



Universidade do Minho
Escola de Engenharia

Criptografia

TP2

						95	1100
JTB PADLH4RQBHV							
24765	93659	55146	09380	18882	67898	69598	
25341	88038	31282	39057	21708	51305	66499	
65096	02819	74377	27960	20471	53361	18687	
19226	31329	55134	83869	26588	24850	81322	
01334	80225	37061	13995	88627	07293	53021	
90865	91712	80927	18799	71311	57151	71976	
98890	61224	59636	08076	65747	36834	49525	
95428	50476	06584	38300	37155	75549	11968	
43041	83175	29737	88523	76769	29465	47144	
77230	19601	57378	51440	48030	63857	15846	
32548	48508	71999	22399	86499	22365	91365	
57311	83798	06280	74855	58916	46616	07784	
10464	00582	08702	30607	80017	50120	76361	
93610	38382	57828	27710	00947	00977	02927	
53217	20255	20839	63759	74408	60213	32159	
31617	14857	97505	25301	14258	36792	42161	
52190	32626	07392	80180	32382	22884	82072	
39585	92345	44974	09467	88114	50678	84634	
44347	73204	49702	60171	56691	11969	32188	
86460	37447	03000	03670	05391	96625	21870	

João Miguel da Silva Alves (83624)

Paulo Jorge Alves (84480)

MESTRADO INTEGRADO EM ENGENHARIA BIOMÉDICA
INFORMÁTICA MÉDICA 2020/2021

O *one-time-pad* usa uma chave aleatória tão grande quanto a mensagem, de modo a que a chave não precise de ser repetida. Além disso, a chave é usada para cifrar e decifrar uma única mensagem, sendo por fim descartada. Cada nova mensagem exige uma nova chave com o mesmo tamanho. Esta cifra é inquebrável. Produz uma saída aleatória que não possui qualquer relacionamento estatístico com o texto claro. Como o criptograma não contém informação sobre o texto limpo, simplesmente não existe um meio de quebrar o código.

No entanto, neste exercício, sabe-se que dois criptogramas resultaram da encriptação com a mesma chave. A regra geral diz que nunca se deve usar a mesma chave mais de uma vez. Caso contrário, a cifra começa a ficar vulnerável. Este é um exemplo deste caso.

Posto isto, considerando C1 um criptograma e C2 outro criptograma, e sendo m1 e m2 as duas mensagens que foram cifradas com a mesma chave k, sabe-se que:

$$C1 = (m1 + k) \bmod 26 \qquad C2 = (m2 + k) \bmod 26$$

Assim, de modo a eliminar k:

$$C1 - C2 = m1 - m2$$

Como C1 e C2 foram encriptados com a mesma chave, a subtração (módulo 26) dos dois criptogramas vai mostrar um padrão de letras "A" (correspondente ao número 0).

Para descobrir os dois criptogramas que foram encriptados com a mesma chave usou-se o código Python abaixo:

```
1 resultado=[]
2 primeiro=0
3 while primeiro<20:
4     resultado.append('\n')
5     segundo=0
6     while segundo<20:
7         resultado.append('\n')
8         K=list(text[primeiro])
9         L=list(text[segundo])
10        S=[]
11        for i in range(len(K)):
12            s=chr((((ord(K[i])-65) - (ord(L[i])-65))%26)+65)
13            S.append(s)
14        resultado.append(''.join(S))
15        segundo=segundo+1
16    primeiro=primeiro+1
17
18 with open("resultado.txt",'w') as f1:
19     f1.write(''.join(resultado))
20
21 i=0
22 while i < len(resultado):
23     if resultado[i] != '\n':
24         string=resultado[i]
25         ocorrencia={'A':0}
26         for b in string:
27             if (b=='A'):
28                 ocorrencia[b]=ocorrencia[b]+1
29         lista=List(ocorrencia.items())
30         print(lista)
31     i=i+1
```

Figura 1 - Código Python

Assim, para resolver o problema em causa, a ideia foi fazer todas as subtrações (módulo 26) possíveis entre os 20 criptogramas, e por fim fazer a contagem das ocorrências da letra “A” em cada um. O texto com maior quantidade de “A”, vai permitir saber quais os criptogramas resultaram da encriptação com a mesma chave.

Começou-se por colocar todos os criptogramas numa lista *text*. Percorrendo essa lista 2 vezes (de forma a operação a realizar seja efetuada entre todos os elementos da lista, ou seja, entre todos os criptogramas), e transformando também essas strings em listas (linhas 8 e 9 do código da figura 1), realiza-se a subtração (módulo 26) entre estas 2 listas.

Para cada uma dessas listas, faz-se a subtração (módulo 26) elemento a elemento, como por exemplo, querendo subtrair (em modulo 26) o “A” pelo “C”:

$$"A" \rightarrow 0 \text{ e } "C" \rightarrow 2, \text{então } "A" - "C" = -2 + 26 = 24 \rightarrow "Y"$$

Desta forma, usando as funções Python `ord()` e `chr()` para o efeito, obtém-se uma lista com todas as subtrações (módulo 26) possíveis.

De seguida, percorrem-se todos os elementos desta lista (sublistas), e registam-se as ocorrências da letra “A” através de um dicionário. No fim é impressa uma lista com os dicionários, em que cada dicionário corresponde às ocorrências de “A” em cada texto, e todos os textos são escritos num ficheiro resultado.txt.

Após analisar todos os dicionários, verifica-se que a letra A aparece 371 vezes (valor máximo de ocorrências em todos os textos) num dos textos (figura 2), sendo esse o texto o corresponde à subtração do criptograma 6 (“LXBVRMY...”) com o 14 (“HCOGIIGQ...”).



Figura 2 - Texto, onde se verificou o maior número de “A”, resultante da subtração do criptograma 6 com o 14.

Assim, chega-se à conclusão de que os dois criptogramas que foram encriptados pela mesma chave correspondem ao criptograma 6 e ao criptograma 14.

Desta forma provou-se que a cifra *one-time-pad* só é inquebrável se a chave for usada uma única vez e por fim descartada. Caso contrário, o adversário consegue ter pistas credíveis sobre a mensagem original, que podem levar à sua decifragem.