Secure Privacy-Friendly Instant Messaging (IM) for Guidepal

Study and Implementation on an IM system for: Guidepal, A Social Networking Mobile Application

ALIREZA ABDI KELISHAMI

KTH Electrical Engineering

Master's Degree Project
Stockholm, Sweden February 15, 2015

XR-EE-LCN 2015:001

ABSTRACT

It is fascinating, and yet often neglected, that a user's privacy can be invaded not only by the absence of security measures and mechanisms, but also by improper or inadequate usage of security services and mechanisms. When designing secure systems, we must consider what services are needed and what is not. The work in this thesis revolves around privacy-friendly instant messaging (IM) systems. In such a system, an inadequate usage of security measures leads to having IM servers being able to intercept or gather users' private conversations. An improper usage of security measures could bring about non-repudiation which is desirable when signing contracts, but unwelcome in IM and private conversations.

We will look into requirements of the desired IM system, study the current state-of-the-art solutions, deploy an IM server, and briefly extend an existing modern privacy-friendly IM protocol and an open source mobile application to meet our security and privacy requirements. This extended IM application is called Guidepal-IM and is available as open source[1]

The thesis work is introduced and carried out at Guidepal, a startup company in Stockholm. It is therefore supervised partly at Guidepal and partly at KTH. Since Guidepal is also looking into possibilities of integrating an IM feature to its current social media apps, our contribution would also briefly extend to studying the limitations and recommendations for Guidepal's social media app to help user privacy preservation.

[1] Guidepal-IM sorce code is available under GNU General Public License at https://bitbucket.org/alirezaabdi/xabber-otr-certificate-edition.

ACKNOWLEDGEMENTS

Each time I look back since the beginning of my thesis project, I firstly hear myself saying like *'boy! I was challenged!'* The project literally brought my weaknesses in front of my face, and made me slightly more humble if not any more. But then I remember all the things I learned, both technical and non-technical. There have been twice the number of technologies mentioned in this report that I have studied, got familiar with and then decided to leave for it turned out that I do not need them in this project. Some example is OpenCA to implement a certificate authority, Google Protocol Buffer, XML RPC, etc. I feel I am technically a stronger person. And therefore it normally is not difficult to feel that I am filled with appreciation and gratefulness for the opportunity that I had to get challenged, specifically for the help of my supervisors, Pontus Naimell and Panagiotis Papadimitratos. I have, for so many times, noticed their support and patience to help me complete this task that I took upon myself.

I also would like to thank my friends who have helped me in this: Mohammad Khodaei has always been there and helped me a lot in this. Hongyu Jin guided me in some of the early challenges I had in programming tricks of the included libraries and protocols.

What is left to express my gratitude for is my family who kept me faithful in this little quest of mine, and http://stackoverflow.com/ website! Had this website feelings I would have thanked it in yet more sentences than one.

CONTENTS

LIST OF FIGURES

1 INTRODUCTION

We live in a society where anything that benefits a connection gets digitally connected. This trend cannot be stopped and will bring us more surprises by the innovations to come. But we also would like to keep it under control and in perspective. Privacy protection is one of the areas of concern with this information era. Privacy needs to be preserved in all aspects where the users would not benefit his personal information being available to another person or authority. It is not natural to have 100 percent surveillance over digital world. It is discussed at times we need some surveillance for national security, maybe to protect the society from threats some of the fashionable of which are terrorist attacks and threats. How we as a society could agree On surveillance issue is not within the scope of this article. This are hot ongoing debates on the way forward is. On the other hand it cannot be denied that history shows a lot of the terrorism is stemming from institutions not rarely supported or allowed by our own governments, eastern or western. We will not get deeper into this topic and a Wikipedia link to state-sponsored terrorism provides enough reference for the unconvinced reader suggesting that this claim is not false [1]. Now we only touched on this vital burning issue, we did not resolve it.

There are many research groups that study and advocate privacy friendly digital services. It is interesting that yet we can design our services such that we keep governmental surveillance possible for certain situations, but make it more difficult, meaning such that they have to have agreements with other parties involved. An example is the design behind the proposed vehicular communication's CA network in [2] where a certificate authority does not have enough knowledge to relate the identity of a data to a real person or vehicle. It therefore does not allow any party become the big brother able to do anything, but it needs a couple of them agreeing together that they really need to know which vehicle or person it was doing something. The reason why we think this is important to discuss is that it increases awareness and our preparation for future.

> *The service we target in this article is online instant messaging, and the philosophy that we espouse to is that no one should be able to read your personal messages except the actual persons in messages, we also*

want to have conversations deniable and this makes it an interesting subject to study from the technical point of view if not any other.

At Guidepal we develop social applications and we need to study implications of responding to the requests from our customers/users. Many applications today connect to user's social media account. For many of these applications it is part of the business plan to have users log in with their social media account, e.g., facebook or twitter account in order to bring their connections for the app company. As the usage of social networking applications spread further, and having mash ups between users' social media account and other applications gets more popular, the need for privacy protection is felt more than ever. We will take a look at IM applications and study how they can fit into our needs, what privacy risks they could pose to users and how to help users protect their privacy better.

This chapter would continue with a more detailed problem statement, assumptions and requirements. It will cast a deeper look at Guidepal and its current product in section 1.4. This would be useful since Guidepal is considering chances for integrating the IM functionality into its main app apart from offering this IM app separately. Chapter two will partly cover the background study and ideas on how and what we extend from the state of the art mechanisms meet our aforementioned requirements. It includes description on how the idea is implemented; and it ends with results. In the end chapter three mentions the contributions made, future work to look forward to, and brief recommendations for Guidepal's social media app. Appendix A includes a brief study on social media apps and their role in user privacy.

1.1 PROBLEM STATEMENT

Guidepal focuses on creating a social recommendation service platform based on the user and his or her friends' experiences enabling users to recommend and give tips - comments - about products, places and services [1]. It wants to enable its users to do online instant messaging securely with its business users. Business users are still user accounts, but represent a company, e.g. a particular travel agency is called a business user, whereas a normal user can be anyone using our app, they are the end users.

Firstly, in this thesis, we will consider Guidepal's requirement to provide an IM functionality where the messages that users send and receive are not

[1] Guidepal Application is accessible for free at http://Guidepal.com.

readable for Guidepal or other third parties. The employed cryptography should not put users in more trouble such that they could not deny that they sent the messages. Of course wire tapping could possibly put someone to trouble in a court even if data was sent unencrypted. The requirement here is not to make deniability more difficult! Hence part of this study will address privacy related issues and challenges for IM systems.

Secondly, we will consider the privacy issues raised as a result of having applications mashing up with and accessing users' social media account information. The raising concern is that although it is under the control of users to agree/disagree to allow the 3rd party application to access their account information, still by the act of allowing, what happens is that e.g., Facebook will allow that 3rd party download the users information on their own servers. Now users' information would exist on two servers: both on the Facebook account, and the 3rd party's server. Applications are required to agree to the Facebook Platform Terms and Policies [3] regarding the handling of the downloaded use's information, but to what extent they respect it, and how strong the 3rd party's data protection mechanisms are remains a concern. The more places the user's information is copied, the higher the risk of information leakage or misuse.

These two will set the ground for this thesis work. The main contribution of this thesis work is to deliver an implementation of a suitable IM mobile application that Guidepal can build upon to deliver to its users. A minor contribution is to study the privacy issues raised should the IM service be integrated into Guidepal's current app.

1.2 ASSUMPTIONS

There are assumptions we made when carrying out the project. The Assumptions made due to the limited time and resources of a master thesis project are:

- Backup - We assume user and system data is properly saved for backed up.
- The app does not get tampered by unauthorized parties, example is that an attacker can not define another fake CA certificate as trusted into the user's phone and this way try to cause damage.
- The IM feature which is the core focus in this thesis will for now be offered in a separate app. It is assumed this IM functionality be also later

well integrated into the Guidepal application. It is now getting more and more common for mobile applications to introduce an IM functionality for its users. Meetup app[1] for example is now letting users have private conversations on top of its core functionality.

- The Certificate Authority (CA) and server is protected against attacks. However we will have a very simplified versions of a CA for the sake of this work.
- All the received certificate signing requests (CSRs) are legitimate and the simplified server signs them without checking their identity through a registration phase that is normally handled by a Registration Authority (RA).
- It is assumed that we will have only one CA, and only one CRL distribution point, and that is hard-coded into the app and is not tampered.
- Requirements are given to the thesis work as input. Hence there is no customer contact involved in this work.

1.3 FUNCTIONAL AND SECURITY REQUIREMENTS

Regardless of our ability to fulfill all the requirements, this chapter will list the security requirements. We categorize requirements into functional and non-functional ones. Functional requirements are the ones that have to do with what your system needs to be able to do, whereas non-functional requirements are the qualities of the system, depicting how the system must be.

1.3.1 Functional Requirements

IM functionality: It is a requirements from Guidepal's business customers that messages must not be readable for Guidepal or other third parties. When offline, the message must be stored on the server. Once delivered, they are expected to be deleted from the server. It is a plus if secure offline messaging is also possible.

Backup: Empowering us and our users to recover the data in case of loss. Although we do not take any action in this thesis regarding providing backup,

[1] http://www.meetup.com/ helps groups of people with shared interests plan events.

we still will briefly touch this topic in this part to clarify what it is that needs backup. We have business users and normal customers in our system. Businesses require to have backup services. Backup should be both for messages and guides. Data that needs backup can be divided in 3 categories:

1. Venues and the tips on them. This is taken care of with the help of services on Amazon EC2 servers.
2. A user's Guides, and his comments, again provided by Amazon's backup mechanisms.
3. Messages, i.e. users' message history, stored safely on the mobile app.

Because they are public, venues and tips are saved on the Guidepal server. This way, after a user changes or loses his phone, this information consisting his previous general comments is already up on the Guidepal server. How about backup guides and the user's descriptions on the guides? This is also taken care of by Guidepal. The guides and the comments on them are public and there is nothing wrong with storing them on the server. How about a backup for private messages i.e. the IM history? No backups are done on our servers, messages are deleted once they are delivered to user's app. Further study of the backup process, mechanisms and its implementation is out of the scope of this work.

1.3.2 Non-functional requirements

Scalability. The system needs to be scalable. Amazon EC2 is what we run our backend services on. It provides features to scale up as the business grows.

Security: Below we discuss facets of security and privacy preservation:

Confidentiality. The messages sent in between users and/or businesses must be confidential. Even Guidepal server must not be able to decode it, although it acts as a relay to send what user A sends to user B.

Sender and Receiver Authentication - Integrity. Users A and B must be able to verify each other to be the ones intended. The stronger the authentication, the better the users are protected against attacks. Sender authentication is that you as the sender authenticate yourself not only to the IM app, and the IM server, but also to the receiver. Receiver authentication is where the receiver is authenticated to the IM app, the IM server and the sender.

Guidepal Server Authentication. The server must authenticate itself to the user's mobile app. This way we will avoid the system from some of the attacks.

Registration authority for Businesses. A malicious party might try to masquerade a business for any purpose. Motives can be user information theft, and destroying a business's reputation by deceiving its customers on Guidepal to reveal personal information to a fake account claiming to be the real entity.

Message Integrity. This means that we must be certain that the messages sent and received are not tampered, nor forged by anyone else.

Forward secrecy. Also referred to as Perfect Forward Secrecy, it means if for any reason your long term private key was compromised, still the messages that you have sent in the past cannot be compromised. We take a deeper look into this in section 2.1.2.

Revocation. Revocation of a permission or certificate of a user when necessary. Guidepal reserves the right to revoke the certificate of a user in certain conditions. Here we want to make sure it has the ability to do that. Users must be able to be notified about validity of another user who presents a certificate.

Generic app. To make it fail-safe, devices that cannot generate cryptographic key pairs for any reason, the communication must be still possible to occur. Same applies if the user's mobile device fails to proceed with any of the phases needed to generate his own certificate and getting it signed by the CA, stored on the device or using it to do the cryptographic operations that it takes to establish a secure channel with user B. Users must be able to communicate with one another and send and receive messages even if their mobile device or the system fails to establish the secure channel. But users will be notified on the messenger screen that the connection is not secure, or verified.

1.4 GUIDEPAL'S SOCIAL APP, A DEEPER LOOK

Let us briefly mention the core functionality of the current Guidepal app. Today it does not provide an IM service to users, but it creates a social recommendation service platform for a user to recommend and give tips about products, places and services [1]. In an example, imagine yourself sitting at a dinner table with friends, at a city you are new to, say Stockholm; and they recommend good restaurants, bars, service providers, etc. that you are tempted to take notes or ask the person to provide you with links to the interesting places and services. Needless to say, there are times where you would like to hear from a friend. No matter how many other people on the web say it, if you hear a friend recommend it, it makes a difference. In this example scenario or many other,

[1] Guidepal application is accessible at http://Guidepal.com.

there is an undeniable interest to have the people already in your network who have e.g. explored the city tell you "New here? - No worries, download this application Guidepal on your mobile phone or sign up in the website and follow me on my Guides: Stockholm's awesome restaurants, dance floors, yoga places, etc."

Guidepal is an application that enables users to recommend one another "venues" and give tips on those venues. Everyone can create guides. A guide can include one or more venues. It runs on both web and mobile platforms. By Feb 2013, it was developed to work on iOS and web, not on Android or other platforms. By August 2014, it already works with android as well, and has grown to more businesses. In Guidepal app, a user will have his own and his friends' comments and tips under their guides." As another example, consider a travel agency that has a guide for its customers flying to the Canarian Islands. It wants to have a guide - or guides - for the customers who are registered with the agency.

Guidepal offers its services at different levels:

1. User-to-user level. Users sharing tips and guides with one another.
2. Business-to-user level: Businesses negotiate with Guidepal, register themselves a page. Take the example of a hotel in the Canary Islands. Note that Guidepal does not get involved in payments and such. A user on Guidepal who has already purchased a service, or in this case, paid for a hotel room, usually gets a reference number in return for his purchase. He can then use this reference number and negotiate with the hotel through the intended hotel's Guidepal page. So, users and business can use Guidepal as a communication channel. But very important is its privacy protection. Patients might communicate with clinics over their appointment with their doctor via this service.

In this case, one might wonder if it is possible to have a mechanism that would make it hard for attackers or even Guidepal itself to understand if a user X is sending some encrypted data to business X that might be a clinic for fertility problems. Even knowing to what business a client is communicating might be unpleasant for many. Not only do we want to make it secure from outsiders, but also the requirement is that we provide privacy at a level that businesses can have enough trust that neither users nor Guidepal can read the content of the messages and files sent over the channel. This idea about securing communications over mobile phones is not novel in that other works have already

implemented it. Gibberbot, e.g., is a secure instant messaging application [1]. It goes without saying that just because there are widely used IM solutions, it does not mean we should not bother about making our communication more secure and privacy friendly.

As of Sep 2014, users can sign up using their Facebook account or their email address. If using email address to connect, they only choose name and password, and they provide their email address which will be their user name later on. Today they can also skip logging in or creating any account.

Each user has a user name i.e. his email address, which is unique from other users. To see other user's guides, you do not need their permission. But if you would like to do instant messaging with them, first you need to send a contact request. User B can accept, reject, or ban a connection request from user A. When user A invites user B as a contact, he sends the invitation only. User B can accept it if he wishes to share contact with him.

Practically speaking, much of privacy protection has to do with giving users more and more control over data related to them. Ideally, users must be able to efficiently control their data and this must be in align with Guidepal's business plan.

1.4.1 SNAs and Privacy Issues

Social networking applications (SNA) are the applications such as Guidepal that connect to a user's Social Networking Servers (SNS) such as Facebook. In the Facebook-SNA information flow model - presented below in figure 1.1 - there are two main categories of privacy breaches:

- The 3rd party application itself or an employee who has access to the downloaded users' data has the ability to expose private information of the user. Because the data that is downloaded from Facebook is not required at all to be encrypted and then saved, in a normal case, the programmers who can see the access tokens can read the data and export it somewhere else for unauthorized usage. So, imagine more people or entities now know your friend list - in case you have granted that specific app permission to read your friend list. Same applies if agencies permitted by law like national intelligence agencies in some countries can force an IT company to give them users' data on their request.

[1] Gibberbot is a secure instant messaging applications which is accessible at https://guardianproject.info/apps/gibber/.

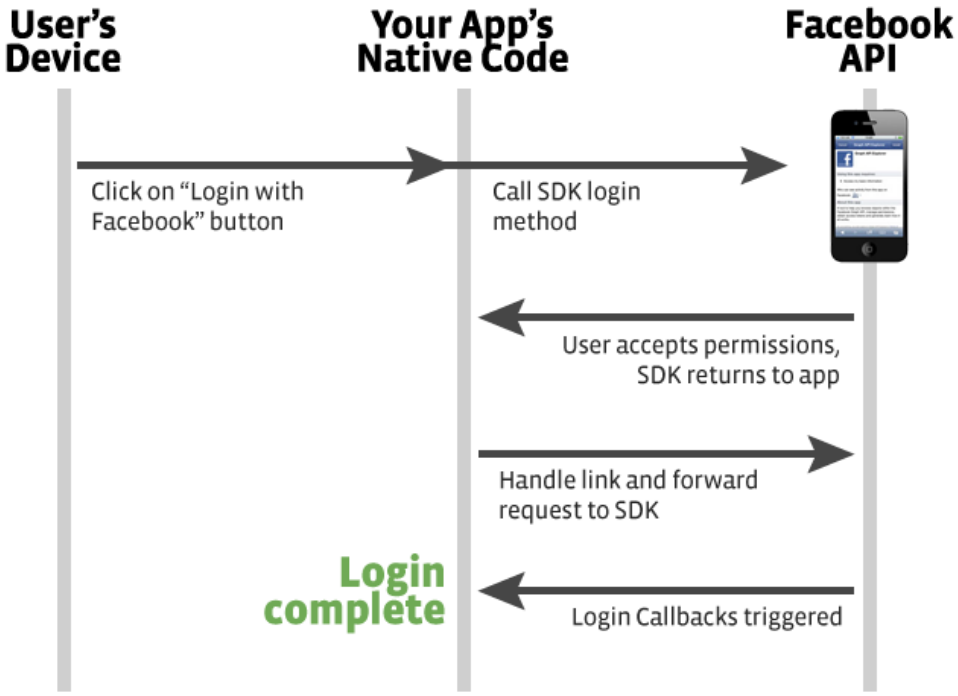

Figure 1.1: Shows the Client-side login flow in Facebook.

- Outsider attackers: Any other unauthorized person that gets to read, in some way, the user's information that was meant to be readable by the 3rd party application only.

Beach et al. [4] analyze the privacy and security issues that rise with the current practice of the mobile social network applications. In this work, the threats are divided into three categories. First is the issue of direct anonymity. It refers to the ability of a given party to link a given user to his real identity in the real world. Some applications even send users' social network ID in clear text. An eavesdropper can link this ID to the corresponding user's public information and this way chances are that the user cannot remain anonymous. Second comes the issue of indirect anonymity problem: Even if the identity of a user is not directly shared, by knowing the preferences of the user, say the list of a user's favorite movies, restaurants, etc. you are revealing information that can possibly help others identify you. The third category they discuss is eavesdropping, spoofing, replay, and wormhole attacks. To solve the problem of identity disclosure on mobile SNAs, they suggest the usage of an identity server. This server allows a user to specify any number of anonymous identities,

namely AIDs, to the same user. By using timeouts, the only way remaining to tie an AID to the identity of a user is to compromise the Identity Server in their architecture.

Good projects have started over the past years that can be thought of as privacy-friendly social network sites. Among these works are projects aiming at using current social networking sites and having users' data unreadable for them using suitable encryption. Beato et al present their work Scramble your social network data [5]. They work under the project PrimeLife [6], a research project that was funded by European Commission. PrimeLife's motto was "Bringing sustainable privacy and identity management to future networks and services." While some research has been done on how to have a secure and privacy-friendly social network by design, fewer studies have been dedicated to studying how we mash up the SNAs with social networks that are used by hundreds of millions to avoid exposing private information of users. For an example, consider an application mashing up with one's Facebook account such as Foursquare [1]. Since it is not the main focus in our work, we only include a brief study on SNA-SNS relationship in appendix A A. There we will also study the implications of having another architecture and data flow of the network of SNS and SNA on user's privacy protection are less exposed or not exposed at all to third party SNAs that download users' data from Facebook.

Krishnamurty et al mention in their work [7] that Facebook did not have today's privacy feedback and control in the past. They suggest that privacy concerns and studies might have been one of the reasons for the little improvements in user privacy preservation. Today, still third party apps gather user's information, in many cases we hear no strong privacy protection strategies are in place; and hence there is an increased concern for user information leakage, and therefore it is necessary to pay it more appropriate attention.

To Trust or Not to Trust!

Problem of trust - When it comes to privacy-friendliness, it makes a tangible difference between showing your application's source code to public, or hiding it and just having users trust you in uncertainty. You might make an effort to assure privacy protection by saying e.g. our application product stores private keys on the the local database of the user's mobile phone and refuses to read it. However, one can question what is the guarantee the software does not maliciously upload the private key to somewhere on the internet or share it with intelligence agencies? It is up to the software developer to solve the problem of trust.

[1] The Foursquare mobile and web applications are accessible via: https://foursquare.com/.

First, areas such as the country's health system have privacy policies and requirements. Second thing to consider is that there are ways to help the problem of trust for users and businesses, one of which is code verification. Although every software person knows that having their code verified is a help to the software's reputation, still most refuse it. To mention some of the reasons, it is for the time that it takes, and high expenses it requires. Each time the software wants to be updated, you need to get it certified, signed by the code verifier, and that takes a long time. Considering the models that startup companies follow, if they follow a model at all, or in ad hoc approaches to software development, code verification just does not fit in. It is not the way forward, but it is looked at as an inefficient way to bring more trust in your product. Another factor adding to the costs of such a decision is the needed expertise for developers and team leader to prepare documentation and write neat code.

Another way to help users and businesses to trust the code easier is to have the code open source. Knowing that other interested developers can take a look at the code, and make sure it is free from severe security weaknesses and malicious code. Another reason is that, you want to show neat code from you, more or less for the same reason you do not walk naked down the street. Companies can judge their code as low-quality and a mess, and therefore avoid putting it public until when they know they have an neat enough code. Refactoring the code can cost a lot, and might be not exactly the next important thing that managers want their developers to do in this fast growing IT industry where speed in market adaptation is key for not only success but also survival. As a result releasing the software code as open source is not a solution to all projects.

Challenges of Mobile Applications from IT Law Perspective

As a software company a software company must know what the rules of the target country or countries are when it comes to data protection and government enforcement possibilities. Otherwise once can be surprised by seeing government forcing to reveal private information of the users. If you believe this should not happen, why not making user data some nonsense unreadable gibberish for yourself too? This way, even if you provide all information that you have about a user, it makes no sense to others.

2 SECURE INSTANT MESSAGING — DISCUSSION AND IMPLEMENTATION

This chapter looks into the related work and field study, and implementation of an IM system that complies with the requirements mentioned in section 1.3. Below we bring forward a discussion on some of the state of the art effort on privacy protection of users in chat and presence applications.

2.1 BACKGROUND

When it comes to studying applications, it is way more simple to talk about open source applications rather than proprietary closed-source applications. Skype for example is not an open source application and therefore a researcher or a user should refer to its terms and conditions to know how his data is treated by skype. Few might try it out in labs and analyze sent and received messages via skype to verify its claims. Moreover, skype does not make it clear if it is not able to view user's messages itself [8]. News reports suggest that user conversations on Skype can be divulged to US government whenever asked by the government[9]. This problem sadly generalizes to many other chat applications such as Facebook chat, Google talk which is a chat system for Google.

The story is different when it comes to open source applications. Think of a scenario where you put together and use a mobile app and server, both of them open source and extensible to suit your needs. However these client and server solutions come with a variety of architecture models, underlying protocol, and features and licenses. The license is a factor that one must also have an eye on not to be surprised by its implications and limitations. For a full list of licenses and their descriptions, refer to [40]. Next, we will take a close look at some of the commonly used instant messaging applications.

2.1.1 | Instant Messaging Clients

Here we will take a brief look at some of the IM applications or clients. An IM client is one that users can install on their phones and connect to our IM server - which will be described in section 2.2.3. We will see how some of the solutions suit our requirements better and is therefore more tempting for us in this project.

- IM client Trillian.
  Trillian[1] is an instant messaging client that lets users chat across platforms and networks including Facebook, Google, MSN, AIM, ICQ, XMPP, Yahoo!

  - Encryption end to end, but its Diffie - Hellman key exchange uses 128 bit long prime numbers which is very weak.
  - No authentication, and therefore susceptible to MITM attack
  - No deniability or perfect forward secrecy
  - Uses like PGP signatures

  Looking at Trillian's privacy policy available on their website, you read "... may access or disclose information about you, including the content of your communications, in order to: comply with the law or respond to lawful requests or legal process". Trillian server and employees have the chance to read your messages. This does not comply with our requirements.

- The Pidgin Encryption. (Previously Gaim Encryption)
  Pidgin[2] is a free software that provides many third party plugins. One of the plugins is Pidgin Encryption. Upon installation of this plugin, the app will generate a key pair and transmit it to others who also have this plugin.

  - Encryption and authentication
  - No perfect forward secrecy
  - Messages digitally signed, and therefore difficult to deny you wrote them.

  Pidgin's privacy policy suggests on their website: "Your privacy is important to us. We do not sell, trade, or rent any of your information. We

[1] Trillian is available in free and paid versions at https://www.trillian.im/.
[2] Pidgin is available for free at http://pidgin.im/.

do use cookies on this site, but only collect anonymous information." Its weakness is mainly that there is no forward secrecy nor deniability. This means in case the user's phone is compromised, the user is in trouble in that he or she would find it difficult if not impossible to deny that they actually signed the messages. And this is one of the reasons why it is not the best choice for us.

- Secure Internet Live Conferencing (SILC)
  SILC [1] is a protocol that is implemented by applications such as Messengeroo [2].

  - uses a completely separate network
  - Share messages (securely) with SILC server, or
  - uses Pre-shared long-term secret, or
  - can do Peer-to-peer communication (hard with Network Address Translation - NATs)

  Should you have a private conversation with someone using SILK, you have to agree on a pre-shared key beforehand. This together with the fact that it uses a separate network (and therefore problems to solve over firewalls) makes SILC not a desirable choice for us. Messengeroo writes in its privacy policy that "We will make every effort to preserve user privacy but MessengeRoo may need to disclose information when required by law." Again, the fact that it can have your messages in plaintext to deliver it to others would cross itself out from our list.

- Jabber Protocol
  Jabber [3] is an instant messaging service based on Extensible Messaging and Presence Protocol (XMPP) [10].

  - Encryption is between Client to Server, not end-to-end.
  - So server can perform MITM attack., or simply read the messages.
  - Or with the server being attacked, all content on it can be exposed to others [11].

  And for these reasons Jabber does not meet our expectations.

[1] SILC is delivered in open source packages and provides secure instant messaging and conferencing services. It is available for free at http://silcnet.org/.
[2] Messengeroo is one of the applications that implement SILC protocol http://www.messengeroo.com/.
[3] Jabber is available at http://www.jabber.org/.

- Off-the-Record messaging (OTR)
  OTR [1] aims at providing conditions close to a private conversation you might have with a friend. This is the motivation behind naming it Off-the-Record. It is designed in Cryptography, Security, and Privacy Research Group (CrySP) at the university of Waterloo.
  It is one of the projects corresponding to the cypherpunk movement [2]. A cypherpunk is someone who believes that individuals must be empowered to be able to protect their privacy over the internet by using strong cryptography. This way the cypherpunk movement helps social and political change. Of all the projects that cypherpunk movement has supported few are: anonymous remailers, Pretty Good Privacy (PGP) for helping users send and receive e-mails securely, Tor network for helping user anonymity over the internet. Julian Assange, the founder of wikileaks, is one of the well-known cypherpunks endeavoring to provide for a privacy-friendly internet.

  - Ships with Linux distributions today
  - Available as a plugin in Pidgin, AIM, several mobile phone applications and platforms.
  - End-2-end encryption and confidentiality
  - Authentication, not with using digital signatures
  - Deniability - anyone who could read the message could also have forged it
  - Forward secrecy - if a key is compromised, previous conversations are still safe!

It is clear that OTR suits our needs and requirements better. There already are desktop and mobile phone applications that have integrated OTR. We will use one of the android applications that is well maintained and is available for free from Redsolution [3]. Due to time limitations of a thesis project, we will only implement for Android. No doubt it is possible for Guidepal to use the idea and do necessary steps to offer similar solutions for other platforms such as iOS[4], or desktop [5]. Next we will take a deeper look into OTR protocol.

[1] OTR is available at https://otr.cypherpunks.ca/.
[2] To read more about cypherpunk movement refer to http://www.activism.net/cypherpunk/.
[3] Redsolution http://www.redsolution.com/ offers the complete source of their android implementation of Jabber (XMPP) client that integrates OTR protocol. Its source code is available at https://github.com/redsolution/xabber-android, and is usable under the GNU General Public License.
[4] For iOS: ChatSecure is one of the iOS applications that integrates OTR in Xabber instant messaging https://github.com/ChatSecure/ChatSecure-iOS
[5] For desktop, there are IM clients such as Pidgin who have OTR support in a security plugin. Pidgin is available on all major operating systems: https://www.pidgin.im/

2.1.2 Off The Record Messaging

We saw that Off-the-Record messaging is a protocol that aims at providing communication conditions close to a person's private communication in reality. We will get deeper into it here and in next chapters. OTR provides the following security and privacy services:

- Malleable encryption to provide plausible deniability. Stream cipher suits this purpose. Why not using SSL with one of its cipher suites? -Because OTR is seeking to provide additional deniability. Diffie - Hellman key agreement [12] is used to generate shared secrets.
- Perfect Forward Secrecy: The two parties in chat generate new Diffie - Hellman key each time the conversation changes direction [13]. This is to ensure the keys are short lived. Furthermore, as we will see below, both parties in chat securely delete key parameters for previous conversations. Therefore should you lose your private keys, the previous conversations are not compromised.
- Authentication, not by digital signature, but using MAC - Message Authenticity Code. (described clearer in the following.) MAC works like this: we have a MAC key that is shared between the two people in communication. Then now we have a message, we produce the MAC by giving the MAC key and message itself into our selected MAC algorithm, and getting the MAC. Now the other party knows that only Alice could have done this. Bob know that only he and Alice know the MAC key, namely MK; and if her knows that he himself has not created the message, then and only then he can personally come to the conclusion that Alice has.
- Deniability, we will see below how the messages can be forged by other parties without affecting Alice and Bob; so a court cannot prove Bob sent a particular message even if Alice's phone gets compromised. Messages sent and received do not have a digital signature on them, keys change each time the conversation changes direction.

Should OTR be described in steps, it can be pinpointed this way:

1. Step 1: Authenticated Key Exchange: Diffie - Hellman
   (a) Use plain (un-authenticated) Diffie - Hellman to establish a channel. (Since OTR V.2, they even send their public key not in plain text, but encrypting it with the session key.)

(b) Sign a MAC on fresh data to prove your identity and that you know the shared secret.

2. Step 2: Message Transmission. We have the encryption key and MAC key correlated closely in that the MAC key is the hash of the encryption key. This will allow further deniability such that anyone who could read the message could have also forged it in the first hand.

   (a) $EK = Hash(s)$; $MK = Hash(EK)$
   Where s is Shared Key; and EK is Encryption Key.
   Alice sends to Bob the following: $E_{EK}(M)$, $MAC_{MK}(E_{EK}(M))$.
   Encryption here is done using AES in counter mode.

   (b) For the receiver to be able to verify the message has not been changed on the way, the sender uses $SHA1-HMAC$ [14], which gives message authentication using keyed hashing.

   (c) Bob verifies the MAC using MK, decrypts M using EK. Confidentiality and authenticity are assured.

3. Step 3: Re-key

   (a) A sends to B: $g^{x'}$, $MAC_{MK}(g^{x'})$.
   B sends to A: $g^{y'}$, $MAC_{MK}(g^{y'})$.
   A and B calculate: $s' = g^{x'y'}$.

   (b) Compute $EK' = Hash(s')$ and $MK' = Hash(EK')$.

   (c) Alice and Bob both securely erase s, x, y, and EK. Alice's secrecy relies on Bob deleting the key information. IF Bob does not erase the key information, Alice can still deny having sent the message. We will see how this is true in next steps.
   They change these parameter set once every time the conversation changes direction. Perfect forward Secrecy is met this way.

4. Step 4: Publish MK

   (a) Why hide/delete it while they can instead publish the old MK along with the next message?

      i. This will let anyone forge messages;
      ii. And that provides extra deniability for Alice and Bob
      iii. This does not threat Alice and Bob to be attacked by forged messages themselves. This is because they have now moved to MK' and they know that if they receive a message authenticated with MK, it should be a forgery. So Alice's deniability does not

rely on Bob, but it relies on herself publishing her MK at this stage.

We can review how we avoided MITM attacks. But first let us answer the question "how did we initially identify one another?"

How can Alice prove to Bob it is really her?

- Initially the two parties use their long-lived public private keys to authenticate their first DH key exchange. The rest of the DH keys are authenticated by using the previous key, not by the public private key [15].
- But how can Alice authenticate Bob's certificate on the first message? Well, it is assumed she does not. More is needed to avoid man-in-the-middle (MITM) attack:
  - Either they should verify the correctness of one another's fingerprints over another secure channel. To get Bob's fingerprint, you measure the MD5 hash value of his public key in base64-encoding.
  - Second way is to use the Socialist Millionaire's protocol - SMP [16] that we describe below.
  - Note: section 2.2.2 will show how we in this thesis will introduce a third way that makes it easier for our customers in Guidepal without compromising the security or privacy of the users.

With an approach of zero-knowledge proof, the aim of SMP is to enable Alice ensure that the secret she knows is what Bob knows without revealing the shared secret. See the details of the SMP protocol in figure 2.1.

Is is still possible to do MITM attack using SMP? Notice that it is not possible for the attacker to relay the messages any more even if he has managed to do MITM attack until this step, because the secret contains the fingerprints of Alice and Bob. So the only way for the attacker is to know the secret, i.e. social engineering. If they have not chosen a secret already before attempting to do OTR, then they can do it over the line but guiding one another to the secret, without saying the secret. We might say the attacker can succeed by knowing some secret about Alice's life, and persuading Alice to choose that as the secret. This can be avoided better if Alice once lets Bob suggest a question and then redoing it this time Alice asking something that she knows only she and Bob know.

Because it takes the participation of both parties in attempting to check a given answer to the authentication challenge, so brute force attack is not possible on SMP.

Let us look closer how OTR supports deniable authentication?

- By using the symmetric encryption and the keyed Hash in the settings above, we already have deniability. Because anybody who knows the message could have generated the message. Still we publish MK to further strengthen deniability.
- By using a counter mode encryption -AES counter mode- instead of a block mode. After publishing the MK, messages can be changed. So this way others can forge new messages. Once the message is modified, it cannot be authenticated properly.
- By publishing the MAC key after the corresponding encryption key was deleted. Which means an encrypted message could have been generated by anyone, i.e. Alice, Bob or any other person. So having an encrypted message and its MAC does not prove that Alice generated them.

Each user generates a public private key pair when installing OTR on his or her device. The applications that implement OTR save the public key of the verified friends on the device. Users must do key verification each time the user wants to do OTR messaging with a new person, or with an old friend who has installed OTR on a new device.

A little more detail from the OTR protocol description [17]:

OTR Message State machine: There are two state variables for an OTR client: Message state, and Authentication state. It is either MSGSTATE-PLAINTEXT, MSGSTATE-ENCRYPTED or MSGSTATE-FINISHED. In otr4j library, this is a variable called SessionStatus, which exists under net.java.otr4j.session.

The authentication state can be one of the following five:

AUTHSTATE_NONE - the actual initial state. AUTHSTATE_AWAITING_DHKEY After when BOb initiates the authentication protocol which is done by sending his D-H initiation message. In this state, he will have to wait for Alice's reply. AUTHSTATE_AWAITING_REVEALSIG After Alice receives that D-H message, and replies with her own D-H initiation Message; now she will have to wait for Bob's reply. AUTHSTATE_AWAITING_SIG After Bob receives Alice's D-H message, he replies a reveal signature message, enters this state and waits for Alice's message.

SMP State Machine: Prerequisite of this is that OTR state machine is in it MSGSTATE-ENCRYPTED format. We need only one variable to keep track of the SMP state. There are four SMP states: SMPSTATE-EXPECT1, SMPSTATE-EXPECT2, SMPSTATE-EXPECT3, and SMPSTATE-EXPECT4.

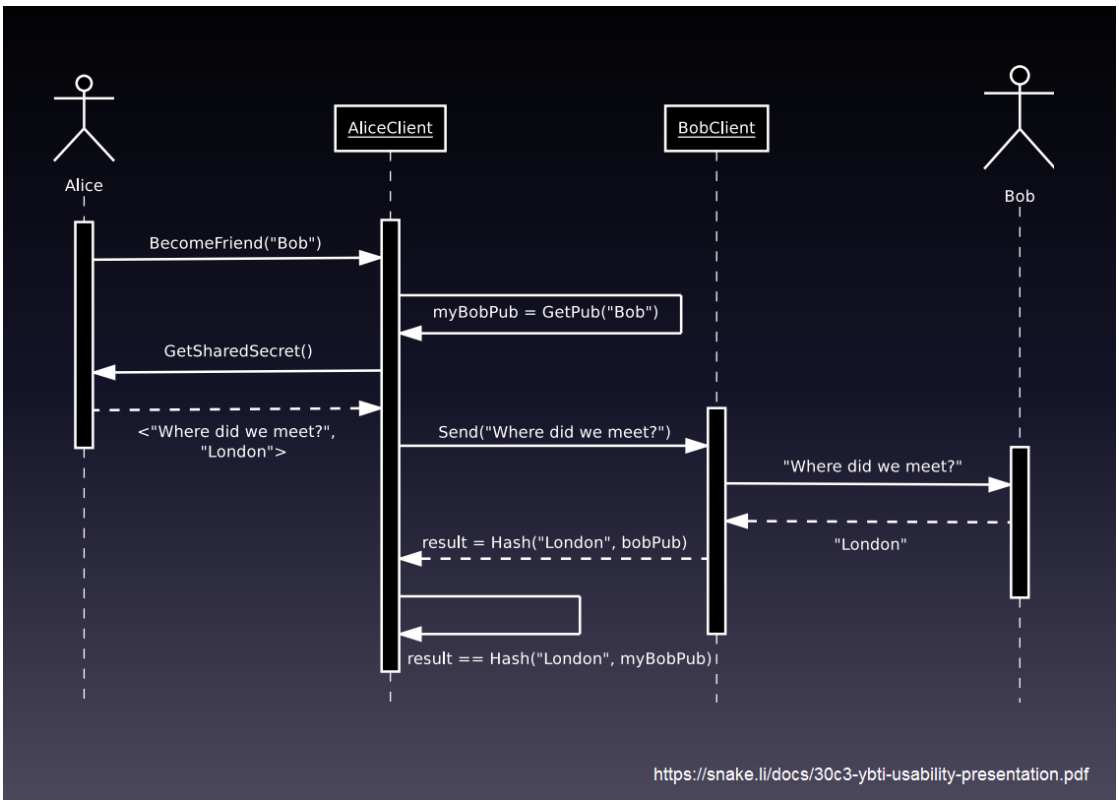

Figure 2.1: Socialist Millionaire Protocol (SMP) diagram, taken from: https://snake.li/

Limitations of the OTR protocol — OTR is not suitable for offline messaging. This is because as pinpointed in the protocol steps, the two parties need to work interactively. Not Alice and Bob, but their software needs to participate in this exchange. In future work, Guidepal can adapt the application and the protocol to enable Alice to send encrypted messages to Bob using Bob's long lived DSA public key. This way we store messages on the server until when they are delivered to the user. The chat application today is such that it stores messages offline on the server anyway. Currently it is not possible to use OTR chat while one of the parties is offline. This is because the protocol requires parties to actively participate in generating keys. It may be that supporting offline messages would be one area where OTR application can be extended to better fit the requirements of Guidepal. This falls into future work.

Using OTR for group chat is CPU extensive, yet there are efforts to support group chat for OTR [18]. However at Guidepal we do not have a requirement today to offer group chat are continuing OTR.

OTR is getting more popular, it is found on some Linux distributions, it is also used on Google Chat and Facebook. When used with Facebook for example, it can be that no User-user authentication is required anymore. The reason is that Bob has already signed into his Facebook account, and that is assumed to be somewhat difficult to break into for hackers, and therefore he has already authenticated himself to OTR and Alice. No doubt there is always chance for profile theft, or similar problems but thinking that Facebook makes it difficult for a hacker to break through by making two step log in available where the user can make it a requirement for login to enter a one-time pin code that is sent to his email address. But our purpose here is to have businesses (that are willing to have IM between themselves and users) verified for their identity and given a certificate trusted by the app. This requirement is not perfectly addressable by the way OTR is offered in Facebook chat for example, because we wish to have businesses registered with us, not to mention that not all people have a Facebook account, and that Facebook has no official check on your identity. So studying the existing work, we will come with design idea in chapter.

In the next section we will discuss how we will extend android Xabber OTR to match the needs in Guidepal.

2.2 DESIGN

Before we move on to finalizing a design for our solution, let us take a look into the adversary model we have.

2.2.1 Adversary Model (Threat Model)

Not thinking about the privacy-aware design behind OTR, one might think that common attacks on an instant messaging application could include: a third party either impersonates the business owner or the customer, or intercepts the messages communicated over the lines and manages to decrypt its contents.

Man in the Middle attack – In case an illegitimate organization gets in the middle of the conversation and acts as if he is Bob, we say man in the middle attack has happened. Registration of Businesses at Guidepal could require that the business negotiate with Guidepal to make themselves a certificate and have it trusted by the chat app.

Traceability – An attacker could trace a user's post that are otherwise allowing only an alias name from him for public. It can be done through social engineering on users who have posts on that venue, or tracing a user based on the information s/he provides on the venue. So no guarantee of non-disclosure of identity can be promised just because a user is using aliases. We know a system's security level is that of its least secure component. This is a typical problem where human is involved in a system.

2.2.2 A Twist to Extend Xabber and OTR

In section 1.3. we discussed the requirements. Customers need to be able to easily authenticate business users. OTR does not authenticate the chat participants from scratch, but it is through a later phase that users authenticate one another by manually verifying the fingerprints or by using some manual challenge/response through SMP. Well, our business customers want to make it easier for users to authenticate them. One way to do this is to introduce the usage of certificates in OTR in such a way that would not disturb the qualities and privacy protection level that OTR offers.

To match our requirements at Guidepal, the OTR and Xabber for Android app are needed to be extended as follows:

In this thesis work, we use the DSA long-lived key pair that a user already generates in OTR, and generate a certificate sign request (CSR). In a separate thread, the client will send his CSR to the server. The server will asynchronously sign it and return a signed certificate to the user. The client hence, also sends his certificate - if he has - to the other client. It must be such that if some user only has the regular standard OTR, the application must still work and they must be able to communicate using the standard OTR.

It is not necessary to have certificates for clients. Because users will not necessarily undertake sensitive tasks that businesses should verify them this way. It is not free to have globally accepted certificates for every end user. So we have two options: first, to have certificates for businesses only. This way, we have server authentication. In case the user needs to authenticate him or herself to the business user, the business user can use the regular SMP and challenge the user with asking for the shared secret. Second: to have certificates for both the businesses and the users.

As we introduce certificates, we need to provide the user with the ability to check certificate revocation as well. To handle this, we have several options, of which two of the major ones are:

- OCSP - Online Certificate Status Protocol, and
- CRL - Certificate Revocation List

As studied by Muñoz et al in [19], CRLs were first meant to help CAs to revoke a previously-signed certificate. But it had some limitations so OCSPA was designed. A CRL typically contains a list of serial numbers that belong to the certificates that have been revoked. This list must be signed by the corresponding CA. Clients are notified where to download a CRL from by looking into the certificate itself and finding the CRL Distribution Point.

But what are the limitations of CRL? - First, CRLs should be updated within several days in between, and if an attack happens in the middle, the clients would probably not know that the CA has revoked that certificate until next time clients would download the CRL. Same would happen if the CRL cannot be downloaded by the client. Second, searching through the list of revoked certificates in a CRL can create an overhead.

What if we go with with OCSP? - First let us quickly review what OCSP is about. It can be a potential replacement for CRL. Here the client can be notified about the revocation of a single certificate. The steps this can take place is: First client sends an OCSP request to the OCSP responder using the serial number

in the desired certificate. Next, the OCSP responder would reply back a signed response of whether or not the Certificate is still valid or not.

OCSP response is smaller than a typical CRL response. This gives it an advantage over CRL. The OCSP responder can be different from the CA. In this case, the CA must be issuer of the OCSP responder's certificate to which the task of OCSP checking and responding is delegated.

But using OCSP for a client introduces a privacy problem in our case of extending the privacy-friendly OTR because our user Alice now has to ask the OCSP server "is this user. Bob, still a valid identity or has his certificate been revoked?" Now OCSP responder has a clue that Alice must have been talking to Bob! Apart from it, since all clients must query the OCSP responder about the status of their desired certificate, there can be a lot of overhead on the OCSP resonpder with all of these requests.

OCSP stapling [20] would be a solution to these problems. The trick in this case would be for the business user to get OCSP response regarding validity of its own certificate in certain intervals and "stamps" it with his own certificate or SSL handshake to the client. Using this would be faster and causing less delay in a normal case. With OCSP itself the client needs to contact DNS to lookup the domain name of the OCSP server of the CA, and establish a connection with the OCSP.

Nevertheless, for the sake of simplicity, we will implement a simple CRL [21] checking functionality into the Xabber app in this phase of development for the sake of this thesis. We will fetch the certificates revocation list from the server once a day, and that way check if the certificate has been revoked or not. In a release version of the app, we will need to consider few more things, like the overhead compared to OCSP.

2.2.3 IM and CA Server

We need a chat server to allow instant messaging in between users. We also would need a server to have our user's certificates signed. For an IM server, the existing solutions seem to be wide and thorough enough so we do not need to implement an IM server from scratch.

For an IM server, we need to be aware of protocols and standards available that provide us with the IM services, i.e. to allow users make an account, be able to search a contact's user-name, show online status - whether online, away, or busy- and write and receive messages, to name a few [22].

Three very common protocols for instant messaging today are Extensible Messaging and Presence Protocol (XMPP), Session Initiation Protocol For Instant Messaging and Presence Leveraging Extensions (SIMPLE), and Internet Relay Chat (IRC). SIMPLE is based Session Initiation Protocol (SIP). SIP which is also used for Simple Mail Transfer Protocol (SMTP) is used for setting up, modifying, and ending communication sessions. SIP can use TCP or UDP or other transport layer protocols which - in case it UDP is used - makes it possible for congestion and message loss. XMPP is one of the most widely adopted open protocol that is used for chat and instant messaging [10, 23]. XMPP has native support for only TCP and includes congestion control by default. Gutiérrez analyzed IM standards in his thesis work [24]. For us this is not a very difficult choice, because OTR does not require server's further support rather than its current IM support.

For the IM server, Prosody, Ejabberd and OpenFire are only some of the servers implementing XMPP chat functionality. Some IM server implementations are open source, some are commercial, each licensed differently. Some of them are available only on Linux or Windows, some on more platforms. We will use OpenFire for a number of reasons. Simply put, it is open source, licensed under Open Source Apache which is not restrictive for us, it is recommended by the OTR users' society, and it can be installed on many platforms i.e. Windows, Linux, Unix, and Mac OS X.

There will be a server to sign users' certificates. This way we empower users with more solution to verify one another's identity - besides other technique available in OTR that is the SMP or verifying fingerprints on a secure channel. Having user's certificate signed in this case requires the use of a Certificate Authority, CA. Having added this node into the network of nodes in the application, we can now use it to sign user's certificate signing requests. More details will be discussed in section 2.2.2 But in general the plan is to have users send the certificate signing requests to the CA server automatically, and the having the CA sign their certificates after identity verification. For using a CA server, we will look into two options:

- Setting up our own simplified CA, and
- Using an already-in-place CA

The benefit of using an already-in-place CA is that we will have no more need to do extensive work to build up a CA on a server. This way we can focus on the application side functionality, and make it work with the external CA. One option is to use a CA like LTCA - Long-Term Certificate Authority - and

the server running it at KTH. This work is described by Khodaei et al. in their paper VESPA - Vehicular Security and Privacy-preserving Architecture [2].

Figure 2.2 shows the information request/response flow between application and a CA Server that would match our need. In case of LTCA; we would also have the user ask for a voucher from the LTCA to use in his certificate sign request [1]. VPKI - vehicular public key infrastructure - requires users to have a voucher in their requests for certification. That is handled in voucher request and response. Users will provide their email addresses and basic data. Assuming that their identity is verified, they will receive a voucher which they will use in next operations. The voucher is user specific and can be used only once. In the case of this thesis, we will not work with LTCA or other in place CAs. The reason why we could not use LTCA is that by Dec 2014, LTCA only supports ECDSA key pair in its certificates. To get the help of LTCA, we needed to change OTR to use ECDSA instead its current DSA key pair.

However, for the limitations we had during this task, we will use a simplified CA on a laptop using the OPENSSL[2] library. In short, the app will generate a CSR, save it on the phone's internal memory. We will then manually read it from the computer, and get it signed by the CA certificate we have on our PC. And then we will sign the CSR and manually send it back to the user's phone. We will also have a CRL manually saved on the user phone to demonstrate that we check the CRL and that we can verify validity of the certificate. More detail about this will come in the implementation section 2.3. The future work section 3.1 will suggest some options for Guidepal to address its need for a CA should this extended OTR app (namely Guidepal-IM) go live.

[1] Vehicular Security and Privacy-preserving Architecture and application programming interface is viewable at https://code.google.com/p/kth-vespa/.

[2] OPENSSL implements SSL and TLS protocols together with some cryptographic functions for creating certificates, certificate signing requests, as well as for signing and revoking the Certificates https://www.openssl.org/.

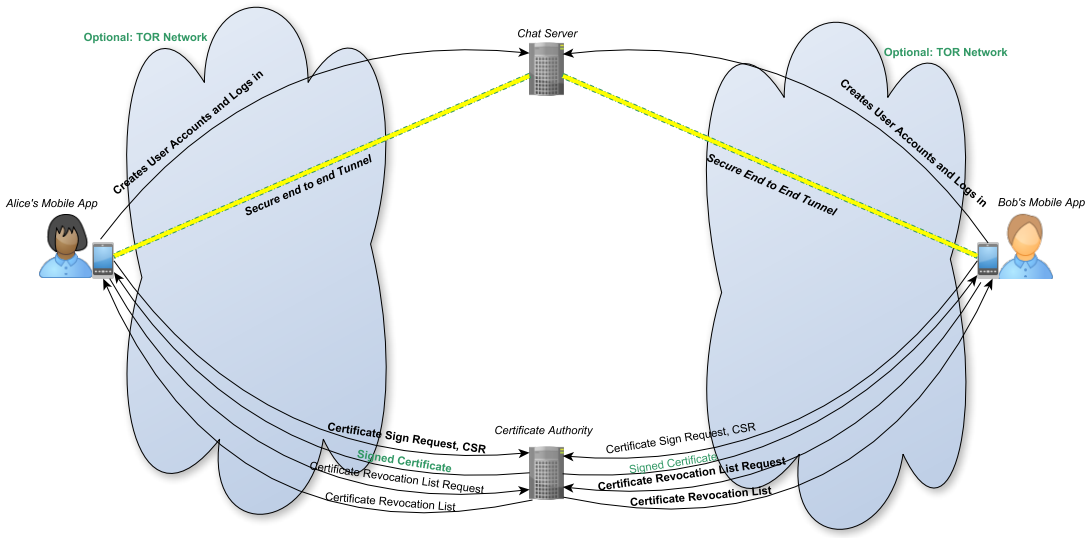

Figure 2.2: Our secure IM solution for Guidepal. Calls to Guidepal IM server are synchronous, while calls to CA are asynchronous. The CA in a real scenario might actually need to perform some background checking on the identity of the certificate request applicant.

Although we deployed a very simple CA for this thesis, but yet we studied how a real CA could be designed for Guidepal. Inspired by LTCA, here are the details:

A. Voucher Request and Response Messages:

As described above, attaining a voucher would actually be part of registration process that we can have for the users wishing to get a certificate.

Voucher Request - Client $\longrightarrow$ Server: This will include:

- Message Type, say C1
- User Name
- Email Address
- Nonce (Protects against Replay attack e.g.)
- Timestamp (Protects against DOS attack e.g.)

Voucher Response - Server $\longrightarrow$ Client:

- Message Type, say S1

- Voucher String
- Nonce
- Timestamp

B. X509 Certification Request and Response Messages:

Client generates certificate sign request. He then sends the certificate sign request to the server.

X509 Certification Request - Client $\longrightarrow$ Server:

- Message Type, say C2
- A Certificate Sign request (encoded in a .PEM file)
- Timestamp
- Nonce

X509 Certification Response - Server $\longrightarrow$ Client:

- Message Type, say S2
- An X509 certificate
- CA's certificate - and if applicable, the certificate chain
- Timestamp
- Nonce Reply
- This whole thing should be signed by CA's private key, named Signature.

Client saves the signed certificate on the phone memory. Alice uses the certificate when chatting with client Bob in that she sends it to her. Bob then verifies that the signature is signed by the correct CA certificate that he knows about as it is embedded in his app.

C. CRL Request, and Response:

CRL Request - Client $\longrightarrow$ Server:

- Message Type, say C3

- CRL Request
- Timestamp
- Nonce

CRL Response - Server $\longrightarrow$ Client:

- Message Type, say S3
- CRL
- Timestamp
- Nonce

2.2.4 | Implications of the Choices

Apart from the recommendations in section 3.1.1, having made the decisions about which applications and solutions we use, there are a number of points that Guidepal needs to consider:

- Certificate Authority (CA). Guidepal needs to develop its own CA. Today we have an over-simplified CA on a PC where we simply sign and revoke user certificates using OPENSSL. Should Guidepal-IM be released, Guidepal needs a real CA. However, that CA does not need to be certified by the CAs trusted by Android platform because it is already made such that the Guidepal's certificate is hard-coded and trusted by the Guidepal-IM android app.
- *As long as you are hiding it, you cannot prove it!* You can only make promises that you are not doing anything malicious. We will call the application developed in this thesis Guidepal-IM. Guidepal-IM is open source[1], but Guidepal app itself is not. As of today, there is no plan to publish Guidepal source code to public. Problem of trust is not negligible if you do not reveal the source code and can only claim e.g. '... we do not act against what we claimed regarding protection of our user privacy'. We know Guidepal-IM is developed to fulfill the security and privacy requirements; therefore it would be verifiable only when the code is released. Guidepal should

[1] Guidepal-IM sorce code is available under GNU General Public License at https://bitbucket.org/alirezaabdi/xabber-otr-certificate-edition.

consider how it is going to integrate the IM solution into the Guidepal application. In this thesis we assume that these apps (Guidepal and the IM service) are separate. Later on Guidepal can link them together to work smoothly. Users download Guidepal and Guidepal-IM apps separately.

- Guidepal is interested in both stand-alone IM solutions and an IM feature added to the Guidepal app. We must give caution regarding integrating the IM application and the privacy solutions. OTR and Guidepal-IM allows having traffic pass through TOR network by providing user with configuration options that allow them to set proxy settings to work with applications like Orbot in Android [25] or equivalent for iPhone mobiles. As we want to integrate OTR Jabber with Guidepal, Guidepal could make sure they will allow Guidepal users to set connection proxy settings so that users could benefit using TOR if they wish. However it needs further analysis if linkability is still possible if users are connecting with their Facebook account.

2.3 IMPLEMENTATION

Chapter 2.2.3 described how we will use a CA server. Figure 2.2 showed in some detail the message flow in between the users, the OpenFire Server and the CA. Below we will show what is sent and received in between a client and the CA in more detail.

Generating the Certificate and getting it work on the app

1. Generate a simple CA (self signed) on the pc. This was done using OPENSSL library in Linux terminal commands.
2. Install this CA cert on the phone so it can verify user's certificate that are signed by this CA in next steps. This is done by putting the CA certificate in the *res/raw* folder in the Guidepal-IM android application source code and later by using the Spongycastle libraries for android[26].
3. Produce a certificate sign req on the phone using OTR key pair, i.e. DSA keypair.
4. Store that CSR on the phone's internal memory.
5. Manually send the CSR from the phone over to the PC. This is done using adb tool with $ *adb pull* command.

6. Sign the CSR with the CA certificate. This is also done using OPENSSL library via the Linux terminal commands.
7. Save the it back on the phone's internal memory as well. This is done using adb tool with $ *adb push* command. We then read the signed certificate by the app.
8. Verify that the certificate was signed by the trusted CA. This is done using the Spongycastle libraries for android.
9. Notify user. We needed to add a new trust status and a new action to OTR library and link it again to the Guidepal-IM app. The new OTR trust status is called: otr_cert_verified and the new action is called action_otr_cert_verified." This is in Git commit # e7fa9ec2 of our source code repository. The user is notified on his chat view that the opponent's certificate is signed by the trusted CA or not.

Generating the CRL and checking certificate revocation status

1. Generate CRL on the pc. This was done using OPENSSL library in Linux terminal commands. Creating a CRL requires a CA certificate because it will be signed by the CA certificate. As mentioned earlier, since CRL distribution point can be different from the CA. However in our case the certificate revocation list signer is the same CA for simplicity.
2. Save the CRL on the phone's internal memory and read it from the computer. This is done using adb tool with $ *adb push* command.
3. Verify that the CRL was generated by the trusted CA. This is also done by using the Spongycastle libraries for android.
4. Notify user. The user is notified on his chat view that the opponent's certificate is checked against CRL and we verify that the certificate seems to be valid. Otherwise the user will be asked not to trust the party who possesses a revoked certificate.

2.4 RESULTS

To verify that the idea works and users can still chat with one another using the previous OTR functions, we installed three different instances of the same Guidepal-IM app on the same phone, which was a Sony Xperia c5503 with Android version 4.4.4 running on it.

Installing different instances of the same application on Android is possible by changing the application's package name in the AndroidManifest.xml file and in our case by changing the value of "development_title" in the *"res/values/-market.xml"* file of the app.

We created two XMPP IM accounts on each of the three instances of the Guidepal-IM app which would login to our IM server's information.

We verified that

- all users can add one another as contacts,
- accept or block these contact requests,
- could change presence status (i.e. budy, away, online, etc.) successfully,
- can send instant messages to each other,
- could do the regular standard OTR operations that they could do before our changes,
- could have the CA certificate embedded in the app and be able to read it,
- could read the opponent's certificate that we manually sent to the phone's memory
- could verify that the opponent's certificate was actually signed by the CA
- the app expectedly failed verifying one of the opponents whose certificate was not signed by the trusted CA
- could read the CRL stored manually on the phone's memory
- could verify either cases if the opponent's certificate was or was not one of the revoked certificates or not, and let the user know about the verification result by printing a sentence on the chat view.

Since generating a CSR only occurs once in the lifetime of an installed application, it appears to be sufficient to provide a little data on the time duration that it takes to do the following steps. Note that the measurements are carried out on a Sony Xperia ZR C5503 phone, that runs Android 4.4.4; a Quad-core 1.5 GHz Krait CPU, and 8 GB Intenral Ram.

- CSR generation - took between 3.293090823 and 6.608032229 seconds in 10 runs. This is done on a separate thread, and is not noticeable for the user at all. The GUI works quite smoothly while this is taking place.

- certificate signature verification: took between 0.044555663 up to 0.120539710 seconds in 50 runs.
- verification against the CRL: took between 0.010474609 up to 0.053710938 seconds in 50 runs.

3 CONTRIBUTION AND CONCLUSION

Why introduce a privacy study at Guidepal? The company is growing since its establishment; and it wants to raise its awareness in this area of privacy, and is seeking to make sure its app and services are offering specific security and privacy services. We think the study achieved its primary purpose.

Earlier in this work, we mentioned few problems that we are facing regarding user privacy. Much of the improvements are to give users more control over their data. Although it is arguably necessary to introduce new effective mechanisms to enforce the security policy for internet services, it is amazing to notice the huge IT businessed do not offer services and users are not given reasonably enough control over their data. E.g. Google, Facebook, Amazon Web Services, Dropbox, WordPress are just to name some of the websites that have implemented a two step verification/authentication Two step authentication/verification is where the user is proposed to register his mobile phone number so that he receives a verification code on his mobile phone whenever he is logging in from a new device. This is a provably good mechanism able to help user privacy and avoid a number of attacks where an attacker accesses user's information on a website using the correct username and password, for example. There have been several other mechanisms to help user security on accessing their data and internet services. But why not offering a service for example where users who wish to have substantial more control over their data could pay some money and instead could e.g. have their Google Drive encrypted end to end such that Google itself would not be able to know what the information really is about. That way even the US government would gain not much if all it can force Google to reveal about user x is just some gibberish.

To help illustrating the meaning of information exposure that is popular today, let us use this simile:

Imagine that your meeting with your closest friend on a sunny day in a park is recorded and accessible and readily available to police, and the Intelligence agencies. It is as if there is a tape recorder or an HD video camera on several angels that could cover whatever you tell one another. The police claims they will record this; but we will not watch this, unless if law indicates that it should be checked for court investigations or what-we-call-sensitive decisions about you in certain situations. To many people, living

a normal life, never being charged for unlawful activities, this is simply an unacceptable, intolerable invasion of privacy.

People normally do not tell everything to everybody; they have subconscious classifications and levels that are important to them. We witness movements among users and organisations to begin claiming some rights of internet users over their personal data.

Looking at today's data protection policies enforced by large Internet corporations such as Google, Facebook, they do record users' data, for it is demanded by the nature of the service, but they do not at all advocate people using encryption mechanisms for communication. For instance, Google Mail should save emails of its users, for the user to look back to his mail archive. Google's privacy policy [27] indicates, in some cases, the data can be exposed to outside Google for "legal reasons." In US, government agencies can relatively easily ask for a copy of a user's private data from email or other internet services' providers based on the Electronic Communications Privacy Act (ECPA) [28]. Based on this law, it does not even require a warrant but only a subpoena for the governmental agency to ask for a copy of a user's stored data and communications [29]. The point, intended here, is that we can argue it should be possible and convenient for a single user to store his data or communication in a form that the service host itself and police agencies cannot understand, like old days where you could meet someone at a park in person and make sure you are only talking to him. Note that confidentiality is different from non-repudiation of origin or other security services. For deeper understanding of the security services, one can refer to literature on security such as Computer Security [30], or Introduction to computer security [31]. Arguably, it is the general public's understanding, behaviour and expectations that shapes the future of the law and practice. An example is acceptability of execution that has changed over the time by the change in the culture in many nations [32].

3.1 FUTURE WORK

There are a number of areas we think would worth considering. Some are just optional, some should be considered more seriously. Below is a list of these recommendations for future work that could not be included further in this thesis study.

- Have a robust CA at Guidepal.
  Setting up a CA for our own — As discussed in chapter [], there are several

CA solutions available. We used that of KTH VeSPA CA for the duration of this study work. Guidepal should use another CA after this CA should it wish to carry on implementing the functionalities proposed in this thesis. Some CAs can be installed on our own server. Not all CA solutions are open source of course. Nicusor Vatra et al ends up comparing OpenCA and EJBCA to propose a PKI for Romania's e-government services [33]. A commonly used open source CA is OPENCA. Both these servers offer most functionalities you would expect a CA be able to do. Nicusor Vatra et al use EJBCA in their proposition for it is more scalable. We study using OpenCA for Guidepal's case.

1. Business users generate and send certificate sign request to CA using the voucher they got.
2. The CA admin observes, authenticates and signs business users' certificates.
   The signed cert will be sent back to the business user's mobile device to be used and stored.
3. Users will get CRLs maximum once a day (when they start using the Guidepal app the first time during the day).

Guidepal will implement a simple CA server for addressing the basic needs of this extension. It is also slightly inspired by VESPA.

- Have a Registration Authority (RA) by which the identity of the business user gets approved.
- Extend both Guidepal-jabber and Guidepal apps to enable Guidepal-jabber fetch contacts from user's Guidepal app on the phone. This way user does not have to memorize jabber-id of the business. To make it more user-friendly, there should be a GUI in Guidepal-jabber with contacts and these contacts need to be transferred securely. This is important if Jabber is going to be integrated with Guidepal-jabber.
- There would be some work to extend the chat application to have secure offline chat. Jabber currently offers online secure chat only.
- Support Certificate Extension functionality. That way users can ask for their certificate be reissued. This would match our business model better where business users have to renew their agreement with us.
- It is necessary that the user checks its own certificate's validity too.
- It is necessary for the CA to have the ability to push revocation notification to all users. Imagine a travel agency's certificate gets compromised, and

the intruder can send in false information or bring much damage to many customers. It is therefore needed for the server to be able to push this notification to all chat users.

- Messages with some businesses must be saved on the mobile device. User must in all cases be able to choose if the certificate can be saved or not. However business users should be able to define a default setting that is also going to be done on the user's phone. A business that has to do with traveling for example can let the default message-saving preference be such that messages be saved on the user's phone. Example necessity is when you chat with your travel agency and get some information that is important to remember along your trip. It would not be nice if the next time you log into your chat app, you see all those data is gone because OTR deleted it. Another business, say a human right watchtower would like to set the default be such that messages do not be saved on user's phone.
  If messages need to be saved, they would better be saved using some other mechanism other than the encryption we use in OTR, for OTR changes keys with every change in the discussion's direction. More study should be done on how to securely save messages. This would require an extension to OTR.

3.1.1 Recommendations for Guidepals Social App

There are several ways to Give users privacy control on their data and still meeting the business plan.

Venues and tips — Tips on a venue are generally public, there is nothing such as private tips on a venue. Note that nobody owns a venue. A venue about a business includes no such information that you can find on its commercial website, but it consists of the address, a picture and an embedded map; the rest are the users' comments. If you post a tip on a venue, it will be visible for all. So, the data will be stored on and fetched from Guidepal servers. This functionality is already implemented and working in Guidepal project. An example of a venue can be McDonald's on Sveavägen 71, Stockholm.

Guides and comments — Users can create Guides, each consisting of one or more venues. One can leave a comment, i.e. tips, on a guide on different guides. These tips on guides can be private. An example is making a guide on: My favorite hangouts for restaurants. Here you add a number of restaurant venues and can optionally leave comments related to each of them, maybe giving tips to your friends or using them as reminders to be used personally later. Today a

guide can only be public. To give users more control over their data, Guidepal can consider to let a guide be private or public, based on the user's choice. If it is private, the user can then add friends to be able to view his guide. You as a user can follow guides that others make and are visible to you. Maybe some of your friends will follow you on your guide. What are the different ways to implement this?

1. To store guides and including comments on Guidepal server without encryption.
2. To provide the mechanism for the user to encrypt and decrypt data behind his guides and yet storing them on a Guidepal server for backups and restoring ease. Having this in place, it would make it more difficult to share a guide with another person, because the idea of protecting them with password was to disable Guidepal be able to read the content of a personal guide. This means the user, the owner of the guide, who encrypts the message, must store the password on his device and send it over to the friend who will have access to this guide. Technical complexity will be hidden from the user, instead merely letting him choose with whom he wants to share the guide. Again tricky part would be that still it is through Guidepal app that this the guides are shown, and since the app is not open-source, it makes little sense to implement this. It would add much more value in case we were open-source.

Since users can choose to create a new account on Guidepal only providing his email address or simply using his Facebook account to login, there is a trade off for the user: either to choose to login with facebook and immediately find friends' as one's guides and use one's existing profile picture and login information; or to create a new account, enter login information manually and find guides from scratch. Providing users with this option is very common among SNAs. It gives slightly better control to users, either to risk their privacy by logging in by their SNS account or not.

Other Recommendations:

- To give users more control over their data, Guidepal can consider to let a guide be private or public, based on the user's choice.
- Enable connecting with TOR for Guidepal users. Like Jabber and Chatsecure application, Guidepal can also give users of its apps the ability to use the TOR network by setting up the proxy settings.
- Enable choosing to have numerous pseudonyms for users when leaving a post or making a guide.

- In case the company decides to integrate the IM feature with its social media app app, it needs to study on how to do it. One solution is to keep Guidepal-IM app open source and just make the current Guidepal app work smoothly with it to enable users and businesses message securely.

A PRIVACY IN SOCIAL NETWORKING APPLICATIONS (SNAs)

The whole idea of this chapter is to see what are the ways to give users more control over their data. The better we are at providing this together with transparency, the more privacy friendly it would be. It is as simple as that. Depending on the nature of a project and its business plan, some solutions for privacy control might be feasible, some not.

Part two of this work is setting the Ground for Further Studies at Guidepal. The purpose of this chapter is to study "a third party app's role in the privacy issues of Social networking applications (SNA), and how to help protecting user's privacy."

A.1 SNA – SNS ARCHITECTURE MODEL

Guidepal is an SNA, benefiting from having users make an account with us trough their facebook account so we can download user's contact list. We therefore will have a look at the architecture first.
Facebook Login Architecture — Figure 1.1 depicted the data flow in current architecture of facebook and apps connecting to it. As a user grants permissions to an application on his mobile device, the user will be given an access token. This token is specific to the application. It has, embedded in it, information about the user, the app, the access token expiration date, and the granted permissions. The API calls to Facebook are mostly required to be signed by this access token for privacy protection. The access tokens are generated using authorization protocol OAuth 2 [34]. It is used to do API calls on behalf of the user [35].

Much of the problem is that the more you share your identity with others, the more hard-to-reach would the dream of anonymity become. Now let us take a look at our product at guidepal. Then we will talk about the problems with the current widespread models.

A.2 A DISCUSSION ON POSSIBLE SOLUTIONS

Maybe we should not have saved user data in readable form in the first place. Imagine a society where the poor get more miserable and the rich to gets richer; now that society claims we do this so that the rich can get powerful enough to help the poor. It resembles our privacy issues here. The moment we we allow ourselves to understand user private data, we have degraded privacy substantially.

By studying the current researches over aforementioned user privacy issues, we can observe the implications of having a different model of information storage and flow on privacy protection than exposing SNA user's data to third party app. Some have suggested more secure architecture for social networking, such as Pidder [1]. But one cannot neglect the cost for users to switch to yet a newer SNS, and e.g. deactivate one's facebook account. Although software technologies replace one another from time to time sooner or later, still incurred costs for the user could be psychological, time for learning the new product, losing many friends who would want to stick to Facebook, etc.

Facebook can read your information. It gives your personal data to intelligence services in US upon request. At the same time it protects your data for you against others. You have no choice or power over what information you want facebook be able to understand or expose to government. Well, the story is a little different with Pidder. Pidder's motto is "Your data, Your Control!". It uses strong encryption that is beyond the focus of this thesis work. One question is can SNAs use the data from secure social media (e.g. Pidder) in any meaningful way? Or is the security and privacy protection measures just a motivation-killer for an SNA developers to mashup with them enough? This is a very serious question. With Facebook, apps can easily work with it, read your account data from Facebook. You have a greater potential to get connected with others if you log in by your facebook account. This fact is motivating for SNA companies.

In the remainder of this article we focus a little more on facebook and see if we can help user privacy even if facebook does not change to disable itself from understanding user data. Next section A.2.2 examines an architecture that would introduce new parties to the scenario. Applications vary much in their targets, and business plans. Some do require users be able to recognize one another more publicly, some do not. Needless to say, it is important to consider scalability and acceptability of the new design proposals for SNS-SNA network,

[1] Pidder -short for private identities demand encrypted resources - is available at https://www.pidder.com/en/index.html.

the developer community and users. Thus in next section we partly discuss and analyze the implications of implementing one possible architecture, and we also try to evaluate if it can introduce any worthwhile added value. We also look into network model and few other issues.

A.2.1 Not Exposing Plaintext Data to Third Party App Servers

In previous sections we discussed our main concern, being that user data is now exposed to several parties other than the social media DBs and servers, i.e. third party SNAs.

The Network Model and Limitations — Does it help user's privacy to get SNAs built in peer to peer network model?
Guidepal is running on a client server network model. We have a server in the middle that stores all data. Although users' mobile phone save some data, yet it does not change anything in this context. But what if we had everything on a peer-to-peer basis? In a peer to peer network model, a node can be both client and server. Observing the ongoing trend in the connected devices, we can note challenges on the way for peer to peer networks. Today 15 percent of data traffic is produced by mobiles. First and foremost, it is interesting to study if having P2P network solves anything for SNAs or SNSs by itself, like the way helps applications like skype. One thing is obvious though, unless if there are open source projects proving that users have full control over their data, there will be no help to user privacy. Second, we must see if it fits into the big picture. P2P networks are not suitable for all projects. For example businesses with sensitive data handling might not be particularly interested in deploying in P2P. Third, the technology for client server is already in place, the IT crowd has far more competence in client server, rather than P2P which are considerably more complex.

Nevertheless we see peer to peer social media being researched and developed in projects like PeerSoN. Developed mainly at the Royal institute of Technology in Sweden, PeerSoN aims at filling some gaps in the area of decentralized social media, while seriously considering privacy issues. Its purpose if to give users control over the data, however it yet has open questions and ongoing research. They stress the fact that deploying the social network on a P2P network model does not solve privacy issues, but introduce new threats that need to be studied and mitigated [36]. Not all peer to peer networks have privacy concerns in focus. 6Search for example, is a collaborative peer network application studied by Wu et. al [37].

We also focus on dealing with the existing network model of Facebook that

knows everything about 800 million users as of today and see if struggles can get us anywhere with our concerns for user privacy.

Due to the nature of this thesis work at the company, the discussions given at this chapter are looked at as a pre-study, something of theoretical value, based on which future development might be affected.

A.2.2 Introducing an Intermediary Trusted Server

Currently the approach that Facebook uses for getting users to allow or disallow a certain app is that Facebook prompts the user if he agrees with the set of permissions to be given to that particular app. Facebook returns to the mobile client application a token that is valid for few hours [38]. The client sends the token to the SNA company (3rd party application server) The server forwards it to Facebook, and gets the desired user's data in return. This way the app company can choose to legally download the information of the user on their own server. This exponentially adds to security concerns, not to mention delivering part of user's data to advertisement and data mining companies.

One raw idea is to simulate and implement a server, that is supposedly working in between Facebook as a social networking server and the third party SNA company. Facebook or some to-be-trusted party could control this server, and the app company can rent it; and this way run its API and queries on the server. The server is only partially controlled by the company, in that the app company cannot view the user information in plain text.

Yet because social networking websites like Facebook are largely used today, a solution is needed to help these cases for when a third party SNA wants to connect to one's social network such as Facebook. The idea suggested in this paragraph suggests Facebook does not give the app company the understandable form of user's information. The user can see in his account settings what information the company has downloaded about him/her. S/he can then even remove those permissions that he gave once to that specific app. The app company can have queries sent to this server after the user gives it the permissions; and the mobile application that the user downloads can be talking to the user. This is a raw idea, and should be studied if it is applicable at all.

Figure A.1 illustrates at a high level the idea of having an intermediate server. The picture depicts the idea of having the intermediary server in between 3rd party app server and Facebook. We discuss if this intermediary server can be scalable.

In this mechanism, the 3rd party SNA company deploys its API on the

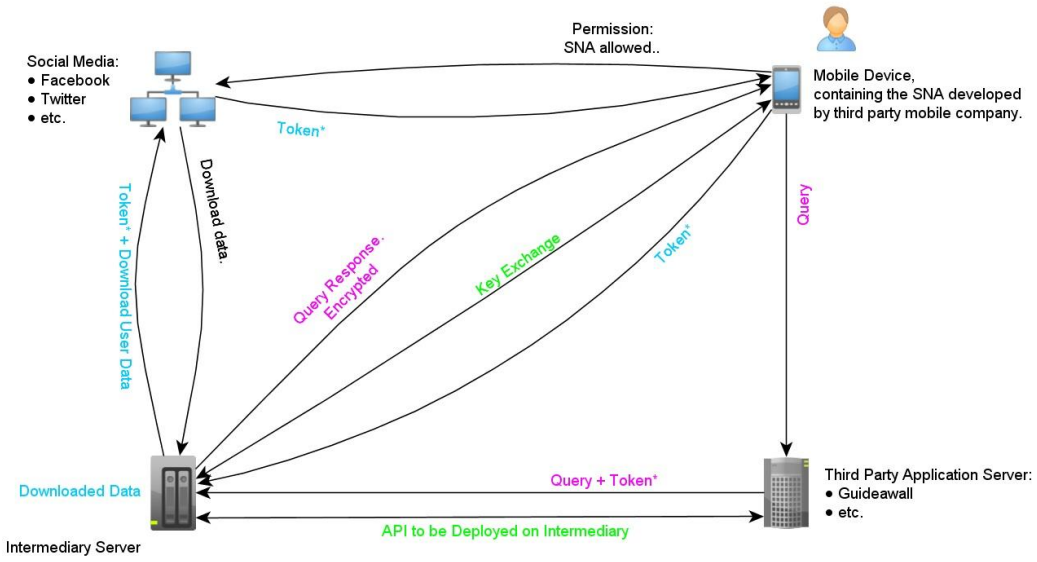

Figure A.1: A twisted architecture of data flow, changed due to the idea of introducing an intermediary server.

intermediary point (shown in green color at the bottom). The user having downloaded the software, exchanges keys with the intermediary server (shown in the picture in green color). The user logs in facebook and gives permission to this particular SNA, and gets a token in return. It then forwards it to the SNA server. SNA server forwards it to the intermediary server, where this server gets and downloads data from SNS, e.g. facebook. The intermediary server then is ready to run the user's queries and encrypt them with the key that is only known to itself and the user.

How practically more private is this? Obviously we can merely get a false sense of privacy protection since we have the user data stay still understandable and readable for Facebook. And Facebook should and does share user data with government, no need for the government to prove to Facebook why it is it is interested in user's information. What more this approach suffers from is the need for the intermediary server to verify the trustworthiness of the 3rd party's API, and the mobile app that the third party has on user's phones to make sure it does not act maliciously. This wouldn't sell, does not justify its purpose, and adds to bureaucracy and complexity at an unacceptable manner. So no need to see what key exchange mechanisms one could use in this scenario.

A.2.3 | Querying Encrypted Data

This part aims at studying the applicability and implications of having the social media encrypt user's data, and then send it to the third party server without having the third party being able to know all the data in plain text. In other words, we could assume that Facebook sends user's data in encrypted form over to the third party's application server. This brings question: how can encrypted data be used by the third party app server? An answer can be using methods to query encrypted data [39]. Yet we have the SNS, Facebook save our data in a way it can read it and expose it, and therefore it is difficult to buy this idea in the first place.

Knowing that many applications do some query or data mining over data in their backend server and API, it does not work in case data is unreadable for the third party application's server. This means the user should fetch the related information from the server on his device, decrypt it with the keys that will be sharing with Facebook, and then query the data himself and show desired results on the screen to the user. This brings a great unacceptable overhead, knowing that we are having users connecting on their mobile phones with limited memory, processing power and battery.

One way to solve this and let some applications be used in an well-encrypted social media network is to have that SNA software operate on a safe environment. The SNA in this case must be a trusted application (and this would mean open source and reviewed) and must not be able to reveal any information on the user. This is to the best of my knowledge today rather hard to monitoring because each time an application needs to be updated, there is a risk of unintentional or intentional wrong behavior of the software compromising users' privacy assumptions. Moreover, having applications open source will not match the case of many businesses. In the end, it seems there are apps that would operate well on a users' plain text data much better. More discussion and studies is required to provide thorough answer to this question: How relevant would a secure, privacy-friendly SNS be to the many applications we have? How much would applications be applicable to a privacy-friendly social network like pidder? We avoid conducting a survey in realizing what users prefer privacy friendly social media or carrying on using Facebook and their own accounts, as it is not particularly the focus of this thesis at Guidepal.

BIBLIOGRAPHY

[1] "State-sponsored terrorism," [Accessed 10-January-2015]. [Online]. Available: http://en.wikipedia.org/wiki/State-sponsored_terrorism

[2] M. Khodaei, H. Jin, and P. Papadimitratos, "Towards deploying a scalable & robust vehicular identity and credential management infrastructure," in *Vehicular Networking Conference (VNC), 2014 IEEE*. IEEE, 2014, pp. 33–40.

[3] Facebook, "Facebook platform policy," [Accessed 10-January-2015]. [Online]. Available: https://developers.facebook.com/policy/

[4] A. Beach, M. Gartrell, and R. Han, "Solutions to security and privacy issues in mobile social networking," in *Computational Science and Engineering, 2009. CSE'09. International Conference on*, vol. 4. IEEE, 2009, pp. 1036–1042.

[5] F. Beato, M. Kohlweiss, and K. Wouters, "Scramble! your social network data," in *Privacy Enhancing Technologies*. Springer, 2011, pp. 211–225.

[6] "Primelife, a research project funded by the european commission's 7th framework programme," [Accessed 10-January-2015]. [Online]. Available: http://primelife.ercim.eu/

[7] B. Krishnamurthy and C. E. Wills, "Characterizing privacy in online social networks," in *Proceedings of the first workshop on Online social networks*. ACM, 2008, pp. 37–42.

[8] "Interview with kurt sauer "phone calls over the internet: How secure is skype really?"," 2007, [In German, Accessed 10-January-2015]. [Online]. Available: http://www.zdnet.de/39151472/telefonieren-uebers-internet-wie-sicher-ist-skype-wirklich/

[9] W. Bleh, "Skype calls untouchable?" 2007, [In German, Accessed 10-January-2015]. [Online]. Available: http://www.intern.de/archiv/news/internet-news/200711232857.html

[10] P. Saint-Andre, "Extensible messaging and presence protocol (xmpp): Core," 2011.

[11] C. Alexander and I. Goldberg, "Improved user authentication in off-the-record messaging," in *Proceedings of the 2007 ACM workshop on Privacy in electronic society*. ACM, 2007, pp. 41–47.

[12] W. Diffie and M. E. Hellman, "New directions in cryptography," *Information Theory, IEEE Transactions on*, vol. 22, no. 6, pp. 644–654, 1976.

[13] N. Borisov, I. Goldberg, and E. Brewer, "Off-the-record communication, or, why not to use pgp," in *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*. ACM, 2004, pp. 77–84.

[14] H. Krawczyk, M. Bellare, and R. Canetti, "Rfc 2104: Hmac: Keyed-hashing for message authentication," 1997.

[15] R. Stedman, K. Yoshida, and I. Goldberg, "A user study of off-the-record messaging," in *Proceedings of the 4th symposium on Usable privacy and security*. ACM, 2008, pp. 95–104.

[16] F. Boudot, B. Schoenmakers, and J. Traore, "A fair and efficient solution to the socialist millionaires' problem," *Discrete Applied Mathematics*, vol. 111, no. 1, pp. 23–36, 2001.

[17] C. OTR, "Off-the-record messaging protocol version 3," [Accessed 10-January-2015]. [Online]. Available: https://otr.cypherpunks.ca/Protocol-v3-4.0.0.html

[18] J. Bian, R. Seker, and U. Topaloglu, "Off-the-record instant messaging for group conversation," in *Information Reuse and Integration, 2007. IRI 2007. IEEE International Conference on*. IEEE, 2007, pp. 79–84.

[19] J. L. Muñoz, J. Forné, and J. C. Castro, "Evaluation of certificate revocation policies: Ocsp vs. overissued-crl," in *2012 23rd International Workshop on Database and Expert Systems Applications*. IEEE Computer Society, 2002, pp. 511–511.

[20] P. Hallam-Baker, "X. 509v3 extension: Ocsp stapling required," 2012.

[21] D. Cooper, "Internet x. 509 public key infrastructure certificate and certificate revocation list (crl) profile," 2008.

[22] M. Mannan and P. C. van Oorschot, "Secure public instant messaging: A survey," *Proceedings of Privacy, Security and Trust*, 2004.

[23] R. B. Jennings, E. M. Nahum, D. P. Olshefski, D. Saha, Z.-Y. Shae, and C. Waters, "A study of internet instant messaging and chat protocols," *Network, IEEE*, vol. 20, no. 4, pp. 16–21, 2006.

[24] A. G. Gutiérrez, "Instant messaging and presence services: Analysis of the standards and example implementation," *Technische Universitat Hamburg-Harburg*, 2004.

[25] "Orbot: The tor project for android users," [Accessed 10-January-2015]. [Online]. Available: https://guardianproject.info/apps/orbot/

[26] S. Castle, "A repackage of bouncy castle for android," [Accessed 10-January-2015]. [Online]. Available: http://rtyley.github.io/spongycastle/

[27] Google, "Google's privacy policy," [Accessed 10-January-2015]. [Online]. Available: http://www.google.com/intl/en/policies/privacy/

[28] T. U. S. D. of Justice, "Electronic communications privacy act of 1986," [Accessed 10-January-2015]. [Online]. Available: http://www.justice.gov/jmd/electronic-communications-privacy-act-1986-pl-99-508

[29] E. P. I. Center, "Electronic communications privacy act (ecpa)," [Accessed 10-January-2015]. [Online]. Available: https://epic.org/privacy/ecpa/

[30] W. Stallings and L. Brown, *Computer Security*. Pearson Education, 2008, no. s 304.

[31] M. Bishop, *Introduction to computer security*. Addison-Wesley Professional, 2004.

[32] A. International, "Abolitionist and retentionist countries," [Accessed 10-January-2015]. [Online]. Available: http://www.amnesty.org/en/death-penalty/abolitionist-and-retentionist-countries

[33] N. VATRA, "A pki architecture using open source software for e-government services in romania," *Indian J. Comput. Sci. Eng*, vol. 2, no. 4, pp. 532–538, 2011.

[34] "The oauth 2.0 authorization framework draft-ietf-oauth-v2-31," [Accessed 10-January-2015]. [Online]. Available: https://tools.ietf.org/html/draft-ietf-oauth-v2-31

[35] Facebook, "Facebook login architecture," [Accessed 10-January-2015]. [Online]. Available: https://developers.facebook.com/docs/facebook-login/overview/v2.2

[36] B. Greschbach and S. Buchegger, "Friendly surveillance–a new adversary model for privacy in decentralized online social networks," in *Current Issues in IT Security 2012, 5th interdisciplinary Conference, Freiburg, Germany, May 08-10, 2012. Proceedings*. Citeseer, 2012, pp. 195–206.

[37] L.-S. Wu, R. Akavipat, A. G. Maguitman, and F. Menczer, "Adaptive peer-to-peer social networks for distributed content-based web search," *Social Information Retrieval Systems: Emerging Technologies and*, p. 155, 2008.

[38] Facebook developer page. [Accessed 10-January-2015]. [Online]. Available: https://developers.facebook.com/docs/facebook-login/permissions/v2.2

[39] Z. Yang, S. Zhong, and R. N. Wright, "Privacy-preserving queries on encrypted data," in *Computer Security–ESORICS 2006*. Springer, 2006, pp. 479–495.