Isaac Lozano Osorio (isaac.lozano@urjc.es)
Antonio González Pardo (antonio.gpardo@urjc.es)

Práctica 3: Criptografía

El objetivo de esta práctica es profundizar en los conceptos explicados en las clases de teoría. De una manera más precisa, se utilizarán los conceptos explicados en el tema de criptografía (Tema 7) y se realizarán ataques a *hashes*, correspondientes al Tema 3.

Ante cualquier duda durante la resolución de la práctica, escribid un correo electrónico al profesor Tomás Isasia (tomas.isasia@urjc.es), poniendo en copia al profesor Isaac Lozano (isaac.lozano@urjc.es). No se responderán a dudas ni se concederán tutorías cuando queden **menos de 48 horas** para la fecha de entrega de la práctica.

Normas de realización

- Para esta práctica se utilizarán los mismos grupos creados para la práctica anterior.
- Dentro de Aula Virtual, encontraréis un archivo con nombre `material.zip` que contiene todo el material necesario para la realización de los ejercicios planteados en esta práctica.

Normas de entrega

- En esta práctica se entregará una memoria donde cada equipo deberá describir los pasos que han realizado y explicar el procedimiento utilizado en cada uno de los ejercicios.
- La memoria será entregada (en formato `pdf`) por un único miembro del equipo. Es imprescindible que **todos** los integrantes del grupo estén correctamente identificados en la entrega.
- La extensión máxima de la memoria será de 10 páginas, incluyendo portada.
- Además, se deberán adjuntar todos los códigos desarrollados, o en su defecto un enlace.
- La fecha límite para entregar la práctica será el **27 de abril a las 23:55**.

Evaluación de la práctica

- La nota máxima que se puede obtener en esta práctica es 10.
- Aquella memoria que esté corrupta con un 0.
- La evaluación de esta práctica se corresponde con el $10\,\%$ de la nota final de la asignatura.
- Las prácticas cuyas memorias superen el límite de páginas (10) se verán penalizadas con dos puntos menos.

1. Criptografía [5 puntos]

En este primer apartado comenzaremos analizando algunos algoritmos criptográficos. En la actualidad, existen una gran cantidad de algoritmos que se utilizan para mantener seguros los datos que utilizamos, o las conversaciones que mantenemos.

Las herramientas que vas a necesitar para realizar los ejercicios son las siguientes:

- https://gchq.github.io/CyberChef/
- https://www.dcode.fr/
- https://aperisolve.fr/

1.1. Cifrado del Cesar [1 punto]

Este cifrado, que lleva el nombre de Julio César, es uno de los tipos de cifrados más antiguos y se basa en el cifrado monoalfabético de desplazamiento, también es común verle llamado "ROT-N", donde N hace referencia al desplazamiento que se va a aplicar.

En este apartado se pide la implementación, en el lenguaje Python, de un codificador y decodificador Cesar. No está permitido el uso de librerías para la realización de este apartado. Su uso supondrá la anulación completa de la puntuación de este ejercicio.
Para verificar el correcto funcionamiento se probará con la cadena "ROGZZ3P4O_Q1J3", el alfabeto utilizado será el inglés (sin ñ). Es interesante destacar que para descifrar este texto, no conocemos el desplazamiento. Por lo que queda a vuestra elección elegir el mejor procedimiento de ataque sobre este cifrado.

1.2. Base64 [1 punto]

Un de las codificaciones más utilizadas es Base64. De la misma forma que en el ejercicio anterior, se pide la implementación, en el lenguaje Python, de un codificador y decodificador de texto en Base64. Está permitido el uso de librerías para la realización de este apartado.
Para verificar el correcto funcionamiento se probará con la cadena:

"VVJKQ3tTTTRMTF9CNFMzXzY0fQ=="

Cabe recordar que, una manera sencilla de identificar cuando un texto está codificado en base64 es prestando atención al último carácter de la cadena. Concretamente, en esta codificación se utiliza el carácter "=" como relleno para obtener una codificación válida.

1.3. Cifrado de Vigenère [1 punto]

Desarrollado en 1553, el cifrado Vigenère es un cifrado basado en diferentes series de caracteres o letras del cifrado César, formando estos caracteres una tabla, llamada tabla de Vigenère, que se usa como clave. El cifrado de Vigenère es un cifrado polialfabético y de sustitución.

El alfabeto que se va a utilizar en este ejercicio es:

```
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
```

el texto cifrado es "mvltykycmjfqlu" y la clave será "URJC".
En este apartado se pide la implementación, en el lenguaje Python, de un codificador y decodificador Cesar. No está permitido el uso de librerías para la realización de este apartado. Su uso supondrá la anulación completa de la puntuación de este ejercicio.

1.4. XOR [1 punto]

Muchos de los algoritmos de cifrado actuales se basan en el operador XOR a nivel de bit. Este XOR es una operación binaria que determina si las dos entradas son iguales (que devuelve un 0), o diferentes (la operación devuelve 1).

Para el completo aprendizaje del cifrado usando XOR, se pide la implementación en el lenguaje Python de un codificador y decodificador de XOR. No está permitido el uso de librerías para la realización de este apartado. Su uso supondrá la anulación completa de la puntuación de este ejercicio. Para verificar el correcto funcionamiento se probará con la cadena y la clave "XOR".

```
({!+8b*+
```

1.5. Cifrado AES [1 punto]

Algo más novedoso es el cifrado AES, que nació en 1998. Este algoritmo usa un esquema de cifrado por bloques adoptado como un estándar de cifrado por el gobierno de los Estados Unidos. En esta práctica se pide la implementación en el lenguaje de Python un codificador y decodificador del cifrado AES-128 con la clave e IV es "SeguridadInforma" y modo de ejecución CBC. La entrada y salida que se espera es en formato hex.

```
F55228945ACF1A291DB0C84409852406
```

Para este apartado se podrá hacer uso de librerías externas siempre y cuando estas estén disponibles en el *Python Package Index* (https://pypi.org/). Revisa que la herramienta esté en uso y no tenga vulnerabilidades.

2. Esteganografía [1 punto]

Se mostrará cuando se imparta el temario.

3. Ruptura de Hash [4 puntos]

Una función criptográfica *hash* es un algoritmo matemático que transforma cualquier bloque arbitrario de datos en una nueva serie de caracteres con una longitud fija. Tienen múltiples usos, pero la

más conocida tiene que ver con el almacenamiento, y envío, de contraseñas.

Si alguna vez olvidas tu contraseña de algún servicio en línea, probablemente tengas que restaurarla. Cuando se restablece una contraseña, por lo general no recibes una clave en texto plano. Eso es debido a que los servicios en línea no almacenan las contraseñas en texto plano, sino que las almacenan bajo el valor **hash** de la contraseña. De hecho, el servicio (a menos que utilices una contraseña demasiado simple, que haga que el valor hash sea ampliamente conocido) no tiene ni idea de cuál es la contraseña real. Esto se debe a que una de las propiedades matemáticas de las funciones hash es que son irreversibles, es decir, dada una función hash, no se puede recuperar la entrada que ha generado esa hash.

En esta práctica usaremos las siguientes herramientas:

- https://crackstation.net/ o https://hashes.com/en/decrypt/hash
- https://github.com/hashcat/hashcat
- https://github.com/blackploit/hash-identifier

El servicio de inteligencia de la URJC ha interceptado un fichero con hashes de cuentas institucionales (fichero **passwords.txt**). Están preocupados por si pueden romper las contraseñas, sospechan que son de cuentas de profesores. ¿Podrías tratar de romperlas? ¿Qué herramientas has utilizado? ¿Sabrías decir con qué función hash se han creado?
Los hashes han sido extraídos de un servicio dónde el mínimo 4 caracteres y máximo 17. Las contraseñas se podían escribir con: letras minúsculas, los símbolos “{” y “}” y números. Dicen que muchas contraseñas tienen que ver con algo llamado “flags”.

La puntuación se distribuye de la siguiente forma:

- Determinar el tipo de hash de las contraseñas (0.50 puntos).
- Obtener al menos 5 contraseñas (0.50 puntos).
- A partir de la 5º, cada contraseña extra obtendrá 0.30 puntos, siendo el máximo 3.50 puntos. 15 contraseñas es la puntuación completa (0.50 por las 5 primeras y $0.30 \cdot 10$).

Notas

- Debéis ver la documentación de hashcat. https://hashcat.net/wiki/.
- Podéis instalar hashcat en vuestro ordenador mediante https://github.com/hashcat/hashcat/releases/download/v6.2.6/hashcat-6.2.6.7z o en caso de error con docker https://hub.docker.com/r/dizcza/docker-hashcat.
- Es posible que os requiere instalar CUDA u OpenCL, es fácil con chocolatey https://community.chocolatey.org/packages/cuda o https://community.chocolatey.org/packages/opencl-intel-cpu-runtime.
- En caso de que hashcat se pare, puede ser por la temperatura de ventiladores, se puede subir o ignorar con `--hwmon-disable` (cuidado si se desactiva).
- Es posible que para obtener la nota máxima en este apartado se deba dejar ejecutando tiempo (1 hora) dependiendo del ordenador.