

Assignment 4

CPSC 526 Fall 2017

November 12, 2017

Mason Lieu	Shane Sims
ID: 10110089	ID: 00300601
Tutorial 04	Tutorial 04
mliu@ucalgary.ca	shane.sims@ucalgary.ca

How to run Server and Client

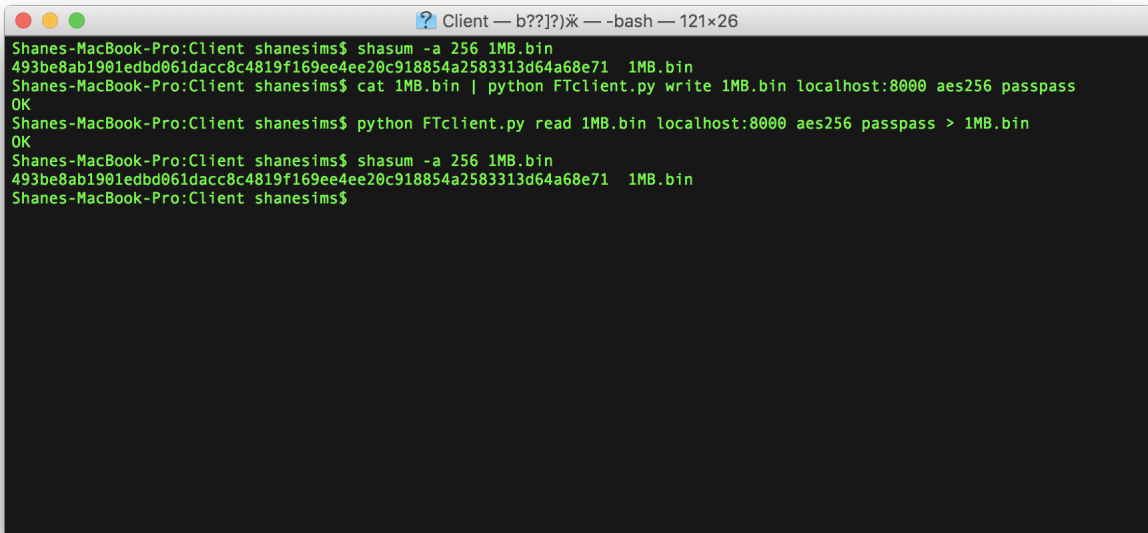
Running the server

```
1 python FTserver.py <port number> <key>
```

Running the client

```
1 python FTclient.py <command> <file name> <host>:<port> <cipher> <key>
```

AES256 upload/download/checksum test

A terminal window titled "Client — b??[?]* — -bash — 121x26" showing a series of commands and their outputs. The commands are: 1. shasum -a 256 1MB.bin, which outputs a long hexadecimal string. 2. cat 1MB.bin | python FTclient.py write 1MB.bin localhost:8000 aes256 passpass, which outputs "OK". 3. python FTclient.py read 1MB.bin localhost:8000 aes256 passpass > 1MB.bin, which outputs "OK". 4. shasum -a 256 1MB.bin, which outputs the same long hexadecimal string as the first command. The terminal background is dark with light green text.

```
Shanes-MacBook-Pro:Client shanesims$ shasum -a 256 1MB.bin
493be8ab1901edbd061dacc8c4819f169ee4ee20c918854a2583313d64a68e71 1MB.bin
Shanes-MacBook-Pro:Client shanesims$ cat 1MB.bin | python FTclient.py write 1MB.bin localhost:8000 aes256 passpass
OK
Shanes-MacBook-Pro:Client shanesims$ python FTclient.py read 1MB.bin localhost:8000 aes256 passpass > 1MB.bin
OK
Shanes-MacBook-Pro:Client shanesims$ shasum -a 256 1MB.bin
493be8ab1901edbd061dacc8c4819f169ee4ee20c918854a2583313d64a68e71 1MB.bin
Shanes-MacBook-Pro:Client shanesims$
```

Figure 1: Test for correctness

Communication Protocol

write command

	Client	Server
1.	send cipher,nonce1	
2.		send challenge nonce2
3.	send sha256(key—nonce2)	
4.		send ACK
5.	send write	
6.		send ACK
7.	send fileName	
8.		echo fileName
9.	encrypt 1024 byte blocks and send	
10.		receive and decrypt blocks
11.	send EOF	
12.		send status/result flag
13.	print status	

read command

	Client	Server
1.	send cipher,nonce1	
2.		send challenge nonce2
3.	send sha256(key—nonce2)	
4.		send ACK
5.	send read	
6.		send ACK
7.	send file name	
8.		send file size
9.		encrypt 1024 byte blocks and send
10.	receive and decrypt blocks	
11.	send status/result flag	
12.	print status	

Timing Tests

File Download Times by Cipher and File Size (median of 10 tests per file size)

FILE SIZE	NULL	AES128	AES 256
1KB	61ms	134ms	137ms
1MB	69ms	219ms	218ms
1GB	7s 389ms	1m 13s 61ms	1m 14s 81ms

File Upload Times by Cipher and File Size (median of 10 tests per file size)

FILE SIZE	NULL	AES128	AES 256
1KB	58ms	141ms	138ms
1MB	74ms	229ms	224ms
1GB	15s 309ms	1m 22s 82ms	1m 25s 777ms

Figure 2:

Summary of Results:

- Download times: From the figure above, it appears that there is a negligible difference in the download times the two AES ciphers used. It might have been expected that the time to download the 1GB file would have resulted in a pronounced difference between the two AES ciphers, but this was not the case. Somewhat surprisingly, within a given AES cipher there the download times for the 1MB file was about twice that of the 1KB file, despite being nearly ten times the size. As was to be expected, the download speed of when using the null cipher was much faster than using the either of the AES ciphers, in all cases of file size.
- Upload times: From the figure above, it can be seen that the results of the time tests for uploading are very similar to those for downloading, for all ciphers and file sizes. The only noteworthy comment that can be made is that on average the upload times were slightly slower than those for downloads.