CPSC 526 Assignment 3: Readme

Albert Luu (10129499) , Ana Paulina Telan (10124075)

# Compiling and running

To run server:

python3 server.py [port] [key]

|  |  |
| --- | --- |
| port | Port number where server will be listening |
| key | Password used for encryption if client decides to use; if not provided, server will generate a random 32-character password |

Note: server is assumed to be running in localhost.

To run client:

python3 client.py [cmd] [filename] [ip]:[port] [cipher] [key]

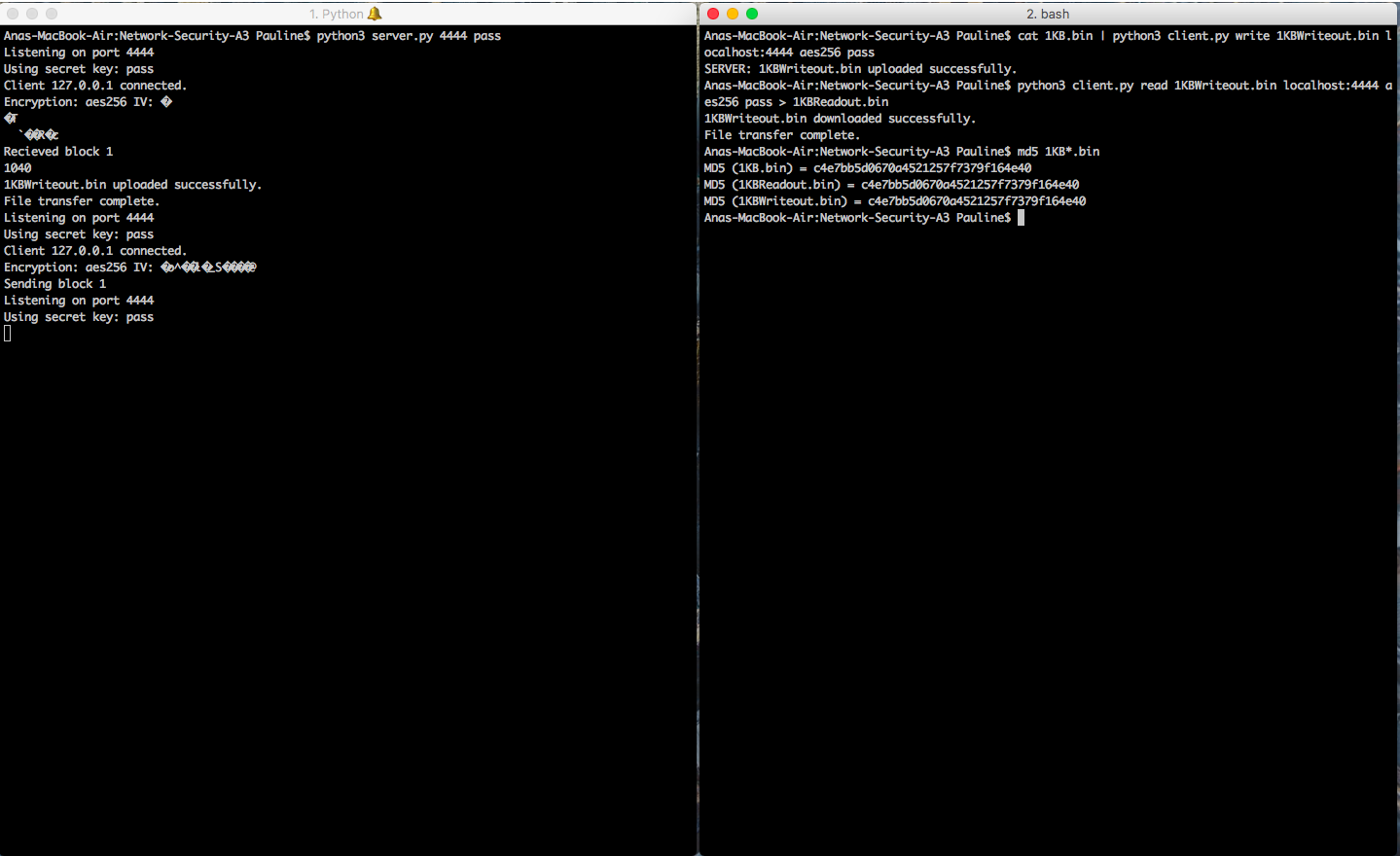
|  |  |
| --- | --- |
| cmd | Determines if client will be uploading/downloading data to/from server either be read/write |
| filename | If cmd is read, server will send contents of filename, else the server will upload contents of stdout to filename |
| ip | IP address of server |
| port | The port the server is listening on |
| cipher | specifies what encryption algorithm is used for communication, can be none, aes128 or aes256 |
| key | Key to be used for encryption. Not applicable when cipher is none. |

# Communication Protocol

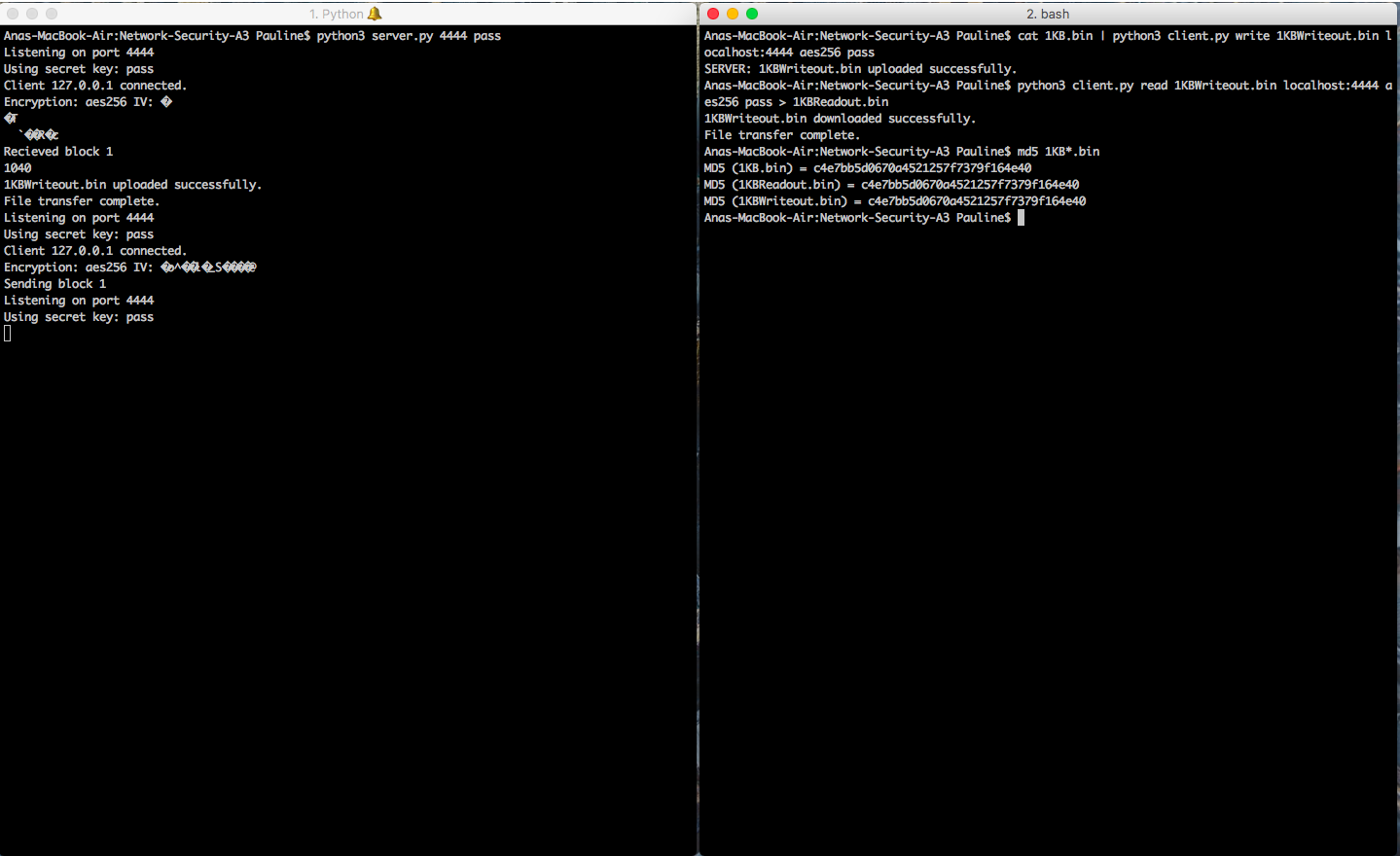
1. Client connects to server
2. Client sends [Cipher, IV] to server in the clear
3. If encryption requested, password authentication:
   1. Client sends encrypted IV to server and vice versa
   2. Both parties compare received encrypted IV to locally encrypted IV
   3. If the same, password checks out. Else, connection is terminated.
4. Client sends [cmd, filename] to server. If cipher is specified, this is encrypted.
5. If cmd=write
   1. client sends data size and then data in 4MB chunks to server
   2. server receives and processes data
   3. once client has sent all data, server sends response to client and client displays server response to console
6. If cmd=read
   1. server sends data size and then data in 4MB chunks to client
   2. client receives and processes data
7. Connection is closed

# Testing AES256 on 1KB bin file using MD5 hash

Server:



Client:



# Timing tests

*Note: a full list of raw values can be found in timedata.txt*

Write

|  |  |  |  |
| --- | --- | --- | --- |
| **File size** | **None** | **AES128** | **AES256** |
| 1KB | 0.684s | 1.100s | 1.108s |
| 1MB | 1.962s | 1.696s | 1.505s |
| 1GB | 13.825s | 56.565s | 58.740s |

Read

|  |  |  |  |
| --- | --- | --- | --- |
| **File size** | **None** | **AES128** | **AES256** |
| 1KB | 0.681s | 0.957s | 0.945s |
| 1MB | 1.116s | 1.449s | 1.582s |
| 1GB | 13.006s | 54.350s | 57.884s |

Prior to the timing of these tests, we hypothesized that on average, choosing to encrypt data would result in a longer run time than not encrypting as there are more operations done to the data before and after it is sent to the destination. Furthermore, we expected that there wouldn’t be a significant difference in terms of time between encrypting with AES128 vs. AES256. Hence, as depicted in our graphs, our results did not come as a big surprise for us although there are some outliers worthy to be discussed.

Comparing the two AES algorithms with differing keys (128 vs. 256), it can be seen that the difference is quite negligible. We believe that this lack of difference depending on the key size is caused by the similarity in the amount of operations between the two. The key size does not dictate a significant change in the communication protocol or the algorithm as the only difference is how the key will be parsed, depending on the password given. In more specific terms, padding/truncating takes constant time hence, there is no notable contrast in time.

For the most part, no encryption proves a significant difference in time as our results dictate that it is way shorter than the other two. Note that when no encryption is used on smaller files, the difference does not quite pass 50% compared to the other ciphers. However, when it is used on bigger files, the difference is greater - regardless whether write/read is used. It is also worth noting that our results for writing on a 1MB file produced a descending trend which is different from the other tests, an anomaly we can’t really explain to be frank.

In conclusion, based on our results, we see that not encrypting traffic only produces a significant difference when the file to be sent is big (ie. 1GB). Furthermore, using AES128 vs. AES256 does not make a time difference worthy of attention hence, using either or should run around the same time length. Therefore, there are significant differences only between the run times of not encrypting versus encrypting but these differences are majorly dictated by the size of the file about to be sent.