# Encrypto

### Eine verschlüsselte Messenger-Applikation

## Ziel des Programmes

Der Ziel meines Programmes ist eine sichere und deswegen verschlüsselte Weise von Kommunikation zu erlauben. Heutzutage Tag nach Tag hört man, dass eine neue Weise ausgefunden wird, damit Regierungen, Privatorganisationen oder Privatpersonen unsere Daten erreichen können, die wir anderwegs nur für uns bleiben gelassen hätten. Natürlich existieren Lösungen dafür, die viel mehr Funktionalität enthalten, als meiner Programm, aber trotzdem finde ich meine Idee nützlich, weil es noch Zeite gibt, wenn man eine einfachere Lösung sucht - es soll nichts spezielles tun, sondern sollte es seine wenige Funktionalitäten sicher und anspruchsvoll ausführen - diesen Niche erfüllt meiner Programm.

## Verwendung des Programmes

Der Programm hat verschiedene logische Funktionalitäten, zwischen denen der Benutzer durch CLI Eingangsparametern wählen kann. Die generale Formel für diese Programm ist:

```
encrypto [-m mode] [-c client] [-d directory] [-k keyfiles]
```

Diese Parameter sind:

1. Mode
    - `server`: Server - Eine Server muss für jede Chat-Room gestartet werden. Diese Server muss an eine solche Maschine laufen, die sichtbar für die Teilnehmer ist - falls einige Leute über die Internet (und nicht über LAN) übermitteln möchten, dann soll diese Server-Maschine eine offene IP haben - diese (in allgemein) bedeutet, dass Port-Forwarding erwarten ist. Über LAN ist es nicht so kompliziert, weil dann sollte nur die gemeinsame Punkt (Router, Switch, usw.) so eingestellt werden, dass es diese Sichtbarkeit erlaubt. Über eine bestimmte (und erlaubte) Port dient es auch eine einfache Representation von den Daten die es bekomment hat - um mehr Sichtbarkeit über die geteilte Informationen zu bieten.
    - `client`: Client - Die Benutzer müssen je eine Client laufen lassen, um die Programm benutzen zu können. Die Funktionalitäten sind:
        - Verschlüsselungsschlüsseln generieren, einladen und speichern - Möglichkeit für unterschiedliche Optionen: RDRAND-basiert (wenn möglich), /dev/random-basier, /dev/urandom-basiert für Generation, PEM für Einladung und Speicherung - Jede Operation an einem nicht swappbaren Memorieteil (durch System Call mlock()) durchfüren und nach dem Benutz die ganze Speicheranteil ausnullen.
        - Verschlüsselung ist jezt nur auf RSA und AES-256 basiert (RSA für Initialisierung, dann eine Session-AES-key generieren für Kommunikation, und AES für die Verschlüsselung der Geheimschlüsseln auf dem Disk.) Diese Operationen sind von der Entwickler dieses Programmes auch implementiert, also sind nicht sicher, und sollten jetzt nicht in Produktion benutzt werden! (Eine libgcrypt oder openSSL Implementation ist geplannt.)
        - Interfacemöglichkeiten sind im Punkt 2 erklärt.

2. Client - Die User-Interface
    - `cli`: CLI-basierte - in dem passenden Terminal laufen lassen, Eingänge aus dem Drücktaste direkt einlesen und Ausgänge direkt zu dem Terminal ausschreiben.
    - `gui-gtk`: GTK±basierte - Native-aussehende Widget Toolkit benutzen, darin verschiedene Elementen für Einlesen und Ausschreiben benutzen.

3. Directory
    - `path/to/directory/`: Die Folder, in dem die verschidene Daten gespeichert werden.

4. Keys
    - `path/to/directory/`: Die Folder, in dem die Schlüsseln gespeichert sind/werden kann.

Beispiel:

`encrypto -c cli -d ~/encrypto/data/ -k ~/keys/` wird den Programm als Client starten (Default, nicht eingegeben), ohne graphicalsche Interface mit dem Working Directory in /home/%USERNAME/encrypto/data, und die Keyfiles in /home/%USERNAME/keys.

## Files

1. Keyfiles - `.key`, folgen die PEM-Standard (Definiert in RFC’s 1421 durch 1424)
2. Datafiles - `.json`, folgen die JSON-Standard (Definiert in ECMA-262 3rd Edition - December 1999.)