

The cryptography Package

The **cryptography** package aims to be “cryptography for humans” much like the **requests** library is “HTTP for Humans”. The idea is that you will be able to create simple cryptographic recipes that are safe and easy-to-use. If you need to, you can drop down to low-level cryptographic primitives, which require you to know what you’re doing or you might end up creating something that’s not very secure.

If you are using Python 3.5, you can install it with pip, like so:

```
pip install cryptography
```



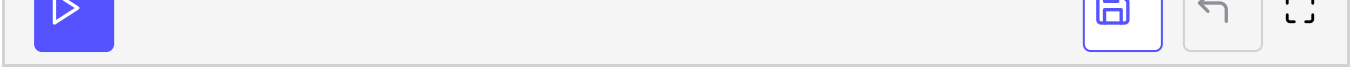
You will see that cryptography installs a few dependencies along with itself. Assuming that they all completed successfully, we can try encrypting some text. Let’s give the **Fernet** module a try. The Fernet module implements an easy-to-use authentication scheme that uses a symmetric encryption algorithm which guarantees that any message you encrypt with it cannot be manipulated or read without the key you define. The Fernet module also supports key rotation via **MultiFernet**. Let’s take a look at a simple example:

```
from cryptography.fernet import Fernet
cipher_key = Fernet.generate_key()
print (cipher_key)
#b'APM1JDVGt8WDGOWBgQv6EIhvx14vDYvUnVdg-Vjdt0o='

cipher = Fernet(cipher_key)
text = b'My super secret message'
encrypted_text = cipher.encrypt(text)
print (encrypted_text)
#(b'gAAAAABXOnV86aeUGADA6mTe9xEL92y_m0_TlC9vcqaF6NzHqRkKjEqh4d21PInEP3C9HuiUkS9f'
# b'6bdHsSlRiCNwbSkPuRd_62zfEv3eaZjJvLAm3omnya8=')

decrypted_text = cipher.decrypt(encrypted_text)
print (decrypted_text)
#b'My super secret message'
```





First off we need to import Fernet. Next we generate a key. We print out the key to see what it looks like. As you can see, it's a random byte string. If you want, you can try running the **generate_key** method a few times. The result will always be different. Next we create our Fernet cipher instance using our key.

Now we have a cipher we can use to encrypt and decrypt our message. The next step is to create a message worth encrypting and then encrypt it using the **encrypt** method. I went ahead and printed out the encrypted text so you can see that you can no longer read the text. To decrypt our super secret message, we just call **decrypt** on our cipher and pass it the encrypted text. The result is we get a plain text byte string of our message.

Wrapping Up

This chapter barely scratched the surface of what you can do with PyCryptodome and the cryptography packages. However it does give you a decent overview of what can be done with Python in regards to encrypting and decrypting strings and files. Be sure to read the documentation and start experimenting to see what else you can do!