Zimmerman

envió su propuesta mediante un telegrama cifrado al embajador en Washington para que, a su vez, retransmitiera el mensaje a su homólogo en México y, éste se lo comunicase al Presidente. El telegrama fue interceptado y descifrado por los británicos quienes, en algún tiempo después, se lo dieron a los estadounidenses provocando su entrada en la guerra.

Con respecto a otros conflictos bélicos, hay que mencionar que una de las claves para que los aliados pudiesen ganar la Segunda Guerra Mundial fue la invención de la máquina *Bomba*, diseñada para descifrar los mensajes de los alemanes cifrados con la famosa máquina *Enigma*. Enseguida, se describe brevemente la historia de tales máquinas recaudada de [32].

La invención de la máquina de escribir eléctrica estableció los medios para producir máquinas electromecánicas de cifrado. En 1915, Edward Hebern usó dos máquinas de escribir eléctricas conectadas aleatoriamente por 26 cables. Las conecciones proporcionaron las ideas básicas para crear un rotor y, en 1918, el alemán Arthur Scherbius obtuvo la patente de una máquina de cifrado a la que bautizó con el nombre de Enigma. Los japoneses compraron la máquina Enigma para su uso en 1934 y desarrollaron su propio sistema de cifrado, el cual era conocido con el nombre de *Purple* por los estadounidenses. Este sistema fue criptoanalizado en 1940 por el *Signal Intelligence Service* de los Estados Unidos.

Por otro lado, los criptógrafos alemanes también tomaron la decisión de adoptar a Enigma como su máquina de cifrado. El sistema de cifrado alemán fue criptoanalizado por los investigadores de la *Code and Cipher School* en Bletchley Park, Buckinghamshire, Inglaterra. Uno de los más importantes investigadores presentes fue Alan Turing. A principios de 1940, Turing indentificó las debilidades de Enigma y rediseñó la máquina Bomba para descifrar los mensajes encriptados con Enigma. Estos hechos fueron relevantes para lograr la victoria de los aliados.

Después de la guerra, la computadora jugó un papel crucial en el desarrollo de la Criptografía ya que con ella los criptoanalistas podían explotar su velocidad y flexibilidad para probar todas las posibles llaves hasta encontrar la correcta. En contraste, los criptógrafos comenzaron a explotar el poder de las computadoras para incrementar la complejidad de los esquemas de cifrado.

En principio, el cifrado mediante computadoras estaba restringido al gobierno y el ejército. A partir de la invención del transistor, en 1947, la computación comercial

se hizo una realidad y con la llegada del circuito integrado en 1959 comenzó una nueva era en la computación. Durante los años 60, las computadoras incrementaron su poder de cómputo al tiempo que se abarataron. Las empresas tuvieron la capacidad económica para adquirirlas y usarlas para cifrar información importante tales como transferencias bancarias o negociaciones comerciales delicadas. A medida que el uso de la Criptografía creció, el tema de la estandarización se volvió una preocupación primaria. En Estados Unidos, el National Bureau of Standards (NBS) lanzó una convocatoria para solicitar propuestas de esquemas de cifrado que permitieran una comunicación secreta en el mundo empresarial. El resultado fue un sistema de cifrado llamado DES (Data Encryption Standard) que permaneció vigente por más de 25 años.

Desafortunadamente, a pesar de la estandarización y fortaleza que ofrecía DES, las compañías se toparon con una situación más complicada, el problema de distribuir la llave. Esto es, cómo dos entidades se ponen de acuerdo en la llave sin que una tercera entidad se entere de ello. La solución llegó en 1976, cuando Whitfield Diffie, Martin Hellman y Ralph Merkle dieron a conocer su esquema para el intercambio de llave, creando con esto, un nuevo tipo de Criptografía, la *Criptografía de llave pública*. Dos años más tarde, se anuncia el primer sistema de cifrado de llave pública, llamado RSA, cuyas letras son las iniciales de los apellidos de sus creadores: Ron Rivest, Adi Shamir y Leonard Adleman. La seguridad del sistema radica en la dificultad computacional de factorizar un número entero.

Desde entonces, se han propuesto numerosos sistemas de cifrado de llave pública. Algunos de éstos son ElGamal, basado en el problema del logaritmo discreto que puede usarse tanto para cifrado como para firma digital; McEliece, basado en códigos detectores-correctores de errores; esquema de Rabin, cuya seguridad está dada por la dificultad de encontrar raices cuadradas módulo un número compuesto. En 1985, Neal Koblitz y V. S. Miller propusieron, de manera independiente, el uso de Curvas Elípticas sobre Campos Finitos en el desarrollo de esquemas de cifrado de llave pública. Aunque no inventaron un nuevo esquema de cifrado, sí implementaron algoritmos de llave pública existentes usando Curvas Elípticas.

Debido a que muchos de estos esquemas basan su seguridad en problemas computacionalmente difíciles, resultan imprácticos para cifrar mucha información como la llave e incluso llegan a ser más lentos que los criptosistemas simétricos. Por lo que se usan en tarjetas inteligentes o en dispositivos móviles para distribuir la llave y después usar un esquema simétrico para cifrar la información.

Para este tiempo, los rápidos avances en la velocidad de cómputo hicieron que el esquema DES fuera objeto de constantes intentos para encontrar la llave correcta probando todas las posibles llaves. Estos intentos dieron frutos cuando en julio de 1998 la Electronic Frontier Foundation (EFF) anunció que había podido descifrar una encripción hecha con DES usando una máquina construida para esa tarea con menos de 250,000 dolares. Esto se logró en menos de tres días. Previamente, en 1997 el National Institute of Standards and Technology (NIST) había abierto una nueva convocatoria con el fin recibir nuevas propuestas de esquemas de cifrado que pudieran reemplazar al sistema DES. El resultado final se dió en el año 2000 con la elección del criptosistema Rijndael como el nuevo estándar de cifrado llamado Advanced Encryption Standard (AES). Este criptosistema sorprendió nuevamente a la comunidad criptográfica debido al uso de un campo finito para definir sus operaciones criptográficas.

Un nuevo paradigma en la computación que puede revolucionar la teoría y práctica de la misma es la computación cuántica. El impacto que una computadora cuántica pudiera tener en la Criptografía recae directamente en los criptosistemas de llave pública, en particular, en la seguridad de RSA y de curvas elípticas. Esto ha impulsado el desarrollo de nuevos esquemas de cifrado que puedan garantizar seguridad en las comunicaciones ante una eventual llegada de las computadoras cuánticas dando origen a lo que se conoce como *Criptografía cuántica*.

A grandes rasgos se han abordado algunos de los hechos históricos más sobresalientes en la evolución de la Criptografía. Dentro de la investigación en el área, es notable el uso de varias ramas de las matemáticas para crear nuevos y más sofisticados esquemas de cifrado, así como para mejorar las técnicas del Criptoanálisis. De manera conjunta, la comercialización de la Criptografía contribuye a su desarrollo al incorporar nuevas tecnologías en los servicios. Más información sobre estas y otras notas históricas pueden consultarse en [15] y en [32].

1.2 Terminología

Para comenzar, supóngase la siguiente situación: una entidad A, conocida como *emisor*, desea comunicarse con otra entidad B, llamada *receptor*, de manera segura a través de un canal que, por sus condiciones físicas, se considera inseguro. Esto es, A debe asegurarse que ningún *intruso* (cualquier entidad distinta a B) entienda

el mensaje que envía a través del canal inseguro.

El primer problema que presenta esta situación es cómo hacer que aunque un intruso intercepte el mensaje y éste lo vea, al final, no pueda reconocer su contenido. Esto se conoce como *privacidad*. Luego, una vez que B ha recibido el mensaje surge el problema de cómo asegurarse que fue A quién efectivamente envió el mensaje. A esto se le llama *autenticidad*. También, el receptor podría preguntarse si el mensaje no ha sido modificado en su contenido. Este problema se conoce como *integridad*. Finalmente, si A negara la autoría de un mensaje enviado a B, y éste último supiera cómo prevenirlo, entonces llevaría a cabo lo que se conoce como *no-repudio*.

La Criptografía representa un conjunto de técnicas orientadas a la solución de los problemas generados al enviar información a través de un canal inseguro. Mientras que el *Criptoanálisis* es el conjunto de métodos que intentan debilitar alguna de las metas de seguridad que se persigue en la Criptografía. Estas dos ramas son el objeto de estudio de lo que se conoce como *Criptología*.

El *cifrado* es el conjunto de operaciones (criptográficas) usadas para hacer que la información que se envía sea incomprensible para cualquiera excepto para el receptor. Este proceso toma un fragmento de información conocido como *texto plano* y lo transforma en un *texto cifrado* o *criptograma* usando un fragmento de información adicional llamado *llave para cifrar*. El *descifrado* es el proceso inverso al cifrado. El receptor recupera el mensaje (texto plano) mediante este proceso a partir del texto cifrado y el uso de la *llave para descifrar*.

El cifrado y descifrado forman, en conjunto, un *esquema de cifrado* o *criptosistema*, ver Figura 1.1.

De manera más formal, se tiene la siguiente definición tomada de [5].

Definición 1.2.1. *Sea Σ un alfabeto. Un esquema de cifrado o criptosistema es una quintupla $(\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ donde:*

- *$\mathcal{M}$, conocido como el espacio de mensajes, es el conjunto de posibles mensajes que se pueden formar con los caracteres del alfabeto Σ.*
- *$\mathcal{C}$, el espacio de mensajes cifrados, es el conjunto de todos los posibles mensajes cifrados.*

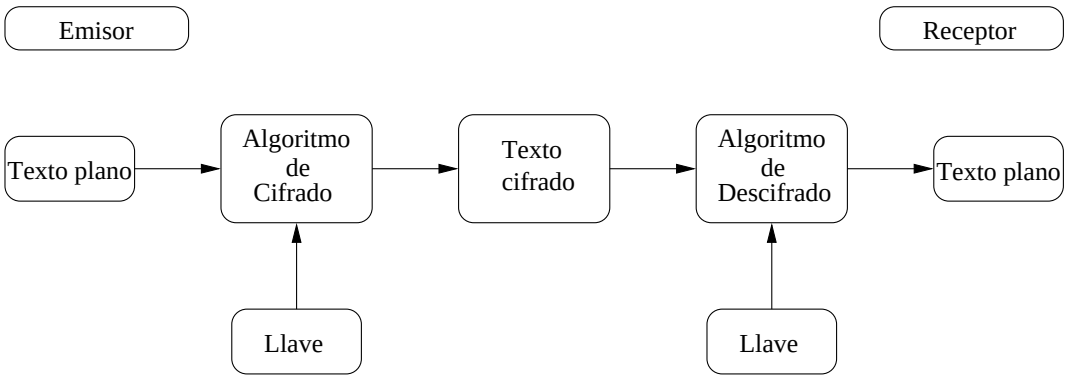

Figura 1.1: Esquema de cifrado.

- $\mathcal{K}$, *el espacio de llaves, es el conjunto de todas las posibles llaves.*
- $\mathcal{E} = \{E_k : k \in \mathcal{K}\}$ *es una familia de funciones* $E_k : \mathcal{M} \to \mathcal{C}$ *en la que cada elemento se llama función de cifrado.*
- $\mathcal{D} = \{D_k : k \in \mathcal{K}\}$ *es una familia de funciones* $D_k : \mathcal{C} \to \mathcal{M}$ *en la que cada elemento se llama función de descifrado.*
- *Para cada* $e \in \mathcal{K}$, *existe* $d \in \mathcal{K}$ *tal que* $D_d(E_e(m)) = m$ *para toda* $m \in \mathcal{M}$.

El primer ejemplo con el que se ilustra la definición anterior es el famoso *cifrado tipo César*. Como se mencionó en la sección anterior, el emperador romano recorría las letras del alfabeto cierto número de posiciones hacia la izquierda respetando el orden en el que aparecían en el alfabeto. Una manera más conveniente de usar este cifrado es asignar a cada letra del alfabeto un número entero según su posición en el alfabeto.

Ejemplo 1.2.1. ***Cifrado tipo César.*** *Supongamos que a cada letra del alfabeto español se le asigna un número entero según su posición en el alfabeto. Por lo que el espacio de mensajes es* $\mathcal{M} = \{0, 1, ..., 26\}$, *el espacio de mensajes cifrados o criptogramas es* $\mathcal{C} = \{0, 1, ..., 26\}$ *y el espacio de llaves es* $\mathcal{K} = \{0, 1, ..., 26\}$. *Por último, las funciones de cifrado y de descifrado quedan como sigue:*

$$\begin{aligned} E_k(m) &= m + k \mod 27 \qquad k \in \mathcal{K} \\ D_k(c) &= c - k \mod 27 \qquad k \in \mathcal{K} \end{aligned}$$

<table>
<tr><td colspan="5">Criptografía</td></tr>
<tr><td colspan="2">Criptografía Clásica</td><td colspan="3">Criptografía Moderna</td></tr>
<tr><td>Esquemas de Substitución</td><td>Esquemas de Transposición</td><td colspan="2">Esquemas de Llave Privada</td><td>Esquemas de Llave Pública</td></tr>
<tr><td></td><td></td><td>Esquemas de Cifrado por Bloque</td><td>Esquemas de Cifrado por Flujo</td><td></td></tr>
</table>

Tabla 1.1: Clasificación de la Criptografía

En este esquema sólo hay 27 posibles llaves por lo que resulta sencillo encontrar el texto plano a partir del texto cifrado probando todas las posibles llaves y revisando con cuál de ellas el texto recuperado adquiere sentido. También, en este caso la llave para descifrar es $d = e = k$. Esto no necesariamente ocurre para otros criptosistemas.

Existen dos categorías dentro de la Criptografía: la *Criptografía de llave privada* o *simétrica* y la *Criptografía de llave pública* o *asimétrica*, ver Tabla 1.1.

Los criptosistemas de llave privada o simétricos son esquemas donde la llave para cifrar puede encontrarse a partir de la llave para descifrar y viceversa. En la mayoría de estos, tanto la llave de cifrado como la llave para descifrar es la misma. Esto requiere que tanto el emisor como el receptor se pongan de acuerdo en la llave antes de iniciar la comunicación que desean asegurar. Los esquemas de llave privada pueden dividirse en dos categorías. Los que operan sobre un bit a la vez del texto plano se conocen como *esquemas de cifrado por flujo*. Los otros esquemas operan sobre grupos de bits (*bloques*) y son llamados *esquemas de cifrado por bloque*. En los criptosistemas de llave pública o asimétricos se usan diferentes llaves en el cifrado y en el descifrado; calcular la llave para descifrar a partir de la llave de cifrado resulta prácticamente imposible. La llave para cifrar, conocida como *llave pública*, puede usarse por cualquiera para cifrar un mensaje, pero sólo la persona que posee la llave para descifrar correspondiente, llamada *llave privada*, puede recuperar el mensaje.

Las *firmas digitales* son una aplicación del uso de los esquemas de cifrado de llave pública. En este caso, los mensajes se cifran con la llave privada y se descifran usando la llave pública. Esto permite verificar la identidad del emisor tal y como lo hace una firma autógrafa.

Por otro lado, el criptanálisis intenta recuperar el texto plano de un mensaje sin conocer la llave, pero suponiendo que el intruso puede acceder a la comunicación a partir del texto cifrado entre un emisor y su receptor. Cuando el Criptoanálisis es efectivo, el intruso recupera el texto plano o la llave para cifrar. Un *ataque* es un intento de Criptoanálisis, el cual se traduce en un algoritmo que permite recuperar elementos protegidos de un esquema de cifrado suponiendo que sólo se conoce el criptosistema que se está usando. El atacante trata de recuperar un mensaje a partir de textos cifrados que intercepta o las llaves que fueron utlizadas. Cuando el atacante intenta encontrar la llave usada probando todas las posibles llaves, esto se conoce como *ataque mediante búsqueda exhaustiva.*

Los esquemas de cifrado pueden ser criptoanalizados mediante los siguientes ataques típicos:

- *Ataque por mensaje cifrado.* El atacante conoce algunos textos cifrados y, a partir de ellos, trata de recuperar los correspondientes mensajes o la llave usada.
- *Ataque por mensaje conocido.* El atacante conoce un mensaje y el texto cifrado correspondiente ó incluso, varios pares de estos, y con ellos trata de encontrar la llave usada ó de descifrar otros textos cifrados.
- *Ataque por mensaje elegido.* El atacante es capaz de cifrar algunos mensajes pero no conoce la llave. Es el ataque por el cuál el atacante puede elegir algunos mensajes y hacer que se cifren, de tal manera que cuando obtiene los criptogramas correspondientes puede recuperar la mayor información posible acerca de la llave usada.
- *Ataque por mensaje cifrado elegido.* El atacante puede elegir sus propios criptogramas y hacer que se descifren sin conocer la llave.

1.3 Criptografía de llave privada

En 1949, Claude E. Shannon publicó su famosos artículo *Communication Theory of Secrecy Systems* [29]. En él, hace un estudio sobre la estructura matemática y propiedades que debe tener un criptosistema. También, formalizó dos conceptos básicos para criptosistemas de llave privada. El primero es la *dispersión*, en el que cada bit del texto plano debe influir en la configuración de muchos bits del texto cifrado, de tal manera que las propiedades estadísticas del texto plano no se

revelen en el texto cifrado. El otro es la *confusión*, en la cual la redundancia en las propiedades estadísticas del texto plano se disipan en las propiedades estadísticas del texto cifrado. Esto es, se refiere a la dependencia de los bits de la salida con respecto a los bits de la entrada. Shannon propuso que al combinar diferentes tipos de esquemas de cifrado sencillos e inseguros se podían crear criptosistemas complejos y seguros. Estos esquemas se conocen como *esquemas tipo producto*. Los detalles de estas ideas pueden consultarse en [29].

Dentro de los esquemas de cifrado de llave privada se distinguen dos categorías: los *esquemas de cifrado por flujo* y los *esquemas de cifrado por bloque*. Las funciones de cifrado por flujo cifran caracteres individuales de un mensaje, uno a la vez, mientras que las funciones de cifrado por bloque procesan fragmentos de texto plano de longitud fija.

1.3.1 Esquemas de cifrado por flujo

Como se ha mencionado anteriormente, los esquemas de cifrado por flujo convierten texto claro en texto cifrado, cifrando caracter por caracter. Se puede decir que antes de la llegada de las computadoras, los esquemas de cifrado actuaban generalmente de esa manera. Por ejemplo, el cifrado de conversaciones telefónicas requiere de este tipo de cifrado ya que el flujo de datos en tiempo real se produce en pequeños fragmentos.

Definición 1.3.1. *Sea $\mathcal{K}$ el espacio de llaves y $\mathcal{M}$ el espacio de mensajes. También, denotaremos como $\mathcal{M}^*, \mathcal{C}^*$ y $\mathcal{K}^*$ a los conjuntos de todas las palabras sobre $\mathcal{M}$, $\mathcal{C}$ y $\mathcal{K}$, respectivamente. Un cifrado por flujo es un criptosistema que procesa un flujo o cascada $m := m_1 m_2 m_3 ... \in \mathcal{M}^*$ de caracteres $m_i \in \mathcal{M}$ como un flujo o cascada $c := c_1 c_2 c_3 ... \in \mathcal{C}^*$ de textos cifrados $c_i \in \mathcal{C}$ usando la cascada de llaves $k := k_1 k_2 k_3 ... \in \mathcal{K}^*$ donde $k_i \in \mathcal{K}$ y la función de cifrado*

$$E_k : \mathcal{M} \to \mathcal{C}$$

para cada k tal que $c_i := E_{k_i}(m_i)$ para cada i.

Típicamente, los caracteres en $\mathcal{M}$ y las llaves en $\mathcal{K}$ son números binarios o bytes. El más famoso ejemplo de un esquema de cifrado por flujo es el *cifrado Vernam*, llamado así en honor a su inventor, Gilbert Vernam.

Ejemplo 1.3.1. *Sea $\mathcal{M} = \mathcal{K} = \mathcal{C} = \{0, 1\}$ y considérense las funciones de cifrado*

y descifrado dadas por:

$$\begin{aligned} E_{k_i}(m_i) &= m_i \oplus k_i \\ D_{k_i}(c_i) &= c_i \oplus k_i \end{aligned}$$

donde $\oplus$ *denota la operación de bits* ***O exclusivo*** *(`XOR`).*

Claramente, el cifrado y descifrado son operaciones inversas. Cada bit de la llave sólo se usa para el cifrado de un mensaje m. Si dicha llave se usara para cifrar los mensajes m y $\overline{m}$, entonces se podría calcular $m \oplus \overline{m}$ a partir de los criptogramas c y $\overline{c}$ y así, obtener información acerca de los mensajes.

Si k se elige de manera aleatoria e independiente, entonces el cifrado se conoce como *libreta de un sólo uso* (one-time pad). Un problema que surge con este último cifrado es que el tamaño de la llave debe ser tan grande como el tamaño del texto plano. Esto dificulta tanto el manejo como la distribución de la llave lo que, a su vez, motiva el diseño de esquemas de cifrado por flujo donde la llave debe generarse de manera pseudoaleatoria a partir de una llave secreta más pequeña con la intención de que dicha llave parezca aleatoria ante algún intruso.

En la práctica, se usan *generadores pseudoaleatotorios* los cuales son algoritmos determinísticos que generan bits a partir de una pequeña semilla aleatoria.

Los esquemas de cifrado por flujo se clasifican comúnmente en *sincronizados* y de *auto-sincronización*. En un esquema sincronizado la llave se genera de manera independiente al texto plano y al texto cifrado, mientras que en un esquema de auto-sincronización la llave se genera como una función de la llave pequeña y de un número fijo de dígitos del texto cifrado.

Como ejemplo del primer esquema, está el conocido como *cifrado por flujo aditivo* el cual aplica la operación $\oplus$ al texto plano y la llave. Esto requiere el uso de un generador de llaves en cascada.

El segundo esquema puede ilustarse describiendo una función recursiva de la llave secreta k que genere la cascada de llaves. Por ejemplo, dado el alfabeto $\Sigma = \{0, 1\}$ y un vector $k = (k_1, k_2, k_3, k_4)$ con $k_i \in \Sigma$, se define la siguiente recursión:

$$\begin{aligned} z_i &= k_i && \text{para } 1 \leq i \leq 4, \\ z_i &= z_{i-3} \oplus z_{i-4} && \text{para } 4 < i. \end{aligned}$$

Muchos de los generadores en cascada de llaves usan un *registro de desplazmiento hacia atrás* como su componente básico, en particular, los *registros de desplazmiento hacia atrás lineales*. Estos últimos son bastante adecuados para implementarse en hardware, pueden producir sucesiones de bits con periodo muy grande y poseen buenas propiedades estadísticas y estructura algebraica.

1.3.2 Esquemas de cifrado por bloque

Los esquemas de cifrado por bloque son versátiles porque permiten la construcción de generadores de números pseudoaleatorios, sirven como componente principal en técnicas de autenticación de mensajes, mecanismos de integridad de información y esquemas de firma digital, entre otros.

En un esquema de cifrado por bloque, el texto plano se divide en fragmentos de longitud fija, esto es, bloques. Cada bloque se convierte en un bloque de texto cifrado de la misma longitud por medio de una substituciones que dependen de una llave.

Definición 1.3.2. *Sea Σ un alfabeto. Un esquema de cifrado por bloque es un criptosistema con $\mathcal{M} = \mathcal{C} = \Sigma^n$ donde n es la longitud del bloque. El espacio de llaves $\mathcal{K}$ es el conjunto de todas las permutaciones de Σ^n, denotado por $\mathcal{S}(\Sigma^n)$. La función de cifrado para una llave fija $k \in \mathcal{S}(\Sigma^n)$ es*

$$E_k : \Sigma^n \to \Sigma^n$$

Obsérvese que el espacio de llaves de este esquema puede ser muy grande ya que contiene $(|\Sigma|^n)!$ elementos. Esto ilustra el argumento de Shannon acerca de los espacios de llaves y de texto plano grandes para obtener esquemas seguros.

Ejemplo 1.3.2. ***Esquema de cifrado DES.*** *Sea $\Sigma = \{0, 1\}$. Defínanse $\mathcal{M} = \mathcal{C} = \Sigma^{64}$ y $\mathcal{K} = \Sigma^{56}$. Para $k = (k_1, ..., k_{16})$ con $k_i \in \mathcal{K}$ se tienen la siguientes funciones de cifrado y de descifrado.*

$$\begin{aligned} E_k &= F_{k_{16}} \circ \cdots \circ F_{k_1} \\ D_k &= F_{k_1}^{-1} \circ \cdots \circ F_{k_{16}}^{-1}. \end{aligned}$$

La función F_{k_j} se explicará a detalle en el siguiente capítulo.

Modos de operación de los esquemas de cifrado por bloque

Estos modos son distintas maneras de llevar a cabo el cifrado de documentos arbitrariamente grandes. A menudo, combinan el esquema de cifrado, y algunas operaciones que deben ser sencillas porque la seguridad es una función del esquema de cifrado y no del modo.

Modo ECB (Electronic Code Book)

El texto plano se divide en bloques de tamaño n y se encripta produciendo un bloque de texto cifrado del mismo tamaño. Teóricamente, sería posible crear un libro con los textos cifrados para cada bloque de texto plano correspondiente, pero a medida que crece el tamaño del bloque, el tamaño del libro aumenta demasiado. El problema con este modo es que si un criptoanalista obtiene el texto cifrado para un texto plano dado que aparezca en varios mensajes, éste puede comenzar a guardarlos e identificar estos bloques en otros mensajes. Por esto, no se recomienda para mensajes más grandes que el tamaño del bloque.

Los modos CBC, CFB y OFB que se describen enseguida necesitan, en su paso inicial, un bloque de información adicional al texto plano llamado vector de inicialización VI. Este vector se genera por cada cifrado y se requiere del mismo vector para la desencripción. Por lo tanto, el vector VI no necesita ser secreto pero tanto emisor como receptor deben conocer la forma de calcularlo. Para mayor información sobre cómo generar tales vectores de inicialización puede consultarse [22].

Modo CBC (Cipher Block Chaining)

En este modo, el cifrado de un bloque no sólo depende de la llave sino de los bloques previos, lo que añade un mecanismo de retroalimentación. Cada bloque se usa para modificar la encripción del siguiente bloque. Esto hace que cada bloque de texto cifrado sea dependiente de todos los bloques de texto plano previos.

En el modo CBC, el bloque de texto plano se opera mediante una suma `XOR` con el bloque de texto cifrado anterior, y enseguida, se encripta. En el caso del primer bloque de texto plano, se necesita un vector de inicialización VI, el cual no tiene que ser secreto pero sí impredecible. Después, el texto cifrado resultante se almacena dentro de un registro para la siguiente encripción. Así, el cifrado de cada bloque depende de todos los bloques anteriores. Para desencriptar un bloque

de texto cifrado, éste se desencripta normalmente y se opera mediante una suma XOR con el bloque descifrado previamente. El modo CBC se ilustra en la Figura 1.2.

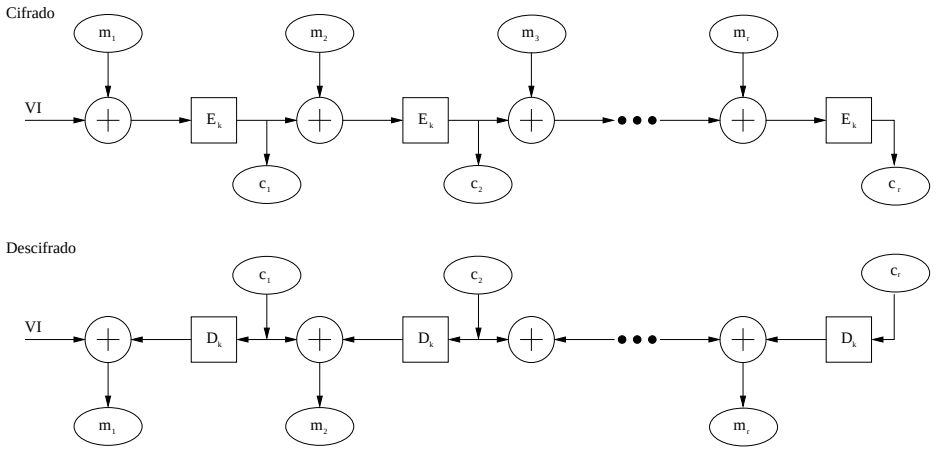

Figura 1.2: Modo CBC.

Modo CFB (Cipher Feedback)

Con los modos anteriores, la encripción no comienza hasta que se tiene un bloque completo de información. Usando el modo CFB, se puede cifrar información en unidades más pequeñas que el tamaño del bloque. Cuando se combina con un registro electrónico, puede usarse para encriptar un bit o byte a la vez. En este modo de cifrado, el primer bloque de entrada es un vector de inicialización al que se le aplica el algoritmo de encripción para producir el primer bloque de salida. Entonces el primer bloque cifrado se genera con una suma XOR entre la primera porción de texto plano y el primer bloque de salida. El proceso se repite con los bloques de entrada siguientes hasta que se ha producido un texto cifrado a partir de todas las porciones de texto plano anteriores. Para el descifrado con el modo CFB, el vector VI es el primer bloque de entrada y cada bloque de entrada siguiente se forma como en el cifrado. La Figura 1.3 describe el modo CFB.

Modo OFB (Output Feedback)

Este modo puede usarse para aplicaciones donde desea evitarse el error de propagación. Es similar al modo CFB y permite el cifrado de varios tamaños en los

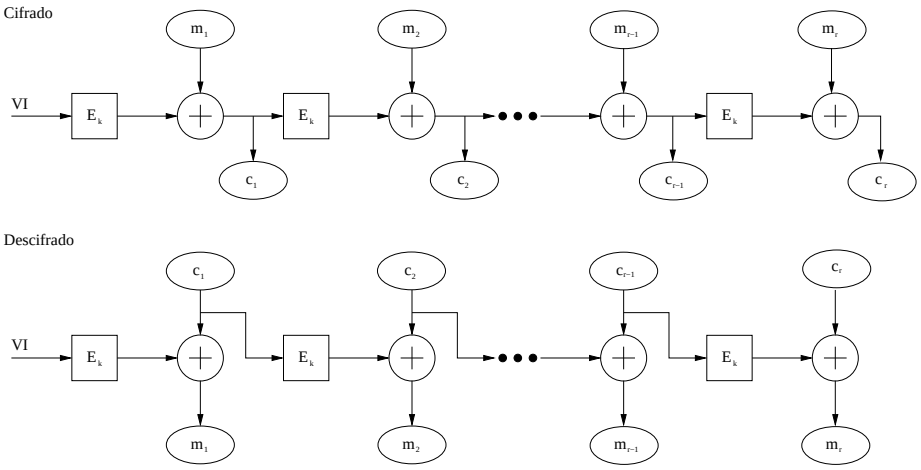

Figura 1.3: Modo CFB.

bloques. En el cifrado con el modo OFB, el vector VI se transforma mediante función de encripción para producir el primer bloque de salida. Después, este bloque se suma al primer bloque de texto plano para generar el primer bloque de texto cifrado. Nuevamente, se aplica la función de encripción al primer bloque de salida creando el segundo bloque de salida. Para formar el segundo bloque de texto cifrado, se debe sumar el segundo bloque de salida con el segundo bloque de texto plano. Así, los bloques de salida siguientes se originan aplicando la función de cifrado al bloque de salida anterior, es decir, no dependen de bloques de texto cifrado anteriores. El descifrado en el modo OFB es similar al cifrado, sólo que se suman los bloques de texto cifrado para recuperar los bloques de texto plano. El modo OFB necesita un único vector VI para cada mensaje que se va a cifrar con una llave dada. De lo contrario, se puede comprometer la confidencialidad de los mensajes que se cifraron usando el mismo vector VI. El modo OFB se ilustra en la Figura 1.4.

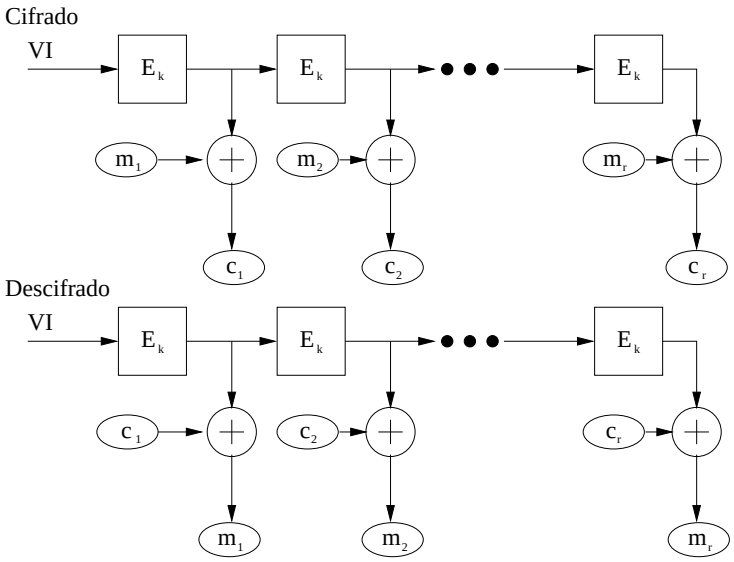

Figura 1.4: Modo OFB.

Capítulo 2
Sistema de cifrado DES: Data Encryption Standard

En este capítulo se introduce el sistema de cifrado DES. Primero, se proporcionan algunos antecedentes históricos referentes al sistema. Después, se presenta su especificación original. Finalmente, el uso de sus componentes tales como tablas de permutación, cajas de substitución y función de Feistel se ilustran mediante un ejemplo.

2.1 Antecedentes

El sistema de cifrado DES ha sido probablemente el esquema de cifrado simétrico moderno más usado y conocido en el mundo. Estuvo vigente por más de 20 años y ha impulsado como ninguno el campo del criptoanálisis ya que nunca antes se había contado con un sistema totalmente abierto al público para su análisis.

A principios de los años 70, dentro del sector comercial de los Estados Unidos se hacía evidente la necesidad de proteger los secretos corporativos. Un claro ejemplo lo ilustra la comunicación entre una máquina (ATM: Automatic Teller Machine) y una computadora central. El usuario inserta una tarjeta magnética y solicita una cantidad de dinero, la máquina ATM envía la solicitud a la computadora y ésta verifica si existen fondos enviando un mensaje en el que autoriza o niega a la máquina ATM otorgar el dinero. Evidentemente, si la información no se protege, un intruso podría intervenir la comunicación e interceptar el mensaje de autorización enviando varias copias del mensaje a la máquina ATM para realizar varias transacciones y obtener más dinero.

Por esa época, un grupo bancario le pidió a la corporación IBM que desarrollara un sistema para cifrar la información de las máquinas ATM. Entonces se formó un equipo de investigadores en dos sedes de IBM (Kingston y Yorktown Heights, Nueva York) para desarrollar el sistema. Algunos de los integrantes del equipo fueron: Roy Adler, Don Coppersmith, Horst Feistel, Edna Grossman, Alan Konheim, Carl Meyer, Bill Notz, Lynn Smith, Walter Tuchman y Bryant Tuckerman.

En mayo de 1973, la oficina de estándares de los Estados Unidos (National Bureau of Standards: NBS) publicó una convocatoria solicitando propuestas de algoritmos de cifrado para proteger la información durante su transmisión o almacenamiento. Algunos de los criterios de diseño que pedían eran los siguientes:

- La seguridad del algoritmo debía residir en la llave y no en mantener el algoritmo en secreto.
- El algoritmo debía ser implementable en dispositivos electrónicos de forma sencilla.
- El algoritmo debía ser eficiente.
- El algoritmo debía ser exportable.

En esa convocatoria no hubo propuestas que cumplieran con los criterios anteriores. Sin embargo, en una segunda convocatoria en 1974 el equipo de IBM registró su sistema, conocido internamente como *LUCIFER*. El algoritmo fue evaluado en forma conjunta y secreta por la Agencia de Seguridad Nacional (NSA: National Security Agency) y por la NBS. Después de algunas modificaciones a las funciones internas del algoritmo y de reducir el tamaño de la llave de 112 a 56 bits, la NBS adoptó el algoritmo como estándar de cifrado a finales de 1976 y lo publicó a principios de 1977 identificándolo con el nombre DES (Data Encryption Standard).

El uso de DES fue obligatorio para todas las transacciones financieras del gobierno estadounidense que concernieran la transferencia electrónica de fondos, incluyendo aquellas dirigidas por bancos miembros del sistema de reserva federal. La adopción de DES por organizaciones de estándares a nivel mundial provocó que éste se convirtiera en el estándar de facto internacional para proporcionar seguridad en el manejo de información comercial y de negocios.

Uno de los hechos más controvertidos fue la intervención de la NSA en la revisión del algoritmo. Mucha gente creía que la NSA había modificado el algoritmo para

instalar una puerta trasera. Se decía que la agencia era la responsable de haber reducido el tamaño original de las llaves de 112 a 56 bits. Sin embargo, a la fecha esto aún no ha sido totalmente aclarado.

Lo que está fuera de duda, es que nunca antes se había publicado un algoritmo evaluado por una agencia de seguridad nacional. Esto impulsó el desarrollo del criptoanálisis como nunca antes.

En 1977, se creía que el esfuerzo para encontrar una llave dentro de 2^{56} posibles era una tarea más que imposible. Sin embargo, en 1997 se mostró que era posible recuperar la llave de DES mediante una búsqueda exhaustiva usando el poder de cómputo de las computadoras conectadas a Internet. En 1998, también se comprobó lo anterior mediante una máquina especializada construida por la (EFF: Electronic Frontier Foundation) con un costo menor a $250,000.00 dólares. Finalmente, en 1999 mediante una máquina especializada combinada con 100,000 PC's en Internet se montó un ataque por búsqueda exhaustiva que tuvo éxito en tan sólo 22 horas. Con la evidencia de que DES ya no era seguro, se diseño una variante del mismo conocida como Triple DES o TDES que usa 2 llaves normales de DES. Walter Tuchman propuso que la operación de cifrado fuera de la forma $E_1D_2E_1$, esto es, cifrar con la primera llave, descifrar con la segunda y volver a cifrar con la primer llave. Mientras que para el descifrado propuso la forma $D_1E_2D_1$ que siginifica descifrar con la primera llave, cifrar con la segunda y volver a descifrar con la primera. Esto es posible gracias a la propiedad de que cifrar y descifrar con la misma llave es lo mismo que descifrar y cifrar con la misma llave. Aunque existen varias maneras de hacer este tipo de combinaciones, Tuchman sugirió estos ya que si las llaves son la misma, el cifrado se convierte en un simple DES. De tal manera que los equipos electrónicos con TDES pudieran interoperar con equipos que sólo contaban con DES como su algoritmo de cifrado. Los estándares usados por los bancos adoptaron este esquema. Los datos expuestos en esta sección fueron consultados en [27].

2.2 Descripción original del sistema DES

Como se especifica en [20], el algoritmo DES es un sistema de cifrado por bloque tipo producto que opera sobre bloques de información de 64 bits de longitud produciendo un bloque de texto cifrado del mismo tamaño. La longitud de la llave es de 56 bits, pero comunmente se expresa como un bloque de 64 bits de los cuales

cada octavo bit se usa para verificar la paridad, estos son los *bits de chequeo de paridad* que finalmente se eliminarán.

En un nivel básico, el algoritmo DES resulta ser la combinación de dos conceptos básicos del cifrado: la confusión y dispersión. El bloque fundamental de DES es una sóla combinación de estos dos conceptos, es decir, una substitución seguida de una permutación.

A grandes rasgos, el sistema DES comienza con una permutación inicial sobre un bloque de texto plano de 64 bits de longitud. Después, el bloque permutado se divide en mitades de 32 bits cada una para aplicar 16 rondas (iteraciones) de una función f tipo Feistel que mezcla la información con la llave original. Los detalles de esta función se proporcionan en la sección 2.2.7. Al término de la décimo sexta ronda, las mitades se juntan nuevamente para formar un sólo bloque al que se le aplica una permutación final. Esta permutación finaliza el algoritmo. La estructura general del algoritmo se puede ver en la Figura 2.1.

Los bits de la llave original se recorren cierto número de posiciones para formar 16 nuevas llaves, llamadas *subllaves*, de 48 bits de longitud cada una y se usa una de ellas por cada ronda. Dentro de la función de Feistel, la mitad derecha de un bloque de información se expande a un bloque de 48 bits de longitud mediante una función de expansión. Luego, se combina con la subllave correspondiente mediante la operación de bits O-exclusivo denotada por `XOR`. El bloque resultante pasa a través de 8 cajas de substitución produciendo un bloque de 32 bits de longitud que finalmente se permuta según una tabla de permutación. La salida de esta función se combina con la mitad izquierda mediante un `XOR` para formar la nueva mitad derecha, mientras que la última mitad derecha se convierte en la nueva mitad izquierda. Este proceso conforma una ronda de DES y como se ha mencionado anteriormente, se repite 16 veces. En seguida se detallan los pasos y componentes del sistema.

2.2.1 Permutación inicial IP y permutación final $\mathbf{IP^{-1}}$

La permutación inicial IP se lleva a cabo antes de la ronda 1, mientras que la permutación final IP^{-1} se realiza después de la ronda 16. La primera tabla transpone el texto plano, mientras que la segunda regresa los bits a su configuración inicial. Ambas tablas, como todas las que se presentan en este capítulo, se leen de izquierda a derecha y de arriba hacia abajo.

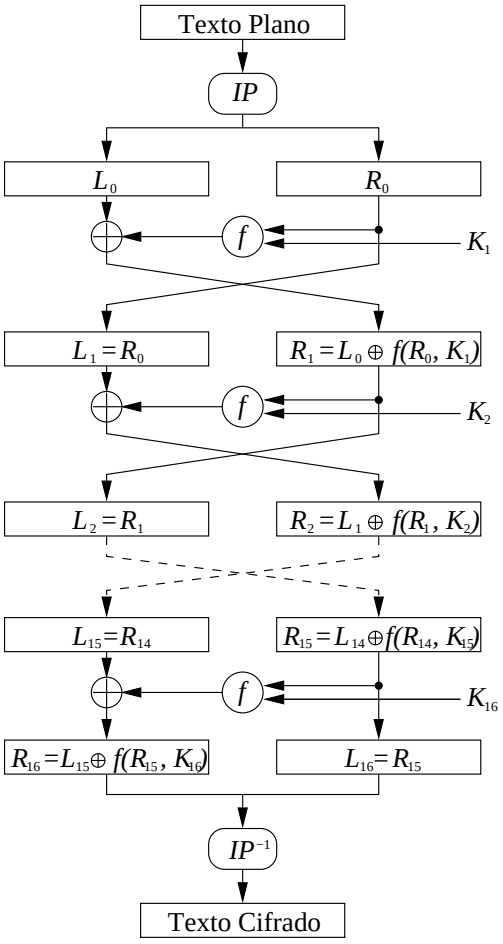

Figura 2.1: Estructura general del sistema DES.

La tabla (a) de la Figura 2.2 corresponde a la permutación inicial IP mientras que la tabla (b) de la misma Figura pertenece a la permutación final IP^{-1}. En la primera tabla se puede observar, por ejemplo, que el bit ubicado en la posición 58 se mueve a la primera posición, el bit 50 se mueve a la segunda posición, el bit 42 se mueve a la tercera posición y así sucesivamente.

Enseguida describiremos las permutaciones y las cajas de subtitución que forman parte de la función f.

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

(a)

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

(b)

Figura 2.2: Tabla de permutación inicial IP y tabla de permutación final IP^{-1}.

2.2.2 Permutación E

Esta tabla expande la mitad derecha de un bloque de información de 32 a 48 bits. Esto se hace duplicando y reordenando la mitad de los bits de la mitad derecha. La selección de estos bits se especifica en la Figura 2.3. Este tipo de tabla se conoce como *permutación de expansión* y tiene dos propósitos. El primero es igualar la longitud de la mitad derecha con la longitud de la llave para llevar a cabo el O-exclusivo. El segundo propósito es proporcionar como resultado un bloque más largo que pueda comprimirse cuando se apliquen las cajas de substitución. Al permitir que un sólo bit afecte a 2 substituciones, se obliga a que la dependencia de los bits de salida sobre los de entrada se propague rápidamente. Esto se conoce como *efecto avalancha*.

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

Figura 2.3: Tabla de permutación E.

2.2.3 Permutación P

Esta permutación es la última etapa en el cálculo de la función tipo Feistel f. Se aplica a un bloque de 32 bits y los reordena de acuerdo a la tabla especificada en la Figura 2.4.

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

Figura 2.4: Tabla de permutación P.

2.2.4 Las cajas de substitución de DES

En el sistema DES, las cajas de susbtitución o S-cajas son 8 tablas de valores diferentes, llamadas $S_1, S_2, ..., S_8$, que substituyen un grupo de 6 bits por otro de 4 bits. Un bloque de 48 bits se divide en 8 grupos de 6 bits. Después, cada grupo se opera con la caja correspondiente, esto es, al primer grupo se le aplica la caja S_1, al segundo grupo se le aplica la caja S_2, y así, sucesivamente.

Cada S-caja tiene 4 filas y 16 columnas. En cada fila, hay una permutación de todos los posibles valores de 4 bits, esto es, una permutación de los números del 0 al 15. La entrada de 6 bits se substituye como sigue: Primero se elige la fila de acuerdo al valor binario formado por la concatenación del primero y sexto bit, mientras que la columna está dada por el valor binario de los 4 bits que quedan en medio. Más específicamente, para un grupo de 6 bits, $b_1b_2b_3b_4b_5b_6$, los bits b_1 y b_6 se concatenan para formar el número binario b_1b_6 cuyos posibles valores son: 0, 1, 2 ó 3. Estos valores son los que, a su vez, identifican al renglón de la correspondiente S-caja. Por otro lado, los posibles valores para la cadena $b_2b_3b_4b_5$ están entre 0 y 15. Éstos corresponden a las columnas de la S-caja. Una vez que se ubica el renglón y la columna de la S-caja, el grupo se substituye por el valor indicado y esta es la salida de la aplicación de la S-caja.

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

Figura 2.5: Cajas de substitución del sistema DES.

Las S-cajas fueron diseñadas cuidadosamente e incluso sometidas a intensas pruebas computacionales. La substitución con las S-cajas es el paso crítico en el sistema DES, representan funciones no lineales de la entrada y esto es lo que le proporciona seguridad al sistema. Las 8 cajas de substitución se muestran en la Figura 2.5.

2.2.5 La función tipo Feistel de DES

La función tipo Feistel f para DES depende de la llave secreta y de un bloque de entrada R, formado por 32 bits. Esta función consiste en 4 operaciones criptográficas:

- Expansión
- Combinación con la subllave
- Substitución
- Permutación

El proceso para aplicar la función f comienza con la expansión del bloque R. Esta expansión se consigue al aplicar la permutación E de la Figura 2.3 a R. El resultado es un bloque de 48 bits de longitud que se combina con la subllave correspondiente a la ronda mediante un `XOR`. Después, el nuevo bloque es separado en 8 grupos de 6 bits de longitud denotados por B_i para $i = 1, ..., 8$. La substitución implica aplicar la caja de substitución S_i al grupo B_i correspondiente, es decir, $S_i(B_i)$ para $i = 1, ..., 8$. Con esto, se obtienen 8 grupos de 4 bits de longitud recuperando un bloque de 32 bits como se tenía originalmente. A éste último se le aplica la permutación P que será la salida de la función f. La Figura 2.6 describe este proceso.

2.2.6 Las rondas del estándar DES

La parte básica del estándar DES es lo que se conoce como una *ronda*. Una ronda es una estructura de Feistel que consiste en separar un bloque de 64 bits en dos partes de igual longitud. La mitad izquierda se identifica con L mientras que la mitad derecha se denota por R. Después, se aplica una función tipo Feistel que depende del bloque R y de la llave correspondiente.

La salida de la función tipo Feistel es un bloque de 32 bits de longitud que se

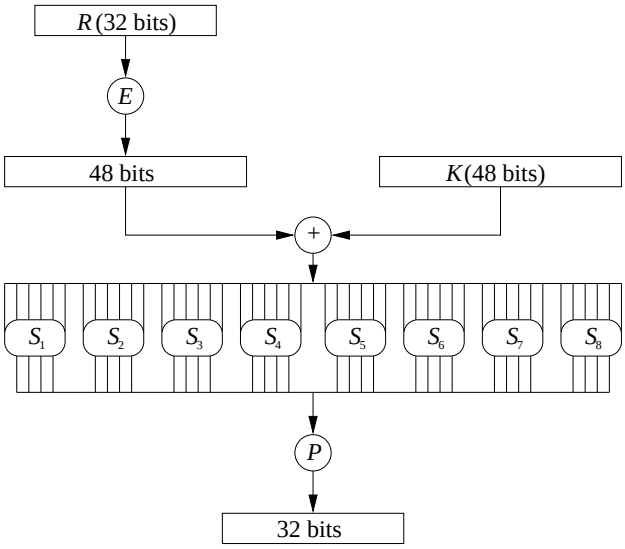

Figura 2.6: Estructura de la función f.

combina con el bloque izquierdo L mediante un `XOR`. El bloque resultante define el nuevo bloque derecho, mientras que el bloque derecho inicial R se convierte en el nuevo bloque izquierdo determinando los nuevos bloques para la siguiente ronda.

A continuación, describiremos el proceso de generación de las llaves. Comenzaremos detallando las permutaciones involucradas en este procedimiento.

2.2.7 Permutación PC_1 y permutación PC_2

Inicialmente, la llave de 64 bits se reduce a una de 56 bits removiendo los bits en la posiciones que son múltiplos de 8. Estos bits pueden usarse como bits de chequeo de paridad para asegurarse que la llave no tenga errores. Después, los bits restantes se permutan según la tabla de permutación PC_1 ver la Figura 2.7. Este tipo de tabla se conoce como *permutación de compresión* porque permuta el orden de los bits y elige un subconjunto de ellos.

A partir de esta llave, se generan 16 subllaves de 48 bits de longitud que se usarán en las 16 rondas de DES. Otra permutación de compresión que también se usa en la generación de las subllaves es la PC_2 la cual toma como entrada un bloque de 56 bits y devuelve un subconjunto permutado de 48 bits, ver la Figura 2.7. Ambas

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

(a)

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

(b)

Figura 2.7: Tabla de permutación PC_1 y tabla de permutación PC_2.

tablas se leen de izquierda a derecha y de arriba hacia abajo. Por ejemplo, en la tabla (a) de la Figura 2.7 el bit ubicado en la posición 57 se mueve a la primera posición, el bit 49 se mueve a la segunda posición, el bit 41 se mueve a la tercera posición y así sucesivamente.

2.2.8 Generación de las subllaves

Para generar las subllaves, primero se aplica la permutación PC_1 a la llave original de 64 bits. La tabla (a) de la Figura 2.7 nos muestra cómo quedan posicionados los bits de la llave. También, se puede apreciar que los 8 bits de paridad de la llave han quedado eliminados. El bloque resultante se divide dos partes de 28 bits cada una, llamadas tradicionalmente, C_0 y D_0. Luego, estas mitades recorren sus bits hacia la izquierda cierto número de posiciones según la tabla de la Figura 2.8. Por ejemplo, recorrer los bits de un bloque de 28 bits en dos posiciones a la izquierda significa que los bits ubicados en las posiciones tercera y cuarta, ahora ocuparán la primera y segunda posición, respectivamente. Los bits posteriores se recorren hasta la posición 26, mientras que los bits 1 y 2, se ubicarán hasta las posiciones 27 y 28, respectivamente. Esto produce 16 bloques C_i y 16 bloques D_i de 28 bits con $i = 1, ..., 16$.

Después, estos bloques se concatenan para formar un sólo bloque C_iD_i, del cual se extraerán 48 bits según la tabla de permutación PC_2 para cada $i = 1, ..., 16$. Los 16 bloques resultantes son precisamente las llamadas subllaves. En resumen,

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

Figura 2.8: Número de desplazmientos hacia la izquierda por ronda.

se tiene el siguiente algoritmo:

$$\begin{aligned}(C_0, D_0) &= PC_1(K)\\ C_i &= C_{i-1} \hookleftarrow v_i\\ D_i &= R_{i-1} \hookleftarrow v_i\\ K_i &= PC2(C_i D_i)\end{aligned}$$

para $i = 1, ..., 16$ donde $\hookleftarrow$ denota el desplazmiento de bits hacia la izquierda. Finalmente, presentamos los algoritmos de cifrado y de descifrado del esquema DES.

2.3 Cifrado y descifrado con DES

El cifrado comienza aplicando la permutación inicial IP a un bloque de texto M. Después, se aplican 16 rondas de la función Feistel y, finalmente, se aplica la permutación final IP^{-1} al bloque resultante:

$$\begin{aligned}(L_0, R_0) &= IP(M)\\ L_i &= R_{i-1}\\ R_i &= L_{i-1} + f(R_{i-1}, K_i)\\ C &= IP^{-1}(R_{16}L_{16})\end{aligned}$$

para $i = 1, ..., 16$ donde L_i y R_i representan la i-ésima mitad izquierda y mitad derecha del bloque, respectivamente. Además, f es la función de Feistel descrita en la sección 2.2.7, es decir,

$$f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i)).$$

El descifrado consiste, esencialmente, en ir hacia atrás. Esto es, se aplican las mismas operaciones que en el cifrado sólo que, en este caso, se comienza con la

subllave K_{16} y se finaliza con K_1. De tal manera que:

$$\begin{aligned}(L_{16}, R_{16}) &= IP(C)\\ R_j &= L_{j+1}\\ L_j &= R_{j+1} + f(L_{j+1}, K_{j+1})\\ M &= IP^{-1}(R_0 L_0)\end{aligned}$$

donde $j = 16 - i$ para $i = 1, ..., 16$. Después de estas rondas, sólo resta aplicar la permutación IP^{-1} para recuperar el bloque original.

2.3.1 Ejemplo

A continuación, se ilustra la forma en cómo opera DES sobre un bloque de texto plano 64 bits y el texto cifrado resultante. Supóngase que se desea cifrar el mensaje "*SaludosTerricola*". Como requerimos una representación binaria del mensaje, el código ASCII de los caracteres de la cadena resulta útil para después encontrar los dígitos binarios de cada código y, así, representar todo el mensaje en notación binaria. El código ASCII de los caracteres del mensaje es el siguiente conjunto:

$$\{83, 97, 108, 117, 100, 111, 115, 84, 101, 114, 114, 105, 99, 111, 108, 97\}.$$

El valor 83 puede expresarse como la cadena 11001010 considerando el bit de la izquierda como el menos significativo. De este modo, el mensaje se expresa de la manera siguiente:

1100101010000110001101101010111000100110111101101100111000101010
1010011001001110010011101001011011000110111101100011011010000110

Como el esquema opera sobre bloques de 64 bits, dividimos el mensaje en dos bloques y ciframos uno a la vez. El primer bloque de texto plano M_1 está dado por:

$$M_1 = 1100101010000110001101101010111000100110111101101100111000101010$$

Mientras que el segundo bloque M_2 es:

$$M_2 = 1010011001001110010011101001011011000110111101100011011010000110.$$

También, el esquema usa llaves de 56 bits. Inicialmente, se almacenan como bloques de 64 bits pero cada octavo bit no se usa y se descarta. Para la llave K

podemos elegir un conjunto de números aleatorios entre 0 y 255 y expresarlos en formato binario, por ejemplo,

$$\{200, 44, 234, 158, 217, 61, 251, 143\}.$$

Cuya representación binaria es

$$K = 0001001100110100010101110111100110011011101111001101111111110001$$

El primer paso es crear las 16 subllaves a partir de la llave K. Para hacerlo, primero hay que remover los bits en una posición múltiplo de 8 de izquierda a derecha y aplicar la tabla de permutación PC_1, ver Figura 2.7. El resultado es el siguiente bloque de 56 bits:

$$K^* = 11110000110011001010101011111010101010110011001111000111 1$$

En seguida, el bloque K^* se separa en mitad izquierda y mitad derecha, C_0 y D_0, con 28 bits cada una:

$$\begin{aligned} C_0 &= 1111000011001100101010101111 \\ D_0 &= 0101010101100110011110001111 \end{aligned}$$

Con estos dos bloques, se generan 16 nuevos bloques, C_i y D_i, $1 \leq i \leq 16$. Cada nuevo par, C_i y D_i, se forma a partir del par anterior, C_{i-1} y D_{i-1}, siguiendo la tabla de desplazmientos de la Figura 2.8. Los nuevos bloques son los siguientes:

$$\begin{aligned} C_1 &= 1110000110011001010101011111 \\ D_1 &= 1010101011001100111100011110 \\ C_2 &= 1100001100110010101010111111 \\ D_2 &= 0101010110011001111000111101 \\ C_3 &= 0000110011001010101011111111 \\ D_3 &= 0101011001100111100011110101 \\ C_4 &= 0011001100101010101111111100 \\ D_4 &= 0101100110011110001111010101 \\ C_5 &= 1100110010101010111111110000 \\ D_5 &= 0110011001111000111101010101 \\ C_6 &= 0011001010101011111111000011 \\ D_6 &= 1001100111100011110101010101 \\ C_7 &= 1100101010101111111100001100 \\ D_7 &= 0110011110001111010101010110 \end{aligned}$$

$$
\begin{aligned}
C_8 &= 00101010101111111110000110011 \\
D_8 &= 10011110001111010101011001 \\
C_9 &= 01010101011111111100001100110 \\
D_9 &= 00111100011110101010110011 \\
C_{10} &= 01010101111111100001100110011001 \\
D_{10} &= 11110001111010101011001100 \\
C_{11} &= 01010111111110000110011001100101 \\
D_{11} &= 11000111101010101100110011 \\
C_{12} &= 01011111111000011001100101010 1 \\
D_{12} &= 00011110101010101100110011111 \\
C_{13} &= 01111111100001100110010101 01 \\
D_{13} &= 01111010101010110011001111 00 \\
C_{14} &= 11111110000110011001010101 01 \\
D_{14} &= 11101010101011001100111100 01 \\
C_{15} &= 11111000011001100101010101 11 \\
D_{15} &= 10101010101100110011110001 11 \\
C_{16} &= 11110000110011001010101011 11 \\
D_{16} &= 01010101011001100111100011 11
\end{aligned}
$$

Las subllaves K_i para $1 \leq i \leq 16$ se forman al aplicar la tabla de permutación PC_2 al bloque concatenado C_iD_i. Cada bloque C_iD_i tiene 56 bits pero al aplicarle la tabla de permutación PC_2 queda un bloque de sólo 48 bits. Por ejemplo, el primer bloque C_1D_1 es el siguiente:

$$C_1D_1 = 11100001100110010101010111111010101011001100111100011110$$

Después de aplicar la tabla de permutación PC_2, se obtiene la subllave K_1:

$$K_1 = 000110110000001011101111111111000111000001110010$$

El resto de las subllaves son las siguientes:

$$
\begin{array}{lcccccccc}
K_2 & = & 011110 & 011010 & 111011 & 011001 & 110110 & 111100 & 100111 & 100101 \\
K_3 & = & 010101 & 011111 & 110010 & 001010 & 010000 & 101100 & 111110 & 011001 \\
K_4 & = & 011100 & 101010 & 110111 & 010110 & 110110 & 110011 & 010100 & 011101 \\
K_5 & = & 011111 & 001110 & 110000 & 000111 & 111010 & 110101 & 001110 & 101000 \\
K_6 & = & 011000 & 111010 & 010100 & 111110 & 010100 & 000111 & 101100 & 101111 \\
K_7 & = & 111011 & 001000 & 010010 & 110111 & 111101 & 100001 & 100010 & 111100 \\
K_8 & = & 111101 & 111000 & 101000 & 111010 & 110000 & 010011 & 101111 & 111011 \\
K_9 & = & 111000 & 001101 & 101111 & 101011 & 111011 & 011110 & 011110 & 000001 \\
K_{10} & = & 101100 & 011111 & 001101 & 000111 & 101110 & 100100 & 011001 & 001111 \\
K_{11} & = & 001000 & 010101 & 111111 & 010011 & 110111 & 101101 & 001110 & 000110 \\
K_{12} & = & 011101 & 010111 & 000111 & 110101 & 100101 & 000110 & 011111 & 101001 \\
K_{13} & = & 100101 & 111100 & 010111 & 010001 & 111110 & 101011 & 101001 & 000001 \\
K_{14} & = & 010111 & 110100 & 001110 & 110111 & 111100 & 101110 & 011100 & 111010 \\
K_{15} & = & 101111 & 111001 & 000110 & 001101 & 001111 & 010011 & 111100 & 001010 \\
K_{16} & = & 110010 & 110011 & 110110 & 001011 & 000011 & 100001 & 011111 & 110101
\end{array}
$$

Una vez que se tienen las subllaves, se procede a encriptar el bloque de información correspondiente. Se comienza con la permutación inicial IP, ver Figura 2.2. Después de aplicar esta permutación, el bloque M_1 queda como sigue:

$$M_1' = 01100001\ 00100100\ 01111110\ 00000000\ 01101011\ 10111100\ 11001001\ 11111111$$

En seguida, el bloque se divide en mitad izquierda L_0 y mitad derecha R_0:

$$
\begin{array}{lcccccc}
L_0 & = & 01100001 & 00100100 & 01111110 & 00000000 \\
R_0 & = & 01101011 & 10111100 & 11001001 & 11111111
\end{array}
$$

Al final, este proceso produce un bloque de texto cifrado que se forma al concatenar $L_{16}R_{16}$. En cada iteración, el bloque derecho del resultado anterior se convierte en el bloque izquierdo de la iteración actual. Para el bloque derecho actual, se ejecuta un `XOR` entre el bloque izquierdo anterior y la salida de la función f. Por ejemplo, para $i = 1$, se tiene

$$
\begin{array}{lcl}
K_1 & = & 000110\ 110000\ 001011\ 101111\ 111111\ 000111\ 000001\ 110010 \\
L_1 & = & R_0 = 01101011\ 10111100\ 11001001\ 11111111 \\
R_1 & = & L_0 + f(R_0, K_1)
\end{array}
$$

Para calcular f, primero se expande el bloque R_{i-1} de 32 bits a 48 bits. Esto es el efecto que tiene la tabla de expansión E, ver Figura 2.3. Por ejemplo, para un bloque R_i de 32 bits, la salida $E(R_i)$ es un bloque de 48 bits. Así, si se calcula $E(R_0)$, a partir de R_0, se obtiene:

$$\begin{array}{lcl} R_0 & = & 0110\ 1011\ 1011\ 1100\ 1100\ 1001\ 1111\ 1111 \\ E(R_0) & = & 101101\ 010111\ 110111\ 111001\ 011001\ 010011\ 111111\ 111110 \end{array}$$

Esta expansión del bloque sirve para el siguiente paso en el cálculo de f, esto es, aplicar $E(R_{i-1}) \oplus K_i$. Para K_1 y $E(R_0)$, se tiene:

$$\begin{array}{lcl} K_1 & = & 000110\ 110000\ 001011\ 101111\ 111111\ 000111\ 000001\ 110010 \\ E(R_0) & = & 101101\ 010111\ 110111\ 111001\ 011001\ 010011\ 111111\ 111110 \\ K_1 \oplus E(R_0) & = & 101011\ 100111\ 111100\ 010110\ 100110\ 010100\ 111110\ 001100 \end{array}$$

Hasta este punto, se tiene un bloque de 48 bits que puede dividirse en 8 grupos de 6 bits. Con estos grupos se localizarán los renglones y las columnas en las S-cajas correspondientes a cada grupo, esto es, cada grupo tiene asignado una S-caja diferente y los bits que lo integran definen un renglón y una columna de dicha caja. El valor de la tabla dado por el renglón y la columna, es un grupo de 4 bits por el que se substituirá el actual grupo de 6 bits. El resultado es que los 8 grupos de 6 bits, ahora se transforman en 8 grupos de 4 bits cada uno creando un nuevo bloque de 32 bits.

El resultado previo, puede escribirse de la siguiente manera:

$$K_n + E(R_{n-1}) = B_1B_2B_3B_4B_5B_6B_7B_8$$

donde cada B_i es un grupo de 6 bits. Para cada uno, se calcula:

$$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$$

donde $S_i(B_i)$ es el grupo de cuatro bits que resulta de aplicar la i-ésima S-caja.

Por ejemplo, de la primera ronda, se tiene el siguiente bloque:

$$K_1 \oplus E(R_0) \quad = \quad 101011\ 100111\ 111100\ 010110\ 100110\ 010100\ 111110\ 001100$$

Para aplicar S_1 al primer bloque 101011, primero se quitan el primer y último bit de la cadena, al concatenarlos se forma una cadena de dos bits que indica el renglón de la tabla S_1. En este caso, la cadena 11 indica el renglón 3 en binario. Luego, los bits restantes del bloque indican la columna de la tabla en notación binaria. Es decir, 0101 se refiere a la columna 5. El valor que aparece en el renglón 3 y columna 5 de la tabla S_1 es 9, ver Figura 2.5. Entonces el bloque 101011 se substituye por el número 9 expresado en notación binaria: 1001. El mismo proceso se aplica para cada bloque y S-caja correspondientes, obteniéndose el siguiente resultado:

$$10010001111001011011001100101011$$

El paso final en el cálculo de la función f es aplicar la tabla de permutación P al resultado anterior, esto es,

$$P(S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8))$$

En particular,

$$P(10010001111001011011001100101011) = 10101001101000110111110110000110$$

Por lo tanto,

$$f(R_0, K_1) = 00100011010010101001101110 11$$

De manera inmediata, se puede calcular $R_1 = L_0 \oplus f(R_0, K_1)$

$$\begin{array}{rcl} R_1 & = & 0110\ 0001\ 0010\ 0100\ 0111\ 1110\ 0000\ 0000 \\ & \oplus & 1010\ 1001\ 1010\ 0011\ 0111\ 1101\ 1000\ 0110 \\ & = & 1100\ 1000\ 1000\ 0111\ 0000\ 0011\ 1000\ 0110 \end{array}$$

En la siguiente ronda, se hace $L_2 = R_1$ y se calcula $R_2 = L_1 \oplus f(R_1, K_2)$. Al final de la última ronda, se tienen los bloques L_{16} y R_{16} los cuales se concatenan en orden inverso, $R_{16}L_{16}$, para formar un bloque de 64 bits al que se le aplicará la permutación IP^{-1}. En el ejemplo, el resultado de la ronda 16 es:

$$\begin{array}{rcl} L_{16} & = & 1010\ 0100\ 1100\ 1101\ 0111\ 0111\ 1100\ 0010 \\ R_{16} & = & 1100\ 0010\ 0101\ 0110\ 1000\ 0110\ 1011\ 1010 \end{array}$$

Después de aplicar IP^{-1}:

$$\begin{array}{rcl} IP^{-1}(R_{16}L_{16}) & = & 00101000\ 01011111\ 10111100\ 00100001 \\ & & 00011001\ 10001001\ 01111010\ 11100111 \end{array}$$

Este es el bloque cifrado de la primera parte del mensaje, siguiendo el mismo proceso se obtiene el segundo bloque cifrado

$$C_2 = 10011110101111100110101001010111101010010100000000111001111011 10.$$

El proceso de descifrado consiste, esencialmente, en aplicar el mismo algoritmo de cifrado pero con el orden de las subllaves invertido. Es decir, la primera ronda comienza con la subllave K_{16} y así, sucesivamente hasta la última ronda que usa K_1. Por ejemplo, retomemos el bloque C_1, para descifrarlo se aplica, como primer paso, la permutación IP. Esto cancela la permutación IP^{-1} aplicada en el cifrado.

$$\begin{aligned} IP(C_1) &= 10110011\ 00011010\ 00010110\ 11111100 \\ &\quad\ 10011100\ 01001111\ 00011010\ 00100110 \end{aligned}$$

Distinguiendo dos partes de $IP(C_1)$ tenemos

$$\begin{aligned} L_0 &= 10110011\ 00011010\ 00010110\ 11111100 \\ R_0 &= 10011100\ 01001111\ 00011010\ 00100110 \end{aligned}$$

Después, se aplica una ronda de a esta cadena junto con la llave K_{16}

$$K_{16} = 110010110011110110001011000011100001011111110101,$$

dando como resultado el bloque

$$\begin{aligned} f(R_0, K_{16}) &= 10100100\ 11001101\ 01110111\ 11000010 \\ &\quad\ 10010110\ 11010111\ 11010110\ 00111111. \end{aligned}$$

Después de aplicar las 15 rondas siguientes, se usa la permutación IP^{-1} para cancelar la permutación IP aplicada en el primer paso del cifrado recuperando, así, el bloque inicial M_1:

$$\begin{aligned} IP^{-1}(R_{16}L_{16}) &= 11001010\ 10000110\ 00110110\ 10101110 \\ &\quad\ 00100110\ 11110110\ 11001110\ 00101010. \end{aligned}$$

Capítulo 3
Descripción polinomial del sistema de cifrado DES

En este capítulo se presenta la descripción algebraica del sistema de cifrado DES. Dicha presentación usa un enfoque polinomial apoyado en estructuras algebraicas como campos finitos y anillos de polinomios para describir el algoritmo de cifrado. La especificación de este sistema en estos términos representa la principal contribución de este trabajo. Enseguida se detallan los componentes del sistema DES descritos mediante operaciones de polinomios.

3.1 Sistema de cifrado DES

Como se describió en el capítulo anterior, el sistema de cifrado DES opera sobre bloques de texto plano de 64 bits de longitud. Los bits de cada bloque se identifican mediante su posición, por lo que un bloque m tiene los siguientes componentes $m = (m_0, m_1, ..., m_{63})$ donde cada $m_i \in \mathbb{F}_2 = \{0, 1\}$. Si $\mathbb{F}_2^{64}$ denota el espacio vectorial de dimensión 64 sobre $\mathbb{F}_2$, esto es,

$$\mathbb{F}_2^{64} = \{a = (a_0, a_1, ..., a_{63}) : a_i \in \mathbb{F}_2\},$$

entonces $m \in \mathbb{F}_2^{64}$. En esta descripción, se desea usar una representación polinomial del bloque m, por lo tanto, se formarán grupos de 8 bits dentro del bloque, a cada grupo se le asociará un polinomio de grado a lo más siete en la variable x y, por lo tanto, dentro de cada bloque se distinguirán a lo más ocho de estos polinomios. Para diferenciar la posición de cada grupo, se usará una potencia en una segunda variable, digamos, y.

Primero, considérese el anillo de polinomios

$$\mathcal{A}_{8,8} := \mathbb{F}_2[x,y]/\langle x^8-1, y^8-1\rangle$$

y defínase el mapeo

$$\begin{aligned} \phi : \mathbb{F}_2^{64} &\rightarrow \mathcal{A}_{8,8} \\ \phi(m) = \phi(m_0, m_1, ..., m_{63}) &\rightarrow m(x,y) = \sum_{i=0}^{7}\sum_{j=0}^{7} m_{i,j}x^i y^j \end{aligned}$$

donde $m_{i,j} \in \mathbb{F}_2$ y (i,j) es la representación de l, $0 \leq l \leq 63$, en base 8, es decir, $l := 8j + i$. Análogamente, la llave original que también es un bloque de 64 bits, $k = (k_0, k_1, ..., k_{63})$, puede representarse mediante un polinomio en $\mathcal{A}_{8,8}$:

$$k = k(x,y) = \sum_{i=0}^{7}\sum_{j=0}^{7} k_{i,j}x^i y^j.$$

Por ejemplo, consideremos la primera parte del mensaje *"SaludosTerricola"* presentada en la sección (2.3), cuya representación binaria es

$$\begin{aligned} m &= 11001010\ 10000110\ 00110110\ 10101110 \\ &\quad 00100110\ 11110110\ 11001110\ 00101010. \end{aligned}$$

Ahora tiene la siguiente representación polinomial:

$$\begin{aligned} m(x,y) &= 1 + x + x^4 + x^6 + (1 + x^5 + x^6)y + (x^2 + x^3 + x^5 + x^6)y^2 \\ &\quad +(1 + x^2 + x^4 + x^5 + x^6)y^3 + (x^2 + x^5 + x^6)y^4 \\ &\quad +(1 + x + x^2 + x^3 + x^5 + x^6)y^5 + (1 + x + x^4 + x^5 + x^6)y^6 \\ &\quad +(x^2 + x^4 + x^6)y^7. \end{aligned} \tag{3.1}$$

Obsérvese que cada polinomio en x, representa un grupo de 8 bits, por ejemplo, el primer grupo 11001010 está representado por $1+x+x^4+x^6$, mientras que el grupo 10000110 tiene como representante al polinomio $1 + x^5 + x^6$. Además, usamos las potencias de y para distinguir cada grupo dentro del bloque. A su vez, la llave

$$\begin{aligned} k &= 00010011\ 00110100\ 01010111\ 01111001 \\ &\quad 10011011\ 10111100\ 11011111\ 11110001 \end{aligned}$$

del mismo ejemplo queda representada por:

$$\begin{aligned} k(x,y) &= (x^3+x^6+x^7)+(x^2+x^3+x^5)y+(x+x^3+x^5+x^6+x^7)y^2 \\ &\quad +(x+x^2+x^3+x^4+x^7)y^3+(1+x^3+x^4+x^6+x^7)y^4 \\ &\quad +(1+x^2+x^3+x^4+x^5)y^5+(1+x+x^3+x^4+x^5+x^6+x^7)y^6 \\ &\quad +(1+x+x^2+x^3+x^7)y^7 \end{aligned} \tag{3.2}$$

3.1.1 Permutación inicial IP y permutación final $\mathbf{IP^{-1}}$

El proceso de cifrado comienza aplicando la permutación IP a un bloque de información m. El efecto de IP sobre un polinomio $m(x,y)$ se logra mediante la siguiente transformación:

$$m(x,y)=\sum_{i=0}^{7}\sum_{j=0}^{7} m_{ij}x^iy^j \xrightarrow{IP} \sum_{i=0}^{7}(x^7\sum_{j=0}^{7} m_{ij}x^{7j})y^k$$

donde $k=\lfloor j/2\rfloor+4(j \mod 2)+4$.

Para construir la tranformación IP, primero se define la transformación π que tiene el efecto de transponer los coeficientes de un polinomio. La transformación $\pi:\mathcal{A}_{8,8}\to\mathcal{A}_{8,8}$ está dada por:

$$\pi(p(x,y))=p(y,x).$$

Cuando se aplica a un polinomio $m(x,y)\in\mathcal{A}_{8,8}$, se obtiene:

$$m(x,y)=\sum_{i=0}^{7}\sum_{j=0}^{7} m_{ij}x^iy^j \xrightarrow{\pi} \sum_{i=0}^{7}\sum_{j=0}^{7} m_{ij}y^ix^j$$

Luego, se aplica la transformación $g:\mathcal{A}_{8,8}\to\mathcal{A}_{8,8}$ definida como:

$$g(p(x,y))=x^7p(x^7,y).$$

Al aplicar g a $k(x,y)$ se obtiene:

$$\sum_{i=0}^{7}\sum_{j=0}^{7} m_{ij}y^ix^j \xrightarrow{g} \sum_{i=0}^{7}(x^7\sum_{j=0}^{7} m_{ij}x^{7j})y^i.$$

Finalmente, se define el índice $k = \lfloor i/2 \rfloor + 4(i \mod 2) + 4$ para cada i, y se tiene:

$$\sum_{i=0}^{7}(x^7 \sum_{j=0}^{7} m_{ij}x^{7j})y^k.$$

Al aplicar la transformación IP al polinomio (3.1) de nuestro ejemplo, se obtiene:

$$\begin{aligned} IP(m(x,y)) &= x + x^2 + x^7 + (x^2 + x^5)y + (x + x^2 + x^3 + x^4 + x^5 + x^6)y^2 \\ &\quad +(x + x^2 + x^4 + x^6 + x^7)y^4 + (1 + x^2 + x^3 + x^4 + x^5)y^5 \\ &\quad +(1 + x + x^4 + x^7)y^6 \\ &\quad +(1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7)y^7 \end{aligned} \tag{3.3}$$

Al final de las 16 rondas se aplica la permutación IP^{-1}. Dicha permutación consiste de las siguientes transformaciones. Se comienza con la transformación π, descrita anteriormente:

$$m(x,y) = \sum_{i=0}^{7}\sum_{j=0}^{7} m_{ij}x^i y^j \xrightarrow{\pi} \sum_{i=0}^{7}\sum_{j=0}^{7} m_{ij}y^i x^j$$

Nuevamente, para cada j se aplica la transformación g:

$$\sum_{i=0}^{7}\sum_{j=0}^{7} m_{ij}y^i x^j \xrightarrow{g} \sum_{j=0}^{7}(y^7 \sum_{i=0}^{7} m_{ij}y^{7i})x^j.$$

La permutación IP^{-1} termina definiendo el índice $k = 7\lfloor j/4 \rfloor + 2(j \mod 4) + 1$ para cada j, y se tiene:

$$\sum_{j=0}^{7}(y^7 \sum_{i=0}^{7} m_{ij}y^{7i})x^k.$$

Las rutinas **IPperm** y **IPInvperm** del apéndice A simulan el efecto de estas permutaciones.

3.1.2 Permutación E

Consideremos un bloque $m(x,y)$ y distingamos dos componentes dentro de él, esto es, $m(x,y) = l(x,y) + y^4 r(x,y)$. Los polinomios $l(x,y)$ y $r(x,y)$ son elementos del anillo $\mathcal{A}_{8,4} = \mathbb{F}_2[x,y]/\langle x^8 - 1, y^4 - 1\rangle$. Entonces a partir de $r(x,y) =$

$\sum_{i=0}^{7}\sum_{j=0}^{3} r_{ij}x^i y^j$ expresamos la permutación de expansión E mediante el siguiente polinomio:

$$\begin{aligned} e(x,y) &:= r_{73} + r_{00}x + r_{10}x^2 + r_{20}x^3 + r_{30}x^4 + r_{40}x^5 \\ &+ (r_{30} + r_{40}x + r_{50}x^2 + r_{60}x^3 + r_{70}x^4 + r_{01}x^5)y \\ &+ (r_{70} + r_{01}x + r_{11}x^2 + r_{21}x^3 + r_{31}x^4 + r_{41}x^5)y^2 \\ &+ (r_{31} + r_{41}x + r_{51}x^2 + r_{61}x^3 + r_{71}x^4 + r_{01}x^5)y^3 \\ &+ (r_{71} + r_{02}x + r_{12}x^2 + r_{22}x^3 + r_{32}x^4 + r_{42}x^5)y^4 \\ &+ (r_{32} + r_{42}x + r_{52}x^2 + r_{62}x^3 + r_{72}x^4 + r_{03}x^5)y^5 \\ &+ (r_{72} + r_{03}x + r_{13}x^2 + r_{23}x^3 + r_{33}x^4 + r_{43}x^5)y^6 \\ &+ (r_{33} + r_{43}x + r_{53}x^2 + r_{63}x^3 + r_{73}x^4 + r_{00}x^5)y^7 \end{aligned}$$

Este polinomio se obtiene de observar la tabla de permutación E y distinguir qué posiciones de los bits son las que se repiten. El polinomio $r(x,y)$ extraido de (3.3) de nuestro ejemplo, es:

$$\begin{aligned} r(x,y) &= x + x^2 + x^4 + x^6 + x^7 + (1 + x^2 + x^3 + x^4 + x^5)y \\ &\quad +(1 + x + x^4 + x^7)y^2 \\ &\quad +(1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7)y^3 \end{aligned} \tag{3.4}$$

Cuando se define $e(x,y)$ a partir de (3.4), éste queda como:

$$\begin{aligned} e(x,y) &:= 1 + x^2 + x^3 + x^5 + (x + x^3 + x^4 + x^5)y \\ &\quad +(1 + x + x^3 + x^4 + x^5)y^2 + (1 + x + x^2 + x^5)y^3 \\ &\quad +(x + x^2 + x^5)y^4 + (x + x^4 + x^5)y^5 \\ &\quad +(1 + x + x^2 + x^3 + x^4 + x^5)y^6 + (1 + x + x^2 + x^3 + x^4)y^7 \end{aligned}$$

Con la rutina **Eperm** se obtiene el polinomio que se define con la permutación E.

3.1.3 Permutación P

Para definir esta permutación usaremos dos campos finitos. Construimos el primer campo finito con el campo $\mathbb{F}_2$ y el polinomio irreducible $\lambda(x) := x^8+x^4+x^3+x^2+1 \in \mathbb{F}_2[x]$:

$$\mathbb{F}_{2^8} := \frac{\mathbb{F}_2[x]}{\langle \lambda(x) \rangle}.$$

Ahora, usamos el polinomio irreducible $v^4 + v + 1 \in \mathbb{F}_2[v]$ para definir el segundo campo finito:

$$\mathbb{F}_{2^4} := \frac{\mathbb{F}_2[v]}{\langle v^4 + v + 1 \rangle}.$$

Los elementos de $\mathbb{F}_{2^8}$ tienen la forma:

$$a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5 + a_6x^6 + a_7x^7 \text{ donde } a_i \in \mathbb{F}_2.$$

Mientras que los elementos de $\mathbb{F}_{2^4}$ se ven como:

$$a_0 + a_1v + a_2v^2 + a_3v^3 \text{ donde } a_i \in \mathbb{F}_2.$$

La permutación P se puede describir mediante varios polinomios de permutación y productos monomiales aplicados a un elemento $p(x, y) \in \mathcal{A}_{8,4}$, esto es, un polinomio $\sum_{i=0}^{7}\sum_{j=0}^{3} r_{ij}x^iy^j$. En este caso, es conveniente observar que un polinomio $p(x, y) \in \mathcal{A}_{8,4}$ tiene 4 polinomios de grado a lo más 7 en la variable x, es decir, 4 elementos del campo $\mathbb{F}_{2^8}$. También, tiene 8 elementos de $\mathbb{F}_{2^4}$, o sea, 8 polinomios de grado a lo más 3 en la variable y.

Para calcular los polinomios de permutación, usaremos la Fórmula de Interpolación de Lagrange para campos finitos, ver [17]. Este método nos permite construir un polinomio que toma valores específicos de un campo finito $\mathbb{F}$ para valores dados en el mismo campo. La Fórmula de Interpolación de Lagrange es la siguiente:

$$f(u) = \sum_{c\in\mathbb{F}} \tau(c)(1 - (u - c)^{|\mathbb{F}|-1}) \tag{3.5}$$

donde τ es una función arbitraria de $\mathbb{F}$ en $\mathbb{F}$ y $|\mathbb{F}|$ denota la cardinalidad de $\mathbb{F}$. En nuestro caso, se requiere que τ se una función uno-uno y sobre, es decir, una permutación de los elementos de $\mathbb{F}$.

Por ejemplo, consideremos un elemento $q(y) = a_0 + a_1y + a_2y^2 + a_3y^3 \in \mathbb{F}_{2^4}$. Supongamos que nuestra permutación consiste sólo en intercambiar la posición del coeficiente que acompaña a y^3 con la del coeficiente de y. Sea $\tau : \mathbb{F}_{2^4} \to \mathbb{F}_{2^4}$ una función de $\mathbb{F}_{2^4}$ en $\mathbb{F}_{2^4}$ dada por

$$\tau(q(y)) = a_0 + a_3y + a_2y^2 + a_1y^3$$

para todo polinomio en $\mathbb{F}_{2^4}$. El proceso se simplifica con *Mathematica v5.0* especificando el dominio e imagen de τ mediante listas y ejecutando la fórmula (3.5):

$$D = \{0, y, y^2, y + y^2, y^3, ..., 1 + y + y^2 + y^3\},$$

$$I = \{0, y^3, y^2, y^2 + y^3, y, ..., 1 + y + y^2 + y^3\}.$$

El polinomio de permutación que describe τ es entonces

$$p(v) = v(1 + y^2) + v^2(1 + y + y^2) + v^4(y + y^2 + y^3) + v^8(1 + y^2 + y^3).$$

El conjunto de operaciones que se requieren para describir P se da en 10 pasos:

1. Multiplicar $x\big(\sum_{i=0}^{7} r_{i2}x^i y^2\big)$ y $x^4\big(\sum_{i=0}^{7} r_{i3}x^i y^3\big)$.
2. Multiplicar:
   (a) $y\big(\sum_{j=0}^{3} r_{0j}y^j\big)$.
   (b) $y^2\big(\sum_{j=0}^{3} r_{1j}xy^j\big)$, $y^2\big(\sum_{j=0}^{3} r_{4j}x^4y^j\big)$ y $y^2\big(\sum_{j=0}^{3} r_{5j}x^5y^j\big)$.
   (c) $y^3\big(\sum_{j=0}^{3} r_{3j}x^3y^j\big)$ y $y^3\big(\sum_{j=0}^{3} r_{7j}x^7y^j\big)$
3. Aplicar:
   (a) el polinomio de permutación
   $$p_1(v) = v(1 + y^2) + v^2(1 + y + y^2) + v^4(y + y^2 + y^3) + v^8(1 + y^2 + y^3)$$
   a los polinomios $\sum_{j=0}^{3} r_{1j}xy^j$ y $\sum_{j=0}^{3} r_{2j}x^2y^j$.
   (b) el polinomio de permutación
   $$p_2(v) = v(y + y^2) + v^2 + v^4(1 + y + y^2) + v^8$$
   al polinomio $\sum_{j=0}^{3} r_{4j}x^4y^j$.
4. Multiplicar:
   (a) $x^7\big(\sum_{i=0}^{7} r_{i1}x^i y\big)$.
   (b) $x^3\big(\sum_{i=0}^{7} r_{i3}x^i y^3\big)$.
5. Aplicar $v + (1 + y)v^2 + y^2v^4 + yv^8$ a $\sum_{j=0}^{3} r_{2j}x^2y^j$.
6. Multiplicar $x^7\big(\sum_{i=0}^{7} r_{i2}x^i y^2\big)$.
7. Aplicar el polinomio de permutación
   $$p_3(v) = v(y + y^2) + v^2(1 + y^3) + v^4(1 + y + y^3) + v^8(1 + y^2)$$
   al polinomio $\sum_{j=0}^{3} r_{3j}x^3y^j$.

8. Multiplicar:
   (a) $x^6\big(\sum_{i=0}^{7} r_{i0}x^i\big)$.
   (b) $x^7\big(\sum_{i=0}^{7} r_{i2}x^i y^2\big)$.
9. Aplicar el polinomio de permutación

$$\begin{aligned} p_4(u) &= u(1+x^4+x^6)+u^2(x^3+x^4+x^5+x^6) \\ &\quad +u^4(x+x^2+x^5+x^6)+u^8(x+x^3+x^7) \\ &\quad +u^{16}(1+x+x^4+x^5+x^6) \\ &\quad +u^{32}(1+x+x^2+x^5)+u^{64}(1+x^2+x^5) \\ &\quad +u^{128}(1+x^2+x^4+x^5+x^7) \end{aligned}$$

   al polinomio $\sum_{i=0}^{7} r_{i1}x^i y$.
10. Aplicar:
    (a) el polinomio de permutación

$$\begin{aligned} p_5(u) &= u(x+x^2+x^5) \\ &\quad +u^2(x^2+x^3+x^4+x^5+x^6+x^7) \\ &\quad +u^4(x^2+x^7)+u^8(x^3+x^4) \\ &\quad +u^{16}(1+x+x^2+x^7)+u^{32}(x+x^2+x^4) \\ &\quad +u^{64}((x^4+x^5+x^6) \\ &\quad +u^{128}(1+x+x^2+x^5+x^6+x^7) \end{aligned}$$

    al polinomio $\sum_{i=0}^{7} r_{i0}x^i$.
    (b) la transformación g al polinomio $\sum_{i=0}^{7} r_{i1}x^i y$. Esto es,

$$(x^7\sum_{i=0}^{7} r_{i1}x^{7i})y.$$

    (c) el polinomio de permutación

$$\begin{aligned} p_7(u) &= u(1+x^2+x^3+x^6) \\ &\quad +u^2(x+x^2+x^4+x^5+x^6+x^7) \\ &\quad +u^4(x+x^2+x^3+x^4)+u^8(x^2+x^7)+u^{16}(x+x^3+x^7) \\ &\quad +u^{32}(x+x^2+x^3+x^4+x^5+x^7) \\ &\quad +u^{64}(1+x+x^2+x^3+x^7)+u^{128}(x+x^3+x^4+x^6+x^7) \end{aligned}$$

al polinomio $\sum_{i=0}^{7} r_{i2}x^i y^2$.

(d) el polinomio de permutación

$$\begin{aligned} p_8(u) &= u(1+x^2+x^3+x^4+x^6) \\ &\quad +u^2(1+x+x^2+x^6+x^7) \\ &\quad +u^4(1+x+x^4+x^5+x^6+x^7) \\ &\quad +u^8(x^3+x^6)+u^{16}(1+x+x^4+x^6) \\ &\quad +u^{32}(1+x+x^2+x^3+x^4+x^6) \\ &\quad +u^{64}(x^2+x^3+x^4+x^6)+u^{128}(1+x^3+x^4+x^5+x^6) \end{aligned}$$

al polinomio $\sum_{i=0}^{7} r_{i3}x^i y^3$.

Para apreciar la aplicación de todos estos polinomios se implementó la rutina **Pperm** del ápendice A.

3.1.4 Las cajas de substitución de DES

Ahora se describirá, en forma polinomial, una de las partes medulares del sistema de cifrado DES, las cajas de substitución. Para cada caja de substitución se calcularon, usando *Mathematica v5.0*, cuatro polinomios en $\mathbb{F}_{2^4}$. Cada polinomio representa a un renglón en la tabla de valores de una S-caja. Por ejemplo, la caja $S1$ (Figura 3.1) está representada por los siguientes cuatro polinomios:

$$\begin{aligned} s1_1(v) &= x+x^2+x^3+v^3x^3+v^7(1+x^2)+v^{14}(1+x^2) \\ &\quad +v^9(x+x^2)+v(1+x+x^2)+v^4(1+x+x^2)+v^{10}(1+x+x^2) \\ &\quad +v^{13}(1+x+x^2)+v^6(1+x^3)+v^2(x+x^3)+v^5(x+x^3) \\ &\quad +v^8(x^2+x^3)+v^{12}(x+x^2+x^3)+v^{11}(1+x+x^2+x^3) \\ s1_2(v) &= v+v^5+v^8x+v^{13}x+v^7x^3+v^9(1+x)+v^2(1+x^2) \\ &\quad +v^3(1+x^2)+v^6(1+x^3)+v^{10}(1+x^3)+v^{14}(1+x+x^3) \\ &\quad +v^{12}(x^2+x^3)+v^{11}(1+x^2+x^3)+v^4(x+x^2+x^3) \\ s1_3(v) &= x^2+v^3x^2+v^4x^2+v^5x^2+v^{10}x^2+v^{14}x^3 \\ &\quad +v^{11}(1+x^2)+v^7(1+x+x^2)+v^6(1+x^3)+v^{13}(1+x+x^3) \\ &\quad +v(x^2+x^3)+v^9(x+x^2+x^3)+v^{12}(1+x+x^2+x^3) \\ s1_4(v) &= 1+x+v^5x+v^{14}x+x^2+x^3 \\ &\quad +v^{12}(1+x)+v^9(x+x^2)+v^2(1+x^3)+v^8(1+x^3) \\ &\quad +v(1+x+x^3)+v^7(x^2+x^3)+v^{11}(x^2+x^3)+v^4(1+x^2+x^3) \end{aligned}$$

La correspondencia entre la tabla de la Figura 3.1 y los polinomios anteriores es la siguiente. Cada S-caja tiene 16 columnas que se enumeran del 0 al 15, usando la representación binaria de estos números, pueden convertirse en un polinomio de grado a lo más 3, esto es, un elemento del campo finito $\mathbb{F}_{2^4}$. Por lo que cada columna queda representada por un elemento de $\mathbb{F}_{2^4}$. A su vez, se hace lo mismo para los renglones de la S-caja y lo que se obtiene es un mapeo específico para cada renglón. Nuevamente, el mapeo se calcula con la fórmula de interpolación de Lagrange.

S_1	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
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

Figura 3.1: Caja de substitución S1.

En el algoritmo de cifrado DES, las S-cajas se aplican a cadenas de 6 bits produciendo grupos de 4 bits. En este caso, sucede algo similar pues se comienza con polinomios de grado a lo más 5 para obtener al final polinomios de grado a lo más 3. Se recordará que para una cadena de 6 bits $a_0a_1a_2a_3a_4a_5$ los bits a_0a_5 determinan el renglón de la S-caja, en tanto que los bits $a_1a_2a_3a_4$ definen la columna de la misma S-caja. Para lograr esto en un polinomio inicial de grado a lo más 5, se emplea un polinomio de permutación auxiliar $sp(t)$ que permutará los coeficientes del polinomio de la forma $a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5$.

$$\begin{aligned} sp(t) &= t(x^3 + x^5) + t^2(1 + x^2 + x^3) + t^4(x + x^2 + x^3) \\ &\quad + t^8(x + x^3 + x^4) + t^{16}(1 + x^3 + x^5) + t^{32}x^3 \end{aligned}$$

Cuando se aplica $sp(t)$ a un polinomio de grado a lo más 5, se obtiene:

$$sp(a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5) = a_1 + a_2x + a_3x^2 + a_4x^3 + a_0x^4 + a_5x^5.$$

Este reordenamiento de los coeficientes del polinomio permite separarlo en dos partes, rescatando los 4 bits que definen la columna de la S-caja como la primera parte del polinomio. Mientras que los 2 bits que determinan el renglón en la S-caja quedan ubicados en la segunda parte del mismo polinomio. Después de ubicar el renglón en la S-caja, la segunda parte del polinomio debe descartarse para obtener un polinomio de grado a lo más 4 al cual aplicarle el polinomio correspondiente de

la S-caja. Para ilustrar lo anterior, supóngase que se desea aplicar la caja S1 al polinomio $x + x^2$. Primero, debe aplicarse $sp(t)$:

$$sp(x + x^2) = 1 + x.$$

Nótese que la segunda parte del polinomio es 0, esto indica que se aplicará el polinomio $s1_1(v)$ al polinomio $1 + x$:

$$s1_1(1 + x) = x + x^3.$$

Los polinomios para el resto de las cajas de substitución aparecen en el apéndice A y la rutina **aplicaSboxes** facilita la aplicación de las S-cajas a un polinomio determinado.

3.1.5 La función tipo Feistel de DES

Como se mencionó en la sección anterior, la función tipo Feistel de DES consiste en cuatro operaciones que se aplican al lado derecho de un bloque de 64 bits. Para describir esta función de una manera algebraica, se requiere distinguir los polinomios que representan cada lado del bloque escribiendo el polinomio $m(x, y)$ de la siguiente manera:

$$m(x, y) = \sum_{i=0}^{7} \sum_{j=0}^{3} m_{ij} x^i y^j + y^4 \sum_{i=0}^{7} \sum_{j=0}^{3} m_{ij} x^i y^j = l(x, y) + y^4 r(x, y)$$

La función de Feistel comienza aplicando la permutación E al polinomio $r(x, y)$, esto es, $e(x, y) = E(r(x, y))$. Después, este polinomio se combina con la llave correspondiente $k(x, y)$ formando un nuevo polinomio. Es decir, $q(x, y) = e(x, y) + k(x, y)$. El proceso continúa con las cajas de substitución. Esto se puede ilustrar con la definición del siguiente polinomio:

$$s(x, y) = S(q(x, y)) = \sum_{j=0}^{7} S_j\big(\sum_{i=0}^{5} q_{ij} x^i\big) y^j = \sum_{j=0}^{7} \big(\sum_{i=0}^{3} s_{ij} x^i\big) y^j$$

Finalmente, la permutación P aplicada al polinomio $s(x, y)$ produce un polinomio $l(x, y) = P(s(x, y))$ que formará la parte izquierda de un nuevo bloque en la siguiente iteración del algoritmo. Estas 4 operaciones definen la función de Feistel $f(r_{i-1}(x, y), k_i(x, y))$.

Para ilustrar la función f, supóngase que $r_0(x,y)$ es el polinomio (3.4), esto es,

$$\begin{aligned} r_0(x,y) &= x+x^2+x^4+x^6+x^7+(1+x^2+x^3+x^4+x^5)y \\ &\quad +(1+x+x^4+x^7)y^2 \\ &\quad +(1+x+x^2+x^3+x^4+x^5+x^6+x^7)y^3 \end{aligned}$$

y $k_1(x,y)$ es la primera subllave generada a partir de (3.2), es decir,

$$\begin{aligned} k_1(x,y) &= (x^3+x^4)+(1+x)y+(x^2+x^4+x^5)y^2+(1+x^2+x^3+x^4+x^5)y^3 \\ &\quad +(1+x+x^2+x^3+x^4+x^5)y^4+(x^3+x^4+x^5)y^5+x^5y^6 \\ &\quad +(1+x+x^4)y^7. \end{aligned}$$

A partir de $r_0(x,y)$ se obtiene $e(x,y)$:

$$\begin{aligned} e(x,y) &:= 1+x^2+x^3+x^5+(x+x^3+x^4+x^5)y \\ &\quad +(1+x+x^3+x^4+x^5)y^2+(1+x+x^2+x^5)y^3 \\ &\quad +(x+x^2+x^5)y^4+(x+x^4+x^5)y^5 \\ &\quad +(1+x+x^2+x^3+x^4+x^5)y^6+(1+x+x^2+x^3+x^4)y^7. \end{aligned}$$

Luego, el polinomio $q(x,y)$ resulta de sumar $k_1(x,y)$ con $e(x,y)$:

$$\begin{aligned} q(x,y) &:= 1+x^2+x^4+x^5+(1+x^3+x^4+x^5)y \\ &\quad +(1+x+x^2+x^3)y^2+(x+x^3+x^4)y^3 \\ &\quad +(1+x^3+x^4)y^4+(x+x^3)y^5 \\ &\quad +(1+x+x^2+x^3+x^4)y^6+(x^2+x^3)y^7. \end{aligned}$$

Ahora, se aplican las S-cajas a los 8 polinomios que integran $q(x,y)$:

$$\begin{aligned} s(x,y) &:= S_1(1+x^2+x^4+x^5)+S_2(1+x^3+x^4+x^5)y \\ &\quad +S_3(1+x+x^2+x^3)y^2+S_4(x+x^3+x^4)y^3 \\ &\quad +S_5(1+x^3+x^4)y^4+S_6(x+x^3)y^5 \\ &\quad +S_7(1+x+x^2+x^3+x^4)y^6+S_8(x^2+x^3)y^7 \end{aligned}$$

Lo que genera:

$$\begin{aligned} s(x,y) &:= (1+x^3)+x^3y \\ &\quad +(1+x+x^2)y^2+(x+x^3)y^3 \\ &\quad +(1+x^2+x^3)y^4+(x^2+x^3)y^5 \\ &\quad +x^2y^6+(1+x^2+x^3)y^7 \end{aligned}$$

Finalmente, el último paso de la función f es aplicar la permutación P. Sin embargo, esta permutación requiere un elemento en $\mathcal{A}_{8,4}$ por lo que se usará un cambio de variable en $s(x,y)$ antes de aplicar P. Ahora, se usa $y^i = y^{i/2}$ si i es par y $y^i = x^4 y^{(i-1)/2}$ si i es impar:

$$\begin{aligned} s'(x,y) &:= 1 + x^3 + x^7 \\ &\quad +(1 + x + x^2 + x^5 + x^7)y \\ &\quad +(1 + x^2 + x^3 + x^6 + x^7)y^2 \\ &\quad +(x^2 + x^4 + x^6 + x^7)y^3 \end{aligned}$$

Al aplicar P a $s'(x,y)$, se obtiene:

$$\begin{aligned} p(x,y) &:= 1 + x^2 + x^4 + x^7 \\ &\quad +(1 + x^2 + x^6 + x^7)y \\ &\quad +(x + x^2 + x^3 + x^4 + x^5 + x^7)y^2 \\ &\quad +(1 + x^5 + x^6)y^3 \end{aligned}$$

Este polinomio es el resultado final de la función Feistel. La rutina **funcionF** aplica todas las operaciones descritas en esta sección y proporciona, como resultado, el polinomio $p(x,y)$.

3.1.6 Las rondas del estándar DES

Una vez descrita la función de Feistel es posible describir las rondas del algoritmo de cifrado. En esta descripción, cada ronda define dos polinomios, $l_i(x,y)$ y $r_i(x,y)$, los cuales conforman el nuevo bloque $m_i(x,y)$ dado por $m_i(x,y) = l_i(x,y) + y^4 r_i(x,y)$. Tomando en cuenta que un polinomio $m(x,y)$ puede expresarse como

$$m(x,y) = \sum_{i=0}^{7}\sum_{j=0}^{3} m_{ij}x^i y^j + y^4 \sum_{i=0}^{7}\sum_{j=0}^{3} m_{ij}x^i y^j = l_0(x,y) + y^4 r_0(x,y),$$

definimos una ronda por

$$\begin{aligned} r_i(x,y) &:= l_{i-1}(x,y) + f(r_{i-1}(x,y), k_i(x,y)), \qquad \text{para } i = 1, ..., 16; \\ l_i(x,y) &:= r_{i-1}(x,y). \end{aligned}$$

El sistema de cifrado DES consta de 16 rondas a partir de un bloque $m(x,y)$ y las 16 subllaves $k_i(x,y)$ formadas en el proceso de generación de las subllaves.

3.1.7 Permutación PC_1

La permutación PC_1 es la primera en aplicarse cuando se generan las subllaves que se usarán en el cifrado. La idea principal para obtener una representación polinomial de esta permutación es que a partir de un bloque ordenado de 64 bits, pero representado por polinomios, se logre la misma configuración que tiene la tabla PC_1 usando operaciones válidas del anillo $\mathcal{A}_{8,8}$ y del campo finito $\mathbb{F}_{2^8}$.

Considerermos los 64 bits de un bloque, ordenados y presentados en un arreglo rectangular como el de la tabla (a) en la Figura 3.2. El objetivo es que mediante desplazmientos de los bits de un renglón o columna y permutaciones de los bits de un renglón o columna logremos la configuración la tabla (b) de la misma Figura. Esta última tabla contiene los bits en el orden en que aparecen en la tabla de permutación PC_1 (ver Figura 2.7) pero mantiene a los bits que son múltiplos de 8.

Bloque Inicial							
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	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

(a)

Bloque Final							
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	32	24	16	8
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	64	56	48	40

(b)

Figura 3.2: Bloque inicial de 64 bits y bloque final de 64 bits.

El sistema DES utiliza una llave de 64 bits pero como cada octavo bit no se usa para cifrar, éstos siempre se quitan. En la representación polinomial, la llave es un elemento $k(x,y) \in \mathcal{A}_{8,8}$ y los bits que ocupan una posición múltiplo de 8 son los coeficientes de x^7. Para que el polinomio $k(x,y)$ continúe como elemento del anillo $\mathcal{A}_{8,8}$, sólo haremos ceros los coeficientes de x^7. Esto se expresa como sigue:

$$\tilde{k}(x,y) = k(x,y) - \sum_{j=0}^{7} k_{7j} x^7 y^j.$$

En particular, el polinomio (3.2) del ejemplo cambia a:

$$\begin{aligned}\tilde{k}(x,y) &= x^3+x^6+(x^2+x^3+x^5)y+(x+x^3+x^5+x^6)y^2 \\ &\quad +(x+x^2+x^3+x^4)y^3+(1+x^3+x^4+x^6)y^4 \\ &\quad +(1+x^2+x^3+x^4+x^5)y^5+(1+x+x^3+x^4+x^5+x^6)y^6 \\ &\quad +(1+x+x^2+x^3)y^7 \end{aligned} \tag{3.6}$$

Después de remover los 8 términos, el polinomio $\tilde{k}(x,y)$ se somete a un conjunto de transformaciones para lograr el efecto de la permutación PC_1 (Figura 2.7).

Primero aplicamos la transformación π, definida en la sección 3.1.1, obteniendo:

$$\tilde{k} = \tilde{k}(x,y) \xrightarrow{\pi} \sum_{i=0}^{7}\sum_{j=0}^{7} k_{ij}x^j y^i.$$

Esta transformación tiene el efecto de transponer los elementos del polinomio $\tilde{k}(x,y)$. En el polinomio (3.6), el efecto se ve de la siguiente manera:

$$\begin{aligned}\tilde{k}^{(1)}(x,y) &= x^4+x^5+x^6+x^7+(x^2+x^3+x^6+x^7)y+(x+x^3+x^5+x^7)y^2 \\ &\quad +(1+x+x^2+x^3+x^4+x^5+x^6+x^7)y^3+(x^3+x^4+x^5+x^6)y^4 \\ &\quad +(x+x^2+x^5+x^6)y^5+(1+x^2+x^4+x^6)y^6 \end{aligned} \tag{3.7}$$

Luego, se aplica la transformación g, definida también en la sección 3.1.1, y se obtiene:

$$\sum_{i=0}^{7}\sum_{j=0}^{7} k_{ij}y^i x^j \xrightarrow{g} \sum_{i=0}^{7}(x^7\sum_{j=0}^{7} k_{ij}x^{7j})y^i.$$

Esta transformación cambia el orden de derecha a izquierda en el que se ubican los bits en cada grupo. Esto es, el primer bit de izquierda a derecha es ahora el primero de derecha a izquierda y así, consecutivamente. Por ejemplo, después de aplicarla a (3.7), se obtiene:

$$\begin{aligned}\tilde{k}^{(2)}(x,y) &= 1+x+x^2+x^3+(1+x+x^4+x^5)y+(1+x^2+x^4+x^6)y^2 \\ &\quad +(1+x+x^2+x^3+x^4+x^5+x^6+x^7)y^3+(x+x^2+x^3+x^4)y^4 \\ &\quad +(x+x^2+x^5+x^6)y^5+(x+x^3+x^5+x^7)y^6 \end{aligned} \tag{3.8}$$

Después, se aplicarán 3 polinomios de permutación que reacomodarán los términos de algunos polinomios. Más específicamente, lo que se busca con estos polinomios

es intercambiar algunas posiciones de los bits de algún renglón o alguna columna en particular que no pueden modificarse con productos monomiales. Pero como estos renglones o columnas son polinomios de grado a lo más siete, se necesita una operación válida en el campo finito $\mathbb{F}_{2^8}$.

En este caso, es conveniente observar que un polinomio $p(x,y) \in \mathcal{A}_{8,8}$ tiene 8 polinomios de grado a lo más 7 en la variable x y 8 polinomios de grado a lo más 7 en la variable y. En ambos casos, estos polinomios son elementos del mismo campo $\mathbb{F}_{2^8}$. El objetivo es reacomodar los términos de algunos de estos polinomios en $\mathbb{F}_{2^8}$ usando polinomios de permutación y ubicarlos en el orden en el que aparecen en PC_1.

Para calcular los polinomios de permutación, nuevamente usaremos la fórmula (3.5). Consideremos un polinomio $q(y) = a_0 + a_1y + a_2y^2 + a_3y^3 + a_4y^4 + a_5y^5 + a_6y^6 + a_7y^7$. Para el primer polinomio de permutación, $pc1_1(u) \in \mathbb{F}_{2^8}[u]$, deseamos cambiar de posición el coeficiente que acompaña a y^7 y ubicarlo como el coeficiente de y^4. También, se pretende que el coeficiente de y^4 sea ahora el coeficiente de y^5, el coeficiente de y^5 lo sea de y^6 y el coeficiente de y^6 se convierta en el coeficiente de y^7. Definimos entonces $\tau_1 : \mathbb{F}_{2^8} \to \mathbb{F}_{2^8}$ como una función de $\mathbb{F}_{2^8}$ en $\mathbb{F}_{2^8}$ tal que

$$\tau_1(q(y)) = a_0 + a_1y + a_2y^2 + a_3y^3 + a_7y^4 + a_4y^5 + a_5y^6 + a_6y^7$$

para todo polinomio en $\mathbb{F}_{2^8}$. Para el segundo polinomio de permutación, $pc1_2(u) \in \mathbb{F}_{2^8}[u]$, queremos cambiar de posición el coeficiente que acompaña a y^7 y ubicarlo como el coeficiente de y^3. También, deseamos que el coeficiente de y^3 sea ahora el coeficiente de y^4, el coeficiente de y^4 lo sea de y^5, el coeficiente de y^5 lo sea de y^6 y el coeficiente de y^6 se convierta en el coeficiente de y^7. Entonces la función $\tau_2 : \mathbb{F}_{2^8} \to \mathbb{F}_{2^8}$ está dada por:

$$\tau_2(q(y)) = a_0 + a_1y + a_2y^2 + a_7y^3 + a_3y^4 + a_4y^5 + a_5y^6 + a_6y^7.$$

Ambos polinomios se calcularon usando *Mathematica v5.0*. Para ello, se especificó tanto el dominio e imagen de la función τ mediante listas y, después, se ejecutó la suma para los elementos dichas listas según (3.6). Por ejemplo, para la función τ_1 del polinomio $pc1_1(u)$ se definieron

$$D = \{0, 1, y, y^2, y^3, y^4, y^5, y^6, y^7, ..., 1 + y + y^2 + y^3 + y^4 + y^5 + y^6 + y^7\}$$

y

$$I = \{0, 1, y, y^2, y^3, y^5, y^6, y^7, y^4, ..., 1 + y + y^2 + y^3 + y^4 + y^5 + y^6 + y^7\}$$

como dominio e imagen de τ_1, respectivamente. Mientras que para τ_2 se usaron

$$D = \{0, 1, y, y^2, y^3, y^4, y^5, y^6, y^7, ..., 1 + y + y^2 + y^3 + y^4 + y^5 + y^6 + y^7\}$$

y

$$I = \{0, 1, y, y^2, y^4, y^5, y^6, y^7, y^3, ..., 1 + y + y^2 + y^3 + y^4 + y^5 + y^6 + y^7\}$$

como dominio e imagen de τ_2, respectivamente. Los polinomios obtenidos son:

$$\begin{aligned} pc1_1(u) &= (1 + y^2 + y^6)u + (1 + y^3 + y^4 + y^7)u^2 + (1 + y^2 + y^3 + y^4)u^4 \\ &\quad +(1 + y^4 + y^6)u^8 + (1 + y + y^5 + y^6 + y^7)u^{16} \\ &\quad +(y + y^5 + y^6 + y^7)u^{32} + (y + y^3 + y^4 + y^6 + y^7)u^{64} \\ &\quad +(y + y^3 + y^6)u^{128} \end{aligned}$$

$$\begin{aligned} pc1_2(u) &= (1 + y^2 + y^5)u + y^2u^2 + (y^3 + y^4 + y^5)u^4 \\ &\quad +(1 + y + y^2 + y^3 + y^4 + y^5 + y^7)u^8 + (y + y^3 + y^4 + y^6)u^{16} \\ &\quad +(1 + y^3 + y^4 + y^5 + y^7)u^{32} + (y + y^2 + y^3 + y^4 + y^5)u^{64} \\ &\quad +(y + y^3 + y^4 + y^5 + y^6)u^{128} \end{aligned}$$

El polinomio $pc1_1(u)$ se aplica a los primeros cuatro polinomios en la variable y del polinomio $\tilde{k}(x, y) \in \mathcal{A}_{8,8}$. Mientras que el polinomio, $pc1_2(u)$ se aplica a los cuatro polinomios restantes. Esto significa que se está actuando sobre las columnas de la tabla.

Al considerar los polinomios en la variable y, el polinomio (3.8) del ejemplo, se reescribe como sigue:

$$\begin{aligned} \tilde{k}^{(2)}(x, y) &= 1 + y + y^2 + y^3 + (1 + y + y^3 + y^4 + y^5 + y^6)x \\ &\quad +(1 + y^2 + y^3 + y^4 + y^5)x^2 + (1 + y^3 + y^4 + y^6)x^3 \\ &\quad +(y + y^2 + y^3 + y^4)x^4 + (y + y^3 + y^5 + y^6)x^5 \\ &\quad +(y^2 + y^3 + y^5)x^6 + (y^3 + y^6)x^7 \end{aligned} \tag{3.9}$$

Aplicar los polinomios $pc1_1(u)$ y $pc1_2(u)$ a $\tilde{k}^{(2)}(x, y)$, se puede expresar como:

$$k'(x, y) = \sum_{j=0}^{3} pc1_1(\sum_{i=0}^{7} k_{ij}y^i)x^{7j+7} + \sum_{j=4}^{7} pc1_2(\sum_{i=0}^{7} k_{ij}y^i)x^{7j+7}.$$

El efecto que logran estos dos primeros polinomios es cambiar de lugar el coeficiente del último término. En el caso de $pc1_1(u)$, lo coloca en la cuarta potencia de

y y recorre los demás coeficientes en una potencia. Para el caso de $pc1_2(u)$, el coeficiente del último término se va hasta la tercera potencia de y recorriendo el resto de los coeficientes en una potencia. El resultado final de aplicar estos polinomios a (3.9) se ve como:

$$\begin{aligned} k'(x,y) &= 1+y+y^2+y^3+(1+y+y^3+y^5+y^6+y^7)x \\ &\quad +(1+y^2+y^3+y^5+y^6)x^2+(1+y^3+y^5+y^7)x^3 \\ &\quad +(y+y^2+y^4+y^5)x^4+(y+y^4+y^6+y^7)x^5 \\ &\quad +(y^2+y^4+y^6)x^6+(y^4+y^7)x^7 \end{aligned} \tag{3.10}$$

En seguida, se reacomodan los términos del quinto polinomio en x, $\sum_{j=0}^{7} k'_{4j}x^j y^4$. Lo que se busca es que los últimos cuatro coeficientes de este polinomio queden como los primeros cuatro y los primeros cuatro se ubiquen a partir de la cuarta potencia de x:

$$k''(x,y)=\sum_{i=0}^{3}(\sum_{j=0}^{7}k'_{ij}x^j)y^i+(\sum_{j=0}^{7}k'_{4j}x^{j+4})y^4+\sum_{i=5}^{7}(\sum_{j=0}^{7}k'_{ij}x^j)y^i.$$

Con esta transformación, el polinomio (3.10) queda como:

$$\begin{aligned} k''(x,y) &= (1+x+x^2+x^3)+(1+x+x^4+x^5)y \\ &\quad +(1+x^2+x^4+x^6)y^2+(1+x+x^2+x^3)y^3 \\ &\quad +(1+x+x^2+x^3)y^4+(x+x^2+x^3+x^4)y^5 \\ &\quad +(x+x^2+x^5+x^6)y^6+(x+x^3+x^5+x^7)y^7 \end{aligned} \tag{3.11}$$

Considérese ahora el polinomio $pc1_3(u)$ definido como:

$$\begin{aligned} pc1_3(u) &= (y^3+y^6)u+(1+y+y^3)u^2+(1+y^2+y^6+y^7)u^4 \\ &\quad +(1+y+y^4+y^6+y^7)u^8+(y+y^3+y^5+y^6)u^{16} \\ &\quad +(y^2+y^4+y^6)u^{32}+(y+y^2+y^7)u^{64} \\ &\quad +(1+y+y^4+y^6+y^7)u^{128}. \end{aligned}$$

Este polinomio se aplica a todos los polinomios en la variable y del polinomio $k''(x,y)$. El objetivo es cambiar de posición los polinomios que se encuentran a partir de la cuarta potencia de y. Como último paso, se requiere que el polinomio que se encuentra como coeficiente de y^7 sea ahora el coeficiente de y^4, el de y^6 sea el

coeficiente de y^5, el de y^5 sea y^6 y, finalmente, que y^4 sea y^8. Este paso determina el polinomio $k^*(x, y)$:

$$k^*(x,y) = \sum_{i=0}^{7} pc1_3(\sum_{j=0}^{7} k''_{ij}x^j)y^i.$$

Finalmente, el resultado de aplicar $pc1_3(u)$ al polinomio (3.11) del ejemplo, es el siguiente:

$$\begin{aligned} k^*(x,y) &= 1+x+x^2+x^3+(1+x+x^4+x^5)y \\ &\quad +(1+x^2+x^4+x^6)y^2+(1+x+x^2+x^3)y^3 \\ &\quad +(x+x^3+x^5+x^7)y^4+(x+x^2+x^5+x^6)y^5 \\ &\quad +(x+x^2+x^3+x^4)y^6+(1+x+x^2+x^3)y^7 \end{aligned} \tag{3.12}$$

Este polinomio es el equivalente al resultado obtenido después de aplicar la permutación PC_1 a la cadena de bits k del ejemplo de la sección anterior:

$$\begin{aligned} k^* &= 1111000\ 0110011\ 0010101\ 0101111 \\ &\quad 0101010\ 1011001\ 1001111\ 0001111. \end{aligned}$$

La rutina **PC1perm** del apéndice A implementa estas transformaciones y permite calcular el último polinomio.

3.1.8 Permutación $\mathbf{PC_2}$

Por otro parte, el último paso en la generación de las llaves es la permutación PC_2. Esta permutación se divide en dos partes, cada una de ellas se aplica a un elemento del anillo

$$\mathcal{A}_{7,4} := \frac{\mathbb{F}_2[x,y]}{\langle x^7-1, y^4-1\rangle}.$$

También, se aprovecharán las estructuras de los campos finitos $\mathbb{F}_{2^4}$ y

$$\mathbb{F}_{2^7} := \frac{\mathbb{F}_2[w]}{\langle w^7+w+1\rangle}.$$

La idea general es usar tanto las características del anillo como las de los dos campos definidos para obtener la misma configuración de la tabla de permutación

PC_2. Recordemos que los números que aparecen en dicha tabla son las posiciones de los bits en el bloque, pero también son las posiciones de los coeficientes de un polinomio. Por esto, se desea ocupar un bloque ordenado y, mediante operaciones con polinomios en el anillo o en cualquiera de los campos, configurarlo en la forma de PC_2.

Las operaciones de polinomios que se requieren, del anillo $\mathcal{A}_{7,4}$, son productos con monomios de la forma v^i o w^i. Estas multiplicaciones representan simplemente i desplazmientos de los bits de un renglón o de una columna. Por otra parte, de los campos $\mathbb{F}_{2^4}$ y $\mathbb{F}_{2^7}$ se necesitan polinomios de permutación que intercambien las posiciones de los bits de un renglón o una columna. Los polinomios de permutación que se presentan en seguida fueron calculados con la fórmula de interpolación de Lagrange para campos finitos.

Debido a la complejidad en el cálculo e implementación de los polinomios de permutación, se determinó usar la menor cantidad de polinomios de permutación y la mayor cantidad posible de productos monomiales. A continuación se describen los nueve pasos que forman la primera parte de la permutación:

1. Multiplicar $y(\sum_{j=0}^{3} r_{0j}y^j)$.
2. Multiplicar:
   (a) $x^6\big(\sum_{i=0}^{6} r_{i0}x^i\big)$.
   (b) $x^5\big(\sum_{i=0}^{6} r_{i1}x^i y\big)$.
   (c) $x^3\big(\sum_{i=0}^{6} r_{i2}x^i y^2\big)$.
   (d) $x^3\big(\sum_{i=0}^{6} r_{i3}x^i y^3\big)$.
3. Aplicar:
   (a) el polinomio de permutación
   $$pc2A_1(v) = v(1+y^2) + v^2 + v^4(y+y^2+y^3) + v^8y^3$$
   al polinomio $\sum_{j=0}^{3} r_{1j}xy^j$.
   (b) el polinomio de permutación
   $$pc2A_2(v) = v(y+y^2) + v^2 + v^4(1+y+y^2) + v^8$$
   al polinomio $\sum_{j=0}^{3} r_{2j}x^2y^j$.

(c) el polinomio de permutación

$$pc2A_3(v) = v(1+y^2) + v^2(1+y+y^2) + v^4(y+y^2+y^3) + v^8(1+y^2+y^3)$$

al polinomio $\sum_{j=0}^{3} r_{3j}x^3y^j$.

(d) el polinomio de permutación

$$pc2A_4(v) = vy + v^2y^3 + v^4(1+y^2) + v^8(1+y^2+y^3)$$

al polinomio $\sum_{j=0}^{3} r_{4j}x^4y^j$.

(e) el polinomio de permutación

$$pc2A_5(v) = v(1+y^2) + v^2(y+y^2) + v^4(y+y^2+y^3) + v^8(1+y^2)$$

al polinomio $\sum_{j=0}^{3} r_{5j}x^5y^j$.

4. Multiplicar:
   (a) $x^4\big(\sum_{i=0}^{6} r_{i1}x^iy\big)$.
   (b) $x\big(\sum_{i=0}^{6} r_{i2}x^iy^2\big)$.
5. Aplicar el polinomio de permutación

$$pc2A_6(v) = v(1+y+y^2) + v^2(y+y^2) + v^4 + v^8y^2$$

al polinomio $\sum_{j=0}^{3} r_{2j}x^2y^j$.

6. Multiplicar:
   (a) $x^3\big(\sum_{i=0}^{6} r_{i1}x^iy\big)$.
   (b) $x\big(\sum_{i=0}^{6} r_{i2}x^iy^2\big)$.
7. Aplicar el polinomio de permutación

$$pc2A_7(v) = v(y+y^3) + v^2y^2 + v^4y^2 + v^8y$$

al polinomio $\sum_{j=0}^{3} r_{0j}y^j$.

8. Multiplicar $x^5\big(\sum_{i=0}^{6} r_{i2}x^iy^2\big)$.
9. Aplicar:

(a) el polinomio de permutación

$$\begin{aligned} pc2A_8(w) &= w(1+x+x^2+x^3+x^5)+w^2x^2 \\ &\quad +w^4(x^3+x^4+x^6)+w^8(x+x^3+x^4+x^5) \\ &\quad +w^{16}x^2+w^{32}(x+x^3+x^4+x^6) \\ &\quad +w^{64}(1+x^2+x^4) \end{aligned}$$

al polinomio $\sum_{i=0}^{6} r_{i0}x^i$.

(b) el polinomio de permutación

$$\begin{aligned} pc2A_9(w) &= w(1+x+x^2+x^5+x^6) \\ &\quad +w^2(1+x+x^2+x^4+x^5+x^6) \\ &\quad +w^4(x+x^2+x^3+x^5+x^6) \\ &\quad +w^8(x^4+x^5+x^6)+w^{16}(x+x^4) \\ &\quad +w^{32}(x+x^2+x^5)+w^{64}(x+x^3+x^4) \end{aligned}$$

al polinomio $\sum_{i=0}^{6} r_{i1}x^i y$.

(c) el polinomio de permutación

$$\begin{aligned} pc2A_{10}(w) &= w(x+x^3+x^6) \\ &\quad +w^2(1+x+x^3+x^5+x^6)+w^4(x+x^6) \\ &\quad +w^8(x+x^2+x^4+x^6) \\ &\quad +w^{16}(1+x^3+x^5+x^6) \\ &\quad +w^{32}(x+x^2+x^5)+w^{64}(x^3+x^4+x^5+x^6) \end{aligned}$$

al polinomio $\sum_{i=0}^{6} r_{i2}x^i y^2$.

(d) el polinomio de permutación

$$\begin{aligned} pc2A_{11}(w) &= w(1+x^2+x^5)+w^2(x^5+x^6) \\ &\quad +w^4(1+x+x^3+x^6)+w^8(1+x^6) \\ &\quad +w^{16}(x+x^2+x^3+x^4+x^6)+w^{32}(x+x^2) \\ &\quad +w^{64}(1+x+x^2+x^4+x^5) \end{aligned}$$

al polinomio $\sum_{i=0}^{6} r_{i3}x^i y^3$.

La segunda parte de la permutación consiste de 12 pasos:

1. Multiplicar $y^3\big(\sum_{j=0}^{3} r_{0j}y^j\big)$.
2. Multiplicar:
   (a) $x\big(\sum_{i=0}^{6} r_{i0}x^i\big)$.
   (b) $x\big(\sum_{i=0}^{6} r_{i2}x^iy^2\big)$.
3. Multiplicar:
   (a) $y^3\big(\sum_{j=0}^{3} r_{1j}xy^j\big)$.
   (b) $y\big(\sum_{j=0}^{3} r_{2j}x^2y^j\big)$.
4. Multiplicar:
   (a) $x^6\big(\sum_{i=0}^{6} r_{i0}x^i\big)$.
   (b) $x^6\big(\sum_{i=0}^{6} r_{i1}x^iy\big)$.
   (c) $x^4\big(\sum_{i=0}^{6} r_{i2}x^iy^2\big)$.
   (d) $x^2\big(\sum_{i=0}^{6} r_{i3}x^iy^3\big)$.
5. Aplicar:
   (a) el polinomio de permutación
   $$pc2B_1(v) = v(1+y^2) + v^2 + v^4(y+y^2+y^3) + v^8y^3$$
   al polinomio $\sum_{j=0}^{3} r_{4j}x^4y^j$.
   (b) el polinomio de permutación
   $$pc2B_2(v) = v(1+y+y^2+y^3) + v^2(y+y^3) + v^4y + v^8(y+y^2)$$
   al polinomio $\sum_{j=0}^{3} r_{5j}x^5y^j$.
6. Multiplicar $x\big(\sum_{i=0}^{6} r_{i1}x^iy\big)$.
7. Aplicar:
   (a) el polinomio de permutación
   $$pc2B_3(v) = v(y+y^2) + v^2 + v^4(1+y+y^2) + v^8$$
   al polinomio $\sum_{j=0}^{3} r_{1j}xy^j$.

(b) el polinomio de permutación

$$pc2B_4(v) = v(y + y^2) + v^2(1 + y^3) + v^4(1 + y + y^3) + v^8(1 + y^2)$$

al polinomio $\sum_{j=0}^{3} r_{3j}x^3y^j$.

8. Multiplicar:

(a) $x^6\big(\sum_{i=0}^{6} r_{i1}x^iy\big)$.

(b) $x^3\big(\sum_{i=0}^{6} r_{i3}x^iy^3\big)$.

9. Aplicar:

(a) el polinomio de permutación

$$pc2B_5(v) = v(1 + y + y^3) + v^2(1 + y^2) + v^8(1 + y + y^2 + y^3)$$

al polinomio $\sum_{j=0}^{3} r_{0j}y^j$.

(b) el polinomio de permutación

$$pc2B_6(v) = v(1 + y^2) + v^2(y + y^2) + v^4(y + y^2 + y^3) + v^8(1 + y^2)$$

al polinomio $\sum_{j=0}^{3} r_{3j}x^3y^j$.

10. Multiplicar $x^3\big(\sum_{i=0}^{6} r_{i2}x^iy^2\big)$.

11. Aplicar el polinomio de permutación

$$pc2B_7(v) = v^2(1 + y + y^3) + v^4(y^2 + y^3) + v^8(1 + y)$$

al polinomio $\sum_{j=0}^{3} r_{5j}x^5y^j$.

12. Aplicar:

(a) el polinomio de permutación

$$\begin{aligned} pc2B_8(w) \;=\; & w(1 + x + x^2 + x^3 + x^6) + w^2(x^3 + x^5) \\ & + w^4(x + x^4 + x^5 + x^6) \\ & + w^8(1 + x^2 + x^4 + x^6) \\ & + w^{16}(x^3 + x^5) + w^{32}(x + x^6) \\ & + w^{64}(x + x^5) \end{aligned}$$

al polinomio $\sum_{i=0}^{6} r_{i0}x^i$.

(b) el polinomio de permutación

$$\begin{aligned} pc2B_9(w) &= w(x^5+x^6)+w^2(x^4+x^5) \\ &\quad +w^4(1+x+x^2+x^3+x^4+x^5) \\ &\quad +w^8(1+x^3+x^4+x^5+x^6) \\ &\quad +w^{32}(1+x+x^3+x^4+x^6)+w^{64}(1+x^5+x^6) \end{aligned}$$

al polinomio $\sum_{i=0}^{6} r_{i1}x^i y$.

(c) el polinomio de permutación

$$\begin{aligned} pc2B_{10}(w) &= w(x+x^3+x^4)+w^2(x+x^5) \\ &\quad +w^4(1+x+x^3+x^4+x^5+x^6)+w^8(x^2+x^3+x^4) \\ &\quad +w^{16}(x^2+x^3+x^4+x^5+x^6) \\ &\quad +w^{32}(1+x)+w^{64}(x+x^5) \end{aligned}$$

al polinomio $\sum_{i=0}^{6} r_{i2}x^i y^2$.

(d) el polinomio de permutación

$$\begin{aligned} pc2B_{11}(w) &= w(1+x+x^5+x^6) \\ &\quad +w^2(x+x^2+x^4+x^5+x^6) \\ &\quad +w^4(x+x^2+x^3+x^4) \\ &\quad +w^8(1+x+x^2+x^4+x^5+x^6) \\ &\quad +w^{16}(1+x^2+x^3+x^4+x^5+x^6) \\ &\quad +w^{32}(1+x^3+x^5+x^6)+w^{64}(x^2+x^3+x^5+x^6) \end{aligned}$$

al polinomio $\sum_{i=0}^{6} r_{i3}x^i y^3$.

Las rutinas **PC2Aperm** y **PC2Bperm** del apéndice A integran estos polinomios y pueden usarse para apreciar el efecto de la permutación PC2.

3.1.9 Generación de las subllaves

Una vez aplicada la permutación PC_1 y obtener el polinomio $k^*(x,y)$, el siguiente paso del algoritmo es generar las subllaves. Para esto, se debe distinguir dos términos de este polinomio. Esto es, hay que expresarlo como la suma de los siguientes dos polinomios:

$$k^*(x,y) = c_0(x,y) + y^4 d_0(x,y) \text{ donde}$$

$$c_0(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{7} k^*_{ij} x^j y^i \qquad \text{y}$$

$$d_0(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{7} k^*_{ij} x^j y^i$$

donde $c_0(x,y)$ y $d_0(x,y)$ son elementos del anillo $\mathcal{A}_{8,4} := \mathbb{F}_2[x,y]/\langle x^8-1, y^4-1\rangle$. Defínase el siguiente mapeo ψ entre el anillo $\mathcal{A}_{8,4}$ y el anillo $\mathcal{A}_{32} := \mathbb{F}_2[z]/\langle z^{32}+1\rangle$ como:

$$\begin{aligned} \psi : \mathcal{A}_{8,4} &\rightarrow \mathcal{A}_{32} \\ \psi(\sum_{i=0}^{3}\sum_{j=0}^{7} p_{i,j} x^j y^i) &\rightarrow p(z) = \sum_{l=0}^{31} p_l z^l \end{aligned}$$

donde $l = 8i + j$. Dicho de otra manera, ψ es el cambio de variable $z^{8i+j} = x^j y^i$, usado para redefinir estos polinomios en términos de una sola variable. En principio, se obtienen polinomios de grado a lo más 31. Sin embargo, en el caso de los polinomios $c_0(x,y)$ y $d_0(x,y)$, el grado es a lo más 27 debido a que los coeficientes de los últimos cuatro términos siempre son cero. En consecuencia, estos polinomios también son elementos de $\mathcal{A}_{28} := \mathbb{F}_2[z]/\langle z^{28}+1\rangle$

$$c_0(z) = \sum_{i=0}^{3}\sum_{j=0}^{7} k^*_{ij} z^{8i+j}$$

$$d_0(z) = \sum_{i=0}^{3}\sum_{j=0}^{7} k^*_{ij} z^{8i+j}$$

En el caso del polinomio (3.12) del ejemplo, los dos nuevos polinomios definidos a partir de $k^*(x,y)$ son:

$$\begin{aligned} c_0(z) &= 1 + z + z^2 + z^3 + z^8 + z^9 + z^{12} + z^{13} \\ &\quad + z^{16} + z^{18} + z^{20} + z^{22} + z^{24} + z^{25} + z^{26} + z^{27} \end{aligned} \tag{3.13}$$

$$\begin{aligned} d_0(z) &= z + z^3 + z^5 + z^7 + z^9 + z^{10} + z^{13} + z^{14} \\ &\quad + z^{17} + z^{18} + z^{19} + z^{20} + z^{24} + z^{25} + z^{26} + z^{27} \end{aligned} \tag{3.14}$$

El paso siguiente es generar 16 polinomios a partir $c_0(z)$ y otros 16 a partir de $d_0(z)$ mediante desplazmientos hacia la izquierda y usando el hecho de que dichos

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$p_i(z)$	z^{27}	z^{27}	z^{26}	z^{26}	z^{26}	z^{26}	z^{26}	z^{26}	z^{27}	z^{26}	z^{26}	z^{26}	z^{26}	z^{26}	z^{26}	z^{27}

Figura 3.3: Polinomios para los desplazmientos hacia la izquierda.

polinomios son elementos del anillo $\mathcal{A}_{28}$. Este paso implica multiplicar estos polinomios por alguna potencia de z dada en la siguiente tabla:
De tal manera que se pueden definir los siguientes polinomios:

$$\begin{aligned} c_i(z) &= p_i(z)c_{i-1}(z) \\ d_i(z) &= p_i(z)d_{i-1}(z) \end{aligned}$$

para $i = 1, ..., 16$.

Así, de los polinomios (3.13) y (3.14) del mismo ejemplo, se pueden generar, digamos, los primeros tres pares de polinomios:

$$\begin{aligned} c_1(z) &= 1 + z + z^2 + z^7 + z^8 + z^{11} + z^{12} + z^{15} \\ &\quad + z^{17} + z^{19} + z^{21} + z^{23} + z^{24} + z^{25} + z^{26} + z^{27} \\ d_1(z) &= 1 + z^2 + z^4 + z^6 + z^8 + z^9 + z^{12} + z^{13} \\ &\quad + z^{16} + z^{17} + z^{18} + z^{19} + z^{23} + z^{24} + z^{25} + z^{26} \end{aligned}$$

$$\begin{aligned} c_2(z) &= 1 + z + z^6 + z^7 + z^{10} + z^{11} + z^{14} + z^{16} \\ &\quad + z^{18} + z^{20} + z^{22} + z^{23} + z^{24} + z^{25} + z^{26} + z^{27} \\ d_2(z) &= z + z^3 + z^5 + z^7 + z^8 + z^{11} + z^{12} + z^{15} \\ &\quad + z^{16} + z^{17} + z^{18} + z^{22} + z^{23} + z^{24} + z^{25} + z^{27} \end{aligned}$$

$$\begin{aligned} c_3(z) &= z^4 + z^5 + z^8 + z^9 + z^{12} + z^{14} + z^{16} + z^{18} \\ &\quad + z^{20} + z^{21} + z^{22} + z^{23} + z^{24} + z^{25} + z^{26} + z^{27} \\ d_3(z) &= z + z^3 + z^5 + z^6 + z^9 + z^{10} + z^{13} + z^{14} + z^{15} \\ &\quad + z^{16} + z^{20} + z^{21} + z^{22} + z^{23} + z^{25} + z^{27} \end{aligned}$$

Nuevamente, se vuelve a expresar los polinomios anteriores como polinomios en dos variables. En este caso, se define el mapeo $\tau : \mathcal{A}_{28} \to \mathcal{A}_{7,4}$ dado por:

$$\tau(\sum_{l=0}^{27} p_l z^l) = \sum_{i=0}^{3}\sum_{j=0}^{6} p_{i,j} x^j y^i$$

donde $l = 7i + j$.

Por ejemplo, los polinomios $c_1(z)$ y $d_1(z)$ quedan ahora como:

$$\begin{aligned} c_1(x,y) &= 1 + x + x^2 \\ &\quad +(1 + x + x^4 + x^5)y \\ &\quad +(x + x^3 + x^5)y^2 \\ &\quad +(1 + x^2 + x^3 + x^4 + x^5 + x^6)y^3 \end{aligned} \tag{3.15}$$

$$\begin{aligned} d_1(x,y) &= 1 + x^2 + x^4 + x^6 \\ &\quad +(x + x^2 + x^5 + x^6)y \\ &\quad +(x^2 + x^3 + x^4 + x^5)y^2 \\ &\quad +(x^2 + x^3 + x^4 + x^5)y^3 \end{aligned} \tag{3.16}$$

La rutina **Subllaves** se encarga de efectuar los desplazmientos descritos y de devolver los polinomios en dos variables. Con las subllaves expresadas como polinomios en dos variables se puede aplicar la permutación PC_2 que se compone de dos partes. La parte A se aplica a los polinomios c_i, mientras que la parte B se aplica a los polinomios d_i, produciendo 16 polinomios en cada parte. Esto es,

$$c_i^*(x,y) = PC_2A(c_i(x,y)) \text{ y } d_i^*(x,y) = PC_2B(d_i(x,y))$$

para $1 \leq i \leq 16$. Para el caso de los polinomios (3.15) y (3.16) del ejemplo, se tienen los siguientes polinomios después de aplicar PC_2:

$$\begin{aligned} c_1^*(x,y) &= x^3 + x^4 \\ &\quad +(1 + x)y \\ &\quad +(x^2 + x^4 + x^5)y^2 \\ &\quad +(1 + x^2 + x^3 + x^4 + x^5)y^3 \end{aligned} \tag{3.17}$$

$$\begin{aligned} d_1^*(x,y) &= 1 + x + x^2 + x^3 + x^4 + x^5 \\ &\quad +(x^3 + x^4 + x^5)y \\ &\quad +x^5y^2 \\ &\quad +(1 + x + x^4)y^3 \end{aligned} \tag{3.18}$$

Finalmente, las llaves que se usarán en el cifrado tienen la forma

$$k_i(x,y) = c_i^*(x,y) + y^4 d_i^*(x,y)$$

para $1 \leq i \leq 16$. Por ejemplo, la primera llave $k_1(x, y)$ queda formada de la siguiente manera:

$$\begin{aligned} k_1(x,y) &= (x^3+x^4)+(1+x)y \\ &\quad +(x^2+x^4+x^5)y^2+(1+x^2+x^3+x^4+x^5)y^3 \\ &\quad +(1+x+x^2+x^3+x^4+x^5)y^4+(x^3+x^4+x^5)y^5 \\ &\quad +x^5y^6+(1+x+x^4)y^7 \end{aligned} \tag{3.19}$$

La rutina **generaLlaves** complementa a la rutina **Subllaves** al incorporar la aplicación de las rutinas **PC2Aperm** y **PC2Bperm** permitiendo generar el número de subllaves especificadas.

3.2 Cifrado y descifrado con DES

Como se mencionó anteriormente, el proceso de cifrado requiere la previa generación de las subllaves. Finalmente, el algoritmo de cifrado para un mensaje $m(x, y)$ se ve como sigue:

$$\begin{aligned} m'(x,y) &:= IP(m(x,y)) = l_0(x,y) + y^4 r_0(x,y) \\ \text{Para} \quad & i = 1, ..., 16 \\ & r_i(x,y) := l_{i-1}(x,y) + f(r_{i-1}(x,y), k_i(x,y)) \\ & l_i(x,y) := r_{i-1}(x,y) \\ & m'_i(x,y) := l_i(x,y) + y^4 r_i(x,y) \\ c(x,y) &:= IP^{-1}(m'_{16}(x,y)) \end{aligned}$$

Después de 16 iteraciones, el polinomio resultante $c(x, y)$ es el mensaje cifrado. La rutina **desRound** ejecuta una ronda del sistema de cifrado. Mientras que la función **desEncrypt** realiza el cifrado de un mensaje para el número de rondas específicado.

Para descifrar $c(x, y)$, se aplica el mismo algoritmo de cifrado con la secuencia de llaves en sentido inverso, es decir, comenzando con $k_{16}(x, y)$. El algoritmo es el

siguiente:

$$\begin{aligned} c'(x,y) &:= IP(c(x,y)) = l_{16}(x,y) + y^4 r_{16}(x,y) \\ \text{Para} \quad & j = 16 - i \text{ con } i = 1, ..., 16 \\ & r_j(x,y) := l_{j+1}(x,y) \\ & l_j(x,y) := r_{j+1}(x,y) + f(l_{j+1}(x,y), k_{j+1}(x,y)) \\ & c'_j(x,y) := l_j(x,y) + y^4 r_j(x,y) \\ m(x,y) &:= IP^{-1}(c'_0(x,y)) \end{aligned}$$

La rutina **desDecrypt** ejecuta el descifrado de un mensaje con el número de rondas especificado.

3.2.1 Ejemplo

A manera de ilustración, consideremos nuevamente los polinomios (3.1) y (3.2) del ejemplo inicial, es decir,

$$\begin{aligned} m(x,y) = \; & 1 + x + x^4 + x^6 + (1 + x^5 + x^6)y + (x^2 + x^3 + x^5 + x^6)y^2 \\ & + (1 + x^2 + x^4 + x^5 + x^6)y^3 + (x^2 + x^5 + x^6)y^4 \\ & + (1 + x + x^2 + x^3 + x^5 + x^6)y^5 + (1 + x + x^4 + x^5 + x^6)y^6 \\ & + (x^2 + x^4 + x^6)y^7, \end{aligned} \tag{3.20}$$

$$\begin{aligned} k(x,y) = \; & (x^3 + x^6 + x^7) + (x^2 + x^3 + x^5)y + (x + x^3 + x^5 + x^6 + x^7)y^2 \\ & + (x + x^2 + x^3 + x^4 + x^7)y^3 + (1 + x^3 + x^4 + x^6 + x^7)y^4 \\ & + (1 + x^2 + x^3 + x^4 + x^5)y^5 + (1 + x + x^3 + x^4 + x^5 + x^6 + x^7)y^6 \\ & + (1 + x + x^2 + x^3 + x^7)y^7 \end{aligned} \tag{3.21}$$

El algoritmo de cifrado comienza por aplicar la permutación IP al polinomio $m(x,y)$. En la sección 3.1.4, se ha descrito esto último y se ha calculado $m'(x,y)$.

$$\begin{aligned} m'(x,y) = \; & x + x^2 + x^7 + (x^2 + x^5)y + (x + x^2 + x^3 + x^4 + x^5 + x^6)y^2 \\ & + (x + x^2 + x^4 + x^6 + x^7)y^4 + (1 + x^2 + x^3 + x^4 + x^5)y^5 \\ & + (1 + x + x^4 + x^7)y^6 \\ & + (1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7)y^7 \end{aligned} \tag{3.22}$$

Es importante distinguir los polinomios $l_0(x,y)$ y $r_0(x,y)$ de $m'(x,y)$ ya que son los que se usan en la primera ronda del algoritmo de cifrado.

$$\begin{aligned} l_0(x,y) &= x + x^2 + x^7 + (x^2 + x^5)y \\ &\quad +(x + x^2 + x^3 + x^4 + x^5 + x^6)y^2 \qquad (3.23) \\ r_0(x,y) &= (x + x^2 + x^4 + x^6 + x^7) + (1 + x^2 + x^3 + x^4 + x^5)y \\ &\quad +(1 + x + x^4 + x^7)y^2 \\ &\quad +(1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7)y^3 \qquad (3.24) \end{aligned}$$

En la primera ronda del algoritmo tenemos:

$$\begin{aligned} l_1(x,y) &:= r_0(x,y) \\ r_1(x,y) &:= l_0(x,y) + f(r_0(x,y), k_1(x,y)) \end{aligned}$$

En la sección 3.1.3, se calcularon las llaves para este ejemplo, mientras que la sección anterior describió cómo se aplica la función f. Por lo que

$$\begin{aligned} f(r_0(x,y), k_1(x,y)) &= 1 + x^2 + x^4 + x^7 + (1 + x^2 + x^6 + x^7)y \\ &\quad +(x + x^2 + x^3 + x^4 + x^5 + x^7)y^2 + (1 + x^5 + x^6)y^3 \end{aligned}$$

De tal manera que

$$\begin{aligned} m'_1(x,y) &= x + x^2 + x^4 + x^6 + x^7 + (1 + x^2 + x^3 + x^4 + x^5)y \\ &\quad +(1 + x + x^4 + x^7)y^2 \\ &\quad +(1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7)y^3 \\ &\quad +(1 + x + x^4)y^4 + (1 + x^5 + x^6 + x^7)y^5 \\ &\quad +(x^6 + x^7)y^6 + (1 + x^5 + x^6)y^7 \end{aligned}$$

Continuando con el resto de las iteraciones obtenemos

$$\begin{aligned} l_{16}(x,y) &= 1 + x^2 + x^5 + (1 + x + x^4 + x^5 + x^7)y \\ &\quad +(x + x^2 + x^3 + x^5 + x^6 + x^7)y^2 + (1 + x + x^6)y^3 \qquad (3.25) \\ r_{16}(x,y) &= 1 + x + x^6 + (x + x^3 + x^5 + x^6)y \\ &\quad +(1 + x^5 + x^6)y^2 + (1 + x^2 + x^3 + x^4 + x^6)y^3 \qquad (3.26) \end{aligned}$$

Y definimos $m_{16}(x,y) := r_{16}(x,y) + y^4 l_{16}(x,y)$

$$\begin{aligned} m'_{16}(x,y) &= 1 + x + x^6 + (x + x^3 + x^5 + x^6)y \\ &\quad +(1 + x^5 + x^6)y^2 + (1 + x^2 + x^3 + x^4 + x^6)y^3 \\ &\quad +(1 + x^2 + x^5)y^4 + (1 + x + x^4 + x^5 + x^7)y^5 \\ &\quad +(x + x^2 + x^3 + x^5 + x^6 + x^7)y^6 + (1 + x + x^6)y^7 \end{aligned}$$

Como paso final, se aplica la permutación IP^{-1} a $m'_{16}(x, y)$:

$$\begin{aligned} c(x,y) &= x^2 + x^4 + (x + x^3 + x^4 + x^5 + x^6 + x^7)y \\ &\quad +(1 + x^2 + x^3 + x^4 + x^5)y^2 + (x^2 + x^7)y^3 \\ &\quad +(x^3 + x^4 + x^7)y^4 + (1 + x^4 + x^7)y^5 \\ &\quad +(x + x^2 + x^3 + x^4 + x^6)y^6 + (1 + x + x^2 + x^5 + x^6 + x^7)y^7 \end{aligned}$$

Lo que coincide con el bloque producido por el algoritmo de cifrado tradicional de DES presentado en el capítulo anterior:

$$\begin{aligned} C_1 &= 00101000\ 01011111\ 10111100\ 00100001 \\ &\quad 00011001\ 10001001\ 01111010\ 11100111 \end{aligned}$$

Capítulo 4
Sistema de cifrado AES: Advanced Encryption Standard

En este capítulo se introduce el sistema de cifrado de llave privada AES (Advanced Encryption Standard). Primero, se proporcionan algunos antecedentes históricos referentes al sistema. Después, se expone la especificación original del sistema basada en [12] y [21]. Se revisan la estructura básica del sistema, las operaciones criptográficas que utiliza y la programación de sus llaves. Finalmente, se proporciona un ejemplo a manera de ilustración.

4.1 Antecedentes

Desde 1977 el sistema DES cumplió satisfactoriamente con su misión de proteger información durante su almacenamiento y transmisión en sistemas computacionales y redes. También, la versión mejorada, conocida como Triple DES, fue ampliamente usada por la industria privada, en particular, la bancaria. Sin embargo, en los años previos al año 2000, la comunidad criptográfica demostró continuamente las limitantes del DES gracias a los avances en la velocidad de cómputo.

A partir de 1997, el Instituto Nacional de Estándares y Tecnología de los Estados Unidos conocido, como NIST por sus siglas en inglés, anunció una convocatoria para presentar sistemas de cifrado como candidatos a reemplazar a DES [12]. El resultado de la convocatoria arrojó 15 candidatos de 12 países que cumplían con las criterios que el NIST había impuesto.

En dicha convocatoria se especificaba que los candidatos debían cumplir los siguientes criterios de aceptación. El algoritmo debía:

- Ser público.
- Ser un algoritmo de cifrado por bloque.
- Estar diseñado de manera que puediese aumentar el tamaño de sus llaves según las necesidades.
- Ser implementable tanto en software como en hardware.
- Estar disponible gratuitamente o bajo los términos del NIST.

Una vez aceptados, los algoritmos serían juzgados por expertos de acuerdo a su seguridad, eficiencia computacional, requisitos de memoria, adecuación de software y hardware, simplicidad de diseño y requisitos de licencia.

El proceso de selección se dividió en varias etapas con una conferencia pública al término de cada una de ellas. Al final de la primera etapa, en agosto de 1998, se contaban con 15 candidatos. Para marzo de 1999, la segunda etapa arrojó varios resultados de los candidatos con respecto a su seguridad, eficiencia y características de implementación. Por lo que en agosto del mismo año quedaban 5 finalistas: MARS, RC6, Rijndael, Serpent y Twofish. Una descripción más detallada de la segunda parte del proceso puede consultarse en [23].

Los finalistas se sometieron a un análisis detallado hasta abril de 2000. Dentro de la tercera conferencia, los resultados de la etapa se presentaron públicamente y los asistentes eligieron Rijndael como el algoritmo favorito. Finalmente, el 2 de octubre del año 2000 el NIST anunció de manera oficial que el sistema Rijndael había sido el ganador de su convocatoria.

El sistema de cifrado Rijndael fue desarrollado por los investigadores belgas Joan Daemen y Vincent Rijmen y surge como un refinamiento de un sistema previo llamado *Square* cuyo diseño se centró en la resistencia al criptoanálisis diferencial y lineal. El nombre del sistema es una combinación de los nombres de sus diseñadores.

Estrictamente hablando, AES no es precisamente Rijndael (aunque en la práctica se les identifique de la misma manera). El sistema Rijndael permite un mayor rango de tamaño en sus bloques y en la longitud de las llaves; mientras que AES tiene un tamaño de bloque fijo de 128 bits y longitudes de llave de 128, 192 ó 256 bits. El sistema Rijndael puede especificarse a una longitud de llave que sea

múltiplo de 32 bits, con un mínimo de 128 bits y un máximo de 256 bits.

A diferencia de su predecesor, el sistema de cifrado DES, el sistema Rijndael posee una estructura tipo red de sustitución-permutación y no una estructura de Feistel. Los cálculos del algoritmo AES se llevan a cabo en el campo finito $GF(2^8)$.

4.2 Descripción original del sistema AES

El sistema Rijndael es un sistema de cifrado por bloque iterativo con tamaño de bloque y tamaño de llave variables. En ambos casos, el tamaño puede ser de 128, 192 o 256 bits de manera independiente.

La unidad básica para procesar información en este sistema es el **byte**, es decir, una cadena de 8 bits. Otra convención a tomar en cuenta es que cada byte se representará en el sistema hexadecimal. Esto se hace dividiendo cada byte en dos partes (4 bits cada una) y asignando a cada parte el número hexadecimal correspondiente a dicha representación binaria (Figura 4.1).

Bin	Hex
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

Bin	Hex
1000	8
1001	9
1010	a/A
1011	b/B
1100	c/C
1101	d/D
1110	e/E
1111	f/F

Figura 4.1: Equivalencia entre los sistemas binario y hexadecimal.

Las diferentes transformaciones que aparecen en el sistema actúan sobre un arreglo rectangular de bytes conocido como *estado*. Cada bloque del mensaje puede verse como un estado con cuatro renglones y número de columnas, denotado por `Nb`, igual al tamaño del bloque dividido por 32. A su vez, la llave de cifrado se visualiza como un estado con cuatro renglones, mientras que el número de columnas, denotado por `Nk`, es igual al tamaño de la llave dividido por 32 (Figura 4.2).

a_{00}	a_{01}	a_{02}	a_{03}	a_{04}	a_{05}
a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_{20}	a_{21}	a_{22}	a_{23}	a_{24}	a_{25}
a_{30}	a_{31}	a_{32}	a_{33}	a_{34}	a_{35}

k_{00}	k_{01}	k_{02}	k_{03}
k_{10}	k_{11}	k_{12}	k_{13}
k_{20}	k_{21}	k_{22}	k_{23}
k_{30}	k_{31}	k_{32}	k_{33}

Figura 4.2: Representación como estados de un bloque y una llave de cifrado.

Cada columna de un estado forma un arreglo unidimensional de cuatro bytes que se identifica como *palabra*.

En términos de bytes, los bloques de entrada y de salida del sistema pueden tener tamaño de 16, 24 o 32 bytes y se enumeran de 0 a $4\texttt{Nb} - 1$. Por su parte, la llave de cifrado puede tener longitud de 16, 24 o 32 bytes numerando cada byte desde 0 hasta $4\texttt{Nk} - 1$.

Como se ha mencionado anteriormente, cada bloque de entrada o de salida puede configurarse como estado. Para hacer esto, se necesita indexar los bytes de dicho bloque de tal manera que formen las entradas del estado. Si n es el índice de algún byte que conforma el bloque, los nuevos índices (i, j) de dicho byte están dados por:

$$i = n \bmod 4, j = \lfloor n/4 \rfloor$$

donde $\lfloor n/4 \rfloor$ representa la parte entera del cociente de la división $n/4$. Por ejemplo, con esta configuración el byte b_7 se convierte en la entrada $b_{3,1}$ del estado.

A continuación, se describen las cuatro operaciones criptográficas que se usan en el esquema de cifrado AES.

4.2.1 Transformación SubBytes

La transformación `SubBytes` sustituye un byte por otro usando una tabla de valores llamada *caja de substitución* o *S-caja* (Figura 4.3). La configuración de dicha tabla es el resultado de aplicar una transformación no lineal sobre el campo $GF(2^8)$. A su vez, esta transformación se compone de otras tres transformaciones que se darán a conocer en la siguiente sección.

La transformación `SubBytes` toma como entrada un estado y aplica la S-caja a cada uno de los bytes del mismo; obteniendo como resultado un nuevo estado. Por ejemplo, siguiendo la caja de substitución de la Figura 4.3, se pueden verificar los valores del estado obtenido después de aplicar la S-caja al primer estado de la Figura 4.4. Para aplicar la caja de substitución, los números de un byte se consideran por separado; el primer número indica el renglón de la caja de substitución, mientras que el segundo valor representa la columna de la misma. Por ejemplo, el byte `19` indica el renglón `1` y la columna `9` de la caja de substitución. Por lo tanto, el valor de substitución correspondiente es el número `d4`. La aplicación de `SubBytes` se denota por `SubBytes(estado)`.

	y															
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
x 7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	87	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Figura 4.3: Configuración de la S-caja.

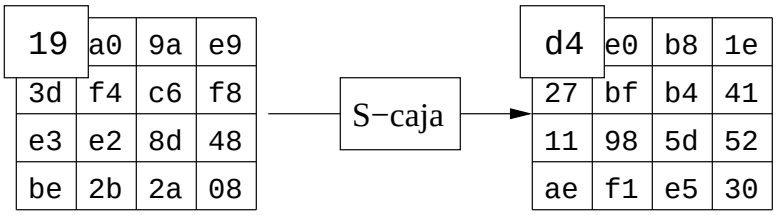

Figura 4.4: Aplicación de la S-caja.

4.2.2 Transformación ShiftRows

Cuando se aplica esta operación, las filas del estado se recorren de manera cíclica hacia la izquierda. Específicamente, los bytes del renglón i se recorren i posiciones a la izquierda para $i = 0, 1, 2, 3$.

El número de desplazmientos depende del tamaño del bloque `Nb`. Estos valores se especifican en la Tabla 4.1. El efecto de la transformación `ShiftRows` se ilustra en

`Nb`	Fila 1	Fila 2	Fila 3
`Nb` $= 4$	1	2	3
`Nb` $= 6$	1	2	3
`Nb` $= 8$	1	3	4

Tabla 4.1: Número de desplazmientos según `Nb`.

la Figura 4.5. La aplicación de esta operación se denota por `ShiftRows(estado)`.

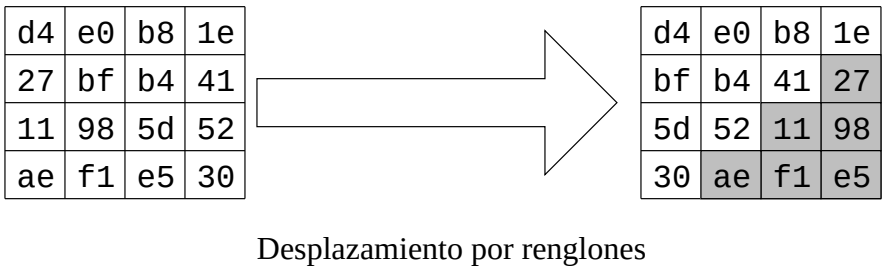

Figura 4.5: `ShiftRows` opera sobre los renglones del estado.

4.2.3 Transformación MixColumns

Para aplicar la operación `MixColumns` se consideran las columnas de un estado y se les aplica una transformación constante a cada una de ellas. Esta transformación puede verse como una multiplicación de matrices:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

La multiplicación por `01` no implica ninguna operación de bits ya que `01` representa el neutro multiplicativo en el campo $GF(2^8)$. Por lo que $\texttt{01} \bullet a_i = a_i$. El producto con `02` se puede describir como un desplazmiento a la izquierda de los bits de un byte a_i y, enseguida, una suma `XOR` con el número `1b` = 00011011, si el bit más significativo de a_i es 1. Por ejemplo, el byte `57` = 01010111, al multiplicarlo por `02`, se convierte en $\texttt{57} \bullet \texttt{02} = 10101110 = \texttt{ae}$. En cambio, $\texttt{ae} \bullet \texttt{02} = 01011100 \oplus 00011011 = 01000111 = \texttt{47}$. Por último, la multiplicación por `03` se simplifica si vemos que $\texttt{03} = \texttt{01} \oplus \texttt{02}$. En la siguiente sección, veremos que estas operaciones son propias del campo $GF(2^8)$. La aplicación de `MixColumns` se denota por `MixColumns(estado)`.

En la Figura 4.6, se ilustra la aplicación de `MixColumns` (Mezcla por columnas). Por ejemplo, $\texttt{02} \bullet \texttt{d4} \oplus \texttt{03} \bullet \texttt{bf} \oplus \texttt{01} \bullet \texttt{5d} \oplus \texttt{01} \bullet \texttt{30} = \texttt{04}$, mientras que $\texttt{01} \bullet \texttt{d4} \oplus \texttt{02} \bullet \texttt{bf} \oplus \texttt{03} \bullet \texttt{5d} \oplus \texttt{01} \bullet \texttt{30} = \texttt{66}$.

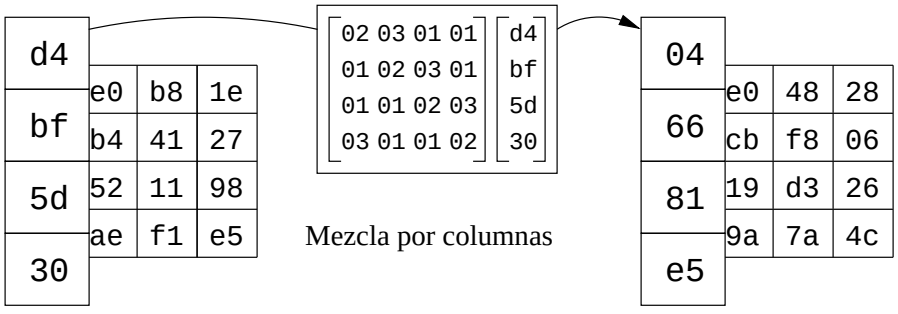

Figura 4.6: Aplicación de `MixColumns`.

4.2.4 Transformación AddRoundKey

En esta transformación se aplica un `XOR` entre los bits de un estado y los de la llave correspondiente a la ronda. Esta llave se genera a partir de la llave privada mediante un proceso llamado *expansión de llaves* que se explicará más adelante. La aplicación de `AddRoundKey` se denota por `AddRoundKey(estado, llave)`.

4.2.5 Las rondas de AES

Una *ronda*, conocida como `Round`, se compone de las 4 transformaciones anteriores: `SubBytes` (Aplicar la S-caja), `ShiftRows` (Hacer desplazmientos por renglones), `MixColumns` (Mezclar columnas) y `AddRoundKey` (Sumar la llave para la ronda correspondiente). La ronda final del cifrado, `FinalRound`, es ligeramente diferente pues sólo incluye las trasformaciones: `SubBytes`, `ShiftRows` y `AddRoundKey`.

El número de rondas se denota por `Nr` y depende de los valores de `Nb` y `Nk` (ver Tabla 4.2). La estructura del algoritmo de cifrado para 10 rondas se muestra en la Figura 4.7. La generación y asignación de las llaves consiste de dos partes: la

`Nr`	`Nb` = 4	`Nb` = 6	`Nb` = 8
`Nk` = 4	10	12	14
`Nk` = 6	12	12	14
`Nk` = 8	14	14	14

Tabla 4.2: Número de rondas según `Nb` y `Nk`.

expansión de la llave y la selección de la llave para la ronda. A continuación se describe cada una de ellas.

4.2.6 Expansión y selección de la llave

La expansión de la llave consiste en extender la llave original en un arreglo rectangular `W` de $N_b(N_r + 1)$ palabras (columnas) conocido como `LlaveExpandida`. La función que expande la llave, `KeyExpansion`, depende del valor N_k; existe una versión para $N_k \leq 6$ y otra para el caso contrario. En ambas versiones, las primeras N_k columnas de `W` son las palabras de la llave original. El resto de las columnas se definen recursivamente en términos de otras columnas previas. Los detalles del algoritmo para ambos casos pueden consultarse en [12] o en [21].

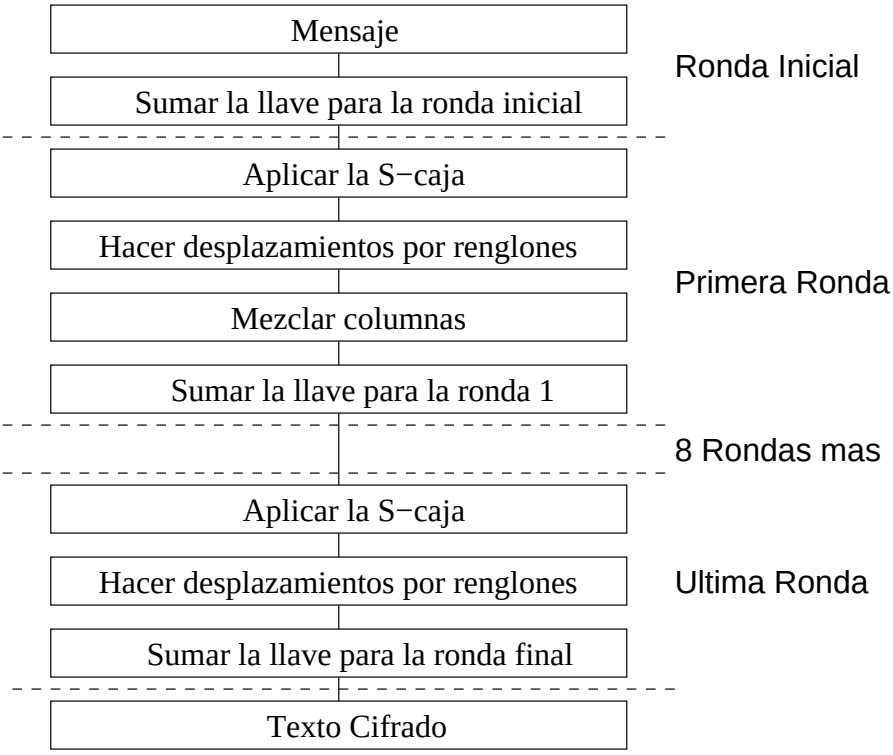

Figura 4.7: Estructura del algoritmo de cifrado AES para 10 rondas.

Básicamente, la función de expansión procesa palabra por palabra. Si i no es un múltiplo de N_k, la columna i es la suma `XOR` de la columna $i - N_k$ y la columna $i-1$. En otro caso, la columna i es la suma `XOR` de la columna $i - N_k$ y una función no lineal de la columna $i - 1$. La función no lineal consiste en llevar a cabo un desplazmiento cíclico de los bytes de una palabra, conocido como `RotByte`; después, se aplica la S-caja de la Figura 4.3 a los bytes de la nueva palabra y se suma una palabra constante, llamada `Rcon`.

Las constantes que se usan para formar las palabras constantes `Rcon` en las primeras 10 rondas se presentan en la Figura 4.8 y se denotan por `RC`. Para formar una palabra `Rcon`$[i]$, simplemente se forma la palabra con el byte `RC`$[i]$ seguida de tres bytes de ceros `00`. Los valores `RC`$[i]$ son independientes de N_k, en la siguiente sección se explicará que se pueden definir de manera recursiva.

Como ejemplo, enseguida se muestra la expansión de la llave para $N_k = 4$ y

i	1	2	3	4	5
RC[i]	01	02	04	08	10
i	6	7	8	9	10
RC[i]	20	40	80	1B	36

Figura 4.8: Constantes por rondas para la generación de las llaves.

$N_b = 4$. Supóngase que la llave original es:

$$k = \begin{array}{|c|c|c|c|}\hline 00 & 00 & 00 & 00 \\ \hline 00 & 00 & 00 & 00 \\ \hline 00 & 00 & 00 & 00 \\ \hline 00 & 00 & 00 & 00 \\ \hline \end{array}.$$

Las columnas de la llave original son las primeras palabras de la expansión, denotadas por $\mathtt{W}[0], \mathtt{W}[1], \mathtt{W}[2], \mathtt{W}[3]$, respectivamente. En este caso, el total de las palabras que se requiere formar es 44. Para generar la palabra $\mathtt{W}[4]$, la cual es múltiplo de 4, primero se rota en una posición los bytes de la palabra $\mathtt{W}[3]$, después se le aplica la S-caja y se le suma la palabra $\mathtt{Rcon}[1]$. Después, a la palabra resultante se le suma la palabra $\mathtt{W}[0]$, esto es,

$$\mathtt{W}[4] = \mathtt{SubByte}(\mathtt{RotByte}(\mathtt{W}[3])) \oplus \mathtt{Rcon}[1] \oplus \mathtt{W}[0]$$

donde

$$\mathtt{SubByte}(\mathtt{RotByte}(00000000)) = \mathtt{SubByte}(00000000) = 63636363.$$

Con lo que se obtiene,

$$\mathtt{W}[4] = 63636363 \oplus 01000000 \oplus 00000000 = 62636363.$$

El proceso continúa con la formación de $\mathtt{W}[5]$, el cual consiste en simplemente sumar $\mathtt{W}[1]$ con $\mathtt{W}[4]$, es decir,

$$\mathtt{W}[5] = \mathtt{W}[4] \oplus \mathtt{W}[1] = 62636363 \oplus 00000000 = 62636363.$$

El resto de las palabras se calcula de manera similar a las dos anteriores. Por ejemplo, para calcular $\mathtt{W}[40]$ primero deberíamos observar que 40 es múltiplo de 4, entonces

$$\mathtt{W}[40] = \mathtt{SubByte}(\mathtt{RotByte}(\mathtt{W}[39])) \oplus \mathtt{Rcon}[10] \oplus \mathtt{W}[36].$$

Esto es,

$$\texttt{W}[40] = \texttt{333b8329} \oplus \texttt{36000000} \oplus \texttt{b1d4d8e2} = \texttt{b4ef5bcb}.$$

Mientras que para `W`[41], solamente se tendría que calcular

$$\texttt{W}[41] = \texttt{W}[40] \oplus \texttt{W}[37] = \texttt{b4ef5bcb} \oplus \texttt{8a7db9da} = \texttt{3e92e211}.$$

En la Figura 4.9 se puede apreciar, de manera general, el proceso de expansión de la llave k.

i	W[i-1]	RotByte()	SubByte()	Rcon[i/Nk]	XOR	W[i-Nk]	W[i]
4	00000000	00000000	63636363	01000000	62636363	00000000	62636363
5	62636363					00000000	62636363
6	62636363					00000000	62636363
7	62636363					00000000	62636363
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
36	3338049f	38049f33	07f2dbc3	1b000000	1cf2dbc3	0e3b9751	b1d4d8e2
37	b1d4d8e2					f9a9061d	8a7db9da
38	8a7db9da					03610afa	1d7bb3de
39	1d7bb3de					3338049f	4c664941
40	4c664941	6649414c	333b8329	36000000	053b8329	b1d4d8e2	b4ef5bcb
41	b4ef5bcb					8a7db9da	3e92e211
42	3e92e211					1d7bb3de	23e951cf
43	23e951cf					4c664941	6f8f188e

Figura 4.9: Expansión de la llave k.

4.3 Cifrado y descifrado con AES

Habiendo generado las llaves mediante la expansión de llaves para la llave inicial `llave`, el proceso de cifrado con AES para un estado `estado` consiste en una ronda inicial en la que solamente se añade la llave inicial `llave` al estado `estado`, esto es, `AddRoundKey`, seguido de `Nr-1` aplicaciones de la tranformación `Round` y, finalmente, una aplicación de la ronda final `FinalRound`. La ronda inicial y cada ronda tienen como entrada un estado y una llave correspondientes a la ronda. La llave para la ronda i se denota por `LlaveExpandida[i]`, mientras que `LlaveExpandida[0]` denota la entrada de la ronda inicial, es decir, `llave`. En resumen, el algoritmo de cifrado se ve como:

```
AES(estado, llave){
   KeyExpansion(llave, Nk);
   AddRoundKey(llave, estado);
   Para i=1 hasta i<Nr
      Round(estado, LlaveExpandida[i]);
   FinalRound(estado, LlaveExpandida[Nr]);
}
```

donde

```
Round(estado,LlaveExpandida[i]){
   SubBytes(estado);
   ShiftRows(estado);
   MixColumns(estado);
   AddRoundKey(estado, LlaveExpandida[i]);
}
```

```
   FinalRound(estado, LlaveExpandida[Nr]){
   SubBytes(estado);
   ShiftRows(estado);
   AddRoundKey(estado, LlaveExpandida[Nr]);
}
```

Las transformaciones que integran la ronda `Round` tienen sus inversos y se usan para describir el proceso de descifrado. Estas transformaciones son `InvSubBytes`, `InvShiftRows`, `InvMixColumns` y `AddRoundKey`.

La transformación `InvShiftRows` es el inverso de `ShiftRows`. Los bytes en los tres últimos renglones de un estado se recorren cíclicamente hacia la derecha el número de posiciones que indica la Tabla 4.1.

La transformación `InvSubBytes` es el inverso de `SubBytes`. En esta transformación se usa el inverso de la tabla de la Figura 4.3.

La transformación `InvMixColumns` es el inverso de `MixColumns`. `InvMixColumns` se aplica a las columnas de un estado. Al igual que con `MixColumns`, esta trans-

formación puede escribirse como una multiplicación de matrices:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} \texttt{0e} & \texttt{0b} & \texttt{0d} & \texttt{09} \\ \texttt{09} & \texttt{0e} & \texttt{0b} & \texttt{0d} \\ \texttt{0d} & \texttt{09} & \texttt{0e} & \texttt{0b} \\ \texttt{0b} & \texttt{0d} & \texttt{09} & \texttt{0e} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

La transformación `AddRoundKey` es inversa de sí misma ya que solamente implica la operación `XOR`. La transformación inversa de `Round` se llama `InvRound` y se compone de los pasos: `AddRoundKey`, `InvMixColumns`, `InvShiftRows` y `InvSubBytes`. Mientras que la transformación inversa de `FinalRound` conocida como `InvFinalRound` consiste en llevar a cabo los pasos: `AddRoundKey`, `InvShiftRows` y `InvSubBytes`. Ambas transformaciones sirven para describir un algoritmo de descifrado *directo* que comienza con la aplicación de `InvFinalRound` seguida de la correspondiente aplicación de `InvRound`.

```
InvRound(estado,LlaveExpandida[i]){
   AddRoundKey(estado, LlaveExpandida[i]);
   InvMixColumns(estado);
   InvShiftRows(estado);
   InvSubBytes(estado);
}

   InvFinalRound(estado, LlaveExpandida[Nr]){
   AddRoundKey(estado, LlaveExpandida[Nr]);
   InvShiftRows(estado);
   InvSubBytes(estado);
}
```

El algoritmo de descifrado directo se ve como sigue:

```
InvFinalRound(estado, LlaveExpandida[Nr]);
   Para i=1 hasta i<Nr
      InvRound(estado, LlaveExpandida[Nr-i]);
AddRoundKey(estado, LlaveExpandida[0]);
```

Equivalentemente, se puede describir un algoritmo de descifrado usando el orden que implementa `Round` pero con las transformaciones inversas detalladas anteriormente. Por razones de implementación resulta conveniente que el primer paso de la ronda sea `SubBytes`. Por esto, es deseable que `InvSubBytes` sea también la primera transformación en aplicarse dentro de la ronda inversa `InvRound`. Para explicar el algoritmo de descifrado equivalente, primero debemos notar dos propiedades [12]:

1. El orden de `InvShiftRows` e `InvSubBytes` es indiferente.

2. El orden de `AddRoundKey` e `InvMixColumns` puede cambiar si la llave de la ronda se adapta convenientemente.

La transformación `InvShiftRows` no tiene efecto alguno sobre los valores de los bytes y la transformación `InvSubBytes` opera sobre los bytes de un estado independientemente de su posición, por lo tanto, pueden conmutar. Por otro lado, la transformación `MixColumns` es lineal, por lo tanto, `InvMixColumns` también lo es. En consecuencia,

$$\texttt{InvMixColumns}(\texttt{estado} \oplus \texttt{llave}) = \texttt{InvMixColumns}(\texttt{estado}) \oplus \texttt{InvMixColumns}(\texttt{llave}).$$

Con esto, se puede definir la ronda `EqRound` como la tranformación que incluye la aplicación de `InvSubBytes`, `InvShiftRows`, `InvMixColumns` y `AddRoundKey`. A su vez, `EqFinalRound` se define como la aplicación de `InvSubBytes`, `InvShiftRows` y `AddRoundKey`. Adicionalmente, la expansión de las llaves sufre una adecuación que consiste en aplicar `InvMixColumns` a todas las llaves generadas excepto la primera y última. Este algoritmo equivalente puede verse en [12].

4.3.1 Ejemplo

A manera de ilustración, a continuación, se aplica una ronda del esquema de cifrado a un bloque de texto plano de 128 bits usando un tamaño de llave de la misma longitud. Nuevamente, consideremos el bloque de texto plano $m =$"*SaludosTerricola*". Recordemos que el código ASCII de los caracteres de m es el conjunto

$$\{83, 97, 108, 117, 100, 111, 115, 84, 101, 114, 114, 105, 99, 111, 108, 97\}.$$

De manera natural, podemos sustituir cada código por su representación binaria o hexadecimal y, así, resulta sencillo ver a m como un estado. La representación hexadecimal de m es la siguiente:

$$m = \begin{array}{|c|c|c|c|} \hline 53 & 61 & 6c & 75 \\ \hline 64 & 6f & 73 & 54 \\ \hline 65 & 72 & 72 & 69 \\ \hline 63 & 6f & 6c & 61 \\ \hline \end{array}$$

Se recordará que con DES el mensaje m se dividió en dos bloques para poderlo cifrar. En este caso, todo el mensaje m forma un estado. Mientras que la llave a utilizar será la misma usada en el ejemplo con DES. Pero como sólo está formada

por 64 bits, repetiremos dos veces el bloque para completar el estado. De esta manera, la llave inicial k, escrita en notación hexadecimal, queda dada por:

$$k = \begin{array}{|c|c|c|c|}\hline 13 & 34 & 57 & 79 \\ \hline 9b & bc & df & f1 \\ \hline 13 & 34 & 57 & 79 \\ \hline 9b & bc & df & f1 \\ \hline \end{array}$$

La expansión de la llave produce 44 palabras. Las primeras 8 de ellas son:

W_0	W_1	W_2	W_3	W_4	W_5	W_6	W_7
13	34	57	79	77	aa	f6	6d
9b	bc	df	f1	ec	16	29	9c
13	34	57	79	ff	22	7e	e5
9b	bc	df	f1	64	9e	a1	14

El primer paso en el algoritmo de cifrado consiste en añadir la llave original al bloque de texto. Esto es:

$$\begin{array}{|c|c|c|c|}\hline 53 & 61 & 6c & 75 \\ \hline 64 & 6f & 73 & 54 \\ \hline 65 & 72 & 72 & 69 \\ \hline 63 & 6f & 6c & 61 \\ \hline \end{array} \oplus \begin{array}{|c|c|c|c|}\hline 13 & 34 & 57 & 79 \\ \hline 9b & bc & df & f1 \\ \hline 13 & 34 & 57 & 79 \\ \hline 9b & bc & df & f1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline 40 & 55 & 3b & 0c \\ \hline ff & d3 & ac & a5 \\ \hline 76 & 46 & 25 & 10 \\ \hline f8 & d3 & b3 & 90 \\ \hline \end{array}$$

Después de aplicar la S-caja al estado anterior, se obtiene:

09	16	38	41
fc	66	5a	66
e2	91	3f	6d
fe	06	ca	60

En seguida, se lleva a cabo la transformación *ShiftRows*:

09	16	38	41
66	5a	66	fc
3f	6d	e2	91
60	fe	06	ca

Luego, el estado resultante se somete a la transformación *MixColumns*:

e7	51	3e	c6
e4	eb	cf	c0
b1	8f	8b	c1
82	ea	c0	21

Finalmente, se suma la llave de la ronda 1 y se obtiene el siguiente estado:

e7	51	3e	c6
e4	eb	cf	c0
b1	8f	8b	c1
82	ea	c0	21

$\oplus$

77	aa	f6	6d
ec	16	29	9c
ff	22	7e	e5
64	9e	a1	14

$=$

90	bd	c1	a2
4e	fd	ed	5e
47	a6	f5	60
ef	76	25	35

Después de aplicar 8 rondas más y una ronda final, `FinalRound`, la cual incluye todas las operaciones criptográficas anteriores excepto `MixColumns` se obtiene el mensaje cifrado

$c =$

bf	94	96	97
11	fd	fe	94
ba	3c	f2	35
f3	8b	5b	bf

Para recuperar el mensaje m, aplicamos a c el algoritmo de descifrado directo descrito anteriormente. Esto es, comenzamos con el inverso de la ronda final, `InvFinalRound`, aplicado a c y, `ExpandedKey[11]`, la última llave generada:

d7	40	df	bb
67	82	b7	07
97	d6	2c	69
1a	e3	30	a8

Después, se llevan a cabo 9 repeticiones de `InvRound` donde se aplican los inversos de las tranformaciones usadas en `Round`, es decir, `InvMixColumns`, `InvShiftRows` y `InvSubBytes`. Finalmente, se suma la llave original, `ExpandedKey[0]`, y se recupera m.

53	61	6c	75
64	6f	73	54
65	72	72	69
63	6f	6c	61

Capítulo 5
Descripción polinomial del sistema de cifrado AES

En este capítulo se describe el sistema AES mediante operaciones algebraicas en el campo finito $GF(2^8)$, dicha descripción se basa en [26]. La principal estructura del sistema, el estado, se presenta como un elemento de un anillo de polinomios. También, algunas de las operaciones criptográficas se representan mediante operaciones dentro de este anillo. Finalmente, se presenta el ejemplo del capítulo anterios usando polinomios.

5.1 Sistema de cifrado AES

La descripción en forma algebraica del sistema de cifrado Rijndael que a continuación se presenta se sigue de [26]. Para hacerla, se manejarán dos representaciones de cada byte, una representación binaria y la otra hexadecimal. Como se recordará cada byte se compone de 8 dígitos binarios, es decir, tiene la forma $a_7a_6a_5a_4a_3a_2a_1a_0$ con $a_i \in \mathbb{F}_2$, pero a su vez, se puede formar usando 2 dígitos hexadecimales, digamos $[\texttt{xy}]_{16}$ donde $[\texttt{x}]_{16} = a_7a_6a_5a_4$ y $[\texttt{y}]_{16} = a_3a_2a_1a_0$. Por ejemplo, el número `9b` representa la cadena `10011011` ya que `9` es la representación hexadecimal de la cadena `1001` y `b` es la representación hexadecimal de la cadena `1011` (Figura 4.1).

También, se requieren tres estructuras algebraicas principales las cuales se describen a continuación. La primera es el espacio vectorial de dimensión 8 sobre $\mathbb{F}_2$, $\mathbb{F}_2^8$. De tal manera que un byte (8 bits) está representado por un elemento en $\mathbb{F}_2^8$, esto es, los dígitos del byte $a_7a_6a_5a_4a_3a_2a_1a_0$ se pueden apreciar como el vector $(a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0) \in \mathbb{F}_2^8$ donde $a_i \in \mathbb{F}_2$ para $i = 0, 1, ..., 8$.

Por otro lado, la siguiente estructura algebraica a considerar será el campo finito $GF(2^8)$ de 256 elementos. Considérese el campo de los números binarios $\mathbb{F}_2 = \{0, 1\}$ y el polinomio irreducible

$$\mu(z) := z^8 + z^4 + z^3 + z + 1 \in \mathbb{F}_2[z].$$

Ahora, se construye el campo $\mathbb{F} := \mathbb{F}_2[z]/\langle \mu(z) \rangle = GF(2^8)$ cuyos elementos tienen la forma:

$$a_7 z^7 + a_6 z^6 + a_5 z^5 + a_4 z^4 + a_3 z^3 + a_2 z^2 + a_1 z + a_0 \text{ donde } a_i \in \mathbb{F}_2.$$

Tanto $\mathbb{F}_2^8$ como $\mathbb{F}$ son espacios vectoriales de dimensión 8 sobre $\mathbb{F}_2$ y se puede establecer una relación entre ambas representaciones mediante el siguiente isomorfismo de $\mathbb{F}_2$-espacios vectoriales. Primero, definamos el mapeo $P : \mathbb{F}_2^8 \to \mathbb{F}$ dado por

$$P(a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0) = a_7 z^7 + a_6 z^6 + a_5 z^5 + a_4 z^4 + a_3 z^3 + a_2 z^2 + a_1 z + a_0.$$

Por ejemplo, el número `9b` $= 10011011$ tiene una representación polinomial dada por

$$P(\texttt{9b}) = P(1, 0, 0, 1, 1, 0, 1, 1) = z^7 + z^4 + z^3 + z + 1.$$

Observemos que el mapeo P es una transformación lineal. Sean $\mathbf{a}$ y $\mathbf{b}$ dos vectores en $\mathbb{F}_2^8$ y c un escalar en $\mathbb{F}_2$. Entonces

$$\begin{aligned} P(\mathbf{b} + \mathbf{a}) &= P(b_7 + a_7, ..., b_1 + a_1, b_0 + a_0) \\ &= (b_7 + a_7)z^7 + ... + (b_1 + a_1)z + (b_0 + a_0) \\ &= (b_7 z^7 + ... + b_1 z + b_0) + (a_7 z^7 + ... + a_1 z + a_0) \\ &= P(\mathbf{b}) + P(\mathbf{a}). \end{aligned}$$

También,

$$\begin{aligned} P(c\mathbf{a}) &= P(ca_7, ..., ca_1, ca_0) \\ &= ca_7 z^7 + ... + ca_1 z + ca_0 \\ &= c(a_7 z^7 + ... + a_1 z + a_0) \\ &= cP(\mathbf{a}). \end{aligned}$$

Más aún, la transformación P es un isomorfismo. Notemos que el único vector $\mathbf{a} \in \mathbb{F}_2^8$ tal que $P(\mathbf{a}) = 0$ es $\mathbf{a} = (0, 0, 0, 0, 0, 0, 0, 0)$. Esto implica que P es uno a

uno. Como $\mathbb{F}_2^8$ y $\mathbb{F}$ tienen la misma dimensión sobre $\mathbb{F}_2$, P también es sobre. Por lo tanto, P es un isomorfismo.

La tercera estructura algebraica importante en el algoritmo Rijndael es el anillo

$$\mathcal{R} := \mathbb{F}[x, y]/\langle x^4 - 1, y^4 - 1\rangle.$$

Cada estado es un elemento $r \in \mathcal{R}$ que tiene la forma

$$r = \sum_{i=0}^{3}\sum_{j=0}^{3} r_{ij}x^i y^j = \sum_{j=0}^{3}\left(\sum_{i=0}^{3} r_{ij}x^i\right) y^j$$

donde $r_{ij} \in \mathbb{F}$.

En las siguientes secciones, se describirá tanto el algoritmo de cifrado y como el de descifrado a través de operaciones polinomiales en dicho anillo. Comenzaremos por detallar las cuatro operaciones básicas de este sistema. Recordemos que una ronda se compone de cuatro transformaciones, cada una de éstas se puede describir en términos algebraicos como se verá a continuación.

5.1.1 Transformación SubBytes

En esta operación cada elemento $r_{ij} \in \mathbb{F}$ se substituye por otro usando una permutación φ del grupo simétrico S_{256}. Dicha permutación φ está formada por las siguientes tres permutaciones:

$$\varphi_1 : \mathbb{F} \mapsto \mathbb{F}, \quad f \mapsto \begin{cases} f^{-1} & \text{si } f \neq 0, \\ 0 & \text{si } f = 0; \end{cases}$$

$$L : \mathbb{F} \mapsto \mathbb{F}, \quad f \mapsto (z^4 + z^3 + z^2 + z + 1)f \bmod (z^8 - 1);$$

$$\varphi_3 : \mathbb{F} \mapsto \mathbb{F}, \quad f \mapsto (z^6 + z^5 + z + 1) + f.$$

Estos mapeos son permutaciones porque son funciones uno a uno y sobre de $\mathbb{F}$ en $\mathbb{F}$. En consecuencia, la composición entre ellos también es una permutación. La permutación φ sobre $\mathbb{F}$ se define entonces como $\varphi := \varphi_3 \circ L \circ \varphi_1$ y se conoce como la S-caja o caja de substitución. A continuación, se verá que cada mapeo puede describirse mediante un polinomio de permutación en $\mathbb{F}[u]$.

Para el mapeo φ_1 el polinomio de permutación correspondiente es $\varphi_1(u) = u^{254}$, puesto que $u^{254}u = 1$, es decir, $u^{-1} = u^{254}$.

En el siguiente caso, el mapeo lineal L es un producto por una constante, esto es, $L(f) = g \cdot f$ donde $g(z) = z^4 + z^3 + z^2 + z + 1$. Además, como $\mathbb{F}$ es un espacio vectorial sobre $\mathbb{F}_2$ de dimensión 8, se puede definir entonces un único polinomio $\mathcal{L}(u) = \sum_{i=0}^{7} \lambda_i u^{2^i} \in \mathbb{F}[u]$ tal que

$$\mathcal{L}(u) = L(f)$$

para toda $f \in \mathbb{F}$ ([17]). Este tipo de polinomios se conoce como linealizados porque tienen las siguientes propiedades:

$$\begin{array}{lcl} \mathcal{L}(\beta + \gamma) & = & \mathcal{L}(\beta) + \mathcal{L}(\gamma) \text{ para cualesquiera } \beta, \gamma \in \mathbb{F}, \\ \mathcal{L}(c\beta) & = & c\mathcal{L}(\beta) \text{ para cualquier } c \in \mathbb{F}_2 \text{ y cualquier } \beta \in \mathbb{F}. \end{array}$$

Lo que se obtiene con esta representación de L es que $\mathcal{L}(u)$ será un polinomio de permutación. Si $\alpha_1, ..., \alpha_8$ es una base de $\mathbb{F}$ sobre $\mathbb{F}_2$ entonces es posible calcular los coeficientes $\lambda_0, \lambda_1, ..., \lambda_7$ a través de las relaciones lineales:

$$\mathcal{L}(\alpha_j) = \sum_{i=0}^{7} \lambda_i \alpha_j^{2^i} = L(\alpha_j), \qquad j = 1, ..., 8. \tag{5.1}$$

El sistema de ecuaciones puede resolverse explícitamente. Para hacer esto se considera la base dual $\{\beta_1, ..., \beta_8\}$ de $\{\alpha_1, ..., \alpha_8\}$ caracterizada por la siguiente condición ([17]):

$$Tr_{\mathbb{F}/\mathbb{F}_2}(\alpha_i \beta_j) = \begin{cases} 1 & \text{si } i = j, \\ 0 & \text{si } i \neq j \end{cases}$$

donde $Tr_{\mathbb{F}/\mathbb{F}_2}(\alpha_i \beta_j)$ denota la traza de $\alpha_i \beta_j$ sobre $\mathbb{F}_2$. Considérense las matrices:

$$A := \begin{pmatrix} \alpha_1 & \alpha_1^2 & \alpha_1^4 & \dots & \alpha_1^{2^7} \\ \alpha_2 & \alpha_2^2 & \alpha_2^4 & \dots & \alpha_2^{2^7} \\ \vdots & \vdots & \vdots & & \vdots \\ \alpha_8 & \alpha_8^2 & \alpha_8^4 & \dots & \alpha_8^{2^7} \end{pmatrix}, B := \begin{pmatrix} \beta_1 & \beta_2 & \dots & \beta_8 \\ \beta_1^2 & \beta_2^2 & \dots & \beta_8^2 \\ \beta_1^4 & \beta_2^4 & \dots & \beta_8^4 \\ \vdots & \vdots & & \vdots \\ \beta_1^{2^7} & \beta_2^{2^7} & \dots & \beta_8^{2^7} \end{pmatrix}.$$

Como $\beta_1, \dots, \beta_8$ es la base dual de $\alpha_1, \dots, \alpha_8$, se tiene que $AB = I_8$.

Sea S la matriz de transición de $\{1, z, ..., z^7\}$, base natural de $\mathbb{F}$, a $\{\alpha_1, \alpha_2, ..., \alpha_8\}$, es decir,

$$\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_8 \end{pmatrix} = S^t \begin{pmatrix} 1 \\ z \\ \vdots \\ z^7 \end{pmatrix}$$

y considérese la matriz

$$L := \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

la cual describe el mapeo lineal con respecto a la base polinomial $1, z, z^2, ..., z^7$. Entonces se tiene el siguiente:

Lema 5.1.1. *Los coeficientes $\lambda_0, \lambda_1, ..., \lambda_7$ del polinomio de permutación $\mathcal{L}(u)$ están dados por:*

$$\begin{pmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_7 \end{pmatrix} = BSL^tS^{-1} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_8 \end{pmatrix}.$$

Demostración. Como S es la matriz de transición de $\{1, z, ..., z^7\}$ a $\{\alpha_1, \alpha_2, ..., \alpha_8\}$, tenemos que S^{-1} es la matriz de transición de $\{\alpha_1, \alpha_2, ..., \alpha_8\}$ a $\{1, z, ..., z^7\}$. Luego, la matriz SL^tS^{-1} describe el cambio de base del mapeo lineal L con respecto a la base $\alpha_1, ..., \alpha_8$. Entonces el sistema de ecuaciones lineales descrito en (5.1) queda como:

$$\begin{pmatrix} \alpha_1 & \alpha_1^2 & \alpha_1^4 & \dots & \alpha_1^{2^7} \\ \alpha_2 & \alpha_2^2 & \alpha_2^4 & \dots & \alpha_2^{2^7} \\ \vdots & \vdots & \vdots & & \vdots \\ \alpha_8 & \alpha_8^2 & \alpha_8^4 & \dots & \alpha_8^{2^7} \end{pmatrix} \begin{pmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_7 \end{pmatrix} = SL^tS^{-1} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_8 \end{pmatrix}.$$

Al multiplicar ambos lados de la ecuación por B se obtiene el resultado deseado. $\square$

Para calcular los coeficientes $\lambda_0, \lambda_1, ..., \lambda_7$ explícitamente se puede usar una base normal. Recordemos que una base normal tiene la forma $\{\alpha, \alpha^q, ..., \alpha^{q^{m-1}}\}$ para algún $\alpha \in \mathbb{F}$. Sea $\alpha := z^5 + 1 \in \mathbb{F}$. El elemento α es primitivo porque es un generador del grupo cíclico $\mathbb{F}^*$. Tanto α como sus conjugados respecto de $\mathbb{F}_2$, es decir, $\{\alpha_i := \alpha^{2^{i-1}} | i = 1, ..., 8\}$, forman una base normal de $\mathbb{F}$ sobre $\mathbb{F}_2$. Dichas bases se conocen como bases normales primitivas. El elemento α es el primero con respecto al orden lexicográfico de los elementos primitivos de $\mathbb{F}$.

La base dual de $\{\alpha_1, ..., \alpha_8\}$ existe y es única ([17]). En este caso, es el conjunto $\{\beta_j := \beta^{2^{j-1}} | j = 1, ..., 8\}$ donde $\beta = z^5 + z^4 + z^2 + 1$. La matriz de cambio de base es:

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

Por consiguiente,

$$\begin{pmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_7 \end{pmatrix} = BSL^tS^{-1} \begin{pmatrix} \alpha \\ \alpha^2 \\ \vdots \\ \alpha^{2^7} \end{pmatrix} = \begin{pmatrix} z^2 + 1 \\ z^3 + 1 \\ z^7 + z^6 + z^5 + z^4 + z^3 + 1 \\ z^5 + z^2 + 1 \\ z^7 + z^6 + z^5 + z^4 + z^2 \\ 1 \\ z^7 + z^5 + z^4 + z^2 + 1 \\ z^7 + z^3 + z^2 + z + 1 \end{pmatrix}$$

los cuales son los coeficientes del polinomio $\varphi(u)$. Finalmente, cuando se componen los tres mapeos, se obtiene:

$$\varphi(u) = \varphi_3 \circ \mathcal{L} \circ \varphi_1(u) = 1 + z + z^5 + z^6 + \mathcal{L}(u^{254}) \bmod (u^{256} + u).$$

5.1.2 Transformación ShiftRows

En esta operación los bytes del i-ésimo renglón se recorren i posiciones de manera cíclica. Esta operación se puede describir algebraicamente de la siguiente forma.

Sea $r = r(x, y) \in \mathcal{R}$ y considérese la transformación monomial $\pi : \mathcal{R} \mapsto \mathcal{R}$ dada por:

$$\pi(r(x, y)) = r(xy^3, y),$$

equivalentemente,

$$\pi\Big(\sum_{i=0}^{3}\sum_{j=0}^{3} r_{ij}x^i y^j\Big) = \sum_{i=0}^{3}\sum_{j=0}^{3} r_{ij}x^i y^{3i+j}.$$

5.1.3 Transformación MixColumns

En esta operación cada columna $r_j(x) = \sum_{i=0}^{3} r_{ij}x^i$ de un estado $r(x, y)$ se multiplica por el polinomio fijo

$$\gamma(x) = (z + 1)x^3 + x^2 + x + z \in \mathcal{R},$$

esto es,

$$\gamma(x)r_j(x) \text{ para } j = 0, 1, 2, 3.$$

Al llevar a cabo esta operación se usa tanto el módulo del anillo $\mathcal{R}$ como el módulo del campo $\mathbb{F}$ para reducir el polinomio resultante. Por ejemplo, para la primera columna, es decir, $j = 1$, se tiene

$$((z + 1)x^3 + x^2 + x + z)(r_{01}y + r_{11}xy + r_{21}x^2y + r_{31}x^3y).$$

5.1.4 Transformación AddRoundKey

En este paso la llave correspondiente a la ronda t, esto es, el polinomio $k^{(t)}(x, y) \in \mathcal{R}$, se suma con un polinomio $r \in \mathcal{R}$ que se obtiene después de aplicar las tres operaciones previas, es decir,

$$r(x, y) + k^{(t)}(x, y) = \sum_{i=0}^{3}\sum_{j=0}^{3}(r_{ij} + k_{ij^{(t)}})x^i y^j \text{ donde } k_{ij}^{(t)}, r_{ij} \in \mathbb{F}.$$

5.1.5 Las rondas de AES

Como se mencionó en la sección anterior, una ronda se compone de las cuatro transformaciones anteriores. De tal manera que dentro del algoritmo de cifrado, algebraicamente una ronda es:

$$\gamma(x)\sum_{i=0}^{3}\sum_{j=0}^{3}\varphi(m_{ij})x^i y^{3i+j} + k(x,y).$$

5.1.6 Expansión y selección de la llave

A partir de la llave original se calculan recursivamente los siguientes elementos $k^{(t)}(x,y) \in \mathcal{R}$ para $t = 0, ..., 10$.

$$\begin{aligned} k^{(0)} &= k; \\ k_0^{(t+1)} &= \left(\sum_{i=0}^{3}\varphi(k_{i,3}^{(t)})x^i\right)x^3 + z^t + k_0^{(t)} \text{ para } t = 0, ..., 9; \\ k_i^{(t+1)} &= k_{i-1}^{(t+1)} + k_i^{(t)} \text{ para } t = 0, ..., 9, i = 1, 2, 3. \end{aligned}$$

Nótese que el polinomio z^t es la representación polinomial de las constantes de la Figura 4.8.

5.2 Cifrado y descrifrado con AES

Para un mensaje $m(x,y)$ y después de generar las llaves de cada ronda, $k^{(t)}(x,y)$, el algoritmo de cifrado es el siguiente:

$$\begin{aligned} m^{(0)} &= m + k^{(0)}; \\ &\vdots \\ m^{(t+1)} &= \gamma\sum_{i=0}^{3}\sum_{j=0}^{3}\varphi(m_{ij}^{(t)})x^i y^{3i+j} + k^{(t+1)} \text{ para } t = 0, ..., 8; \\ &\vdots \\ c = m^{(10)} &= \sum_{i=0}^{3}\sum_{j=0}^{3}\varphi(m_{ij}^{(9)})x^i y^{3i+j} + k^{(10)}. \end{aligned}$$

El polinomio de permutación $\varphi(u)$ que representa la S-caja tiene elemento inverso $\psi(u) \in \mathbb{F}[u]$ con grado a lo más 255. El polinomio $\psi(u)$ es un polinomio de permutación y puede consultarse en [26]. Por otro lado, el polinomio $\gamma(x) \in \mathcal{R}$ es invertible también y su inverso está dado por

$$\gamma^{-1}(x) = (z^3 + z + 1)x^3 + (z^3 + z^2 + 1)x^2 + (z^3 + 1)x + (z^3 + z^2 + z).$$

Con estos dos polinomios podemos describir el descifrado en términos polinomiales. El algoritmo de descifrado es el siguiente:

$$\begin{aligned}
c^{(0)} &= c + k^{(10)}; \\
&\vdots \\
c^{(t+1)} &= \gamma^{-1} \sum_{i=0}^{3} \sum_{j=0}^{3} \psi(c_{ij}^{(t)}) x^i y^{i+j} + \gamma^{-1} k^{(9-t)} \text{ para } t = 0, ..., 8; \\
&\vdots \\
c^{(10)} &= \sum_{i=0}^{3} \sum_{j=0}^{3} \psi(c_{ij}^{(9)}) x^i y^{i+j} + k^{(0)}.
\end{aligned}$$

Tanto el cifrado como el descifrado tienen el mismo orden en la aplicación de sus respectivas transformaciones, sólo cambia la programación de las llaves. En el caso del descifrado se comienza por $k^{10}, k^9, ..., k^1, k^0$.

5.2.1 Ejemplo

Para llevar a cabo la representación polinomial del ejemplo del capítulo anterior, primero definimos los polinomios $m(x, y)$ y $k(x, y)$, esto es, el mensaje y la llave, respectivamente. Recordemos que cada byte es, a su vez, un elemento del campo $\mathbb{F}$. Por ejemplo, el primer byte de m, $\texttt{53} = 01010011$, tiene como representante al polinomio $z^6 + z^4 + z + 1$. Una vez que se tienen todos los bytes como polinomios,

se define un elemento r del anillo $\mathcal{R}$. Este elemento es el polinomio:

$$\begin{aligned} m(x,y) &= 1+z+z^4+z^6+x(z^2+z^5+z^6) \\ &\quad +x^2(1+z^2+z^5+z^6)+x^3(1+z+z^5+z^6) \\ &\quad +y\Big(1+z^5+z^6+x(1+z+z^2+z^3+z^5+z^6) \\ &\quad +x^2(z+z^4+z^5+z^6)+x^3(1+z+z^2+z^3+z^5+z^6)\Big) \\ &\quad +y^2\Big(z^2+z^3+z^5+z^6+x(1+z+z^4+z^5+z^6) \\ &\quad +x^2(z+z^4+z^5+z^6)+x^3(z^2+z^3+z^5+z^6)\Big) \\ &\quad +y^3\Big(1+z^2+z^4+z^5+z^6+x(z^2+z^4+z^6) \\ &\quad +x^2(1+z^3+z^5+z^6)+x^3(1+z^5+z^6)\Big). \end{aligned}$$

Este polinomio representa el texto que se desea cifrar, mientras que la llave inicial es:

$$\begin{aligned} k(x,y) &= 1+z+z^4+x(1+z+z^3+z^4+z^7) \\ &\quad +x^2(1+z+z^4)+x^3(1+z+z^3+z^4+z^7) \\ &\quad +y\Big(z^2+z^4+z^5+x(z^2+z^3+z^4+z^5+z^7) \\ &\quad +x^2(z^2+z^4+z^5)+x^3(z^2+z^3+z^4+z^5+z^7)\Big) \\ &\quad +y^2\Big(1+z+z^2+z^4+z^6+x(1+z+z^2+z^3+z^4+z^6+z^7) \\ &\quad +x^2(1+z+z^2+z^4+z^6)+x^3(1+z+z^2+z^3+z^4+z^6+z^7)\Big) \\ &\quad +y^3\Big(1+z^3+z^4+z^5+z^6+x(1+z^4+z^5+z^6+z^7) \\ &\quad +x^2(1+z^3+z^4+z^5+z^6)+x^3(1+z^4+z^5+z^6+z^7)\Big). \end{aligned}$$

La expansión de la llave $k(x,y)$ genera las llaves $k^0(x,y) = k(x,y)$ y

$$\begin{aligned} k^1(x,y) &= 1+z+z^2+z^4+z^5+z^6+x(z^2+z^3+z^5+z^6+z^7) \\ &\quad +x^2(1+z+z^2+z^3+z^4+z^5+z^6+z^7)+x^3(z^2+z^5+z^6) \\ &\quad +y\Big(z+z^3+z^5+z^7+x(z+z^2+z^4) \\ &\quad +x^2(z+z^5)+x^3(z+z^2+z^3+z^4+z^7)\Big) \\ &\quad +y^2\Big(z+z^2+z^4+z^5+z^6+z^7+x(1+z^3+z^5) \\ &\quad +x^2(z+z^2+z^3+z^4+z^5+z^6)+x^3(1+z^5+z^7)\Big) \\ &\quad +y^3\Big(1+z^2+z^3+z^5+z^6+x(z^2+z^3+z^4+z^7) \\ &\quad +x^2(1+z^2+z^5+z^6+z^7)+x^3(z^2+z^4)\Big). \end{aligned}$$

Como $m^0(x,y) = m(x,y) + k(x,y)$, entonces

$$\begin{aligned} m^0(x,y) &= z^6+x(1+z+z^2+z^3+z^4+z^5+z^6+z^7) \\ &\quad +x^2(z+z^2+z^4+z^5+z^6)+x^3(z^3+z^4+z^5+z^6+z^7) \\ &\quad +y\Big(1+z^2+z^4+z^6+x(1+z+z^4+z^6+z^7) \\ &\quad +x^2(z+z^2+z^6)+x^3(1+z+z^4+z^6+z^7)\Big) \\ &\quad +y^2\Big(1+z+z^3+z^4+z^5+x(z^2+z^3+z^5+z^7) \\ &\quad +x^2(1+z^2+z^5)+x^3(1+z+z^4+z^5+z^7)\Big) \\ &\quad +y^3\Big(z^2+z^3+x(1+z^2+z^5+z^7) \\ &\quad +x^2z^4+x^3(z^4+z^7)\Big). \end{aligned}$$

En la siguiente iteración se aplica la S-caja representada por el polinomio $\varphi(u)$. Recordemos que este polinomio se aplica a cada elemento de $\mathbb{F}$, por lo que el

siguiente estado se ve como:

$$\begin{aligned}
& \varphi(z^6) + x\varphi(1+z+z^2+z^3+z^4+z^5+z^6+z^7) \\
+ \;& x^2\varphi(z+z^2+z^4+z^5+z^6) + x^3\varphi(z^3+z^4+z^5+z^6+z^7) \\
+ \;& y\Big(\varphi(1+z^2+z^4+z^6) + x\varphi(1+z+z^4+z^6+z^7) \\
+ \;& x^2\varphi(z+z^2+z^6) + x^3\varphi(1+z+z^4+z^6+z^7)\Big) \\
+ \;& y^2\Big(\varphi(1+z+z^3+z^4+z^5) + x\varphi(z^2+z^3+z^5+z^7) \\
+ \;& x^2\varphi(1+z^2+z^5) + x^3\varphi(1+z+z^4+z^5+z^7)\Big) \\
+ \;& y^3\Big(\varphi(z^2+z^3) + x\varphi(1+z^2+z^5+z^7) \\
+ \;& x^2\varphi(z^4) + x^3\varphi(z^4+z^7)\Big).
\end{aligned}$$

Con lo que obtenemos:

$$\begin{aligned}
& 1 + z^3 + x(z^2+z^3+z^4+z^5+z^6+z^7) \\
+ \;& x^2(z+z^5+z^6+z^7) + x^3(z+z^2+z^3+z^4+z^5+z^6+z^7) \\
+ \;& y\Big(z+z^2+z^4 + x(z+z^2+z^5+z^6) \\
+ \;& x^2(1+z^4+z^7) + x^3(z+z^2)\Big) \\
+ \;& y^2\Big(z^3+z^4+z^5 + x(z+z^3+z^4+z^6) \\
+ \;& x^2(1+z+z^2+z^3+z^4+z^5) + x^3(z+z^3+z^6+z^7)\Big) \\
+ \;& y^3\Big(1+z^6 + x(z+z^2+z^5+z^6) \\
+ \;& x^2(1+z^2+z^3+z^5+z^6) + x^3(z^5+z^6)\Big)
\end{aligned}$$

La transformación *ShiftRows* tranforma el polinomio anterior en:

$$\begin{aligned}
& 1 + z^3 + x(z+z^2+z^5+z^6) \\
+ \;& x^2(1+z+z^2+z^3+z^4+z^5) + x^3(z^5+z^6) \\
+ \;& y\Big(z+z^2+z^4 + x(z+z^3+z^4+z^6) \\
+ \;& x^2(1+z^2+z^3+z^5+z^6) + x^3(z+z^2+z^3+z^4+z^5+z^6+z^7)\Big) \\
+ \;& y^2\Big(z^3+z^4+z^5 + x(z+z^2+z^5+z^6) \\
+ \;& x^2(z+z^5+z^6+z^7) + x^3(z+z^2)\Big) \\
+ \;& y^3\Big(1+z^6 + x(z^2+z^3+z^4+z^5+z^6+z^7) \\
+ \;& x^2(1+z^4+z^7) + x^3(z+z^3+z^6+z^7)\Big)
\end{aligned}$$

En la transformación *MixColumns* cada polinomio $r_j(x) = \sum_{i=0}^{3} r_{ij}x^i$ de $r(x,y)$ se multiplica por el polinomio:

$$\gamma(x) = \texttt{03}x^3 + \texttt{01}x^2 + \texttt{01}x + \texttt{02} = (z+1)x^3 + x^2 + x + z$$

Por ejemplo, si $j = 1$, el producto que hay que llevar a cabo es:

$$\Big((z+1)x^3 + x^2 + x + z\Big)\Big(z + z^2 + z^4 + x(z + z^3 + z^4 + z^6) \\ +x^2(1 + z^2 + z^3 + z^5 + z^6) + x^3(z + z^2 + z^3 + z^4 + z^5 + z^6 + z^7)\Big).$$

Al realizar los cuatro productos, se obtiene el polinomio:

$$\begin{aligned} & 1 + z + z^2 + z^5 + z^6 + z^7 + x(z^2 + z^5 + z^6 + z^7) \\ + \;& x^2(1 + z^4 + z^5 + z^7) + x^3(z + z^7) \\ + \;& y\Big(1 + z^4 + z^6 + x(1 + z + z^3 + z^5 + z^6 + z^7) \\ + \;& x^2(1 + z + z^2 + z^3 + z^7) + x^3(z + z^3 + z^5 + z^6 + z^7)\Big) \\ + \;& y^2\Big(z + z^2 + z^3 + z^4 + z^5 + x(1 + z + z^2 + z^3 + z^6 + z^7) \\ + \;& x^2(1 + z + z^3 + z^7) + x^3(z^6 + z^7)\Big) \\ + \;& y^3\Big(z + z^2 + z^6 + z^7 + x(z^6 + z^7) \\ + \;& x^2(1 + z^6 + z^7) + x^3(1 + z^5)\Big) \end{aligned}$$

Finalmente, se suma la llave $k^1(x,y)$ y se obtiene el polinomio:

$$\begin{aligned} m^1(x,y) \;=\; & z^4 + z^7 + x(z + z^2 + z^3 + z^6) \\ & +x^2(1 + z + z^2 + z^6) + x^3(1 + z + z^2 + z^3 + z^5 + z^6 + z^7) \\ & +y\Big(1 + z^2 + z^3 + z^4 + z^5 + z^7 + x(1 + z^2 + z^3 + z^4 + z^5 + z^6 + z^7) \\ & +x^2(z + z^2 + z^5 + z^7) + x^3(z + z^2 + z^4 + z^5 + z^6)\Big) \\ & +y^2\Big(1 + z^6 + z^7 + x(1 + z^2 + z^3 + z^5 + z^6 + z^7) \\ & +x^2(1 + z^2 + z^4 + z^5 + z^6 + z^7) + x^3(1 + z^2 + z^5)\Big) \\ & +y^3\Big(z + z^5 + z^7 + x(z + z^2 + z^3 + z^4 + z^6) \\ & +x^2(z^5 + z^6) + x^3(1 + z^2 + z^4 + z^5)\Big). \end{aligned}$$

Después, de 8 ejecuciones de la ronda anterior se obtiene

$$\begin{aligned} m^9(x,y) &= 1+z+z^2+z^4+z^5+z^6+z^7+x(1+z+z^2+z^4+z^5+z^6+z^7) \\ &\quad +x^2(z^2+z^3+z^4+z^7)+x^3(z+z^5) \\ &\quad +y\Big(1+z+z^2+z^3+x(1+z^3+z^4) \\ &\quad +x^2(1+z^2)+x^3(1+z+z^2+z^5+z^7)\Big) \\ &\quad +y^2\Big(z+z^3+z^4+z^5+z^6+z^7+x(1+z+z^3+z^5+z^6) \\ &\quad +x^2(z^2+z^5+z^7)+x^3(1+z+z^2+z^7)\Big) \\ &\quad +y^3\Big(1+z+z^3+z^5+z^6+z^7+x(1+z+z^3+z^4+z^5+z^7) \\ &\quad +x^2(z^2+z^5+z^7)+x^3(z+z^6)\Big). \end{aligned}$$

Con la ronda final se obtiene el mensaje cifrado

$$\begin{aligned} c(x,y) &= 1+z+z^2+z^3+z^4+z^5+z^7+x(z^2+z^4+z^7) \\ &\quad +x^2(z+z^2+z^4+z^7)+x^3(1+z+z^2+z^4+z^7) \\ &\quad +y\Big(1+z^4+x(1+z^2+z^3+z^4+z^5+z^6+z^7) \\ &\quad +x^2(z+z^2+z^3+z^4+z^5+z^6+z^7)+x^3(z^2+z^4+z^7)\Big) \\ &\quad +y^2\Big(z+z^3+z^4+z^5+z^7+x(z^2+z^3+z^4+z^5) \\ &\quad +x^2(z+z^4+z^5+z^6+z^7)+x^3(1+z^2+z^4+z^5)\Big) \\ &\quad +y^3\Big(1+z+z^4+z^5+z^6+z^7+x(1+z+z^3+z^7) \\ &\quad +x^2(1+z+z^3+z^4+z^6)+x^3(1+z+z^2+z^3+z^4+z^5+z^7)\Big). \end{aligned}$$

Conclusiones

En este trabajo se ha presentado una descripción algebraica de los sistemas de cifrado DES y AES. Para el caso de AES, se siguió de cerca la representación polinomial que Joachim Rosenthal hace en [26]. En el caso de DES, se usaron varios resultados del Álgebra Conmutativa y la Teoría de Campos Finitos para describirlo. Por ejemplo, para representar tanto los bloques de texto como la llave del sistema, se usó el anillo de polinomios $\mathcal{A}_{8,8}$, mientras que la mitad derecha y la mitad izquierda de un bloque de texto se distinguieron como polinomios en el anillo $\mathcal{A}_{8,4}$. Aprovechando las operaciones polinomiales que se pueden definir en estas estructuras, se pueden caracterizar algunos componentes de DES como las permutaciones IP y IP^{-1}. Otros componentes como las tablas de permutación P, PC_1, PC_2 y las cajas de substitución se pueden describir mediante operaciones de elementos en algunos campos finitos tales como $\mathbb{F}_{2^8}$, $\mathbb{F}_{2^7}$ y $\mathbb{F}_{2^4}$. Mediante polinomios de permutación, calculados con la Fórmula de Interpolación de Lagrange, y productos monomiales se obtuvo una serie de operaciones polinomiales para describir cada una de las permutaciones mencionadas anteriormente. Finalmente, al agrupar todas las operaciones polinomiales previas se pudo establecer tanto el algoritmo de cifrado como el algoritmo de descifrado para DES. Esta representación es la parte central de este trabajo, cumpliendo así, con el objetivo planteado al principio de este escrito.

Enseguida, repasamos algunos de los logros más importantes que conforman la representación polinomial del sistema DES dada en el capítulo 3.

Un resultado importante fue la representación de las cajas de substitución. Como se recordará, en estas tablas reside la seguridad del sistema DES, los polinomios calculados tienen coeficientes en el campo $\mathbb{F}_{2^4}$ y se requirieron 4 para representar

cada S-caja. Por ejemplo, los polinomios que reemplazan a la caja S1 son:

$$\begin{aligned}
s1_1(v) &= x + x^2 + x^3 + v^3x^3 + v^7(1 + x^2) + v^{14}(1 + x^2) \\
&\quad +v^9(x + x^2) + v(1 + x + x^2) + v^4(1 + x + x^2) + v^{10}(1 + x + x^2) \\
&\quad +v^{13}(1 + x + x^2) + v^6(1 + x^3) + v^2(x + x^3) + v^5(x + x^3) \\
&\quad +v^8(x^2 + x^3) + v^{12}(x + x^2 + x^3) + v^{11}(1 + x + x^2 + x^3), \\
s1_2(v) &= v + v^5 + v^8x + v^{13}x + v^7x^3 + v^9(1 + x) + v^2(1 + x^2) \\
&\quad +v^3(1 + x^2) + v^6(1 + x^3) + v^{10}(1 + x^3) + v^{14}(1 + x + x^3) \\
&\quad +v^{12}(x^2 + x^3) + v^{11}(1 + x^2 + x^3) + v^4(x + x^2 + x^3), \\
s1_3(v) &= x^2 + v^3x^2 + v^4x^2 + v^5x^2 + v^{10}x^2 + v^{14}x^3 \\
&\quad +v^{11}(1 + x^2) + v^7(1 + x + x^2) + v^6(1 + x^3) + v^{13}(1 + x + x^3) \\
&\quad +v(x^2 + x^3) + v^9(x + x^2 + x^3) + v^{12}(1 + x + x^2 + x^3), \\
s1_4(v) &= 1 + x + v^5x + v^{14}x + x^2 + x^3 \\
&\quad +v^{12}(1 + x) + v^9(x + x^2) + v^2(1 + x^3) + v^8(1 + x^3) \\
&\quad +v(1 + x + x^3) + v^7(x^2 + x^3) + v^{11}(x^2 + x^3) + v^4(1 + x^2 + x^3).
\end{aligned}$$

En cuanto a las permutaciones del sistema DES como $PC1$, $PC2$, IP, IP^{-1}, también se usaron algunos polinomios de permutación y operaciones en diferentes anillos de polinomios para representarlas. Por ejemplo, la permutación $PC1$ se definió en el anillo de polinomios $\mathcal{A}_{8,8}$ y las transformaciones que se aplicaron en este anillo son:

$$\begin{aligned}
r(x, y) &\xrightarrow{\pi} \sum_{i=0}^{7}\sum_{j=0}^{7} r_{ij}x^j y^i, \\
r(y, x) &\xrightarrow{g} \sum_{i=0}^{7}(x^7\sum_{j=0}^{7} r_{ij}x^{7j})y^i.
\end{aligned}$$

Mientras que los polinomios de permutación obtenidos para esta permutación están definidos en $\mathbb{F}[u]$:

$$\begin{aligned}
pc1_1(u) &= (1+y^2+y^6)u + (1+y^3+y^4+y^7)u^2 + (1+y^2+y^3+y^4)u^4 \\
&\quad +(1+y^4+y^6)u^8 + (1+y+y^5+y^6+y^7)u^{16} \\
&\quad +(y+y^5+y^6+y^7)u^{32} + (y+y^3+y^4+y^6+y^7)u^{64} \\
&\quad +(y+y^3+y^6)u^{128}, \\
pc1_2(u) &= (1+y^2+y^5)u + y^2u^2 + (y^3+y^4+y^5)u^4 \\
&\quad +(1+y+y^2+y^3+y^4+y^5+y^7)u^8 + (y+y^3+y^4+y^6)u^{16} \\
&\quad +(1+y^3+y^4+y^5+y^7)u^{32} + (y+y^2+y^3+y^4+y^5)u^{64} \\
&\quad +(y+y^3+y^4+y^5+y^6)u^{128}, \\
pc1_3(u) &= (y^3+y^6)u + (1+y+y^3)u^2 + (1+y^2+y^6+y^7)u^4 \\
&\quad +(1+y+y^4+y^6+y^7)u^8 + (y+y^3+y^5+y^6)u^{16} \\
&\quad +(y^2+y^4+y^6)u^{32} + (y+y^2+y^7)u^{64} \\
&\quad +(1+y+y^4+y^6+y^7)u^{128}.
\end{aligned}$$

Finalmente, se ha proporcionado tanto un algoritmo de cifrado como uno de descifrado del sistema DES en términos de operaciones polinomiales dentro de varios campos finitos y algunos anillos de polinomios. Además, se llevó a cabo la implementación de estos algoritmos en *Mathematica v5.0*. Todo esto conforma las representación polinomial del sistema de cifrado DES. El algoritmo de cifrado se puede describir de la siguiente manera:

$$\begin{aligned}
m'(x,y) &:= IP(m(x,y)) = l_0(x,y) + y^4 r_0(x,y) \\
&\text{Para} \quad i = 1, ..., 16 \\
&\qquad r_i(x,y) := l_{i-1}(x,y) + f(r_{i-1}(x,y), k_i(x,y)) \\
&\qquad l_i(x,y) := r_{i-1}(x,y) \\
&\qquad m'_i(x,y) := l_i(x,y) + y^4 r_i(x,y) \\
c(x,y) &:= IP^{-1}(m'_{16}(x,y)).
\end{aligned}$$

Mientras que el algoritmo de descifrado es:

$$\begin{aligned} c'(x,y) &:= IP(c(x,y)) = l_{16}(x,y) + y^4 r_{16}(x,y) \\ \\ \text{Para} \quad & j = 16 - i \text{ con } i = 1, ..., 16 \\ & r_j(x,y) := l_{j+1}(x,y) \\ & l_j(x,y) := r_{j+1}(x,y) + f(l_{j+1}(x,y), k_{j+1}(x,y)) \\ & c'_j(x,y) := l_j(x,y) + y^4 r_j(x,y) \\ \\ m(x,y) &:= IP^{-1}(c'_0(x,y)). \end{aligned}$$

Una diferencia notable entre esta descripción polinomial y la tradicional es el tiempo de ejecución de sus implementaciones correspondientes. Por ejemplo, usando la función **Timing** de *Mathematica v5.0*, hemos podido apreciar que el cifrado del mensaje *"SaludosTerricola"*, presentado en los capítulos II y III, se lleva a cabo en 0.016 segundos con la versión de DES tradicional, **DES**. Mientras que con la implementación polinomial de DES, **DESPolinomial**, el tiempo de cifrado es de 2.044 segundos. También, el cifrado del archivo correspondiente a la portada de este trabajo, *portada.tex* (594 bytes), se realizó en 146.859 segundos con **DESPolinomial**. Entretanto, el cifrado llevado a cabo con **DES** duró solamente 0.39 segundos. La razón principal de estas diferencias radica en el costo computacional que tienen algunos componentes básicos de DES en la implementación polinomial. Cuando se representa una caja de substitución de DES mediante 4 polinomios de permutación en el campo finito $\mathbb{F}_{2^4}$, el tiempo de ejecución es mayor que el tiempo de ejecución al representar la S-caja como una tabla en la versión tradicional de DES. Por otro lado, las tablas de permutación como P, PC_1 y PC_2 involucran, además de productos monomiales, varios polinomios de permutación en distintos campos finitos, lo cual contribuye también con un tiempo de ejecución mayor que el de sus análogos en la implementación tradicional de DES. Estas pruebas se realizaron en un equipo Windows 7 con procesador Intel Core i5 (2.53 Ghz). Queda claro que la implementación polinomial de DES no mejora el tiempo de ejecución que ofrece DES tradicional por lo que no resulta útil para fines prácticos. Sin embargo, en este trabajo se ha tratado de explorar la posible estructura matemática que DES pudiera tener. En este sentido, la implementación polinomial de DES nos permite apreciar las operaciones polinomiales encontradas para DES.

La fuerte estructura algebraica que presenta AES ha motivado a los investigadores a explorarlo através de novedosos ataques algebraicos [14, 10, 7]. Debido a esto, recientemente, las representaciones algebraicas de otros esquemas de cifrado por

bloque va en aumento, por ejemplo, DES, Serpent, entre otros, [11, 31]. Nicolas Courtois y Gregory Bard en [11] dejan abierta la posibilidad de encontrar mejores métodos para expresar DES como un sistema de ecuaciones enfatizando que es mejor tener un sistema con ecuaciones de bajo grado a pesar de que su tamaño se incremente. Esto se debe a que las técnicas para resolver estos sistemas de ecuaciones son aún poco eficientes y otras sólo son útiles para casos particulares o para versiones reducidas de los esquemas de cifrado. Por ejemplo, el uso de bases de Gröbner para simplificar los sistemas de ecuaciones derivados de esquemas de cifrado es común. Sin embargo, esta técnica sólo es factible para un número pequeño de variables [9]. Por ello, dichas técnicas representan también temas de actual investigación. Si, bien, este trabajo no representa la solución a la pregunta planteada por los autores de [11], puede considerarse un intento en esa dirección.

Perspectivas

Este trabajo abre las siguientes líneas de trabajo a futuro:

1. Encontrar el sistema de ecuaciones asociado a los polinomios de permutación que se encontraron en este trabajo. Se puede explorar la posibilidad de simplificar las ecuaciones obtenidas en las S-cajas para, posteriormente, intentar montar un ataque algebraico para DES.

2. Representar polinomialmente otros sistemas de cifrado conocidos con el objetivo de reconocer sus estructuras algebraicas. Es probable que mientras más y mejores representaciones algebraicas de los esquemas de cifrado por bloque haya, más posibilidades se tienen de mejorar las técnicas para simplificar y solucionar estos sistemas.

3. Usar polinomios de permutación en más de una variable para representar las S-cajas de DES. Como una posibilidad para mejorar la expresión de DES como un sistema de ecuaciones se pueden usar polinomios de permutación en más de una variable.

4. Encontrar condiciones para determinar cuáles y cuántos de todos los mapeos de un campo finito en sí mismo son polinomios de permutación. Este tipo de polinomios son todo un tema de investigación muy importante y con muchas aplicaciones, por ejemplo, en la Teoría de Códigos, Criptografía, Turbo Códigos, entre otros.

Apéndice A

PC1perm

```
PC1perm[p_] :=
    Module[{l, m, tm, rm, mm, ll, llt, l5, r1, tr1, mmxy, poli, polixxx,
        polinomios, pol1, pol2, pol3, clUlt},
    pol1 = {1 + y^2 + y^6, 1 + y^3 + y^4 + y^7, 1 + y^2 + y^3 + y^4,
        1 + y^4 + y^6, 1 + y + y^5 + y^6 + y^7, y + y^5 + y^6 + y^7,
          y + y^3 + y^4 + y^6 + y^7, y + y^3 + y^6};
    pol2 = {1 + y^2 + y^5, y^2, y^3 + y^4 + y^5,
        1 + y + y^2 + y^3 + y^4 + y^5 + y^7, y + y^3 + y^4 + y^6,
        1 + y^3 + y^4 + y^5 + y^7, y + y^2 + y^3 + y^4 + y^5,
        y + y^3 + y^4 + y^5 + y^6};
    pol3 = {y^3 + y^6, 1 + y + y^3, 1 + y^2 + y^6 + y^7,
        1 + y + y^4 + y^6 + y^7, y + y^3 + y^5 + y^6,
        y^2 + y^4 + y^6, y + y^2 + y^7, y^2 + y^3 + y^5 + y^6 + y^7};
    l = CoefficientList[p, {y, x}];
    m = PadRight[l, {8, 8}];
    tm = Transpose[m];
    rm = Reverse /@ tm;
    mm = Transpose[rm];
    poli = mm . {1, y, y^2, y^3, y^4, y^5, y^6, y^7};
    polinomios = {};
    For[j = 1, j <= 4,
        potencias = {};
        For[i = 0,  i <= 7,
            temp = PolynomialPowerMod[poli[[j]],2^i,
                {y^8 + y^4 + y^3 + y^2 + 1,2}];
            AppendTo[potencias,temp];
            i++];
        tmpx = PolynomialMod[potencias . pol1,
                {y^8 + y^4 + y^3 + y^2 + 1,2}];
```

```
    AppendTo[polinomios,tmpx];
    j++];
For[j = 5, j <= 8,
    potencias = {};
    For[i = 0,i <= 7,
        temp = PolynomialPowerMod[poli[[j]],2^i,
            {y^8 + y^4 + y^3 + y^2 + 1,2}];
        AppendTo[potencias,temp];
        i++];
    tmpx = PolynomialMod[potencias . pol2,
            {y^8 + y^4 + y^3 + y^2 + 1,2}];
    AppendTo[polinomios, tmpx];
    j++];
ll = PadRight[CoefficientList[polinomios, y], {8,8}];
llt = Transpose[ll];
l5 =  RotateRight[llt[[5]], 4];
rl =  ReplacePart[llt, l5, 5];
trl = Transpose[rl];
poli = trl . {1, y, y^2, y^3, y^4, y^5, y^6,y^7};
ultimos = {};
For[j = 1, j <= 8,
    potencias = {};
    For[i = 0,i <= 7,
        temp = PolynomialPowerMod[poli[[j]],2^i,
            {y^8 + y^4 + y^3 + y^2 + 1,2}];
        AppendTo[potencias,temp];
        i++];
    tmpx = PolynomialMod[potencias . pol3,
        {y^8 + y^4 + y^3 + y^2 + 1,2}];
    AppendTo[ultimos,tmpx];
    j++];
clUlt = CoefficientList[ultimos, y];
mmxy = PadRight[clUlt, {8, 8}];
polixxx = Transpose[mmxy] . {1, x, x^2, x^3, x^4, x^5, x^6, x^7};
Return[polixxx . {1, y, y^2, y^3, y^4, y^5, y^6, y^7}]]
```

PC2Aperm

```
PC2Aperm[p_] :=
  Module[{l, lt, cl, ml, row1, row2, row3, row4, stp1, stp2, stp3, stp4, stp5,
       stp6, stp7, stp8, polis, pol, polpol, polix, permpols, pot, col, p6,
        p7, q},
    pol = {{1 + y^2, 1, y + y^2 + y^3, y^3}, {y + y^2, 1, 1 + y + y^2, 1},
           {1 + y^2, 1 + y + y^2, y + y^2 + y^3, 1 + y^2 + y^3},
           {y, y^3, 1 + y^2, 1 + y^2 + y^3},
           {1 + y + y^2 + y^3, y + y^3, y, y + y^2}};
    p6 = {1 + y + y^2, y + y^2, 1, y^2};
    p7 = {y + y^3, y^2, y^2, y};
    polpol = {{1 + x + x^2 + x^3 + x^5, x^2, x^3 + x^4 + x^6,
               x + x^3 + x^4 + x^5, x^2, x + x^3 + x^4 + x^6, 1 + x^2 + x^4},
           {1 + x + x^2 + x^5 + x^6, 1 + x + x^2 + x^4 + x^5 + x^6,
               x + x^2 + x^3 + x^5 + x^6, x^4 + x^5 + x^6, x + x^4,
               x + x^2 + x^5, x + x^3 + x^4},
           {x + x^3 + x^6, 1 + x + x^3 + x^5 + x^6, x + x^6,
               x + x^2 + x^4 + x^6, 1 + x^3 + x^5 + x^6, x + x^2 + x^5,
               x^3 + x^4 + x^5 + x^6},
           {1 + x^2 + x^5, x^5 + x^6, 1 + x + x^3 + x^6, 1 + x^6,
               x + x^2 + x^3 + x^4 + x^6, x + x^2, 1 + x + x^2 + x^4 + x^5}};
    l = PadRight[CoefficientList[p, {y, x}], {4, 7}];
    lt = Transpose[l];
    row2 = RotateRight[lt[[2]], 1];
    stp1 = ReplacePart[lt, row2, 2];
    l = Transpose[stp1];
    stp2 = {RotateRight[l[[1]], 6], RotateRight[l[[2]], 5],
        RotateRight[l[[3]], 3], RotateRight[l[[4]], 3]};
    lt = Transpose[stp2];
    polis = lt . {1, y, y^2, y^3};
    permpols = {};
    For[j = 2, j <= 6,
        potencias = {};
        For[i = 0, i <= 3,
                temp = PolynomialPowerMod[polis[[j]], 2^i, {y^4 + y + 1, 2}];
                AppendTo[potencias, temp];
                i++];
        tmpx = PolynomialMod[potencias.pol[[j - 1]], {y^4 + y + 1, 2}];
        AppendTo[permpols, tmpx];
      j++];
```

```
stp3 = Join[{polis[[1]]}, permpols, {polis[[7]]}];
l = PadRight[CoefficientList[stp3, y], {7, 4}];
lt = Transpose[l];
row2 = RotateRight[lt[[2]], 4];
row3 = RotateRight[lt[[3]], 1];
stp4 = {lt[[1]], row2, row3, lt[[4]]};
lt = Transpose[stp4];
q = lt[[3]] . {1, y, y^2, y^3};
pot = {};
For[i = 0, i <= 3,
    temp = PolynomialPowerMod[q, 2^i, {y^4 + y + 1, 2}];
    AppendTo[pot, temp];
    i++];
col = PolynomialMod[pot.p6, {y^4 + y + 1, 2}];
stp5 = ReplacePart[lt, PadRight[CoefficientList[col, y], 4], 3];
l = Transpose[stp5];
row2 = RotateRight[l[[2]], 3];
row3 = RotateRight[l[[3]], 1];
stp6 = {l[[1]], row2, row3, l[[4]]};
lt = Transpose[stp6];
q = lt[[1]] . {1, y, y^2, y^3};
pot = {};
For[i = 0, i <= 3,
    temp = PolynomialPowerMod[q, 2^i, {y^4 + y + 1, 2}];
    AppendTo[pot, temp];
    i++];
col = PolynomialMod[pot.p7, {y^4 + y + 1, 2}];
stp7 = ReplacePart[lt, PadRight[CoefficientList[col, y], 4], 1];
l = Transpose[stp7];
row3 = RotateRight[l[[3]], 5];
stp8 = ReplacePart[l, row3, 3];
polis = stp8 . {1, x, x^2, x^3, x^4, x^5, x^6};
permpols = {};
For[j = 1, j <= 4,
    potencias = {};
    For[i = 0, i <= 6,
            temp = PolynomialPowerMod[polis[[j]], 2^i, {x^7 + x + 1, 2}];
            AppendTo[potencias, temp];
            i++];
    tmpx = PolynomialMod[potencias.polpol[[j]], {x^7 + x + 1, 2}];
```

```
      AppendTo[permpols, tmpx];
    j++];
  cl = PadRight[CoefficientList[#1, x], 7] & /@ permpols;
  ml = ReplacePart[#1, 0, 7] & /@ cl;
  polix = ml . {1, x, x^2, x^3, x^4, x^5, x^6};
  Return[polix . {1, y, y^2, y^3}]]
```

PC2Bperm

```
PC2Bperm[p_] :=
  Module[{l, lt, ml, cl, row1, row2, row3, row4, stp1, stp2, stp3, stp4, stp5,
      stp6, stp7, stp8, stp9, stp10, polis, pol, permpols, pot, col, col1,
    col2, col3, p1, p2, p3, p4, p5, p6, p7, polpol, q, polix},
  p1 = {1 + y^2, 1, y + y^2 + y^3, y^3};
  p2 = {1 + y + y^2 + y^3, y + y^3, y, y + y^2};
  p3 = {y + y^2, 1, 1 + y + y^2, 1};
  p4 = {y + y^2, 1 + y^3, 1 + y + y^3, 1 + y^2};
  p5 = {1 + y + y^3, 1 + y^2, 0, 1 + y + y^2 + y^3};
  p6 = {1 + y^2, y + y^2, y + y^2 + y^3, 1 + y^2};
  p7 = {0, 1 + y + y^3, y^2 + y^3, 1 + y};
  polpol = {{1 + x + x^2 + x^3 + x^6, x^3 + x^5, x + x^4 + x^5 + x^6,
      1 + x^2 + x^4 + x^6, x^3 + x^5, x + x^6, x + x^5},
    {1 + x + x^2 + x^5 + x^6, 1 + x + x^2 + x^4 + x^5 + x^6,
      x + x^2 + x^3 + x^5 + x^6, x^4 + x^5 + x^6, x + x^4, x + x^2 + x^5,
      x + x^3 + x^4},
    {1 + x + x^2 + x^3 + x^6, x^3 + x^5, x + x^4 + x^5 + x^6,
      1 + x^2 + x^4 + x^6, x^3 + x^5, x + x^6, x + x^5},
    {1 + x + x^2 + x^3 + x^6, x^3 + x^5, x + x^4 + x^5 + x^6,
      1 + x^2 + x^4 + x^6, x^3 + x^5, x + x^6, x + x^5}};
  l = PadRight[CoefficientList[p, {y, x}], {4, 7}];
  lt = Transpose[l];
  col1 = RotateRight[lt[[1]], 3];
  stp1 = ReplacePart[lt, col1, 1];
  l = Transpose[stp1];
  row1 = RotateRight[l[[1]], 1];
  row3 = RotateRight[l[[3]], 1];
  stp2 = {row1, l[[2]], row3, l[[4]]};
  lt = Transpose[stp2];
  col2 = RotateRight[lt[[2]], 3];
  col3 = RotateRight[lt[[3]], 1];
  stp3 = {lt[[1]], col2, col3, lt[[4]], lt[[5]], lt[[6]], lt[[7]]};
```

```
q = stp3[[5]] . {1, y, y^2, y^3};
pot = {};
For[i = 0, i <= 3,
    temp = PolynomialPowerMod[q, 2^i, {y^4 + y + 1, 2}];
    AppendTo[pot, temp];
    i++];
col = PolynomialMod[pot.p1, {y^4 + y + 1, 2}];
stp4 = ReplacePart[stp3, PadRight[CoefficientList[col, y], 4], 5];
q = stp3[[6]] . {1, y, y^2, y^3};
pot = {};
For[i = 0, i <= 3,
    temp = PolynomialPowerMod[q, 2^i, {y^4 + y + 1, 2}];
    AppendTo[pot, temp];
    i++];
col = PolynomialMod[pot.p2, {y^4 + y + 1, 2}];
stp4 = ReplacePart[stp4, PadRight[CoefficientList[col, y], 4], 6];
lt = Transpose[stp4];
row2 = RotateRight[lt[[2]], 1];
stp5 = ReplacePart[lt, row2, 2];
lt = Transpose[stp5];
q = lt[[2]] . {1, y, y^2, y^3};
pot = {};
For[i = 0, i <= 3,
    temp = PolynomialPowerMod[q, 2^i, {y^4 + y + 1, 2}];
    AppendTo[pot, temp];
    i++];
col = PolynomialMod[pot.p3, {y^4 + y + 1, 2}];
lt = ReplacePart[lt, PadRight[CoefficientList[col, y], 4], 2];
q = lt[[4]] . {1, y, y^2, y^3};
pot = {};
For[i = 0, i <= 3,
    temp = PolynomialPowerMod[q, 2^i, {y^4 + y + 1, 2}];
    AppendTo[pot, temp];
    i++];
col = PolynomialMod[pot.p4, {y^4 + y + 1, 2}];
stp6 = ReplacePart[lt, PadRight[CoefficientList[col, y], 4], 4];
l = Transpose[stp6];
row2 = RotateRight[l[[2]], 6];
row4 = RotateRight[l[[4]], 3];
stp7 = {l[[1]], row2, l[[3]], row4};
```

```
lt = Transpose[stp7];
q = lt[[2]] . {1, y, y^2, y^3};
pot = {};
For[i = 0, i <= 3,
    temp = PolynomialPowerMod[q, 2^i, {y^4 + y + 1, 2}];
    AppendTo[pot, temp];
    i++];
col = PolynomialMod[pot.p5, {y^4 + y + 1, 2}];
lt = ReplacePart[lt, PadRight[CoefficientList[col, y], 4], 2];
q = lt[[4]] . {1, y, y^2, y^3};
pot = {};
For[i = 0, i <= 3,
    temp = PolynomialPowerMod[q, 2^i, {y^4 + y + 1, 2}];
    AppendTo[pot, temp];
    i++];
col = PolynomialMod[pot.p6, {y^4 + y + 1, 2}];
stp8 = ReplacePart[lt, PadRight[CoefficientList[col, y], 4], 4];
l = Transpose[stp8];
row3 = RotateRight[l[[3]], 3];
stp9 = ReplacePart[l, row3, 3];
lt = Transpose[stp9];
q = lt[[6]] . {1, y, y^2, y^3};
pot = {};
For[i = 0, i <= 3,
    temp = PolynomialPowerMod[q, 2^i, {y^4 + y + 1, 2}];
    AppendTo[pot, temp];
    i++];
col = PolynomialMod[pot.p7, {y^4 + y + 1, 2}];
stp10 = ReplacePart[lt, PadRight[CoefficientList[col, y], 4], 6];
l = Transpose[stp10];
polis = l . {1, x, x^2, x^3, x^4, x^5, x^6};
permpols = {};
For[j = 1, j <= 4,
    potencias = {};
    For[i = 0, i <= 6,
            temp = PolynomialPowerMod[polis[[j]], 2^i, {x^7 + x + 1, 2}];
            AppendTo[potencias, temp];
            i++];
    tmpx = PolynomialMod[potencias.polpol[[j]], {x^7 + x + 1, 2}];
    AppendTo[permpols, tmpx];
```

```
      j++];
    cl = PadRight[CoefficientList[#1, x], 7] & /@ permpols;
    ml = ReplacePart[#1, 0, 7] & /@ cl;
    polix = ml . {1, x, x^2, x^3, x^4, x^5, x^6};
    Return[polix . {1, y, y^2, y^3}]]
```

Subllaves

```
Subllaves[p_, nr_Integer] :=
  Module[{l, a, b, c = {}, d = {}, cparts, dparts, cpolis = {}, dpolis = {},
      shifts},
    shifts = {1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1};
    l = PadRight[CoefficientList[p, {y, x}], {8, 8}];
    a = Drop[Join[l[[1]], l[[2]], l[[3]], l[[4]]], -4];
    b = Drop[Join[l[[5]], l[[6]], l[[7]], l[[8]]], -4];
    For[i = 1, i <= nr,
        AppendTo[c, RotateLeft[a, shifts[[i]]]];
        AppendTo[d, RotateLeft[b, shifts[[i]]]];
        a = RotateLeft[a, shifts[[i]]];
        b = RotateLeft[b, shifts[[i]]];
      i++];
    cparts = Partition[#1, 7] & /@ c;
    dparts = Partition[#1, 7] & /@ d;
    For[i = 1, i <= nr,
        l1 = cparts[[i]] . {1, x, x^2, x^3, x^4, x^5, x^6};
        q1 = l1 . {1, y, y^2, y^3};
        AppendTo[cpolis, q1];
        l2 = dparts[[i]] . {1, x, x^2, x^3, x^4, x^5, x^6};
        q2 = l2 . {1, y, y^2, y^3};
        AppendTo[dpolis, q2];
      i++];
    Return[{cpolis, dpolis}]]
```

generaLlaves

```
generaLlaves[k_, nr_Integer] := Module[{pc1, c = d = {},
        s = llaves = {}},
    pc1 = PC1perm[QuitaBitsDeParidad[k]];
    s = Subllaves[pc1, nr];
    c = PC2Aperm /@ s[[1]];
    d = PC2Aperm /@ s[[2]];
    Return[c + Collect[Expand[y^4 d], y]]]
```

IPperm

```
IPperm[p_] := Module[{l, m, tm, rm, ip = {}, polix, rl},
    l = CoefficientList[p, {y, x}];
    m = PadRight[l, {8, 8}];
    tm = Transpose[m];
    rm = Reverse /@ tm;
    For[i = 0, i <= 7,
        AppendTo[ip, rm[[Mod[Quotient[i, 2] + 4*(Mod[i, 2] + 1), 8] + 1]]];
        i++];
    polix = ip . {1, x, x^2, x^3, x^4, x^5, x^6, x^7};
    rl = Partition[polix, 4];
    Return[rl . {1, y, y^2, y^3}]]
```

IPInvperm

```
IPInvperm[p_] := Module[{l, m, rm, t = {}, ipinv, polix},
    l = CoefficientList[p, {y, x}];
    m = PadRight[l, {8, 8}];
    rm = Reverse /@ m;
    For[i = 0, i <= 7,
        AppendTo[t, rm[[Mod[7 Quotient[i, 4] + 2*(Mod[i, 4]) + 1, 8] + 1]]];
        i++];
    ipinv = Transpose[t];
    polix = ipinv . {1, x, x^2, x^3, x^4, x^5, x^6, x^7};
    Return[polix . {1, y, y^2, y^3, y^4, y^5, y^6, y^7}]]
```

Eperm

```
Eperm[p_] := Module[{l, m, longlist, prim, ult, ll, elist, polix},
    l = CoefficientList[p, {y, x}];
    m = PadRight[l, {4, 8}];
    longlist = Join @@ m;
    prim = First[longlist];
    ult = Last[longlist];
    ll = RotateRight[longlist];
    ll = Append[ll, ult];
    ll = Append[ll, prim];
    elist = Partition[ll, 6, 4];
    polix = elist . {1, x, x^2, x^3, x^4, x^5};
    Return[polix . {1, y, y^2, y^3, y^4, y^5, y^6, y^7}]]
```

aplicaSboxes

```
aplicaSboxes[p_] :=
    Module[{l, m, polis, permpols, spolis, pot, lpares, q},
    S = {{1 + x + x^2 + u^5 (1 + x) + u^10 (1 + x) + u^3 (1 + x^2) +
            u^9 (1 + x + x^2) + u (1 + x^3) + u^7 (1 + x^3) +
            u^13 (x + x^3) + u^6 (1 + x + x^3) + u^14 (1 + x + x^3) +
            u^11 (1 + x^2 + x^3) + u^8 (x + x^2 + x^3) + u^2 (1 + x +
            x^2 + x^3) + u^12 (1 + x + x^2 + x^3),
            u^7 + u^14 + u^4 x + u x^2 + u^11 x^2 + u^8 (1 + x) +
            u^6 (1 + x^2) + u^13 (1 + x^2) + u^3 (x + x^3) + u^9 (1 +
            x^2 + x^3) + u^10 (1 + x^2 + x^3) + u^2 (x + x^2 + x^3),
            u^3 + u^14 + x + u^10 x^3 + u^2 (1 + x^2) + u^12 (1 + x^2)
            + u^11 (1 + x + x^2) + u^5 (x + x^3) + u^9 (x + x^3) +
            u^7 (x^2 + x^3) + u^4 (1 + x^2 + x^3) + u^6 (1 + x^2 + x^3)
            + u^8 (x + x^2 + x^3),
            1 + u^14 + x + x^2 + u^9 x^2 + x^3 + u x^3 + u^7 (x + x^2)
            + u^11 (x + x^2) + u^4 (1 + x + x^2) + u^5 (1 + x^3) +
            u^6 (1 + x^3) + u^10 (1 + x^3) + u^12 (1 + x^3) + u^2
            (1 + x^2 + x^3) + u^8(1 + x^2 + x^3) + u^3(1 + x + x^2 + x^3)},
        {1 + u^3 + x + u^8 x + u^11 x + u^12 x + x^2 + x^3 + u^14 (1 + x)
            + u^4 (1 + x^2) + u (1 + x + x^3) + u^2 (1 + x + x^3) +
            u^5 (x^2 + x^3) + u^7 (x^2 + x^3) + u^9 (x^2 + x^3) + u^10
            (1 + x^2 + x^3) + u^13 (1 + x^2 + x^3) + u^6 (1 + x + x^2 + x^3),
            u^6 + u^13 + u^4 x + u^12 x + x^2 + u x^2 + u^7 x^2 + x^3 +
            u^8 x^3 + u^2 (x + x^2) + u^11 (1 + x^3) + u^5 (1 + x + x^3)
            + u^3 (x^2 + x^3) + u^9 (x^2 + x^3) + u^14 (x^2 + x^3) +
            u^10 (1 + x + x^2 + x^3),
            u^7 (1 + x^2) + u^2 (x + x^2) + u^5 (1 + x + x^2) + u^11
            (1 + x + x^2) + u^9 (1 + x^3) + u^3 (x + x^3) + u^10 (x + x^3)
            + u^13 (x + x^3) + u^6 (1 + x + x^3) + u (x^2 + x^3) +
            u^4 (x^2 + x^3) + u^14 (x^2 + x^3) + u^12 (1 + x^2 + x^3),
            1 + x + u x^2 + u^2 x^2 + x^3 + u^13 x^3 + u^5 (1 + x) + u^6
            (1 + x) + u^9 (1 + x) + u^8 (x + x^2) + u^11 (x + x^2) +
            u^12 (1 + x^3) + u^3 (x + x^3) + u^7 (x + x^3) + u^10(x + x^3)
            + u^4 (x + x^2 + x^3)},
        {1 + u^12 x + x^2 + u^3 x^2 + u^13 x^2 + u^6 x^3 + u^9 x^3 +
            u^4 (1 + x) + u^7 (1 + x) + u^8 (1 + x) + u^10 (1 + x + x^2)
            + u (1 + x + x^3) + u^11 (1 + x + x^2 + x^3) +
            u^14 (1 + x + x^2 + x^3),
            1 + u^6 + x + u^14 x + x^3 + u^2 (1 + x) + u^13 (1 + x^2)
```

```
    + u^8 (1 + x^3) + u^12 (1 + x^3) + u^7 (1 + x + x^3) +
    u (1 + x^2 + x^3) + u^10 (1 + x^2 + x^3) +
    u^3 (x + x^2 + x^3) + u^4 (1 + x + x^2 + x^3) +
    u^5 (1 + x + x^2 + x^3) + u^11 (1 + x + x^2 + x^3),
    1 + u^6 + u^9 + u^12 + x + u^7 x + u^10 x^2 + x^3 + u^4 x^3
    + u^11 (1 + x) + u^8 (1 + x^3) + u^3 (x^2 + x^3) +
    u^5 (x^2 + x^3) + u^2 (1 + x^2 + x^3) + u^14 (x + x^2 + x^3) +
    u (1 + x + x^2 + x^3) + u^13 (1 + x + x^2 + x^3),
    u^3 x + x^3 + u (1 + x) + u^6 (1 + x) + u^12 (1 + x^2) +
    u^8 (1 + x + x^2) + u^13 (1 + x + x^2) + u^5 (1 + x^3) +
    u^11 (1 + x^3) + u^7 (x + x^3) + u^9 (1 + x + x^3) + u^4
    (x^2 + x^3) + u^10 (x^2 + x^3) + u^14 (x^2 + x^3)},
{u + u^8 + x + x^2 + u^7 x^2 + u^14 x^2 + x^3 + u^2 x^3 + u^5
    (1 + x^2) + u^6 (1 + x^3) + u^9 (x + x^3) + u^13 (1 + x + x^3)
    + u^4 (1 + x^2 + x^3) + u^10 (1 + x^2 + x^3) + u^11
    (1 + x^2 + x^3) + u^3 (x + x^2 + x^3),
    1 + u^2 + x + u^7 x^2 + x^3 + u^14 x^3 + u^13 (1 + x) +
    u^3 (1 + x^2) + u^11 (x + x^2) + u^9 (x + x^3) +
    u^4 (1 + x + x^3) + u^5 (1 + x + x^3) + u^12 (x + x^2 + x^3) +
    u^6 (1 + x + x^2 + x^3) + u^10 (1 + x + x^2 + x^3),
    1 + u^6 + u^7 + u^11 x + x^2 + u^10 x^2 + u x^3 + u^4 x^3 +
    u^12 (1 + x^2) + u^5 (1 + x + x^2) + u^14 (x + x^3) + u^8
    (1 + x + x^3) + u^2 (x + x^2 + x^3) + u^9 (x + x^2 + x^3) +
    u^3 (1 + x + x^2 + x^3),
    u^2 x + u^6 x + u^12 x + x^2 + u^5 x^2 + x^3 + u^4 x^3 +
    u^7 (1 + x) + u^13 (1 + x^2) + u^10 (1 + x + x^2) + u^9
    (x + x^3) + u (1 + x + x^2 + x^3) + u^8 (1 + x + x^2 + x^3)},
{u^11 x + x^2 + u^14 x^2 + u^4 x^3 + u^13 x^3 + u^2 (1 + x^2) +
    u^10 (1 + x^2) + u^9 (x + x^2) + u^5 (1 + x + x^2) + u^6
    (1 + x^3) + u (x + x^3) + u^7 (x + x^3) + u^8 (1 + x + x^3)
    + u^3 (1 + x + x^2 + x^3) + u^12 (1 + x + x^2 + x^3),
    1 + x + u^6 x + x^2 + u^4 x^3 + u^7 x^3 + u (1 + x) + u^8
    (x + x^2) + u^12 (1 + x + x^2) + u^14 (x + x^3) +
    u^10 (1 + x + x^3) + u^3 (1 + x^2 + x^3) + u^2 (x + x^2 + x^3)
    + u^9 (1 + x + x^2 + x^3),
    u + u^7 + u^11 + x + u^10 x + u^8 x^3 + u^2 (1 + x) + u^13
    (x + x^2) + u^3 (1 + x + x^2) + u^4 (1 + x + x^2) + u^9
    (1 + x^3) + u^5 (x + x^3) + u^6 (1 + x^2 + x^3) + u^14
    (1 + x^2 + x^3),
    1 + u^12 + x^2 + x^3 + u^6 (1 + x) + u^8 (1 + x) + u^10
```

```
    (1 + x^2) + u^14 (1 + x^2) + u (1 + x^3) + u^5 (x + x^3) +
    u^9 (1 + x + x^3) + u^3 (x^2 + x^3) + u^13 (x^2 + x^3) +
    u^11 (1 + x^2 + x^3) + u^2 (1 + x + x^2 + x^3) + u^4
    (1 + x + x^2 + x^3) + u^7 (1 + x + x^2 + x^3)},
{1 + u^4 + u^11 + x + u^6 x^2 + u^2 x^3 + u^10 (1 + x) +
    u^12 (1 + x + x^2) + u^7 (1 + x + x^3) + u^3 (x^2 + x^3) +
    u^14 (x^2 + x^3) + u^5 (x + x^2 + x^3) + u^13 (x + x^2 + x^3)
    + u (1 + x + x^2 + x^3) + u^8 (1 + x + x^2 + x^3),
    1 + x^2 + u^10 x^2 + u^2 x^3 + u (1 + x) + u^7 (1 + x) +
    u^3 (1 + x^2) + u^6 (1 + x^2) + u^14 (x + x^2) + u^4 (1 + x^3)
    + u^12 (1 + x^3) + u^11 (1 + x + x^3) + u^8 (x^2 + x^3) +
    u^5 (x + x^2 + x^3) + u^9 (1 + x + x^2 + x^3) + u^13
    (1 + x + x^2 + x^3),
    1 + u^6 x + u^14 x + x^3 + u^7 x^3 + u^5 (1 + x) + u^9 (1 + x)
    + u^8 (x + x^2) + u^12 (x + x^2) + u^13 (1 + x^3) + u^11
    (1 + x + x^3) + u (1 + x^2 + x^3) + u^2 (1 + x^2 + x^3) + u^4
    (1 + x^2 + x^3) + u^3 (x + x^2 + x^3) + u^10 (x + x^2 + x^3),
    u^8 + u^12 + x + u^10 x^2 + u^3 (1 + x) + u^5 (1 + x^2) + u
    (x + x^2) + u^7 (1 + x + x^2) + u^11 (1 + x + x^2) + u^4
    (x + x^3) + u^9 (x^2 + x^3) + u^14 (1 + x^2 + x^3) + u^6
    (1 + x + x^2 + x^3) + u^13 (1 + x + x^2 + x^3)},
{x + u^2 x^2 + u^7 x^2 + u^8 x^2 + u^10 x^2 + u^5 x^3 + u^6 x^3 +
    u^9 x^3 + u (x + x^2) + u^13 (x + x^2) + u^3 (1 + x^3) +
    u^14 (1 + x^3) + u^11 (x + x^3) + u^4 (x^2 + x^3),
    1 + x + u^4 x + u^7 x^2 + x^3 + u^10 x^3 + u^5 (1 + x^2) +
    u^12 (1 + x^2) + u^13 (1 + x^2) + u (1 + x + x^2) + u^8
    (x + x^3) + u^11 (x + x^3) + u^2 (1 + x + x^2 + x^3) + u^6
    (1 + x + x^2 + x^3) + u^9 (1 + x + x^2 + x^3) + u^14
    (1 + x + x^2 + x^3),
    u + u^7 x + u^12 x + u^4 x^2 + u^6 x^2 + u^8 x^2 + x^3 +
    u^2 x^3 + u^11 (x + x^2) + u^3 (1 + x + x^2) + u^10
    (1 + x + x^2) + u^13 (x + x^3) + u^5 (1 + x + x^3) +
    u^9 (1 + x + x^3) + u^14 (x^2 + x^3),
    x + u^8 x + u^10 x + x^2 + u x^2 + u^9 (1 + x^2) + u^2 (x + x^2)
    + u^4 (1 + x^3) + u^7 (1 + x^3) + u^13 (1 + x^3) + u^6
    (x^2 + x^3) + u^11 (x^2 + x^3) + u^14 (x + x^2 + x^3) + u^12
    (1 + x + x^2 + x^3)},
{1 + u^8 + u^11 + x + u^4 x + x^3 + u (1 + x^2) + u^13 (x + x^2)
    + u^5 (1 + x^3) + u^7 (1 + x + x^3) + u^3 (1 + x^2 + x^3) +
    u^12 (1 + x^2 + x^3) + u^14 (1 + x^2 + x^3) + u^6 (x + x^2 + x^3)
```

```
        + u^9 (x + x^2 + x^3),
        u^4 + u^12 + u^9 x^2 + u^13 x^2 + x^3 + u^11 (1 + x) + u^14
        (1 + x) + u^6 (x + x^2) + u^10 (x + x^2) + u^2 (1 + x + x^3)
        + u^5 (1 + x + x^3) + u^8 (1 + x + x^3) + u (1 + x + x^2 + x^3)
        + u^3 (1 + x + x^2 + x^3),
        x + x^2 + u^8 x^2 + u^14 x^2 + x^3 + u^13 x^3 + u^4 (1 + x) +
        u^10 (x + x^2) + u^11 (1 + x + x^2) + u^7 (1 + x^3) + u^2 (x + x^3)
        + u^12 (x + x^3) + u (1 + x + x^3) + u^6 (1 + x + x^3) + u^9
        (1 + x^2 + x^3) + u^3 (1 + x + x^2 + x^3) + u^5 (1 + x + x^2 + x^3),
        u^8 + x^2 + u^2 x^2 + u^13 x^2 + u^4 x^3 + u (1 + x) + u^12 (1 + x)
        + u^6 (1 + x^2) + u^11 (1 + x^2) + u^14 (x + x^2) + u^9 (x + x^3)
        + u^7 (1 + x + x^3) + u^10 (1 + x + x^3) + u^5 (x + x^2 + x^3)}};
p1 = {x^3 + x^5, 1 + x^2 + x^3, x + x^2 + x^3, x + x^3 + x^4, 1 + x^3 + x^5,
      x^3};
l = CoefficientList[p, {y, x}];
m = PadRight[l, {8, 6}];
polis = m . {1, x, x^2, x^3, x^4, x^5};
permpols = {};
For[j = 1, j <= 8,
    potencias = {};
    For[i = 0, i <= 5,
        temp = PolynomialPowerMod[polis[[j]], 2^i, {x^6 + x + 1, 2}];
        AppendTo[potencias, temp];
        i++];
    tmpx = PolynomialMod[potencias . p1, {x^6 + x + 1, 2}];
    AppendTo[permpols, tmpx];
    j++];
l = CoefficientList[permpols, x];
m = PadRight[#1, 6] & /@  l;
spolis = {};
For[i = 1, i <= 8,
    pol = Take[m[[i]], 4] . {1, x, x^2, x^3};
    row = FromDigits[Take[m[[i]], -2], 2] + 1;
    perm = S[[i]][[row]];
    expos = Exponent[perm, u, List];
    coef = PadRight[CoefficientList[perm, u], 15, 0];
    pot = {};
    For[j = 0,j <= 14,
        If[MemberQ[expos, j],
            tmp = PolynomialPowerMod[pol, j, {x^4 + x + 1, 2}],
```

```
                tmp = 0];
            AppendTo[pot, tmp];
            j++];
        tmpx = PolynomialMod[pot . coef, {x^4 + x + 1, 2}];
        AppendTo[spolis, tmpx];
    i++];
    lpares = Partition[spolis, 2];
    q = Expand[lpares . {1, x^4}];
    Return[q . {1, y, y^2, y^3}]]
```

funcionF

```
funcionF[p_, q_] := Module[{e, z, s},
    e = Eperm[p];
    z = PolynomialMod[e + q, 2];
    s = aplicaSboxes[z];
    Return[Pperm[s]]]
```

desRound

```
desRound[lprev_, rprev_, k_] := Module[{lsig, rsig},
    lsig = rprev;
    rsig = PolynomialMod[lprev + funcionF[rprev, k], 2];
    Return[{lsig, rsig}]]
```

desEncrypt

```
desEncrypt[m_, k_, nr_Integer] :=
  Module[{ek = generaLlaves[k, nr], ipm = IPperm[m], lprev, rprev},
    lprev = ipm[[1]];
    rprev = ipm[[2]];
    For[ronda = 1, ronda <= nr,
        lr = desRound[lprev, rprev, ek[[ronda]]];
        lprev = lr[[1]];
        rprev = lr[[2]];
      ronda++];
    Return[lprev + y^4 rprev]]
```

Bibliografía

[1] Angel, J.J. y Morales-Luna, G., *Breve Descripción de la Criptografía en la Revolución Mexicana*, Revista Digital Universitaria, Vol. 9, No. 3, DGSCA-UNAM, 10 de marzo de 2008.

[2] Biham, E. and Shamir, A., *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, London, 1993.

[3] Bloisi, D. and Iocchi, L., *Image Based Steganography and Cryptography*, Proceedings of the Second International Conference on Computer Vision Theory and Applications, Volume 1, Barcelona, Spain, March 8-11, 2007.

[4] Boghardt, T., *The Zimmermann Telegram: Diplomacy, Intelligence and the American Entry into World War I*, Working Paper Series of The BMW Center for German and European Studies Edmund A. Walsh, School of Foreign Service, Georgetown University, Working Paper No.6-04, November 2003.

[5] Buchmann, J., *Introduction to cryptography*, Undergraduate Texts in Mathematics, Springer-Verlag, New York, 2001.

[6] Campbell, K. and Wiener, M., *DES is not a Group*, CRYPTO '92 Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology, 1992.

[7] Cid, C., Murphy, S. and Robshaw, M., *Algebraic Aspects of the Advanced Encryption Standard*, Springer, 2006.

[8] Coppersmith, D. and Grossman, E., *Generators for certain alternating groups with applications to cryptography*, SIAM Journal of Applied Mathematics Vol. 29, No. 4, 1975.

[9] Courtois, N., Klimov, A., Patarin, J. and Shamir, A., *Efficient algorithms for solving overdefined systems of multivariate polynomial equations*, Advances in Cryptology: Eurocrypt '00, B. Preneel, ed., pp. 392-407, Springer-Verlag, Berlin, 2000.

[10] Courtois, N. and Pieprzyk, J., *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations*, Advances in Cryptology - ASIACRYPT 2002, Lecture Notes in Computer Science, 2002, Volume 2501/2002, 267-287, Springer Verlag, Heidelberg, 2002.

[11] Courtois, N. and Bard, G., *Algebraic Cryptanalysis of the Data Encryption Standard*, Cryptography and Coding, Lecture Notes in Computer Science, 2007, Volume 4887/2007, 152-169, Springer Verlag, Heidelberg, 2007.

[12] Daemen, J. and Rijmen, V., *The design of Rijndael: AES-The Advanced Encryption Standard*, Springer-Verlag, Berlin, Heidelberg, 2002.

[13] Even, S. and Goldreich, O., *DES-like functions can generate the alternating group*, IEEE Transactions on Information Theory Vol. IT-29, No. 6, 1983.

[14] Ferguson, N., Schroeppel, R., and Whiting, D. *A Simple Algebraic Representation of Rijndael*, Proceedings of Selected Areas in Cryptography, 103-111, 2001.

[15] Kahn, D., *The Codebreakers*, The New American Library, Inc., 1973.

[16] Kaliski B.S., Rivest R.L., and Sherman A.T., *Is the Data Encryption Standard a Group? (Results of Cycling Experiments on DES)*, Journal of Cryptology, Vol. 1, No. 1, pp. 3-36, 1988.

[17] Lidl, R. and Niederreiter, H., *Finite Fields, Encyclopedia of Mathematics and its Applications Volume 20* , Cambridge University Press, 2000.

[18] Matsui, M., *Linear Cryptanalysis Method for DES Cipher*, in Advances in Cryptology: Eurocrypt '93, T. Helleseth, ed., Springer-Verlag, Berlin, pp. 386-397, 1994.

[19] Menezes, A., van Oorschot, P. and Vanstone, S., *Handbook of Applied Cryptography*, CRC Press, 1996.

[20] National Institute of Standards and Technology (NIST), *"FIPS-46-3: Data Encryption Standard (DES)"*, 1999. Disponible en: `http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf`

[21] National Institute of Standards and Technology (NIST), *"FIPS-197: Advanced Encryption Standard"* 2001. Disponible en `http://csrc.nist.gov/publications/fips/fips197/fips197.pdf`

[22] National Institute of Standards and Technology (NIST), *"Recommendation for Block Cipher Modes of Operation"* 2001. Disponible en `http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf`

[23] National Institute of Standards and Technology (NIST), *"Report on the Development of the Advanced Encryption Standard (AES)"* 2000. Disponible en `http://csrc.nist.gov/archive/aes/round2/r2report.pdf`

[24] Paar, C., Pelzl, J., *Understanding Cryptography A Textbook for Students and Practicioners*, Springer-Verlag, Berlin, Heidelberg, 2010.

[25] Pieprzyk, J., Hardjono, T. and Seberry, J., *Fundamentals of Computer Security*, Springer-Verlag, Berlin, Heidelberg, 2003.

[26] Rosenthal, J., *A polynomial description of the Rijndael Advanced Encryption Standard*, Journal of Algebra and Its Applications Vol. 2, No. 2, 223-236, 2003.

[27] Schneier, B., *Applied Cryptography Second Edition : Protocols, algorithms, and source code in C*, John Wiley & Sons, Inc., 1996.

[28] Shannon, C., *A Mathematical Theory of Communication*, The Bell System Technical Journal, Vol. 27, pp. 379-423, 623-656, 1948.

[29] Shannon, C., *Communication Theory of Secrecy Systems*, 1949. Disponible en `http://netlab.cs.ucla.edu/wiki/files/shannon 1949.pdf`

[30] Shimoyama, T. and Kaneko, T., *Quadratic Relation of S-box and its Application to the Linear Attack of Full Round DES*, Advances in Cryptology - Crypto 98, volume 1462 of LNCS, pp200-211, Springer-Verlag, 1998.

[31] Singh, B., Alexander, L. and Burman, S., *On Algebraic Relations of Serpent S-Boxes*, Cryptology ePrint Archive, Report 2009/038, 2009. Disponible en: `http://eprint.iacr.org/2009/038.pdf`

[32] Singh, S., *The code book: The science of secrecy from ancient Egypt to quantum cryptography*, Anchor Books, 2000.

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA

ACTA DE EXAMEN DE GRADO

No. 00070
Matrícula: 205384134

DESCRIPCION POLINOMIAL DE LOS SISTEMAS DE CIFRADO DES Y AES

En México, D.F., se presentaron a las 11:00 horas del día 16 del mes de diciembre del año 2011 en la Unidad Iztapalapa de la Universidad Autónoma Metropolitana, los suscritos miembros del jurado:

DR. RUBEN VAZQUEZ MEDINA
DR. HORACIO TAPIA RECILLAS
DR. JOSE NOE GUTIERREZ HERRERA

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

MAESTRO EN CIENCIAS (MATEMÁTICAS APLICADAS E INDUSTRIALES)

DE: PAULO SERGIO GARCIA MENDEZ

PAULO SERGIO GARCIA MENDEZ
ALUMNO

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

Aprobar

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

REVISO

LIC. JULIO CESAR DE LARA ISASSI
DIRECTOR DE SISTEMAS ESCOLARES

DIRECTOR DE LA DIVISIÓN DE CBI

DR. JOSE ANTONIO DE LOS REYES HEREDIA

PRESIDENTE

DR. RUBEN VAZQUEZ MEDINA

VOCAL

DR. HORACIO TAPIA RECILLAS

SECRETARIO

DR. JOSE NOE GUTIERREZ HERRERA