CENTRO DE CIENCIAS BÁSICAS

DEPARTAMENTO DE SISTEMAS ELECTRÓNICOS

SEGURIDAD E INTEGRIDAD DE SISTEMAS

9° "A"

TAREA: PRIMEROS PASOS CIFRANDO

Profesor: Arturo Ocampo Silva

Alumnos:

Almeida Ortega Andrea Melissa

Espinoza Sánchez Joel Alejandro

Fecha de Entrega: Aguascalientes, Ags.**, 26** de agosto de 2022

Índice

Introducción

En informática, el cifrado es importante en términos de seguridad. La empresa Kaspersky (2022) comenta que “es la base principal de la seguridad de datos. Es la forma más sencilla e importante para garantizar que la información de un sistema de computadora no pueda robarla ni leerla alguien que desee utilizarla con fines maliciosos.”

El cifrado implica convertir texto sin formato legible por humanos en un texto incomprensible, conocido como texto cifrado. En esencia, esto significa tomar datos legibles y cambiarlos para que se vean como algo aleatorio. El cifrado implica utilizar una clave criptográfica; un conjunto de valores matemáticos que acuerdan tanto el emisor como el receptor. El receptor utiliza la clave para descifrar los datos y volver a convertirlos en texto sin formato legible.

Entre algunos tipos de cifrado que se han tratado a nivel educativo son el cifrado César y el cifrado Atbash. El cifrado César consiste en tomar un abecedario y al tomar una frase, recorrer una determinada cantidad de veces las letras de este abecedario. Un cifrado César-1 convierte las letras “a” en “b”, las “z” en “a” y eso sucede para todas las letras del abecedario. Un cifrado César-2 convierte las “a” en “c”, las “z” en “b” y eso sucede para todas las letras del abecedario en uso.

Por otra parte, el cifrado Atbash consiste en una conversión de texto en donde se inviertan las letras del abecedario en uso. En este caso, una letra “a” se convierte en una “z”, una “z” se convierte en “a”, una “b” se convierte en “y” y una “y” en una “b”. Esto sucede para todo el abecedario.

En el presente se tomará el espacio práctico para implementar estos dos tipos de cifrado usando el lenguaje de programación Python.

Objetivo

Objetivo: Implementar en un lenguaje de programación el cifrado César y Atbash usando el abecedario estándar.

Método: Mediante el lenguaje de programación Python, se enfocará a realizar una lógica de cómo se hace este cifrado y llevarlo a instrucciones de código de este lenguaje de programación.

Predicción: El equipo predice que mediante un enfoque a funciones puede realizarse la implementación independiente de los dos procesos para que ambos miembros del equipo trabajen a la par. De esta forma se predice que no se batallará mucho para realizar esta tarea.

Desarrollo

Primeramente, se decidió realizar esta práctica programada en el lenguaje de programación Python. Se decidió realizar dos funciones donde cada una realizara el procedimiento para el cifrado César y para el cifrado Atbash.

Cifrado César

Al realizar el cifrado César se planteó el uso de caracteres a través del código ASCII. Se planteó tener en cuenta los siguientes caracteres:

Carácter				Código			
(Caracter de espacio)				32			
Caracter	Código	Carácter	Código	Caracter	Código	Caracter	Código
A	65	N	78	a	97	n	110
B	66	O	79	b	98	o	111
C	67	P	80	c	99	p	112
D	68	Q	81	d	100	q	113
E	69	R	82	e	101	r	114
F	70	S	83	f	102	s	115
G	71	T	84	g	103	t	116
H	72	U	85	h	104	u	117
I	73	V	86	i	105	v	118
J	74	W	87	j	106	w	119
K	75	X	88	k	107	x	120
L	76	Y	89	l	108	y	121
M	77	Z	90	m	109	z	122

Teniendo una cadena de texto de entrada que contenga sólo los caracteres mencionados previamente y un valor para recorrer el abecedario, se pensó en realizar la siguiente función:

- Sea `cadena` la cadena de texto de entrada.
- Sea `cesar` el valor para recorrer el abecedario.
- Sea `newCadena` la cadena de salida producida al recorrer el abecedario.

El procedimiento es el siguiente:

1) Se lee y se guarda el texto en `cadena`.
2) Se guarda en `cesar` el valor en el que se recorrerá el abecedario.
3) Se inicializa `newCadena` como vacía.
4) `cesar = cesar MOD 26`
5) Por cada carácter de `cadena`:
    a. Se obtiene el valor en ASCII del carácter y se guarda en `x`.
    b. Si `x` es igual a 32:

i. Se trata de un caracter de espacio, por lo tanto se le añade a newCadena el caracter de espacio.

c. Si x está entre 65 y 90:
   i. Se trata de un caracter en mayúscula, por lo tanto, se aplica la función `y = x + cesar`.
   ii. Si y es mayor a 90:
      1. `y = 65 + (y - 90) + 1`.
      2. El caracter que le corresponde al valor de código ASCII de y se añade a `newCadena`.
   iii. Si y es menor a 65:
      1. `y = 90 - (65 - y) - 1`.
      2. El caracter que le corresponde al valor de código ASCII de y se añade a `newCadena`.

d. Si x está entre 97 y 122:
   i. Se trata de un caracter en minúscula, por lo tanto, se aplica la función `y = x + cesar`.
   ii. Si y es mayor a 122:
      1. `y = 97 + (y - 122) + 1`.
      2. El caracter que le corresponde al valor de código ASCII de y se añade a `newCadena`.
   iii. Si y es menor a 65:
      1. `y = 122 - (97 - y) - 1`.
      2. El caracter que le corresponde al valor de código ASCII de y se añade a `newCadena`.

Esta lógica se implementó en Python resultando la siguiente función:

```python
def cesar(cadena, cesar):
  newCadena = ''
  for i in range(len(cadena)):
    # Pasar el caracter de espacio
    if cadena[i] == ' ':
      newCadena = newCadena + ' '
    # Mayúsculas
    if (ord(cadena[i]) >= 65) and (ord(cadena[i]) <= 90):
      x = ord(cadena[i])
      y = x + cesar
      if y < 65:
        y = 90 - (65 - y) + 1
      if y > 90:
        y = 65 + (y - 90) - 1
      newCadena = newCadena + chr(y)
    # Minúsculas
    if (ord(cadena[i]) >= 97) and (ord(cadena[i]) <= 122):
      x = ord(cadena[i])
      y = x + cesar
      if y < 97:
        y = 122 - (97 - y) + 1
      if y > 122:
        y = 97 + (y - 122) - 1
      newCadena = newCadena + chr(y)

  return newCadena
```

El código en Python se ejecuta y se llama a la función desde la consola de Python, pasando como primer parámetro la cadena de texto y como segundo parámetro el valor con el que se recorrerá el abecedario. Algunos ejemplos de ejecuciones de código son los siguientes:

```
> cesar('abcdefg',1)
'bcdefgh'
```

```
> cesar('abcdefg',-1)
'zabcdef'
```

```
> cesar('Hola este es un ejemplo',5)
'Mtqf jxyj jx zs jojruqt'
```

Cifrado Atbash

Para el cifrado Atbash también se usó el enfoque mediante el uso de código ASCII. Aquí debía cambiarse la lógica para que invirtiera los números del código; por ejemplo, en el caso de las mayúsculas, el carácter 65 debe convertirlo a 90 y el 90 debe convertirlo a 65. Matemáticamente, una función que pueda realizar esto es la siguiente:

$$y = 155 - x$$

En el caso de las minúsculas, es necesario una función que pueda convertir el 97 a 122 y el 122 a 97. Una función matemática que haga lo anterior es:

$$y = 219 - x$$

Con esto definido, se toma como base la lógica del código César y se hacen las modificaciones pertinentes:

1) Se lee y se guarda el texto en `cadena`.
2) Se inicializa `newCadena` como vacía.
3) Por cada carácter de `cadena`:
    a. Se obtiene el valor en ASCII del carácter y se guarda en `x`.
    b. Si `x` es igual a 32:
        i. Se trata de un caracter de espacio, por lo tanto se le añade a newCadena el caracter de espacio.
    c. Si `x` está entre 65 y 90:
        i. Se trata de un caracter en mayúscula, por lo tanto, se aplica la función `y = 155 - x`.
        ii. El caracter que le corresponde al valor de código ASCII de y se añade a `newCadena`.
    d. Si `x` está entre 97 y 122:
        i. Se trata de un caracter en minúscula, por lo tanto, se aplica la función `y = 219 - x`.
        ii. El caracter que le corresponde al valor de código ASCII de y se añade a `newCadena`.

Esta lógica se implementó en Python resultando la siguiente función:

```python
def atbash(cadena):
  newCadena = ''
  for i in range(len(cadena)):
    # Pasar el caracter de espacio
    if cadena[i] == ' ':
      newCadena = newCadena + ' '
    # Mayúsculas
    if (ord(cadena[i]) >= 65) and (ord(cadena[i]) <= 90):
      x = ord(cadena[i])
      y = 155 - x
      newCadena = newCadena + chr(y)
    # Minúsculas
    if (ord(cadena[i]) >= 97) and (ord(cadena[i]) <= 122):
      x = ord(cadena[i])
      y = 219 - x
      newCadena = newCadena + chr(y)
  return newCadena
```

El código en Python se ejecuta y se llama a la función desde la consola de Python, pasando como parámetro la cadena de texto. Algunos ejemplos de ejecuciones de código son los siguientes:

```
> atbash('ABCXYZ')
'ZYXCBA'
```

```
> atbash('Este es un ejemplo de Atbash')
'Vhgv vh fm vqvnkol wv Zgyzhs'
```

Conclusión

Almeida Ortega Andrea Melissa: Por medio de este trabajo vimos de manera más específica los dos distintos tipos de cifrado César y Atbash, conociéndolos a detalle para poder realizar un programa con ellos, gracias a eso vemos como es que se protegen los datos a pesar de que estos son los cifrados más básicos ya nos da una idea de como se deben de proteger los datos.

Espinoza Sánchez Joel Alejandro: Mediante este espacio práctico para que implementemos el cifrado César y Atbash observamos un par de formas básicas para cuidar nuestros datos. Tal vez con un enfoque a menor escala, nos damos cuenta que podemos cuidarnos al proteger mensajes que enviamos y tan sólo con dos tipos de cifrado y muy básicos. Podemos llevar estos conocimientos a gran escala en el ámbito empresarial o incluso personal para tener cuidado con la información que pueda ser interceptada y usada con fines maliciosos.

Conclusión del equipo: Nos dimos cuenta que a nivel práctico podemos implementar cifrado de información para proteger datos, en este caso, a nivel de mensajes. Cuestión de seguridad que se hacía desde hace mucho tiempo por precaución. Podemos llevar este ideal de seguridad a más áreas de nuestra vida y protegernos de personas que quieran usar de forma indebida la información que intercepten.

Bibliografía

- Kaspersky Lab. (2022). *¿Qué es el cifrado de datos? Definición y explicación*. Agosto 17, 2022, de Kaspersky Sitio web: https://latam.kaspersky.com/resource-center/definitions/encryption.
- Wikipedia. (2018). *Atbash*. Agosto 17, 2022, de Wikipedia Sitio web: https://es.wikipedia.org/wiki/Atbash.
- Wikipedia. (2019). *Cifrado César*. Agosto 17, 2022, de Wikipedia Sitio web: https://es.wikipedia.org/wiki/Cifrado_C%C3%A9sar.

Anexos

Anexo 1: Código de cifrado en lenguaje de programación Python:

```
#%% Portada
'''
           Universidad Autónoma de Aguascalientes
                 Centro de Ciencias Básicas
            Departamento de Sistemas Electrónicos
             Seguridad e Integridad de Sistemas
                          9° "A"

              Tarea: Primeros Pasos Cifrando

               Profesor: Arturo Ocampo Silva

            Alumnos:
                Almeida Ortega Andrea Melissa
                Espinoza Sánchez Joel Alejandro

Fecha de entrega: Aguascalientes, Ags., 26 de agosto del 2022
'''


#%% Función para hacer cifrado César
def cesar(cadena, cesar):
  newCadena = ''
  for i in range(len(cadena)):
    # Pasar el caracter de espacio
    if cadena[i] == ' ':
      newCadena = newCadena + ' '
    # Mayúsculas
    if (ord(cadena[i]) >= 65) and (ord(cadena[i]) <= 90):
      x = ord(cadena[i])
      y = x + cesar
      if y < 65:
        y = 90 - (65 - y) + 1
      if y > 90:
        y = 65 + (y - 90) - 1
      newCadena = newCadena + chr(y)
    # Minúsculas
    if (ord(cadena[i]) >= 97) and (ord(cadena[i]) <= 122):
      x = ord(cadena[i])
      y = x + cesar
      if y < 97:
        y = 122 - (97 - y) + 1
      if y > 122:
        y = 97 + (y - 122) - 1
      newCadena = newCadena + chr(y)

  return newCadena

#%% Función para hacer cifrado Atbash
def atbash(cadena):
  newCadena = ''
```

```
for i in range(len(cadena)):
  # Pasar el caracter de espacio
  if cadena[i] == ' ':
    newCadena = newCadena + ' '
  # Mayúsculas
  if (ord(cadena[i]) >= 65) and (ord(cadena[i]) <= 90):
    x = ord(cadena[i])
    y = 155 - x
    newCadena = newCadena + chr(y)
  # Minúsculas
  if (ord(cadena[i]) >= 97) and (ord(cadena[i]) <= 122):
    x = ord(cadena[i])
    y = 219 - x
    newCadena = newCadena + chr(y)
return newCadena
```