

UNIVERSIDAD DEL ISTMO

FACULTAD DE INGENIERÍA

INFORME ALGORITMO DES

SISTEMAS OPERATIVOS

KEVIN ALEXANDER CHAMPNEY LEÓN

ELISA MARGARITA MONZÓN GODOY

Guatemala, 24 de septiembre de 2019

ÍNDICE

[INTRODUCCIÓN 1](#_Toc20145286)

[CONTENIDO 2](#_Toc20145287)

[Descripción de Algoritmo DES 2](#_Toc20145288)

[Principio de funcionamiento 3](#_Toc20145289)

[Diagrama de flujo 3](#_Toc20145290)

[Lógica descriptiva 4](#_Toc20145291)

[Código descriptivo 4](#_Toc20145292)

[Rondas de Feistel 4](#_Toc20145293)

[Algoritmo 5](#_Toc20145294)

[Variables mutex y condicionales 5](#_Toc20145295)

[Mutex 5](#_Toc20145296)

[Condicionales 5](#_Toc20145297)

[Catálogo de funciones principales de programa 5](#_Toc20145298)

[CONCLUSIONES 6](#_Toc20145299)

[BIBLIOGRAFÍA 7](#_Toc20145300)

# INTRODUCCIÓN

# CONTENIDO

## Descripción de Algoritmo DES

DES, Data Encryption Estándar[[1]](#footnote-1), es un esquema de encriptación simétrico, desarrollado en 1977 por el Departamento de Comercio y la Oficina Nacional de Estándares de Estados Unidos, en colaboración con la empresa IBM. Fue creado con el objeto de proporcionar al público en general un algoritmo de cifrado normalizado para las redes de ordenadores, que contara con los siguientes requisitos:

* Ofrecer un alto nivel de seguridad relacionado con una pequeña clave utilizada para cifrado y descifrado
* Comprensible
* No depender de la confidencialidad del algoritmo
* Adaptativo y económico
* Eficaz y exportable

El algoritmo se basa en un sistema mono alfabético, con un algoritmo de cifrado consistente en la aplicación sucesiva de varias permutaciones y sustituciones. Inicialmente el texto en claro a cifrar se somete a una permutación, con un bloque de entrada de 64 bits, para posteriormente ser sometido a la acción de dos funciones principales, una función de permutación con entrada de 8 bits y otra de sustitución con entrada de 5 bits, en un proceso que consta de 16 etapas de cifrado.

## Principio de funcionamiento

Se trata de un sistema de cifrado simétrico por bloques de 64 bits, de los que 8 bits se utilizan como control de paridad (para la verificación de la integridad de la clave). Cada uno de los bits de la clave de paridad se utiliza para controlar uno de los bytes de la clave por paridad impar, es decir, que cada uno de los bits de paridad se ajusta para que tenga un número impar dentro del byte al que pertenece. Por lo tanto, la clave tiene una longitud útil de 56 bites.

El algoritmo se encarga de realizar combinaciones, sustituciones y permutaciones entre el texto a cifrar y la clave, asegurándose al mismo tiempo de que las operaciones puedan realizarse en ambas direcciones (para el descifrado). La combinación entre sustituciones y permutaciones se llama cifrado del producto.

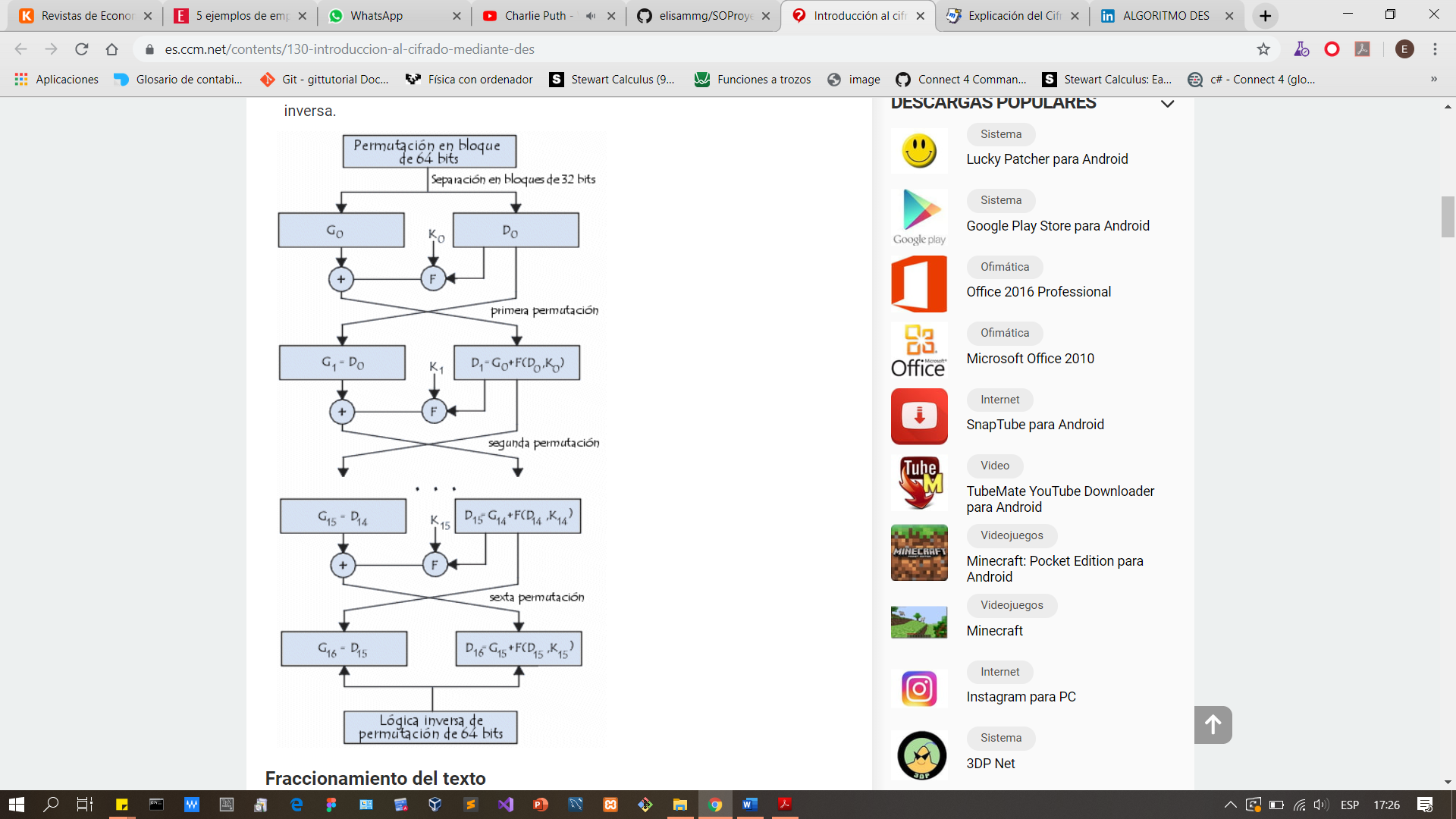
La clave es codificada en 64 bits y se compone de 16 bloques de 4 bits, generalmente anotadas de *k1* a *k16*. Dado que "solamente" 56 bits sirven para el cifrado, ¡puede haber hasta 256 (o 7.2\*1016) claves diferentes

## Diagrama de flujo

Las partes principales del algoritmo constan de:

* Fraccionamiento del texto en bloques de 64 bits
* Permutación inicial de los bloques
* Partición de los bloques en dos partes:
  + Izquierda 🡪 I
  + Derecha 🡪 D
* Fases de permutación y de sustitución repetidas 16 veces (rondas)
* Reconexión de las partes izquierda y derecha, seguida de la permutación inicial inversa.

Imagen No.1: Diagrama algoritmo DES



Fuente: CCM, 2019.

## 

## Rondas de Feistel

El cifrado de Feistel es un método de cifrado en bloque, con una estructura particular, y es implementado por el algoritmo Data Ecnryption Estándar, ya que presenta la ventaja de ser reversibles por lo que las operaciones de cifrado y descifrado son idénticas, requiriendo únicamente invertir el orden de las subclaves utilizadas.

### Algoritmo

Este [algoritmo](https://es.wikipedia.org/wiki/Algoritmo) se denomina simétrico por rondas, es decir, realiza siempre las mismas operaciones un número determinado de veces (denominadas rondas). Los pasos de la red de Feistel son entre algunos más:

1. Se selecciona una cadena, N, normalmente de 64 o 128 bits, y se la divide en dos subcadenas, L y R, de igual longitud (N/2)
2. Se toma una [función](https://es.wikipedia.org/wiki/Funci%C3%B3n_matem%C3%A1tica), F, y una clave Ki
3. Se realizan una serie de operaciones complejas con F y Ki y con L o R (solo una de ellas)
4. La cadena obtenida se cambia por la cadena con la que no se han realizado operaciones, y se siguen haciendo las rondas.

## Variables mutex y condicionales

### Mutex

Las variables mutex son un mecanismo de sincronización entre hilos, que ayudan a proteger una sección critica del código. Estas variables son de tipo pthread\_mutex\_t y los métodos para emplear los mutex son los siguientes:

* int pthread\_mutex\_init(pthread\_mutex\_t \*mutex, pthread\_mutexattr\_t \* attr);
  + Inicializa el mutex
* int pthread\_mutex\_destroy(pthread\_mutex\_t \*mutex) ;
  + Destruye el mutex (lo elimina de la memoria).
* int pthread\_mutex\_lock(pthread\_mutex\_t \*mutex);
  + Bloquea el mutex si no lo tiene nadie. Si alguien tiene bloqueado el mutex el proceso espera hasta que el proceso que lo tiene bloqueado lo libera.
* int pthread\_mutex\_unlock(pthread\_mutex\_t \*mutex);
  + Libera el mutex.

### Condicionales

Las variables condicionales son variables de sincronización asociadas a un mutex que ayudan a sincronizar nuestros bloques entre hilos. Por ejemplo, realizamos un bloqueo, pero tenemos que esperar a que otro recurso realice una acción, con estas variables podemos esperar dicha acción. Estas variables pueden realizar principalmente dos acciones: *wait* y *signal* y conviene realizarlas entre *lock* y *unlock* de un mutex. Estas variables son de tipo pthread\_cond\_t y los métodos para emplear los mutex son los siguientes:

* int pthread\_cond\_init(pthread\_cond\_t\*cond, pthread\_condattr\_t\*attr);
  + Inicializa la variable condicional.
* int pthread cond destroy(pthread\_cond\_t \*cond);
  + Destruye la variable condicional.
* int pthread\_cond\_wait(pthread\_cond\_t\*cond, pthread\_mutex\_t\*mutex);
  + Mediante este método suspendemos el hilo en el que se ejecuta y liberamos el mutex. El proceso queda suspendido hasta que recibe una señal de activación. Una vez que se activa se vuelve a luchar por el control del mutex.
* int pthread\_cond\_signal(pthread\_cond\_t \*cond);
  + Se reanuda la ejecución de uno o más hilos que estén esperando por la variable condicional (cond).
  + En caso de no haber ningún proceso esperando no ocurre nada.
* int pthread\_cond\_broadcast(pthread\_cond\_t \*cond);
  + Se reanudan todos los hilos que estaban suspendidos por la variable condicional (cond).

## Catálogo de funciones principales de programa

Las principales funciones que utilizaremos para el programa de encripción y desencripción son las siguientes:

* Declaración de variables
  + pthread\_mutex\_t mutex
  + pthread\_cond\_t cond
  + pthread\_t thread
* Inicialización de funciones
  + pthread\_mutex\_init(&mutex, NULL)
  + pthread\_cond \_init(&cond, NULL)
  + pthread\_create(&thread, NULL, subrutina, NULL)

# 

# CONCLUSIONES

# BIBLIOGRAFÍA

<http://neo.lcc.uma.es/evirtual/cdd/tutorial/presentacion/des.html>

<https://es.ccm.net/contents/130-introduccion-al-cifrado-mediante-des>

1. En castellano: Algoritmo de Cifrado de Datos [↑](#footnote-ref-1)