TEMA
Data Encryption Standard - DES

El algoritmo Data Encryption Standard (DES), también conocido como Data Encryption Algorithm (DEA), es el método de cifrado simétrico más ampliamente conocido. Ha sido el primer algoritmo moderno de cifrado para intereses comerciales, difundido de forma abierta y completa a todo el mundo, en todos sus detalles de implementación. Su estándar queda recogido en American Standard FIPS 46–2 y FIPS 46–3 (donde se añade el algoritmo TripleDES o TDEA: Triple Data Encryption Algorithm). Estos documentos están disponibles y se encuentran fácilmente en la web: vale la pena disponer de ellos.

(FIPS: "Federal Information Processing Standards" son estándares anunciados públicamente desarrollados por el gobierno de los Estados Unidos para su utilización por parte de todas las agencias del gobierno no militares y por los contratistas del gobierno. Muchos estándares FIPS son versiones modificadas de los estándares usados en las comunidades más amplias como por ejemplo ANSI, IEEE, ISO, etc.)

1. UNA BREVE INTRODUCCIÓN HISTÓRICA (www.wikipedia.es).

Los orígenes de DES se remontan a principios de los 70. En 1972, la autoridad de estándares estadounidense NBS (National Bureau of Standards) —ahora rebautizado **NIST** (National Institute of Standards and Technology)— señaló, en un estudio sobre las necesidades del gobierno en el ámbito de la seguridad informática, la necesidad de un estándar a nivel gubernamental para cifrar información confidencial.

El 15 de mayo de 1973, tras consultar con la NSA (National Security Agency), el NBS solicitó propuestas para un algoritmo que cumpliera rigurosos criterios de diseño: ninguna de las propuestas que recibió pareció adecuada. Se realizó una

segunda petición el **27 de agosto de 1974**. Esta vez **IBM** presentó un algoritmo que fue considerado aceptable, desarrollado durante el periodo 1973–1974 y basado en otro anterior, el algoritmo **Lucifer** de **Horst Feistel**.

El papel de la NSA en el diseño. El **17 de marzo de 1975**, la propuesta de IBM fue publicada en el Registro Federal. En algunos sectores se criticó la seguridad del algoritmo, señalando dos aspectos que podían hacerlo vulnerable:

- **La corta longitud de la clave**.
- Las misteriosas **S-cajas** (más adelante, en esta presentación de DES se verá a qué nos referimos), que evidenciaban la inadecuada interferencia de la NSA.

La sospecha era que el algoritmo había sido debilitado de manera secreta por la agencia de inteligencia de forma que ellos —y nadie más— pudiesen leer mensajes cifrados fácilmente.

Algunas de las sospechas sobre puntos débiles ocultos en las S-cajas fueron descartadas en 1990, con el descubrimiento independiente y la publicación libre por Eli Biham y Adi Shamir del **criptoanálisis diferencial**, un método general para romper cifrados de bloque. Las S-cajas de DES eran mucho más resistentes al ataque que si hubiesen sido escogidas al azar, lo que sugería que IBM conocía la técnica allá en los 70. Éste era de hecho, el caso: en 1994, se publicaron los criterios de diseño originales para las S-cajas; IBM había descubierto el criptoanálisis diferencial en los 70 y, tras asegurar DES, la NSA les ordenó mantener en secreto la técnica. De hecho no hay evidencias de una influencia negativa en el diseño de DES destinada a debilitar el criptosistema. Al menos en lo que hace referencia a las S-cajas.

Las otras críticas —sobre que la longitud de la clave era demasiado corta— se fundaban en el hecho de que la razón dada por la NSA para reducir la longitud de la clave de 64 bits a 56 era que los 8 bits restantes podían servir como bits de paridad, lo que en cierto modo resultaba sospechoso (también esta característica de la clave se verá más adelante). Se sabe que la NSA animó, o incluso persuadió a IBM para que redujera el tamaño de clave de 128 bits a 64, y de ahí a 56 bits; con frecuencia esto se ha interpretado como una evidencia de que la NSA poseía suficiente capacidad de computación para romper claves de este tamaño incluso a mediados de los 70.

El algoritmo como estándar. A pesar de la polémica, DES fue aprobado como estándar federal en noviembre de 1976, y publicado el 15 de enero de 1977

como FIPS PUB 46, autorizado para el uso no clasificado de datos. Fue posteriormente confirmado como estándar en 1983, 1988 (revisado como FIPS-46-1), 1993 (FIPS-46-2), y de nuevo en 1998 (FIPS-46-3), éste último definiendo "TripleDES". El **26 de mayo de 2002**, DES fue finalmente reemplazado por **AES** (**Advanced Encryption Standard**), tras una competición pública. Hasta hoy día DES continúa siendo ampliamente utilizado. Fue un ataque por fuerza bruta en 1998 el que demostró que DES podría ser atacado en la práctica, y se destacó la necesidad de un algoritmo de repuesto.

2. FUNDAMENTOS TEÓRICOS PREVIOS DE DES

El diseño de DES se basa en dos conceptos generales de criptografía: en el **cifrado de producto** y en el **cifrado de Feistel**. Es conveniente, antes de entrarnos en la descripción del algoritmo DES, presentar estos dos conceptos, que dan luego forma al algoritmo. Veamos pues cada uno de estos dos conceptos:

Cifrado de producto.

La idea básica del cifrado de producto es la de **construir una función de cifrado compleja a partir de la composición de varias operaciones simples que se ofrecen de forma complementaria, y que son individualmente insuficientes para lograr un criptosistema robusto**. Sus operaciones básicas incluyen las trasposiciones, las traslaciones (por ejemplo, las realizadas mediante el operador XOR) y transformaciones lineales, operadores aritméticos, multiplicación modular y simple sustitución.

Un cifrado de producto combina dos o más transformaciones con una intención determinada, de manera que el cifrado resultante es más seguro que el de cada transformación por separado. Muchos de los algoritmos simétricos de cifrado en la práctica son cifrados de producto.

Llamamos ITERACIÓN DE CIFRADO DE BLOQUE a un cifrado de bloque que implica la repetición secuencial de una función interna. Los parámetros que definen este cifrado son el número de vueltas o iteraciones, r, el tamaño del bloque, n, y el número de bits, k, del total de la clave K que tomaremos en cada

vuelta para crear cada una de las r sub-claves K_i: Cada una de las r iteraciones se realiza con su correspondiente clave K_i.

Cifrados de Feistel.

El cifrado de Feistel es un cifrado iterado, que transforma un texto plano $m = (L_0, R_0)$ de longitud $2 \cdot t$ bits en un texto cifrado $c = (L_r, R_r)$, al que se llega después de repetir un proceso r veces, donde $r \geq 1$. Cada bloque L_0 y R_0 tiene una longitud igual a t. El tamaño de la clave de cifrado $K \in \mathcal{K}$ es de k bits.

Cada iteración $(1 \leq i \leq r)$ que realiza la transformación $(L_{i-1}, R_{i-1}) \xrightarrow{K_i} (L_i, R_i)$ se define de la siguiente manera: $L_i = R_{i-1},\ R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$, donde K_i es una clave derivada de la clave de cifrado K y donde $\oplus$ es la operación OR exclusivo, también llamada de suma módulo 2

Queda pendiente determinar cómo se generan las sucesivas K_i a partir de la clave $K \in \mathcal{K}$, que pertenece al campo de claves del criptosistema. También queda pendiente definir la función f_{K_i}. Eso será una cuestión de cada criptosistema que use el esquema de los cifrados de Feistel.

La seguridad del cifrado de Feistel depende de la seguridad de su cifrado de bloque interno (la función f_{K_i}). La seguridad aumenta al iterar repetidamente esa transformación. En los cifrados de Feistel se toma $r \geq 3$, y habitualmente un valor par. Como se ha visto, la estructura de los algoritmos de Feisel lleva a un texto cifrado de la forma (R_r, L_r), y no (L_r, R_r); de forma general, después de la última vuelta, los dos bloques se intercambian.

Para descifrar un texto cifrado con la técnica de Feistel se sigue el proceso inverso: $R_{i-1} = L_i;\ L_{i-1} = R_i \oplus f(L_i, K_i)$, para $1 \leq i \leq r$. Realizando esta operación r veces con la secuencia de claves a la inversa, $K_r, K_{r-1}, \ldots K_1$, llegaremos al texto plano (R_0, L_0).

3. UN ALGORITMO TIPO DES SIMPLIFICADO.

Presentamos un algoritmo de aspecto muy semejante al DES, pero mucho más reducido. Al igual que DES es un algoritmo de cifrado por bloques. Como se verá, es un algoritmo basado en los cifrados de Feistel. Este algoritmo fue presentado por el profesor Edward Schaefer, de la Universidad de Santa Clara. Es un algoritmo válido para la docencia, y no pretende ser un algoritmo de seguridad criptográfica. Tiene propiedades y estructura muy similares a las del algoritmo

DES, con parámetros muchos menores. Puede ser un buen ejercicio implementar este algoritmo y probar su funcionamiento.

Como cada bloque se cifra por separado, vamos a suponer que nuestro mensaje a cifrar consiste en un solo bloque. El mensaje plano $m = L_0R_0$ tiene 12 bits, que dividimos en dos mitades: L_0 consiste en el bloque de los 6 primeros bits del mensaje, y R_0 los 6 bits últimos.

La clave K es de 9 bits.

La iteración $i-ésima$ del algoritmo transforma una entrada $L_{i-1}R_{i-1}$ en la salida L_iR_i, usando una subclave K_i de 8 bits, derivada de la clave K. (Esquema de Feistel.) La función $f(R_{i-1}, K_i)$ de Feistel toma una entrada de 6 bits R_{i-1} y una entrada K_i de 8 bits y produce una salida de 6 bits. La salida de la $i-ésima$ iteración queda definida así: $L_i = R_{i-1}$, y $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$. Este proceso se itera r veces y produce el texto cifrado L_rR_r.

Para descifrar, comenzamos por el texto cifrado L_rR_r. En un primer paso, intercambiamos los bloques de la izquierda y la derecha para obtener R_rL_r: ésa es la cadena que vamos a someter al descifrado. Ahora realizamos las mismas operaciones que antes, pero usando las claves K_i en orden inverso: El primer paso toma como entrada R_rL_r y da como salida $[L_r]$ y $[R_r \oplus f(L_r, K_r)]$.

Por el procedimiento seguido en el cifrado, sabemos que $L_r = R_{r-1}$, y que $R_r = L_{r-1} \oplus f(R_{r-1}, K_r)$, Entonces $[L_r]\,[R_r \oplus f(L_r, K_r)] = [R_{r-1}]\,[L_{r-1} \oplus f(R_{r-1}, K_r) \oplus f(L_r, K_r)] = [R_{r-1}]\,[L_{r-1}]$.

Para la última igualdad hemos tenido en cuenta que $L_r = R_{r-1}$, así que $f(R_{r-1}, K_r) \oplus f(L_r, K_r) = 0$.

De forma similar, el segundo paso del proceso de descifrado transforma $L_{r-1}R_{r-1}$ en $L_{r-2}R_{r-2}$. Y así, vemos que al final del proceso se habrá regresado al valor R_0L_0. Intercambiando los bloques de izquierda y derecha llegamos finalmente al texto plano original L_0R_0.

El proceso de descifrado es esencialmente el mismo que el de cifrado. Simplemente se han de intercambiar inicialmente los bloques izquierdo y derecho y se han de usar las claves en orden inverso. Por lo tanto, tanto el que envía como el que recibe usan una misma clave común, y pueden usar máquinas idénticas: el mismo algoritmo.

Queda pendiente explicar en qué consiste la función f y aclarar cómo se generan los claves K_i. Cualquier f podría trabajar en el procedimiento antes descrito. Pero algunas elecciones son mucho mejores que otras. El tipo de la función f usado en DES es similar al que se describe a continuación. Para definir ahora nuestra función f necesitamos construir primero sus componentes:

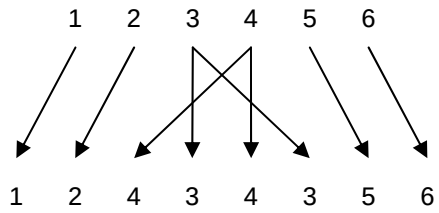

Figura 1: La función de expansión.

El primer componente es una función de **expansión**. Tiene una entrada de 6 bits y una salida de 8: por ejemplo, podría ser la que se recoge en la figura 1.

Por ejemplo, 011001 se expande a 01010101: El tercer bit ocupa la cuarta y la sexta posición; el cuarto bit ocupa la tercera y quinta posición.

Los componentes principales de nuestro algoritmo de cifrado se llaman S–boxes (**cajas**). Ahora usaremos dos:

$$S_1 = \begin{bmatrix} 101 & 010 & 001 & 110 & 011 & 100 & 111 & 000 \\ 001 & 100 & 110 & 010 & 000 & 111 & 101 & 011 \end{bmatrix}$$

$$S_2 = \begin{bmatrix} 100 & 000 & 110 & 101 & 111 & 001 & 011 & 010 \\ 101 & 011 & 000 & 111 & 110 & 010 & 001 & 100 \end{bmatrix}$$

La entrada a las cajas es de 4 bits. El primer bit especifica qué fila se va a utilizar: un 0 indica la fila de arriba; un 1 indica la fila de abajo. Los otros tres bits representan la codificación binaria de la columna que se va a utilizar: desde la columna 0 (000) situada en el extremo izquierdo hasta la columna 7 (111) situada en el extremo derecho. Por ejemplo, la entrada 1010 para S_1 da como salda el valor de la fila de abajo, en la columna tercera: 110.

Ya hemos indicado que la clave K está formada por 9 bits y que las distintas claves K_i para cada una de las iteraciones de cifrado se obtienen de K. El procedimiento que se va a seguir para la obtención de cada una de las claves es tomar 8 bits de K, comenzando por el bit en la posición i. Por ejemplo, si $K = 010011001$, entonces $K_4 = 01100101$ (después de tomar del bit 4 al bit 9, tomamos los dos restantes del inicio de los bits de K).

Con todos estos elementos o componentes ya podemos definir la función $f(R_{i-1}, K_i)$ (ver figura 2):

1. La entrada R_{i-1} está formada por 6 bits: primero la **expandimos** con la función de expansión y obtenemos 8 bits.
2. Con el resultado se realiza la operación **OR exclusivo** con la clave K_i.
3. Los nuevos 8 bits obtenidos se dividen en dos bloques de 4 bits: los 4 primeros se toman como entrada para S_1, y los cuatro últimos como entrada para S_2. Cada una de las dos cajas da una salida de 3 bits, que al concatenar forman una cadena de 6 bits.

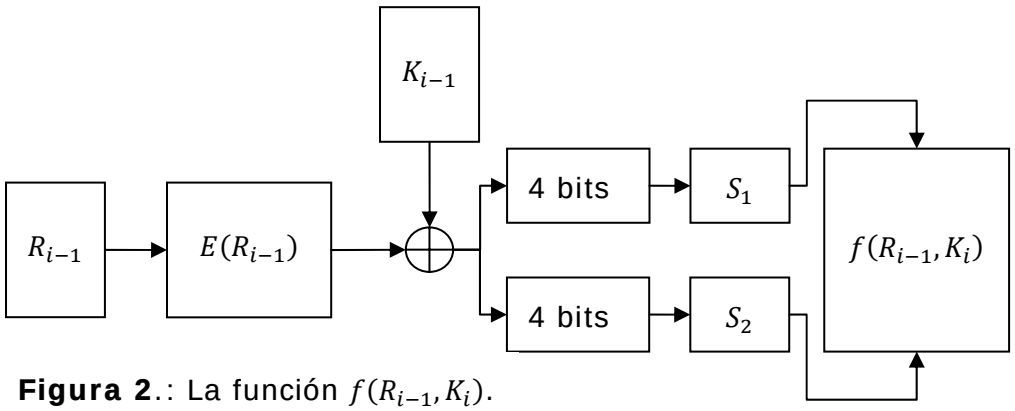

Figura 2.: La función $f(R_{i-1}, K_i)$.

La concatenación de las salidas de las dos cajas forma la salida de $f(R_{i-1}, K_i)$.

Por ejemplo, si $R_{i-1} = 100110$ y $K_i = 01100101$, tendremos que $E(100110) \oplus K_i = 10101010 \oplus 01100101 = 11001111$. Los primeros cuatro bits se envían a S_1, y los cuatro últimos a S_2. Tendremos las salidas 000 y 100, que concatenados nos da $f(R_{i-1}, K_i) = 000100$.

Veamos una iteración completa. Supongamos que tenemos la entrada $L_{i-1}R_{i-1} = 011100\,100110$ y $K_i = 01100101$. Hemos tomado los valores de antes, y $R_{i-1} = 100110$, por lo que $f(R_{i-1}, K_i) = 000100$. Por lo tanto tenemos que la salida final es $L_i = R_{i-1}$, y $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$, es decir, $L_iR_i = 100110\,011000$.

4. DES.

Es un algoritmo de cifrado de bloques de 64 bits. Su clave es de 56 bits, pero viene expresada en una cadena de 64 bits: En cada byte tenemos introducido un bit de paridad. La salida cifrada será otro bloque de 64 bits.

Los espacios de los textos planos y los textos cifrados son $\mathcal{M} = \mathcal{C} = \{0,1\}^{64}$. El espacio de las claves está formado por todas las cadenas de 64 bits que verifiquen la siguiente propiedad: si dividimos esos 64 bits en sus 8 bytes, entonces la suma de esos ocho bites en cada byte debe ser impar: eso significa que siete de los ocho bits determinan el valor del octavo. Es decir, el espacio de claves es

$\mathcal{K} = \{(b_1, \dots, b_{64}) \in \{0,1\}^{64} : \sum_{i=1}^{8} b_{8k+i} \equiv 1 \bmod 2; 0 \leq k \leq 7\}$, donde $b_{8\cdot i}$ $(1 \leq i \leq 8)$ son los bits de paridad.

El número de claves posibles es $2^{56} \sim 7.2 * 10^{16}$. Un ejemplo de clave DES válida dada en hexadecimal es, por ejemplo, 13 34 57 79 9B BC DF F1.

El algoritmo DES de cifrado comienza con un bloque de texto plano $m = (m_1 m_2 \dots m_{64})$, de 64 bits, y sigue los siguientes tres pasos (en la figura 3 se puede ver un esquema del funcionamiento del algoritmo):

1. Los bits de m son permutados mediante una **permutación** fija inicial (ver tabla 1), para obtener $m_0 = L_0 R_0$, donde L_0 es el bloque formado por los 32 primeros bits de m_0, y R_0 el bloque formado por los 32 últimos bits de m_0: $L_0 R_0 \leftarrow IP(m_1 m_2 \dots m_{64})$; es decir, $L_0 = (m_{58} m_{50} \dots m_8)$ y $R_0 = (m_{57} m_{49} \dots m_7)$.
2. Para $1 \leq i \leq 16$, se realizan las siguientes **operaciones**: $L_i = R_{i-1}$; $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$.

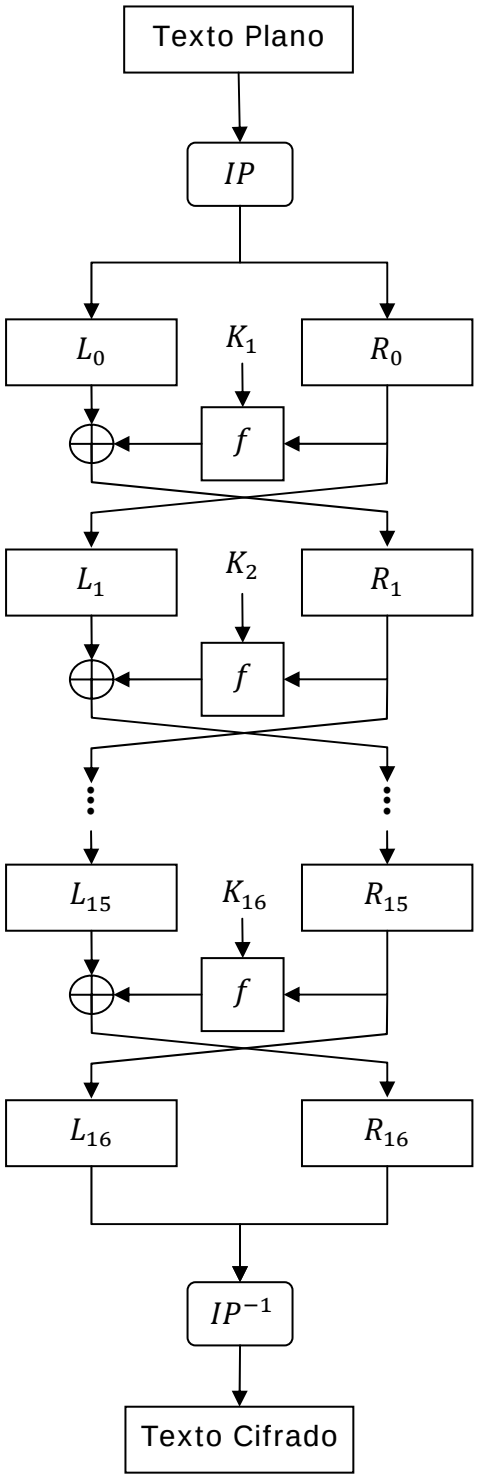

Figura 3: Algoritmo DES.

Donde K_i es una cadena de 48 bits obtenida de la clave K, y f es una función que describiremos más tarde.

3. Se intercambian derecha e izquierda para obtener $R_{16}L_{16} = (b_1 b_2 \dots b_{64})$ (bloques derecho e izquierdo de 32 bits, sobre los que se ha aplicado ya 16 veces el paso 2 del algoritmo). Se aplica entonces **la inversa de la permutación** inicial (ver de nuevo la tabla 1) para llegar finalmente al texto cifrado: $c = IP^{-1}(R_{16}L_{16}) = (b_{40} b_8 \dots b_{57} b_{25})$.

La operación de descifrado se realiza de forma análoga a la del cifrado, excepto que las claves $K_1, \dots, K_{16}$ se usan en orden inverso.

Permutación IP

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

Permutación IP inversa

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

Tabla 1: Permutación inicial DES (IP) y final inversa.

Veamos con más detalle cada uno de los tres pasos descritos.

PERMUTACIÓN.

La permutación inicial no tiene ningún valor criptográfico. Quizá se definió para otorgar mayor eficiencia al algoritmo en los chips de los años 70 (¿?). La permutación se realiza de acuerdo con la tabla 1.

Esto quiere decir que la cadena que tomaremos comienza con el bit 58, seguido del bit 50, y luego el bit 42,… La permutación inversa es la que se realiza como último paso del descifrado.

FUNCIÓN $f(R, K_i)$

En nuestro proceso interno de cifrado, nuestro alfabeto es $\{0,1\}$, la longitud de nuestros bloques es de 32 bits y el espacio de sus claves es $\{0,1\}^{48}$.

La función de cifrado es $f_K: \{0,1\}^{32} \to \{0,1\}^{32}$, que se calcula en una serie de pasos:

1. **Primero, el argumento $R \in \{0,1\}^{32}$ será expandido** mediante la función $E: \{0,1\}^{32} \rightarrow \{0,1\}^{48}$.
2. **A continuación se calcula $E(R_{i-1}) \oplus K_i$**, y el resultado se divide en **ocho bloques B_i, $1 \leq i \leq 8$, de longitud 6**, donde $B_i \in \{0,1\}^6$.
3. Luego se somete a cada bloque a una operación llamada S–boxes, de la siguiente forma: $S_i: \{0,1\}^6 \rightarrow \{0,1\}^4$, para $1 \leq i \leq 8$. **Cada B_i de seis bits obtiene una salida C_i de cuatro bits**. Pasamos, pues, de una cadena de 48 bits a una más corta de 32.
4. Finalmente se somete **a la cadena resultante de 32 bits a una permutación**.

Veamos cada uno de los pasos brevemente enunciados:

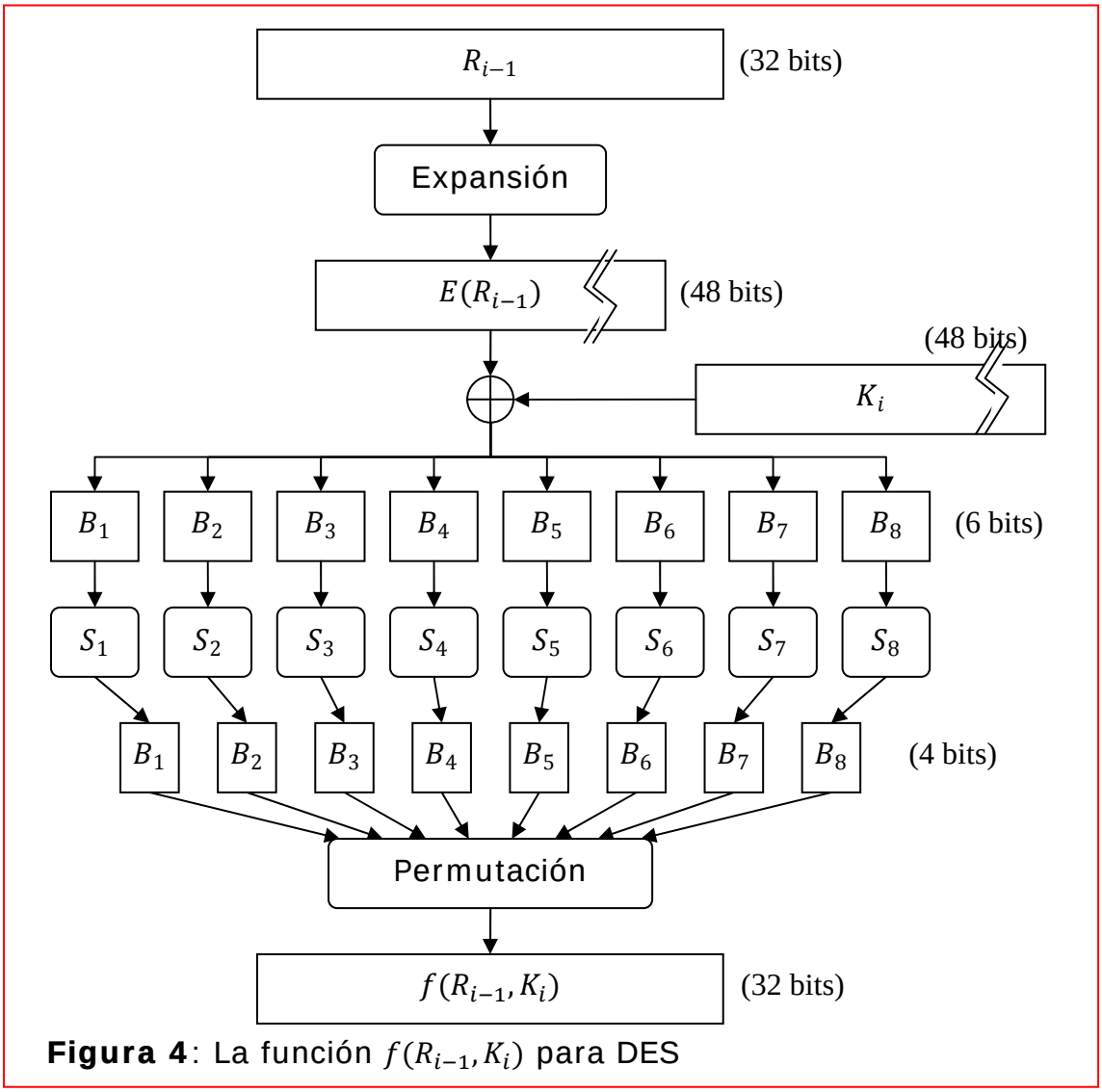

Figura 4: La función $f(R_{i-1}, K_i)$ para DES

1. Primero, $\boldsymbol{R}$ **se expande**, de acuerdo con lo indicado en la tabla 2.

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

Tabla 2: Función de expansión $E(R)$

   El primer bit de $E(R)$ es el 32 de R, etc. La función de expansión tiene una entrada de 32 bits y una salida de 48 bits. Quedan remarcados en negrita y recuadrados los bits que se replican en la función de expansión.

2. Luego **se calcula** $\boldsymbol{E(R) \oplus K_i}$ (más adelante veremos cómo se definen las distintas K_i: queda claro por ahora que son valores de 48 bits). El resultado de esta operación lo dividimos en 8 bloques de 6 bits cada uno: $E(R) \oplus K_i = B_1B_2B_3B_4B_5B_6B_7B_8$.
3. Tomamos ahora ocho **S–boxes** según se definen en la tabla 3. Cada B_i es la entrada de la correspondiente caja S_i. Si tomamos $B_i = b_1b_2b_3b_4b_5b_6$, la fila de la caja S_i viene indicado por b_1b_6, mientras que la columna viene indicada por $b_2b_3b_4b_5$.

   Por ejemplo, si $B_3 = 001001$, entonces tomaremos la fila 01 que es la segunda (la primera fila es la 00), y la columna 0100, que es la quinta (la primera columna es la 0000). El valor obtenido es el 3, que se obtendrá en codificación binaria con cuatro bits: $C_3 = 0011$: todas las salidas se codifican con 4 dígitos binarios. Obtenemos, por tanto, 4 bits de cada caja; en total 32 bits: $C_1C_2C_3C_4C_5C_6C_7C_8$.

   Las S–boxes constituyen el corazón de DES y son los que proveen de seguridad al criptosistema.
4. Los 32 bits de la cadena $C_1C_2C_3C_4C_5C_6C_7C_8$ **permutan** de acuerdo con la tabla 4. El resultado de esa permutación es la cadena de salida $\boldsymbol{f(R, K_i)}$**,** de 32 bits.

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

Tabla 4: Permutaciones de los bits de salida de las S–boxes.

S – box 1															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	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	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S – box 2															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	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	0	5	14	9
S – box 3															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S – box 4															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S – box 5															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S – box 6															
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S – box 7															
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	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	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S – box 8															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Tabla 3: S–Boxes para DES

LAS CLAVES K_1, ..., K_{16}.

Queda pendiente describir cómo se obtienen las dieciséis claves K_i. Como se recordará, la clave del criptosistema DES, K, es una clave de 64 bits, de los cuales, 8 son bits de paridad: se dispone de un total de 56 bits de clave útil. Los pasos a seguir para obtener las sucesivas K_i son los siguientes:

1. Descartar los bits de paridad y, con los bits que resten, realizar una

permutación según indica la tabla 5. Una vez permutada la clave, la tomamos como C_0D_0, (con C_0 y D_0 de 28 bits).

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

Tabla 5: **Permutación** inicial para la obtención de la claves K_i. Supone también una **selección** de bits: se eliminan los bits de paridad: bits $b_{8\cdot k}$, para $1 \leq k \leq 8$. Esta permutación se llama PC-1.

2. Para $1 \leq i \leq 16$ realizamos la operación: $C_i = LS_i(C_{i-1})$, y $D_i = LS_i(D_{i-1})$. LS_i es una función de rotación a izquierda en una o en dos posiciones, de acuerdo con la tabla 6.

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

Tabla 6: Desplazamiento de la clave (fila 2) en cada una de las 16 iteraciones (fila 1).

3. En cada iteración tomamos 48 de los 56 bits de la cadena C_iD_i de acuerdo con las indicaciones de la tabla 7. La salida de esta operación es ya K_i. De esta forma cada bit de la clave es usado aproximadamente 14 de las 16 iteraciones.

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

Tabla 7: Bits seleccionados de la cadena C_iD_i para la construcción de la clave K_i. Esta operación supone una selección y una permutación de los bits tomados. Esta permutación se llama PC-2.

5. SEGURIDAD DE DES.

Desde su invención, la seguridad que ofrece el algoritmo DES ha sido estudiada de forma intensiva. De hecho, DES ha sido el criptosistema estándar durante los 20 últimos años del siglo XX. Se han presentado algunas técnicas de criptoanálisis, pero el ataque más eficaz resulta ser el de la búsqueda exhaustiva de la clave en su espacio de claves. DES se demostró finalmente inseguro en julio de 1998, cuando la Electronic Frontier Foundation (EFF) anunció que había logrado romper un criptograma usando una maquina de diseño especial, llamada DES–Cracker, construida por menos de 250.000 dólares. El ataque se tomó menos de 3 días.

El sistema de ataque consiste en recorrer todas las posibles claves. Además de llegar así a encontrar la clave correcta dentro del amplio dominio de claves, es necesario que el algoritmo discierna cuándo la clave probada es realmente la clave buscada. Para ello, el algoritmo debe reconocer un texto plano correcto en un determinado idioma cada vez que realiza la operación de descifrado sobre el texto cifrado. Debe asumir diferentes idiomas válidos. Y debe asumir la posibilidad de que el texto haya sido comprimido antes de su cifrado, cosa que es bastante habitual. Por último, debe aceptar también que el mensaje plano use un alfabeto amplio, con signos de puntuación, o caracteres numéricos.

Ataque por fuerza bruta. Para cualquier tipo de cifrado, el método de ataque más simple es el ataque por fuerza bruta, probando una por una cada posible clave. La longitud de clave determina el número posible de claves, y por tanto la factibilidad del ataque. En el caso de DES, ya en sus comienzos se plantearon cuestiones sobre su longitud de clave, incluso antes de ser adoptado como estándar, y fue su reducido tamaño de clave, más que el criptoanálisis teórico, el que provocó la necesidad de reemplazarlo.

Académicamente, se adelantaron varias propuestas de una máquina para romper DES. En 1977, Diffie y Hellman propusieron una máquina con un coste estimado de 20 millones de dólares que podría encontrar una clave DES en un sólo día. Hacia 1993, Wiener propuso una máquina de búsqueda de claves con un coste de un millón de dólares que encontraría una clave en 7 horas. La vulnerabilidad de DES fue demostrada en la práctica en 1998 cuando la Electronic Frontier Foundation (EFF), un grupo dedicado a los derechos civiles en el ciberespacio, construyó una máquina a medida para romper DES, con un coste aproximado de 250.000 dólares. Su motivación era demostrar que se podía romper DES tanto en la teoría como en la práctica: "Hay mucha gente que no creerá una verdad

hasta que puedan verla con sus propios ojos. Mostrarles una máquina física que pueda romper DES en unos pocos días es la única manera de convencer a algunas personas de que realmente no pueden confiar su seguridad a DES." La máquina rompió una clave por fuerza bruta en una búsqueda que duró poco más de 2 días; Más o menos al mismo tiempo, un abogado del Departamento de Justicia de los Estados Unidos proclamaba que DES era irrompible.

Ataques más rápidos que la fuerza bruta. Existen tres ataques conocidos que pueden romper las dieciséis rondas completas de DES con menos complejidad que un ataque por fuerza bruta: el criptoanálisis diferencial (CD), el criptoanálisis lineal (CL) y el ataque de Davies. De todas maneras, éstos ataques son sólo teóricos y no es posible llevarlos a la práctica.

El criptoanálisis diferencial fue descubierto a finales de los 80 por Eli Biham y Adi Shamir, aunque era conocido anteriormente tanto por la NSA como por IBM y mantenido en secreto. Para romper las 16 rondas completas, el criptoanálisis diferencial requiere 2^{47} textos planos escogidos. DES fue diseñado para ser resistente al CD.

El criptoanálisis lineal fue descubierto por Mitsuru Matsui, y necesita 2^{43} textos planos escogidos (Matsui, 1993); el método fue implementado (Matsui, 1994), y fue el primer criptoanálisis experimental de DES que se dio a conocer. No hay evidencias de que DES fuese adaptado para ser resistente a este tipo de ataque. Una generalización del CL —el criptoanálisis lineal múltiple— se propuso en 1994 Kaliski and Robshaw), y fue mejorada por Biryukov y otros (2004); su análisis sugiere que se podrían utilizar múltiples aproximaciones lineales para reducir los requisitos de datos del ataque en al menos un factor de 4 (es decir, 2^{41} en lugar de 2^{43}). Una reducción similar en la complejidad de datos puede obtenerse con una variante del criptoanálisis lineal de textos planos escogidos (Knudsen y Mathiassen, 2000). Junod (2001) realizó varios experimentos para determinar la complejidad real del criptoanálisis lineal, y descubrió que era algo más rápido de lo predicho, requiriendo un tiempo equivalente a 2^{39}–2^{41} comprobaciones en DES.

El ataque mejorado de Davies: mientras que el análisis lineal y diferencial son técnicas generales y pueden aplicarse a multitud de esquemas diferentes, el ataque de Davies es una técnica especializada para DES. Propuesta por vez primera por Davies en los 80, y mejorada por Biham y Biryukov (1997). La forma

más potente del ataque requiere 2^{50} textos planos escogidos y tiene un 51% de probabilidad de éxito.

Existen también ataques pensados para versiones del algoritmo con menos rondas, es decir versiones de DES con menos de dieciséis rondas. Dichos análisis ofrecen una perspectiva sobre cuántas rondas son necesarias para conseguir seguridad, y cuánto «margen de seguridad» proporciona la versión completa.

Propiedades criptoanalíticas. DES presenta la propiedad complementaria, dado que $E_K(m) = c \Leftrightarrow E_{\bar{K}}(\bar{m}) = \bar{c}$, donde $\bar{x}$ es el complemento a bit de x. La propiedad complementaria implica que el factor de trabajo para un ataque por fuerza bruta se podría reducir en un factor de 2 (o de un único bit) asumiendo un ataque con texto plano escogido.

6. DES NO ES UN GRUPO.

Un posible modo de incrementar de modo efectivo el tamaño de la clave de DES sería realizar una doble operación de cifrado sobre el texto plano. Tomando K_1 y K_2 y el texto plano m, podríamos obtener el criptograma a través de la doble transformación $c = E_{K2}(E_{K1}(m))$. Para descifrar, bastaría hacer ahora $m = D_{K1}(D_{K2}(c))$. Y ahora para el atacante la tarea es enormemente mayor: encontrar ambas claves que juntamente llegan a nuestra transformación.

¿Aumentaría eso la seguridad de nuestro criptosistema?

En realidad, para muchos criptosistemas, aplicar dos veces el proceso de cifrado con dos claves diferentes es equivalente a realizar el proceso de cifrado con una tercera clave diferente. ¿Es así para DES?: Dadas las claves K_1 y K_2, ¿existe una clave K_3 tal que $E_{K3} = E_{K2}E_{K1}$? Esta cuestión se plantea frecuentemente de la siguiente forma: ¿Es DES un grupo?, o también ¿Es DES una transformación cerrada bajo la operación de composición?

La respuesta es que NO. Afortunadamente no es así, en el caso de DES: la composición de dos cifrados DES no es equivalente a un cifrado DES con una tercera clave en principio distinta a las otras dos. Una demostración de esta afirmación se puede encontrar en [3], en § 4.5., en un apartado titulado “DES is not a group”.

Cabe pensar, por tanto, que efectivamente, cifrar DES dos veces es equivalente a trabajar con un algoritmo de longitud de clave 112.

Pero no es así, por otro motivo. Existe una forma de ataque que anula esa posibilidad. Para levarla a cabo basta con tener la herramienta que rompe DES y una enorme capacidad de memoria. El procedimiento para atacar esta cifra es el siguiente:

1. Supóngase que el atacante ha interceptado el mensaje plano a transmitir, m, y el criptograma $c = E_{K2}(E_{K1}(m))$. Quiere ahora hallar K_1 y K_2.
2. Computa y almacena en memoria todos los valores $E_k(m)$ para todos los posibles valores $k \in \mathcal{K}$.
3. Computa y almacena en memoria todos los valores $D_k(c)$ para todos los posibles valores $k \in \mathcal{K}$.
4. Compara ambas listas en busca de encontrar cadenas de texto iguales. Entre las claves que han conducido a estas cadenas coincidentes están las dos claves buscadas.

Por tanto, el atacante ha de probar todas las claves para el paso n. 2, y ha de probar todas las claves para el paso n. 3. Por tanto, si N es el número total de claves posibles, al realizar el proceso de cifrado dos veces deberá realizar una búsqueda de $2 \cdot N$ claves, y no de N^2 como habíamos esperado al principio.

A este ataque se le ha dado el nombre “meet–in–the–middle attack”, o ataque de “encuentro–en–la–mitad”.

7. TRIPLE DES

Doble–DES queda por tanto descartado como criptosistema seguro una vez se ha logrado comprometer el criptosistema DES. ¿Qué ocurre en el caso de Triple–DES? En ese caso, el algoritmo de ataque meet–in–the–middle no funciona correctamente.

Existen dos formas habituales de aplicar Triple–DES:

1. Con tres claves K_1, K_2 y K_3, aplicamos la transformación $c = E_{K1}(E_{K2}(E_{K3}(m)))$
2. Con dos claves K_1 y K_2, aplicamos la transformación $c = E_{K1}(D_{K2}(E_{K1}(m)))$. En este caso, evidentemente, cuando $K_1 = K_2$ el algoritmo Triple–DES se reduce al DES tradicional.

Ambas formas dan gozan de una seguridad equivalente a la de un criptosistema de longitud de clave igual a 112 bits. Ambas formas son resistentes, como ya se ha dicho, al ataque meet–in–the–middle. Existen, sin embargo, otras formas de

ataque definidas, de factura similar a la presentada en este algoritmo, pero requieren una cantidad de memoria tal que la hacen impracticable. Queda indicada, por tanto, la posible debilidad, que no supone hoy por hoy una amenaza viable.

Rivest tiene presentada otra vía para reforzar DES. Tomando tres claves K_1, K_2 y K_3, realizamos la transformación $c = K_3 \oplus E_{K2}(K_1 \oplus m)$: Modificar el texto plano con la operación XOR con la clave K_1; al resultado aplicar la transformación DES con la clave K_2, y operar el resultado obtenido de nuevo con el operador XOR y ahora la clave K_3. Este método se conoce como DESX y se ha mostrado también suficientemente seguro.

REFERENCIAS

[1] "Handbook of Applied Cryptography". A. Menezes, P. van Oorschot, and S. Vanstone. CRC Press, Inc. 1997.

[2] "Cryptography and Network Security. Principles and practices". William Stallings. Prentice Hall. Pearson Education. Third edition. 2003.

[3] "Introduction of Cryptography with coding theory". Wade Trappe and Lawrence C. Washington. Prentice Hall, 2002.

[4] "Introduction to Cryptography". Johannes A. Buchmann. Springer Verlag, 2004. Second Edition.

[5] "Data encryption Standard (DES)", FIPS PUB 46–3. 1999 October 25. http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf.

[6] Wikipedia. http://es.wikipedia.org/wiki/Data_Encryption_Standard.