Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

Criptografía - Data Encryption Standar (DES) (25117 lectures)

Per **Eduard Llull**, ***Daneel*** ()

Creado el 01/02/2003 18:02 modificado el 01/02/2003 18:02

Con este artículo espero empezar una serie dedicada a criptografía, ya que es un tema que suele despertar bastante curiosidad y no es algo de lo que se suelan publicar muchos artículos. Pensando en aquellos que no tengan ningún tipo de conocimiento sobre el tema intentaré no dejarme ningún detalle para que la explicación sea lo más completa posible.

1. Introducción.

Empecemos introduciendo un poco de terminología:

- Texto en claro o *plain text*: es la información original, el mensaje, que se quiere cifrar.
- Texto cifrado o criptograma: es la información resultante una vez se ha cifrado el mensaje.

A continuación clasificaremos los distintos típos de mecanismos de cifrado, indicando sus diferencias, pros y contras. Por una parte podemos dividir los sistemas de cifrado según operen sobre bloques o sobre streams (flujos de bits). Los primeros dividen la información a cifrar en bloques de un determinado tamaño y aplican una serie de operaciones sobre ese bloque para producir el criptograma. Los segundos, cifran la información bit a bit.

Por otra parte, podemos dividir los sistemas de cifrado en simétricos y asimétricos. En los primeros, también conocidos como sistemas de clave secreta, se utiliza la misma clave tanto para cifrar como para descifrar, por lo tanto, las distintas partes involucradas en la comunicación de los datos cifrados (emisor y receptor) deben compartir el conocimiento de esta clave. Los sistemás asimétricos o sistemas de clave pública utilizan dos claves, una para cifrar y la otra para descifrar, de tal manera que el criptograma producido por una de ellas sólo puede ser descifrado por la otra. Esta división de dos claves permite la existencia de lo que se ha venido a conocer como firmas digitales, pero esto lo dejamos para un futuro artículo. Centremonos en los sistemas de cifrado simétricos, y más concretamente en los de bloque.

---

2. Esquema de los sistemas de cifrado de bloques.

El esquema de funcionamiento general es bastante simple, se divide la información a cifrar en bloques de un mismo tamaño y a cada uno de ellos se le aplican una serie de transformaciones para producir el correspondiente bloque de texto cifrado. Esquemáticamente:

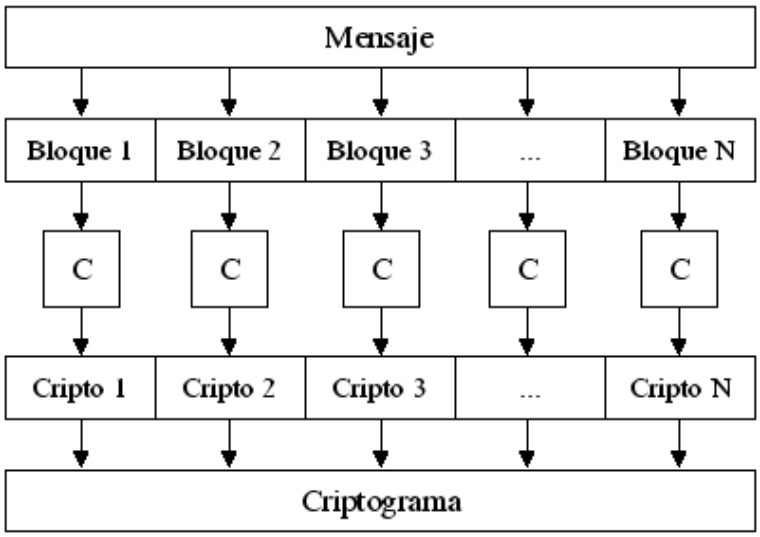

Ilustración1 - Esquema de cifrado por bloques

Dentro de este tipo de sistemas criptográficos, el más conocido es el DES o Data Encryption Standar. Este sistemas fue desarrollado a principio de los años '70 por un grupo de trabajo de IBM. En 1981 la ANSI aprovó el DES como estándar, el X3.92. Por su parte, la ISO hizo lo mismo en 1987 dandole el nombre de DEA-1. Las principales características de este sistema de cifrado es que utiliza operaciones lógicas simples (transposiciones, desplacamientos y XOR's) sobre grupos reducidos de bits, lo que permite una fácil y eficiente implementación del algoritmo en hardware.

3. Data Encryption Standar.

Como ya hemos dicho, el DES es un sistema de cifrado de bloques. En este caso, el algoritmo toma la información en bloques de 64 bits produciento un bloque de texto cifrado también de 64 bits. Las claves utilizadas por este sistema son de 56 bits, aunque se suelen distribuir en forma de un número de 64 bits, donde cada octavo bit (el lsb o less-significant bit) de cada uno de los ocho bytes de la calve es un bit de paridad.

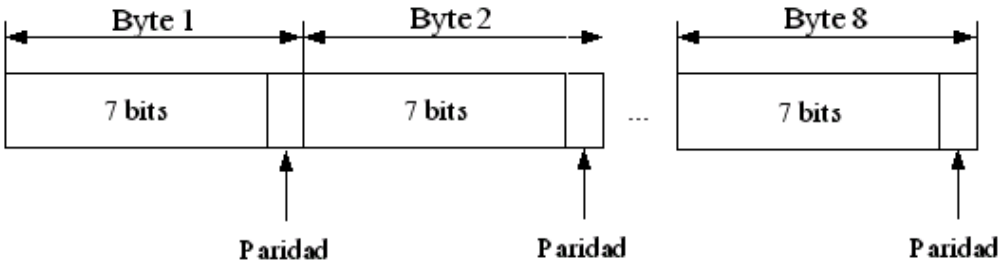

Ilustración 2 - Esquema clave DES

3.1. Algoritmo de cifrado.

Centremonos ahora en el algoritmo de cifrado.

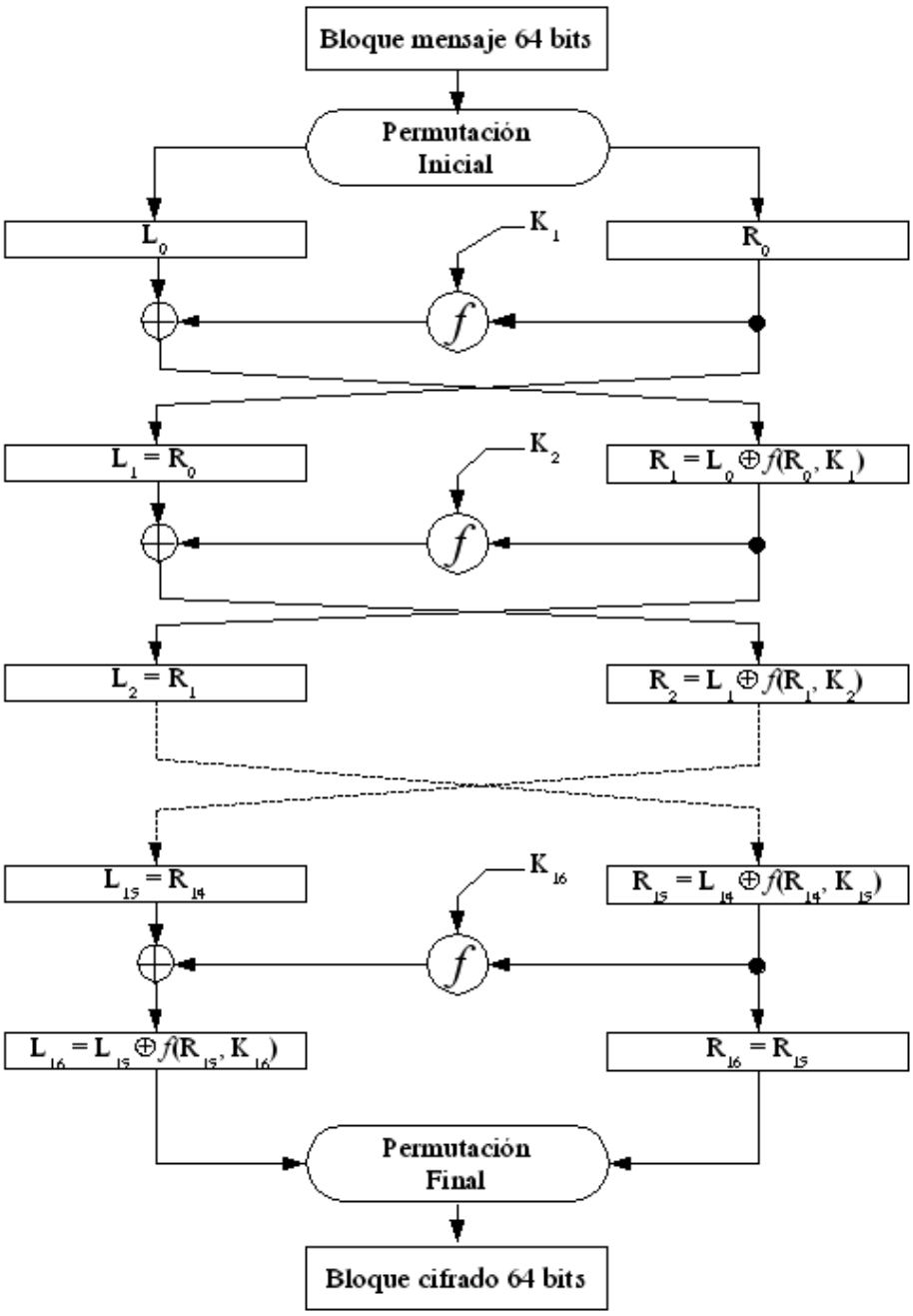

Ilustración 3 - Esquema del DES

Lo primero que se realiza es una permutación de los 64 bits del bloque de entrada. Realmente, esta permutación no añade seguridad alguna y su principal función es la de facilitar la carga de los bits de información en bloques de 8 bits a un chip especializado en DES (debemos recordar que el desarrollo del DES es anterior a los procesadores de 16 o 32 bits). Al final del algoritmo hay otra permutación que es la inversa de esta, lo que vuelve a dejar a los bits en su posición inicial.

58	50	42	34	26	18	10	02	60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06	64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01	59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05	63	55	47	39	31	23	15	07

Tabla 1 - Permutación inicial

Esta tabla debe leerse de izquiera a derecha y de arriba a bajo. Esto quiere decir, que el primer bit a la salida de la permutación es el que en la entrada ocupa la posición 58; el segundo, el que ocupaba la 50; y así sucesivamente.

Despues de esta permutación inicial, los 64 bits resultantes se dividen en dos partes de 32 bits cada una. Lo que viene a continuación se repite diecisesis veces:

1. La mitat de la derecha (R_i) se introduce en una funcion (*f*) donde es combinada con la clave.
2. Se realiza una XOR con el resultado de la función y con la mitad de la izquierda (L_i).
3. La parte de la derecha de esta ronda (R_i) pasará a ser la parte izquierda de la siguiente (L_{i+1}) y el resultado de la XOR anterior pasará a ser la parte de la derecha de la ronda siguiente (R_{i+1}).

$$L_{i+1} = R_i$$
$$R_{i+1} = L_i \text{ XOR } f(R_i, K)$$

La única excepción a este proceso está en la última ronda, en la que el entrecruce de las dos partes no se produce.

El esquema de esta función *f* es el siguiente:

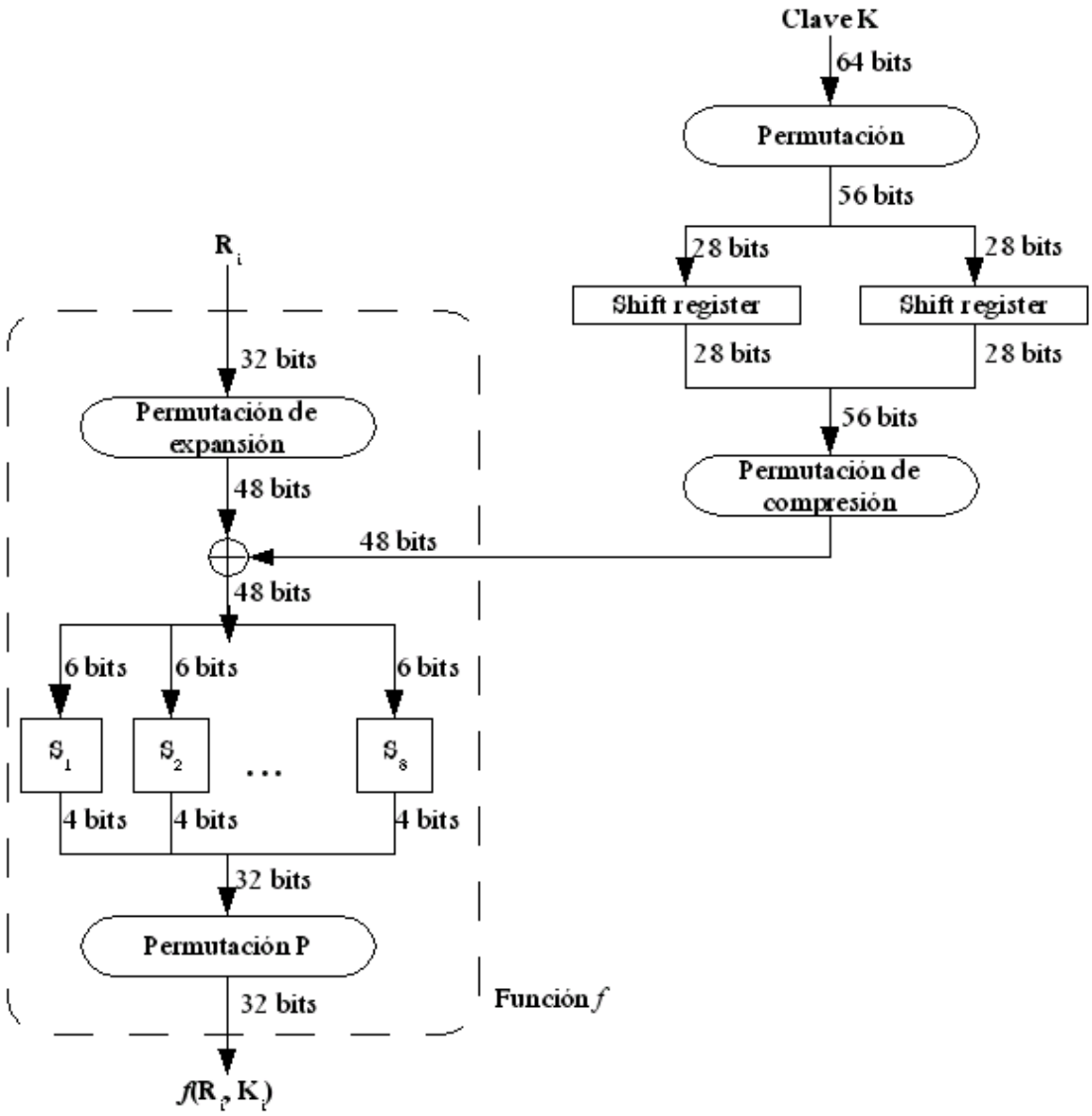

Ilustración 4 - Esquema de la función f

En cada ronda, se genera una subclave K_i distinta de la siguiente manera: como ya habiamos dicho, de los 64 bit de la clave, se descartan los 8 bits de paridad quedando una clave de 56 bits. Esto se realiza mediante la siguiente

permutación:

57	49	41	33	25	17	09	01	58	50	42	34	26	18
10	02	59	51	43	35	27	19	11	03	60	52	44	36
63	55	47	39	31	23	15	07	62	54	46	38	30	22
14	06	61	53	45	37	29	21	13	05	28	20	12	04

Tabla 2 - Permitación de la calve

Como se puede observar, no aparecen los bits 8, 16, 24, 32, 40, 48, 56 ni 64, que son los que hacen la función de bits de paridad.

La clave resultante se divide en dos mitades de 28 bits que alimentan unos registros que rotan hacia la izquierda cada una de las mitades un número de bits que viene determinado por la ronda en la que nos encontramos (recordar que en el DES existen 16 rondas)

Ronda	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Desplazamiento	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Tabla 3 - Desplazamientos según la ronda para el cifrado

De los 56 bits resultantes se seleccionan 48 modificando su orden. Esta operación es denominada permutación de compresión.

14	17	11	24	01	05	03	28	15	06	21	10
23	19	12	04	26	08	16	07	27	20	13	02
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 4 - Permutación de compresión

En la salida de la permutación de compresión tendríamos la subclave K_i de 48 bits correspondiente a la ronda i-ésima del algoritmo. Una vez ya hemos explicado como se obtienen las subclaves de las rondas, centremonos en las modificaciones que sufre la parte de la derecha R_i dentro de la función. Primero, mediante la permutación de expansión, se expanden los 32 bits de R_i a 48 bits (repitiendo alguno de los bits) cambiando, además, su posición. Esto se hace para que este operando tenga el mismo tamaño que la subclave de la ronda, y además hace que los bits del criptograma dependan todavía con más fuerza de los bits de la entrada. Esto hecho se conoce como el efecto avalancha: un pequeño cambio en el mensaje de entrada provoca grandes modificaciones en la sequencia de bits de salida.

32	01	02	03	04	05	04	05	06	07	08	09
08	09	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	01

Tabla 5 - Permutación de expansión

Los 48 bits resultantes de la XOR entre la parte derecha R_i expandida y la subclave K_i se introducen de seis en seis en ocho bloques que son conocidos como S-Box, de tal manera que el primer bloque de 6 bits se introduce en la S-Box 1, el segundo en la S-Box 2, y así sucesivamente. Como cada S-Box toma 6 bits como entrada y produce 4 bits como salida, al final obtenemos 32 bits (8 S-Box · 4 bits/S-Box = 32 bits)

Cada S-Box es una tabla de cuatro filas y dieciseis columnas, donde cada posición de la tabla es un número de 4 bits. De los 6 bits de la entrada b1 b2 b3 b4 b5 b6, los bits b1 y b6 indican la fila, mientras que los otros cuatro indican la

columna. Es decir, si la entrada de una S-Box es 35, esto en binario es 100011, lo que significa que hemos de tomar el valor de la fila 3 (11) y columna 1 (0001). El contenido de cada una de las S-Box es distinta del de las demás.

Contenido de las S-Box

14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
00	15	07	04	14	02	13	01	10	06	12	11	09	05	03	08
04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

Tabla 6 - S-Box 1

15	01	08	14	06	11	03	04	09	07	02	13	12	00	05	10
03	13	04	07	15	02	08	14	12	00	01	10	06	09	11	05
00	14	07	11	10	04	13	01	04	08	12	06	09	03	02	15
13	08	10	01	03	15	04	02	11	06	07	12	00	05	14	09

Tabla 7 - S-Box 2

10	00	09	14	06	03	15	05	01	13	12	07	11	04	02	08
13	07	00	09	03	04	06	10	02	08	05	14	12	11	15	01
13	06	04	09	08	15	03	00	11	01	02	12	05	10	14	07
01	10	13	00	06	09	08	07	04	15	14	03	11	05	02	12

Tabla 8 - S-Box 3

07	13	14	03	00	06	09	10	01	02	08	05	11	12	04	15
13	08	11	05	06	15	00	03	04	07	02	12	01	10	14	09
10	06	09	00	12	11	07	13	15	01	03	14	05	02	08	04
03	15	00	06	10	01	13	08	09	04	05	11	12	07	02	14

Tabla 9 - S-Box 4

02	12	04	01	07	10	11	06	08	05	03	15	13	00	14	09
14	11	02	12	04	07	13	01	05	00	15	10	03	09	08	06
04	02	01	11	10	13	07	08	15	09	12	05	06	03	00	14
11	08	12	07	01	14	02	13	06	15	00	09	10	04	05	03

Tabla 10 - S-Box 5

12	01	10	15	09	02	06	08	00	13	03	04	14	07	05	11
10	15	04	02	07	12	09	05	06	01	13	14	00	11	03	08
09	14	15	05	02	08	12	03	07	00	04	10	01	13	11	06
04	03	02	12	09	05	15	10	11	14	01	07	06	00	08	13

Tabla 11 - S-Box 6

04	11	02	14	15	00	08	13	03	12	09	07	05	10	06	01
13	00	11	07	04	09	01	10	14	03	05	12	02	15	08	06
01	04	11	13	12	03	07	14	10	15	06	08	00	05	09	02
06	11	13	08	01	04	10	07	09	05	00	15	14	02	03	12

Tabla 12 - S-Box 7

13	02	08	04	06	15	11	01	10	09	03	14	05	00	12	07
01	15	13	08	10	03	07	04	12	05	06	11	00	14	09	02
07	11	04	01	09	12	14	02	00	06	10	13	15	03	05	08
02	01	14	07	04	10	08	13	15	12	09	00	03	05	06	11

Tabla 13 - S-Box 8

Los 32 bits resultantes de la anterior operación entran en la permutación P, en la que los bits son permutados otra vez pero esta vez no se eliminan ni añaden bits como se habia hecho en las permutaciónes de compresión y expansión.

16	07	20	21	29	12	28	17	01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09	19	13	30	06	22	11	04	25

Tabla 14 - Permutación P

Para finalizar la ronda, la salida de la permutación P alimenta una XOR en la que el otro operando es la parte izquiera L_i. Si no estamos en la ronda 16, el resultado de la XOR anterior se convierte en la parte derecha de la siguiente ronda R_{i+1}, y la parte derecha de la ronda en la que estamos R_i pasará a ser la parte izquiera de la siguiente L_{i+1}. Y así sucesivamente hasta completar las 16 rondas.

Finalmente, la concatenación de L16 y R16 se introduce en una permutación final, que ya habiamos comentado al inicio de la explicación del algoritmo. Esta permutación es la inversa de la permutación inicial quedando de la siguiente manera:

40	08	48	16	56	24	64	32	39	07	47	15	55	23	63	31
38	06	46	14	54	22	62	30	37	05	45	13	53	21	61	29
36	04	44	12	52	20	60	28	35	03	43	11	51	19	59	27
34	02	42	10	50	18	58	26	33	01	41	09	49	17	57	25

Tabla 15 - Permutación final

Se podría haber hecho que en la ronda final se intercambiaran las partes derecha e izquierda, y adaptar la permutación final para que tuviera en cuenta este intercambio de más. Pero no se hizo porque de esta manera se puede utilizar el

mismo algoritmo tanto para el cifrado como para el descifrado.

3.2. Algoritmo de descifrado.

El algoritmo de descifrado es el mismo que el de cifrado, con la única matización de que las subclaves deben utilizarse en el orden inverso. Es decir, si en el algoritmo de cifrado las subclaves de cada ronda eran K1, K2, ..., K16, en el descifrado las subclaves son K16, K15, ..., K1, entendiendo que la primera subclave (K1 en el cifrado y K16 en el descifrado) es la que se utiliza en la primera ronda; la segunda subclave (K2 en el cifrado y K15 en el descifrado), en la segunda ronda; etc.

Además, el algoritmo mediante el cual se generan las subclaves es circular. Así que para generar las subclaves necesarias para el descifrado, los registros de rotación en lugar de desplazar hacia la izquierda, lo deben hacer hacia la derecha y el número de desplazamientos viene dado por la siguiente tabla:

Ronda	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Desplazamiento	0	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Tabla 16 - Desplazamientos según la ronda para el descifrado

3.3. Consideraciones sobre las claves.

Debido al modo en el que se generan las subclaves en las diferentes rondas, existen una serie de claves que no son aptas para ser utilizadas. Las primeras de estas claves son conocidas como claves débiles (o *weak keys*). La debilidad de estas claves es que todos los bits de cada una de las dos mitades que entran en los registros de rotación son todo unos o todo ceros. Entonces, por mucho que rotemos estas palabras de bits, siempre obtendremos la misma subclave.

Claves Débiles	**Subclave**
101010101010101	0000000 0000000
1F1F1F1F0E0E0E0E	0000000 FFFFFFF
E0E0E0E0F1F1F1F1	FFFFFFF 0000000
FEFEFEFEFEFEFEFE	FFFFFFF FFFFFFF

También es posible encontrar pares de claves de tal manera que el criptograma generado por una de las claves del par puede ser descifrado por la otra clave. O lo que es lo mismo, las dos claves del par generan el mismo criptograma a partir del mismo texto en claro. Esto se debe a que esas claves, en lugar de generar 16 subclaves distintas, sólo generan dos subclaves, y cada una es usada ocho veces. A estas claves se le llaman claves semidébiles (*semiweak keys*) y existen 12 claves que cumplen esta condición.

Para finalizar, existen otras claves que producen únicamente cuatro subclaves distintas, que son usadas cuatro veces cada una. Estas son conocidas como claves posiblemente débiles (*possibly weak keys*). De este tipo de claves, existen 48.

Pero a pesar de la existencia de estas 64 claves débiles (4 + 12 + 48), hay que decir que son una minuscula porcion de las 2^{56} = 72.057.594.037.927.936 claves que puede utilizar el DES.

4. Conclusiones

Como se ha podido ver, las operaciones utilizadas en el sistema de cifrado DES son muy simples, permutaciones, XORs e indexado de tablas; siendo estas tres operaciones muy simples de realizar utilizando hardware: las permutaciones se pueden conseguir simplemente cambiando el orden de las pistas, para las XOR disponemos de circuitos integrados que ejecutan esa función, y el indexado de tablas se puede conseguir mediante ROM's. Este criterio, el de simplicidad de implementación, es uno de los que se tuvieron en cuenta durante el desarrollo del

algoritmo. Y el hecho de que se pudiera implementar a nivel de hardware permitió que se puediera usar en aplicaciones en las que la latencia es un factor importante. Otro de los criterios es que la robustez del sistema no debía depender de que el algoritmo fuera secreto, y en este caso toda la seguridad del sistema reside en la clave.

Espero que este artículo os haya parecido interesante. Ya estoy preparando el siguiente que tratará los sistemas de cifrado asimétricos.

Articulo basado en el capítulo 12 de *"Applied Criptography"* de **Bruce Schneier**

E-mail del autor: daneel _ARROBA_ bulma.net