Copyright © 2015 Jonathan Jensen

1. INTRODUCTION

1.1 Purpose

This software design document describes the architecture and system design of Jonathan Jensen's TauNet project.

1.2 Scope

In an attempt to protect sensitive communications from probing eyes or oppressive government bodies, TauNet will be a secure communications network meant for a group of trusted and verified members.

1.3 Reference Material

TauNet Communications Protocol v0.2 (revision 1)

Can be found at:

https://docs.google.com/document/d/1juKX1KE8FnpVBpb9S6RHwLm2DpCJv0RnuKHOfOK-h7Y/edit

1.4 Definitions and Acronyms

Term	Definition
RC4	Form of encryption
Node	A raspberry pi, running TauNet, that is connected to the network/ An instance of TauNet
User	An assumed human that is using a TauNet Node to communicate within the network.
System	The TauNet system/program

2. SYSTEM OVERVIEW

TauNet is a simple messaging application. On the client side it is able to view a .csv file consisting of contacts and let the user select one of those contacts. It then makes a socket connection with the user selected contact and allows the user to input a message to send to that contact. TauNet then encrypts the message and sends it to the connected contact.

On the server side, TauNet will listen for incoming socket connections, accept an incoming connection, and then accept the incoming byte stream. It will decrypt the byte stream and save it to a text file that is also accessible by the client for viewing. In addition to that it will write connection information to a server log text file.

3. SYSTEM ARCHITECTURE

3.1 Architectural Design

At a relatively high level, TauNet will simply be a client subsystem and a server subsystem. In order to simplify concurrency, the client and server subsystems will be implemented as different projects and therefore different processes. Instead of a dedicated server for the network as a traditional client server application would have, each TauNet node will be running its own "server". Both subsystems will have access to a Utilities class that will be made up of all the functionality that is relevant to both subsystems, such as encryption. In this way the subsystems are separate but may interact resulting in ideally, a lowly coupled but highly cohesive design.

3.2 Design Rationale

The client server model was selected because of its ease of implementation. Because of its prevalence there are many resources to reference when needed. Since the client[person that system is being built for] expressed that a "single point of failure" design must be avoided, the server module was integrated into the same solution as the client, therefore deviating from the traditional client server model which uses a single server module for the entire network. This introduced the issue of concurrency between the client and server. The server needs to be listening for incoming connections at all times, even when the client is making a connection with a different not or performing other operations. To simplify the design and ensure reliable concurrency, the client and server modules will be separated and run as separate processes, the operating system that TauNet is running on will be responsible for the concurrency in that aspect. This results in a system that uses a client and server module, but sort of operates as a peer to peer network.

4. DATA DESIGN

4.1 Data Description

A SQL based database was considered for this implementation but ultimately, it was decided that a fully featured database was not necessary. Storing each message as an individual object in memory was also considered. This would benefit security given that the messages would surely be destroyed upon closing the application.

5. HUMAN INTERFACE DESIGN

5.1 Overview of User Interface

The user interface will be a simple console based command driven interface. The user will be guided through the system by prompts showing up in the console window.