Universidad del Valle de Guatemala

Microprocesadores

Juan Celada

Ciclo 2, 2020

Proyecto 2

Gabriel Quiroz 19255

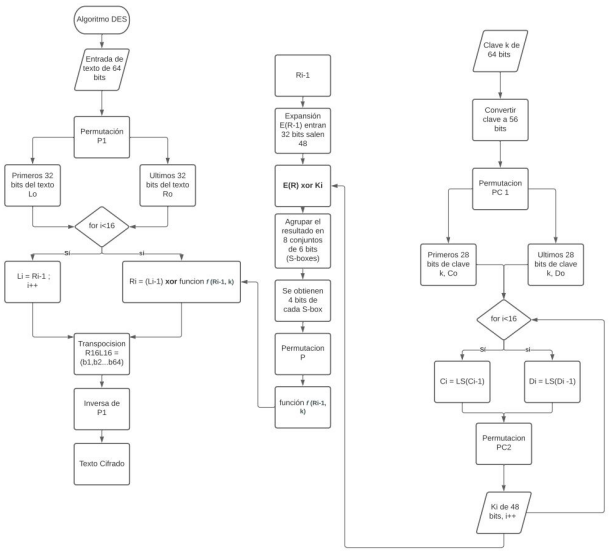
Jose Ponce 19092

Índice

Introducción

El algoritmo de encriptación de datos (DES) es un método para cifrar información. Este método es bastante antiguo ya que fue creado en 1973 y 1974, fue desarrollado por un equipo de IBM en cual destacaban Feistel y Walter Tuchman. El algoritmo se basa en tomar un texto de 64 bits y convertirlo en otro totalmente diferente mediante distintas operaciones y funciones. En el presente trabajo se realizó una implementación del algoritmo, utilizando mecanismos de programación paralela para hacer más eficiente la funcionalidad del algoritmo.

Diagrama de Funcionamiento

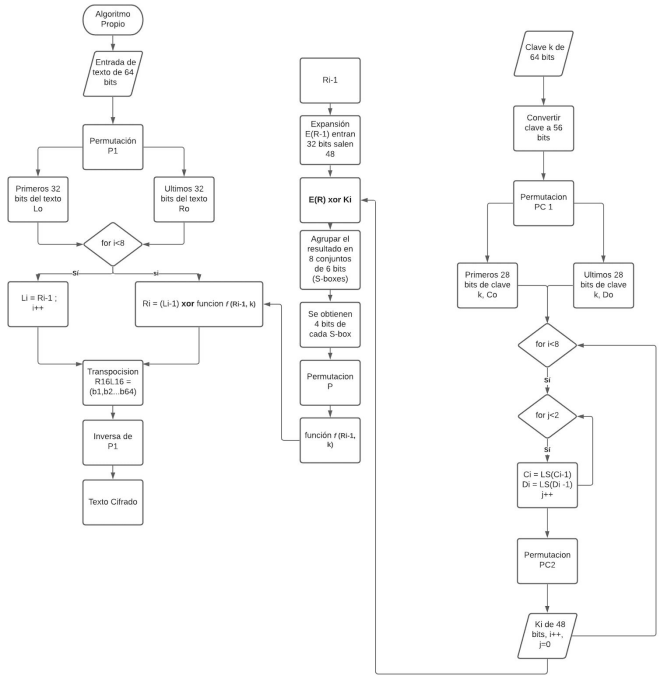


Descripción del Funcionamiento

El algoritmo Des primeramente realiza la permutación inicial 1 con el texto de 64 bits convertido anteriormente a binario, seguido de esto se toman los 64 bits y se dividen en dos partes, la primera tomando los primeros 32 y la segunda parte tomaría los últimos 32 bits. Por otro lado se realiza la permutación PC1 a la clave de 64 bits para convertirla a 56 bits.

Al tener hechas las 2 permutaciones mencionadas, a través de 16 rondas, primero se separa la clave en dos grupos de 28 bits y se realiza un desplazamiento circular de un bit a la izquierda a cada uno de los dos grupos. Luego, se realiza la permutación PC2 a la clave para obtener la clave final que será utilizada en los S-boxes. Para obtener los S-boxes se toma el grupo derecho de 32 bits tras la primera permutación, a estos se les aplica una expansión para llevarlo a 48 bits los cuales se operan mediante un XOR con una de las claves anteriormente mencionadas. Con el resultado de la anterior operación se hacen 8 grupos de 6 bits, se obtiene el primero y el último y se concatenan, lo mismo con los bits del medio. Los bits extremos van a indicar el número de fila y los centrales el número de columna para finalmente obtener 4 bits de cada S-box. Luego de obtener 4 bits de cada S-box se juntan y se tienen 32 a los cuales se les aplica la permutación P para finalmente obtener la función F. Por otro lado, los primeros 32 bits del texto a cifrar pasan a ser los últimos 32 bits y los últimos 32 bits a través de un xor con los primeros 32 bits del texto a cifrar y la función F pasan a ser los primeros 32 bits. Este proceso mencionado se realiza 15 veces más. Por último, se realiza la permutación inversa y de esta manera estaría cifrado el texto a través del algoritmo DES.

Diagrama de Funcionamiento del Algoritmo Propuesto



Descripción del Funcionamiento del Algoritmo Propuesto

El algoritmo propuesto funciona de una manera muy similar al algoritmo original, cuenta con todos los pasos, permutaciones y S-boxes. Salvo unas modificaciones tales como en lugar de ser 16 rondas se hicieron 8, en lugar de hacer corrimiento de 1 bit se realizó de 2 bit al momento de obtener la clave y se modificó la obtención de los bits las S-boxes.

Catálogo de Rutinas

void* Leer(void* palabra1) : Parámetro palabra1 contiene la palabra leída del archivo de texto. Subrutina utilizada para leer el archivo de texto y posteriormente convertir los caracteres a binario y almacenarlos en el buffer global buffer[64].

void* Leer2(void* palabra1) : Parámetro palabra1 contiene la clave ingresada por el usuario. Subrutina para leer la clave y posteriormente convertir los caracteres a binario y almacenarlos en el buffer global bufferclave64[64].

void* PermutacionP1() : Subrutina para realizar la permutación P1 al texto a cifrar y almacenar los primeros y últimos 32 bits en los buffer globales izquierda[32] y derecha[32].

void* PermutacionPC1F() : Subrutina de permutación PC1 a clave de 64 bits para convertirla a 56 bits y almacenarla en la variable global bufferclave56permutacionpc1[56].

void* ClaveDesplazamientoLS(): Subrutina utilizada para el desplazamiento circular de 2 bits a la izquierda a la clave y almacenarla en el buffer global izquierdayDerechaClaveDesplazamiento2bit[56].

void* PermutacionPC2F() : Subrutina utilizada para la permutación final PC2 para obtener la clave de 48 bits y almacenarla en la variable global bufferclave48permutacionpc2[48].

void* rondasLiRi() : Subrutina utilizada para realizar las rondas de intercambio y xor con la función F, de los primeros y últimos 32 bits de la palabra a cifrar y almacenarla en las variables globales izquierdanueva[32] y derechanueva[32].

void* PermutacionINVERSA() : Subrutina utilizada para realizar la permutación final inversa para obtener el texto completamente cifrado y se almacena en la variable global bufferfinal[64].

void* expansion(): Subrutina utilizada para realizar la expansión de 32 a 48 bits, no recibe ningún parámetro y tampoco devuelve. Utiliza buffers globales para realizar la expansión mediante la matriz base de la expansión

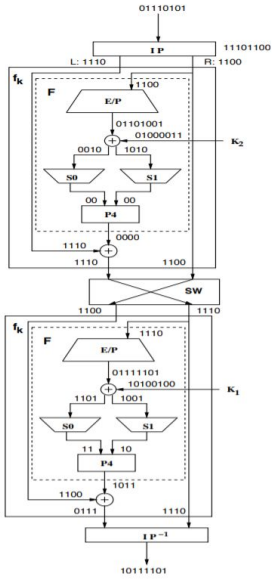
int binarioadecimal(string binario): Subrutina utilizada para convertir un numero binario a decimal. Esta subrutina recibe un string y retorna un número entero.

bitses<4>* enteroaBinario(int valor): Subrutina utilizada para convertir un numero decimal a una cadena de 4 bits. Esta subrutina recibe un número entero y retorna una cadena de 4 bits.

void* Sboxes(): Subrutina utilizada para la creación de las S-boxes y posteriormente para realizar la permutación P con los bits obtenidos de las S-boxes. Esta subrutina utiliza binarioadecimal y enteroaBinario, además, hace uso de buffers globales para realizar las operaciones.

string binarioascii(string binario): Subrutina utilizada para convertir un numero binario a ascii. Esta subrutina recibe un string y retorna un string.

Ilustraciones Importantes



(ESCOM, 2017)
Imagen que describe a grandes rasgos el descifrado del algoritmo DES

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

Permutación Inicial (P1)							
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

(Rafael Net, 2011)
Imagen donde se puede observar como es el orden de los valores del texto a cifrar luego de permutación Inicial P1.

Sub-bloque C0 = 0000000 0111111 1111111 1100000

Sub-bloque D0 = 1100010 1110011 0010010 1001001

Al ser la primer vuelta, el desplazamiento es de un bit a la izquierda como se indica en la tabla, dando como resultado C1 y D1.

Sub-bloque C1 = 0000000 1111111 1111111 1000000

Sub-bloque D1 = 1000101 1100110 0100101 0010011

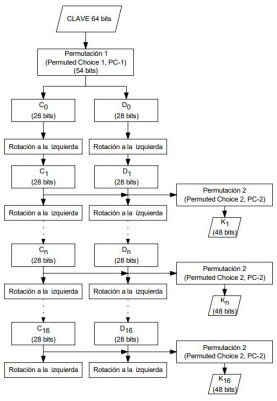
(Rafael Net, 2011)
Imagen donde se puede observar el desplazamiento de un bit en los dos bloques de 28 bits de la clave.

Tabla de 64 bits de K1 Inicial							
0	1	0	1	0	0	1	1
0	1	1	0	0	0	0	1
0	1	1	0	1	1	1	0
0	1	1	1	0	1	0	0
0	1	1	0	1	0	0	1
0	1	1	0	0	0	0	1
0	1	1	0	0	1	1	1
0	1	1	0	1	1	1	1

Tabla de 56 bits							
0	1	0	1	0	0	1	
0	1	1	0	0	0	0	
0	1	1	0	1	1	1	
0	1	1	1	0	1	0	
0	1	1	0	1	0	0	
0	1	1	0	0	0	0	
0	1	1	0	0	1	1	
0	1	1	0	1	1	1	

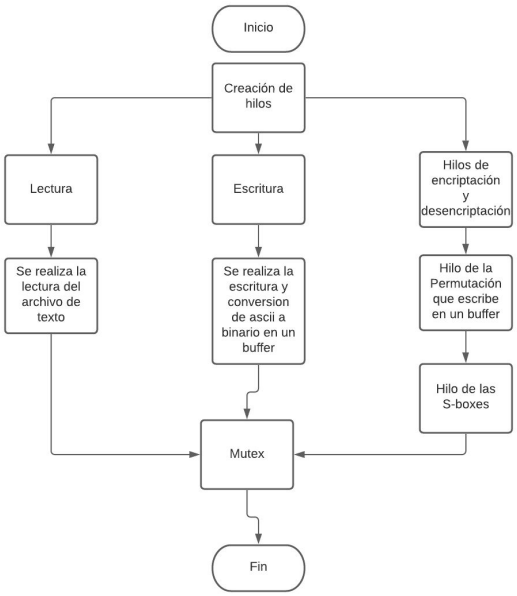
Permutación PC1						
0	0	0	0	0	0	0
0	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	0	0	0	0	0
1	1	0	0	0	1	0
1	1	1	0	0	1	1
0	0	1	0	0	1	0
1	0	0	1	0	0	1

(Rafael Net, 2011)
Imagen donde se puede observar cómo termina la clave luego de pasarla a 56 bits y realizar la permutación PC1.



(Sánchez, 1999)
Diagrama donde se puede observar el mecanismo de rondas de la clave para obtener la clave de 48 bits en cada una y seguir realizando las rotaciones circulares.

Diagrama de Mecanismos Paralelos y de Sincronía



Explicación de Mecanismos Paralelos y de Sincronía

Para la realización del programa se implementaron mecanismos de paralelismo y sincronía como pthreads y mutex. Se utilizaron pthreads en la creación de hilos para cada proceso, se utilizó uno para lectura del documento de texto, otro para llenar el buffer con los bits del texto ingresado y posteriormente se utilizó un hilo para cada proceso y operación, como para las permutaciones y las S-boxes. Posteriormente se utilizó Mutex para sincronizar los procesos y evitar que los hilos crearan algún conflicto al ejecutar su respectivo proceso.

Conclusión y Discusión sobre mejoras

Está claro que a través del algoritmo propuesto se puede obtener el cifrado de un texto y a través de los los mecanismos paralelos y de sincronía utilizados se pudo aumentar el desempeño del algoritmo.

Por otro lado, a futuro se podría aumentar el cifrado del texto utilizando diversas llaves y a través de las habilidades adquiridas para la programación paralela se podrían trabajar mutuamente y determinar en qué momentos se puede aplicar este tipo de programación para que no existan conflictos al momento de ejecutar el algoritmo.

Bibliografía

- Rafael Net (2011) Explicación del Cifrado en Bloques Simétrico DES, Extraido de, https://www.monografias.com/trabajos20/cifrado-en-bloques/cifrado-en-bloques.shtml

- Suárez, F. P., & Coruña, A. Estudio del algoritmo de cifrado Rijndael. Comparativa entre los algoritmos de cifrado DES y Rijndael.

- Jorge Sanchez (1999) Descripcion del algoritmo DES (Data Encryption Standard), Extraido de, http://www.satorre.eu/descripcion_algoritmo_des.pdf

- Ferguson, N., Schneier, B., & Kohno, T. (2010). Cryptography engineering. *Design Princi*.

- Cristoff Par Jan Pelzl (1998) Understanding Cryptography, Extraido de, http://swarm.cs.pub.ro/~mbarbulescu/cripto/Understanding%20Cryptography%20by%20Christof%20Paar%20.pdf

- Keith W. Campbell, Michael J. Wiener: DES is not a Group. CRYPTO 1992: pp512–520

- Don Coppersmith. (1994). The data encryption standard (DES) and its strength against attacks. *IBM Journal of Research and Development*, **38**(3), 243–250.