



TAMPERE UNIVERSITY OF TECHNOLOGY

MIKKO VATAJA
SOCIAL APPLICATION PRIVACY IN MOBILE AD HOC
NETWORKS

Master of Science Thesis

Examiners: Professor Kari Systä,
Teemu Laukkarinen
Examiners and topic approved by
the Faculty Council of the Faculty
of Computing and Electrical
Engineering on 6th March 2013

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY

Master's Degree Programme in Information Technology

VATAJA, Mikko: Social Application Privacy in Mobile Ad Hoc Networks

Master of Science Thesis, 57 pages

May 2016

Major: Software Science

Examiner: Kari Systä, Teemu Laukkarinen

Keywords: Privacy, mobile, ad hoc networks, social networks

Social applications on mobile platform typically use a central server. Use of social applications usually involves private information such as user profiles, that users wish to share securely and in a controlled manner. The central server can function as a trusted central authority that provides the necessary security mechanisms for maintaining the privacy of the users. An alternative for central servers is to use ad hoc networks where there is no central server, but all the communication is done strictly on a peer-to-peer basis. This poses certain challenges in regards to user privacy. The lack of a central server means that the necessary security mechanisms must be in place on each singular device.

This thesis analyzes how the privacy of a user can be maintained in social applications in the context of ad hoc networks. An example social application, Social Index Engine (SoInx), that was designed to run on ad hoc networks is introduced. A significant component of the application was sharing profiles between users via interest matching.

This thesis presents the security mechanisms of SoInx in detail and their efficacy is evaluated using a hostile client that attempts to breach the privacy of the users running the SoInx application. The hostile client was run in a simulated environment, where the simulator models movement patterns of large populations in an urban area, thereby allowing natural encounters between SoInx users and the hostile client to occur.

The hostile client attempted to find out whether an encountered user has been seen before. The results demonstrated that the information SoInx interest matching protocol reveals is sufficient to break the privacy property unlinkability, making it feasible to track people's movements around the city.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO

Tietotekniikan koulutusohjelma

VATAJA, MIKKO: Sosiaalisten sovellusten yksityisyys mobiileissa ad hoc -verkoissa

Diplomityö, 57 sivua

Toukokuu 2016

Pääaine: Ohjelmistotiede

Tarkastajat: Kari Systä, Teemu Laukkarinen

Avainsanat: Yksityisyys, mobiili, ad hoc -verkot, sosiaaliset verkot

Sosiaaliset sovellukset mobiiliympäristössä hyödyntävät tyypillisesti keskuspalvelinta. Sosiaalisten sovellusten käyttö sisältää yleensä yksityistä informaatiota, kuten käyttäjäprofileja, joiden sisältöä käyttäjät haluavat jakaa turvallisesti ja hallitusti. Keskuspalvelin voi toimia luotettuna keskusauktoriteettina, joka tarjoaa tarvittavat tietoturvamekanismit käyttäjien yksityisyyden ylläpitämiseksi. Vaihtoehtona keskuspalvelimille on käyttää ad hoc -verkoja, joissa ei ole keskuspalvelinta, vaan kaikki tiedonvälitys tapahtuu vertaisverkossa suoraan käyttäjien kesken. Tämä aiheuttaa haasteita käyttäjien yksityisyyden suhteen. Keskuspalvelimen puuttuminen tarkoittaa, että tarvittavat tietoturvamekanismit täytyy sisällyttää jokaiseen yksittäiseen laitteeseen.

Tässä diplomityössä analysoidaan, miten käyttäjän yksityisyyttä voidaan ylläpitää sosiaalisissa sovelluksissa ad hoc -verkoissa. Työssä esitellään sosiaalinen esimerkkisovellus, Social Index Engine (SoInx), joka toimii ad hoc -verkoissa. Merkittävä osa sovellusta oli käyttäjien profiilien jakaminen vertailemalla yhteisiä mielenkiinnon kohteita.

Tässä diplomityössä kuvaillaan SoInx-sovelluksen tietoturvamekanismit yksityiskohtaisesti, ja niiden tehokkuutta arvioidaan vihamielisen sovelluksen avulla, joka pyrkii murtamaan samassa ympäristössä ajettavien SoInx-käyttäjien yksityisyyttä. Vihamielistä sovellusta ajettiin SoInx-sovellusten kanssa simuloidussa ympäristössä, jossa simulaattori mallintaa suurten ihmismäärien liikkeitä kaupunkialueella. Tämä simuloi luonnollisia kohtaamisia SoInx-käyttäjien ja vihamielisen sovelluksen kanssa.

Vihamielinen sovellus pyrki selvittämään, onko kohdattu käyttäjä nähty aiemmin. Tulokset osoittivat, että SoInxin yhteisiä mielenkiinnon kohteita selvittävä protokolla jakaa riittävästi informaatiota rikkomaan yksityisyyttä, mahdollistaen ihmisten liikkeiden seuraamista kaupunkialueella.

PREFACE

This thesis is a part of a research project called "Social Index Engine" (SoInx) which took place between 2012 and 2013. SoInx project focused on social applications using local social networking. The project was done in cooperation between Nokia Research Center (NRC) and the Department of Computer Systems (later the Department of Pervasive Computing) at Tampere University of Technology.

I would like to thank everyone who were involved in the SoInx project: people at NRC and my colleagues.

CONTENTS

1	Introduction.....	1
1.1	Background.....	3
1.2	Structure of the Thesis.....	4
2	SoInx.....	5
2.1	SoInx Basics.....	5
2.2	Environment and Constraints.....	5
2.3	Functional Requirements.....	6
2.4	Technical Requirements.....	7
2.5	Inputs for Interests.....	7
2.5.1	Social Networks.....	7
2.5.2	Personal Data on the Device.....	8
2.5.3	Other Interest Sources – Plugin System.....	8
2.6	User Interface.....	8
2.6.1	Radar View.....	9
2.6.2	Importing Interests.....	9
2.6.3	Contact Information.....	10
2.6.4	Configuring Privacy Settings.....	12
2.6.5	User View.....	14
2.6.6	User Interest List.....	15
2.6.7	Send Contact.....	16
2.7	Mobility Generator and Simulator.....	17
2.7.1	Mobility Generator.....	18
2.7.2	Simulator Core.....	20
2.7.3	Network Model.....	21
2.8	Constraints of the Simulator Environment	22
2.9	Simulation Run Statistics.....	23
2.9.1	Reports.....	23
2.10	Related Works.....	24
2.10.1	Highlight.....	24
2.10.2	E-SmallTalker.....	25
2.10.3	Serendipity.....	25
2.10.4	FindU.....	25
3	Privacy.....	27
3.1	Terminology.....	27
3.2	Privacy Properties and Threats.....	27
3.2.1	Unlinkability.....	30
3.2.2	Anonymity and Pseudonymity.....	30
3.2.3	Plausible Deniability.....	30

	3.2.4	Undetectability and Unobservability.....	30
	3.2.5	Confidentiality.....	31
	3.2.6	Content Awareness.....	31
	3.2.7	Policy and Consent Compliance.....	31
	3.3	Privacy in Ad Hoc Networks.....	31
	3.3.1	Location Tracking Based on Network Identifiers.....	31
	3.3.2	Important Considerations.....	32
4		Security Mechanisms of SoInx.....	33
	4.1	Common Interest Exchange Protocol.....	33
	4.2	Encryption Protocol.....	34
	4.3	SoInx Messages.....	34
	4.3.1	Query Messages.....	35
	4.3.2	Match Messages.....	35
	4.4	Query IDs and Match IDs.....	36
	4.4.1	Salt.....	38
	4.5	Handling Received Messages.....	38
	4.5.1	Handling a Query Message.....	38
	4.5.2	Handling a Match Message.....	39
	4.6	Common Interest Verification Process Example.....	39
5		Privacy Analysis of SoInx.....	41
	5.1	Using Network Identifier For Linkability.....	41
	5.2	Broadcasting Sensitive Information.....	41
	5.3	Seeing Broadcasted Interests.....	42
	5.4	Eavesdropping on Private Communication.....	42
	5.5	Using Query ID Sets to Link Encounters.....	42
	5.6	Using Query ID Set Sizes as Fingerprints.....	42
	5.7	Histogram of Query IDs as a Fingerprint.....	43
	5.8	Physical Observations.....	43
	5.9	Other Applications Leaking Identity.....	43
	5.10	Recognized and Unrecognized Query ID sets.....	43
	5.11	Summary of Security Mechanisms for Maintaining Privacy in SoInx.....	44
6		Hostile Client.....	45
	6.1	Hostile Client With Massive Interest Database.....	45
	6.2	Implementation.....	46
	6.2.1	Collected Data.....	46
	6.2.2	Post-processing.....	47
	6.3	Results of Hostile Client Execution.....	48
7		Evaluation of SoInx.....	49
	7.1	User Tests.....	49
	7.2	Encryption Based on Common Interests.....	50

7.3	Ideas for Improvement.....	50
7.3.1	Diffie-Hellman and Other Key Exchange Protocols.....	50
7.3.2	Tweaking Hashing Parameters.....	51
7.3.3	Discouraging Easily Crawlable Interest Sources.....	51
7.3.4	Recognizing and Reacting to Certain Attacks.....	52
7.3.5	Context Depending Behavior.....	52
7.3.6	PSI & PCSI protocols.....	52
8	Conclusions.....	54
	Bibliography.....	55

TERMS AND ABBREVIATIONS

Ad hoc	In this context a technology that enables peer-to-peer communication without infrastructure such as access points or routers
AES	Advanced Encryption Standard
C++	A programming language
Computationally difficult	Impossible for modern supercomputers to do in a reasonable amount of time. A term commonly used in cryptography to describe infeasibility of breaking an encryption.
DTN	Delay-tolerant networking
MAC address	Media access control address; physical address; a unique identifier for a network interfaces for physical layer
PCSI	Private cardinality of set-intersection
POI	Point of interest
PSI	Private set-intersection
Python	A programming language
Qt	A cross-platform application framework
SMC	Secure multi-party computation
SoInx	Social Index Engine, an example social application designed for ad hoc networks. Also the name of the project.
WLAN	Wireless local area network

1 INTRODUCTION

Wireless connections between mobile devices are usually implemented via a central server and a wireless Internet connection. A less widespread way to connect between devices is using technologies that communicate in a peer-to-peer manner, where devices in close enough proximity form an ad hoc network. Examples for technologies enabling this type of communication are the ad hoc mode of Wireless LAN and Bluetooth. This peer-to-peer manner of communication has some advantages, namely it does not require any pre-existing infrastructure such as WLAN access points or mobile base stations. However, it requires physical proximity between the communicating devices.

Ad hoc connection extends communication capabilities of mobile devices by increasing their awareness of their surroundings. This opportunistic access allows different kinds of social applications to emerge, namely proximity based services and interactions between users.

The devices can form and maintain the network even while individual devices appear or disappear from the communication range of other devices. It is not necessary for every device in an ad hoc network to be within the range of every other device; one device (or possibly several devices) can act as an intermediary and relay messages between two devices that otherwise would not be able to communicate with one another. This means that ad hoc networks are self-organizing and adaptive [1]. This is illustrated in Figure 1.1. A solid arrow represents a direct connection between devices, i.e. they are in the communication range. A dotted arrow represents a connection that has to be routed through one or several intermediary devices.

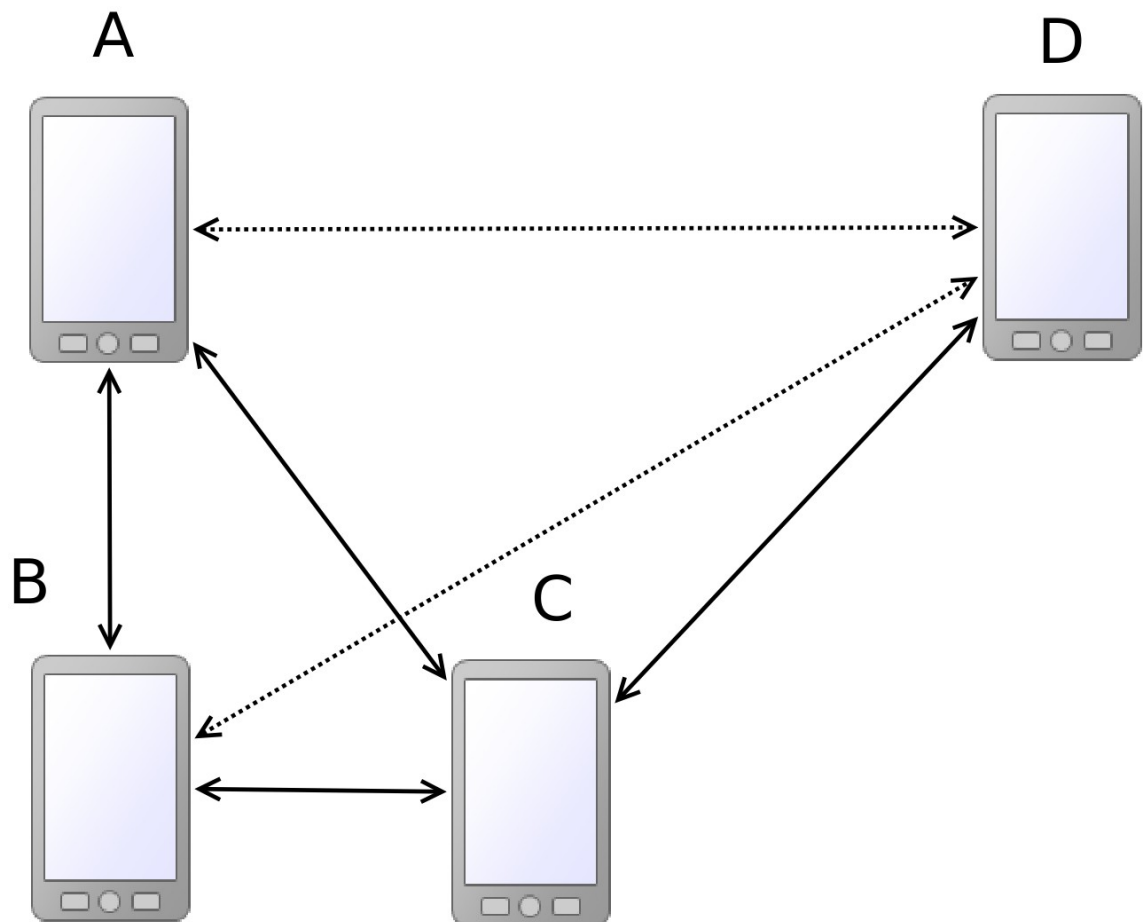


Figure 1.1: An example ad hoc network. The connections from devices A and B with device D are routed via device C.

Social applications on mobile devices are nowadays commonplace, but most of them are implemented to utilize a central server [2]. Since users of social applications typically supply and share private information such as user profiles, security mechanisms are needed to do this privately. The availability of a central server permits the use of the server as a trusted central authority that manages most of the security mechanisms.

When the social application runs on ad hoc networks, the peer-to-peer manner of connectivity requires a different approach for security and privacy issues. The lack of a central server means that each device must have all the privacy and security solutions implemented in its hardware and software. However, this means that no third party has to be trusted to keep the users' data and communications safe. The privacy requirements must also be balanced with other requirements of mobile applications such as minimizing energy consumption and data throughput.

To research social applications in the ad hoc environment, a prototype application called Social Index Engine, SoInx, was developed by a research group at Tampere University of Technology, which the author of this thesis was a part of. SoInx tries to find new interesting people and content in proximity, based on shared interests between users. SoInx uses several sources (inputs) of interests, and tries to match them between people in close enough proximity.

A simulator was used for testing proximity based applications. The simulator includes a mobility model that models movement patterns of large populations in an urban area. The simulator allows large scale testing of social applications reliant on ad hoc networking between proximate devices.

In this thesis the privacy of SoInx is analyzed against known privacy threats and evaluated with a help of a hostile client that attempts breach the privacy of SoInx users. The security mechanisms of SoInx for maintaining privacy are described in detail, and possible attacks against these security mechanisms are discussed. The hostile client implements an attack where it attempts to recognize whether an encountered user has been seen before. This is done by utilizing a large interest database. The hostile client was tested in a simulated environment with SoInx clients. The results of the hostile client execution showed that the attack was a feasible way to track the movements of singular users and masses of users alike.

1.1 Background

This thesis introduces the SoInx application, and privacy and security matters related to it. The SoInx research project was a co-operative project between Nokia Research Center and Department of Computer Systems (later Department of Pervasive Computing) at Tampere University of Technology. The SoInx application was developed as a prototype social application for Nokia Instant Community platform.

The SoInx project is a continuation of a research project called TWIN, which produced a social mobile application that allows people to chat, send files, and form communities in the context of local wireless ad hoc networking.

The author of this thesis joined the SoInx research team after the initial version of the SoInx application was developed. The work of the author included further development to the SoInx application and the SoInx simulator with the research team, and also more comprehensive privacy analysis of the SoInx application and its environment. The SoInx simulator was developed earlier during the TWIN project [3]. The hostile client was also the responsibility of the author.

In september 2013, an article about SoInx application and its user tests was published and presented at a workshop on Mobile Cloud and Social Perspective (MoCSop) [4].

1.2 Structure of the Thesis

This thesis is organised as follows. Chapter 2 introduces the prototype implementation of the example social application SoInx and describes related works on proximity based social applications. Chapter 3 presents an introduction to privacy issues and discusses how they map to proximity based social applications on ad hoc networks. Chapter 4 describes SoInx security mechanisms in detail. Chapter 5 describes possible privacy issues and explains how the security elements of SoInx solve those issues. Trade-offs between chosen and rejected security mechanisms are also discussed. Chapter 6 contains an assessment of SoInx privacy using a hostile client that aims to breach user privacy. The methods of evaluation and possible types of attacks are discussed. One type of attack is described in detail and the implemented hostile client and its results are discussed. Chapter 7 contains the evaluation of SoInx application via user testing. Possible improvements and further development ideas are also discussed. Finally, chapter 8 gives the conclusion of the thesis.

2 SOINX

This chapter introduces a social application called Social Index Engine, SoInx, and describes the prototype implementation.

SoInx is a proximity based social application that runs on local ad hoc network environment. This means that it only communicates with other SoInx running devices that are in close enough proximity. SoInx uses no external infrastructure such as WLAN routers or mobile Internet; all communication between devices is instantiated in a peer-to-peer manner.

The main purpose of the application is to find new information and interesting people.

2.1 SoInx Basics

The main function of SoInx is to discover new interesting people based on common interests. Proximity is an important part of the concept; only with SoInx users that are in close enough proximity (i.e. in the range of the ad hoc networking technology) any kind of interest matching process is started. This means that people with matching interests are already physically close to one another, enabling people to quickly form relationships if they wish.

A single piece of interest can be for example a Facebook like or a LinkedIn contact. Based on user interaction SoInx tries to learn which interests and what kind of people the user finds most interesting.

2.2 Environment and Constraints

SoInx works in an environment where the devices see only other SoInx running devices that are in close enough proximity. The range is technology dependent. One example is the ad hoc mode of WLAN which was chosen for the application. Another example is Bluetooth [5], where the range is significantly smaller which is not large enough for the purposes of this application.

The prototype implementation uses an experimental WLAN mesh communication called Nokia Instant Community implemented by Nokia Research Center [6]. Nokia Instant Community adds power saving functionality on top of the WLAN technology.

All the messages in SoInx are sent in a broadcast manner. Since in ad hoc networks everyone close enough can see every message, messages directed to a single person should be encrypted so that the outsiders are prevented from viewing the contents of such messages. This is achieved using the SoInx encryption protocol that is described in more detail in section 4.2.

The ad hoc network may be used by other devices and applications. Other than valid SoInx messages are simply ignored. This makes it unlikely for other applications to disturb communication between SoInx clients, besides consuming communication bandwidth.

The users that communicate together do not form any explicit communities. Interests are broadcasted to everyone in the range, and after establishing common interests subsequent messages, although broadcasted, are done in a one-to-one manner.

Due to privacy issues, the network identifier is changed every hour. Whenever a new SoInx user appears with a previously unseen network identifier, SoInx does not attempt to link it to previous encounters; all the new encounters are assumed to be previously unknown from the perspective of the application.

This type of proximity based peer-to-peer communication may prove to be useful especially in areas where mobile Internet is inaccessible or unstable, such as developing countries.

2.3 Functional Requirements

Background operation. SoInx application was designed to run in the background most of the time. It should find interesting people without any user input, and then possibly alert the user when an especially interesting person is found.

Tolerant to network changes. The nature of mobile ad hoc networks is that the network topology can change at any moment. For example any node can appear or disappear at any moment. SoInx application should be able to handle these kinds of situations.

Touch based UI. SoInx was designed to run on Nokia N9 mobile phone. This means that user interaction is implemented via a touch based user interface with a screen the size of 480 x 854 pixels.

Privacy. Main purpose of SoInx is finding new interesting people based on common interests. This should be accomplished without revealing too much personal information to other users, and making it sufficiently hard for any outside observer to gather such information. This is a major perspective this thesis discusses SoInx application.

2.4 Technical Requirements

Since SoInx application is running on a mobile device that is usually not connected to an external power source, minimizing battery consumption is an important requirement. This was taken into account by avoiding algorithms requiring high performance, and by minimizing network traffic. Both the size and the number of messages sent were reduced.

Most of the time SoInx application runs in the background. While in this state, broadcasting type messages are sent only periodically. The SoInx messaging scheme is discussed more thoroughly in section 4.3. Lower power consumption is also achieved using the Nokia Instant Community platform [6].

2.5 Inputs for Interests

Different interest sources – inputs – are used to gather interests that the user may wish to share with others. These interests could include music, movies, friends, likes (in the context of social networks), or any piece of information that user finds interesting that can be used to connect different people.

SoInx application uses a plugin architecture for different sources of inputs; new sources can be added when deemed necessary. The main interest sources implemented in the prototype were social networks, and personal user data on the device.

2.5.1 Social Networks

Social network sites are useful sources for pieces of information that the user might be interested in. Many social networking sites have ways for the users to define friend relationships between themselves. This works as a SoInx interest, since having common friends could be a good indication that people may wish to get to know one another.

Another widespread notion among social networking sites is the concept of a like. This concept may have different names in separate social networking sites (such as '+1' on Google+, or a thumbs up on Youtube), but they all denote a similar intent: the user likes something. This also is a good piece of information to serve as a SoInx interest.

Social networking sites usually allow access to users' likes, friend relationships, and other data programmatically using their defined API (application programming interface). Using this type of API usually requires a user specific secret key, or the user's username and password.

Facebook was the first social network input implemented for SoInx. Facebook's popularity allows to collect a large number of interests, both likes and friend relationships. Facebook uses OAuth 2.0 for authenticating a user and allowing to extract user data via the Facebook API [7].

Twitter was also implemented as an input. A user's followers, followees (whom the user follows), and tweets (Twitter posts) were used as interests. Twitter uses OAuth 1.0A for authentication [8].

Other social networking sites such as LinkedIn and Google+ were considered as inputs, but they were not implemented due to other features being considered as having higher priorities. Implementing them would not have brought significant advantage, considering that interests from Facebook and Twitter were comprehensive enough for the prototyping purposes.

SoInx attempts periodically to redownload the interests from the utilized social networks. This allows SoInx to receive more interests as the user gathers more likes and friend relationships etc.

2.5.2 Personal Data on the Device

Mobile devices nowadays, i.e. smart phones, contain plenty of personal user data. Some of the data the user may wish to share with others. SoInx uses this data found on the mobile device as a source of interests.

Typical data that works as SoInx interests would be for example PDF and media files found on the device, bookmarks from web browsers, contact information on the address book, and installed third party applications on the device.

SoInx can take this type of information and use it as interests.

2.5.3 Other Interest Sources – Plugin System

The number and nature of interest sources are not hard coded in the system. SoInx uses a plugin system that allows adding new interest sources later on. This permits the utilization of future social networks and other upcoming web applications as interest sources.

In addition to input sources, the user can also manually input things they are interested in. These interests take the form of textual keywords, such as "hiking", "basketball", etc.

2.6 User Interface

SoInx prototype application was designed to run on a handheld Nokia N9 phone with a touch screen. Two basic requirements for the user interface were ease of use, and scalability.

Scalability in this context means that the user interface should be usable regardless of how many SoInx users are in proximity; in an environment with lots of users this could mean that only the most interesting users are displayed.

2.6.1 Radar View

Radar view was considered as the main view in the SoInx application. It visualizes other SoInx users in proximity. This is illustrated in Figure 2.1.

The center of the view represents the user holding the device, and icons floating on different circle perimeters represent other SoInx users in proximity. The type and color of the icon is decided on the common interests and how interesting the other user is deemed by the learning algorithms of SoInx. The icon colors vary between green and red, where green indicates a less interesting person and red indicates a more interesting person. The distance from the center depicts the number of hops a message must pass through to reach the user.



Figure 2.1: Radar view

2.6.2 Importing Interests

The user can import interests into SoInx from several sources. The user can choose from a list of possible sources for importing interests. This is shown in Figure 2.2.

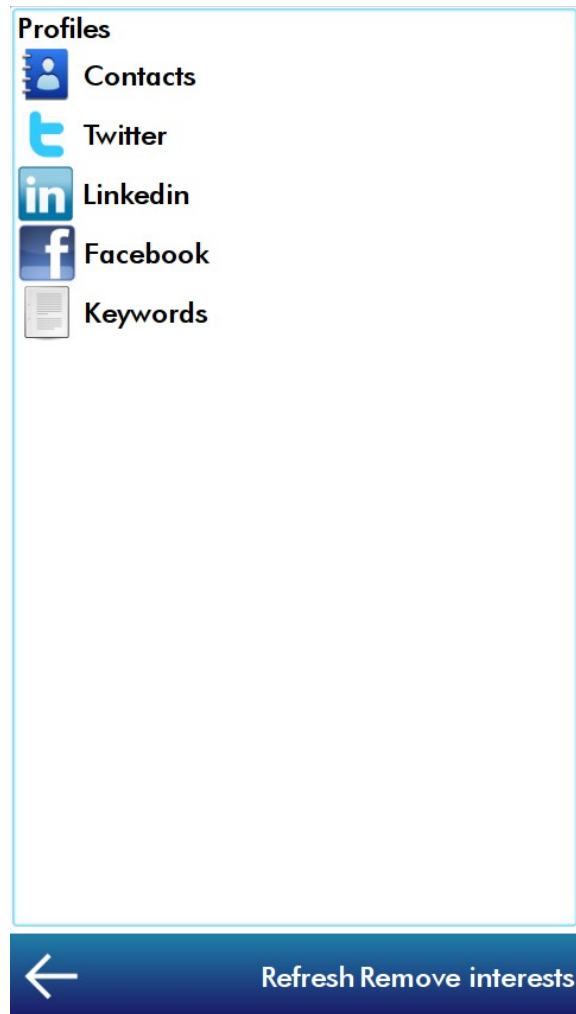


Figure 2.2: User interface for importing interests.

The user can highlight a source such as Facebook and tap "Refresh". The user's Facebook username and password is asked, after which the user's Facebook likes and friend relationships are downloaded and imported as interests. This is one of the few use cases when an Internet connection is required while using SoInx.

2.6.3 Contact Information

SoInx application allows the user to fill a contact card with the user's contact information. This information can be sent to other SoInx users that seem interesting. This is shown in Figure 2.3.

Set Contact

Save Contact information

Website

Phone Number

Nick Name

Name

Email Address



Figure 2.3: User interface for setting up contact information.

2.6.4 Configuring Privacy Settings

Some data interpreted as interests may contain information that the user wishes to keep private. Therefore, SoInx application must include a way to indicate which interests are fine for others to see, and which interests should be kept hidden.

SoInx solves this by having a simple user interface to configure the privacy settings for each source of interests. This is shown in Figure 2.4. The possible privacy values are public, shared secret, and hidden. Public means that interests from this source are broadcasted normally in query messages. Shared secret means that interests from this source are never broadcasted, but they are used for matching the same interests when seen broadcasted by other users. Querying and matching are explained in more detail in Chapter 4. Hidden means that this input source is never taken in consideration in the functionality of SoInx application.

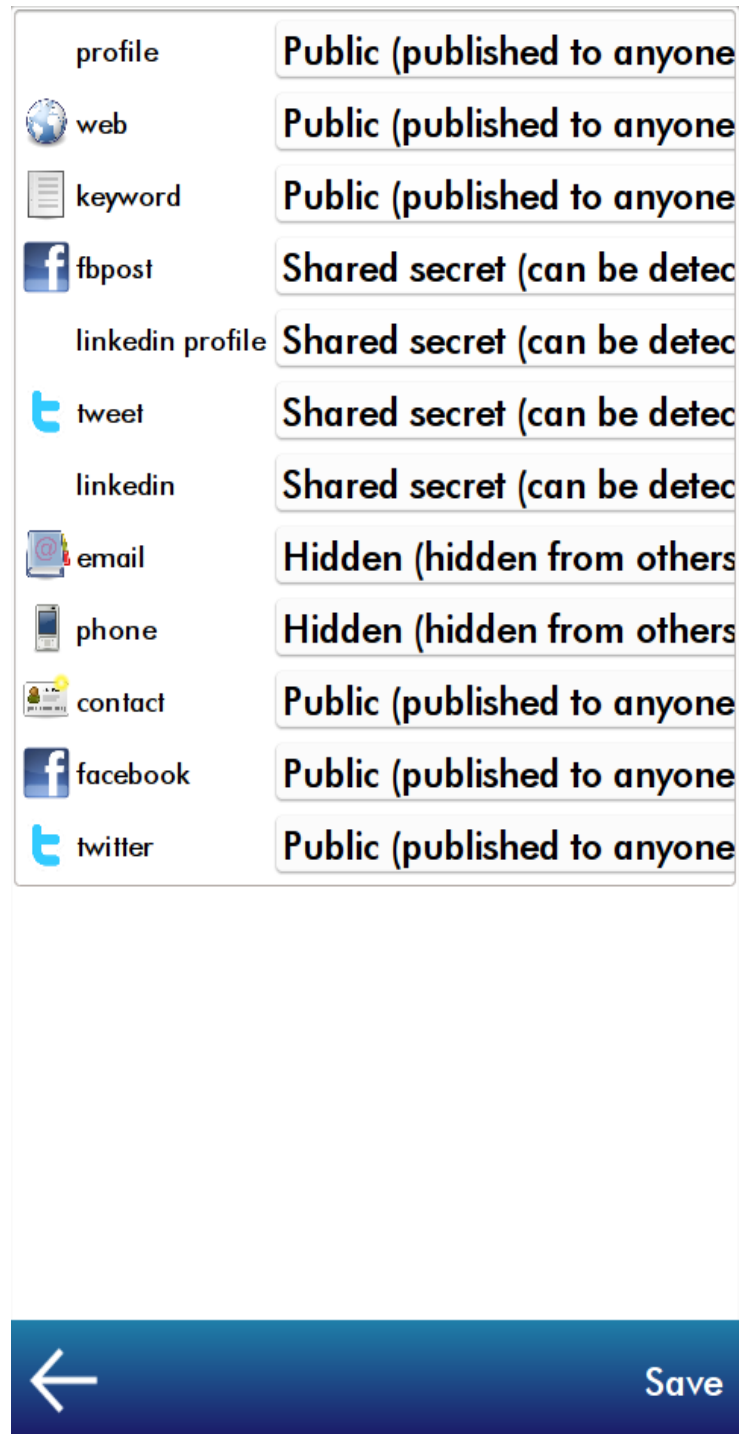


Figure 2.4: User interface for privacy settings.

2.6.5 User View

After clicking a user icon on the radar view, a user view is opened. This is shown in Figure 2.5.

User view informs if the user has common interests with the match. The top interest is shown (decided by the learning algorithm). This view allows the user to like or dislike the match (the icons "Like" and "Not Interesting" respectively). Clicking these are used as inputs for the learning algorithm that determines how important certain interests are for the user. Tapping "Later" leaves this view and goes back to the radar view, and "See More" opens up the user interest view that is discussed in section 2.6.6.

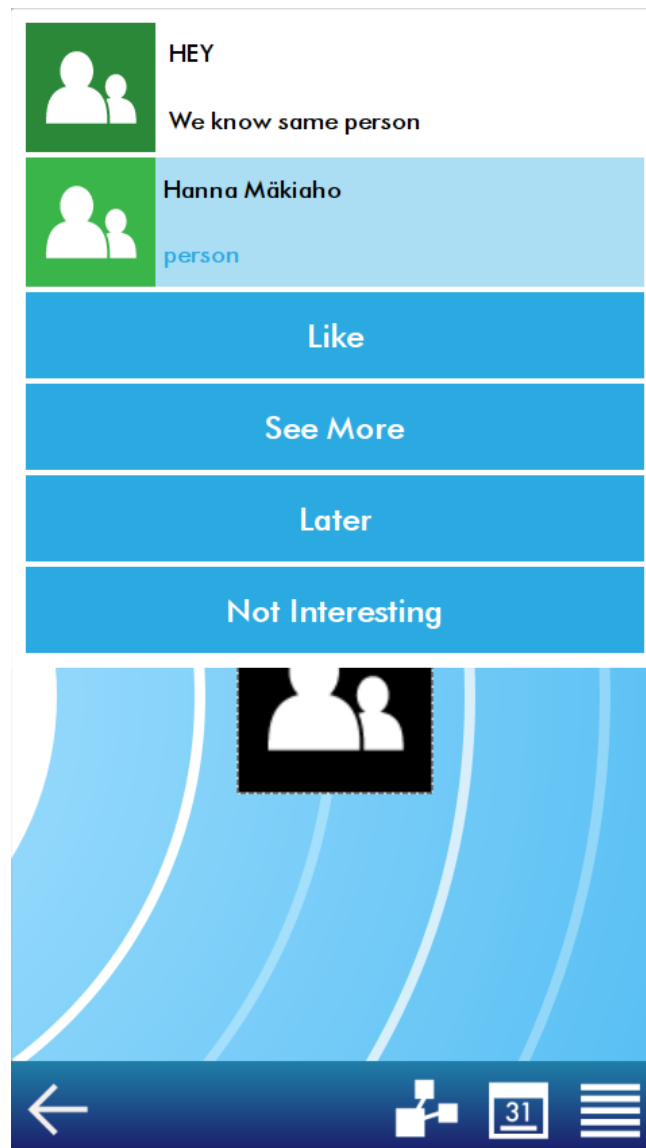


Figure 2.5: Main view showing information on another user.

2.6.6 User Interest List

User interest view shows a list of all the common interests the user has with a specified match. The list is scrollable, if there are more interests than what fits on the screen.

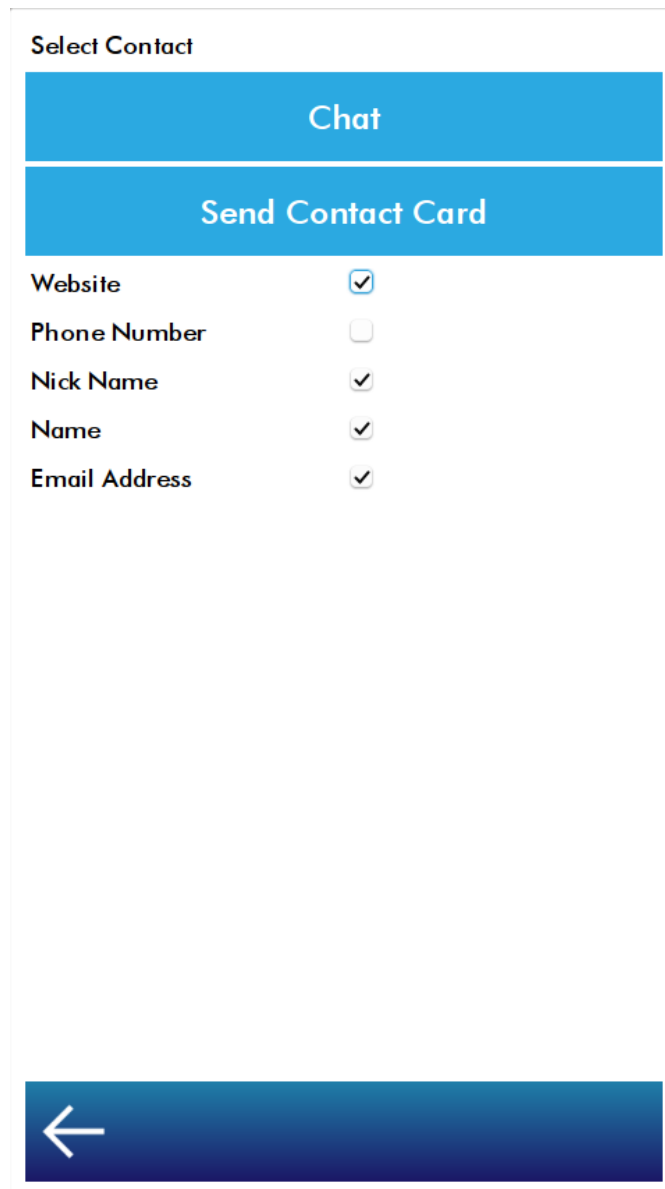
The user interest list is shown in Figure 2.6. The view also includes a button to contact the person. This opens a view described in section 2.6.7.



2.6.7 Send Contact

Send Contact view allows the user to contact a person he/she deems interesting.

The user has two ways of contacting the person; one is to send a contact card which can be filled using the contact information view (section 2.6.3). The other way is to establish a chat connection with the person. The actual chatting was not implemented in this prototype due to prioritizing reasons, but tapping the chat button was logged and used as an input for the learning algorithm.



Select Contact

Chat

Send Contact Card

Website

Phone Number

Nick Name

Name

Email Address

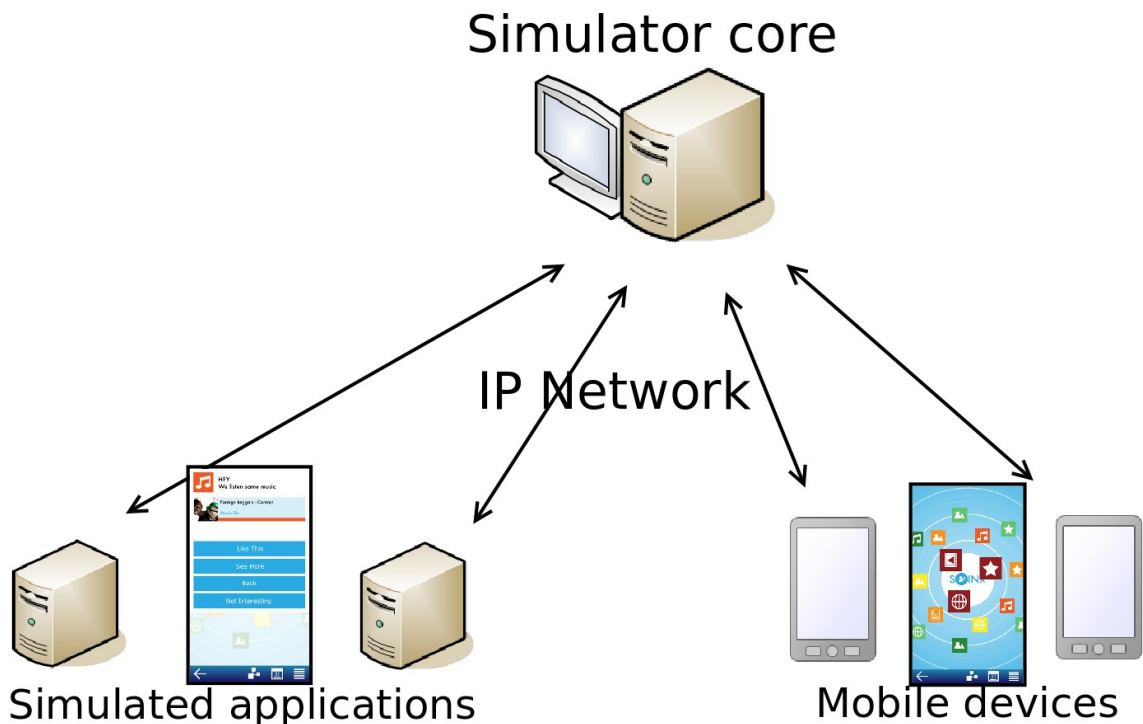
2.7 Mobility Generator and Simulator

A simulator was used for testing the SoInx application with a large number of users to get a sense of the communication patterns and functionality when the application is used by a larger population. The simulator was developed during the preceding TWIN research project, and is described in more detail in Janne Kulmala's thesis [3]. The simulator was designed to facilitate faster and easier testing of social applications in ad hoc networks. This was achieved by modeling movement patterns of a large population in an urban area. The model corresponds to the real life movement patterns, since it simulates the behavior of people commuting between home, the place of work or study, and places of interest.

The simulator simulates daily movements of several hypothetical mobile device carrying people. For every simulated person, one client application, which represents the ad hoc network using software (in this case the SoInx application), is started. The client applications can be run on separate desktop machines or mobile devices. The communication between the core simulator machine and client machines is done via an IP network. The system architecture of the simulator is illustrated in Figure 2.8.

While the simulator application is running, it transmits messages between the running clients that are in close enough proximity in the simulated environment. The concept of time is also provided to the client applications by the simulator. The simulator also includes a user interface that visualizes the people's movements and message passing while the simulation runs. This is described more rigorously in section 2.7.2.

The simulator environment was used to test and evaluate user privacy with a hostile client that attempts to breach the privacy of SoInx users.



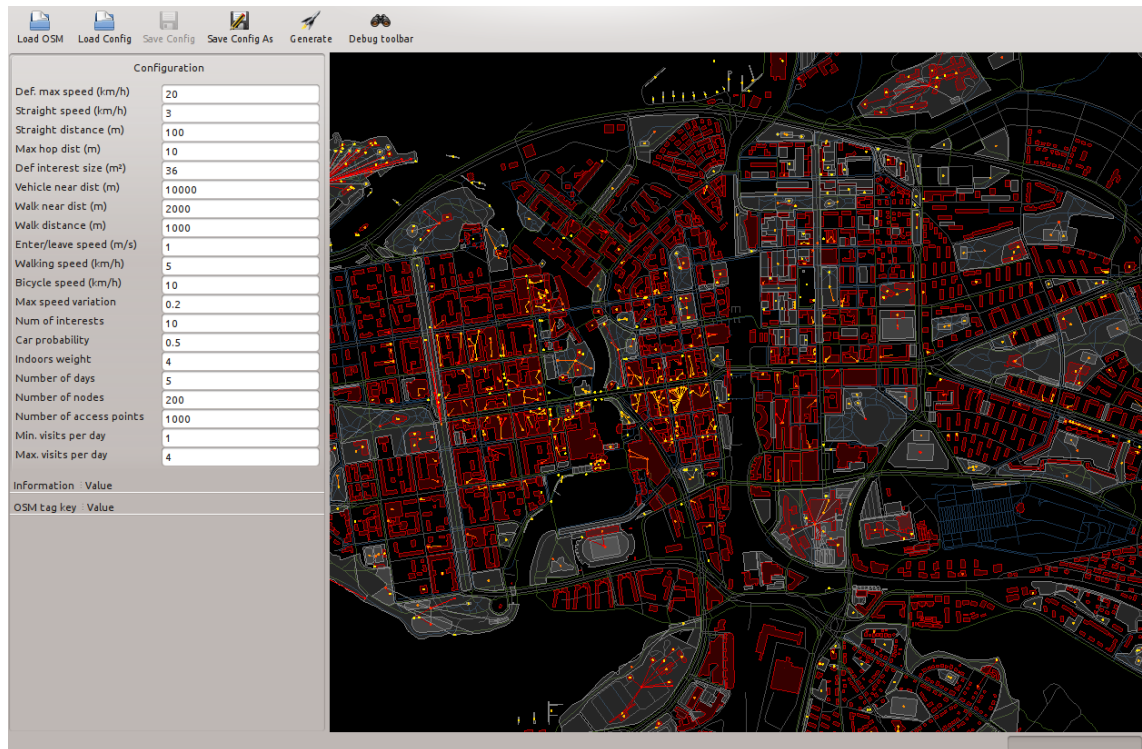
2.7.1 Mobility Generator

Mobility Generator is a tool that generates daily schedules for a group of people and models their movements around a city or other urban area, to be used in the simulation. The area where people's movements are generated, is taken from OpenStreetMap and can be any urban region.

Inputs for Mobility Generator are the area map (taken from OpenStreetMap) and mobility parameters. The map is analyzed, and movement patterns for simulated people are generated. The movements patterns are saved into a file, which in turn is used by the core simulator application. The mobility parameters are adjustable, and they include person specific parameters such as the walking speed, the probability that the simulated person uses a car, how many places are visited per day etc. They also include simulation specific parameters, such as the time frame of the simulation i.e. how many days are simulated, and how many people are simulated.

Mobility Generator includes a user interface for configuring the mobility parameters, and for visualizing the map of the urban area where the simulation takes place in. The map visualization allows zooming into specific POIs (point of interest) in the urban area and viewing their type such as school, restaurant, grocery store, et cetera. The map view can also be used for finding problems such as errors in the path finder. The user interface of the Mobility Generator is shown in Figure 2.9.

Mobility information. For each simulated person, mobility information is generated. The mobility information of a person consists of a path, represented in the form of time/location pairs. Time refers to a certain point in simulated time and the location refers to a 2D coordinate on the map. During the simulation a simulated person's location and velocity at a certain time can be calculated by linear interpolation between two consecutive path points [3].



Daily schedule. For every simulated user a daily schedule is generated for every simulated day. A simulated person is assumed to have a home, where every night is spent in. The home for each simulated user is chosen randomly from the set of buildings without any interest information in the OpenStreetMap data; these are assumed to be residential buildings. The probability of choosing a specific building is weighted by the building size.

A person is also assumed to have one important place to visit every day. This represents the person's place of work or study. This place is chosen randomly from POIs on the map data. The POIs are weighted by their category, for example schools are more likely to be chosen.

After spending the day at the place of work or study, the simulated person visits a small number of interesting places that represent places of hobbies, grocery store visits etc. These places are also chosen randomly, weighted by their category and distance from home and the place of work or study.

A simulated person is assigned randomly to own a car, which determines the velocity at which the person can move between places of visit. The route between visiting location is along the roads (from the OpenStreetMap data), and is calculated by A* pathfinder algorithm [9], where the cost function is the travel time between two road nodes (OpenStreetMap data includes speed limits for road nodes).

Real world applicability of the generated mobility information. In the TWIN project, a user trial with 300 devices was undertaken. The trial showed that the real world movement patterns gathered from the user trial corresponded to the generated mobility information of this mobility model [3].

2.7.2 Simulator Core

The simulator takes mobility data generated by Mobility Generator. It uses the data to simulate people's movements.

The simulator is responsible for maintaining the state of the simulation, which includes keeping track of the simulated time, the simulated people's locations, and the state of the network. Whenever people are in close enough proximity, the simulator allows them (i.e. their mobile devices) to send messages between one another. Also the simulated network identifiers are changed periodically, to simulate the privacy enhancing feature of Nokia Instant Community platform.

For each simulated user, a client application is started. The application represents the software the simulated people are running on their devices. The simulator provides a client API that provides the state of the simulated environment, and enables the client applications to communicate with each other via the simulated ad hoc network. The client applications use the API also for obtaining the current time and scheduling events in the future. This allows the simulation to run on variable speed.

The client API works via UNIX sockets or TCP sockets. This allows for the client applications to be run on both the same machine as the simulator is running on, and on different machines that communicate via IP network with the simulator machine. In the case of same machine, the client applications are run on different processes than the core simulator application. This is to achieve better decoupling between the simulator and the client applications, and also to prevent accidental sharing of state between the client applications and the simulator application.

The simulator includes a user interface to view the state of the simulated environment. This includes a map view where the simulated clients are shown as colored circles. The area of the circle represents the reach of the ad hoc network of a specific device in the simulation. The colors are same for those clients who at a specific point of (simulated) time are in close enough proximity to be able to send messages to one another, forming an ad hoc network. The user interface of the simulator is shown in

Figure 2.10. The figure also includes a single SoInx application running in the simulated environment.

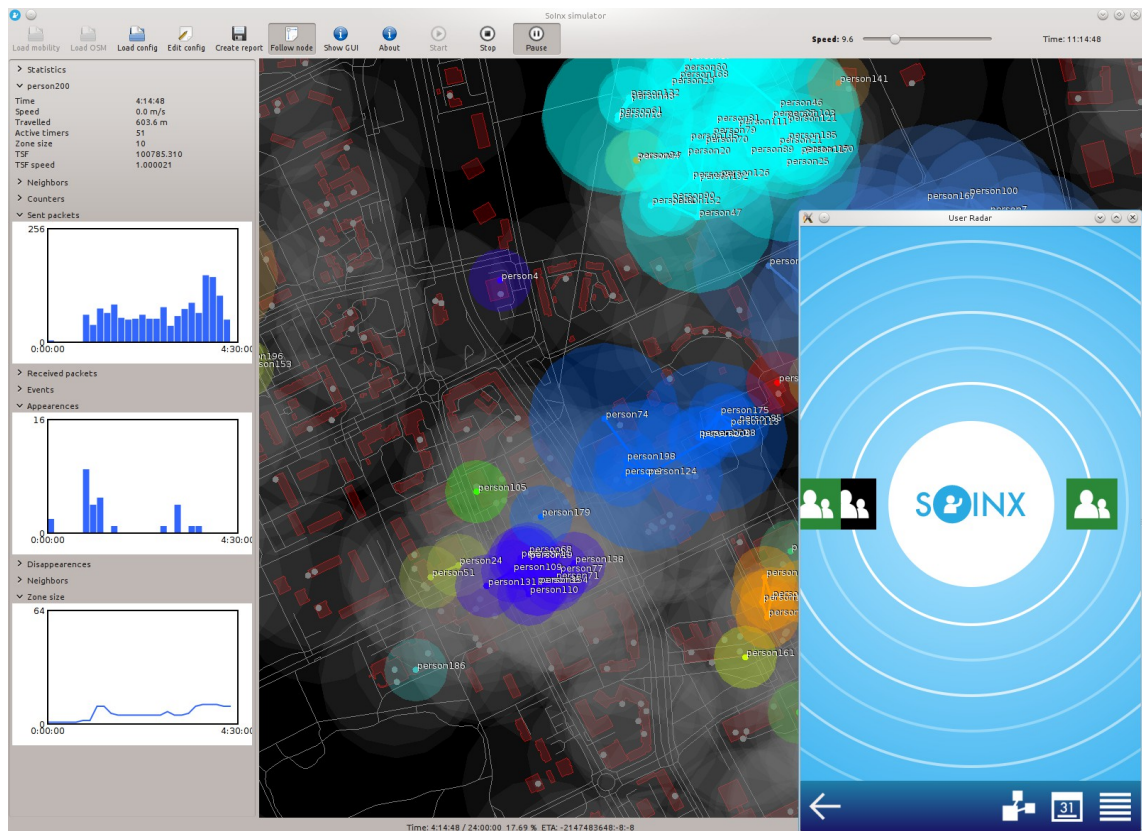


Figure 2.10: Simulation run with a SoInx client on the foreground (bottom right). On the left some statistics specific to a simulated user are shown, and in the middle is the map view.

The user interface allows to adjust the speed of the simulation, and allows to view simulated user specific statistics in real time. This includes statistics such as sent messages, encountered users, how many users constitute the current ad hoc network in the simulated location, et cetera.

After the simulation run is complete, the simulator generates a report. This is discussed more in section 2.9.1.

2.7.3 Network Model

The simulated network in the simulation estimates a radio channel and a network protocol to achieve statistically similar behavior of a real world ad hoc network. It is modeled after an ad hoc network that implements multi-hop features and power saving features. The communication in the network also has a possibility of signal degradation in the form of packet loss, the likelihood of which increases as the distance between the devices grows.

Multi-hop. Multi-hop in an ad hoc network refers to the feature that singular devices will forward traffic by re-transmitting packets that they receive. This makes it possible for two devices to communicate even when out of each other's radio range. This is illustrated in Figure 1.1.

Packet loss. The network model also implements the possibility of packet loss. This is because a radio channel has reliability issues caused by background noise, other radio transmissions in proximity, and signal decay over distance. The packet loss probability uses a linear model, where the probability of packet loss increases as the distance between the communicating devices grows, and transmission is not possible when the distance is more than the radio range.

Power saving. To prolong battery life the devices are asleep much of the time, and they wake up only periodically to send and receive traffic. Devices that wish to communicate between one another has to be awake at the same time. This is achieved by periodically staying awake a longer time, to synchronize the wake up periods of the devices.

Communication zones. Communication zone refers to a set of devices following the same power saving cycle. The zones are supposed to be as large as possible, so that the maximum number of devices are able to communicate between one another. When a device notices that it is in the range of several communication zones that are on different power saving cycles, the device synchronizes with the communication zone that is the largest (i.e. it consists of the largest number of devices). This feature causes a domino effect of all the devices migrating to the same communications zone, sometimes one by one after a delay of several power saving cycles.

The network model is described more rigorously in the thesis of Janne Kulmala [3].

2.8 Constraints of the Simulator Environment

The computer running the simulator places some constraints on the scope and the number of tested client applications. The client applications can be run on the same computer, or on different machines while communicating with the simulator via network. When run on the same machine, RAM and CPU limits the memory footprint, performance, and the number of client applications. When run on different machines, the network bandwidth places limits on the number and size of the messages the client applications are sending.

SoInx applications had several optimizations to reduce the amount of RAM and processing power required. The user interface was made optional; it was started only for those simulated clients where user interaction was required. Also the number of messages sent between simulated clients was aggressively reduced whenever their communication was deemed uninteresting. An example situation was user testing; only

the messages relating to the person doing the user testing were interesting. Other messages between simulated clients who were not in proximity of the tested user were just ignored in the context of the user test. Henceforth, those messages were not even sent to reduce CPU load and messaging bandwidth.

Most of our simulations (including the user tests) were done with 900 simulated users in the area of Tampere. The simulated SoInx clients were run on the same desktop machine as the simulator application. The simulator was able to run 5 days of simulated time in less than one hour of CPU time.

2.9 Simulation Run Statistics

During the simulation run, statistics are gathered for each simulated user. Several types of data are gathered: Sent packets, Received packets, Events, Appearances and Disappearances.

This data is used to generate reports after the simulation run.

2.9.1 Reports

Total amounts of gathered data for each person are shown in the main report view. The report is shown in Figure 2.11.

Created 2012-12-18 13:14:38

Node	Sent packets	Received packets	Events	Appearances	Disappearances	
person1	934	920	2043	153	153	CSV
person10	3385	10259	14369	525	525	CSV
person100	1381	2447	4113	244	244	CSV
person101	1916	3125	5679	476	476	CSV
person102	2961	4877	8582	551	551	CSV
person103	1417	7746	9632	355	355	CSV
person104	3899	5326	9989	565	565	CSV
person105	2543	3699	6803	420	420	CSV
person106	1281	2009	3831	384	384	CSV
person107	7816	6120	14542	457	456	CSV
person108	2851	6187	9406	297	297	CSV
person109	3763	5472	9862	456	456	CSV
person11	2080	798	3224	297	297	CSV
person110	1968	8090	10765	510	510	CSV
person111	3079	9588	13421	514	514	CSV
person112	4110	20472	25462	616	616	CSV
person113	7413	12923	20826	364	362	CSV
person114	3371	6046	9736	259	259	CSV
person115	8312	17240	26694	807	806	CSV
person116	4111	8402	13503	693	693	CSV
person117	6992	2744	10056	277	276	CSV
person118	1571	2836	4766	298	298	CSV
person119	2693	2184	5158	233	233	CSV
person12	3230	2913	6821	492	492	CSV
person120	7423	13287	21729	722	721	CSV
person121	7062	6681	14204	369	368	CSV
person122	4486	11157	16431	629	629	CSV
person123	3646	7911	11961	329	329	CSV
person124	3521	7040	10946	288	288	CSV
person125	2608	9964	13222	464	464	CSV
person126	3527	3454	7399	352	352	CSV
person127	3742	11389	15696	412	412	CSV
person128	896	1406	2582	202	202	CSV
person129	4170	14453	19544	655	655	CSV
person13	3487	4847	9035	489	489	CSV
person130	385	335	733	12	12	CSV
person131	783	439	1373	137	137	CSV
person132	7037	3947	11240	216	215	CSV
person133	501	914	1469	45	45	CSV
person134	4293	9667	14653	546	546	CSV
person135	3027	3919	7457	367	367	CSV
person136	3682	2478	6636	362	362	CSV
person137	2470	4007	9015	400	400	CSV

Figure 2.11: Report generated by the simulator

Also a more detailed report is generated for each simulated person; the report shows the number of sent messages etc. during a certain time frame. This helps to see at which times the communication between the SoInx users is most active. A single user report view is shown in Figure 2.12.

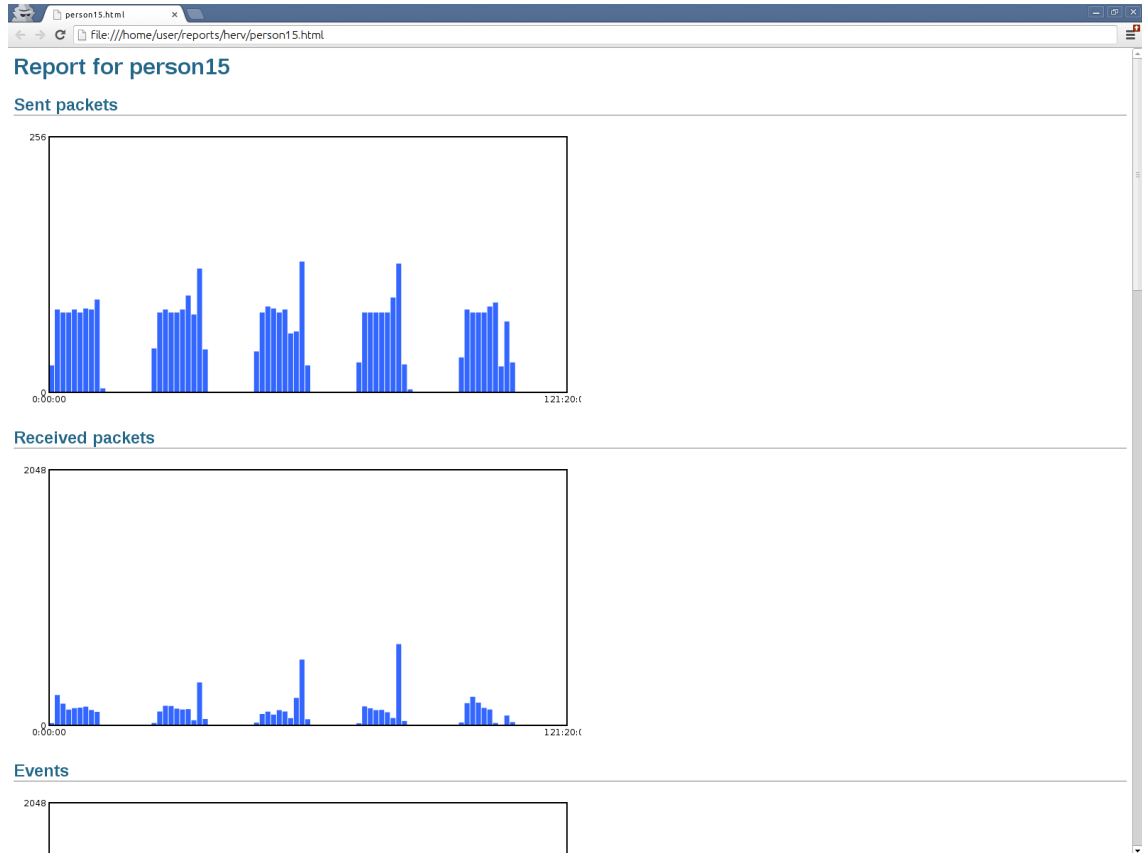


Figure 2.12: Report generated for a single user

2.10 Related Works

Related works consist of applications and research related to mobile social networking. Some use ad hoc networking, others are server based. The social networking aspect is described, and the way the application takes privacy into account, is discussed.

2.10.1 Highlight

Highlight is a social networking application for iOS and Android [10]. The application is similar to SoInx in that it shows nearby people who have common interests with you. Instead of using ad hoc networking, Highlight is server based; the user position is periodically sent to a central server, and the server uses this position data to notify users when there are friends or interesting users nearby. This also means that the ad hoc networking related privacy risks are avoided.

Highlight has attracted some criticism for disclosing private information to strangers [11]. Since SoInx application uses similar sources for interests (Facebook likes and friend relationships), it is likely that if published, it would receive similar criticism.

2.10.2 E-SmallTalker

E-SmallTalker is a distributed mobile system for social networking in physical proximity [12]. It aims to initiate meaningful small talk between two strangers by discovering shared interests, and suggesting them for topics of conversation.

E-SmallTalker uses Bluetooth Service Discovery Protocol as the ad hoc networking technology. Interest matching is achieved using a low bandwidth query methods utilizing Bloom filters [13]. Bloom filters use one-way hashing to encode interests, which makes it difficult to derive the original set of interests, therefore protecting privacy of the users. However, E-SmallTalker is vulnerable to a dictionary attack [14].

2.10.3 Serendipity

Serendipity is a match making software that calculates similarity scores between encountered users' profiles, and notifies the user when the score is above a certain threshold. Serendipity uses a combined ad hoc networking and server based implementation; Bluetooth discovery protocol is used to discover nearby users, and the user profiles and the match making are implemented on a central server. The users' profiles are associated with their Bluetooth identifiers [15].

Serendipity takes user privacy into account by revealing information on the encountered user only when the similarity score is high enough. Only the common interests are revealed, in addition to optional user specified contact information. Since Serendipity requires for Bluetooth to be on, it exposes the user to the privacy threat of linkability (section 3.2.1), allowing someone to track people's movements based on the Bluetooth identifier.

2.10.4 FindU

FindU allows the user to find the user with the profile that is the most similar to his/hers from the group of users in proximity. The existence of a trusted third party is not assumed.

FindU uses two privacy levels; higher privacy level leverages PCSI (private cardinality of set-intersection) and the lower one leverages PSI (private set-intersection) protocols. The efficiency of these protocols is improved by using SMC (secure multi-party computation) and polynomial secret sharing, and several key enhancements [14].

FindU is more secure and privacy preserving than SoInx, but its functionality is slightly different; e.g. FindU assumes established secure communication channels between each pair of users in proximity. SoInx use cases using FindU protocols would require a lot more computation and message passing than using SoInx protocols.

3 PRIVACY

This work focuses on privacy issues. Hence a more general introduction to privacy is beneficial for understanding the general framework related to privacy issues.

Privacy properties and threats are introduced, and the properties are discussed in the context of SoInx. The privacy considerations in ad hoc networking context are discussed. The security mechanisms of SoInx for maintaining security and privacy are introduced in Chapter 4.

3.1 Terminology

Privacy issues have some established terminology [16]. Terms are used in later sections describing security mechanisms of SoInx.

Here some common privacy related terms are explained:

Action. Something a user or the client application does in the context of the privacy environment in examination. This can be for example sending a message.

Actor, also used: **subject**. Typically a user in the context of the privacy environment in examination. For example a sender of a message.

Actee, also used: **object**. A target of an action that an actor does. For example the recipient of a message sent by an actor.

IOI, Item of Interest. Item of interest is a piece of information that some actor might be interested in. Examples: an action, a message, a location of a user, a user's identity.

Hostile client, Also used: **attacker, adversary client**.

A hostile client is an actor in the environment that intends to breach the security or privacy of the user or users. An example hostile client that aims to resolve whether an encountered user has been seen before or not was developed. This is discussed more in Chapter 6.

3.2 Privacy Properties and Threats

Privacy properties are usually divided into hard privacy and soft privacy [16]. Privacy can also be examined via privacy threats; each privacy property has a corresponding privacy threat.

The concept of privacy can also be divided into server level privacy and network level privacy. Since SoInx assumes the nonexistence of a trusted third party (i.e. a server), this work focuses solely on the network level privacy.

Hard privacy properties and their explanations and the corresponding privacy threats are listed in Table 3.1.

Table 3.1: Hard privacy properties and threats

Privacy property	Explanation
Unlinkability	<p>Corresponding privacy threat: Linkability.</p> <p>Unlinkability from an attacker's point of view means that it is not possible to sufficiently distinguish whether two IOIs such as subjects, messages, or actions are related or not. Linkability is the negation of unlinkability i.e. the attacker can sufficiently distinguish whether two IOIs are related or not.</p> <p>Essentially, unlinkability means hiding the link between two or more actions, identities, or pieces of information [17].</p>
Anonymity & Pseudonymity	<p>Corresponding privacy threat: Identifiability.</p> <p>Anonymity of a subject means the subject is not identifiable within a set of subjects, the anonymity set. The anonymity set is the set of all possible subjects.</p> <p>Essentially, anonymity means hiding the link between an identity and an action or a piece of information. Since actions, identities, and pieces of information can be considered as IOIs, anonymity can be described in terms of unlinkability.</p> <p>Pseudonymity refers to the use of pseudonyms as identifiers. An identifier of a subject other than its real name would be considered a pseudonym. Pseudonymity does not automatically imply anonymity between a subject's true identity and its pseudonym(s).</p> <p>It is possible to build a reputation on a</p>

	pseudonym. Several pseudonyms can be used for different purposes.
Plausible deniability	<p>Corresponding privacy threat: Non-repudiation.</p> <p>Plausible deniability in the context of privacy means the ability to deny having performed a certain action that other parties should not be able to confirm or contradict.</p> <p>This privacy property is in direct contradiction with the classical security property non-repudiation.</p>
Undetectability & unobservability	<p>Corresponding privacy threat: Detectability.</p> <p>Undetectability of an item of interest means that the attacker cannot sufficiently distinguish whether it exists or not. For example messages in a transfer channel should not be able to be distinguished from random noise.</p> <p>Unobservability of an item of interest means that 1) undetectability of the IOI from the perspective of the uninvolved subjects, and 2) anonymity of the subject(s) involved even with regards to other subjects involved in the IOI.</p>
Confidentiality	<p>Corresponding privacy threat: Disclosure of information.</p> <p>Confidentiality means hiding the data content or the controlled release of the data content, for example transferring an encrypted message.</p>

Soft privacy properties and their explanations and corresponding privacy threats are listed in Table 3.2.

Table 3.2. Soft privacy properties and threats

Privacy property	Explanation
Content awareness	<p>Corresponding privacy threat: Content unawareness.</p> <p>Content awareness property is more related with service providers such as Web 2.0 services. It makes sure that users are aware of what</p>

	information related to them is stored and that only the minimum necessary information should be used for functionality it is needed in.
Policy and consent compliance	<p>Corresponding privacy threat: Policy and consent non-compliance.</p> <p>Policy and Consent Compliance property means that the user should be informed beforehand about the system's privacy policy, and the user should be allowed to consent to this policy in compliance with the legislation.</p>

3.2.1 Unlinkability

Since SoInx is a social application, plenty of actions and pieces of information should be related to one another. For example, if an interesting person is discovered, the user should be able to chat with that person. Therefore linkability should clearly be allowed in a controlled manner.

In many cases unlinkability is generally desirable, for example adversary observers should not be able to follow users' movements around the city.

3.2.2 Anonymity and Pseudonymity

SoInx does not explicitly utilize the concept of identity or pseudonymity; the users are assumed to be anonymous between one another. However, the network identifier of the device running SoInx could be used as an implicit pseudonym. The network identifier changes periodically, but during the time interval it is constant, it is used for identifying the users who are sending messages related to SoInx functionality.

3.2.3 Plausible Deniability

In SoInx the ability to deny having performed a certain action that other parties should not be able to confirm or contradict, is not built into the application. There are still implicit ways to achieve this; for example a user may claim that a certain action was done by someone else who was spoofing his/her network identifier.

3.2.4 Undetectability and Unobservability

Undetectability in SoInx is fairly limited; when messages are sent, any observer can determine that it was sent, and can even see the format of the message. Only some parts

of the messages (the encrypted payload) is what would be considered undetectable in SoInx.

The second part of unobservability (anonymity of the subject(s) involved even with regards to other subjects involved in the IOI) is usually upheld; SoInx does not reveal user's identity unless the user explicitly does so by chatting or sharing contact information.

3.2.5 Confidentiality

Confidentiality in SoInx applies mostly to the way the common interests between users are shared, using the common interest verification protocol, and how further communications are encrypted using the set of common interests as the encryption key (section 4.2). This would be considered as controlled release of data content.

3.2.6 Content Awareness

Since SoInx does not use any central server, this privacy property does not apply to SoInx. However, the way SoInx allows users to define the privacy settings for different kinds of interest sources is tangential to this privacy property. The users should be made aware of how these sources of interests and the interests themselves are used for the application.

3.2.7 Policy and Consent Compliance

Since SoInx in its current form is just a prototype, policy and consent compliance has not really been an issue. Any kinds of explicit privacy policies were not drafted for this application.

If SoInx was released commercially, the users should be made aware of the ways SoInx affects the privacy of the user. This would probably take the form of a terms of use approval before using SoInx.

3.3 Privacy in Ad Hoc Networks

Privacy in the context of ad hoc networks should be considered separately from server based privacy. With the peer-to-peer type environment of ad hoc networks, there is no single trusted party (i.e. the central server) that is relied upon for implementing sufficient security measures for preserving user privacy. Instead, every actor has to implement their own security mechanisms.

3.3.1 Location Tracking Based on Network Identifiers

Devices using ad hoc networking use some kind of network identifier to access the network. One example is the MAC address of the WLAN adapter. An adversary controlling multiple access points can use the network identifier to track the location of a user. In the case of WLAN, very accurate (up to 1 meter accuracy) location can be determined if multiple access points are used to collect Signal-to-Noise-Ratio measurements [18].

This type of location tracking can be avoided by using disposable interface identifiers; for example by changing the WLAN MAC address periodically [19].

3.3.2 Important Considerations

There are certain considerations that apply especially well in the context of proximity based ad hoc networking. These should be taken into account when developing applications running in this context.

Broadcasting. In ad hoc networking context every sent message is broadcasted via a radio channel, which means that everyone with the proper equipment in close enough proximity can see every piece of data that is sent on the ad hoc network. Some type of encryption is highly advisable for any private data directed to a specific user.

Man-in-the-middle. When sent data is relayed through an intermediary device in the network, that device is by definition a man-in-the-middle, so man-in-the-middle attacks have to be taken into account [20]. These attacks can include various adversary actions, ranging from simply observing the sent data and trying to break the encryption, to tampering with the data before it is relayed to the other devices.

Battery life. These kinds of applications are usually used with a mobile device, which means that the battery life has to be accounted for when deciding security and privacy measures. Computational intensity and bandwidth requirements must be balanced with the efficacy of the security protocols.

4 SECURITY MECHANISMS OF SOINX

This chapter describes the security mechanisms SoInx uses for maintaining user privacy and data security. The main security mechanism is the common interest exchange protocol based on sending hashed versions of interests between users. The verified set of common interests are also used as a basis for encrypting further messaging between the users.

The section 4.1 explains the high level overview of the common interest exchange protocol and the section 4.2 explains the encryption scheme. Later sections go into the details and specifics. The section 4.6 gives an example of the interest sharing and verification process when two users have two common interests.

4.1 Common Interest Exchange Protocol

SoInx communicates with other devices by broadcasting messages. Since the messages often include things that the user finds interesting, it is important that only the intended receivers can decipher the contents of the message; it is not desirable to share this potentially personal information with everyone in proximity. When trying to discover other people with similar interests, only those interested in a certain subject should be able to deduce that you also are interested in that same subject.

SoInx uses its own common interest exchange protocol to perform this kind of profile information exchange securely. The protocol between two users is based on message chains where the initiator sends a query message (section 4.3.1), the other one responds with a match message (section 4.3.2) to which the original initiator responds with another match message. The match messages are used to verify for the recipient that common interests were recognized. Both parties work periodically as an initiator; the more these query-match-match chains are sent, the more common interests can be discovered. These messages do not contain actual interests, only query and match IDs which are derived via one-way hashing (section 4.4).

This is not always sufficient for solving every privacy threat. An adversary observer can deduce some information from the type of data sent between clients; for example, if after a few query messages two clients start sending several match messages, the adversary observer can conclude that they have at least some common interests. This means that some of the information is still detectable in the SoInx communication.

The network identifier (i.e. the MAC address) of the mobile device can be used to identify a user. This is not a desirable feature, and it is handled on the device level; the network identifier is changed periodically to improve the privacy and anonymity of the user.

4.2 Encryption Protocol

In addition to a secure interest exchange scheme, any further messaging between two users (such as chatting or sending the contact card) must be done securely, i.e. encrypted. In ad hoc networks with previously unseen encounters, the local context must be used to compose a common encryption key. This causes privacy problems that are difficult to solve, because plenty of things in the local context are also accessible to adversary observers in proximity. SoInx bases the encryption on common interests.

After verifying common interests between two SoInx users, the set of common interests is used for encrypting further communication (e.g. chat messages) between the two SoInx users. The encryption algorithm SoInx uses is AES (Advanced Encryption Standard) [21]. AES is a symmetric-key cryptographic algorithm, so the decryption key is the same as the encryption key. The key is constructed from the set of common interests discovered by the interest exchange protocol. The encryption key is an SHA1 [22] hash of the concatenation of all the verified interests sorted alphabetically plus the current salt (section 4.4.1), separated by null characters.

These encrypted messages are placed in the payload field of match messages. These types of match messages differ slightly from the match messages used purely for interest verification purposes. Section 4.3.2 presents these in detail.

4.3 Solnx Messages

SoInx is sending two types of messages: query and match messages. Query messages are broadcasted when new devices appear in the range of the ad hoc network, although the broadcasting is throttled to avoid message flooding when several new devices appear consecutively in short bursts. When no new users are encountered, the query messages are sent periodically and the time interval between query messages becomes longer if no new users are encountered. Match messages are sent to verify that one or more interests were recognized from query messages. More interest verification is done in yet another match message, sent by the original query message sender. Match messages may include an encrypted payload; this payload can be considered sent through a secure communications channel between two SoInx users.

The messages below are described using JSON-representation, but in the application the messages are encoded with Bencode [23]. Bencode is a data encoding scheme suited

for storing and transmitting loosely structured data. Bencode was chosen because it is fairly efficient and for each possible value, there is only one valid Bencoded value. This latter feature permits simple comparisons and identical hashing values between identical data structures.

4.3.1 Query Messages

Query message is the basic broadcasting message the device will be sending out that contains a randomly chosen and randomly ordered list of the interests (20 at maximum per query message) in query ID form, derived from hashed interests (section 4.4). For each chosen interest one query ID of the five possible is chosen randomly. The query ID list in the message is in a string format; the different query ID values are concatenated together.

The number of query IDs per message is limited to 20 because otherwise the user could possibly be recognized by using the number of query IDs in a message as a fingerprint.

The format of the message is:

```
{'type' : 'query', 'ids' : list-of-query-ids}
```

4.3.2 Match Messages

Match messages are used for common interest verification and also for sending encrypted messages (payload) for specific users.

The format of the message is:

```
{'type' : 'msg', 'ids' : list-of-match-ids, 'data' : encrypted-payload, 'iv' : random-initialization-vector-for-encryption, 'pad' : padding, 'tok' : token}
```

The list of the match IDs is non-empty only when the match message is used for common interest verification purposes, as explained in section 4.1. When match messages are used for secure encrypted messaging between users, the match ID list is empty. The field **tok** refers to a token that denotes the set of verified interests at the time of the message construction. The field **data** contains an encrypted payload for communicating between users. The field **iv** contains a random initialization vector used for the encryption and the field **padding** contains the number of padding bytes used for the encryption.

If the match message is used for common interest verification purposes, the encrypted payload is just information about the interests being verified. In the receiving device this information is simply ignored after decryption in the current version of the prototype. In these cases the match ID list of the message is also empty.

If the match message is not for common interest verification purposes, then the payload includes private messaging between two users such as chat messages, like-notifications, or contact card information.

4.4 Query IDs and Match IDs

Query IDs and match IDs are generated via one-way hashing. Two hashes are calculated from every interest: the query hash and the match hash. Query refers to query messages and match refers to match messages, both of which are used in the common interest verification process.

A query hash is divided into query IDs and a match hash is divided into match IDs. The hashing algorithm used in SoInx is SHA1, but the concept can be extended to function with other hashing algorithms as well. The query hash and the match hash are salted with salts derived from the time of day and the salt is changed periodically. Everyone aware of the time is able to derive the salts; the salt is same during the whole time period before it is changed. The processes for deriving query IDs and match IDs are visualized in Figure 4.1 and Figure 4.2 respectively.

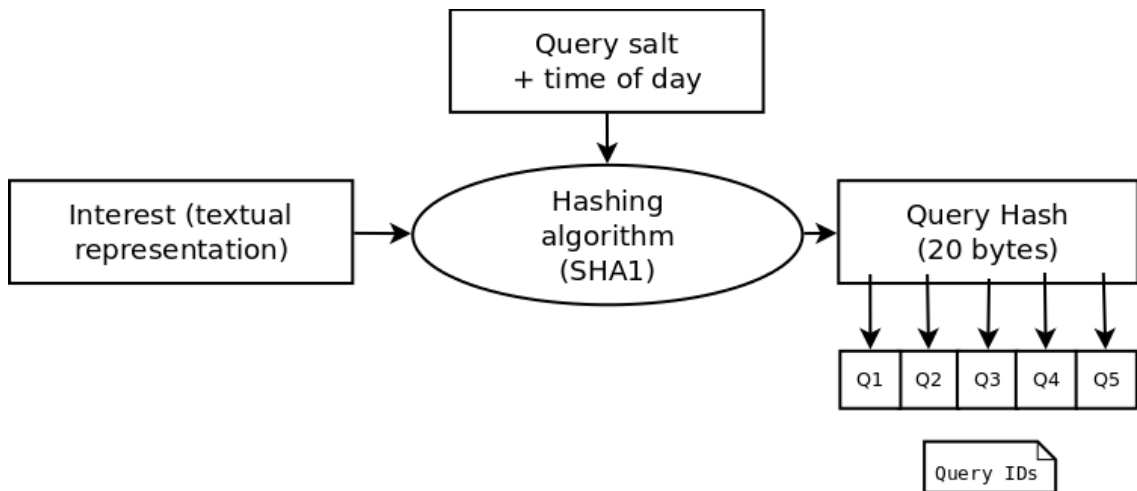


Figure 4.1: Deriving query IDs

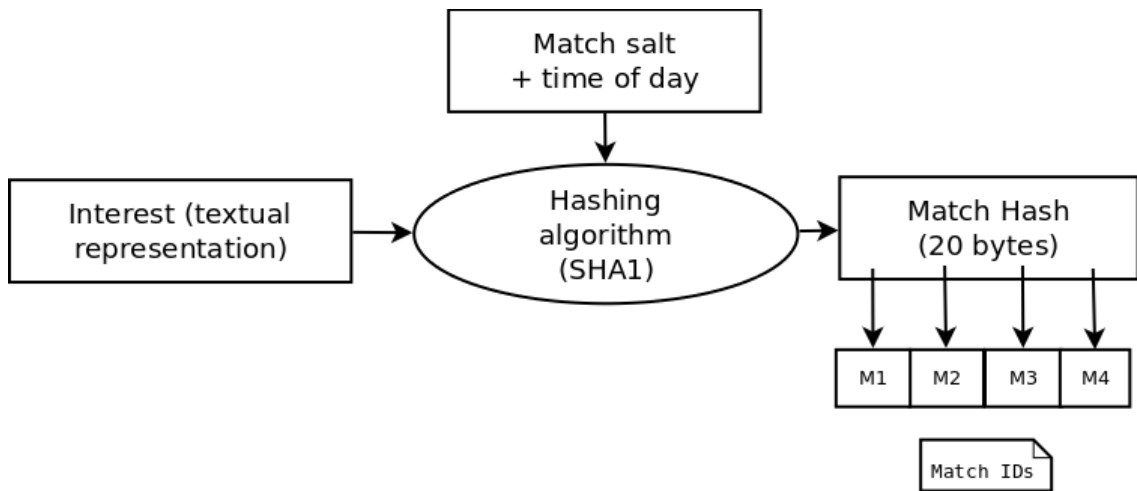


Figure 4.2: Deriving match IDs

In the prototype implementation the exact input for the SHA1 hashing algorithm is a concatenation of five text strings. For the query hash: the textual representation of the interest, the null character, the string literal "query", another null character, and the current salt. For the match hash: the textual representation of the interest, the null character, the string literal "match", another null character, and the current salt.

SHA1 produces hashes with the length of 20 bytes [22]. One query hash of an interest is divided into five query IDs (Q1 through Q5 in Figure 4.1). This means that every query ID is 4 bytes long. The first query ID, Q1, is simply the first 4 bytes of the query hash, the second query ID, Q2, is the next 4 bytes and so on. The match hash is similarly divided into four match IDs (M1 through M4 in Figure 4.2), which means that every match ID is 5 bytes long.

Several query IDs and match IDs per interest are generated due to privacy issues (section 5.6). This makes it more difficult for an adversary observer to recognize the user using the set of seen query IDs as a fingerprint. The actual numbers (5 and 4 for query IDs and match IDs respectively) were chosen somewhat arbitrarily, but it has some implications: query ID collisions are more likely than match ID collisions. These collisions can make certain types of attacks more difficult while still keeping user experience intact. For reference: to reach a 1 % probability of a collision in a single query message of 20 query IDs, a set of 431659 unique interests is required. With a match message of 20 match IDs, the required number of interests is 138130764.

These query IDs and match IDs are used in query and match messages, when referred to as interests in an encrypted form. Each query message contains query IDs for 20 interests; one query ID per interest, which is chosen randomly.

4.4.1 Salt

Salts for both query and match hashes are based on the time of day in such a way that each salt is valid for a certain time interval at a time. The prototype implementation uses one hour long time intervals.

Some crypto applications generate a new salt for every sent message and include this salt into the message. This is not realistic for SoInx purposes; then query IDs would have to be recomputed for each interest for every received message when searching for matching interests. This time-based salting requires this recomputation only once an hour.

The current salt can be derived by taking the current time in seconds and dividing it by the time interval (also in seconds). The result of the division is rounded down:

$$salt_{current} = \left\lfloor \frac{current\ time}{time\ interval} \right\rfloor$$

The prototype implementation uses Unix time (also known as POSIX time) for the current time. It is defined as the number of seconds that have elapsed since 00:00:00 Coordinated Universal Time (UTC), Thursday, 1 January 1970 [24].

When the application is run on the simulator, the current time is received from the simulator, which uses its own time concept. The simulated time is simply the elapsed seconds since the start of the simulation (the time in the simulated environment, not the actual time).

4.5 Handling Received Messages

When receiving messages, the first thing to do is trying to decode it, the assumption being that it is in a Bencode format. If the decoding fails, the message is either garbled or it is sent by some other application, and the message is not SoInx compatible. In this case, the message is simply ignored.

If decoding succeeds, the field 'type' of the message is examined. The only acceptable 'type' values are 'query' (a query message) and 'msg' (a match message), otherwise the message is again ignored.

4.5.1 Handling a Query Message

Processing a query message constitutes mainly of trying to figure out whether you have any of the same interests that the query message contains. If query IDs are recognized, the verification process will be started, where SoInx attempts to confirm that the different SoInx users indeed have the same interests. The process is showcased in section 4.6

Whenever a query message is seen, it is first confirmed that the field 'ids' of the Bencoded message is a string, otherwise the message is ignored. The contained query IDs are examined. For each recognized query ID, a corresponding interest is taken, and the set of recognized interests (i.e. corresponding match IDs) is used for the match messages during the verification process.

The recognized interests are marked as unverified common interests between the sender of the query message.

4.5.2 Handling a Match Message

When a received match message includes a non-empty match ID list, this means that the sender has started a verification process. The sender aims to verify some common interests. If all the match IDs are recognized, the recognized interests are used to form an encryption key that is used in decrypting the payload of the match message.

If the decryption succeeds, the receiver can be sure that the interests recognized from the match ID list are common for both parties. This is when the unverified common interests can be marked as verified common interests. The sender is still unsure (i.e. the interests are marked as unverified common interests from the sender's point of view); the verification has to happen in the other direction as well. This is done by sending a similar verification match message to the original sender of the received match message. Next section presents a detailed example of the common interest verification protocol.

When the match ID list of the match message is empty, the message is used for communicating encrypted private information. The verification steps described above are skipped, and the encrypted payload is simply decrypted with the set of verified common interests, and the contents of the decrypted message are used for further processing, such as displaying the received chat message, displaying the received contact information, etc.

4.6 Common Interest Verification Process Example

The verification process for common interests consists of sending several query and match messages. The process between a SoInx client **A** and a SoInx client **B** who have two common interests **i1** and **i2** is as follows:

Phase 1. Client **A** sends a query message. Client **B** sees the query message and recognizes that the interests **i1** and **i2** correspond to two query IDs in client **A**'s query message. Client **B** marks interests **i1** and **i2** as unverified common interests.

Phase 2. User **B** constructs a match message with a list of match IDs that correspond to the interests **i1** and **i2**, and a payload encrypted with an encryption key derived from

i1 and **i2**, and broadcasts it. The payload is just information about **i1** and **i2** in a textual form.

Phase 3. Client **A** sees the match message and recognizes that the match IDs found in the message were derived from the interests **i1** and **i2**. Client **A** constructs an encryption key from **i1** and **i2**, and tries to decrypt the payload in the match message. The decryption succeeds; now client **A** can mark **i1** and **i2** as verified common interests.

Phase 4. Client **A** constructs a match message with match IDs that correspond to the interests **i1** and **i2** and a payload encrypted with an encryption key derived from **i1** and **i2**, and broadcasts it.

Phase 5. Client **B** sees the match message and recognizes that the containing match IDs correspond to the interests **i1** and **i2**. Client **B** constructs an encryption key from **i1** and **i2**, and tries to decrypt the payload in the match message. The decryption succeeds; now client **B** can mark interests **i1** and **i2** from unverified to verified common interests.

After this process, both users **A** and **B** now know that they share two common interests **i1** and **i2**.

5 PRIVACY ANALYSIS OF SOINX

This chapter discusses privacy issues in a mobile social application running on ad hoc networks, such as SoInx, and describes how the security mechanisms of SoInx work to solve those issues.

There are rigorous and formal privacy threat analysis frameworks [17]; the approach here is less formal: each privacy issue description explains how the user's privacy is being affected, then the SoInx solution is discussed. If the current version of SoInx does not address the issue, possible solutions are proposed.

5.1 Using Network Identifier For Linkability

Network identifiers such as MAC addresses or Bluetooth addresses can easily be used to link encountered users with previous encounters, henceforth breaking the privacy property of unlinkability. One way to solve this is to change the network identifier periodically. This may require hardware support.

SoInx application assumes this type of solution. The network identifier change coincides with the salt change described in section 4.4.1, causing the generation of new query and match IDs. The simulator provides this kind of changing network identifiers, as described in section 2.7.2.

During the time interval the network identifier is constant, it is trivial to link messages, but this is desired behavior and required for the functionality of the application.

5.2 Broadcasting Sensitive Information

User's interests may contain some private and sensitive information. Therefore, it should be up to the user to decide which interests can and cannot be broadcasted in query messages, and matched using the verification protocol.

To achieve this, SoInx has a user interface to define which sources of information are sensitive enough to be completely private in the context of SoInx application. This is described in section 2.6.4.

5.3 Seeing Broadcasted Interests

User's interests may contain some private information. It is not desired that every interest from a broadcasted query message can be examined by every actor operating in close enough proximity, but the user might still wish to reveal these interests to people with same interests.

The solution to this in SoInx is that instead of using plain text representations of interests, query (and match) IDs of the interests are used. Since query and match IDs are generated via one way hashing, only those with the same interests can recognize them.

5.4 Eavesdropping on Private Communication

When two SoInx users have established a set of common interests, they may wish to contact each other for example for chatting purposes. It is not desirable that everyone in proximity can examine the contents of these messages, as the messages might contain private information.

SoInx solves this by encrypting all further messages between SoInx users after establishing common interests. The encryption key is derived from the verified common interests.

5.5 Using Query ID Sets to Link Encounters

The set of seen query IDs from a user can be used as a fingerprint to recognize that user. Whenever a new user is encountered, its broadcast message can be compared to earlier broadcast messages. If the broadcast message has the same set of query IDs as an earlier broadcast message, it is almost certain that the user is the same one as in the earlier encounter.

SoInx solves this by generating new query IDs periodically. This problem still exists during the time interval the query IDs are unchanged. That is why SoInx also uses several different query IDs for each interest. The query ID for a certain interest used in a query message is chosen randomly. Query ID generation mechanics are discussed in more detail in 4.4.

5.6 Using Query ID Set Sizes as Fingerprints

Using the number of query IDs a user is sending could be used as a rough, albeit uncertain, fingerprint to recognize whether the user has been seen before.

The SoInx makes this harder using a two-fold solution; there is a limit on the query IDs a query message can contain. In the prototype implementation it is 20. Another mechanism is the generation of several query IDs for each interest, in the prototype

implementation 5 query IDs. In query messages, a query ID for a certain interest is chosen at random.

5.7 Histogram of Query IDs as a Fingerprint

Some interests can be more important to a user than others. Therefore, it could be desirable to send query IDs of those interests more often than query IDs of less interesting ones. This would more quickly find matches that the user deems important.

A downside of this is that the frequencies of different query IDs broadcasted can be used to form a histogram. This histogram could be used as a rough fingerprint to recognize users.

SoInx solves this by always using a random sample of interests for each broadcasted query message.

5.8 Physical Observations

Since a SoInx device communicates with other devices in close enough proximity, in some situations it is possible to recognize the SoInx user just by looking around. For example, if SoInx clients are communicating and you can see only one person closeby, it is likely that the person is carrying the device running the SoInx application. This type of situation can easily break anonymity.

SoInx application does not address this type of privacy threat. One way to solve this is to just shut down the application in situations like this. This would require user action and vigilance from the user to notice these kinds of situations.

5.9 Other Applications Leaking Identity

It is possible that other applications running on the device can leak the identity of the user. Therefore in spite of all the precautions related to privacy and data security, the privacy of the user could be trivially compromised.

This is not addressed in the SoInx application, but one way to solve this could be to make sure that SoInx is not using the same network identifier as other applications. This may require hardware support.

5.10 Recognized and Unrecognized Query ID sets

The set of recognized interests from query IDs, combined with the number of unrecognized query IDs in relation to the total number of query IDs seen from a SoInx user can possibly be used as a fingerprint to link a new encounter to a previously seen

SoInx user. This would require an interest database for recognizing some of the query IDs seen, and this type of attack is analogous to a dictionary attack.

The implemented hostile client described in section 6.2 shows that this could be a feasible attack that breaks the privacy property of unlinkability of SoInx users.

5.11 Summary of Security Mechanisms for Maintaining Privacy in Solnx

The main security mechanisms of SoInx for maintaining privacy are the encryption protocol based on common interests between two SoInx users, and the common interest exchange protocol based on one-way hashing of interests.

The protocols allow for SoInx users to establish a set of common interests between two users, and use these common interests as an encryption key to encrypt any further communication between users. The protocols have several implementation choices that are not initially obvious (such as using several query and match IDs, forced randomness of chosen interests in query messages, etc.) but as discussed in this chapter, they prove to be important components in maintaining user privacy.

6 HOSTILE CLIENT

One way to evaluate privacy and security of a social application is to create hostile clients to study how well data security and privacy of the users are maintained. Hostile clients work in the same environment that the application, but instead of participating in the ordinary functionality of the application, they try to perform actions to breach the user privacy or security. Possible ways to achieve this could include for example sending specifically crafted messages to reveal something unwanted about the receiving user, or just passively listening to messages other users are sending to one another and exploiting the possible inherent insecurities in the communication protocol.

There are several different types of hostile clients that could be run in the SoInx environment to extract unwanted information. A hostile client could just be a passive observer, or it could disguise itself as an ordinary SoInx client while examining the communication between other SoInx clients and attempting to find holes in privacy and data security. One hostile client was implemented during the SoInx project. It is discussed in more detail in section 6.2.

6.1 Hostile Client With Massive Interest Database

Since the encryption of SoInx messages are based on common interests between parties, a hostile client with a database of all the possible interests would theoretically be able to break the encryption on every message sent by the SoInx application, and also to recognize with a high probability if an encountered user has been seen before. This type of hostile client would work well as a passive observer. This type of attack is equivalent to a dictionary attack.

Having a database of all the possible interests is not feasible, since the number of possible interests is limitless. As an example, in our crawled set of 46500 Facebook profiles, there are over 3 million unique interests (all the unique likes and friends found in the profiles). Taking into account other sources of interests, constructing a comprehensive set of interests would be computationally difficult. However, the crawled set of Facebook profiles shows that it is feasible to mine social networking sites for millions of interests.

Even if only some of the interests can be recognized from query messages, this attack may still be used to link the encountered users with previous encounters; the

recognized interests and the number of unrecognized ones could be used as a fingerprint for recognizing SoInx users. As the example implementation shows, this approach is a feasible way to breach user privacy in the SoInx environment.

6.2 Implementation

The type of hostile client implemented during the project was a passive listener that functions in the SoInx environment observing messages. It uses an interest database and intends to recognize users it has seen before, therefore linking new encounters with previous encounters. Performing this with a high enough success rate would break the privacy property of unlinkability, and it would allow to track movements of SoInx users.

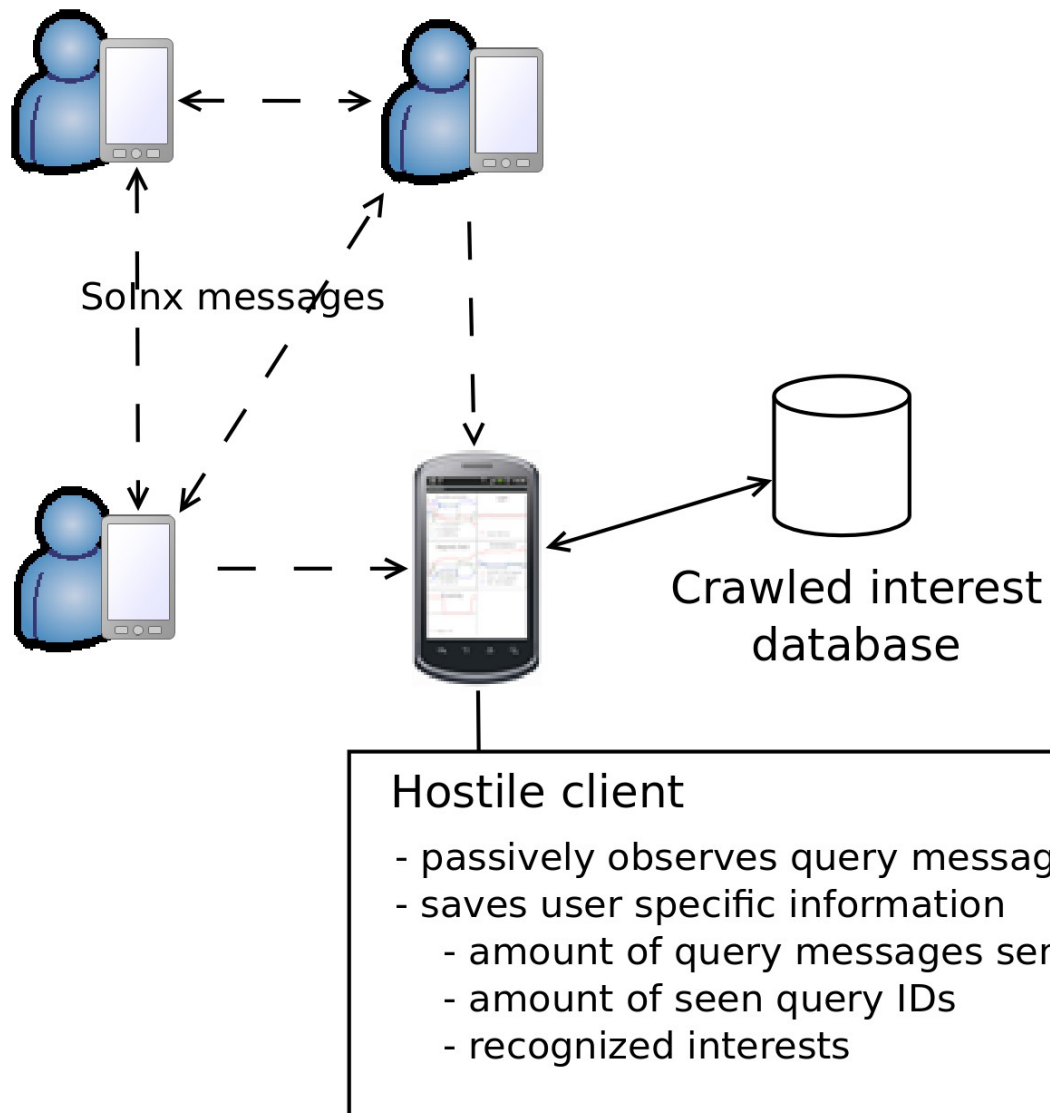
The high level functionality of the hostile client is illustrated in Figure 6.1. The hostile client collects data, and the data is analyzed (for attempting to link encounters with each other) as post-processing. This type of separation is useful because it allows for multiple hostile clients to collect data and send them to an aggregate database via the Internet. The aggregated data can then be analyzed as a whole. The implementation consisted of only one hostile client.

6.2.1 Collected Data

The hostile client collects data whenever it sees SoInx users sending messages to one another. The only messages observed are query messages.

The data observed in a query message is associated with the network identifier of the SoInx user and with the time of the sent message. Since the network identifier is changed periodically, the collected data is later analyzed to deduce whether the user with one network identifier is the same user with a different network identifier at a different point of time.

The collected data for a single network identifier is derived from every broadcasted query message seen during the time block the user has a constant network identifier.



6.2.2 Post-processing

In the post-processing phase, the different encounters are compared with one another and collected data related to those encounters is analyzed to deduce whether two encounters are from the same SoInx user or not.

In the analysis, following data is considered when comparing two encounters **E1** and **E2**: the total number of recognized interests (for both encounters) **r1** and **r2**, the total number of unrecognized query IDs (for both encounters) **u1** and **u2**, the total number of recognized interests that are found in both encounters **c** (c as in common).

The number of seen query IDs for an encounter depends on the number of seen query messages, and the number of seen query messages depends on how long the encounter has lasted. This must be taken into account when trying to decide whether two

encounters are from the same user or not. To account for this, ratios were used (calculated only if $r1 > 0$ and $r2 > 0$):

$$\mathbf{ratio1} = \mathbf{u1} / \mathbf{r1}$$

$$\mathbf{ratio2} = \mathbf{u2} / \mathbf{r2}$$

The preliminary algorithm for this decision was fairly simple:

The encounters **E1** and **E2** are considered same if and only if

$$c > 0 \wedge \frac{\min(\mathbf{ratio1}, \mathbf{ratio2})}{\max(\mathbf{ratio1}, \mathbf{ratio2})} > 0.6$$

6.3 Results of Hostile Client Execution

An example run was done with a database of 100000 interests. These interests were the most common interests found in the database of crawled Facebook profiles. The total number of unique interests found in the profiles was over 3 million, so the hostile client interest database size was only about 3 % of the possible number of encountered interests.

In total, there were 2814 comparisons between different encounters. 142 of those were from same users, and 2672 of those were from different users.

Of those 142 cases where the two encounters were from the same user, the hostile client was able to recognize 119, and in the remaining 23 cases it falsely assumed the encounters to be from different users.

Of those 2672 cases where the two encounters were from different users, the hostile client was able to correctly recognize 2608, and in the remaining 64 cases it falsely assumed the encounters to be from same users.

So 83.8 % of the linkable encounters were recognized. This percentage is likely to improve with a larger interest database and by using more sophisticated algorithms for deducing whether two encounters are from the same user. From a privacy standpoint this poses a linkability threat in the SoInx application.

7 EVALUATION OF SOINX

This chapter discusses the evaluation of the SoInx application based on the user testing. The evaluation of SoInx security mechanisms for maintaining privacy of the users is also discussed, and several avenues for improvements are presented.

7.1 User Tests

User tests were conducted during the SoInx project. The SoInx application was tested, mainly to find bugs and to evaluate the user interface and to get new ideas for improving the software.

The tests were conducted on a desktop computer, using the simulated environment. The visualization of the simulation (section 2.7.2) and the SoInx client connected to the simulation were shown on the same screen. The user being tested was explained their simulated location would be shown on the map, and the SoInx client view was what would be shown on their device. An ongoing user test is shown in Figure 7.1.



Figure 7.1: A user test subject interacting with the SoInx application connected to the simulator.

The simulation lasted for 5 days of simulated time with 899 simulated user clients and 1 user client taking input from the test subject. The simulated area was the city of Tampere.

A single user test lasted for about 2 hours, after which the test subject answered questions regarding the SoInx client and the test. Earlier answers were used to improve the test questions and to make them more specific. The answers provided also good input for improving the simulator and the SoInx client.

Whenever a test had an encounter in the simulation, they could use the user interface to indicate several possible things: they liked the encountered person, they wished to chat with the encountered person, and/or they wished to share contact information with the encountered person. Not choosing to do some or any of those actions was also considered significant data, and was recorded.

The concern over privacy was almost non-existent among the test subjects. This was even without explaining the security mechanisms that SoInx uses. Among reasons were that they did not care if Facebook data (likes and friend relationships) was public, and that they were aware that this kind of data is already known for corporations such as Google or Facebook.

The results of the user tests are covered more thoroughly in the thesis of Saku Rautiainen [25].

7.2 Encryption Based on Common Interests

The interest database based adversary client proved to be feasible, especially when interests used in SoInx are mainly derived from social networking sites that are easy to crawl. Therefore, the encryption that is based on shared interests may need some reconsideration, especially when privacy and security are more essential requirements of an application.

7.3 Ideas for Improvement

There are several possible avenues for improving the privacy of the users of the SoInx application (or similar social applications running in the environment of mobile ad hoc networks).

7.3.1 Diffie-Hellman and Other Key Exchange Protocols

Diffie-Hellman key exchange method is a way for two parties to establish a shared secret key [26]. This key can in turn be used to encrypt ensuing communication between each other.

This key exchange and encryption could be used as an extra layer on the encryption protocol used by SoInx. The Diffie-Hellman encryption could be used before or after the common interest based encryption, but after the interest (query ID) broadcasting step, leaving user privacy still vulnerable to the example hostile client attack.

Using the Diffie-Hellman encryption before the basic interest broadcasting is feasible only when the number of proximate SoInx users is fairly low, since every user would have to establish an encrypted channel with every other user it encounters before it can broadcast its interests (query IDs), which would multiply the amount of communication required for the basic interest sharing use case. This is in conflict with the requirements of minimizing network traffic and power consumption. Also, the hostile client could just establish a Diffie-Hellman based encrypted channel with each user it encounters, and then view the same information it could see by just examining the broadcasted query IDs.

As an extra encryption layer this would eliminate (i.e. make it computationally difficult) the attack where the adversary client knows all the common interests between communicating parties, and uses the information to calculate the shared encryption key used to establish the encrypted channel between two users.

Diffie-Hellman is also vulnerable to a man-in-the-middle attack [26].

7.3.2 Tweaking Hashing Parameters

The algorithms and parameters used for hashing can be modified to make calculations for interest verification steps more difficult, therefore making calculation based attacks computationally more difficult. This has the unfortunate effect of making the normal application use calculations to consume more power, so a proper balance has to be found.

The hashing algorithm could be swapped with a more difficult one, or even with an algorithm where the computational difficulty can be adjusted such as bcrypt [27]. Also more hash values could be computed from an interest, and subsequently more query IDs could be generated from the interests, to make associating query/match IDs with their original interests more difficult.

7.3.3 Discouraging Easily Crawlable Interest Sources

Certain interest sources may be troublesome in the context of privacy. For example the crawled interest database proved that Facebook is fairly easily crawlable and therefore Facebook based interests are vulnerable to attacks based on interest databases. This concern is likely valid for many other social networking sites as well.

An example of an interest source that is easily generatable is phone numbers. Since phone numbers tend to obey a previously known form, generating a large phone number

based interest database is straightforward. Therefore phone numbers as such may not be the best possible interest source. A way to make phone numbers feasible is to attach the contact name to the phone number. This would require normalization of the names, such as lower casing all the letters, removing special characters, and sorting first, middle, and last names alphabetically, because interest matching requires interests to be exactly the same in their source form due to hashing algorithms used (even one bit difference produces completely different query IDs).

7.3.4 Recognizing and Reacting to Certain Attacks

SoInx could be modified to notice certain abnormal network behavior, that could indicate that someone is performing an attack on SoInx. If certain behavior is detected, communications could be turned completely off until the attack ceases.

For example, an attacker could be flooding the network with query messages in order to reveal faster what interests users have. Currently this type of situation is not detected.

Another example of abnormal behavior would be when someone has suspiciously high number of common interests with you. This may indicate that someone is performing an attack using a large interest database. This would be a good situation to scale down communications.

7.3.5 Context Depending Behavior

The application could be configured to behave differently depending on the context, in relation to privacy. This could take the form of different modes, such as Paranoid-mode, Feeling Social -mode et cetera, where the number of interests broadcasted and overall communications depend on the current mode. These modes could partly be deduced from the calendar / current profile of the device – e.g. when the calendar shows that the user is in an important meeting, SoInx communication could be turned completely off.

7.3.6 PSI & PCSI protocols

Research have already been done on how parties can privately compute an intersection of their respective sets of data. This can be done using PSI (private set intersection) protocols [28]. These cryptographic protocols could be used where the sets of data would be the interests and the intersection would be the common interests between two parties.

Instead of computing the intersection of data, it is also possible to compute only the size of the intersection. This can be achieved with a PCSI (private cardinality of set-interesection) protocol.

Instead of directly revealing the common interests with the encountered user, a layer could be added where only the number of common interests is revealed without

revealing the actual interests. If the number of common interests is above a certain threshold, the current interest matching protocol would be used. This may impair the user experience; the user may want an alert every time a certain very important interest is matched with another user.

8 CONCLUSIONS

This thesis introduced basics of privacy in the ad hoc networking context, emphasis on social applications. An example social application, SoInx, based on ad hoc networking, was introduced. Also a simulator environment for testing such applications was described.

A major objective of the thesis was to analyze how privacy of a user can be maintained in this type of applications. The security mechanisms of SoInx were described in detail and also how they solve certain privacy and security issues.

The security mechanisms were evaluated by implementing a hostile client that attempts to breach user privacy. The hostile client attempts to figure out whether newly encountered SoInx users had been seen before by utilizing a database of interests (this was basically a type of dictionary attack). This attack proved to be feasible; it breaks the privacy property of linkability, making it possible to track people's movements around the city. The common interest verification scheme and the shared interest based encryption were found lacking in the context of privacy.

Social applications running purely on ad hoc networking context also have some fundamental privacy threats based on physical reality, such as deducing the identity of a user when there are few people in adversary's physical proximity as described in section 5.8. Also the lack of a trusted party, such as in central server based applications, reduces the certainty of the privacy measures.

Currently almost all popular social applications are central server based. These include services such as Facebook, Tinder, Hilight, FourSquare, etc.

It is the opinion of the author that the current security mechanisms used in SoInx are not sufficient enough for maintaining user privacy, and the subject matter requires further research.

BIBLIOGRAPHY

- [1] C. K. Toh, *Ad Hoc Mobile Wireless Networks: Protocols and Systems*. Upper Saddle River 2002, Prentice Hall. 336 p.
- [2] Y. Wang, A. V. Vasilakos, Q. Jin, J. Ma, Survey on mobile social networking in proximity (MSNP): approaches, challenges and architecture. *Wireless Networks*, Volume 20, Issue 6, December 2013. Springer US. pp. 1295-1311.
- [3] J. Kulmala, *Developing Local Social Applications on Mobile Devices*. Tampere University of Technology, 2013. 56 p.
- [4] J. Kulmala, M. Vataja, S. Rautiainen, T. Laukkarinen, M. Hännikäinen, Social Index: A Content Discovery Application for Ad Hoc Communicating Smart Phones. *Advances in Service-Oriented and Cloud Computing, Workshops of ESOC 2013*, Málaga, Spain, September 2013. Springer Berlin Heidelberg. pp. 244-253.
- [5] Bluetooth SIG, *Bluetooth Specification Version 2.0*. Bluetooth SIG, November 2004
- [6] Nokia Conversations, Nokia Instant Community gets you social [WWW]. [accessed on 17.7.2013]. Available at: <http://conversations.nokia.com/2010/05/25/nokia-instant-community-gets-you-social/>.
- [7] Facebook, Facebook Login Overview [WWW]. [accessed on 27.1.2014]. Available at: <https://developers.facebook.com/docs/facebook-login/overview/>.
- [8] Twitter, OAuth [WWW]. [accessed on 27.1.2014]. Available at: <https://dev.twitter.com/docs/auth/oauth>.
- [9] P.E. Hart, N.J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, Volume 4, July 1968. IEEE. pp. 100-107.
- [10] Math Camp Inc, highlight. [accessed on 27.2.2014]. Available at: <http://highlig.ht/>.
- [11] NBCNews.com, Trendy iPhone app 'Highlight' is a privacy nightmare [WWW]. [accessed on 27.2.2014]. Available at: <http://www.nbcnews.com/technology/trendy-iphone-app-highlight-privacy-nightmare-386931>.
- [12] Z. Yang, B. Zhang, J. Dai, A. C. Champion, D. Xuan, D. Li, E-SmallTalker: A Distributed Mobile System for Social Networking in Physical Proximity. *ICDCS '10 Proceedings of the 2010 IEEE 30th International Conference on Distributed Computing Systems*, Via Opera Pia 15A, 16145 Genova, Italy, June 21-25 2010. Washington, DC, USA, 2010, IEEE Computer Society. pp. 468-477.
- [13] B. H. Bloom, Space/Time Trade-offs in Hash Coding with Allowable Errors.

- Communications of the ACM, Volume 13 Issue 7, July 1970. ACM. pp. 422-426.
- [14] M. Li, S. Yu, N. Cao, W. Lou, Privacy-Preserving Distributed Profile Matching in Proximity-Based Mobile Social Networks. *IEEE Transactions on Wireless Communications*, Volume 12, Issue 5, April 2013. IEEE. pp. 2024-2033.
- [15] N. Eagle, A. Pentland, Social serendipity: Mobilizing social software. *IEEE Pervasive Computing*, Volume 4, Issue 2, April 2005. IEEE. pp. 28-34.
- [16] A. Pfitzmann, M. Hansen, A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management. . Available at: http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf.
- [17] M. Deng, K. Wuyts, R. Scandariato, B. Preneel, W. Joosen, A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements, 2011
- [18] P. Bahl, V. N. Padmanabhan, RADAR: An in-building RF-based user location and tracking system. *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings, Volume 2, March 2000*. IEEE. pp. 775-784.
- [19] M. Gruteser, D. Grunwald, Enhancing location privacy in wireless lan through disposable interface identifiers: a quantitative analysis. *Mobile Networks and Applications*, Volume 10, June 2005. Kluwer Academic Publishers. pp. 315-325.
- [20] W. Trappe, L. C. Washington, *Introduction to Cryptography with Coding Theory*. New York 2005, Pearson. 257 p.
- [21] Federal Information, Announcing the ADVANCED ENCRYPTION STANDARD (AES), Processing Standards Publication 197, 2001
- [22] FIPS 180-4, Secure Hash Standard (SHS). [accessed on 15.4.2016]. Available at: <http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>.
- [23] Bencode, Bencoding Specification. [accessed on 16.5.2016]. Available at: <https://wiki.theory.org/BitTorrentSpecification#Bencoding>.
- [24] IEEE, The Open Group Base Specifications Issue 7 [WWW]. [accessed on 5.5.2014]. Available at: http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap04.html#tag_04_15.
- [25] S. Rautiainen, Prototype for Automatic Content Discovery in Mobile Ad Hoc Networks. Tampere University of Technology, 2013. 62 p.
- [26] W. Diffie, M. Hellman, New directions in cryptography. *IEEE Transactions on Information Theory*, Volume 22 Issue 6, November 1976. IEEE Press Piscataway. pp. 644-654.
- [27] N. Provos, D. Mazières, T. J. Sutton, A Future-Adaptable Password Scheme. *USENIX Annual Technical Conference, Proceedings, 1999*. USENIX. pp. 81-92.
- [28] M. J. Freeman, K. Nissim, B. Pinkas, Efficient Private Matching and Set

Intersection. *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, May 2004*. Springer Berlin Heidelberg. pp. 1-19.