Eric Watson

ECE440 Homework 3 – Network Application

This homework assignment required us to take the TCP server and client functions that we had experimented with in our last assignment and introduce a measure of security to prevent messages from being read by someone monitoring the communication channel. To do so we created a substitution cipher, which would substitute one ASCII character for another to encrypt the message. I chose to implement my substitution cipher by adding a specified 'key' value to each ASCII character of the message, shifting forward along the alphabet and then wrapping back. To simplify, we limited the input character set to only use upper case letters, which are represented as 65 to 90 in ASCII format. It was also necessary to check for space characters, line feed, carriage return, and null characters during the implementation, all of which were transmitted without any changes during encryption. Wrapping the alphabet values meant reverting the value back to the start of this sequence of 26 numbers (65 to 90) once the shifted value had passed 90, simulating modulo arithmetic. For example, using *key =4* if the message contained the letter "Y" then this would be substituted with the character "C". The encrypted message would then be transmitted from the TCP server to the TCP client. The TCP client would write back the message as confirmation, and the TCP server would verify the messages match. Once received and verified as matching, the TCP client would then decrypt the message using the same method of substitution, shifting in the opposite direction using the key value.

The captured terminal output below demonstrates the functionality of my code. After initiating the TCP Server, my server program prints the original input message followed by the encrypted message that is sent. Similarly, my TCP Client program prints the received encrypted message, followed by the decrypted message after arrival.

TCP SERVER

```
Erics-MacBook-Pro-2:HW03 Watson$ ./tcp_server localhost

Starting server
Making socket
Binding to port 0
opened socket as: fd (3) on port (52859) for stream i/o
Server:
              sin_family        = 2
              sin_addr.s_addr   = 0
              sin_port          = 52859

Making a listen queue of 5 elements
Waiting for a connection

Got a connection
* NOTICE: All messages will be converted into uppercase.
Messages should not contain any special characters or numbers. *
Please input your message: hello this is a test xyz

 Original Message: "HELLO THIS IS A TEST XYZ"
 Encrypted Message: "LIPPS XLMW MW E XIWX BCD"

Sending "LIPPS XLMW MW E XIWX BCD" to client
The messages match
Closing the socket
Waiting for a connection
```

TCP CLIENT

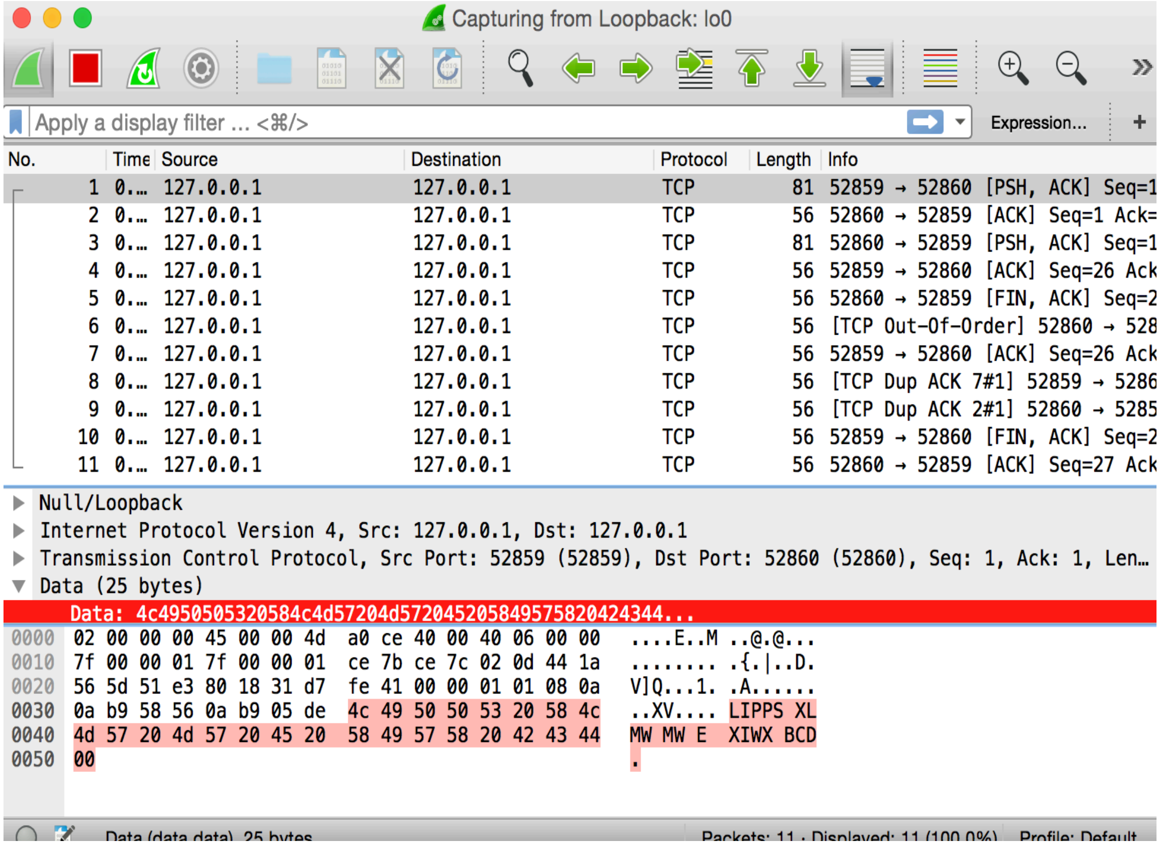
```
Erics-MacBook-Pro-2:HW03 Watson$ ./tcp_client localhost 52859

Making a socket
Connecting to localhost on port 52859
Received "LIPPS XLMW MW E XIWX BCD" from server

Writing "LIPPS XLMW MW E XIWX BCD" to server
Decrypted Messaged: "HELLO THIS IS A TEST XYZ"

Closing socket
```

Wireshark allows us to monitor network activity. The screenshot below shows the encrypted message being transmitted between TCP server and client.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.…	127.0.0.1	127.0.0.1	TCP	81	52859 → 52860 [PSH, ACK] Seq=1
2	0.…	127.0.0.1	127.0.0.1	TCP	56	52860 → 52859 [ACK] Seq=1 Ack=
3	0.…	127.0.0.1	127.0.0.1	TCP	81	52860 → 52859 [PSH, ACK] Seq=1
4	0.…	127.0.0.1	127.0.0.1	TCP	56	52859 → 52860 [ACK] Seq=26 Ack
5	0.…	127.0.0.1	127.0.0.1	TCP	56	52860 → 52859 [FIN, ACK] Seq=2
6	0.…	127.0.0.1	127.0.0.1	TCP	56	[TCP Out-Of-Order] 52860 → 528
7	0.…	127.0.0.1	127.0.0.1	TCP	56	52859 → 52860 [ACK] Seq=26 Ack
8	0.…	127.0.0.1	127.0.0.1	TCP	56	[TCP Dup ACK 7#1] 52859 → 5286
9	0.…	127.0.0.1	127.0.0.1	TCP	56	[TCP Dup ACK 2#1] 52860 → 5285
10	0.…	127.0.0.1	127.0.0.1	TCP	56	52859 → 52860 [FIN, ACK] Seq=2
11	0.…	127.0.0.1	127.0.0.1	TCP	56	52860 → 52859 [ACK] Seq=27 Ack

```
▶ Null/Loopback
▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▶ Transmission Control Protocol, Src Port: 52859 (52859), Dst Port: 52860 (52860), Seq: 1, Ack: 1, Len…
▼ Data (25 bytes)
    Data: 4c4950505320584c4d57204d57204520584957582042434 4...
```

```
0000  02 00 00 00 45 00 00 4d  a0 ce 40 00 40 06 00 00   ....E..M ..@.@...
0010  7f 00 00 01 7f 00 00 01  ce 7b ce 7c 02 0d 44 1a   ........ .{.|..D.
0020  56 5d 51 e3 80 18 31 d7  fe 41 00 00 01 01 08 0a   V]Q...1. .A......
0030  0a b9 58 56 0a b9 05 de  4c 49 50 50 53 20 58 4c   ..XV.... LIPPS XL
0040  4d 57 20 4d 57 20 45 20  58 49 57 58 20 42 43 44   MW MW E  XIWX BCD
0050  00                                                 .
```

Data (data.data), 25 bytes   Packets: 11 · Displayed: 11 (100.0%)   Profile: Default