momedo

Open Source Mobiler Messenger für kommunale und schulische Zwecke

View project on GitHub

momedo

Open Source Mobiler Messenger für kommunale und schulische Zwecke mit Verschlüsselung

Gliederung:


```
Bibliographischer Verweis:
Momedo: Open Source Mobiler Messenger für kommunale und schulische Zwecke mit Verschlüsselung,
Github, URL: https://momedo.github.io/momedo/
&   https://github.com/momedo/momedo/blob/master/README.md , 2018
```


1 Ausgangslage

Mitarbeiterinnen und Mitarbeiter (mit allen Begriffen sind jeweils immer alle Geschlechter gemeint) vieler Kommunalverwaltungen in zahlreichen Städten sowie auch Schüler und Lehrer in den kommunalen Schulen vernetzen sich zunehmend nicht nur für privaten Kontakt unter Kollegen, sondern auch für berufliche Zusammenhänge und Arbeitsprozesse

über moderne Messenger Applikationen auf mobilen Smartphones. Eine beliebte Applikation ist dabei die zu Facebook gehörende Applikation WhatsApp.

Wir wollen mit dieser Expertise der Fragestellung nachgehen, ob WhatsApp auch für die Arbeitsprozesse der Kommunalverwaltung einer Stadt bzw. die Kommunikation der Beschäftigen der Kommunalverwaltung darüber eingesetzt werden kann.

Um die Antwort vorwegzunehmen: Sie lautet: Nein: WhatsApp kann – auch wenn sie im privaten Bereich eine der Beliebtesten Applikationen auf dem Smartphone zu sein scheint, aus rechtlichen, technischen, sicherheitsrelevanten und auch arbeitsschritt-prozessualen Gesichtspunkten nicht angewandt werden.

Es muss daher nach Alternativen gesucht werden, die die Spezifikationen, Anforderungen und Bedürfnisse der Kommunalen Fachbereiche abdecken, zu denen die vorliegende Expertise eine Handlungsempfehlung abgeben will für erstens die Erkenntnis, dass hier dringend in den genannten Feldern Handlungsbedarf besteht und zweitens für einen Gestaltungsprozess, beispielsweise über ein Pilotprojekt, dass für die Anforderungen der kommunalen Fachbereiche eine flexible Applikation erstellt. Diese sollte auch an die Arbeitsprozesse der Kommunalverwaltung anbindbar sein.

Was ist also zu tun, wenn Mitarbeiter Arbeitsprozesse über WhatsApp regeln, diese jedoch damit gegen rechtliche Auflagen, sicherheitstechnische Bedenken verstoßen und damit ein Practice-Modell etablieren, dass es nicht ermöglicht, z.B. durch Plugins arbeitsschritt-prozessuale Anforderungen der Digitalisierung in der Kommunalverwaltung einzubinden und um zugleich Arbeitsprozesse überhaupt noch selbst definieren zu können?

Whatsapp hat also entscheidende Auswirkungen auf die Ausgestaltung, Sicherheit und den Workflow der Arbeitsprozesse und ist rechtlich bedenklich.

Datenschützer haben z.B. zurecht darauf hingewiesen, dass WhatsApp z.B. an Schulen rechtlich nicht eingesetzt werden darf (vgl. Smoltczyk 2016, DSGVOApp 2018). Durch den Upload des gesamten privaten Telefonbuches an WhatsApp werden die Rechte des Kontaktes verletzt, der theoretisch gegen seinen Freund oder Geschäftspartner eine Abmahnung und Unterlassungserklärung erwirken kann, wenn dieser die Daten ungefragt ohne seine Zustimmung an WhatsApp weitergibt.

Ein Beispiel: Ein Mitarbeiter der Kommunalverwaltung sendet per WhatsApp die Mitteilung an einen Hausmeister, dass in einem Gebäude eine Glühlampe nun repariert werden könne und die Freigabe erteilt sei. Der Hausmeister könnte nun gegen den Mitarbeiter der Kommunalverwaltung klagen – wenn er nicht über WhatsApp kontaktiert werden möchte – dass der Mitarbeiter der Kommunalverwaltung seine Telefonnummer an einen amerikanischen Dienst ungefragt übermittelt hat. Bislang gibt es diese Klagen nicht, jedoch ist es eine reine klare rechtliche Ausgangslage, die als Risikobewertung in unsere Analyse einfließen muss. Sollte jemand klagen, drohen Strafen mit 4% des Jahresumsatzes bzw. Haushaltsplanes (Iseninfos 2018).

WhatApp gehört zu Facebook und gleicht auch die Daten und Telefonnumer mit diesem Konzern ab – was rechtlich ebenso unzulässig ist, so dass bereits Strafzahlungen in von mehreren ländern in Europa ausgesprochen wurden (vgl. Spanien und Irland, Greis 2018).

Die ab Mai 2018 gültige EU-Datenschutz-Grundverordnung (DSGVO 2016, Wikipedia 2018) ist auch für die Kommunalverwaltungen der Städte umzusetzen und gilt nicht nur für die Rechte der Kunden und Kommunikationsteilnehmer in der Kommunalverwaltung, sondern auch für die Sicherheitsauflagen der Geschäftsprozesse. Eine verschlüsselte Kommunikation ist daher nach DSGVO anzubieten (z.B. generell beim Upload von Bewerbungsunterlagen bei Stellenausschreibungen) und auch relevante Unterlagen und Vorgänge sind vor den Augen unbefugter Dritter zu schützen (z.B. Sachbearbeitung und Kundenkontakt sind durch bauliche Maßnahmen zu trennen, damit der nächste Kunde nicht auf die herumliegenden Daten des vorherigen Kunden schauen kann):

Für eine Kommunikation von Beschäftigen einer Stadtverwaltung untereinander und zu Lieferanten und Partnern in der Arbeitsprozess-Kette kann über WhatsApp dieses Schutzbedürfnis nach EU-DSGVO nicht optimal umgesetzt werden und ist verboten (Art 2 Abs 2 lit c EU-DSGVO). Auch aus diesen Gründen der datenschutzrechtlichen Anforderungen wird ein hauseigener Betrieb eines mobilen Kommunikations-Servers und einer App für Android und Apple notwendig.

Weiterhin ist mit BYOD – Bring your own Device – seine Situation eingetreten, dass berufliche Kommunikation über Whatsapp über dienstliche und auch private Handies eingetreten ist (Tung 2017). Dieses ist als völliger Wildwuchs zu bezeichnen, Stadtverwaltungen haben hier keine Kontrolle mehr über die Kommunikationswege und damit auch der dienstlichen Nutzung und beispielsweise Geheimhaltungnotwendigkeiten von Geschäftsinformationen wie auch geschäftlicher Daten.

Auch arbeitsschutzgesetzlicher und -rechtlicher Anforderungen können bei der kommunalen Kommunikation über amerkanische Server-Apps auf Privattelefonen nicht mehr geregelt werden. Eine Betriebsvereinbarung wie sie manche Automobilhersteller mit den Personalräten getroffen, haben, dass die E-Mail-Kommunikation nach 18 Uhr auf Mobilfunkgeräte ausgesetzt wird, um bei den Beschäftigten eine Entgrenzung von Arbeitszeitreglungen durch Technik zu verhindern, könnten Betriebsräte z.B. nicht mehr einfordern, wenn dazu private Geräte und Kanäle über amerikanischen Servern genutzt werden (Frankfurter Rundschau 2014).

Wenn zu berücksichtigen ist, dass auch beispielsweise Dateianhänge über WhatsApp versandt werden, die kaufmännische oder steuerliche Relevanz haben, sind diese revisionssicher mindestens fünf bzw. zehn Jahre aufzubewahren (Handelskammer 2018).

In den Kommunalverwaltungen unserer Städte gibt es zahleiche dienstliche Handies, die einen kommunalen Messenger erfordern und ggf. derzeit WhatsApp installiert haben. Die Anzahl der Beschäftigen und die Anzahl der Lieferanten und Arbeitsprozess-Partner, die die Kommunikation mit der Stadtverwaltung über private Geräte über WhatsApp aufnehmen, kann jeweils nur geschätzt werden – zeigt jedoch den strukturellen Bedarf auf, diesen Wildwuchs der Komunikationsprozesse mit einer eigenen Messenger App die kommunalen Zwecke für eine jeweilige Stadt zu begegnen.

Neben der Nutzer- und Gerätevielfalt, geht es aber auch vor allem um den Erhalt der Gestaltungsfähigkeit der eigenen Arbeitsprozesse. Die Digitalisierung von Arbeitsprozessen greift in allen Fachbereichen und Ämtern einer Stadt und geschlosse, nicht quell-offene Systeme wie WhatsApp erlauben nicht die Optionen z.B. mittels Plugins an einen solchen Messengern Arbeitsdaten und Arbeitsschritte anzuflanschen.

Zunehmend werden Speicherorte von Daten bewusster ausgewählt - sie sollen im eigenen Lande bzw. Haus liegen: Die Speicherung von einer Organisationskommunikation sollte daher über eigene Server stattfinden, die auch eine Verschlüsselung der Kommunikation ermöglichen.

Hier geht es nicht nur um den Schutz vor an Internetknoten möglicherweise abhörenden Agenturen, sondern z.B. um ganz einfache Prozesse auf Dienst-Handys, mit denen berufliche Dateien und Informationen mit einem Wisch auch in die Kontaktdaten-Kanäle zu Privatkontakten weitergegeben werden könnten. Beschäftigte mit erweiterten und spezifischen Datenschutzanforderungen wird es somit sehr leicht gemacht, Geschäftsprozess-Informationen an unbefugte Dritte weiterzugeben – ohne dass die Organisation es nachweisen kann, weil sie die Kommunikationsprozesse nicht mehr durch das Angebot einer Alternative mit eigenem Server steuern kann. Oder: Die berufliche Kommunikation und deren Daten findet gleich komplett in semi-privaten Informationskanälen statt, auf die Kommunen keinen Einfluss bei der Geschäftsprozessmodelierung mehr haben.

Zusammenfassend lassen sich also mit folgenden Stichworten die Begründungszusammenhänge finden, warum eine Organisationskommunikation in der Verwaltung wie auch in kommunalen Schulen über WhatsApp nicht tragbar ist und daher eine kommunale Alternative zu schaffen ist.

7 Gründe für die Abkehr von kommunaler Kommunikation über WhatsApp:

```
* ********************************************************************************************************
*
*            7 Gründe für die Abkehr von kommunaler Kommunikation über WhatsApp:
*
*   1  ABMAHN-SICHERHEIT: Rechtliche Abmahnung in Deutschland möglich aufgrund technischem
*      inhärenten Prozess, dass Kontaktdaten vorwiegend ungefragt auf den zentralen Server
*      geladen werden.
```

```
*
*   2  DATENSCHUTZ: Datenschutzanforderungen und neuere Erkenntnisse zur Notwendigkeit von
*      Verschlüsselung erfordern die Abkehr von amerikanischen Servern hin zur
*      Kommunikationsservern, die in eigener, kommunaler Hand sind.
*
*   3  EU-DSGVO: Einbezug der erweiterten neuen Anforderungen der EU-DSGVO
*      Wildwuchs von Organisationskommunikation über private Telefone, die nicht mehr
*      gestaltet werden können.
*
*   4  RECHTLICHE ANFORDERUNGEN: Rechtliche Anforderungen an Revisionssicherheit bei
*      Kommunikation und Daten bzw. Dateianhängen
*      z.B. hinsichtlich kaufmännischer oder gar steuerlicher Relevanz
*
*   5  MESSENGER ALS ETABLIERTER STANDARD: Ein System über E-Mail auf mobilen Geräten
*      wird erfahrungsgemäß nicht genutzt
*      und ist derzeit nicht etabliert für Verschlüsselten versandt über mobile Geräte.
*      Hierzu werden Messenger zur Kommunikation genutzt.
*
*   6  GEFAHR VON NICHT ADMINISTRIERBARER PARALLEL-KOMMUNIKATION BEI NUTZUNG PRIVATER GERÄTE:
*      Gefahr der Auslagerung von organisationaler Kommunikation auf private Kanäle, bei der
*      die Organisationalen Arbeitsprozesse und deren Regelungen nicht mehr gestaltbar sind.
*
*   7  KEINE BEDARFSGERECHTE ENTWICKLUNGSMÖGLICHKEITEN IN PROPRIETÄREN SYSTEMEN: Ein Messenger
*      macht nur Sinn, wenn dieser keine proprietäre Lösung ist, sondern ein offenes, d.h.
*      quelloffenes System darstellt, mit dem man Anforderungen der Fachbereiche
*      im Zuge der Digitalisierung einbauen kann, z.B. über Plugins.
*
* ********************************************************************************************
```

Um diesen Anforderungen und Risikobetrachtungen bei der Nutzung von Whatsapp Einhalt zu gebieten bzw. eine Alternaitive auszurollen, wäre nach Erstellung einer quelloffenen, flexibel anpassbaren Alternative auch zu überlegen, ggf. ein Mobile Device Management-(MDM)-Werkzeug auf kommunalen Dienst-Handies zu nutzen, um die Installation einer eigenen kommunalen mobilen Messenger-App zu befördern. Darüber hinaus sollte eine Organisation über eine Mobile-Security-Strategie verfügen (vgl. Blackberry-Studie 2017), mit der die Nutzung eines konformen eigenen Messengers befördert bzw. die Nutzung von Whatsapp über MDM ausgeschlossen werden kann.

Für die kommunalen schulischen Institutionen haben zahlreiche Länder der Nutzung von WhatsApp gleich flächendeckend rechtlich verboten, wie z.B. Rheinland-Pfalz, Baden- Württemberg oder Niedersachesen und weitere (vgl. z.B. Landesbeauftragte 2017, SVZ 2017).

Rechtlich gesehen haben Geschäftsführer, IT-Leiter, Schulleiter und auch Beschäftigte und insbesondere Personalräte ein hohes Interesse aufgrund der vorgenannten Risiken an der Sperrung von WhatApp auf dienstlichen Geräten oder an Alternativen für dienstliche Zwecke eingesezte private Geräte. Eine Altnertive muss also her, die es jedem Amt und jeder Schule erlaubt, einen eigenen Kommunikationsserver zu betreiben.

Dieses will das Projekt momedo befördern und entwickeln auf Basis einer Umgebungs- und Landschaftsanalyse bestehender quelloffener (1), mobiler (2) Lösungen mit Verschlüsselung (3) und der Entscheidung für eine Code-Basis, mit der eine solche Alternative weiterentwickelt und den Interessierten quelloffen, kostenfrei und einfach in der Anwendung bereitgestellt werden kann.

Durch eine konzertierte Aktion von Android/iOS Entwicklern, Beförderern und beteiligten kommunalen Implementierern kann eine solche Lösung auf die Beine gestellt und angewandt werden.

2 Zielsetzung, Fachbereichsbefragung sowie Anforderungen und Spezifikationen

Die Zielsetzung dieses momedo Projektes besteht u.a. in

(1) der Analyse der Anforderungen von Fachbereichen kommunaler Stadtverwaltungen und schulischer Institutionen hinsichtlich einer mobilen Messenger Applikation,

(2) in der Verdeutlichung der Notwendigkeit, eigene Server und quelloffene Klienten mit Integration von Verschlüsselung zu nutzen, um bedarfsgerechte Arbeitsschritte im Zuge der Digitalisierung in die Applikation einbauen zu können sowie

(3) der Darstellung eines Überblicks der Landschaft solcher bisherigen mobilen quelloffenen und verschlüsselnden Applikationen mit Fokussierung auf die Empfehlung der Nutzung und Weiterentwicklung einer quelloffenen Lösung mit Fundierung von Entscheidungsempfehlungen

(4) und Start eines Pilotprojektes, um eine solche App für Android und IOS ggf. zu entwickeln und anzubieten. Hierzu sind weitere Interessierte, Community-Beteiligte und kommunale Befähigte und Implementierer sowie Programmierer herzlich eingeladen, diesem Projekt beizutreten.

Methodisch soll dabei zunächst auch eine Befragung von einzelnen kommunalen Fachbereichen sowie zweier Schulen einfließen, um darauf aufbauend eine Projekt- und Pilotierungsempfehlung abschätzen zu können. Auf Basis eines strukturierten Gesprächsleitfadens wurde eine handvoll Ansprechpartner dazu befragt.

Auf dieser Basis sollen hier nun ein paar Beispiele für die Nutzung und Anforderungen an eine Mobile App hier exemplarisch dargestellt werden und diese sollen anderen kommunalen Entscheidern und Befähigten für eine Diskussion und z.B. Mitgestaltung eines solchen Pilotprojektes dargelegt werden.

Im Falle einer positiven Entscheidung zur Mitwirkung weiterer kommunaler Handlungsträger an diesem momedo Rahmenverbund-Projektes kann die Erstellung einer quelloffenen WhatsApp-Alternative als Modellversuch im Rahmen des weiter unten spezifizierten Pilotprojektes angesehen werden, das nach einer gewissen Testphase kommunal evaluiert werden könnte, so dass sich weitere Empfehlungen für nächste Schritte mit den Handelnden aus der kommunalen und erweiterten Community gemeinsam abstimmen lassen.

Ein erster Schritt ist mit der Ermittlung der Anforderungen und Spezifikationen also getan, um darauf aufbauend nach derzeitiger Lage entsprechende quell-offene mobile Applikationen mit Verschlüsselung, eigenem Serverbetrieb und der Option der Anbindung von bedarfsgerechten Arbeitsprozess-Schritten als Empfehlung prüfen und benennen zu können.

Der weitere Schritt wäre sodann die Kompilierung und ggf. angepasste Entwicklung der mobilen Applikation zunächst für Android und auch sodann Apples iOS.

Im Folgenden werden Ergebnisse und Empfehlungen aus (anonymisierten) Gesprächen mit kommunalen Verantwortlichen hinsichtlich der Einschätzungen zu einer quelloffenen WhatsApp-Alternative zusammengefasst und für technische und prozessualen Anforderungen und Spezifikationen aus kommunaler Sicht aufbereitet und um die technischen Anforderungen aus IT-Sicht ergänzt.

Modellhafte und lediglich kurz skizzierte Fallbeispiele verdeutlichen die Notwendigkeit, auf eine flexible und eigenprogrammierte, quell-offene Code-Basis zu setzen, um Anpassungen an kommunale Geschäftsprozesse vorzunehmen.

Folgende Beispiele für kommunale Anwendungsfälle sowie Anforderungen und Spezifikationen lassen sich finden:

Beispiele für praktische Anforderungen und Spezifikationen:

```
Beispiel 1: Ein Schule möchte im Klassenverband einen Messenger einsetzen,
um alle Schüler über den Ausfall einer Unterrichtsstunde des frühen Morgens zu informieren.
```

```
Beispiel 2: Ein Hausmeister soll darüber informiert werden, dass an einem kommunalen
Gebäude eine Glühbirne ausgetauscht werden muss (vgl. Beispiel oben).
Die Freigabe zur Reparatur erfolgt aus den Prozessen der Kommune nach Bewilligung.
```

```
Beispiel 3: Die Bilder einer Ordnungswidrigkeit sollen mit Standortangaben
an das Ordnungsamt gesandt werden, die diese dann direkt aus dem Messenger
mit einem Upload nach SAP versehen können, ohne diese manuell zu bearbeiten
oder zu transferieren.
```

Aus den Rückmeldungen in Gesprächen mit kommunalen Beteiligten lassen sich folgende Anforderungen und Spezifikationen festhalten:

1) Mobile App zur Kommunikation, die Seitens einer Kommunalverwaltung zur Verfügung gestellt wird, die WhatsApp ersetzt und die Basis-Funktion der Nachrichten-Übermittlung ermöglicht 2) Möglichkeit der Übersendung von Bildern über den Messenger, um z.B. ein Schadensfoto an den Bearbeiter des Reparaturauftrages zu übersenden 3) Einfache Übermittlung von Text-Informationen über den Messenger an einen Kollegen 4) Einrichtung eines Gruppen-Channels, um alle Kollegen zu erreichen 5) Anbindung des mobilen Messengers an kommunale Software, um manuelle Übertragungen und Arbeitsschritte zu erleichtern.

Technische Anforderungen / Hinzu kommen aus Sicht von IT verschiedene technische Aspekte, die es zu erfüllen gilt

A) einfach zu administrierender lokaler Server „on Premise“, um die Kommunikation und Verschlüsselungsoptionen nicht auszulagern und in eigenen Händen zu behalten. Sollten die Ergebnisse des Pilotprojektes als anwendbar und erfolgreich gelten, könnten auch weitere Zielgruppen wie Schulen oder andere Fachbereiche und Kommunen von dieser im Rahmen dieses Projektes erarbeiteten Lösung profitieren, so dass ein eigener Server kostengünstiger ist als eine proprietäre und kosten- sowie lizenzpflichtige Lösung mit Kommunikationsserver in fremder Hand.

B) Die Server-Software sollte ebenso wie die Klienten-Software quell-offen sein und für verschiedene Betriebssysteme wie die drei POSIX-Systeme Windows, Linux und Mac vorliegen, idealerweise ggf. auch für das Betriebssystem Android.

C) Der Klient sollte keine DHT (Distributed Hash Table) Lösung umfassen, sondern auf eine Server-Klient-Lösung setzen, da mobile Netzwerke bzw. deren Sicherheitsarchitektur sogenannte „Servents“ im DHT Netzwerk oftmals nicht zulassen.

D) Um die Lösung später auch in Hand z.B. von Informatik-Lehrern an kommunalen Schulen geben zu können, die dann eine Instanz für die Schule oder den Fachbereich aufsetzen wollen, ist nochmal die Einfachheit der Administration der Serversoftware zu betonen. Ggf. sind bei differierenden Angeboten Praxistests und Evaluationen von Einrichtern von Serversoftware zu erstellen. Die Einfachheit der Administration der Messenger-Serversoftware ist mittelfristig ein entscheidendes Erfolgskriterium für ein Roll-Out der Applikation nach einer ersten Pilotierungsphase.

E) Verschlüsselung: Grundsätzlich sollte der kommunale Messenger Verschlüsselung anbieten, dieses betrifft zugleich eine einfache Handhabung der Verschlüsselung im Hintergrund und nicht jede einzelne Nachricht sollte verschlüsselt werden, wie es derzeit bei Email der Fall wäre, sondern wenn einmal die Schlüssel getauscht wurden, sollte die Verschlüsselung kontinuierlich funktionieren und den Benutzer nicht weiter vor Aktionen stellen.

F) Schlüsselmanagement: Verschlüsselung erfordert, dass jeder Nutzer einen Schlüssel für die Verschlüsselung nutzt und den Schlüssel seines Kommunikationspartners kennt. Dazu ist ein Management der Schlüssel der Nutzer erforderlich. Dazu werden in der Regel Schlüssel-Server als zentrale Instanz genutzt oder die Nutzer tauschen beide ihre Schlüssel gegenseitig. Beide Optionen sollten die Software einfach und im Hintergrund anbieten können.

G) Das Messaging sollte zwischen mobiler Anwendungen wie auch zu einem Desktop Klienten möglich sein. Während Android Anwendungen in Java geschrieben werden und Desktop-Anwendungen meisten in C++, stellt sich die Notwendigkeit dar, dass das Verschlüsselungsmanagement zwischen Java und C++ mit unterschiedlichen Schlüssel-Formaten meist inoperabel ist. Wenige Lösungen, zu denen wir später noch Empfehlungen abgeben, haben einen Kommunikationsweg zwischen Mobiler Applikation und Desktop-Applikation gefunden.

H) Die Applikation sollte quelloffen sein, nur dann macht es Sinn, die weiteren Projektschritte damit zu definieren, denn Ziel muss es mittel- bis langfristig sein, je nach Anforderung der weiteren Digitalisierung von Arbeitsprozessen in den kommunalen Fachbereichen entsprechende Plugins und Schnittstellen in die Messenger Softwarte selbst entwickeln und einzubinden zu können. Vordringliches Ziel ist nicht, mit dem Messenger eine Nachricht von A nach B zu senden, sondern eine quelloffene Applikation zu haben, die es ermöglicht, kommunale Arbeitsprozesse mit Workflow-Schritten in den Messenger zu integrieren und so die Arbeitsprozesse selbst weiter ausgestalten zu können.

I) Die Code Basis sollte gut strukturiert und compilierbar bzw. anpassbar sein für beide mobile Betriebssysteme: Android und iOS.

J) Um den datenschutzrechtlichen Anforderungen gerecht zu werden, sollte der den Nutzer identifizierenden Identifier keine Telefonnummer sein wie es bei WhatsApp scharf kritisiert wird. Eine ID ist daher erforderlich, die idealerweise aufgrund der Verschlüsselung der Nachrichten nicht mit der regulären Emailadresse des Nutzers gewählt wird. Idealerweise wird eine kurze ID wie z.B. „49K492“ gebildet, um den Nutzer auf technischer Ebene zu indexieren und auch das Schlüsselmanagement darüber ablaufen zu lassen.

K) Identifier auf Basis der Emailadresse sollten weiterhin also ebenso vermieden werden, wie insbesondere auch die Einrichtung von Parallel-Emailadressen um keine zweite Mailadresse für einen Messenger zu haben, die ggf. noch durcheinandergebracht wird.

L) Die Nachrichten sollen auf verschiedenen Mobilgeräten ersichtlich werden, so dass eine Instanz auf dem Smartphone mit denselben Nachrichten ausgestattet wird wie eine Instanz auf einem Tablet. Dieses erfordert ebenso eine Analyse der gewählten Verschlüsselung und deren Einrichtung bzw. Export zu einem Zweigerät.

M) Abrufen von Nachrichten an Kollegen, die Offline sind (Queue oder Buffer im Klienten bzw. Server).

N) Gruppenchat sollte sowohl über symmetrische als auch asymmetrische Verschlüsselung möglich sein, um z.B. entsprechende kompromittierte Schlüssel begegnen zu können.

O) Klient und Server sollten keine Lizenz-Gebühren erfordern, sondern eine aktive Entwicklung bzw. Ausgereiftheit aufweisen.

P) Die Benutzeroberfläche des Messenger Klienten sollte für zumindest eines der beiden mobilen Betriebssysteme ausbaufähig vorliegen und ein Übertrag nach Analyse der Codebasis für das jeweils weitere Betriebssystem leicht möglich sein.

Q) Idealerweise Einsatz von verschiedenen und kompatiblen Algorithmen für die Verschlüsselung

R) Die Lizenz sollte quelloffen sein und ohne weitere Restriktionen, idealerweise BSD oder GPL.

S) Kernel-Programmteile oder API / SDK Programmteile sollten veränderbar und mit ebenso quelloffener Lizenz bestehen wie der Hauptteil der Software

T) Der Software-Code sollen frei sein von allzu großen Abhängigkeiten zu anderen Bibliotheken für Kernel, Verschlüsselung und anderen Bestandteilen der Software, um so auch eine Kompilier-Kompatibilität zu garantieren für die unterschiedlichen mobilen Betriebssysteme.

U) Die Serververfügbarkeit sollte überdurchschnittlich hoch sein, um den Betrieb im öffentlichen Dienst auch mit zusätzlicher Garantie zu versehen.

3 Landschafts- und Kontext-Analyse

Mobile Messenger Lösungen mit quelloffenem Messaging-Server und Messaging-Klienten, die eine Verschlüsselung einsetzen und einen leicht administrierbaren Server beinhalten, finden sich in der Software Landschaft nicht in Hülle und Fülle, jedoch in arbeitsfähiger und ausreichender Grundlage.

Proprietäre Lösungen mit Lizenzgebühren (wie Telegram, Whatsapp und andere) scheiden nicht nur aus Kostengründen, sondern auch aus Gründen der fehlenden Anpassbarkeit an die lokalen Arbeitsprozesse der Ämter und Schulen aus, da sie keine spezifische API-Anpassung oder eigene Kompilierungen ermöglichen. Um Anforderungen der Digitalisierung von Arbeitsprozessen in den Fachbereichen gerecht zu werden, muss auf eine Open Source Lösung gesetzt werden, die es ermöglicht, durch Plugins und Schnittstellen weitere Funktionen und Arbeitshilfen in die Software selbst einzubauen.

Damit ist die Entscheidung “Make statt Buy” zu favorisieren und es soll im Folgenden daher auf quelloffene Lösungen fokusiert werden.

Auch Lösungen, die auf einen DHT (Distributed Hasch Table) setzen (wie Messenger Antox oder Messenger Briar und andere) statt auf eine klassische Klient-Server-Architektur, sind im mobilen Bereich nur schwierig eizusetzen, da nicht jede Mobile Netzwerkkommunikation einen sogenannten „Servent“ - also sowohl Klient und Server - auf dem Mobiltelefon ermöglicht und zugleich auch die Serveraktivität auf mobilen Geräten energieintensiver sein kann.

XMPP Messaging Anwendungen und Server sind entwickelt worden für unverschlüsseltes Messaging und spezifiziert vorwiegend für den Desktop. Die Versuche, Verschlüsselung nachzurüsten sowohl in Klienten wie auch in deren Servern, die nicht ohne entsprechende Patches und Sicherheitsmodule dafür ausgerüstet sind, und das Verschlüsselungsverfahren derzeit von OTR auf Double-Rachet umwandeln, ist äußerst fragmentiert und wenig technisch wie flächendeckend ausgeprägt und setzt auf eine nicht native Verschlüsselung hinsichtlich Einfachheit in der Anwendung, Schlüsselstärken und Schlüsselmanagement. Auch die Pilotversuche bei mobilen Klienten (wie das experimentelle Chatsecure (nur für IOS) und andere eher experimentelle Projekte) sollen daher nicht weiter einbezogen werden, da sie von den vorgenannten Voraussetzung doch sehr abhängig und geprägt sind. Schlichtweg ist die Serveradministration und deren Verschlüsselungspatches auch nicht sehr praktikabel für nicht fortgeschrittenere IT-Experten. Nicht jeder Informatiklehrer wird sich mit der Einrichtung eines prosody oder ejabberd Servers auseinandersetzen, um dort die entsprechenden Verschlüsselunspatches einzuspielen. Da sind die im folgenden dargestellten verschlüsselnden Kommunikationsserver wesentlich einfacher zu bedienen und bieten sich als Solution-Erfahrung an.

Wie bereits herausgestellt sollten auch für den Erfolg des Projektes Applikationen und Architekturen außen vor bleiben, die zwar eine quelloffene Messenger-Applikation anbieten (wie Telegram, s.o. und andere) aber keinen quelloffenen Messaging-Server vorhalten – wie auch Applikationen, die für den Server nur eine API anbieten oder deren Setup und Replikation bislang auch in Fachkreisen noch nicht zielführend gelungen ist (wie z.B. die Signal Server Software und andere).

Nachdem Verschlüsselungsexperte Bruce Schneier (2016) eine umfassende Liste sämtlicher seinerzeit in der Welt existierender Verschlüsselungswerkzeuge erstellt hat, lässt sich festhalten, dass unter den oben getroffenen Annahmen insbesondere zwei Applikationen übrig bleiben, die als mobile Messenger diese Voraussetzungen ideal zu erfüllen scheinen.

Es handelt ich hierbei um den Messenger Smoke Chat und einmal um den Messenger Delta Chat.

Beide Applikationen sind junge und inzwischen gereifte Projekte insofern durchaus etabliert in der Softwarelandschaft und bieten mit ihrem quelloffenen Code eine ausführliche Basis, um eine kommunalspezifische Anwendung daraus mit geringen Anpassungen (z.B. Branding) in der Benutzeroberfläche erstellen zu können. Insbesondere die Codebasis für die

Serversoftware gilt es zu betrachten.

Im Folgenden sollen daher beide Applikationen und Architekturen zunächst kurz vorgestellt werden, um in einem zweiten Schritt beide Applikationen auf kommunale und oben genannte Anforderungen und Spezifikationen zu beziehen.

Smoke Chat

```
Smoke Chat:
Mobile, quelloffene und verschlüsselnde Chat Applikation für Android
Repositorium: https://github.com/textbrowser/smoke
Projektseite: https://textbrowser.github.io/smoke/
Manual: https://github.com/textbrowser/smoke/raw/master/Documentation/Smoke.pdf
Serversoftware ist die einfach zu administrierende Software SmokeStack.
Ebenso bestehen weitere Serverprojekte, so dass die Serversoftware in Java, C++
bzw. auch für Android Betriebssystem vorliegt. Dieses ermöglicht die größtmögliche
Flexibilität in der Administration des Messenger Servers.
Die Server Software ist sehr einfach zu administrieren.
Nachteil: Die Benutzeroberfläche müsste noch etwas aufgehübscht werden
und zu iOS portiert werden.
```

Delta Chat

```
Delta Chat:
Mobile, quelloffene und verschlüsselnde Chat Applikation für Android
Repositorium: https://github.com/deltachat/deltachat-android
Projektseite: https://github.com/deltachat/deltachat-core
Die Serversoftware basiert auf einem IMAP Server. Diese Architektur erfordert
spezifische Kenntnisse in der Einrichtung eines solchen Servers und die Ausstattung
der Nutzer mit einer zweiten E-Mailadresse, die nur für das Messaging genutzt wird.
Nachteil: IOS Portierung wäre ebenso erforderlich und IMAP ist nicht ganz zeitgerecht
in der Übermittlung und externe IMAP Mailadressen werden das Verschlüsselungsprotokoll
ggf. unterbinden, so dass ein Wechsel zu einem eigenen Server und damit die Erzeugung von
Paralleladressen notwendig wird (die ein neuer Nutzer ggf. nicht auseinanderhalten kann,
was Messenger und was E-Mailadresse ist und an welche dann keine Plaintext-Nachricht zu senden ist).
```

3.1 Vergleich Smoke und Delta

Sowohl Smoke Chat wie auch Delta Chat sind probate Grundlagen eine Art kommunales quelloffenes WhatsApp mit wenig Aufwand und leichten Anpassungen in der Benutzeroberfläche zu erstellen.

Anhand der aufgestellten Kriterien sollen beide Applikationen und Server nun eingeschätzt werden, denn es handelt sich bei beiden um unterschiedliche Architekturen, die nicht nur technisch zu bewerten sind, sondern auch Einfluss auf die Handhabung und Nutzung haben werden und damit die kommunalen Arbeitsprozesse beeinflussen werden.

KRITERIUM	SMOKE CHAT	DELTA CHAT
TECHNISCHE SPEZIFIKATIONEN		
Einfache Administration Server	Sehr einfache Serveradministration, z.B. mit Serversoftware SmokeStack oder	Nutzung von IMAP Servern. Damit werden Email-Server für Chat umfunktioniert, d.h. es gibt auch ggf.

	Spot-on. Serversoftware liegt für zahlreiche Betriebssysteme vor wie auch in Java und C++. Die Serveradministration erfolgt sehr einfach mit wenigen Klicks über die Einrichtung eines HTTPS Listeners.	Verzögerungen in der Zustellung. Es muss ausgeschlossen werden, dass die Nutzer andere IMAP Server als den für die jeweilige Kommune bzw. Schule und dieses Programm dediziert nutzen. Dieses wäre faktisch unmöglich, und E-MailAdressen könnten verwechselt werden. Die IMAP Serveradministration ist nur etwas für Experten.
Quelloffener Servercode	Ja, BSD license. Kostenfreie Serversoftware in der Architektur vorhanden.	Lizenz in Hand Dritter, den Anbietern des IMAP Servers. Lizenzkosten für das Aufsetzen ggf. eines Exchange Servers. Es entstehen Kosten. Open-Xchange, Scalix, Zimbra und Zarafa als Exchange-Ersatz nicht erprobt im einfachen Einsatz und im Zusammenspiel mit Delta.
Ohne DHT	Ohne DHT	Ohne DHT
Roll-out Leichtigkeit Server	Sehr einfach, auch in Fachbereichen ohne IT-Support.	Experten erforderlich für Einrichtung eines IMAP-Servers.
Verschlüsselung	PKI (Public Private Key Infrastructure) basierend auf den Algorithmen RSA und McEliece, die untereinander kompatibel sind.	GPG mit nur einem Algorithmus im Angebot.
Schlüsselmanagement	SmokeStack Server sind zugleich auch Schlüssel-Server und ermöglichen einfaches Schlüssel-Management. Sowie: Nutzer können Schlüssel auch von Klient zu Klient übertragen.	Nutzer übertragen Schlüssel von Klient zu Klient durch einen automatischen Schlüsseltausch. Programmierung eines Schlüsselservers wäre ggf. erforderlich.
Mobile zu Dekstop Kommunikation	Messages können von der mobilen App auch zur Desktop App übertragen werden – trotz unterschiedlicher Schlüsselformate für Java und C++. Einfache Möglichkeit von mobiler Anwendung an einen Desktop Klienten zu chatten. Es bestehen dedizierte Desktop Anwendungen, die mit Smoke Messaging Nachrichten austauschen können.	Durch den Einsatz von GPG gibt es derzeit keine Messaging-Desktop Anwendungen für den Chat, ein Import der verschlüsselten Nachricht in einen E-Mail-Klienten ermöglicht keine automatisierte Entschlüsselung der Nachricht. Hoher Aufwand zur Entschlüsselung auf Desktop-Klienten, die es dediziert für Delta nicht gibt.
Quelloffen, um fachbereichsspezifische Funktionen einbauen zu können	Quelloffen für Klient und Server. BSD Lizenz.	Quelloffen für Klient. Server über IMAP, meist Exhange und closed source. Klient hat zusötzlich eine Core-Lib, beide GPLv3.
Code Basis gut und strukturiert	Ja. Gilt für Klient und Server.	Ja für Klient, jedoch mit Core Kernel.
Betriebssysteme IOS und Android	Derzeit verfügbar für Android (Java). Es besteht auch c++ Code (aufgrund des Desktop Clients), der bei IOS als Grundlage für Objective C genutzt werden kann. Portierung der App zu IOS also erforderlich.	Derzeit verfügbar für Android, IOS in Planung. Keine Desktop-Anwendung in C++ vorhanden. Portierung der App daher als Support erforderlich.
Nutzer-Identifier	Nutzer werden über eine Kurze zufällig Kennung wie „82K3JB4“ definiert und erkannt. Über diese Kennung funktioniert auch einfach und im Hintergrund der Schlüsseltausch für die Nutzer. Eine elegante Art des Schlüsselaustausches ohne großen Aufwand für den Nutzer.	Nutzer werden über eine E-Mailadresse definiert. Es ist eine Emailadresse des dedizierten IMAP Servers erforderlich und es ist sicherzustellen, dass Nutzer als Mitarbeiter einer Kommune oder Schule keine anderen IMAP Server anwenden. Es entstehen sonst für jeden Nutzer zwei E-Mailadressen, eine für Email, eine für den Messenger, die in der Adressierung von E-Mails nicht verwechselt werden sollten - auch aufgrund der Verschlüsselung.

Multiple Instanzen	Es ist möglich, die Message sowohl auf dem Tablet als auch auf dem Handy zu erhalten und zu entschlüsseln.	Die GPG Details der Verschlüsselung erfordern einen manuellen Import/Export der Verschlüsselungs-Credentials auf das zweite Gerät.
Nachricht an Offline-Kollegen	Möglich über verschiedene Wege (Abrufgen über Postfach („Ozone“ genannt) im Server oder Versand, wenn Kollege online ist	Storage im IMAP Server.
Gruppenchat	Vorhanden über verschiedene Möglichkeiten (symmetrisch wie asymmetrisch). Sofortige Verfügbarkeit wie im IRC oder einer WhatsApp Gruppe.	Gruppenchat wird über Emails an mehrere Personen erreicht, ein Real-time Chat ggf. mit Verzögerungen je nach IMAP Server Abruf der einzelnen Teilnehmer.
Gebühren	Frei von Lizenz-Gebühren	Ggf. Gebühren und Lizenzen sowie deren Verwaltung für IMAP Server erforderlich.
Benutzeroberflöche	Für Android, Bedarf der Aufhübschung ggf. mit eigenen Mitteln	Für Android, Benutzeroberfläche am Nutzer orientiert und etwas elaborierter.
Verfügbare Algorithmen	KI, RSA und McEliece	GPG (RSA)
Code-Lizenzen	BSD	GPLv3
Lib, Kernel, Core, Abhängigkeiten	Geringe Abhänigkeiten von Bibliotheken.	Abhängigkeiten von GPG code, Mail core der App
FACHBEREICHS SPEZIFIKATIONEN		
Ersatz für Whatsapp	Grundsätzlich Ja, nach Weiterentwicklung der GUI des Klienten	Ja im privaten Bereich, aber ggf. differierende Architektur.
Versand von Bildern	Derzeit in Implementierung für Inline-Versand.	Ja.
Übermittlung Text-Nachrichten	Ja.	Ja.
Erstellung von Gruppenchats	Ja, mehrere Methoden.	Ja, an mehrere Teilnehmer.

Aufgrund dieser Analyse und einer Einschätzung pro Zeile ergibt sich auch aus einer Gewichtung der Einzel-Kriterien aus der hier vorliegenden Sicht, dass Smoke die besseren technischen Ausstattungen, geringeren Kosten und umfangreichere Erweiterungsmöglichkeiten für kommunale Zwecke bietet, auch wenn für die Benutzeroberfläche ein Re-Design erforderlich ist. Die etwas elaboriertere Benutzeroberfläche von Delta rechtfertigt nicht die technischen Nachteile und vagen Implementierungsrisiken wie doppelte Emailadressen, Zeit-Delays bei Zustellung von Nachrichten, Notwendigkeit der Exklusion von anderen IMAP-Servern im kommunalen Geschäftsprozess oder der Notwendigkeit, einen Schlüssel-Server noch programmieren zu müssen, der in der Serversoftware SmokeStack des Servers von Smoke Chat bereits zu Verfügung steht. Die Entwicklung der Benutzeroberfläche von Smoke ist daher im notwendigen Branding der Applikation für eine Kommunalverwaltung in einem Schritt mit Leichtigkeit und wenigen entsprechenden Java-Programmierern zu erreichen. Die Portierung auf Apples IOS Betriebssystem ist bei beiden Varianten erforderlich.

Übersicht der Pro und Contra Indikationen für eine Entscheidung für die Code Basis von Smoke Messenger

	PRO	CONTRA
SMOKE Code	* Schlüssel-Server in der Code Basis vorhanden. * Technisch ausgereift mit vielen Optionen und umfangreichen Schnittstellen zu Erweiterungsfunktionen. * SmokeStack Serversoftwar einfach zu administrieren. * Serversoftware und Code Basis für Java und C++ vorhanden. * Serversoftware läuft auf allen POSIX Maschinen und auch unter Android Betriebssystem.	* Benutzeroberfläche benötigt leichte Anpassungen für Android. * Portierung nach Apples iOS als nächster Schritt erforderlich.

	* Einsatz auf multiplen Geräten möglich. * Code Basis vorhanden für den Chat von mobilen Geräten (Java) auch zu Desktop Klienten (C++). * Geringe Abhängigkeiten in der Codebasis.	
DELTA Code	*Ansprechende Benutzeroberfläche	* Portierung nach Apples iOS als nächster Schritt erforderlich. * IMAP Server könnte Gebühren verursachen. * Kein einfaches Server-Setup für weitere Zielgruppen wie Schulen. * Schlüssel-Server müsste extra programmiert werden. * IMAP Protokoll nicht ideal für Messaging ohne Delay-Wunsch. * Risiko der Doppelten Mailadressen und Nutzung privater Emailadressen (= selbe Konfliktsituation wie bei WhatsApp). * GPG erlaubt keine Chat Verbindung zu einem Desktopklienten mangels Code-Basis. * Zahlreiche Abhängigkeiten in der Codebass zu Bibliotheken und GPG Grundlagen, Core für den Mail-Kern der App.

Zusammenfassend soll daher Smoke Chat als Applikation zur Implementierung

> nach gewisser Anpassung in der Android Benutzeroberfläche
> und der Portierung nach Apple iOS zur weiteren Nutzung empfohlen werden. Neben der obigen Beurteilung der fachbereichsspezifischen und technischen Anforderungen und Spezifikationen lassen sich im folgenden Abschnitt weiterhin auch übergeifende strategische Bereiche benennen, die die Softwareauswahl insbesondere für die bereits vorgestellte Lösung unterstützen.

Die Entscheidung „Make oder Buy“ ist bei diesem Projekt der Erstellung einer quelloffenen kommunalen Alternative zu WhatsApp mit eigenem Server einfach zu treffen aus folgenden Gründen: Der Messenger-Markt bietet zwar proprietäre Lösungen, die eingekauft werden könnten, bieten jedoch dann nicht die Möglichkeit, entsprechende Erweiterungen selbst für die Bedürfnisse und Anforderungen der kommunalen Fachbereiche implementieren zu können. Ein Lizenz-Gebühren-Modell würde dauerhaft laufende Kosten erfordern, und ein Return in Invest wäre nicht möglich. Daher ist dieses Vorhaben in der Entscheidung „Make“ mit Bordmitteln insbesondere in der Programmierung zu gestalten.

Das Interesse an einer Messenger Applikation für kommunale Zwecke mit einfach administrierbarem Server ist sehr hoch, so dass konzertierte Aktionen mehrerer Kommunen oder Einzelner und Programmierern aus der Community entstehen könnten. Dieses rechtfertigt eine kurze Anpassungsphase in der Benutzeroberfläche wie auch in der Portierung zu Apple IOS.

4 Die Messenger App „Smoke“ bzw. deren Code-Basis als ein mögliches Pilotprojekt zum Ausbau

Die so evaluierte und zur weiteren Verwendung vorgeschlagene Code-Basis des Messenger Smoke Chat soll daher in einigen weiteren Bereichen etwas vertieft vorgestellt werden. Die Smoke Dokumentation gibt dazu auch viele weitere technische Details (Smoke 2017).

4.1 Kommunikation

Die Kommunikation des Smoke Messengers ist eine Klient-Server Architektur. Nutzer können sich mit ihrem mobilen Gerät über eine einfache HTTPS Verbindung zu einem spezifischen Kommunikationsserver verbinden und die Nachrichten sind unmittelbar auf dem Gerät der Gegenstelle sichtbar.

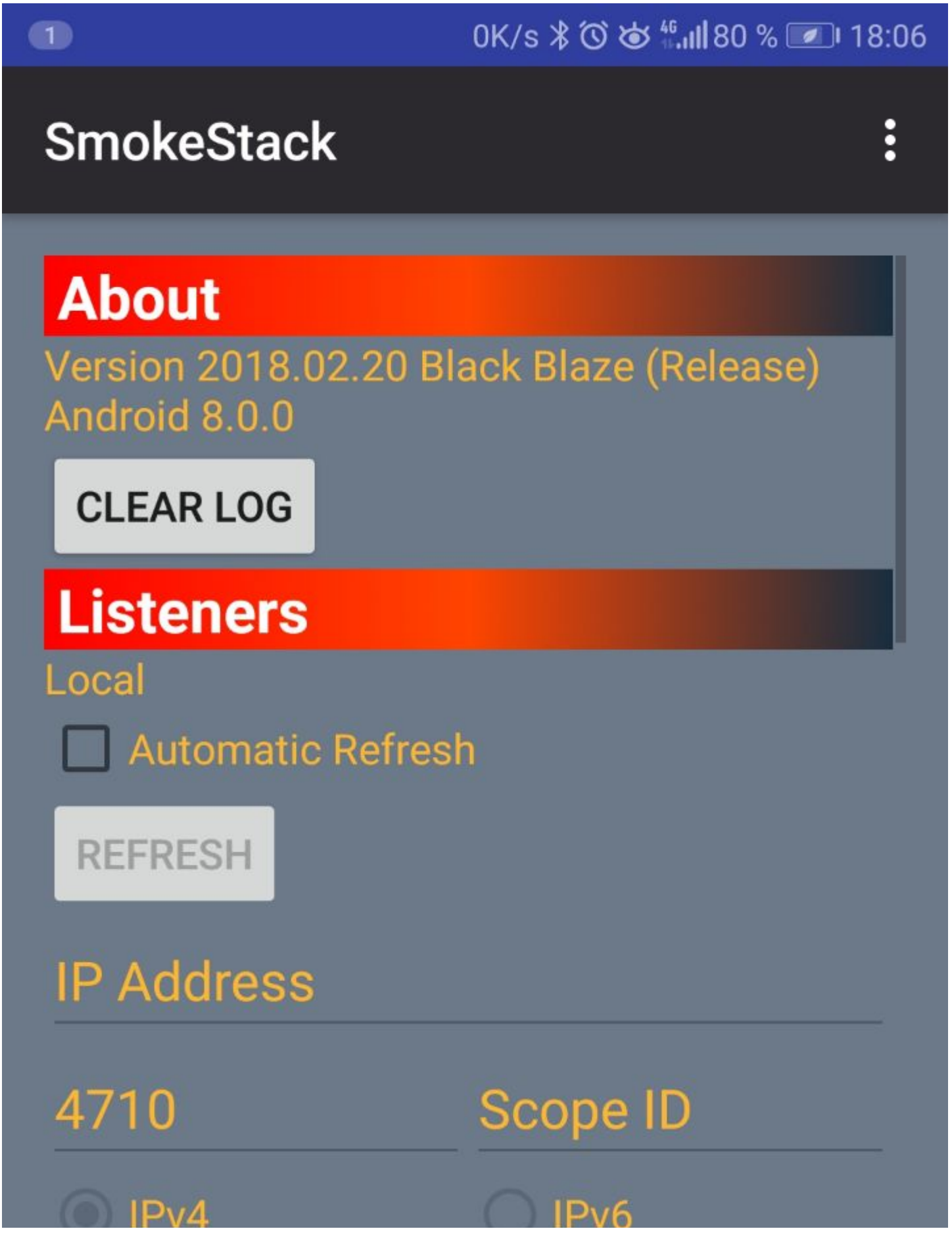
4.2 Leiche Administrierbarkeit des Servers

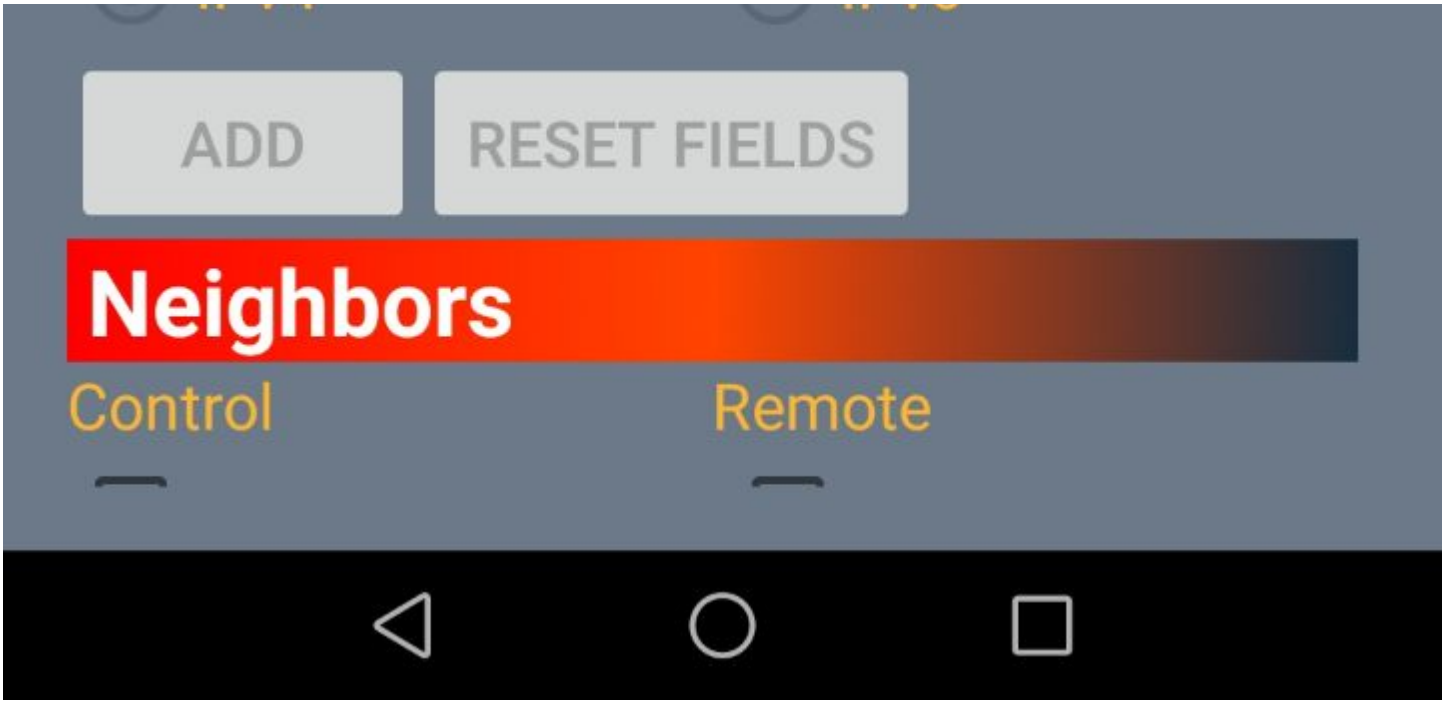
Der Server ist sehr leicht administrierbar. Es ist lediglich auf der IP der hostenden Maschine ein Listener an einem

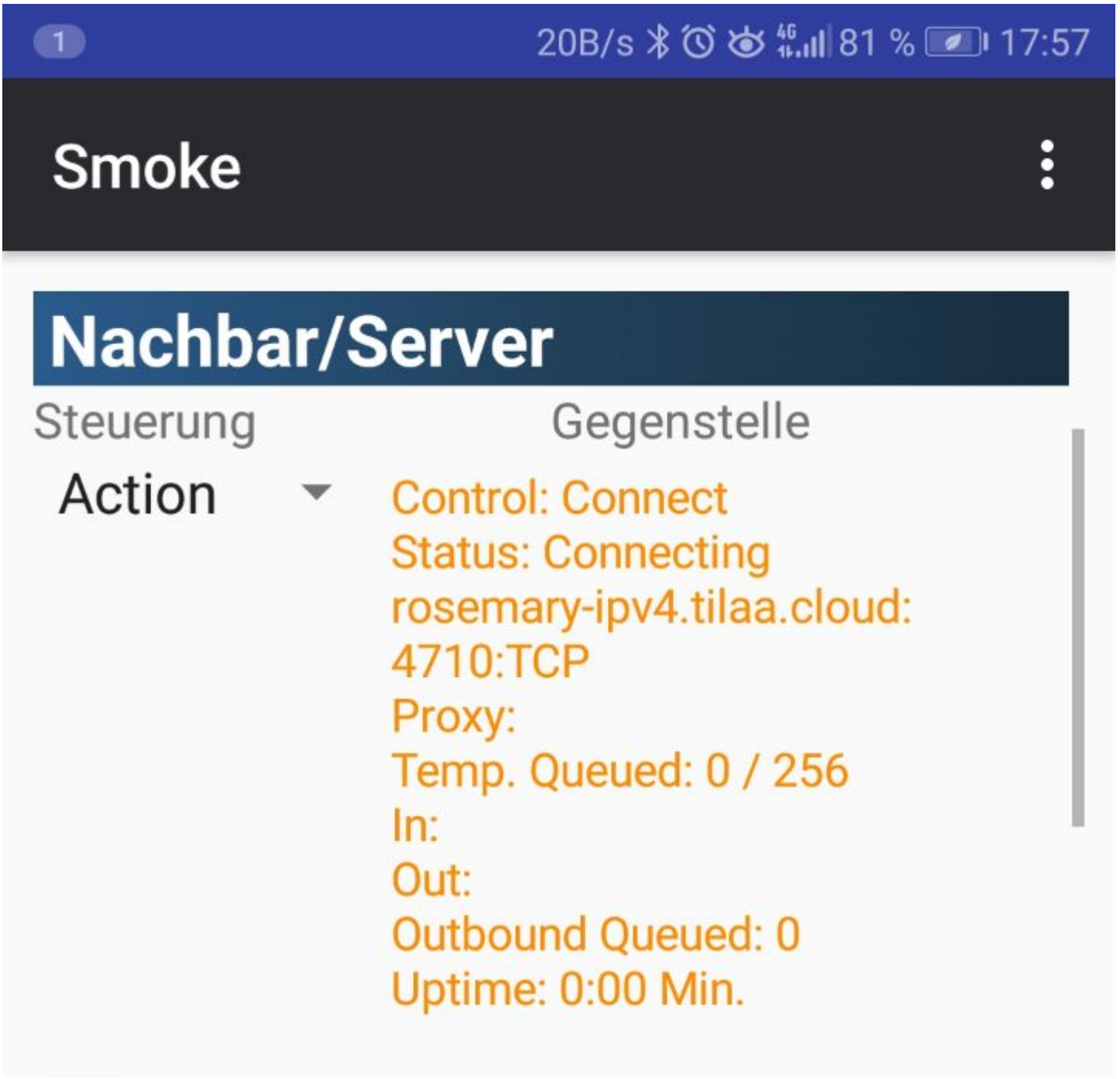
wählbaren Port zu öffnen und schon können zwei Nutzer darüber die Kommunikation laufen lassen.

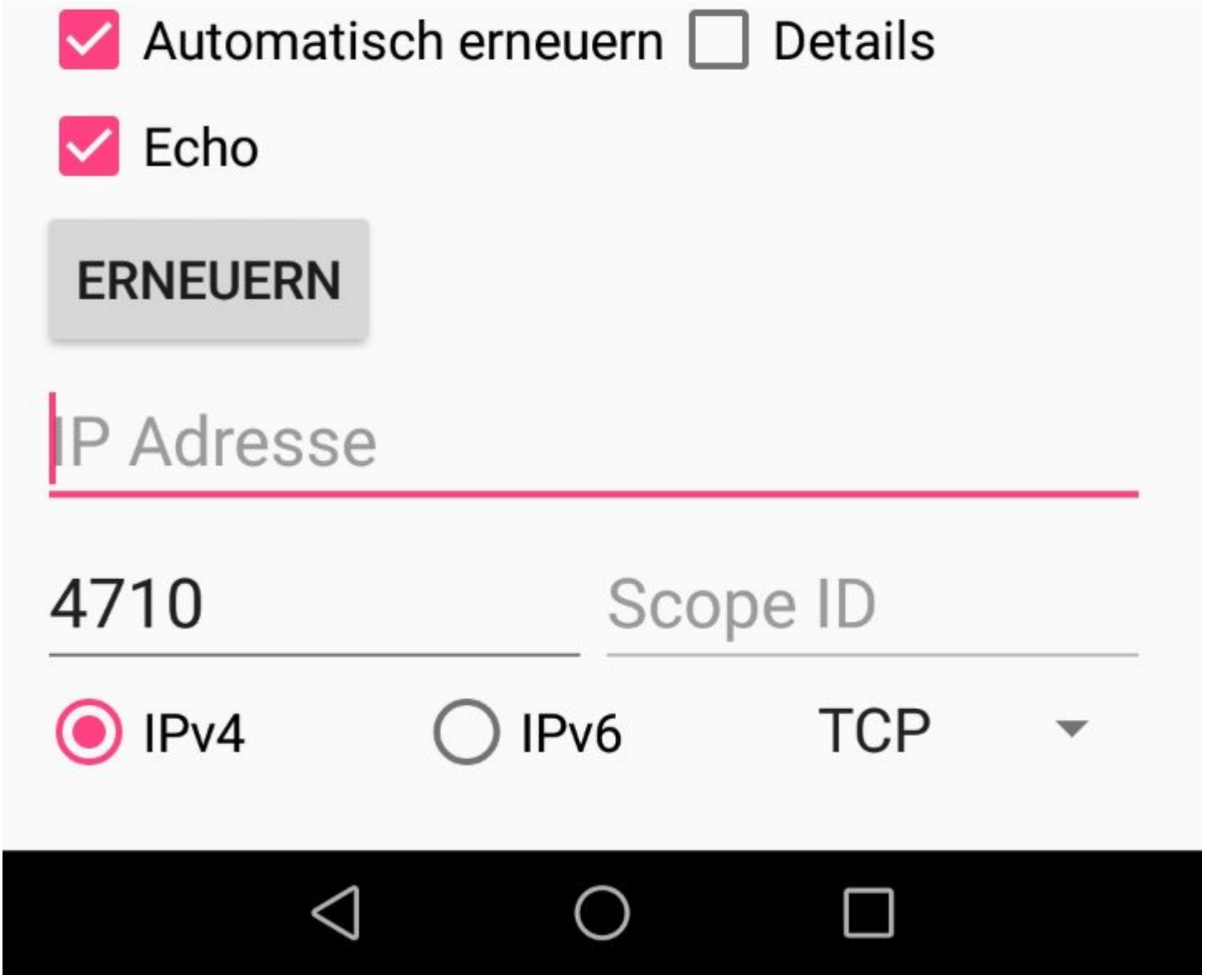
Die Einfachheit der Administrierbarkeit des Servers bietet sehr große Chancen auf eine Nutzung auch von anderen Zielgruppen, wie von Informatik-Lehrern, die für eine Klasse oder Schule einen kommunalen Messenger anbieten möchten.

*Screenshot der Serversoftware in Java




ADD
RESET FIELDS
Neighbors
Control
Remote


1
20B/s
81 %
17:57
Smoke
Nachbar/Server
Steuerung
Gegenstelle
Action
Control: Connect
Status: Connecting
rosemary-ipv4.tilaa.cloud:
4710:TCP
Proxy:
Temp. Queued: 0 / 256
In:
Out:
Outbound Queued: 0
Uptime: 0:00 Min.

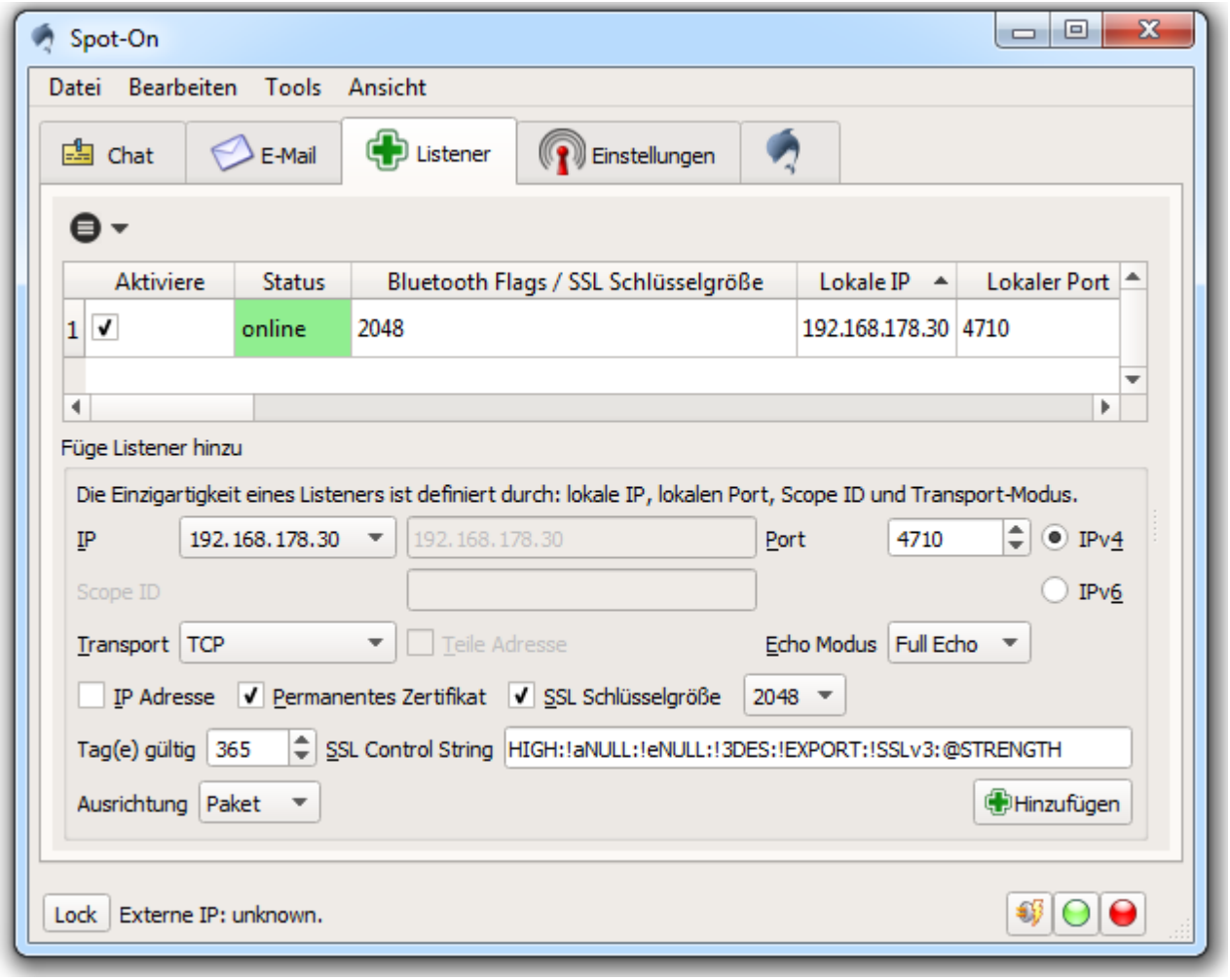


Die Abbildungen zeigen einmal die Android Java Anwendung für den Server SmokeStack und die Verbindung aufgebaut vom Klienten Smoke zum Listener der Servers.

Der Kommunikations-Server kann mit einem Button-Klick durch Bestimmung von Port und IP hinzugefügt werden.

Im Klienten Smoke ist die IP und Port des Servers direkt in der App eingebaut und verbindet sich direkt automatisch.

Screenshot der Serversoftware in C++



Neben Java besteht auch eine Code Basis für einen Kommunikationsserver in der Programmiersprache C++. Für den Fall, dass der Server unter einem der drei üblichen Betriebssysteme Windows, Linux oder Mac laufen soll oder auf einem Raspberry Pi installiert werden soll.

Die Serveradministration ist auch hier sehr einfach, mit wenigen Mausklicken lässt sich ein Listener einrichten.

Zusätzlich gibt es die Serversoftware auch als Lite Version für eine Linux Kommandozeilen-Administration.

Es sind keine weitergehenden oder spezifischen Kenntnisse erforderlich als mittels ein paar weniger Mausklicks einen Listener an einer IP mit Port freizugeben.

4.3 Einsatzfähigkeit der Serversoftware

Die Serversoftware läuft auf allen POSIX Maschinen und auch unter dem Betriebssystem Android.

D.h. die Serversoftware hat sowohl eine Code Basis für Java und als auch für C++ . Die Serversoftware für Android heißt SmokeStack und die Serversoftware für C++ heißt Spot-on (bzw. Spot-on lite). Damit ist es möglich, einen Kommunikationsserver in großer Bandbreite zu erstellen, sei es unter einen Windows Server, einer kleinen Android Kiste wie sie als Media Server in privaten Wohnzimmern oft genutzt wird, auch eine Linux Maschine eines Raspberry PI ist geeignet, um diese Serversoftware einzusetzen.

Dieses bietet auch erhebliches Explorations- und Forschungspotential z.B. bei technisch Interessierten oder auch bei Lehrern, für ihre kommunale Institution die Serversoftware einsetzen und einen kommunalen Messenger explorieren wollen oder als Lernobjekt für Schüler nutzen einbinden wollen.

4.4 Adressierung des Nutzers und Einsatz auf multiplen Geräten

Technisch ausgereift sind andere Alternativen nicht dahingehend, dass der Klient mit der gleichen Verschlüsselung auf multiplen, verschiedenen Geräten eingesetzt werden kann, beispielsweise auf einem Handy und zugleich auch auf einem Tablet. Bei der Code Basis des Smoke Messenger ist dieses problemlos möglich und spricht für die technische Elaboration.

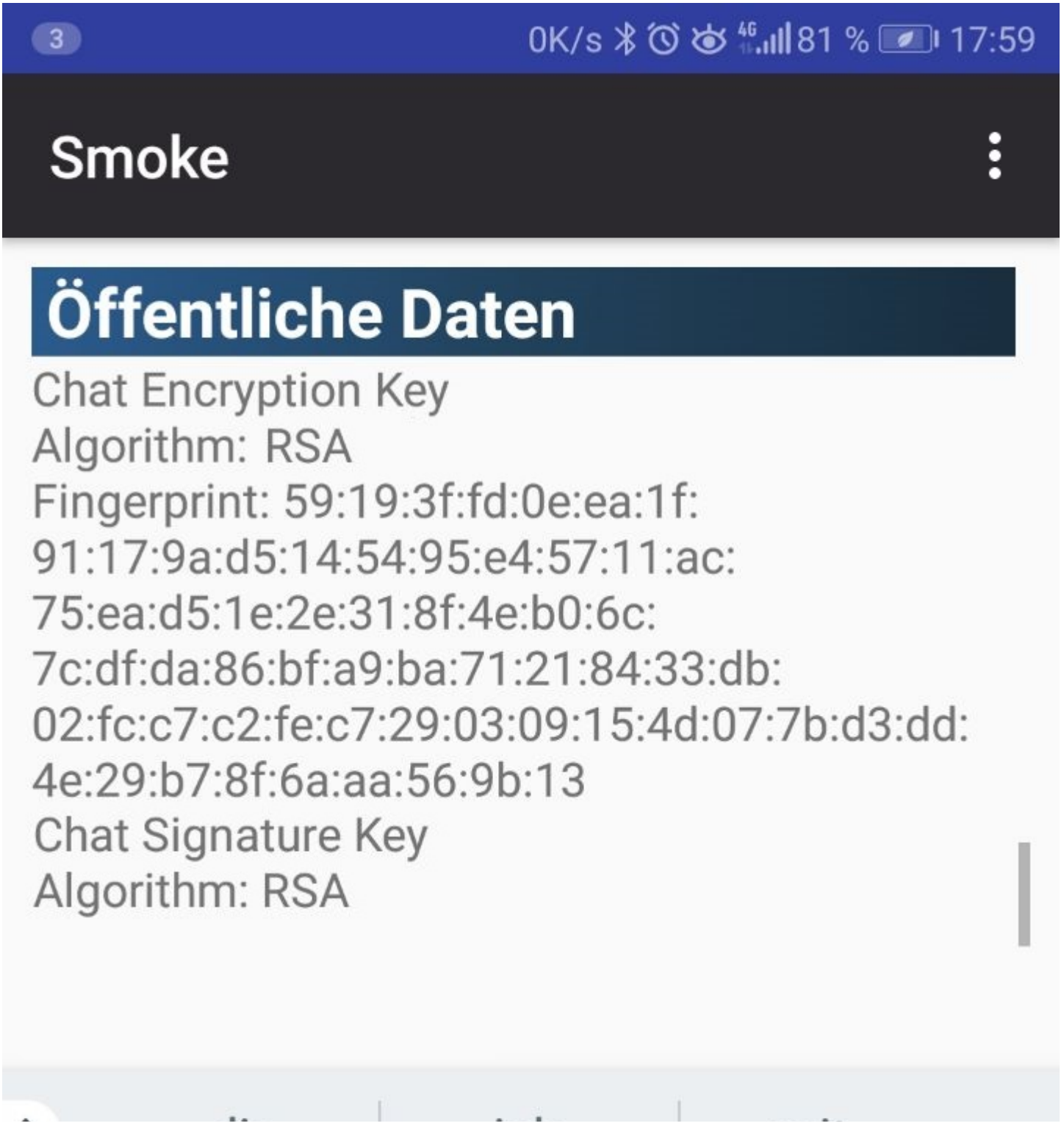
4.5 Verschlüsselung

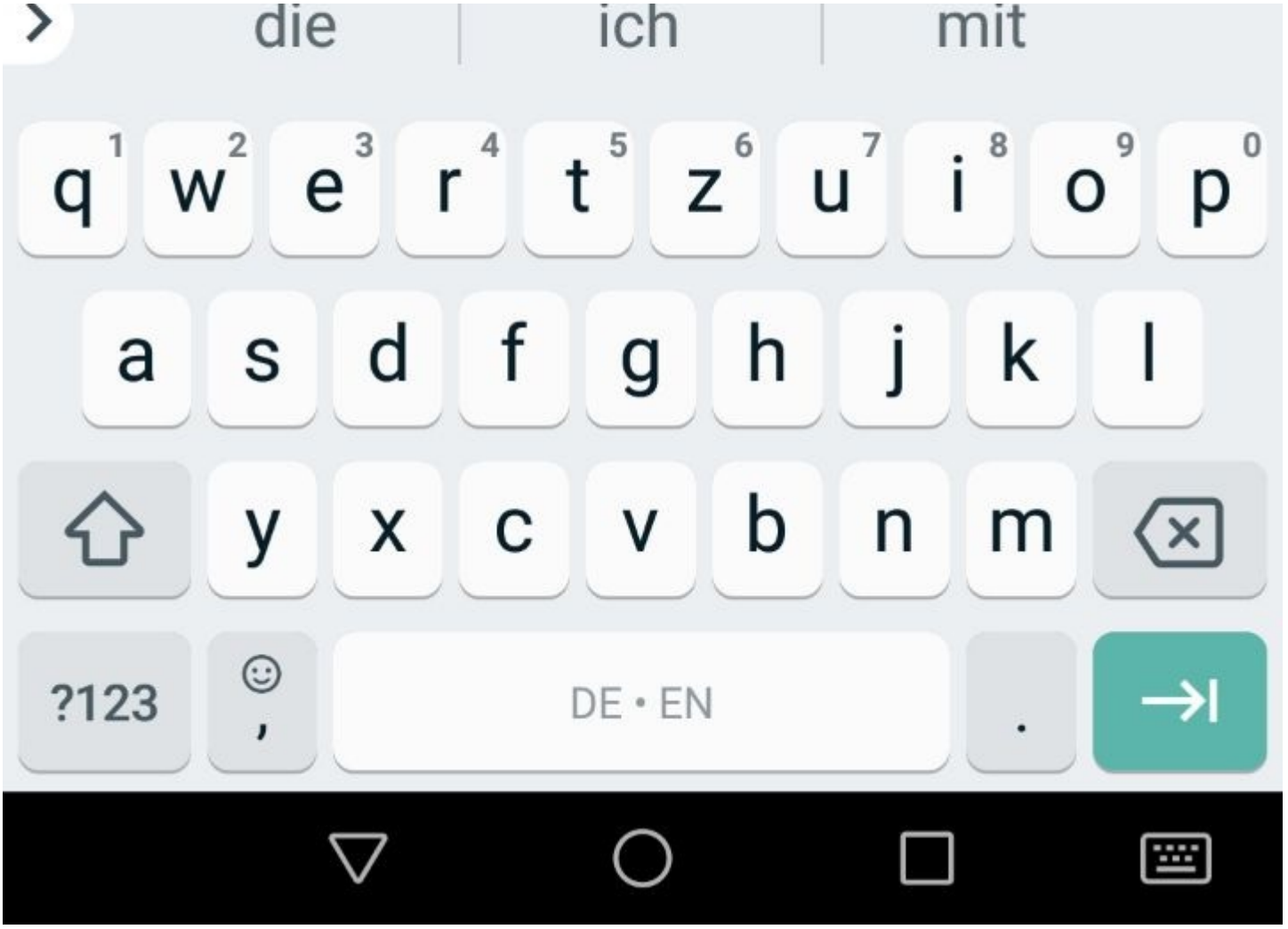
Die Verschlüsselung setzt in für den Nutzer wenig aufdringlichen Prozessen dennoch entsprechende Standards sein.

So kann zwischen einer symmetrischen wie auch asymmetrischen Verschlüsselung gewählt werden. Ebenso werden neben RSA auch der alternative und quantum computing Angriffen resistentere Algorithmus McEliece eingesetzt.

Das Verschlüsselungsverfahren setzt standardmäßig auf eine Infrastruktur aus öffentlichen und privaten Schlüsseln (PKI) basierend auf dem Algorithmus RSA.

Screenshot der Verschlüsselungsinformation Key und Signatur





Die Verschlüsselungsdaten werden ausgewiesen und hier der Fingerprint des öffentlichen Schlüssels dargestellt. Für den Nutzer kann diese zusätzliche Darstellung z.B. in angepassten Kompilierungen verborgen werden.

Nutzer haben dabei keine größeren Anforderungen, ihre Schlüssel zu tauschen, da dieses durch die Applikation automatisiert erfolgt über zwei Wege, die weiter unten beschrieben werden.

So besteht eine notwendige Verschlüsselung und zugleich ein minimaler Aufwand für den Nutzer, sich damit zu beschäftigen, da es im Hintergrund ablaufen kann.

4.6 Brückenschlag zwischen mobilem Klienten und Desktop Klienten

Die Code Basis ist vorhanden für den Chat von mobilen Geräten (Java) auch zu Desktop Klienten (C++), da beide Programmiersprachen unterschiedliche Schlüsselformate haben, ist bei anderen Applikationen normalerweise eine Kommunikation von Java-Klienten zu nativen C++ Klienten nicht möglich.

Die Code Basis ist auch hier insofern innovativ, als dass eine Messaging Möglichkeit gefunden wurde: - durch Verschlüsselung, die Programmierung sowie die Prozessroutinen, um einen Chat vom C++ Desktop-Klienten an die mobilen Java Geräte zu senden.

Dies funktioniert auch im Gruppenchat. So kann beispielsweise eine Führungskraft oder Lehrer auch vom C++ Desktop-Klienten die mobilen Mitarbeiter bzw Kollegen oder Schüler mit Java-Endgeräten vom Desktop adressieren.

Dieses Beispiel zeigt nur als ein Fall die Ausgereiftheit der Code Basis auf. Andere Applikationen bieten diese technische Ausgereiftheit nicht oder umgehen die Schlüssel-Format-Inkompatibilität über Webbrowser Zugriffe, die wiederum über (kryptographisch sehr unsichere) Javascript-Bibliotheken erreicht werden.

4.7 Schlüsselmanagement

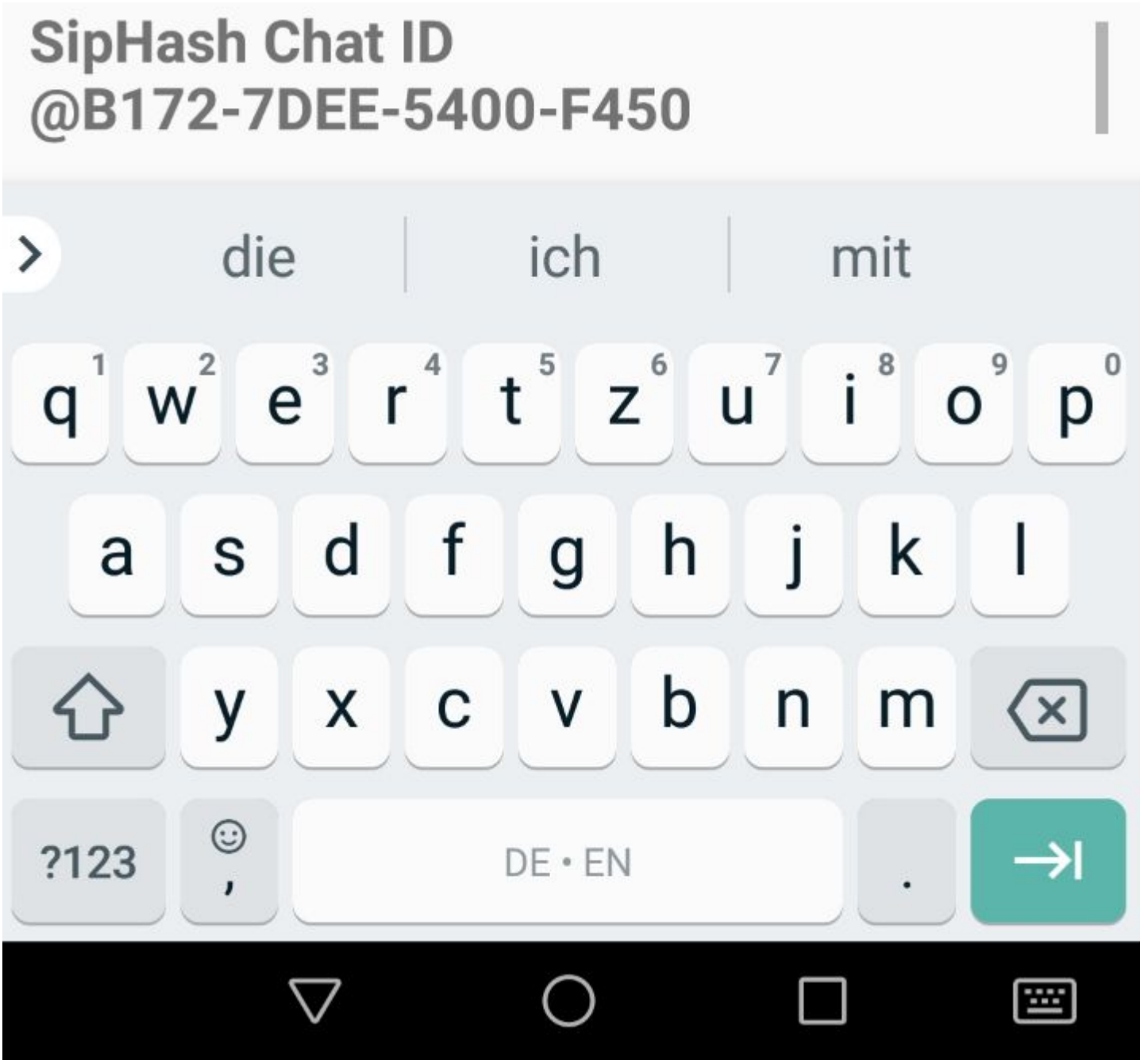
Auch ein ggf. notwendiger oder im Prozess helfender Schlüssel-Server ist in der Code Basis vorhanden. Er findet sich im SmokeStack-Server und dort können die Schlüssel von Kollegen ebenso deponiert und abgerufen werden.

Andere Lösungen erfordern ggf. eine eigene Programmierung und Anbindung eines Schlüsselservers.

Die Code Architektur bietet jedoch auch einen zweiten Weg, den Schlüssel zu tauschen in semi-automatisierter Form: Jeder Nutzer wird beim Smoke Messenger nicht über die Telefonnummer oder E-Mailadresse identifiziert, sondern über einen sogenannten SIP-Hash. Dieses ist eine kurze Folge von Ziffern und Zahlen wie „M8N22HT6“.

Und mit dieser Kennung wird ein verschlüsselter Kanal zwischen Kollegen eröffnet, über den die Applikation sodann automatisiert die öffentlichen RSA-Schlüssel sendet und somit mit der Gegenstelle tauscht.

Screenshot SIP HASH-ID

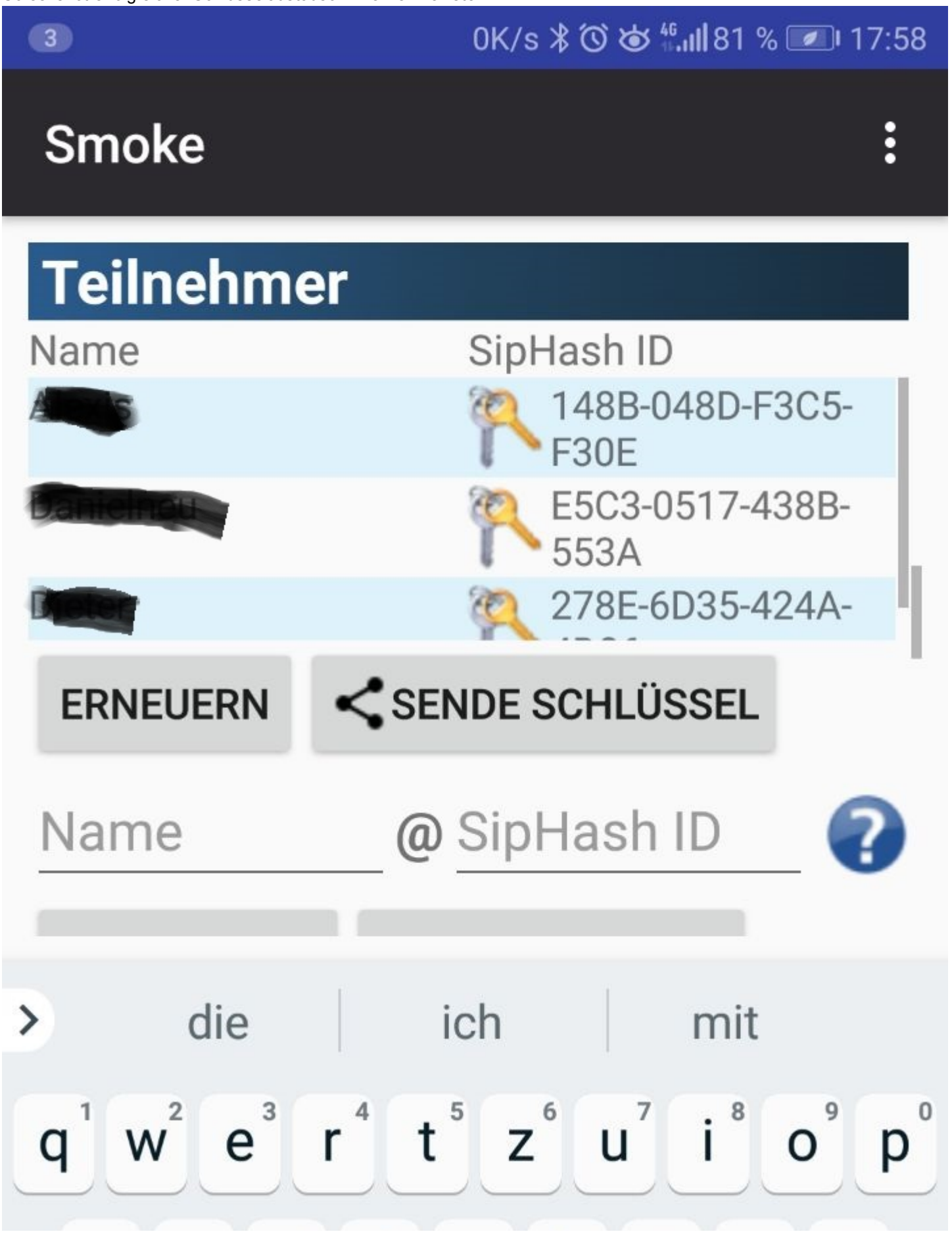


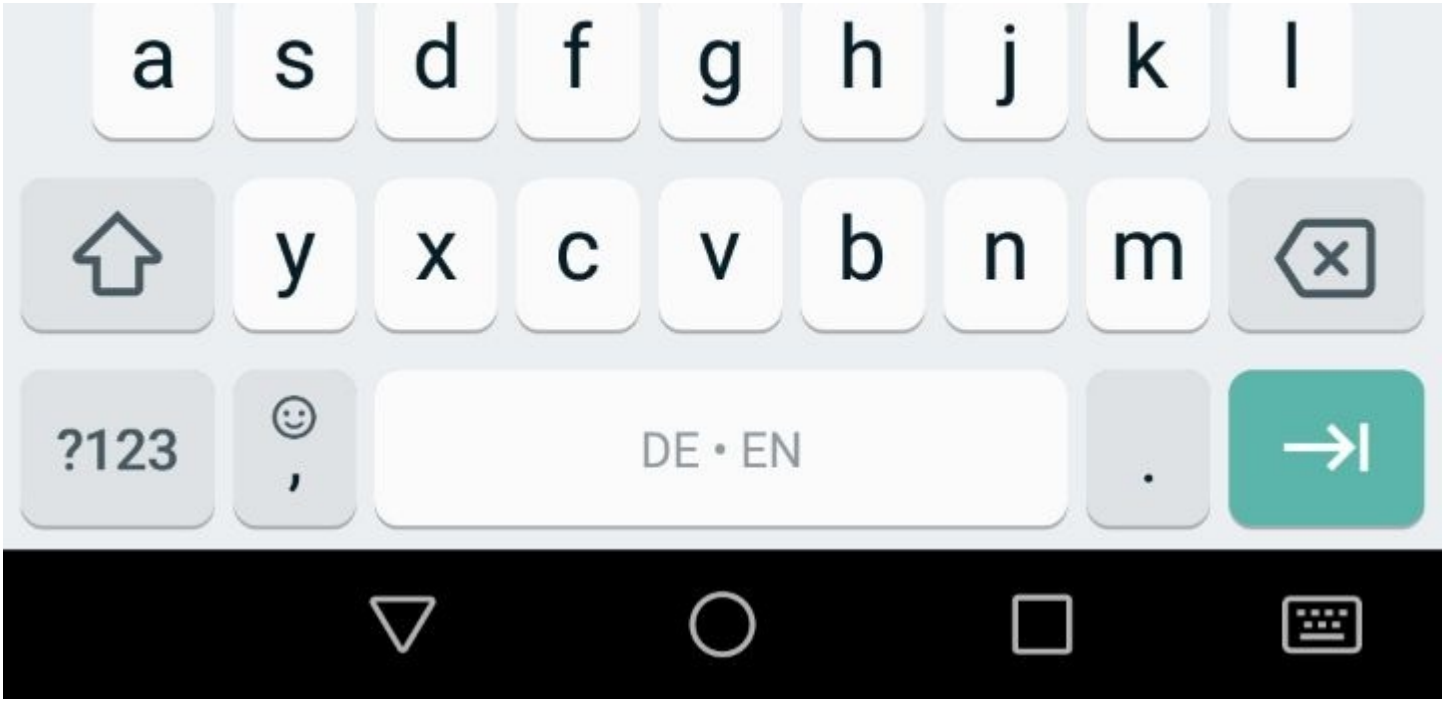
Der Identifier des Smoke Messengers ist eine SIP-HASH ID, ein kurzer String wie er von Lizenzschlüsseln bekannt ist. Darüber lässt sich jeder Nutzer identifizieren und auch das Schlüsselmanagement betreiben. Ergänzend ist in der Serversoftware auch noch ein Schlüsselmanagement integiert, das bei anderen Lösungen ggf. zusätzliche gebaut werden

müsste. Smoke hat mit dem Server SmokeStack einen Schlüssel-Server bereits integriert und stellt Schlüssel automatisch zur Verfügung.

Ein Symbol zeigt in der Teilnehmerliste den erfolgreichen Schlüsselaustausch der Teilnehmer an.

Screenshot erfolgreicher Schlüsselaustausch / Teilnehmerliste



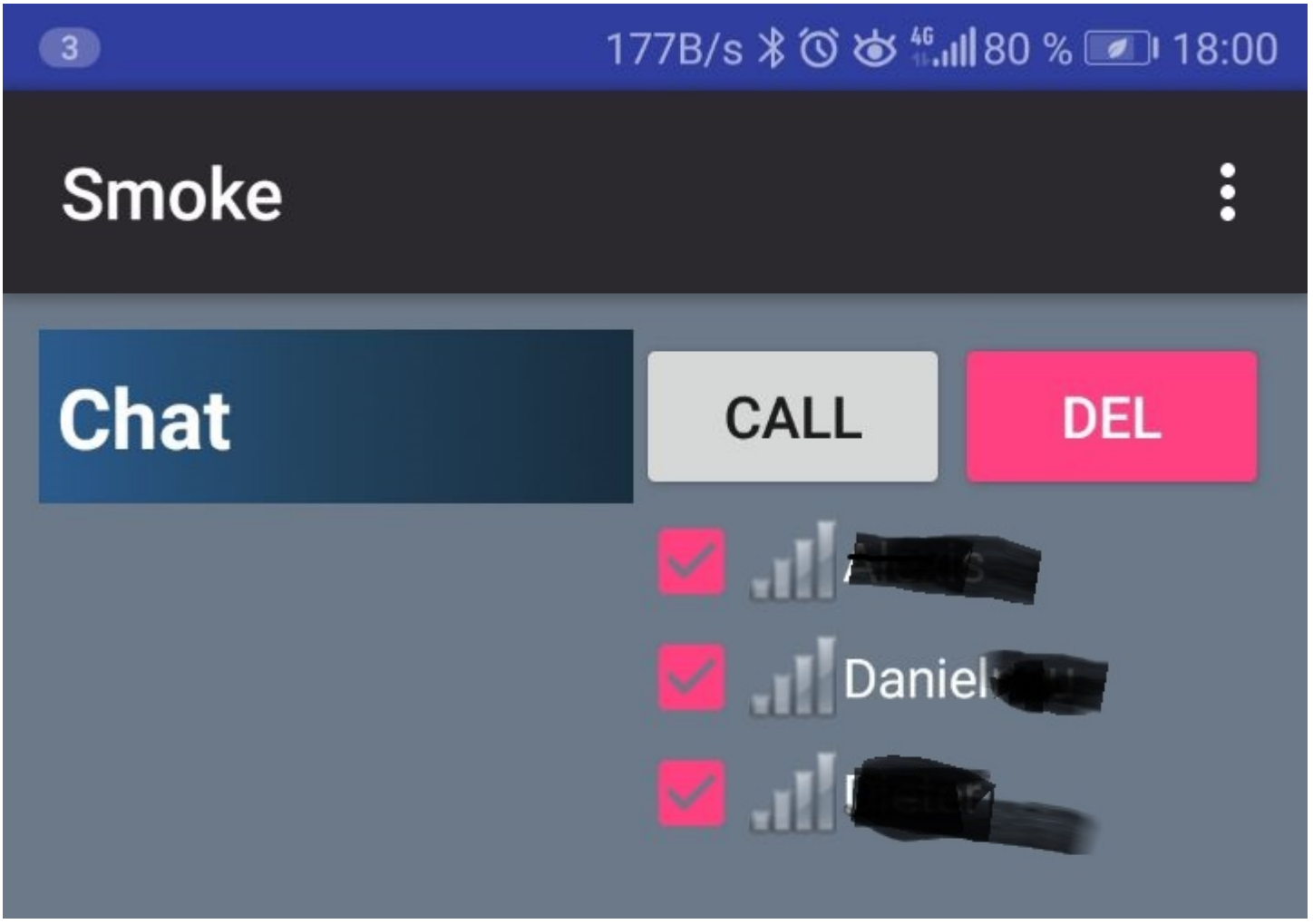


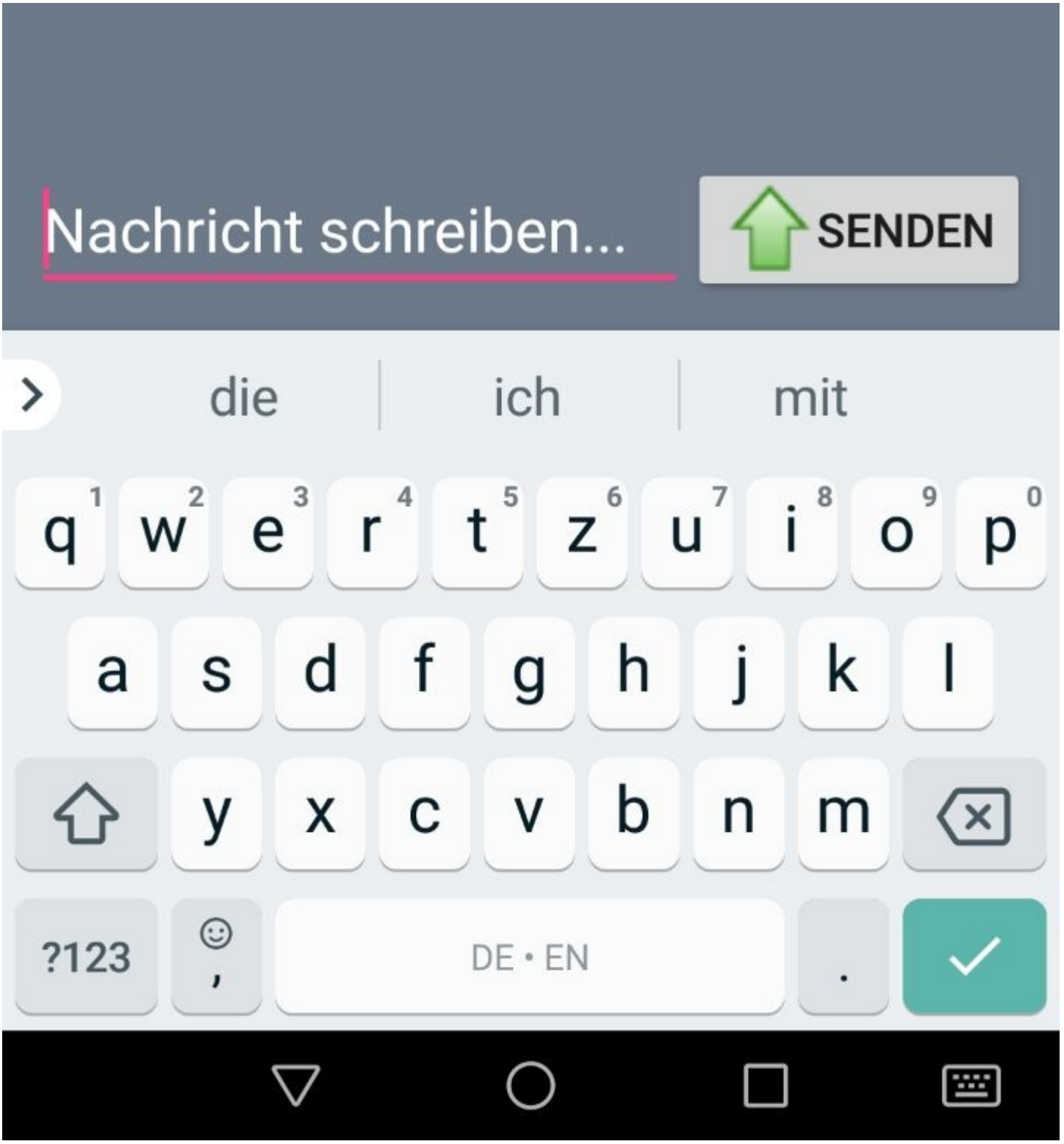
Die Teilnehmerliste in Smoke Messenger besteht aus Teilnehmer und Sip-Hash-Kennung. Das Symbol vor der Sip-Hash-ID zeigt an, dass die Schlüssel für die Verschlüsselung erfolgreich von Nutzer zu Nutzer übertragen wurden.

4.8 Benutzeroberfläche

Die Benutzeroberfläche ist derzeit noch nicht in eigene Seiten pro Teilnehmer gesplittet. Dieses müsste umprogrammiert werden, so dass jeder Nutzer eine eigene Timeline mit den Nachrichten erstellt.

Screenshot der Benutzeroberfläche





Die Benutzeroberfläche ist derzeit im IRC-Stile und zeigt sämtliche Nachrichten auf einer Seite an. Hier ist die Anpassung vorzunehmen, dass für jeden Nutzer eine eigene Message-Historie aufgebaut wird. Dieses kann durch kurzfristige Java Programmierung mit hauseigenen Bordmitteln eine wie bei anderen Messangern gestaltete Oberfläche erwirkt werden. Dieses ist die eigentliche Aufwendung und Investition in den Messenger bzw. dessen Code Basis, die bei der Kompilierung und dem Branding des Messengers auf eine jeweilige Stadterwaltung oder Schule erfolgen kann.

Die Benutzeroberfläche zeigt auch den “Calling” Knopf an, mit dem der Smoke Client auch das “Cryptographische Calling” unterstützt. Dieses ermöglicht neue Schlüssel zu generieren (vgl. auch die ausführlichere Beschreibung des Cryptographischen Callings in der technischen Dokumentation der Applikationen im Source). Mit einem Druck auf die Teilnehmer eröffnet sich auch ein Kontext-Menü, mit dem weitere Funktionen und kryptographsiche Funktionen ermöglicht werden. So kann beispielsweise für die Sitzung manuell und individuell ein Ende-zu-Ende verschlüsselndes Passwort in zwei Klienten zusätzlich definiert werden.

4.9 Quelloffene Codebasis und aktive Entwicklung

Die Codebasis ist quelloffen und damit ideal, aus den Fachbereichen gewünschte Anforderungen dort in Eigenprogrammierung einzubinden.

Der Vorteil der Code Basis ist weiterhin, dass geringe Abhängigkeiten zu anderen Code-Fragmenten wie Bibliotheken und anderen verschlüsselungsspezifischen Programmteilen bestehen.

Die Software-Architektur und auch die Verschlüsselung ist daher technisch ausgefeilt mit vielen Optionen und der Möglichkeit zu umfangreichen Schnittstellen für kommunale fachspezifische Erweiterungsfunktionen.

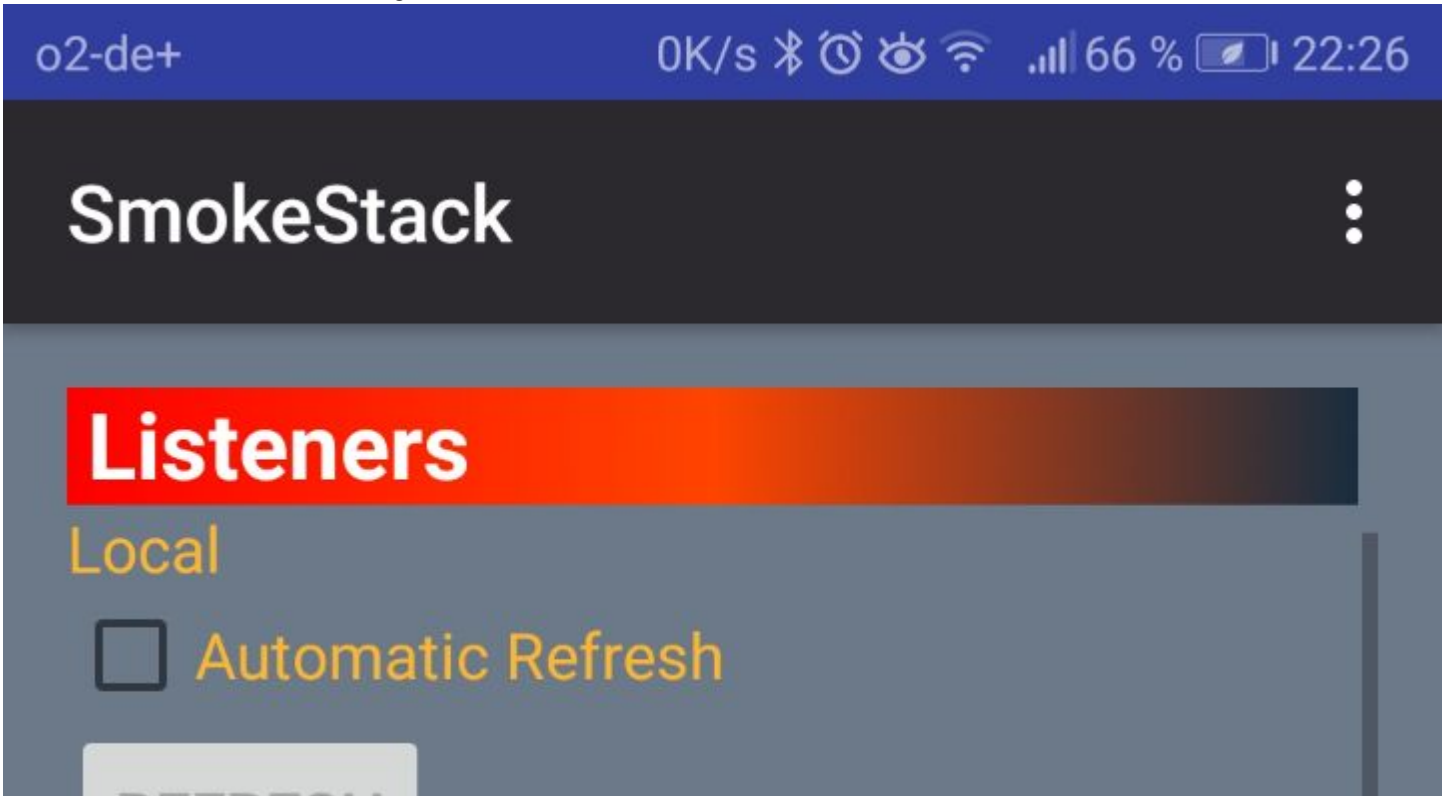
Aufgrund der Aktivität des Projektes in der Entwicklung kann eine Weiterentwicklung der Benutzeroberfläche des Android-Klienten gut von der bisherigen Entwicklung profitieren und diese einbinden.

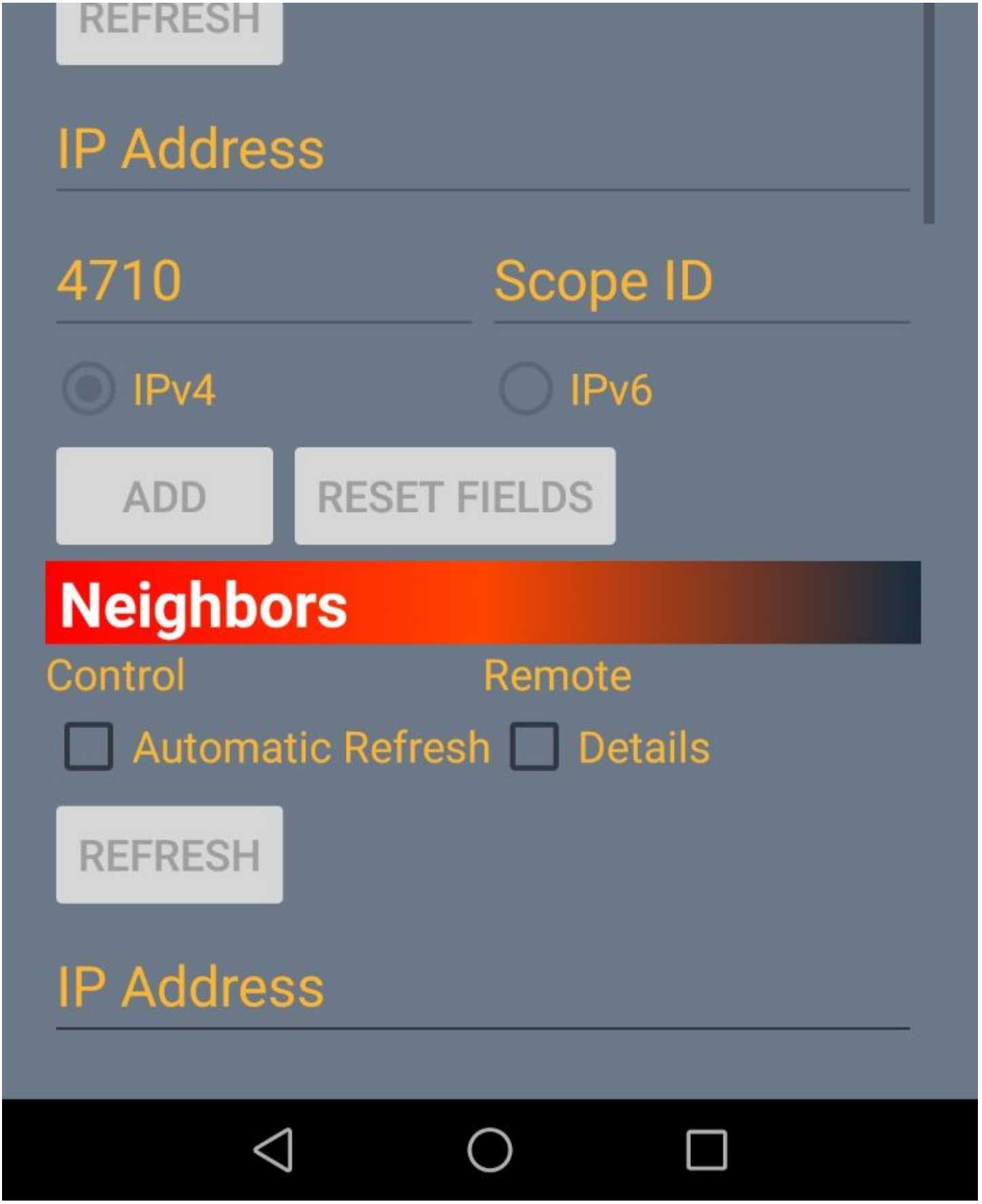
5 Funktionsweise des Smokestack Servers im Detail

Nachdem wir nun erstens festgestellt haben, dass ein kommunaler Messenger mit Verschlüsselung auf einer adäquaten quelloffenen Codebasis beruhen sollte, um weitere kommunale Funktionen darin einbauen zu können und dass zweitens der Setup des Kommunikationsservers ebenso einfach wie in eigener Hand möglich sein muss, soll hier nun für die ausgewählte Grundlage des Smoke Messengers im Folgenden nun der SmokeStack Komunikations-Server im Setup und mit seinen Funktionen ausführlicher beschrieben werden. Die Leichtigkeit, einen eigene (schulischen oder kommunalen) Kommunikatonsserver für eine verschlüsselt kommunizierende mobile App aufsetzen zu können, ist das A und O für die Gestaltung einer kommunalen Umsetzungen mobiler Kommunikation.

Wie beim Server Spot-On sowie Spot-On-Lite, der Deamon Variante von Spot-On, kann auch mit dem Server SmokeStack eine Einrichtung durch wenige Mausklicks erfolgen. Es ist ledichlich ein Listener zu eröffnen, was durch jeden Laien erfolgen kann. Wer die SmokeStack APK für Anroid heruntergeladen hat und auf seinem Tablet, Android-Media-Server oder Telefon installiert hat, wird nach der Setzung eines Passwortes um die Applikation aufzurufen direkt im Feld des Listeners nach der Eingabe einer IP und eines Ports gefragt. Es ist hier die IP zu nehmen, an der das Gerät angeschlossen ist (wird in Android oder auch im Router wie der Fritz Box angezeigt). Der Port wird idealerweise bei 4710 belassen. Ist die IP eine lokale im eigenen Haus-Netz, sollte sie durch weitere Geräte im eigenen Haus-Netz sofort erreichbar sein. Soll der Smoke-Stack-Server vom Internet aus erreichbar sein, ist die Firewall bzw. der Router für den gewählten Port entsprechend weiterzuleiten.

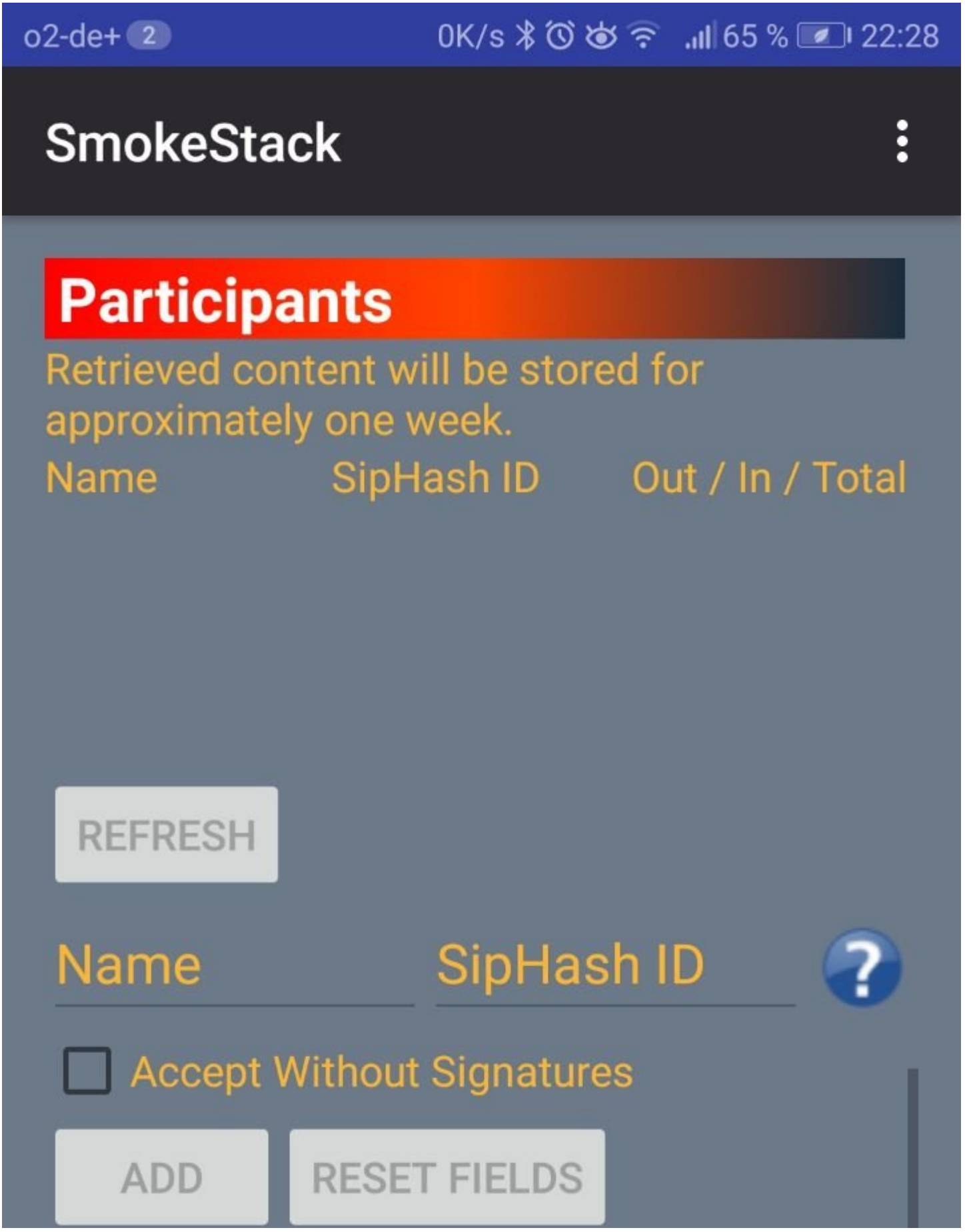
Screenshot SmokeStack Messenger Server: Listener





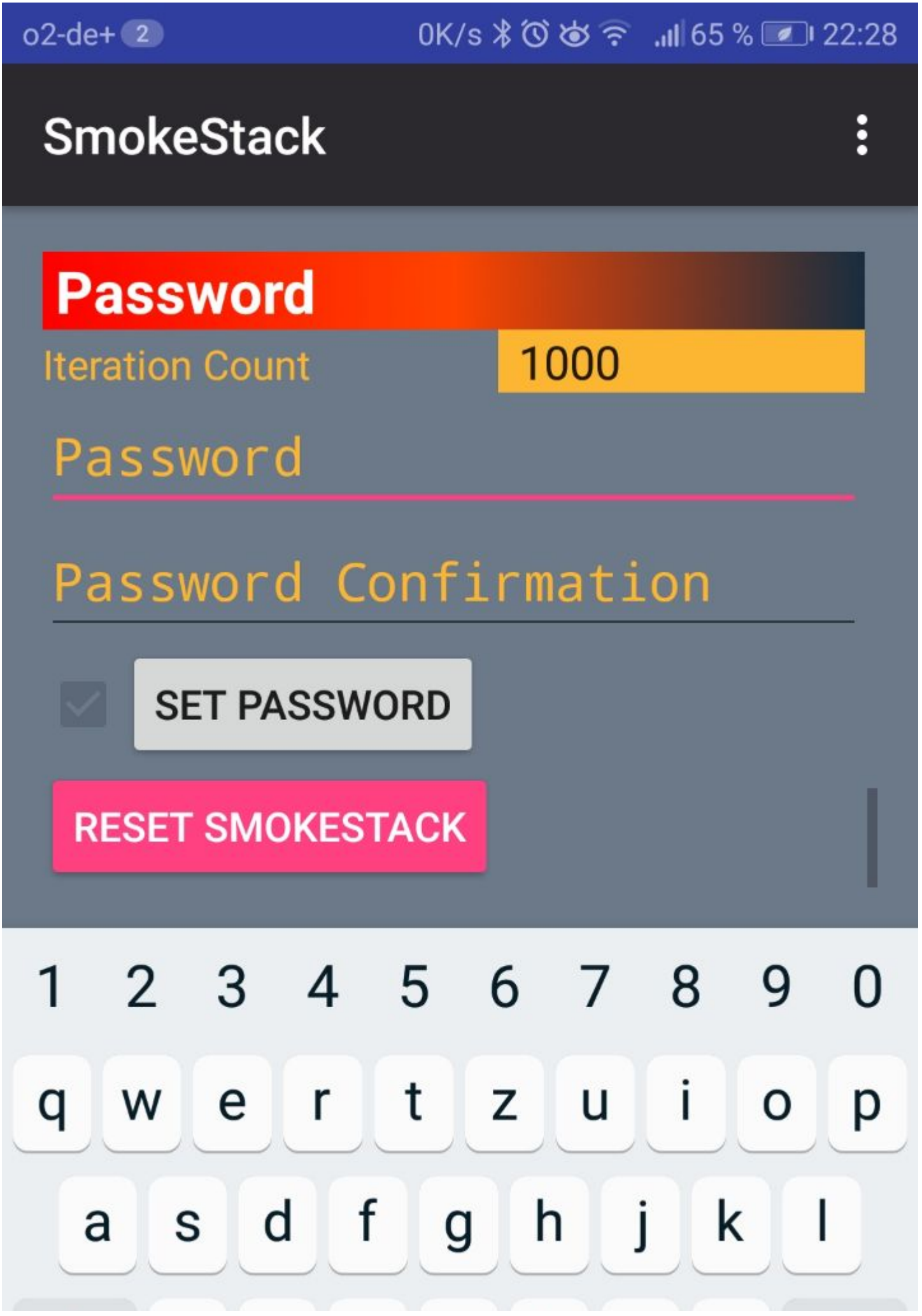
Der Router als Listener ist damit funktionsfähig. Um jedoch für die Nutzer weitere Annehmlichkeiten vorzuhalten, ist es sinnvoll, die Nutzer auf dem Server mit Namen und Sip-Hash-ID am SmokeStack Server zu definieren. Dieses betrifft insbesondere die sodann die Schlüsseltausche und auch das Speichern von Offline-Nachrichgten mittels der (weiter unten beschriebenen) Funktion “Ozone”. Wird der Finger länger auf einen Teilnehmer gehalten, erscheint das Kontext-Menü, mit dem weitere Optionen wie das Umbenennen oder Enfernen des Teilnehmers möglich wird. Aber auch Funktionen, den Schlüssel zu tauschen oder in anderen Instanzen zu hinterlegen, wird mit dem Kontext-Menü pro Teilnehmer möglich.

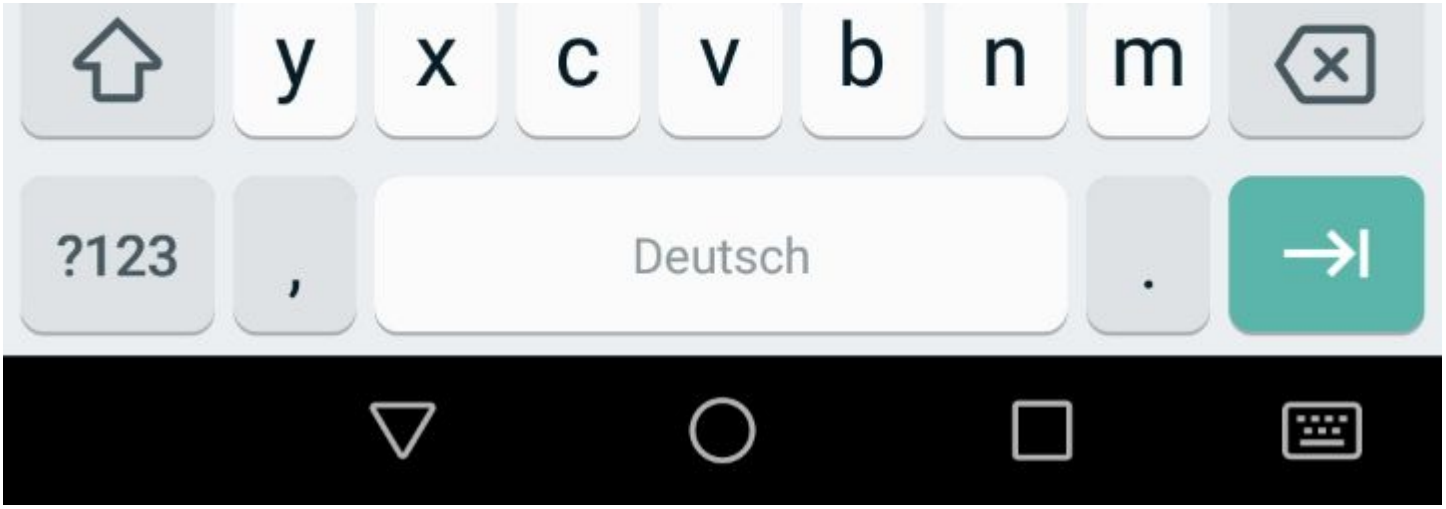
Screenshot SmokeStack Messenger Server: Participants



Die Funktionalität der “Nachbarn” ist dafür sinnvoll, wenn andere Server an den eigenen Server angeschlossen werden sollen. Die Server können also föderiert werden und somit stehen den Nutzern auch mehrere Server zur Verfügug, was z.B. die Sicherheit der Verfügbarkeit erhöht. Nutzer können also mit einem, mit beiden oder mit einem alternativen Server angebunden werden.

Screenshot SmokeStack Messenger Server: Neighbors



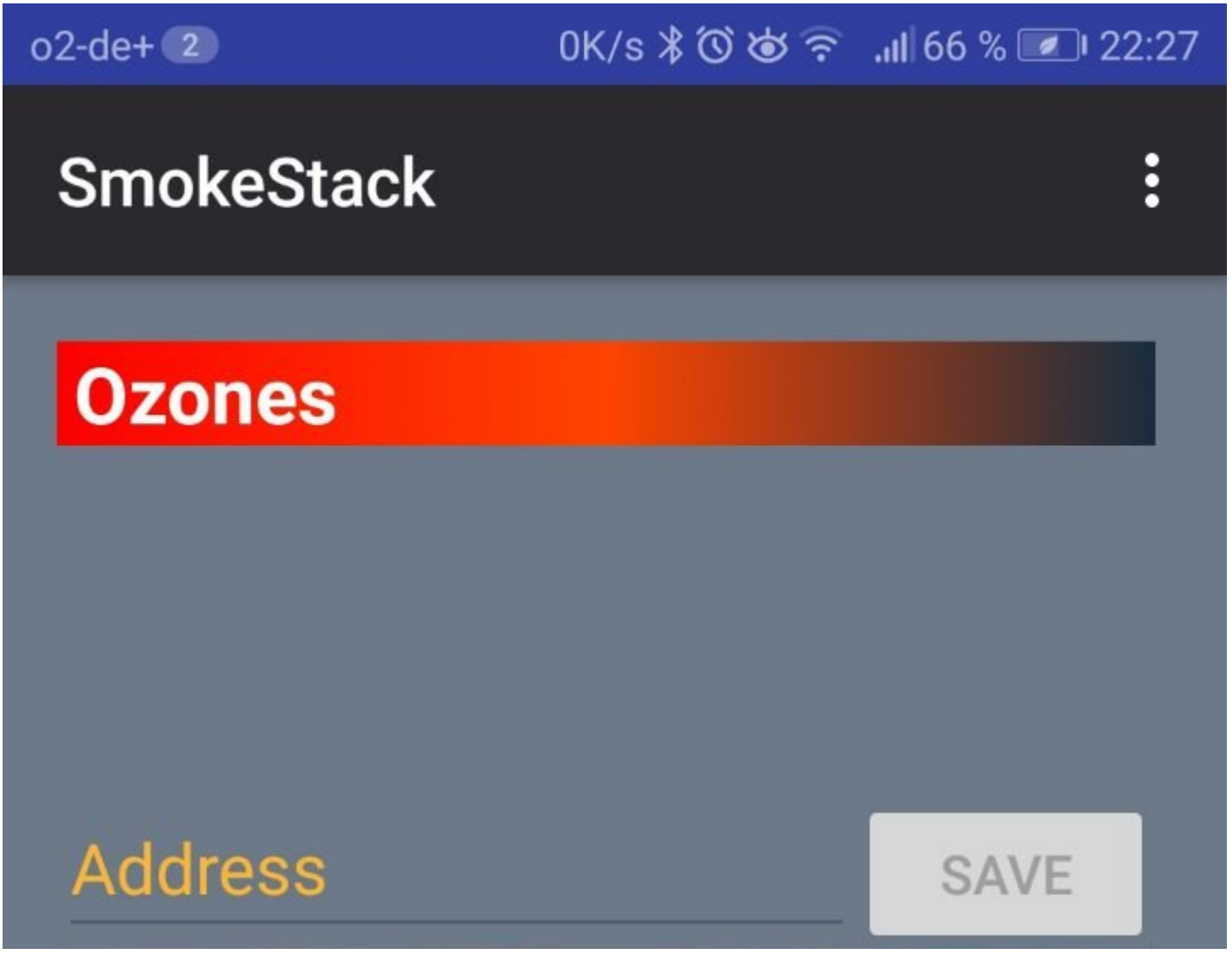


Die Funktionalität, ein Postfach auf dem Server einzurichten, wird in SokeStack “Ozone” genannt. Damit kann ein Nutzer, wenn sein Adressat offline ist, die Nachrichten im Ozone Postfach des Servers hinterlegen und abrufen, wenn er online kommt. Nutzer vereinbaren daher einfach ein Passwort in ihrem Smoke-Klienten und hinterlegen es ebenso auf dem SmokeStack Server. Mittels der cryptographischen Prozesse, können die Nachrichten dann nicht nur sicher gelagert werden, sondern beim Online-Kommen dann auch empfangen werden.

Zusätzlich zur Ozone-Funktion der Speicherung der Nachrichten, ist SmokeStack ein Schlüssel-Server für die private-öffentliche Schlüssel-Infrastruktur. Ein zusätzlicher Schlüsselserver ist also nicht mehr erforderlich, da dieses bereits im SmokeStack Server und im Flow mit dem Smoke-Clienten direkt eingebaut ist.

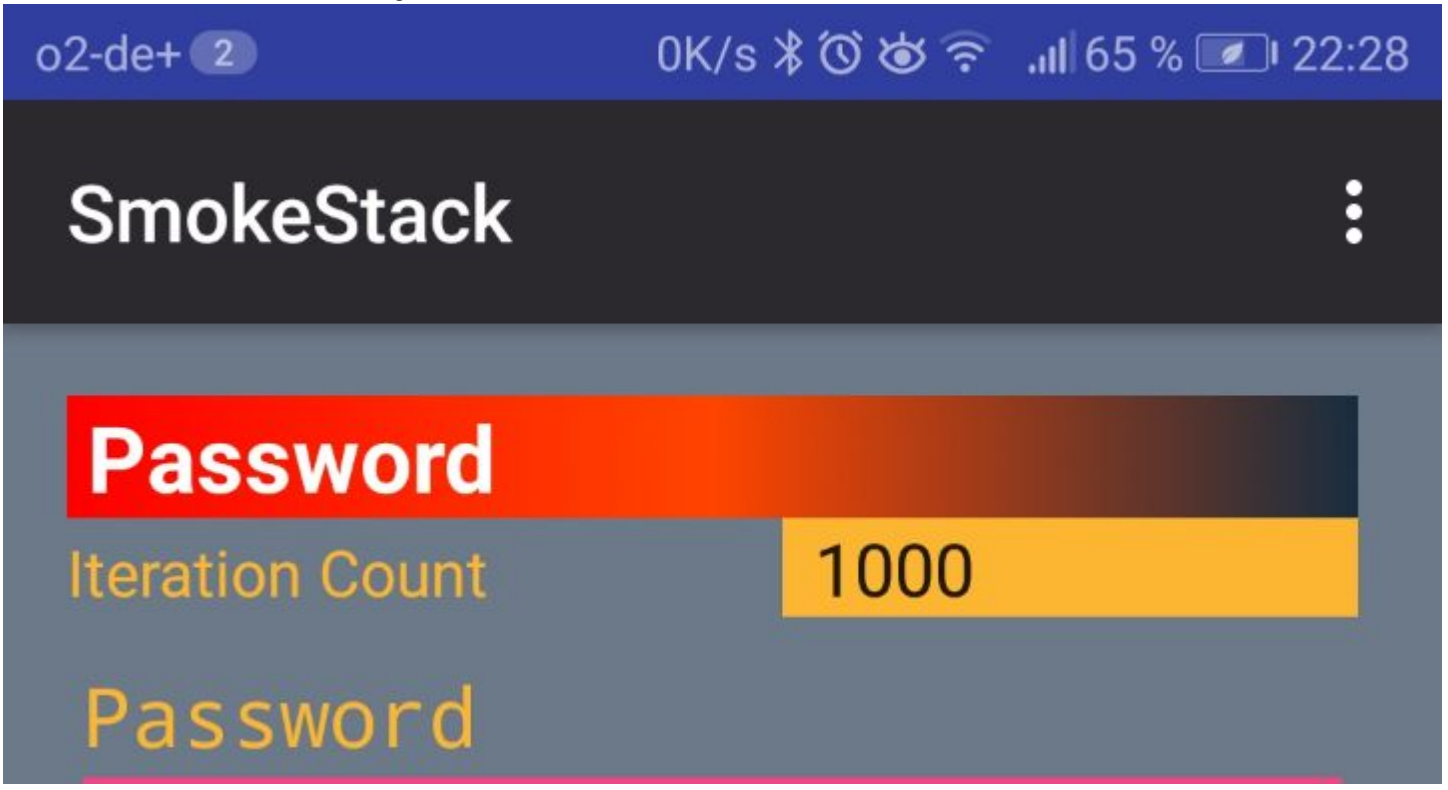
Durch Hinzufügung der Teilnehmer in SmokeStack ist der Server-Administrator verantwortlich für die Koordination der Hinterlegung der Schlüssel. Der Nutzer kann sodann mit anderen Nutzern die öffentlichen Schlüssel von spezifischen Teilnehmern ebenso über die Ozone-Funktionalität abrufen. Die Schlüssel werden ebenso mit digitalen Signauten gesichert. Die Signaturen sind Teil des EPKS-Bündels (Echo Public Key Share). Sind die Signaturen für den (RSA) Schlüssel gültig, wird der öffentliche Schlüssel akzeptiert.

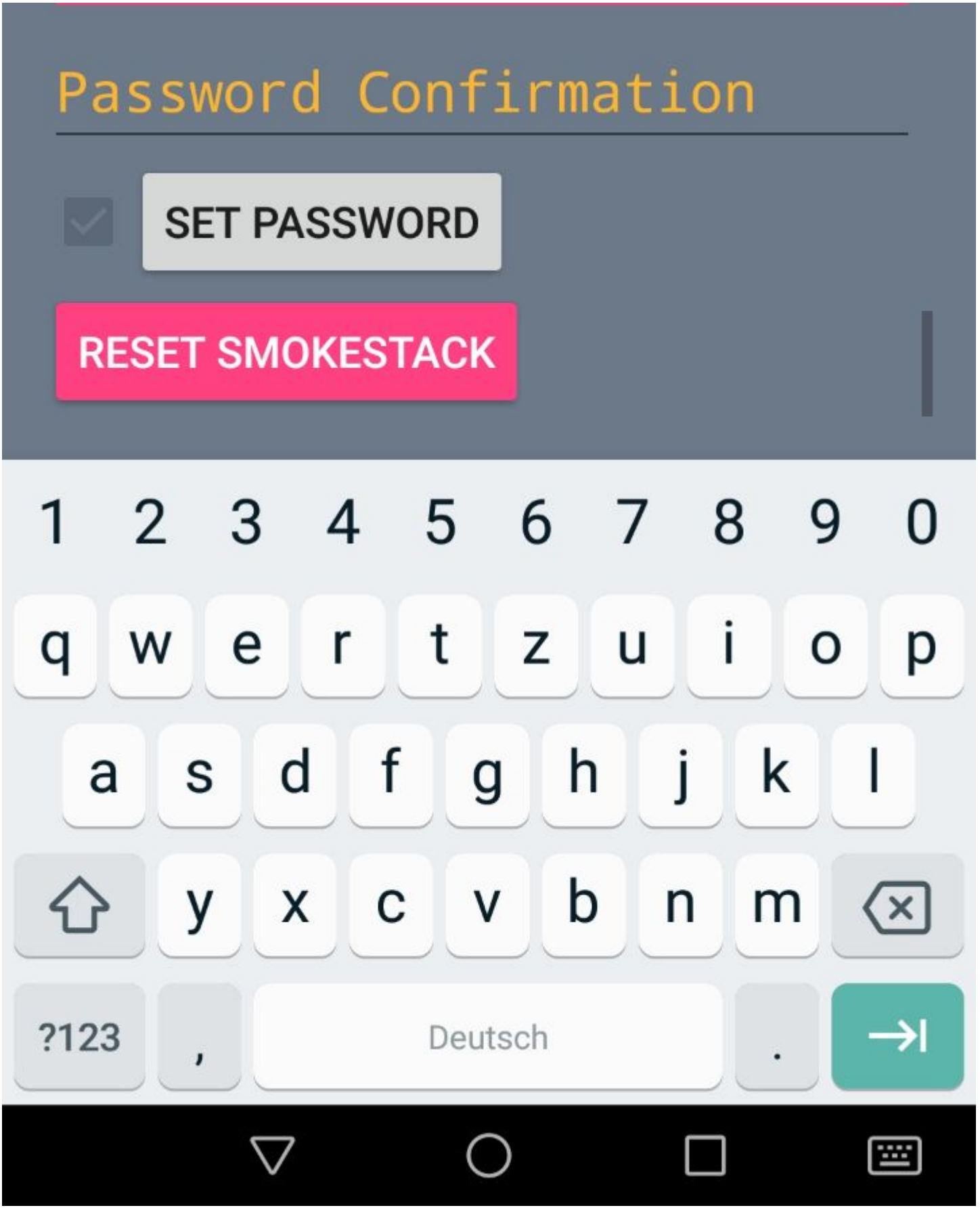
Screenshot SmokeStack Messenger Server: Ozones



Der Server wird wie genannt auf dem Android-Betriebssystem mit einem Passwort gesichert, so dass die Funktionalitäten oder der Aufruf der App nur zur Verfügung steht, wenn eine Authentifizierung erfolgt ist.

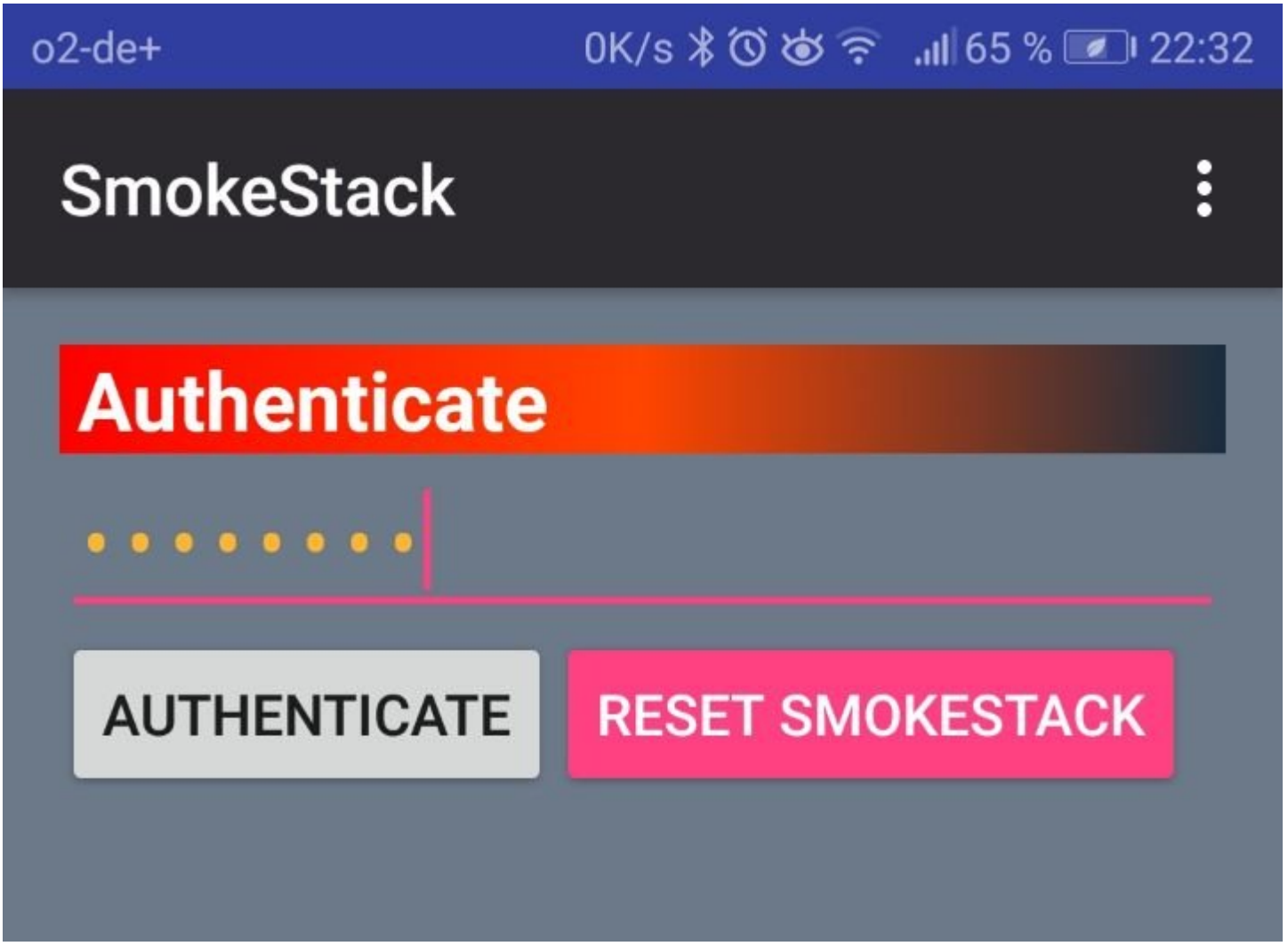
Screenshot SmokeStack Messenger Server: Password





Nach eingegebenem Passwort sieht die Startseite zur Authentifizierung wie folgt aus:

Screenshot SmokeStack Messenger Server: Authenticate



SmokeStack ist eine schnelle und mobile und zugleich stabile und funktionalitätenreiche Appkikation, um für ein Amt oder eine Schulklasse oder bei einem Workshop mit einem einfachen Android Gerät zahlreichen Nutzern einen Kommunikationsserver zu eröffnen. Darüber hinaus stehen ebenso die C++ Varianten Spot-On und Spot-On-Lite zur Verfügung. Es ist ein leichtes, mit drei Android-Geräten einen intermediären SmokeStack Server zwischen zwei kommunizierenden Smoke-Clienten auszutesten.

6 Management Summary / Ausblick und Zusammenfassung

Der Einsatz von WhatsApp in der Kommune oder Schule ist nicht nur aus rechtlichen, technischen, sicherheitsrelevanten und organisations-prozessualen Gesichtspunkten als Risiko zu betrachten und rechtlich sowie Datenschutzgründen zu verbieten. Klagen und Abmahnungen gegen Schulen und Kommunen sind denkbar.

Daher muss gehandelt werden und eine in eigener Hand administrierbare Software-Lösung gefunden werden.

Dabei geht es nicht um lediglich des Wechsels des Kommunikations-Channels, sondern auch um die Option, die Arbeitsprozesse der kommunalen Fachbereiche, die sich derzeit und auch in Zukunft zunehmend digitalisieren werden, an eine Messenger Applikation anzubinden. Dieses ist nur über die Wahl einer quelloffenen Code-Basis möglich, so dass hausinterne Programmierer Extensionen; Plugins und Schnittstellen programmieren können. Dazu wurden mehrere Anforderungen und Spezifikationen kommunaler Fachbereiche gelistet.

Es ist daher eine Make Statt Buy Entscheidung erforderlich, um mit einem geringen eigenen Invest über Bordmittel und ggf. hausinternen Java-Programmierern ein Produkt zu erstellen, dass flexibel durch die Kommune an eigene Arbeitsprozesse angepasst werden kann und zusätzlich die Prozesse und Serverkommunikation in eigenen kommunalen Händen hält und auch hinsichtlich der notwendigen Verschlüsselung sicherheitsrelevante Aspekte vertraulich handhaben kann.

Wie analysiert ist dieses mit der Empfehlung der Codebasis des Messenger Smoke Chat ideal umsetzbar und mit zahlreichen Gründen gegenüber weiteren Alternativen belegt, so dass vorgeschlagen wird, hiermit mit dem Projekt momedo ein Modellprojekt für Fachbereiche von Kommunalverwaltungen bzw auch von Schulen zu beginnen.

Ziel des Projektes momedo ist eine Analyse dieses Prozesses sowie sodann auch eine Aufhübschung/Weiterentwicklung bzw. Anpassung der Benutzeroberfläche der Applikation Smoke für Android mit Test und Evaluation und schließlichodann auch eine Portierung für Apple Smartphones.

7 Literaturverzeichnis

BlackBerry-Studie: Datei-Übertragungen (PDF, Excel, per E-Mail) und DSGVO – eine explosive Mischung, in: URL: https://www.it-finanzmagazin.de/blackberry-studie-datei-uebertragungen-dsgvo-explosiv-62029/, 08.12.2017.

DGSVO: Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), http://data.consilium.europa.eu/doc/document/ST-12399-2016-INIT/en/pdf, 27.10.2016

DSGVOAPP: Whatsapp & DSGVO – Kann das zusammenpassen?, https://www.dsgvoapp.at/2018/01/16/whatsapp-dsgvo-kann-das-zusammenpassen/, 27.01.2018

Frankfurter Rundschau: Recht auf Unerreichbarkeit im Job E-Mail-Sperre nach Feierabend, URL: http://www.fr.de/wirtschaft/arbeit-soziales/recht-auf-unerreichbarkeit-im-job-e-mail-sperre-nach-feierabend-a-612421, 18,02,2014

Handelskammer Hamburg: Steuer- und Handelsrecht: Aufbewahrungsfristen von Geschäftsunterlagen, URL: https://www.hk24.de/produktmarken/beratung-service/recht_und_steuern/steuerrecht/abgabenrecht/aufbewahrungsfristen-geschaeftsunterlagen/1157174, 17,04.2018

Iseninfos: DGSVO: WhatsApp auf Firmenhandy nicht mehr erlaubt, URL: http://www.iseninfos.de/mitarbeiter-unternehmen-whatsapp-firmenhandy/, 27.01.2018

Landesbeauftragte für den Datenschutz Niedersachsen: Merkblatt für die Nutzung von „WhatsApp“ in Schulen, URL: tps://www.lfd.niedersachsen.de/download/124022/Merkblatt_fuer_die_Nutzung_von_WhatsApp_in_Schulen.pdf, 19.10.2017.

Greis, Friedhelm: Strafe verhängt: Diese Nutzerdaten teilt Whatsapp weiterhin mit Facebook, https://www.golem.de/news/strafe-verhaengt-diese-nutzerdaten-teilt-whatsapp-weiterhin-mit-facebook-1803-133372.html, 16.03.2018

Momedo: Open Source Mobiler Messenger für kommunale und schulische Zwecke mit Verschlüsselung, Github, URL: https://github.com/momedo/momedo/blob/master/README.md , 2018

Schneier, Bruce / Seidel, Kathleen / Vijayakumar, Saranya: A Worldwide Survey of Encryption Products, February 11, 2016 Version 1.0., zit. nach Adams, David / Maier, Ann-Kathrin (2016): BIG SEVEN Study, open source crypto-messengers to be compared - or: Comprehensive Confidentiality Review & Audit of GoldBug, Encrypting E-Mail-Client & Secure Instant Messenger, Descriptions, tests and analysis reviews of 20 functions of the application based on the essential fields and methods of evaluation of the 8 major international audit manuals for IT security investigations including 38 figures and 87 tables, URL: https://sf.net/projects/goldbug/files/bigseven-crypto-audit.pdf - English / German Language, Version 1.1, 305 pages, June 2016

Smoltczyk, Maja: Datenschutzmängel bei Nutzung von Whatsapp an Schulen - Tätigkeitsbericht der Berliner Datenschutzbeauftragten 2016, URL: https://www.datenschutz-berlin.de/, zit. nach: Schulzki-Haddouti, Christiane: Datenschutzbeauftragte sieht schwere Datenschutzmängel bei Nutzung von Whatsapp an Schulen, URL: https://www.heise.de/newsticker/meldung/Datenschutzbeauftragte-sieht-schwere-Datenschutzmaengel-im-Berliner-

Gesundheitswesen-3678056.html, 07.04.2017

Smoke: Documentation of the Android Messenger Application Smoke with Encryption, URL: https://github.com/textbrowser/smoke/raw/master/Documentation/Smoke.pdf , 2017

SVZ: Mangelnde Aufklärung : Lehrer sollten Whatsapp und Co. nicht für Klassenchat nutzen, URL: https://www.svz.de/18100191, 18.10.2017.

Tung, Liam: WhatsApp is the most blacklisted app on BYOD iPhones in the enterprise, URL: https://www.cso.com.au/article/628110/whatsapp-most-blacklisted-app-byod-iphones-enterprise/, 04.10.2017

Wikipedia: Datenschutzgrundverordnung (DSGVO), URL: https://de.wikipedia.org/wiki/Datenschutz-Grundverordnung, 17.04.2018

momedo is maintained by momedo.

This page was generated by GitHub Pages.