Laboratory_01: Basic Network Communication and AES in Python

In this lab you will write a very simple client and server using AES.

Your task

Your job is to write a client and a server that connects to the server, retrieves ciphertext, decrypts it, and then displays the fortune to the user. When you connect to this server over TCP it sends you an encrypted fortune. Here is some important information:

- The server encrypts using AES in CBC mode.
- The key used for all encryption is 0x00112233445566778899aabbccddeeff.
- The initial vector used for the encryption is sent first (as 16 raw bytes).
- After the IV, the CT is sent (as raw bytes).
- The server closes the connection when it is finished sending bytes.
- Consider verify the traffic using wireshark or tshark.

References

1. Consider at the socket api for Python. (The link is is for Python 3.6, but you can change the version in the upper left of the page.)

Hints

1. Socket programming is a big topic, so here is some sample code to open a new connection to a given port, read some bytes, and print them:
2. import socket
3. sock = socket.create_connection(('rainmaker.wunderground.com', 23))
4. data = sock.recv(8192)
5. print(data)
6. sock.close()

7. Note that recv(num_bytes) will receive *at most* num_bytes but may return less. If you want to make sure you get a specific number of bytes, you may need to build a loop to do so. (Although frequently you are just lucky and get the number of bytes you expected...)
8. If you call recv() and there is no data available to be read, then it will block and your program will wait until there is data to be read, the connection is closed, or a timeout occurs.
9. When passing a key or IV to the pycrypto library, you should be passing raw bytes. For example, a 128-bit key (16 bytes) should be a 16-byte buffer containing the bytes, *not* an integer.