Encryption & Decryption using Fernet and hashlib – Michael Geiger – Unit 4

Different approaches have been used to encrypt a text, which relate to different possible applications.

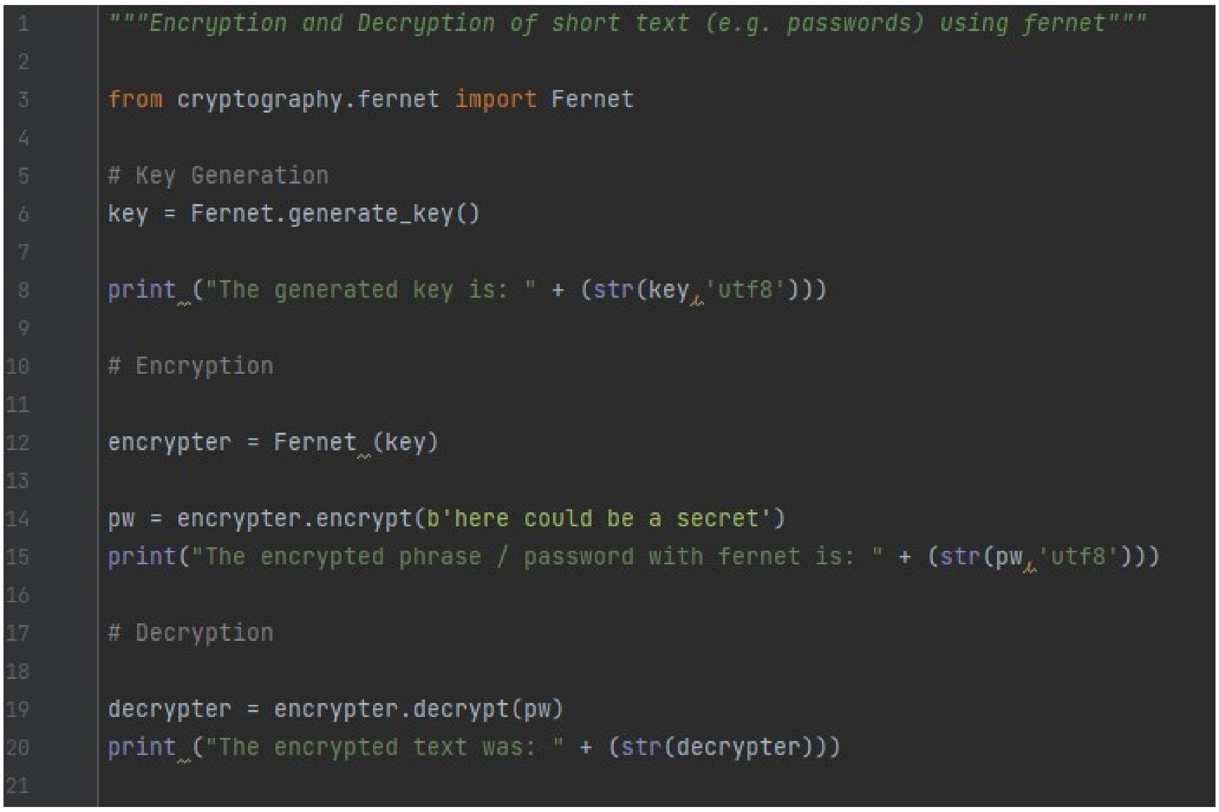
As a challenge for the creation of the encryption code, the approach chosen was to keep the encryption as short and simple as possible while at the same time meeting the requirements of the GDPR.

With the encryption and decryption method using the Fernet module, encrypted transmission of information can be enabled. Since a new key is generated for each encryption process, the possibility for hackers who intercept the transmission to decrypt the text is significantly reduced, since a new key is used for each subsequent transmission. Even if malicious people manage to decrypt one information transmission, a great deal of effort would have to be expended to decrypt the next one, since the same key is not used again. This fulfills the requirements of the GDPR regulations.

With the encryption method using the hashlib module, user names and passwords can be saved in encrypted form. However, decryption is not easily possible. However, this method comes with a relatively high threat. The generated encryption is always made with the same key, the encryption is therefore unique and can be reproduced. While direct decryption is not possible, a database can be created containing all possible word combinations and sha256 ciphers. Now the encryption can be compared with the database. Since such databases are freely accessible and can also be created relatively quickly and easily, this type of encryption is comparatively insecure.

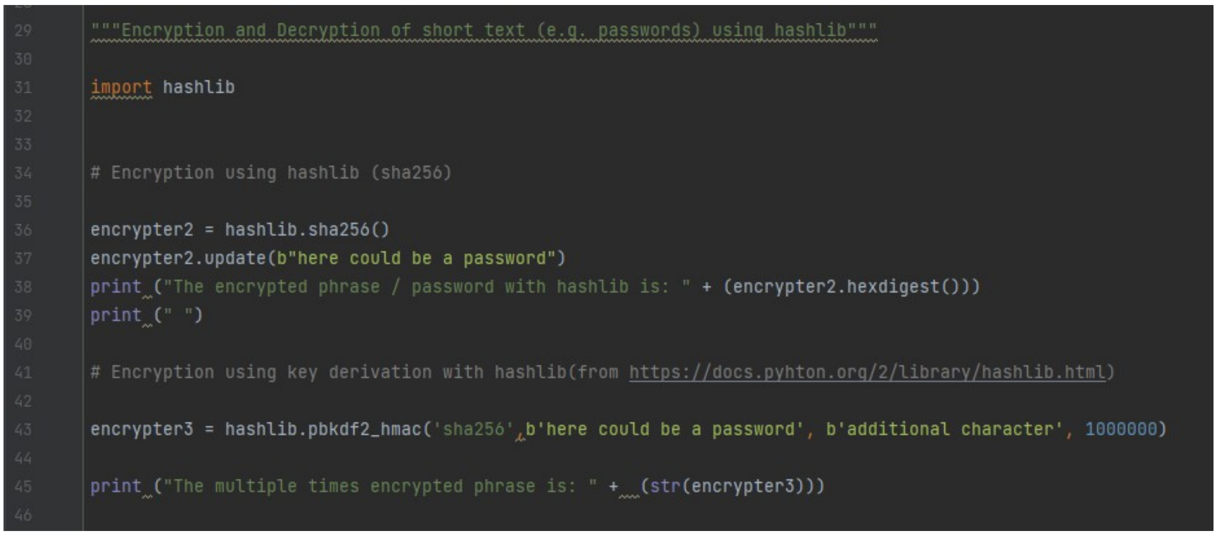
However, in order to increase the security of this encryption, further implementations can be made. On the one hand, an additional text can be included in the text to be encrypted, whereby the original encryption differs from the new one. It has thus become more difficult for a hacker to obtain the actual information. On the other hand, the encrypted information can be re-encrypted several times with sha256, which means that a hacker can no longer easily check the encryption against a database. It must be known how often the encryption was performed in order for the encryption to be successfully checked against a database. This significantly increases security and fulfills the GDPR regulations.

Encryption & Decryption using Fernet:



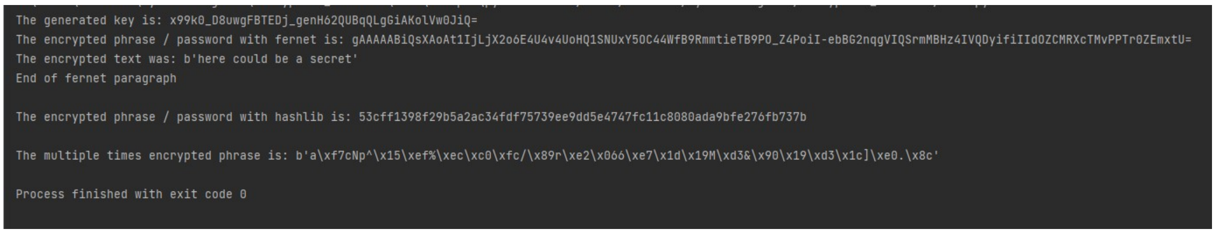
```python
"""Encryption and Decryption of short text (e.g. passwords) using fernet"""

from cryptography.fernet import Fernet

# Key Generation
key = Fernet.generate_key()

print ("The generated key is: " + (str(key,'utf8')))

# Encryption

encrypter = Fernet (key)

pw = encrypter.encrypt(b'here could be a secret')
print("The encrypted phrase / password with fernet is: " + (str(pw,'utf8')))

# Decryption

decrypter = encrypter.decrypt(pw)
print ("The encrypted text was: " + (str(decrypter)))
```

Encryption using hashlib:



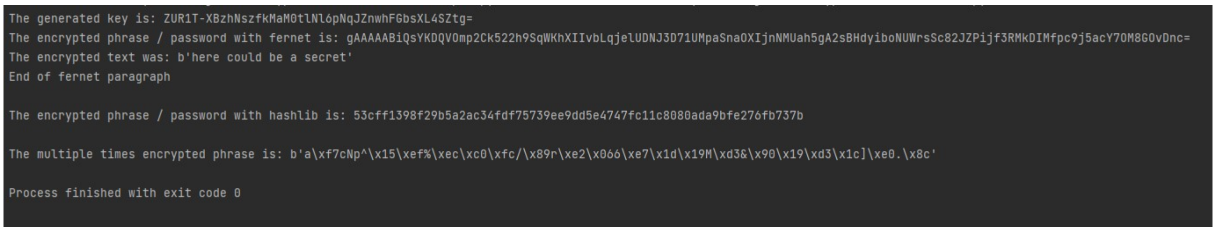
```python
"""Encryption and Decryption of short text (e.g. passwords) using hashlib"""

import hashlib


# Encryption using hashlib (sha256)

encrypter2 = hashlib.sha256()
encrypter2.update(b"here could be a password")
print ("The encrypted phrase / password with hashlib is: " + (encrypter2.hexdigest()))
print (" ")

# Encryption using key derivation with hashlib(from https://docs.pyhton.org/2/library/hashlib.html)

encrypter3 = hashlib.pbkdf2_hmac('sha256',b'here could be a password', b'additional character', 1000000)

print ("The multiple times encrypted phrase is: " + (str(encrypter3)))
```

Example 1:



```
The generated key is: x99k0_D8uwgFBTEDj_genH62QUBqQLgGiAKolVw0JiQ=
The encrypted phrase / password with fernet is: gAAAAABiQsXAoAt1IjLjX2o6E4U4v4UoHQ1SNUxY50C44WfB9RmmtieTB9PO_Z4PoiI-ebBG2nqgVIQSrmMBHz4IVQDyifiIIdOZCMRXcTMvPPTr0ZEmxtU=
The encrypted text was: b'here could be a secret'
End of fernet paragraph

The encrypted phrase / password with hashlib is: 53cff1398f29b5a2ac34fdf75739ee9dd5e4747fc11c8080ada9bfe276fb737b

The multiple times encrypted phrase is: b'a\xf7cNp^\x15\xef%\xec\xc0\xfc/\x89r\xe2\x066\xe7\x1d\x19M\xd3&\x90\x19\xd3\x1c]\xe0.\x8c'

Process finished with exit code 0
```

Example 2:



```
The generated key is: ZUR1T-XBzhNszfkMaM0tlNl6pNqJZnwhFGbsXL4SZtg=
The encrypted phrase / password with fernet is: gAAAAABiQsYKDQVOmp2Ck522h9SqWKhXIIvbLqjelUDNJ3D71UMpaSnaOXIjnNMUah5gA2sBHdyiboNUWrsSc82JZPijf3RMkDIMfpc9j5acY7OM8GOvDnc=
The encrypted text was: b'here could be a secret'
End of fernet paragraph

The encrypted phrase / password with hashlib is: 53cff1398f29b5a2ac34fdf75739ee9dd5e4747fc11c8080ada9bfe276fb737b

The multiple times encrypted phrase is: b'a\xf7cNp^\x15\xef%\xec\xc0\xfc/\x89r\xe2\x066\xe7\x1d\x19M\xd3&\x90\x19\xd3\x1c]\xe0.\x8c'

Process finished with exit code 0
```