Project Report

- This project is written in C using the OpenSSL libraries for encrypting data.

- Connections are made using sockets.

- The usage of this program is exactly as specified in the problem statement given.

- The encryption/decryption of the network payload is done using AES 128 CTR mode.

- The pbproxy program was tested using a NETCAT session for simple interchange of messages between one system and the other. There were no problems observed. The program runs as expected.

To use Netcat the following commands were used.

- Terminal 1 - “nc -l -p 7777” {Netcat Server}

- Terminal 2 - “./pbproxy -l 2222 -k mykey localhost 7777localhost” {PBPROXY Server}

- Terminal 3 - “./pbproxy -k mykey localhost 2222” {PBPROXY Client}

{Snapshots are attached below.}

- The program was also checked with an SSH session.

{Warning: The program runs perfectly when both the client and the server are on the same system. I however encountered problems when running the code on my other pc running Ubuntu 16.04. The cause is generally attributed to some problem with either the drivers or the configuration of the Ethernet device. The program ran on other people's machines without a hitch however. It was also running on my older pc which ran Ubuntu 14.04}.

Commands used were:

- Terminal 1 - “./pbproxy -k mykey -l 2222 localhost 22” {Pbproxy server}

- Terminal 2 – 'ssh -o "ProxyCommand ./pbproxy -k mykey localhost 2222" localhost' {ssh client using pbproxy as proxy service}

- The ssh server was configured to use port 22.

- The pbproxy server was started, and then an ssh client was opened using the pbproxy program (in client mode) as the proxy. Upon entering the correct password, I was able to use the SSH Session.

- The server keeps on listening for connections requests. Every time a connection is accepted, details are passed to a thread function which then runs communications between the client and SSH server. Each thread opens a line of communication between the client and service, acting as a proxy in-between them.

- All communication is encrypted using AES with an Initialization Vector of size 8 which is passed along with the payload.

- A symmetric key is read from a file specified by the user.

- After a session is ended, the server continues to listen for further connections, unless forcibly closed.

- ONLY 1 connection at a time is allowed.

- Attached in the zip file is a sample keyfile that I have been using for encryption in my testing.

- To build the code, simply enter the folder and run the “make” command.

PS : There are screenshots of the code running on my machine attached to the submitted zip file. Please take a look.

For references, I have used information obtained from various websites such as Wikipedia, StackOverflow, and so on. Some important ones are mentioned below.

- <https://stackoverflow.com/questions/38255433/parameter-details-of-openssls-aes-ctr128-encrypt>

- <https://stackoverflow.com/questions/37498912/openssl-aes-128-ctr-in-c>

- <http://www.gurutechnologies.net/blog/aes-ctr-encryption-in-c/>

- <http://stackoverflow.com/questions/174531/easiest-way-to-get-files-contents-in-c>

- <http://manpages.ubuntu.com/manpages/zesty/man2/fcntl.2.html>

- <http://www.roman10.net/2011/12/02/simple-tcp-socket-client-and-server-communication-in-c-under-linux/>

- <http://www.delorie.com/gnu/docs/glibc/libc_525.html>

- http://timmurphy.org/2010/05/04/pthreads-in-c-a-minimal-working-example/