Universidad del Valle de Guatemala
Departamento de Ciencias de la Computación
Programación de Microprocesadores
Juan Zelada

Proyecto 02:
Catálogo Descriptivo

Jorge Súchite 15293
Alexa Bravo, 18831
Joséi Tzoc, 13322

Rondas Feistel

- Es de tipo void, la parte cifrada se realizará en un archivo global dentro de la función.
- Nombre de la función: RFeistel.
- Los parámetros de entrada son de tipo void ya que el archivo que lee es global.
- No tienen un uso específico los parámetros de entrada, sin embargo utilizará el archivo global para obtener la cadena de bits.
- En esta función, se divide el bloque inicial en dos partes: izquierda (L) y derecha (R). A la parte derecha se le aplica una función f que aporta la difusión adecuada. En esta función un lugar importante lo ocupa la clave, ésta debe permanecer en secreto y sólo la deben conocer el emisor y el receptor del mensaje.
- Esta función no tiene retorno, sólo lo modifica.

DES

- Esta función retorna un string de 32 bits.
- Nombre de la función: FDes.
- Tiene dos parámetros de entrada:
    - Clave: String 32 bits.
    - Cadena: String 48 bits.
- El parámetro de entrada sirve para obtener la cadena de 32 bits que usa la ronda Feistel
- El resultado de esta función es aplicado a la parte izquierda del bloque mediante un XOR.
- Retorna una variable con el texto encriptado:
    - Mensaje: String 32 bits.

Divide el Texto

- Es una función de tipo void
- Nombre de la función: DivideT
- Tiene un parámetro de entrada:
    - Parte: 32 bits.
- Se divide en dos partes el parámetro de entrada.
- En esta función, se divide el bloque inicial en dos partes: izquierda (L) y derecha (R).
- Esta función no tiene retorno, sólo lo divide.

Descripción de la sección del algoritmo general donde se usará cada función.

Texto plano

Definición del bloque de entrada de 64 bits definida en el programa en una matriz de nxn. Esto con el fin de ver cómo se va a hacer todo el trasfondo y lo primero antes de encriptar.

Entonces, debido a que el texto plano . se le va a pedir el mensaje al usuario para luego dividirla en una clave de 8 bytes esto con el fin de también quitar los bits menos significativos en nuestro proceso.

Se definirá un diccionario de esta envergadura con el fin de convertir decimal a hexadecimal para mayor facilidad de interpretación de la computadora. Para luego, después de la conversión.

```
mp['0']= "0000";
mp['1']= "0001";
mp['2']= "0010";
mp['3']= "0011";
mp['4']= "0100";
mp['5']= "0101";
mp['6']= "0110";
mp['7']= "0111";
mp['8']= "1000";
mp['9']= "1001";
mp['A']= "1010";
mp['B']= "1011";
mp['C']= "1100";
mp['D']= "1101";
mp['E']= "1110";
mp['F']= "1111";
```

```
mp["0000"]= "0";
mp["0001"]= "1";
mp["0010"]= "2";
mp["0011"]= "3";
mp["0100"]= "4";
mp["0101"]= "5";
mp["0110"]= "6";
mp["0111"]= "7";
mp["1000"]= "8";
mp["1001"]= "9";
mp["1010"]= "A";
mp["1011"]= "B";
mp["1100"]= "C";
mp["1101"]= "D";
mp["1110"]= "E";
mp["1111"]= "F";
```

Luego de realizar las permutaciones ya antes explicadas y las claves que se realizarán en cada permutación juntamente con cada ronda de permutación que tenga en su arquitectura las rondas feisel. definimos funciones tales como:

int i, x -contadores que servirán tanto para definir las rondas y las permutaciones
chart str[] - definición del parámetro del límite del mensaje a encriptar

variable c - tipo string donde guardará el mensaje y a la par se imprimirá un mensaje en donde se le pida al usuario ingresar mensaje.

name matrix1-2-3-4-5-6-7-8-9-10-11-12[][] - matriz donde se guardarán las permutaciones realizadas por nuestro algoritmo

función permutar (string , int array, int x) hace la permutación
función shift left (string k , int shifts)
función que permuta hacia la izquierda

función shift right(string k , int shifts)
función que permuta hacia la derecha

función xor (string a, string b)
suma los bits en la permutación

función encrypt(scring t, vector<string> rkt)
función encriptadora de hexa a dec con el fin de hacer la permutación
utiliza las funciones anteriores mencionadas

cout << "primera permutacion" + encrypt>>;

función x= xorK para las keys con fin de realizar la key 1 hasta la n permutación

string combi = left + right función que combina el texto o números encriptados después del shift left and right habrán más de estas funciones las veces que sea necesaria o permutaciones que se van a hacer.

variable key esta variable empezará con una key random en la cual se debe realizar la primera permutación, luego cambiará según la permutación en la que esté

keyp = permute(key, key1); variable que hace permutaciones con las keys con el fin de realizarlo aparte que el mensaje a encriptar.

int printf(round1,round2,round3, round4, round5 ……); printeará todas las rondas hechas por el algoritmo y la última ronda será guardada en un txt y printeada en el programa.

Nota: Esto no está escrito en piedra y puede que esta descripción sea modificada en la entrega final del proyecto, tanto variables como las definiciones y cantidad de rondas juntamente con rondas feisel.