Week 6 - 18/11/2022

Exercício 1

```
#!/usr/bin/env sage -python
import sys
import numpy as np
from sage.all import *

p = 7
q = 11
c = 20
e = 13

phi = (p-1)*(q-1) #cáluclo do phi

_,d,_= xgcd(e,phi) #cálculo do inverso multiplicativo de phi que corresp

m = (20**37) % 77 #plaintext

print(m)
```

48

Exercício 2:

Dado o n ser um número pequeno, é possível fatorizá-lo no terminal com recurso ao comando $ factor, depois, sabendo a chave pública e tendo os 2 primos que compõe n, fica fácil obter a chave privada, do seguinte modo:

```
#!/usr/bin/env sage -python
import sys
import numpy as np
from sage.all import *

n = 2881
e = 65
p = 43 #Obtido com recurso a factor
q = 67 #Obtido com recurso a factor

phi = (p-1)*(q-1) #Cálculo do phi

_,d,_ = xgcd(e, phi)
# cálculo do inverso multiplicativo de phi que corresponde à chave priva


#condição necessária para o resultado dar correto
if (d<0):
    d = phi+d



print('(d: '+ str(d) + ', n:' + str(n) + ')')
```

(d: 725, n:2881)

Exercício 3:

Ao não gerar um novo módulo, o Bob leva a que um atacante, tenha acesso à antiga chave privada publicada e, dado o caráter público da mesma, à antiga chave pública. Assim sendo, tendo acesso a estas duas chaves, é possível descobrir o phi(n) que o Bob não quer mudar. Com este phi(n) e com a nova chave pública, que também se encontra disponível para o atacante, é possível calcular o inverso multiplicativo e descobrir a chave privada, o que torna esta primitiva, insegura.

Exercício 4:

Dado o atacante possuir a chave pública e o tamanho da mensagem enviada ser bastante reduzido (assumindo que é um número portugês, seriam 9 números), utilizando a chave pública pode realizar um ataque de dicionário no qual cifra todas as combinações de 9 dígitos possíveis (sendo que este número pode reduzir assumindo que começará por 9 se for um número móvel) e ver quando é que o texto cifrado coincide com o enviado.