TauNet Software Design Document

Document Version 1.0

Abstract
This document describes the architectural design of the TauNet system. This document covers component breakdown, component functionality, and design rationale.

Status of This Memo
This document is provided as part of the requirements for TauNet, as specified in Portland State University CS 300 Section 1 Fall 2015.

1.0 Introduction

1.1 Purpose
The purpose of this document is to inform developers about the inner workings of the TauNet system. This document describes the architecture of the TauNet system, the coupling of the individual subsystems, and the function of those subsystems.

1.2 Scope
The purpose of this software is to provide a platform for encrypted communication.

1.3 Overview
This document describes the overall system architecture, the subsystem design, and the rationale behind those designs.

2.0 System Architecture
This system follows a loose object oriented design. Functionality is separated into roughly independent subsystems. Each subsystem that needs the services of another will contain a pointer to the required subsystem.

2.1 Architecture Design
The functionality of this program is separated into several subsystems described in sections 2.1.x

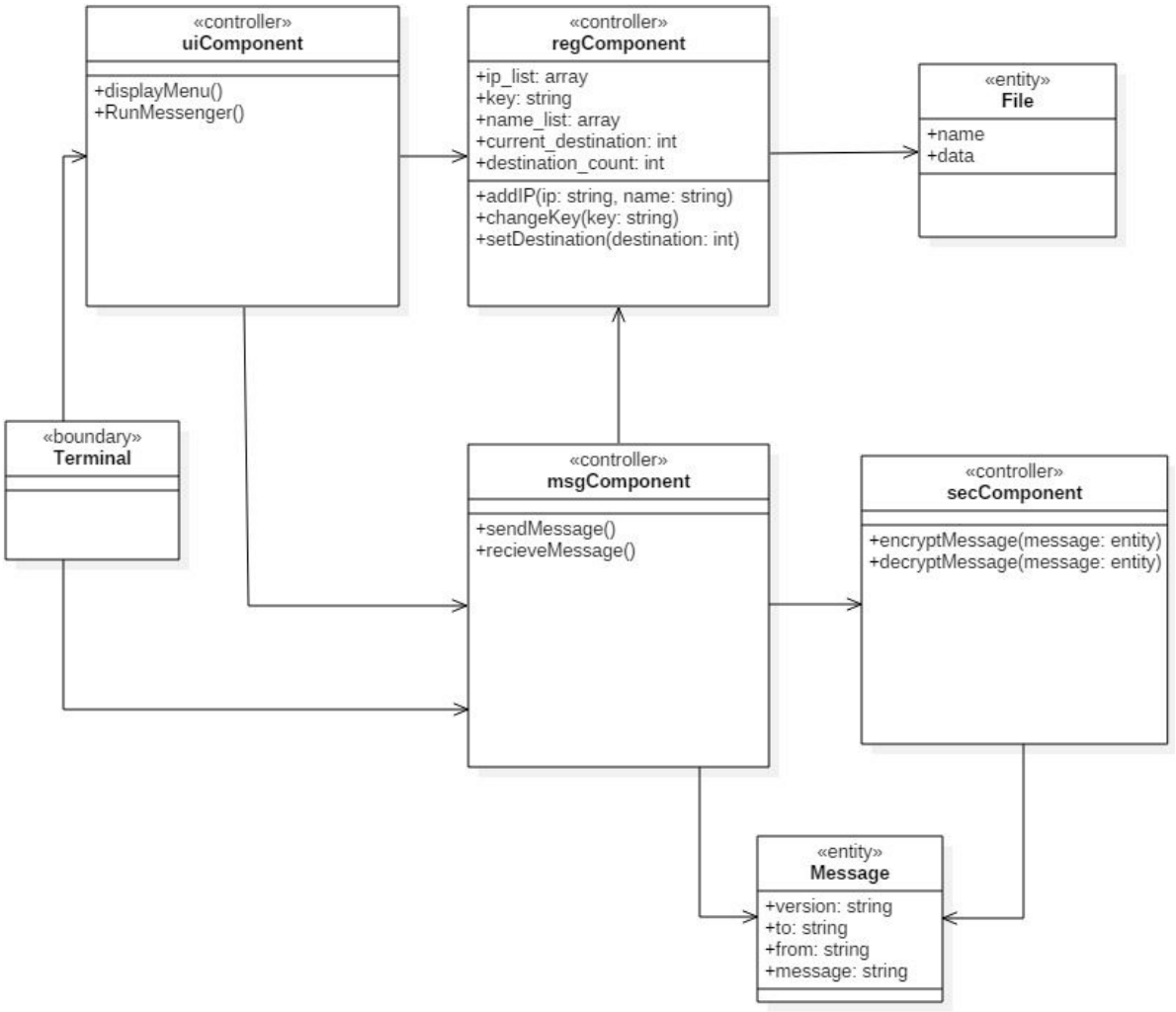
2.1.1 User Interface
The User Interface Menu is contained in the *uiComponent*. This component is responsible for drawing menus and handling user input. This subsystem allow the modification of the user table, the ip table, and the crypto key via the settings menu. This subsystem uses the *regComponent* to modify program data.
User Interface drawing during messaging is handled by the *msgComponent*.

2.1.2 Data Passing and File Management
The data and user files of the TauNet system are managed by the *regComponent*.

2.1.3 Messaging / Network Connections
The network connections and message sending are managed by the *msgComponent*. This component uses the *regComponent* and *secComponent* during operation.

2.1.4 Data Encryption
The encryption and decryption of messages is the responsibility of the *secComponent*.

2.2 Decomposition Design



2.3 Design Rationale
The rational for an object oriented design to to provide a system with high cohesion and low coupling between subsystems. The separation of functionality into subsystems creates boundaries that are useful for testing. Decomposed subsystem are also useful to simplify design and understanding of the system.