Práctica 1

Simplified Data Encryption Standard (S-DES)

1. Objetivo de la práctica

El objetivo de esta práctica es familiarizarse con la operativa de los algoritmos de criptografía simétrica, en particular, con el algoritmo Data Encription Standard (DES).

- Analizar la operativa de un algoritmo criptográfico de bloque.
- Comprender los conceptos de clave y algoritmo, cifrado y descifrado, etc.
- Evaluar las operaciones básicas de permutación y sustitución empleadas en los algoritmos criptográficos.
- Implementación y validación a nivel binario de un algoritmo criptográfico simple.

2. Simplified DES

En el año 1975 se aprobó y publicó el algoritmo de cifrado simétrico Data Encryption Standard (DES) [2], que se emplea, entre otros, para proteger conversaciones telefónicas, para el cálculo de Personal Identification Numbers (PIN) o en transacciones de finacieras (i.e. micropagos).

DES se presenta como un algoritmo de criptografía moderna que permite el cifrado de un texto plano, y la operación recíproca, mediante el empleo de una clave secreta. Aunque su uso está muy extendido, los numerosos parámetros y funciones que incluye lo hacen complicado de desarrollar en un entorno educacional.

Por este motivo, el profesor Edward Schaefer de la Universidad de Santa Clara ideó el Simplified DES (S-DES) [1] en 1996 como una versión reducida del DES. Concebido no como un algoritmo criptográfico seguro, sino como un enfoque meramente educativo, S-DES se basa en una estructura y propiedades similares a DES, pero con un conjunto de parámetros y operaciones muy inferior.

Para la versión simplificada del DES, dos usuarios cualesquiera comparten una clave secreta de 10 bits. El algoritmo opera segmentando el mensaje en bloques de 8 bits, cifrando y descifrando cada bloque de forma independiente, de forma que bloques idénticos generan el mismo cifrado dada una clave.

El algoritmo de cifrado, resumido en la figura 1, consiste en la aplicación de 5 funciones: una permutación inicial (IP); una función compleja, denominada f_K, que incluye permutaciones y sustituciones y cuyo resultado depende de una clave; una permutación simple que intercambia las dos mitades de los datos (SW); la función f_K nuevamente; y finalmente una función de permutación inversa de la permutación inicial.

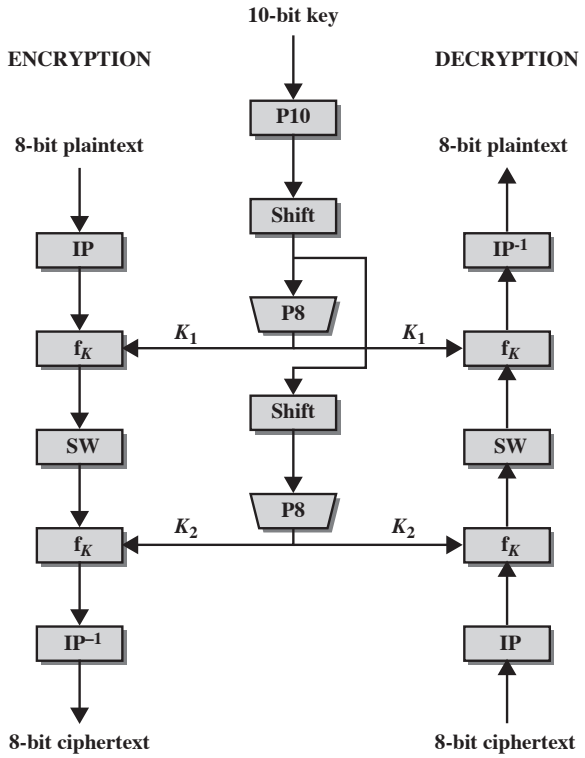

Figura 1: Esquema del Simplified DES

La función f_K se define como una función con 2 parámetros de entrada: los datos a cifrar y una clave de 8 bits. El algoritmo S-DES podría haberse diseñado para emplear una clave de 16 bits, de forma que una subclave de 8 bits se empleara en cada llamada a la función f_K. Del mismo modo podría haberse considerado una clave de 8 bits que se empleara dos veces. Sin embargo, en su diseño, con el objetivo de mantener una mayor similitud con la operativa del DES, se estableció el uso de una clave de 10 bits de la que se derivan 2 subclaves de 8 bits.

De forma resumida se pueden expresar las operaciones de cifrado y descifrado como:

$$ciphertext = IP^{-1}\left(f_{K_2}\left(SW\left(f_{K_1}\left(IP\left(plaintext\right)\right)\right)\right)\right)$$

$$plaintext = IP^{-1}\left(f_{K_1}\left(SW\left(f_{K_2}\left(IP\left(ciphertext\right)\right)\right)\right)\right)$$

donde

$$K_1 = P8\left(Shift\left(P10\left(key\right)\right)\right)$$

$$K_2 = P8\left(Shift\left(Shift\left(P10\left(key\right)\right)\right)\right)$$

2.1. Generación de claves

S-DES emplea una clave de 10 bits compartida entre los dos extremos de la comunicación. Esta clave es la semilla de 2 subclaves de 8 bits que se emplean en diferentes etapas del proceso de cifrado y descifrado.

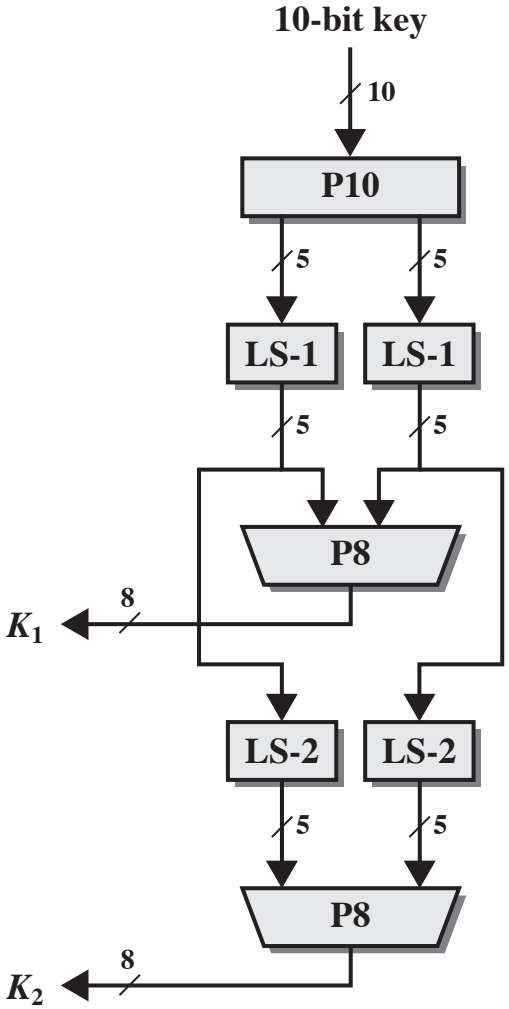

Figura 2: Detalle del proceso de generación de claves

En la figura 2 se describe el proceso de generación de estas subclaves. Considérese que la clave de 10 bits se define como $(k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9, k_{10})$.

El primer paso supone la permutación P10 de los bits de la clave según:

$\text{P10}(k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9, k10) = (k_3, k_5, k_2, k_7, k_4, k_{10}, k_1, k_9, k_8, k_6)$

De forma más genérica se puede expresar la transformación en formato tabla, según se indica en la tabla 1.

Tabla 1: Permutación P10

P10									
src	3	5	2	7	4	10	1	9	8	6
dst	1	2	3	4	5	6	7	8	9	10

Tras la aplicación de la permutación P10, se realiza un desplazamiento circular a la izquierda o rotación binaria (LS-1) a los primeros 5 bits y a los 5 bits finales, de forma independiente.

A continuación, se aplica la permutación P8, según la tabla 2, que selecciona 8 de los 10 bits de la clave. Resultado de la misma se obtiene la primera clave de 8 bits, K_1.

Tabla 2: Permutación P8

P8								
src	6	3	7	4	8	5	10	9
dst	1	2	3	4	5	6	7	8

Para obtener K_2, se realiza un desplazamiento circular doble a la izquierda (LS-2) al resultado obtenido anteriormente de hacer la permutación P10. Tras ello se lleva a cabo la permutación P8, generando K_2.

2.2. Cifrado

El procedimiento de cifrado supone la aplicación secuencial de 5 funciones sobre un bloque de datos de 8 bits. En la figura 3 se muestra el detalle del procedimiento que seguidamente se describe.

2.2.1. Permutaciones inicial y final

En la primera etapa, el bloque de 8 bits de información se permuta empleando la función IP, según la tabla 3. Esta permutación mantiene los 8 bits de información, pero los entremezcla.

Tabla 3: Permutación IP

IP								
src	2	6	3	1	4	8	5	7
dst	1	2	3	4	5	6	7	8

Al final del algoritmo, se realiza la permutación inversa IP^{-1}, tal como se muestra en la tabla 4.

Tabla 4: Permutación IP^{-1}

IP^{-1}								
src	4	1	3	5	7	2	8	6
dst	1	2	3	4	5	6	7	8

Se puede comprobar fácilmente esta segunda permutación es la inversa de la inicial, por lo que se puede afirmar que $IP^{-1}(IP(X)) = X$.

2.2.2. Función f_K

El procedimiento más complejo del algoritmo S-DES es la función f_K, ya que supone la combinación de funciones de permutación y sustitución.

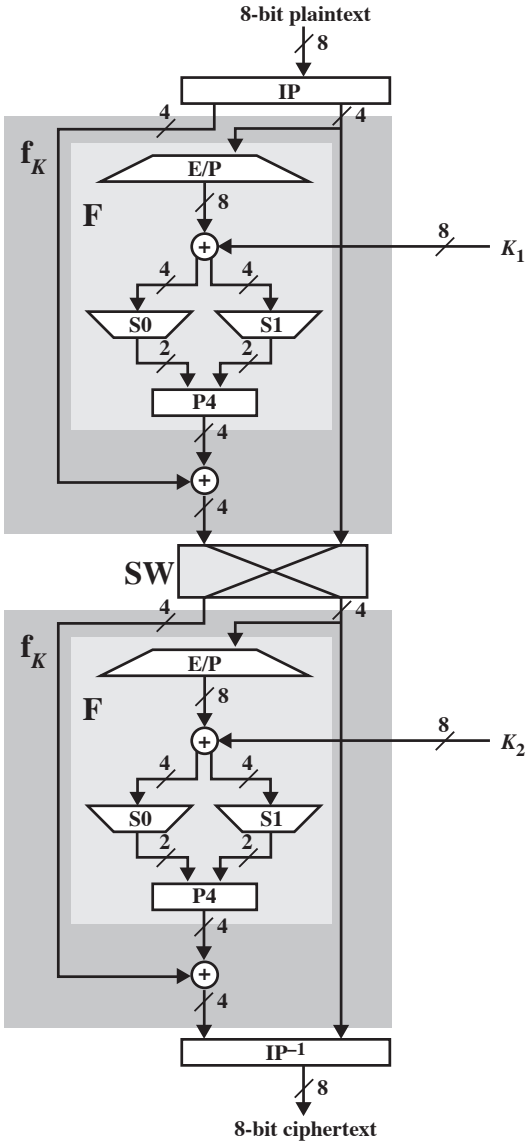

Figura 3: Detalle del proceso de cifrado

Siendo L y R los 4 bits más y menos significativos respectivamente de la entrada de datos, y F una función de correspondencia de secuencias de 4 bits en otra secuencia de 4 bits, se define f_K como:

$$f_K(L, R) = (L \oplus F(R, SK), R)$$

donde SK es una subclave y $\oplus$ es la operación OR exclusiva bit a bit (XOR).

La función F, que toma como argumentos de entrada una secuencia de 4 bits de datos $(n_1 n_2 n_3 n_4)$ y una subclave de 8 bits, consiste en la siguiente secuencia de operaciones:

1. Expansión/permutación de los bits de entrada según la tabla 5.

Tabla 5: Expansión / Permutación

E/P							
src	4	1	2	3	2	3	4	1
dst	1	2	3	4	5	6	7	8

2. Operación XOR del resultado de la operación anterior con la subclave de 8 bits.

3. Aplicación de la transformación mediante las cajas S, cuyos valores se indican en la tabla 6.

Tabla 6: Cajas S

S0

Fila \ Columna	0	1	2	3
0	1	0	3	2
1	3	2	1	0
2	0	2	1	3
3	3	1	3	2

S1

Fila \ Columna	0	1	2	3
0	0	1	2	3
1	2	0	1	3
2	3	0	1	0
3	2	1	0	3

Para la operativa con las cajas S, se divide el resultado del paso anterior en dos cuádruplas de bits. Para la primera se aplica la transformación según S0, y para la segunda según S1. El primer y el cuarto bit de cada cuádrupla conforma el valor de la fila, y el segundo y el tercer bit el valor de la columna. Los valores de fila y columna se consideran en base 2.

4. Los 4 bits resultado de las transformaciones mediante las cajas S se concatenan (S0 S1) y se les aplica la permutación P4, reflejada en la tabla 7.

Tabla 7: Permutación P4

P4				
src	2	4	3	1
dst	1	2	3	4

La salida de la función F es el resultado tras realizar la permutación P4.

2.2.3. Conmutación intermedia

La función f_K unicamente altera los 4 bits más significativos de los datos de entrada. En este sentido, la función de conmutación intermedia (SW, SWitch function) intercambia los 4 bits más y menos significativos, de forma que la segunda aplicación de la función f_K se realice sobre una secuencia diferente de bits.

2.3. Ejemplo

A continuación, se detalla un ejemplo de la operativa del algoritmo S-DES para los siguientes parámetros:

- Clave (K): 1010000010
- Datos (P): 01110010

Los pasos para la generación de las dos claves de 8 bits, K1 y K2, son:

1. Aplicar a K la permutación P10: 1000001100
2. Rotar a la izquierda 1 posición de cada mitad de la clave: 00001 11000
3. Aplicar la permutación P8 para obtener K1: 10100100
4. Rotar a la izquierda 1 posición de cada mitad de la clave: 00001 11000
5. Aplicar la permutación P8 para obtener K2: 01000011

K1 y K2 se usan como entradas en las dos etapas de cifrado y descifrado.

Partiendo de los datos P y conocidas las subclaves, se procede a su cifrado:

1. Aplicar la permutación inicial IP a P : 10101001
2. Separar en dos mitades, L and R: L=1010, R=1001
3. Aplicar la expansión y permutación E/P sobre R: 11000011
4. Realizar la XOR con la clave K1: 10100100 XOR 11000011 = 01100111
5. Aplicar las cajas S
   a) Para S0, la entrada es 0110, siendo la fila 00 y la columna 11, por lo que la salida es 10.
   b) Para S1, la entrada es 0111, siendo la fila 01 y la columna 11, por lo que la salida es 11.
6. Unir las salidas anteriores (1011) y aplicar la permutación P4: 0111
7. Realizar la XOR con L del paso 2: 0111 XOR 1010 = 1101
8. Tras aplicar la función f_k por primera vez, intercambiar L obtenida del paso 2 y R original: 1001 1101
9. Aplicar la expansión y permutación E/P sobre R (1101): 11101011
10. Realizar la XOR con la clave K2: 11101011 XOR 01000011 = 10101000
11. Aplicar las cajas S

   a) Para S0, la entrada es 1010, siendo la fila 10 y la columna 01, por lo que la salida es 10.
   b) Para S1, la entrada es 1000, siendo la fila 10 y la columna 00, por lo que la salida es 11.

12. Unir las salidas anteriores (1011) y aplicar la permutación P4: 0111
13. Realizar la XOR con L del paso 8: 0111 XOR 1001 = 1110
14. A la salida anterior, unida con la mitad derecha del paso 8, se le aplica la permutación IP inversa:
   a) Entrada: 1110 1101
   b) Salida: 01110111

Por tanto, el resultado de cifrar 01110010 con la clave 1010000010 es 01110111.

3. Desarrollo

Tras estudiar el comportamiento del algoritmo S-DES, se pide realizar la implementación software de dicho algoritmo.

3.1. Requerimientos

La solución desarrollada deberá cumplir los siguientes requerimientos mínimos:

- Cifrar y descifrar cualquier mensaje y fichero binario.
- Operativa a nivel de bit con los bloques y con las claves, es decir, los datos no se transformarán en una secuencia de caracteres a valor 1 o 0, sino como una unidad binaria.
- Soporte multibloque ilimitado, es decir, permitir el cifrado y descifrado de mensajes con una longitud superior al tamaño de un bloque.

Adicionalmente se valorará la implementación de diferentes modos de operación (ECB, CBC, etc.) cuando se realice el cifrado/descifrado de multiples bloques.

Por otro lado, en caso que se considere necesario, en función de la metodología de entrada de los datos utilizada, se deberá incluir un procedimiento de relleno o padding adecuado.

El programa deberá aceptar al menos tres argumentos de entrada:

- Operación a realizar (cifrado o descifrado).
- Secuencia de caracteres hexadecimal y/o binaria representando la clave.
- Ruta al fichero de entrada o *plaintext*.
- Ruta al fichero de salida o *ciphertext*.

A modo de ejemplo, la ejecución del mismo se lanzará mediante comandos similares a los siguientes:

```
$ ./sdes <mode> <key> <inputfile> <outputfile>
$ ./sdes -c -k <key> -i <inputfile> -o <outputfile>
```

3.2. Entorno de desarrollo

El estudiante puede emplear el lenguaje de programación o plataforma de desarrollo que desee, si bien no está permitida la implementación en Matlab y se recomienda el uso del lenguaje C.

Se proporciona, a través del Aula Virtual (Moodle) de la asignatura [3], una máquina virtual de Ubuntu cuyas credenciales de acceso son el usuario `student` con la contraseña `telematica`.

El alumno podrá modificar el contenido de dicha máquina virtual a su voluntad para adaptarla al lenguaje de programación y entorno de desarrollo de su elección. No obstante, la configuración de dicha máquina virtual incluye las herramientas básicas para el trabajo con el lenguaje C en entornos Linux.

Para la edición del código del programa se proporciona el entorno Visual Studio Code. Para la compilación de los ficheros en C, se incluye `gcc`, que se puede emplear desde la consola o terminal, o bien a través del propio Visual Studio Code mediante extensiones que deberá instalar el alumno.

Considerando que el fichero que contiene el código fuente y el programa ejecutable se alojarán en la misma carpeta, a modo ejemplificativo, el proceso de compilación/linkado se realiza mediante una comando similar al siguiente:

```
$ gcc sdes.c -o sdes
```

La ejecución del programa se lanzará con un comando similar al siguiente:

```
$ ./sdes <mode> <key> <inputfile> <outputfile>
```

En los anexos A y B se proporcionan un conjunto de funciones de ayuda que podrán ser utilizadas como base para el desarrollo del algoritmo. Adicionalmente, se dispone para la descarga un fichero `.h` en el que se incluye la definición de las diferentes matrices con los índices para las operaciones de permutación, expansión, etc.

4. Evaluación

Al finalizar la práctica y tras comprobar que la implementación del algoritmo S-DES funciona correctamente, será necesario responder a un cuestionario que se habilitará en el Aula Virtual (Moodle) y entregar una memoria descriptiva en la que se demuestren las tareas realizadas, además del código desarrollado.

El informe, en formato PDF, deberá incluir un breve resumen del código desarrollado, pudiendo hacer referencia a las funcionalidades descritas en la sección 2 y al código implementado. Se tendrá en cuenta la correcta estructura del documento, su formato, así como la claridad de las explicaciones.

Además el informe deberá incluir una sección con el cuestionario y las respuestas al mismo. El cuestionario se centrará principalmente en obtener datos que validen la correcta implementación del algoritmo por parte del alumno, si bien podrá incluir alguna cuestión con un desarrollo breve.

Se valorará positivamente estructurar adecuadamente el código (mediante el uso de funciones, cuando sea procedente), así como la utilización de comentarios que permitan interpretarlo de manera más sencilla.

El envío de los documentos se realizará a través de la tarea correspondiente en el Aula Virtual (Moodle) antes del 16 de diciembre de 2021.

> *Nota: Los asuntos de los correos electrónicos relacionados con esta práctica deberán contener la cadena* `[CRIPTO21_PR]` *seguida de una descripción breve del mensaje.*

5. Referencias

[1] Edward F. Schaefer (1996) *A SIMPLIFIED DATA ENCRYPTION STANDARD ALGORITHM*, Cryptologia, 20:1, 77-84, DOI: 10.1080/0161-119691884799

[2] National Institute of Standards and Technology (1979). *FIPS-46: Data Encryption Standard (DES)*. Revised as FIPS 46-1:1988, FIPS 46-2:1993, FIPS 46-3:1999, available at http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[3] Luis Muñoz, Jorge Lanza, *G844 - Criptografía y Seguridad en Redes y Servicios - Curso 2021-2022*, https://moodle.unican.es/course/view.php?id=11954

A Utilidades de ayuda

- Generar un fichero con contenido aleatorio de un tamaño determinado.

```
$ dd if=/dev/urandom of=<file> bs=1 count=<num_blocks>
```

- Generar un volcado hexadecimal de un archivo.

```
$ echo 'En formato hexdump'
$ xxd <fileName>
$ echo 'En formato hex plano'
$ xxd -p <fileName>
```

- Generar un fichero a partir de una secuencia hexadecimal

```
$ echo -n 0102030405060708 | xxd -p -r > out.bin
$ xxd out.bin
0000000: 0102 0304 0506 0708 ........
```

B Operativa binaria en C

- Desplazamiento a la izquierda

```
1 int main() {
2   unsigned int x, y, shift;
3
4   x = 0x00D1;
5   shift = 3;
6   y = x << shift;
7
8   printf("Expected output: 0x00D1 desplazado 3 bits a la
      izquierda es 0x0688\n");
9   printf("Output: 0x%04X desplazado %d bits a la izquierda
      es 0x%04X\n", x, shift y);
10 }
```

- Desplazamiento a la derecha

```
1 int main() {
2   unsigned int x, y, shift;
3
4   x = 0x1D1;
5   shift = 3;
6   y = x >> shift;
7
8   printf("Expected output: 0x01D1 desplazado 3 bits a la
      derecha es 0x003A\n");
9   printf("Output: 0x%04X desplazado %d bits a la derecha es
      0x%04X\n", x, shift y);
10 }
```

- Obtener el valor de un bit específico

```c
int main() {
  unsigned int x, y, bit;

  x = 0x03A;
  bit = 3;

  y = (x >> 3) & 0x01;

  printf("Expected output: El valor del bit 3 de 0x003A es 1\n");
  printf("Output: El valor del bit %d de 0x%04Xes %d\n", bit, x, y);
}
```

- Fijar el valor de un bit específico a 1

```c
int main() {
  unsigned int x, y, mask, bit;

  x = 0x03A;
  bit = 7;
  mask = 1 << (bit - 1);

  y = x | mask;

  printf("Expected output: Se ha fijado el bit 5 de 0x003A a 1, resultando 0x007A\n");
  printf("Output: Se ha fijado el bit %d de 0x%04X a 1, resultando 0x%04X\n", bit, x, y);
}
```

- Fijar el valor de un bit específico a 0

```c
int main() {
  unsigned int x, y, mask, bit;

  x = 0x03A;
  bit = 5;
  mask = 1 << bit;

  y = x & (~mask);

  printf("Expected output: Se ha fijado el bit 5 de 0x003A a 0, resultando 0x001A\n");
  printf("Output: Se ha fijado el bit %d de 0x%04X a 0, resultando 0x%04X\n", bit, x, y);
}
```

- Conmutar el valor de un bit específico

```c
int main() {
  unsigned int x, y, mask, bit;

  x = 0x03A;
  bit = 5;
  mask = 1 << bit;

```

```
8    y = x ^ mask;
9
10   printf("Expected output: Se ha conmutado el bit 5 de 0
       x003A, resultando 0x001A\n");
11   printf("Output: Se ha conmutado el bit %d de 0x%04X,
       resultando 0x%04X\n", bit, x, y);
12 }
```

- Fijar el valor de un bit específico

```
1 int main() {
2   unsigned int x, y, mask, bit, val, newval;
3
4   x = 0x03A;
5   val = 1;
6   bit = 5;
7   mask = 1 << bit;
8
9   newval = !!val;
10
11  y = x & (~mask);
12
13  printf("Expected output: Se ha fijado el bit 5 de 0x003A a
      1, resultando 0x001A\n");
14  printf("Output: Se ha fijado el bit %d de 0x%04X a %d,
      resultando 0x%04X\n", bit, x, newval, y);
15 }
```