Crypto Homework 3 Key

# Python3 on Ubuntu

The first solution will be on Ubuntu, the second on Windows with Python and IDLE.

# Encryption

From a BASH prompt, enter Python.  
python3

Execute the following in Python.

from Crypto.Cipher import AES

key = b'This is my key!7'

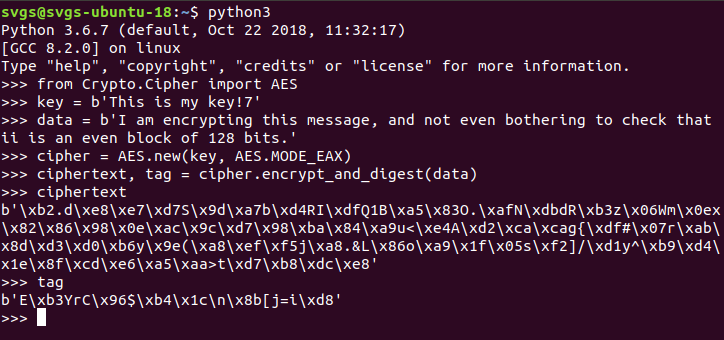
data = b'I am encrypting this message, and not even bothering to check that ii is an even block of 128 bits.'

cipher = AES.new(key, AES.MODE\_EAX)

ciphertext, tag = cipher.encrypt\_and\_digest(data)

ciphertext

tag



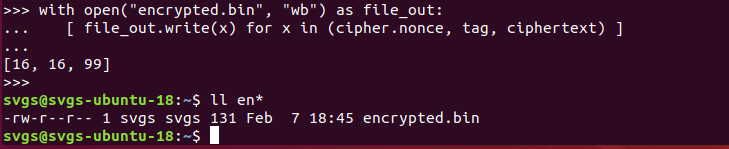
We need to send the nonce, tag, and ciphertext to our recipient so we will save them in one file, ecrypted.bin. I prefer to use the “with open” Python syntax, with open since it automatically closes the file when we are done.

with open("encrypted.bin", "wb") as file\_out:

[ file\_out.write(x) for x in (cipher.nonce, tag, ciphertext) ]

exit()

Now, back at BASH, let’s make sure we created the file  
svgs@svgs-ubuntu-18:~$ ll en\*  
-rw-r--r-- 1 svgs svgs 131 Feb 7 18:45 encrypted.bin



Now we move to the other computer. The file encrypted.bin is emailed to the other student, given on a flash drive, whatever.

# Decryption

Enter Python on second computer.  
python3

Import the AES module and read the encrypted message from encrypted.bin. Note that the code below saves the first 16 bytes from encrypted.bin in nonce, the second 16 bytes in tag, and the rest in ciphertext. (Hopefully that matches the way we created encrypted.bin)  
from Crypto.Cipher import AES  
with open("encrypted.bin", "rb") as file\_in:  
 nonce, tag, ciphertext = [ file\_in.read(x) for x in (16, 16, -1) ]

We received the key from our partner in a secure method, this time as a written note from our partner.

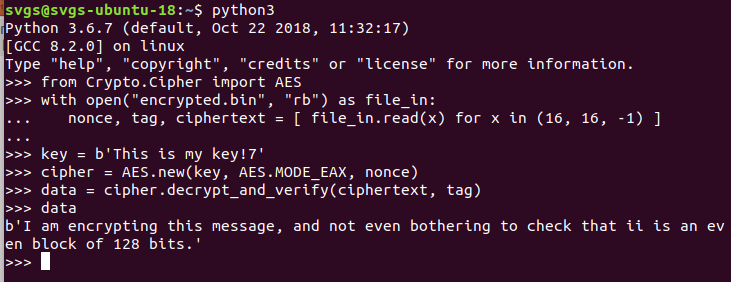
key = b'This is my key!7'

cipher = AES.new(key, AES.MODE\_EAX, nonce)

data = cipher.decrypt\_and\_verify(ciphertext, tag)

When we check the value of the variable data, we see that we have recovered our message.

b'I am encrypting this message, and not even bothering to check that ii is an even block of 128 bits.'



# Python3 on Windows with IDLE

Execute the following in Python.

from Crypto.Cipher import AES

key = b'This is my key!7'

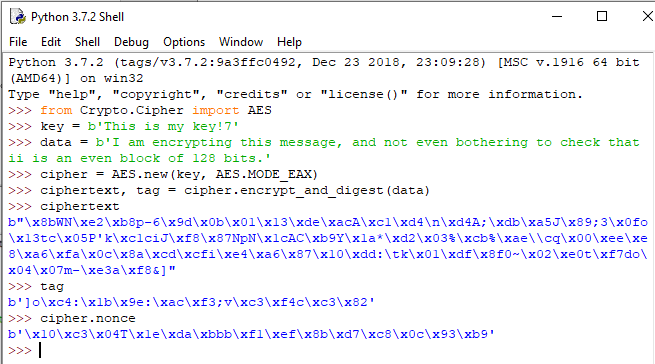
data = b'I am encrypting this message, and not even bothering to check that ii is an even block of 128 bits.'

cipher = AES.new(key, AES.MODE\_EAX)

ciphertext, tag = cipher.encrypt\_and\_digest(data)

ciphertext

tag

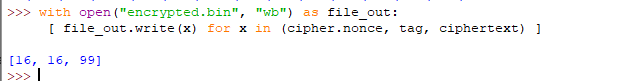


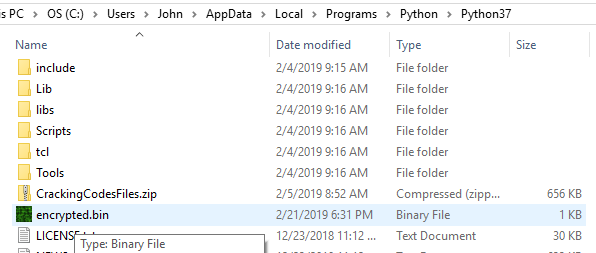
Note that the values for the ciphertext, tag, and nonce are all different from the previous example even though the key and plaintext (data) are unchanged. The nonce introduces the random component that does this.

We need to send the nonce, tag, and ciphertext to our recipient so we will save them in one file, ecrypted.bin. I prefer to use the “with open” Python syntax, with open since it automatically closes the file when we are done.

with open("encrypted.bin", "wb") as file\_out:

[ file\_out.write(x) for x in (cipher.nonce, tag, ciphertext) ]



The file encrypted.bin was created and is visible in file explorer.  


# Decryption

Enter Python on second computer.

Import the AES module and read the encrypted message from encrypted.bin. Note that the code below saves the first 16 bytes from encrypted.bin in nonce, the second 16 bytes in tag, and the rest in ciphertext. (Hopefully that matches the way we created encrypted.bin)  
from Crypto.Cipher import AES  
with open("encrypted.bin", "rb") as file\_in:  
 nonce, tag, ciphertext = [ file\_in.read(x) for x in (16, 16, -1) ]

We received the key from our partner in a secure method, this time as a written note from our partner.

key = b'This is my key!7'

cipher = AES.new(key, AES.MODE\_EAX, nonce)

data = cipher.decrypt\_and\_verify(ciphertext, tag)

When we check the value of the variable data, we see that we have recovered our message.

b'I am encrypting this message, and not even bothering to check that ii is an even block of 128 bits.'

