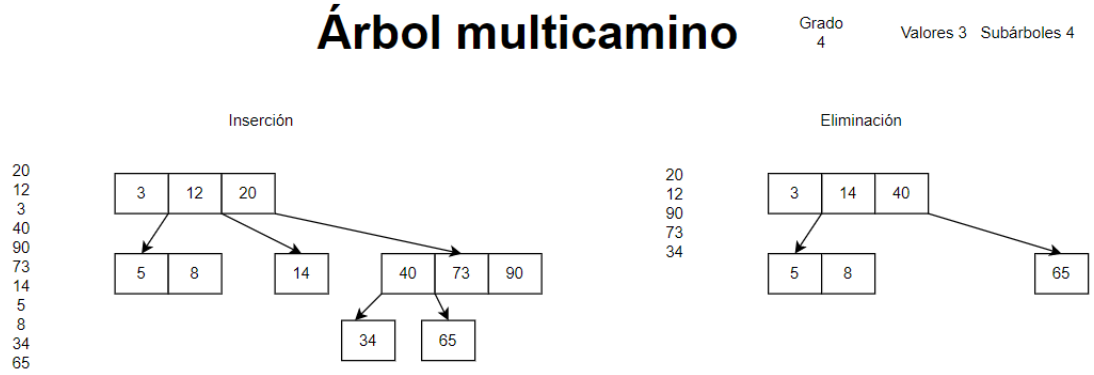
Árbol Multicamino

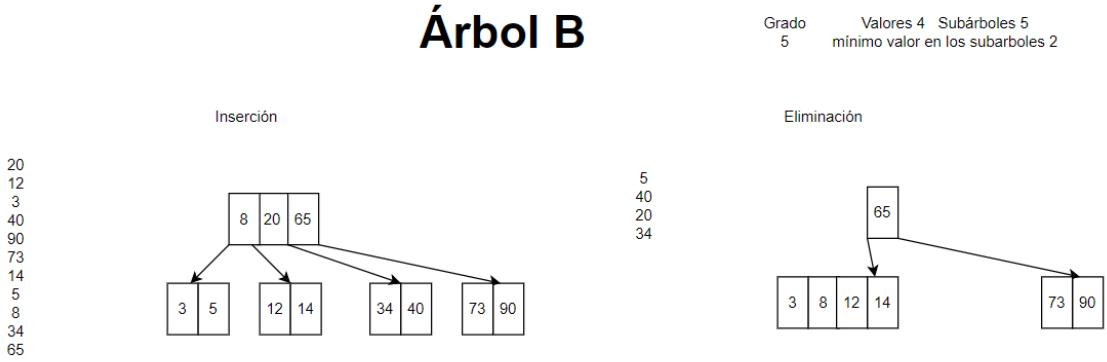
→ LogmN - Niveles para insertar N valores en un árbol de grado m.
→ Valores en cada nodo *m – 1*
→ Cantidad de subárboles *m*
→ $m^x - (m-1)$ → cantidad de nodos de un árbol de grado m en x niveles.
→ $(m-1)(m^x - (m-1))$ → cantidad de valores de un árbol de grado m en x niveles (máximo).

Ejemplo



Árbol B

→ Se autobalancean.
→ Ideal utilizar grados impares
→ Se inserta en las hojas.
→ Mínimo $\frac{m}{2} - 1$ valores y $\frac{m}{2}$ subárboles
→ Máximo *m - 1* valores y *m* subárboles.
→ El árbol B mejora respecto al



Árbol B+

→ Todos los valores están en las hojas.
→Grados pares para que la división de los nodos sea más sencilla
→ Copia el valor solo cuando las hojas se separan.
→ Los valores de nodos intermedios son copias de los valores originales (índices).
→ Todos los nodos hoja están encadenados.
Eliminación: cuando la hoja está en underflow se elimina la raíz en común y se unen los 2 nodos. (Si es necesario, se vuelven a separar).
→ Ventaja respecto a un B*: Permite búsquedas secuenciales entre rangos

Árbol B*

→ En los valores que **NO SON LA RAÍZ** $\frac{2m-1}{3}$ valores.

→ En **LA RAÍZ** $\frac{4}{3}(m-1)$ valores como máximo.
Inserción: unir 2 nodos y 1 raíz (escoger valor mínimo y resultan 3 nodos)
Eliminación: unir 3 nodos y 2 raíces (escoger valor máximo y resultan 2 nodos).

Compresión

→ Se puede utilizar compresión con pérdida para imágenes, pero no para textos
→ Entropía de la información: Cantidad de información promedio que contienen los símbolos usados
→ Cuando todos los símbolos son igualmente probables (distribución de probabilidad plana), todos aportan información relevante y la entropía es **máxima**
→ La razón de compresión debe ser menor a 1
→ El factor de compresión debe ser mayor a 1
→ Lo que intenta remover la compresión es la redundancia.

Aritmético

Compresión
$(L_s - L_i)\ P + V_{anterior}$
Descompresión
→ Se hace el mismo proceso, pero se va buscando el intervalo en el que se encuentra en el que se encuentra el número comprimido (El resultado de la compresión)
→ Ordenar de mayor a menor

Huffman

→ Menores a la derecha
→ Mayores a la izquierda
→ Se inserta el nodo resultante en la cola de prioridad (De menor a mayor)
→ Izquierda = 0
→ Derecha = 1
→ Se asigna el valor a cada letra
→ Se cifra la palabra
→ Se dividen en 8 y se saca el valor en decimal, se coloca el valor del código ASCII
→ Se hace el árbol con la metadata
→ Se toma el primer byte y se buscan las coincidencias

Valor	Cantidad	Frecuencia

LZW

→ Buscan bytes repetidos en una ventana corrediza

- Lempel
- Ziv
- Storer
- Swymanky

→ Encontrar la cadena "S" más larga que exista en el diccionario + un símbolo o la cadena más corta que no exista.
→ Para la descompresión se queda la tabla original.
→ El valor más grande del índice determina la cantidad de bits que tendrá cada número

LZ77

→ Creado en 1977
→ Base de LZW, GIF, PNG, ZIP
→ Se basa en la repetición de frases (dos o más caracteres)
→ Cuando ha repetición se usa un apuntador a la última vez que aparece la frase
→ Crea un diccionario **adaptativo.**

Diccionario: Estructura de datos que utiliza una llave para obtener un valor

→ Son mejores con entradas más grandes y con muchas repeticiones.

Cadena	Dirección	Longitud	Nuevo Símbolo
holacomohalo	0	0	o

LZ78
→ Se queda la compresión del carácter y se decide si se quiere o no hacer algo con él.

Salida	Index	Cadena
0, c(carácter)	1	a

Cifrados

Transposición

- Una clave úncia
- Emisor
- Receptor

Cesar

→ Una palabra desorganiza el abecedario

ZigZag

→ Se necesita un número n, para determinar la cantidad de niveles.

→ Se debe quedar en el punto antes de volver a empezar

Ruta

→ Vertical o Espiral

→ Mensaje en una matriz de n x m

Vertical (Cifrado) – Se llena verticalmente y se recorre en horizontal.

Vertical (Descifrado) – Se llena horizontalmente y se recorre en vertical.

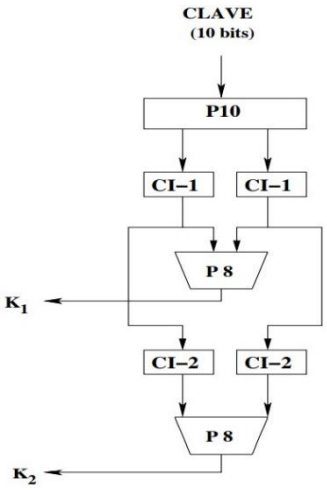
Espiral (Cifrado) – Se llena en espiral y se recorre en horizontal.

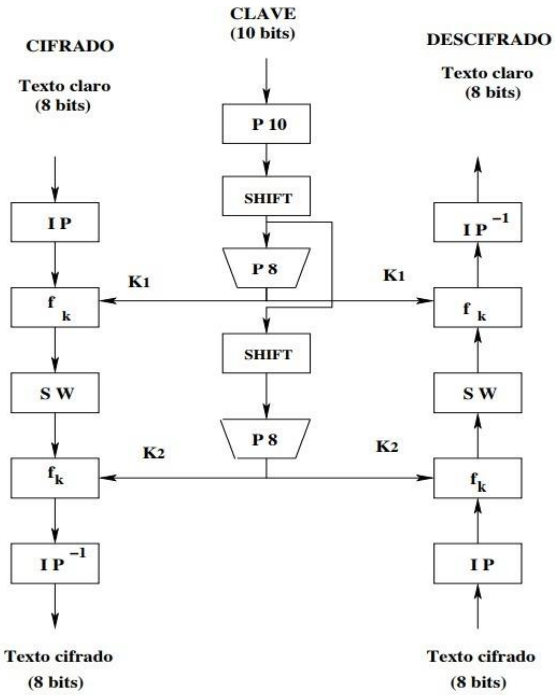
Espiral (Descifrado) – Se llena horizontalmente y se recorre en espiral.

SDES

	DES	SDES
Entrada	64 bits	8 bits
Salida	64 bits	8 bits
Clave	56 bits	10 bits
Rondas	16	2
F operaciones	32 bits	4 bits
S_boxes	8	2

CI- 1 -> Corrimiento Izquierda
CI -1 -> Corrimiento 2 Izquierda





S_1	**00**	01	10	11
00	00	01	10	11
01	10	00	01	11
10	11	00	01	00
11	**10**	01	00	11

S_0	00	01	**10**	11
00	01	00	11	10
01	11	10	01	00
10	00	10	01	11
11	11	01	**11**	10

Cifrado simétrico

Claves (Llaves)

- Son de información
- Secuencias de números o letras
- Permitir la autorización de acceso a un servicio o sistema
- Debe permanecer “secreta”

Tipos

- Key – Una palabra
- Passphase – Una frase

Características

- Longitud → Fuerza bruta
- Aleatoriedad → Evita ataques de Diccionario
- Período de uso → Cambio constante de contraseña reduce el riesgo de ataques

Sal (Salt)

Bits aleatorios

Es un valor que se agrega a la función de cifrado

De una sola clave

- Clave secreta o única: Acuerdo previo para definir la clave

Principios de Kerchoff

- Sistema no es irrompible
- Efectividad no depende que el diseño sea secreto
- La clave debe ser memorizable
- Los ciptogrmas deber resultar en valores alfanuméricos
- Sistema debe ser operable por 1 persona
- Sistema debe ser fácil de usar

LA SEGURIDAD ESTÁ EN LA CLAVE

RSA

Generación de llaves

p	53	q	73	
n = p*q				3869
phi = (p-1)(q-1)				3744
e	Es primo			23
	1 < e < phi			
	No debe ser factor de phi, ni n			
d	2279			

65

Cifrar	c^e mod n	3440
Descifrar	x^d mod n	65

	phi	phi
	e	1
	3744	3744
162	23	1
1	18	3582
3.6	5	163
1	3	3093
1	2	814
	1	2279