Technische Universität München
Department of Informatics

Bachelor's Thesis in Informatics

Security Analysis of Mobile Messaging Traffic with an Automated Test Framework

Johannes Zirngibl

TECHNISCHE UNIVERSITÄT MÜNCHEN

DEPARTMENT OF INFORMATICS

BACHELOR'S THESIS IN INFORMATICS

Security Analysis of Mobile Messaging Traffic with an Automated Test Framework

Sicherheitsanalyse des Netzwerkverkehrs mobiler Nachrichtenanwendungen mittels einer automatisierten Testumgebung

Author	Johannes Zirngibl
Supervisor	Prof. Dr.-Ing. Georg Carle
Advisor	Dipl.-Ing. Univ. Quirin Scheitle, Dr. Matthias Wachs
Date	November 15, 2015

Informatik VIII
Chair for Network Architectures and Services

I confirm that this thesis is my own work and I have documented all sources and material used.

Garching b. München, November 15, 2015

Signature

Abstract

The non-local nature of communication paths used by modern mobile messaging applications can have a significant negative impact on the security and privacy of communications between users. Such applications are a popular form of communication, but their often centralized infrastructure limits the use of direct communication paths between users, thereby increasing the number of opportunities for outsiders to intercept messages as they pass by. This thesis analyzes and compares the communication paths of four well known mobile messaging applications in large scale by exchanging over 6000 mobile messages. The senders and receivers are distributed across 28 countries worldwide to present a global perspective. The paths taken by messages are reproduced using forward path measurements and mapped to geolocations. These communication paths are additionally compared to direct network paths between the countries to derive the influence of the mobile messaging applications. The mobile messaging applications used in this work are chosen based on a taxonomy depicting their popularity, security, service architecture and server locations. This thesis introduces a transparent, extensible and automated test framework to conduct automated, large scale measurements with mobile applications. The framework allows message exchanges between multiple emulated countries and reproduces the communication paths.

The analysis of the message paths shows that architecture and infrastructure of applications have significant impact on users' data locality. The selected applications all use a centralized architecture, located in a single country. The thesis shows that this architecture results in deviations from direct paths for more than two thirds of the communication paths. This has the implication that messages need to transition multiple additional countries, even if the parties are in the same location, which drastically reduces the users' data security and communication privacy.

Zusammenfassung

Die fehlende Lokalität der Kommunikationspfade moderner mobiler Nachrichtenanwendungen kann die Sicherheit von Nutzern und die Privatsphäre von Kommunikationen erheblich negativ beeinflussen. Mobile Nachrichtenanwendungen sind eine populäre Form der Kommunikation. Ihre zentralisierte Infrastruktur beschränkt jedoch direkte Kommunikationspfade zwischen Nutzern. Hierdurch erhöhen sich die Möglichkeiten für Außenstehende, die Nachrichten während ihrer Übertragung abzuhören.

Diese Arbeit analysiert und vergleicht die Kommunikationspfade von vier bekannten mobilen Nachrichtenanwendungen anhand von über 6000 ausgetauschten Nachrichten. Um eine globale Sicht auf die Kommunikationspfade zu erhalten, sind Sender und Empfänger der Nachrichten auf 28 Länder weltweit verteilt. Die Pfade der Nachrichten werden durch Pfadmessungen nachgestellt und auf ihre geographischen Standorte abgebildet. Zusätzlich dazu wird dieser Nachrichtenpfad mit direkten Netzwerkpfaden zwischen den Ländern verglichen, um den Einfluss der mobilen Nachrichtenanwendungen zu ermitteln. Die Auswahl der in dieser Arbeit verwendeten Anwendungen erfolgt auf der Basis einer Taxonomie. Die Kriterien der Taxonomie sind Popularität, Sicherheit, Architektur der Dienstleistung und Standorte der Server. Des Weiteren stellt diese Arbeit eine transparente, erweiterbare und automatisierte Testumgebung vor, die es ermöglicht, automatisierte und umfangreiche Messungen von mobilen Anwendungen durchzuführen.

Die Analyse der Kommunikationspfade zeigt, dass die Architektur und Infrastruktur der Anwendungen einen signifikanten Einfluss auf die Lokalität der Daten eines Nutzers haben. Alle ausgewählten Anwendungen nutzen eine in einem Land zentralisierte Infrastruktur. Darüber hinaus wird gezeigt, dass dadurch über zwei Drittel der Kommunikationspfade von einem direkten Pfad abweichen. Dies hat zur Folge, dass Nachrichten zusätzliche Länder passieren müssen, wodurch die Sicherheit der Daten eines Nutzers und die Privatsphäre einer Kommunikation stark verringert werden.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The non-local nature of communication paths used by modern mobile messaging applications can have a significant negative impact on the security and privacy of communications between users.
Mobile messaging applications, like WhatsApp or WeChat have become immensely popular channels of communicating via the Internet. WhatsApp is known as the most popular in the Western hemisphere and has been able to acquire 900 million monthly active users since its release in 2009 [1]. Furthermore one of the founders announced in January 2015 that WhatsApp is delivering over 30 billion messages per day [2].
Mobile messaging applications are focusing mainly on mobile devices, often without clients for other devices. They provide the users with the possibility to send and receive instant messages of multiple types, for example texts or pictures.
Comparing them to emails, another established way of communicating over the Internet, shows large differences. On the one hand, the email service is based on well known and open protocols, used by multiple providers. Therefore a user is not dependent on only one centralized provider and interoperability is given. On the other hand, mobile messaging applications are often based on their own closed protocols and closed source codes without any interoperability to other providers. Furthermore, the infrastructure of the service is often centralized and not necessarily located in the same jurisdiction as the user. This leads to a high dependency between the user and the provider of an application whereby the user has to fully trust the provider and its responsible jurisdiction. Different reports and exposures in the last years have shown that this trust is at least questionable. Sometimes Internet companies work together with intelligence agencies and provide them access to user data, either voluntarily or out of necessity and they are usually not allowed to inform their users about this [3]. Furthermore different jurisdictions are known to use their intelligence agencies to intercept traffic at large Internet exchange points (IXPs), for example the GCHQ in the United Kingdom or the BND in Germany, who has most likely intercepted traffic at the DE-CIX for several years [4, 5].
Deviations from a direct communication path between a sender and a receiver increase

the number of such possible points of attack and therefore impact security and privacy of communications between users.

1.1 Goals of the Thesis

This thesis analyzes the impact of mobile messaging applications on the communication paths of the network traffic delivering the messages between a sender and a receiver. To cover this issue, the following research questions will be answered:

1. Does mobile messaging influence the locality of network traffic?
2. Do the results differ based on the regions of the communication partners?
3. How strong is the influence on communication paths?
4. Does the influence differ between the various mobile messaging applications?

1.2 Definition of Local Traffic

Traffic is defined as *local* in this thesis when it remains in the region of the communication partners. If they are located in different regions local traffic remains in the set of both regions. The regions are the five regions defined by the United Nations Geoscheme: Africa, Asia, Europe, Oceania and the Americas. The Americas are further split into Latin America and the Caribbean – from now on labeled as South America – and Northern America, labeld as North America [6].
For example, when the sender resides in Germany and the receiver in Spain, both located inside of Europe, *local traffic* would only be routed inside of Europe and would not pass other regions.
This definition is based on the assumption of a user without any knowledge about network topologies and routing policies. Such a user is expected to assume that a sent message remains in the regions of the sender and receiver. Even a path between Europe and Oceania could be expected as local. Even though they lie far apart, a direct path can be assumable with submarine communications cables or satellite Internet connections.

1.3 Methodology

To answer the research questions, real applications are employed to measure mobile messaging application traffic in a global view. The selection of the applications is based on a carefully created taxonomy that considers different aspects and is applied to a list of mobile messaging applications.

To intercept the traffic and emulate plenty of geographical locations, the mobile phones running the selected applications are integrated into a framework. The framework is able to intercept the traffic and tunnel it through the Internet to multiple countries. Mobile messages are exchanged between the mobile phones in multiple measurements to analyze their behavior. The intercepted traffic is analyzed to extract message passing network connections. The communication paths are reproduced with forward path measurements to the targets of these network connections to obtain all hops in between the sender an receiver. Those paths will be called *application paths* below. The locality of mobile messaging traffic will be analyzed, based on these path measurements.
With the definition of local traffic, resulting deviations could also emerge from network topologies or routing policies. Simply routing network traffic through the Internet could already contradict with the assumption of a user and has to be considered. To make sure the actual influence, resulting from the applications, is evaluated, the application path is compared to another path, the *network path*. It is defined as the route between the sender and receiver on the Internet and measured through additional path measurements directly between them.

In the course of this thesis, a paper was written and submitted to the Passive and Active Measurement Conference 2016. Parts of this thesis result from the joint work with the research group for the paper. The affected parts of this work are labeled accordingly. [7]

1.4 Outline

The thesis is structured as follows: Chapter 2 provides necessary background about distributed testbeds for understanding the thesis. Chapter 3 presents related work to this thesis and explains the remaining differences. Chapter 4 gives an overview of different mobile messaging applications and introduces and applies the taxonomy that is created to select interesting applications for this thesis. The framework that is built to conduct all necessary measurements is explained in Chapter 5. It includes entities to generate, capture, tunnel and analyze the traffic of the respective applications. Chapter 6 describes the exact processes of the different measurements, realized through this framework. It explains the particular steps that were made to obtain the application and network paths. The resulting dataset is post-processed through different methods described in Chapter 7. Resulting from this the locality of mobile messages is analyzed and evaluated in Chapter 8 to answer the research questions. Interesting further experiences and challenges that have arisen are mentioned in Chapter 9. Finally, the thesis is concluded in Chapter 10 and an outlook about further research topics is provided.

Chapter 2

Background on Distributed Testbeds

Testing and analyzing software or products developed primarily for network use is often limited by small testbeds or the availability of a few distributed nodes. Therefore results are often not comparable to the real life behavior over the Internet. To overcome this problem there are different possibilities to gain access to globally distributed testbeds.

2.1 PlanetLab

The PlanetLab testbed is a project to globally run and test network services. It started in 2003 at Princeton University and today, users have access to 1353 nodes which are located at 717 different sites worldwide. Most of the nodes are bunched in North America, Europe and Asia but there are some in South America, Oceania and Africa as well. Every site has to host at minimum one node to gain access to the testbed. Having done so they are able to create multiple accounts for their members. The users possess so-called *slices*, a logical structure to organize their workflow. Each node is divided into multiple GNU/Linux *vservers*, so called *slivers*, that share the node's hardware but provide the users with encapsulated areas they can work on. Users can add multiple *slivers* from different nodes to their *slices* and access them with a UNIX shell access. However a disadvantage of the GNU/Linux *vserver* approach is, that the users are only provided with limited authorizations to modify the settings and especially the network configuration. The limitation of the root access is necessary due to the shared hardware to secure the access and configurations for other users deploying the same node. [8]

2.2 NLNOG Ring

NLNOG Ring is a similar testbed to PlanetLab, initiated by the Netherlands Network Operators Group (NLNOG). Today the testbed is globally distributed. There are 388 different nodes in 48 countries available. An organization gains access to the testbed by hosting at least one node itself. Afterwards they can access all other nodes of the testbed. That is why it is called a *ring*. A node runs different virtual machines users can connect to. Users have no *root* or *sudo* rights to preserve the security and consistency of the system. Therefore the configuration of the virtual machine can not be changed and no additional software can be installed by a user. [9]

2.3 VPN Gate

VPN Gate is a project of the University of Tsukuba in Japan that provides an infrastructure for volunteers to host relay servers for virtual private networks (VPN). The relay servers are only gateways, providing users with the possibility to tunnel the complete network traffic trough them but without direct access to execute measurements on them. The infrastructure can be used free of charge and volunteers are hosting relay servers around the world. A disadvantage is, that because of the volunteer based system the uptime of the relay servers is rather short often only some hours per day and the availability over a longer period of time differs a lot. [10]

2.4 RIPE Atlas

RIPE Atlas is a measurement testbed established by the RIPE NCC. The testbed connects 8951 *probes* to run different measurements. A probe is a hardware device, running different measurements: *pings*, *path measurements*, *DNS queries* and *SSL Certificate measurements*. The results are collected by components in the testbed to create maps and overviews of traffic behavior. Users can participate by running their own probe. Afterwards they can access all collected results and can execute individual measurements with the testbed. The probes are distributed over all regions with the highest concentration in Europe and parts of Asia. [11]

Chapter 3

Related Work

To the best of my knowledge there has been no other research that analyzes the traffic of mobile messaging applications focusing on the locality of messages and the security or privacy issues resulting from it. Research about the traffic and security of mobile messaging applications tends to take another direction. Furthermore there is no documentation of a framework, similar to the implementation for this thesis.

3.1 Analysis of Mobile Messaging Traffic

Understanding the impact of mobile messaging traffic on the network is often the main purpose of existing research as in [12–14].
Pierdomenico Fiadino et al. analyze the traffic of Facebook and WhatsApp to understand major *Online Social Networks* and their network behavior. Therefore they distinguish the server footprints and data flow characteristics of both. They locate the backend servers of the two systems and inspect how different message types are handled by them. They state that mobile text messages are often sent with longer data flows but smaller data itself and multi media messages often have shorter data flows but are of larger size. Furthermore, in [12], they analyze how the server footprint and data flow of WhatsApp influences the quality of experience for the users. Their analysis is based on datasets of two large cellular networks. Furthermore they resolve the domain names of the services from a distributed testbed. They map the resolved IPs to geolocations to locate the service infrastructures. [12, 13]

Qun Huang et al. want to provide an insight into the impact of WeChat traffic on cellular networks and improve the user experience. Therefore they analyze the different tasks accomplished by the application and the traffic characteristics of WeChat by dissecting large datasets provided to them by a cellular network. Their main findings are, that

a high percentage of the general Internet users use WeChat, that different network protocols are used for different tasks and that the downlink traffic is often higher than the uplink traffic. [14]
Both research teams do not create datasets but rely on measurements provided by cellular networks. Furthermore they analyze the characteristics of the traffic and the footprint of the service infrastructure but do not consider the locality of the traffic or security risks.

3.2 Security Analysis of Mobile Messaging Applications

Several research works analyze the security of mobile messaging applications and reveal privacy risks and user protection issues as in [15–18].
The Electronic Frontier Foundation (EFF) gives an overview of the security of multiple mobile messaging applications. They compare the security of many mobile messaging applications based on different criteria and present their findings in a clear scorecard. Only a few of the applications reach the highest possible score. Usually the messages are not end-to-end encrypted or the possibility of an independent code review and analysis is missing. The impact of non-locality on the security of the messages is however not a part of their analysis. [15]

To call attention to vulnerabilities of mobile messaging applications, Schrittwieser et al. published two papers about different vulnerabilities based on a weak authentication mechanism. The first paper, published in 2012 analyzes nine different applications [16]. The second paper re-evaluates those and adds eight further applications [17]. The connection between all these mobile messaging applications is that the authentication takes place based on the phone number of the user. They mainly analyze whether account hijacking is possible because of this mechanism, whether an attacker is able to trigger unrequested text messages and whether enumeration attacks on the user base are possible. The enumeration takes place between the uploaded phone book of the users and the connection of the different phone numbers and user accounts corresponding to the applications. The research team comes to the conclusion that many applications are afflicted by these security issues and most of them were not solved until the re-evaluation. Similarly to this thesis, they integrate mobile phones into a framework to collect data, but they have to configure proxy settings directly on the phones and the actions on the phones have to be performed manually.
Coull and Dyer analyze the security impact of information side channels of encrypted mobile messaging traffic. They exemplarily analyze whether encrypted traffic of Apple's iMessage service is still vulnerable to passive analysis methods even with end-to-end encryption. They come to the conclusion that an eavesdropper can collect information about the messages even when traffic is encrypted. With different methods that run with a high accuracy they can get insights into the message type, the text length or

the language used. The methods are mostly based on the payload size of the encrypted and intercepted packets. The research group proposes that encryption alone is not sufficient to protect all meta data but that other methods, especially some that vary the length of the payload have to be added. They claim that all their presented attacks are basically applicable on other messaging applications as well. They implement their own framework that automates the message sending and traffic interception, but do not integrate a mechanism to send from distributed locations. Furthermore traffic locality does not matter in their security and privacy analysis. [18]

3.3 Location Analysis of Application Traffic

Ferreira et al. provide knowledge about privacy and security issues of Android applications combined with user opinions. They mainly analyze which permissions an application requires and the opinions of the users about all those permissions and whether they consider them to be alarming. Furthermore they track network connections that are established by applications and especially the destination servers of those connections. They analyze the countries those servers are located in and conclude whether this connection is basically secure only based on the network port used and the protocol normally using this port. They come to the conclusion that most of the connections are established to North America and are insecure by design of the protocol used. They implement an Android application called *Securacy* to analyze other applications and hand it out to a set of users to collect data for their research. They do not create a framework similar to the one in this thesis but rely on users actively using *Securacy*. In addition, they do not focus on mobile messaging applications but only on the endpoints of a connection and neglect the locality of the network paths. [19]

3.4 Summary

This chapter presented other research analyzing mobile messaging applications. Some approaches analyzed the traffic behavior and communication patterns of mobile messaging applications, but the paths of the messages are not considered. Furthermore other works highlighted different security aspects, but there is no research so far that analyzes the security risks based on the non-locality of message passing traffic.

Chapter 4

Application Selection Based on a Taxonomy

In this chapter a list of different mobile messaging applications is presented and a taxonomy is applied to the list to gain a subset of applications worth studying. Those will be examined further. In this thesis only applications available for Android are considered.

Many Android applications provide users with messaging services. Table 4.1 holds a list of applications for sending text and media messages between users. Nearly all of them support group chats and a high percentage of them can be used to make phone calls. It has to be mentioned that this list does not claim to contain all available applications with those features.

In the context of this work only a subset of applications can be considered. Therefore a taxonomy is created to compare the applications based on different aspects and to select a subset of suitable applications. The four aspects are: popularity, security, architecture of the service and distribution of the servers provided for the application. The values for each application are shown in Table 4.2. There are some values missing whenever no information was found during the research.

4.1 Popularity of Mobile Messaging Applications

The first aspect is the popularity of the applications. Many companies provide the *number of monthly active users* (MAU) of their mobile messaging application. The MAUs are published by the companies providing the applications through different channels, like blogs or financial reports. They do not explain how they have raised them or what active exactly means. However the MAU gives an impression of the popularity of an application. Therefore the MAU is considered to be a comparable indicator for the popularity in the taxonomy.

Application	Version[1]	Text/Media Messages	Group Chats	Telephony
WhatsApp	2.12.176	✓	✓	✓
WeChat	6.2.4	✓	✓	✓
TextSecure	2.24.1	✓	✓	✗
Threema	2.41	✓	✓	✗
Bleep	1.0.616	✓	✗	✓
Skype	5.8.0.13549	✓	✓	✓
Facebook Messenger	34.0.0.22.211	✓	✓	✓
Telegram	3.1.2	✓	✓	✗
SIMSme	1.03.001.127086	✓	✓	✗
QQ International	5.0.10	✓	✓	✓
Silent Text	1.12.2	✓	✓	✗
Viber	5.5.0.2477	✓	✓	✓
Wickr	2.4.7	✓	✓	✗
Kik	8.7.0.1643	✓	✓	✗
Yahoo Messenger	1.8.8	✓	✓	✓
Tango	3.18.168535	✓	✓	✓
LINE	5.2.5	✓	✓	✓

1: Version of the Android application available on 06.09.2015

Table 4.1: Functionality of Different Mobile Messaging Applications

Table 4.2 shows that the MAU varies greatly between the different applications. The values range from millions, for example from Threema, to hundreds of millions of monthly active users. The applications with the highest given numbers are WhatsApp, with 500 to 900 million, WeChat with up to 500 million, the stand alone Facebook Messenger, with up to 600 million and QQ International, with up to 830 million monthly active users.

Based on these results the decision which services to look at was made in favor of WhatsApp and WeChat even though Facebook Messenger and QQ International have a comparable or even bigger number of monthly active users. Facebook and QQ International are mainly social networks built as web based desktop oriented services and not as mobile messaging applications. They obtain a large amount of users through these services. In a second step, they added the mobile messenger to their products. In contrast, WhatsApp and WeChat have a massive amount of monthly active users as well and were primary built for mobile phones as mobile messaging services and obtain the majority of their users from the mobile phone segment.

WhatsApp belongs to the U.S. company Facebook Inc. but is provided independently from the social network Facebook and the affiliated messenger. After a registration with the phone number, it is possible to send messages to other registered users.

WeChat known as *Weixin* in Asia, was developed by the Chinese company Tencent. Similar to WhatsApp, Tencent also provides QQ as a social media platform and both are independent as well. Besides the possibility to create an account with the phone number they allow their users to log in with a Facebook account.

Both are available for all common mobile operating systems and they have recently implemented web interfaces to enable users to send and receive messages on their browsers as well.

4.2 Security Analysis

The second aspect of this taxonomy is aimed at the security of the applications. Analyzing the security of the applications of Table 4.1 is out of scope of this thesis. As mentioned in Chapter 3, the Electronic Frontier Foundation regularly analyzes different messaging applications and publishes the results on a scorecard. They have created a taxonomy with seven criteria:

1. Is the message encrypted on transit?
2. Is the message end-to-end encrypted?
3. Is there a possibility to verify the identity and key of a communication partner?
4. Does the application use perfect forward secrecy?
5. Can the code be reviewed by another institution?
6. Is there a documentation of the security implementation?
7. Was an independent security audit done in the recent past?

Thus, their evaluation leads to a maximum score of seven points, one for each criterion. In this thesis, this score is considered to choose an application that is categorized as secure.
Some of the applications from Table 4.1 have not been evaluated by the Electronic Frontier Foundation so far. The bigger part of the evaluated applications only reaches one to two points for encryption on transit and a recent code audit in the past. With the default settings, Telegram reaches only four points. The missing three points for end-to-end encryption, perfect forward secrecy and the possibility to verify the identity of a communication partner are only reached if the user explicitly activates so called *secure chats*. Threema is missing points for its closed source code and no recent code audit and Wickr is closed source as well and has no proper documentation of its security protocol. Finally, two applications of the list in Table 4.1 score the maximum of seven points: TextSecure and Silent Text. [15] Here TextSecure is chosen because it is free of charge. Silent Text however charges its users a monthly fee.

TextSecure was built by Open Whisper Systems and is only available for Android devices. Open Whisper Systems developed another application called Signal for iOS. At the moment TextSecure and Signal are independently treated by Open Whisper Systems even though communicating between them is possible. CyanogenMod[1], an open source firmware based on Android, has directly integrated TextSecure into the firmware to send mobile messages. As a result, ten million users were added to the user base of TextSecure, but CyanogenMod uses its own service infrastructure. Communication between both infrastructures is possible [20]. In this thesis only the version available in the Google Play store will be considered. Furthermore, the possibility to make calls is implemented as a standalone application named RedPhone. Accounts for TextSecure are created using the phone number of the mobile phones.

4.3 Architecture of the Service

Nearly all applications use a client-server approach where a sender sends the message to the infrastructure of the application. The receiver of the message has to connect to the infrastructure and pull the message from it afterwards.

Bleep developed by BitTorrent, Inc., uses peer-to-peer (P2P) as a contrasting approach to deliver the messages between two clients. The main idea is that the message itself does not have to pass the infrastructure of the application but is sent directly to the receiver. This is accomplished through a distributed hashing table (DHT), where a mapping between each user and its current IP address is stored whenever a user goes online with its application client. Before sending a message, the sender has to request the mapping between the receiver and its IP address and can afterwards send the message to the received IP address and therefore directly to its communication partner. This only holds true if both communication partners are online at the same time and if a direct path is possible and not denied due to e.g. NATs or firewalls. Otherwise, Bleep switches to a client-server approach by sending the message to the DHT. When the receiver goes online and connects to the DHT to store its current IP address the message is delivered to the receiver from the DHT. [33, 34]

Bleep does not need a mobile phone number or email address to create an account but individual identifiers are assigned to the users. The phone number and email address can still be added so other users can find someone more easily presuming they know these data.

4.4 Geographical Server Distribution

The geographical distribution of the backend infrastructure is the criterion of the taxonomy where the least information is available. Some companies make explicit statements about the location backend infrastructure. On the one hand, some companies state that

[1]http://www.cyanogenmod.org/

Application	Version[1]	Monthly Active Users[2]	EFF Scorecard Points
WhatsApp	2.12.176	500 - 900 mio [1, 21, 22]	2
WeChat	6.2.4	500 mio [23]	n/a
TextSecure	2.24.1	>10 mio [20][3]	7
Threema	2.41	3 mio [24]	5
Bleep	1.0.616	n/a	n/a
Skype	5.8.0.13549	300 mio [25]	1
Facebook Messenger	34.0.0.22.211	600 mio [26]	2
Telegram	3.1.2	50 mio [27]	4 (7 in secure chats)
SIMSme	1.03.001.127086	1 mio [4] [28]	n/a
QQ International	5.0.10	830 mio [23]	2
Silent Text	1.12.2	n/a	7
Viber	5.5.0.2477	249 mio [29]	1
Wickr	2.4.7	n/a	5
Kik	8.7.0.1643	n/a	1
Yahoo	Messenger 1.8.8	n/a	1
Tango	3.18.168535	70 mio [30]	n/a
LINE	5.2.5	211 [31]	n/a

Application	Version[1]	Architecture	Server Distribution
WhatsApp	2.12.176	client-server	n/a
WeChat	6.2.4	client-server	n/a
TextSecure	2.24.1	client-server	distributed[5] [32]
Threema	2.41	client-server	Switzerland
Bleep	1.0.616	peer-to-peer	n/a
Skype	5.8.0.13549	client-server	n/a
Facebook Messenger	34.0.0.22.211	client-server	n/a
Telegram	3.1.2	client-server	distributed[5]
SIMSme	1.03.001.127086	client-server	Germany
QQ International	5.0.10	client-server	n/a
Silent Text	1.12.2	client-server	n/a
Viber	5.5.0.2477	client-server	n/a
Wickr	2.4.7	client-server	n/a
Kik	8.7.0.1643	client-server	n/a
Yahoo Messenger	1.8.8	client-server	n/a
Tango	3.18.168535	client-server	n/a
LINE	5.2.5	client-server	n/a

1: Version of the Android application available on 06.09.2015
2: Values determined till 06.09.2015
3: Only users from the implementation in CyanogenMod
4: Registered Users
5: Distributed servers in multiple countries worldwide

Table 4.2: Criteria for the Selection of the Applications

they host globally distributed servers as shown in Table 4.2. With the decision to evaluate TextSecure because of the high EFF score, one of those applications is already in the selected subset. On the other hand, there are Threema and SIMSme, two applications that promise to only host backend servers in specific countries, Threema in Switzerland and SIMSme in Germany. This automatically leads to non-local traffic for all regions except Europe because the messages have to pass these servers. The question remains about the degree of this influence compared to the other applications. Even though Threema and SIMSme have comparably low monthly active user numbers, Threema is chosen because of the higher MAU in direct comparison.

Threema is developed by the Swiss company Threema GmbH. Besides the centralized server structure it also has a relatively high EFF score with 5 points. It is available for Android, iOS and Windows Phone 8. Accounts are not created with the mobile phone number, but individual identifiers are assigned to users by the application to preserve their privacy. Furthermore, Threema provides a lot of information about the application and the security protocols used on their website[2].

4.5 Summary

In this chapter, a taxonomy was applied to a list of mobile messaging applications for Android. Using this taxonomy resulted in a subset of five applications—WhatsApp, WeChat, Threema, TextSecure and Bleep—which will be used for the upcoming locality analysis. WhatsApp and WeChat were chosen because of their popularity, TextSecure was chosen because of its security, Bleep promises P2P as a different service architecture and Threema completes the subset with its promise to only host servers in Switzerland.

[2]https://threema.ch/en

Chapter 5

Structure and Implementation of a Measurement Framework

To collect the application paths, an automated framework is built to generate, capture and analyze the traffic of the applications. This chapter is divided into several parts: First of all, the necessity of the framework and the requirements it has to meet will be explained. Then a logical overview of the framework is given. In a final step the implementation of the individual components is explained in detail.

5.1 Necessity and Requirements for the Framework

The initial position so far, without a framework, is a set of applications. Most of them are closed source and can therefore only be analyzed from the outside. To create a standard environment for the applications and to not influence the behavior and therefore the results, they have to be installed natively on mobile phones and not on virtual machines or emulators. The mobile phones are bound to one geographical location and can not easily be moved to many different countries, but the goal of this thesis is to send and receive messages from multiple locations. All those preconditions lead to the necessity of a framework to enable automated and reproducible measurements with the same conditions for the set of applications and the possibility to enable measurements in a global view.

The central goal behind the framework is to send real messages to and from different countries. Therefore the network traffic has to be tunneled through the Internet to different countries. Besides tunneling the traffic, the mobile phones need to be convinced that they are located in different geographical locations to make sure the mobile phones do not change their behavior. Most applications require the permission to request the *global positioning system* (GPS) coordinates from the operating system. The coordinates have to be consistent with the location, the network traffic is tunneled trough.

Another goal is to intercept the entire network traffic for further analysis. Intercepting the traffic has to be done while the mobile phones run the applications and send messages and is therefore swapped to extra measurement points. The mobile phones have to be connected to the Internet through the measurement points to provide the measurement points access to the entire network traffic. Additionally, the connection to the measurement points should be the only network connection the mobile phones can use. One more requirement to facilitate a capture, as cleanly as possible, is to suppress traffic from other applications running on the mobile phones by using firewalls.

The last requirement regarding the framework is the automation and control of the measurement process. This implies multiple subaspects: First of all the network components and tunnels should be set up automatically to different remote nodes and the coordinates should be falsified accordingly. Second, the capture has to start simultaneously. Third, the mobile phones and applications need to be controlled to connect to the network, execute the applications and send messages.

The identified requirements of the framework can be summarized as follows:

1. Running the applications natively on mobile phones
2. Establishing points of measurement to capture the network traffic
3. Tunneling the network traffic transparently for the mobile phones
4. Emulating different geographical locations of sender and receiver
5. Automating the process

Besides the primary requirements, secondary requirements arise. All components should be implemented extensible and independently to create a framework for further use.
Furthermore first analysis steps should be done in the framework. The analysis is no direct requirement of the framework and could be done afterwards, but some analysis steps rely on the tunnel infrastructure to reproduce the application paths. Integrating them into the framework saves measurement time. Therefore the following steps of analysis are considered during the framework implementation: extraction of all resolved hostnames, investigation of the established network connections and execution of first path measurements to the targets of the network connections to reproduce the paths of the traffic.

5.2 Overview of the Framework

The combination of the requirements leads to the overview, illustrated in Figure 5.1. In the context of this thesis the number of communication partners is limited to two. The mobile phones, running the mobile messaging applications and sending messages, are

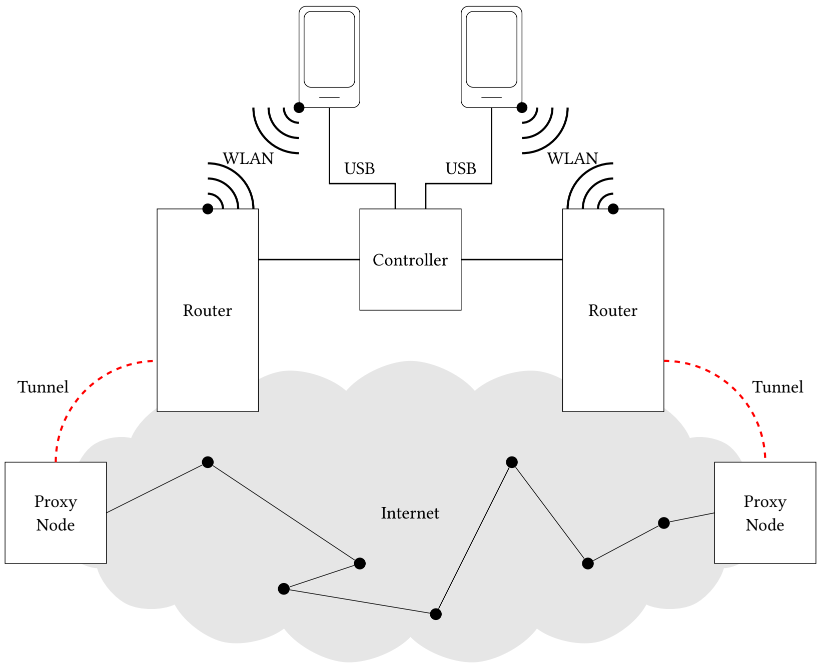


Figure 5.1: Overview of the Framework

the first components shown at the top. The measurement points that capture the traffic are two routers. They span wireless networks, the mobile phones use to connect to the Internet to make sure those measurement points can capture the entire traffic. Even though the routers are directly connected to the Internet, the traffic from the mobile phones is forwarded through tunnels to different remote nodes to emulate different geographical network access points. The traffic is finally routed between the remote nodes through the Internet independently from the framework. This path is the main subject of the further analysis. To automate the processes, a controller is added to the framework. The controller needs to be connected to both mobile phones and the routers, to fulfill all its tasks.

5.3 Configuration of the Mobile Phones

The applications are installed on physical Android mobile phones. The devices are two Motorola Moto E (2. Generation) with Android Lollipop 5.0.2 (Build number: LXI22.50-24.1) installed. Both devices are rooted to gain complete access to the features of Android. They are connected to Google accounts with individually generated email addresses and possess SIM cards with German mobile phone numbers. The Google accounts are used to download the applications from the Google Play Store directly. The Bleep accounts are created with individual user names and connected to the email addresses of the Google accounts. The WhatsApp, WeChat and TextSecure accounts are created with the phone numbers. Automatic application updates are disabled on the phones to make sure the same application versions are used for all measurements. No other services or applications are installed, to limit the network traffic.

By disabling the cellular network connection it is secured that the complete traffic is sent over the wireless network the mobile phones will be connected to and can be intercepted by the framework. Furthermore, the mobile phones have no access to cell information that could be used to calculate the geographical location.

5.3.1 Controlling Network Traffic with Firewalls

Nevertheless traffic generated by sources other than the messaging applications can not be excluded completely and has to be taken into account in the analysis of the captured data. Still a first step to suppress remaining, unwanted traffic from the mobile phones is the use of firewalls in the framework. Android users are provided with two possibilities to set up firewalls. First, several firewall applications are available in the Google Play Store. Second, Android supports iptables natively. The framework uses the second possibility and manually sets up firewalls with iptables to maintain full control over the firewalls. A whitelist approach is used where all traffic is generally denied and only necessary system functions and the executed applications are whitelisted and therefore able to establish network connections. Whitelisting applications under Android works over the user identifier (UID) of the applications. A UID is assigned to every application and the network traffic generated by that application is tagged with the UID [35]. The iptables rules apply an owner match on outgoing traffic based on these UIDs and reject everything not matching. The configuration of iptables requires root access.

The list of necessary system functions is elevated empirically by starting with a whitelist, on which only the messaging applications are listed. Afterwards different UIDs are added until the messaging applications act properly. In the framework, the following list of functions is necessary:

1. *root* (UID: 0)
2. *system* (UID: 1000)
3. *wifi* (UID: 1010)
4. *dhcp* (UID: 1014)
5. *Motorola Services:*
   3c_devicemanagement, 3c_checkin, cce, 3c_notification (UID: 10000)
6. *CQATest* (UID: 10005)
7. *Google Partner Setup* (UID: 10019)

To create clean and separate results for each mobile messaging application, only the active mobile messaging application sending messages, is whitelisted during a measurement. All other mobile messaging applications are not whitelisted and their network traffic is therefore suppressed.

5.3.2 Changing GPS Coordinates

The application XPrivacy[1] is used to change the GPS coordinates of the mobile phones and to adapt the locations to the countries, the traffic will be tunneled to. It requires root access on the mobile phones and is based on the XPosed framework[2]. XPrivacy can limit the access of applications to different categories of data. It can either completely restrict the access or return fake values. The framework uses XPrivacy to restrict the *location* category and to change the longitude and latitude. After setting up XPrivacy on the mobile phones, it returns the changed values to the mobile messaging applications if they request the geographical location from the operating system. To efficiently change the longitude and latitude for each location, the XPrivacy versions are upgraded to the pro version. This allows to export the settings as an XML file, change the values accordingly in this file and re-import the settings from the changed file.

5.4 Controlling and Automating the Mobile Phone Activities

The main challenge in controlling and automating the interaction with the mobile phones is the fact that most of the Android system functionalities and applications are only designed for an interaction through a graphical user interface and do not offer command-line interfaces. To deal with this challenge, the *Android Debug Bridge* (ADB) is used. It is officially developed and supported by Google and part of the Android software development kit[3]. It is a client-server based command line tool that provides the possibility to connect from a command line to Android mobile phones over USB or TCP connections. To avoid unnecessary network traffic, the possibility to connect over USB is used in the framework. It requires the activation of the USB debugging mode, in the developer options of the mobile phones. Each mobile phone connected is identified by a unique serial number. This can be used to connect multiple devices to one controller and forward commands explicitly to individual devices with the `-s` parameter. [36]

In principle, the ADB provides different commands to interact with the connected mobile phone. It can be used for example to copy data from or to the mobile phone and

[1]https://github.com/M66B/XPrivacy/
[2]http://repo.xposed.info/
[3]https://developer.android.com/sdk/index.html

access logs or the Android internal shell. The framework combines multiple of those functionalities to implement all necessary activities.

Before explaining the implementation of the different activities the possibilities of the Android shell in combination with the ADB have to be explained. Two different ways can be used to interact with the shell. Firstly, the Android shell can be accessed with the command `adb shell`. Afterwards commands can be used directly in the Android shell. Secondly, commands can be forwarded to the Android shell one by one. Therefore the instruction `adb shell` has to be added to each command. Due to this, sequences of multiple commands can be scripted and executed on a controller. Table 5.1 holds a list of different Android shell commands used by the framework.

The command `am` and `svc` can be used to interact with applications and services. Each application defines a name in its Android Manifest, that can be used to force-stop it, and some activities that can be started over the `am` command [37]. With `svc` different Android services can be for example enabled or disabled. In the framework the `wifi` service is controlled like this.

Most of the applications only define a main activity that can be used to start the applications. Especially the chosen messaging applications do not provide a feature to directly send messages with this approach. Therefore a workaround is implemented with the `input` command. It provides different options to simulate interactions with the touchscreen or other input devices for example a keyboard. The `tap` and `swipe` option can be used to interact with the display, that has to be turned on. Each Android mobile phone spans a coordinate system over its touchscreen to define where a user has interacted with the touchscreen. The grid used by the Motorola devices has approximately a width of 500 units and a height of 950, with the origin at the upper left corner. The option `text` allows to send a string to the device. Additionally, there are some predefined events mapped to numbers that can be used with the `input` command. Those so called *keyevents* stand for example for the sleep or power key [38]. Combining those commands, applications with a static user interface and a deterministic behavior can be controlled. All chosen messaging applications come with those preconditions and can be controlled with the same approach.

The `screenrecord` command records the screen for up to three minutes with different quality options. This feature is added to the framework to create videos of major tasks and to provide the possibility to verify the functionality by inspecting videos.

All different activities that are necessary to send messages between the mobile phones are explained in more detail in the following subsections.

Command	Parameters	Description
su	-c "Command"	executes the command as root
am	start "Acticivity"	starts activities of applications, e.g. com.whatsapp/.Main
	force-stop "Application"	forces an application to stop, e.g. com.whatsapp
svc	"Service" enable/disable	starts/stops a service, e.g. wifi
input	tap X Y	simulates a tap on the screen at the position (X,Y)
	swipe X Y X' Y'	simulates a swipe on the screen from (X,Y) to (X',Y')
	text "Text"	writes the text on the mobile phone
	keyevent "Event"	triggers pre-defined Android events, e.g. 26 stands for the power button
screenrecord	[options] "filename"	records the screen

Table 5.1: Commands for the ADB Shell

5.4.1 Display Activities

The framework contains four necessary activities concerning the display of the mobile phone: turning the display on, turning it off, starting the screenrecord and terminating the recording and copying the file to the controller. Turning the display on is done by sending the keyevent *26*, that stands for the power key of the mobile phone. Afterwards the screen needs to be unlocked. On the Motorola devices used, this can be done by swiping from the bottom of the screen to the top. Turning the display off can be done in one step by sending the keyevent *223*—that stands for sleep—to the mobile phone.

```
//Turning the Display On
adb -s TA36402CQL shell input keyevent 26
adb -s TA36402CQL shell input swipe 275 800 275 180
```

```
//Turning the Display Off
adb -s TA36402CQL shell input keyevent 223
```

Listing 5.1: Turning the Display On and Off

The recording of the screen is started by the according ADB command with a size of 360×640 and a bit rate of 1Mbps. It remains as a running background process on the controller. The process stops by default after three minutes or earlier if it is terminated with a Unix interrupt signal on the controller. Afterwards the video file is pulled from the mobile device to the controller and stored permanently. The relevant commands can be seen in Listing 5.2.

```
// Starting the recording
adb -s TA36402CQL shell screenrecord --size 360×640
    --bit-rate 1000000 /sdcard/video.mp4
```

```
// Pulling the result from the mobile phone to the controller
adb -s TA36402CQL pull /sdcard/video.mp4 example_recording.mp4
```

Listing 5.2: Recording the Mobile Phone Screen

5.4.2 Network Activities

The network activities are split into two groups, one for enabling and disabling the *wifi* service and one for appropriately setting up the firewall. Enabling the *wifi* service needs an additional step to test if the mobile phones are actually connected to the wireless network. It sometimes happens, that the mobile phones do not connect automatically, even with only one wireless network configured on each mobile phone. Therefore, after enabling the service the controller has to check if the phone is connected to the wireless network. Android provides no direct command-line tool to do so. Therefore a workaround is created with a ping based on ICMP echo requests. The ping is sent to the IP 8.8.8.8. If the mobile phones are not connected to the wireless networks, the ping results in the error message "connect: Network is unreachable". If the ping command fails, the controller retries the enabling process several times. Empirical tests showed, that three retries are enough. The value worked for all measurements afterwards. The *wifi* service is disabled by executing the `svc disable` command. The `scv` command has to be exectued as root.
The firewall needs to be set up initially with a sequence of iptables commands. First of all an extra chain is created and basically everything from the OUTPUT chain is forwarded to this chain. A rule is inserted into the new chain that rejects all UIDs and afterwards rules are inserted which only accept the described system functionalities by their UID. An example of the initial firewall can be seen in Listing 5.3. Another activity adds the rule to accept the traffic for each active messaging application separately and removes it after the measurements. Finally there is an activity that resets the firewall completely and removes all added rules and the chain created.

5.4.3 Application Activities

All messaging applications have similar graphical user interfaces and behaviors. Therefore the same steps executed with different parameters can be used to automate the message sending process.
Before sending messages the controller has to start the mobile messaging application and access the conversation with the other mobile phone. Starting a mobile messaging

```
Chain OUTPUT (policy ACCEPT)
num  target      prot opt src      dest
1    measurement all  --  anywhere anywhere

Chain measurement (1 references)
1    RETURN      all  --  anywhere anywhere    owner UID match
     root (0)
2    RETURN      all  --  anywhere anywhere    owner UID match
     system (1000)
3    RETURN      all  --  anywhere anywhere    owner UID match
     wifi (1010)
4    RETURN      all  --  anywhere anywhere    owner UID match
     dhcp (1014)
5    RETURN      all  --  anywhere anywhere    owner UID match
     u0_a0 (10000)
6    RETURN      all  --  anywhere anywhere    owner UID match
     u0_a5 (10005)
7    RETURN      all  --  anywhere anywhere    owner UID match
     u0_a19 (10019)
8    reject      all  --  anywhere anywhere    owner UID match
     0-999999999 reject-with icmp-port-unreachable
```

Listing 5.3: Firewall

application is done with the `am` command followed by the main activity of the application. After a startup time a screen with a list of all possible conversations appears on the mobile phones. The mobile phones are only set up for the measurement and contain only one conversation for each mobile messenger. Therefore the conversation is on a static position at the top of the conversations list and can be entered with the *input tap* command on this static position. The final step of starting the mobile messaging application is to tap on the text field. This is not necessary for all applications, some directly jump into the text field but for the sake of consistency, the step is done for all mobile messaging applications.
A message can now be sent by entering a text over the `input text` command into the already entered text field and tapping on the send button. The framework sends randomly chosen sentences from the play "Death of a Salesman". The input text has to consist of ASCII letters. Spaces can be written with `%s`. Double quotes and semicolons have to be escaped with a backslash.
Listing 5.4 shows the necessary sequence of ADB commands to open a mobile messaging application, enter a conversation and send a message with the example of WhatsApp.

At the end, the application can be closed executing an activity that triggers the `am force-stop` command on the mobile device, with the application name.
Fixed timeouts are set between all input events, to make sure they result in the correct

```
// Starting WhatsApp and entering the converstion
adb -s TA36402CQL shell am start com.whatsapp/.Main
adb -s TA36402CQL shell input tap 200 200
adb -s TA36402CQL shell input tap 230 847

//Writing a text and tapping on the send button
adb -s TA36402CQL shell input text "Hello%sworld"
adb -s TA36402CQL shell input tap 511 445
```

Listing 5.4: ADB Commands to Start WhatsApp and Send a Message

```
<XPrivacy>
  <Setting Id="" Type="" Name="Latitude" Value="48.13"/>
  <Setting Id="" Type="" Name="Longitude" Value="11.57"/>
</XPrivacy>
```

Listing 5.5: xprivcy_coordinates.xml Example

actions on the mobile devices. The mobile messaging application names, activities, timeouts and coordinates used in the framework are shown in Appendix A.1.

5.4.4 XPrivacy Activity

The initial XPrivacy set up is done manually to restrict the location data of the mobile messaging applications. The mobile phone controller has to change the values of the longitude and latitude. As already mentioned, the pro version of XPrivacy provides the possibility to export and import the settings of XPrivacy. It provides an activity to trigger the import process, that can be used with the `am` command. However the activity only opens the related GUI process and does not allow to import from a file directly as it is considered a security risk. The controller creates the file with the new values and pushes the file to the mobile phone. Listing 5.5 shows an exemplary input file changing only the latitude and longitude. Afterwards the import activity of XPrivacy is triggered. It opens a file manager, that jumps to the recently used directory. By importing a file once manually before the measurement and always pushing the input file afterwards to the same directory, overwriting the old file, it is certain that XPrivacy jumps to this directory and the file is visible at the same static position. Finally the `input tap` command of the Android shell is used to tap on the file and confirm the import. Listing 5.6 shows the sequence of ADB commands to import a file to XPrivacy. The `-eia UidList` parameter holds the UIDs of the mobile messaging applications.

```
//Pushing the import file to the mobile phone
adb −s TA36402CQL push xprivacy_coordinates.xml
      /sdcard/xprivacy_import/xprivacy_coordinates.xml
// Starting the import activity
adb −s TA36402CQL shell
     am start −a biz.bokhorst.xprivacy.action.IMPORT
     −−eia UidList 10101,10105,10106,10107
     −−ez Interactive true
//Tapping on the import file
adb −s TA36402CQL shell input tap 217 180
//Confirming the import
adb −s TA36402CQL shell input tap 394 292
```

Listing 5.6: ADB Commands to Import a file to XPrivacy

5.5 Measurement Points to Intercept the Network Traffic

The traffic has to be transparently and independently intercepted from the mobile phones. Therefore measurement points are set up that have access to the network traffic and can intercept it. To make sure the measurement points can actually intercept the complete network traffic, the traffic has to be routed between the mobile phones and the Internet via those measurement points. Therefore, they additionally serve as wireless access points (WAP). In the framework two low budget GNU/Linux computers running *Debian Jessie* are used. They are connected to the Internet directly over Ethernet interfaces. Furthermore they are equipped with wireless interfaces, used to span wireless networks the mobile phones can connect to. The authentication of the mobile phones is carried by the hostapd utility[4]. It is used to secure the wireless network with WPA2.

After the authentication to the WAP, the mobile phones receive their further network configuration via DHCP. Therefore DHCP servers[5] are deployed on the measurement points. The DHCP servers propagate addresses out of the private address space 10.0.0.0/8 [39]. The two WAPs assign addresses from different ranges to facilitate the differentiation of the two networks and therefore of the mobile phones in the measurements afterwards. Furthermore the DHCP leases propagate the DNS servers the mobile phones should use to resolve domain names.

After connecting the mobile phones to these WAPs, the complete network traffic passes them because cell data is disabled on the mobile phones. Therefore tcpdump[6] can be used on the wireless interface of the WAPs to intercept the entire network traffic originating from the mobile phones. Tcpdump is used without any filters and all analysis and filter steps are done afterwards on the resulting traces.

[4]https://w1.fi/hostapd/

[5]https://www.isc.org/downloads/dhcp/

[6]http://www.tcpdump.org/

5.6 Network Tunnel

Tunneling the network traffic from the framework to remote nodes is influenced by two major requirements itself. Firstly the endpoints have to be stable and globally distributed to gain a global view and reproducible initial conditions for measurements. Secondly the tunnels have to be completely transparent to the mobile phones to exclude the possibility of a different behavior by the applications as far as possible. The mobile phones are only connected to the wireless networks of the WAPs and do not know anything about the set up of the tunnels. However the complete network traffic should be relayed over them, including DNS queries. The traffic should appear afterwards as if it originated actually from the network of the remote node. Therefore even the DNS servers of the remote nodes should be used for DNS requests by the mobile phones.

5.6.1 Selection of a Distributed Testbed for Remote Node Access

To face the first requirement of the network tunnel, different easily accessible distributed testbeds are compared. The four testbeds PlanetLab, NLNOGRing, VPN Gate and RIPE Atlas have already been introduced in Chapter 2. All of them provide a large amount of globally distributed nodes. VPN Gate would provide the straightest way to tunnel the network traffic, as it is built explicitly for virtual private networks (VPN), but the relatively short uptimes of the gateways would be an enormous drawback for a long measurement that relies on stable conditions. RIPE Atlas would provide the largest amount of measurement entities with 8951 probes, but the probes are only able to run the four mentioned measurements and traffic tunneling is not possible. NLNOG Ring and PlanetLab provide nodes in similar countries and have comparably stable uptimes, but without *root* access on NLNOG Ring, PlanetLab wins in the direct comparison.

To integrate PlanetLab into the framework, two *slices* are used, one for each network tunnel from the routers. Due to this, the end points on the remote nodes are located in different virtual machines, possessed by each *slice* and do not interfere with each other especially when the tunnel endpoints are the same PlanetLab node.

5.6.2 Proxy Software to Tunnel Network Traffic

To actually tunnel the traffic to those nodes an implementation from another thesis is used. A. Loibl developed a way to tunnel the complete network traffic over a remote node. This project only needs SSH access on the remote node and *root* rights to create raw sockets and run user level binaries on it. Both requirements are fulfilled by the PlanetLab nodes even with the limitation of the *root* access on the nodes.

To serve the second requirement of the tunnel – the transparency – the starting points are designed as virtual network devices – tun devices – on the WAPs. To send something through the tunnel the traffic has to be forwarded only to this tun device. Actually tunneling the network traffic from these virtual devices to the remote nodes is solved by a combination of two solutions, one for UDP and ICMP and one for TCP.

UDP and ICMP can be handled on the user level of the remote nodes. Because of the connectionless implementation of the protocol, forwarding is possible. To establish a stable connection between the tunnel start- and endpoint with open ports the STUN protocol [40] combined with UDP hole punching is used to traverse possible NATs and firewalls between the framework WAPs and the PlanetLab nodes. After reaching the remote nodes, user level binaries handle the traffic with network sockets. All incoming UDP and ICMP packets from the WAPs are intercepted by sockets and retransmitted with the properties of the PlanetLab nodes as the new source address and port. The corresponding answers from the Internet are again intercepted by sockets on the PlanetLab nodes and sent back to the WAPs afterwards.

TCP is handled by a different solution. Because of the connection-oriented protocol, a solution similar to UDP and ICMP would cause TCP over TCP connections. To avoid this, a different approach is used. TCP traffic is encapsulated in the SOCKS protocol, implemented to traverse firewalls [41]. OpenSSH supports the version 5 of SOCKS, but only the TCP protocol. The encapsulated TCP traffic is sent via a SSH connection to the PlanetLab node. OpenSSH takes care of forwarding this traffic to its destination and handles the TCP connections. To encapsulate and forward the traffic from the tun device to the SOCKS tunnel on the routers, the program tun2socks[7] is used.

In summary, the implementation brings all necessary steps to set up the tunnel. First of all a SSH connection to the remote node is established. The binaries for the UDP and ICMP proxy are transmitted via SCP to the remote node and executed. Then a new SSH connection to the remote node is established to serve as a SOCKS tunnel. The tun device is set up and tun2socks is started to connect the SOCKS tunnel and the tun device. A feature that automatically copies the list of DNS servers from the PlanetLab node to the starting point is added to the tunnel setup explicitly for the framework. For further information about the measurement proxy software please refer to [42].

5.6.3 Connection to the Wireless Access Point

The starting point of the tunnels are the GNU/Linux computers, that are already serving as wireless access points. The tun devices created are located in their own network namespaces[8] as shown in Figure 5.2. Namespaces are copies of the network stack with individual routing tables, firewall rules and their own network devices. The main

[7] https://code.google.com/p/badvpn/wiki/tun2socks

[8] http://man7.org/linux/man-pages/man8/ip-netns.8.html

advantage of the decision to locate the tun devices in their own network namespaces is that normal network activities of the routers, for example SSH connections, are still handled in the main namespaces and routed over the standard Ethernet devices, directly connected to the Internet. Network activities created in the tunnel namespace are routed over the tun device, therefore path measurements or other tasks trough the tunnel are possible for the integrated data analysis.
A small workaround is created to forward the traffic from the wireless device to the tunnel, as it is not possible to move a physical device—the wlan0 devices—out of the main namespace. Therefore connecting the namespaces is handled by two additional virtual devices. One of them, veth0, is located in the main namespace, the other one, veth1, is located in the tunnel namespace. Those two are connected and forward the traffic automatically between each other. Finally policy routing is used to route the traffic between the interfaces wlan0 and veth0 independent from the main routing table. Everything originating from those devices is handled by a separate routing table. The routing table on the tunnel namespace uses the tun0 device as default route. Finally network address translation is applied at the tun0 device to masquerade the private address space of the WAPs and the traffic can be tunneled. Traffic originating from a mobile phone is therefore routed from the wlan0 device over the routing table *Tunnel* to veth0, forwarded to veth1 and finally routed in the namespace tunnel to the tun0 device. Afterwards the traffic is tunneled to a proxy node. The path is illustrated in Figure 5.2.
Besides the routing, the DNS resolution has to be consistent with the remote nodes as well. The list of DNS servers from the remote node is copied to the router during the set up. It is used as default configuration in the tunnel namespace. To propagate the DNS servers to the mobile phones as well, the list is extracted and the different servers are set in the configuration of the DHCP server from the WAPs.

5.6.4 Network Controller

Besides the automation and control of the mobile phones, the network setup has to be automated and controlled to allow clean measurements. The controller has to set up the tunnels from the routers to two PlanetLab nodes. Even if the uptimes are relatively stable, it can happen that PlanetLab nodes are not responsive and need to be exchanged by the controller. Furthermore there are some nodes differently configured with DNS servers with private IP addresses. Using those DNS servers on the tunnel starting points does not work. All DNS responses reaching the tunnel starting point are rejected by the router and not routed to the mobile phones. Thus DNS resolution is not possible with these DNS servers in combination with the tunnels. At this point, the PlanetLab node has to be excluded permanently from the measurement. A list of nodes that were excluded in this thesis is given in Table 5.2.
After establishing the tunnels and confirming their functionality, the configurations of the DHCP servers of the WAPs have to be changed to serve the new DNS servers

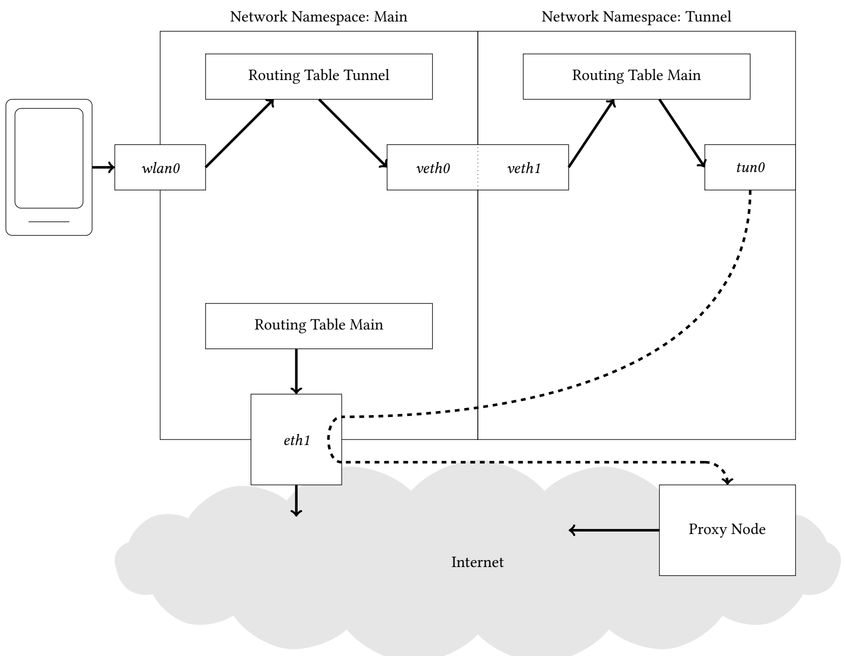


Figure 5.2: Connection between the Network Namespaces and Network Devices

Country	Node
Germany	planetlab3.net.in.tum.de
Hungary	planetlab1.tmit.bme.hu
	planetlab2.tmit.bme.hu
Switzerland	icnalplabs2.epfl.ch
France	anateus.ipv6.lip6.fr
Réunion	lim-planetlab-2.univ-reunion.fr

Table 5.2: PlanetLab Nodes with Problematic DNS Servers

and they have to be restarted to adapt the changes. Finally the interception software has to be started and the mobile phone controller can send the messages. After the measurement finishes all network components have to be torn down and cleaned up to facilitate the next measurement.

5.7 Integrated Data Analysis

So far, the framework is only able to intercept the complete traffic without any constraints or analysis steps. This leads to dump files containing the data. The intercepted data still has to be parsed and path measurements to the target IPs have to be executed. Those path measurements have to be executed with the emulated geographical locations as well to reproduce the paths of the beforehand intercepted traffic. Therefore an inte-

gration into the framework is implemented to use the already established infrastructure. The dump files are examined with scapy[9] to extract several information. First, DNS resolutions are inspected and all mappings between hostnames and according IPs are extracted for further analysis. Second, all established network connections are extracted as a triple that contains the destination IP, the used transport layer protocol and the according port number. Before the path measurements are done, a filter is applied to the set of triples, that filters out several values that are irrelevant, for example connections to the UDP port 123 belonging to the network time protocol [43] or connections to addresses out of private ranges according to RFC1918 [39]. Third, path measurements are conducted with `traceroute`. The path measurements are made either with the `-T` option for TCP traceroutes with *syn probes* or the `-U` option for UDP traceroutes, both to the according destination port wit the `-p` option, to stay as close as possible to the intercepted network traffic.
At the time of the measurement the implementation of the proxy software was not finished completely and only provided for the framework in a beta version. Because of this, TCP path measurements over the tunnel were not possible at that moment. To still be consistent with the emulated geographical locations of the mobile phones the path measurements are started directly on the according PlanetLab nodes and the results are copied back to the framework. So far only commodity hardware is used in the framework. To permanently store the data, it is copied to a more reliable storage server at the end of each measurement.

5.8 Summary

In this Chapter the framework implemented for this thesis was described. The requirements were deduced based on the necessary tasks to answer the research questions and combined in a logical composition. Afterwards the implementations of the components were described in more detail. All of them were successfully implemented and combined to form a functional framework as described in the overview. The framework contains the mobile phones and sends messages between them. Geographical locations can be emulated to allow a global view on the locality of mobile messages. The accumulating traffic can be intercepted completely and is subjected to first analysis steps resulting in path measurements for all established network connections. To minimize the amount of path measurements the network traffic is controlled by firewalls and a first filter is applied. The framework is made publicly available[10]. It contains all necessary source codes and further technical information in a README. Exemplary router and controller configurations are added to the repository to support the set up of the framework.

[9]http://www.secdev.org/projects/scapy/
[10]https://github.com/tumi8/matador

Chapter 6

Procedure of Data Collection

Different measurements and tests are done over the whole period of the thesis. Together the results create the dataset for the further analysis. Those measurements include all steps from the application installation to the final path measurements. At the beginning, the application installation is described and it is explained why it is part of the resulting dataset. Afterwards the results and consequences of functional tests of the applications in the framework are shown. Furthermore the procedures to elevate all application and network paths, and to create a basis for a traffic blacklist are described. Findings and resulting statistics are shown in the next section. The last section describes the structure of the published dataset .

6.1 Application Installation inside the Framework

The first measurement conducted with the framework is the installation of the applications to make sure initial steps, which could influence the later measurements are at least captured and observable even if the traffic could be encrypted. Such initial steps could be the exchange of IP addresses, that will not be resolved again in later use or the explicit exchange of the geographical location of a user. The whole process from downloading the applications and installing them is carried out using parts of the framework. Furthermore the measurement includes the first start of the applications, the account creation and first messages between the two mobile phones. Because creating accounts is limited to two, due to the number of mobile phones only two countries are used to tunnel the traffic. Because the download is a component of this measurement as well, the respective remote nodes have to be relatively fast. Therefore one in the U.S. and one in Germany are used. All interactions with the mobile phones to install the applications are still made by hand and no forward path measurements are done based on the intercepted traffic.
Inspecting the resulting network traces afterwards verifies the expectation that the

relevant traffic is encrypted. Therefore, nothing suspicious can be seen but the traces are still provided in the resulting dataset.

6.2 Functional Tests of the Applications in the Framework

The integration of the messengers into the framework is tested before the main path measurements. Even though the applications are installed on real mobile phones and the tunnels and network components are designed transparently for the mobile phones, the traffic generated by the applications is still influenced by the framework. Therefore multiple scenarios with tunnels to different PlanetLab nodes are run through. In each scenario multiple messages are sent between the mobile phones. These tests show that WhatsApp, Threema, TextSecure and WeChat have a functional behavior in the framework. Sometimes WeChat messages are transmitted extremely slowly between the two mobile phones with different PlanetLab nodes but this only comes with the drawback of longer measurement times.

However, Bleep shows no functional behavior. First, it is always extremely slow, even while looking up if the conversation partner is online and messages can therefore be sent peer-to-peer. Second, when executing Bleep inside the framework with working tunnels, messages often are not transmitted, and the use over time even leads to unsuspected errors with the tunnel connection leading to complete malfunction. Third, even when the traffic is routed from the WAPs directly to the Internet without any tunnels, the sending process is slow. Fourth, inspecting several network traces shows that no peer-to-peer connections can be established and the messages are sent over relay servers. This happens most likely due to the use of NATs between the wireless network for the mobile phones and the exit point from the routers to the Internet, directly and with tunnels.
Because of the indeterministic behavior and the fact that peer-to-peer is most likely not working in the framework, Bleep has to be excluded from the list of applications for this thesis.

6.3 Collecting Application Paths with an Application Measurement

To actually be able to talk about the paths of the messages, multiple messages have to be exchanged between different countries for each application. Therefore an application measurement is conducted with the framework. A country list and the application list is used as input data. The exact procedure and combination of the framework functionalities is described in the following steps:

Step 1: The initialization of the measurement consists of one step to set up the mobile phone firewall.

All steps from two to ten are executed for each pair of countries, combined from the country input list.

Step 2: The two tunnels are established from the routers to the PlanetLab nodes for the two countries. At the same time, the GPS coordinates of these PlanetLab nodes are imported to Xprivacy on the mobile phones.

Step 3: After the two tunnels are established, the controller checks whether the tunneling works. If so, the routers change the settings of their DHCP server to propagate the newly received DNS servers to the mobile phones later on. If the tunnel is not working properly, the sequence has to jump back to Step 2.

All steps from four to nine are executed consecutively for each application.

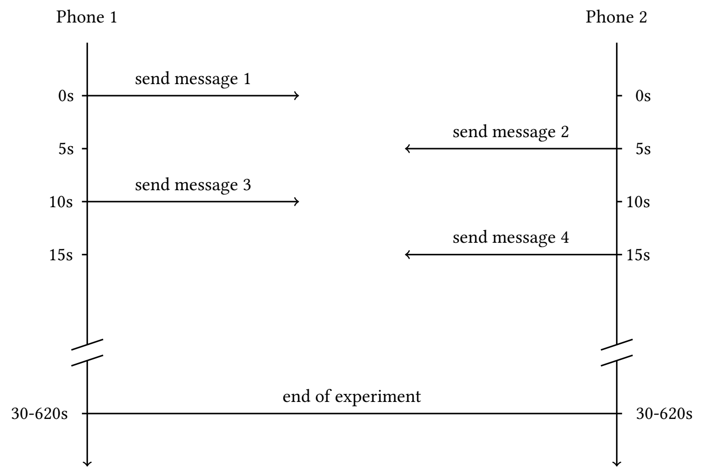
Step 4: The routers start tcpdump to intercept the traffic, generated in the next steps.

Step 5: The displays of the mobile phones are activated and the recordings of the screens started.

Step 6: The firewall is opened for the currently measured application. Afterwards the *wifi* service is enabled and the mobile phones connect themselves to the WAPs.

Step 7: Finally the mobile messaging applications are started on each phone and the messages are sent. Overall, four messages are sent in this step, alternating between the two mobile phones. Multiple post-processing steps need to differentiate between network traffic containing the messages and additional irrelevant network traffic afterwards. To simplify this, the messages are sent with a fixed timeout of five seconds between each other. Afterwards a timeout is waited to make sure all sent messages were received and therefore captured in the same network trace as the sent messages. This timeout varies widely dependent on the application and PlanetLab node used. The timeouts for, WhatsApp, Threema and TextSecure are between 5 and 15 seconds for all PlanetLab nodes used. The timeout for WeChat however fluctuates between five seconds, primarily for nodes in Europe and Asia, and some minutes, e.g. for the nodes in Argentina and Australia. The timeouts at the end are adapted for each part of the measurement based on a set of countries and applications, to minimize the overall measurement time as far as possible. Figure 6.1 illustrates the message sending procedure and all timeouts in between and afterwards.

Step 8: After the final timeout, the mobile phones are restored to a clean state to enable the next application execution. Therefore the applications are terminated, the *wifi* services disabled, the firewalls closed again, the recording of the screens terminated and copied to the controller and finally the displays are turned off again.



Created in Joint Work with Quirin Scheitle for [7]

Figure 6.1: Timeouts between Message Sending

Step 9: Tcpdump is terminated on both routers and based on these results, the first integrated analysis steps are started, as described in Section 5.7.

Step 10: After all applications from the input list sent messages with this network setup and the first analysis steps are done, the results are finally copied to a persistent storage server and the tunnels are torn down.

Step 11: After all country pairs are used, the last step is to remove the firewalls from the mobile phones and bring them back to completely clean states.

The messages are sent while both communication partners are connected to the network and the mobile messaging applications are active on both mobile phones. This results in a limitation that services, used to notify mobile phones about events when they are not connected to application infrastructures are not considered in the measurement, e.g. Google Cloud Messaging[1] (GCM).

6.4 Collecting Data for a Filter Blacklist

Even with the firewalls, additional traffic, independent from the messaging applications can not be suppressed completely. Some applications and tasks, especially from Google or the operating system can not be blocked by the firewall without interfering with the fundamental behavior of the mobile phones. Therefore such traffic can be observed in the results of the application measurement as well and has to be identified, affirmed as irrelevant and filtered during the post-processing. To build a basis for a blacklist, listing such traffic, an *empty measurement* where no applications are executed is done. The

[1]https://developers.google.com/cloud-messaging/

measurement is performed similarly to the application measurement. All steps to set up the network (1-3, 9-11) and measurement tools (4, 9) are executed as well, but instead of executing the applications and sending messages (5-8) a timeout of 60 seconds is waited without doing anything on the mobile phones. For this measurement a subset of three nodes—one in China, Germany and the U.S. respectively—are used, leading to six combinations. By analyzing the results, this empty measurement provides a blacklist that can be used to adjust the application measurements from background sounds.

6.5 Collecting Network Paths with a Direct Measurement

The additionally necessary network paths are performed by further path measurements with `traceroute`. The path measurements are directly made between all pairs of used PlanetLab nodes leading in a direct network path, messages would need to travel without being influenced by the mobile messaging applications and passing any corresponding infrastructure. They are bidirectionally made to absorb differences based on the direction. Listing 6.1 shows the command used for the path measurements. The majority of conducted path measurements in the application measurement are TCP path measurements. Therefore, TCP path measurements with the `-T` parameter are used for the direct measurement to stay as close as possible to the application measurement. To increase the possibility to not interfere with any running applications on the PlanetLab nodes, a high port number (37503) is chosen as destination port. TCP path measurements can only be executed as root, therefore the command is executed with `su -c`.

```
su -c "traceroute -T -p 37504 destination"
```

Listing 6.1: Traceroute Command for the Direct Measurement

The path measurements are combined in one script, shown in Listing 6.2. The script is based on a hostname list of PlanetLab nodes, used in the application measurement. The hostname list is named: `node_list`. The script iterates through all nodes in the hostname list and executes the path measurement of Listing 6.1. In each iteration, the path measurement is made in parallel from all elements of the hostname list to one node after another. The parallelization is made with `parallel-ssh`[2] that parallelizes OpenSSH[3] and related tools. The command is used to establish connections to the PlanetLab nodes in parallel and execute the path measurements. The `-x` parameter forwards the following expression to OpenSSH. The `-t` parameter from OpenSSH forces a pseudo-terminal allocation on the destination of the SSH connection. Tests have shown that a pseudo-terminal is necessary on some PlanetLab nodes, for example iraplab1.iralab.uni-karlsruhe.de, to execute commands as root over SSH. Furthermore

[2]https://code.google.com/p/parallel-ssh/
[3]http://www.openssh.com/manual.html

```
for node in `cat node_list`
do
  parallel-ssh -x "-tt" -h node_list -l tumple_messaging -t 0 -p 100
      -o tmp_results/ "su -c \"traceroute -T -p 37504 $node\""
  #Rename the output files
  for i in `ls tmp_results/`
  do
    begin="traceroute_from_"
    mid="_to_"
    end=$(date +"_%Y%m%d_%H%M").txt"
    newname=$begin$i$mid$j$end
    mv "tmp_results/"$i" final_results/"$newname
  done
done
```

Listing 6.2: ParallelSSH Script for the Direct Measurement

`-t` did not work, but `-tt` was necessary.
Returning to the `parallel-ssh` parameters, the `-h` parameter defines the list of hosts, SSH connections are established to. `parallel-ssh` executes the command on all nodes listed in the file. The `-I` parameter sets the username for the SSH connections. The `-p` parameter sets the maximal number of parallel SSH connections. `-t` sets the SSH timeout for each connection. A value of zero means no timeout. `-o` defines the directory where outputs of the executed command are stored. The filename for the output of each connection is the according hostname of the PlanetLab node. Moving the results to a different directory and unique filenames after each execution of `parallel-ssh` prevents the script to overwrite the results in each iteration. Finally the explicit command executed via each SSH connection is added.

6.6 Measurement Statistics

Due to the automation a large amount of different countries was used in the measurement. The PlanetLab nodes finally used are not determined by any criteria other than availability. During the measurement around 200 nodes out of the announced 1323 were accessible. They are located in 28 different countries, listed in Table 6.1. Unfortunately besides a good coverage of Europe, Asia, Oceania and North America, only nodes in two countries of South America were accessible and none of the countries in Africa had accessible nodes during the measurement. For each application measurement a pair of countries is needed one for each mobile phone. Including the combinations from each country with itself, the measurement can be divided into $(28 * 29)/2 = 406$ different subtasks, representing one country combination each. Table 6.2 shows the distribution

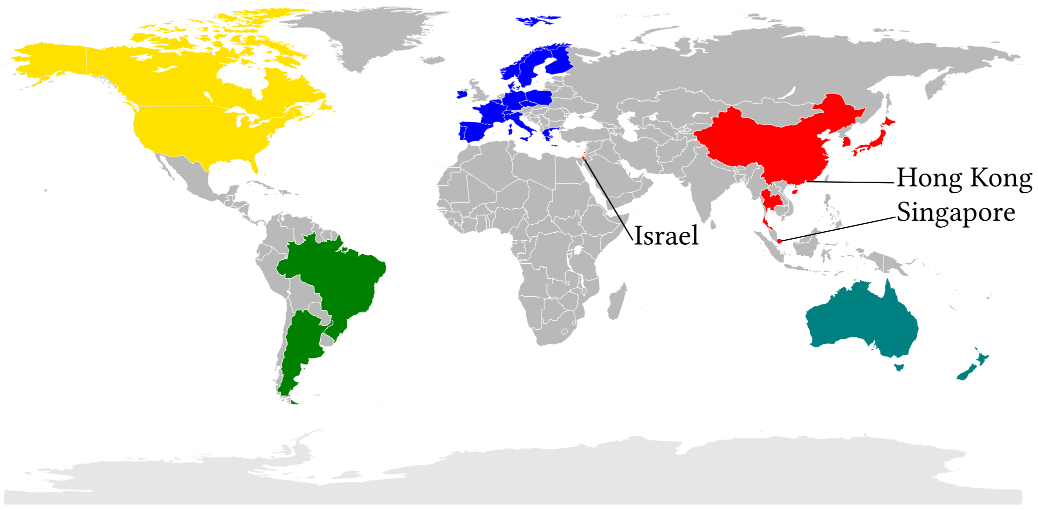


Figure 6.2: Countries with Accessible PlanetLab Nodes

of the different subtasks after dividing the countries into their geographical regions.

The whole measurement was not executable as one large task because of the unstable behavior of the PlanetLab nodes and the fact that WeChat is extremely slow tunneling it through some nodes. Measuring everything as one large measurement would have taken several days. By splitting it into smaller parts, the monitoring could be done better, problems were detected and failed measurements repeated. Therefore the recorded videos were inspected at random and the logs created by the framework checked to find problems with the procedure of the measurement.
As a first step, the initial list of available remote nodes was split into sets based on their country information and geographical region. Afterwards, the applications list was split into two groups, one for TextSecure, Threema and WhatsApp, and one for WeChat. This leads to several measurement parts with a remote node list and an application list for each part.

The measurement parts were done over a time period of two weeks between the 30.09.2015 and 12.10.2015 without changing the version of the applications. The active measurement time was around 96 hours. At the end one last combination was missing, the country combination of Israel with New Zealand for WeChat only. However since then, no Israeli PlanetLab node has been accessible and the measurement could not be conducted resulting in a small gap in the dataset.

Region	Country	#[1]	Node Used
Europe	Belgium	1	planetlab1.extern.kuleuven.be
	Czech Republic	4	planetlab1.cesnet.cz[2]
	Denmark	2	planetlab1.diku.dk
	Germany	14	iraplab1.iralab.uni-karlsruhe.de
	Finland	1	planetlab4.hiit.fi
	France	13	planetlab1.jcp-consult.net
(15)	Greece	2	planetlab3.cslab.ece.ntua.gr
	Ireland	2	planetlab-coffee.ait.ie
	Italy	3	planet-lab-node1.netgroup.uniroma2.it
	Norway	2	planetlab1.ifi.uio.no
	Poland	6	planetlab1.mini.pw.edu.pl
	Portugal	3	planet2.servers.ua.pt
	Spain	5	planetlab1.um.es
	Sweden	5	planetlab-1.ida.liu.se
	Switzerland	4	planetlab2.inf.ethz.ch
Asia	China	9	pl1.6test.edu.cn
	Hong Kong	3	planetlab1.ie.cuhk.edu.hk
	Israel	3	planetlab1.mta.ac.il
(7)	Japan	7	pl1.sos.info.hiroshima-cu.ac.jp
	Korea, Republic of	1	netapp6.cs.kookmin.ac.kr
	Singapore	3	planetlab1.comp.nus.edu.sg
	Thailand	2	ple1.ait.ac.th
Oceania	Australia	2	pl1.eng.monash.edu.au
(2)	New Zealand	4	planetlab1.cs.otago.ac.nz
North America	Canada	9	cs-planetlab3.cs.surrey.sfu.ca
(2)	U.S.	71	planetlab01.cs.washington.edu
South America	Argentina	2	planet-lab2.itba.edu.ar
(2)	Brazil	3	planetlab1.pop-mg.rnp.br

1: Number of accessible nodes during the measurement
2: The Czech PlanetLab node was not accessible for a period of time. Some measurements use planetlab1.fit.vutbr.cz

Table 6.1: Countries with Accessible PlanetLab Nodes

	Europe	Asia	Oceania	North America	South America
Europe	120	105	30	30	30
Asia		28	14	14	14
Oceania			3	4	4
North America				3	4
South America					3

Table 6.2: Distribution of all Subtasks over the Regions

6.7 Structure of the Dataset

The resulting dataset is made publicly available, to allow other researchers to validate the results and and do further analysis.[4] It contains all results from the application installation, application measurement, empty measurement and direct measurement. The application installation contains the network traces, intercepted during the installations and a short additional description for each installation. The direct measurement consists of the outputs from all path measurements stored as text files. The application measurement is divided into further directories for the regions: Europe, Asia with Oceania, the Americas and the combinations of those three. Furthermore they are divided into directories for each application: WhatsApp, TextSecure, Threema, WeChat and XPrivacy. All application measurement directories and the empty measurement consits of the same type of files:

Description: Each directory holds a short description of the measurement data and messenger timeouts.

Logs: Log files are added to the dataset. They hold the timestamps for each country combination, the names of the used PlanetLab nodes and information, which experiment failed and when it was remeasured.

Network Traces: All network traces are available in the dataset as *.dump* files. The file names hold the used PlanetLab nodes during the measurement.

Extracted DNS Resolutions: The DNS resolutions are extracted from the traces for each experiment. They are available as text files. The names of the files contain the keyword *DNS* and the resolved domain name. All IPs resolved are written line for line in the file.

Path Measurements: The output of the executed path measurements for all network connections in each trace is available as text files. The filename contains the keyword *traceroute*. Furthermore it contains the hostname of the PlanetLab node used, the target IP and port number, and the method used.

Screen Records: One version of the dataset additionally contains the screen record for each mobile messaging application and XPrivacy as *.mp4* files. Another version leaves out all video files to minimize the size of the dataset.

6.8 Summary

This chapter described all processes that were executed with the framework to acquire all necessary data for answering the research questions. The application installation was

[4]http://net.in.tum.de/pub/mobile-messaging

made additionally to complete the dataset, but has no direct influence on the analysis, because the information exchange of the applications with the infrastructure of the services is encrypted as expected. The functional tests showed that Bleep can not be integrated into the framework and had to be dropped from the list of applications. All application and network paths are measured and collected in a dataset for all further steps.

Chapter 7

Post-Processing of the Dataset

Different steps are applied to post-process the dataset: First, irrelevant network connections are filtered out. Second, the domain name structure of the messaging infrastructure is analyzed. Third, the remaining IPs are mapped to their geolocation. Fourth, gaps on the measured paths are characterized.

7.1 Filtering non Message Passing Traffic

The focus of the analysis is on the network connections that handle message passing between the sender, the infrastructure of the service and the receiver. Therefore, all irrelevant path measurements are filtered based on the target IP addresses and according domain names. Starting with the *empty measurement* a list of potentially irrelevant domain names and according IPs is created. To make sure all entries of this list can be filtered out of the application measurement, multiple traces from the application measurement are inspected. The inspection confirms the blacklisting can be done. The domain names and according IPs rarely appear in the traces and the times when they appear do not match the predetermined sending times.

After applying this blacklist a large amount of path measurements remains. They can be divided into three groups: First, a group of multiple IPs that can be mapped to domain names that directly relate to the applications. Second multiple IPs that can not be directly mapped to any DNS resolution. So far those groups are considered relevant and inspected in more detail in the next section. Third, a smaller amount of IPs and according traces that can be added to the blacklist after inspecting multiple measurements they belong to. Again they are not considered as relevant for message sending based on their frequency of occurrence and their activity times. The extensive list of filtered domain names and additional IPs is given in Appendix A.2.

7.2 DNS Resolutions from the Messaging Applications

After the filtering step, a list of different DNS resolutions but also a list of IPs, that are not assignable to any DNS resolution remains. It turns out that all non-assignable addresses appear in WeChat measurements. To check if they are already integrated in the application or obtained through another mechanism than DNS, the intercepted traffic of the measurements with such unknown IPs is inspected manually. It arises, that WeChat implements an own mechanism to obtain IPs. It is mainly based on HTTP sent to port 80 or 8080 but there are additional unencrypted HTTP communications to port 443. The mobile phones send HTTP requests with URIs, including "/cgi-bin/micromsg-bin/newgetdns". Afterwards they receive answers in XML format containing plenty of IPs. Listing 7.1 shows parts of a DNS-over-HTTP result from the inner-country measurement of Belgium. The answers contain a domainlist with multiple domains and their according IPs. An additional timeout value for each domain indicates a DNS timeout of 1800. The timeout is most likely in seconds, based on the frequency, such DNS-over-HTTP requests appear. They can be seen approximately every 30 minutes. Unfortunately this interferes with the runtime of the executed measurements.

TShark[1] is used to extract DNS-over-HTTP results from all traces in the dataset. Listing 7.2 shows a script applying `tshark` on all traces and converting the outputs to readable XML Files:
The combination of `-T fields` and `-e data` outputs the data field of intercepted packets. `TShark` treats all TCP connections established to port 443 as HTTPS, but with the `-d tcp.port==443,http` option, the connections are treated as HTTP. This is possible in this case as the DNS-over-HTTP mechanism from WeChat to port 443 is using the HTTP protocol.
The `-r` parameter specifies the input file and the `-Y` parameter applies a filter on all intercepted packets in the input file. Inspecting several traces shows that the filter `'http.response.code == 200 && http.request.version == "HTTP/1.0"'` uniquely identifies packets containing DNS-over-HTTP results.
The data field is written out as hexadecimal string with line breaks. The line breaks have to be removed to convert the output. The `binascii`[2] Python module is used to convert the output to a binary string and readable ASCII letters. Listing 7.3 shows the used implementation, converting the text of an input file and printing the result on the standard output.
All extracted XML files are added to the available dataset.

Finally all available DNS resolutions are analyzed, including the results from the installation traces and the DNS-over-HTTP mechanism. Table 7.1 holds an overview of the results. The Table and parts of the following post-processing step are made in

[1] https://www.wireshark.org/docs/man-pages/tshark.html
[2] https://docs.python.org/2.7/library/binascii.html

```
<?xml version="1.0" encoding="utf-8"?>
<dns>
  <retcode>0</retcode>
  <domainlist>
   <domain name="extshort.weixin.qq.com" timeout="1800">
     <ip>103.7.31.152</ip>
   </domain>
   <domain name="localhost" timeout="1800">
     <ip>127.0.0.1</ip>
   </domain>
   <domain name="long.weixin.qq.com" timeout="1800">
     <ip>103.7.31.151</ip>
   </domain>
   <domain name="minorshort.weixin.qq.com" timeout="1800">
     <ip>103.7.31.152</ip>
   </domain>
   ⋮
   <domain name="hkshort.weixin.qq.com" timeout="1800">
     <ip>203.205.151.160</ip>
     <ip>203.205.147.168</ip>
     <ip>203.205.129.101</ip>
   </domain>
  </domainlist>
  <builtiniplist>
   <ip>203.205.151.164</ip>
   <ip>203.205.143.141</ip>
   <ip>203.205.129.102</ip>
  </builtiniplist>
  <clientip>193.190.168.49</clientip>
  <clientispid>0</clientispid>
  <timestamp>1443628800</timestamp>
  <signature>MDwCHFqMSx/
      PdY6OtMi59uAjQ0JAqe3ZsJ8IUn7Fii0CHDuAtxbwX/XV094cGvj00r83+
      iHY2fFwYKZqmqs=</signature>
</dns>
```

Listing 7.1: Examplary DNS-over-HTTP Result

```
#!/bin/bash
for i in *.dump
do
tshark -T fields -d tcp.port==443,http -e data -r $i
    -Y 'http.response.code == 200 && http.request.version == "HTTP/1.0"'
    | tr -d '\n' > tempfile
python unhexlify.py tempfile > dns_over_http/$i.xml
done
```

Listing 7.2: Bash Script to Extract DNS-over-HTTP with TShark

```
import binascii
import sys
string = open(sys.argv[1],'r').read()
sys.stdout.write(binascii.unhexlify(string))
```

Listing 7.3: Converting from Hexadecimal to Ascii with Python

joint work with Quirin Scheitle for [7]. TextSecure uses one domain name, textsecure-service.whispersystems.org., that resolves to two unique IPs out of a subnet from the Amazon Web Services. Both IPs are located in the U.S.. This disproves the results of the research for the application taxonomy of Chapter 4. According to [32], TextSecure uses globally distributed servers. However, during the measurements TextSecure only used an infrastructure located in the U.S..
Threema uses three different domain names, api.threema.ch., g-40.0.threema.ch. and g-e8.0.threema.ch.. The first one resolves to one unique IP address which is infrequently used. By checking multiple traces it can be shown that the connections to this IP address can not be part of the sending mechanism because it is used only for a short period of time outside of the sending window predetermined by the application measurement process. The domain name and IP address is added to the filter blacklist. The second and third domain name resolve to the same two addresses. Both of them seem to be used for sending and receiving messages. All three IPs are part of one subnet, owned by Threema itself.
WhatsApp divides their IPs over multiple domain names, e[1-16].whatsapp.net.. Each domain name resolves to 25 unique IPs. Therefore 400 unique IPs of WhatsApp are resolved during the application measurement. They are distributed over different subnets owned by SoftLayer. The Planet Internet Services—owner of the subnet 184.172.0.0/15—is part of Softlayer as well, after they merged in 2010 [44]. All 400 IPs are located in the U.S..
So far there seems to be no relationship between the location of the requesting phones and the received answers. All domain names globally resolve to the same set of IPs.

Beside the two different mechanisms for domain name resolving at WeChat another interesting result is the difference between the IPs resolved in China on the one hand and in the rest of the measured countries on the other hand. Starting with the direct DNS resolutions, the first domain name is mp.weixin.qq.com.. It resolves to one IP from China and a different IP from the rest. In China the IP is part of the China Education and Research Network and in the rest it is part of a subnet directly owned by Tencent. Both unique IPs were never used, but other addresses out of the Tencent subnet were used, thus, the network is considered as relevant in Table 7.1. The second domain name is masdk.3g.qq.com., the only domain name resolving to the same IPs out of a subnet owned by China Unicom Shanghai from both regions. Outside of China DNS resolutions are done for the domain names hkextshort.weixin.qq.com. and hklong.weixin.qq.com.. They resolve to three unique IPs each, all of them part of a subnet owned by Tencent. In China, only one IP is resolved for hklong.weixin.qq.com. from a subnet of China Unicom Guangdong. The last domain name resolved over DNS is dns.weixin.qq.com.. Outside of China, it resolves to three unique IPs, again part of the network from Tencent. In China it also resolves to three unique IPs, they are located in different regional networks of China Unicom. One from Zhejiang, one from Sichuan and the last one from Tianjin. Those IPs are used to conduct the DNS-over-HTTP mechanism.

The results from the DNS-over-HTTP mechanism can be divided into three large blocks. The first block contains multiple domain names similar to the already mentioned names hkextshort.weixin.qq.com. and hklong.weixin.qq.com.. The following list shows all of them. They are all under weixin.qq.com.:

1. caextshort
2. calong
3. caminorshort
4. cashort
5. extshort
6. hkextshort
7. hklong
8. hkminorshort
9. hkshort
10. long
11. minorshort
12. sh2tjextshort
13. sh2tjlong
14. sh2tjminorshort
15. sh2tjshort
16. shextshort
17. short
18. sz2tjextshort
19. sz2tjlong
20. sz2tjminorshort
21. sz2tjshort
22. szextshort
23. szlong
24. szminorshort
25. szshort
26. tjextshort
27. tjlong
28. tjshort

They seem to divide the traffic to different servers based on Chinese regions. A precise analysis was not possible because the resolved IPs are largely distributed over the different domain names. The important networks the domain names resolve to, are 203.205.128.0/19 outside of China, again a subnet from Tencent and 14.16.0.0/12 and 183.61.32.0/19 inside of China, both owned by ChinaNet Guangdong. All other resolved subnets are listed in Table 7.1 but are not used in the measurement by WeChat and therefore considered as irrelevant.

The elements of the second block seem to be different media and ad services, neither of the according IPs were used in the measurements. For the sake of consistency, they are listed in Table 7.1.

The third block, delivered through the DNS-over-HTTP mechanism is not part of the domainlist but labeled as *buitliniplist*, shown in Listing 7.1. Outside of China the

Textsecure DNS results

Domain names resolved	Unique IPs	Longest routed prefix (Inetnum owner)	AS	Location[1]	Relevant[2]?
textsecure-service.whispersystems.org.	2	52.4.0.0/14 (Amazon AWS)	14618 (Amazon)	US	✓

Threema DNS results

Domain names resolved	Unique IPs	Longest routed prefix (Inetnum owner)	AS	Location[1]	Relevant[2]?
g-[40|e8].0.threema.ch.	2	5.148.175.192/27 (Threema)	29691 (NINE)	CH	✓
api.threema.ch.	1	5.148.175.192/27 (Threema)	29691 (NINE)	CH	✓

WhatsApp DNS results

Domain names resolved	Unique IPs	Longest routed prefix (Inetnum owner)	AS	Location[1]	Relevant[2]?
e[1-16].whatsapp.net.	400[3]	108.168.128.0/17 (SoftLayer)	36351 (SoftLayer)	US	✓
		158.85.0.0/16 (SoftLayer)	36351 (SoftLayer)	US	✓
		169.54.0.0/18 (SoftLayer)	36351 (SoftLayer)	US	✓
		173.192.0.0/15 (SoftLayer)	36351 (SoftLayer)	US	✓
		184.172.0.0/15 (ThePlanet)	36351 (SoftLayer)	US	✓
		50.22.0.0/15 (SoftLayer)	36351 (SoftLayer)	US	✓
		50.97.0.0/16 (SoftLayer)	36351 (SoftLayer)	US	✓

WeChat DNS results

Domain names resolved	Unique IPs	Longest routed prefix (Inetnum owner)	AS	Location[1]	Relevant[2]?
mp.weixin.qq.com.	1 (in China)	115.25.209.0/24 (Cernet[6])	4538 (CERNET[6])	CN	✗
	1 (outside China)	203.205.128.0/19 (Tencent)	132203 (Tencent)	CN	✓
masdk.3g.qq.com.	4	140.207.54.0/23 (CU[4] Shanghai)	17621 (CU[4] Shanghai)	CN	✓
	1 (in China)	124.161.0.0/16 (CU[4] Sichuan)	4837 (CU[4])	CN	✗
	1 (in China)	125.36.0.0/14 (CU[4] Tianjin)	4837 (CU[4])	CN	✗
	3 (outside China)	203.205.128.0/19 (Tencent)	132203 (Tencent)	CN	✓
hkextshort.weixin.qq.com.	3[5] (outside China)	203.205.128.0/19 (Tencent)	132203 (Tencent)	CN	✓
hklong.weixin.qq.com.	3[5] (outside China)	203.205.128.0/19 (Tencent)	132203 (Tencent)	CN	✓
	1 (in China)	163.177.70.0/23 (CU[4] Guangdong)	17623 (CU[4] Shenzen)	CN	✗
dns.weixin.qq.com.	1 (in China)	124.160.0.0/16 (CU[4] Zhejiang)	4837 (CU)	CN	✓

WeChat DNS-over-HTTP results

Domain names resolved	Unique IPs	Longest routed prefix (Inetnum owner)	AS	Location[1]	Relevant[2]?
several[7].weixin.qq.com	13 (outside China)	203.205.128.0/19 (Tencent)	132203 (Tencent)	CN	✓
	2 (outside China)	103.7.31.0/24 (Tencent)	132203 (Tencent)	CN	✗
	9 (outside China)	203.205.160.0/24 (Tencent)	132591 (Tencent)	CN	✗
	1 (outside China)	203.205.179.0/24 (Tencent)	132591 (Tencent)	CN	✗
several[7].weixin.qq.com	6 (in China)	101.224.0.0/13 (CN[8] Shanghai)	4812 (CN[8] Shanghai)	CN	✗
	2 (in China)	123.151.8.0/21 (DINGSHENG-LTD)	17638 (CT[9] Tianjin)	CN	✗
	7 (in China)	14.16.0.0/12 (CN[8] Guangdong)	4314 (CT[9])	CN	✓
	5 (in China)	183.61.32.0/19 (CN[8] Guangdong)	4816 (CT[9])	CN	✓
	1 (in China)	183.232.96.0/22 (China Mobile)	56040 (China Mobile)	CN	✗
	4 (in China)	61.151.128.0/17 (CN[8] Shanghai)	4812 (CN[8] Shanghai)	CN	✗
media & ad services[10]	1 (outside China)	103.7.31.0/24 (Tencent)	132203 (Tencent)	CN	✗
	1 (outside China)	184.104.0.0/15 (HE[12])	6939 (HE[11])	US[12]	✗
	4 (outside China)	203.205.128.0/19 (Tencent)	132203 (Tencent)	CN	✓
	2 (outside China)	220.249.243.0/24 (CN[8] Guangdong)	17623 (CU[7] Shenzen)	CN	✗
	1 (outside China)	54.229.0.0/17 (Amazon)	16509 (Amazon)	IE[12]	✗
	2 (outside China)	54.232.192.0/18 (Amazon)	16509 (Amazon)	BR[12]	✗
media & ad services[11]	2 (in China)	61.151.128.0/17 (CN[8] Shanghai)	4812 (CN[8] Shanghai)	CN	✗
	1 (in China)	140.207.56.0/21 (CU[4] Shanghai)	17621 (CU[4] Shanghai)	CN	✗
	4 (in China)	59.37.96.0/19 (CN[8] Guangdong)	4816 (CT[9])	CN	✗
	11 (in China)	101.224.0.0/13 (CN[8] Shanghai)	4812 (CN[8] Shanghai)	CN	✗
	2 (in China)	122.136.0.0/13 (CU[4] Jilin)	4837 (CU[4])	CN	✗
	2 (in China)	123.125.64.0/18 (CU[7] Beijing)	4808 (CH CNC)	CN	✗
	3 (in China)	218.8.0.0/15 (CN[8] Heilongjiang)	4837 (CU[4])	CN	✗
	2 (in China)	218.60.0.0/16 (CU[4] Liaoning)	4837 (CU[4])	CN	✗
	4 (in China)	220.194.192.0/18 (CU[4] Comms)	4837 (CU[4])	CN	✗
	2 (in China)	42.224.0.0/12 (CU[4] Henan)	4837 (CU[4])	CN	✗
builtiniplist[14]	3[15] (outside China)	203.205.128.0/19 (Tencent)	132203 (Tencent)	CN	✓
	1 (outside China)	203.205.167.0/24 (Tencent)	132591 (Tencent)	CN	✓
	1 (in China)	115.236.128.0/19 (CN[8] Zhejiang)	58461 (CT[9] Hangzhou)	CN	✗
	1 (in China)	123.151.8.0/21 (DINGSHENG-LTD)	17638 (CT[9] Tianjin)	CN	✗
	1 (in China)	182.140.164.0/22 (CN[8] Sichuan)	38283 (CN[8] Sichuan)	CN	✗
	1 (in China)	183.60.0.0/20 (CN[8] Guangdong)	4816 (CT[9])	CN	✗

1: Determined by manual analysis of intermediate hops and latencies for traceroutes from every source country
2: If we could not exclude the subnet from message passing, it is considered relevant
3: Precisely 25 IP addresses per hostname, with no overlap between hostnames 4: China Unicom
5: Same specific IPs as through DNS-over-HTTP from outside China 6: CERNET: China Education and Research Network
7: shextshort,[tjextshort|tjlong|tjshort],[long|short|minorshort|extshort], [sh2tj|sz2tj|sz|ca|hk][extshort|long|minorshort|short]
8: ChinaNet 9: China Telecom 10: wx.qlogo.cn, mmsns.qpic.cn,shp.qlogo.cn
11: Hurricane Electric 12: Not validated as not relevant
13: wx.qlogo.cn, mmsns.qpic.cn,shp.qlogo.cn, weixinc2c.tc.qq.com, wxsnsad.tc.qq.com, vcloud1023.tc.qq.com, vweixinf.tc.qq.com
14: DNS-over-HTTP holds built in IP lists, with IPs that are used frequently
15: Same specific IPs as through DNS for dns.weixin.qq.com. outside of China

Created in Joint Work with Quirin Scheitle for [7]

Table 7.1: DNS Resolution per Messenger

buitliniplist contains four IPs over all measurements. Three of them are the same specific IPs the domain name dns.weixin.qq.com. resolves to through DNS. One additional IP, part of the subnet 203.205.167.0/24 owned by Tencent, only appears as built in IP. The IP is not resolved from any domain name or mapped to a domain in the DNS-over-HTTP results. Inside of China, the *builtiniplist* contains four unique IPs, out of four different subnets. All four IPs are not mapped to a domain name in the dataset neither with the results of DNS nor DNS-over-HTTP. However all four IPs are used infrequently during the WeChat measurements. Manually inspecting the according network traces shows that all of them are only destination IPs of connections used for the DNS-over-HTTP mechanism and the connections do not deliver messages.

Considering the domains of all applications already implies that mobile messaging can not be local for the most regions. All infrastructures are only located in one country each. The country mapping is done with the method described in the next Section. Therefore these countries have to be part of each path, interfering with the locality.

7.3 Mapping IPs to Geolocations

The IPs from the path measurements have to be mapped to their according geolocations to analyze the regions a path traverses. The mapping is initially done with the IP-Country database provided by IP2Location[3]. Such databases are often not entirely correct and some IPs are mapped to the wrong geolocation. To not fully rely on the database, non-local paths are manually inspected. Some regions have to be definitely traversed by communication paths because the sender and receiver or the infrastructure of the used mobile messaging application is located there. These regions are excluded from the manual verifications. The IPs of all additional regions are checked until one IP mapping of the database can be verified for each region or all mappings can be changed to other regions. Listing 7.4 shows the result of a forward path measurement between the used PlanetLab node in Norway and a server of WeChat in China. The application of the IP-Country database results in three regions: Europe, Asia and North America. Europe and Asia has to be definitively traversed because of the start and end point of the path. However, North America is an additional region. The according IP is 109.105.97.73 in hop ten, mapped to the U.S..

Several steps are made to check the mapping between IPs and the results of the IP-Country database: First, the AS number and INETNUM owner is inspected to get an overview of the subnet and specific IPs. However most of the inspected subnets and IPs are assigned to large ISPs with globally distributed networks and no geolocation can be derived. 109.105.97.73 is for example part of AS2603 and the INETNUM owner is NORDUnet. NORDUnet hosts a network with gateways in the U.S..

[3]http://www.ip2location.com/

```
traceroute to 203.205.143.143 (203.205.143.143) , 30 hops max, 60 byte packets
1 193.157.115.249 (193.157.115.249) 0.521 ms 0.463 ms 0.525 ms
⋮
8 dk−uni.nordu.net (109.105.97.133) 17.450 ms 17.427 ms 17.401 ms
9 de−hmb.nordu.net (109.105.97.14) 23.933 ms 24.032 ms 24.009 ms
10 de−ffm.nordu.net (109.105.97.73) 32.318 ms 32.296 ms 32.431 ms
11 br02. frf02 .pccwbtn.net (80.81.192.50) 33.550 ms 33.738 ms 33.776 ms
12 TenGE0−1−0−18.br02.hkg08.pccwbtn.net (63.223.29.14) 341.750 ms
     TenGE0−0−0−4.br02.hkg08.pccwbtn.net (63.218.204.154) 340.910 ms
     TenGE0−1−0−18.br02.hkg08.pccwbtn.net (63.223.29.14) 341.912 ms
13 TenGE0−0−0−4.br02.hkg08.pccwbtn.net (63.218.204.154) 340.571 ms
     TenGE0−1−0−18.br02.hkg08.pccwbtn.net (63.223.29.14) 341.421 ms *
⋮
16 203.205.143.143 (203.205.143.143) 341.920 ms 342.733 ms 334.028 ms
```

Listing 7.4: Traceroute from planetlab1.ifi.uio.no to a WeChat Server

Second, the reverse DNS labels of the IPs are identified. Some of them contain location information that can help to correct the mapping of the geolocations. The reverse DNS name of 109.105.97.73 is *de-ffm.nordu.net* indicating a geolocation in Germany.
As this step is still inaccurate as shown in [45], the results of the path measurements are manually analyzed to verify or disprove the placement with regard to different aspects as a third step:
Round Trip Time (RTT): The RTT is used to estimate the distance between the hop and the origin of the path measurements. The origins of the paths are always PlanetLab nodes with well-known locations. In the example of Listing 7.4 the RTT between the PlanetLab node in Norway and the inspected IP is 32.318 ms. This RTT strengthens the assumption that the correct geolocation is Germany as the time is too short for a hop in the U.S..
Location of the neighbors: The geolocations of neighboring hops and their RTT in comparison to inspected IPs support the assumptions of the steps before. Small deviations in the RTT indicate the same region or country for the IPs. Larger deviations indicate a transition between regions. In the example, hop nine and eleven are mapped to Germany by the IP2location database and the reverse DNS names and RTTs support the mapping this time. Comparing the RTT of the inspected IP with its neighbors shows that they do not differ greatly. The IP 109.105.97.73 is therefore mapped manually to Germany.

7.4 Characterization of Different Gaps on the Paths

One last step before analyzing the locality of the messages is to consider gaps on the measured paths and characterize them. There are two reasons for possible gaps for the

```
 traceroute  to  5.148.175.201  (5.148.175.201) , 30 hops max, 60 byte packets
1 host129.190−227−163.telecom.net.ar  (190.227.163.129)  1.395 ms 1.378 ms 1.454 ms
⋮
8 xe−11−0−0.edge2.miami1.Level3.net  (63.209.150.165)  130.908 ms 137.789 ms 138.165
    ms
9 ae−2−26.bar1.Zurich1.Level3.net  (4.69.142.133)  278.270 ms 278.336 ms 278.079 ms
10 ae−2−26.bar1.Zurich1.Level3.net  (4.69.142.133)  248.524 ms 272.676 ms 272.715 ms
11 NINE−INTERN.bar1.Zurich1.Level3.net (213.242.67.94)  263.285 ms 266.851 ms *
12 * * *
⋮
16 * ps1.threema.ch  (5.148.175.201)  259.566 ms 255.001 ms
```

Listing 7.5: Traceroute from planet-lab2.itba.edu.ar to a Threema Server

application and network paths and one additional reason for the application paths.

First, path measurements only represent a logical path considering all routers and nodes between two nodes. The physical layer, connecting all routers and nodes on the logical path are not considered. The physical layer can possibly traverse multiple countries or regions. These countries can not be measured but are nevertheless relevant and are security threats as it is possible to access the data directly on the link by wiretapping or similar attacks. Listing 7.5 shows a path measurement from the PlanetLab node used in Argentina to the infrastructure of Threema in the Switzerland. Hop eight of the path is a router located in the U.S.. The next hop is located in the Switzerland. They are logical hops on the path responding to path measurements. They have to be connected over a physical layer. As the Switzerland has no cost and is not bordering directly to the U.S., the physical layer has to traverse other countries.
Second, not all hops are responsive during the path measurements. Even though a large amount of nodes respond to the methods, used to conduct path measurements, there are some nodes not answering, leading to gaps in the path measurements, for example hop twelve in Listing 7.5. By inspecting the hop before and after such a gap, the likelihood that this missing hop is in the same country can be estimated relatively well, but there still remains the possibility that countries are missing. Without further knowledge about all routing policies that influence the paths and the topology of all networks traversed, an analysis of those gaps is impossible and therefore they are only mentioned here, but not further considered in this thesis.

Additionally, the message path can only be measured between the two used PlanetLab nodes and the application service infrastructure. In each conversation only a subset and often only a single IP is used from each mobile phone. Especially for WhatsApp and WeChat the amount of available IPs is relatively high and the possibility that both mobile phones establish connections to the same IPs is accordingly small. Figure 7.1 shows an exemplary service infrastructure with four servers. Mobile phone A establishes a

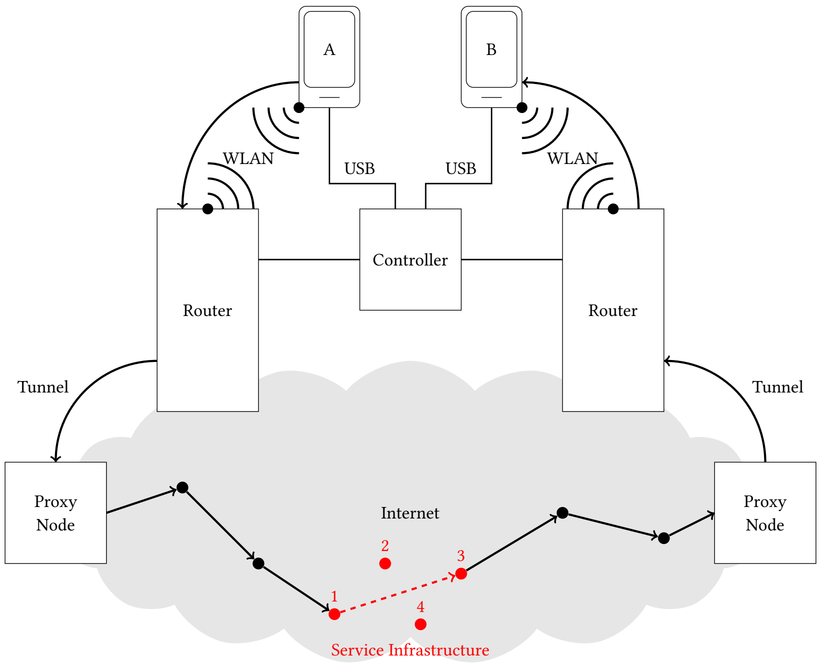


Figure 7.1: Application Path trough a Service Infrastructure

connection to server 1. However, mobile phone B establishes a connection to server 3. A message from A to B is routed over the router and tunnel to server 1. Afterwards the message has to pass the gap between server 1 and 3 to be delivered. From server 3 the message is routed over the tunnel and router to mobile phone B. The gap between server 1 and 3 can not be measured by the framework. For each messenger, the IPs are located in only one country, therefore this gap is most likely passed within the same country as well and has no impact on the locality of the message. This gap is still another non-observable factor that affects the user security and privacy and increases the trust a user has to have in mobile messaging applications.

7.5 Summary

This chapter described all post-processing steps applied to the dataset to enable an analysis. All irrelevant path measurements were filtered out of the dataset by creating and applying a blacklist. Afterwards the domain names and their resolutions were inspected for all applications to identify relevant traffic for message passing. To analyze the locality, the IPs were mapped to their geolocations with a database. The results of the mapping were corrected for 103 IPs. A full list of those can be found in Appendix A.3. The possible gaps of the path measurements were characterized, but will not be considered in the further analysis.

Chapter 8

Evaluation of the Locality of Mobile Messages

After post-processing the dataset, the network and application paths remain and can be analyzed. All paths are categorized as *local* as defined in Chapter 1 or deviating from a local path. The magnitude of deviation, is considered, resulting in further categories: *local+1*, *local+2* and *local+n* (for the amount of additional regions one to n). Firstly, the locality is analyzed for all communication paths within the same region. Secondly, all remaining combinations between different regions, are analyzed. Thirdly, the magnitude of the deviation is evaluated, the deviation traced back to the mobile messengers and the differences between the messengers carved out.

8.1 Locality Analysis of Communications in the same Region

Communications within the same region are expected to stay within this region. Table 8.1 shows the results of the analysis for all messengers, based on each region. The following passages explain each value:

Europe: The 15 accessible countries in Europe add up to 120 combinations and communication paths. Considering the direct network paths, all of them stay inside Europe. All paths for Threema are routed over Switzerland but still remain inside Europe. WhatsApp and TextSecure break the locality adding North America to the message paths. WeChat even adds two more regions to the application paths for most of the combinations by routing to Asia via North America. Only the direct communication within Switzerland is routed over an allegedly direct link from Europe to the WeChat infrastructure in Asia.

Asia: Asia has accessible nodes in seven countries, leading to 28 combinations. The network paths only remain local in Asia for 22 paths. Israel never seems to be locally routed to other Asian countries. Israel to Thailand is routed only via Europe whereas

Israel to the other five Asian nodes is routed via Europe and North America. Israel to Israel remains inside Asia. WhatsApp and TextSecure break the locality, adding North America to all paths, resulting in 21 paths containing Asia and North America and in seven paths for Israel routing via Europe and North America, leading to two more regions in the path. Four application paths of Threema are *local+1*. Israel and Thailand are directly routed to Europe, resulting in three paths. The path between Thailand and Korea is the fourth path directly routed to Europe. The remaining 24 paths are routed via North America to Europe. The application paths for WeChat are only local for 14 combinations even though the WeChat infrastructure is also located in Asia. Traffic from Singapore seems to be routed via North America to WeChat servers, leading to seven paths in the category *local+1*. Again, Israel is routed via Europe and North America. This time, the communication within Israel is affected as well.

Oceania: Combining New Zealand and Australia as Oceania results in three possible combinations. All network paths remain local in the region. WhatsApp and TextSecure are routed to North America due to the infrastructure of the services adding one more region to the paths. The application paths of Threema contain Switzerland and therefore Europe and are additionally routed through North America. The paths from Australia to the infrastructure of WeChat contain hops in North America, adding two more regions to the paths. New Zealand is directly routed to the infrastructure in Asia, resulting in only one additional region in the combination with itself.

North America: North America consists of three different combinations as well. The network, WhatsApp and TextSecure paths remain all local. Using Threema adds Europe as one more region to the paths and WeChat even adds two more regions to the paths, because for both—the U.S. and Canada—the traffic is routed to the WeChat infrastructure in Asia over Oceania.

South America: The last remaining region is South America, again with two countries and three combinations. Only the communications within a country stay in South America. The network path between Brazil and Argentina already contains North America. The WhatsApp and TextSecure paths additionally contain North America for all three paths. Threema contains Europe besides North America and WeChat contains Asia besides North America.

8.2 Locality Analysis of Communications between two Regions

Communicating between two regions results in ten combinations which will be analyzed separately in this section. This time a local path is defined as the set of regions, to stay with the definition of locality. Again the categories *local+n* stand for *n* additional regions. The results are summarized in Table 8.2 and explained in the following passages:

Region	#Paths	Network/ Application	Local #	Local %	Local+1 #	Local+1 %	Local+2 #	Local+2 %
Europe	120	Network	120	100%	0	0%	0	0%
		WhatsApp	0	0%	120	100%	0	0%
		TextSecure	0	0%	120	100%	0	0%
		Threema	120	100%	0	0%	0	0%
		WeChat	0	0%	1	1%	119	99%
Asia	28	Network	22	78%	1	4%	5	18%
		WhatsApp	0	0%	21	75%	7	25%
		TextSecure	0	0%	21	75%	7	25%
		Threema	0	0%	4	14%	24	86%
		WeChat	14	50%	7	25%	7	25%
Oceania	3	Network	3	100%	0	0%	0	0%
		WhatsApp	0	0%	3	100%	0	0%
		TextSecure	0	0%	3	100%	0	0%
		Threema	0	0%	0	0%	3	100%
		WeChat	0	0%	1	33%	2	67%
North America	3	Network	3	100%	0	0%	0	0%
		WhatsApp	3	100%	0	0%	0	0%
		TextSecure	3	100%	0	0%	0	0%
		Threema	0	0%	3	100%	0	0%
		WeChat	0	0%	0	0%	3	100%
South America	3	Network	2	67%	1	33%	0	0%
		WhatsApp	0	0%	3	100%	0	0%
		TextSecure	0	0%	3	100%	0	0%
		Threema	0	0%	0	0%	3	100%
		WeChat	0	0%	0	0%	3	100%

Green highlighted: Recommended application based on the locality analysis

Table 8.1: Locality of Messages for Communication Partners in the same Regions

Europe × Asia: Combining the 15 European countries with the seven Asian ones, leads to 105 path combinations. Fifty-seven network paths are local and the remaining 48 contain North America as an additional region. The network paths from Korea (15 paths), Japan (15 paths) and Hong Kong (15 paths) to all European countries are routed via the U.S.. Furthermore, the network paths between France on the one side and China, Singapore and Thailand on the other are additionally routed via North America. All application paths of WhatsApp and TextSecure are non-local, adding North America to all paths. Threema preserves the locality for 34 combinations. They are divided into 15 paths from Thailand and 15 paths from Israel to all European countries and four paths from Korea to Germany, Spain, France and Poland respectively. The remaining 71 additionally contain North America. The WeChat paths are mostly routed through North America, only the communications between Switzerland on the one side and China, Hong Kong, Korea, Japan and Thailand on the other are local.

Europe × Oceania: These regions lead to 30 combinations. Not a single one is local. All of the network paths contain at least North America as an additional region. The path between Portugal and New Zealand even traverses Asia. On the one side, the path from Portugal to New Zealand is routed through China and Australia to New Zealand. On the other side, the path from New Zealand to Portugal is routed through the U.S. and the UK. As all paths are measured bidirectionally, the combination contains both: North America and Asia. All 30 paths of WhatsApp, TextSecure and Threema are additionally routed through North America. WeChat again deviates the most from local paths, adding not only Asia due to the location of the service infrastructure in China but North America as well for most of the paths. Only the application path between Switzerland and New Zealand leaves out North America. Traffic from these two countries to the WeChat infrastructure is routed directly to Asia as seen in the inner-region communication for Europe and Oceania.

Europe × North America: The 15 European countries with the two North American ones add up to 30 combinations. The network paths and the application paths for WhatsApp, TextSecure and Threema are all local. The application paths for WeChat contain Asia and Oceania additionally for nearly all paths. Only the path between Belgium and the U.S. is not routed via Oceania, still breaking the locality with one additional region.

Europe × South America: The 30 network paths from Europe to South America are local in 16 instances, whereas 14 paths are additionally routed via North America. 13 paths from Brazil to Europe are local. The remaining two Brazilian paths to Switzerland and France are routed through Miami in the U.S.. Argentina routes twelve times via the U.S.. The paths to Finland, Norway and Sweden a routed via a direct link between Buenos Aires and London and do not pass North America. Mobile Messaging applications break the locality for all paths. All WhatsApp, TextSecure and Threema paths in addition contain North America. WeChat adds not only Asia but North America to the region set, leading to two more traversed regions.

Asia × Oceania: Asia combined with Oceania results in 14 country pairs. Only seven network paths are local. The network paths from China, Hong Kong and Japan to Australia, and the network paths from Japan and Korea to New Zealand are routed over North America. Israel is again routed through Europe and North America to Oceania. The message paths for WhatsApp and TextSecure are in twelve cases categorized as *local+1*, due to additional hops in North America and twice categorized as *local+2*, due to the behavior of traffic from Israel. All 14 Threema paths are categorized as *local+2*, due to the message paths through North America to the infrastructure in Europe.
WeChat is only represented by 13 paths, due to the inaccessibility of the PlanetLab nodes in Israel at the end of the measurement and therefore due to the missing messages between New Zealand and Israel as explained in Section 6.6. The measured paths for WeChat are spread over all categories. The five paths from New Zealand to China, Hong Kong, Japan, Korea and Thailand respectively are local, the single Israeli path is again *local+2* and the paths from Australia and Singapore are routed over North America to the WeChat infrastructure as in the inner-region communication of Oceania and Asia, resulting in 7 paths under the category *local+1*.

Asia × North America: All Asian countries combined with the North American ones lead to 14 paths. The categorization of the network paths and the application paths for WhatsApp and TextSecure is once more the same, with twelve local paths. The two paths from Israel to the U.S. and Canada add Europe to the region set. The North American countries again route all their traffic to the WeChat infrastructure via Oceania, resulting in twelve paths under the category *local+1* and two paths under *local+2*. The two outliers can again be traced back to the behavior of Israel routing via Europe to Asia and North America. All Threema paths are plus one due to the routing through Europe.

Asia × South America: None of the paths are local for the 14 communications between Asia and South America. The network, WhatsApp, TextSecure and WeChat paths all span at least North America and again additionally go through Europe for the communications with Israel. All Threema paths contain North America and Europe in addition.

Oceania × North America: Combining the two regions leads to four possible combinations. The network, WhatsApp and TextSecure paths remain all local. Using Threema only adds Europe and WeChat only adds Asia to the paths.

Oceania × South America: The categorization for Oceania and South America is similar to Oceania × North America but incremented by one for each category. The network, WhatsApp and TextSecure paths are all *local+1*, adding North America to the paths. Using Threema adds Europe and North America and using WeChat adds Asia and North America to the paths.

North America $\times$ South America: The last region pair, the communication between the Americas leads to four communications. The network paths and the application paths of WhatsApp and TextSecure are all local. All application paths of Threema are *local+1*, adding Europe to the set of traversed regions and WeChat is *local+2* adding Asia and Oceania.

8.3 Magnitude of Deviation from the Locality

Higher instances of deviation from the locality increase the amount of possible attack points on communications. The definition of locality and the partition of the world into six regions could possibly have resulted in five more regions for inner-region communications and in four more regions for inter-region communications. Analyzing the locality of all communication paths showed that the highest measured deviation is two additional regions. The majority of commucations has deviations of a magnitude of one, especially for communications within Europe and North America. The traffic from South America, Oceania and Asia is often routed via additional regions resulting in a magnitude of two. The main transit country is the U.S., often used to connect the other regions.

8.4 Evaluation of Influence from the Messenger

The locality analysis shows that the assumption of a user as defined in this thesis is often false and a large number of paths are non-local. But there remains the question whether the resulting deviation can be traced back to a direct influence from the mobile messaging applications, using their service infrastructure, or whether the deviation is only based on exterior influences, for example network topologies or routing policies. These exterior influences are measured and analyzed through the network paths, and reflected in the non-locality of these paths. Therefore, the application paths are compared to the network paths in Table 8.1 and 8.2.

To illustrate this controversy, the inner-region communication of Asia is used exemplarily. Considering the network paths, five are categorized already as *local+2* since Israel is routed via Europe and North America to other Asian countries. Looking at the application paths of all messengers in Asia, plenty of paths are categorized as *local+2* as well. The increased number of paths categorized as *local+2* shows, that the mobile messengers have a direct influence on traffic locality, but for five communications of each messenger a deviation already existed in the network path and is therefore not solely caused by the use of the messenger. This means that an influence from the messenger is a given, but influences, independent from the mobile messaging applications affect the locality of the communications. The magnitude of the deviations caused by

		Network/	Local		Local+1		Local+2	
Region	#Paths	Application	#	%	#	%	#	%
	105	Network	57	54%	48	46%	0	0%
Europe		WhatsApp	0	0%	105	100%	0	0%
×		TextSecure	0	0%	105	100%	0	0%
Asia		**Threema**	34	32%	71	68%	0	0%
		WeChat	5	5%	100	95%	0	0%
	30	Network	0	0%	29	97%	1	3%
Europe		**WhatsApp**	0	0%	30	100%	0	0%
×		**TextSecure**	0	0%	30	100%	0	0%
Oceania		**Threema**	0	0%	30	100%	0	0%
		WeChat	0	0%	1	3%	29	97%
	30	Network	30	100%	0	0%	0	0%
Europe		**WhatsApp**	30	100%	0	0%	0	0%
×		**TextSecure**	30	100%	0	0%	0	0%
North America		**Threema**	30	100%	0	0%	0	0%
		WeChat	0	0%	1	3%	29	97%
	30	Network	16	53%	14	47%	0	0%
Europe		**WhatsApp**	0	0%	30	100%	0	0%
×		**TextSecure**	0	0%	30	100%	0	0%
South America		**Threema**	0	0%	30	100%	0	0%
		WeChat	0	0%	0	0%	30	100%
	14	Network	7	50%	5	36%	2	14%
Asia		WhatsApp	0	0%	12	86%	2	14%
×		TextSecure	0	0%	12	86%	2	14%
Oceania		Threema	0	0%	0	0%	14	100%
	13[1]	**WeChat**	5	38%	7	54%	1	8%
	14	Network	12	86%	2	14%	0	0%
Asia		**WhatsApp**	12	86%	2	14%	0	0%
×		**TextSecure**	12	86%	2	14%	0	0%
North America		Threema	0	0%	14	100%	0	0%
		WeChat	0	0%	12	86%	2	14%
	14	Network	0	0%	12	86%	2	14%
Asia		**WhatsApp**	0	0%	12	86%	2	14%
×		**TextSecure**	0	0%	12	86%	2	14%
South America		Threema	0	0%	0	0%	14	100%
		WeChat	0	0%	12	86%	2	14%
	4	Network	4	100%	0	0%	0	0%
Oceania		**WhatsApp**	4	100%	0	0%	0	0%
×		**TextSecure**	4	100%	0	0%	0	0%
North America		Threema	0	0%	4	100%	0	0%
		WeChat	0	0%	4	100%	0	0%
	4	Network	0	0%	4	100%	0	0%
Oceania		**WhatsApp**	0	0%	4	100%	0	0%
×		**TextSecure**	0	0%	4	100%	0	0%
South America		Threema	0	0%	0	0%	4	100%
		WeChat	0	0%	0	0%	4	100%
	4	Network	4	100%	0	0%	0	0%
North America		**WhatsApp**	4	100%	0	0%	0	0%
×		**TextSecure**	4	100%	0	0%	0	0%
South America		Threema	0	0%	4	100%	0	0%
		WeChat	0	0%	0	0%	4	100%

1: The path between Israel and New Zealand is missing
Green highlighted: Recommended application based on the locality analysis

Table 8.2: Locality of Messages for Communication Partners in different Regions

the mobile messaging service infrastructure is additionally influenced by the standard network behavior. The additional influence from independent sources does not reduce the security and privacy issues for a user and the communication path but reduces the direct impact of the messengers and their infrastructure on the locality of messages and has to be kept in mind when judging mobile messengers or other network locality measurements.

The influence of the service infrastructure can lead to three possible changes of application paths compared to network paths. The locality of application paths can be better, even or worse than the locality of network paths. Table 8.3 shows the absolute number and percentage of the three possible changes for each region combination and messenger.
First, the locality of the application path can be better than the locality of the network path. Table 8.3 shows that only one communication path is better in terms of the locality. One network path for the cross-region communication between Europe and Oceania is categorized as *local+2* in Table 8.2 due to hops in Asia and North America for the communication between Portugal and New Zealand. Using WhatsApp, TextSecure and Threema positively influences the locality in this case, as they have no more paths categorized as *local+2* but only *local+1* paths, eliminating the hops in Asia.
Second, the locality of the application path can be equal to the locality of the network path. This is mostly the case when the service infrastructure of a mobile messaging application is in the same region as one of the communication partners, for example all Threema paths in the communications within Europe or all WhatsApp and TextSecure paths in the communications within North America. However, the communications between Europe and Oceania show that equal locality results can be seen as well if the service infrastructure of a mobile messaging application is not in the same region as one of the communication partners. The WhatsApp and TextSecure infrastructures are located in the U.S.—not in Oceania or Europe—but because the network path is already routed over North America for most of the country combinations, the locality of the application paths and network paths is even.
Third, the locality of the application path could be worse than the locality of the network path. Sending the messages via the service infrastructure often leads to deteriorations, for example the communications inside Europe, where the network path would be local. Threema, with its infrastructure in Switzerland does not affect the locality. However WhatsApp and TextSecure messages need to be routed to their infrastructure in the U.S. causing a deviation from the defined locality and the network path. WeChat traffic needs to be routed to China in Asia causing a deviation as well. The inner-country communication of Switzerland shows that a direct path to Asia is possible. However for the remaining European countries, the result is intensified by routing policies, resulting in paths via North America for the majority of the messages.

			Better		Even		Worse	
Region	#Paths	Application	#	%	#	%	#	%
Europe	120	WhatsApp/TextSecure	0	0%	0	0%	120	100%
×		Threema	0	0%	120	100%	0	0%
Europe		WeChat	0	0%	0	0%	120	100%
Asia	28	WhatsApp/TextSecure	0	0%	6	22%	22	78%
×		Threema	0	0%	6	22%	22	78%
Asia		WeChat	0	0%	20	71%	8	29%
Oceania	3	WhatsApp/TextSecure	0	0%	0	0%	3	100%
×		Threema	0	0%	0	0%	3	100%
Oceania		WeChat	0	0%	0	0%	3	100%
North America	3	WhatsApp/TextSecure	0	0%	3	100%	0	0%
×		Threema	0	0%	0	0%	3	100%
North America		WeChat	0	0%	0	0%	3	100%
South America	3	WhatsApp/TextSecure	0	0%	1	33%	2	67%
×		Threema	0	0%	0	0%	3	100%
South America		WeChat	0	0%	0	0%	3	100%
Europe	105	WhatsApp/TextSecure	0	0%	48	46%	57	54%
×		Threema	0	0%	82	78%	23	22%
Asia		WeChat	0	0%	53	50%	52	50%
Europe	30	WhatsApp/TextSecure	1	3%	29	97%	0	0%
×		Threema	1	3%	29	97%	0	0%
Oceania		WeChat	0	0%	2	7%	28	93%
Europe	30	WhatsApp/TextSecure	0	0%	30	100%	0	0%
×		Threema	0	0%	30	100%	0	0%
North America		WeChat	0	0%	0	0%	30	100%
Europe	30	WhatsApp/TextSecure	0	0%	14	47%	16	53%
×		Threema	0	0%	14	47%	16	53%
South America		WeChat	0	0%	0	0%	30	100%
Asia	14	WhatsApp/TextSecure	0	0%	7	50%	7	50%
×		Threema	0	0%	2	14%	12	86%
Oceania	13[1]	WeChat	0	0%	13	100%	0	0%
Asia	14	WhatsApp/TextSecure	0	0%	14	100%	0	0%
×		Threema	0	0%	2	14%	12	86%
North America		WeChat	0	0%	2	14%	12	86%
Asia	14	WhatsApp/TextSecure	0	0%	14	100%	0	0%
×		Threema	0	0%	2	14%	12	86%
South America		WeChat	0	0%	14	100%	0	0%
Oceania	4	WhatsApp/TextSecure	0	0%	4	100%	0	0%
×		Threema	0	0%	0	0%	4	100%
North America		WeChat	0	0%	0	0%	4	100%
Oceania	4	WhatsApp/TextSecure	0	0%	4	100%	0	0%
×		Threema	0	0%	0	0%	4	100%
South America		WeChat	0	0%	0	0%	4	100%
North America	4	WhatsApp/TextSecure	0	0%	4	100%	0	0%
×		Threema	0	0%	0	0%	4	100%
South America		WeChat	0	0%	0	0%	4	100%

1: The path between Israel and New Zealand is missing

Table 8.3: Changes of the Locality of Application Paths Compared to Network Paths

8.5 Evaluation of Differences between the Messengers

The main differences between the messengers in connection with the thesis' objectives are the locations of the infrastructures of the mobile messaging applications with the resulting locality deviations. All other differences do not seem to influence the locality behavior of the messages. Nothing exceptional, besides the DNS-over-HTTP mechanism, could be discovered and this mechanism seems to work in a similar way to DNS. Only WeChat incorporates a differentiation resolving domain names over DNS and DNS-over-HTTP between China and the rest of the countries used in this thesis. The WhatsApp, TextSecure and Threema domain names globally resolve to the same set of IPs and the IPs are used without observable patterns based on the location. Further differentiations of the sender or receiver country, implemented in the applications do not seem to exist. The infrastructures are centralized in one country for each messenger as shown in Chapter 7. Noteworthy are the infrastructures of WhatsApp and TextSecure. They are both located in the U.S. but hosted by different organizations. Nonetheless they lead to the exact same results analyzing the locality with the granularity of this thesis. Comparing the absolute values, Threema with its servers inside Europe leads to the best results. This value is however not that significant, because of the fact that Europe was represented by 15 PlanetLab nodes in the measurement, North America on the other hand by only two nodes. Therefore the different messengers are compared based on the percentage values.
Recommendations which mobile messaging applications to use in which region are highlighted green in Table 8.1 and 8.2, e.g. when someone in Argentina wants to communicate with someone in New Zealand, he should tend towards WhatsApp and TextSecure, considering only the locality of the communication paths.
Table 8.4 shows the locality categorization of the mobile messaging applications summed up in a global view. WhatsApp and TextSecure lead to the best results. A third of the communications even stays local and only three percent are routed via two additional regions. The paths for Threema are still local for 16.5% but already 38% are *local+2*. WeChat results in the worst outcome. Only around six percent of the communications stay local and even 61.4% of the communications are routed through more than one additional region. Even countries in Asia, like Singapore do not route to WeChat directly but through an Internet exchange point in the U.S.. The results for each messenger are the best for communications in their regions. The application paths of Threema all stay local in Europe, WhatsApp and TextSecure stay local in North America and WeChat has the best results in Asia, with at least 50% local traffic. The grading is reflected in the changes of the locality from the application paths compared to the network paths in a global view as well. WhatsApp and TextSecure result in better locality results for 0.2% of the paths as well as Threema but 66.3% of the application path categorizations are even to the network path categorizations and only 33.5% are worse. The application paths of Threema are even for 32.4% in the global view and 67.4% are worse than the

Application	Local	Local+1	Local+2
Network	65.8%	30.9%	3.3%
WhatsApp	32.4%	64.1%	3.5%
TextSecure	32.4%	64.1%	3.5%
Threema	15.5%	45.5%	39.0%
WeChat	6.2%	32.4%	61.4%

Table 8.4: Locality of Messages in a Global View

Application	Better	Even	Worse
WhatsApp	0.2%	66.3%	33.5%
TextSecure	0.2%	66.3%	33.5%
Threema	0.2%	32.4%	67.4%
WeChat	0%	22.8%	77.2%

Table 8.5: Changes of the Locality in a Global View

network paths. WeChat leads to the worst results again, with zero better paths, only 22.8% equal application paths compared to the network paths and 77.2% of the WeChat application paths are worse than the network paths.

8.6 Summary

In this chapter, the locality of mobile messages was analyzed and the results evaluated. It was shown that plenty of the analyzed communication paths are non-local. Every messenger breaks the locality even considering network topologies and routing policies. WhatsApp and TextSecure lead to the best results directly compared to the other two messengers. The magnitude of the deviation lies between one and two but not above. No node in Africa was accessible for the measurement and Africa was also not traversed by a single path. Furthermore the chapter provides recommendations which mobile messaging application to use for the best locality results based on the regions of the sender and receiver and compares all mobile messaging applications in a global view if a user wants to choose only one application for global communications.

Chapter 9

Experiences and Challenges

9.1 Interval Based Approach of the Mobile Phone Controller

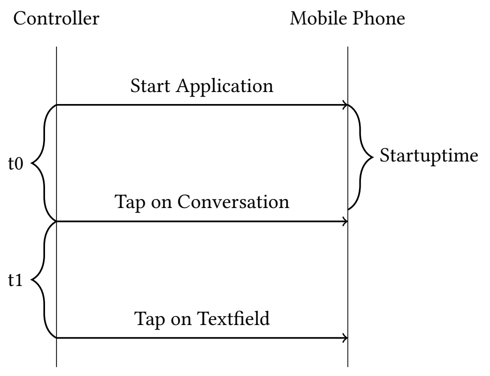
The implementation of the framework by now comes with one major disadvantage. The approach to control the mobile phones and the different applications is interval based. This means that an action, like starting an application or sending a message, can be executed but afterwards a static timeout is set to make sure the result has happened. The following example illustrates a problem with this approach.

The two diagrams in Figure 9.1 show the interaction between the controller and a mobile phone to start an application and enter a conversation based on a timeout. The controller sends the command to start the application to the mobile phone and afterwards it waits a fixed time t0 until it sends the next command to tap on the conversation. Figure 9.1a shows a scenario where no problem occurs. The startuptime of the application is lower than the timeout t0 and the tap is possible. Figure 9.1b shows a scenario where the startuptime is higher than the timeout and the tap will not lead to any or to a wrong result. The next tap on the textfield will fail as well.

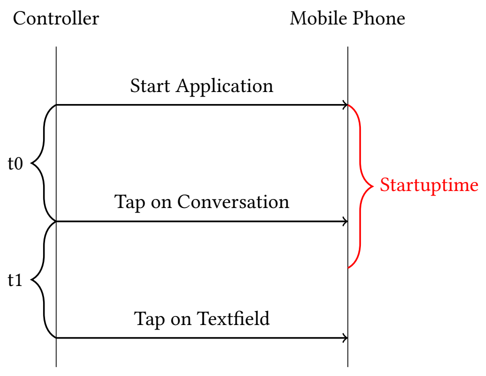
Unfortunately this startup and many other activities are influenced by many different factors like the quality of the network connection or the momentary performance of the mobile phones. However the likelihood of the problem can be minimized by higher timeouts. This comes with the disadvantage of longer measurement times.

9.2 DNS Requests Despite Firewalling with iptables

The use of iptables should block as much traffic as possible. During the analysis of the dataset, an unforeseen case occurred. It was expected that there would be no traffic in relation to mobile messaging applications, that were not active in an experiment.



(a) Working Starting Process



(b) Failing Starting Process

Figure 9.1: Time Based Diagram of the Application Starting Process

However it turned out that DNS resolutions that could be definitely dedicated to non-active applications were still executed. Looking at the DNS mechanism of Android shows that DNS resolutions are done by the *libc* C library [46]. Applications use this library to send DNS requests. They are sent tagged with the root UID responsible for this library and not tagged with the application UID. Therefore they pass the iptables firewalls and are visible in the dataset. This is still the only traffic related to non-active mobile messaging applications.

9.3 External Influences on the Mobile Phones and Messengers

The mobile phones and mobile messaging applications are set up in a standard mode and therefore integrated in the normal environment of the services. Most of the analyzed messengers are preset to accept all incoming communications even though the counterpart is unknown. It is sufficient for someone to send a message to the account of the messenger and for WhatsApp, WeChat and TextSecure the account identifier is primary the mobile phone number. Therefore if someone sends a message to this number during a measurement intentionally or by mistake, the measurement is profoundly influenced. This occurred once during the measurements for this thesis: someone sent a WhatsApp message to one of the measurement mobile phones by mistake. In WhatsApp, new conversations shift to the top and due to the static configuration of our controller—tapping always on the uppermost conversation—the messages from this mobile phone were sent to the unknown conversation partner for all further country pairs of the measurement. All failed country pairs had to be repeated afterwards.

Chapter 10

Conclusion

In this thesis the locality of mobile messages was analyzed. It is a security and privacy issue, that effects all mobile messaging application users. The thesis sought to analyze the influence of mobile messaging applications on the traffic delivering the messages. The analysis was made with a global perspective based on messages exchanged and the reproduction of their behavior with path measurements. The thesis aimed to answer the following questions based on this data:

1. Does mobile messaging influence the locality of network traffic?
2. Do the results differ based on the regions of the communication partners?
3. How strong is the influence on communication paths?
4. Does the influence differ between the various mobile messaging applications?

The results of the thesis show that an influence of the mobile messengers on traffic locality is given. The thesis statement holds true for all tested messengers. The influence is caused by the centralized infrastructure the messages have to pass. All messengers only provide servers located in one country, even WhatsApp with 900 million users worldwide. Sending messages in countries in the same region as the infrastructure tend to result in a better locality. Choosing a messenger based on the proximity to this infrastructure would therefore improve locality of the messages but restrict the freedom of choice for users. The deviation from a local path lies mostly between an additional one or two regions, based on the regions of sender and receiver of messages and additional reinforcements through network topologies and routing policies. WhatsApp and TextSecure resulted in the best locality values. The combination of their centralized infrastructure in the U.S. and the strong linkage of North America to other regions positively complement each other. Nonetheless, two thirds of the communications are non-local and endanger the users. A possible solution for the companies providing mobile messaging applications to protect messages of users would be a decentralized infrastructure, distributed globally and therefore located closer to the users.

A limitation of large scale locality analysis is the dependency on geolocation databases, which are only accurate to a certain percentage. They are still the best way to map many IPs to their geolocation. To not only rely on the database, several entries were corrected to increase the accuracy on this level of granularity. Furthermore the availability of remote nodes in distributed testbeds varies uncontrollably. PlanetLab provided accessible nodes in 28 countries for over two weeks, but till the end no PlanetLab node was accessible in Israel, leading to a gap in the dataset.

10.1 Future Work

The analysis covered four popular messengers and their behavior on the granularity of regions. Future work could improve this granularity and gain more information on traffic locality in relation to individual countries. The analysis could improve the insights into unexpected behavior for example the routing from Israel to other Asian countries or the routing from Singapore via the U.S. to WeChat in China. The dataset created provides a basis for a more detailed analysis and a large number of communication paths. The main focus to further analyze the dataset has to be the improvement of geolocation databases to make sure all IPs of the network paths and application paths are correctly mapped to the respective countries.
In addition, the dataset can be used to analyze the messengers looking at other aspects than the locality. The network traces contain the complete traffic of the mobile messaging applications from start to termination for each measurement. The results about the DNS-over-HTTP mechanism from WeChat show the potential of the dataset besides the locality analysis. It holds further communication patterns of the messengers and information about network protocols and security mechanisms. It provides a basis for further areas of research, e.g. certificate analysis of the four messengers or detailed protocol analysis in a global perspective similar to [12–14].
Besides the research and results in this thesis, the locality analysis offers further interesting directions, for example other mobile messaging applications or application versions for different operating systems could be analyzed in further works to enlarge the insights on different approaches, e.g. the P2P applications Bleep or TextSecure with the CyanogenMod version and the iOS version Signal. Different functionalities and services of the operating system, for example notification systems, could be used by mobile messaging applications. Therefore the locality analysis can lead to different results based on the operating system the applications are installed in.
Furthermore, the impact of non-local traffic on the security and privacy of users' data is not bound to mobile messaging applications but affects all applications sending users' data through the Internet.

The implemented framework provides a functional basis for this thesis, but the main

principles and design decisions make it usable for further research as well. Possible improvements and features of the framework can be for example:
Integration of further operating systems: The mobile phone controller is designed as independent from the network and measurement components and can be extended to control iOS or Windows Phone as well. The other components, to establish network connections, tunnel the traffic and intercept and analyze the traffic, can remain unchanged. Mobile phone controllers for other mobile phone operating systems cause the same challenge of controlling features—designed for a use over a GUI—over a command-line tool. Tools similar to the ADB are necessary for a successful integration.
Expansion of communication partners: By now, two mobile phones are integrated in the framework, but as it is built with commodity hardware, an expansion is imaginable. All components are built with possible extension in mind and the automation ensures that higher number of measurements can be realizable. The expansion can be used to gain insight into further functionalities of applications, e.g. group chats. Furthermore the expansion can be used to parallelize multiple conversations to increase the message throughput and decrease the measurement time.
Upgrade from the interval based to an event based approach: An event based approach could possibly solve the drawbacks of the current interval based approach. The main challenge of an event based approach is to implement a controller that has access to necessary information about the occurrence of events at all components of the framework. Especially events in the mobile phones take place with non-obvious feedback for the controller, for example the arrival of mobile messages on the phone. There is no activity implemented into the mobile messaging applications that could be used via the ADB to check if messages have arrived on the mobile phone. Android logs or the notification mechanisms of the operating systems could be a starting point to implement workarounds. Advantages of an event based approach are: variable definitions of timeouts to decrease the overall measurement time and more accurate error handling and logging to reduce the necessity of watching screen records as a controlling mechanism.
Integration of DNS Servers: Another possible feature could be the integration of DNS servers in the framework. The mobile phone would use those servers to resolve domain names and each server resolves the domain name over the DNS servers of the tunnel remote nodes recursively. This would not change the basic functionality of the resolution process and would lead to the same results, but adds the DNS servers as additional entities to the framework. The advantages are, that these entities are fully manageable and logging features increase the information about DNS resolutions in addition to the results gained from the traffic traces.
SSL Man in the Middle Proxies: Many mobile applications do not implement end-to-end but only on-transit encryption with SSL. The routers of the framework that already route the complete network traffic between the mobile phones and the Internet could be extended as *man in the middle SSL proxies*. This would allow further security analysis

besides the locality analysis. Secure implementations of the encryption and certificate mechanisms could be inspected in an automated way and a global perspective without implementing a new framework.

Appendix A

Additional Information

A.1 Application Parameters on the Motorola Devices

The following values are necessary ADB parameters for the mobile phone controller. The application names and activities are correlated to the applications. In addition the coordinates are correlated to the configuration of the used Motorola devices.

Application	Name	Start Activity
WhatsApp	com.whatsapp	~^{1}/.Main
TextSecure	org.thoughtcrime.securesms	~^{1}/.RoutingActivity
Threema	ch.threema.app	~^{1}/.activities.MainActivity
WeChat	com.tencent.mm	~^{1}/.ui.LauncherUI

1: Name of the application

Table A.1: Names and Start Activities of the Mobile Messaging Applications

Action	WhatsApp	TextSecure	Threema	WeChat
Start Application	5s	2s	3s	25s
Enter Conversation	2s			
Time until Messages Receiving	10s			20-600s

Table A.2: Timeouts after all Application Actions

	WhatsApp		TextSecure		Threema		WeChat	
Action	X	Y	X	Y	X	Y	X	Y
Conversation	200	200	265	174	237	226	246	155
Textfield	230	847	233	846	250	852	200	854
Send Button	511	445	505	455	520	441	513	439

Table A.3: Coordinates for Interactions with the Messengers

A.2 Filter List of Domain Names and Additional IPs

The following list contains all domain names that were filtered out during the post-processing step described in Section 7.1. Network connections and path measurements to all IPs resolved from these domain names were filtered out of the dataset before the analysis. Three IPs without relations to domains were additionally filtered.

- 2.android.pool.ntp.org.
- android.clients.google.com.
- android.googleapis.com.
- argo.svcmot.com.
- clients3.google.com.
- clients4.google.com.
- mail.google.com.
- mclients.googleapis.com.
- mtalk.google.com.
- play.googleapis.com.
- safebrowsing.google.com.
- www.googleadservices.com.
- www.googleapis.com.
- xtra1.gpsonextra.net.
- xtra2.gpsonextra.net.
- xtra3.gpsonextra.net.
- moto-cds.appspot.com.
- www.google.com.
- csi.gstatic.com.
- xtrapath1.izatcloud.net.
- xtrapath2.izatcloud.net.
- api.threema.ch.

103.7.31.157: This IP is only used once during a WeChat measurement. Inspecting the according trace shows that it is not used for message passing

120.198.189.100: This IP appears only two times in traces from WeChat measurements. Again, inspecting the according traces shows that it is not used for message passing.

139.80.64.1: This is the nameserver of the New Zealand PlanetLab node. The integrated filter in the framework is configured to filter UDP DNS requests. However the mobile phones sens TCP DNS requests to this nameserver. Therefore this IP can be excluded from message passing and is filtered.

A.3 List of Changes in the IP2Location Database

The initial mapping is done with the IP2Location[1] IP-Country database from 9 October 2015. The following table holds a list of all mappings between IP addresses and geolocations that were changed during the thesis with the in Section 7.3 described method.

[1]http://www.ip2location.com/

	Geolocation	
IP Address	Old	New
4.68.111.178	US	NL
4.69.137.81	US	CH
61.8.59.37	AU	JP
61.8.59.38	AU	JP
62.115.140.203	US	FR
62.216.145.74	GB	HK
63.216.156.106	US	HK
63.216.156.86	US	HK
63.218.230.73	US	DE
63.223.29.10	US	HK
63.223.29.14	US	HK
63.223.29.18	US	HK
77.19.128.210	NO	CH
80.77.0.181	EG	HK
80.77.0.182	EG	HK
82.197.168.121	HR	CH
85.95.25.105	US	HK
85.95.25.90	US	HK
85.95.26.102	AE	HK
89.221.41.175	IT	US
89.221.41.177	IT	US
109.105.97.73	US	DE
129.250.6.162	JP	US
149.3.181.101	IT	US
149.3.181.65	IT	US
149.3.181.87	IT	US
149.3.181.91	IT	US
149.3.181.95	IT	US
149.3.181.97	IT	US
149.3.181.99	IT	US
150.99.188.201	JP	US
176.32.125.136	IE	US
176.32.125.138	IE	US
176.32.125.140	IE	US
176.32.125.143	IE	US
176.32.125.145	IE	US
176.32.125.147	IE	US
176.32.125.148	IE	US
176.32.125.152	IE	US
176.32.125.155	IE	US
176.32.125.159	IE	US
176.32.125.160	IE	US
176.32.125.162	IE	US
176.32.125.164	IE	US
176.32.125.167	IE	US
176.32.125.169	IE	US
176.32.125.171	IE	US
176.32.125.172	IE	US
176.32.125.174	IE	US
176.32.125.176	IE	US
176.32.125.179	IE	US
176.32.125.181	IE	US
176.32.125.183	IE	US
176.32.125.186	IE	US
176.32.125.191	IE	US
176.32.125.193	IE	US
176.32.125.196	IE	US
176.32.125.198	IE	US
176.32.125.200	IE	US
176.32.125.203	IE	US
176.32.125.207	IE	US
176.32.125.208	IE	US
176.32.125.210	IE	US
176.32.125.212	IE	US
176.32.125.215	IE	US
176.32.125.217	IE	US
176.32.125.219	IE	US
176.32.125.220	IE	US
176.32.125.224	IE	US
176.32.125.227	IE	US
176.32.125.229	IE	US
176.32.125.231	IE	US
178.236.3.24	IE	US
178.236.3.96	IE	US
178.236.3.98	IE	US
185.70.203.37	IT	AR
185.70.203.61	IT	AR
185.70.203.63	IT	AR
195.22.199.177	IT	US
195.22.219.145	IT	BR
195.22.219.147	IT	BR
195.22.219.171	IT	BR
195.22.219.175	IT	BR
195.22.219.177	IT	BR
195.22.219.179	IT	BR
198.32.141.11	US	SG
198.32.141.146	US	SG
202.147.40.114	AU	US
202.147.58.146	AU	US
202.147.58.147	AU	GB
202.147.58.150	AU	US
202.147.58.151	AU	GB
202.84.140.234	AU	SG
202.84.143.253	AU	HK
202.84.221.25	AU	SG
202.84.221.26	AU	KR
202.84.249.161	AU	HK
203.233.2.234	KR	US
212.162.10.81	US	DE
212.162.10.82	US	DE
213.242.73.74	US	CH
217.239.45.6	DE	US
217.6.51.246	DE	US
-	-	-
-	-	-

Table A.4: Changes in the IP2Location Database

Bibliography

[1] J. Koum, "Whatsapp - now serving 900,000,000 monthly active users," Website, September 2015, accessed: Sep. 23, 2015. [Online]. Available: https://www.facebook.com/jan.koum/posts/10153580960970011

[2] ——, "Whatsapp - 700 million monthly avtive users and 30 billion messages per day," Website, January 2015, accessed: Oct. 26, 2015. [Online]. Available: https://www.facebook.com/jan.koum/posts/10152994719980011

[3] G. Greenwald and E. MacAskill, "NSA Prism program taps in to user data of Apple, Google and others," Website, June 2013, accessed: Oct. 26, 2015. [Online]. Available: http://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data

[4] E. MacAskill, J. Borger, N. Hopkins, N. Davies, and J. Ball, "GCHQ taps fibre-optic cables for secret access to world's communications," Website, June 2013, accessed: Oct. 26, 2015. [Online]. Available: http://www.theguardian.com/uk/2013/jun/21/gchq-cables-secret-world-communications-nsa

[5] A. Meister, "How the German Foreign Intelligence Agency BND tapped the Internet Exchange Point DE-CIX in Frankfurt, since 2009," Website, March 2015, accessed: Oct. 26, 2015. [Online]. Available: https://netzpolitik.org/2015/how-the-german-foreign-intelligence-agency-bnd-tapped-the-internet-exchange-point-de-cix-in-frankfurt-since-2009/

[6] United Nations Statistic Division, "Composition of macro geographical (continental) regions, geographical sub-regions, and selected economic and other groupings," Website, October 2013, accessed: Oct. 26, 2015. [Online]. Available: http://unstats.un.org/unsd/methods/m49/m49regin.htm

[7] Q. Scheitle, M. Wachs, J. Zirngibl, and G. Carle, "Analyzing Locality of Mobile Messaging Traffic using the MATAdOR Framework," Currently under submission, 2015.

[8] PlanetLab, "PlanetLab: An open platform for developing, deploying, and accessing planetary-scale services," Website, accessed: Oct. 06, 2015. [Online]. Available: http://www.planet-lab.org/

[9] J. Snijders, "NLNOG Ring," Website, accessed: Oct. 06, 2015. [Online]. Available: https://ring.nlnog.net/

[10] VPN Gate, "VPN Gate: Public VPN Relay Servers," Website, accessed: Oct. 06, 2015. [Online]. Available: http://www.vpngate.net/en/

[11] RIPE NCC, "RIPE Atlas," Website, accessed: Nov. 03, 2015. [Online]. Available: https://atlas.ripe.net/

[12] P. Fiadino, M. Schiavone, and P. Casas, "Vivisecting WhatsApp in Cellular Networks: Servers, Flows, and Quality of Experience," in *Traffic Monitoring and Analysis.* Springer, 2015, pp. 49–63.

[13] P. Fiadino, P. Casas, M. Schiavone, and A. D'Alconzo, "Online Social Networks anatomy: On the analysis of Facebook and WhatsApp in cellular networks," in *IFIP Networking Conference (IFIP Networking), 2015*, May 2015, pp. 1–9.

[14] Q. Huang, P. P. Lee, C. He, J. Qian, and C. He, "Fine-Grained Dissection of WeChat in Cellular Networks," in *Quality of Service (IWQoS), 2015 IEEE/ACM International Symposium on*, June 2015.

[15] Electronic Frontier Foundation, "Secure Messaging Scorecard: Which apps and tools actually keep your messages safe?" Website, accessed: Sep. 17, 2015. [Online]. Available: https://www.eff.org/secure-messaging-scorecard

[16] S. Schrittwieser, P. Frühwirt, P. Kieseberg, M. Leithner, M. Mulazzani, M. Huber, and E. R. Weippl, "Guess Who's Texting You? Evaluating the Security of Smartphone Messaging Applications." in *Network and Distributed System Security Symposium*, 2012.

[17] R. Mueller, S. Schrittwieser, P. Fruehwirt, P. Kieseberg, and E. Weippl, "What's new with WhatsApp & Co.? Revisiting the Security of Smartphone Messaging Applications," in *Proceedings of the 16th International Conference on Information Integration and Web-based Applications & Services*, 2014, pp. 142–151.

[18] S. E. Coull and K. P. Dyer, "Traffic Analysis of Encrypted Messaging Services: Apple iMessage and Beyond," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 5–11, 2014.

[19] D. Ferreira, V. Kostakos, A. R. Beresford, J. Lindqvist, and A. K. Dey, "Securacy: An Empirical Investigation of Android Applications' Network Usage, Privacy and Security," in *Proceedings of the 8th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2015.

[20] M. Marlinspike, "Textsecure, now with 10 million more users," Website, December 2013, accessed: Oct. 19, 2015. [Online]. Available: https://whispersystems.org/blog/cyanogen-integration/

[21] WhatsApp, “WhatsApp Blog Post: 500.000.000,” Website, April 2014, accessed: Sep. 17, 2015. [Online]. Available: http://blog.whatsapp.com/613/500000000

[22] J. Koum, “Whatsapp - now serving 800,000,000 monthly active users,” Website, April 2015, accessed: Sep. 23, 2015. [Online]. Available: https://www.facebook.com/jan.koum/posts/10153230480220011

[23] Tencent, “TENCENT ANNOUNCES 2015 FIRST QUARTER RESULTS,” Website, May 2015, accessed: Sep. 17, 2015. [Online]. Available: http://www.tencent.com/en-us/content/at/2015/attachments/20150513.pdf

[24] Threema, “If you value security and privacy,” Website, September 2014, accessed: Sep. 17, 2015. [Online]. Available: https://threema.ch/press-files/1_press_info/Press-Info_Threema_EN.pdf

[25] Lovina McMurchy, “Skype Connection Hub Ads Provide Increased Scale for Marketers,” Website, December 2014, accessed: Sep. 17, 2015. [Online]. Available: http://advertising.microsoft.com/en/blog/29331/skype-connection-hub-ads-provide-increased-scale-for-marketers

[26] Facebook, “Messenger at F8,” Website, March 2015, accessed: Sep. 17, 2015. [Online]. Available: http://newsroom.fb.com/news/2015/03/messenger-at-f8/

[27] The Telegram Team, “Telegram Reaches 1 Billion Daily Messages,” Website, December 2014, accessed: Sep. 17, 2015. [Online]. Available: https://telegram.org/blog/billion

[28] Deutsche Post DHL Group, “Deutsche Post SIMSme messenger app tops one million mark,” Website, March 2015, accessed: Sep. 17, 2015. [Online]. Available: http://www.dpdhl.com/en/media_relations/press_releases/2015/deutsche_post_simsme_messenger_app_tops_one_million_mark.html

[29] Rakuten, Inc., “FY2015 First Quarter Consolodated Financial Results,” Website, March 2015, accessed: Sep. 17, 2015. [Online]. Available: http://linecorp.com/en/pr/news/en/2015/1043

[30] Tango, “200 Million Members!” Website, March 2014, accessed: Sep. 17, 2015. [Online]. Available: http://www.tango.me/blog/200-million-members

[31] LINE Corporation, “LINE Corporation Announces 2015 Q2 Earnings,” Website, Juli 2015, accessed: Sep. 17, 2015. [Online]. Available: http://linecorp.com/en/pr/news/en/2015/1043

[32] Open Whisper System: Support Center, “Servers Location,” Website, January 2013, accessed: Sep. 17, 2015. [Online]. Available: http://support.whispersystems.org/customer/portal/questions/6621379-servers-location

[33] F. Fadaie, "How Does Bleep Work?" Website, September 2014, accessed: Oct. 15, 2015. [Online]. Available: http://engineering.bittorrent.com/2014/09/17/how-does-bleep-work/

[34] ——, "Bleep now supports asynchronous offline messaging," Website, December 2014, accessed: Oct. 15, 2015. [Online]. Available: http://blog.bittorrent.com/2014/12/22/bleep-now-supports-asynchronous-offline-messaging/

[35] Android, "Data Usage Tags Explained," Website, accessed: Oct. 12, 2015. [Online]. Available: https://source.android.com/devices/tech/datausage/tags-explained.html

[36] Android Developers, "Android Debug Bridge," Website, accessed: Sep. 17, 2015. [Online]. Available: http://developer.android.com/tools/help/adb.html

[37] ——, "App Manifest," Website, accessed: Oct, 14 2015. [Online]. Available: http://developer.android.com/guide/topics/manifest/manifest-intro.html

[38] ——, "KeyEvent," Website, accessed: Oct. 16, 2015. [Online]. Available: http://developer.android.com/reference/android/view/KeyEvent.html

[39] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear, "Address Allocation for Private Internets," RFC 1918 (Best Current Practice), Internet Engineering Task Force, Feb. 1996, updated by RFC 6761. [Online]. Available: http://www.ietf.org/rfc/rfc1918.txt

[40] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing, "Session Traversal Utilities for NAT (STUN)," RFC 5389 (Proposed Standard), Internet Engineering Task Force, Oct. 2008, updated by RFC 7350. [Online]. Available: http://www.ietf.org/rfc/rfc5389.txt

[41] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones, "SOCKS Protocol Version 5," RFC 1928 (Proposed Standard), Internet Engineering Task Force, Mar. 1996. [Online]. Available: http://www.ietf.org/rfc/rfc1928.txt

[42] A. Loibl, "Measurement Proxy for distributed Network Measurements," Bachelor's Thesis, Technische Universität München, 2015.

[43] D. Mills, J. Martin, J. Burbank, and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification," RFC 5905 (Proposed Standard), Internet Engineering Task Force, Jun. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5905.txt

[44] SoftLayer Technologies®, "SoftLayer® and The Planet® Begin Merged Operations," Website, November 2010, accessed: Oct. 14, 2015. [Online]. Available: http://www.softlayer.com/de/node/1796

[45] B. Huffaker, M. Fomenkov, and k. claffy, “DRoP:DNS-based Router Positioning,” *ACM SIGCOMM Computer Communication Review (CCR)*, vol. 44, no. 3, pp. 6–13, Jul 2014.

[46] Google Source, “Google source: Android 5.0.2,” Website, accessed: Oct. 12, 2015. [Online]. Available: https://android.googlesource.com/platform/bionic/+/android-5.0.2_r1/libc