UNIVERSITY OF WOLLONGONG AUSTRALIA

Performing symmetric encryption using Python

Please **ensure you have installed** the **latest "cryptography" package** for your Python.

To install Cryptography module on Python:

- On Command Prompt, type:

pip install cryptography

```
C:\Users\jcl960>pip install cryptography
Defaulting to user installation because normal site-packages is not writeable
Collecting cryptography
  Downloading cryptography-39.0.2-cp36-abi3-win_amd64.whl (2.5 MB)
     ---------------------------------------- 2.5/2.5 MB 9.2 MB/s eta 0:00:00
Collecting cffi>=1.12
  Downloading cffi-1.15.1-cp311-cp311-win_amd64.whl (179 kB)
     ---------------------------------------- 179.0/179.0 kB 10.5 MB/s eta 0:00:00
Collecting pycparser
  Downloading pycparser-2.21-py2.py3-none-any.whl (118 kB)
     ---------------------------------------- 118.7/118.7 kB 6.8 MB/s eta 0:00:00
Installing collected packages: pycparser, cffi, cryptography
Successfully installed cffi-1.15.1 cryptography-39.0.2 pycparser-2.21

[notice] A new release of pip available: 22.3.1 -> 23.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

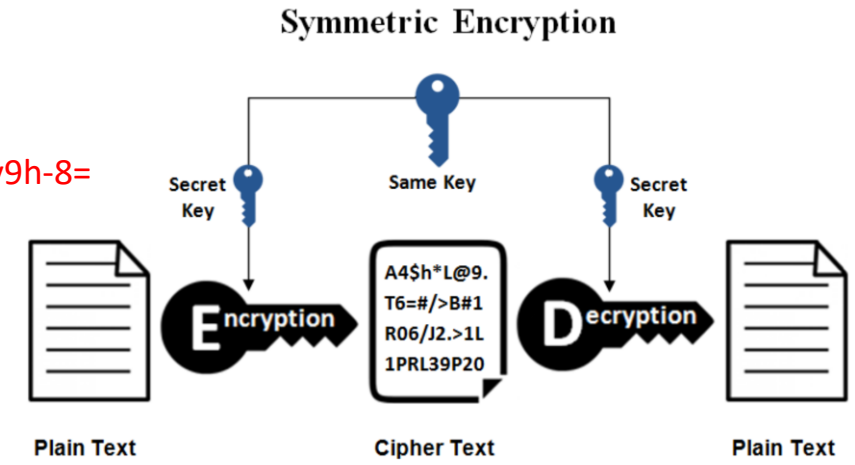
```
Requirement already satisfied: cryptography in c:\users\guais\appdata\local\packages\pythonsoftwarefoundation.python.3.1
0_qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (39.0.2)
Requirement already satisfied: cffi>=1.12 in c:\users\guais\appdata\local\packages\pythonsoftwarefoundation.python.3.10_
qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from cryptography) (1.15.1)
Requirement already satisfied: pycparser in c:\users\guais\appdata\local\packages\pythonsoftwarefoundation.python.3.10_q
bz5n2kfra8p0\localcache\local-packages\python310\site-packages (from cffi>=1.12->cryptography) (2.21)

[notice] A new release of pip available: 22.3.1 -> 23.0.1
[notice] To update, run: C:\Users\guais\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.10_qbz5n2k
fra8p0\python.exe -m pip install --upgrade pip
```

Performing symmetric encryption using Python

What is “symmetric encryption”?

- Using the **same secret key** for both encryption and decryption of data

Secret Key: S_aIcd2d2e4-TMZ9rMLGLOn8BM1Tih3sEiww1pv9h-8=



Plain Text: Hello World

Cipher Text: gAAAAABkDzr0bpuf3BCOCjUhtTmkemS23-nJVhBxaeEyJZ33vLM6wch-nmB50yDpMhlBw4gvlSpSyTbR_YJy0CAPrWN9mS1c7w==

https://www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences

UNIVERSITY OF WOLLONGONG AUSTRALIA

Performing symmetric encryption using Python

Create a new file, namely “symenc.py”

Import library

Use “Tab” instead of spacing!

```
from cryptography.fernet import Fernet

def create_key():
    key = Fernet.generate_key()
    return key

secretkey = create_key()
print("Encryption Key:", secretkey)
```

“def” means defining a function

Result:

```
C:\Users\guais\Desktop\New folder>python symenc.py
Encryption Key: b'HdiD9L_o9nm_tfponagFweq1psW8AO1lk0YdQmtoPvk='
```

You will see the **random key starting with b’xxxxxx’.** This means the key is **represented as “byte”**

Performing symmetric encryption using Python

```
String format: hello world
Byte format: b'hello world'
```

Why is it important?

A function may require “Input”

The input must conform to a defined format.

For example: `Fernet(key).encrypt(ptext)`

- Fernet(X).encrypt(Y) requires two inputs: X (key) and Y (ptext)
- Based on what Fernet defined: X and Y must be in **Byte format**

UNIVERSITY OF WOLLONGONG AUSTRALIA

Performing symmetric encryption using Python

```
from cryptography.fernet import Fernet

def create_key(): # def means “function”
    key = Fernet.generate_key()
    return key

def encrypt(key, ptext):
    ctext = Fernet(key).encrypt(ptext)
    return ctext

secretkey = create_key()
print("Encryption Key:", secretkey)
plaintext = input("Enter your message to encrypt: ")

ciphertext = encrypt(secretkey, plaintext)
print("Ciphertext: ", ciphertext)
```

```
Encryption Key: b'x_q__1or7DUAZEUHiNvsOPOtrnZ4bF6LVkdMVTvBD0I='
Enter your message to encrypt: Hello World
Traceback (most recent call last):
  File "C:\Users\guais\Desktop\New folder\symenc.py", line 17, in <module>
    ciphertext = encrypt(secretkey, plaintext)
  File "C:\Users\guais\Desktop\New folder\symenc.py", line 10, in encrypt
    ctext = Fernet(key).encrypt(ptext)
  File "C:\Users\guais\AppData\Local\Packages\PythonSoftwareFoundation.Pyt
es\Python310\site-packages\cryptography\fernet.py", line 51, in encrypt
    return self.encrypt_at_time(data, int(time.time()))
  File "C:\Users\guais\AppData\Local\Packages\PythonSoftwareFoundation.Pyt
es\Python310\site-packages\cryptography\fernet.py", line 55, in encrypt_at
    return self._encrypt_from_parts(data, current_time, iv)
  File "C:\Users\guais\AppData\Local\Packages\PythonSoftwareFoundation.Pyt
es\Python310\site-packages\cryptography\fernet.py", line 60, in _encrypt_f
    utils._check_bytes("data", data)
  File "C:\Users\guais\AppData\Local\Packages\PythonSoftwareFoundation.Pyt
es\Python310\site-packages\cryptography\utils.py", line 30, in _check_byte
    raise TypeError("{} must be bytes".format(name))
TypeError: data must be bytes
```

The code encounters error message!

How to convert plaintext to byte?

Performing symmetric encryption using Python

Instead of running

```
ciphertext = encrypt(secretkey, plaintext)
```

We modify it as

```
ciphertext = encrypt(secretkey, plaintext.encode())
```

The additional command: xxxx.encode() changes the format of “plaintext” from String to Byte

String_variable.encode() → Byte_variable

Performing symmetric encryption using Python

Your program should now be able to **encrypt** the entered plaintext!

```
C:\Users\guais\Desktop\New folder>python symenc.py
Encryption Key: b'c_soq_e18Y0TLne2l2TvpEfBUsokyTgScf6qO4_Jkug='
Enter your message to encrypt: hello world
Ciphertext:  b'gAAAAABkD0GvzVH6AW0VRJyR09BHQc7KeGWw3xILlJOq_X71J2MMyEG
79_SopKuHFcZWFG83FgRrAKko9j4S1hwTj9CaGGlLLQ=='
```

Now, given a ciphertext, how can we **decrypt and obtain** the **original plaintext**?

Performing symmetric encryption using Python

To decrypt the ciphertext, we need the **same secret (encryption)** key!

```
def decrypt(key, ctext):
    ptext = Fernet(key).decrypt(ctext)
    return ptext
```

We then call the decryption function – input the same secret key

```
print( "Plaintext (decrypted): ", decrypt(secretkey, ciphertext))
```

By calling Fernet(X).decrypt(Y), we manage to **recover plaintext** if the used key is correct/same.

Result:

```
Enter your message to encrypt: hello world
Ciphertext:  b'gAAAAABkD0RU3uya0TDwXpIF5sqNwyo01sYKaXQKqTmnCnJ
OSZRAr9u_BYLtr1gktfQG1sl14WWa7yBwnyNyy7tMJ-CZFDsL0Q=='
Plaintext (decrypted):  b'hello world'
```

UNIVERSITY OF WOLLONGONG AUSTRALIA

Performing symmetric encryption using Python

Suppose a wrong key is used

```
fakekey = create_key()
print("Plaintext (decrypted, fake key): ", decrypt(fakekey, ciphertext))
```

You will receive an error message, which indicates the function is unable to decrypt the ciphertext with the given “wrong key”

Performing symmetric encryption using Python

Now, the decrypted message is in Byte format

```
Plaintext (decrypted):  b'hello world'
```

How should we change it to String format?

```
print("Plaintext (decrypted): ", decrypt(deckey, ciphertext))

print("Plaintext (decrypted): "+ decrypt(deckey, ciphertext).decode())
```

Results:

```
Plaintext (decrypted):  b'hello'
Plaintext (decrypted): hello
```

Byte_variable.decode() → String_variable

Performing symmetric encryption using Python

The same for showing key or ciphertext in String

From `print("Ciphertext: ", ciphertext)` to `print("Ciphertext: ", ciphertext.decode())`

Or, we define a new variable:

```
string_ciphertext = ciphertext.decode()

print("Ciphertext: ", string_ciphertext)
```

Byte_variable.decode() → String_variable