Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC3253 — Criptografía y Seguridad Computacional 2021-1

NOMBRE: Matías Duhalde

Tarea 2

Pregunta 1

Parte a)

Como se menciona en el enunciado, *key schedule* corresponde a un algoritmo que calcula las llaves a usar en las rondas a partir de una llave inicial. Para el caso particular de DES, este es utilizado para generar las llaves a usar durante la ejecución según la cantidad de rondas (16 en la definición estándar).

Key scheduling en DES funciona de la siguiente manera: se comienza con la llave inicial de 64 bits (también llamada Master Key), de la cual sólo 56 bits corresponden efectivamente a la clave, debido a que el octavo bit en cada byte indica paridad del byte. Los bits de la Master Key son permutados según función definida, resultando en un string de 56 bits de largo. La matriz que describe esta permutación inicial es la siguiente:

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

Esta matriz indica que el i-ésimo carácter del nuevo string corresponderá al carácter del string original en la posición del i-ésimo valor de la matriz. Por ejemplo, el carácter de la posición 33 ocupará el cuarto lugar. Notar que esta matriz 56 valores, saltándose los bits de paridad (múltiplos de 8), reduciendo el string de 64 a 56 bits.

El string resultante luego se separa por la mitad en dos substrings, C y D, de $\frac{56}{2} = 28$ bits cada una.

Luego, para cada ronda, se aplica un *left-rotate* a C y D de dos posiciones, excepto en las rondas 1, 2, 9, y 16, donde el *rotate* es de una posición. Luego, se concatena C y D, formando un string de 56 bits, el cual se vuelve a permutar con otra función fija, representada por la siguiente matriz:

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

Notar que nuevamente se reduce el tamaño de 56 a 48 bits. El valor resultante de esta permutación corresponde a la *key* de la ronda k_r. Para obtener las siguientes llaves, se vuelve a repetir con C y D resultantes el proceso de *left-rotate* y permutación un total de 16 veces, para obtener todas las llaves requeridas por DES.

Parte b)

Ignorando el primer y ultimo paso de permutación de DES (Initial permutation y Final permutation), los elementos del algoritmo tendrían la siguiente estructura:

$$
\begin{aligned}
L_0 &= m[: \frac{|M|}{2}] \\
R_0 &= m[\frac{|M|}{2} :] \\
L_1 &= R_0 \\
R_1 &= L_0 \bigoplus f(k_1, R_0) \\
L_2 &= R_1 \\
R_2 &= L_1 \bigoplus f(k_2, R_1) \\
L_3 &= R_2 \\
R_3 &= L_2 \bigoplus f(k_3, R_2) \\
c &= L_3 + R_3
\end{aligned}
$$

Componiendo las variables, se obtiene finalmente

$$
\begin{aligned}
&= L_0 \bigoplus f(k_1, R_0) \bigoplus f(k_3, L_3) \\
&= L_0 \bigoplus f(k_1, R_0) \bigoplus f(k_3, R_2)
\end{aligned}
$$

Supongamos que hacemos un ataque de texto plano, donde tenemos una cierta cantidad de pares (m, c), donde m es el mensaje original, y c es el resultado de encriptar el mensaje con el algoritmo señalado.

Si tuviéramos dos pares de mensajes, $m = L_0||R_0$, y $m' = L'_0||R'_0$, ambos encriptados con la misma llave, entonces podemos componer las funciones encontradas de la siguiente manera:

$$
R_3 \bigoplus R'_3 = L_0 \bigoplus L'_0 \bigoplus f(k_1, R_0) \bigoplus f(k_3, R_2) \bigoplus f(k_1, R'_0) \bigoplus f(k_3, R'_2)
$$

Si escogemos m y m' tal que $R_0 = R'_0$ (y en consecuencia $f(k_1, R_0) = f(k_1, R'_0)$), podemos reducir la segunda relación a lo siguiente:

$$
\begin{aligned}
R_3 \bigoplus R'_3 &= L_0 \bigoplus L'_0 \bigoplus f(k_3, R_2) \bigoplus f(k_3, R'_2) \\
R_3 \bigoplus R'_3 \bigoplus L_0 \bigoplus L'_0 &= f(k_3, R_2) \bigoplus f(k_3, R'_2) \\
R_3 \bigoplus R'_3 \bigoplus L_0 \bigoplus L'_0 &= f(k_3, L_3) \bigoplus f(k_3, L'_3)
\end{aligned}
$$

Ahora, si analizamos la función f, tenemos que:

$$
\begin{aligned}
f(k_3, R_2) &= P/S(E(L_3) \bigoplus k_3) \\
f(k_3, R'_2) &= P/S(E(L'_3) \bigoplus k_3)
\end{aligned}
$$

Donde P/S corresponde a la función que aplica la red de permutación-sustitución de f, y E corresponde a la función de extensión de largo (aumenta el argumento de 32 a 48 bits, para permitir XOR).

Dado que $R_1 = L_2 = f(k_3, L_3) \bigoplus R_3$, y $R_1 = L_0 \bigoplus f(k_1, R_0)$, podemos formular

$$
f(k_3, L_3) \bigoplus R_3 = L_0 \bigoplus f(k_1, R_0)
$$

Ya obtenida la subkey, se puede *invertir* la permutación hecha durante el key-schedule, realizando la permutación inversa. Sin embargo, quedarían 8 bits de C y D que no se pueden obtener directamente, debido a que la permutación reduce la llave desde 56 bits a 48. Los 8 bits restantes se pueden obtener con fuerza bruta en 2^8 pasos. A partir de los C y D generados se pueden generar las subkeys correspondientes para todas las rondas fácilmente. Además, se puede

Así, es posible ejecutar un ataque para este algoritmo en alrededor de 2^24 pasos.