Pujan Bhatta
10162769
T01
Mohammad Albaiti

How to run

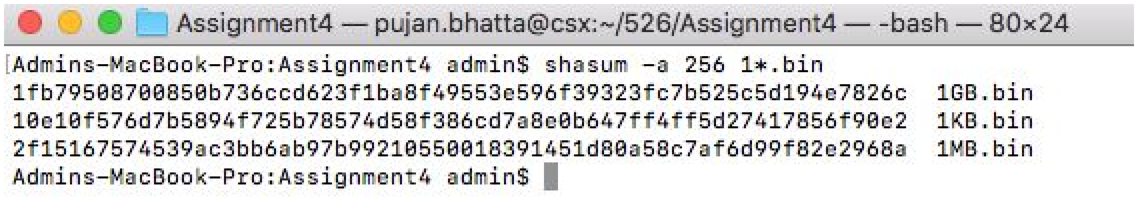
1)First run the server

    python3 Server.py port secret

2) Run the client

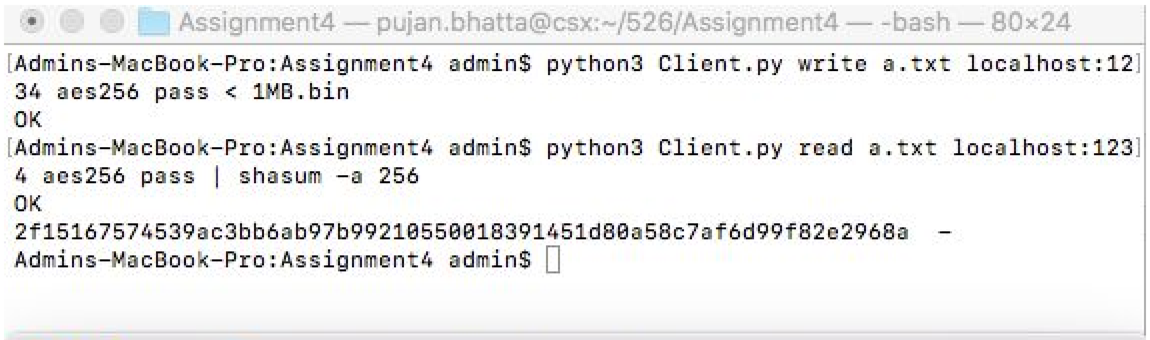
    Python3 Client.py command filename hostname:port cipher key
    ** Redirection using <> is supported

Test



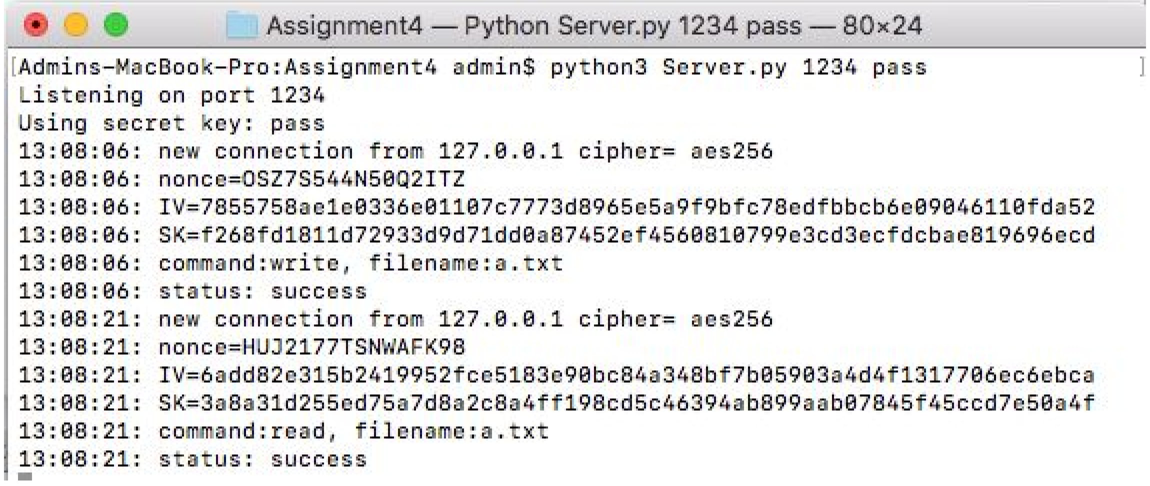
Assignment4 — pujan.bhatta@csx:~/526/Assignment4 — -bash — 80×24

```
Admins-MacBook-Pro:Assignment4 admin$ shasum -a 256 1*.bin
1fb79508700850b736ccd623f1ba8f49553e596f39323fc7b525c5d194e7826c  1GB.bin
10e10f576d7b5894f725b78574d58f386cd7a8e0b647ff4ff5d27417856f90e2  1KB.bin
2f15167574539ac3bb6ab97b99210550018391451d80a58c7af6d99f82e2968a  1MB.bin
Admins-MacBook-Pro:Assignment4 admin$
```



Assignment4 — pujan.bhatta@csx:~/526/Assignment4 — -bash — 80×24

```
Admins-MacBook-Pro:Assignment4 admin$ python3 Client.py write a.txt localhost:12
34 aes256 pass < 1MB.bin
OK
Admins-MacBook-Pro:Assignment4 admin$ python3 Client.py read a.txt localhost:123
4 aes256 pass | shasum -a 256
OK
2f15167574539ac3bb6ab97b99210550018391451d80a58c7af6d99f82e2968a  -
Admins-MacBook-Pro:Assignment4 admin$
```



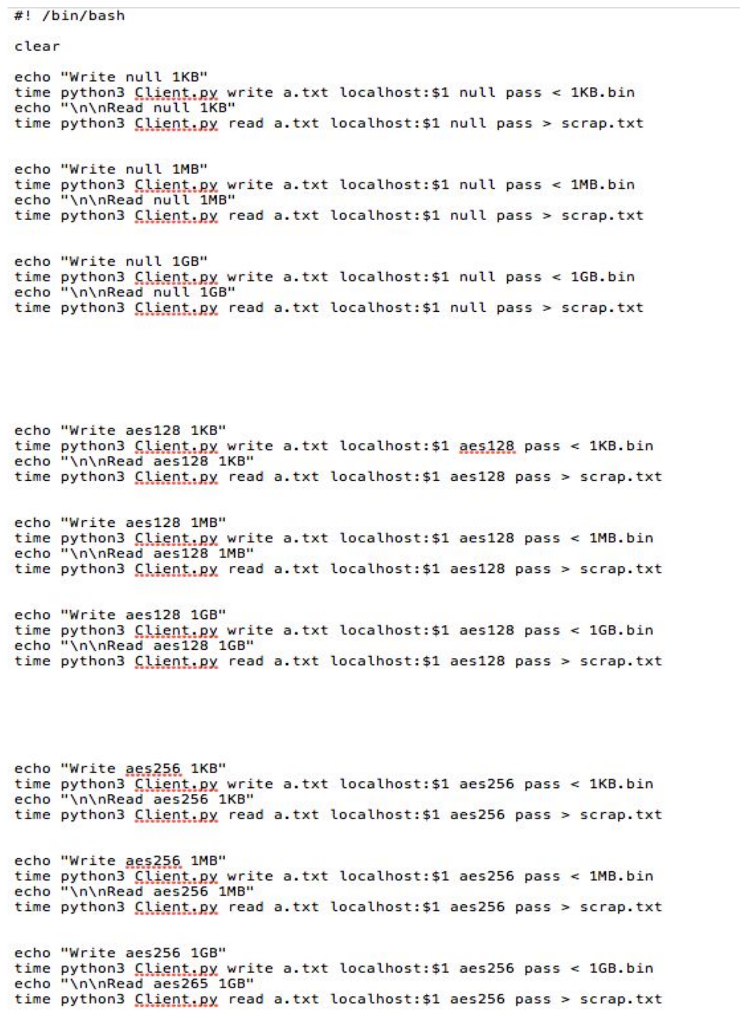
Assignment4 — Python Server.py 1234 pass — 80×24

```
Admins-MacBook-Pro:Assignment4 admin$ python3 Server.py 1234 pass
Listening on port 1234
Using secret key: pass
13:08:06: new connection from 127.0.0.1 cipher= aes256
13:08:06: nonce=OSZ7S544N50Q2ITZ
13:08:06: IV=7855758ae1e0336e01107c7773d8965e5a9f9bfc78edfbbcb6e09046110fda52
13:08:06: SK=f268fd1811d72933d9d71dd0a87452ef4560810799e3cd3ecfdcbae819696ecd
13:08:06: command:write, filename:a.txt
13:08:06: status: success
13:08:21: new connection from 127.0.0.1 cipher= aes256
13:08:21: nonce=HUJ2177TSNWAFK98
13:08:21: IV=6add82e315b2419952fce5183e90bc84a348bf7b05903a4d4f1317706ec6ebca
13:08:21: SK=3a8a31d255ed75a7d8a2c8a4ff198cd5c46394ab899aab07845f45ccd7e50a4f
13:08:21: command:read, filename:a.txt
13:08:21: status: success
```

Protocol

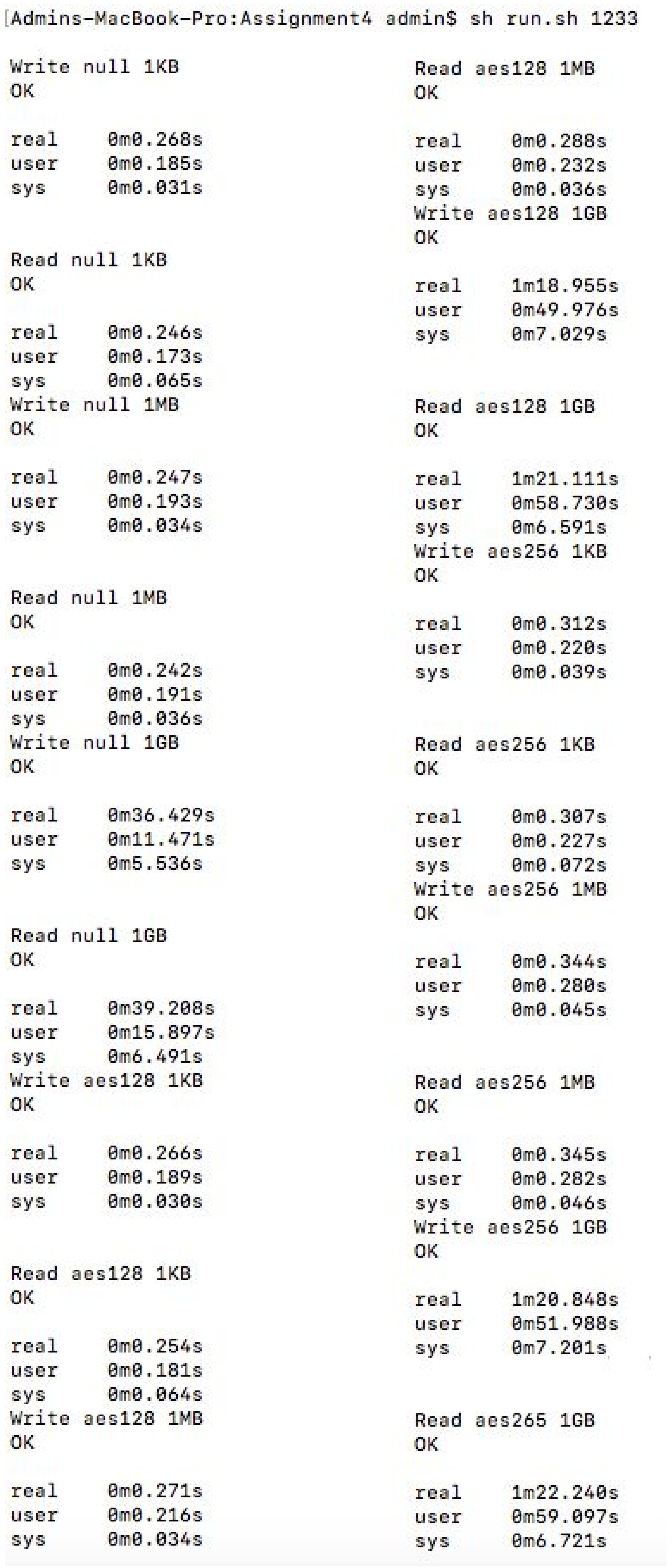
We used the python Cryptography and socket socket. First cipher and the nonce is sent to the server. The IV and SK are generated using the sha256 key|nonce| (“IV or “SK”)respectively by both client and server. For aes128 the key is computed by taking the md5 of the SK and for aes256 we take the first 32 bytes of the SK. Then there is a challenge where server send a random 8 byte data to client which is encrypted. If the key is correct and cipher type is null then sha256(key|data) is sent, key cannot be recovered even if someone get that data. If there is a cipher type then we try to decrypt, if that fails we throw an exception and then the client is disconnected, server and client will show error. When the authentication passes filename and command is sent. The data chunks are all sent and received in 40 bytes blocks which are encrypted if being sent or decrypted if its received. If the command is upload the server listens, then writes data sent by client to the file. If the command is download then it reads from the file and sends it to client, the client prints it to stdout. Errors are printed by both client and server if can't read or write. The client is then disconnected. Client displays error to stderr. If everything is finished without a problem then Server sends Success message to the client. When the client gets the messages it prints OK to stderr and disconnects.

Timing



```
#! /bin/bash

clear

echo "Write null 1KB"
time python3 Client.py write a.txt localhost:$1 null pass < 1KB.bin
echo "\n\nRead null 1KB"
time python3 Client.py read a.txt localhost:$1 null pass > scrap.txt

echo "Write null 1MB"
time python3 Client.py write a.txt localhost:$1 null pass < 1MB.bin
echo "\n\nRead null 1MB"
time python3 Client.py read a.txt localhost:$1 null pass > scrap.txt

echo "Write null 1GB"
time python3 Client.py write a.txt localhost:$1 null pass < 1GB.bin
echo "\n\nRead null 1GB"
time python3 Client.py read a.txt localhost:$1 null pass > scrap.txt




echo "Write aes128 1KB"
time python3 Client.py write a.txt localhost:$1 aes128 pass < 1KB.bin
echo "\n\nRead aes128 1KB"
time python3 Client.py read a.txt localhost:$1 aes128 pass > scrap.txt

echo "Write aes128 1MB"
time python3 Client.py write a.txt localhost:$1 aes128 pass < 1MB.bin
echo "\n\nRead aes128 1MB"
time python3 Client.py read a.txt localhost:$1 aes128 pass > scrap.txt

echo "Write aes128 1GB"
time python3 Client.py write a.txt localhost:$1 aes128 pass < 1GB.bin
echo "\n\nRead aes128 1GB"
time python3 Client.py read a.txt localhost:$1 aes128 pass > scrap.txt




echo "Write aes256 1KB"
time python3 Client.py write a.txt localhost:$1 aes256 pass < 1KB.bin
echo "\n\nRead aes256 1KB"
time python3 Client.py read a.txt localhost:$1 aes256 pass > scrap.txt

echo "Write aes256 1MB"
time python3 Client.py write a.txt localhost:$1 aes256 pass < 1MB.bin
echo "\n\nRead aes256 1MB"
time python3 Client.py read a.txt localhost:$1 aes256 pass > scrap.txt

echo "Write aes256 1GB"
time python3 Client.py write a.txt localhost:$1 aes256 pass < 1GB.bin
echo "\n\nRead aes265 1GB"
time python3 Client.py read a.txt localhost:$1 aes256 pass > scrap.txt
```

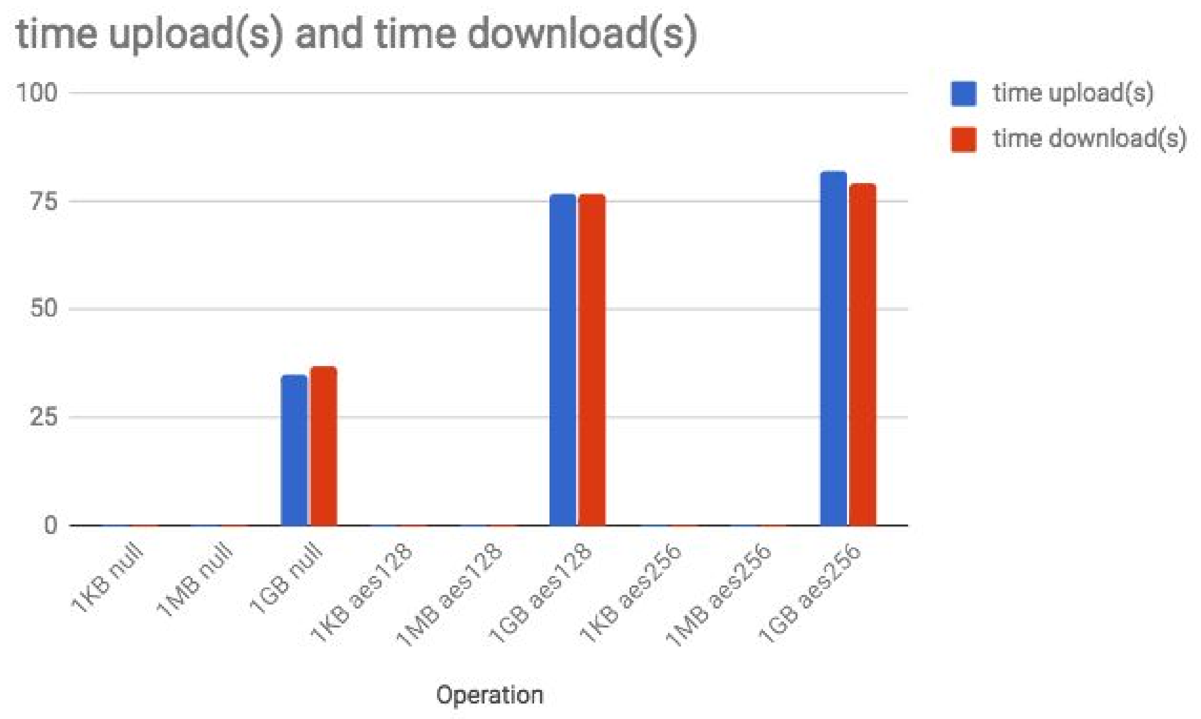
**Bash script for the test



```
Admins-MacBook-Pro:Assignment4 admin$ sh run.sh 1233
Write null 1KB
OK

real    0m0.268s
user    0m0.185s
sys     0m0.031s


Read null 1KB
OK

real    0m0.246s
user    0m0.173s
sys     0m0.065s
Write null 1MB
OK

real    0m0.247s
user    0m0.193s
sys     0m0.034s


Read null 1MB
OK

real    0m0.242s
user    0m0.191s
sys     0m0.036s
Write null 1GB
OK

real    0m36.429s
user    0m11.471s
sys     0m5.536s


Read null 1GB
OK

real    0m39.208s
user    0m15.897s
sys     0m6.491s
Write aes128 1KB
OK

real    0m0.266s
user    0m0.189s
sys     0m0.030s


Read aes128 1KB
OK

real    0m0.254s
user    0m0.181s
sys     0m0.064s
Write aes128 1MB
OK

real    0m0.271s
user    0m0.216s
sys     0m0.034s
```

```
Read aes128 1MB
OK

real    0m0.288s
user    0m0.232s
sys     0m0.036s
Write aes128 1GB
OK

real    1m18.955s
user    0m49.976s
sys     0m7.029s


Read aes128 1GB
OK

real    1m21.111s
user    0m58.730s
sys     0m6.591s
Write aes256 1KB
OK

real    0m0.312s
user    0m0.220s
sys     0m0.039s


Read aes256 1KB
OK

real    0m0.307s
user    0m0.227s
sys     0m0.072s
Write aes256 1MB
OK

real    0m0.344s
user    0m0.280s
sys     0m0.045s


Read aes256 1MB
OK

real    0m0.345s
user    0m0.282s
sys     0m0.046s
Write aes256 1GB
OK

real    1m20.848s
user    0m51.988s
sys     0m7.201s


Read aes265 1GB
OK

real    1m22.240s
user    0m59.097s
sys     0m6.721s
```

** Sample run, the image has been cropped and modified to look like this



From the graph and data we can conclude that the huge factors that affects the time is the file size, also that there is no difference in time between aes128 and aes256 but using no encryption also decreases the time by more than 50%. This is to be expected because encryption takes time as it encrypts the data block by block which takes a longer time for larger files. If there is no encryption then we can send the data right away. The results also tell me that half the time is used for encryption and the other half is used for file transfer. The evaluation is done in my personal macbook because trying to ssh into the school computer and running the file gives a permission denied error for the Cryptography libary.