Criptografía

Práctica 2: FSR y cifrado en flujo.

__TrollScience__

Sergio Balbuena Pantigas
Jesús Carabaño Bravo
Pablo Caro Martín
Francisco Miguel Dios Buitrago

Descripción del Software

Este programa ha sido implementado sobre el lenguaje de programación **Python** en su más reciente versión **3.2.2** y explota las funcionalidades que esta brinda, por lo que será necesario utilizar un interprete en su versión 3.x para lograr su correcto funcionamiento.

El contenido del programa se ha desglosado en tres secciones. La primera incluye los algoritmos solicitados para el estudio de secuencias pseudoaleatorias, con los cuales se permite determinar el periodo de secuencias y comprobar si dicho periodo verifica los tres postulados de Golomb (el segundo postulado era de carácter opcional). La segunda parte agrupa los distintos generadores de secuencias (NLFSR, LSFR, A5/1), los cuales se sirven de lo implementado en la primera sección de la práctica para llevar a cabo su finalidad. Por último, la tercera sección recoge el cifrado y descifrado de archivos la cual, además del generador A5/1, también permite utilizar como secuencia cifrante la contenida en un fichero externo (parte opcional).

Propiedades del Software

Las características más destacables de nuestro programa son la **facilidad de uso**, la **eficiencia** de ejecución y la **simplicidad del código** elaborado. Veamos por qué las destacamos:

- **Facilidad de Uso.** No existen más opciones de las requeridas ni menos de las necesarias y aunque la interfaz del tipo consola sea simple es más que suficiente para su labor. Utilizar un algoritmo solo requiere de dos sencillas interacciones con el teclado (también existen algunos atajos para elegir algoritmo mediante parámetro, sin pasar por el menú principal).
- **Eficiencia.** Es de sobra conocido que el lenguaje Python se encuentra a un nivel de abstracción superior a los clásicos lenguajes de programación de alto nivel. Este hecho repercute en la velocidad de ejecución de los programas generados, siendo normalmente más lentos que con otros lenguajes. Sin embargo, a manos de un buen programador esas deficiencias pueden solventarse e incluso es posible conseguir mejores rendimientos haciendo uso de las utilidades avanzadas que Python ofrece.
- **Simplicidad del código.** Esta propiedad no es visible al usuario medio pero es interesante conocer que, por ejemplo, con Python no es necesario controlar el overflow de sus números, pues internamente gestiona cualquier tamaño que sea necesario. Esta y otras características dotan de simplicidad al código y hacen posible una reutilización más agil y sencilla.

Manual de Uso

La utilización del programa es muy simple. En primer lugar ha de ejecutarse sobre algún interprete Python 3.2. Al lanzar la aplicación aparecerá un menú de elección en el que podrá decidirse la funcionalidad a utilizar mediante el teclado. Cuando se elija algún algoritmo el programa solicitará los parámetros que necesita para funcionar. Una vez introducido todo lo necesario comenzará la ejecución, mostrando los resultados por pantalla o generando un archivo resultado (al cifrar – descrifrar) cuando acabe.

Para el caso del cifrador se ha incluído una barra de progreso que permite ver de forma cómoda el tiempo restante para que el proceso termine de generar resultados.

Además, al generar una secuencia con cualquiera de los tres generadores de secuencia, preguntará si se quiere realizar un análisis del periodo. Se buscará el periodo de dicha secuencia, y se ejecutará el análisis basado en los Postulados de Golomb.

Existe la posibilidad de ejecutar el cifrador usando parámetros, sin necesidad de pasar por el menú. La sintaxis es la siguiente:

python P2.py -c archivo_a_cifrar archivo_llave [-ff]

- Si en archivo_llave se detecta una semilla válida para A5/1, se cifra con A5/1 a no ser que exista el parámetro -ff.
- Si no, se cifra usando el archivo como llave.

Nota:

Comparativa NLFSR

Función booleana (archivos de minterms)

Archivo 1:
0,1,2,4,5,9,12,13,14,15,16,17,18,19,20,23,24,26,29,32,33,35,36,37,38,43,45,46,50,52,53,56,59,61,62,65,68,72,73,75,80,82,83,84,85,92,93,94,96,97,99,100,102,105,106,107,110,114,117,119,126

Archivo 2:
1,3,4,5,10,11,13,14,19,20,21,23,24,25,30,32,33,34,35,37,38,40,41,42,43,44,48,49,50,52,57,60,61,63,65,67,69,70,71,75,77,79,82,85,89,91,92,93,94,95,99,105,106,107,109,112,113,114,115,116,118,120,122,123,126

Archivo 3:
5,6,9,11,14,17,19,21,22,23,24,25,26,29,31,33,37,38,39,40,42,44,46,48,50,52,53,54,55,58,61,65,67,68,69,70,71,72,73,75,76,77,85,89,90,91,93,96,97,98,99,101,102,103,106,108,109,113,114,115,117,118,119,121,122,123,125

En la siguiente tabla se puede comprobar que las secuencias generadas por un NLFSR con un registro de desplazamiento de 7 celdas tienen, generalmente, un periodo muy pequeño. Con distintas semillas, podemos encontrar periodos de en torno a 25 bits.

Función	Semilla	(longitud - descartados) Secuencia	Resultado
A1	1011100	(100-15) 0100101111011000001100101001011110110000011001010010111101100000110010100101111011000001100101001011110110000011001	Golomb 1º = V Golomb 2º = F Golomb 3º = F **Periodo = 23**
	1110111	(100-15) 0110000011001010010111101100000110010100101111011000001100101001011110110000011001010010111101100000110010100101111	Golomb 1º = V Golomb 2º = F Golomb 3º = F **Periodo = 23**
A2	1111111	(100-30) 0100100000111010011011000010010000011101001101100001001000001110100110110000100	Golomb 1º = F Golomb 2º = F Golomb 3º = F **Periodo = 25**
	1011110	(100-30) 0000011101001101100001001000001110100110110000100100000011101001101100001001	Golomb 1º = F Golomb 2º = F Golomb 3º = F **Periodo = 25**
A3	0011001	(100-30) 0101100110110111100000110101100110110111100000110101100110110111100000110101011	Golomb 1º = F Golomb 2º = F Golomb 3º = F **Periodo = 24**
	1011111	(100-30) 0110111100000110101100110110111100000110101100110110111100000110101100110110101	Golomb 1º = F Golomb 2º = F Golomb 3º = F **Periodo = 24**

Notas adicionales

En esta práctica, los generadores LFSR y A5/1 se han realizado utilizando funciones generadoras del lenguaje Python. Estas funciones son "co-procesos", funciones que no terminan su ejecución al devolver un valor, sino que quedan en suspensión esperando hasta que se pida otro valor o se cierre. Es decir, los generadores generan los bits en tiempo real mientras se les pide, en lugar de generar una lista de longitud fija.

Este tipo de implementación da sentido al cifrado en tiempo real, ya que no se requiere saber a priori la longitud de los datos a cifrar.

Para mejorar aún más la eficiencia, el cifrador A5/1 utiliza una técnica de buffering, que almacena los datos obtenidos en tiempo real en bloques pequeños, lo que disminuye tanto el tiempo de cómputo necesario como los recursos de memoria usados del ordenador.