Universidad Diego Portales

# Laboratorio 1: Protegiendo Mensajes en una Red Segura con Cifrado Vigenère

Estructuras de Datos y Algoritmos
2025-1

Profesores:
Valentina Aravena
Cristián Llull
Marcos Fantoval

# 1. Introducción

En una empresa de telecomunicaciones, los empleados necesitan enviar mensajes cifrados para evitar que información confidencial sea interceptada por terceros. Sin embargo, debido a la sensibilidad de la información, las claves utilizadas para el cifrado son extremadamente largas, tanto que no pueden almacenarse en una sola variable convencional.

Para resolver este problema, usted debe implementar un sistema de cifrado basado en Vigenère, en el cual la clave será tratada como un número de gran tamaño almacenado en un arreglo de tamaño fijo, y el alfabeto estará representado en una matriz bidimensional.

# 2. Objetivo

Implementar una clase BigVigenere en Java que permite cifrar y descifrar mensajes utilizando el cifrado de Vigenère con una clave numérica extremadamente larga, almacenada en un arreglo de tamaño fijo, y un alfabeto representado en una matriz.

# 3. Especificación de la clase BigVigenere

Usted tendrá que implementar la clase BigVigenere con los siguiente **atributos**.

- `int[] key`: Arreglo de enteros que almacena la clave numérica de gran tamaño.
- `char[][] alphabet`: Matriz bidimensional que representa el alfabeto utilizado en el cifrado, compuesto por caracteres alfanuméricos (`'a' - 'z'`, `'A' - 'Z'` y `'0' - '9'`) [1].

Luego tendrá que implementar los siguientes métodos:

- `public BigVigenere()`: Constructor que solicita al usuario una `key` para luego guardar el el arreglo númerico correspondiente al atributo `key`. También genera la matriz del alfabeto para el algoritmo Vigenère.
- `public BigVigenere(String numericKey)`: Constructor que recibe la clave en forma de String y luego la transforma a la clave númerica correspondiente para almacenarla en el atributo `key`. También genera la matriz del alfabeto para el algoritmo Vigenère.
- `public String encrypt(String message)`: Método que cifra un mensaje utilizando la clave numérica almacenada y la matriz del alfabeto usando el algoritmo de Vigenère.
- `public String decrypt(String encryptedMessage)`: Método que descifra un mensaje cifrado con la misma clave numérica usando Vigenère.
- `public void reEncrypt()`: Método que permite cambiar la clave utilizada en el cifrado. Con el siguiente paso a paso:
  - Se debe solicitar el mensaje encriptado.
  - Se debe descifrar el mensaje actual con la clave actual (como input desde la terminal).
  - A continuación se debe solicitar la nueva clave.
  - Luego se debe cifrar con la nueva clave.
  - Por último se imprime el nuevo mensaje.
- `public char search(int position)`: Método que busca la letra correspondiente a la posición buscada, se realiza una búsqueda iterativa de izquierda a derecha, arriba hacia abajo. Retorna el carácter de la posición buscada.

[1] El abecedario considera la letra ñ|Ñ, teniendo un total de 64 carácteres en total.

- `public char optimalSearch(int position)`: Método que realiza la búsqueda del carácter correspondiente de acuerdo a la posición indicada. Se busca realizar una búsqueda más eficiente que el caso anterior.

# 4. Experimentación

Para la experimentación tendrá que seguir los siguientes pasos.

1. **Diseñar y ejecutar** pruebas unitarias para verificar que los métodos encrypt y decrypt funcionen correctamente.
2. **Implementar** la clase solicitada BigVigenere con sus atributos y métodos.
3. **Evaluar y graficar** el tiempo de ejecución del cifrado y descifrado con claves de distintos tamaños `L = {10, 50, 100, 500, 1000, 5000}` y un mismo mensaje (El mensaje a cifrar debe tener al menos 10000 carácteres).
4. **Comparar la eficiencia** del cifrado a medida que aumenta el tamaño de la clave. Analizar el tiempo de ejecución con la notación Big O.

# 5. Informe

Dentro del informe tendrá que incluir las siguientes secciones:

1. **Implementación**: Realizar una descripción detallada de la implementación del constructor (método `public BigVigenere`), donde se presente la construcción final de la matriz `alphabet` y el paso a paso para la construcción del atributo `key`.
2. **Métodos**: Descripción de los métodos implementados, y explicación de cómo se óptimizó la búsqueda de la posición en `optimalSearch(int position)`.
3. **Experimentación y Resultados**: Descripción de los experimentos realizados, se puede presentar el mensaje a cifrar. Posteriormente se debería mostrar el gráfico de tiempo de ejecución, con la comparación de eficiencia con la notación Big O.
4. **Conclusiones**: Realizar observaciones con respecto al laboratorio, respondiendo las siguientes preguntas: £Cuales dificultades se presentaron y cómo se resolvió? £Cómo se logró optimizar la búsqueda de las posiciones?, en relación con el tamaño de la clave a utilizar £qué recomendaciones entregaría para la creación de la clave?.

# 6. Ayudas

## 6.1. Creación de la matriz `alphabet`

Para el cifrado de Vigenère se puede utilizar **opcionalmente** una matriz de 64x64, de tal forma que esta podrá verse de la siguiente manera:[2]

$$\begin{bmatrix} a & b & c & d & e & f & ... \\ b & c & d & e & f & g & ... \\ c & d & e & f & g & h & ... \\ d & e & f & g & h & i & ... \\ e & f & g & h & i & j & ... \\ f & g & h & i & j & k & ... \\ ... & ... & ... & ... & ... & ... & ... \end{bmatrix}$$

Así, si se tiene una `key` y un `mensaje`, la primera letra para el mensaje encriptado corresponderá al carácter en la posición `l = alphabet[posición_carácter_mensaje][posición_carácter_key]`

[2]Note que la primera fila contiene todo el vocabulario de a-z, A-Z y 0-9, luego la siguiente fila tendrá el vocabulario de b-z, A-Z, 0-9 y la letra "a" al final.