



## PRÁCTICA: CRIPTOGRAFÍA

### Objetivo:

Contestar un conjunto de preguntas / ejercicios relacionados con la materia aprendida en el curso.

### Detalles:

En esta práctica el alumno aplicará las técnicas y utilizará las diferentes herramientas vistas durante el módulo. Para aquellos ejercicios para los que se haya usado código, se entregará el código junto con el documento de respuestas.

### Evaluación

Es obligatorio la entrega de un informe para considerar como APTA la práctica. Este informe ha de contener:

- Los enunciados seguido de las respuestas justificadas.
- En el caso de que se hayan usado comandos / herramientas también se deben nombrar y explicar los pasos realizados.

El código escrito para la resolución de los problemas se entrega en archivos separados junto al informe.

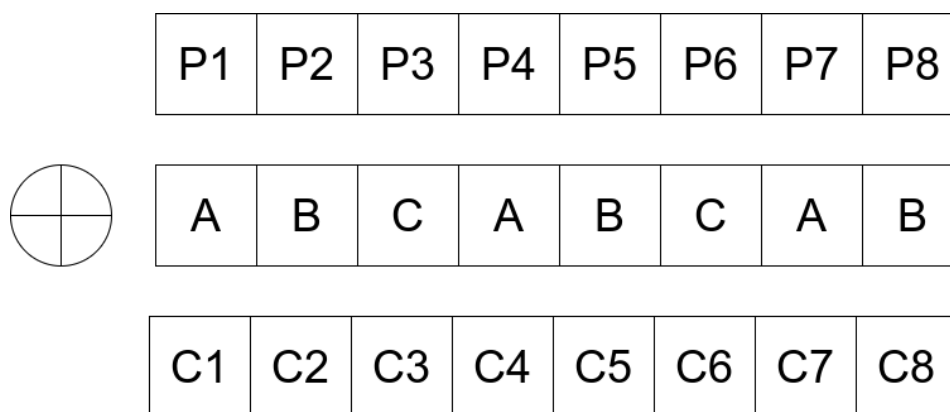
Se va a valorar el proceso de razonamiento aunque no se llegue a resolver completamente los problemas. Si el código no funciona, pero se explica detalladamente la intención se valorará positivamente.



## Ejercicios:

1. Junto a este documento se adjunta un archivo ej1.txt que contiene ciphertext en base64. Sabemos que el método para encriptarlo ha sido el siguiente:

La clave es de 10 caracteres. Frodo ha pensado que era poco práctico implementar one time pad, ya que requiere claves muy largas (tan largas como el mensaje). Por lo que ha decidido usar una clave corta, y repetirla indefinidamente. Es decir el primer byte de cipher es el primer byte del plaintext XOR el primer byte de la clave. Como la clave es de 10 caracteres, sabemos que el carácter 11 del ciphertext también se le ha hecho XOR con el mismo primer carácter de la clave. Debajo vemos un ejemplo donde la clave usada es “ABC”.



¿Por qué no ha sido una buena idea la que ha tenido Frodo? Explica cómo se puede encontrar la clave y descryptar el texto a partir del archivo.

Programa el código y encuentra la clave junto con el texto.

Pista: no debería ser muy costoso, ya que se puede reutilizar mucho código de un ejercicio hecho en clase ;)



2. En el archivo ej2.py encontramos dos funciones. Una de ellas obtiene una string cualquiera y genera una cookie encriptada con AES CTR, la función genera cuentas con rol user, y se encarga de verificar que la string no contiene ni ';' ni '='. La otra función desencripta la cookie y busca la string ';admin=true;'. Realizando llamadas a estas dos funciones, genera una cookie y modifica el ciphertext tal que al llamar la función de check te devuelva permisos de admin. (Obviamente se puede acceder a la clave de encriptación de la clase, pero el ejercicio consiste en modificar el sistema tal y como si el código se ejecutara en un servidor remoto al que no tenemos acceso, y solo tuvieramos acceso al código).

**Pista:** Piensa en qué efecto tiene sobre el plaintext en CTR cuando cambia un bit del ciphertext

3. Explica cómo modificarías el código del ejercicio 2 para protegerte de ataques en los que se modifica el ciphertext para conseguir una cookie admin=true.
4. Existe otro error en la implementación del Server que le puede hacer vulnerable a otro tipo de ataques, concretamente en los parámetros que usa para CTR. ¿Sabrías decir cuál es? ¿Cómo lo solucionarías?
5. En el archivo ej5.py encontrarás una implementación de un sistema de autenticación de un servidor usando JWT. El objetivo del ejercicio es encontrar posibles vulnerabilidades en la implementación del servidor, y explicar brevemente cómo podrían ser atacadas.
  - La función register crea un nuevo usuario en el sistema (en este caso guardado localmente en el objeto AuthServer en la lista users).
  - La función login verifica que el password es correcto para el usuario, y si es así, devuelve un JWT donde sub = user y con expiración 6h después de la creación del token.
  - La función verify verifica que la firma es correcta y devuelve el usuario que está autenticado por el token.

Para JWT usamos la librería PyJWT: <https://pyjwt.readthedocs.io/en/latest/>

Para funciones criptográficas, usamos la librería PyCryptoDome usada anteriormente en clase.

6. Modifica el código para solucionar las vulnerabilidades encontradas.
7. ¿Qué problema hay con la implementación de AuthServer si el cliente se conecta a él usando http? ¿Cómo lo solucionarías?



8. Sabemos que un hacker ha entrado en el ordenador de Gandalf en el pasado. Gandalf entra en internet, a la página de Facebook. En la barra del navegador puede ver que tiene un certificado SSL válido. ¿Corre algún riesgo si sigue navegando?
  
9. En el archivo sergi-pub.asc mi clave pública PGP:
  - Comprueba que la fingerprint de mi clave es:  
**4010179CB16FF3B47F7D09C4751DD875E2FBBF59**
  - Genera un par de claves, y añade tu clave pública a los archivos entregados, el archivo con tu clave pública tiene que estar encriptado para que lo pueda ver sólo con mi clave privada.
  - Usa tu clave privada para firmar el archivo de entrega, y adjunta la firma en un documento firma.sig (Recuerda que deberás hacer la firma con la versión final que entregues, ya que si realizas algún cambio, la firma no será válida).

**Explica los pasos que has seguido.**