


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

INFORME DE LABORATORIO

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	Seguridad Informática				
TÍTULO DE LA PRÁCTICA:	FUNCIONES ELEMENTALES DE LA CRIPTOGRAFÍA				
NÚMERO DE PRÁCTICA:	01	AÑO LECTIVO:	2023	NRO. SEMESTRE:	A
FECHA DE PRESENTACIÓN	09/06/2023	HORA DE PRESENTACIÓN	15/06/2023		
INTEGRANTE (s): FRANK BERLY QUISPE CAHUANA				NOTA:	
DOCENTE(s): Juan Carlos Zuñiga					

SOLUCIÓN Y RESULTADOS
<p>I. SOLUCIÓN DE EJERCICIOS/PROBLEMAS</p> <p><i>Sobre el texto claro mostrado a continuación:</i></p> <p align="center"> <i>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!</i> </p>

*¡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.
ALBA, Federico García Lorca*

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

```
import re

# Realizar las siguientes sustituciones: axo, hxi, ñxm, kxl, uxv, wxv, z
# xy, xxr
# (tanto mayúsculas como minúsculas).
def sustituciones(texto):
    sustituciones_dict = {
        'a': 'o',
        'h': 'i',
        'ñ': 'm',
        'k': 'l',
        'u': 'v',
        'w': 'v',
        'z': 'y',
        'x': 'r'
    }

    for k, v in sustituciones_dict.items():
        texto = re.sub(r'{}'.format(k), v, texto, flags=re.IGNORECASE)
    return texto
```

2. Elimine las tildes

```
#Elimine las tildes
def eliminar_tildes(texto):
    tildes_dict = {
        'á': 'a',
        'é': 'e',
        'í': 'i',
        'ó': 'o',
        'ú': 'u',
        'Á': 'A',
        'É': 'E',
        'Í': 'I',
        'Ó': 'O',
        'Ú': 'U'
    }
    for k, v in tildes_dict.items():
        texto = texto.replace(k, v)
    return texto
```

3. *Convierta todas las letras a mayúsculas*

```
# Convierta todas las letras a mayúsculas
def mayusculas(texto):
    return texto.upper()
```

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

```
# Elimine los espacios en blanco y los signos de puntuación
```

```
def eliminar_espacios_puntuacion(texto):  
    texto = re.sub(r'[\s\W]+', '', texto)  
    return texto
```

```
# Indique cuál sería el alfabeto resultante y cuál su longitud
```

```
def obtener_alfabeto(texto):  
    texto = eliminar_espacios_puntuacion(texto)  
    alfabeto = set(texto)  
    longitud = len(alfabeto)  
    return alfabeto, longitud
```

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

- 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

```
# implementar una función que calcule una tabla de frecuencias para cada
# letra de la 'A' a 'Z'.
def frecuencias(archivo):

    frecuencias_dict = {}
    with open(archivo, 'r') as file:
        texto = file.read()

    for letra in 'ABCDEFGHIJKLMNOPQRSTUVWXYZ':
        frecuencias_dict[letra] = texto.count(letra)

    return frecuencias_dict
```

```
# Reconozca en el resultado obtenido los cinco caracteres de mayor frecuencia
def caracteres_mayor_frecuencia(frecuencias_dict):
    frecuencias_mayor = sorted(frecuencias_dict.items(), key=lambda x: x[1], reverse=True)[:5]
    return frecuencias_mayor
```

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)

```
# , 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)
def obtener_trigramas_y_distancias(texto):
    trigramas = set()
    distancias = []

    for i in range(len(texto)-2):
        trigrama = texto[i:i+3]
        if texto.count(trigrama) > 1:
            trigramas.add(trigrama)
            indice = texto.index(trigrama)
            distancia = texto.index(trigrama, indice+1) - indice
            distancias.append(distancia)

    return trigramas, distancias
```

```
# Hexadecimal
def convertir_a_hexadecimal(numero):
    digitos_hex = "0123456789ABCDEF"
    resultado = ""

    if numero == 0:
        return "0"

    while numero > 0:
        residuo = numero % 16
        resultado = digitos_hex[residuo] + resultado
        numero = numero // 16

    return resultado
```

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

```
# Volver a preprocesar el archivo cambiando cada carácter según UNICODE-8
def preprocesar_a_unicode8(archivo_entrada, archivo_salida):
    with open(archivo_entrada, 'r') as file:
        texto = file.read()

    texto_unicode8 = ""
    for caracter in texto:
        texto_unicode8 += "U+" + str(convertir_a_hexadecimal(ord(caracter)).zfill(4)) + " "

    with open(archivo_salida, 'w', encoding='utf-8') as file:
        file.write(texto_unicode8)
```

8. 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

```
# 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_con_padding(archivo_entrada, archivo_salida):
    with open(archivo_entrada, 'r') as file:
        texto = file.read()

    texto_procesado = ""
    contador = 0
    for caracter in texto:
        if contador % 20 == 0 and contador > 0:
            texto_procesado += "AQP"
            texto_procesado += caracter
            contador += 1
        else:
            texto_procesado += caracter
            contador += 1

    if len(texto_procesado) % 4 != 0:
        padding = 4 - (len(texto_procesado) % 4)
        texto_procesado += "X" * padding

    with open(archivo_salida, 'w') as file:
        file.write(texto_procesado)
```

Main - Ejecucion

```
if __name__ == '__main__':
    # Leer el archivo de texto
    with open('text.txt', 'r') as file:
        texto = file.read()

    # Realizar las sustituciones, eliminar tildes, convertir a mayúsculas y eliminar es
    # pacios y puntuación
    texto = sustituciones(texto)
    texto = eliminar_tildes(texto)
    texto = mayusculas(texto)
    texto = eliminar_espacios_puntuacion(texto)

    # Guardar el resultado en un archivo
    with open('POEMA_PRE.txt', 'w') as file:
        file.write(texto)

    # Obtener el alfabeto y su longitud
    alfabeto, longitud = obtener_alfabeto(texto)

    # Imprimir el resultado
    print(f'Texto preprocesado: {texto}')
    print(f'Alfabeto: {alfabeto}')
    print(f'Longitud del alfabeto: {longitud}')

    # Calcular las frecuencias de cada letra
    frecuencias_dict = frecuencias('POEMA_PRE.txt')

    # Imprimir las cinco letras de mayor frecuencia
    caracteres_mayor_frecuencia_dict = caracteres_mayor_frecuencia(frecuencias_dict
)
    print(f'Caracteres de mayor frecuencia: {caracteres_mayor_frecuencia_dict}')

    # Obtener los trigramas y distancias
    with open('POEMA_PRE.txt', 'r') as file:
        poema_pre = file.read()
        trigramas, distancias = obtener_trigramas_y_distancias(poema_pre)

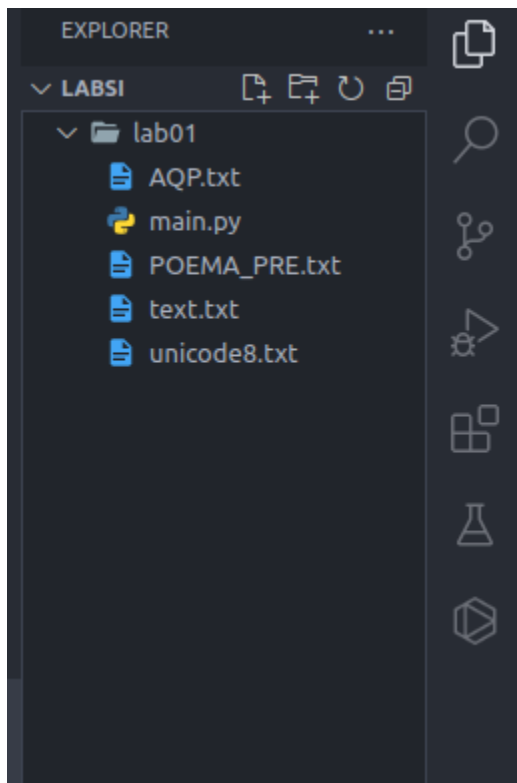
    # Imprimir los trigramas y distancias
    print(f'Trigramas encontrados: {trigramas}')
    print(f'Distancias: {distancias}')

    # Preprocesar a UNICODE-8
    preprocesar_a_unicode8('text.txt', 'unicode8.txt')

    # Preprocesar con padding
    preprocesar_con_padding('POEMA_PRE.txt', 'AQP.txt')
```





```
frank@unsa:~/documents/unsa/labSI/lab01$ python3 main.py
Texto preprocesado: AICORDYONOPRIINIDOSIENTESVNTLOOLBORODGELDOLORDESVSOPNRESYELSVRODELSDISTONCLOLVDELQVOROLLOVESEMIILLEROSDENOSTOLCISYLOTRISTEYOSINLOSXOSDELQMEDVLDELLOLLOLGRONTVMBDELONOCIESVNEGROVELOL
EVONTOPOROSVLTORONELDLOLJINWINGSOVMBREESTRELLLOQVQVIOREVSOSBRETSOSCOMOSCOGEGENONIMOSYROMOSROBRODDELLOVROVRYLLLENODNOCIEELONQVETIORESTITENESVSOSJASVERTOSLOSLVCSCLOROSYNODISENTIRHICORNEELCOLORDETVSHIRODO
SPONQVETEPERIPORSTIEMPRENOQVVELLOTORDECLORLOTOVMIPECTOSTARESECOCOMOVNOSTRELLLOQVODGODLOLBOFEDERICOGORCITOLORCDO
Alfabeto: ['A', 'J', 'V', 'S', 'Q', 'R', 'O', 'N', 'E', 'P', 'I', 'C', 'B', 'Y', 'D', 'T', 'F', 'L', 'M', 'G']
Longitud del alfabeto: 20
Caracteres de mayor frecuencia: [('O', 120), ('E', 65), ('L', 45), ('R', 39), ('S', 39)]
Trigramas encontrados: ('IEN', 'IOR', 'ORD', 'OCO', 'OLB', 'ORE', 'OOV', 'VMB', 'EST', 'OMO', 'VSO', 'OCV', 'POR', 'OSV', 'COM', 'STR', 'BRE', 'OVR', 'COG', 'SIE', 'ORO', 'TRE', 'EVO', 'OES', 'ELO', 'SCO', 'NOC', 'ISE', 'CLO', 'OSO', 'EEL', 'ROE', 'ESE', 'STO', 'ATO', 'OSC', 'ROV', 'ELL', 'DOS', 'EST', 'OIO', 'REE', 'ESV', 'LOT', 'ORC', 'OSI', 'OOL', 'LOO', 'DOS', 'LOS', 'COR', 'VEL', 'LLE', 'EES', 'CIE', 'ROO', 'ICO', 'LRO', 'EVO', 'QVE', 'VEI', 'REL', 'QOV', 'EIO', 'ODE', 'OCI', 'OJO', 'DES', 'OMO', 'TOR', 'LOR', 'MOS', 'OLO', 'ELO', 'SOJ', 'ONT', 'LOL', 'ODO', 'RCO', 'VRO', 'LOH', 'ENT', 'LOD', 'VOS', 'RIC', 'ROR', 'IOL', 'RES', 'CIO', 'LLO', 'GRO', 'TVS', 'DEN', 'LEV', 'ORD', 'DEL', 'ENO', 'OSO', 'TOS', 'ROS')
Distancias: [377, 377, 377, 31, 296, 166, 385, 115, 402, 251, 353, 169, 3, 14, 57, 3, 456, 456, 31, 250, 282, 172, 14, 316, 345, 345, 323, 131, 287, 230, 192, 271, 87, 19, 19, 68, 39, 55, 41, 376, 137, 14, 75, 1
9, 19, 57, 288, 288, 288, 288, 31, 9, 95, 95, 9, 255, 39, 33, 199, 195, 41, 158, 311, 55, 118, 118, 115, 68, 113, 288, 288, 288, 39, 33, 19, 19, 172, 87, 87, 19, 19, 75, 14, 22, 30, 68, 87, 19, 19, 166, 136, 136
, 136, 131, 22, 244, 19, 75, 95, 95, 30, 169, 284, 266, 31, 23, 230, 294, 300, 172, 137, 14, 23, 60, 24, 24, 24, 241, 241, 241, 196, 196, 87, 202, 79, 79, 79, 79, 192, 118, 118, 113, 24, 24, 24, 41,
89, 6, 6, 285, 6, 6, 231, 251, 6, 158, 230, 6, 258, 87, 282, 87, 19, 19, 57, 208, 288, 288, 288, 31, 296, 9, 195, 87, 199, 195, 136, 136, 136, 69, 19, 172, 238, 79, 79, 79, 79, 192, 271, 251, 24, 58, 287, 28
9, 288, 288, 89, 113, 14, 68, 82, 316, 31, 255, 158, 78, 87, 323, 60, 323, 377, 377, 377, 69, 14, 316, 345, 345, 58, 258, 282, 385, 286, 79, 286, 402, 24, 195, 79, 79, 244, 196, 196, 311, 238, 345, 345, 62, 316,
31, 78, 376, 17, 24, 271, 90, 205, 230, 17, 24, 241, 241, 196, 196, 57, 284, 16, 282, 3, 456, 456, 377, 231, 16, 294, 376, 137, 14, 316, 294, 300]
```



II. SOLUCIÓN DEL CUESTIONARIO

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

Algunos criptosistemas pueden utilizar la permutación u otras operaciones matemáticas como parte de su proceso de cifrado. Por ejemplo, el Estándar de Cifrado Avanzado (AES, por sus siglas en inglés) utiliza una combinación de sustitución, permutación y otras operaciones matemáticas para cifrar el texto plano. Estas operaciones son necesarias para hacer el proceso de cifrado más seguro y resistente a ataques. Al aplicar

	<p align="center">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 10

estas operaciones al texto plano, el texto cifrado resultante se vuelve más difícil de descifrar sin la clave de descifrado correcta.

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

Uno de ellos es la posibilidad de introducir errores en los datos durante el preproceso, lo que podría afectar la integridad de la información original. Además, si el preproceso no se realiza correctamente, podría debilitar la seguridad del sistema y facilitar posibles ataques o vulnerabilidades. Por lo tanto, es importante llevar a cabo un análisis cuidadoso y riguroso al implementar cualquier operación de preprocesamiento para minimizar estos riesgos y garantizar la robustez del criptosistema.

III. CONCLUSIONES

En el estudio de la criptografía, hemos explorado conceptos fundamentales como el cifrado, el preprocesamiento y el método Kasiski. A través de ejemplos prácticos y el uso de funciones en Python, hemos aprendido a realizar diversas operaciones criptográficas, como la sustitución de caracteres, el cálculo de frecuencias y el análisis de trigramas. Estas habilidades nos permiten comprender y aplicar técnicas criptográficas para garantizar la seguridad de la información.

También hemos comprendido la importancia del preprocesamiento adecuado en los criptosistemas. El manejo correcto de caracteres, tildes, codificaciones y relleno de texto son aspectos esenciales para lograr un cifrado eficiente y seguro. Sin embargo, debemos ser conscientes de los riesgos potenciales, como vulnerabilidades y pérdida de información, al implementar operaciones de preprocesamiento, por lo que se requiere un enfoque cuidadoso y una evaluación constante.

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: 11