Práctica 1: Criptografía Clásica

28 de septiembre de 2023

Índice

Resumen

El objetivo de esta práctica es la familiarización con métodos clásicos de cifrado. Los cifrados por sustitución, tanto monoalfabeto como polialfabeto, y los cifrados por transposición, así como con algunos de los métodos empleados para su criptoanálisis.

Introducción

La práctica puede ser realizada en lenguaje C de programación, o en Python. Se deberá elaborar una memoria sobre la práctica (en un fichero en formato pdf) que explique detalladamente la realización de todos los apartados con todos sus resultados correspondientes. En la memoria se podrá integrar el código necesario para entender la práctica correctamente. La elaboración de esta memoria es indispensable para la superación de la práctica. En el caso de la realización de la práctica en Python, se podrá presentar como memoria el PDF del "Jupyter Notebook — IPython"(explicando detalladamente la realización de todos los apartados con todos sus resultados correspondientes), a parte de los ficheros "*.ipynb"para poder ejecutar la práctica. En el caso de e la realización de la práctica en lenguaje C de programación, se presentarán los códigos de C utilizados más la memoria en PDF.

Problemas

1. Sustitución Monoalfabeto

Criptografía

a. Método afín (USO OBLIGATORIO DE GMP)

Programa el método afín utilizando los procedimientos numéricos de precisión múltiple de la biblioteca `GMP`. El programa se llamará `afin`. Los parámetros del programa podrían ser los siguientes:

`afin {-C|-D} {-m` $|Z_m|$`} {-a` $N_\times$`} {-b` N_+`} [-i` $file_{in}$`] [-o` $file_{out}$`]`

La explicación de los parámetros es la que aparece a continuación, y será la misma en los ejercicios siguientes, por lo que no se repetirá:

-C el programa cifra

-D el programa descifra

-m tamaño del espacio de texto cifrado

-a coeficiente multiplicativo de la función afín

-b término constante de la función afín

-i fichero de entrada

-o fichero de salida

El programa verificará que los números a y b determinan una función afín inyectiva. Si no lo cumplen, se generará un mensaje de error.

Si no se proporciona el parámetro `-i`, se procesará la entrada estándar. Si no se proporciona el parámetro `-o`, se utilizará la salida estándar. No obstante la estructura de parámetros de los programas es meramente orientativa y se puede usar la que crea más conveniente para cada problema. **Los algoritmos de Euclides y Euclides extendido se deben implementar explícitamente, siguiendo los algoritmos explicados en clase.**

Criptoanálisis

b. Aplicación Práctica (USO OBLIGATORIO DE GMP)

Diseña y genera un método de cifrado afín no trivial (basado en afín, pero con mayor fortaleza en el número de posibles claves que este) y explica el método para criptoanalizarlo aplicándolo a un ejemplo en concreto. Da una estimación de la fortaleza del método presentado, respecto al numero de claves posibles.

2. Sustitución Polialfabeto

Criptografía

a. Método de Hill (USO DE GMP NO OBLIGATORIO)

Programa el método de Hill. Se desarrollará un programa llamado `hill`. Los parámetros del programa podrán ser los siguientes:

`hill {-C|-D} {-m` $|Z_m|$`} {-n` N_K`} {-k` $file_K$`} [-i` $file_{in}$`] [-o` $file_{out}$`]`

Los parámetros introducidos en este caso son:

-m cardinalidad de Z_m

-n dimensión de la matriz de transformación

-k fichero que contiene la matriz de transformación, con el formato:

$$\begin{matrix} k_{1,1} & k_{1,2} & \cdots & k_{1,n} \\ k_{2,1} & k_{2,2} & \cdots & k_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ k_{n,1} & k_{n,2} & \cdots & k_{n,n} \end{matrix}$$

El programa verificará que la matriz K define una función inyectiva. Si no lo cumple, se generará un mensaje de error.

Si el tamaño del fichero de entrada no es múltiplo de n, se añadirá al final el número necesario de caracteres para que lo sea (*padding*).

b. Método de Vigenere (USO DE GMP NO OBLIGATORIO)

Implementa el método de Vigenere. Se creará para ello un programa llamado `vigenere`. El programa, por ejemplo, aceptará los siguientes parámetros de entrada:

`vigenere {-C|-D} {-k clave} [-i` $file_{in}$`] [-o` $file_{out}$`]`

-k cadena de caracteres usada como clave

Prueba que funciona tu programa cifrando y descifrando, por ejemplo Don Quijote de la Mancha, o cualquier otro libro que encuentres en la web. Utiliza diferentes tamaños de claves para los experimentos.

Criptoanálisis

c. Criptoanálisis del Vigenere (GMP NO OBLIGATORIO)

Implementa las funciones que estimes necesarias para criptoanalizar documentos cifrados mediante el método de Vigenere. Para sacar la longitud de la clave tienes que utilizar obligatoriamente los dos métodos: **el test de Kasiski** e **Índice de Coincidencia**. Considera utilizar las tablas de frecuencias que están al final de esta memoria. Las funciones que implementes deben conservar la filosofía del formato de entrada que se han especificado en secciones anteriores. Por ejemplo para el índice de coincidencia podría ser:

`IC {-l` N_{grama}`} [-i` $file_{in}$`] [-o` $file_{out}$`]`

-l longitud de n-grama buscado

Para realizar el criptoanálisis completo, una vez que se obtiene el tamaño de la clave, se extraerán cada una de las componentes de la misma, mediante la metodología explicada en clase.

Genera textos cifrados con diferentes tamaños de clave para Vigenere y aplica el criptoanálisis implementado para los textos cifrados. Muestra los resultados para todos los casos. Investiga y muestra con datos experimentales que sucede con el criptoanálisis cuando el tamaño de clave se va acercando a a la longitud del texto a cifrar.

En la memoria se debe explicar **muy claramente** los datos de salida que genera cada uno de los programas que utilizas para criptoanalizar los textos cifrados: i)Por qué ii) Para qué iii) Qué conclusiones sacas del análisis.

3. Cifrado de flujo (GMP NO OBLIGATORIO)

Crea un programa que realice un cifrado de flujo sobre un fichero de texto, por ejemplo Don Quijote de la Mancha, o cualquier otro libro que encuentres en la web (cifrando carácter por carácter con un

cifrado de desplazamiento). Para la generación de la secuencia de claves cifrante utiliza el algoritmo de generación de números aleatorios que consideres oportuno (no utilizar la función del RC4 para generar la secuencia cifrante), motivando así su utilización. Por ejemplo podéis utilizar un algoritmo de generación de números aleatorios basándoos en la idea de LFSR o NLFSR, y quizás mirar el capitulo 6 del Libro de criptografía aplicada: A. J. Menezes; P. C. van Oorschot; S. A. Vanstone (1997). Handbook of Applied Cryptography, para sacar alguna idea. Comprueba que tanto el cifrado como el descifrado funcionan correctamente. Explica en que se basa la fortaleza de criptosistema que has elegido y si es fácilmente atacable por un criptoanalista.

4. Método de transposición (GMP NO OBLIGATORIO)

Particularizar el método de Hill a un método de cifrado por transposición. Explica cómo se utilizaría el cifrado de Hill para realizar cifrado por permutación. Haz un programa, basado en el programa `hill`, llamado `transposición` que recibirá los siguientes argumentos:

`transposición {-C | -D} {-p permutación | -n` N_{perm}`} [-i` $file_{in}$`] [-o` $file_{out}$`]`

-p Cadena conteniendo la permutación. Por ejemplo, si queremos emplear esta permutación:

1	2	3	4	5
4	1	2	5	3

haremos: `transposición -p ‘‘4 1 2 5 3’’`

-n Número de elementos que se permutan. Sólo se utilizará con la opción -C

Cuando se utilice -n, el programa generará aleatoriamente una permutación del número de elementos indicado. Esta permutación se escribirá en un fichero llamado `permutacion.dat` para poder utilizarla posteriormente.

5. Producto de criptosistemas permutación

De modo similar al cifrado de Hill, asume que los bloques de cifrado de un criptosistema están constituidos por $M \times N$ elementos de texto claro, de forma que el texto claro P será una consecución de matrices de M filas y N columnas. El criptosistema consiste en hacer una doble permutación, primero se hace por fila, y posteriormente por columnas. En este caso la clave secreta de nuestro sistema de cifrado viene dada por dos vectores K_1 y K_2, que son empleados para permutar las filas y columnas de P respectivamente. Así explícitamente dados P, K_1 y K_2 cada uno de los bloques de texto C de dimensión $M \times N$ se calcula según el siguiente procedimiento:

A. Permutación de filas. La fila 1 de cada uno de los bloques de C viene dada por fila $K_1(1)$ de cada uno de los bloques de P, la fila 2 del mismo bloque C es la fila $K_1(2)$ del mismo bloque C, etc.

B. Permutación de columnas. La columna 1 de cada uno de los bloques de C será la columna $K_2(1)$ de cada uno de los bloques de P, la fila 2 del mismo bloque C viene definida por la fila $K_2(2)$ del mismo bloque C, y así sucesivamente.

Implementar este método de cifrado con un programa del estilo: `permutación {-C | -D} {-k1` K_1 `-k2` K_2`} [-i` $file_{in}$`] [-o` $file_{out}$`]` ¿Que posible criptoanálisis podría sufrir este doble cifrado de permutación y bajo que supuestos de modelo de seguridad?

6. Criptoanálisis mediante texto claro escogido del doble cifrado de permutación (APARTADO OPCIONAL)

El ataque por texto claro escogido [1] es un tipo de criptoanálisis en el que se asume que se tiene acceso a la máquina de cifrado. En este contexto el atacante puede cifrar tantos textos como considere oportuno para inferir la clave secreta de cifrado (o una parte de la misma). Con objeto de estudiar el ataque por texto claro escogido, se consideran los criptosistemas basados únicamente en la permutación de los elementos del texto claro del punto anterior de la memoria. A modo de ejemplo considérese $M = N = 4$, $K_1 = (3, 1, 4, 2)$ y $K_2 = (4, 2, 1, 3)$. Para esta configuración, el siguiente bloque de texto claro

$$P = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{pmatrix}$$

da lugar al bloque de texto cifrado

$$C = \begin{pmatrix} 4 & 2 & 1 & 3 \\ 4 & 2 & 1 & 3 \\ 4 & 2 & 1 & 3 \\ 4 & 2 & 1 & 3 \end{pmatrix}$$

La matriz P es una matriz con todas las filas iguales y, en consecuencia, al cifrarla todas sus filas siguen siendo iguales. Teniendo en cuenta esta propiedad, es posible deducir el valor del vector de permutaciones de columnas sin más que estudiar una cualquiera de las columnas de C. Así, el análisis de la primera fila de C nos permite inferir el contenido del vector K_2. En efecto, la columna 1 de dicha fila es igual a 4, lo que lleva a $K_1(1) = 4$; la columna 2 de cualquier fila de C es igual a 2, con lo que $K_2(2) = 2$; la columna 3 es igual a 1, de modo que se tiene $K_2 = 1$; la columna 4 es igual a 3, lo que implica $K_2 = 3$.

Basándose en esta elección particular de texto claro conocido anterior generar el número de bloques de textos claros necesarios para deducir K_1 y K_2, en un ejemplo del criptosistema de doble permutación del apartado anterior. Se puede demostrar que ese número de bloques de textos claros es igual $log_m(M \times N)$, donde m es la cardinalidad del lenguaje. Así por ejemplo en un tamaño de bloque de 4×4 y con 4 símbolos del lenguaje, el número de bloques de textos claros necesarios para criptoanalizar el sistema es igual a 2. En definitiva realizar un programa que "rompa" el cifrado de doble permutación, mediante texto claro escogido de manera apropiada, para un tamaño de bloque $M \times N$ y un alfabeto de cardinalidad m ¿Por qué el número de bloques de textos claros escogidos necesario para criptoanalizar este criptosistema es igual a $log_m(M \times N)$?

Información complementaria

Alfabeto El alfabeto con el que se trabajará es el formado por las letras que van de la A (código `ascii` 65) a la Z (código `ascii` 90).

Comienzo y entrega Comienzo: 28/09/2023, Entrega: 01/11/2023 (23:55 horas)

Tablas de frecuencia de las letras del alfabeto en castellano y en inglés

	Castellano	Inglés
A	11.96	8.04
B	0.92	1.54
C	2.92	3.06
D	6.87	3.99
E	16.78	12.51
F	0.52	2.30
G	0.73	1.96
H	0.89	5.49
I	4.15	7.26
J	0.30	0.16
K	0.0	0.67
L	8.37	4.14
M	2.12	2.53
N	7.01	7.09
O	8.69	7.60
P	2.77	2.00
Q	1.53	0.11
R	4,94	6.12
S	7.88	6.54
T	3.31	9.25
U	4.80	2.71
V	0.39	0.99
W	0.0	1.92
X	0.06	0.19
Y	1.54	1.73
Z	0.15	0.19

Referencias

[1] D. Arroyo, J. Diaz, F.B. Rodriguez 2013. Cryptanalysis of a one round chaos-based Substitution Permutation Network. Signal Processing 93(5):1358-1364.