Week 4 - 14/10/2022

Exercício 1

```
F = FiniteField(2**(130)-5)
PR.<X> = PolynomialRing(F)
Key = (F.random_element(), F.random_element())
Message = [F.random_element() for x in range(5)]
def universal(K, M):
    K1 = Key[0]
    K2 = Key[1]
    X = K2
    return K1 + PR(Message)
```

Como a função hash universal poly1305 é um construção MAC e o tamanho dos blocos no Poly1305 é de 128 bits, existem colisões após 2^64 hashes geradas, por causa do paradoxo do aniversário. Assim, a probabilidade de colisão da hash de duas mensagens fixas é de 1/(2^64).

Exercício 2

```
from Cryptodome.Cipher import AES
import hashlib
import binascii
import os


plaintext='meter o plaintext aqui'
password=os.urandom(16)

def encrypt(plaintext,key, mode):
    encobj = AES.new(key, AES.MODE_GCM)
    ciphertext,authTag=encobj.encrypt_and_digest(plaintext)
    return(ciphertext,authTag,encobj.nonce)

def decrypt(ciphertext,key, mode):
    (ciphertext,  authTag, nonce) = ciphertext
    encobj = AES.new(key,  mode, nonce)
    return(encobj.decrypt_and_verify(ciphertext, authTag))


print("\nMessage:\t",plaintext)
print("Key:\t\t",password)


ciphertext = encrypt(plaintext.encode(),password,AES.MODE_GCM)

print("Auth Msg:\t",binascii.hexlify(ciphertext[1]))
print("Nonce:\t\t",binascii.hexlify(ciphertext[2]))
print("Cipher:\t\t",binascii.hexlify(ciphertext[0]))


res= decrypt(ciphertext,password,AES.MODE_GCM)


print ("\n\nDecrypted:\t",res.decode())
```

```
Message:     meter o plaintext aqui
Key:         b'T\x9e\xcc\xc7\xa7\xe0\x8a\xf6\x93\x1c.\xfc\x98\xdf\x89"'
Auth Msg:    b'9f56e6a7b44da5726164aa92e70bcb01'
Nonce:       b'bb3b4e2b261504076b63fb999cc8cdb1'
Cipher:      b'89e870c7e908e01d1a4cc6c880778a1bc4ed3385ff9f'


Decrypted:   meter o plaintext aqui
```

Para usar AES - CTR basta substituir MODE_ GCM por MODE_ CTR e criar um counter

através de um Syntetic IV, para utilizar em vez do nonce.

Se mudarmos o ficheiro encriptado, não é possivel desencriptar, já que a autenticação não é bem sucedida.