AES-Verschlüsselung in Python

Installieren von pyCryptobome erforderlich

```
pip install pycryptodome
```

Danach Import von AES aus dem Modul:

```
from Cryptodome.Cipher import AES
```

Der Schlüssel hat eine Länge von 128, 192 oder 256 Bit. Die Nachricht muss als Bytestream vorliegen, damit diese verschlüsselt werden kann. Dies hat den Vorteil, dass es möglich ist eine Datei direkt einzulesen. In diesem Beispiel muss jedoch ein String in Bytes kodiert werden. Der SChlüssel besteht hier ebefalls aus 24 Byte.

```
nachricht = "Hallo ITO"
key = b'09865rfqghlafgtz78nafg3q'
data = str.encode(nachricht)

print("type of data", type(data))
print("Type of key: ", type(key))
```

Das Verschlüsselungsverfahren wird hier als das Objekt cipher implementiert. Dies setzt sich zum einen aus dem SChlüssel und dem eigentlichen Algorithmus (hier: Mode EAX) zusammen. Bei *Mode EAX* handelt es sich um ein VErfahren, dass neben der Vertraulichkeit (Verschlüsselung) auch die Authentizität über einen Hashwert sicherstellt. Die Methode *encrypt_and_digest()* der Klasse AES hat als Rückgabewert zwei Bytestreams. In Ihr findet die eigentliche Verschlüsselung statt. In *ciphertext* wird der verschlüsselte Nachricht gespeichert, in *messageDigest* der Hashwert zu der Klartextnachricht.

```
cipher = AES.new(key, AES.MODE_EAX)
ciphertext, messageDigest = cipher.encrypt_and_digest(data)

print("Verschlüsselte Daten: ", ciphertext, type(ciphertext), len(ciphertext))
print("Hashwert: ", messageDigest, type(messageDigest), len(messageDigest))
```

Eine Entschlüsselung der Daten kann wie folgt implementiert werden:

```
#Decrypt
nonce = cipher.nonce
cipher = AES.new(key, AES.MODE_EAX, nonce=nonce)
plaintext = cipher.decrypt(ciphertext)

print(plaintext)
```

Den Quellcode oben finden Sie als Beispielimlentierung hier. Er basiert u.a. auf diesem YouTube-Video.