# Criptografía

# Grupo 09

---

# Laboratorio 1

---

## Integrantes


| Nombre | CI | Correo |
|---|---|---|
| Germán Ouviña | 4.823.566-1 | german.ouvina@fing.edu.uy |

# Índice

# 1. Introducción

El siguiente documento es un informe de las actividades realizadas en el contexto del **Laboratorio 1** del curso de Criptografía de Fing, UdelaR, edición 2019. Dicho laboratorio tiene como objetivo generar un conjunto de herramientas de asistencia para el criptoanálisis de distintos cifrados y emplearlas para encontrar las claves de cifrados de ejemplo.

El informe se compone de las siguientes secciones:

- Una sección **Desarrollo**, donde se tratan los puntos principales del laboratorio, dividiéndose en las subsecciones **Diseño**, donde se trata el enfoque utilizado al desarrollar cada herramienta, y **Análisis**, donde se emplea las herramientas desarrolladas para analizar los cifrados pertinentes,
- Una sección **Guía de Uso**, donde se especifíca como correr cada herramienta en particular.
- Una sección **Referencias**, donde se lista la bibliografía y sitios consultados durante el proceso de desarrollo.

# 2. Desarrollo

## 2.1. Parte 1 - Diseño de herramientas

La primera parte del laboratorio consistió en desarrollar distintas herramientas de utilidad para el criptoanálisis de los criptosistemas *Shift*, *Afín*, *Vigenere* y *Sustitución*.

Cabe mencionar que se tomaron ciertas presunciones en lo que concierne a todas las herramientas en general. A continuación, un resumen de dichas consideraciones:

**Especificaciones Técnicas:** Se utilizó el lenguaje de programación *Python* para desarrollar las herramientas, utilizando *Python 3.7* para su ejecución. A su vez, se utilizaron bibliotecas de *Python* nativas como *math* y *operator*, y la biblioteca *NumPy* que debe ser instalada de forma externa.

**Especificaciones Generales:** Las herramientas se desarrollaron de forma tal que el **mapa** o **alfabeto** a utilizar sea "portable", es decir, al definir un archivo *mapa* y pasarlo a la herramienta en cuestión, esta realizará el análisis considerando que el texto se compone de caracteres pertenecientes a dicho alfabeto. De esta forma, es posible utilizar las herramientas para analizar textos con otros caracteres, siempre y cuando se genere el archivo *mapa* correspondiente. Dicho esto, se utilizó un alfabeto del español, colocando a la letra ñ al final del mismo (luego de la z). El archivo se encuentra en la carpeta principal bajo el nombre *"texto_mapa.txt"*.

**Especificaciones Estructurales:** Además de definir un módulo (es decir, un archivo) por herramienta, se utilizaron módulos auxiliares para definir funciones de manejos de archivos, de operaciones matemáticas y de funciones auxiliares utilizadas en más de una herramienta. Esta modularización se realizó con el objetivo de aumentar la reutilización de código y simplificar su lectura.

A continuación se repasa brevemente el objetivo y los lineamientos seguidos en el diseño de cada herramienta:

### 2.1.1. Análisis de frecuencias

Recibiendo como parámetro un *texto* y un *mapa*, la herramienta *frecuencias.py* determina la frecuencia de cada letra, digrama y trigrama presente en *texto* y perteneciente a *mapa*, calculando su cantidad de ocurrencias y mostrando una lista ordenada descendentemente por frecuencia y alfabéticamente.

### 2.1.2. Análisis de índice de coincidencia

Recibiendo como parámetro un *texto* y un *mapa*, la herramienta *coincidencia.py* determina el **índice de coincidencia** de *texto*, utilizando el contador de letras definido para *frecuencias*.

### 2.1.3. Análisis de índice de coincidencia por bloques

Recibiendo como parámetro un *texto*, un *mapa* y un *largo de clave* $n$, la herramienta *coincidencia_bloques.py* genera $n$ bloques vacíos y a partir de *texto*, múltiples particiones de largo $n$, tomando luego la i-ésima letra de cada partición y agregandola al i-ésimo bloque. Luego, determina el **índice de coincidencia** para cada bloque, utilizando la fórmula definida para *coincidencia*.

### 2.1.4. Análisis de índice de coincidencia mutua

Recibiendo como parámetro un *texto*, un *mapa* y un *largo de clave* $n$, la herramienta *mutua.py* genera $n$ bloques utilizando la misma lógica que en *coincidencia_bloques*. Luego, determina el **índice de coincidencia mutua** para cada letra en cada bloque, utilizando la fórmula definida en la página 35 de [1]. De esta forma, permite determinar una posible clave de largo $n$, eligiendo para cada bloque, la letra con mayor índice.

### 2.1.5. Test de Kasiski

Recibiendo como parámetro un *texto* y un *mapa*, la herramienta *kasiski.py* corre el **test de kasiski** para *texto*, devolviendo el largo de clave posible para *texto*. Es **crucial** destacar que se devuelve el máximo común divisor entre todas las distancias del trigrama encontrado, lo cual representa una pérdida de información (en la sección de análisis se expande este punto).

### 2.1.6. Análisis de clave parcial

Recibiendo como parámetro un *texto*, un *mapa* y una *clave parcial*, la herramienta *sustitucion_clave_parcial.py* sustituye para cada letra presente en *texto* y en *clave parcial*, la letra correspondiente a la posición de cada letra examinada. En otras palabras, descifra *texto* utilizando *clave*, dejando aquellos caracteres que no puede descifrar.

## 2.2. Parte 2 - Criptoanálisis de cifrados

La segunda parte del laboratorio consistió en utilizar las anterior herramientas para analizar distintos cifrados y averiguar su clave.

A continuación se repasa brevemente el proceso seguido al descifrar cada cifrado:

### 2.2.1. Cifrado Shift

Se atacó el problema con los siguientes pasos:

1. Se utilizó *frecuencias* para encontrar las letras con mayor frecuencia en el texto cifrado.
2. Evaluando las frecuencias del idioma español y las frecuencias del texto tanto en letras como en digramas y trigramas, se determinó que la letra **M** en el cifrado correspondía a la letra **E** o **A**.
3. Tomando **M** como *E*, la distancia entre ambas es de 8. Al probar la clave, funcióno.

Se determinó que la clave es **8**.

### 2.2.2. Cifrado Afín

Se atacó el problema con los siguientes pasos:

1. Se utilizó *frecuencias* para encontrar las letras con mayor frecuencia en el texto cifrado.
2. Evaluando las frecuencias del idioma español y las frecuencias del texto tanto en letras como en digramas y trigramas, se determinó que las dos letras más frecuentesen el cifrado correspondían a las letras **E** y **A** (muy probablemente).
3. Estableciendo un sistema de congruencias con ambas presunciones, se encontró un par de claves 5 y 8. Al probar dicha clave, funcionó.

Se determinó que la clave es **5 8**.

### 2.2.3. Cifrado Vigenere

Se atacó el problema con los siguientes pasos:

1. Se utilizó *kasiski* para encontrar el posible largo de clave. El test retornó 1, pero al evaluar el máximo común divisor entre cada par de distancias, se obtuvieron los valores 7 y 2.
2. Se utilizó *coincidencia_bloques* para ver el índice de coincidencia para un largo de clave 2 y 7. Para un largo 7, el índice oscilaba en valores cercanos a 0.73, el índice de coincidencia del idioma español.
3. Se utilizó *mutua* con largo de clave 7 y se encontró la posible clave *gabriel*. Al probar dicha clave, funcionó.

Se determinó que la clave es **gabriel**.

### 2.2.4. Cifrado por Sustitución

Se atacó el problema con los siguientes pasos:

1. Se utilizó *frecuencias* para encontrar las letras con mayor frecuencia en el texto cifrado.
2. Evaluando las frecuencias del idioma español y las frecuencias del texto tanto en letras como en digramas y trigramas, se determinó aquellas letras que representaban **E** y **A**.
3. Se utilizó *sustitucion_clave_parcial* y a través de ensayo y error, se fue completando la clave, leyendo el texto parcialmente descifrado. Se empezó intentando posicionar la letra **L** antes de las **A** encontradas, así como se descifró que letra era **D** en base a suposiciones de las frecuencias.
4. Siguiendo con esa línea de razonamiento, se encontraron suficientes letras para encontrar palabras reconocibles, determinandose finalmente la clave parcial *nqgrtwovjk-sdxyfbelmcp-aui-* y descifrando el texto completamente. Al asignar h,z,ñ a los lugares vacíos y probar dicha clave, funcionó.

Se determinó que la clave es **nqgrtwovjkhsdxyfbelmcpzauiñ**.

# 3. Guía de uso

Para ejecutar cualquiera de las 6 herramientas, es necesario tener instalada una versión de *Python* superior a 3.6, así como la biblioteca *NumPy*, la cual puede ser instalada a través de un gestor de paquetes como *PIP*.

Tomando esto en cuenta, para ejecutar cada herramienta, es necesario correr el siguiente comando:

- Para *frecuencias*, se ejecuta: *python frecuencias.py texto.txt mapa.txt*, siendo *texto* el cifrado y *mapa* el archivo con el alfabeto.
- Para *coincidencia*, se ejecuta: *python coincidencia.py texto.txt mapa.txt*, siendo *texto* el cifrado y *mapa* el archivo con el alfabeto.
- Para *coincidencia_bloques*, se ejecuta: *python coincidencia_bloques.py texto.txt mapa.txt n*, siendo *texto* el cifrado, *mapa* el archivo con el alfabeto y $n$ el largo de clave a probar.
- Para *mutua*, se ejecuta: *python mutua.py texto.txt mapa.txt n*, siendo *texto* el cifrado, *mapa* el archivo con el alfabeto y $n$ el largo de clave a probar.
- Para *kasiski*, se ejecuta: *python kasiski.py texto.txt mapa.txt*, siendo *texto* el cifrado y *mapa* el archivo con el alfabeto.
- Para *sustitucion_clave_parcial*, se ejecuta: *python sustitucion_clave_parcial.py texto.txt mapa.txt clave.txt*, siendo *texto* el cifrado, *mapa* el archivo con el alfabeto y *clave* el archivo con la clave parcial en el formato indicado.

# 4. Referencias

[1] **Cryptography, Theory and Practice** - Douglas Stinson
La fuente principal de información para esta tarea, libro donde se especifícan las técnicas empleadas.

[2] **Kasiski Analysis: Breaking the Code** - https://crypto.interactive-maths.com/kasiski-analysis-breaking-the-code.html
Herramienta web que permite realizar test de kasiski y otros análisis útiles.