**Introducción**

Este proyecto se realizó con el objetivo de aplicar métodos de encriptación los cuales se dedujeron en la materia de criptografía y seguridad, la dificultad de los algoritmos se basa en el diferente lenguaje de programación en el que se desarrollaron.

La encriptación y desencriptación de un texto plano se realizará únicamente a textos del alfabeto español, sin contar los números ni caracteres especiales, en algunos métodos no se pudo aplicar encriptación con la letra ‘ñ’, en el resumen de cada meto especificaremos las precondiciones para poder encriptar un archivo de texto o una cadena cualquiera introducida por el usuario.

Existen distintas maneras de representar estos métodos de encriptación, las que hemos aplicado en este proyecto creemos que son las más adecuadas y precisas para representarlos, tomando en cuenta el tiempo dado para el proyecto y la complejidad del lenguaje de programación.

La interfaz está hecha mediante consola ya que el lenguaje de programación en el que se desarrolló este proyecto es PERL. Para poder ejecutar el programa, se debe tener instalado en el entorno del sistema dicho lenguaje.

Acto seguido se debe abrir la consola del sistema (también conocido como símbolo del sistema). Debemos acudir mediante comandos de consola hasta la carpeta en la que se encuentra el archivo PROYECTO.pl, una vez en la carpeta, debemos ejecutar el comando ‘perl proyecto.pl’ y acto seguido el programa se iniciará.

Se podrá encriptar y desencriptar archivos de textos guardados en el sistema y cadenas de texto ingresadas por el usuario.

**Métodos de encriptación**

1. **Deslizamiento puro**

Este método de cifrado es muy simple y es el más usado ya que sólo consiste en desplazar el abecedario. Para realizar este método usé la aritmética modular del cifrado de Cesar, transformando las letras en números del 0 al 25, para esto necesité del código ASCII y obtener el número de cada letra del texto a cifrar, pero no solo es suficiente con obtener este número, sino que además restarle cierta cantidad, 97 o 65, dependiendo de si es mayúscula o minúscula, para que así esté en el rango de 0 a 25. Luego le aplicamos la aritmética modular: **c = x + Desplazamiento mod 26** y así obtenemos el texto encriptado.

Para desencriptarlo apliqué el mismo método con esta única diferencia de que ahora será: **c = x – Desplazamiento mod 26**.

Solo encriptará textos que incluyan las letras del alfabeto excepto la ñ, los demás caracteres no los alterará.

1. **Deslizamiento puro con palabra clave**

Para realizar el cifrado y descifrado puro con Palabra clave se procedió a investigar previamente como se realizaba el cifrado con desplazamiento puro, que era mucho más sencillo, puesto que solo se realizaba el desplazamiento respectivo y se reconstruía el abecedario para su consiguiente sustitución de caracteres.

Para realizar el cifrado, se procedió a dividir los trabajos en funciones auxiliares para que no haya errores al momento de cifrar y descifrar. En primer lugar, el parámetro de entrada es una cadena, la palabra clave y el desplazamiento.

Como el mensaje a cifrar entra como una cadena, procedemos a convertirla a un arreglo para poder trabajar al momento de generar el nuevo alfabeto, con la ayuda de nuestra función ***CadToArraySinR***, eliminamos los caracteres repetidos. Luego de generamos el nuevo Alfabeto.

Para poder reconstruirlo, con la palabra clave ya modificada, generamos el nuevo alfabeto concatenando la palabra clave al principio con el abecedario, habiendo eliminado las letras del abecedario original previamente. En la función ***generarAlfConPalD*** *realizamos esto.*

Teniendo ya nuestras herramientas de apoyo, procedemos a cifrar nuestro mensaje. En nuestra función principal: **cifrarCesarP**, para cifrar el mensaje, o **descifrarCesarP**, para descifrar el mensaje.

El desarrollo del ejercicio permitió aplicar la lógica adquirida durante los años de práctica en la universidad. En este caso, aprender un nuevo lenguaje no es sencillo, pero siempre es útil porque nos nutrimos de más conocimiento y reforzamos nuestras habilidades en distintos lenguajes para poder programar. PERL es un lenguaje peculiar, utiliza comandos y funciones definidas que ayudar y retrasan el procedimiento del desarrollo de algoritmos.

En mi experiencia, son útiles, pero hay que investigar bastante. Aprender algo nuevo cada día, es subir un peldaño hacia el éxito.

1. **Transposición por grupos**

Para realizar el cifrado por transposición por grupos se tuvo que investigar el método, su desarrollo y complejidad. En este caso, consiste en agrupar por un conjunto especifico de números, que pueden ser agrupación de números primos, números impares, y números pares. Es posible también intercalar los índices desordenados de una cadena, como método de encriptación, pero la complejidad aumenta y es más difícil de descifrar. Para lo que haremos nosotros, utilizaremos agrupación de índices impares seguidos de índices pares de una cadena correspondiente.

Para realizar esto, creamos 2 funciones auxiliares, una que corresponde para el cifrado y otra para el descifrado. La primera función auxiliar **separa(),** como su nombre lo dice, separa las palabras por una cantidad de grupos ingresada como parámetro. Por ejemplo, si tenemos HOLACOMOESTAS, y el número de grupos es 3, se separará en HOL ACO MOE STA SXX, siendo estos últimos caracteres, añadidos en la misma función, cuando la última cadena no contiene la cantidad de caracteres igual al número de grupos.

Se procede a cifrar utilizando esta función auxiliar, separando los caracteres de la cadena en índices impares, concatenando a una nueva cadena, seguidos de los índices pares de la misma cadena.

Para descifrar, la otra función auxiliar **inter(),** la cual separa la cadena en 2, impares y pares, las cuales ayuda para hacer la intercalación de índices uno a uno, para formar el mensaje original. En este caso, PERL maneja comandos algo complicados de trabajar, pero nada imposible. Utilizamos la función de cadena chop() para poder sacar carácter uno a uno e ir intercalando los caracteres en una cadena nueva.

Al finalizar, la cadena quedará lista, pero invertida. Entonces, procedemos a utilizar una vez más la función chop() e ir concatenando en una cadena nueva, para tener como resultado el mensaje descifrado original.

1. **Transposición por series**

Esta encriptación se realizó en base a una serie de números pares, el mayor problema al realizar el ejercicio fue que el método carecía de más aseries para encriptar, así que se decidió hacer la serie solo con números pares e intercalar las posiciones al intercalar. Para desencriptarlo solo se necesitó saber si la longitud era par o impar, y así sacar carácter a carácter y luego unirlo en dos cadenas.

Este método es capaz de encriptar textos con la letra ñ y cualquier carácter, ya que solo las desplaza sobre sus índices.

1. **Transposición por columnas**

Para hacer más sencillo este método de cifrado, usé una matriz, llenando la matriz por filas con cada letra del texto a cifrar, donde la cantidad de columnas será igual al desplazamiento. Una vez llena la matriz saco cada una de las letras, pero esta vez por columnas y así formo mi texto ya cifrado.

Para desencriptar el texto cifrado aplique la misma lógica, solo que, en lugar de llenar la matriz por filas, las lleno por columnas y por lo tanto la cantidad de columnas seguirá siendo el desplazamiento. Así se va concatenando las letras y se vuelve a obtener el texto en plano.

Este método no tiene restricciones, puede encriptar cualquier texto ya que solo altera la posición de las letras en el texto.

1. **Transposición por filas**

Para desarrollar el algoritmo de Cifrado por Transposición por filas, gráficamente utilizamos una matriz, escribiendo el mensaje de forma vertical (o por columnas), según un numero de filas constante. Para cifrar, leemos horizontalmente cada fila y trascribimos el mensaje.

Para realizar este ejercicio, se utilizó la idea de una matriz, pero no fue necesario implementar una, sino que se trabajó realizando pruebas de algoritmos matemáticos para encontrar fórmulas que puedan ayudar a realizar esta transposición de una manera más sencilla.

Realizamos una función auxiliar **newPalabra**, para que realice esto. La función consiste en que, dado el mensaje a cifrar, la cantidad de filas y el carácter, si es caso faltara elementos en la matriz, reconstruye la nueva palabra, concatenando la cantidad restante de elementos fundamentales al final del mensaje dado por parámetro.

Como es una función genérica, se tuvo que validar tanto para mensajes, cuya longitud de la cadena sea >9, longitud de la cadena < cantidad de filas, o longitud de la cadena >cantidad de filas, pero menor a 10 caracteres.

Para cifrar, se introduce el mensaje, la cantidad de filas y el carácter de apoyo, si es que fuera necesario. Teniendo la nueva palabra (dada por la función ***newPalabra***), tenemos de pivotes o de los primeros ***$nfilas*** caracteres *($nfilas= cantidad de filas*), realizamos saltos acordes a las *$nfilas* dadas por parámetro, hasta los *$nfilas* primeros caracteres. Una vez sacado un carácter, concatenamos en una cadena vacía para ir reconstruyendo el mensaje cifrado.

El salto se realiza hasta que sea posible no pasándonos de la longitud de la cadena, y cada vez que llegue a su límite, se aumenta un contador y se va eliminando el primer carácter de la cadena pivote hasta llegar a $nfilas, dado como resultado el mensaje cifrado en la nueva cadena.

Para descifrar, el único parámetro que se calcula es el número de columnas para cifrar, esto es: ***$ncolum=$longitud/$nfilas*** (parte entera). Luego se procede a realizar los mismos pasos del cifrado, dado como resultado el mensaje descifrado.

La experiencia adquirida al realizar este ejercicio fue buena, puesto que se recordó a utilizar los desplazamientos por matrices y se buscó una manera más sencilla de recorrerlos.

La validación de algunos campos en el cifrado causó problemas, puesto que, como se hacía una división con parte entera, necesitábamos hacer un redondeo con incremento al siguiente digito, para realizar esto, nos ayudamos de la librería *integer*, almacenando el valor en una variable e incrementándole 1 al valor anterior.

El contenido del cifrado fue sencillo de comprender, puesto que también se lo aprendió en clase, y la implementación del algoritmo no fue muy tediosa de realizar.

1. **Transposición por zigzag**

Un método de encriptación muy complicado para programarlo, ya que hay que considerar las longitudes en función a la altura de la matriz a crear para poder encriptar. EL mayor problema encontrado al hacer este ejercicio, fue sacar los caracteres que están en el centro de las matrices creadas, ya que había que hacer muchas deducciones para poder encontrar un patrón lógico a usar. Pero al usar la lógica matemática y un poco de cálculos aritméticos se logró descifrar una fórmula para poder encriptar el texto. Para desencriptarlo se aplicó una lógica un poco inversa, con la diferencia que algunos cálculos se los hace a la inversa del original. Las precondiciones para utilizar este método es que la altura del zigzag no sea extremadamente alta.

1. **Sustitución monoAlfabética**

Para la encriptación monoalfabética se utiliza un alfabeto cualquiera y se reemplaza cada letra por la letra en la misma posición del nuevo alfabeto. Es un método muy sencillo para programar y es uno de los menos seguros en el área de encriptación. La forma para desencriptar es una de las más fáciles de resolver, por lo tanto, no es muy segura para proteger tus datos. Las precondiciones para encriptar con este método es que el alfabeto nuevo no sea más largo ni más corto que nuestro alfabeto común.

1. **Sustitucion poliAlfabetica**

Este método de cifrado forma su propia tabla a partir de varias series de cifrado Cesar. Para realizar este método usé la aritmética modular del cifrado de VINGENERE, transformando las letras del texto en números del 0 al 25, tanto del texto en plano como de la clave. Obtengo el número de la letra en código ASCII y dependiendo de si es mayúscula o minúscula le resto 65 o 97 para así obtener un número que esté entre el 0 al 26. Luego le aplico la aritmética modular del cifrado de VINGENERE que es: **c = letraClave + letraTexto mod 26**. Así obtengo el texto cifrado.

Para desencriptar aplico el mismo método solo con una pequeña alteración en la fórmula que sería: **c = letraClave - letraTexto mod 26**, y así desencriptamos el texto cifrado.

Este método es solo para letras del alfabeto sin incluir la ñ, por lo tanto, no los alterará.