Cifrado César

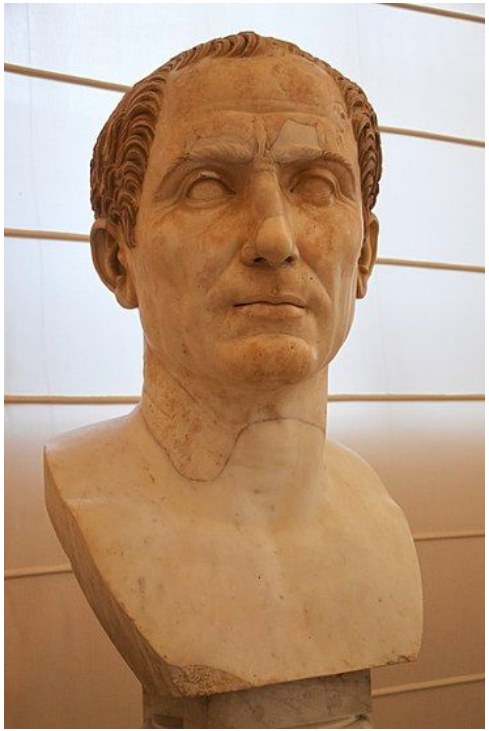
by Canek García (kaan.ek@ciencias.unam.mx)

Agenda

- Breve historia
- Cifrado César
- Implementación

Breve historía...

- Un tipo de cifrado por sustitución
- El nombre de este cifrado se debe a que Julio César utilizaba este método para comunicarse con sus generales en los campos de batalla.
- En el algoritmo ROT13 se usa el cifrado César con un desplazamiento de 13.
- Un cifrado similar al de César fue utilizado por la reina María Estuardo de Escocia, para conspirar junto con los españoles contra su prima Isabel I.
- El capo Bernardo Provenzano, utilizaba para comunicarse, en pleno siglo XXI, notas escritas con una máquina de escribir, codificadas mediante cifrado de César.



Cifrado César

Descripción del algoritmo

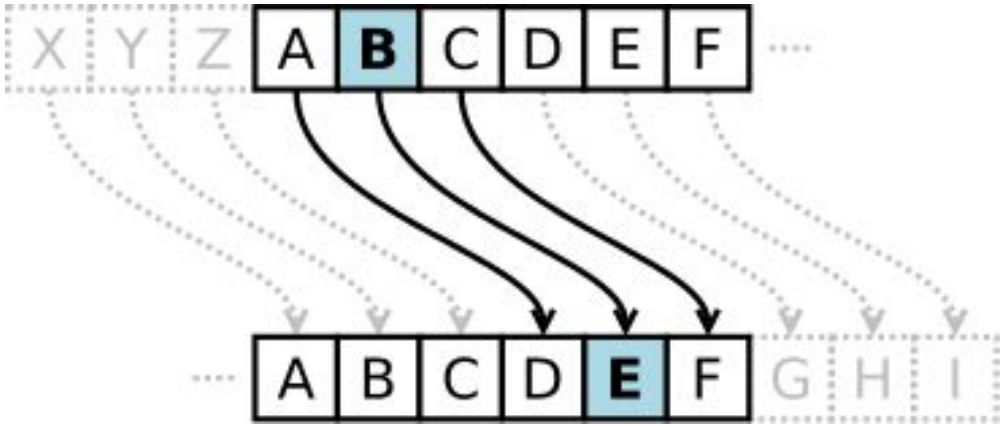
Es un tipo de cifrado por sustitución, en el que una letra en el texto original es reemplazada por otra letra que se encuentra un número fijo de posiciones más adelante en el alfabeto.

Alfabeto en claro:	A B C D E F G H I J K L M N Ñ O P Q R S T U V W X Y Z
Alfabeto cifrado:	D E F G H I J K L M N Ñ O P Q R S T U V W X Y Z A B C

Ejemplos

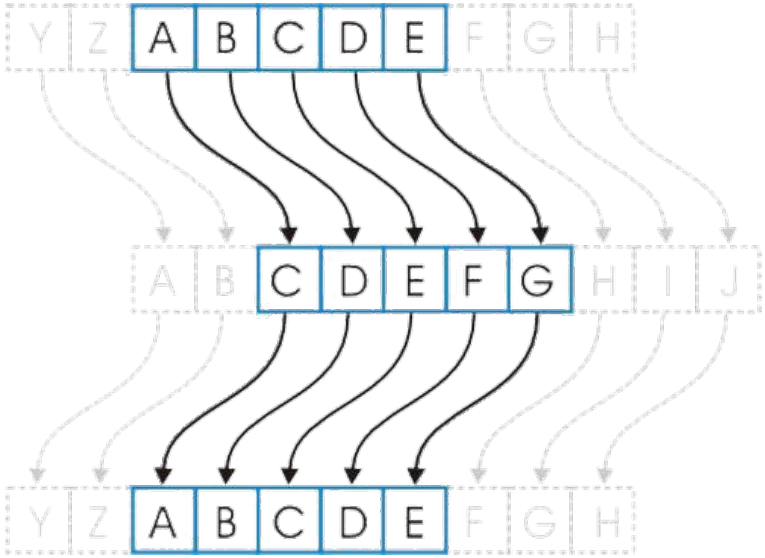
Aquí se usa un desplazamiento de tres espacios.

La letra B en el texto original se convierte en una E en el texto codificado.



Aquí se usa un desplazamiento de dos espacios.

La letra B en el texto original se convierte en una D en el texto codificado.



Implementación

Idea

Matemáticamente, podemos describir el método usado por Julio César como una función lineal del tipo:

$$E_n(x) = (x + n) \mod 26.$$

Código (Java)

```java
/**
 * E(x) = x+n (mod N)
 */
private static String codificar(String cadena, int desplazamiento){
    String rtnCadena = "";

    for(int i=0; i<cadena.length(); i++){

        if ( alfabetoMinusculas.contains( String.valueOf(cadena.charAt(i)) ) ||
             alfabetoMayusculas.contains( String.valueOf(cadena.charAt(i)) ) ) {

            char cipherChar;

            if ( alfabetoMinusculas.contains( String.valueOf(cadena.charAt(i)) ) ) {

                int x  = alfabetoMinusculas.indexOf(cadena.charAt(i));
                // E(x)=  x +        n             (mod N)
                int Ex = (x + desplazamiento) % alfabetoMinusculas.length();

                cipherChar = alfabetoMinusculas.charAt(Ex);

            }else{

                int x  = alfabetoMayusculas.indexOf(cadena.charAt(i));
                // E(x)=  x +        n             (mod N)
                int Ex = (x + desplazamiento) % alfabetoMayusculas.length();

                cipherChar = alfabetoMayusculas.charAt(Ex);
            }

            rtnCadena += cipherChar;

        }else{
            rtnCadena += cadena.charAt(i);
        }

    }

    return rtnCadena;
}
```

Decodificación

De manera análoga, podemos describir el método como una función lineal del tipo:

$$D_n(x) = (x - n) \mod 26.$$

Código (Java)

```java
/**
 * D(x) = x-n (mod N)
 */
private static String decodificar(String cadena, int desplazamiento){
    String rtnCadena = "";

    for(int i=0; i<cadena.length(); i++){

        if ( alfabetoMinusculas.contains( String.valueOf(cadena.charAt(i)) ) ||
             alfabetoMayusculas.contains( String.valueOf(cadena.charAt(i)) ) ) {

            char cipherChar;

            if ( alfabetoMinusculas.contains( String.valueOf(cadena.charAt(i)) ) ) {

                int x  = alfabetoMinusculas.indexOf(cadena.charAt(i));
                // D(x)=  x -        n             (mod N)
                int Dx = (x - desplazamiento) % alfabetoMinusculas.length();

                cipherChar = alfabetoMinusculas.charAt(Dx);

            }else{

                int x  = alfabetoMayusculas.indexOf(cadena.charAt(i));
                // D(x)=  x -        n             (mod N)
                int Dx = (x - desplazamiento) % alfabetoMayusculas.length();

                cipherChar = alfabetoMayusculas.charAt(Dx);
            }

            rtnCadena += cipherChar;

        }else{
            rtnCadena += cadena.charAt(i);
        }

    }

    return rtnCadena;
}
```

Código fuente

https://github.com/kanekko/cryptography_2019-1/blob/master/Lab00/Caesar.java