Hands-on Brute Force Attack

Python and Cryptography package

- I assume each of you have a python distribution installed
- In Windows I have used the package Cryptography (and installed with `pip install cryptography`)
- For instructions on the packages you may look at https://pypi.org/
- Install the package with **`python pip install name_package`**:
  - **`pip install cryptography`**
- Note that the package has many functionalities, we will see only very few of them

Using the Cryptography package

```
import base64
import os
from cryptography.fernet import Fernet
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC

passwd = b"XXXXXXX"
cleartext = b"this is an example of plaintext to encrypt and decrypt"
salt = os.urandom(16)
print('Salt = ',salt)
kdf = PBKDF2HMAC(
     algorithm=hashes.SHA256(),
     length=32,
     salt=salt,
     iterations=100000,
)
key = base64.urlsafe_b64encode(kdf.derive(passwd))
f = Fernet(key)
cyphertext = f.encrypt(cleartext)
print('Cyphertext = ',cyphertext)
print(f.decrypt(cyphertext))
```

The Base64 encoding is used to convert bytes that have binary or text data into ASCII characters. Encoding prevents the data from getting corrupted when it is transferred or processed through a text-only system.

Python has a built-in os module with methods for interacting with the operating system, like creating files and directories, management of files and directories, input, output, environment variables, process management, etc.

PBKDF2 (Password Based Key Derivation Function 2) is typically used for deriving a cryptographic key from a password. Here it sets its parameters.

Derives the key from the password and sets the key in the library

Encrypts and decrypts

Decrypting...

```
import base64
import os
from cryptography.fernet import Fernet
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC

def decrypt(passwd):
    salt = b"\x............" # you should write here the salt obtained when
encrypting
    cyphertext =  b'............' # you should write here the cyphertext obtained
when encrypting
    kdf = PBKDF2HMAC(
        algorithm=hashes.SHA256(),
        length=32,
        salt=salt,
        iterations=100000,
    )
    key = base64.urlsafe_b64encode(kdf.derive(passwd))
    f = Fernet(key)
    try:
        print(f.decrypt(cyphertext))
        print('right password: '+str(passwd)+'\n')
    except:
        print('wrong password: '+str(passwd))
        pass

decrypt(b"TestPass")
```

Exercise

I have used a simple and unsafe password: it is a combination of my own name “Stefano” combined with a number and a lowercase character (any arbitrary placement of these two extra characters is possible, for example “xStefa9no” or “St9xefano”).

I have used this salt for encryption:

```
salt = b"\x9aF\xdb^\xd5\x18\xb0\xe2k\r\xfc\xf3\x7f3\xe0\xb5"
```

```
cyphertext =  b'gAAAAABlJ678-
7eprVhp3wnTslVPcDZzK33bXpQ8WTctjUI8mTobjVwYa7LQfASyRzD2rh1RkB8ufPKsL-
xHJyYaUGJa-dDi8wzx2XQzYV6dnnwbw1NJWxsfeb_Ol9_DhGcxQMm8nqjZw-
6JHzR3_YtQpiZ4083_btWasC_Jg1EEjupDRp0-
vXTwuTuwgYWMLlxwyFox9pCabsieEasHhb8mJFeBhw7xCDbUlLEJLPeUalSUSSv1JuA='
```

The exercise for you is to decrypt the cyphertext. Once decrypted the cleartext you will have discovered my password.