UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO
FACULTAD DE CIENCIAS FÍSICO-MATEMÁTICAS
"Luís Manuel Gutiérrez Rivera"

EL DATA ENCRYPTION STANDARD Y LA ROBUSTEZ DE MAPEOS REGULARES

TESIS

QUE PARA OBTENER EL TITULO DE:
LICENCIADO EN CIENCIAS FÍSICO MATEMÁTICAS

PRESENTA:

JESÚS CASTAÑEDA RIVERA

DIRECTOR DE TESIS:

DR. HUMBERTO CÁRDENAS TRIGOS

MORELIA, MICHOACÁN, ENERO DE 2008.

UNIVERSIDAD MICHOACANA DE
SAN NICOLÁS DE HIDALGO
FACULTAD DE CIENCIAS FÍSICO-MATEMÁTICAS
"Luís Manuel Gutiérrez Rivera"

EL DATA ENCRYPTION STANDARD Y LA ROBUSTEZ DE MAPEOS REGULARES

Por

Jesús Castañeda Rivera

Director de Tesis:

Dr. Humberto Cárdenas Trigos

Morelia, Michoacán
Enero de 2008.

A LA MEMORIA DE MI PADRE:

En este breve trabajo quisiera hacer homenaje ala memoria de mi padre, quien ha sido el incentivo en mis estudios, fuente de reflexión y cariño. A quien agradezco tan bellos momentos asu lado y en el cielo espero se sienta satisfecho con el trabajo logrado.

Por creer en mí y en mí trabajo.

Muchas gracias papá!!

Este trabajo tiene un significado especial, pues representa tantos esfuerzos que ha hecho mi familia para que yo continuara mis estudios, especialmente mis padres, mi mama que me ha acompañado a lo largo de toda mi vida, que me ha dado su amor y su cariño, sus cuidados y su apoyo.

Gracias mamá!!

Agradezco a mis hermanos, por siempre tener una sonrisa para mí y por todo su apoyo para que pudiera llegar a concluir mi carrera profesional.

Gracias ¡!!!!!

AGRADECIMIENTOS

Agradezco de manera especial al profesor ***Humberto Cárdenas Trigos***, amigo y sin lugar ha duda el mejor matemático que he conocido, quien ha sido mi tutor en todos mis estudios, me ha guiado con sencillez y humildad, realmente este breve trabajo no hubiese sido posible sin todas sus enseñanzas y todo el tiempo que me ha dedicado.

Asu lado he tenido la oportunidad de tener un excelente guía en el aprendizaje de las matemáticas. Primeramente, en combinatoria y espacios parcialmente lineales, después en teoría de códigos y criptografía, por regalarme un poco de sus conocimientos, su inmensa creatividad, anécdotas y pensamientos, por ayudarme ha crecer como persona, como estudiante y como matemático.

A mis amigos y compañeros de la Facultad de Ciencias, por su amistad, apoyo y colaboración.

Agradezco a ***Dr. Mario Cesar Suárez Arriaga*** por sus consejos y su constante motivación, ***Dr. Rigoberto Vera Mendoza, Dr. Carlos Cortes Zavala*** y ***Dr. Francisco Domínguez Mota*** por su interés en este trabajo y sus importantes observaciones.

A nuestra ***Facultad de Ciencias Físico Matemáticas "Mat. Manuel Gutiérrez Rivera"de la Universidad Michoacana de San Nicolás de Hidalgo*** por permitirme pasar por sus aulas, por formarme con profesores excepcionales y conocer un poco de ciencia. Gracias al ***Instituto de Matemáticas UNAM Campus Morelia*** por todo su apoyo y disposición a lo largo de mis estudios.

Jesús Castañeda Rivera
Morelia, Michoacán.
Enero de 2008.

CONTENIDO

RESUMEN

El Data Encryption Standard (DES) es un sistema criptográfico standard (desde 1977) para la protección de la información confidencial en sistemas bancarios y comerciales, es el sistema criptográfico más utilizado en la historia de la criptografía moderna. Constantemente ha sido puesto a prueba sobre su seguridad informática y en 1999 el DES fue remplazado por el AES (Advanced Encryption Standard) para proteger la información confidencial de bancos y agencias federales de los Estados Unidos de América. Sin embargo, todos los algoritmos criptográficos posteriores al DES presentan construcciones similares, estos sistemas criptográficos son llamados *sistemas criptográficos simétricos tipo DES*.

La seguridad informática de estos sistemas criptográficos se basa principalmente en funciones que en la sucesión de sus valores cada elemento se repite el mismo número de veces, estas funciones se llaman *regulares* y en el caso particular del DES se les llama *S-cajas*.

Asociado a estas funciones regulares, se define un número llamado robustez de la función regular que se interpreta como una medida de la seguridad que proporciona la función regular en un sistema criptográfico tipo DES; Para funciones regulares más robustas mayor será la seguridad del sistema criptográfico.

El problema de encontrar las mejores funciones regulares ha sido tratado desde 1977, pues en nuestros días es de mucho interés contar con sistemas criptográficos de alta seguridad. La teoría principal se debe a los trabajos de J. Seberry, Zhang, Zheg, Coppersmith y E. Biham.

En este trabajo continuaremos en la búsqueda de encontrar las mejores S-cajas para sistemas criptográficos simétricos en espacios vectoriales pequeños (ver [4], [5], [6]). Proponemos un método eficiente para el cálculo de la robustez de funciones regulares considerando un conjunto de reducciones en el espacio de funciones regulares debido a la acción del grupo afin $Aff(n,2)$. Al parecer, la acción del grupo afín en el espacio de funciones regulares no había sido identificada por otros autores.

Probamos la invarianza de la robustez de una función regular a las transformaciones lineales y a las traslaciones, presentamos un programa de simulación del Data Encryption Standard y calculamos la robustez de todas funciones regulares de $S_{3,2}$ y $S_{4,2}$ mediante la implementación de programas en GAP 3.0 para Linux.

INTRODUCCIÓN

El Data Encryption Standard (DES) es un sistema criptográfico que permite cifrar información confidencial mediante un conjunto de claves secretas. La información confidencial es partida en bloques de 64 bits de información, por ello el DES es conocido como un sistema de cifrado a bloques con clave secreta.

El Data Encryption Standard fue construido en 1975 por la IBM en respuesta a la convocatoria del Nacional Bureau of Standards (NBS) en la que solicitaba el desarrollo de nuevos sistemas criptográficos (1973). Desde entonces el DES ha sido sometido a multitud de estudios y controversias acerca de su seguridad. Se adopto como standard en 1977 para aplicaciones no clasificadas, aplicaciones en las redes de comunicaciones bancarias y comerciales, y es el sistema de cifrado más utilizado mundialmente.

En el Data Encryption Standard los mensajes son divididos en bloques de 64 bits, a los cuales se les aplican 16 claves de 48 bits y finalmente el mensaje cifrado es también de 64 bits. El método de cifrado consiste en aplicar en 16 ocasiones una función especial DES con una clave k, el método de descifrado consiste en aplicar en 16 ocasiones la función DES inversa.

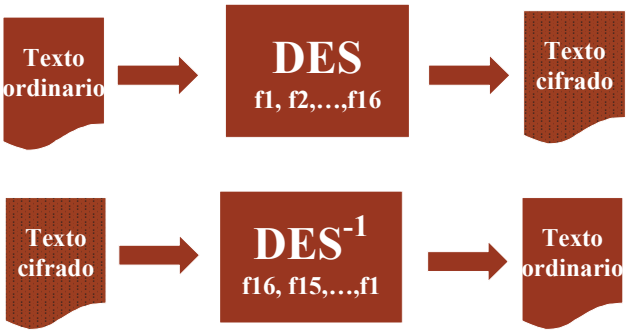

Fig. 1. Representación del sistema criptográfico Data Encryption Standard (DES)

El Data Encryption Standard se compone de una permutación inicial de 64 elementos, una función de expansión que transforma vectores de 32 bits en vectores de 48 bits, un conjunto de claves secretas de 48 bits, una función $S \square\square_{4\square} \rightarrow \square_{32}$ compuesta de 8 funciones llamadas S-cajas $S \square\square_{\square} \rightarrow \square_{4}$ que transforman a cada bloque de 6 bits en un bloque de 4 bits, una permutación final de 32 elementos y una función de selección.

Cabe mencionar, que la seguridad de este sistema se encuentra asociada a la estructura de sus S-cajas. Las S-cajas son funciones entre espacios vectoriales sobre el campo de dos elementos que llamaremos funciones regulares.

Sea $\square_n$ el espacio vectorial de dimensión *n* sobre el campo de dos elementos $\square_2$. *Una función* $S \square\square_n \to \square_\square$ *se llama regular si en la sucesión de valores de la función cada elemento j de* $\square_\square$ *aparece* $2^{n-\square}$ *veces. Podemos asociar a cada función regular un número* $0 \le \square\square\square(S) < \square$ *llamado robustez de S que interpretamos como una medida de la seguridad del sistema criptográfico, si este número se aproxima a 1 diremos que el sistema es muy seguro, si se aproxima a cero el sistema será inseguro.*

Para calcular la robustez de una función regular S es necesario primeramente calcular su tabla de distribución diferencial que es una matriz $\square^{(S)} = \left(\square_{\square\square}(S) \right)$ *de* $2^n \times 2^\square$ *en donde sus entradas son elementos*

$$\square_{\square\square}(S) \equiv \left| \left\{ \square \in \square_n / S(\square) + S(\square + \square) = \square \right\} \right|.$$

En el **capitulo 1**, describimos al Data Encryption Standard (DES) y presentamos un programa de simulación en GAP de este sistema criptográfico. Abordaremos el problema de encontrar las mejores S-cajas que proporcionen a este sistema la mayor seguridad informática posible.

Una forma posible de resolver este problema es calculando todas las funciones regulares del espacio de funciones del DES. Sin embargo, el número de funciones regulares de este espacio es

150264313212607405157249461229631844127711223505326160201417800000000000000

por lo que es difícil calcular la robustez de todas las funciones regulares de $\square_\square$ *a* $\square_4$*, considerando realizar 1,000 cálculos por segundo, después de un año abríamos calculado la robustez de 3,153,6000,000 funciones y tardaríamos 476,485,011,455,502,933,654.3932 años en calcular la robustez de cada una de las funciones regulares de este espacio.*

Considerando la complejidad de este caso, estudiamos el espacio de funciones de $\square_4$ *en* $\square_2$ *que tiene 63, 063,000 funciones regulares y que podemos calcular en aproximadamente 17.5175 horas, calculando la robustez de 1000 funciones por segundo.*

Otra forma de resolver este problema es considerando algunos resultados sobre la robustez de las funciones regulares y la acción de algunos grupos en los espacios de funciones regulares.

En 1993, Seberry, Zhang y Zheng obtuvieron una cota para la robustez de funciones regulares, definida por el valor

$$\varepsilon' = \left(\square - \frac{\square}{2^n} \right) \left(\square - 2^{-\square+\square} \right)$$

Posteriormente en 1997, H. Tapia-Recillas, G. Vega y E. Daltabuit introdujeron otra cota para la robustez de un mapeo regular S.

$$\varepsilon^0 = \left(\square - \sqrt{2^{-\square} - 2^{-n}}\right)^2 \qquad n>m>1$$

Tal que $\varepsilon' > \varepsilon^0$.

En el **capitulo 2,** estudiamos algunos resultados sobre la tabla de distribución diferencial y la cota de la robustez de una función regular presentados por H. Tapia-Recillas, G. Vega y E. Daltabuit en 1997, probamos estos resultados de manera alternativa y presentamos algunos nuevos resultados.

En el **capitulo 3**, Consideramos la acción del grupo afín $Aff(n,2)$ sobre el espacio de funciones regulares y calculamos la robustez de las funciones regulares de $\square_4$ en $\square_2$, probamos la invarianza de la robustez de funciones regulares a la acción del grupo afín *Aff(2,2)*. Cabe mencionar, que la acción de este grupo en el espacio de funciones regulares permite simplificar la búsqueda de funciones regulares a un número reducido de estas.

Para el espacio de funciones regulares de $\square_4$ *en* $\square_2$ *solo necesitamos considerar 2, 627,625 funciones regulares, a las cuales podemos calcular su robustez en aproximadamente 43.7937 minutos, esto reduce significativamente el tiempo de computo.*

Consideramos también la acción del grupo lineal ***GL(4,2)****, este opera naturalmente en* $\square_4$ *y al aplicar una transformación lineal a una función regular nos da otra con la misma robustez. Entonces, solo es necesario considerar 11,550 funciones regulares que reduce nuestro tiempo de computo a solo 11.55 segundos.*

Al parecer, la acción del grupo afín sobre el espacio de funciones regulares es tratada por primera ocasión en esta tesis.

Finalmente, en el **capitulo 4** se presentan programas en GAP 3.0 (para linux) de Simulación del Data Encryption Standard, cálculo de la robustez de S-Cajas del Data Encryption Standard, cálculo de la robustez de funciones regulares de $\square_{\square}$ a $\square_4$, cálculo de funciones regulares y la tabla de distribución diferencial de una función regular. Los cálculos se llevaron a cabo en una computadora personal de procesador Intel Pentium 4.0 y memoria RAM de 1 Gigabyte.

Las aplicaciones en ciencias de la computación y criptografía que tienen estos resultados son interesantes pues le dan al especialista una manera de encontrar funciones que le proporcionen a su sistema de cifrado una alta seguridad, de manera que se puede conjeturar que para espacios de dimensión mayor se puedan realizar reducciones semejantes.

1.0 EL DATA ENCRYPTION STANDARD.

El Data Encryption Standard (DES) fue implementado el 15 de junio de 1977 en los Estados Unidos de América como un sistema de cifrado standard para ser usado en las agencias y departamentos federales, es un algoritmo de clave secreta para operaciones no clasificadas y su principal aplicabilidad es la seguridad informática.

Ha tenido varias implementaciones en dispositivos de seguridad bancaria y comercial, chips electrónicos VLSI, microprocesadores de *Read Only Memory (ROM)*, memorias programables *Programmable Read Only Memory (PROM)* y demás aplicaciones en seguridad y protección de software utilizando *Ramdon Access Memory (RAM).*

El Data Encryption Standard especifica un algoritmo para ser implementado en dispositivos de hardware electrónico y usado para la protección criptográfica de datos de la computadora.

1.1 SISTEMAS CRIPTOGRÁFICOS SIMÉTRICOS

Un sistema criptográfico simétrico consta de un conjunto de mensajes que son partidos en bloques como elementos de un espacio vectorial, de un conjunto de claves *K*, una función de cifrado $\square_f$ que mediante una función *f* entre espacios vectoriales encripta el mensaje *m* con una clave *k*, un conjunto de mensajes cifrados y una función $\square^{-\square}_f$ que descifra el mensaje cifrado por $\square_f$. En estos sistemas, es importante que la función $\square_f$ sea invertible para poder recuperar el mensaje enviado inicialmente.

La función de cifrado $\square_f$ es una composición de funciones no necesariamente biyectivas. Sin embargo, esta función es biyectiva. Como observaremos mas adelante, el algoritmo de cifrado permite construir funciones invertibles independientemente de cómo sea la función *f*.

Sea $\square_n$ el espacio vectorial de dimensión *n* sobre el campo de dos elementos $\square_2$. Denotaremos a los elementos de $\square_n$ con los enteros $\square$, $0 \leq \square \leq 2^{n-\square}$, el entero $\square$ representara al vector $(\square_\square, \square_2, \square\square, \square_{n-\square})$ donde $\square = \square_0 + 2\square_\square + \square\square + 2^{n-\square}\square_{n-\square} = \sum_{\square=\square}^{n} 2^{\square+\square}\square_{\square+\square}$ (aquí los $\square_\square$ se interpretan como elementos del campo $\square_2$). Por ejemplo, el vector $(\square,0,\square,0) \in \square_4$ es el número $(\square,0,\square,0) = \square + 0 \bullet 2 + \square \bullet 2^2 + 0 \bullet 2^3 = \square.$

Sea $f \square\square_n \to \square_n$ una función, podemos definir la función

$$\square_f \square\square_n \times \square_n \to \square_n \times \square_n$$
$$(\square, f) \mapsto (\square, \square + f\square)$$

Donde $(L,f)\in V_n\times V_n$ y tiene sentido que $fR=f(R)$. La función E_f cifra el mensaje M partido en L y R. La función inversa $E_f^{-1}(L,R)=(-fL+R,L)$ es la función que descifra el mensaje *m*.

Observe que $\begin{pmatrix}-f & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}0 & L\\ 0 & L+R\end{pmatrix}=\begin{pmatrix}0 & L\\ 0 & R\end{pmatrix}$ y $\begin{pmatrix}-f & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}0 & L\\ 0 & R\end{pmatrix}=\begin{pmatrix}0 & -fR+R\\ 0 & L\end{pmatrix}$.

1.2 DESCRIPCIÓN DEL DATA ENCRYPTION STANDARD

El DES consiste en un algoritmo de cifrado-descifrado de bloques de 64 bits, mediante una clave *k* de 48 bits.

Los 64 bits de entrada (texto original), formando un bloque *m*, son transformados mediante una permutación inicial (IP) del conjunto {*1,2,…,64*} escrita en forma de matriz, en un bloque. $m_0=IP(m)$

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

Este bloque m_0 es dividido en dos sub-bloques L_0 y R_0 de 32 bits cada uno, los cuales están sujetos aun conjunto de 16 transformaciones, con una función *f(,)* y 16 claves k_i *(i = 1,2…,16)* cada una de 48 bits.

A continuación mostramos las claves utilizadas en el programa de simulación DES escritas en GAP.

key:=[[1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1], [1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1], [1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0], [0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0,0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1], [1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0], [1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1], [0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1,

1, 0, 1, 0, 1, 0, 0, 0, 1], [0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0], [1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1], [0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0], [0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0], [1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0], [1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0], [0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1], [1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1]];

Al final de aplicar las claves, el bloque resultante de concatenar L_{16} y R_{16} es sometido a la permutación inversa inicial $(IP)^{-1}$.

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

Entre la permutación (IP) y la permutación $(IP)^{-1}$, el algoritmo DES ejecuta 16 iteraciones que combinan substituciones y transposiciones. Sea $\square_i$ el resultado de la i-esima iteración, $\square_i = L_i R_i$, donde L_i y R_i son respectivamente, los 32 bits mas a la izquierda y los 32 bits mas a la derecha de $\square_i$. Entonces, el algoritmo de cifrado es

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

Donde k_i es una clave de 48 bits.

El siguiente diagrama muestra esquemáticamente el procedimiento de cifrado del Data Encryption Standard.

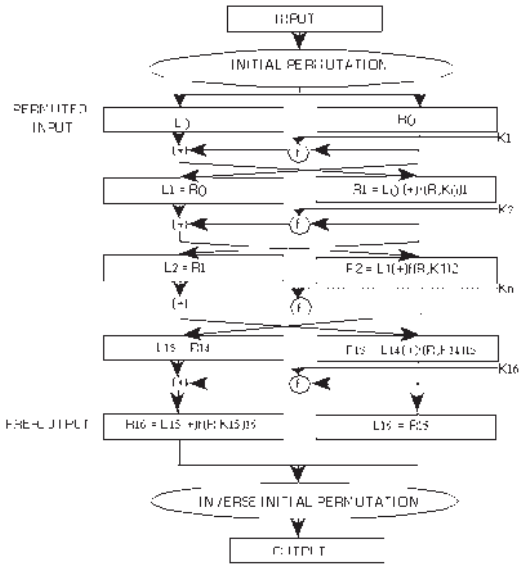

Fig. 2- Diagrama del sistema de cifrado del Data Encryption Standard.

1.3 LA FUNCIÓN DE SELECCIÓN.

Esta función transforma los 32 bits del bloque R_{i-1}, mediante una clave $k_i \in V_{48}$, en los 32 bits de $f(R_{i-1}, k_i)$. Primeramente, se aplica una función de expansión $E: V_{32} \to V_{48}$ definida por $x = (x_1, \ldots, x_{32}) \mapsto (x_{E(1)}, \ldots, x_{E(48)}) = E(x)$ que transforma un bloque de 32 bits en un vector de 48 bits.

En GAP podemos escribir esta función de la siguiente forma:

```
ext:=function ( x )
   local  a, b, i;
   a := x;
   b := [ ];
   for i  in [ 1 .. 32 ]  do
      b[i] := a[i];
   od;
   for i  in [ 33 .. 48 ]  do
      b[i] := a[i - 32];
   od;
   return b;
end;
```

Posteriormente, calculamos $E(R_{i-1}) \oplus k_i$ cuyo resultado es un vector de 48 bits que dividimos en 8 bloques $\square_i$ de 6 bits cada uno.

$$\square = E(R_{i-1}) \oplus k_i = (\square_1, \square_2, \square_3, \square_4, \square_5, \square_6, \square_7, \square_8)$$

Cada uno de estos bloques es usado como entrada de una tabla de selección-substitución $\square_i$, que da 4 bits de salida. Esta tabla de selección es llamada Caja y se compone de 8 funciones $\square_i$ llamadas S-cajas que transforman un bloque de 6 bits en un bloque de 4 bits.

La Caja se define

$$\square \square V_{48} \longrightarrow V_{32}$$
$$\square \mapsto \square(\square)$$

Donde cada S-caja esta definida

$$\square_i \square V_6 \longrightarrow V_4$$
$$\square_i \mapsto \square_i(\square_i)$$

Para *i = 1,2,…,8*. Podemos interpretar $V_{48} \cong V_6^8$ $\square\square V_{32} \cong V_4^8$ $\square\square\square$

$$\square(\square_1, \square_2, \square_3, \square_4, \square_5, \square_6, \square_7, \square_8) = (\square_1(\square_1), \square_2(\square_2), \square_3(\square_3), \square_4(\square_4), \square_5(\square_5), \square_6(\square_6), \square_7(\square_7), \square_8(\square_8))$$

Enseguida presento la Caja utilizada en el Programa de Simulación DES:

caja:=[[[1, 5, 0, 2], [2, 4, 5, 6], [2, 5, 6, 7], [3, 5, 6, 7], [12, 13, 15, 5], [3, 5, 7, 10], [2, 6, 1, 8], [2, 4, 6, 2], [2, 6, 8, 9], [12, 15, 0, 1], [1, 5, 15, 9], [8, 9, 10, 14], [3, 5, 7, 9], [13, 15, 6, 7], [13, 5, 7, 9], [11, 15, 7, 9]] ,

[[11, 0, 9, 12], [0, 4, 5, 6], [2, 15, 6, 10], [3, 5, 6, 9], [12, 13, 15, 5], [9, 15, 7, 10], [0, 6, 11, 8], [4, 4, 6, 2], [9, 6, 8, 9], [12, 15, 0, 1], [5, 0, 5, 9], [9, 9, 10, 14], [3, 0, 7, 9], [3, 5, 6, 7], [3, 5, 10, 9], [7, 15, 7, 0]] ,

[[4, 0, 9, 4], [4, 7, 5, 7], [2, 5, 6, 0], [3, 5, 6, 9], [2, 3, 15, 5], [9, 5, 7, 10], [7, 6, 1, 8], [4, 4, 7, 2], [9, 6, 4, 7], [1, 15, 0, 1], [5, 0, 7, 9], [7, 9, 7, 4], [3, 0, 7, 9], [3, 5, 6, 7], [3, 2, 7, 9], [7, 8, 7, 0]] ,

[[4, 1, 9, 4], [1, 7, 5, 7], [2, 1, 6, 10], [1, 5, 6, 9], [12, 13, 15, 1], [9, 5, 1, 10], [7, 6, 1, 8], [4, 4, 7, 11], [9, 6, 4, 7], [11, 15, 0, 11], [5, 0, 7, 11], [7, 9, 11, 4], [3, 1, 7, 9], [3, 5, 11, 7], [3, 2, 7, 11], [7, 8, 11, 0]],

[[14, 11, 9, 3], [1, 3, 15, 7], [12, 11, 3, 10], [3, 5, 3, 9], [3, 3, 5, 11], [9, 5, 1, 1], [7, 3, 1, 8], [14, 14, 13, 11], [9, 6, 4, 3], [1, 5, 0, 1], [13, 10, 14, 11], [7, 9, 1, 3], [3, 3, 7, 9], [3, 14, 11, 7], [3, 2, 3, 1], [7, 8, 1, 3]] ,

[[15, 6, 9, 0], [1, 6, 5, 7], [12, 1, 3, 0], [3, 5, 3, 0], [3, 3, 15, 6], [9, 5, 0, 6], [7, 3, 1, 8], [4, 4, 3, 6], [9, 6, 10, 6], [1, 5, 10, 1], [13, 0, 14, 1], [7, 9, 10, 6], [13, 13, 7, 9], [13, 14, 6, 7], [13, 12, 10, 6], [7, 8, 11, 6]] ,

[[5, 7, 9, 7], [1, 7, 5, 7], [12, 1, 3, 0], [3, 5, 3, 0], [13, 13, 5, 6], [9, 7, 0, 6], [7, 7, 1, 8], [14, 14, 13, 7], [9, 7, 0, 6], [11, 7, 0, 1], [13, 10, 14, 7], [7, 7, 0, 6], [3, 7, 7, 9], [3, 4, 7, 7], [13, 12, 10, 7], [7, 7, 1, 7]] ,

[[15, 9, 9, 9], [1, 7, 15, 7], [12, 1, 13, 0], [3, 9, 13, 0], [13, 13, 15, 9], [9, 7, 0, 6], [7, 7, 1, 8], [14, 14, 3, 9], [9, 12, 0, 6], [11, 7, 0, 1], [13, 0, 4, 9], [7, 13, 10, 6], [13, 7, 7, 9], [3, 4, 9, 7], [13, 2, 0, 9], [7, 12, 11, 7]]];

Estos bloques de salida, concatenados forman un nuevo bloque C de 32 bits.

$$\square = \square(\square) = \square(E(R_{i-1}) \oplus k_i) = (\square_1(\square_1), \square_2(\square_2), \square_3(\square_3), \square_4(\square_4), \square_5(\square_5), \square_6(\square_6), \square_7(\square_7), \square_8(\square_8))$$

La función **Box( )** escrita en GAP asocia a cada bloque de B una S-Caja.

```
Box:=function(x)
local a,b,c,i;
a:=x;b:=part(a);c:=[];
for i in [1..8] do
c[i]:=box(b[i],i);
od;
return c;
end;
```

Finalmente, la salida de la función $f(,)$ es el resultado de aplicar un conjunto $\square$ de 16 permutaciones de los elementos $\{1,2,\ldots,32\}$ al bloque C de 32 bits.

$$P(\square) = P(\square(\square)) = P(\square(E(R_{i-1}) \oplus k_i))$$

La siguiente matriz esta compuesta por 16 permutaciones de los elementos $\{1,2,\ldots,32\}$.

permutacion:=[[15, 17, 19, 9, 25, 16, 7, 23, 26, 12, 5, 13, 28, 20, 14, 21, 1, 27, 3, 32, 6, 2, 22, 29, 18, 31, 24, 11, 4, 10, 8, 30], [32, 6, 5, 25, 12, 28, 9, 31, 21, 18, 15, 8, 27, 26, 2, 1, 19, 23, 17, 16, 10, 30, 7,24,13, 22, 11, 14, 29, 3, 20, 4], [3, 26, 28, 1, 14, 11, 21, 4, 12, 29, 19, 27, 15, 16, 2, 17, 18, 13, 6, 24, 8, 9,20,25, 10, 32, 23, 7, 22, 30, 31, 5], [29, 10, 26, 22, 8, 16, 2, 3, 7, 1, 24, 4, 9, 20, 12, 11, 32, 17, 6, 15, 19, 23, 14, 30, 27, 5, 28, 13, 25, 18, 21, 31], [2, 19, 18, 14, 20, 22, 23, 17, 11, 15, 28, 12, 24, 29, 31, 3, 21, 26, 5, 9, 6, 27, 30, 7, 8, 10, 4, 25, 1, 32, 16, 13], [13, 7, 27, 19, 6, 32, 2, 3, 31, 28, 15, 24, 21, 30, 20, 26, 12, 5, 11, 10, 8, 23, 1, 25, 29, 18, 9, 4, 22, 16, 17, 14], [16, 22, 29, 12, 14, 30, 24, 18, 6, 11, 25, 15, 31, 20, 7, 3, 9, 1, 32, 5, 17, 10, 19, 27, 23, 13, 8, 21, 4, 28, 26, 2], [4, 28, 18, 7, 31, 11, 9, 22, 17, 13, 8, 15, 6, 14, 10, 29, 20, 25, 21, 12, 1, 24, 5, 2, 30, 27, 32, 23, 26, 3, 19, 16], [30, 5, 25, 11, 28, 20, 14, 6, 16, 22, 24, 27, 9, 1, 4, 23, 10, 7, 18, 3, 29, 31, 13, 32, 21, 17, 26, 8, 2, 19, 12, 15], [28, 30, 4, 2, 22, 10, 32, 8, 31, 21, 26, 5, 13, 6, 15, 24, 16, 19, 18, 3, 11, 23, 29, 17, 1, 25, 12, 27, 9, 20, 7, 14], [21, 14, 23, 15, 12, 9, 5, 2, 24, 6, 26, 31, 4, 3, 32, 16, 25, 22, 27, 7, 10, 28, 29, 17, 18, 13, 20, 1, 19, 8, 11, 30], [12, 2, 27, 10, 11, 17, 25, 30, 13, 8, 28, 15, 31, 23, 16, 19, 6, 1, 22, 18, 14, 20, 3, 26, 29, 24, 5, 21, 9, 7, 32, 4], [18, 32, 24, 16, 31, 23, 27, 8, 30, 21, 25, 17, 6, 9, 15, 5, 10, 13, 12, 11, 28, 1, 26, 22, 14, 20, 2, 29, 4, 7, 19, 3], [9, 20, 7, 21, 2, 27, 6, 19, 14, 8, 31, 29, 13, 4, 22, 24, 12, 3, 26, 16, 5, 15, 11, 10, 18, 23, 30, 25, 32, 28, 1, 17], [5, 1, 22, 6, 24, 26, 28, 2, 29, 17, 9, 12, 8, 7, 10, 27, 4, 16, 20, 3, 15, 19, 13, 31, 25, 14, 18, 11, 30, 32, 21, 23], [3, 17, 2, 24, 11, 25, 28, 6, 8, 7, 32, 9, 20, 23, 13, 4, 1, 31, 27, 30, 12, 16, 29, 18, 5, 10, 15, 21, 14, 22, 26, 19]];

La función **Per(,)** aplica a cada una de las S-cajas una permutación de la matriz anterior**.**

```
Per:=function(x,y)
local a,b,c,i;
a:=x;b:=permutacion[y];
c:=[];
for i in [1..32] do
c[i]:=a[b[i]];
od;
return c;
end;
```

Para cada iteración con clave k_i hay una permutación correspondiente. La función se selección $f(\ ,\ )$ esta definida como una composición de funciones

$$V_{32} \xrightarrow{E\square\square} V_{48} \xrightarrow{k_i} V_{48} \xrightarrow{\square\square\square} V_{32} \xrightarrow{P} V_{32}$$
$$f(R_{i-1}, k_i) = P(\square) = P(\square(\square)) = (P_i(\square(E(R_{i-1}) \oplus k_i)))$$

Para $i = 1,2,\ldots,16$. El siguiente diagrama muestra esquemáticamente una descripción de la función $f(\ ,\ )$.

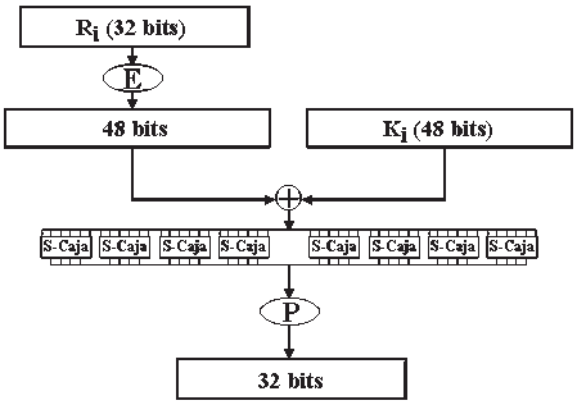

Fig. 3-Diagrama de la función de selección.

En GAP podemos programar a nuestra función de selección de la forma siguiente:

```
F:=function(x,i)
local a,b,k;
a:=x;k:=i;
b:=Per(glue(Box(ad(ext(a),key[k]))),k);
return b;
end;
```

Observe que la construcción del programa de simulación del Data Encryption Standard es muy similar a su forma matemática, la función **ext(R)** es la aplicación de la función de expansión a un vector de 32 bits **R** , mediante **ad(ext(R), key[k])** sumamos el vector extendido **ext(R)** a una clave **key[k]**de 48 bits, la función **(Box(ad(ext(R), key[k])))** consiste en aplicar a este vector resultante de 48 bits cada una de las S-cajas que lo transformaran en un vector de 32 bits. Posteriormente, aplicamos **glue(Box(ad(ext(R), key[k])))** que consiste en concatenar el vector C que fue partido en 8 bloques de 4 bits por la acción de la función **Box()**, finalmente aplicamos la permutación final **Per(glue(Box(ad(ext(R), key[k]))),k);** y obtenemos la función de selección de una de las 16 iteraciones.

El procedimiento anterior se realiza en 16 ocasiones, así obtenemos el mensaje cifrado mediante la función **encryp(m)**, que en GAP escribimos:

```
encrypt:=function(x)
local a,l,r,i,ia,da,fa;
a:=x;
l:=[];r:=[];
l[1]:=int(a,1,32);
r[1]:=int(a,33,64);
for i in [2..17] do
l[i]:=r[i-1];
r[i]:=ad(l[i-1],F(l[i],i-1));
od;
ia:=l[17];
da:=r[17];

fa:=Concatenation(ia,da);
return fa;
end;
```

Para descifrar el mensaje, utilizamos las claves en orden inverso; Es decir, k_{16} es utilizada en la primera iteración, k_{15} en la segunda y así sucesivamente hasta llegar a la ultima iteración donde utilizamos a k_1. Esto se debe a que la permutación inicial y final son inversa una de la otra. En este caso,

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

En GAP, utilizamos la función **desencrypt( )** para descifrar nuestro mensaje cifrado. Esto es, **desencrypt(encrypt(m))=m.**

```
desencrypt:=function(x)
local a,l,r,i,ia,da,fa;
a:=x;
```

```
l:=[];r:=[];
l[1]:=int(a,1,32);
r[1]:=int(a,33,64);
for i in [2..17] do
r[i]:=l[i-1];
l[i]:=ad(r[i-1],F(r[i],18-i));
od;
ia:=l[17];da:=r[17];

fa:=Concatenation(ia,da);
return fa;
end;
part:=function(x)
local a,b,c,i,s;
a:=x;b:=[];c:=0;s:=[1,7,13,19,25,31,37,43];
for i in s do
c:=c+1;b[c]:=int(a,i,i+5);
od;
return b;
end;
```

2.0 LA ROBUSTEZ DE MAPEOS REGULARES.

Las S-cajas son unos de los componentes principales de los sistemas de cifrado simétrico tipo DES. En estos sistemas la seguridad depende de un conjunto de claves *K* y de la estructura de estas S-cajas.

El problema de seguridad de los sistemas criptográficos simétricos consiste en encontrar S-cajas que den al sistema la mayor seguridad informática.

Este problema ha sido estudiado con anterioridad por muchos matemáticos y especialistas de la computación desde 1977, principalmente los trabajos de *Jennifer Seberry, X. Zhang y Young Zheng* han contribuido en la búsqueda de funciones S-cajas que proporcionan sistemas criptográficos con la mayor seguridad posible. *A. Shamir* y *E. Biham* han presentado investigaciones sobre el criptoanálisis diferencial de criptosistemas tipo DES, *Webster* y *Tavares* han contribuido al mejoramiento en el diseño de S-cajas, han construido S-cajas cada vez más fuertes a criptoanálisis, aun que la teoría de las funciones regulares se debe principalmente a *Jennifer Seberry.*

En este trabajo presentaré nuevas características de estas funciones y encontraremos las S-cajas que dan la mayor seguridad en este sistema criptográfico para espacios pequeños, de esta manera es posible mejorar notablemente la estructura de las S-cajas y construir sistemas criptográficos más seguros. Esto es, resolver el problema de la seguridad de sistemas criptográficos simétricos tipo DES para espacios de funciones regulares pequeños.

2.1 NOTACIÓN Y DEFINICIONES BASICAS.

Sea V_n el espacio vectorial de dimensión n sobre el campo de dos elementos $\mathbb{Z}_2$. Denotaremos a los elementos de V_n con los enteros x, $0 \le x \le 2^{n-1}$, el entero x representará al vector $(x_1, x_2, \ldots, x_{n-1})$ donde $x = x_0 + 2x_1 + \cdots + 2^{n-1}x_{n-1} = \sum_{i=1}^{n} 2^{i-1}x_{i-1}$ (aquí los x_i se interpretan como elementos del campo $\mathbb{Z}_2$). Por ejemplo, el vector $(1,0,1,0) \in V_4$ es el número $(1,0,1,0) = 1 + 0 \bullet 2 + 1 \bullet 2^2 + 0 \bullet 2^3 = 5$.

2.1.1 Tabla de Distribución Diferencial.

Dada una función $f: V_n \to V_m$ definimos la función $f_a: V_n \to V_m$ como $f_a(i) = f(x) + f(x+i)$ y formamos la matriz $M = \left(f_a(i)\right)$ de $2^n \times 2^m$, donde a indica el renglón y i la columna.

Definición 1. *Si S es una función de V_n en V_m, su matriz de distribución es la matriz $M^{(f)} = \left(f_i(a)\right)$ de $2^n \times 2^m$ en donde $f_i(a) = \left|\{x \in V_n / f(x) + f(x+i) = a\}\right|$. Es decir, es el número de entradas en la columna i de la matriz M con valores iguales a j.*

Consideremos algunas propiedades de las funciones $f: V_n \to V_m$:

2.1.1.1) En la tabla de distribución $M^{(f)}$ el elemento $f_{0,0}(x) = 2^n$ y los elementos $f_{0,a}(x) = 0$ para toda j diferente de cero.
2.1.1.2) Dada una función $f: V_n \to V_m$, las entradas de la matriz $M^{(f)} = \left(f_i(a)\right)$ son números enteros pares.
Prueba. En efecto, si en la columna i aparece $f_a(i) = f(x) + f(x+i) = a$ también aparece $f_{x+i}(i) = f(x+i) + f(x+i+i) = f(x+i) + f(x) = a$. Es decir, cada número aparece dos veces.
2.1.1.3) La suma de las entradas de un renglón es 2^n.
Prueba. Considere la función $f_i: V_n \to \text{mn}(i)$ dada por $f_i(x) = f(x+i) + f(x)$, entonces, por definición $f_i(a) = \left|f_i^{-1}(a)\right|$. Por tanto, $\sum_{a=0}^{2^n-1} f_i(a) = \sum_{a=0}^{2^n-1} \left|f_i^{-1}(a)\right| = \left|V_n\right| = 2^n$.
2.1.1.4) La función $f_a: V_n \to V_m$ definida por $f_a(i) = f(x) + f(x+i)$ se puede interpretar como la composición de las traslaciones $\tau_1: V_n \to V_n$, $\tau_2: V_m \to V_m$ y la función $f: V_n \to V_m$.

$$\begin{array}{ccccccc} V_n & \xrightarrow{\tau_1} & V_n & \xrightarrow{f} & V_m & \xrightarrow{\tau_2} & V_m \\ i & \mapsto & x+i & \mapsto & f(x+i) & \mapsto & f(x+i)+f(x) \end{array}$$

2.1.2 Funciones Regulares

Una función $\square\square V_n \to V_m$ *se llama regular si* $\left|\square^{-1}(\square)\right| = 2^{n-m} \quad \forall \square \in V_n$. *Es decir, en la sucesión de valores de la función cada elemento j de* V_m *aparece* 2^{n-m} *veces.*

Consideremos la siguiente observación:

2.1.2.1) Si S es una función regular $\square\square V_n \to V_m$ *y* $f\ \square V_n \to V_n$; $\square\square V_m \to V_m$ *son funciones biyectivas, entonces la composición* $V_n \xrightarrow{f} V_n \xrightarrow{\square} V_m \xrightarrow{\square} V_m$ *es también regular. En particular, las funciones* $\square_{\square}$ *se pueden interpretar como la composición de las traslaciones* T_1, T_2 *y la función regular S. De lo anterior, si S es regular el renglón* $\times$ *de la matriz M es la sucesión de valores de la función* $\square_{\square}$, *cada elemento de* V_m *aparece* 2^{n-m} *veces. Por otro lado, la matriz M tiene* 2^n *renglones donde cada elemento j de* V_m *aparece* $2^n\left(2^{n-m}\right)$ *veces.*

Proposición 2. *La suma de las entradas* $\square_{i,0} + \square_{i,1} + \square\square + \square_{i,2^i-1}$ *de la columna i de la matriz* $\square^{(\square)}$ *es igual a* $2^n\left(2^{n-m}\right)$, *si y solamente si* $\square\square V_n \to V_m$ *es una función regular.*

Prueba. Como $\square_{i\square}(\square)$ es el número de veces que aparece j en la columna i, $\sum_{\square=0}^{2^n-1} \square_{i\square}(\square)$ es el número total de j que aparece en $\square^{(\square)}$ y por la observación (2.1.2.1) $\sum_{\square=0}^{2^n-1} \square_{i\square}(\square) = 2^n(2^{n-m})$. Recíprocamente, si suponemos $\sum_{\square=0}^{2^n-1} \square_{i\square}(\square) = 2^n(2^{n-m})$ cada elemento *j* de V_m aparece 2^{n-m} veces y por tanto *S* es una función regular.

Proposición 3. *Sea* $\square\square V_n \to V_m$ *una función regular, el valor* $\square_{1,0}(\square) \neq 2^n - 2$ *en la tabla de distribución diferencial de S.*

Prueba. Supongamos que $\square_{1,0}(\square) = 2^n - 2$. Sea $\square = \left\{\square \in V_n \middle| \square(\square+1) + \square(\square) = 0\right\}$, por definición $\square_{1,0}(\square) = \left|\square\right| = 2^n - 2$. Consideremos el conjunto de las imágenes inversas $\square = \square^{-1}(0) \cup \square^{-1}(1) \cup \square\square \cup \square^{-1}(2^m - 2)$, entonces $\left|\square\right| = 2^n - 2^{n-m} = 2^{n-m}(2^m - 1)$, y se cumple que $\left|\square\right| + \left|\square\right| - \left|\square \cap \square\right| \leq 2^n$, de lo anterior $2^n - 2 + 2^{n-m}(2^m - 1) - \left|\square \cap \square\right| \leq 2^n$ y $2^{n-m}(2^m - 1) - 2 \leq \left|\square \cap \square\right| \leq 2^{n-m}(2^m - 1)$. Consideremos las siguientes afirmaciones:

(1) Si $\square \in \square$ *entonces* $(\square+1) \in \square$. *(2)* $\left|\square \cap \square^{-1}(i)\right|$ *es un número par,* $i = 0, \square\square, 2^m - 1$.

Si $\square \in \square \cap \square^{-1}(0)$ entonces $(\square+1) \in \square$, como $\square(\square) = \square(\square+1)$ y $\square(\square) = 0$, entonces $\square(\square+1) = 0$. De lo anterior, el número de elementos de $\square \cap \square$ es $2^{n-m}(2^m - 1)$ o $2^{n-m}(2^m - 1) - 2$. Supongamos que $\left|\square \cap \square\right| = 2^{n-m}(2^m - 1)$, por tanto $\square \subseteq \square$. Sea $\square = \square \cup \{\square, \square\}$, esto significa que $\{\square, \square\} \in \square^{-1}(2^m - 1)$. Solo falta probar que

$\square(\square+1) = 2^m - 1$, supongamos lo contrario; Esto es, $\square(\square+1) \neq 2^m - 1$, entonces $(\square+1) \in \square^{-1}(k)$, $0 \leq k < 2^m - 1$ y $\square^{-1}(k) \subset \square$. Entonces, $k = \square(\square+1) = \square(\square) = 2^m - 1$ lo que contradice la hipótesis inicial.

Si $|\square \cap \square| = 2^{n-m}(2^m - 1) - 2$, se cumple que $|\square \cap \square^{-1}(k)| = 2^{n-m} - 2$ para $0 \leq k < 2^m - 1$, podemos formar el conjunto $\square' = \square^{-1}(0) \cup \square^{-1}(1) \cup \square\square\square \cup \square^{-1}(2^m - 1)$ y $|\square \cap \square'| = 2^{n-m}(2^m - 1)$ que es el caso anterior.

2.2 UNA COTA PARA LA ROBUSTEZ DE MAPEOS REGULARES.

La robustez de un mapeo regular S es un número entre 0 y 1 que se interpreta como tener mayor seguridad en el sistema criptográfico si este número es aproximado a 1 y una menor seguridad si este se aproxima a 0.

El problema de calcular cotas para la robustez de un mapeo regular es importante, pues permite saber cual es el alcance de seguridad de una función regular en un sistema criptográfico.

En 1993, Seberry, Zhang y Zheng obtuvieron una cota para la robustez de n×m cajas de mapeos regulares, definida por el valor

$$\varepsilon' = \left(1 - \frac{1}{2^n}\right)\left(1 - 2^{-m+1}\right)$$

Posteriormente en 1997, H. Tapia-Recillas, G. Vega y E. Daltabuit introdujeron otra cota para la robustez de un mapeo regular S.

$$\varepsilon^0 = \left(1 - \sqrt{2^{-m} - 2^{-n}}\right)^2 \qquad n>m>1$$

Tal que $\varepsilon' > \varepsilon^0$.

A continuación definiremos la robustez *rob(S)* de un mapeo regular de manera formal.

Definición 4. *Sea* $\square\square V_n \to V_m$, *L el mayor valor de la tabla de distribución diferencial de S (omitiendo el primer renglón); sea* R *el número de entradas distintas de 0 en la primera columna de la tabla (omitiendo el primer renglón). La robustez de S se define como*

$$\square\square\square(\square) = \left(1 - \frac{R}{2^n}\right)\left(1 - \frac{L}{2^n}\right). \qquad 0 \leq \square\square\square(\square) < 1.$$

Consideremos las siguientes afirmaciones:

2.2.1) $\left(1-\frac{L}{2^n}\right)\left(1-\frac{2^{n-m}}{L}\right)\geq\left(1-\frac{R}{2^n}\right)\left(1-\frac{L}{2^n}\right)$.

Prueba. Ya que $2^n(2^{n-m}-1)\leq RL$ para $\square_i \leq R, i=0,\square,L$ se tiene que $2^n(2^{n-m})-2^n=\square_0+\square_1+\square\square+\square_L\leq RL$. Entonces $\frac{2^{n-m}-1}{R}=\frac{L}{2^n}$. Por tanto,

$$\left(1-\frac{L}{2^n}\right)\left(1-\frac{2^{n-m}}{L}\right)\geq\left(1-\frac{R}{2^n}\right)\left(1-\frac{L}{2^n}\right).$$

Por otro lado, el valor máximo de L *es el entero positivo par más próximo y menor que* $L_0=\sqrt{2^n(2^{n-m}-1)}$. *Al considerar este valor* L_0 *en* $\square\square(\square)=\left(1-\frac{R}{2^n}\right)\left(1-\frac{L}{2^n}\right)$ *se tiene que* $\varepsilon^0=\left(1-\sqrt{2^{-m}-2^{-n}}\right)^2$ *es una cota para la robustez de* S.

3.0 LA ACCIÓN DEL GRUPO AFÍN *Aff(n,2)* EN EL ESPACIO DE FUNCIONES REGULARES.

En este capitulo estudiaremos la acción del grupo afín $\square ff(n,2)$ sobre el espacio de funciones regulares y calculamos la robustez de las funciones regulares de V_4 en V_2, probamos la invarianza de la robustez de funciones regulares a la acción del grupo afín *Aff(2,2)*. Cabe mencionar, que la acción de este grupo en el espacio de funciones regulares permite simplificar la búsqueda de funciones regulares a un número reducido de estas.

Al parecer, la acción del grupo afín en el espacio de funciones regulares es considerada por primera ocasión en esta tesis.

3.1 GRUPOS QUE OPERAN EN FUNCIONES REGULARES.

Para definir las S-cajas en el Data Encryption Standard no se consideran todas las funciones posibles (ver [2], [4], [5], [6], [10], [12]), se toman en cuenta solamente las llamadas funciones regulares.

El número de funciones regulares de V_n en V_m es

$$\frac{2^n\square}{\left(2^{n-m}\right)\square^{2^m}}$$

Esto es consecuencia de que el grupo simétrico $□_{2^n}$ opera transitivamente en las funciones regulares y el subgrupo que deja fija una función regular es $□_{2^{n-m}} \times □□ \times □_{2^{n-m}}$ de 2^m factores.

Consideremos el conjunto $□_{n,m}$ de todas las funciones regulares de V_n en V_m. Los grupos simétricos $□_{2^n}$, $□_{2^m}$ operan en $□_{n,m}$ de la siguiente manera:

$$□_{2^m} \times □_{n,m} \times □_{2^n} \to □_{n,m}$$
$$(\sigma, □, \tau) \mapsto (\tau \circ □ \circ \sigma)(□)$$

$\forall □ \in V_n$.

Note que si S es regular, entonces $\tau \circ □ \circ \sigma$ *es también regular.*

3.1.1 El Grupo Afín $□ff(n,2)$.

El grupo afín $□ff(n,2)$ se define como el producto semidirecto $□ff(n,2) = □L(n,2) \succ V_n$ con respecto a la operación natural de $□L(n,2)$ en V_n □El grupo $□ff(n,2)$ □□□□□□□□□□ □□□□□□□□□□ □□□□□□$□_{2^n}$.

Las transformaciones afines están generadas de transformaciones lineales y traslaciones, en ambos casos; La aplicación de estas transformaciones a una función regular deja invariante la robustez de la función.

3.1.1.1 Traslaciones

Consideremos las siguientes proposiciones:

Lema 4. *Sea* $□ □V_m \to V_m$ *una traslación* □□□$_{i□}(□)$ □□$_{i,((□\circ□)(□))}(□\circ□)$ *elementos de las tablas de distribución diferencial* $□^{(□)}$, $□^{(□\circ□)}$ *respectivamente. Entonces* $□_{i□}(□) = □_{i,((□\circ□)(□))}(□\circ□)$.

Prueba. En efecto,

$$□_{i,(□\circ□)(□)}(□\circ□) = □_{i,□(□(□))}(□\circ□)$$
$$= \left|\left\{□\in V_n \,\middle|\, (□\circ□)(□)+(□\circ□)(□+i)\right\}\right| = \left|\left\{□\in V_n \,\middle|\, □(□(□))+□(□(□+i))\right\}\right|$$
$$= \left|\left\{□\in V_n \,\middle|\, (□(□)+□)+(□(□+i)+□)\right\}\right| = \left|\left\{□\in V_n \,\middle|\, □(□)+□(□+i)\right\}\right| = □_{i□}(□)$$

Lema 5. *Sean* $□ □V_n \to V_n$ *traslación,* $□ □V_n \to V_m$ *función regular,* $□ = \left\{□\in V_n \,\middle|\, □(□)+□(□+i) = □\right\}$ *y* $□' = \left\{□\in V_n \,\middle|\, (□\circ□)(□)+(□\circ□)(□+i) = □\right\}$, *se cumple que* $□\in □ \Leftrightarrow □(□)\in □'$ □

Prueba. Supongamos que $\square\in\square$, es decir; x cumple que $\square(\square)+\square(\square+i)=(\square\circ\square)(\square(\square))+(\square\circ\square)(\square(\square+i))=\square$ por tanto $(\square(\square))\in\square'$. Ahora, supongamos que $\square(\square)\in\square'$, entonces x satisface que

$$(\square\circ\square)(\square)+(\square\circ\square)(\square+i)=\square(\square(\square))+\square(\square(\square+i))=\square.$$

Por tanto, $\square\in\square$. Por consiguiente $\square'=\square(\square)$ y podemos decir que $|\square|=|\square'|$.

En consecuencia de los lemas anteriores, se presenta la siguiente proposición:

Proposición 6. *Sean* $\square_{i,((\square\circ\square)(\square))}(\square\circ\square)$, $\square_{i\square}(\square)$, $\square_{i,((\square\circ\square)(\square))}(\square\circ\square)$ *elementos de la tabla de distribución diferencial* $\square^{(\square\circ\square)}$, $\square^{(\square)}$, $\square^{(\square\circ\square)}$ *respectivamente. Entonces,*

$$\square_{i,((\square\circ\square)(\square))}(\square\circ\square)=\square_{i\square}(\square)=\square_{i,((\square\circ\square)(\square))}(\square\circ\square)$$

Además, por consiguiente

$$\square\square\square(\square\circ\square)=\square\square\square(\square)=\square\square\square(\square\circ\square)$$

3.1.1.2 Transformaciones Lineales.

Consideremos los siguientes resultados:

Lema 7. *Sea* $\square_{i\square}(\square)$ *un elemento de* $\square^{(\square)}$, $\square\square V_m\to V_m$ *una transformación lineal invertible,* $\square=\{\square\in V_n\,|\,\square(\square(\square))+\square(\square(\square+i))=\square(\square)\}$ *y* $\square'=\{\square\in V_n\,|\,\square(\square)+\square(\square+i)=\square\}$. *Se satisface que* $\square\in\square'\Leftrightarrow\square\in\square$.

Prueba. Si $\square\in\square'$ entonces $\square(\square)+\square(\square+i)=\square$, si aplicamos la transformación lineal a cada elemento de $\square'$, tenemos que $\square(\square(\square(\square))+\square(\square(\square+i)))=\square(\square)$ y por lo tanto $\square\in\square$. Si $\square\in\square$ entonces $\square(\square(\square(\square))+\square(\square(\square+i)))=\square(\square)$, aplicamos la inversa de la transformación lineal $\square^{-1}\square(\square(\square(\square))+\square(\square(\square+i)))=\square^{-1}\square(\square)$ y por consiguiente $\square(\square)+\square(\square+i)=\square$ y $\square\in\square'$.

Lema 8. *Sea* $\square_{i\square}(\square)$ *un elemento de* $\square^{(\square)}$, $\square\square V_n\to V_n$ *una transformación lineal invertible,* $\square=\{\square\in V_n\,|\,\square(\square(\square))+\square(\square(\square+i))=\square\}$ *y* $\square'=\{\square\in V_n\,|\,\square(\square)+\square(\square+i)=\square\}$. *Se tiene que* $\square\in\square'\Leftrightarrow\square^{-1}(\square)\in\square$.

Prueba. Observemos que si $\square\in\square'$ entonces x satisface que $\square(\square)+\square(\square+i)=\square$, por consiguiente x cumple que $\square(\square(\square^{-1}(\square)))+\square(\square(\square^{-1}(\square+i)))=\square$ que es igual ha $\square(\square)+\square(\square+\square^{-1}(i))=\square$, lo cual significa que la transformación lineal cambia el orden de los renglones de la tabla de distribución diferencial, esto se debe a que $\square(\square(\square^{-1}(\square+i)))=\square(\square(\square^{-1}(\square)+\square^{-1}(i)))$ y entonces $\square^{-1}(\square)\in\square$. Supongamos que $\square^{-1}(\square)\in\square$ entonces $\square^{-1}(\square)$ satisface que $\square(\square(\square^{-1}(\square)))+\square(\square(\square^{-1}(\square+i)))=\square(\square)+\square(\square+i)=\square\square$Por tanto $\square\in\square'$.

Observe que la tabla de distribución diferencial conserva invariante al cero al aplicar transformaciones lineales. El elemento $d_{i0}=|\{x\in V_n\,|\,s(x)+s(x+i)=0\}|$ *y el elemento*

$d_{i,\square(0)}(\square\circ\square)=\left|\left\{x\in V_\square \,\middle|\, \square(\square(x))+\square(\square(x+i))=\square(0)=0\right\}\right|$ *son iguales.*

En consecuencia de los lemas anteriores, se presenta la siguiente proposición:

Proposición 9. *Sean* $\square: V_\square \to V_\square, \square': V_\square \to V_\square$ *dos transformaciones lineales invertibles,* $\square: V_\square \to V_\square$ *función regular. Entonces,* $\square\square\square(\square'\circ\square\circ\square)=\square\square\square(\square)=\square\square\square(\square\circ\square\circ\square')$.

Cabe mencionar, que si $\square: V_\square \to V_\square$, $\square': V_\square \to V_\square$ *son transformaciones lineales invertibles,* $\square: V_\square \to V_\square$ *función regular, entonces la composición* $V_\square \xrightarrow{\square} V_\square \xrightarrow{\square} V_\square \xrightarrow{\square'} V_\square$ *es una función regular de igual robustez que S.*

3.2 CÁLCULO DE ROBUSTEZ DE FUNCIONES REGULARES

Los resultados acerca de los grupos afines y la robustez de mapeos regulares nos permiten simplificar el problema de calcular la robustez de todas las funciones regulares en espacios vectoriales pequeños.

El grupo $\square_4$ opera en el espacio de funciones regulares $\square_{\square,2}$ por composición:

$$\square_4 \times \square_{\square,2} \to \square_{\square,2}$$
$$(\sigma, \square) \mapsto \sigma\circ\square$$

Note que todas las orbitas de $\square_{\square,2}$ *respecto del grupo* $\square_4$ *tienen 24 elementos, esto es debido a que si* $\sigma\circ\square=\square$ *entonces* $\sigma = i_d$. *Por otra parte,* $\square_4$ *es isomorfo al grupo Aff(2,2) y por los resultados 2.3.1,2.3.1.1 y 2.3.1.2 los elementos de Aff(2,2) no cambian la robustez de una función regular. Por tanto, para el espacio de funciones regulares* $\square_{3,2}$ *solo es necesario considerar* $\frac{2{,}520}{4!}=105$ *funciones regulares y para el espacio de funciones regulares* $\square_{4,2}$ *solo necesitamos* $\frac{16!}{(24)^5}=2{,}627{,}625$.

Caso I: *Consideremos el espacio de todas las funciones de* V_3 *a* V_2, *el número de funciones regulares es* $\frac{2^3!}{(2)!^4}=2{,}520$. *Sin embargo, solo es necesario considerar 105 funciones. En este caso, las funciones solo tienen dos posibles robustez 0 y ¼. Hay 1,344 funciones con robustez ¼ y 1,176 funciones con robustez cero. La máxima robustez obtenida al considerar todas las funciones regulares fue ¼ y la cota teórica correspondiente es 0.416. Podemos observar que la cota es significativamente mayor que el resultado experimental.*

Caso II. *Para el espacio de todas las funciones regulares de* V_4 *a* V_2**,** *el número de funciones regulares es* $\frac{2^4!}{(2^{4-2})!^4} = \frac{16!}{24^4} = 63{,}063{,}000$. *Para reducir el número de funciones regulares observamos que los elementos del subgrupo Aff(2,2) de* $\square_{16}$ *que permutan los ciclos* $(1,2,3,4),(5,6,7,\square),(9,10,11,12),(13,14,15,16)$ *al aplicarlos a una función regular nos dan otra que tiene la misma robustez. Este subgrupo Aff(2,2) tiene orden 24 y por consiguiente, solo necesitamos considerar* $\frac{16!}{(24)^5} = 2{,}627{,}625$ *funciones regulares. Considerando calcular la robustez de 1000 funciones regulares por segundo, podemos calcular su robustez en aproximadamente 17.5175 horas, esto reduce significativamente el tiempo de computo.*

Consideramos también la acción del grupo lineal GL(4,2), este opera naturalmente en V_4 *y como en el caso anterior, al aplicar una transformación lineal a una función regular nos da otra con la misma robustez. (Ver 2.3.1.2). Obtuvimos que la máxima robustez es 7/32, número significativamente menor que la cota teórica 0.3.*

Observemos que las particiones se pueden clasificar considerando la acción del grupo lineal *GL(4,2)* sobre los subconjuntos de 4 elementos {*0,a ,b, c*}.

(0,1,2,3)	(4,5,6,7)	(□,9,10,11)	(12,13,14,15)
↓	↓	↓	↓
0	1	2	3

Tomemos en cuenta el subconjunto que tiene al 0. Diremos que dos particiones son del mismo tipo si tienen al mismo subconjunto. Utilizando esta propiedad podemos ver que hay 455 clases, sin embargo los subconjuntos que contiene al cero solo son de dos tipos:

Tipo 1. {*a, ,b c*} *son linealmente independientes.*
Tipo 2. {*a, b*} *son linealmente independientes y* $a + b = c$.

Entonces, cualquiera de las 455 clases bajo el grupo lineal se transforma en 2. Solo es necesario considerar $2 \times 5{,}775 = 11{,}550$ *funciones regulares, que reduce nuestro tiempo de computo a solo 11.55 segundos.*

Las observaciones anteriores nos permiten reducir casos de espacios de funciones mayores a cálculos que se puedan efectuar en la computadora en un tiempo razonable.

En el caso de las S-cajas del Data Encryption Standard (en el espacio $\square_{6,2}$**)**, el número de funciones regulares es

64!/(24)^16 = 150264313212607405157249461229631844127711223505326160201417800000000000000

Es muy difícil calcular la robustez de todas las funciones regulares de V_6 a V_4, considerando realizar 1,000 cálculos por segundo, después de un año abríamos calculado la robustez de 3,153,6000,000 funciones y tardaríamos 476,485,011,455,502,933,654.3932 años en calcular la robustez de cada una de las funciones regulares de este espacio.

Mediante el programa **encrypdes** calculamos la robustez de cada una de S-cajas utilizadas en el DES.

ROB(B1)	**81/256**	**0.31**
ROB(B2)	**93/256**	**0.36**
ROB(B3)	**675/2048**	**0.32**
ROB(B4)	**15/32**	**0.46**
ROB(B5)	**825/2048**	**0.40**
ROB(B6)	**93/256**	**0.36**
ROB(B7)	**87/256**	**0.33**
ROB(B8)	**21/64**	**0.32**

Cabe mencionar, que la cota de la robustez para este espacio es 0.6 que es significativamente mayor que la robustez experimental de cada S-caja.

4.0 PROGRAMAS EN GAP

4.1 Programa de Simulación del Data Encryption Standard.

En el siguiente programa podemos simular al DES, este análisis particularmente nos presenta el funcionamiento de este algoritmo de cifrado, lo cual nos puede permitir sugerir algunas modificaciones en sus S-cajas para aumentar su seguridad y observar varias propiedades de sus componentes.

```
dec:=function ( x )
   local  a, b, n, i;
   a := x;
   b := Size(a);
   n := 0;
   for i  in [ 1 .. b ]  do
      n := n + a[i] * 2 ^ (i - 1);
   od;
   return n;
end;
```

```
key:=[ [ 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0,
1, 0, 0, 1, 0, 0, 0, 1, 1 ], [ 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1,
1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1 ],   [ 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0,
```

```
1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0 ], [ 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0,0, 1, 1, 1, 1, 0,
1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0 ], [ 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1 ], [ 1, 1, 1, 0, 0, 1, 0, 1,
1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0 ], [ 1, 0,
1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1,
1, 0, 1 ], [ 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1,
1, 0, 1, 0, 1, 0, 0, 0, 1 ], [ 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1,
0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0 ], [ 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1 ], [ 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0,
1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0 ], [ 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1,
0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0 ], [ 1, 0, 1, 0, 1, 1, 1, 0, 0,
0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0 ], [ 1, 0, 1,
0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0,
0, 1, 0 ], [ 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0,
1, 1, 1, 1, 1, 1, 0, 0, 1 ], [ 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0,
1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1 ] ];

m:=function ( x )
   local  a, k, i;
   k := x;
   a := [  ];
   for i  in [ 1 .. k ]  do
      a[i] := Random( 0, 1 );
   od;
   return a;
end;

p:=function ( x )
   local  s, k, i;
   s := [  ];
   k := [ 1 .. x ];
   for i  in [ 1 .. x ]  do
      s[i] := Random( k );
      k := Difference( k, [ s[i] ] );
   od;
   return s;
end;

permutacion:=[ [ 15, 17, 19, 9, 25, 16, 7, 23, 26, 12, 5, 13, 28, 20, 14, 21, 1, 27, 3, 32, 6, 2, 22, 29, 18, 31,
24, 11, 4, 10, 8, 30 ], [ 32, 6, 5, 25, 12, 28, 9, 31, 21, 18, 15, 8, 27, 26, 2, 1, 19, 23, 17, 16, 10, 30, 7,24,13,
22, 11, 14, 29, 3, 20, 4 ], [ 3, 26, 28, 1, 14, 11, 21, 4, 12, 29, 19, 27, 15, 16, 2, 17, 18, 13, 6, 24, 8, 9,20,25,
10, 32, 23, 7, 22, 30, 31, 5 ], [ 29, 10, 26, 22, 8, 16, 2, 3, 7, 1, 24, 4, 9, 20, 12, 11, 32, 17, 6, 15, 19, 23, 14,
30, 27, 5, 28, 13, 25, 18, 21, 31 ], [ 2, 19, 18, 14, 20, 22, 23, 17, 11, 15, 28, 12, 24, 29, 31, 3, 21, 26, 5, 9, 6,
27, 30, 7, 8, 10, 4, 25, 1, 32, 16, 13 ], [ 13, 7, 27, 19, 6, 32, 2, 3, 31, 28, 15, 24, 21, 30, 20, 26, 12, 5, 11, 10,
8, 23, 1, 25, 29, 18, 9, 4, 22, 16, 17, 14 ], [ 16, 22, 29, 12, 14, 30, 24, 18, 6, 11, 25, 15, 31, 20, 7, 3, 9, 1, 32,
5, 17, 10, 19, 27, 23, 13, 8, 21, 4, 28, 26, 2 ], [ 4, 28, 18, 7, 31, 11, 9, 22, 17, 13, 8, 15, 6, 14, 10, 29, 20, 25,
21, 12, 1, 24, 5, 2, 30, 27, 32, 23, 26, 3, 19, 16 ], [ 30, 5, 25, 11, 28, 20, 14, 6, 16, 22, 24, 27, 9, 1, 4, 23, 10,
7, 18, 3, 29, 31, 13, 32, 21, 17, 26, 8, 2, 19, 12, 15 ], [ 28, 30, 4, 2, 22, 10, 32, 8, 31, 21, 26, 5, 13, 6, 15, 24,
16, 19, 18, 3, 11, 23, 29, 17, 1, 25, 12, 27, 9, 20, 7, 14 ], [ 21, 14, 23, 15, 12, 9, 5, 2, 24, 6, 26, 31, 4, 3, 32,
16, 25, 22, 27, 7, 10, 28, 29, 17, 18, 13, 20, 1, 19, 8, 11, 30 ], [ 12, 2, 27, 10, 11, 17, 25, 30, 13, 8, 28, 15,
```

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

```
diad:=function ( x, y )
   local  a, b, s, c, r, i;
   a := x;
   b := y;
   s := [ ];
   for i  in [ 1 .. b ]  do
      r := EuclideanRemainder( a, 2 );
      s[i] := r;
      a := EuclideanQuotient( a, 2 );
   od;
   return s;
end;

e1:=function(x)
 local a,b ;
 a:=x; b:=Random([1..2^16]);
return b;
end;

e2:=function(x,y)
 local a,b,c ;
 a:=x; b:=y; c:=Random([2^16..2^32]);
return c;
end;
```

caja:=[[[1, 5, 0, 2], [2, 4, 5, 6], [2, 5, 6, 7], [3, 5, 6, 7], [12, 13, 15, 5], [3, 5, 7, 10], [2, 6, 1, 8], [2, 4, 6, 2], [2, 6, 8, 9], [12, 15, 0, 1], [1, 5, 15, 9], [8, 9, 10, 14], [3, 5, 7, 9], [13, 15, 6, 7], [13, 5, 7, 9], [11, 15, 7, 9]] ,

[[11, 0, 9, 12], [0, 4, 5, 6], [2, 15, 6, 10], [3, 5, 6, 9], [12, 13, 15, 5], [9, 15, 7, 10], [0, 6, 11, 8], [4, 4, 6, 2], [9, 6, 8, 9], [12, 15, 0, 1], [5, 0, 5, 9], [9, 9, 10, 14], [3, 0, 7, 9], [3, 5, 6, 7], [3, 5, 10, 9], [7, 15, 7, 0]] ,

[[4, 0, 9, 4], [4, 7, 5, 7], [2, 5, 6, 0], [3, 5, 6, 9], [2, 3, 15, 5], [9, 5, 7, 10], [7, 6, 1, 8], [4, 4, 7, 2], [9, 6, 4, 7], [1, 15, 0, 1], [5, 0, 7, 9], [7, 9, 7, 4], [3, 0, 7, 9], [3, 5, 6, 7], [3, 2, 7, 9], [7, 8, 7, 0]] ,

[[4, 1, 9, 4], [1, 7, 5, 7], [2, 1, 6, 10], [1, 5, 6, 9], [12, 13, 15, 1], [9, 5, 1, 10], [7, 6, 1, 8], [4, 4, 7, 11], [9, 6, 4, 7], [11, 15, 0, 11], [5, 0, 7, 11], [7, 9, 11, 4], [3, 1, 7, 9], [3, 5, 11, 7], [3, 2, 7, 11], [7, 8, 11, 0]],

[[14, 11, 9, 3], [1, 3, 15, 7], [12, 11, 3, 10], [3, 5, 3, 9], [3, 3, 5, 11], [9, 5, 1, 1], [7, 3, 1, 8], [14, 14, 13, 11], [9, 6, 4, 3], [1, 5, 0, 1], [13, 10, 14, 11], [7, 9, 1, 3], [3, 3, 7, 9], [3, 14, 11, 7], [3, 2, 3, 1], [7, 8, 1, 3]] ,

```
[[ 15, 6, 9, 0 ], [ 1, 6, 5, 7 ], [ 12, 1, 3, 0 ], [ 3, 5, 3, 0 ], [ 3, 3, 15, 6 ], [ 9, 5, 0, 6 ], [ 7, 3, 1, 8 ], [ 4, 4, 3, 6 ],
[ 9, 6, 10, 6 ], [ 1, 5, 10, 1 ], [ 13, 0, 14, 1 ], [ 7, 9, 10, 6 ], [ 13, 13, 7, 9 ], [ 13, 14, 6, 7 ], [ 13, 12, 10, 6 ], [
7, 8, 11, 6 ]] ,

 [[ 5, 7, 9, 7 ], [ 1, 7, 5, 7 ], [ 12, 1, 3, 0 ], [ 3, 5, 3, 0 ], [ 13, 13, 5, 6 ], [ 9, 7, 0, 6 ], [ 7, 7, 1, 8 ], [ 14, 14, 13,
7 ], [ 9, 7, 0, 6 ], [ 11, 7, 0, 1 ], [ 13, 10, 14, 7 ], [ 7, 7, 0, 6 ], [ 3, 7, 7, 9 ], [ 3, 4, 7, 7 ], [ 13, 12, 10, 7 ], [ 7,
7, 1, 7 ]] ,

 [[ 15, 9, 9, 9 ], [ 1, 7, 15, 7 ], [ 12, 1, 13, 0 ], [ 3, 9, 13, 0 ], [ 13, 13, 15, 9 ], [ 9, 7, 0, 6 ], [ 7, 7, 1, 8 ], [ 14,
14, 3, 9 ], [ 9, 12, 0, 6 ], [ 11, 7, 0, 1 ], [ 13, 0, 4, 9 ], [ 7, 13, 10, 6 ], [ 13, 7, 7, 9 ], [ 3, 4, 9, 7 ], [ 13, 2, 0, 9],
[ 7, 12, 11, 7 ]] ];

suma:=function ( x, y )
   local  a, b, c;
   a := x;
   b := y;
   if a = b  then
      c := 0;
   else
      c := a + b;
   fi;
   return c;
end;

ad:=function ( x, y )
   local  a, b, c, m, i;
   a := x;
   b := y;
   m := Size( a );
   c := [  ];
   for i  in [ 1 .. m ]  do
      c[i] := suma( a[i], b[i] );
   od;
   return c;
end;

ext:=function ( x )
   local  a, b, i;
   a := x;
   b := [  ];
   for i  in [ 1 .. 32 ]  do
      b[i] := a[i];
   od;
   for i  in [ 33 .. 48 ]  do
      b[i] := a[i - 32];
   od;
   return b;
end;
```

```
box:=function(x,y)
 local a,b,c,e,f;
 a:=x;

 c:=dec([a[2],a[3],a[4],a[5]])+1;
 b:=dec([a[1],a[6]])+1;
 e:=caja[y][c][b];
 f:=diad(e,4);
return f;
end;

int:=function(x,y,z)
local a,i,j,k,s;
a:=x; i:=y; j:=z; s:=[];
for k in [1..j-i+1] do
s[k]:=a[i+k-1];
od;
return s;
end;

part:=function(x)
local a,b,c,i,s;
a:=x;b:=[];c:=0;s:=[1,7,13,19,25,31,37,43];
for i in s do
c:=c+1;b[c]:=int(a,i,i+5);
od;
return b;
end;

Box:=function(x)
local a,b,c,i;
a:=x;b:=part(a);c:=[];
for i in [1..8] do
c[i]:=box(b[i],i);
od;
return c;
end;

glue:=function(x)
local a,b,i;
a:=x;b:=[];
b[1]:=a[1];
for i in [2..8] do
b[i]:=Concatenation(b[i-1],a[i]);
od;
return b[8];
end;
```

```
Per:=function(x,y)
local a,b,c,i;
a:=x;b:=permutacion[y];
c:=[];
for i in [1..32] do
c[i]:=a[b[i]];
od;
return c;
end;

F:=function(x,i)
local a,b,k;
a:=x;k:=i;
b:=Per(glue(Box(ad(ext(a),key[k]))),k);
return b;
end;

encrypt:=function(x)
local a,l,r,i,ia,da,fa;
a:=x;
l:=[];r:=[];
l[1]:=int(a,1,32);
r[1]:=int(a,33,64);
for i in [2..17] do
l[i]:=r[i-1];
r[i]:=ad(l[i-1],F(l[i],i-1));
od;
ia:=l[17];
da:=r[17];

fa:=Concatenation(ia,da);
return fa;
end;

desencrypt:=function(x)
local a,l,r,i,ia,da,fa;
a:=x;
l:=[];r:=[];
l[1]:=int(a,1,32);
r[1]:=int(a,33,64);
for i in [2..17] do
r[i]:=l[i-1];
l[i]:=ad(r[i-1],F(r[i],18-i));
od;
ia:=l[17];da:=r[17];

fa:=Concatenation(ia,da);
```

```
return fa;
end;
part:=function(x)
local a,b,c,i,s;
a:=x;b:=[];c:=0;s:=[1,7,13,19,25,31,37,43];
for i in s do
c:=c+1;b[c]:=int(a,i,i+5);
od;
return b;
end;
```

Observe que, para un mensaje **m** de 64 bits, **encryp(m)** es el mensaje cifrado con la función DES y que **desencrypt(encrypt(m))=m** es el mensaje descifrado.

4.2 S-Cajas del Data Encryption Standard.

La siguiente lista de funciones regulares de V_6 en V_4 son las s-cajas utilizadas en DES (ver [12]).

B1:=[14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 16, 7, 16, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8, 4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 16, 15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3 ,14, 10, 16, 6, 13];

B2:= [15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 16, 5, 10, 3, 13, 4, 7, 15, 2, 8, 14, 12, 16, 1, 10, 6, 9, 11, 5, 16, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15, 13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 16, 5, 14, 9];

B3:=[10, 16, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8, 13, 7, 16, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1, 13, 6, 4, 9, 8, 15, 3, 16, 11, 1, 2, 12, 5, 10, 14, 7, 1, 10, 13, 16, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12];

B4:=[7, 13, 14, 3, 16, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15, 13, 8, 11, 5, 6, 15, 16, 3, 4, 7, 2, 12, 1, 10, 14, 9, 10, 6, 9, 16, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4, 3, 15, 16, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14];

B5:=[2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 16, 14, 9, 14, 11, 2, 12, 4, 7, 13, 1, 5, 16, 15, 10, 3, 9, 8, 6, 4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 16, 14, 11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 16, 9, 10, 4, 5, 3];

B6:=[12, 1, 10, 15, 9, 2, 6, 8, 16, 13, 3, 4, 14, 7, 5, 11, 10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 16, 11, 3, 8, 9, 14, 15, 5, 2, 8, 12, 3, 7, 16, 4, 10, 1, 13, 11, 6, 4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 16, 8, 13];

B7:=[4, 11, 2, 14, 15, 16, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1, 13, 16, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6, 1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 16, 5, 9, 2, 6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 16, 15, 14, 2, 3, 12];

B8:=[13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 16, 12, 7, 1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 16, 14, 9, 2, 7, 11, 4, 1, 9, 12, 14, 2, 16, 6, 10, 13, 15, 3, 5, 8, 2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 16, 3, 5, 6, 11];

4.3 Cálculo de la Robustez de Funciones Regulares de V_6 a V_4.

Mediante al siguiente programa podemos calcular la robustez de una función regular, en este caso el programa que se presente permite calcular la robustez de las S-cajas del DES. Sin embargo, al modificar el campo de la suma podemos utilizarlo para cualquier espacio de funciones regulares.

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

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

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

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

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

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

sum:=function ( x, y )
   local  a, b, c;
   a := x;
   b := y;
   c := S[a + 1][b + 1];
   return c;
end;

dij:=function ( a,b,c)
   local  s, i, j, d, x;
   s := a;
   i := b;
   j := c;
   d:= 0;
   for x in [ 0 .. 63 ]  do
      if sum( s[sum( x, i ) + 1], s[x+1] ) = j  then
         d := d + 1;
      fi;
   od;
   return d;
end;

R:=function(x)
local R1, r, R, i, j, a, n, s;
s:=x;
R:=[];
R1:=[];
n:=0;
for i in [1..63] do
for j in [0..15] do
n:=n+1;
R[n]:=dij(s,i,j);
od; od;
R1:=Elements(R);
a:=Size(R1);
r:=R1[a];
return r;
end;

L:=function(a)
local s, x, L;
L:=0;
s:=a;
```

```
for x in [1..63] do
if dij(s,x,0)<>0 then L:=L+1;
fi;
od;

return L;
end;


ROB:=function(x)
local s, r;
s:=x;
r:=(1-L(s)/64)*(1-R(s)/64);
return r;
end;
```

Note que los valores de L, R son los mismos que aparecen en la definición *4.0,* y que la acción **ROB(S)** es calcular la robustez de la función S.

4.4 Programas Alternos.

4.4.1 Cálculo de las Funciones Regulares de un Espacio Vectorial de Dimensión pequeña.

El siguiente programa calcula todas las funciones regulares de V_4 en V_2.

```
 f:=function(x,y)
 local m,a,b,m1,m2;
a:=y;b:=x[1];
m:=Difference(x,[b]);
  m1:=UnionSet([b],Combinations(m,3)[a]);
  m2:=Difference(x,m1);
  return [m1,m2];
  end;
  bal:=function(x,y,z)
  local a,b,c,m1,m2,m3,p1,p2,p3,p4,g,i;
  a:=x;b:=y;c:=z;g:=[];
  p1:=f([1..16],a)[1];
  for i in p1 do g[i]:=0;od;
  m1:=f([1..16],a)[2];
  p2:=f(m1,b)[1];
  for i in p2 do g[i]:=1;od;
  m2:=f(m1,b)[2];
  p3:=f(m2,c)[1];
  for i in p3 do g[i]:=2; od;
  m3:=f(m2,c)[2];
  p4:=f(m3,1)[1];
```

```
for i in p4 do g[i]:=3;od;

return g;
end;
Print(" en bal(a,b,c) ");
Print(" 0 < a < 456; 0 < b <166; 0 < c < 36                    ");
```

4.4.2 Cálculo de la Tabla de Distribución Diferencial y La Robustez de una Función Regular.

El siguiente programa calcula la tabla de distribución diferencial y la robustez de las funciones regulares de V_4 en V_2. Observe que al cambiar el campo $\mathbf{F_4}$ este programa funciona para otros espacios de funciones regulares.

El Programa *(4.3)* calcula la robustez de una función regular con una rapidez superior a este programa. Sin embargo, este programa nos permite observar la tabla de distribución diferencial de la función regular.

```
F4:=Tuples(GF(2),4);;

 num:=function(x)
 local a,i;
 a:=x;
 for i in [1..16] do
 if a=F4[i] then break; fi;
 od;
 return i;
 end;

 suma:=function(x,y)
 local a,b,c;
 a:=x;
 b:=y;
 c:=num(F4[a+1]+F4[b+1])-1;
 return c;
 end;

vsuma:=function(x,y)
 local a,b,c,i;
 a:=x;
 b:=y;
 c:=[];
 for i in [1..16] do
 c[i]:=suma(a[i],b[i]);
 od;
```

```
 return c;
 end;

presuma:=function(x,y)
 local a,b,i,c;
 a:=x;
 b:=y;
 c:=[];
 for i in [1..16] do
 c[i]:=a[suma(i-1,b)+1];
 od;
 return c;
 end;

md:=function(x)
 local a,m,i;
 a:=x;
 m:=[];
 for i in [1..16] do
 m[i]:=presuma(a,i-1);
 od;
 return m;
 end;

reng:=function(x)
 local a,b,p,q,r,s,i;
 a:=x;
 b:=[];
 p:=0;q:=0;r:=0;s:=0;
 for i in [1..16] do
 if a[i]=0 then p:=p+1;fi;
 if a[i]=1 then q:=q+1;fi;
 if a[i]=2 then r:=r+1;fi;
 if a[i]=3 then s:=s+1;fi;
 od;
 b:=[p,q,r,s];
 return b;
 end;

td:=function(x)
 local a,b,i;
 a:=md(x);
 b:=[];
 for i in [1..16] do
 b[i]:=reng(vsuma(a[i],a[1]));
 od;
 return b;
 end;
```

```
rob1:=function ( x )
   local  a, s, i;
   a := td( x );
   s := 0;
   for i  in [ 1 .. 15 ]  do
      if a[i + 1][1] <> 0  then
         s := s + 1;
      fi;
   od;
   return s;
end;

 rob2:=function(x)
local a,m,i;
a:=td(x);m:=[];
for i in [1..15] do
m:=UnionSet(m,a[i+1]);
od;
return m[Size(m)];;
end;
rob:=function(x)
local a,r;
a:=x;r:=(1-rob1(a)/16)*(1-rob2(a)/16);
return r; end;

rb:=function(x,y)
 local m,n,i,r;
 m:=x;
 n:=y; r:=[];
 for i in [m..n] do
 r:=UnionSet(r,[rob(bal[i])]);
 od;
 return r;
 end;
```

Note que dada una función regular S, **td(S)** nos imprime los valores de la tabla de distribución diferencial y **rb(S)** nos da el valor de su robustez.

5.0 CONCLUSIONES.

En este trabajo abordamos el problema de encontrar las mejores S-cajas en un sistema criptográfico simétrico tipo DES, para ello estudiamos las características de estas funciones en los trabajos de J. Seberry , H. Tapia-Recillas y J. Castañeda ([4],[5],[6],[10] , [14]).

El resultado principal de esta tesis es ofrecer un método eficiente para el cálculo de la robustez de funciones regulares considerando un conjunto de reducciones en el espacio de funciones regulares debido a la acción del grupo afin $\square\square\square(\square,2)$. Al parecer, la acción del grupo afín en el espacio de funciones regulares no había sido identificada por otros autores.

Se presentan los siguientes resultados:

1. La función $\square_x : V_\square \to V_\square$ *definida por* $\square_x(i) = \square(x) + \square(x+i)$ *se puede interpretar como la composición de las traslaciones* $\square_1 : V_\square \to V_\square$, $\square_2 : V_\square \to V_\square$ *y la función* $\square : V_\square \to V_\square$.

$$V_\square \xrightarrow{\square_1} V_\square \xrightarrow{\square} V_\square \xrightarrow{\square_2} V_\square$$
$$i \mapsto x+i \mapsto \square(x+i) \mapsto \square(x+i)+\square(x)$$

2. Dada una función $\square : V_\square \to V_\square$, *las entradas de la matriz* $\square^{(\square)} = \left(d_{i\square}(\square)\right)$ *son números enteros pares.*

3. Dada una función regular $\square : V_\square \to V_\square$ *y* $\square : V_\square \to V_\square$; $\square : V_\square \to V_\square$ *funciones biyectivas, entonces la composición* $V_\square \xrightarrow{\square} V_\square \xrightarrow{\square} V_\square \xrightarrow{\square} V_\square$ *es también regular.*

4. Sea $\square : V_\square \to V_\square$ *una función regular, el valor* $d_{1,0}(\square) \neq 2^\square - 2$ *en la tabla de distribución diferencial de S.*

5. Sean $\square : V_\square \to V_\square$, $\square' : V_\square \to V_\square$ *traslaciones y* $\square : V_\square \to V_\square$ *función regular, entonces* $\square\square\square(\square \circ \square) = \square\square\square(\square) = \square\square\square(\square \circ \square')$.

6. Sean $\square : V_\square \to V_\square$, $\square' : V_\square \to V_\square$ *dos transformaciones lineales invertibles,* $\square : V_\square \to V_\square$ *función regular. Entonces,* $\square\square\square(\square' \circ \square \circ \square) = \square\square\square(\square) = \square\square\square(\square \circ \square \circ \square')$.

Considerando los resultados anteriores, se calculo que la máxima robustez de funciones regulares para el espacio $\square_{3,2}$ *es de ¼ y para el espacio* $\square_{4,2}$ *es 7/32, para calcular la robustez de estas funciones es necesario calcular todas las funciones regulares del espacio dado. Sin embargo, al considerar estos resultados fue posible reducir el número de funciones necesarias, de* $\frac{2^3!}{(2)!^4} = 2{,}520$ *a solo 105 funciones para el espacio* $\square_{3,2}$ *y de* $\frac{2^4!}{(2^{4-2})!^4} = \frac{16!}{24^4} = 63{,}063{,}000$ *a solo 11,550 funciones para el espacio* $\square_{4,2}$. *En este segundo caso, logramos reducir el tiempo de calculo computacional de 17.5175 horas a solo 11.5 segundos. Cabe mencionar, que debido a que* $\square_4$ *es isomorfo al grupo afin Aff(2,2) el análisis presentado en esta tesis facilita el estudio de la acción de* $\square\square\square(\square,2)$ *en el espacio de las*

funciones regulares $\square_{\square 2}$. *Sin embargo,* $\square\square\square(\square,\square) \leq \square_{2\square}$ *pero no necesariamente* $\square_{2\square} \cong \square\square\square(\square,\square)$, *comprender como afecta la acción del grupo* $\square\square\square(\square,\square)$ *a la robustez de las funciones regulares cuando* $\square_{2\square} \ncong \square\square\square(\square,\square)$ *es un problema más complejo.*

6.0 REFERENCIAS

[1] Biham Eli, Shamir Adi, *Differential cryptanalysis of the Full 16-round DES*. Lecture Notes in Computer Science: Advances in Cryptology-Proceedings of CRYPTO 92, Springer-Verlag, 1990, pág 2-21.

[2] Ding Cunsheng , *The Data Encryption standard in Detail*. Department of Computer Science, Hong Kong University of Science and Technology Clearwater Bay, Kowloon, Hong Kong, China. 2000.

[3] D. Coppersmith, *The Data Encryption Standard (DES) and its strength against attacks*. IBM J. RES. DEVELOP. VOL: 38 No. 3 may 1994.

[4] **Castañeda R. Jesús**, *El Data Encryption Standard y Mapeos Regulares*. Memorias del 4to Congreso Internacional y 2do Nacional de Métodos Numéricos y Matemáticas Aplicadas. Sociedad Mexicana de Métodos Numéricos. Morelia, Michoacán, México. Enero de 2007.ISBN. 978-84-96736-08-5.

[5] **Castañeda R. Jesús**, *La Robustez de Mapeos Regulares*. Memorias del 4to Congreso Internacional y 2do Nacional de Métodos Numéricos y Matemáticas Aplicadas. Sociedad Mexicana de Métodos Numéricos. Morelia, Michoacán, México. Enero de 2007. ISBN. 978-84-96736-08-5.

[6]. **Castañeda R. Jesús**, *The Affine Group Aff(n,2) and The Robust of Regular Mappings*, Memories of the 8th. World Congress on Computational Mechanics (WCCM8) and 5th. European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2008),Venice, Italy. Junio de 2008.

[7] J. Rifa, Ll. Huguet. *Comunicación Digital*. Masson, S. A., Barcelona, 1991.

[8] Conway H. John, *Tree Exceptional Lecture of Group Theory*, Sphere Packing, Codes and Groups. Springer Verlag, 2003.

[9] Cárdenas T. Humberto, Lluís Emilio, *Módulos Semisimples y Representaciones de Grupos Finitos,* Serie 1, Sociedad Matemática Mexicana, Editorial Trillas, 1970.

[10] H. Tapia-Recillas, G. vega, E. Daltabuit. *Some Results on Regular Mappings*. Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, Lecture Notes in Computer Science. 12th International Symposium, AAECC-12, France, June 1997.

[11] Hall Marshall, *The Theory of Groups*, Macmillan Company. 1973.

[12] National Bureau of Standards FIPS PUB 46-2 Supersedes FIPS PUB 46-1. *Data Encryption Standard*, Federal information Processing Standards Publication 46-2. 1993.

[13] Roman Joseph. *An* Introduction *to the Theory of Groups*, Springer Verlag, 1993.

[14] Seberry, J., Zhang, X. M. M and Zheng, Y. *Systematic generation of cryptographically robust S-boxes*. In Proceedings of the first ACM Conference on Computer and Communications Security, the Association for Computing Machinery, New York (1993) 172-182.