Presentación

En esta práctica nos familiarizaremos con el lenguaje de programación Python. Para ello implementaremos un criptosistema derivado de dos criptosistemas históricos: la Rejilla de Cardano y el cifrado de desplazamiento o de tipo César. La Rejilla de Cardano es un criptosistema que fue reinventado por Gerolamo Cardano hacia 1550 y usa la misma idea que la Máscara China, una técnica implementada en la antigua China. El cifrado de tipo César lo hemos estudiado en el módulo 1.

Objetivos

Los objetivos de esta práctica son:

1. Familiarizarse con el entorno de trabajo en Python y el *framework unittest*.
2. Implementar un criptosistema histórico.

Descripción de la Práctica a realizar

El funcionamiento del cifrado de desplazamiento o de tipo César lo podéis consultar en el módulo 1.

La Rejilla de Cardano (o Máscara China) consiste en el uso de una cartulina (papel, madera, etc) rectangular con agujeros en ciertas posiciones. Esta cartulina, que actúa como clave, cuando se coloca sobre un texto cifrado, muestra por los agujeros el texto en claro.

Para codificar un mensaje, se coloca la cartulina sobre un papel y se escribe el mensaje a cifrar en los agujeros de la cartulina. A continuación, se retira la cartulina y se rellena el texto faltante con texto un inocuo que no levante sospechas.

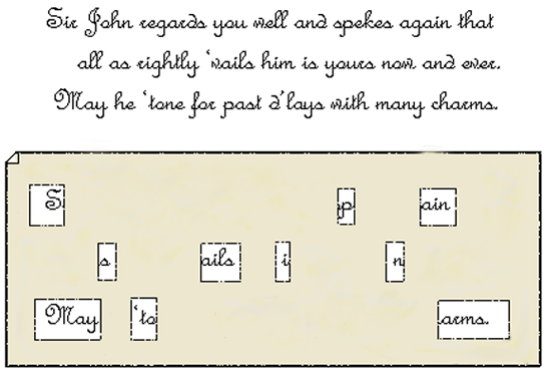

Figura 1: Imagen obtenida de Wikipedia

Para la práctica, vamos a usar un criptosistema que funciona como se explica a continuación:

1. Primero realizaremos un primer cifrado del texto usando el cifrado de desplazamiento, con el desplazamiento que se indique. El criptosistema final usará como valor del desplazamiento el número de agujeros en la rejilla. Es decir, que si la rejilla tiene 7 agujeros, el desplazamiento que se usará será de 7 posiciones.
2. A continuación, aplicaremos la Rejilla de Cardano. Para ello, no vamos a usar una rejilla de cartulina. En su lugar, usaremos una lista de Python que hará de máscara. Así, pondremos un 1 en las posiciones correspondientes a los agujeros y un 0 en el resto de posiciones. Las posiciones en las que hay un uno las elegiremos aleatoriamente como parte del proceso de generación de la clave.
3. Una vez disponemos de la clave de rejilla, podemos cifrar un texto colocándolo en las posiciones correspondientes a los agujeros (los unos). Si el texto es más largo que la rejilla, colocaremos de nuevo la rejilla a continuación. Así, tantas veces como sea necesario. El resto de posiciones, las que tienen un cero, serán rellenadas con caracteres aleatorios.
4. El cifrado con rejilla se realiza usando como entrada el texto cifrado usando el sistema de desplazamiento.
5. Para descifrar un mensaje, solo tendremos que colocar nuestra rejilla tantas veces como sea necesario, leyendo los caracteres que se corresponden con los agujeros de la rejilla. Después de aplicar la rejilla para obtener el mensaje, lo que tenemos es un mensaje cifrado con el sistema de desplazamiento. Por lo que para terminar de descifrarlo tendremos que aplicar el desplazamiento en sentido contrario.
6. En este criptosistema usaremos un alfabeto de 41 caracteres, que incluirán las 26 letras del abecedario en inglés más los 10 dígitos numéricos, del 0 al 9, así como los 4 caracteres siguientes: “.”,“,”,“:”,“?” y el espacio en blanco. Es decir, un total de 41 caracteres.

En esta práctica sólo podéis usar las librerías estándar de Python. Es decir, **no se permite el uso de librerías externas**. Para poder estar seguros de qué módulos forman parte de la librería estándar, podéis usar este enlace.

Para implementar el criptosistema dividiremos el trabajo en las siguientes funciones:

1. Cifrado de desplazamiento.
2. Descifrado de desplazamiento.
3. Generación de la clave de rejilla.
4. Cifrado usando la rejilla.
5. Descifrado usando la rejilla.
6. Cifrado completo de un texto en claro.
7. Descifrado completo de un texto cifrado.

Cada uno de los ejercicios corresponde a la programación de una de estas funciones. A continuación, se describen los detalles.

1. Cifrado de desplazamiento (1 punto)

En este ejercicio implementaremos el cifrado de desplazamiento.

La función recibirá como argumento un mensaje y un entero positivo, que indicará el desplazamiento del cifrado y retornará el texto cifrado.

La función, que encontraréis en el esqueleto proporcionado, recibe los siguientes parámetros:

- La variable `message`: Mensaje a cifrar.
- La variable `shift`: Número de posiciones del desplazamiento.
- La función retornará una cadena de texto con el mensaje cifrado.

Ejemplo:

- `message`: `ABCD`
- `shift`: `3`
- `return`: DEFG

2. Descifrado de desplazamiento (1 punto)

En este ejercicio implementaremos el descifrado del sistema de desplazamiento implementado en el ejercicio anterior.

La función recibirá como argumento un entero positivo, que indicará el desplazamiento del descifrado en el sentido contrario y retornará el texto cifrado.

La función, que encontraréis en el esqueleto proporcionado, recibe los siguientes parámetros:

- La variable `message`: Mensaje a descifrar.
- La variable `shift`: Número de posiciones del desplazamiento.
- La función retornará una cadena de texto con el mensaje en claro.

Ejemplo:

- `message`: `DEFG`
- `shift`: `3`
- `return`: ABCD

3. Generación de la clave (2 puntos)

En este ejercicio implementaremos la generación de la clave de rejilla.

La función recibirá como argumentos dos enteros, que determinarán el tamaño total de la rejilla y el número de agujeros que tendrá.

La función retornará una lista de Python que contendrá ceros y unos. Las posiciones a uno indicarán los agujeros de la rejilla. Esta lista será la clave que usaremos para cifrar y descifrar.

Importante: La clave retornada no puede ser siempre la misma. Es recomendable consultar el módulo `random` de la librería estándar de Python.

La función, que encontraréis en el esqueleto proporcionado, recibe los siguientes parámetros:

- La variable `grille_len`: contendrá un entero que determinará el tamaño de la rejilla.
- La variable `num_holes`: contendrá un entero que determinará el número de agujeros en la rejilla.
- La función retornará una lista con la clave.

Ejemplo:

- `grille_len`: `10`
- `num_holes`: `5`
- `return`: `[0, 0, 0, 1, 1, 0, 0, 1, 1, 1]`

4. Cifrado de rejilla (2 puntos)

En este ejercicio implementaremos el cifrado para el criptosistema descrito.

La función recibirá como argumentos la clave y el texto a cifrar y retornará el texto cifrado.

La función, que encontraréis en el esqueleto proporcionado, recibe los siguientes parámetros:

- La variable `key`: Clave de rejilla.
- La variable `plaintext`: Texto a cifrar.
- La función retornará el texto cifrado.

Ejemplo:

- `key`: `[0, 0, 0, 1, 1]`

- `plaintext`: `HELLO`
- `return`: `2MIHEGV8LL9NMO`

5. Descifrado de rejilla (2 puntos)

En este ejercicio implementaremos el descifrado para el criptosistema descrito.

La función recibirá como argumentos la clave de rejilla y el texto cifrado, y retornará el texto descifrado.

La función, que encontraréis en el esqueleto proporcionado, recibe los siguientes parámetros:

- La variable `key`: Clave de rejilla.
- La variable `ciphertext`: Texto cifrado.
- La función retornará el texto descifrado.

Ejemplo:

- `key`: `[0, 0, 0, 1, 1]`
- `ciphertext`: `2MIHEGV8LL9NMO`
- `return`: `HELLO`

6. Criptosistema completo (2 puntos)

En este ejercicio implementaremos el criptosistema propuesto usando las funciones que se han implementado en los ejercicios anteriores. Por lo tanto, este ejercicio, a diferencia de los anteriores, requiere de la implementación de dos funciones: la de cifrado y la de descifrado.

Para cifrar, habrá que:

1. Realizar un primer cifrado usando el método de desplazamiento.
2. Cifrar usando el método de rejilla.

Y para descifrar, habrá que:

1. Descifrar usando el método de rejilla.
2. Descifrar el resultado usando el método de desplazamiento.

Criterios de valoración

Formato y fecha de entrega

La puntuación de cada ejercicio se encuentra detallada en el enunciado.

Por otro lado, es necesario tener en cuenta que el código que se envíe debe contener los comentarios necesarios para facilitar su seguimiento. En caso de no incluir comentarios, la corrección de la práctica se realizará únicamente de forma automática y no se proporcionará una corrección detallada. No incluir comentarios puede ser motivo de reducción de la nota.

La fecha máxima de envío es el **30/03/2023** (a las 24 horas).

Junto con el enunciado de la práctica encontraréis el esqueleto de la misma (fichero con extensión .py). Este archivo contiene las cabeceras de las funciones que hay que implementar para resolver la práctica. Este mismo archivo es el que se debe entregar una vez se codifiquen todas las funciones. No se tiene que enviar el fichero comprimido. No se aceptarán ficheros en otros formatos que no sean .py.

Adicionalmente, también os proporcionaremos un fichero con tests unitarios para cada una de las funciones que hay que implementar. Podéis utilizar estos tests para comprobar que vuestra implementación gestiona correctamente los casos principales, así como para obtener más ejemplos concretos de lo que se espera que retornen las funciones (más allá de los que ya se proporcionan en este enunciado). Nótese, sin embargo, que los tests no son exhaustivos (no se prueban todas las entradas posibles de las funciones). Recordad que no se puede modificar ninguna parte del archivo de tests de la práctica.

La entrega de la práctica constará de un único fichero Python (extensión .py) donde se haya incluido la implementación.