Entwicklerdokumentation

Bei dem vorgestellten Programm handelt es sich um einen Messenger, über den mithilfe des Client-Server-Modells zwei Chatpartner miteinander kommunizieren können. Dabei werden eingegebene Nachrichten vor dem Versenden verschlüsselt und erst beim Empfänger wieder entschlüsselt.

Allgemeine Informationen

* Das Programm wurde in einer UNIX-Umgebung entwickelt und ist auch nur für diese gedacht. Es wurde auf OS X 10.14.2 getestet. Es ist nicht möglich, das Programm in einer UNIX-fremden Umgebung zu nutzen.
* Zur Nutzung des Programms wird eine bestehende Internetverbindung benötigt. Außerdem muss der Client Kenntnis über die IP-Adresse des Servers sowie dessen ausgewählten Port haben.
* Um den Code zu kompilieren, muss lediglich das Makefile mit dem Befehl „make“ ausgeführt werden. Anschließend lässt sich das Programm wie gewohnt starten.

* Nachdem das Programm kompiliert wurde, wird es wie folgt gestartet:
  + Der Server startet das Programm in der Kommandozeile (./Messenger) und wählt Option zwei („Create host“) aus. Er gibt eine Portnummer zwischen 2000 und 65535 ein.
  + Der Client startet das Programm auf gleiche Weise. Er wählt Option eins („Connect to host“). Anschließend gibt er zunächst die IP-Adresse des Servers und dann den ausgewählten Port ein.
  + Nun können beide miteinander kommunizieren.

Schlüsseldateien:

main.c

* Funktion: Die main-Funktion wird zu Beginn des Programms aufgerufen. Sie dient lediglich dazu, zu entscheiden, ob es sich um den Server oder den Client handeln soll.
* Eingebundene Bibliotheken:
  + <stdio.h>

server.c

* Funktion: Der Server erstellt zunächst einen neuen Socket und wartet anschließend darauf, dass sich ein Client mit diesem verbindet. Sobald das geschehen ist, wird eine Schleife gestartet, in der der Benutzer immer wieder gefragt wird, ob er eine neue Nachricht senden oder nach eingegangenen Nachrichten suchen will.
* Eingebundene Bibliotheken:
  + <stdio.h>
  + <string.h>
  + <netinet/in.h>

client.c

* Funktion: Der Client erstellt ebenfalls einen neuen Socket und verbindet sich mithilfe des Wissens über IP-Adresse und Portnummer mit dem Socket des Servers. Danach wird die gleiche Schleife aufgerufen, die auch der Server nutzt, da Client und Server ab diesem Punkt gleichwertige Chatpartner sind.
* Eingebundene Bibliotheken:
  + <stdio.h>
  + <stdlib.h>
  + <string.h>
  + <netdb.h>

functions.c

* Funktion: Die Datei functions.c enthält die Funktionen, welche sowohl vom Client, als auch vom Server aufgerufen werden.
* Eingebundene Bibliotheken:
  + <stdio.h>
  + <stdlib.h>
  + <string.h>
  + <unistd.h>

encryption.c

* Funktion: Die Datei enthält die Funktionen zum Ver- und Entschlüsseln der Nachrichten.
* Eingebundene Bibliotheken:
  + <stdio.h>
  + <string.h>

makefile

* Funktion:

Funktionen und Unterfunktionen

main.c

* main()
  + Unterteilt den Nutzer in Server und Client.
  + Eingabeparameter: void
  + Ausgabeparameter: void

server.c

* startServer()
  + Erstellt einen Socket für den Server und wartet auf einen Client. Ruft anschließend eine Schleife auf, in der der Nutzer aufgerufen wird, auszuwählen, ob er eine neue Nachricht versenden oder nach neuen empfangenen Nachrichten suchen will.
  + Eingabeparameter: void
  + Ausgabeparameter: void

client.c

* startClient()
  + Erstellt einen Socket fürden Client und verbindet sich mit dem Server. Ruft anschließend die gleiche Schleife auf wie der Server.
  + Eingabeparameter: void
  + Ausgabeparameter: void

functions.c

* error()
  + Ausgabe einer Fehlermeldung und Beenden des Programms
  + Eingabeparameter: char\*
  + Ausgabeparameter: void
* writeMessage()
  + Sendet eine neue Nachricht an den Chatpartner.
  + Eingabeparameter: int, char\*
  + Ausgabeparameter: void
* readMessage()
  + Sucht nach neuen empfangenen Nachrichten und gibt diese aus.
  + Eingabeparameter: int
  + Ausgabeparameter: void
* enterPort()
  + Fordert den Nutzer zur Eingabe einer Portnummer auf.
  + Eingabeparameter: int\*
  + Ausgabeparameter: void

encryption.c

* len()
  + Ersetzt strlen().
  + Eingabeparameter: const char\*
  + Ausgabeparameter: int
* intlen()
  + Zählt die Länge eines Integers.
  + Eingabeparameter: const int\*
  + Ausgabeparameter: int
* conv\_key()
  + Konvertiert den Schlüssel zu Zahlen.
  + Eingabeparameter: const char\*
  + Ausgabeparameter: int\*
* crypter()
  + Verschlüsselt einen Text.
  + Eingabeparameter: const char\*, const char\*
  + Ausgabeparameter: char\*
* decrypter()
  + Entschlüsselt eine Nachricht.
  + Eingabeparameter: const char\*, const char\*
  + Ausgabeparameter: char\*