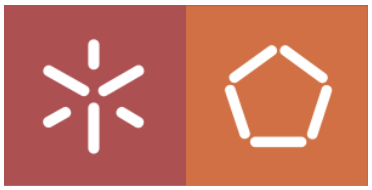


Universidade do Minho
Escola de Engenharia

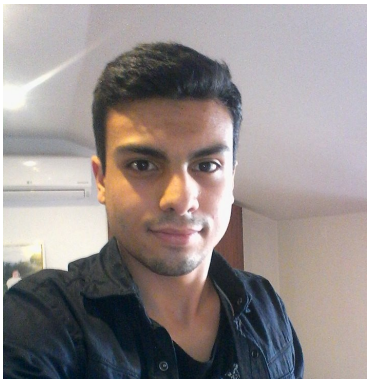
Universidade do Minho
Mestrado Integrado em Engenharia Informática

Tecnologia Criptográfica

Trabalho prático 2

6 Dezembro 2020

Ana Margarida Campos
(A85166)



Nuno Pereira
(PG42846)

Contents

Introdução

1.1 Contextualização

O presente relatório foi elaborado no âmbito do primeiro Trabalho Prático da Unidade Curricular de Tecnologia Criptográfica, que se insere no 1º semestre do 4º ano do Mestrado Integrado em Engenharia Informática (1º ano MEI).

1.2 Objetivos e Trabalho Proposto

Para este trabalho foram fornecidos 20 criptogramas gerados a partir de *One Time Pad* que foi utilizado de uma forma incorreta. Estes criptogramas correspondem a duas mensagens de texto-limpo distintas cifradas com chaves diferentes, excepto no caso de dois criptogramas que foram cifrados utilizando a mesma chave. O objetivo principal deste trabalho prático consiste na descoberta de quais os criptogrmas que utilizam a mesma chave.

1.3 Estrutura do Relatório

Este relatório encontra-se divido em três secções. Na primeira secção foi feita uma breve introdução do trabalho proposto. Na secção seguinte é abordado o método criptográfico *One Time Pad*. Posteriormente é descrita a estratégia utilizada para o desenvolvimento do código em *python*. Por último, é apresentada uma conclusão à cerca do trabalho desenvolvido.

One Time Pad

2.1 Definição

One Time Pad é uma técnica criptográfica que combina, caractere a caractere, a mensagem de texto-limpo com uma chave secreta, gerada de preferência de uma forma aleatória. De forma a ser seguro e impenetrável, a chave tem de ter no mínimo o mesmo tamanho que o texto-limpo e só pode ser usada uma única vez. Se de facto estes requisitos forem cumpridos, é considerado um algoritmo de segurança perfeita. Esta técnica criptográfica também é vista como sendo computacionalmente segura uma vez que, mesmo utilizando uma máquina, é necessário muito tempo (certas vezes tempo infinito) para quebrar a cifra.

Neste método a cada letra é atribuído um valor numérico: por exemplo A=0, B=1,..., Z=25. Após a passagem da mensagem e da chave para os seus valores numéricos correspondentes, a técnica de cifragem utilizada consiste em combinar a chave e a mensagem usando a adição modular. De modo a decifrar um criptograma, sabendo a chave, é utilizado o processo inverso ou seja a subtração modular.

Programa Desenvolvido

3.1 Estratégia Utilizada

A estratégia utilizada consistiu na conversão de caracteres dos criptogramas para valores numéricos, como explicado anteriormente, e na posterior subtração modular de cada criptograma com todos os outros menos com ele próprio. O resultado das subtrações entre dois criptogramas distintos que utilizam a mesma chave é constituído por mais zeros que os restantes. Isto porque se por exemplo, uma mensagem for cifrada usando a mesma chave e posteriormente subtraída por ela própria o resultado será todo composto por zeros, visto que todas as letras são iguais e os correspondentes valores numéricos também. Baseando-nos nesta ideia, os criptogramas que usam a mesma chave, se tiverem letras iguais o resultado da subtração modular dará zero. Ou seja, o nosso programa foi desenvolvido com o intuito de descobrir quais os dois criptogramas que subtraídos dão mais zeros e que portanto usaram a mesma chave. O programa realizado encontra-se nos anexos deste relatório.

3.2 Resultados Obtidos

Utilizando a estratégia descrita anteriormente, os criptogramas que usam a mesma chave correspondem aos criptogramas números 6 e 14 do ficheiro disponibilizado. O resultado das subtrações modulares dá um número de zeros mais elevado que os outros criptogramas, sendo este valor de 371 zeros. Logo, visto ser o maior valor existe uma maior probabilidade de estes terem sido cifrados partilhando a mesma chave.

Conclusão

Com este trabalho prático foi possível pôr em prática vários conhecimentos obtidos durante as aulas de Tecnologia Criptográfica, nomeadamente sobre o método criptográfico *One Time Pad*.

Com o trabalho desenvolvido podemos concluir que, de facto, este método não é seguro se a mesma chave for utilizada. Para o método ser seguro é preciso, como dito ao longo do relatório, a chave ser gerada aleatoriamente, nunca ser reutilizada e ter um tamanho igual ou superior ao tamanho da mensagem de texto-limpo.

Código do Programa

```
'''classe para guardar uma lista com as subtrações modulares
   das letras dos criptogramas convertidas para valores
   numéricos e os respetivos ids. Zeros indica o número
   de zeros de cada subtração.
'''
class cripto:
    def __init__(self, lista, id1, id2, zeros):
        self.lista = lista
        self.id1 = id1
        self.id2 = id2
        self.zeros = zeros


''' Recebe os criptogramas e dá como resultado
    os criptogramas cuja subtração deu
    mais zeros
'''
def getCriptogramas():
    f = open("criptogramas", "r")
    lista_criptogramas = []
    index = 0
    string = ""
    ''' cópia de cada criptograma do ficheiro para uma lista '''
    for i, char in enumerate(f.read()):
        if char != '\n':
            string += char
        if char == '\n':
            lista_criptogramas.insert(index, string)
            index += 1
            string = ""
    lista_numeros_total = []
    ''' Conversão dos criptogramas para valores numéricos '''
    for l in lista_criptogramas:
        lista_numeros = []
        for char in l:
            lista_numeros.append(ord(char)-65)
        lista_numeros_total.append(lista_numeros)

    indice = 0
    lista_subtracoes = []
    ''' Subtrações de todos os criptogramas com todos menos com ele próprio '''
```

```
    for i in range(20):
        for j in range(20):
            if i != j:
                lista = subtract(lista_numeros_total[i], lista_numeros_total[j])
                lista_subtracoes.append(cripto(lista, i, j, -1))
                indice += 1

    i = 0
    ''' conta o número de zeros de cada criptograma'''
    for l in lista_subtracoes:
        for t in l.lista:
            if t == 0:
                i += 1
        l.zeros = i
        i = 0

    ''' Coloca os valores na classe e vê qual o que tem mais zeros'''
    maior = cripto([], -1, -1, -1)
    for indice in enumerate(lista_subtracoes):
        if (maior.zeros < indice.__getitem__(1).zeros):
            maior = cripto(indice.__getitem__(1).lista, indice.__getitem__(1).id1,
                           indice.__getitem__(1).id2, indice.__getitem__(1).zeros)
    return maior


''' Subtração modular de duas listas '''
def subtract(list1, list2):
    lista = []
    for i in range(len(list1)):
        lista.append((list1[i]-list2[i]) % 26)
    return lista


if __name__ == '__main__':
    maior = getCriptogramas()
    print(maior.id1 + 1, maior.id2 + 1) # somamos + 1 porque o índice da lista começa em 0
```