Práctica 3: Criptografía (Parte 1)

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

Ante cualquier duda durante la resolución de la práctica, escribir un email a antonio.gpardo@urjc.es con copia a isaac.lozano@urjc.es. En caso de no poderse resolver la duda vía mail, se puede concertar una tutoría, siempre y cuando se concierte en un período de **hasta 48 horas antes** de 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 llamado `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 o invitación al repositorio donde estén almacenados. (Nombre de usuario de GitHub: `isaaclo97`)
- 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 se evaluará con un 0.
- La evaluación de esta práctica se corresponde con el 10% de la nota final de la asignatura.

1 Criptografía

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://github.com/ReFirmLabs/binwalk
- 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 de cesar (**sin uso de librerías**).

Para verificar el correcto funcionamiento se probará con la cadena "MyaolcxuxChzilguncwuWymul", 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]

Además, otra de las codificaciones más utilizadas en los últimos años es la llamada Base64. De la misma forma que en el ejercicio anterior, se pide la implementación, en el lenguaje Python, de un codificador y decodificador de base64.
Para verificar el correcto funcionamiento se probará con la cadena:

"MjAyM19TZWd1cmlkYWRJbmZvcm1hdGljYUJhc2U2NA=="

Una manera sencilla de identificar las codificaciones base64 es mediante el último carácter de la cadena, ya que es muy corriente que los textos cifrados finalicen con el carácter =, que se usa como *padding*.

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.

Para este apartado se pide la implementación en el lenguaje de Python un codificador y decodificador del cifrado Vigenère (**sin uso de librerías**). El alfabeto que se va a utilizar en este ejercicio es: `abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ` "QqmiaiiiYmisqmwmxijs" y la clave será "Vigenere"

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 (**sin uso de librerías**). Para verificar el correcto funcionamiento se probará con la cadena "+*5-=;i.61!47=?9;;;." con la clave "XOR".

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.

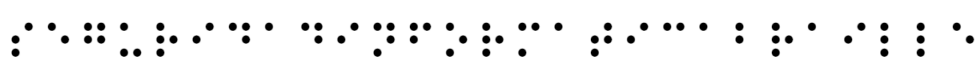
"`9bc43d7ec1aa11f64302287b17be9f7b`"

2 Esteganografía [1 punto]

Como se ha explicado en clase, existen una gran multitud de algoritmos y métodos para cifrar contenidos. En esta sección se proponen diferentes retos relacionados con la esteganografía (tanto con archivos de imágenes como audios).

Vuestra labor será la de intentar analizar los diferentes archivos, para ver si contienen información oculta. En la memoria deberéis detallar todos los pasos que habéis realizado y las conclusiones a las que llegáis.

- Esta imagen **image.jpg** tiene algo escondido ¿puedes encontrar el qué? (0.25 puntos)
- En el audio **AUDIO.wav** hay algo escondido ¿puedes encontrarlo? (0.25 puntos)
- Encuentra el secreto de este fichero **secreto.png** (0.25 puntos)
- Analiza el mensaje que contiene la siguiente imagen **codificacion.png** (0.25 puntos)



3 Ruptura de Hash

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 hash no se puede recuperar la entrada que ha generado esa hash.

Si alguna vez recibes una contraseña en texto plano, quiere decir que el servicio en línea que estás utilizando no está haciendo un hashing de tu contraseña (consejo: sospecha de ese sitio).

Como os podéis imaginar, existen mecanismos para la ruptura de hashes que se han explicado en clase.
En esta práctica usaremos las siguientes herramientas:

- https://crackstation.net/
- https://github.com/hashcat/hashcat
- https://github.com/blackploit/hash-identifier

El servicio de inteligencia de la URJC ha interceptado ciertas hashes, y necesitan de vuestra ayuda para intentar recuperar las posibles contraseñas que han generado dichas hashes. Tienen la sospecha de que estas contraseñas pertenecen a dos profesores del departamento y que un alumno las ha publicado. ¿Podrías ayudarles a determinar las contraseñas? ¿Sabrías decir con qué función hash se han creado?
Las hashes en cuestión son las siguientes:

- `c8ac686a2a44646d8ccc5501b27f7cce` (0.5 puntos)
- `970337e97c313fdbf46e4ad5ace0011a33f85540` (0.5 puntos)

El servicio de inteligencia ha sido capaz de identificar a los alumnos, y ha obtenido sus hashes, pero no conocen las contraseñas, así que no pueden vengarse.
Las hashes de los alumnos son estas:

- `19fee8879af928cff552a55a256a7808` (1.5 puntos)
- `4e538e3ef8fed7180d9d422566f7ab9aa9c15349` (1.5 puntos)

Además, saben que estos estudiantes no eran muy buenos en seguridad informática, ya que utilizan en sus contraseñas letras minúsculas y/o números. Por último, creen que el primer alumno usa contraseñas de 7 caracteres siempre, mientras que el segundo establece contraseñas de 8 caracteres.