Project: MP4
Course: ITM-513
Author: Brian T. Bailey

Project Description:

The objective of this project is to create a Python application that demonstrates network programming. In the project we need to transfer textual messages through network sockets from a client to a server. The client script needs to read the message content from a text file stored on the filesystem and store it in a Message object.

We also need to create encryption and decryption methods to encrypt the message before sending it to the server and decrypt it once the server receives it. The encryption should be some arithmetic modification of the message text using a key.

Once the server receives the encrypted message it needs to decrypt it and save it to a text file. The text file should include a receive timestamp, the message length and the decrypted text.

A driver script should be provided that will run the server, process the multiple clients, retrieve the server generated files, and show their content. The driver will retrieve the files from the server using the ftplib. It is ok to run both the client and server scripts on the same computer and simulate the network connections.

Installation, Compile and Run-Time Requirements:

This project was written in Python using version 2.7.1. The scripts were written in BBEdit version 10.1.2 on the Macintosh platform. The computer used was a 2.7 GHz dual-core Intel Core i7 13” MacBook Pro with 8GB of RAM running OS X Lion 10.7.4.

This application has been uploaded to the glenellyn server in a mp4 directory within my home directory. A driver script called mp4_driver.py is supplied to run the application. The driver script starts the server as a thread and runs the client scripts and ftp requirements.

Insights and Expected Results:

I decided to upload all the code to the glenellyn server and simulate the client-server communication there. I tried to run the client on my local machine and the server on glenellyn but ran into connection issues. I found others in the discussion board having issues too. Everything worked simulated on the localhost so I decided to go that route for now.

For the encryption and decryption methods I built a simple module. The methods I built for encryption in this module do a basic form of encryption. Although the encrypted text looks like random noise, these methods are not cryptographically very strong and should not be used in real code. An organization like the NSA would probably have no problem decrypting the cipher text. There are plenty of real crypto libraries, written by real cryptographers to use in production code.

For my crypto module I created a very basic symmetric block cipher. My cipher works in 32 byte blocks and requires a 256 bit (32 byte) key. I provided a basic password-based key derivation function (PBKDF). There is a very popular and secure function that is used in production code called PBKDF2 which is part of RSA's standards, but I decided to write my own very simplistic version. My function takes a string password as the argument and returns a string version of the 256 bit key. Inside the function there are a couple basic steps. First the provided password is hashed using sha256 which hashes the password into a 256 bit string. Next I enter a loop. In the loop I concatenate the hash with the original password and hash it again with sha256. The result of that hash is used at the top of the loop to add to the password and hash again. This continues for 20 rounds.

My encryption cipher is rather straightforward. First I make a byte array out of the key string. Next I calculate the amount of 32 byte blocks and what the remainder of bytes I have in the message using division and modulus. I then create a list of those 32 byte blocks. Depending on the remainder value, I pad the last block so it is 32 bytes or if the message is an even number of blocks I add a 32 byte padding block to the end of the list anyway. The padding value is the number of padding bytes added. This framework helps me strip the padding off later.

For the actual encryption I loop though the block list and for each 32 byte block I first create a byte array from the string. Then I reverse the order of the bytes. Once I have that, I use a loop to XOR the key and block arrays and concatenate everything to a string buffer. My final step is to base64 encode that buffer. That is the cipher text I return from the function.

The decryption method is basically similar but in a reverse order. First I break up the blocks. Then I decrypt with the XOR then the byte swap. Finally I remove the padding from the string. For that, all I have to do is read the last byte and strip off that many characters from the end of the string.

For the client script I wrote a standalone script that takes two arguments. The first is the path and filename of the message text file and the second is a string password for the encryption. The script connects to the server and transfers the key then the encrypted message together.

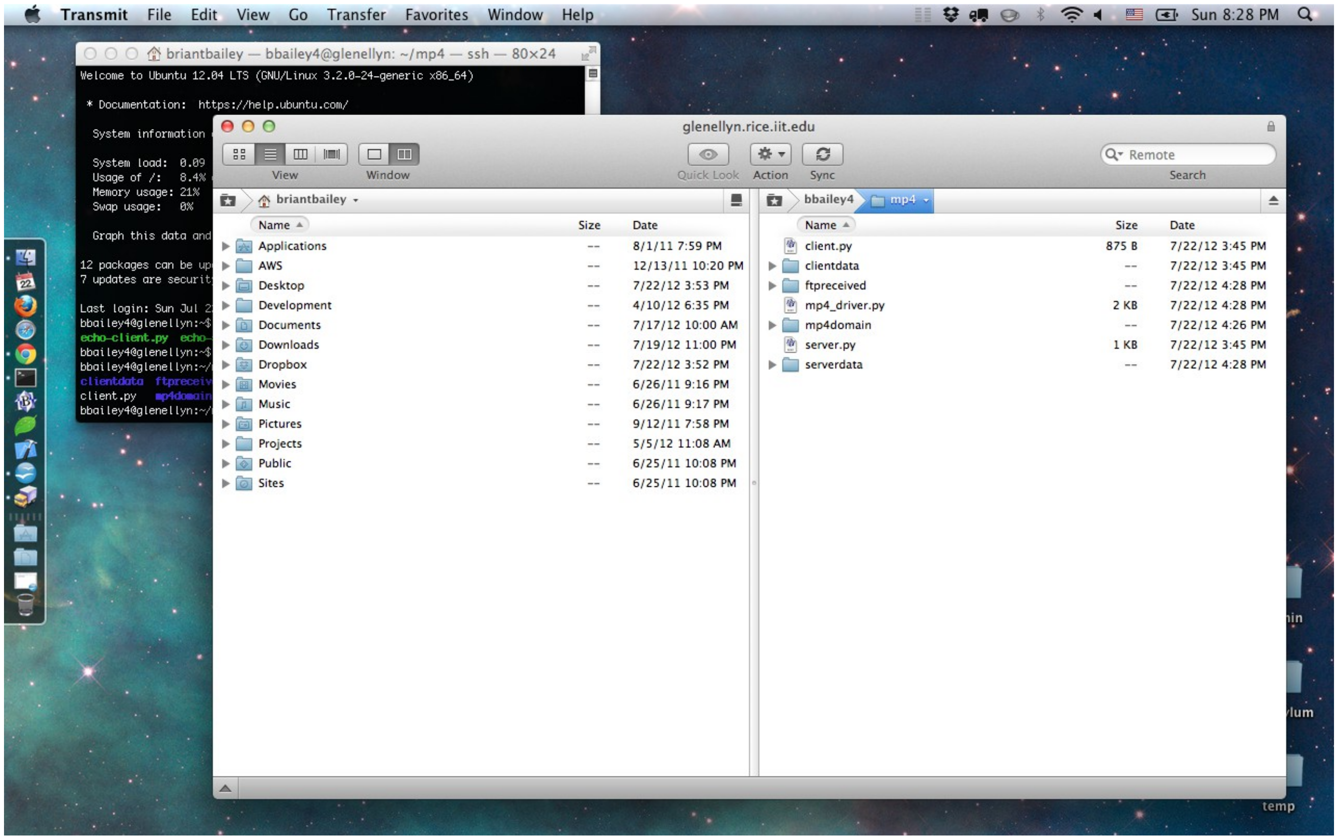
The server code receives the encrypted message from the client and takes the first 32 bytes of the message to use as the key to decrypt the remainder of the message. It then proceeds to write the log file for that message transfer with the required information. I decided to write the encrypted data to the end of the file too just so you could see the differences in the plaintext and encrypted text.

Since I was running both client and server on the same machine, I decided to build a driver that took care of running the entire simulation. The driver file first starts up the server code as a separate thread. Although I did the server in a thread, I also provided the code as a standalone server file that can be run separately in case they were not on the same system or you wanted it to run as a separate file.

After the driver starts the server, it uses the subprocess module to call the client script 6 different times with 6 different input files. Using the subprocess module allowed me to simulate running the client script from the command line with the two arguments. After all the client files run, the driver then uses the ftplib to download all the log files the server produced and stores them in a ftpreceived directory. The final step in the driver loops through all those files and prints their contents to the terminal console. After that the driver is done running and since the server thread is a daemon thread it terminates also.

Screenshots Demonstrating Application:

This screenshot shows the mp4 project directory on the glenellyn server.



```
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.2.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information

  System load:  0.09
  Usage of /:   8.4%
  Memory usage: 21%
  Swap usage:   0%

  Graph this data and

12 packages can be up
7 updates are securit

Last login: Sun Jul 2
bbailey4@glenellyn:~$
echo-client.py  echo-
bbailey4@glenellyn:~$
bbailey4@glenellyn:~/
clientdata  ftpreceiv
client.py   mp4domain
bbailey4@glenellyn:~/
```

glenellyn.rice.iit.edu

briantbailey

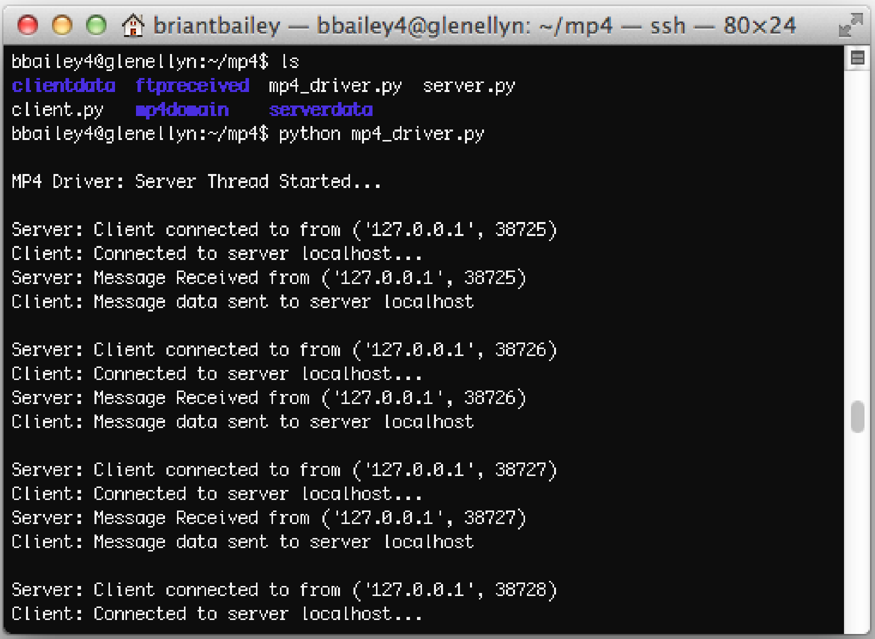
Name	Size	Date
Applications	--	8/1/11 7:59 PM
AWS	--	12/13/11 10:20 PM
Desktop	--	7/22/12 3:53 PM
Development	--	4/10/12 6:35 PM
Documents	--	7/17/12 10:00 AM
Downloads	--	7/19/12 11:00 PM
Dropbox	--	7/22/12 3:52 PM
Movies	--	6/26/11 9:16 PM
Music	--	6/26/11 9:17 PM
Pictures	--	9/12/11 7:58 PM
Projects	--	5/5/12 11:08 AM
Public	--	6/25/11 10:08 PM
Sites	--	6/25/11 10:08 PM

bbailey4 mp4

Name	Size	Date
client.py	875 B	7/22/12 3:45 PM
clientdata	--	7/22/12 3:45 PM
ftpreceived	--	7/22/12 4:28 PM
mp4_driver.py	2 KB	7/22/12 4:28 PM
mp4domain	--	7/22/12 4:26 PM
server.py	1 KB	7/22/12 3:45 PM
serverdata	--	7/22/12 4:28 PM

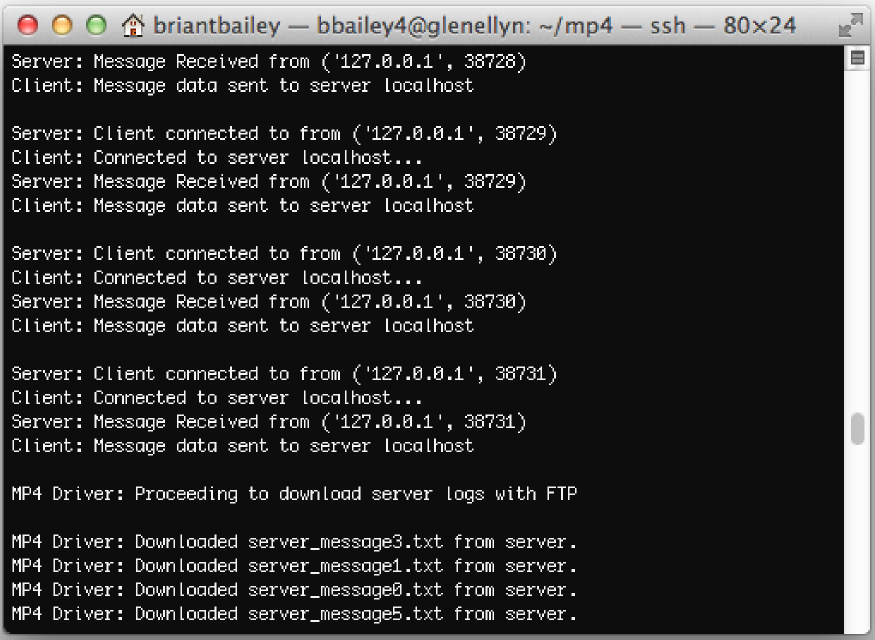
Transmit File Edit View Go Transfer Favorites Window Help
Sun 8:28 PM
briantbailey — bbailey4@glenellyn: ~/mp4 — ssh — 80×24
View
Window
Quick Look
Action
Sync
Remote
Search
temp

Screenshot that shows the mp4_driver.py file being executed and starting from the terminal window on the glenellyn server.



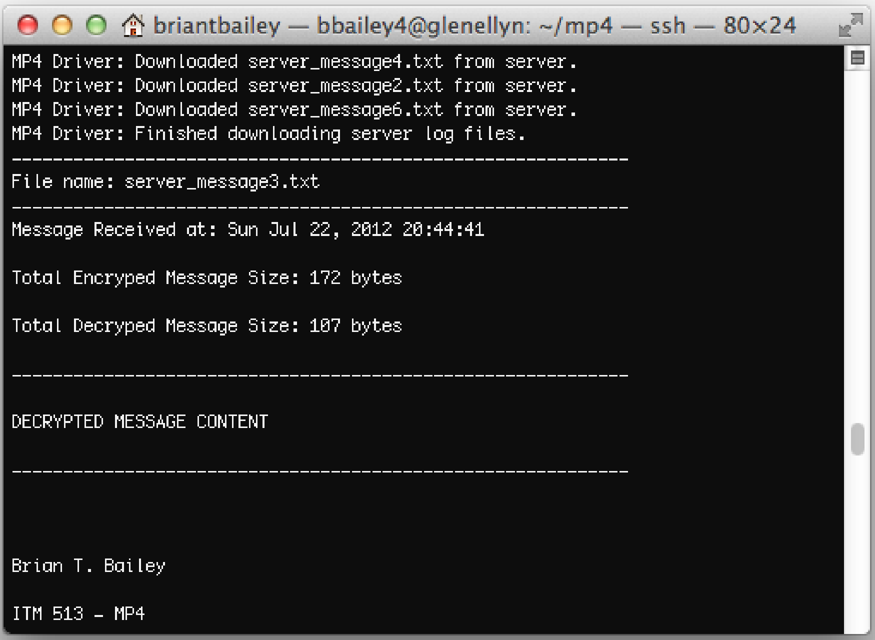
```
bbailey4@glenellyn:~/mp4$ ls
clientdata  ftpreceived  mp4_driver.py  server.py
client.py   mp4domain    serverdata
bbailey4@glenellyn:~/mp4$ python mp4_driver.py

MP4 Driver: Server Thread Started...

Server: Client connected to from ('127.0.0.1', 38725)
Client: Connected to server localhost...
Server: Message Received from ('127.0.0.1', 38725)
Client: Message data sent to server localhost

Server: Client connected to from ('127.0.0.1', 38726)
Client: Connected to server localhost...
Server: Message Received from ('127.0.0.1', 38726)
Client: Message data sent to server localhost

Server: Client connected to from ('127.0.0.1', 38727)
Client: Connected to server localhost...
Server: Message Received from ('127.0.0.1', 38727)
Client: Message data sent to server localhost

Server: Client connected to from ('127.0.0.1', 38728)
Client: Connected to server localhost...
```

Screenshot that shows the application driver running with client and server status updates. At the end of this window the ftp function starts transferring the server log files.



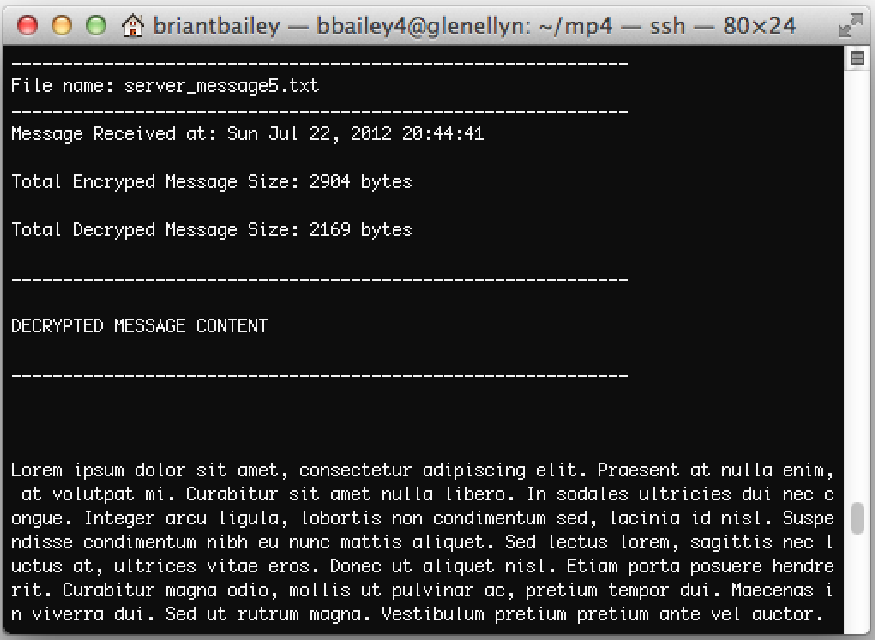
```
Server: Message Received from ('127.0.0.1', 38728)
Client: Message data sent to server localhost

Server: Client connected to from ('127.0.0.1', 38729)
Client: Connected to server localhost...
Server: Message Received from ('127.0.0.1', 38729)
Client: Message data sent to server localhost

Server: Client connected to from ('127.0.0.1', 38730)
Client: Connected to server localhost...
Server: Message Received from ('127.0.0.1', 38730)
Client: Message data sent to server localhost

Server: Client connected to from ('127.0.0.1', 38731)
Client: Connected to server localhost...
Server: Message Received from ('127.0.0.1', 38731)
Client: Message data sent to server localhost

MP4 Driver: Proceeding to download server logs with FTP

MP4 Driver: Downloaded server_message3.txt from server.
MP4 Driver: Downloaded server_message1.txt from server.
MP4 Driver: Downloaded server_message0.txt from server.
MP4 Driver: Downloaded server_message5.txt from server.
```

Screenshot that shows the ftp transfer completing and the driver starting to display the contents of the server log files.


briantbailey — bbailey4@glenellyn: ~/mp4 — ssh — 80×24

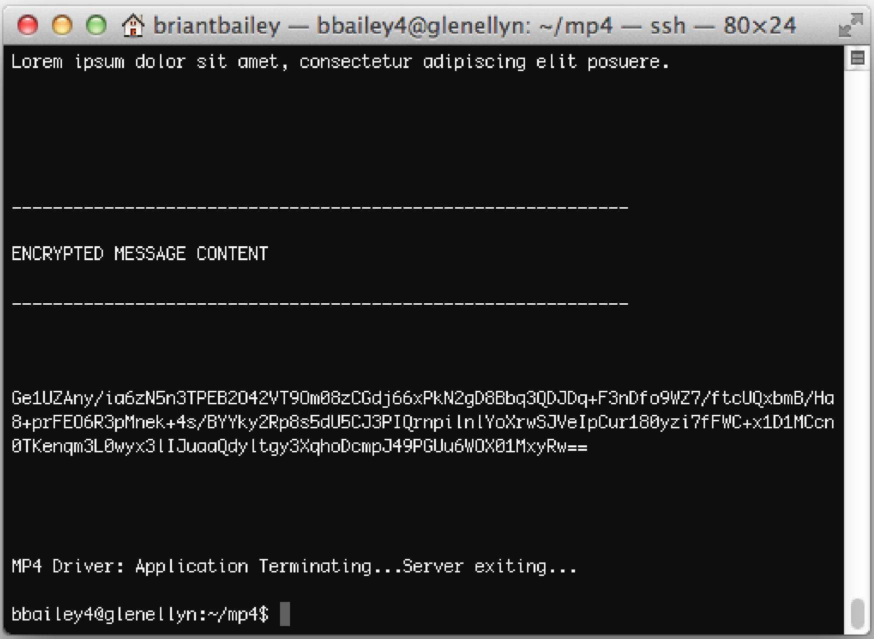
```
MP4 Driver: Downloaded server_message4.txt from server.
MP4 Driver: Downloaded server_message2.txt from server.
MP4 Driver: Downloaded server_message6.txt from server.
MP4 Driver: Finished downloading server log files.
------------------------------------------------------------
File name: server_message3.txt
------------------------------------------------------------
Message Received at: Sun Jul 22, 2012 20:44:41

Total Encryped Message Size: 172 bytes

Total Decryped Message Size: 107 bytes

------------------------------------------------------------

DECRYPTED MESSAGE CONTENT

------------------------------------------------------------



Brian T. Bailey

ITM 513 - MP4
```

This screenshot shows the application continuing to display the server log files.



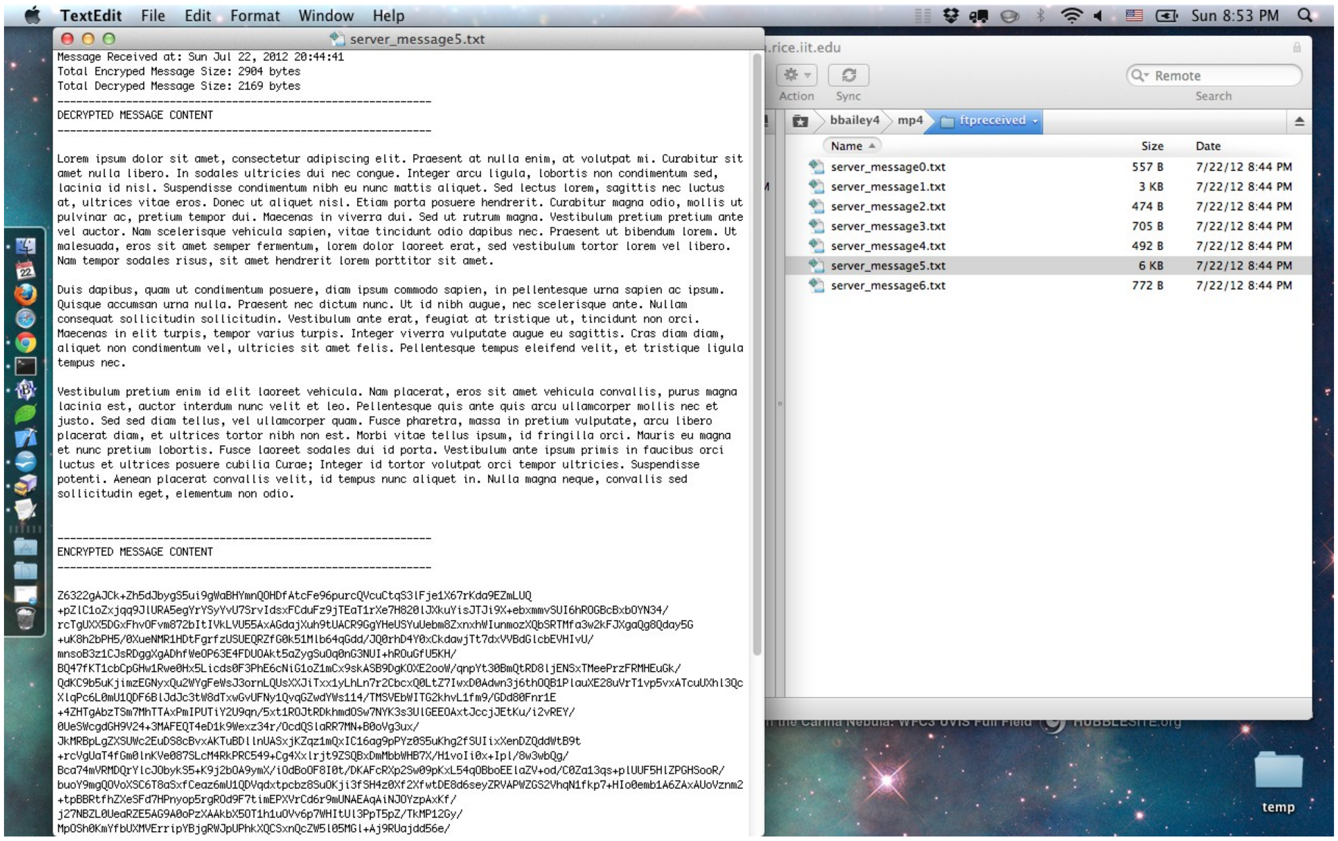
```
------------------------------------------------------------
File name: server_message5.txt
------------------------------------------------------------
Message Received at: Sun Jul 22, 2012 20:44:41

Total Encryped Message Size: 2904 bytes

Total Decryped Message Size: 2169 bytes

------------------------------------------------------------

DECRYPTED MESSAGE CONTENT

------------------------------------------------------------



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent at nulla enim,
 at volutpat mi. Curabitur sit amet nulla libero. In sodales ultricies dui nec c
ongue. Integer arcu ligula, lobortis non condimentum sed, lacinia id nisl. Suspe
ndisse condimentum nibh eu nunc mattis aliquet. Sed lectus lorem, sagittis nec l
uctus at, ultrices vitae eros. Donec ut aliquet nisl. Etiam porta posuere hendre
rit. Curabitur magna odio, mollis ut pulvinar ac, pretium tempor dui. Maecenas i
n viverra dui. Sed ut rutrum magna. Vestibulum pretium pretium ante vel auctor.
```

This screenshot shows the application driver finishing displaying the files and ending the application.



```
Lorem ipsum dolor sit amet, consectetur adipiscing elit posuere.





------------------------------------------------------------

ENCRYPTED MESSAGE CONTENT

------------------------------------------------------------




Ge1UZAny/ia6zN5n3TPEB2O42VT9Om08zCGdj66xPkN2gD8Bbq3QDJDq+F3nDfo9WZ7/ftcUQxbmB/Ha
8+prFEO6R3pMnek+4s/BYYky2Rp8s5dU5CJ3PIQrnpilnlYoXrwSJVeIpCur180yzi7fFWC+x1D1MCcn
0TKenqm3L0wyx3lIJuaaQdyltgy3XqhoDcmpJ49PGUu6WOX01MxyRw==




MP4 Driver: Application Terminating...Server exiting...

bbailey4@glenellyn:~/mp4$
```

This final screenshot shows the ftpreceived directory on the right half of the screen and the server_message5.txt file on the left half of the screen.



```
Message Received at: Sun Jul 22, 2012 20:44:41
Total Encryped Message Size: 2904 bytes
Total Decryped Message Size: 2169 bytes
-------------------------------------------------------------
DECRYPTED MESSAGE CONTENT
-------------------------------------------------------------

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent at nulla enim, at volutpat mi. Curabitur sit
amet nulla libero. In sodales ultricies dui nec congue. Integer arcu ligula, lobortis non condimentum sed,
lacinia id nisl. Suspendisse condimentum nibh eu nunc mattis aliquet. Sed lectus lorem, sagittis nec luctus
at, ultrices vitae eros. Donec ut aliquet nisl. Etiam porta posuere hendrerit. Curabitur magna odio, mollis ut
pulvinar ac, pretium tempor dui. Maecenas in viverra dui. Sed ut rutrum magna. Vestibulum pretium pretium ante
vel auctor. Nam scelerisque vehicula sapien, vitae tincidunt odio dapibus nec. Praesent ut bibendum lorem. Ut
malesuada, eros sit amet semper fermentum, lorem dolor laoreet erat, sed vestibulum tortor lorem vel libero.
Nam tempor sodales risus, sit amet hendrerit lorem porttitor sit amet.

Duis dapibus, quam ut condimentum posuere, diam ipsum commodo sapien, in pellentesque urna sapien ac ipsum.
Quisque accumsan urna nulla. Praesent nec dictum nunc. Ut id nibh augue, nec scelerisque ante. Nullam
consequat sollicitudin sollicitudin. Vestibulum ante erat, feugiat at tristique ut, tincidunt non orci.
Maecenas in elit turpis, tempor varius turpis. Integer viverra vulputate augue eu sagittis. Cras diam diam,
aliquet non condimentum vel, ultricies sit amet felis. Pellentesque tempus eleifend velit, et tristique ligula
tempus nec.

Vestibulum pretium enim id elit laoreet vehicula. Nam placerat, eros sit amet vehicula convallis, purus magna
lacinia est, auctor interdum nunc velit et leo. Pellentesque quis ante quis arcu ullamcorper mollis nec et
justo. Sed sed diam tellus, vel ullamcorper quam. Fusce pharetra, massa in pretium vulputate, arcu libero
placerat diam, et ultrices tortor nibh non est. Morbi vitae tellus ipsum, id fringilla orci. Mauris eu magna
et nunc pretium lobortis. Fusce laoreet sodales dui id porta. Vestibulum ante ipsum primis in faucibus orci
luctus et ultrices posuere cubilia Curae; Integer id tortor volutpat orci tempor ultricies. Suspendisse
potenti. Aenean placerat convallis velit, id tempus nunc aliquet in. Nulla magna neque, convallis sed
sollicitudin eget, elementum non odio.


-------------------------------------------------------------
ENCRYPTED MESSAGE CONTENT
-------------------------------------------------------------
```

```
Z6322gAJCk+Zh5dJbygS5ui9gWaBHYmnQOHDfAtcFe96purcQVcuCtqS3lFje1X67rKda9EZmLUQ
+pZlC1oZxjqq9JlURA5egYrYSyYvU7SrvIdsxFCduFz9jTEaT1rXe7H820lJXkuYisJTJi9X+ebxmmvSUI6hROGBcBxbOYN34/
rcTgUXX5DGxFhvOFvm872bItIVkLVU55AxAGdajXuh9tUACR9GgYHeUSYuUebm8ZxnxhWIunmozXQbSRTMfa3w2kFJXgaQg8Qday5G
+uK8h2bPH5/0XueNMR1HDtFgrfzUSUEQRZfG0k51Mlb64qGdd/JQ0rhD4Y0xCkdawjTt7dxVVBdGlcbEVHIvU/
mnsoB3z1CJsRDggXgADhfWeOP63E4FDUOAkt5aZygSuOq0nG3NUI+hROuGfU5KH/
BQ47fKT1cbCpGHw1Rwe0Hx5Licds0F3PhE6cNiG1oZ1mCx9skASB9DgKOXE2ooW/qnpYt30BmQtRD8ljENSxTMeePrzFRMHEuGk/
QdKC9b5uKjimzEGNyxQu2WYgFeWsJ3ornLQUsXXJiTxx1yLhLn7r2CbcxQ0LtZ7IwxD0Adwn3j6thOQB1PlauXE28uVrT1vp5vxATcuUXhl3Qc
XlqPc6L0mU1QDF6BlJdJc3tW8dTxwGvUFNy1QvqGZwdYWs114/TMSVEbWITG2khvL1fm9/GDd80Fnr1E
+4ZHTgAbzTSm7MhTTAxPmIPUTiY2U9qn/5xt1ROJtRDkhmdOSw7NYK3s3UlGEEOAxtJccjJEtKu/i2vREY/
0UeSWcgdGH9V24+3MAFEQT4eD1k9Wexz34r/OcdQSlaRR7MN+B0oVg3ux/
JkMRBpLgZXSUWc2EuDS8cBvxAKTuBDllnUASxjKZqz1mQxIC16ag9pPYz0S5uKhg2fSUIixXenDZQddWtB9t
+rcVgUaT4fGm0lnKVe087SLcM4RkPRC549+Cg4Xxlrjt9ZSQBxDmMbbWHB7X/H1voIi0x+Ipl/8w3wbQg/
Bca74mVRMDQrYlcJObykS5+K9j2bOA9ymX/iOdBoOF8I0t/DKAFcRXp2Sw09pKxL54qOBboEElaZV+od/C0Za13qs+plUUF5HlZPGHSooR/
buoY9mgQOVoXSC6T8aSxfCeaz6mU1QDVqdxtpcbz8SuOKji3fSH4z0Xf2XfwtDE8d6seyZRVAPWZGS2VhqN1fkp7+HIo0emb1A6ZAxAUoVznm2
+tpBBRtfhZXeSFd7HPnyop5rgROd9F7timEPXVrCd6r9mUNAEAqAiNJOYzpAxKf/
j27NBZL0UeaRZE5AG9A0oPzXAAkbX5OT1h1uOVv6p7WHItUl3PpT5pZ/TkMP12Gy/
MpOSh0KmYfbUXMVErripYBjgRWJpUPhkXQCSxnQcZW5l05MGl+Aj9RUajdd56e/
```

Name	Size	Date
server_message0.txt	557 B	7/22/12 8:44 PM
server_message1.txt	3 KB	7/22/12 8:44 PM
server_message2.txt	474 B	7/22/12 8:44 PM
server_message3.txt	705 B	7/22/12 8:44 PM
server_message4.txt	492 B	7/22/12 8:44 PM
server_message5.txt	6 KB	7/22/12 8:44 PM
server_message6.txt	772 B	7/22/12 8:44 PM