Wason 0.1

En esta versión usarás RC4 para encriptar los mensajes comunicados con otros usuarios de Wason.

RC4 es un generador seudoaleatorio. Recibe una cadena (semilla o llave) de tamaño entre 1 y 256 bytes, y devuelve un flujo de bytes tan largo como se requiera. Recordemos que usando un generador seudoaleatorio G se obtiene directamente un análogo al cifrado one-time pad, haciendo G(k) xor m. A esta construcción se le llama cifrador de flujo (stream cipher), aunque a veces se le llama cifrador de flujo al generador seudoaleatorio (como a RC4 en nuestro caso).

En laboratorio introdujimos el concepto de vector de inicialización (IV), que convierte nuestro algoritmo de cifrado en un método no determinista, más bien en un método aleatorizado. De esta forma, si se obtiene c1 = Enc(k, hola), y más adelante se obtiene c2 = Enc(k, hola), tendremos la seguridad de que c1 y c2 son distintos y no hay ninguna correlación evidente.

¿Para cuándo?

Los que quieran podemos revisarlo el miércoles 13 de marzo en el laboratorio. La entrega en el Classroom es el domingo 17 de marzo.

¿Qué hay que hacer?

- Haz los cambios necesarios en tu cliente de Wason para que los mensajes enviados a otros usuarios estén cifrados con RC4, usando un vector de inicialización de 5 bytes para cada mensaje cifrado.
- El vector de inicialización se debe generar con os.urandom en Python. Para Javascript puedes usar el generador de http://bitwiseshiftleft.github.io/sjcl/ o https://developer.mozilla.org/en-US/docs/Web/API/Crypto/getRandomValues.
- La llave que se usará para cifrar los mensajes será introducida manualmente al abrir el cliente.
- Solo los mensajes enviados a otros usuarios van cifrados, es decir, solamente el campo “msj” tendrá una cadena de bytes cifrados.
- Los mensajes cifrados además deben ser codificados con Base64 para simplificar el formato en los objetos JSON.

El siguiente seudocódigo muestra cómo se verían las funciones de cifrado y descifrado.

```
función cifrar(llave : bytes, mensaje : bytes)
    IV = random(5)         // os.random en Python
    G = RC4.new(IV + llave)        // + denota concatenación
    cifrado = list(mensaje.length)
    for i = desde 0 hasta mensaje.length - 1
        cifrado[i] = mensaje[i] xor G.get_byte()
    return IV + bytes(cifrado)

función descifrar(llave, cifrado)
    IV = cifrado[0] + cifrado[1] + cifrado[2] +
         cifrado[3] + cifrado[4]
    cifrado = cifrado a partir de la posición 5
    G = RC4.new(IV + llave)
    mensaje = list(cifrado.length)
    for i = desde 0 hasta cifrado.length - 1
        mensaje[i] = cifrado[i] xor G.get_byte()
    return bytes(mensaje)
```

Puedes usar alguna implementación de RC4 que directamente encripta los mensajes, pero toma en cuenta el uso del vector de inicialización.