
	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 1</p>

INFORME DE LABORATORIO

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	Seguridad Informatica				
TÍTULO DE LA PRÁCTICA:	FUNCIONES ELEMENTALES DE LA CRIPTOGRAFIA				
NÚMERO DE PRÁCTICA:	01	AÑO LECTIVO:	2023	NRO. SEMESTRE:	A
FECHA DE PRESENTACIÓN	15/06/2023	HORA DE PRESENTACIÓN			
INTEGRANTE (s): Yoset Cozco Mauri				NOTA:	
DOCENTE(s): Juan carlos Zuñiga					

SOLUCIÓN Y RESULTADOS
<p>I. SOLUCIÓN DE EJERCICIOS/PROBLEMAS</p> <p>Sobre el texto claro mostrado a continuación:</p> <p>Mi corazón oprimido Siente junto a la alborada El dolor de sus amores Y el sueño de las distancia. La luz de la aurora lleva Semilleros de nostalgias Y la tristeza sin los ojos De la médula del alma. La gran tumba de la noche Su negro velo levanta Para ocultar con el día La inmensa cumbre estrellada. ¡Qué haré yo sobre estos campos Cogiendo niños y ramas Rodeado de la aurora</p>

Y llena de noche el ama!
¡Qué haré si tienes tus ojos
Muertos a las luces claras
Y no ha de sentir mi carne
El calor de tus miradas!
¿Por qué te perdí por siempre
En aquella tarde clara?
Hoy mi pecho está reseco
Como una estrella apagada.

Implementar las siguientes operaciones de preprocesamiento, en cada caso debe mostrar el código de la implementación de la operación y la salida parcial resultante en cada paso (podrá usar como herramienta de desarrollo C++, python o java, pero no por una librería)

4.1 Realizar las siguientes sustituciones: axo, hxi, ñxm, kxl, uxv, wxv, zxy, xxr (tanto mayúsculas como minúsculas).

```
def sustitucion(texto):  
    # Realizamos las sustituciones solicitadas  
    texto = texto.replace('a', 'o')  
    texto = texto.replace('h', 'i')  
    texto = texto.replace('ñ', 'm')  
    texto = texto.replace('k', 'l')  
    texto = texto.replace('u', 'v')  
    texto = texto.replace('w', 'v')  
    texto = texto.replace('z', 'y')  
    texto = texto.replace('x', 'r')  
    # same replace for uppercase letters  
    texto = texto.replace('A', 'O')  
    texto = texto.replace('H', 'I')  
    texto = texto.replace('Ñ', 'M')  
    texto = texto.replace('K', 'L')  
    texto = texto.replace('U', 'V')  
    texto = texto.replace('W', 'V')  
    texto = texto.replace('Z', 'Y')  
    texto = texto.replace('X', 'R')  
    return texto
```

4.2 Elimina las tildes

```
def del_tildes(texto):  
    # Eliminamos las tildes  
    texto = texto.replace('á', 'a')  
    texto = texto.replace('é', 'e')  
    texto = texto.replace('í', 'i')  
    texto = texto.replace('ó', 'o')  
    texto = texto.replace('ú', 'u')  
  
    # same replace for uppercase letters  
    texto = texto.replace('Á', 'A')  
    texto = texto.replace('É', 'E')  
    texto = texto.replace('Í', 'I')  
    texto = texto.replace('Ó', 'O')  
    texto = texto.replace('Ú', 'U')  
    return texto
```

4.3 Convierta todas las letras a mayúsculas

```
def conver_upper(texto):  
    # Convertimos el texto a mayúsculas  
    return texto.upper()
```

4.4 Elimine los espacios en blanco y los signos de puntuación Indique cuál sería el alfabeto resultante y cuál su longitud

```
def del_spaces_punt(texto):  
    # Eliminamos los espacios en blanco y los signos de puntuación  
    texto = re.sub(r'\s+', '', texto)  
    texto = re.sub(r'^\w\s', '', texto)  
    return texto  
  
# Aplicamos las funciones al texto
```

```
texto = "Mi corazón oprimido Siente junto a la alborada El dolor de sus amores  
Y el sueño de las distancia. La luz de la aurora lleva Semilleros de nostalgias  
Y la tristeza sin los ojos De la médula del alma La gran tumba de la noche Su  
negro velo levanta Para ocultar con el día La inmensa cumbre estrellada. ;Qué  
haré yo sobre estos campos Cogiendo niños y ramas Rodeado de la aurora Y llena  
de noche el ama! ;Qué haré si tienes tus ojos Muertos a las luces claras Y no  
ha de sentir mi carne El calor de tus miradas! ;Por qué te perdí por siempre En  
aquella tarde clara? Hoy mi pecho está reseco Como una estrella apagada."  
  
# Guardamos el resultado en un archivo  
with open("POEMA_PRE.TXT", "w") as f:  
    f.write(texto)
```

GUARDE EL RESULTADO EN EL ARCHIVO "POEMA_PRE.TXT" (el que deberá ser adjuntado)

4.5 Abra el archivo generado e implemente una función que calcule una tabla de frecuencias para cada letra de la 'A' a 'Z'. La función deberá definirse como `frecuencias(archivo)` y deberá devolver un diccionario cuyos índices son las letras analizadas y cuyos valores son las frecuencias de las mismas en el texto (número de veces que aparecen). Reconozca en el resultado obtenido los cinco caracteres de mayor frecuencia

```
def frecuencias(archivo):  
    # Abrimos el archivo en modo lectura  
    with open(archivo, 'r') as f:  
        texto = f.read()  
  
    # Creamos un diccionario para almacenar las frecuencias  
    frecuencias = {}  
  
    # Recorremos cada letra de la 'A' a la 'Z'  
    for letra in 'ABCDEFGHIJKLMNOPQRSTUVWXYZ':  
        # Contamos la frecuencia de la letra en el texto y la almacenamos en el  
        # diccionario  
        frecuencias[letra] = texto.count(letra)  
  
    return frecuencias
```

```
# Usamos la función para calcular las frecuencias en el archivo "POEMA_PRE.TXT"
frecuencias = frecuencias("POEMA_PRE.TXT")

# Imprimimos las frecuencias
for letra, frecuencia in frecuencias.items():
    print(f'{letra}: {frecuencia}')

# Identificamos los cinco caracteres de mayor frecuencia
top_5 = sorted(frecuencias.items(), key=lambda x: x[1], reverse=True)[:5]

# Imprimimos los cinco caracteres de mayor frecuencia
print("\nLos cinco caracteres de mayor frecuencia son:")
for letra, frecuencia in top_5:
    print(f'{letra}: {frecuencia}')
```

4.6 Obtener la información que el método Kasiski requiere para implementar un ataque, para ello deberá recorrer el texto preprocesado y hallar los trigramas en el mismo (sucesión de tres letras seguidas que se repiten) y las distancias (número de caracteres entre dos trigramas iguales consecutivos)

```
import re

def kasiski(archivo):
    # Abrimos el archivo en modo lectura
    with open(archivo, 'r') as f:
        texto = f.read()

    # Creamos un diccionario para almacenar los trigramas y sus distancias
    trigramas = {}

    # Recorremos el texto buscando trigramas
    for i in range(len(texto) - 2):
        trigrama = texto[i:i+3]
        if re.match(r'[A-Z]{3}', trigrama):
            if trigrama not in trigramas:
                trigramas[trigrama] = []
```

```
        else:
            ultima_ocurrencia = trigramas[trigrama][-1] if
trigramas[trigrama] else 0
            trigramas[trigrama].append(i - ultima_ocurrencia)

# Abrimos el archivo "trigrama.txt" en modo escritura
with open("trigrama.txt", "w") as f:
    # Escribimos los trigramas y sus distancias en el archivo
    for trigrama, distancias in trigramas.items():
        f.write(f'{trigrama}: {distancias}\n')

return trigramas

# Usamos la función para calcular los trigramas y sus distancias en el archivo
"POEMA_PRE.TXT"
trigramas = kasiski("POEMA_PRE.TXT")

# Imprimimos los trigramas y sus distancias
for trigrama, distancias in trigramas.items():
    print(f'{trigrama}: {distancias}')
```

4.7 Volver a preprocesar el archivo cambiando cada carácter según UNICODE-8

```
def preprocesar_unicode(archivo):
    # Abrimos el archivo en modo lectura
    with open(archivo, 'r') as f:
        texto = f.read()

    # Convertimos cada carácter a su número Unicode y lo almacenamos en una
lista
    unicode_texto = [str(ord(c)) for c in texto]

    # Unimos los números Unicode en una cadena y la retornamos
    return ' '.join(unicode_texto)
```

```
# Usamos la función para preprocesar el archivo "POEMA_PRE.TXT"
unicode_texto = preprocesar_unicode("POEMA_PRE.TXT")

# Imprimimos el texto preprocesado
print(unicode_texto)
```

4.8 Volver a preprocesar el archivo cambiando cada carácter según alfabeto de su elección
Volver a preprocesar el archivo insertando la cadena AQP cada 20 caracteres, el texto resultante deberá contener un número de caracteres que sea múltiplo de 4, si es necesario rellenar (padding) al final con caracteres X según se necesite

```
def preprocesar_insertar(archivo):
    # Abrimos el archivo en modo lectura
    with open(archivo, 'r') as f:
        texto = f.read()

    # Insertamos 'AQP' cada 20 caracteres
    texto = ''.join(texto[i:i+20] + 'AQP' for i in range(0, len(texto), 20))

    # Comprobamos si la longitud del texto es múltiplo de 4
    while len(texto) % 4 != 0:
        # Si no lo es, añadimos 'X' al final del texto
        texto += 'X'

    return texto

# Usamos la función para preprocesar el archivo "POEMA_PRE.TXT"
texto = preprocesar_insertar("POEMA_PRE.TXT")

# Imprimimos el texto preprocesado
print(texto)
```

II. SOLUCIÓN DEL CUESTIONARIO

Describe alguna otra operación o función de preprocesamiento que se implemente sobre el texto claro en los criptosistemas, justifique ¿por qué esta etapa es necesaria?

En criptografía, a menudo se realiza un preprocesamiento para estandarizar el texto antes del cifrado. Esto puede incluir convertir todo a mayúsculas o minúsculas, eliminar espacios o signos de puntuación, entre otros. Este paso es crucial para garantizar un cifrado y descifrado eficaces y para limitar la cantidad de pistas que un potencial atacante podría obtener del texto cifrado.

2. ¿Qué riesgo implicaría el implementar el preproceso de la información?

Sin embargo, el preprocesamiento no está exento de riesgos. Si se hace incorrectamente, puede introducir errores en los datos o perder información importante. Además, si un atacante conoce el método de preprocesamiento, puede usarlo para ayudar a descifrar el texto. Por lo tanto, es vital que los métodos de preprocesamiento sean bien pensados y probados.

III. CONCLUSIONES

En conclusión, el preprocesamiento es una etapa esencial en los criptosistemas. Ayuda a estandarizar y preparar el texto para el cifrado, lo que facilita el proceso de cifrado y descifrado. Sin embargo, es importante tener en cuenta que el preprocesamiento debe realizarse con cuidado. Si se hace incorrectamente, puede introducir errores o perder información valiosa. Además, si un atacante conoce el método de preprocesamiento, puede usarlo para ayudar a descifrar el texto. Por lo tanto, es crucial diseñar y probar cuidadosamente los métodos de preprocesamiento para garantizar que sean seguros y efectivos. En última instancia, un buen preprocesamiento puede mejorar la seguridad y la eficiencia de un criptosistema.

RETROALIMENTACIÓN GENERAL

REFERENCIAS Y BIBLIOGRAFÍA



UNIVERSIDAD NACIONAL DE SAN AGUSTIN
FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA



Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 9

Link archivos : https://github.com/ycozco/nsa23_02/tree/main/seguridad_informatica/lab