PRÁCTICA 5: NOTAS AUXILIARES PARA SU CORRECTO DESARROLLO E IMPLEMENTACIÓN

La práctica tiene por objetivo efectuar el cifrado de flujo de un texto llano binario (sin cifrar) para obtener un texto cifrado binario (criptograma), utilizando autómatas celulares 1-D binarios (con alfabeto de estados $A = \{0,1\}$) con "buenas" propiedades de pseudoaleatoriedad:

¿Qué autómatas celulares binarios debo investigar para encontrar candidatos con "buenas propiedades de pseudoaleatoriedad?

Potencialmente, podrían ser muchos (basta con hacer crecer la vecindad r tanto como queremos); en nuestro caso, limitaremos la búsqueda poniendo $k = 2, r = 1$.

De entre los autómatas celulares a investigar ¿cuáles consideramos que tienen "buena" aleatoriedad?

Desde un punto de vista de rigor matemático, serían aquellos que superasen un determinado número de test de aleatoriedad. Nosotros escogeremos un enfoque más sencillo, estableciendo arbitrariamente los siguientes criterios:

- La distancia de Hamming media $\overline{d_H}$ (para 1024 células y 1000 generaciones de evolución) es superior a 300.
- La entropía espacial media de configuración $\overline{H_C}$ (media de las entropías espaciales de configuración) es superior a 0.8.
- La entropía temporal celular media $\overline{H_t}$ (para la célula situada en el punto medio del array en la posición 499) medida sobre las 2000 generaciones es superior a 0.8

Nota: estos criterios reitero que son arbitrarios para el desarrollo de la práctica, pero no son rigurosos (ni en términos matemáticos ni de seguridad de la información) si nuestro esquema de cifrado en flujo estuviera destinado a aplicaciones criptográficas en entornos de riesgo.

¿Cómo hago la búsqueda en el espacio de reglas establecido? Preferentemente, de manera automatizada. El programa irá regla por regla, cargará una configuración inicial aleatoria para las 1024 células, ejecutará una simulación (obviamente no hace falta visualizarla en pantalla para esto) de 1000 generaciones, medirá los tres parámetros $\overline{d_H}$, $\overline{H_C}$ y $\overline{H_t}$ (tampoco hace falta pintar las curvas), verificará si cumple con los criterios establecidos y, en caso positivo, anotará en un fichero en disco la regla; el nombre del fichero será "reglas.txt". De esta forma, al terminar, el usuario dispondrá en el fichero de texto de una pequeña base de datos de reglas con "buenas" propiedades.

¿QUÉ ES UN CIFRADO DE FLUJO (CIFRADO DE VERNAM)?

Es una técnica de criptografía simétrica de clave privada, donde el mensaje representado por una cadena de bits, es cifrado utilizando un flujo de bits pseudoaleatorio (que se obtiene a partir de una clave privada compartida por emisor y receptor), aplicando la función $\oplus$ (XOR).

La siguiente Figura ilustra el funcionamiento de este tipo de cifrados:

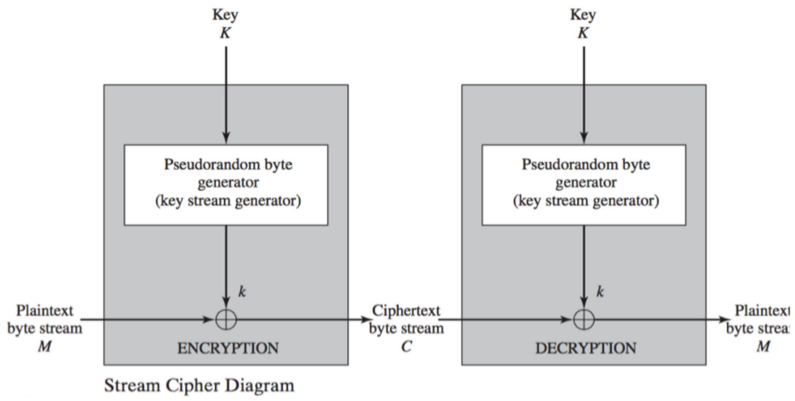

Figura 1: Cifrado en Flujo de Bits.

El esquema de cifrado se explica por sí mismo y en principio, no requiere mucho más análisis. En todo caso, sugiero en este punto la lectura de las páginas 191 a 194 (Capítulo 6) del *Handbook of Applied Criptography*, cuyo pdf tenéis disponible en la carpeta de la práctica. Si tras ello os queda alguna duda del funcionamiento de la técnica de cifrado, me preguntáis.

CIFRADO DE FLUJO CON AUTÓMATAS CELULARES

Ahora: para propósitos de cifrado,

- reducimos el tamaño del array de células a 512.
- y nos quedamos con la regla escogida para efectuar el cifrado de flujo(aquí, quedaría más "chulo" ofrecer un desplegable con lo lista de reglas que pasaron el filtrado, y que el usuario escoja una; vosotros mismos).

¿Cómo utilizo un autómata celular para cifrar un mensaje expresado mediante una cadena binaria de símbolos m_i? Pues la literatura para esto propone varias aproximaciones. Vosotros escogeréis aquella que utiliza como cadena de bits de cifrado k_i a los valores que la célula central del array (situada en la posición 256 de un total de 512) toma a lo largo del tiempo, obteniendo así el bit de criptograma mediante la ecuación $c_i = m_i \oplus k_i$, para $i = 1, 2, \ldots, n$; aquí n representa la longitud (en bits) del mensaje sin cifrar (texto llano). La siguiente fotografía (tomada en su día de esta misma explicación en pizarra) ilustra la idea:

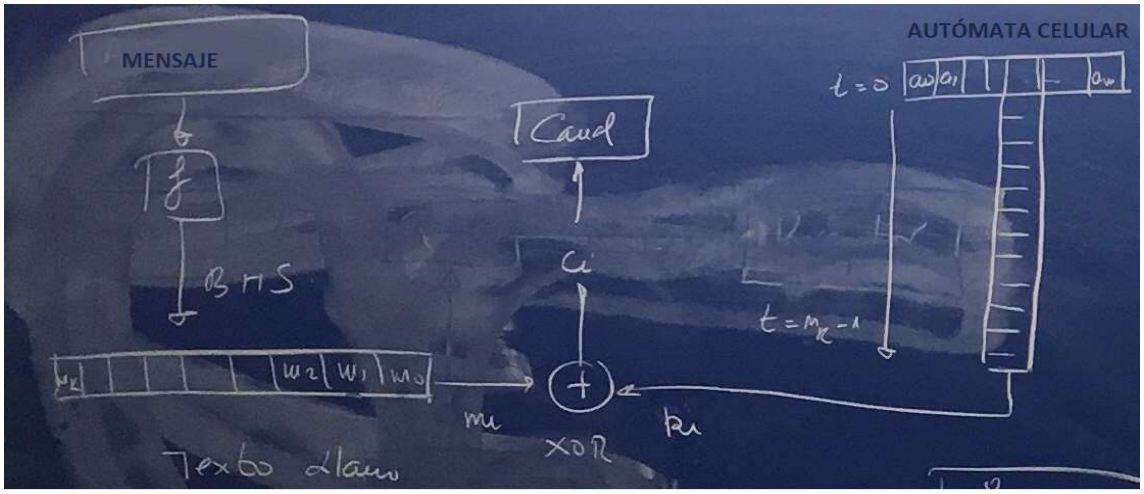

Figura 2: Esquema de Cifrado en Flujo con un AC-1D.

En la imagen, vemos a la izquierda un cuadro rotulado como "Mensaje", que contiene el texto a cifrar. Ese texto, a través de una función f adecuada se transforma a representación binaria (más abajo indico cómo), obteniéndose los bits de mensaje $m_0, m_1, ...$; a la derecha, está el autómata celular de cifrado. Vemos que se escoge una columna de valores para una células concreta (habitualmente la central) para obtener la secuencia de bits de cifrado k_i; finalmente, en el centro de la imagen, los bits de mensaje m_i y de cifrado k_i se combinan mediante la función $\oplus$ a través de la ecuación $c_i = m_i \oplus k_i$ para formar un bit de criptograma c_i, que se envíe al canal de comunicación.

Ayuda: hay un programita en la carpeta de la práctica llamada `cifraVernam.java` que muestra un núcleo básico de cifrado

¿Cuántas generaciones del autómata celular hay que computar? Tantas como bits tenga el mensaje que queremos cifrar, expresado mediante representación binaria, es decir, n.

¿Cuál es la clave del cifrado en flujo con autómatas celulares? La configuración inicial del autómata celular.

¿Cómo introduzco esa clave? Como hemos indicado, ahora, la configuración inicial del autómata celular no la estableceremos mediante un generador aleatorio, sino que la elegirá el usuario mediante un texto normal (un password de los de toda la vida) lo bastante largo, del que luego se obtendrá la representación binaria para cargar la configuración inicial del autómata celular.

¿Cómo paso el password y mensaje original (texto llano) a representación binaria? Pues echadle imaginación; habitualmente se usa un código numérico intermedio (ASCII, Unicode) y ya… En el siguiente URL tenéis una sencilla explicación de cómo hacerlo:

https://www.rapidtables.com/convert/number/ascii-to-binary.html

Ayuda: hay un programita en la carpeta de la práctica llamada `string2Ascii.java` que igual os resuelve esto.

¿Cómo paso el criptograma (texto cifrado) de representación binaria a un código habitual como ASCII ? Mediante el procedimiento inverso al seguido en el apartado anterior.

¿Qué inputs debe tener vuestro cifrado (integrado en el GUI)?

1. Cuadro para introducir el texto con el password.
2. Cuadro para introducir el texto llano con el mensaje a cifrar en un cuadro de texto.
3. OPCIONAL: si se permite escoger varios autómatas para cifrar, un desplegable con la lista de reglas para ello.

¿Qué outputs debe tener vuestro cifrado (integrado en el GUI)?

1. Cuadro de texto con el mensaje cifrado.

¿Qué elementos de controlo debe tener vuestro GUI?

1. Botón "Cifrar", que una vez introducidos password y texto llano activa el núcleo de cifrado, y presentar el texto cifrado en el cuadro respectivo.
2. Botón "Limpiar", para despejarlo todo y comenzar de nuevo.