Laboratorio 1: Tel252

Javiera Fuentes, Nicolás Vergara, Pablo Gajardo, Camilo Muñoz y Vicente Moya

August 26, 2024

1 Introducción

1.1 Contexto

El presente laboratorio está diseñado para introducir a los estudiantes en el uso de la plataforma CoCalc para la implementación y análisis de métodos de cifrado clásicos, específicamente el cifrado César, utilizando la herramienta SageMath. La criptografía, siendo una disciplina fundamental en la seguridad de la información, requiere una comprensión profunda de los algoritmos que han sido la base de su desarrollo a lo largo de la historia. En este sentido, el cifrado César representa un primer acercamiento a los conceptos esenciales de la criptografía, permitiendo a los estudiantes experimentar con la codificación y descodificación de mensajes, así como con técnicas básicas para romper sistemas de cifrado [1].

El uso de SageMath en este contexto no solo facilita la implementación de algoritmos criptográficos, sino que también permite realizar ataques como el de fuerza bruta, proporcionando una plataforma práctica para la experimentación. Este laboratorio busca fortalecer la comprensión teórica a través de la aplicación práctica, estableciendo las bases para estudios más avanzados en criptografía y seguridad informática.

1.2 Objetivos

- **Familiarización con CoCalc y SageMath:** Facilitar a los estudiantes la comprensión y uso de la plataforma CoCalc, enfocándose en las funcionalidades de SageMath para la implementación de algoritmos criptográficos.
- **Implementación del cifrado César:** Guiar a los estudiantes en la implementación de funciones que realicen tanto el cifrado como el descifrado de textos utilizando el cifrado César, abordando también variantes para diferentes configuraciones de texto.
- **Realización de ataques criptográficos básicos:** Instruir a los estudiantes en la realización de ataques de fuerza bruta sobre textos cifrados, explorando la efectividad de este método y analizando los resultados obtenidos.
- **Consolidación de conocimientos teóricos a través de la práctica:** Integrar la teoría criptográfica con la práctica, permitiendo a los estudiantes visualizar y comprender los principios que sustentan la criptografía clásica [1].

2 Metodología

2.1 Flujo del laboratorio

- En primera instancia tanto el ayudante como el profesor introducen la herramienta CoCalc y a continuación dan tiempo a los estudiantes para crear una cuenta en la plataforma.
- Posterior a esto se enseña a los estudiantes como realizar un proyecto y el sistema de archivos que utiliza CoCalc. Se indica que se trabajará con el lenguaje SageMath y como crear archivos con esta extensión.

- Una vez dentro de un archivo *.sagews* se indican algunos comandos básicos, como crear funciones y sintaxis del lenguaje.
- Luego se muestra el código césar y su funcionamiento, así también, como desencriptarlo y mostrar los resultados por pantalla.
- Finalmente se explica el concepto de ataque y en esta oportunidad, se utiliza un ataque de fuerza bruta sobre el codigo césar para mostrar su vulnerabilidad.

2.2 Ejercicios desarrollados durante la clase

- A partir de un string que contiene todas las letras del alfabeto inglés, se crean las funciones auxiliares para transformar el índice de cada letra en un número y viceversa.
- Se crea la función para encriptar usando el código césar.
- Se crea la función para desencriptar el string generado a partir del código césar.
- Se realizan pruebas tanto para ambas funciones, mostrando que fueron implementadas de forma correcta.
- Se crea una función de ataque por fuerza bruta a partir de distintas posibles claves.
- Se muestra como es posible desencriptar el código generado por la función de encriptación a través de este método para distintos casos.

  El código generado para esta experiencia puede ser encontrado en el siguiente repositorio de github.

3 Ejercicios Prácticos

3.1 Parte 1

Implementar funciones que pueden ser utiles para los algoritmos de cifrados clásicos.

1. Implementa una función que retorne "True" si y sólo si el caracter 'c' pertenece al alfabeto inglés.

```
"""
se evalua que el caracter este en la tabla de valores ASCII, especificamente en el
rango de letras minusculas que corresponde al 97 hasta el 122.
"""
def ingles_alfabeto(letra,bool=False):
    if(ord(letra.lower())>=97 and ord(letra.lower())<=122):
        bool = True
    return bool
```

2. Implementa una función que convierta un caracter simple en su valor numerico correspondiente.

```
"""
dado que la letra 'a' se ubica en la posicion 97 en ASCII, se debe hacer el desplzamiento
de cada posicion respecto 97 unidades.
"""
def convertir_numero(letra):
    return ord(letra.lower())-97
```

3. Implementa una función que retorne el caracter correspondiente a 'x mod 26'.

```
"""
funcion que retorna la operacion mod 26 sobre la suma de key mas el valor de la letra
como parametro (c+k), siendo este llave_letra.
"""
def mod(letra_llave):
    return letra_llave % 26
```

3.2 Parte 2

Implementar funciones de Sage para el cifrado/descifrado con el cifrado. César, así como ataques

1. Implementa funciones de cifrado/descifrado.

```
"""
funcion cifrado de una cadena de texto mediante algoritmo de cesar.
"""
def cifrado(key,palabra):
    palabra_cifrada = []
    for letra in palabra:
        if(ingles_alfabeto(letra)):
            mover = mod((convertir_numero(letra))+key)
            palabra_cifrada.append(chr(mover+97))
        else:
            palabra_cifrada.append(letra)
    return ''.join(palabra_cifrada)

"""
funcion de descifrado, aqui se optimiza el uso de lineas.
"""
def descifrado(key,palabra):

    original = [

        chr(mod((convertir_numero(letra))-key)+97)
        if(ingles_alfabeto(letra))
        else letra
        for letra in palabra
    ]
    return ''.join(original)
```

2. Implementa una función que realice un ataque de fuerza bruta sobre un texto cifrado.

```
"""
funcion de ataque
"""
def ataque(palabra,subcadena=''):
    descifrada = []

    for key in range(26):
        descifrada = descifrado(key,palabra)
        if(subcadena in descifrada and subcadena != ''):

            print('Key : {0}  --> Palabra : {1}'.format(key,descifrada))

        if(subcadena == ''):

            print('Key : {0}  --> Palabra : {1}'.format(key,descifrada))
```

3. Muestra la salida de tu función de cifrado (parte a) en los siguientes pares (clave, texto plano):

   - $k = 6$, texto plano = "Get me a vanilla ice cream, make it a double."
   - $k = 15$, texto plano = "I don't much care for Leonard Cohen."
   - $k = 16$, texto plano = "I like root beer floats."

```
prueba_1 = {
    6:'et me a vanilla ice cream, make it a double.',
    15:'don't much care for Leonard Cohen.',
    16:'I like root beer floats.'
}

for llave,palabra in prueba_1.items():
    print(cifrado(llave,palabra))

# Resultado en el terminal:

kz sk g bgtorrg oik ixkgs, sgqk oz g juahrk.
sdc'i bjrw rpgt udg atdcpgs rdwtc.
y byau heej ruuh vbeqji.
```

4. Muestra la salida de tu función de descifrado (parte a) en los siguientes pares (clave, texto cifrado)

   - $k = 12$, texto cifrado = "nduzs ftq buzq oazqe."
   - $k = 3$, texto cifrado = "fdhvdu qhhgv wr orvh zhljkw."
   - $k = 20$, texto cifrado = "ufgihxm uly numnys."

```
prueba_2 = {

    12:'nduzs ftq buzq oazqe.',
    3:'fdhvdu qhhgv wr orvh zhljkw.',
    20:'ufgihxm uly numnys.'
}

for llave,palabra in prueba_2.items():
    print(descifrado(llave,palabra))

# Resultado en el terminal:

bring the pine cones.
caesar needs to lose weight.
almonds are tastey.
```

5. Muestra la salida de tu función de ataque (parte b); si se especifica una palabra clave opcional:
   - texto cifrado = 'gryy gurz gb tb gb nzoebfr puncry.', palabra clave = 'chapel'
   - texto cifrado = 'wziv kyv jyfk nyve kyv tpdsrcj tirjy.', palabra clave = 'cymbal'
   - texto cifrado = 'baeeq klwosjl osk s esf ozg cfwo lgg emuz.', sin palabra clave"

```
prueba_3 = {

    'gryy gurz gb tb gb nzoebfr puncry.':'chapel',
    'wziv kyv jyfk nyve kyv tpdsrcj tirjy.':'cymbal',
    'baeeq klwosjl osk s esf ozg cfwo lgg emuz.':''
}

for palabra,subcadena in prueba_3.items():
    print('\nAtaque de descifrado de : ',palabra,' con sub cadena : ',subcadena)
    ataque(palabra,subcadena)

# Resultado en el terminal:

Ataque de descifrado de :  gryy gurz gb tb gb nzoebfr puncry.  con sub cadena :  chapel
Key : 13  --> Palabra : tell them to go to ambrose chapel.

Ataque de descifrado de :  wziv kyv jyfk nyve kyv tpdsrcj tirjy.  con sub cadena :  cymbal
Key : 17  --> Palabra : fire the shot when the cymbals crash.

Ataque de descifrado de :  baeeq klwosjl osk s esf ozg cfwo lgg emuz.  con sub cadena :
Key : 0  --> Palabra : baeeq klwosjl osk s esf ozg cfwo lgg emuz.
Key : 1  --> Palabra : azddp jkvnrik nrj r dre nyf bevn kff dlty.
Key : 2  --> Palabra : zycco ijumqhj mqi q cqd mxe adum jee cksx.
Key : 3  --> Palabra : yxbbn hitlpgi lph p bpc lwd zctl idd bjrw.
Key : 4  --> Palabra : xwaam ghskofh kog o aob kvc ybsk hcc aiqv.
Key : 5  --> Palabra : wvzzl fgrjneg jnf n zna jub xarj gbb zhpu.
Key : 6  --> Palabra : vuyyk efqimdf ime m ymz ita wzqi faa ygot.
Key : 7  --> Palabra : utxxj dephlce hld l xly hsz vyph ezz xfns.
Key : 8  --> Palabra : tswwi cdogkbd gkc k wkx gry uxog dyy wemr.
Key : 9  --> Palabra : srvvh bcnfjac fjb j vjw fqx twnf cxx vdlq.
Key : 10  --> Palabra : rquug abmeizb eia i uiv epw svme bww uckp.
Key : 11  --> Palabra : qpttf zaldhya dhz h thu dov ruld avv tbjo.
Key : 12  --> Palabra : posse yzkcgxz cgy g sgt cnu qtkc zuu sain.
Key : 13  --> Palabra : onrrd xyjbfwy bfx f rfs bmt psjb ytt rzhm.
Key : 14  --> Palabra : nmqqc wxiaevx aew e qer als oria xss qygl.
Key : 15  --> Palabra : mlppb vwhzduw zdv d pdq zkr nqhz wrr pxfk.
Key : 16  --> Palabra : lkooa uvgyctv ycu c ocp yjq mpgy vqq owej.
Key : 17  --> Palabra : kjnnz tufxbsu xbt b nbo xip lofx upp nvdi.
Key : 18  --> Palabra : jimmy stewart was a man who knew too much.
Key : 19  --> Palabra : ihllx rsdvzqs vzr z lzm vgn jmdv snn ltbg.
Key : 20  --> Palabra : hgkkw qrcuypr uyq y kyl ufm ilcu rmm ksaf.
Key : 21  --> Palabra : gfjjv pqbtxoq txp x jxk tel hkbt qll jrze.
Key : 22  --> Palabra : feiiu opaswnp swo w iwj sdk gjas pkk iqyd.
Key : 23  --> Palabra : edhht nozrvmo rvn v hvi rcj fizr ojj hpxc.
Key : 24  --> Palabra : dcggs mnyquln qum u guh qbi ehyq nii gowb.
Key : 25  --> Palabra : cbffr lmxptkm ptl t ftg pah dgxp mhh fnva.
```

4 Conclusiones

4.1 Análisis general

Se implementaron las funciones de búsqueda de caracteres, transformación de caracteres y módulo 26, todas las cuales entregaron los resultados esperados para su uso en funciones más complejas. Las funciones de encriptamiento y desencriptamiento, se probaron utilizando los ejemplos proporcionados en la guía del laboratorio. Los textos planos y encriptados procesados por las funciones cumplen con los resultados de las operaciones necesarias para simular el algoritmo de César. Además,la función de ataque de fuerza bruta fue implementadas correctamente, incluyendo la opción para insertar un subtexto que permite encontrar posibles claves que contengan dicha palabra.

Finalmente se cumplieron los objetivos principales de esta entrega. Se utilizó la herramienta SageMath para desarrollar el código de encriptamiento. Estos constituyen los primeros pasos y la base para avanzar en el curso. Gracias al trabajo práctico planteado en el laboratorio, se ha logrado una comprensión más profunda de la metodología de encriptación básica.

4.2 Reflexiones

El laboratorio realizado ha permitido profundizar en el entendimiento de los conceptos básicos de encriptación, así como para familiarizarse con el uso de SageMath como herramienta de desarrollo para la creacion de algoritmos. La implementación de funciones esenciales como la búsqueda y transformación de caracteres y el módulo 26 no solo ha permitido establecer una base sólida para operaciones más complejas, sino que también ha demostrado la importancia de estos componentes en la simulación de algoritmos de encriptación, específicamente el algoritmo de César. Ademas, se pudo conocer las vulnerabilidades del mismo, mediante la implemnetacionde un algoritmo de descifrado, el ataque de fuerza bruta. Finalmente, en esta experiencia se pudo verificar y validar el funcionamiento del algoritmo de César. De esta manera crear conciencia de los casos que seria beneficioso implementarlo y tener siempre en cuenta sus vulnerabilidades de manera que podamos evitar o contener un ataque en caso de ser necesario.

References

[1] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 7th ed. London, UK: Pearson, 2017.