Fundamentos de criptografía y seguridad informática 2016/2017

PRÁCTICA 2: SEGURIDAD PERFECTA Y CRIPTOGRAFÍA SIMÉTRICA

Nombres:

Miguel Andrés Russián Rojas

Víctor Martín Hernández

Grupo: 1461

Ejercicio 1: Seguridad Perfecta

Comprobación empírica de la Seguridad Perfecta del cifrado de desplazamiento

El programa calcula las probabilidades condicionales del texto plano al texto cifrado promediando las probabilidades extraídas cifrando el texto múltiples veces (un número constante determinado por la macro `SAMPLE_SIZE` en `algoritmos.h`) con claves generadas o bien de manera equiprobable o bien de manera sesgada, según se explicite como argumento del programa.

Para crear claves de manera equiprobable, hemos creado una clave nueva para cada cifrado distinto, de tamaño igual del texto a cifrar, clave cuyos bytes son creados mediante la funcion `random()` de `stdlib.h`. En contraparte, para el cifrado sesgado se utiliza un número fijo (macro `KEYSPACE`, por el tamaño del espacio de claves) de claves generadas aleatoreamente, todas menos la última con probabilidades de ser usadas de p, p^2, p^3... salvo la última que concentra la probabilidad restante, donde p viene dada por la macro `BIASED_P`.

Tras probar con varios tamaños de texto (de 50 a 4000 palabras) ambos modos de operación, observamos que la seguridad era perfecta para el cifrado con claves equiprobables, es decir, la probabilidad de encontrar un caracter en texto plano es la misma que la suma de las probabilidades de ese mismo caracter condicionadas a cada caracter posible del texto cifrado, con independencia del tamaño del texto. Sin embargo, para el cifrado sesgado, en promedio la diferencia de probabilidad entre ambas probabilidades no variaba con respecto al tamaño del texto, aunque sí su varianza, que tendía a reducirse con textos más grandes.

Ejercicio 2: Implementación del DES

Programación del DES

En este ejercicio número dos se nos pedía la implementación del criptosistema *DES*. En este caso, el *DES* pedido será el de 16 rondas en el modo *ECB*. El modo de ejecución del programa será el siguiente.

```
./desECB {-C | -D -k clave} [-i filein] [-o fileout]
```

El modo *ECB* se caracteriza por la capacidad de poder cifrar de manera paralela debido a la división del texto plano en bloques y el empleo de las mismas claves generadas aleatoriamente. Normalmente es utilizado para el envío de mensajes cortos (como transferencias de claves) ya que, para mensajes largos estructurados, no es seguro debido a la posible repetición de bloques y clave; pudiendo obtener repeticiones en el cifrado. El esquema seguido es el siguiente:

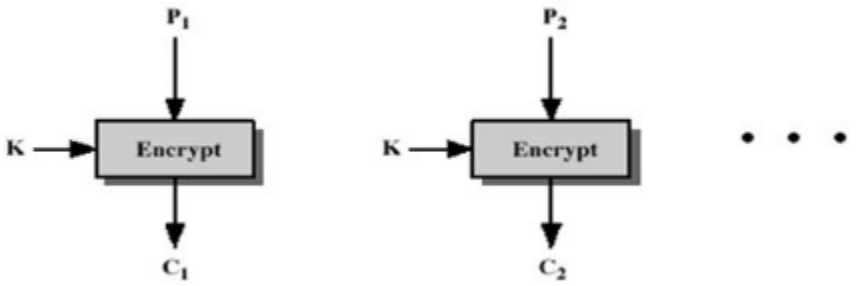


Figura 1: Modo ECB del DES

Previa implementación del *DES* elaboramos dos librerías propias, una para trabajar con bits y otra que implementase cada función necesaria del criptosistema. Posteriormente, realizamos el programa en sí, controlando que los parámetros de entrada sean los correctos y, teniendo en cuenta dos detalles:

1. La longitud en bits del fichero de entrada no es múltiplo de 64. En ese caso, se añadirán los carcateres necesarios para que lo sean. Hemos optado por añadir tantas 'A' como sean necesarias.
2. Modo de cifrado. En ese caso, se generará automáticamente la clave de cifrado, compuesta por números y letras, mostrándose por la salida estándar para poder utilizarla en el modo de descifrado.

Cabe destacar que la librería que implementa las funciones propias del *DES* hace uso del fichero facilitado en *moodle,* que contiene los valores numéricos de las 8 *S-BOXES* y las diferentes permutaciones necesarias. El esquema seguido para el cifrado es el mostrado en la *Figura 2*.

Por tanto, en el descifrado se aplicarán las mismas funciones que en el cifrado pero la introducción de las claves generadas serán en orden inversa, es decir, desde K_{16} a K_1 (siguiendo el esquema de la *Figura 2*). Esto es posible gracias a las propiedades de la función XOR.

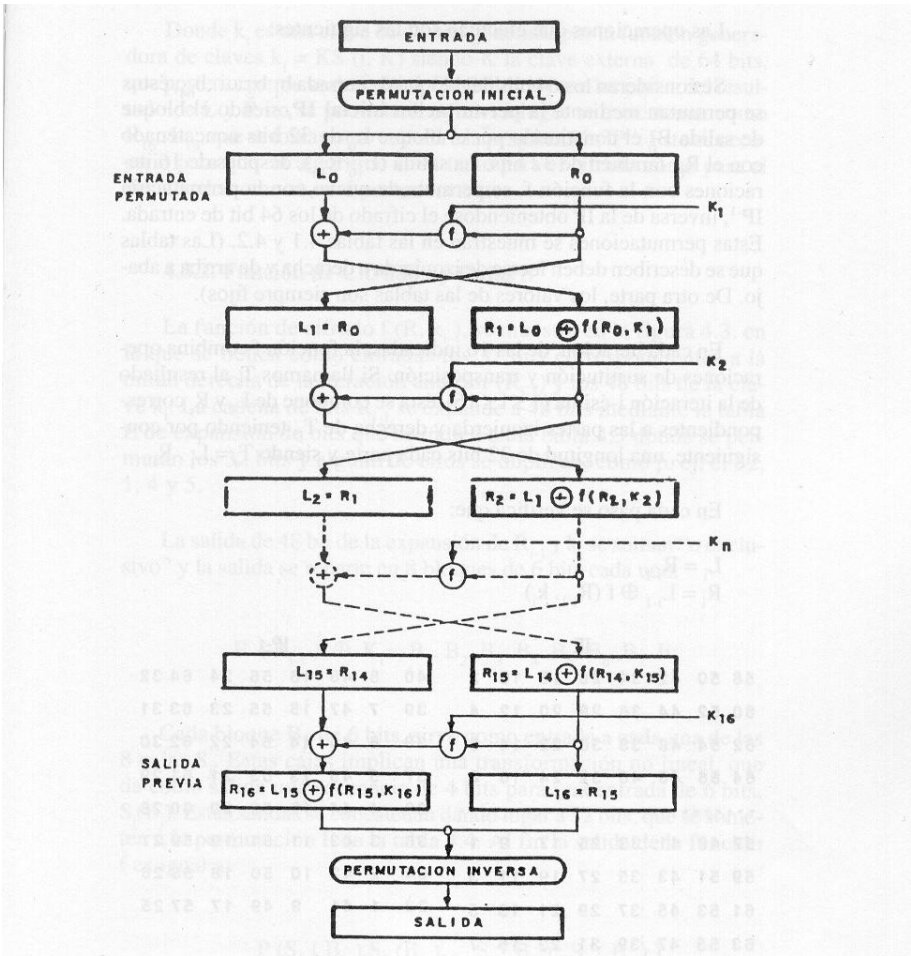


Figura 2: Computación del cifrado

Programación del Triple DES

En este segundo apartado del ejercicio 2 se nos pedía implementar el *Triple DES* en el modo de operación *CBC* utilizando las funciones utilizadas en el apartado anterior. El modo de ejecución del programa es el siguiente:

```
./tripleDES {-C | -D -k clave -v IV} [-i filein] [-o fileout]
```

Para tratar de conseguir un programa lo más similar posible, hemos considerado que al ejecutar el programa con la opción -C tanto la clave como el vector de inicialización se generen de manera aleatoria por el programa y se muestren por pantalla al cifrar un texto determinado.

Algunas de las diferencias que encontramos en el modo CBC frente al ECB visto anteriormente son las siguientes:

- No permite la paralelización debido al encadenamiento existente entre los diferentes cifrados de bloques, es decir, es necesario la salida del anterior para poder realizar el cifrado del siguiente. Esto permite que los cambios producidos a la salida sean aún mayores.
- Introducción de un nuevo secreto, el vector de inicialización

El esquema que sigue dicho modo es el mostrado en la *Figura 3*.

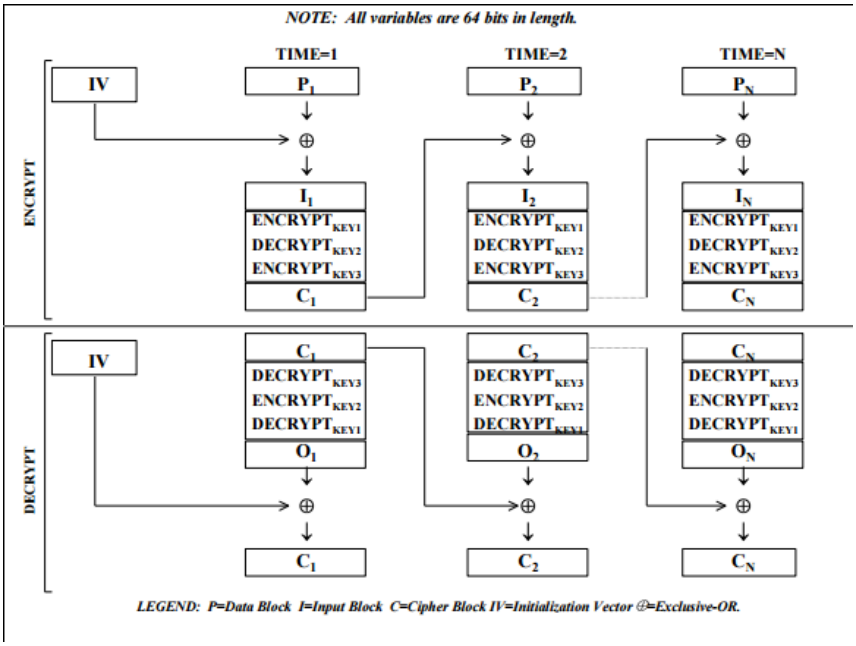


Figura 3: Modo CBC para el Triple Des

Ejercicio 3: Principios del DES

Estudio de la no linealidad de las S-Boxes

Una primera prueba de las S-Boxes del DES es la medición de la variación de la salida para un valor de entrada concreto con respecto a otros valores de entrada cercanos: en nuestras pruebas hemos observado que por lo general la diferencia entre salidas para un número y su siguiente tiende al 55% considerando todas las entradas posibles, lo cual es bastante propicio para disociar la entrada de la salida.

Al repetir estas pruebas con palabras de 48 bits para usar todas las S-Boxes, la variación promedio cae al 6.16% (unos 2 bits), con un máximo de 10 bits cambiados, debido a que incrementar la entrada cambia por lo general solo su parte menos significativa, y por tanto en la concatenación de las salidas de las S-Boxes, cambia una única porción de 4 bits de toda la salida.

La tercera prueba sería la demostración de la independencia lineal de entrada y salida, pero no de cada entrada a su salida (lo cual es improbable en vista de que hay menos salidas que entradas y la mayoría son números no primos), sino de dos entradas no relativamente primas con salidas tampoco relativamente primas. Esto significa que dada una entrada y su salida, buscar la entrada correspondiente a una salida sería más sencillo probando primero con múltiplos primero. Sin embargo, nuestras pruebas demuestran que no hay correlanción tal entre entradas y salidas

Los resultados de ambas pruebas se han obtenido mediante el programa desNLTest (test de no linealidad), con estos parámetros:

```
./desNLTest [-n num_pruebas] [-k num_pruebas] [-d num_pruebas]
```

Donde la opción -n realiza la primera prueba sobre todos los números hasta el especificado (existen 64 posibles), -k realiza la segunda prueba sobre todas los números posibles hasta el especificado (de 0 a 2^{48}-1, es decir todas las entradas posibles de la caja S completa), y la opción -d realiza la tercera prueba, de nuevo sobre 64 valores que son los correspondientes a todas las entradas posibles de una caja S

Hubo una última prueba que por restricciones computacionales no se pudo llevar a cabo, que probaba de manera empírica el principio de no linealidad de las entradas con la salida, es decir $f(a \oplus b) \neq f(a) \oplus f(b)$. El algoritmo de comparación tiene una complejidad de $O(n^2)$, donde en el caso peor n es 2^{48}. La prueba se encuentra implementada pero comentada dado que incluso para muestras poco significativas (2^{13} de 2^{48}, es decir, $2.9 \cdot 10^{-9}$ %) el tiempo de computación es alto.

Estudio del Efecto de Avalancha

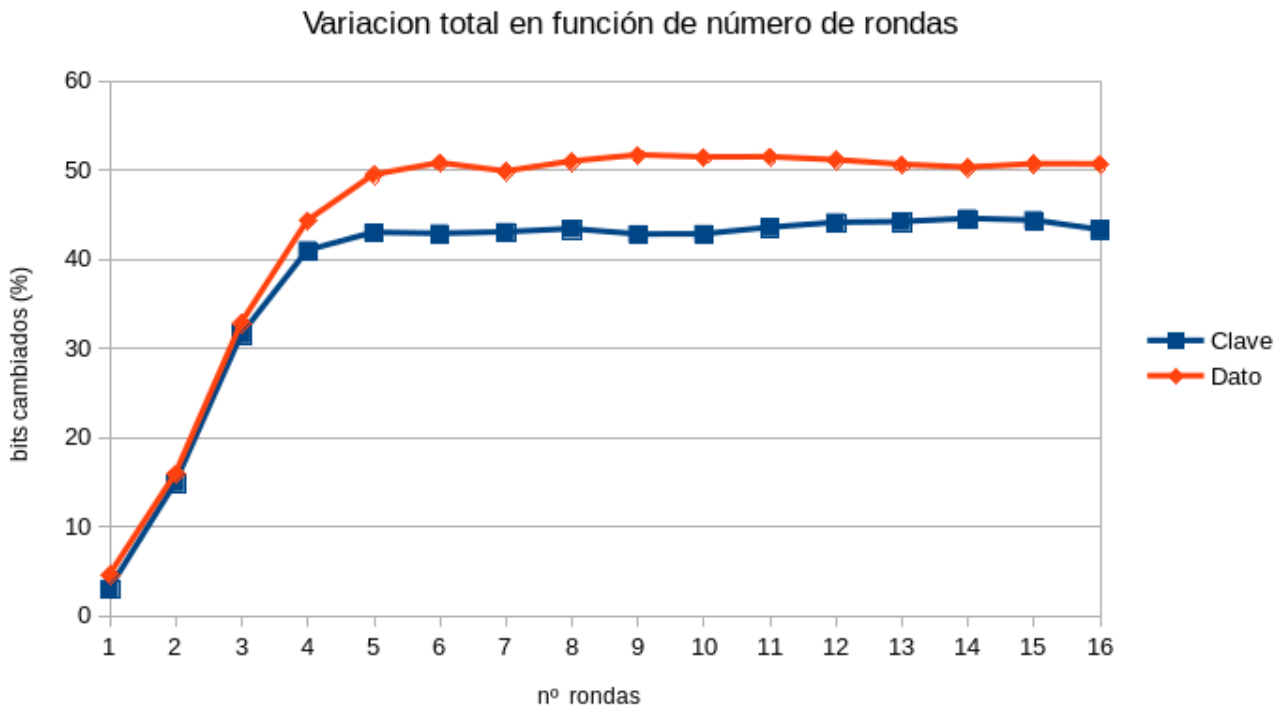
Para conocer el efecto que tienen en la salida del algoritmo pequeñas variaciones en las entradas (dato a cifrar y clave), hemos creado el programa desAvTest con las siguientes opciones:

```
./desAvTest {-I | -K} {-n max_rondas} {-k clave} {-d dato_prueba}
```

En donde se provee un bloque o dato a cifrar (opción -d), la clave (-k), el número máximo de rondas con que cifrar (-n) y bien si se varía la clave (con la opción -K) o el bloque de entrada. El programa presenta el promedio y varianza de los cambios ocurridos al cambiar un solo bit de entre todos los posibles de la palabra escogida, tanto para todas las rondas como si formasen su propio DES de una ronda, como para el DES completo con las rondas hasta la especificada.

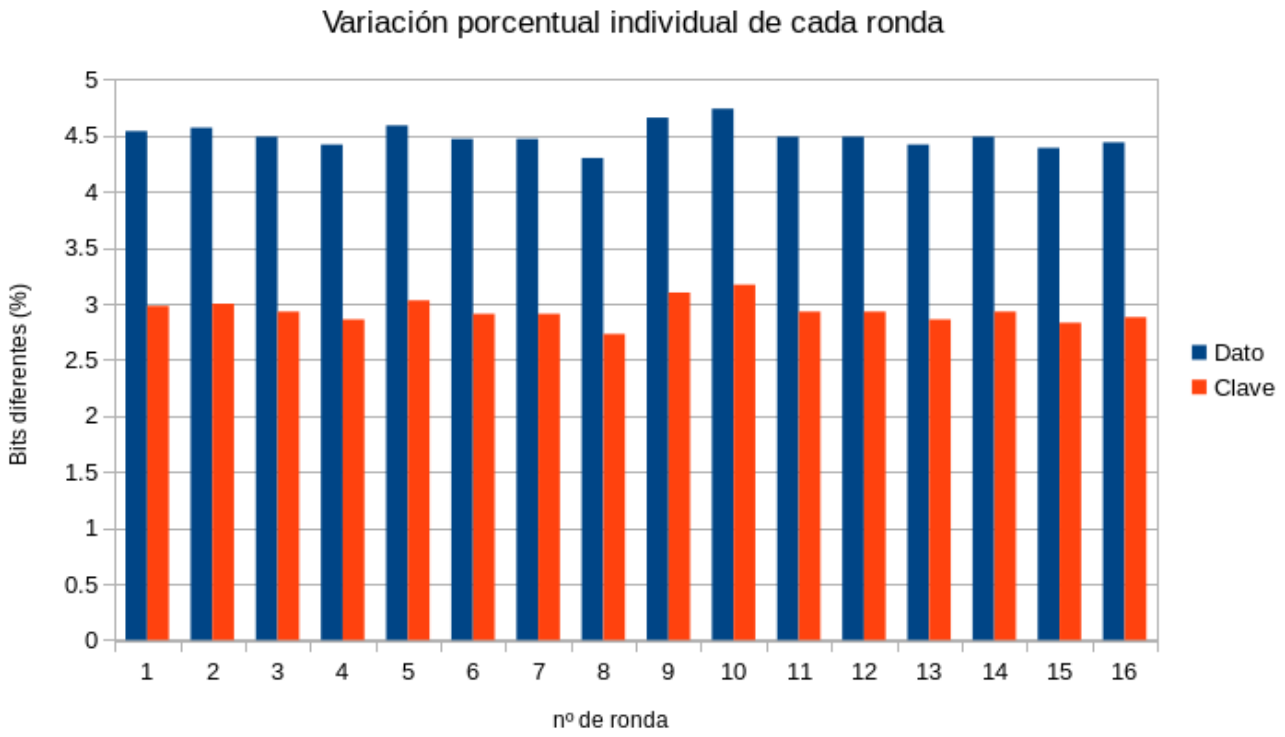
Mediante un script que tomaba un par de palabras aleatorias para la clave y el bloque en claro e iteraba sobre los posibles números de rondas totales, hemos obtenido algunos resultados interesantes que analizamos a continuación:

En primer lugar, el número de rondas, como es de esperar, fue influyente en el número de bits que cambiaban en la salida al variar un bit en la entrada, pero hasta cierto punto



Como podemos apreciar, a partir de la quinta ronda la variación con respecto a la entrada y la entrada con un bit cambiado se estabiliza alrededor del 50%. Esto es deseable porque la ganacia de información conociendo el bloque cifrado (y su negado) se minimiza, en vista de que tiene el menor parecido posible con ambos. También observamos que la variación de la clave tiene un efecto menor que la cambiar el dato, lo que se puede deber al menor número de operaciones ligadas a la generación de claves frente al propio algoritmo de cifrado

En segundo lugar, y en consonancia con los resultados del apartado anterior, cada ronda independiente tiene un efecto de cambio semejante al resto de rondas



Recordemos que sumando 1 a la entrada, los cambios en promedio rondaban el 6%, por tanto al variar solo un bit (lo cual es un cambio menor al no tener que considerar acarreos de suma), es razonable que la variación sea menor. Esto también muestra que las permutaciones tienen poco efecto en la variación de bits, puesto que su objetivo es el de mezclar más que sustituir.

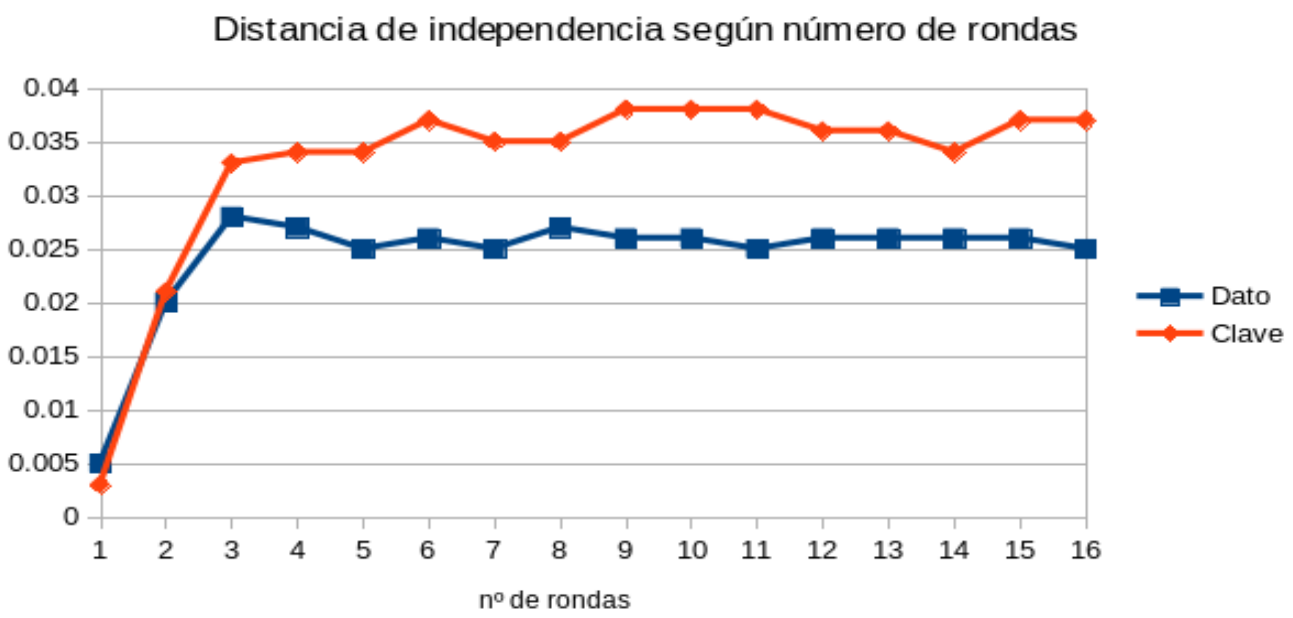
El Criterio de Avalancha Estricto (SAC) y el Criterio de Independencia de Bits (BIC)

Para estudiar estos principios de diseño se creó un programa nuevo con la siguiente interfaz:

```
./desFTests {-S | -B} {-I | -K} {-n max_rondas} {-k clave} {-d dato_prueba}
```

El nombre se refiere a que se realizan pruebas sobre los fundamentos de diseño del DES. La interfaz es similar al del programa anterior, pero añade las opciones -S para mostrar los resultados del análisis del SAC y -B para el análisis del BIC. Otra diferencia es que se imprime solo el resultado aplicando el número de iteraciones proporcionadas únicamente.

Para el BIC, la siguiente gráfica ilustra nuestros resultados:

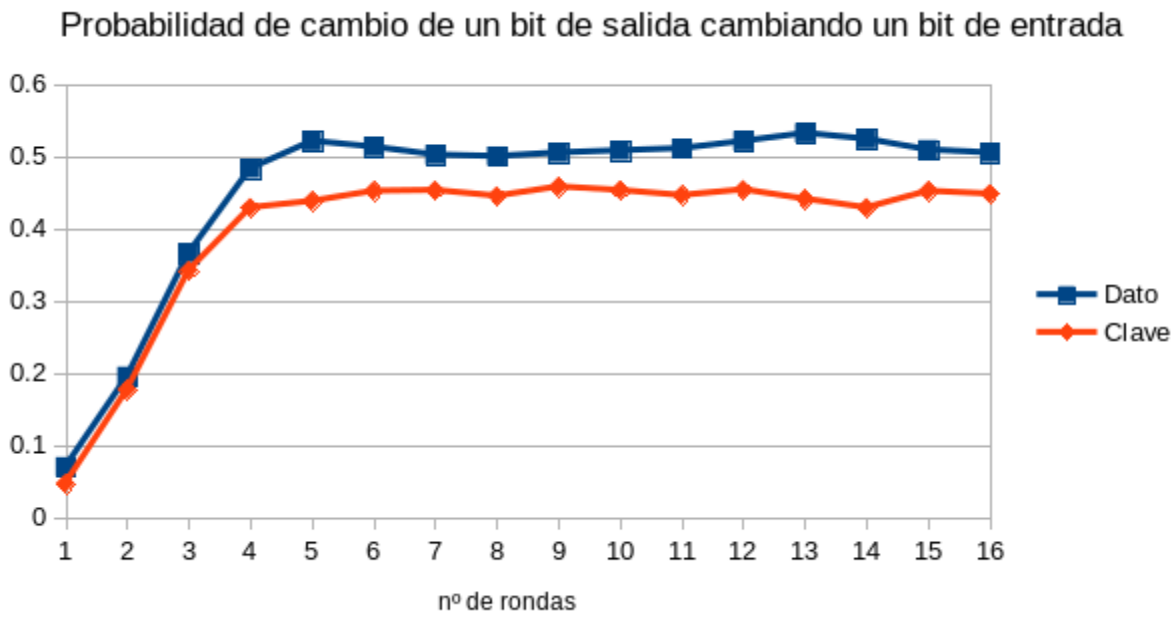


Definimos la distancia a independencia estadística como |P(A,B) – P(A)·P(B)| donde A y B son bits, y se mide la probabilidad de ser invertidos. Recordemos que para cumplir el criterio BIC, se debe cumplir que las probabilidades de cambio de cualesquiera dos bits deben ser independientes para cualquier cambio en la entrada, por lo cual P(A,B) = P(A)·P(B), luego la distancia a la independencia promedio mide la conformidad con el BIC promedio cambiando cada uno de los bits de entrada.

Podemos observar que en general la distancia es menor al 4%, y que al variar el dato es siempre menor al 3%, lo cual es bastante deseable en vista de que el dato a cifrar cambia con mucha más frecuencia que la clave. También cabe destacar que la menor distancia está en la primera ronda, en donde es más sencillo ajustar las cajas S y otros parámetros necesarios para el cifrado de tal manera que se adecúe lo mejor a los principios de diseño.

Por otro lado está el criterio SAC, que consiste en que la probabilidad de cambio de un bit cualquiera en la salida cambiando cualquier bit en la entrada es de 0.5. Esta prueba es más rigurosa porque no mide el cambio total entre salidas, sino las probabilidades de cambio de cada bit independientemente.

Los resultados obtenidos se muestran en la siguiente gráfica:



Aunque se aprecian resultados similares a los del estudio del efecto avalancha en el apartado anterior, vemos que a partir de 4 rondas se cumple este criterio. Cabe destacar que también a partir de la ronda 4, la varianza ronda el 0.01 con variación de pocas milésimas, lo que significa que todos los bits tienen una probabilidad de cambio cercana a la media, que se ve en la gráfica.

Ejercicio 4: Principios de diseño del AES

Estudio de la no linealidad de las S-Boxes del AES

Gracias a que la tabla de sustitución es única y el tamaño de la entrada y la salida es la misma, el análisis se simplifica bastante.

Para entradas consecutivas, los bits que difieren entre entrada y salida son 3.5 en promedio (21.92% del byte) para el cifrado, mientras que para el descifrado es de 4.88 (30.51% del byte). Esto implica que sin complementar la sustitución con otras funciones y rondas, sería posible realizar un análisis estadístico (eso sin contar con que las tablas de sustitución son públicas).

Por otro lado, para entradas consecutivas, las salidas difieren en 1.5 bits en promedio en ambos modos, es decir, para tres entradas consecutivas dadas, el número de bits cambiados entre parejas de salidas cambia 1.5 bits.

Finalmente, se ha encontrado que la mayoría de las parejas de entradas son linealmente independientes de sus salidas, pero para el cifrado no lo son un 14.66% de las parejas, y para el descifrado un 22.89% de ellas. Considerando que las tablas recorren todos los posibles enteros entre 0 y 255, no es de extrañar que algunas parejas de valores sean linealmente dependientes. De nuevo, gran parte de la robustez radica en el producto de criptosistemas.

Generación de las S-Boxes del AES

Para la creación de las S-Boxes del AES resultó apropiado adoptar la manipulación de bits dentro de bytes en lugar de la creación dinámica de cadenas representando bits, para lo cual se implementaron funciones para rotar bits en ambas direcciones, filtrar y componer bytes en palabras de 32 bits. Luego se desarrollaron y probaron por separado las funciones necesarias para obtener el grado del polinomio y operar en aritmética $GF(2^8)$, como divisón, xtime, multiplicación y búsqueda del inverso multiplicativo en $GF(2^8)$ módulo un polinomio dado.

El programa tiene la interfaz:

```
./SBOX_AES {-C | -D} [-o file_out]
```

Y genera las tablas para la sustitución en el cifrado (opción -C) o descifrado (-D), las cuales se imprimen para una mejor lectura como una tabla similar a las imágenes 7 y 14 del FIPS 197 (secciones 5.1.1 y 5.3.2). Se ha probado que las tablas resultantes coinciden con las tablas de sustitución proporcionadas.

Programación del AES

Aunque no es parte del estándar, se ha programado un algoritmo AES de 16 rondas en lugar de 10, para nk = 4 y nb = 4. Para ello se ha aprovechado el código de apartados anteriores pero también se ha implementado funciones relativas a aritmética de polinomios con coeficienctes $GF(2^8)$, los pasos de mezcla de columnas, rotación de filas, generación y adición de claves intermedias, siguiendo de manera fidedigna y trazable el pseudo-código expuesto en el documento FIPS 197.

Para el cifrado *Blockchain* primero se ha programado y probado una función para cifrar un único bloque siguiendo la filosofía UNIX de ”hacer una cosa, y hacerla bien”. El resto de la implementación fue sencillo una vez que se tenía el cifrado de bloques individuales.

El programa se invoca mediante la siguiente interfaz:

```
./AesCBC {-C | -D -k clave} [-i file_in ] [-o file_out]
```

Siguiendo la interfaz de programas anteriores. El argumento -k representa un archivo donde se encuentra la clave de 128 bits como nk bloques legibles como números en hexadecimal, separados por espacios. Un archivo de este formato es generado al cifrar con -C.

De ser necesario, el relleno de bloques es el caracter ‘\0’, que se aprecia en el descifrado