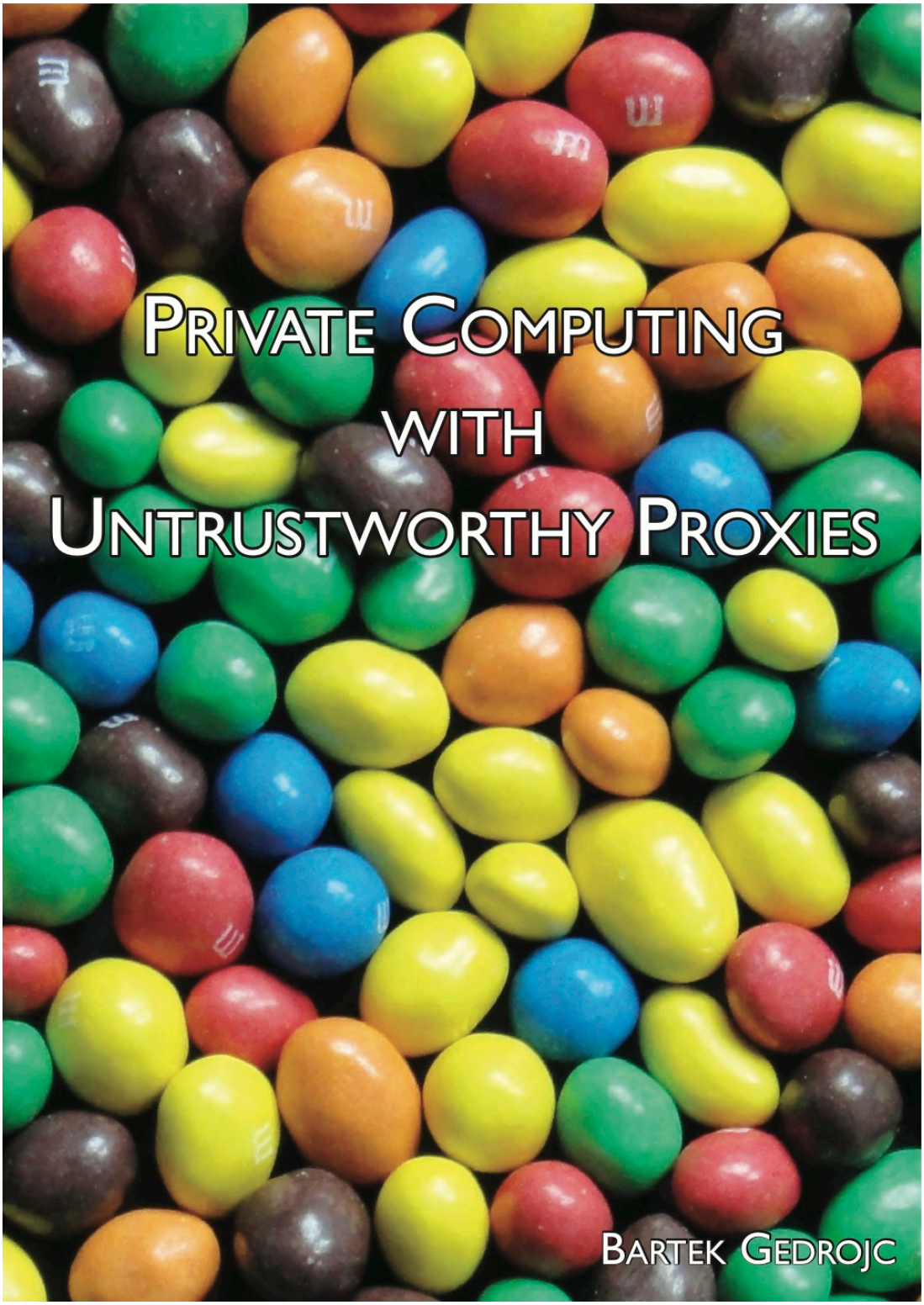

PRIVATE COMPUTING
WITH
UNTRUSTWORTHY PROXIES
BARTEK GEDROJC

Private Computing
with
Untrustworthy Proxies

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op dinsdag 4 oktober 2011 om 15:00 uur
door Bartlomiej GEDROJC
elektrotechnisch ingenieur
geboren te Poznań, Polen.

Dit proefschrift is goedgekeurd door de promotor:
Prof.dr.ir. R.L. Lagendijk

Copromotor:
Dr.ir. J.C.A. van der Lubbe

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof.dr.ir. R.L. Lagendijk,	Technische Universiteit Delft, promotor
Dr.ir. J.C.A. van der Lubbe,	Technische Universiteit Delft, copromotor
Prof.dr.ir. S.M. Heemstra de Groot,	Technische Universiteit Delft
Prof.dr.ir. H.J. Sips,	Technische Universiteit Delft
Prof.dr. Y.-H. Tan,	Technische Universiteit Delft
Prof.dr. S. Etalle,	Technische Universiteit Eindhoven
Prof.dr. P.H. Hartel,	Universiteit Twente

Published by VSSD

ISBN 978-90-6562-282-2

Cover: M&M'S ®

to Q

Preface

The research for this thesis was conducted within the PAW project. PAW (Privacy in an Ambient World) was a Dutch Ministry of Economic Affairs funded project through the IOP GenCom program, managed by Senter, which started October 2003 for a period of 4 years. PAW build on the results obtained during the European Union funded PISA (Privacy Incorporated Software Agent) project. The objective of the PAW project was to develop a privacy protecting architecture that could provide full privacy of the user in an ambient world.

The following parties participated in the PAW project: TNO ITSEF, TNO Telecom, Radboud University Nijmegen, University of Twente and Delft University of Technology.

The task of Delft University of Technology within the PAW project was to answer two questions in the area of Private Computing. The first question was whether it is theoretically possible to protect mobile software against privacy and security attacks while these programs are executed at untrustworthy proxies. The second question was whether the conventional cryptographic algorithms are applicable in this mobile environment. A likely answer to this question was no and then new algorithms had to be developed in order to be able to provide full privacy at these untrustworthy proxies. The result of this work is presented in this thesis.

B. Gedrojc, Rijswijk, September 2011.

Table of Contents

Chapter 1

Introduction

At the end of the 20th century, with the birth of internet, the digital globalization was a fact. The growth of the Internet to a worldwide scale made it possible to acquire a vast amount of information within seconds and gave the possibility to communicate virtually with everybody around the globe like they where standing next to us.

Technically this was made possible by connecting various devices and computer networks with each other, creating a web of connected devices which pass-on information from one device to another. Connecting to the web requires a proxy capable of registering connections to the web in order to facilitate proper communications. Internet Service Providers are proxies capable of providing Internet connection to companies and consumers via wired connections; and within the mobile domain (Mobile Phones, Smartphone, Personal Digital Assistants) the mobile operators are proxies able to provide Internet access to their mobile users via wireless connections.

Once connected to the Internet, the web provides various services e.g. email, access to dictionaries, encyclopedia, search engines. These, usually 'free', services are all provided through proxies e.g. Microsoft, Yahoo, Google. To remain free these proxies use advertisements. In order to provide the most interesting advertisement for the correct person, private information is needed. For example, search engines base their advertisement on the search query. Free webmail shows the advertisement based on the emails in your inbox. Mobile devices using global positioning systems or based on network triangulation provide advertisement based on the location of the user.

Having Internet connection and using services on the internet, we are neither directly connected to the source of information we are querying for directly connected to nor the person we are speaking to. The reason for this is that various proxies are always in between our communication and therefore always have access to our private data. For example, when accessing the mobile phone network, the mobile phone operator always knows our geographic location. When searching the Internet, our Internet Service Provider learns what we are searching for and the search engine learns what we are querying. Furthermore, when private information is stored at a proxy, this

proxy has complete control over our private data.

Proxies are trusted not to reveal the Internet behavior and private data of their users to other parties (unless the user makes everything explicitly publicly available e.g. when using Twitter or publishing pictures on Facebook). Unfortunately, private data leaks out unwanted [149] or proxies keep records of all private communications for business purposes [47] or due to governmental legislation [16, 48]. Proxies should be considered untrustworthy by their users when handling private data i.e. untrustworthy proxies are not trusted of keeping private information private. Users should especially consider the consequence of free Internet services when advertisement is the main source of income.

When browsing the internet, proxies are inevitable. Although they pose a potential threat to our privacy, they can be used to perform automated and powerful computations e.g. query hundreds of websites simultaneously. The question is, can proxies perform these computations while their users still remain in control of their privacy?

1.1 Privacy

There exist various definitions for privacy. The definition by Alan Westin [156] defines privacy from a 'control of information' point of view: "The ability to determine for ourselves when, how, and to what extent information about us is communicated to others". Privacy is an abstract and controversial notion which can be divided into four categories [119, 132]:

Personal privacy – Concerned with limitations on interference into home, private life, personal property, public space or workplace e.g. video surveillance or identity checking.

Physical privacy – Concerned with the protection of the integrity of person's body e.g. in case of genetic testing, drug testing and medical testing. It is also concerned with the protection against individuals having access to information about the physical health of a person e.g. doctors, employers and insurance companies.

Communications privacy – Concerned with individuals being able to communicate amongst themselves using mail, telephones, e-mail or any other forms of communication without being routinely monitored by other persons or organizations.

Information privacy – Concerns that private data will only be used for the purpose for which it was collected and not disclosed to others without consent e.g. credit information, and medical and government records. Also referred to as data privacy.

Due to the technological advances like the Internet, we should all be concerned with the privacy of our private data i.e. information privacy. In our perspective the challenge

on the Internet is to share data while protecting Personal Identifiable Information (PII) i.e. any information related to an entity or individual who is the subject of the information. PII is considered non-public data as defined by law or personal policy and can be used to identify, contact, or locate a person e.g. his/her social security number, birthday, health information, ethnicity or home address.

Private data or PII has become a commodity [127] and is used to gain discount at the local supermarket or to access to free services on the Internet, like search or webmail. Protecting our privacy on the Internet can be regulated, but not be guaranteed, by legislation. To accomplish information privacy on the Internet it needs to be enforced by technology such as Privacy Enhancing Technologies (PET), defined by [18]:

> *"A system of ICT measures protecting informational privacy by eliminating or minimising personal data thereby preventing unnecessary or unwanted processing of private data, without the loss of the functionality of the information system."*

By using privacy enhancing technologies, users can employ a range of programs and systems that provide a degree of privacy and security. We have divide privacy enhancing technologies into four separate but related categories:

1. The first category is self-determination, by letting the user remain in control over the information that is communicated to a service provider. For example, by minimising the communication of private data. Only the relevant information can be collected by the service providers. Another solution is to use a unidirectional network where information flows from the service provider to the user but not the other way around [89].

2. The second category is that there is an agreement between the user and the service provider about how to handle private data. This can vary from data tracking, where the service provider is enforced to use techniques that log and archive the handling of the users private data e.g. cryptographic traces [151, 152]. Or based on policies, which can be derived from legal rights, about data handling like inspection, manipulation or deletion.

3. The third category is to separate the private data from the information related to a user i.e. removing the link between the data and the identity of the user. For example by choosing a degree of anonymity and replacing the identity with multiple fake identities (pseudonyms). Other solutions are electronic cash [34, 37], anonymous digital credentials [35], mix networks [33, 90] and onion routing [79].

4. The fourth and final category is to protect private data while preserving the relation between the data and the information related to the user. This can be achieved by using for example encryption tools which prevent unauthorised access to the private data. Other solutions are Oblivious Transfer [3, 128, 148], Homomorphic encryption [62, 125, 131], Multi-Party Computation [36, 77, 158] or Threshold Cryptography [51, 53, 142].

The first category, self-determination, is a fundamental process in preserving privacy and relates to all other categories from privacy enhancing technologies. Solutions from the second category are used to indicate afterwards who was responsible for the privacy invasion, while the solutions from the third and fourth category preserve privacy before it is send to the service providers.

There are numerous solutions in achieving privacy enhancing technologies of the third category e.g. connecting anonymously to the Internet using free and open wireless internet connections, or by creating multiple pseudonym email addresses. There are various cryptographic solutions which address the fourth category of privacy enhancing technologies. Nevertheless, there is a limited subset of cryptographic techniques for the fourth category which address privacy enhancing technologies in case of untrustworthy proxies. These privacy enhancing technologies are denoted in this thesis as 'Private Computing with Untrustworthy Proxies'.

We define 'Private Computing with Untrustworthy Proxies' as:

Private Computing – A subset of privacy enhancing technologies that maintains the relation between the private data and the information related to the user, while preserving privacy when the private data is processed by an untrustworthy proxy.

1.2 Problem Addressed

Access control systems are proxies which grant access (or not) to devices or services based on passwords. After the user has provided his password to the proxy, the proxy is able to compare the password with a previously provided password by the user, which is used for reference. But the proxy is unable to learn the password since it is hashed by the user. If the hidden passwords are equal, access is granted, otherwise not. With this mechanism, the proxy is able to learn that the passwords are equal (or not) without learning the actual passwords.

Similar techniques can also be used to verify two users hold the same information without revealing this information to each other beforehand. Both users cryptographically hash their information and submit independently their hash to the proxy. The values are compared by the proxy without having to interact with the users. The result of the comparison is communicated to both users. After comparing the hashes the user and the proxy learn the data between the users is equal or not. If the data is equal both users know they share the same information but this is not learned by the proxy. If the hashes are unequal all parties learn only that the submitted information was not the same and nothing more.

The restriction of these access control systems is that they are limited in functionality, since the proxies are only able to learn that the passwords/hashes are equal or not. The proxies in the access control systems only compute the equality function e.g. if $a = a'$ where a, a' are hashed passwords. The challenge is to extend the functionality of the proxies by letting proxies compute more complex comparison functions such as

inequality functions e.g. less than $<$ or greater than $>$ [11]. These functions could extend access control systems by comparing if two values are equal to each other within a pre-defined margin of error e.g. if $a+\varepsilon > a' > a-\varepsilon$ where a, a' are hashed/encrypted passwords and ε is the error-margin.

The computation of complex functions by untrustworthy proxies is addressed in the research about mobile software agents, mobile code (privacy) [30] and computing with encrypted functions [136, 137, 138]. Agents are software being able to execute tasks for their users autonomous and independent. Agents are able to move over networks where they can communicate with other agents. Furthermore agents can be intelligent and cooperate in order to achieve a common purpose [82].

The main challenge with mobile agents is the environment where they are executed on. Can the agent trust the environment not to look into the information the agents is carrying? How can the agent communicate privately with other agents while the environment is constantly involved within their communication? This problem is comparable to the Man-in-the-Browser (MitB) attack [124], where users do online banking while their browser is compromised by malware or trojans to manipulate the banking transactions. In both cases the executing environment is an unavoidable untrustworthy proxy where information has to be processed.

Privacy-preserving protocols allow multiple parties with private inputs to perform joint computation while preserving the privacy of their respective inputs. An important cryptographic primitive for designing such privacy-preserving protocols is secure function evaluation (SFE). The classic solution for SFE by Yao [158] uses a gate representation of the function that the two parties want to jointly compute. Other cryptographic techniques such as garbled circuits, homomorphic encryptions or combinations of algebraically homomorphic encryptions and garble circuits are also used [75]. Applications of SFE can be found in Privacy-Preserving Genomic Computations [70, 92, 146], Remote Diagnostics [21], Graph Algorithms [22], Data Mining [112], Credit Checking [8], Medical Diagnostics [10], Face Recognition [64, 134], or Policy Checking [69]. Although SFE, but also Multi-Party Computation (MPC) and Two-Party Computation (2PC) give the ability for two or more parties to execute any function securely, their solutions [91, 120, 158] do not consider the use of an untrustworthy proxy to execute the function privately.

Another restriction of SFE is the ability to collect information privately from multiple sources while an untrustworthy proxy has to facilitate the communication and execution of the collection function. Although secure 2PC can be extended to MPC protocols, the use of untrustworthy proxy is not considered. A common technique within MPC is Oblivious Transfer (OT) where a sender sends some information to a receiver, but remains oblivious as to what is received [45, 128]. Applications of OT are e.g. database access [26] or mutually authenticated key exchange based on possibly weak passwords [116].

Research on cryptographic where proxies are a central part is scarces and mainly limited to research on mobile software agents in untrustworthy environments. The objective of this thesis is to address two basic cryptographic functionalities which provide

privacy for the user while proxies are involved in the communication and computation between parties: Collecting Private Data (discussed in Section 1.2.1) and Comparing Private Data (discussed in Section 1.2.2).

1.2.1 Collecting Private Data

The first challenge is to let a proxy collect private information from various sources. Therefore the first problem addressed is:

> *"Develop protocols for letting an untrustworthy proxy retrieve information from various sources while keeping all the collected information private in respect of the various sources."*

By selecting and collecting of private data, a user selects various sources where information must be collected while letting a proxy collecting this information without being able to learn what information was selected or collected.

The following example, about Location Based Service (LBS), demonstrates the process of collecting private data. Mobile devices connected to the mobile network are traceable by the mobile network operator. Information, based on the location of the mobile device of a user, can be requested from a LBS. This service can vary from requesting the nearest restaurant, ATM machine, locating people on a map, traffic information or warning the nearest doctor in case of an emergency. Considering that the communication between the mobile device and the LBS has to go through the mobile network operator, it has a great impact on the privacy of the user. First of all, the location of the user can be tracked by the LBS. Secondly, the mobile network learns which service was requested, and finally learns the outcome of the LBS. An additional consideration is that the collection of the selected information preferably is done without the aid of the mobile device because mobile devices are limited in processing power and communication with the mobile network is costly. Within this specific example communication with the mobile device should be minimized and all computations should be done by the proxy.

We can derive the following set of assumptions and requirements for the new private computing collection schemes:

- There is an unavoidable proxy which performs the collection and most of the computations.
- The inputs and outputs must remain private to the users i.e. confidentiality should be guaranteed.

Although various approaches seem suitable for solving our problem such as oblivious transfers [128], they do not take into account that the information must be transmitted through an unavoidable and untrustworthy proxy. Two approaches are considered. The first approach will combine basic SFE techniques such as homomorphic encryption, blind signatures and oblivious transfers into a solution which can cope with the untrustworthy proxy. The second approach will be based on cryptographic techniques

used within mobile software agents [143]. This latter approach will show that solutions for collecting private information is not bound by standard techniques used in SFE solutions i.e. not using oblivious transfers, homomorphic encryptions or garble circuits.

1.2.2 Comparing Private Data

The second challenge is in letting the untrustworthy proxy compare the collected private information. As addressed in Section 1.2 the equality function is a basic function for comparing private inputs. Furthermore, from SFE we know that basically every function can be computed securely but due to the ideal/real model [28, 29] solutions omit the use of proxies. Cachin [24] addresses a SFE private bidding problem by using an oblivious third party to ensure fairness. Homomorphic cryptosystems such as RSA [131], ElGamal [62], Goldwasser-Micali [81], Benaloh [14] or Paillier [125, 126] provide algebraic operations on the plaintext while the inputs are encrypted.

Yao's Millionaires' Problem [158] is an important problem in cryptography and used within e-commerce, data mining, linear programming and operational research. Due to the importance of the comparing operation within many (cryptographic) application this fundamental building block should also be available within private computing. The second problem addressed is therefore:

> *"Develop protocols for letting an untrustworthy proxy compare private information by computing an inequality function in such a way that the private information remains private in respect of the users, but the results of the comparison can be made public if required."*

Can a proxy compare various values to one another while they are hidden, and say something useful about their relation? As already described, access control systems are able to compare whether two independently obtained values e.g. passwords, are equal to each other or not. With this result, the system learns one bit of information about the relation between the two values, but does not learn what the actual passwords are. These password systems are only able to compute the equality function while the challenge is to compute more complex functions resulting in more sophisticated access control systems.

Let us give an example of comparing private data in an Internet environment based on mobile software agents. As described in the introduction of Section 1.2, agents are autonomous pieces of software that run on (remote) hosts in order to carry out tasks on behalf of its user. The code of the agent itself is transported over the Internet and executed on the remote hosts. This is in contrast to 'normal' code, which is executed on the local host while data is transported between the remote host and the local host.

Buyer agents are mobile software agents that help users to buy items on the Internet. They move around the Internet in search for seller agents willing to sell the requested item for the right price. Before an item can be purchased the buyer agent and the seller agent have to come to an agreement about the price. The buyer agent has a maximum

price he is willing to pay, while the seller agent has a minimum selling price. The agents don't trust each other but they also do not trust the environment where they are executed i.e. untrustworthy proxies.

The same requirements, as for the collection scheme, also hold for the new private computing comparison schemes with additional requirements regarding the comparison process:

- The proxy performs the comparison computations.
- The inputs must remain private to the users.
- The comparison function computed by the proxy should be an inequality function.
- The proxy learns (some) information of the output (e.g. greater than or not) without being able to learn the actual inputs.
- The output should be made public by the proxy (All users should be able to learn the same information as the proxy).

Our approach is to adapt a well known SFE solution by Cachin [24]. Cachin uses an untrustworthy proxy to ensure fairness, but does not provide the capability to compute the result of the comparison without leaking information (in a controlled matter). We start our approach with the requirement of computing one inequality function and continue with an approach capable of computing multiple inequality functions.

1.3 Organization and Contribution

1.3.1 Organization

An overview of the content of this thesis is given in figure 1.1. The content is divided into various parts. The first part is the skeleton containing Chapters 1 and 2. It starts with the introduction where a relation between privacy, privacy enhancing technologies and private computing is given. Chapter 2 starts with the problem of collecting and comparing private data by an untrustworthy proxy and continues with the private computing problems based on a general private computing scenario. Furthermore two private computing problems are dissected resulting in four unique approaches to the problems addressed.

The second part of this thesis composes the body and is divided into two sections. The first section consists of Chapters 3 and 4 which address the selection and collection problem. Both chapters tackle the selection and collection problem from a different point of view. Chapter 3 is mainly based on oblivious transfers and letting the proxy communicate with all parties simultaneously i.e. in parallel. While Chapter 4 is based on hash chaining and the collection of information is based on minimum communication with the proxy.

The second part, consisting of Chapters 5 and 6, address the collection problem. Both chapters get down to the problem based on multi-party computation techniques and homomorphic encryptions. The challenge in Chapter 5 is to compute a single comparison function, while Chapter 6 focusses on the comparison of multiple values.

In Chapter 7 we conclude with a discussion where we reflect on the results and the choices made. Furthermore we evaluate and compare the different presented solutions. Finally we give recommendations and an outlook to the near future with respect to private computing.

Appendix A describes various building blocks used within private computing and which are used in chapters 3, 4, 5 and 6.

1.3.2 Contribution

We summarise the contributions of this thesis:

- Chapter 3 builds on the idea on developing a framework where the user's location and subscription are processed in the encrypted domain, as a result we achieve privacy in a new way for certain classes of Location Based Services. To ensure privacy, our approach uses cryptographic protocols such as oblivious transfer and homomorphic encryption. The research on privacy friendly Location Based Services has been carried out together with Markulf Kohlweiss, which resulted in a publication at [99]. The research was extended together with Sebastian Faust, Lothar Fritsch and Bart Preneel and published at the PET 2007 workshop [98].
- In Chapter 4 the solution is a novel approach for comparing private data while dealing with untrustworthy parties. This contribution enables the proxy to retrieve information from various untrustworthy users, followed by a computation on the retrieved information. This paper builds on the work by [143], by minimizing the amount of communication between the parties. This paper has been carried out in close collaboration with Martin van Hensbergen and resulted in the conference papers [73, 87].
- In Chapter 5 by combining the Private Agent Communication Algorithm by [30, 31] and the fair multi-party computation protocol by [24] we have made it possible to compare the maximum of two independent inputs without the intervention of the users. This resulted in paper [72].
- Chapter 6 extends the Φ-hiding assumption by using the concept of [24]. As a result an efficient inexact matching protocol has been developed, capable of comparing two or more inputs by a public party without the interaction of the user. A paper was published with this result in [74].

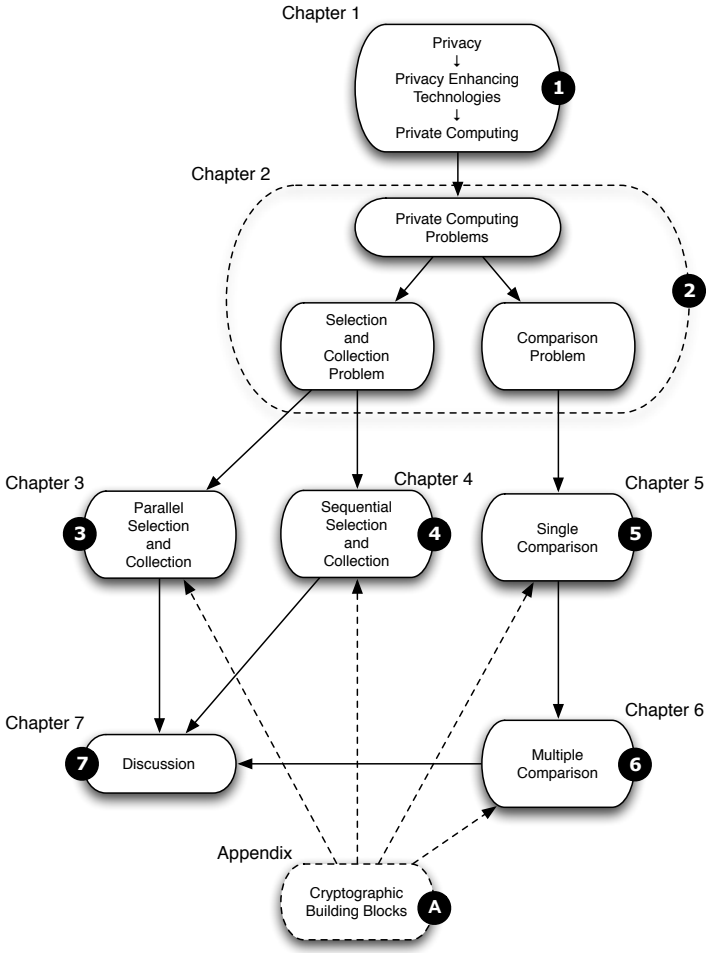


Figure 1.1: Thesis Overview

Chapter 2

Private Computing Problems

Private computing preserves privacy when private data is processed by a proxy. There are various applications for letting a proxy process our private data. For example, for authentication purposes, where the proxy needs to learn some information from the private data to decide if access will be granted or not. Or it provides more functionality for the user in letting a proxy do the computations instead of doing it on its own. This chapter describes the general private computing scenario used within this thesis and two specific private computing problems: the collection problem and the comparison problem.

2.1 Introduction

Within a private computing scenario, various parties should communicate with each other even though they do not trust each other. Figure 2.1 gives an overview of a private computing scenario where each parties has its own role. The following parties are present in the scenario:

User – The user is the initiator of the scenario and is considered to be trusted by all parties.

Hosts – The hosts are untrustworthy parties from the user point of view, willing to participate in the process but curious about the private data of the user. Untrustworthy is also known as semi-trusted or passive attacker [115] which threatens the confidentiality of the private data.

Proxy – The proxy executes the private computing processes and is considered to be untrustworthy from the user and the hosts point of view.

The general private computing scenario considers a user using an untrustworthy proxy to process private data while keeping it private, as defined in Section 1.1. Depending on the application, the private data can be acquired from the user but also from various

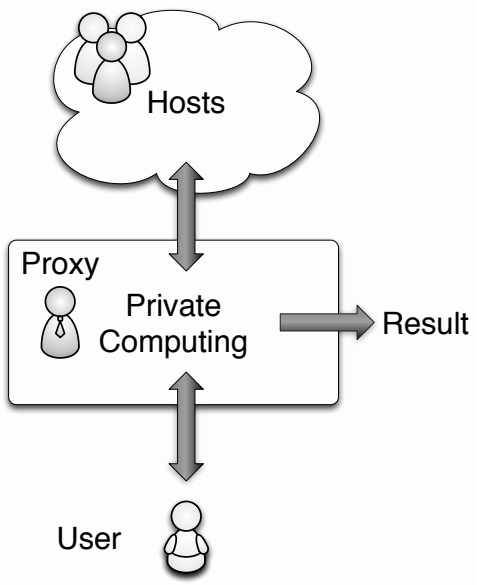


Figure 2.1: Private computing scenario

hosts. Furthermore, depending on the application the result of the computation can be given to the user or can be made public (while maintaining privacy). In Section 1.2 a motivation has been given regarding the two private computing problems addressed in this thesis. The first is the collection of private data and the second is the comparison of private data. Although both problems can be addressed independently they can also be merged together into one scenario. For example, in figure 2.1 the user selects the various hosts he wants to communicate with. Based on this selection the data is collected by the proxy and compared with the collected data received from the user.

Since proxies are unavoidable, the assumption for private computing is that the collection and computation has to be done by the proxy while keeping all inputs private to all parties, excluding the user. This also indirectly presumes that all communication between the hosts and the user has to go through the proxy i.e. there is no direct communication between them.

The primary threat is that the user looses control of his private data to the proxy i.e. looses confidentiality. Data appears in various forms e.g. the location or names of the hosts which are queried, the private data which was collected from the hosts or the information which is send to the proxy. This also holds for the data which is processed by the proxy, to data which is made public.

There are various private computing applications which use a proxy to carry out the communications or computations:

- An access control system acts as a proxy and is able to authenticate users while

keeping their private data confidential.

- Location-Based Services, where a user with a mobile device is using various services over a mobile network (proxy).
- Online banking, where a browser acts as a proxy between the user and the bank.
- Internet Service Providers are the proxies (gateways) to the Internet.
- Search engines are the proxies handling our search queries.

This thesis addresses two private computing problems. The first problem considers the challenge of letting a proxy collect private data without learning what was selected or collected. This is described in section 2.2. The second private computing problem addresses the comparison problem where a proxy compares private data in such a way that the data remains private while the results are made public. The second problem is described in section 2.3.

2.2 The Selection and Collection Problem

The selection and collection problem wants an user to select various information sources from which an untrustworthy proxy must collect this information without being able to learn what information was selected and collected. This problem is similar to the privacy preserving database querying problem [61, 157] where a user has a query Q and a host has a database $M = \{m_1, \ldots, m_j\}$. The user wants to know whether there exists a value m_i in the host's database that matches Q. The requirement is that the host cannot learn the query and the result of the matching, and the user should not learn the complete content of the database of the host other than the result of the query. This matching problem can theoretically be solved using general techniques of Multi-Party Computation [44, 77] and has been considered widely in literature [41, 42, 44] using Private Information Retrieval (PIR) techniques.

Although Multi-Party Computation based on PIR techniques are suitable for various applications like privacy preserving database querying or privacy preserving data mining [60, 105], they do not consider the use of untrustworthy proxies being involved in the problem. The selection and collection problem addresses the problem of letting an untrustworthy proxy collect information from various hosts while keeping all the collected information private.

2.2.1 Scenario

A user selects privately which hosts have to be queried. He hides his selection and sends this to the proxy. The proxy ensures that the correct information is retrieved at the selected hosts. The hosts provide the requested information by the user while hiding it for the proxy. At the end of the collection process the information is processed by the proxy and the result is made public or presented to the user.

2.2.2 Assumptions

Various assumptions apply to the selection and collection problem:

Communication – The user is unable to communicate directly with the hosts. Therefore all communication between the hosts and the user passes the proxy. But, the hosts are able to communicate with each other without the aid of the proxy, since the proxy only acts as an intermediary between the host and the user. This assumption, regarding communication, is a direct consequence of the general assumption of this thesis, that a proxy is unavoidable, i.e. cannot be circumvented.

PKI – A public key infrastructure (PKI) is assumed to be available, which is used to facilitate key distribution for secure communication.

Trust – The user is trustworthy while the hosts and the proxy are untrustworthy parties. The proxy is seen as an untrustworthy 'man-in-the-middle' which processes all the information correctly but is curious regarding the content of the information i.e. the proxy provides integrity but not confidentiality.

2.2.3 Threats

The threat model of the selection and collection problem is based on the trust we have in the parties involved [105, 107, 111]. When comparing Multi-Party Computation with private computing, the approach within Multi-Party Computation is not to assume the ideal situation, with a trusted proxy doing all the computation; but to assume the real situation where no external proxy exists that can be trusted by all parties. This is called the ideal/real model [28, 29]. Private computing assumes the proxy to be unavoidable and untrustworthy i.e. it follows the protocol but attempts to learn private information from the received data; by deviating from the protocol specifications. Therefore, the real situation within private computing is the existence of an proxy which has to perform all computation but cannot be trusted by all parties.

Since the proxy is unavoidable and the proxy together with the hosts are untrustworthy, it makes them all a threat to the users' privacy. The following list of threats applies to the selection and collection problem:

Threats towards the selection by the user – The user selects the hosts he wants to collect information from. If the proxy remains oblivious to the hosts selected, the proxy is unable to manipulate the collection process in his advantage; other than not performing the collection process or randomly discarding hosts during the collection. If the hosts are known by the proxy, the proxy will be able to manipulate the hosts which are being queried and therefore influence the collection process in advantage of the user.

Knowing which hosts are being queried also threatens the privacy of the user i.e. linking the user to specific hosts.

Threat towards the collected information – The result of the collection is only intended for the user and should not be known by other hosts or the proxy. Collected information can consist of sensitive and private information e.g. the age of the user or the location of the user.

The user could be mislead or misinformed if the collected information would be modified before it is received by the user.

2.2.4 Requirements

Requirements are based on the threats within the selection and collection problem.

Selection of information requirements – The user should control how much information is learned by the hosts or proxy regarding which hosts are being queried. Therefore, a mechanism should be applied which ensures privacy of the selected hosts without letting other hosts and the proxy learn which hosts were queried.

Collection of information requirements – The collected information must remain confidential to the proxy and the hosts i.e. hosts and the proxy should be unable to view the content of the collected information, other than the content which they have submitted.

An integrity mechanism is required that prevents alteration of collected information. The proxy or the user should be able to verify the collected information.

2.2.5 Problem addressed and Approach

Based on the threats and requirements the collection scenario addresses the problem of letting an untrustworthy proxy retrieve information from various hosts, while keeping all the collected information private.

We introduce two approaches to the selection and collection problem, the sequential approach and the parallel approach. The sequential approach is shown in figure 2.2 and the parallel approach is shown in figure figure 2.3. Both approaches are based on the general private computing model shown in figure 2.1.

The two approaches are comparable with approaches within MANETs (Mobile Ad hoc Networks) [5]. MANETs are self-configuring networks of mobile devices. Within these network the Base Station (BS) plays a critical role, since it is involved in the communication with every mobile device available. Compared with private computing, the BS is the proxy and the mobile devices are the hosts. Two models are used within MANETs, Single-Hop (Single-path) routing and Multi-Hop (Multi-path) routing. The difference between the two models is that in case of Single-Hop all communication between mobile devices (the hosts) is routed strictly through the proxy (BS), while with Multi-Hop the communication is routed between the mobile devices directly.

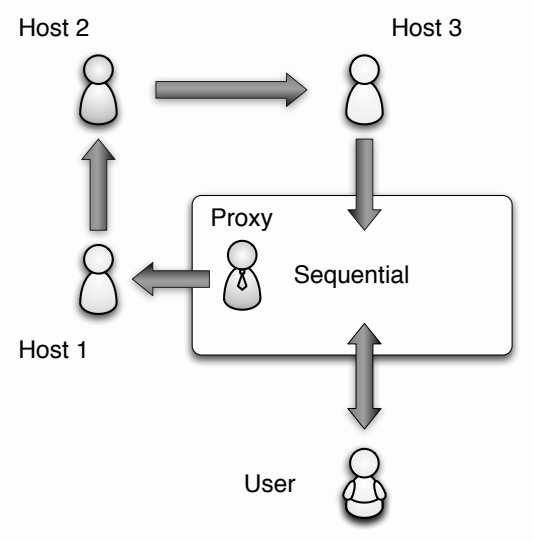


Figure 2.2: The sequential selection and collection approach

Wireless Sensor Networks (WSNs) are related to MANETs [71]. WSNs consist of distributed autonomous devices which are made up of tiny sensor that have the goal to collect information and to make a decision based on this information. WSNs are used to monitor cooperatively physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants, at different locations; or in healthcare, home automation, and traffic control applications.

MANETs and WSNs are in some ways similar, since they are both distributed wireless networks which may involve the use of intermediate (proxy) sensors between routing. The difference is that WSNs are mostly battery powered and therefore have different resource and computing constraints, opposite to MANETs [71]. Due to limited power and data storage, one of the main security concerns with WSNs is that a sensor is interacting within an untrustworthy environments, with the threat of loosing private information [153].

Sequential Approach – The sequential approach is comparable with the multi-hop distributed wireless networks. The approach is sequential since all communication is passed on from host to host (hops). The user starts the scenario by submitting a query to the proxy. Within this query the user pre-defines the itinerary i.e. the path or hops which have to be visited in a particular order. The proxy learns from the query to which host he needs to forward the information received from the user. After a host has processed his part of the query, the host submits the query and the result of the process, to the next host. This process continues until the query has reached the proxy again. Finally, the proxy computes the remainder of the query and submits his result to the user.

To overcome the problem that hosts are unwilling to participate or are unaccessible during the collection process, a helper procedure needs to skip any unwilling or unreachable host. This sub-protocol should ensure that the itinerary is not stopped during the process and all the preceding information can still be collected.

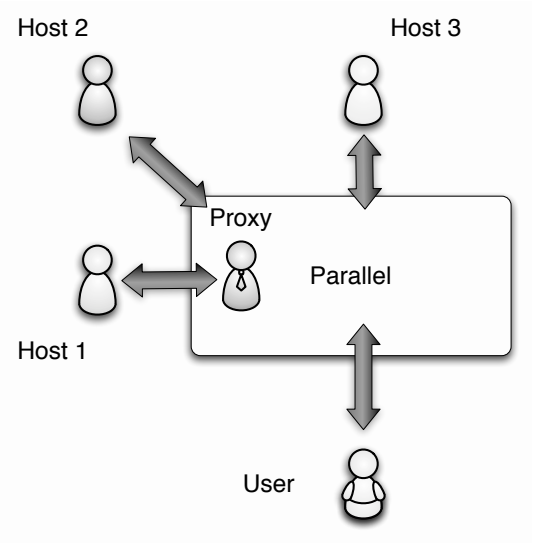


Figure 2.3: The parallel selection and collection approach

Parallel Approach – The parallel approach can be seen as a single-hop distributed wireless network where all communication with the hosts and user is strictly executed by the proxy. There is no direct communication between the various host, as it is with the sequential approach. Furthermore, opposite to the sequential approach, the proxy communicates with all hosts in parallel.

The user pre-defines from which hosts he wants to collect information and provides this query to the proxy. The proxy communicates in parallel with all available hosts, without being able to learn from which host information is actually being collected. After the collection process has finished the proxy processes the result of the collection and submit this to the user.

2.2.6 Related work

With the sequential and parallel approach known cryptographic protocols will be used. Various cryptographic building blocks will be combined to create new private computing protocols.

Sequential Approach – Mobile agents and mobile code [30] are the basis for the sequential approach. As described in Section 1.2, mobile agents are pieces of software moving from one host to another, while collecting information and performing tasks.

After a mobile agent is transferred from one host (sender) to another (receiver) over a network the mobile code within the agent is executed on the recipient's side. The execution of software on the recipient's computer presents a number of security issues. From the recipients' point of view, the execution of 'untrustworthy' software can harm the recipients computer e.g. the software can contain a virus, a worm or a Trojan. A number of techniques have been suggested, and used, to alleviate some of these issues, like sand boxing [102] and the use of certificates [145]. From the senders' point of view, the execution of the software on the recipients computer can be manipulated or spied on because the recipient has full control over the software.

In [138] the threat of executing mobile code in untrustworthy environments is acknowledged. They introduce the general problem of *non-interactive evaluation of encrypted functions* (EEF), where a host has a private function f and a user has an input x. The user wants to compute $f(x)$ but the host wants the user to learn nothing about that function. Two approaches are explained, the first based on homomorphic functions and the second is functional decomposition problem based [159]. Although both solutions are leaking information and are therefore not the general solution for mobile agents protection, they provide evidence that software can at least be partially protected against malicious hosts.

White box cryptography [93] prevents disclosure of secret keys in untrustworthy host environments. It is a special class of software obfuscation [9]. Although this research field deals with untrustworthy hosts, it focusses on the implementation of the software and the protection of keys, rather than on the execution of specific functions. White box cryptography research focusses on the implementation of specific cryptographic algorithms like DES [108] and AES [43].

In [143] an e-commerce problem with untrustworthy hosts has been addressed using the concept of mobile agents. The proposed scenario is similar to the sequential problem and the solution uses hash chaining techniques from [95]. The limitations of this solutions is that it is based on multiple mobile agents e.g. for the protection of the private keys. It does not provide a complete solution in the case that a single agent would be used.

Location-based privacy in Wireless Sensor Networks (WSNs) was addressed by [113, 114]. The focus of this research was on the contextual information which remains unprotected during operation e.g. location of an object which is tracked by the sensor network. The main threat within this model was that the adversary is able to intercept any traffic information within the network. The result of this research show that location-based privacy can be achieved with a minimal communications overhead. Although this paper solves a location-based privacy threat, it does not provide a solution of collecting information privately based on the location of the user.

Parallel Approach – With the parallel approach each host has information and the user wants to learn a single piece of this information. Cryptographically this relates to an Oblivious Transfer (OT). With OT we usually have a host (sender) and a user (receiver). The host holds a list of n values $x_1, \ldots, x_n$, and the user wants to learn x_i. The problem is that the user does not want to send i to the host; while the host does

not want to send the complete list x_j (for $j \neq i$) to the user. With OT the hosts transfers x_i to the user without knowing i. OT was first introduced by Rabin [128] and further developed by e.g [3, 57, 109].

In principle OT protocols do not take into account the use of proxies between a sender and receiver. The approach, for using OT within the parallel problem would be to combine various OT protocols; by exploiting the fact that the user is not querying a single host but is querying multiple host.

Private Information Retrieval (PIR) [12, 25, 42] is a cryptographic primitive related to Oblivious Transfer. Also called symmetric Private Information Retrieval, because both the user and the host have a privacy requirement. PIR allows a user to retrieve information from a host without revealing which item they are retrieving. In a single host scenario the only way to achieve privacy is for the host to send the entire list of n values to the user. One way to solve this problem is to assume multiple non-cooperative hosts each having a copy of the entire list of n values.

2.3 The Comparison Problem

The comparison problem copes with an untrustworthy proxy who wants to calculate an inequality function with the private inputs from various hosts. The challenge is to keep the private inputs private but the result of the comparison can be made public. Two basic comparison functions in mathematics are equality and inequality. Equality functions compare information and give as output whether the information is identical or not. This is a valuable function when comparing static passwords. With inequality [11] we mean the less than $<$, greater than $>$, less than or equal to $\leqslant$, greater than or equal to $\geqslant$, not greater than $\not>$ and not less than $\not<$. Inequality functions are used to compare the relative size (or order) of the information.

The millionaires' problem is a Multi-Party Computation problem introduced by Yao [158]. The problem describes two millionaires who want to compare their wealth with each other and compute who is the richest, while keeping their actual wealth private. In an ideal situation this can be achieved using a mutually trusted party. In a real situation there are no trusted parties and the millionaires have to solve this problem together. With private computing the proxy does all the computations for the millionaires without being able to learn how much millions they own.

Furthermore, the challenge in case of MPC is not to leak any information. The millionaires are privately able to learn the outcome of the comparision (inequality function) but should know nothing more e.g. the wealth of the other millionaire. With private computing the proxy learns the outcome of the comparison but nothing more.

The millionaires problem was the first problem [158] within MPC and gave way to the generalization of MPC by [78]. The millionaires problem is an example of calculation of the inequality function where both parties calculate the greater than function. To set the first steps for generalization of private computing we focus on an important build-

ing block, the inequality functions, such as Yao [158] focussed on the millionaires' problem with his initial paper on MPC

2.3.1 Scenario

The comparison scenario in figure 2.4 shows the comparison of information without loosing privacy:

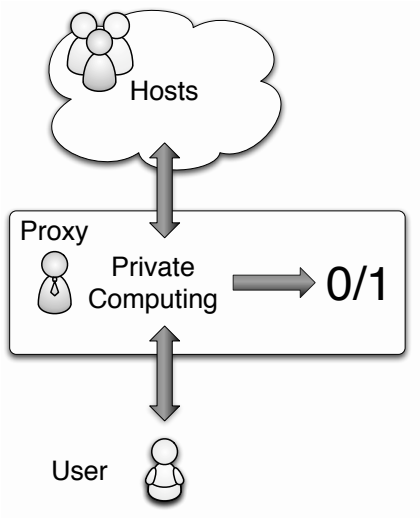


Figure 2.4: The comparison scenario

The user hides his private information and sends this to the proxy. The hosts also hide their private information and submit this to the proxy. The proxy compares the received values with each other and is able to make the result of the comparison public without being able to learn what the initial values where. The result of the comparison is binary, since the proxy is calculating inequality functions. For example, when the proxy compares which of two encrypted values (a,b) is the greatest, the public result will be 0, if e.g. a is the greatest, and 1 otherwise.

2.3.2 Assumptions

The same assumptions apply to the comparison model as to the collection model, given in Section 2.2.2. One additional assumption is added:

Binary output – The comparison model will focus on the inequality function and the output of the function is binary i.e. 0 or 1, Yes or No.

2.3.3 Threats

The following threats apply to the comparison model.

Threats towards the inputs – The inputs should remain confidential and are not intended to be known by other users, since knowledge of the inputs by any other party can be used to influence the outcome of the computation.

Threats towards the comparison process – With every comparison process information is learned by the proxy or made public. This process leaks, with every comparison, one bit of information about the relation between the two private inputs. The threat is that the proxy performs a chosen plaintext attack repeatedly, where the proxy feeds the comparison process with chosen inputs based on a binary search algorithm, it would be able to learn the hidden inputs.

2.3.4 Requirements

The following is a list of requirements for the comparison model.

Input requirements – The inputs must remain confidential and should not be known by the proxy when they are received or learnt when they are processed.

Output requirements – The proxy learns the outcome of the comparison process. Furthermore, the proxy learns 1-bit of information about the relation between the two inputs without learning what these inputs actually where.

Comparison process requirements – An inequality function will leak 1-bit of information with every comparison. The proxy which compares the private inputs will only learn the relationship between the inputs and should not learn anything about the plaintext relation. Therefore the proxy should not be able to perform a chosen plaintext attack by replaying the comparison with chosen plaintext.

2.3.5 Problem addressed and Approach

The comparison model addresses the problem of letting an untrustworthy proxy compare private data in such a way that the data remains confidential but the results of the comparison is made public.

Single Comparison Approach – The first approach, see figure 2.5, in solving the comparison problem is based on techniques from Multi-Party Computation (MPC), where we especially look at approaches where untrustworthy proxies where used to compare two inputs. Although these proxies compute a public function they are unable to learn the outcome of the computation. Our first approach is to adept these MPC protocols by enhancing the capabilities of the proxies. We start with the comparison of two input (a, a').

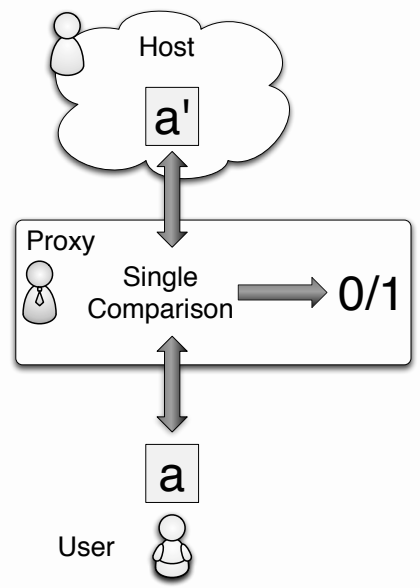


Figure 2.5: Single Comparison

Multiple Comparison Approach – The second approach, multiple comparison, see figure 2.6, compares multiple values. By using for example two inequality functions we are able to create a distance function or approximation function. Let us assume an error ε we want to decide if a is approximately equal to a'. We assume they are approximately equal if the $Distance(a,a') < \varepsilon$. We can therefore also compute $a-\varepsilon < a'$ and $a' < a+\varepsilon$. If we would use two single inequality functions and the result of the comparison would be that $a \neq a'$ then the proxy will learn which of the boundaries failed, $a-\varepsilon$ or $a+\varepsilon$. However when using multiple inequality functions in one pass, the user will not learn which boundary failed.

The host in this scenario only acts as public storage for the user since the user is the only party submitting values. The values are encrypted by the user $E(\cdot)$ to provide confidentiality. The proxy uses the encrypted values as reference when comparing the values from the user, with the already stored values at the host.

2.3.6 Related work

When comparing values we want to hide our private data but still be able to perform some computations on it. Homomorphic encryption [62, 81, 122, 125, 126, 131] can be used to combine two ciphertexts and create a third ciphertext where the plaintext is related to the first two. The drawback of these homomorphic encryptions is that they are limited to an addition, the plaintext values of the two cipertexts are added to each other in the third ciphertext, or to a multiplication, where the original plaintext

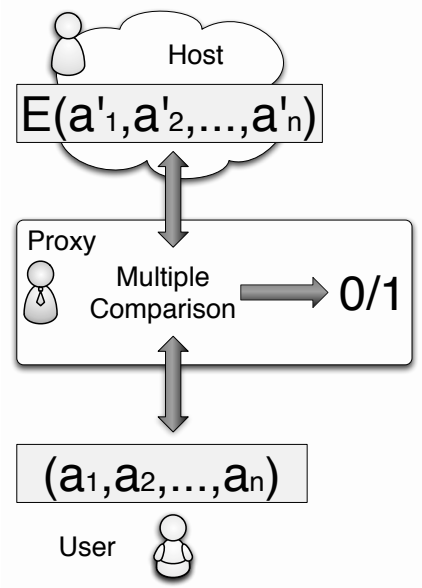


Figure 2.6: Multiple Comparison

values are multiplied within the third ciphertext. They are suitable as building blocks for various cryptosystems [24, 50, 88, 104] but on their own unable to compute an inequality function.

As introduced in Section 2.3, MPC is a problem introduced by Yao [158] and further described by [1, 13, 36, 77, 78, 129]. Yao's solution for the millionaires' problem is solved without the help of a Trusted Third Party (TTP), while remaining secure with untrustworthy adversaries. Lindell et al. [106] propose a secure two-party protocol which provides security against malicious adversaries. In general, the MPC problem is equivalent to having a TTP receiving the private inputs x_i from players i. It computes the function f and returns outcome y_i to every player i.

The comparison problem has some similarities with MPC. Within both problems a function must be computed privately amongst various parties. The fundamental difference is that the MPC problem is based on getting rid of a proxy to execute the function with multiple parties, while private computing wants to use explicitly an untrustworthy proxy which performs all the computations.

Another difference is who learns the outcome of the computation. With MPC the parties involved learn the outcome of the computation while with private computing the proxy learns the outcome of the computation. The difference between a proxy and a user is that the proxy does not submit any private inputs and only uses the private inputs of the hosts and the user. With MPC the users learn the outcome of the computation while they are in their own private environment, this in contrary to

private computing where the proxy is considered semi-trusted by the users but still needs to compute the outcome of the comparison function.

The consequence of these differences is that most MPC protocols are unsuitable for private computing since they do not use an untrustworthy proxy for the computation of the public function and do not provide a solution for letting an untrustworthy proxy learn the outcome of the computation.

Claessens et al [46] gives a survey on mobile agents doing secure electronic transactions in untrustworthy environments. Six areas are defined for solutions against untrustworthy environments: avoid the problem, server-aided computation, protect the data, minimizing the risk, execution integrity and execution privacy. Private computing relates to execution privacy since it describes the problem of executing functions in untrustworthy environments. Examples of execution privacy solutions are function hiding [136] and computing with encrypted functions.

Sander and Tschudin [137, 138] were one of the first to write about computations on untrustworthy hosts. They claim that there is no reason why programs must be executed in plaintext form. In Algesheimer *et al* [4] it is stated that a software only solution for secure mobile computation schemes do not exist and that if the host learns information that depends on the agent's code, it cannot be secure. They suggest to use at least one trusted user or proxy. Nevertheless, Cachin [24] shows that a semi-trusted proxy can be used to compute a function privately.

2.4 Discussion

The presented private computing scenario shows the basic principle of private computing where an untrustworthy proxy has to process private data without loosing privacy. Furthermore we choose to solve two private computing problems independently. The first, collecting private information, is a fundamental step in remaining in control of your private information when information is not directly accessible by the user and has to be collected from e.g. the Internet. The second, comparing private information, aims to extend the functionality of proxies while preserving privacy.

The advantage of addressing the Private Computing Scenario as two seperate problems is that the solutions to the problems can be used independently of each other in various other scenarios. Seperation also makes the solutions more flexible for combining them with other private computing building blocks.

Private computing should not be confused with Multi-Party Computation (MPC). Although various solutions from MPC can be classified (and used) as private computing solutions and *vice versa*. But, this does not imply that private computing is equal to MPC. Due to the ideal/real paradigm [28, 29] MPC does not assume that in the ideal situation a trusted party is doing all the computation, but it assumes the real situation where no external party exists that can be trusted by all parties. The fundamental concept behind MPC is to compute collectively a function without revealing the private

inputs and each user should be able to learn his own private output based on the public function and the private inputs. While private computing uses explicitly a proxy to perform the computation for all users without being able to learn the private inputs other then the result of the output of the computation.

Basically we can say that with MPC all parties try to compute one problem together, while with private computing the problem is given to the proxy and the user waits for an answer. MPC and private computing problems use similar techniques to solve their problems, but this does imply that private computing is a subset of MPC. Most MPC protocols will in principle be unsuitable for private computing since they require the users to compute the result of the public function privately after it has been handled by all other users involved in the computation. MPC also does not consider that a proxy should be able to learn information (binary) from the computation.

Chapter 3

Parallel Selection and Collection

How can a user obtain information from various sources while this information needs to be collected by an untrustworthy proxy? This chapter answers this question by describing a selection and collection protocol using a parallel approach whereby a proxy collects information simultaneously from various sources without loosing the users privacy. The solution is worked out within a location-based service (LBS) scenario, with the goal of secure processing of the location data. Its properties are the avoidance of personal information on the services side. Although a LBS scenario is presented, our solution can also be used in other, more generic, scenarios where a proxy acts as a man in the middle e.g. querying multiple databases or websites without loosing privacy.

3.1 Introduction

Privacy is an enormous topic [56]. It has many legal and commercial implications. The different ways in which location is used by a LBS influences our privacy experience highly. Are we interacting only with the service or with other users of the service? Is our location only used at the time of our request, or are we tracked constantly and notified upon certain events? Rather than covering all of these topics, we focus on a very specic sub-problem where information is collected privately based on the location of the user. Some of the techniques employed can however also be used for improving the privacy properties of other types of LBS protocols as surveyed in [100].

The research for this chapter has been carried out during a three months academic visit at the Katholieke Universiteit Leuven, Belgium at the COSIC group of Prof. Bart Preneel in 2006. The results where published in cooperation with Markulf Kohlweiss in [99] and with further cooperation with Sebastian Faust, Lothar Fritsch and Bart Preneel in [98].

We do not consider solutions involving anonymity or specific privacy policies since we focus on Privacy Enhancing Technologies which protect private data while preserving the relation between the data and the information related to the user, as described in Section 1.1. Moreover, we consider only solutions where no information at all about the user's location is revealed to the LBS. After the execution of the protocol a malicious LBS (even if collaborating with the proxy) cannot reveal anything, he (they) could not have revealed before.

With the growing availability of mobile internet the possibilities of LBS applications becomes a fact. An evident application of LBS is the ability to locate individuals or devices in emergency situations e.g. when a mobile phone is lost, the user can lock his mobile phone remotely or track the location of his mobile device. Navigation services has emerged in the last decade and has been augmented with real-time traffic information based on the location and itinerary of the user. Information service guides users through cities or provides users with location specific weather reports. For instance tracking services make it possible to track users while they are driving a car and bill them based on the time driven, distance covered and locations visited. Augmented reality is emerging on mobile phones where location specific information is projected on a physical real-world image e.g. a video camera pointing in a certain direction on a specific location. This chapter proposes a sophisticated Privacy Enhancing Technique based on oblivious transfers where we see that in recent years the tendency to incorporating privacy in LBS is limited to switching the service on/off or letting the user choose which level of detail, regarding his physical location, is used with LBS.

Our solution is not explicit for LBS and can be used in various scenarios where information has to be queried from untrustworthy proxies. For example, when information is stored in remote databases under control of others. Typically before storing this information it is protected by encryption. To retrieve the encrypted information efficiently and securely a collection mechanism is needed capable of searching in encrypted data while keeping it private. Solutions can be based on indices, where instead of searching in the encrypted data, the actual search is performed in an added index e.g. based on hashes of the encrypted records [84, 85, 150]. Another solution is to use trapdoor encryptions where users are able to perform operations on encrypted data without using the decryption key [2, 19, 76, 144, 155]. Distributing information over multiple servers is a solution where the information is retrieved using a secret sharing protocol between the user and hosts [41, 42, 101]. Solutions based on homomorphic encryptions give the possibility to perform some simple operations on encrypted data without decrypting it first [41, 42, 59]. The downside to these solutions is that they do not consider the usage of an untrustworthy proxy.

Another approach of letting a user collect information from various hosts is Oblivious Transfer (OT), introduced by Rabin [128]. With OT a host (sender) holds a list of n messages $M = (m_1, \ldots, m_n)$ while the user (receiver) wants to learn m_i, $1 \leq i \leq n$ i.e. the i-th message. The problem is that the user does not want to send i to the host, since this will reveal to the host which message the user is querying. The host does not want to send the complete list M to the user. With OT the hosts transfers m_i to the user without knowing i.

Fundamental results with OT by [65], lets the host hold two messages m_0, m_1 while the user has a bit b. The user wants to receive m_b from the host without letting the host learn which message was retreived i.e. the user does not want to reveal the bit b. Another constraint is that only one message can be send from the host to the user. This is called 1-out-of-2 Oblivious Transfer, $\binom{1}{2}$-OT. Further generalization of OT was established by [57] resulting in a 1-out-of-n Oblivious Transfer, $\binom{1}{n}$-OT.

OT is related to Private Information Retrieval (PIR) such as [3, 96, 109]. For example, a single host holds a list of n messages $M = (m_1, \ldots, m_n)$. Identical copies of this list are stored by all other hosts ($\geqslant 2$) while the hosts do not communicate with each other. The user has index i and wants to learn m_i. To achieve this he queries each host such that each host will have no information about the index i. The query to each host is distributed independently of i and therefore each host does not gain any information about i. As established by [57] any non-trivial PIR protocol implies an 1-out-of-n OT. The advantage of PIR schemes is that they have a lower communication complexity than OT. The disadvantage of using PIR is that multiple hosts have to store identical copies of the list.

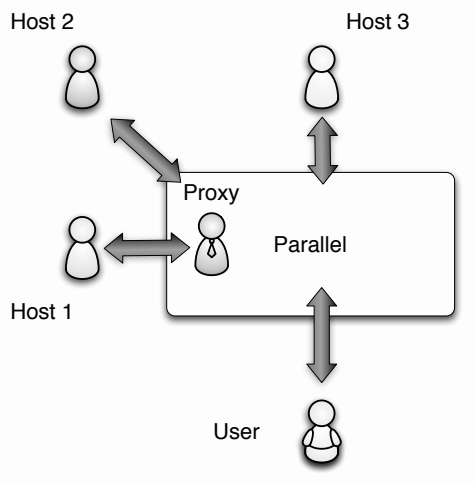


Figure 3.1: Parallel selection and collection Scenario

Approach. In Figure 3.1 a conceptual overview of the parallel selection and collection scenario is given. In the concept of location-based services (LBS) a user accesses various hosts via a mobile device. The goal is to obtain location specific information on topics of which is of interest to the user. This information is collected and served by hosts called service providers. A third party who knows the user's location and acts as a proxy between users and hosts, this can be the mobile operator of the user or an organization associated with it, is responsible for the security of the location information.

Navigation system or other electronic maps can be augmented with online and real-

time information provided by location-based services. This way users can find what they need faster or more accurately. With LBS, location privacy is at stake. To achieve privacy, it is advisable to limit access to identity and location information. Even the regular observation of service usage patterns, might reveal private information.

Today LBS are provided in two ways. Either all the service specific data is made available to the user who computes the result locally, this is the case e.g. with (car) navigation systems; or the service is provided remotely. The latter is the dominant approach for providing LBS in mobile communication networks. Such LBS can be seen as a side product of the GSM system (Global System for Mobile Communications), as the location of subscribers is already used for mobility management [141]. Given these constraints, we aim at achieving privacy for the users who want to use LBS. We presume that the location will be gathered from a mobile network, while the service will be provided by external service providers.

We use cryptographic protocols to ensure privacy: Oblivious transfer and homomorphic encryption. By developing a framework where the users' location and subscription are processed in the encrypted domain, we achieve privacy for location-based services. Unlike classic approaches using Mixes or anonymous credentials [100], our approach achieves the following strong privacy properties: First, the mobile operator learns nothing in addition to what he already knows, with the exception of the set of users that are all interested in using LBS. Thus, no usage profiles can be collected. Second, the service providers only learn the number of subscribers to their service. Thus, service providers do not learn the users' location. In particular, our protocol offers the additional privacy property of service unobservability. Even the service providers do not know whether a user is accessing their service or not.

Threat Model. The parallel selection and collection protocol is build around the user, which is considered to be trustworthy operating from a trusted environment. The host and the proxy are curious adversaries. The protocol is designed to protect the assets of the user i.e. the users' location and the subscription to various hosts. The proxy handles all communication between the user and the host. Being curious, the proxy monitors all traffic during communication and observes all computations being performed for leakage of valuable information. The proxy as attacker would like to learn the information collected by the user and to learn from which host this information was collected. The host as attacker is interested to learn which user is collecting information and what the location is of this user. The proxy is aware of the users' location and could basically leak this information to any other party. Even if the proxy colludes with all hosts none of the parties will learn which information was collected by the user.

Structure. Section 3.2 elaborates on the requirements and gives a high level approach of our privacy protecting parallel selection and collection scheme based on the LBS scenario. The protocol is given in section 3.3. We analyze the efficiency and security of our solution in section 3.4 and finally give conclusions in section 3.5.

3.2 Requirements and High-level Overview

The parallel selection and collection protocol should protect the assets and interests of all involved parties. The assets that need to be protected are: the user's location, the user's query and the databases of the hosts. More precisely, we consider the following requirements:

Location privacy – The protocol should not reveal the user's location to the service.

Query privacy – Even when the proxy and the hosts collude, the secrecy of the user's query should remain protected. This includes message privacy; i.e., only the users can decrypt the messages of the hosts.

Database secrecy – The user should from the databases only learn the requested, location specific, information and nothing more.

In a LBS scenario the users are typically connected via a mobile network to the proxy who obtains the users' location as a result of the GSM handover protocol [140]. Higher location resolutions may be achieved by using triangulation techniques or GPS. The hosts are connected to the proxy over high bandwidth wired line connections. We assume that parties communicate via secure channels, that the proxy and the hosts are able to authenticate to each others, and that they are able to sign messages using their identity. This can for instance be realized using a public key infrastructure (PKI).

3.2.1 Phases

The selection and collection protocol is divided into three phases: Setup, Select and Collect:

1. Setup. In the setup phase the involved parties generate their keys. The user $\mathcal{U}$ needs keys for the encryption of the query and the hosts for the encryption of their database. We assume that the user only queries a single host. Furthermore, each host $\mathcal{L}_j$ encrypts its specific database and transmits it to the proxy.

2. Select. In the select phase a user $\mathcal{U}$ creates an encrypted query and sends it to the proxy. The query consists of ℓ elements, one for each host $\mathcal{L}_j$. Each element indicates in an encrypted form whether $\mathcal{U}$ is querying a host or not. For example, there are three hosts ($\ell = 3$) and we are querying host $\mathcal{L}_2$. Then the selection of the user $\mathcal{U}$ will be: $Q_1, Q_2, Q_3 = E_{pkU}(0), E_{pkU}(1), E_{pkU}(0)$. Eventually the proxy forwards the j^{th} subscription element to the respective host $\mathcal{L}_j$. E_{pkU} is an homomorphic encryption using the public key pkU of user $\mathcal{U}$. The database of a host $\mathcal{L}_j$ is represented as a one-dimensional vector with one element for each location. Let n be the number of locations, then this vector is $m_{(1,j)}, \ldots, m_{(n,j)}$. For simplicity of presentation we assume the same n for all hosts.

3. Collect. In the collect phase the proxy runs a protocol with every host $\mathcal{L}_j$ and

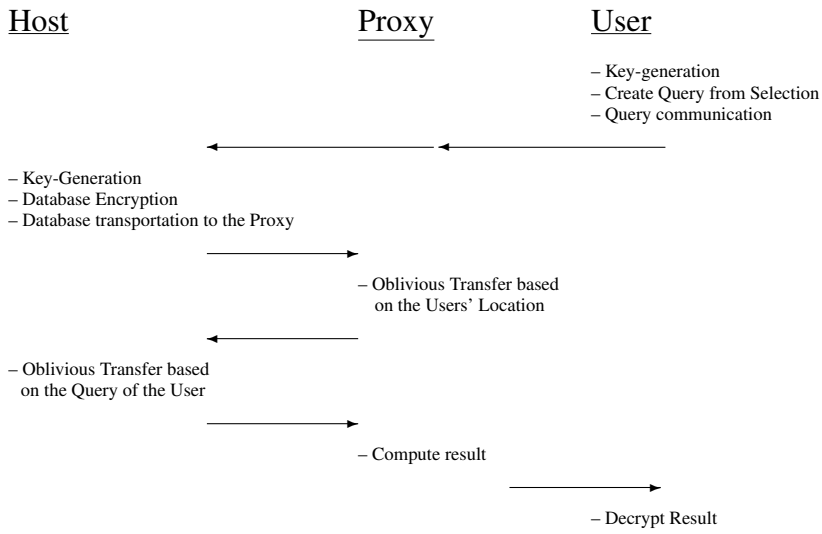
obtains an encrypted result (under the key used by the user). For efficiency the proxy combines all results into a single encrypted result for the user, such that the user only receives the data of the host he was querying.

Notation. We write cryptographic primitives as $\mathsf{Alg}_k(x)$, where x denotes the processed inputs of the algorithm $\mathsf{Alg}(.)$ and k denotes keys.

3.2.2 High-level Overview

We follow a constructive approach in the description of our protocol. We take building blocks from Appendix A, put them into place, and describe their function. Some of the security requirements can be fulfilled by the functionality provided by individual building blocks; others require a complex interplay between building blocks. As a consequence the transformation from building blocks to the sub-protocols of our solution is not one-to-one. In Table 3.1 we sketch our approach of the complete protocol with all the sub-protocols, like Oblivious Transfer, as they get assembled from their building blocks.

Table 3.1: Parallel Selection and Collection Approach



Our main building blocks are homomorphic encryption (Appendix A.1) and two variants of oblivious transfer (OT) protocols (Appendix A.3). Two OT protocols are used, one based on blind signatures and the other based on homomorphic encryptions. The blind signature OT scheme is suitable when the input database remains fixed, while the index varies. The homomorphic encryption OT is efficient in the opposite case; it is easy to use for fixed indices.

During the protocol execution, a single proxy interacts with a multitude of users and multiple hosts. The first building block we put into place is a blind signature based OT protocol. It is executed with the proxy acting as the requester and one of the hosts as the sender. It allows the proxy to retrieve location specific information $m_{\sigma_1,j}$ for a user at location σ_1 without host $\mathcal{L}_j$ learning the user's location. This guarantees *location privacy*. The proxy executes this sub-protocol with all hosts in parallel. This assures *query privacy* at the hosts side. In this way the proxy obtains an information vector $m_{\sigma_1,1},\ldots,m_{\sigma_1,\ell}$; where ℓ is the number of hosts.

Our second building block is a homomorphic encryption based OT protocol. It runs with the proxy acting as the sender (using the aforementioned vector as input) and the user acting as the requester (using the index of the host $\mathcal{L}_{\sigma_2}$ as input i.e. the selected host by the user is indicated as σ_2). Finally, the protocol allows the user to learn m_{σ_1,σ_2} without the proxy learning the user's query; we achieve full *query privacy*.

In the first OT, the same database is queried by the proxy multiple times as the user moves around. Nevertheless, the database needs to be encrypted and transferred to the proxy only once. For the second OT between user and proxy, it is sufficient to send the first (and expensive) query of the homomorphic OT only once.

This gives us a first instantiation of the protocol phases. The outline of the protocol is depicted in Table 3.1. For ease of presentation we use a simplified notation. The detailed protocol description is given in Section 3.3.

1. Setup. User $\mathcal{U}$ generates a public key-pair (pkU, skU) for a homomorphic encryption scheme. These keys are used for the oblivious transfer based on homomorphic encryption (Appendix A.3.2). Every host $\mathcal{L}_j$ generates a public key-pair (pkB_j, skB_j) that is used for OT based on blind signatures (Appendix A.3.1). The database of the host $\mathcal{L}_j$ consists of n elements $m_{(1,j)},\ldots,m_{(n,j)}$. Each of the elements is encrypted with its own symmetric key $H(k_{(i,j)})$ that is computed by hashing the signature $k_{(i,j)} = Sign_{skB_j}(i)$ of the index using the secret key skB_j of each host $\mathcal{L}_j$. The encrypted database $DB_j = (C_{(1,j)},\ldots,C_{(n,j)})$, with $C_{(i,j)} = E_{H(k_i)}(m_{i,j})$ is sent to the proxy.

2. Select. A user's selection consists of ℓ ciphertexts, one for each host, $Q_1,\ldots,Q_{\sigma_2},\ldots,Q_\ell$. The ciphertext Q_j of the homomorphic encryption scheme is obtained by using the public key of the user, E_{pkU}. Q_j is the result of an encryption of 1 for the host $\mathcal{L}_{\sigma_2} = Q_{\sigma_2}$ which the user has selected and of 0 otherwise i.e. hosts which are not selected to be queried. In addition, to ensure the security of the oblivious transfer the user could prove, using zero-knowledge, that the Q_j are constructed correctly.

3. Collect. In the data retrieval phase a user obtains location-specific data from the hosts. The proxy is involved since he is aware of the user's location and stores the encrypted databases of the hosts. Recall that these databases are encrypted using hashed signatures as keys. The proxy acts on behalf of the user and can request decryption of individual items without revealing the location of the user. To guarantee query privacy the proxy has to repeat the following steps for every host $\mathcal{L}_j$:

- The proxy blinds the actual user' location σ_1 using a blinding factor b and sends the blinded value $Blind_{pkB_j}(\sigma_1, b)$ to each host. The hosts reply with the blinded signature $Sign_{skB_j}(Blind_{pkB_j}(\sigma_1, b))$. The proxy then unblinds the users location $Unblind_{pkB_j}(Sign_{skB_j}(Blind_{pkB_j}(\sigma_1, b)), b)$. This will result in $Sign_{skB_j}(\sigma_1) = k_{(\sigma_1,j)}$ which is identical to the symmetric key $H(k_{(\sigma_1,j)})$. Eventually the proxy computes $m_{\sigma_1,j} = D_{H(k_{(\sigma_1,j)})}(C_{(\sigma_1,j)})$, which completes the first OT. The proxy collects $m_{(\sigma_1,1)}, \ldots, m_{(\sigma_1,\ell)}$ and continues with the second OT (the user's first message is taken from the subscription). The proxy takes each Q_j and computes for all $1 \leq j \leq \ell$, $M_j = Q_j^{m_{(\sigma_1,j)}}$. This corresponds to an encryption of $m_{(\sigma_1,\sigma_2)}$ for $\mathcal{L}_{\sigma_2}$ and an encryption of 0 otherwise.
- As a last step the proxy combines the M_j by homomorphically adding all the encryptions, not knowing which of them contain the message. This way all encryptions of 0 cancel out. The result is transferred to the user, who decrypts M_j to obtains $m_{(\sigma_1\sigma_2)}$.

The main flaws of this simple construction is the fact that the proxy learns the $m_{i,j}$ vector for the locations of the user. This is a compromise of *database secrecy*.

3.2.3 Database Secrecy revision

We address the lack of *database secrecy* by intertwining the first OT with the second. To this end we let the proxy pass on Q_j to $\mathcal{L}_j$. Now (after agreeing on who sends which bit range) both $\mathcal{L}_j$ and the proxy can act as senders in the second OT without learning each others inputs. This is made possible by the properties of homomorphic encryption, which lets everyone manipulate encrypted data. Informally, the last message of the first OT will be transferred as part of the encrypted payload of the second OT. This guarantees that only the user with her secret decryption key can obtain the results of both protocols.

More concretely the following changes have to be made in the select and collect phases. During the collection the proxy blinds the location σ_1 and sends the blinded value $Blind_{pkB_j}(\sigma_1, b) = \sigma_{1,j}$ to the hosts. To ensure that only the user (and not the proxy) can decrypt $C_{(\sigma_1,j)}$, the hosts encrypt the blinded signature $Sign_{skB}(\sigma_{1,j})$. This is done with an additive homomorphic encryption scheme. Remember that during setup the user (through the proxy) provided the host $\mathcal{L}_{\sigma_2}$, to which he wants to collect information from, with an encryption $Q_{\sigma_2} = E_{pkU}(1)$. The host $(j = \sigma_2)$ computes $M_{\sigma_2} = Sign_{skB_{\sigma_2}}(\sigma_{1,\sigma_2}) \otimes Q_{\sigma_2} = E_{pkU}(Sign_{skB_j}(\sigma_{1,\sigma_2}) \cdot 1) = E_{pkU}(Sign_{skB_j}(\sigma_{1,\sigma_2}))$. These requests are done for all hosts, including those the user did not select. The latter receives $Q_j = E_{pkU}(0)$ and all the operations result in $M_j = E_{pkU}(0)$, for $j \neq \sigma_2$. All results are sent to the proxy who adds the blinding factor b and corresponding database entries $C_{(\sigma_1,j)}$ to each message M_j.

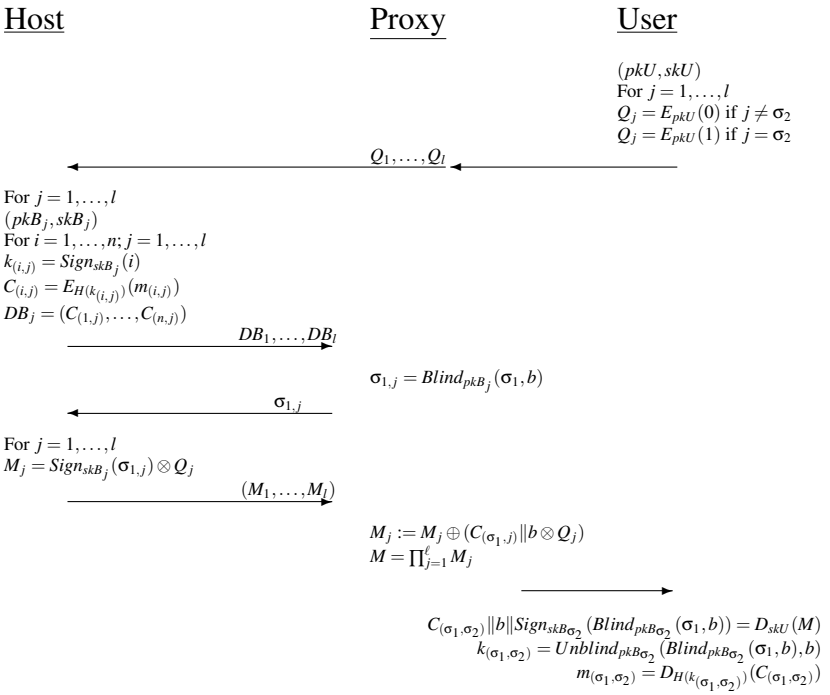
As a last step the proxy computes the homomorphic sum of all encryptions; not knowing which of them contain the unblinding information, the encrypted message, and the blinded signature. This way all encryptions of 0 cancel out. The result is

transferred to the user, who decrypts M, obtains $b\|C_{(\sigma_1,\sigma_2)}\|Sign_{skB_{\sigma_2}}(\sigma_{1,j})$, computes $H(Unblind_{pkB_{\sigma_2}}(Sign_{skB_{\sigma_2}}(Blind_{pkB_{\sigma_2}}(\sigma_1,b)),b)) = H(k_{(\sigma_1,\sigma_2)})$, and finally computes $m_{(\sigma_1\sigma_2)} = D_{H(k_{(\sigma_1,\sigma_2)})}(C_{(\sigma_1,\sigma_2)}) = D_{H(k_{(\sigma_1,\sigma_2)})}(E_{H(k_{(\sigma_1,\sigma_2)})}(m_{\sigma_1,\sigma_2}))$.

3.3 Parallel Selection and Collection Protocol

We will describe the complete parallel selection and collection protocol in details based on the high-level approach from section 3.2 together with the database secrecy revision from the same section. The complete protocol is also shown in Table 3.2, which shows the execution and communication flows between the user, proxy and hosts.

Table 3.2: Parallel Selection and Collection Protocol



1. Setup. The protocol starts with the generation of various key pairs with public key pk and private key sk. Every host $\mathcal{L}_j$ generates the key pair (pkB_j, skB_j) for the unique blind signature protocol. The proxy does not need to generate keys. The user $\mathcal{U}$ generates a key pair (pkU, skU) for the additive homomorphic Damgård-Jurik cryptosystem [52] used for homomorphic OT [123].

Each host $\mathcal{L}_j$ encrypts its location specific information $m_{(1,j)},\ldots,m_{(n,j)}$. A host $\mathcal{L}_j$

uses his secret key skB_j to compute $DB_j = (C_{(1,j)}, \ldots, C_{(n,j)})$:

$$k_{(i,j)} = Sign_{skB_j}(i)$$
$$C_{(i,j)} = E_{H(k_{(i,j)})}(m_{(i,j)})$$

The cryptographic hash function H is used for computing the symmetric key. Each resulting database DB_j is sent to the proxy i.e. $DB_1, \ldots, DB_l$.

2. Select. User $\mathcal{U}$ must select a host $\mathcal{L}_{\sigma_2}$ to receive location related information from and proceeds as follows:

1. $\mathcal{U}$ uses his public key pkU to compute the query Q_j, $1 \leq j \leq \ell$:

$$Q_j = \begin{cases} E_{pkU}(1) & \text{if } j = \sigma_2 \\ E_{pkU}(0) & \text{otherwise} \end{cases}$$

   The Q_j items are used to request the location specific information from the host $\mathcal{L}_{\sigma_2}$.
2. The resulting $Q_1, \ldots, Q_\ell$ are sent to the proxy.

The proxy passes each query Q_j on to the respective host $\mathcal{L}_j$.

3. Collect. The proxy runs a set of algorithms together with the hosts $\mathcal{L}_j$ to request location specific information for user $\mathcal{U}$. The input of these algorithms is the database DB_j and the current location σ_1 of the user $\mathcal{U}$. The proxy's output is an encryption of $m_{(\sigma_1,j)}$ based on the location of the user. The following is computed:

1. The proxy chooses a random b and computes

$$\sigma_{1,j} = Blind_{pkB_j}(\sigma_1, b)$$

   The random blinding factor b hides the location σ_1 of the user in $\sigma_{1,j}$ for each host.

2. The proxy sends $\sigma_{1,j}$ to $\mathcal{L}_j$, $1 \leq j \leq \ell$, which computes

$$M_j = Sign_{skB_j}(\sigma_{1,j}) \otimes Q_j$$

3. Every host $\mathcal{L}_j$ sends M_j back to the proxy.

4. The proxy enriches M_j with $C_{(\sigma_1,j)}$ and b. This is done by computing $M_j := M_j \oplus (C_{(\sigma_1,j)} \| b \otimes Q_j)$. This only changes the content of M_j if Q_j is an encryption of 1.

After the computations by every $\mathcal{L}_j$ and receiving the corresponding M_j, the proxy computes $M = \prod_{j=1}^{\ell} M_j$. This will combine all the encryptions into one message using the homomorphic property of the cyphertext.

Finally the user decrypts M by computing the following algorithms:

$$C_{(\sigma_1,\sigma_2)} \| b \| Sign_{skB_{\sigma_2}}(Blind_{pkB_{\sigma_2}}(\sigma_1, b)) = D_{skU}(M)$$
$$k_{(\sigma_1,\sigma_2)} = Unblind_{pkB_{\sigma_2}}(Blind_{pkB_{\sigma_2}}(\sigma_1, b), b)$$
$$m_{(\sigma_1,\sigma_2)} = D_{H(k_{(\sigma_1,\sigma_2)})}(C_{(\sigma_1,\sigma_2)})$$

3.4 Security and Efficiency

3.4.1 Efficiency analysis

For our efficiency analysis we focus on two main factors. The first is the limited computation and communication resources when users are using mobile devices. Although mobile devices are becoming more powerful we see (through cloud based computing) that most computation is done on the server side i.e. the hosts and the proxy. The other factor is scalability for the hosts with respect to location resolution, i.e. number n of map cells and the number of ℓ hosts.

User. The costs incurred by the user are key generation and decryption. Key generation involves the generation of a single key. Decryption requires a single Damgård-Jurik decryption. For low power devices with limited bandwidth, we suggest to do the key generation at the user's desktop PC. After the key generation, the private key $sk_{\mathcal{U}}$ can be synchronized to the mobile device of the user. For additional security, when the public and private key-pair where generated on the mobile device, the public key $pk_{\mathcal{U}}$ can be moved to the user's desktop PC.

Hosts. The system scales optimally with multiple users since the key setup and database encryption are independent of the number of users. Furthermore, database encryption is only linear in the number of locations. Queries are linear in the number of hosts; optimal as all hosts need to be involved to guarantee query privacy. Practically data transfer is independent of the number of locations and again only depends on the number of hosts.

The most prominent cost brought upon by the host is database encryption requiring one signature operation per location. The most dominant cost for $\mathcal{L}_j$ and proxy is the transmission of the encrypted database DB_j. Moreover, these costs are brought upon whenever any of the $m_{(i,j)}$ should be updated; since then the whole database has to be updated and retransmitted. By relaxing the database security requirement the host could only update a subset of updated locations and communicate this with the proxy.

3.4.2 Security Analysis

Our main goal is to implement location-based services without revealing additional information about the user's location and his/her service usage i.e. query. More pre-

cisely, an adversary involved in our protocol should learn nothing except what he already could have learned by being involved in a scenario without location-based services. For the proxy this implies that he is allowed to know the user's location but should not learn anything about the query of the user. For hosts this implies that both the user's location and the user's query have to be concealed.

The proxy helps to solve fairness conflicts between users and hosts. This is supported by considering the rationality of the proxy: his core competence is setting up the communication in such a way that hosts and users can communicate in a fair way. Cheating or not co-operating in resolving fairness disputes decreases his reputation.

In cases where accusing the cheater would endanger the users service privacy, we could make use of a privacy trustee as an additional off-line trusted party. The usage of a trusted third party to resolve fairness problems is common in the literature of fair exchange protocols. In fact, [6, 7] have shown that the problem of fairly exchanging data requires at least an off-line trusted party.

Location privacy. In our protocol hosts only get in contact with location related information in the collection phase. However, there the OT based on blind signatures protects location related information from being revealed unintentionally. The security of the OT scheme is based on the signature's blindness property. This property guarantees that for any host viewing message M and message M' is computationally indistinguishable i.e. the two messages are effectively the same. As the user's location σ_1 is hidden, the location privacy of our protocol can be straightforward reduced to the blindness property of the used blind signature scheme.

Query privacy. Achieving query privacy is more challenging. This has two reasons: First, the relationship user/host plays a role during the collection of data. Second, we consider a stronger, but realistic, adversary model and allow for a malicious proxy, who possibly collaborates with any host. This implies that the query privacy cannot rely on the help of the proxy. Therefore, in the selection phase, the user's query is protected by the semantic security of the underlying homomorphic encryption scheme i.e. it is infeasible for a computationally-bounded adversary to derive significant information about the plaintext given the ciphertext and corresponding public key. The semantic security guarantees that two different queries are indistinguishable.

Database secrecy. In contrast to location and query privacy the database secrecy protects the interest of the hosts. It guarantees that a user gets no more than the requested information even if he collaborates with the proxy. The database secrecy of our scheme, fundamentally, relies on two aspects: First, the host encrypts his database before he sent it to the proxy. Second, as a result of the collection phase the user only gets to know the requested data. This is due to the so called 'Database security' [121] of the underlying secure oblivious transfer protocol.

Threat Model. Based on the threat model in Section 3.1 the adversaries (proxy and hosts) are unable to learn more information, other than the information they already knew. The proxy already knows the location of the user, but does not learn which

host the user is querying and does not learn which information is retrieved from the hosts. The probability of the proxy guessing from which host the user is retrieving information from decreases if the amount of hosts increase due to the combination of the oblivious transfer using blind signatures and the homomorphic encryption during collection. Even if the proxy would reveal the location of the user to all the hosts (i.e. all adversaries collude), the proxy and the hosts will be unable to learn which information has been queried by the user due to the homomorphic encryption. An individual host remains oblivious to the information being queried from his database unless he colludes with the proxy. Even if the host knows the location of the user, the host remains oblivious in knowing if he has been queried by the user or not.

Complexity. We measure the strength of the protocol by analyzing the computational complexity of the cryptographic techniques and algorithms used. Assuming the worst-case scenario, where the proxy colludes with the hosts which implies that the location of the user is compromised; then the security of the protocol is solely based on the homomorphic encryption. Since Paillier is used for his additive homomorphic properties (Appendix A.1.3), the complexity of breaking the parallel selection and collection protocol is as hard as factoring the modulus n.

3.5 Discussion

We presented the first privacy-preserving parallel selection and collection protocol based on cryptographic techniques, namely, on oblivious transfer and homomorphic encryption. The privacy of the user is protected by hiding the user's location from the services and by not revealing information on the user/host relationship.

Various privacy enhancing technologies (PET) have been proposed for LBS. Most of these techniques focus on providing pseudonymity and anonymity for LBS. Federath *et al.* [66] proposed the use of a trusted fixed station and Mixes [33] for hiding the linkage of real world identities to location data in GSM networks. While our protocol can be adapted to such a privacy enhanced GSM network by letting the fixed station localize the user, we explicitly focus on the less privacy friendly but more practical setting where a third party knows the users location.

Researchers started to develop LBS specific PETs called mix-zones (see [15] and [83]). Mix-zones allow users to switch pseudonyms associated to their location in an unobservable way. Kölsch *et al.* [100] use pseudonymization techniques in the following realistic setting. A network operator (or a party connected to multiple operators) knows the users location, while the LBS are provided by independent service providers that know the user only under short lived pseudonyms. Unlike them we do not rely on the anonymization of location data disclosed to services. Basically, as location information is inherently attached to a persons life, reidentification is often easy. Location needs to be hidden, not anonymized.

Chapter 4

Sequential Selection and Collection

This chapter addresses the selection and collection problem using a sequential method where information sources (hosts) are selected by choosing a fixed itinerary and ensuring that collected information is passed on from host to host. Since only the first and last host of the itinerary should communicate with the proxy, communication is minimal. To overcome the problem of ending the itinerary when hosts are unreachable, a helper protocol is introduced which is capable of evading hosts from the itinerary, when needed.

4.1 Introduction

Searching, comparing and buying e.g. airline tickets on the Internet is still a major problem, especially if this task should be done on a mobile device. It takes time to query multiple websites and to compare the variety of offers e.g. from airline companies. Although mobile devices are getting more powerful, most computation is still performed on the server side.

In chapter 3 information was collected from various hosts using a proxy which queried each host individually i.e. parallel selection and collection. That approach made it possible to keep the proxy oblivious to the information that was collected and oblivious from which host it was collected. A different approach for collecting information from various hosts is to query each host independently based on a fixed path i.e. itinerary. Only when all hosts have been visited the collection process returns to the proxy for

The research for this chapter has been carried out in close collaboration with M. van Hensbergen MSc. student of the Delft University of Technology, now with Fox-IT. The results of this chapter have been published in [73] and [87].

the final result. Comparable to a lorry collecting packages at various locations before driving back to the distribution center.

This so-called sequential selection and collection approach is based on hash chaining together with symmetric and asymmetric encryption. Hash chaining is a method of producing one-time passwords [103], but also suitable for the collection of information from a fixed number of hosts [143]. The combination results in a sequential selection and collection approach where the risk of loosing private data is minimized, the collected information is kept private and the user remains in control over the predefined hosts that should be visited.

Motivation. One of the classic approaches to the sequential problem are based on mobile agents and mobile code operating in untrustworthy environments [30]. Mobile agents are autonomous pieces of software that run on remote hosts in order to carry out a task on behalf of its user. The code of the agent itself, together with some other data, is transported via the Internet as opposed to data in traditional communication. Usually these agents are considered in an e-commerce scenario (e.g. buying airline tickets) where the goal is to purchase a desired item for a user. The agents visit a number of hosts that may or may not sell the tickets in question, ask them for an offer and eventually decide which offer is the best. The best offer is signed digitally by the agent to commit itself to the offer made by the host.

Mobile agent technology is a blending of a number of technologies, in particular artificial intelligence and mobile code. Mobile code is the transfer of code, from one host (sender) to another (receiver), which is executed on the recipient's side. Examples are Java Applets, which are transferred from a website to the user's computer through a web browser. It is often used in cases where bandwidth limitations make it more efficient to send the code together with the data rather than executing the code on the server hosting the website. In combination with artificial intelligence, mobile code may be more autonomous and may travel a more complex path to execute more complex functions. For example, mobile agents can be used in Multi-Hop MANETs [40]. Multi-Hop MANETs are comparable to the sequential problem since all communication is passed on from host to host (hops) without having to return to the proxy.

As discussed in chapter 2, the execution of software on the recipient's computer possesses a number of security issues. Execution of 'untrustworthy' code on the recipients computer can contain a virus, worm or a trojan, and can potentially harm the users 'trusted' environment. Techniques for mitigating these threats are for example sandboxing [102] or using certificates [145]. Sandboxes are mechanisms for running programs in a isolated environment within the host. It is a valid security measure for the host which do not trust the code to be executed on their system, but it does not provide a solution for executing trusted code in an untrustworthy environment. The disadvantage of using certificates is that an additional trusted party is needed to act as Certificate Authority (CA) for all parties. This trusted party is able to verify the code which needs to be executed on the remote host.

Karjoth *et al.* [95] defined a number of security properties on how mobile agents need to protect collected information against adversaries. Information is usually collected

as a chain of encapsulated pieces of information with security properties such as: confidentiality, forward privacy, forward integrity, or non-repudiation. A number of solutions have been proposed based on public key cryptography, digital signatures and hash chaining [95, 110, 160, 161], while some of these protocols possess weaknesses [133].

Sander and Tschudin wrote various papers on the execution of mobile code in untrustworthy environments [138, 137, 136]. They introduced functions which can be executed in encrypted form based on homomorphic cryptosystems.

In the work by Singelée and Preneel [143] an e-commerce problem with untrustworthy hosts has been addressed using the concept of mobile agents. The proposed scenario is similar to the sequential problem and the solution uses hash chaining techniques from [95]. The limitations of these solutions is that they are based on multiple mobile agents e.g. for the distributed protection of the private keys. It does not provide a complete solution when a single agent would be used. This limitation, it provides a firm basis for the development of a new sequential selection and collection approach as will be described in this chapter.

The above mentioned literature is mostly presented within a mobile agents scenario. Since this thesis does not focus on the concept of intelligent and autonomous mobile agents, we therefore omit mobile agents in our solution as presented in this chapter.

Structure of the chapter. Section 4.2 gives the definitions of the parties, the model and the requirements. The general approach to the problem is described in section 4.3. The sequential selection and collection protocol is given in section 4.4. For more understanding an example is given in section 4.5. We analyze the protocol in section 4.6 and finally conclude in section 4.7.

4.2 Definitions

In this section we present the parties, the general model and the requirements of our sequential selecting and collecting protocol.

4.2.1 Model

The issue with visiting hosts is that they are in principle untrustworthy and therefore could tamper with the correct execution of the protocols. Figure 4.1 shows a schematic overview of the sequential selection and collection model. The model is build around three entities: the user $\mathcal{U}$, a proxy $\mathcal{P}$ and hosts $\mathcal{L}_1, \ldots, \mathcal{L}_\ell$. The user $\mathcal{U}$ is situated in a trusted environment where he is able to execute the protocols securely and privately.

The proxy $\mathcal{P}$ is a curious party which will execute the protocols correctly and therefore will not tamper with the protocol, but it is interested in the private information. The hosts $\mathcal{L}_1, \ldots, \mathcal{L}_\ell$ are considered malicious. They are able to alter the protocol,

choose to partially execute the protocol or not participate at all. Protocols executed by hosts should not be able to leak any private information. It is impossible to prevent tampering of the hosts, but it should be possible to detect misuse.

The itinerary is a list of hosts that the user $\mathcal{U}$ wants to visit in a particular arrangement in order to obtain information. The itinerary is executed by the proxy $\mathcal{P}$ and furthermore carried out by the hosts $\mathcal{L}_1, \ldots, \mathcal{L}_\ell$. The itinerary ends at the proxy $\mathcal{P}$ and the result of the query is sent to the user $\mathcal{U}$ by proxy $\mathcal{P}$.

Threat model. The threat model of the sequential selection and collection protocol is based on the trust we have in the involved parties. The user is trusted and operating from a trusted environment. The proxy is untrustworthy and curious regarding the communication and computation being performed. The hosts are active attackers capable of manipulating the computation and communication. They are capable of performing replay attacks on the protocol or clone the data in order to gain information. Hosts can attack the itinerary to retrieve information of the hosts to visit, which could be used against the itinerary of the user e.g. to prevent the protocol in reaching a specific host. Colluding hosts would not gain much information, other than they already know, unless they also involve the proxy. If the itinerary has returned to the proxy, a host colluding with a proxy could gain more information then wanted. In that case, security would fail.

4.2.2 Requirements and Assumptions

The goal of our approach is to protect the resources and private information of all involved parties. Therefore, the solution to our model should fulfill the following

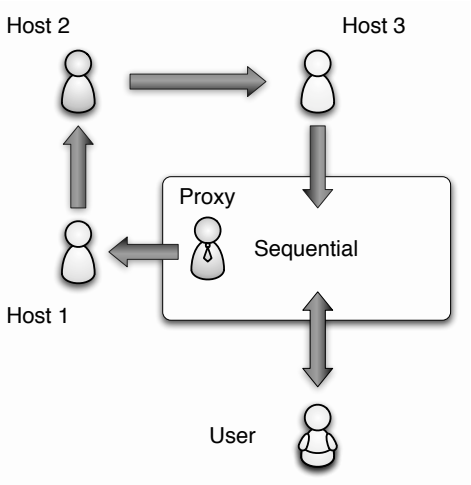


Figure 4.1: Sequential approach

requirements:

1. The collected information from the hosts should be hidden for other hosts. For example, in the case of collecting offers from airline companies, these companies should not be able to offer a price for a ticket based on already collected offers from other companies. The goal is therefore to have confidentiality of the retrieved offers.
2. The itinerary should not reveal the full list of hosts which are planned to be visited.
3. The amount of network traffic the protocol generates should be minimized. This holds especially w.r.t. the interaction with the users due to the use of a mobile device which has minimum computational power.
4. The final requirement is related to the tampering of the malicious hosts. One cannot prevent tampering of the protocols by the hosts since the host has full control of the protocol executed in his private environment. But, any host should be able to detect malicious activity.

The assumptions to our model are:

1. A Public Key Infrastructure (PKI) is available to all parties to authenticate the communication by identifying the public keys of various parties i.e. to ensure secure communication channels.
2. This model does not consider a denial of service attack.
3. Hosts do not collude with each another.

4.3 Approach

The approach of this chapter is based on splitting up tasks (collect and decide). By letting the hosts $\mathcal{L}_1, \ldots, \mathcal{L}_\ell$ collect information from each other, the private information of the owners (hosts) cannot be revealed, because the hosts are not able to process collected information. For example, hosts are unable to match their provided information with previously collected information. This decision process in only available to the proxy $\mathcal{P}$. The gathering of information is a rather neutral activity since hosts are only used for the collection of information, while the decision logic is kept away from these untrustworthy environments i.e. hosts.

There is also a helper protocol involved, which will be invoked when the protocol cannot be sent to the next host. This helper protocol will reside within the proxy $\mathcal{P}$. The helper protocol is only needed when a host $\mathcal{L}_j (1 \leq j \leq \ell)$ does not function, otherwise the helper protocol will just shut down after a pre-determined time.

The groundwork of our approach is based on [143] which present a scenario in which data can be collected securely, where collected data can be protected against untrustworthy hosts and where transaction can be digitally signed in untrustworthy environ-

ments. In order to protect different aspects of our selection and collection protocol a variety of cryptographic techniques and concepts is needed. The encryption of messages m_1 and m_2 with a symmetric encryption algorithm using key K_j of host $\mathcal{L}_j$ is denoted as $E_{K_j}(m_1, m_2)$, whereas the messages are concatenated denoted by the comma. The encryption of m_1, m_2 using the public key of a particular host $\mathcal{L}_j$ is denoted as $E_{pk_{\mathcal{L}_j}}(m_1, m_2)$. The digital signature of m_1, m_2 by a host $\mathcal{L}_j$ will be represented by $Sign_{sk_{\mathcal{L}_j}}(m_1, m_2)$.

Itinerary Security. The itinerary is based on an end-to-end encryption where in each step a layer is revealed, with the result that the size of the itinerary gets smaller. The itinerary should be controlled to ensure that every host is visited and that the order of visit is controlled. The itinerary should be protected in that sense that the location of the hosts is not revealed prior of visiting the host and that no additional hops can be added to the itinerary. The approach in this chapter for itinerary security is based on the hash chaining method described in Appendix A.4.

Threshold Signature Scheme. The proxy needs to sign a message at the end of the itinerary. Therefore a threshold signature scheme is chosen, where the signing key is split into two parts. Each party in the signature process can use its share to create a partial signature on a message. These partial signatures can then later be combined to create one complete signature. In our model we let the hosts partially sign their messages all with their secret signing key given by the itinerary. Once the proxy receives all messages, it creates a partial signature and combines the two partial signature to one complete signature. This way, the complete signing key is never reconstructed in one place and can therefore not be misused by the hosts or the proxy.

4.4 Sequential Selection and Collection Protocol

The sequential selection and collection protocol is divided into three phases. The initial phase is the initialization and selection treated in section 4.4.1 where all the necessary information is selected, prepared and send to the appropriate parties. Section 4.4.2 deals with the collection of information and how the protocol acts when the itinerary is unable to reach a host. Finally, section 4.4.3 explains how the proxy and the user process the collected information.

4.4.1 Initialization and Selection

For each initiation of the sequential selection and collection protocol the following three sub-protocols need to be executed: the itinerary protocol, the proxy protocol and the helper protocol.

Itinerary protocol. User $\mathcal{U}$ chooses a fixed itinerary $\mathcal{L}_1, \ldots, \mathcal{L}_\ell, \mathcal{P}$ where $\mathcal{L}_j (1 \leq j \leq \ell)$ are hosts, ℓ is the total amount of hosts to visit and $\mathcal{P}$ the proxy i.e. the final destination of the itinerary.

For each host $\mathcal{L}_j$ to be visited a symmetric encryption key K_j is generated by the user. When host $\mathcal{L}_j$ receives the itinerary, it is able to extract his designated symmetric encryption key K_j using his secret asymmetric key $sk_{\mathcal{L}_j}$. Next, the symmetric encryption key is used to expose the location of the following host to be visited from the itinerary. In addition, the itinerary is reduced and only contains information accessible by the next host.

A random nonce n_j is generated for every host $\mathcal{L}_j$, where n_j is used within the helper protocol. The nonce is used at the end of the selection and collection protocol in order to verify which host was skipped during the itinerary.

Using a $(2,2)$-threshold signature scheme between every host and the proxy the signing key is split into two parts, one for each host s_j and one for the proxy $s_{\mathcal{P}}$. Splitting of the signing keys can vary on the scheme used. We ensure that the proxy signing key $s_{\mathcal{P}}$ remains the same with all signing key-pairs. We compute for every host $\mathcal{L}_j$:

$$(s_j, s_{\mathcal{P}}) \tag{4.1}$$

The user generates the itinerary by starting with the itinerary entry of the last party to be visited: the proxy $\mathcal{P}$:

$$I_{\mathcal{P}} = [E_{pk_{\mathcal{P}}}(K_{\mathcal{P}}, s_{\mathcal{P}})] \tag{4.2}$$

At the end of the itinerary the proxy $\mathcal{P}$ will be able to decrypt the symmetric key $K_{\mathcal{P}}$ by using his private asymmetric key $sk_{\mathcal{P}}$. The rest of the itinerary is computed iteratively for the remaining hosts to visit, $\mathcal{L}_j$ where $j = \{\ell, \ldots, 1\}$:

$$I_j = [E_{pk_j}(K_j, s_j), E_{K_j}(Q_j, \mathcal{L}_{j+1}, I_{j+1})] \tag{4.3}$$

Where $\mathcal{L}_{\ell+1} = \mathcal{P}$, $I_{\ell+1} = I_{\mathcal{P}}$ and $s_j, s_{\mathcal{P}}$ are the secret signing keys of the hosts and the proxy. The itinerary is constructed like an onion. Each host removes a layer of the itinerary to uncover: the query Q, the next host $\mathcal{L}_\ell$ or $\mathcal{P}$ to sent the itinerary to, and the remainder of the itinerary I.

By using an onion like approach the privacy of the itinerary can be achieved. Since hosts only learn which hosts have sent the itinerary to them and which host they should sent the itinerary to, they do not learn the complete itinerary and also not the order of the itinerary. All parties know the location of the proxy $\mathcal{P}$ since every host should be able to skip the next host in the itinerary by contacting the proxy. The advantage of this approach is that the itinerary should be executed in a fixed order to be completed successfully.

The protocol can only finish correctly if the itinerary is completed in the correct order, because only then the proxy $\mathcal{P}$ will receive the correct symmetric key to decrypt the encrypted storage, which is used to verify the complete process.

Every host should receive his part of the itinerary separately, if the itinerary would not use the onion construction. The disadvantage of not using the onion construction is that the user looses control of the hosts to be visited. The user looses control of the order in which the hosts are visited. He looses privacy of the complete itinerary and the amount of communication with the proxy grows. Finally, the complete itinerary I_1 is sent to the proxy $\mathcal{P}$.

Proxy protocol. The user creates an encrypted storage using the symmetric encryption key $K_{\mathcal{P}}$ of the $\mathcal{P}$. The encrypted storage is:

$$E_{K_{\mathcal{P}}}(\mathcal{L}_1, n_1, \mathcal{L}_2, n_2, \ldots, \mathcal{L}_\ell, n_\ell, F) \tag{4.4}$$

F is the function that is performed by the proxy over the offers made by the hosts. $\mathcal{L}_j$ holds the name, location or credentials of the host.The encrypted storage can only be accessed by the symmetric key which becomes available to the proxy when the itinerary has visited all hosts. Using his private asymmetric key $sk_{\mathcal{P}}$, the proxy is able to decrypt equation 4.2 and gain access to this symmetric key.

The advantage of this approach is that the proxy does not learn anything about the itinerary before the initiation. Prior knowledge of the itinerary by the proxy could be used to alter the host to visit and creating a privacy concern regarding the requirements of the user.

Helper protocol. When a host is unable to reach the next host in the itinerary, we initiate a helper protocol using a look-up table. We want to make sure that the look-up table 4.1 is not used by any adversary or by the proxy as an oracle that provides any information for manipulating the collection process.

Table 4.1: Helper protocol lookup-table

Verification	**Value**
$h(E_{K_j}(\mathcal{L}_{j+1}))$	$E_{pk_j}(K_{j+1}, n_{j+1})$

The host has decrypted his symmetric key K_j from the itinerary and has learned the location of the next host $\mathcal{L}_{j+1}$ to sent the itinerary to. In case the host is not responding he decides to contact the helper protocol. To verify that the host is the preceding host in the itinerary, the host sends $E_{K_j}(\mathcal{L}_{j+1})$ and $\mathcal{L}_{j+1}$ to the proxy. The proxy verifies whether $\mathcal{L}_{j+1}$ is inactive and verifies the encrypted value that was sent by comparing it with the verification column in the table.

Since only the host knows which host he can visit next, he is the only one able to generate $E_{K_j}(\mathcal{L}_{j+1})$. After verification, the user sends the value to the host which can use it to decrypt it and use the new symmetric key to recover the query Q_{j+2}, the next host to visit $\mathcal{L}_{j+2}$ and the remainder of the itinerary. This construction ensures that the proxy or any other who gets access to the lookup table is unable to use it as an oracle.

4.4.2 Collection

After the user has selected his itinerary and has executed the three protocols discussed in section 4.4.1, all generated information is sent to the proxy. The proxy selects which information is relevant for him and the rest is sent to the first host to visit. This will be known by the proxy since he needs to know where to sent the information to.

During the information collection process the hosts will use the collection protocol and sent the appropriate information to the next host. If a host is not reachable by another host, that host is able to initiate the helper protocol. The explanation on how to use the helper protocol is given in 'Using the Helper protocol'.

Collection protocol. Every host executes the following protocol (after having received the itinerary):

1. Host $\mathcal{L}_j$ uses his private key sk_j to decrypt the first element of I_j as given in equation 4.3, to obtain his symmetric key K_j. This, so-called session key, can be used to decrypt the second element of I_j to obtain the personal query Q_j of the user, the location $\mathcal{L}_{j+1}$ of the next host and the remaining part of the itinerary I_{j+1}.
2. The host $\mathcal{L}_j$ reads the query Q_j and decides to respond with the message m_j. Since the message is only intended for the user it is encrypted with the users public key $E_{pk_{\mathcal{U}}}$ resulting in $E_{pk_{\mathcal{U}}}(m_j)$.
3. The encrypted message is then signed by the user using the partial signing key s_j. The signed message $E_{pk_{\mathcal{U}}}(m_j)$ is denoted as $c_{j,1} = Sign_{s_j}(E_{pk_{\mathcal{U}}}(m_j))$.
4. The encapsulated message M_j is then computed by the host:

$$M_j = Sign_{sk_{\mathcal{L}_j}}[E_{pk_{\mathcal{P}}}(E_{pk_{\mathcal{U}}}(m_j), c_{j,1}), H_j] \tag{4.5}$$

$$H_j = h(M_{j-1}, \mathcal{L}_{j+1}) \tag{4.6}$$

   whereby, $h(\cdot)$ is the chain relation (see Appendix A.4) of the previously encapsulated message M_{j-1} together with the next host to be visited $\mathcal{L}_{j+1}$.
5. If necessary the helper protocol is used by the host as described in the next paragraph.
6. If no helper protocol was used M_0 to M_j and H_1 to H_j are sent to the next destination on the itinerary. When the helper protocol was executed M_0 to M_{j+1} and H_1 to H_{j+1} are sent.

Using the Helper protocol. When the next host in the itinerary is unreachable the helper protocol can be used by the host to skip one host.

1. Host $\mathcal{L}_j$ learns that host $\mathcal{L}_{j+1}$ is unreachable.

2. Since host $\mathcal{L}_j$ has the knowledge of K_j and the location of the next host $\mathcal{L}_{j+1}$ to be visited he computes $h(E_{K_j}(\mathcal{L}_{j+1}))$ and sends this to the proxy $\mathcal{P}$.
3. The proxy verifies that host $\mathcal{L}_{j+1}$ is down, offline, unreachable or unwilling to participate.
4. The proxy verifies whether $h(E_{K_j}(\mathcal{L}_{j+1}))$ is part of the helper protocol lookup-table.
5. If the verification is correct he sends the corresponding value $E_{pk_j}(K_{j+1}, n_{j+1})$ to the host. If not, the protocol is aborted.
6. With this knowledge host $\mathcal{L}_j$ is able to compute the following encapsulated message

$$M_{j+1} = Sign_{sk_{\mathcal{L}_j}}[E_{pk_{\mathcal{P}}}(n_j), H_{j+1}] \tag{4.7}$$

$$H_{j+1} = h(M_j, \mathcal{L}_{j+2}) \tag{4.8}$$

7. Finally, M_0 to M_{j+1} and H_1 to H_{j+1} are sent to the next host $\mathcal{L}_{j+2}$.

4.4.3 Finalization

Decision protocol. Once the itinerary has reached the proxy $\mathcal{P}$, the proxy will be able to finalize the selection and collection protocol.

1. Proxy $\mathcal{P}$ uses his private key $sk_{\mathcal{P}}$ to decrypt the final element of the itinerary $I_{\mathcal{P}}$. This reveals the session key $K_{\mathcal{P}}$ and the proxy signing key $s_{\mathcal{P}}$.
2. The encrypted storage $E_{K_{\mathcal{P}}}(\mathcal{L}_1, n_1, \mathcal{L}_2, n_2, \ldots, \mathcal{L}_\ell, n_\ell, F)$ can now be decrypted using the session key $K_{\mathcal{P}}$. This gives the proxy knowledge of the itinerary that was followed.
3. The proxy decrypts all encapsulated messages M_j and verifies that they where signed by the correct party. If verification fails, the user $\mathcal{U}$ is informed and the protocol is aborted.
4. Since the proxy knows which party was skipped during the itinerary, the proxy can verify whether the encrypted message belonging to that proxy contains the nonce n_j. If this fails, the protocol is aborted and the user is notified.
5. The function F is executed by the proxy on the encrypted messages $E_{pk_{\mathcal{U}}}(m_j)$ to e.g. decide which of the messages should be send to the user.
6. The proxy signs the message $c_{j,2} = Sign_{s_{\mathcal{P}}}(E_{pk_{\mathcal{U}}}(m_j))$ which resulted from the computation of function F, and combines it with $c_{j,1}$ to obtain a valid signature c_j.
7. Finally, the result of the computation is sent to the user who is able to verify and decrypt the signed message c_j.

4.5 Example

Initialization and Selection. Let us assume the presence of a proxy $\mathcal{P}$ and three ($\ell = 3$) hosts $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$, see figure 4.2. First the user generates the symmetric encryption keys $K_1, K_2, K_3, K_{\mathcal{P}}$ and the random nonces n_1, n_2, n_3. Secondly, the user $\mathcal{U}$ generates the itinerary using equations 4.2 and 4.3, which results in:

$$I_1 = [E_{pk_1}(K_1, s_1), E_{K_1}(Q_1, \mathcal{L}_2, [E_{pk_2}(K_2, s_2), E_{K_2}(Q_2, \mathcal{L}_3, \ldots \\ \ldots, [E_{pk_3}(K_3, s_3), E_{K_3}(Q_3, \mathcal{P}, [E_{pk_{\mathcal{P}}}(K_{\mathcal{P}}, s_{\mathcal{P}})])])])]$$

The secret signing keys are based on a threshold signature scheme (e.g. [55]) where for every host the signing key is split between the host and the proxy. This will result in three valid signature pairs with proxy signing key $s_{\mathcal{P}}$:

$$(s_1, s_{\mathcal{P}}) = s_a$$
$$(s_2, s_{\mathcal{P}}) = s_b$$
$$(s_3, s_{\mathcal{P}}) = s_c$$

The encrypted storage for the proxy $\mathcal{P}$ is calculated and sent to the proxy:

$$E_{K_{\mathcal{P}}}(\mathcal{L}_1, n_1, \mathcal{L}_2, n_2, \mathcal{L}_3, n_3, F)$$

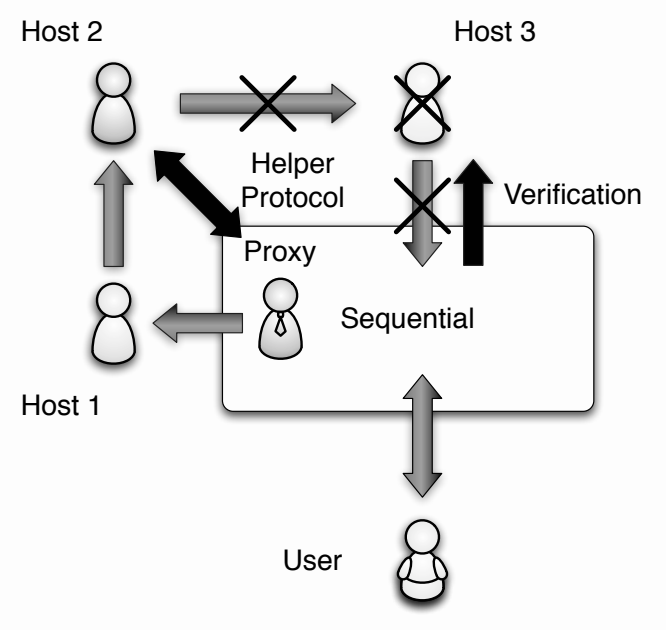


Figure 4.2: Example of the sequential approach

Table 4.2: Example helper protocol lookup-table

Verification	**Value**
$h(E_{K_1}(\mathcal{L}_2))$	$E_{pk_1}(K_2, n_2)$
$h(E_{K_2}(\mathcal{L}_3))$	$E_{pk_2}(K_3, n_3)$

The helper protocol generates table 4.2 which is only calculated for host $\mathcal{L}_1$ and $\mathcal{L}_2$. Host $\mathcal{L}_3$ does not need to skip a host because he can directly reach the proxy $\mathcal{P}$ who is assumed to be always to be available.

Collection. Host $\mathcal{L}_1$ uses his private key sk_1 to decrypt I_1 which reveals the session key K_1 and the partial signing key s_1. Using K_1 the user is also able to reveal the personal query Q_1 and the next host to be visited $\mathcal{L}_2$.

$\mathcal{L}_1$ reads the query Q_1 and decides to respond with the message m_1. Using the public key of the user $\mathcal{U}$ the message is encrypted, $E_{pk_\mathcal{U}}(m_1)$ and with the partial signing key s_1 the message is signed: $c_{1,1} = Sign_{s_1}(E_{pk_\mathcal{U}}(m_1))$.

The encapsulated message is computed by the host:

$$M_1 = Sign_{sk_{\mathcal{L}_1}}[E_{pk_\mathcal{P}}(E_{pk_\mathcal{U}}(m_1), c_{1,1}), H_1]$$

$$H_1 = h(M_0, \mathcal{L}_2)$$

Finally all the information is sent to the next host to visit, $\mathcal{L}_2$.

Host $\mathcal{L}_2$ receives I_2 and retrieves $K_2, s_2, Q_2, \mathcal{L}_3, I_3$. He then signs his message $c_{2,1} = Sign_{s_2}(E_{pk_\mathcal{U}}(m_2))$ and he computes the encapsulated message:

$$M_2 = Sign_{sk_{\mathcal{L}_2}}[E_{pk_\mathcal{P}}(E_{pk_\mathcal{U}}(m_2), c_{2,1}), H_2]$$

$$H_2 = h(M_1, \mathcal{L}_3)$$

Trying to sent his information to $\mathcal{L}_3$ he learns that it is not available. Therefore he decides to contact the $\mathcal{P}$ and initiate the helper protocol by sending $h(E_{K_2}(\mathcal{L}_3))$ to him. The proxy verifies that host $\mathcal{L}_3$ is down. Since this is a value within the helper protocol lookup-table, the proxy responds to the host by sending back the corresponding value $E_{pk_2}(K_3, n_3)$.

After decrypting the received message the host is able to retrieve the missing information because he has learned the symmetric key K_3 of host $\mathcal{L}_3$. He learns that the next host to visit is the proxy $\mathcal{P}$. Finally, he computes the missing encapsulated message and sends this information together with the rest of the encapsulated messages to the proxy $\mathcal{P}$, which is the next host to visit:

$$M_3 = Sign_{sk_{\mathcal{L}_2}}[E_{pk_\mathcal{P}}(n_3), H_3]$$

$$H_3 = h(M_2, \mathcal{P})$$

Finalization. The proxy receives the last part of the itinerary ($I_{\mathcal{P}} = [E_{pk_{\mathcal{P}}}(K_{\mathcal{P}}, s_{\mathcal{P}})]$) and decrypts it using his private key $sk_{\mathcal{P}}$ resulting in the exposure of the session key $K_{\mathcal{P}}$ and the partial signing key $s_{\mathcal{P}}$.

By using $K_{\mathcal{P}}$ the encrypted storage will also reveal, the location of all hosts $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$, the nonces n_1, n_2, n_3 and the function F.

The proxy verifies the signatures of M_1, M_2, M_3 and verifies the integrity of the itinerary using H_1, H_2, H_3. Next, the proxy decrypts:

$$\begin{aligned} D_{sk_{\mathcal{P}}}[E_{pk_{\mathcal{P}}}(E_{pk\mathcal{U}}(m_1), c_{1,1})] &= E_{pk\mathcal{U}}(m_1), c_{1,1} \\ D_{sk_{\mathcal{P}}}[E_{pk_{\mathcal{P}}}(E_{pk\mathcal{U}}(m_2), c_{2,1})] &= E_{pk\mathcal{U}}(m_2), c_{2,1} \\ D_{sk_{\mathcal{P}}}[E_{pk_{\mathcal{P}}}(n_3)] &= n_3 \end{aligned}$$

With n_3 the proxy can also verify that host $\mathcal{L}_3$ was skipped during the itinerary.

Function F is executed by the proxy on the encrypted messages received. For example, the function computes which of two messages m_1, m_2 should be send to the user, without decrypting the encrypted messages $E_{pk\mathcal{U}}(m_1), E_{pk\mathcal{U}}(m_2)$. We define that the messages represent numbers and that $m_1 = 10$ and $m_2 = 5$. Furthermore, function F computes a Greater Than function with two inputs. Since $m_1 > m_2$ then the result of the function F will be $E_{pk\mathcal{U}}(m_1)$.

The proxy finally computes $c_{1,2} = Sign_{s_{\mathcal{P}}}(E_{pk_{\mathcal{U}}}(m_1))$ and combines this with $c_{1,1}$, which will result in $c_1 = Sign_{s_a}(E_{pk_{\mathcal{U}}}(m_1))$.

c_1 is send to the user $\mathcal{U}$, who is capable of verifying the combined signature since the user has generated the threshold signatures.

4.6 Security

In this section a security analysis is given of the selection and collection protocol described above. The security analysis is based on the complete system and therefore the protocols from section 4.4 are not discussed individually.

4.6.1 Protection Against Replay Attacks and Copying

With a replay attack, it is the hosts $\mathcal{L}_j$'s attention to gain information about the decision logic in order to derive the best strategy to get its message accepted by executing the collection protocol multiple times with varying inputs and observing the outputs. Since this logic is not included in the itinerary, this decision logic can not be derived from interaction with the itinerary.

The host can execute the collection protocol multiple times and add a different message to the storage with each execution. The proxy can verify a-posteriori if hosts have not stored more messages than they were allowed to, so this strategy will be

detected. The host can also copy the itinerary and store a different message in each one in the hope that the message which is most beneficial to him will reach the proxy first. But the first itinerary to reach the proxy will initiate the decision protocol and all the itineraries that arrive after this itinerary will be ignored. Therefore, still only one message of this host is considered and the decision cannot be undone by another itinerary.

4.6.2 Redundant Itinerary

A fixed itinerary by the user, opposed to letting each host decide the next hop of the itinerary, has the advantage of being able to audit better if an itinerary has fulfilled its task properly, but it has the downside of having a rigid path across the hosts. Especially if the requirement is also to protect the identities of the other hosts, it poses a problem if one or more of the hosts in its path is not working properly.

The use of the helper protocol, introduces a form of safety net in the mission by helping the itinerary to get past non functioning hosts.

4.6.3 Protection of Itinerary

Not only alteration of the itinerary be will detected, a host $\mathcal{L}_j$ cannot generally see who the other hosts are in the system. Each host $\mathcal{L}_j$ can only see where the itinerary came from (previous destination) and where the itinerary is going next $\mathcal{L}_{j+1}$. Therefore each host $\mathcal{L}_j$ can only see 2 out of j other hosts.

4.6.4 Weak protection of Signing Key

It is the digital signing of the best message by the hosts which seals the process. Once the hosts have used the signing key to sign their message, this signature can be taken as of by the host and the user that they have agreed to a certain message.

Since the signing key can be used to sign messages on behalf of the user, it is important that the signing key does not get compromised. Any party who has the signing key could use it to sign other messages which the user may not necessarily agree to.

In this model, the malicious hosts only get one part of the signing key, namely s_j. The other part, $s_{\mathcal{P}}$ is stored in encrypted form on the proxy. Therefore, the malicious hosts will not be able to get the complete signing key. Malicious hosts do not see the partial signatures $c_{j,1}$ from the other hosts, but even if they did, this would not help the malicious hosts in reconstructing the complete signing key.

Assuming the malicious hosts and the proxy do not collude, the latter cannot derive the secret key either. If the itinerary, for whatever reason, does not return to the proxy it will never be able to see s_j as it cannot decrypt the proxy's stored information. Nor will it see the other half as this is stored in the itinerary.

If the itinerary does return to the proxy, then that host can see $s_{\mathcal{P}}$ and the partial signatures $c_{1,1}, \ldots, c_{j,2}$. Due to the properties of the threshold scheme, it is infeasible to derive s_j from the collection of partial signatures. Therefore it cannot reconstruct the complete signing key. It can, in principle, make all of these partial signatures valid. But the fact that the proxy can at most sign the pre-defined messages, as opposed to signing an arbitrary message if it were to find the complete signing key, is an advantage.

4.6.5 Threat Model and Complexity

From a communication perspective the proxy is the most visited party in the protocol. The user sends encrypted selection information to the proxy, after the computation of the complete itinerary the protocol is executed at the proxy, and the helper protocol resides at the proxy. Therefore, security wise, the proxy plays an important part within the protocol since the user trusts the proxy not to be malicious. This in comparison with the hosts which may be malicious.

Based on the threat model in Section 4.2 we look at the worst case scenario where the proxy is malicious. We consider two cases, before the itinerary is complete (i.e. during processing by the hosts) and after the itinerary has returned to the proxy. The itinerary is computed at the trusted environment of the user and is protected by the hash chaining. Since the proxy does not know the private keys of the hosts and the symmetric keys of the hosts chosen by the user, the proxy is unable to decrypt the itinerary and manipulate it other than replacing the itinerary completely.

Considering the case where the proxy is malicious and the itinerary has not completed. This means the collection protocol has not ended and no collected information has returned to the proxy. In this case the security of the protocol is not compromised and no additional information will leak from the protocol. The proxy is unable to retrieve the session key $K_{\mathcal{P}}$ which is used to decrypt the encrypted storage. Without having access to the encrypted storage, the proxy is unable to retrieve the full itinerary of the user. Since the signing key $s_{\mathcal{P}}$ is also not compromised, the proxy is unable to construct a valid signature based on the threshold signature scheme. The strength of this case relies on the encryption used for the encrypted storage e.g. using AES-128 the time complexity is 2^{128} [17].

In the case the itinerary has completed and the proxy is malicious or when all host collude with the proxy, then the security of the protocol cannot be maintained. With every completion of the itinerary the proxy will learn the content of the encrypted storage. This itself only reveals the hosts which were visited in the past i.e. after the completion of the itinerary. If all hosts would collude together with the proxy, the proxy and a particular host could influence the itinerary during its execution at the host. For example, they could use the helper protocol to skip various hosts during the execution of the itinerary. Hosts which did not collude with the proxy and encrypted a message $E_{pk_{\mathcal{U}}}(m_j)$ intended for the user can still rely on the strength of the asymmetric encryption system. A final undesirable effect of a malicious proxy is that it can

manipulate the signing of the result by replacing this with a valid signed message m_j from a malicious and colluding host $\mathcal{L}_1$.

4.7 Discussion

This chapter introduced a selection and collection protocol which could be used as a basis for an e-commerce setting. By defining strict tasks for each protocol, sensitive information is prevented from being exposed to the possibly malicious hosts during the various processes. Even though we use a fixed itinerary to travel to the different hosts, hosts see only a very small part of the itinerary. In case of failures in the itinerary, a help protocol can be executed to overcome this difficulty. If all of the hosts function properly this help protocol is not needed at all.

The sequential selection and collection protocol is based on previous work by [143]. The protocol is adopted to be used within a private computing problem where a proxy is involved. The proxy is used as a semi-trusted hub from which the itinerary starts and ends. The hosts can be active attackers, since the introduced helper protocol is capable of circumventing malicious hosts.

The basis for the sequential selection and collection protocol are mobile software agents. The main issue is that only a limited amount of papers is related to the threat of untrustworthy proxies. Solutions based on encrypted functions [138] and homomorphic functions [159] are capable of performing limited operations within untrustworthy environments. But they do not provide solutions for protecting the itinerary (path) of the protocol and they do not provide a mechanism to skip hosts if needed.

Various papers have been published on mobile agents in distributed sensor networks [38, 39, 135, 147]. Mobile agents, due to their nature of being autonomous and intelligent, seem suitable for distributed sensor networks. Mobile agents can be used to greatly reduce the communication cost on low-bandwidth connections by moving the processing to the remote sensors rather than bringing the data to a central sensor. The main challenge remains to incorporate them efficiently within sensor networks rather than using them for security or privacy.

Chapter 5

Single Comparison

In this chapter it is studied how an untrustworthy proxy compare private information by computing an inequality function in such a way that the private information remains private in respect of the proxy, but the results of the comparison can be made public if required. This chapter addresses the aforementioned problem by using Multi-Party Computation (MPC) techniques together with the Private Agent Communication Algorithm by [30, 31].

5.1 Introduction

In this fast growing digital era we are more often surrounded by intelligent devices, such as mobile phones or tablets PC's, which can support us in our daily life. While information flows through these ambient devices, which are not only connected to local networks but also to the outside world, we encounter security and privacy threats. The most difficult threat [46] is the malicious adversary who has the ability to create malicious software, to spy on insecure communication channels but also to create untrustworthy proxies.

Every proxy has complete control over the information which is executed in his digital 'world' and it can therefore manipulate the software running on it. One topic that so far has received relatively little attention in literature is the processing of private information within untrustworthy proxies. Most research focusses on malicious information and communications, and assumes that the information is processed on trusted proxies. With private computing it should be possible to possess personal information of a user e.g. credit card numbers, personal preferences, while being able to process this information within untrustworthy proxies without compromising security and privacy.

The research of this chapter has been carried out together with ir. K. Cartrysse and dr.ir. J.C.A. van der Lubbe at the Delft University of Technology. The result has been published in a research paper [72].

In Chapters 3 and 4 we have covered the private computing - selection and collection problem, where a proxy retrieves information from selected hosts without learning what was selected and collected. In Chapter 5 and 6 we want this proxy to compute an inequality function with private information where the challenge is to keep the private information private but the result of the comparison can be learned by the proxy. This challenge has similarities with Multi-Party Computation (MPC).

Multi-Party Computation

Multi-Party Computation deals with protocols which allow e.g. n hosts $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n$ in a network to compute jointly a multi-variable function $f(x_1, x_2, \ldots, x_n) = y$ on their individual private inputs x_i, in such a way that each host only learns the output y [158] and not the inputs x_i of the other hosts. In other words the goal is to preserve during the protocol's process, the maximal privacy of the x_i's and to guarantee the correctness of the common result y. Privacy is achieved if the adversary cannot learn more about the honest hosts's input than what is revealed by the output of the function. Correctness is achieved when the function f is always computed correctly, even in presence of untrustworthy hosts.

MPC protocols where introduced in 1982 by Andrew Yao [158] with the famous 'Millionaires' Problem'. A short description of the problem: two millionaires (Alice and Bob) want to know who is richer without showing their private wealth. So they want to compute a function with their private wealths, respectively x_1 and x_2. This function can be of the form:

$$\begin{aligned} \text{if } x_1 < x_2 \text{ then} \quad & f(x_1, x_2) = 1 \\ \text{else} \quad & f(x_1, x_2) = 0. \end{aligned}$$

So, Alice and Bob discover only who is the richest without knowing each other's wealth. The strength of this protocol is based on the power of one-way functions, that means, functions which are easy to compute but almost impossible to be inverted (in the $\bmod p$ sense). We describe the protocol using the RSA encryption scheme [131]. The protocol initiates with the following initial conditions:

1. Alice has i millions and Bob has j millions, where $1 \leqslant i, j \leqslant 10$;
2. M is the set of all N-bit nonnegative integers;
3. Q_N is the set of all $1-1$ onto functions from M to M;
4. The public key of Alice pk_a is generated randomly from Q_N;
5. The two users don't cheat.

The protocol proceeds as follows:

1. Bob takes randomly an N-bit number x from the set M;
2. And computes privately $k = E_{pk_a}(x)$;
3. Bob then computes $k - j + 1$ and sends it to Alice;

4. Alice computes $y_u = D_{sk_a}(k - j + u)$, for $u = 1, \ldots, 10$, and where sk_a is Alice's secret key;
5. Alice generates randomly a prime number p of $N/2$-bit and computes $z_u = y_u \bmod p$ for all u;
6. She verifies that all z_u differ by at least 2 in the $\bmod p$ sense, if not, then she picks up another prime p and computes the new z_u. This verification guarantees that no number appears twice in the sequence generated by Alice.
7. Alice sends then to Bob the prime number p and the following list:

$$z_1, \ldots, z_i, z_{i+1} + 1, \ldots, z_{10} + 1$$

8. Bobo picks up the j^{th} number from the list and checks whether $z_j = x \bmod p$;
9. If it is true, then $i \geqslant j$, otherwise $i < j$;
10. Bob gives to Alice the response.

The protocol fulfills the requirements of security and privacy. Alice cannot know Bob's wealth i because Bob computes $E_{pk_a}(x)$, while Alice computes $D_{sk_a}(k - j + u)$, she cannot find j.

After Yao several other MPC protocols have been designed, and nowadays the field of MPC is expanding in several areas, for example [80]: Electronic Elections, Electronic Bidding for Contracts, Private and Secure database access, Joint Signatures, Joint Decryptions.

Secure Multi-Party computation is important for cryptography since basically MPC protocols simulate a Trusted Third Party (TTP) where none exists, under the assumption that at most a certain fraction of the hosts involved are corrupted. In other words, MPC protocols 'cancel' the problem of not having the TTP without decreasing the level of security and privacy, because the trust is distributed all over the hosts.

Since we assume the involvement of an untrustworthy proxy within private computing, the aforementioned description of MPC contradicts with our untrustworthy proxy assumption. This also reflects in the description of Yao's protocol which does not use any proxy to perform the computation, since both parties compute the function together. Although not even this most basic MPC protocol fulfills the private computing requirement of using a mandatory proxy, there are MPC protocols which can be adapted for private computing such as [24].

This chapter addresses the single comparison problem based on the private bidding protocol by [24]: Alice wants to buy some goods from Bob if the price is less than a and Bob would like to sell, but only for more than b and neither of them wants to reveal the secret bounds. The private bidding scenario actually compares two values with each other. The objective of this chapter is to demonstrate that comparison (Private Bidding) is also feasible within an untrustworthy environment i.e. proxy. To achieve this goal we are using a proxy to compare the inputs of two users, without learning their private inputs.

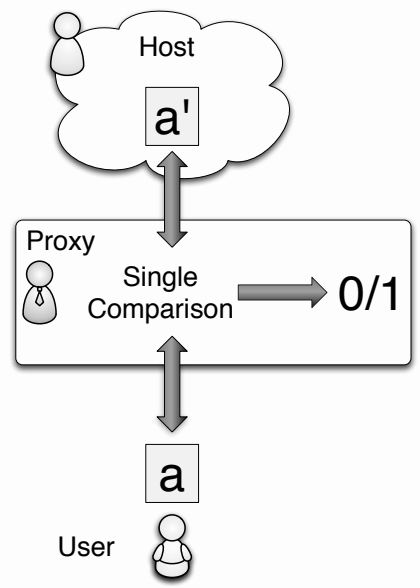


Figure 5.1: Single comparison Scenario

In section 5.2 we present our scenario. The homomorphic E-E-D protocol used within this chapter is described in section 5.3. The single comparison protocol is given in section 5.4, while section 5.5 discusses the security of the system. Section 5.6 ends with the conclusion.

5.2 Scenario

There are similarities between the original private bidding protocol from [24] and the single comparison protocol we present in this chapter. The main difference is that within our protocol the communication between the users always passes through the proxy, because the users are unable to communicate directly with each other. Furthermore, within the original private bidding protocol the untrustworthy party, named Oblivious Third Party (OTP), was unable to compute a result of the computation. While in our protocol the proxy is able to learn the outcome of the comparison.

Figure 5.1 gives an overview of the parties involved within the scenario. The user, host and the proxy are the three parties involved in this scenario. The user initiates the protocol. The proxy is an untrustworthy party that will correctly execute the private inputs a, a' given by the user and host, without being able to learn any information about the inputs. The goal of the parties is to determine whether $a > a'$ privately. In an e-commerce setting the single comparison could be applied as the maximum price a buyer $(e.g.\ user)$ is willing to pay is greater than the minimum price the seller

(*e.g. host*) is willing to receive. Nevertheless, the parties will only reveal their bids a, a' if $a > a'$, otherwise they will keep their bids private.

Threat model. The proxy is an untrustworthy adversary as was introduced by [128]. Consequently, the proxy is a server executing the protocol of the user and the host, while storing all information from the protocol and all communication between the user and host. The proxy is not a malicious party trying to modify the protocol. The main threat within the single comparison problem is breaching confidentiality of the inputs. The proxy could learn viable information during the processing of the protocol and the communication with the user and the host. Furthermore, the proxy could replay the comparison process with different inputs to retrieve private information. A colluding host and proxy are able to retrieve private information from the user.

Requirements. Our goal is to ensure that the proxy does not learn the private inputs a, a' before the users decide privately to reveal their secret inputs e.g. when $a > a'$.

The requirements for our single comparison scenario are:

- The proxy performs the comparison computations.
- The inputs must remain private to the users.
- The comparison function computed by the proxy should be an inequality function.
- The proxy learns (some) information of the output
- The output should be made public by the proxy (All users should be able to learn the same information as the proxy).

5.3 Homomorphic E-E-D

In [31] a protocol is described which is able to encrypt a message *m* with the public key of the user and send it to the host which encrypts the message again with his own public key. The encrypted message is send back to the user, who is able to decrypt the message in such a way that the result is a message encrypted by the host. This protocol, as shown in figure 5.2 is know as Encryption-Encryption-Decryption (E-E-D).

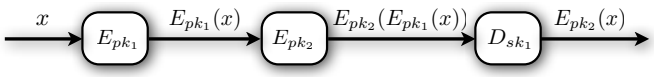


Figure 5.2: E-E-D

Homomorphic E-E-D is based on the homomorphic (addition) property of ElGamal [62]. The concept behind the algorithm is to encrypt two messages with different keys

and adding them together. Next, the messages are decrypted with the first key and the result will be the addition of the two messages which are only encrypted with the second key, see figure 5.3.

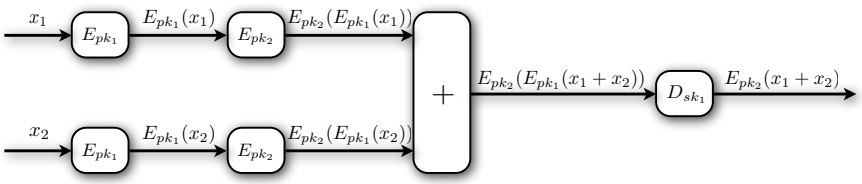


Figure 5.3: Homomorphic E-E-D

We generate a large prime p and choose a generator $g \in \mathbb{Z}_p^*$. We choose a random integer $a_1, 1 \leq a_1 \leq p-2$ and compute $\omega_1 = g^{a_1} \bmod p$. We also select a random integer $k_1, 1 \leq k_1 \leq p-2$ and compute the following:

$$\gamma_1 = g^{k_1} \bmod p,$$
$$\delta_{1,x_1} = x_1 \omega_1^{k_1} \bmod p = E_{pk_1}(x_1).$$

The message is x_1, the cipher text is $c_{1,x_1} = (\gamma_1, \delta_{1,x_1})$ and the secret key sk_1 is a_1. Next, we encrypt δ_{1,x_1}, using a new public key (p, g, ω_2). We also compute $\omega_2 = g^{a_2} \bmod p$ and use this to encrypt the message:

$$\gamma_2 = g^{k_2} \bmod p,$$
$$\delta_{2,x_1} = \delta_{1,x_1} \omega_2^{k_2} \bmod p = x_1 \omega_1^{k_1} \omega_2^{k_2} \bmod p = E_{pk_2}(E_{pk_1}(x_1)).$$

The public key pk_2 is (p, g, ω_2), the cipher text is $c_{2,x_1} = (\gamma_1, \delta_{2,x_1})$ and the secret key sk_2 is a_2. For convenience we write δ_{2,x_1} as $E_{pk_2}(E_{pk_1}(x_1))$ which represents the two top left function blocks shown in figure 5.3.

We encrypt a message x_2 with the same keys and the same random integers k_1, k_2 as for message x_1, which can be calculated as indicated above. The result is written as $E_{pk_2}(E_{pk_1}(x_2))$ which represents the two bottom left function blocks from figure 5.3. Furthermore, we add the two ciphertexts δ_{2,x_1}, δ_{2,x_2} together:

$$\begin{aligned} x = \delta_{2,x_1} + \delta_{2,x_2} &= E_{pk_2}(E_{pk_1}(x_1)) + E_{pk_2}(E_{pk_1}(x_2)) \\ &= \delta_{1,x_1} \omega_2^{k_2} \bmod p + \delta_{1,x_2} \omega_2^{k_2} \bmod p \\ &= x_1 \omega_1^{k_1} \omega_2^{k_2} \bmod p + x_2 \omega_1^{k_1} \omega_2^{k_2} \bmod p \\ &= (x_1 + x_2) \omega_1^{k_1} \omega_2^{k_2} \bmod p \\ &= \delta_{2,x_1+x_2} = E_{pk_2}(E_{pk_1}(x_1 + x_2)). \end{aligned}$$

The two encrypted messages δ_{2,x_1}, δ_{2,x_2} can only be added to one another when they are both encrypted with the same public keys and with the same random integers k_1,

k_2. The result, $E_{pk_2}(E_{pk_1}(x_1+x_2))$, is decrypted by the first private key a_1 and the following equation:

$$\begin{aligned} x' = D_{sk1}(x) &= \frac{\delta_{2,x_1+x_2}}{\gamma_1^{a_1}} \bmod p \\ &= \frac{(x_1+x_2)\omega_1^{k_1}\omega_2^{k_2}}{g^{k_1 a_1}} \bmod p \\ &= (x_1+x_2)\omega_2^{k_2} \bmod p. \end{aligned}$$

We now have the first and the second message added to one another and encrypted with the second public key pk_2, which can be written as follows:

$$D_{sk1}(E_{pk2}(E_{pk1}(x_1+x_2))) = E_{pk2}(x_1+x_2) = x'$$

Consequently, decrypting x' produces the following:

$$\begin{aligned} D_{sk1}(x') &= \frac{x'}{\gamma_2^{a_2}} \bmod p \\ &= \frac{(x_1+x_2)\omega_2^{k_2}}{g^{k_2 a_2}} \bmod p \\ &= x_1+x_2. \end{aligned}$$

which is the addition of the two messages without encryption.

It has to be noticed that according to the Diffie-Hellman problem it is impossible to compute $g^{ak} \bmod p$ from $g^a \bmod p$ and $g^k \bmod p$ [58]. Furthermore, the key a and the random integer k must remain secure, and preferably should be changed frequently.

5.4 Protocol

The following protocol is carried out by three parties: the user, the proxy and the host. The protocol is initialized by the user and the proxy who will choose, generate and compute various variables, keys and encryptions. The host then receives information from the user through the proxy, processes it and sends it back the proxy. In the end, the proxy is able to compute a greater than function over two secret values without leaking any information.

The protocol begins by letting the proxy generate an (ElGamal) key-pair with the public encryption key E_{pkt} and the secret decryption key D_{skt}. The user is unwilling to reveal his private value a. This value can be represented as a binary string. Next, random and independent values are chosen:

$$\begin{aligned} x_l, x_{l-1}, \ldots, x_0 &\in \mathbb{Z} \\ r_{l-1}, \ldots, r_0 &\in \mathbb{Z} \\ s_{l-1}, \ldots, s_0 &\in \mathbb{Z} \end{aligned}$$

where l is chosen as the maximum number of bits used for the representation of the private value of the user a and the private value of the host b.

The user needs to define an n-bit prime $p_a = \lambda(t_a)$ where $t_a \in \{0,1\}^n$ is randomly chosen. Using $\lambda(x)$ the user can map a string x to an n-bit prime p e.g. by finding the smallest prime greater than x. Then the user generates a random m_a using the Φ-Hiding Assumption, as described in Appendix A.5, which hides her prime p_a. The user also computes $G_a = \frac{\phi(m_a)}{p_a}$ which will be used at the end of the protocol to determine which of the private values is the largest.

Next, a key κ for the hash function H_κ is generated and published. The user calculates $\varphi_{a,j}$ using the hash function $H_\kappa(x_j + s_j)$ (for $j = l-1,\ldots,0$):

$$\varphi_{a,j} = H_\kappa(x_j + s_j) \oplus t_a.$$

Where all values except $t_a \in \{0,1\}^n$ are $\in \mathbb{Z}$.

The user chooses a secret value a which needs to remain confidential. The user represents the value a as a string of bits a_j for $j = l-1,\ldots,0$ and encrypts it using his public-key $E_{pka,j}$ and random integers $k_{a,j}$. This has to be *"sealed"* using the public-key of the proxy and the random integers $k_{t,j}$

$$y_{a,j} = \begin{cases} E_{pkt,j}(E_{pka,j}(x_j - x_{j+1} + r_j)) & \text{if } a_j = 0 \\ E_{pkt,j}(E_{pka,j}(x_j - x_{j+1} + s_j + r_j)) & \text{if } a_j = 1 \end{cases}$$

The bidding is prepared for the host by the user using the same cryptosystem with random integers $k_{t,j}$, $k_{a,j}$ and public-keys $E_{pkt,j}$:

$$y_{b,j} = \begin{cases} y_{b0,j} = E_{pkt,j}(E_{pka,j}(-r_j)) & \text{if } b_j = 0 \\ y_{b1,j} = E_{pkt,j}(E_{pka,j}(-s_j - r_j)) & \text{if } b_j = 1 \end{cases}.$$

He computes,

$$\Psi_{b,j} = H_\kappa(x_j - s_j),$$

which is send to the host through the proxy.

The host is able to acquire the public $y_{b,j}$. He also uses his secret value b, which is represented as a string of bits b_j for $j = l-1,\ldots,0$. Because $y_{a,j}$ is also publicly available he can compute $y_{ab,j}$:

$$y_{ab,j} = y_{a,j} + y_{b,j}. \tag{5.1}$$

To be sure that the proxy will be unable to construct b from y_{ab}, the host encrypts (5.1) again, by computing:

$$w_{ab,j} = E_{pkt,j}(y_{ab,j}).$$

The only requirement with this protocol is that the host and the user are using the samen cryptosystem i.e. they have the same generator g and prime p but they can have a different k.

The user has already chosen a $t_b \in \{0,1\}^n$, defined an n-bit prime $p_b = \lambda(t_b)$ and generated a random m_b. He also computes $G_b = \frac{\phi(m_b)}{p_b}$ used for determining the end result. The host received from the user $\Psi_{b,j}$ and computes before sending information back to the proxy, for $j = l-1,\ldots,0$:

$$\varphi_{b,j} = \Psi_{b,j} \oplus t_b.$$

The user and the host have prepared their values using the public information and are now able to send their information to the proxy. The host sends $\varphi_{b,j}$, m_b, $w_{ab,j}$ and G_b to the user with help of the proxy.

The user is able to decrypt $w_{ab,j}$ because he is in possession of the secret-key $D_{ska,j}$:

$$\begin{aligned} z_j &= D_{ska,j}(w_{ab,j}) \\ &= D_{ska,j}(E_{pkt,j}(y_{ab,j})). \end{aligned}$$

The User sends the following values to the Proxy

$$\begin{aligned} &\kappa, x_l, z_{l-1}, \ldots, z_0, \varphi_{a,l-1}, \ldots, \varphi_{a,0}, \\ &\varphi_{b,l-1}, \ldots, \varphi_{b,0}, m_a, m_b, \\ &G_a. \end{aligned}$$

The proxy lets $c_l = x_l$ and chooses $g_{a,l} \in QR_{m_a}$, $g_{b,l} \in QR_{m_b}$ by selecting a random element of $\mathbb{Z}_{m_a}$, respectively $\mathbb{Z}_{m_b}$, and squaring it. For $j = l-1,\ldots,0$ the following five steps are repeated

1. $c_j = c_{j+1} + D_{skt,j}(D_{skt,j}(z_j))$
2. $q_{a,j} = \lambda(H_\kappa(c_j) \oplus \varphi_{a,j})$
3. $g_{a,j} = (g_{a,j+1})^{q_{a,j}} \bmod m_a$
4. $q_{b,j} = \lambda(H_\kappa(c_j) \oplus \varphi_{b,j})$
5. $g_{b,j} = (g_{b,j+1})^{q_{b,j}} \bmod m_b$

After this iteration the proxy chooses $r_a \in \mathbb{Z}_{m_a}$, $r_b \in \mathbb{Z}_{m_b}$ randomly, computing:

$$\begin{aligned} h_a &= (g_{a,0})^{r_a} \\ h_b &= (g_{b,0})^{r_b}, \end{aligned}$$

where h_a is sent to the user and h_b is sent to the host.

All parties can test the result of the computation by using the factorization of m_a and m_b. The user can check whether $a > b$

$$h_a^{G_a} = h_a^{\frac{\phi(m_a)}{p_a}} \equiv 1 \bmod m_a.$$

Similarly, the host can check whether $a < b$ by computing

$$h_b^{G_b} = h_b^{\frac{\phi(m_b)}{p_b}} \equiv 1 \bmod m_b.$$

The proxy can either choose G_a or G_b to compute the result of the protocol and learn if $a > b$, or not, without being able to learn a or b.

5.5 Security Analysis

The original private bidding protocol by [24] did not use ElGamal for the communication between the user and the host. We choose ElGamal due to the E-E-D property. The protocol is secure if the comparison takes place at the untrustworthy proxy and as long the proxy does not learn the private-key of the user to decrypt $w_{ab,j}$.

The proxy has G_a or G_b to compute the result of the comparison. G_a, G_b can be used to gain more information during the five steps iteration process performed by the proxy, which is able to discover when it contains a *p-th root* modulo m_a or m_b. The result is that the proxy is able to learn how many bits the strings a_j and b_j differ from each other i.e. the proxy is able to learn the distance $d = a - b$ without learning a or b. The proxy will therefore only learn information regarding the relationship between de values, but nothing else.

The protocol is fair because both the user and the host are able to check individually what the result is of the comparison computed by the proxy.

ElGamal is insecure if we keep the random k identical for every encryption. In our case we also keep the k the same but we claim this is secure. The user is the only one who encrypts the messages $y_{a,j}$, $y_{b0,j}$ and $y_{b1,j}$, therefore he is the only one who knows the content of the messages; which is also not known to the host. Suppose the same k is used to encrypt two messages x_1 and x_2 and the result is the ciphertext pairs $(\gamma_1, \delta_{1,x_1})$ and $(\gamma_2, \delta_{2,x_2})$. Then

$$\frac{\delta_{1,x_1}}{\delta_{2,x_2}} = \frac{x_1}{x_2} \tag{5.2}$$

is easily computed if x_1 or x_2 is known [115]. In our case, the messages are only known to the user and therefore no other entity is capable of computing (5.2). We therefore claim that the use of the identical k does not make our protocol insecure.

The user initiates the comparison process, therefore he can cheat the protocol. This could be prevented by adding commitments and letting the proxy be more actively involved in the comparison process.

Threat model and Complexity. In Section 5.2 we formalized the threat model for the single comparison protocol. The proxy is curious to the inputs which are being compared but the protocol will prevent the proxy to learn any additional information. The functionality is based on the homomorphic E-E-D (Section 5.3) and the security is based on the Φ-Hiding Assumption (Appendix A.5). Furthermore, the protocol only requires one round of communication with the user and the proxy. The complexity of breaking the single comparison protocol is as hard as breaking the factorization of the Euler totient function $\phi(m)$.

If a curious proxy would turn malicious the protocol would rely on the cryptographic primitives but fail to produce a reliable outcome of the computation since the proxy is able to alter the protocol completely. For example, the proxy would be able to

randomly choose values from $y_{a,j}$ and compare them with random values from $y_{b,j}$, which randomizes the result.

If the complete set of values $y_{b,j}$ is compromised together with the comparison value G_a already know by the proxy, the malicious proxy is be able to perform a replay attack on the input values. By repeatedly changing the input values of the host, the proxy is able to retrieve the private value of the user. This can be prevented by encrypting the set of values with the public key of the host. The downside of this encryption is that the user must already know the host he is going to compare the values with.

If the host would collude with the proxy and get access to the comparison value G_a of the user, he could use the proxy as an oracle by querying it and gaining access to the users' private information.

5.6 Discussion

We have presented a single comparison protocol that keeps the private values of the user and the host private while the communication and the comparison is performed by an untrustworthy proxy.

This chapter is based on previous work by [24], which was based on Multi-Party Computation (MPC) techniques. The issue with MPC is that most solutions are unsuitable for private computing with untrustworthy proxies since most solutions avoid the usage of these curious proxies. The paper by [24] focusses on fairness and therefore was bound to use a third party. By adopting the original protocol with various techniques such as E-E-D and the usage of ElGamal in a special way we where able to improve the protocol for usage with untrustworthy proxy. In the original paper the proxy was unable to learn any information other than having to contact the user or host. In this chapter the proxy is able to learn one-bit of information related to the inputs.

In other approaches such as "Non-interactive CryptoComputing for NC^1" by Sander, Young and Yung [139] a user sends a private message x to the proxy, who processes it using his private function f and sends his output $f(x)$ back to the user. In their solution the user learns only the output $f(x)$ and not about the function f while the proxy learns nothing about the message x but also nothing about the output. In our approach the proxy only learns the outcome and nothing else.

Freedman *et al* [68] and Li *et al* [86] discuss single comparison in the context of finding common data elements (intersection) in two databases without revealing private information. This is different from our approach, where the goal is to compare two approximately equal inputs together instead of finding the intersection.

The single comparison protocol only computes one greater than function while in various scenario it is more interesting to compare multiple greater than functions. In the next chapter we will provide a solution for multiple comparisons based on the same paper as single comparison.

Chapter 6

Multiple Comparison

Consider a scenario for comparing multiple values, where a user wants a set of secret values to be compared with another set of secret values by an untrustworthy proxy. This comparison is done in such a way that the proxy only learns whether the inputs are approximately equal to each other or not. Theproxy does not learn any other information about the secret values. This protocol is based on multi-party computational techniques together with the Φ-hiding assumption, which is extended to hide multiple primes.

6.1 Introduction

One of the challenges of private computing is the problem of comparing 'noisy' values. Suppose, a user provides his secret values $F = \{a_1, \ldots, a_n\}$ to a trusted device, which encrypts the information and the result $E(F)$ is stored on a public database. Next, the user gives another set of secret values $F' = \{a'_1, \ldots, a'_n\}$ to a trusted device. These values are also encrypted and the result $E(F')$ is sent to the proxy. The proxy compares the two samples, while only obtaining one-bit of information about the two values. During this comparison the proxy should only learn whether $F \approx F'$ without knowing F or F'. We use the term 'noisy' to indicate that the compared values are approximately equal (or not) based on a pre-defined error-margin. We assume that proxy follows the protocol correctly, but is interested in the content that he is processing i.e. untrustworthy behavior. Because the proxy is a threat, he should be prevented to repeatedly compare various values with the already submitted values; with the goal of gaining knowledge about the values without decrypting it (replay-attack).

This problem is similar to password protection, where the hash-value of a password P is computed based on a one-way function H and the result $H(P)$ is stored in a database.

The research of this chapter has been carried out together with dr.ir. J.C.A. van der Lubbe at the Delft University of Technology. The result has been published in a paper [74].

During comparison the user enters his password P' and the hash-value $H(P')$ (based on the same function H) is compared with the previously obtained hashed password $H(P)$. The two values compare (are equal) if $H(P) = H(P')$ i.e. $P = P'$. The hash function produces a fixed-length output string for an arbitrary length password P and because it is computationally infeasible to find two passwords P and P' such that $P \neq P'$ but $H(P) = H(P')$ (i.e. collision-free), this simple scenario succeeds.

The problem is that comparing 'noisy' values based on one-way functions is infeasible, because a small difference in the inputs will result in a large difference in the output. Therefore if $F \approx F'$ it will not mean that $H(F) \approx H(F')$.

Threat model. The multiple comparison problem is based on the same threat model as the single comparison problem, described in Section 5.2.

We present a multiple comparison protocol which only needs one round of communication and which is based on the private bidding protocol [24], instead of using one-way functions. In section 6.2 we present our scenario and discuss the relation with previous work. Section 6.3 describes the building blocks that are necessary to construct the protocol. In section 6.4 the protocol is presented and the security of the protocol is discussed in section 6.5. Finally, section 6.6 ends with concluding remarks.

6.2 Problem Statement

Our objective is to demonstrate that an untrustworthy proxy can determine whether two messages from user are equal within some pre-defined error margin, without revealing those messages i.e. we want proxy to compare two values while maintaining privacy.

The main problem is how to compare two noisy values with each other. The values are retrieved from the same user at two different times. The first value is used as reference for the comparison and stored in a public database *DB*. Values retrieved later are compared during the comparison with the reference values from the database, afterwards these new values are not stored but discarded. The user of the system is trusted not to tamper with the protocol i.e. trustworthy. Because the values must remain private, the user will also provide a key k which remains secret using e.g. PIN code or personal card.

The proxy controls the comparison process and is considered to be untrustworthy, which implies that he will execute the protocol correctly but he is interested in the secret values F. This passive untrustworthy proxy, should only learn from the comparison process that $F \approx F'$ but not F or F'. Furthermore, the proxy should not be able to compare values from possible other users or values generated by the proxy itself. Finally, eavesdroppers should not learn any information from the values sent over public channels.

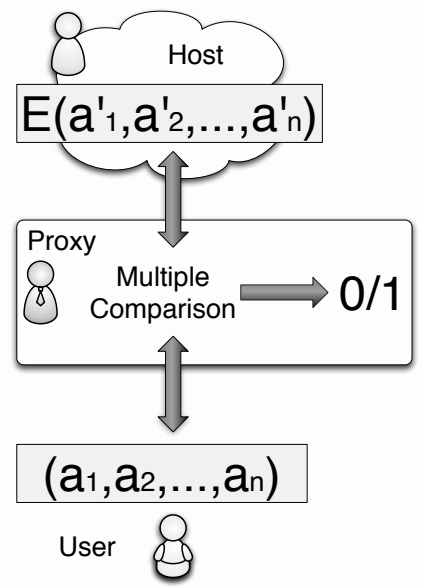


Figure 6.1: Multiple comparison

6.3 Cryptographic Techniques

The comparison of 'noisy' values is based on the two-party private bidding protocol of Cachin [24] which is modified to suit our requirements. The algorithms that are used for the multiple comparison protocol are discussed in Appendix A and modifications are presented in the next paragraphs.

Private Bidding versus Multiple Comparison

The private bidding protocol of [24] consists of three different parties: buyer, seller and Oblivious Third Party (OTP). Within our multiple comparison scenario (Section 6.2) we also have three parties: a user how starts the protocol, a host who stores the secret values and a proxy which does the comparison. The main difference between the two protocols is that the Oblivious Third Party does not learn anything about the comparison while the proxy does. Another difference is that the buyer and the seller are two independent parties submitting their own secret values. With multiple comparison, the user and the host are two different parties but the values provided by the host are created by the user $(a'_1, \ldots, a'_n)$ and only kept confidential at the host using encryption $E(a'_1, \ldots, a'_n)$. The user provides new secret values $(a_1, \ldots, a_n)$ which can be compared with the encrypted reference values $E(a'_1, \ldots, a'_n)$ stored by the host. This multiple comparison scenario is shown in figure 6.1.

Due to the 'noise' in the values we need to compare the values using a distance function $D(F,F') < \varepsilon$ where ε is the threshold or error-margin. After defining the threshold, the distance function can be divided into two Greater Than (GT) functions

$$F + \varepsilon > F' > F - \varepsilon \tag{6.1}$$

The private bidding protocol does not fulfill the requirements of letting the proxy learn information about the comparison, therefore we need to make alterations to the original protocol and apply this to our multiple comparison scenario. Taking the requirements of comparison into account the following must be changed. Using private bidding, the OTP (proxy) is not able to learn anything from the comparison; it is only capable of computing the GT function and sent the result back to the buyer and seller. With multiple comparison the proxy should learn the result of the distance function but nothing more. Also, with private bidding the buyer sends his encrypted secret values to the seller without sending it directly to the proxy. In multiple comparison we require the user to give his (encrypted) secret directly to the proxy.

The proxy learns nothing about the relationship between F and F' during the private bidding protocol, because the proxy sends the encrypted result back to the seller and buyer. In the multiple comparison scenario the proxy should learn only the relationship between F and F' but noting more. Since F and F' consist of multiple values they have to be compared separately. Two approaches are possible. Compare each value of $F = (a_1, \ldots, a_n)$ and $F' = (a'_1, \ldots, a'_n)$ separately where each comparison result is learned by the proxy. Or, compare each value separately but let the proxy only learn the overall result of the comparison. The downside of the first approach is that the proxy learns the result of each comparison and therefore learns more than just if F is or is not greater dan F'. The second approach has the advantage that when at least one of the distance functions does not fulfills the requirement then all distance functions will fail.

6.3.1 Extended ΦHA

Private bidding [24] is based on the Φ-hiding assumption (ΦHA) [25], which states the difficulty of deciding whether a small prime divides $\phi(m)$, where m is a composite integer whose factorization is unknown. The original protocol can only hide one prime. Next we will discuss how it can be extended to hide multiple primes so it can be used within the computation of multiple GT functions.

In the extended ΦHA, let:

$$m_n = q'_0 \prod_{i=1}^{n} q'_i = q'_0 q'_1 q'_2 \ldots q'_n \tag{6.2}$$

where n is the amount of primes we want to hide, q'_0 is a safe prime of the form $2q_0 + 1$, q_0 is a prime and q'_i are quasi-safe primes of the form $2p_i q_i + 1$. The primes p_i, q_i are odd primes and should satisfy $p_i \neq q_i$ for all i. The modulus m_n will only hide the

primes $p_1, \ldots, p_n$ if $\prod_{i=1}^{n} p_i$ divides $\phi(m_n)$. Using equation (6.2) the totient function will always have the form $\phi(m_n) = 2^{n+1} q_o \prod_{i=1}^{n} p_i q_i$ when we are hiding multiple primes.

Consider a set of primes $\mathcal{P} = \{p_1, \ldots, p_\eta\}$ where $\eta \geq n$, a modulus m_n that hides the primes $\prod_{i=1}^{n} p_i$, and a random value $g \in \mathbb{Z}_{m_n}$. Then the ΦHA can be used to determine whether $\mathcal{P}$ contains a $\prod_{i=1}^{n} p_i$-th root modulo m_n using the factorization of m_n. To check this one computes:

$$\alpha = \frac{\phi(m_n)}{\prod_{i=1}^{n} p_i} \prod_{j=1}^{\eta} p_j,$$
$$\beta = g^{\alpha}.$$

If $\beta \equiv 1 \bmod m_n$, then the set $\mathcal{P}$ hides $\prod_{i=1}^{n} p_i$ else it does not. The set $\mathcal{P}$ will only hide $\prod_{i=1}^{n} p_i$ if it divides $\prod_{i=j}^{\eta} p_j$ for $\eta \geq n$. It is important that the starting value $g \in \mathbb{Z}_{m_n}$ should not be a $\prod_{i=1}^{n} p_i$-th residue modulo m_n i.e. there should not exist a μ such that $\mu^{\prod_{i=1}^{n} p_i} \equiv g \bmod m_n$.

For a randomly chosen m_n and the product of primes $\Psi_1 = \prod_{i=1}^{n} p_i, \Psi_2 = \prod_{j=1}^{n} p_j$ such that m_n hides Ψ_1 but not Ψ_2, the tuples (m_n, Ψ_1) and (m_n, Ψ_2) are computationally indistinguishable. This means it is hard to distinguish whether Ψ_1 or Ψ_2 is a factor of $\phi(m_n)$.

6.4 Protocol

We will describe the protocol for one distance function as shown in equation 6.1. We require to compute two Greater than function $F + \varepsilon > F'$ and $F' > F - \varepsilon$. With private bidding the ΦHA assumption was used to hide one prime for one greater than function. With multiple comparison we require to compute multiple greater than function and therefore we also need to hide multiple prime numbers. For this we will use the extended ΦHA. In section 6.5 we will discuss the security and explain the advantages of using the extended ΦHA when multiple GT functions have to be computed.

The user provides this secret value and defines the error-margin ε. This will result in a l'-bit input value F which should be dividid into two greater than functions (equation 6.1) to compute the distance function:

$$a = F - \varepsilon,$$
$$b = F + \varepsilon,$$

where a is the l''-bit lower bound and b is the l-bit upper bound of F. The lower and upper bound are used for comparison of F with a new secret value. The values F, a are zero padded at the most significant bits. We denote a_j as the j^{th}-bit of a and b_j as the j^{th}-bit of b, where $j = l-1, \ldots, 0$. During comparison the protocol will compare each j^{th}-bit of a_j with b_j.

Initialization The user chooses randomly a k'-bit prime $p_1 = \lambda(T_a)$ where $T_a \in \{0,1\}^{k'}$. Let $\lambda(x)$ be a mapping of value x to a k'-bit prime. The same is done for $p_2 = \lambda(T_b)$ where $T_b \in \{0,1\}^{k'}$. A simple way to map the $\lambda(x)$ function is to find the smallest prime greater than x.

The user uses the extended ΦHA to hide the randomly chosen primes p_1 and p_2. He selects three primes at random, q_0, q_1, q_2 and computes $m = q_0' q_1' q_2'$ where $q_0' = 2q_0 + 1, q_1' = 2p_1q_1 + 1$ and $q_2' = 2p_2q_2 + 1$. Denote that m should be larger than l-bits. Let $\phi(m)$ be the totient function of m. The user computes:

$$h = \frac{\phi(m)}{p_1 p_2}, \tag{6.3}$$

which h is sent to the proxy.

Next, the user selects random values $x_i, y_i \in \mathbb{Z}_m$ for $0 \leq i \leq l$ and $s_i \in \mathbb{Z}_m$ for $0 \leq i \leq l-1$. He also chooses another set of random values $t_i \in \mathbb{Z}_m$ for $0 \leq i \leq l-1$, which will be used to combine the secret values. The values x_i, y_i are used for comparison of the bits. Since x_i, y_i, s_i, t_i are also needed during the comparison, they should remain private.

The user should make sure the he has access to the public encryption key E_{pk} of proxy $\mathcal{P}$ which is used to transmit the data securely over the untrustworthy channel.

Storage at the Host The user uses a trusted device to encrypt the lower bound a. For each bit of a denoted by a_j, for $j = l-1, \ldots, 0$ the user computes:

$$z_{a,j} = \begin{cases} E_{pk}(x_j - x_{j+1} + t_j) & \text{if } a_j = 0 \\ E_{pk}(x_j - x_{j+1} + t_j + s_j) & \text{if } a_j = 1. \end{cases}$$

Then, the upper bound b is encrypted as follows:

$$z_{b,j} = \begin{cases} E_{pk}(y_j - y_{j+1} + t_j) & \text{if } b_j = 0 \\ E_{pk}(y_j - y_{j+1} + t_j + s_j) & \text{if } b_j = 1. \end{cases}$$

Finally, the encrypted upper- and lower bound:

$$z_{a,l-1}, \ldots, z_{a,0}, z_{b,l-1}, \ldots, z_{b,0}.$$

are sent to the host and stored as a reference values to be used during the comparison.

The second secret value The user provides another l-bit secret value $F' = c$ which is approximately equal to F if it satisfies $a < c < b$. Let c_j be the j^{th}-bit of c where $j = l-1, \ldots, 0$. He chooses random values $r_i \in \mathbb{Z}$ for $0 \leq i \leq l-1$ and defines $r_l = 0$. By using his private values s_j, t_j the sample c_j can be encrypted for $j = l-1, \ldots, 0$ as follows:

$$z_{c,j} = \begin{cases} E_{pk}(-t_j + r_j - r_{j+1}) & \text{if } c_j = 0 \\ E_{pk}(-t_j + r_j - r_{j+1} - s_j) & \text{if } c_j = 1. \end{cases}$$

He also needs to compute:

$$\delta_{a,j} = H(x_j + r_j - s_j) \oplus T_a$$
$$\delta_{b,j} = H(y_j + r_j + s_j) \oplus T_b.$$

where H is a one-way function and which together with $z_{c,l-1}, \ldots, z_{c,0}$ is send to the proxy.

Comparison The proxy must merge the reference samples $z_{a,j}, z_{b,j}$ with the 'fresh' sample $z_{c,j}$ for $j = l-1, \ldots, 0$. It computes:

$$z_{ac,j} = z_{a,j} \cdot z_{c,j} \tag{6.4}$$
$$z_{bc,j} = z_{b,j} \cdot z_{c,j} \tag{6.5}$$

before starting the comparison process. Computing (6.4) and (6.5) will result in the addition of the encrypted values of a_j, b_j, c_j because we are using an additive homomorphic cryptosystem e.g. Paillier [125, 126].

Let $d_l = x_l$, $e_l = y_l$ and choose $g_l \in QR_m$. The following steps must be repeated to compute the comparison of the lower and upper bound for $j = l-1, \ldots, 0$

1. $d_j = d_{j+1} + D_{sk}(z_{ac,j})$,
2. $e_j = e_{j+1} + D_{sk}(z_{bc,j})$,
3. $q_{ac,j} = \lambda(H(d_j) \oplus \delta_{a,j})$,
4. $q_{bc,j} = \lambda(H(e_j) \oplus \delta_{b,j})$,
5. $g_j = (g_{j+1})^{q_{ac,j} q_{bc,j}}$.

To acquire the result of the comparison the proxy needs equation (6.3) to check whether

$$(g_0)^h \equiv 1 \bmod m \tag{6.6}$$

succeeds. If equation (6.6) is equal to 1 the proxy can conclude that the secret value F' provided by the user is approximately equal to the reference secret value F (provided by the user through the host) i.e. $F' \approx F$. If the output is 0 then $D(F', F) > \varepsilon$.

With only one round of communication the proxy was able to compare the data samples of the user only learning the relationship between the secret values but nothing more.

6.5 Security

This section will give a security analysis of the multiple comparison protocol during various situations.

Communication Channel. The information sent between the user , the host and the proxy goes through an untrustworthy channel, where eavesdroppers can monitor all

communications. Also, considering our comparison protocol we are looking for a cryptosystem with additive homomorphic properties. Consequently, Paillier public-key cryptosystem is considered as a good candidate. According to [32] Paillier is currently considered as the best candidate because of its efficiency, the semantic security is proven under the decisional composite residuosity assumption and so far no adaptive chosen-ciphertext attack to recover the secret key is known.

Hidden primes. Assuming that the cheating proxy is able to retrieve the hidden primes P, then the protocol will only give away a minimum amount of information about the relationship between the two input messages F, F'. The GT function compares two bit-strings starting with the most significant bits. If the GT criteria (e.g. $c > a$ or $b > c$) is met, the protocol will generate the hidden prime; in all other cases the protocol generates a random prime. The consequence is: First the proxy will learn the bit location where the two bit strings differ and secondly he will also know those actual bits. Finally, he can conclude that the bits prior to the first difference must be equal, but he does not know their value. For example, with bit-string $a = 1111$ and bit-string $b = 1100$ it is evident that $a > b$. We also see that the two most significant bits are the same. The first difference occurs at the third bit from the left, which will be known to the proxy. When enough secret values are compared the proxy would be able to derive the reference secret value F. Therefore, in a practical situation F must be renewed regularly.

Reconstruction. Even if the proxy is able to recover some bits of the original secret value, he is unable to reuse this information to compare it with other secret values. Because s_j can be seen as the private key of the user, nobody else is able to construct messages that belong to the user. This also protects the secret values against replay attacks. By adding the random string r_j to the message and using the one-way function, during the comparison, no user is able to compare various secret values of the same user. Finally t_j is added to bind the stored (at the host) secret value with new secret values, which prevents comparison with other users secret values.

Threat model. Chapters 5 and 6 are based on the same threat model as described in Section 5.2. Although they are based on the same paper [24], their security outcome is different due to the improvement in the ΦHA and the usage of a different homomorphic encryption scheme.

If the proxy follows the protocol correctly then he will only learn that the secret values of the user are equal, within a predefined threshold ε.

The main threat is that the proxies becomes malicious and starts executing the protocol differently then expected. For example, if the proxy is able to recover some bits of the original secret value, he is unable to reuse this information to compare it with other secret values. Because s_j can be seen as the private key of the user, nobody else is able to construct messages that belong to the user. This also protects the secret values against replay attacks. By adding the random string r_j to the message and using the one-way function, during the comparison, no user is able to compare various secret values of the same user. Finally t_j is added to bind the stored (at the host) secret value

with new secret values, which prevents comparison with other users secret values.

If user X tries to match random values using the proxy he would not be able to produce verifiable encrypted values because he does not have s_j, x_j which function as a private key or string.

A malicious host will not learn more that a cheating proxy. The host holding secret values could be accessible by many users and it should be seen as an untrustworthy party. The data stored at the host is encrypted using the Paillier scheme, hence the confidentiality of the data depends on the strength of the cryptosystem and on the policies of the proxy about the distribution and usage of the private key. The initialization of the protocol and the translation of the data into an encrypted digital form depends on the user, located at a trusted environment.

Colluding proxy and host will not gain any additional information from each other since the host only stores the encrypted information.

Complexity. After the secret values of the user are transferred to the untrustworthy proxy, the proxy will attempt to manipulate the comparison protocol with the intention of retrieving the user's secret information. The proxy is able to decrypt the transmitted messages because he is in possession of the private key; the encryption was only intended to have a secure communication. A cheating proxy has the possibility to modify the iterative steps during the comparison. Considering two GT functions ($n = 2$), the extended ΦHA will hide two primes p_1, p_2. Knowing h, m and choosing a random $g \in QR_m$ the proxy can individually compare all primes $q_{ac,j}, q_{bc,j}$ by raising them to the power of $g^h \bmod m$. Because two primes are hidden, the proxy needs to compare a minimum of two primes per run. By creating the following set for $j = l-1, \ldots, 0$:

$$\mathcal{S} = \{(q_{ac,l-1} \cdot q_{bc,l-1}), (q_{ac,l-1} \cdot q_{bc,l-2}), \ldots, (q_{ac,0} \cdot q_{bc,0})\}.$$

he is able to find the hidden primes with $Prob[\frac{1}{|\mathcal{S}|}]$ where $|\mathcal{S}| = l^2$. This will only hold if:

$$g^{h \cdot q_{ac,l-1} \cdots q_{ac,0} \cdot q_{bc,l-1} \cdots q_{bc,0}} \equiv 1 \bmod m.$$

i.e. if it satisfies equation (6.6); otherwise the proxy does not learn anything. Generally, let n be the number of GT functions, then the total amount of primes the proxy will compute is $n \cdot l$ when using the λ function. Also the set S will grow to $|\mathcal{S}| = l^n$ because all computed primes have to be compared with each other. For that reason the probability of finding the hidden primes is

$$Prob[P = \{p_1, p_2, \ldots, p_n\} | g^{\frac{\phi(m_n)}{\prod_{i=1}^{n} p_i} \prod_{j=1}^{\eta} p_j} \equiv 1 \bmod m_n] = \frac{1}{l^n} \tag{6.7}$$

where $P = \{p|$ hidden primes p from the ΦHA$\}$ and $\eta \geq n$. From equation (6.7) we can conclude that the complexity of finding the hidden primes without any additional knowledge grows exponentially with the number of GT functions. On the other hand, the complexity of computing all primes and therefore comparing two data samples only grows linearly.

The protocol, when using n GT functions, is as efficient as using n times the private bidding protocol, because no additional communications are added. Also, the computational costs do not grow exponentially with the number of GT functions because the modulus m is dependent on l and not on n.

6.6 Discussion

Our protocol is secure for untrustworthy proxies willing to participate in the protocol i.e semi-trusted. But, when the proxy is malicious the protocol will leak some information about the relationship of the secret values. Depending on the amount of values provided by the user, the malicious proxy is able to recover the secret values. Multiple comparison of noisy samples using a malicious proxy still remains an open problem. The security of the protocol lies in the use of multiple greater than (GT) functions instead of using one GT function as was proposed in the private bidding protocol.

This chapter is based on previous work by [24] and on the research results of Chapter 5. As described, [24] is a Multi-Party Computation protocol which provides fairness within a bidding protocol for two users. The private bidding protocol seemed suitable since it assumes the usage of an Oblivious Third Party which is involved in the computation but does not provide any inputs. The disadvantage of this third-party was, that it is unable to learn any information without the aid of the user. Furthermore, the protocol was initially unsuitable for multiple comparisons since every comparison would be independent while our requirement was to have one result for the total computation.

Chapter 7

Discussion

The objective in this thesis was to provide solutions for untrustworthy proxies that have to collect and compare private information. This chapter gives a summary and discussion of the selection and collection problems and continues with the comparison problem. Finally, recommendations are given for future research.

7.1 Summary of the Results

7.1.1 Collecting Private Data

The first problem in this thesis was to research how an untrustworthy proxy can retrieve information from various sources while keeping collected information private. A user wants to retrieve information from various sources and asks a proxy to perform this task for him. The user trusts the proxy that it will execute the task correctly but does not trust him to handle his private information. The user therefore hides the name (location) of the entities he want's to query, in such a way that the proxy should still be able to collect the information from the entities. Furthermore, the collected information from the entities must remain private to the user and must not be accessible by the proxy. This problem was addressed by two approaches, parallel and sequential.

Parallel Approach. The basic concept behind the parallel selection and collection approach is that all entities are queried independently of each other by the proxy. The entities all communicate through the proxy and do not eschange information directly with each other. The privacy of the selected entities is based on cryptographic primitives. Therefore, compared to the sequential approach all entities can be queried simultaneous i.e. parallel.

The entities selected by the user to be queried, remain oblivious using an oblivious transfer protocol based on homomorphic encryption. The collected information re-

mains private to the proxy by using a different oblivious transfer based on blind signatures. In addition, the blind signatures provide the integrity of the collected information.

Sequential Approach. The second approach to the collection and sellection of private data is the sequential approach. The user selects a set of hosts he wants to collect information from and arranges them in a fixed order. This pre-defined set of hosts by the user is called an itinerary and defines the sequential order on how the hosts will communicate with each other without the intervention of the proxy.

The itinerary is sent to the proxy and kept private by means of hash chaining. Even though the proxy receives the complete itinerary he is only able to learn the location of the first entity to query since he has to provide the complete itinerary to this entity. This itinerary constructions provides the confidentiality of the users selection.

The integrity of the collected information is provided by a threshold signature scheme where the signature key is split between the hosts and the proxy. Based on this scheme the hosts cannot deny they have provided their information i.e. non-repudiation. The confidentiality of the collected information is based on public key encryption where each entity encrypts the requested information with the public key of the user.

In addition a protocol is used to aid the selection and collection protocol when one host is unable to participate within the collection process. This helper protocol ensures that the collected information remains private but needs to provide the location (name) of the host that is not responding and the location of the host that initiated the helper protocol. The helper protocol can be used when a trade-off has to be made between strict itinerary privacy and completion of the collection protocol.

Parallel and Sequential. Both approaches fulfill the requirements of the selection and collection model. But from a private computing perspective the task that is performed by the proxy is different. In the parallel approach the proxy is more involved in the complete process and truly acts as a 'man-in-the-middle'. While the proxy in the sequential approach is only involved in the process when the itinerary is stopped and at the end of the itinerary. Technically the approaches differ a lot. With the parallel approach most techniques came from multi-party computation while the sequential approach uses the ideas of mobile code and mobile software agents. But eventually, both solutions can be used complimentary to each other.

7.1.2 Comparing Private Data

The second problem in this thesis was to discover how to let an untrustworthy proxy compare private information by computing an inequality function such that the private information remains private even when the result of the computation are made public.

The user wants to compare his private information and asks the proxy to perform this task for him. The user trusts the proxy that it will execute the comparison correctly

but does not trust the proxy to handle his private information. The user therefore encrypts the private information he wants to be compared and sends this to the proxy. Another user or perhaps the same user (depending on the scenario) also submit private information to the proxy. The proxy is able to compare the received inputs by computing an inequality function. The result of the computation reveals minimum information about the provided private inputs. The proxy learns which of the inputs is the greatest without being able to learn what the actual inputs where. The problem was addressed based on two approaches, single comparison approach and multiple comparison approach.

Single Comparison Approach. The single value comparison approach is able to compute the greater than function of two encrypted natural numbers $(1, 2, \ldots)$ (secret values). The approach is based on three steps. In the first step the secret values are represented as a bit string and encrypted using ElGamal. In the second step the information is combined based on the homomorphic property of ElGamal and the use of multiple encryptions with different keys. In the final step the Φ-hiding assumption is applied to compare the encrypted values and to provide the proxy with information which of the two values is the largest. Due to the confidentiality of the private keys of the users, the proxy is unable to perform a chosen plaintext attack by generating various new encrypted inputs and using the public comparison protocol to compare them with the already known secret values. Due to the openness of the comparison protocol and the provided information by the users, the proxy is also able to learn how many bits difference the two inputs differ in plaintext values. Nevertheless, the secret values remain confidential.

Multiple Comparison Approach. The fundamental idea of encrypting bits and comparing them using the Φ-hiding assumption was also used with the multiple comparison approach. The difference between the single comparison and multiple comparison approach is that multiple values are compared with another set of values of the same length instead of one value. The approach makes it possible to compare multiple greater than function where for every comparison individually can be decided by the user whether the result should be positive for the larger value or for the smaller value. Based on this notion, the result of the computation by the proxy will only output a positive result when all comparisons are computed positive. Also, when the result is negative (because one or more greater than functions where negative), the proxy is unable to learn which greater than function was negative. Consequently, the proxy remains oblivious to the complete computation.

The result of the computation is, as with the comparison, based on the Φ-hiding assumption, which is extended to cope with the multiple values. Also, instead of using ElGamal, this approach is based on a simple threshold scheme for splitting secrets amongst multiple users together with the homomorphic public key encryption scheme of Paillier.

Single and multiple comparison. Multiple comparison is an extension to the single comparison. Using the single comparison with multiple values is not as secure as

multiple comparison, while multiple comparison will provide a similar approach when applied on a single value. Although both solutions fulfill the requirements of the comparison model, we recommend to use the multiple comparison approach when computing greater than functions with untrustworthy proxies.

7.2 Discussion

Although this thesis tried to present a complete research in private computing, still many recommendations for future research can be given.

Multi-Party Computation. A well defined problem in cryptography is Multi-Party Computation (MPC) which is comparable to private computing. But there is a fundamental difference between the two, although all solutions in this thesis share the cryptographic techniques used in MPC.

The fundamental concept of MPC is to have a set of users who all have a secret value which they do not want to reveal. There is a public function which can be computed by any users based on these secret value. The output of the function gives information about the relationship between the secret values, but does not reveal the actual inputs. The main concept of MPC is to mutually compute a public function instead of using a proxy trusted by all users. WIth private computing we want to compute the same public functions as MPC, but require it to be done by an untrustworthy proxy.

Example of applications where private computing could be applied:

- Browsing and searching within databases without revealing to the proxy and database service the information you are looking for [23].
- An encrypted High Definition television broadcast is received and transcoded by the proxy with the purpose of being re-transmitted through a low bandwidth medium, like a GSM network, to a limited processing device, like a mobile phone. The proxy has great computer power to perform the transcoding but should be unable to learn what was transcoded [130].
- Dynamic passwords instead of static passwords. With dynamic passwords you create the flexibility for the user to enter his password with a small margin of error while still granting access. For example, granting access when a user submitted almost the correct password 1234 but made a small typo and submitted 1224. This concept can be used to create visual passwords where a user points or clicks on a digital image. Since the pixels on the image are to small to be picked exactly, a small error of margin can be used to define if a user has chosen the correct points on the image or not [63].
- Firewalls (proxies) that can process and filter encrypted data without having to decrypt the content [49, 97].

Can MPC protocols be used within these applications or do we require to look for a new concept such as private computing where untrustworthy proxies are part of the scenario? For example, Yao [158] solution to the millionaires' problem lets a millionaire compute the comparison function. This user learns the result of the computed function but needs to use his secret value in plaintext to do so. This could, as it is, not be used within a private computing scenario since the proxy would learn the secret value of, at least, one millionaire.

Combining Collection and Comparison. Although both problems are described seperately in this thesis, they can be combined into one single solution where the user lets the proxy collect information from various sources and only gives the result of the comparison to the user. Combining the solutions is not an out-of-the-box task, since both approaches require specific inputs. Further research could find the right combination of solutions and even optimize the final solution with respect to efficiency and security.

Appendix A

Cryptographic Building Blocks

The following appendix provides more explanation of the cryptographic building blocks used in solving the private computing problems in this thesis. We start with homomorphic encryption which it is an important building block for computing with encrypted values. Since we are dealing with multiple parties the use of threshold cryptography and secret sharing techniques provides a valuable approach for collecting distributed information. We continue with an important primitive within Secure Function Evaluation and Multi-Party Computation, namely Oblivious Transfer. When the integrity of the collected information from various parties has to be provided, a hash chaining algorithm could be used. Finally, the $\Phi-$Hiding Assumption is described which could be used as a building block for comparing private data.

A.1 Homomorphic Encryption

An encryption algorithm $E()$ is homomorphic if given two ciphertexts $E(m_1)$ and $E(m_2)$, it is possible to obtain a third ciphertext $E(m_1 \bigodot m_2)$ without decrypting $E(m_1)$ and $E(m_2)$ (in order to compute $E(m_1 \odot m_2)$) for some operation $\bigodot$. More specifically, let $\bigodot_M$ denote a binary operator in the plaintext space M and $\bigodot_C$ a binary operation in the ciphertext space C. Then the encryption algorithm is said to be homomorphic if it satisfies:

$$\forall\ m_1, m_2 \in M, \quad E(m_1 \bigodot_M m_2) = E(m_1) \bigodot_C E(m_2) \tag{A.1}$$

Where the '=' means "can be directly computed from" without any intermediate decryption [67]. Note that sometimes $\bigodot_M = \bigodot_C$.

There are two groups of homomorphic cryptosystems, ones based on the *decisional composite residuosity assumption* (DCRA) and ones based on the *decisional Diffie-Hellman assumption* (DDH). DCRA states that given an integer z and a composite n (positive integer which has positive divisors other than one or itself), it is hard to

decide whether there exists y such that $z \equiv y^n \pmod{n^2}$. Considers a cyclic group G of order q with generator g. The DDH assumption state that given g^a and g^b for randomly chosen $a, b \in \mathbb{Z}_q$, the value g^{ab} is a random element of the cyclic group G. Furthermore, homomorphic cryptosystems can be probabilistic or deterministic.

A homomorphic encryption is additive if there is a suitable binary operation $\oplus$ such that:

$$E(m_1 + m_2) = E(m_1) \oplus E(m_2) \tag{A.2}$$

Or muliplicative if there is a suitable binary operation $\otimes$ such that:

$$E(m_1 \times m_2) = E(m_1) \otimes E(m_2) \tag{A.3}$$

The RSA cryptosystem [131] and the ElGamal cryptosystem [62] are multiplicative homomorphic. The Goldwasser-Micali cryptosystem [81], the Paillier cryptosystem [125] and the Damgård-Jurik cryptosystem [52] which is a generalization of the Paillier cryptosystem, are all additive homomorphic.

A.1.1 RSA

The RSA cryptosystem [131] was one of the first multiplicative homomorphic encryption schemes. The system lets a user compute an integer $n = pq$ where p and q are chosen large prime numbers. The user also computes an integer e such that $gcd(e, \phi(n)) = 1$ and an integer d which is the inverse of e mod $\phi(n)$ i.e. $ed = 1 \bmod \phi(n)$ where $\phi(n)$ denotes the Euler function $\phi(n) = \phi(pq) = (p-1)(q-1)$. The Euler totient function $\phi(n)$ is defined as the number of positive integers $\leq n$ which are relative prime to n. The public key of the user becomes (n, e) and the private key is d, furthermore the users keeps both p and q secret.

To demonstrate the homomorphic property of RSA one has to encrypt the messages m_1, m_2 according to $E_1 = E(m_1) = {m_1}^e \bmod N$ and $E_2 = E(m_2) = {m_2}^e \bmod N$. Then the multiplication of $E_1 \times E_2$ becomes ${m_1}^e \times {m_2}^e \bmod N = (m_1 \times m_2)^e \bmod N$. This shows the multiplicative homomorphic property: $E(m_1 \times m_2) = E(m_1) \otimes E(m_2)$.

A.1.2 ElGamal

ElGamal [62] is a multiplicative homomorphic cryptosystem. In order to generate a key-pair the user choses a large prime integer p and a generating element g of the cyclic group ${Z_p}^*$. The user picks a random $x \in Z_q$ with the order of the group $q = p - 1$ and computes $h = g^x$ in ${Z_p}^*$. The public key becomes (g, q, h) and the private key is x.

Encrypting the messages m_1, m_2 we pick the randoms values $k_1, k_2 \in Z_q$ and compute $E_1 = (a, b) = (g^{k_1}, m_1 h^{k_1})$, $E_2 = (c, d) = (g^{k_2}, m_2 h^{k_2})$. The multiplication of $E_1 \times E_2 = (a, b) \times (c, d) = (g^{k_1}, m_1 h^{k_1}) \times (g^{k_2}, m_2 h^{k_2})$ becomes $(ac, bd) = (g^{k_1 + k_2}, (m_1 m_2) h^{k_1 + k_2})$. This shows the multiplicative property of ElGamal: $E(m_1 \times m_2) = E(m_1) \otimes E(m_2)$.

A.1.3 Paillier

The Paillier cryptosystem [125] is a probabilistic additive homomorphic cryptosystem where probabilistic means that an additional uniform random number is used as input. If not, then it's called deterministic. Comparable to RSA the user picks two primes p,q and let $n = pq$ where n is satisfying $gcd(n,\phi(n)) = 1$. Note that all elements have order dividing $\phi(n^2) = n \cdot \phi(n) = n \cdot \phi(p) \cdot \phi(q)$. The public key becomes (n,g) where g has order multiple of n. The private key becomes $\lambda(n)$ where $\lambda(n) = lcm(p-1,q-1)$. $\lambda(n)$ is the Carmichael function (defines the exponent of the multiplicative group of integers modulo n) and 'lcm' is the lowest common multiple.

To encrypt the message $m_1,m_2 \in Z_N$ we choose $x_1,x_2 \in Z_N{}^*$ and compute $E(m_1) = g^{m_1}x_1{}^n \bmod n^2$,$E(m_2) = g^{m_2}x_2{}^n \bmod n^2$. The multiplication of $E(m_1)$ and $E(m_2)$ leads to $E(m_1+m_2) = g^{m_1+m_2}x_1x_2{}^n \bmod n^2$. This demonstrates the additive homomorphic property of Paillier: $E(m_1+m_2) = E(m_1) \oplus E(m_2)$.

A.1.4 Damgård-Jurik cryptosystem

The Damgård-Jurik cryptosystem [52] is a generalization and adaptation of the Pallier cryptosystem [125] based on the decisional composite residuosity assumption (DCRA) as described in Section A.1. It uses computations module n^{s+1} where $s = 1$ leads to the Paillier's scheme.

A.2 Threshold Cryptography

Using threshold cryptographic systems like [51, 52] a message is encrypted with a public key while the corresponding private key is shared among a group of parties where only a qualified subset of parties is allowed to decrypt the encrypted message. In a comparable way it is possible to create threshold signature schemes [53, 54] where a message is signed by a subset of parties using a shared private key while the signature of the message can be verified by anybody using the corresponding public key.

A.2.1 Secret Sharing

When sharing a secret F among a set of n participants in such a way that t participants can compute the value F but no subset smaller than t can reconstruct F, it is called a (t,n)-threshold scheme. The first threshold scheme was proposed in 1979 by Shamir [142] and many others have followed.

In *secret splitting* a secret F is split among n participants so that all n participants are needed to reconstruct the secret i.e. secret splitting is a (n,n)-threshold scheme. A simple example from [154] is to split the secret number F among n participants. Give

$n-1$ participants each one of the random number $r_1, \ldots, r_{n-1} \bmod m$. The remaining participant receives $F - \sum_{\pi=1}^{n-1} r_\pi (\bmod m)$ (m is an integer larger than all possible messages).

A similar scheme can be used for *secret binding*, which has the purpose to let a user N add (bind) secrets together, learning only the complete secret F if all partial secrets are joint together; without being able to compute the partial secrets. To achieve this, all n participants have to share a secret k which should not be available to user N. In the following example user A has a secret F_A, user B has a secret F_B and they both share a secret key k. They want to bind these secrets together by addition which results in the complete secret $F = F_A + F_B$. This addition should be performed by user C in such a way that he does not learn F_A, F_B. Therefore, user A hides his secret and gives $F_A + k$ to user C. User B provides $F_B - k$ to user C which is able to compute $F_A + k + F_B - k = F$ without learning anything else but F.

When using a additive homomorphic encryption scheme like [125, 126] it is possible to bind the secrets together without decrypting them. This is also used in our solution in chapter 6, where user A provides two noisy samples which should remain unknown to user B; but he should be able to bind those secret values together and use our protocol to match them.

A.3 Oblivious Transfer (OT)

Oblivious transfer was first introduced by Rabin [128]. The primitive captures the notion of a protocol by which a sender sends some information to the receiver, but remains oblivious as to what is sent. The paradox is resolved by recognizing that it are the actions of the *receiver and the sender* that determine the outcome of the protocol. Even *et al.* [65] generalized it to 1-out-of-2 oblivious transfer (OT_1^2). The receiver determines which message out of two possible messages she is going to receive. In turn it was shown how to construct OT_1^n from n [20] and even $\log n$ [116] applications of OT_1^2. [3, 94, 118] provided direct constructions for OT_1^n based on the decision Diffie-Hellman and quadratic residuosity assumptions.

A.3.1 Adaptive OT from Blind Signatures

Adaptive oblivious transfer protocols were proposed in [45, 117, 121]. Camenisch et al. [27] recognized that the schemes in [45, 121] are based on a common principle to construct adaptive oblivious transfer from unique blind signature schemes, first described by Chaum in [34].

Suppose, all messages $m_1, \ldots, m_n$ are symmetrically encrypted by the sender using the hashed signature of the index i, $1 \leq i \leq n$, as the key. Thus $C_i = E_{H(Sign_{sk}(i))}(m_i)$. H is a symmetric hash function, E is a symmetric cipher, *Sign* is a unique blind signature scheme, and *sk* is the signing key of the sender that will be used for creating the blind

signature. The ciphertexts $C_1,\ldots,C_n$ are transferred from the sender to the receiver. In order to obtain message m_k, where $1 \leq k \leq n$, the receiver runs a blind signature protocol with the sender on index k to obtain the symmetric key $H(Sign_{sk}(k))$. The receiver blinds the index k and sends the result $Blind(k,b)$ with some random number b to the sender. The sender signs the received message $Sign_{sk}(Blind(k,b))$ with the signing key of the sender and sends the result back to the receiver. The receiver unblinds the message $Unblind(Sign_{sk}(Blind(k,b)),b)$ using the blinding factor b. The blind signature protocol is designed to reduce the message to $Sign_{sk}(k)$, which can be used to obtain message m_k.

A.3.2 OT using Homomorphic Encryption

Additive homomorphic encryption has the homomorphic addition of plaintexts property e.g using Paillier as described in Section A.1.3. Given the encryptions $E(m_1)$ and $E(m_2)$, the multiplication of the two ciphertexts will lead to $E(m_1)\cdot E(m_2) = E(m_1)\oplus E(m_2) = E(m_1+m_2)$. Furthermore, the additive homomorphic encryption of Paillier also has the property of homomorphic multiplication of plaintexts. Given an encryption $E(m)$ and constant k and raising the encryption to a constant k will result in $k\otimes E(m) = E(m)^k = \prod_{j=1}^{k} E(m)_j = E(k\cdot m)$.

Ostrovsky et al. [123] recognized that given the encryption $E(1)$ it is also possible to compute an encryption message m as $m\otimes E(1) = E(1)^m = \prod_{j=1}^{m} E(1)_j = E(m\cdot 1) = E(m)$. However, if the encryption is given as $E(0)$ the same operation does not change anything i.e. $m\otimes E(0) = E(0)^m = \prod_{j=1}^{m} E(0)_j = E(m\cdot 0) = E(0)$.

Suppose, a sender has n messages $m_1,\ldots,m_n$. The receivers wants to obtain message m_k with index k where $1 \leq k \leq n$. He constructs a vector $E_1,\ldots,E_n$ where all encryptions are $E(0)$ except index k, which is $E_k(1)$ e.g. $E_1(0),\ldots,E_k(1),\ldots,E_n(0)$. The sender receives the vector from the receiver and computes the oblivious transfer based on the homomorphic property, $E_1(0)^{m_1},\ldots,E_k(1)^{m_k},\ldots,E_n(0)^{m_n}$, which results in $E_1(0),\ldots,E_k(m_k),\ldots,E_n(0)$. Finally, the sender computes the product of the result $E_1(0)\cdot E_2(0)\cdot\ldots\cdot E_k(m_k)\cdot\ldots\cdot E_n(0) = E_k(m_k)$ to reduce the complexity of the communication. The receiver collects the result of the oblivious transfer and learns m_k. The sender does not learn which message was collected by the receiver.

A.4 Hash Chaining

Hash Chaining is a method of providing integrity of data when this data is being accessed by multiple parties. Hash Chains are based on hash functions which are functions that compress an input of arbitrary length to a result with a fixed length. The following method of hash chaining is taken from [143]. A user is visiting various hosts $\mathcal{L}_j$ where he collects information which has to remain private. Using Hash Chaining the user is able to store the collected information o_j in an so called encrypted storage. Furthermore, the host also makes a commitment that he adds it to the correct storage,

by including a hash of the previous storage state i.e. the storage collected so far. Each host $\mathcal{L}_j$ will follow the following protocol for storing its offer o_j to the storage.

- Encapsulated offer:

$$O_j = Sign_{K_j}(E_{pk_Q}(o_j, r_j), H_j), 0 \leq j \leq n \tag{A.4}$$

- Chaining relation:

$$H_0 = h(o_0, \mathcal{L}_1) \tag{A.5}$$
$$H_j = h(O_{j-1}, \mathcal{L}_{j+1}), 1 \leq j \leq n \tag{A.6}$$

Sign is the signing algorithm, K_j is the signing key of Host j, o_0 is initial information (e.g. identitiy of the collector), o_j the offer of host $\mathcal{L}_j$, O_j the encapsulated offer from host $\mathcal{L}_j$, r_j a random number generated by $\mathcal{L}_j$, $E_{pk_Q}(v)$ is the encryption of value v with the public key of the final receiver Q of the information and $h(.)$ a cryptographic hash function. The encrypted storage will consist of the chain $O_0, O_1, \ldots, O_n$, where n is the total amount of hosts visited.

The essence of the protocol is that a host $\mathcal{L}_j$ signs both its offer and a hash value taken over the last encapsulated offer and the next destination of the itinerary. If a malicious host $\mathcal{L}_j$ would like to delete, for example, an offer from the storage, then this will be detected during verification of the hash chain because the committed value will fail the verification.

A.5 Φ-Hiding Assumption

The Φ-Hiding Assumption (Φ-HA), by [25], states the difficulty of deciding whether a small prime divides $\phi(m)$, where m is a composite integer of unknown factorization. The Φ-HA is used as follows: Choose randomly m that hides a prime p_0 so it is hard to distinguish if p_0 or p_1 is a factor of $\phi(m)$, where p_1 is chosen randomly and independently i.e. it is computationally indistinguishable.

Let us choose $m = p'q'$ where p' is a safe prime and q' is a quasi-safe prime. Prime p' is a safe prime if $\frac{p'-1}{2}$ is also a prime. An odd prime q' is quasi-safe if $q' = 2\hat{p}+1$ where $\hat{p}$ is a prime power, i.e. $\hat{p} = p^{\alpha}$ for some prime p. We compute $p' = 2q_1 + 1$ and $q' = 2pq_2 + 1$, where q_1 is a prime and p, q_2 are odd primes.

The Φ-HA continues with the Euler totient theorem:

$$g^{\phi(m)} \equiv 1 \bmod m \tag{A.7}$$

where $g \in \mathbb{Z}$ and m is a composite integer. To hide the prime p the Euler totient function is computed with the chosen safe and quasi-safe primes:

$$\begin{aligned} \phi(m) &= \phi(p'q') = (p'-1)(q'-1) \\ &= (2q_1 + 1 - 1)(2pq_2 + 1 - 1) \\ &= (2q_1)(2pq_2) = 4pq_1q_2 \end{aligned} \tag{A.8}$$

Combining (A.7) and (A.8) we can compute:

$$g^{4pq_1q_2} \equiv 1 \bmod m \tag{A.9}$$

The Φ-HA assumption can be used to determine if from a set of primes $\mathcal{P} = \{p_1, \ldots, p_n\}$ where $n \geq 0$, one of the primes is the hidden prime p or not. The assumption also works if the product of the set of primes is given, i.e. $\prod_{i=0}^{n} p_i$:

$$g^{\frac{4pq_1q_2}{p} \prod_{i=0}^{n} p_i} \equiv 1 \bmod m \tag{A.10}$$

If A.10 is congruent to 1 modulo m then the set $\mathcal{P}$ holds (hides) p else is does not. Note that m hides p if and only if p divides $phi(m)$. It is important that the $g \in \mathbb{Z}$ should not be a *p-th root* module m i.e. there exists no μ such that $\mu^p \equiv g \bmod m$. Therefore, we should choose $g \in QR_m$ (Quadratic Residue of m) because p should not be an even prime.

Bibliography

[1] M. Abadi and J. Feigenbaum. Secure circuit evaluation. *Journal of Cryptology*, 2(1):5–21, 1990.

[2] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 563–574, New York, NY, USA, 2004. ACM.

[3] W. Aiello, Y. Ishai, and O. Reingold. Priced oblivious transfer: How to sell digital goods. In B. Pfitzmann, editor, *Advances on Cryptology — EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 119–135, Innsbruck, Austria, 6–4 May 2001. Springer-Verlag.

[4] J. Algesheimer, C. Cachin, J. Camenisch, and G. Karjoth. Cryptographic security for mobile code. *Proceedings 2001 IEEE Symposium on Security and Privacy, IEEE*, pages 2–11, 2001.

[5] F. Alizadeh-Shabdiz and S. Subramaniam. Analytical models for single-hop and multi-hop ad hoc networks. *Mob. Netw. Appl.*, 11(1):75–90, 2006.

[6] N. Asokan, Matthias Schunter, and Michael Waidner. Optimistic protocols for fair exchange. In *CCS '97: Proceedings of the 4th ACM conference on Computer and communications security*, pages 7–17, New York, NY, USA, 1997. ACM Press.

[7] N. Asokan, Victor Shoup, and Michael Waidner. Asynchronous protocols for optimistic fair exchange. *sp*, 00:0086, 1998.

[8] M. Atallah, K Frikken, and C. Zhang. Privacy-preserving credit checking. Technical report, Purdue University, 2005.

[9] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. *Crypto 2001, Lecture Notes in Computer Science*, pages 1–18, 2001.

[10] B. Barni, P. Failla, V. Kolesnikov, R. Lazzeretti, A-R. Sadeghi, and T. Schneider. Secure evaluation of private linear branching programs with medical applications. In *Computer Security - ESORICS 2009, 14th European Symposium on*

Research in Computer Security, Saint-Malo, France, September 21-23, 2009. Proceedings, pages 424–439, 2009.

[11] E. Beckenbach and R. Bellman. *An Introduction to Inequalities*. Random House, New York, 1961.

[12] A. Beimel, Y. Ishai, and E. Kushilevitz. General constructions for information-theoretic private information retrieval. *Journal of Computer Systems Sciences*, 71(2):247–281, 2005.

[13] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10, 1988.

[14] J. Benaloh. Dense probabilistic encryption. In *In Proceedings of the Workshop on Selected Areas of Cryptography*, pages 120–128, 1994.

[15] A.R. Beresford and F Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, 2003.

[16] F. Bignami. Protecting privacy against the police in the european union: The data retention directive. Technical Report 13, Duke Law School – Science, Technology and Innovation, January 2007.

[17] A. Biryukov, O. Dunkelman, N. Keller, D. Khovratovich, and A. Shamir. Key recovery attacks of practical complexity on aes variants with up to 10 rounds. Cryptology ePrint Archive, Report 2009/374, 2009. http://eprint.iacr.org/.

[18] G.W. van Blarkom, J.J. Borking, and J.G.E. Olk, editors. *Handbook of Privacy and Privacy-Enhancing Technologies, The case of intelligent software agents*. College Bescherming Persoonsgegevens, 2003. Chapter 3, PET, G.W. Blarkom, J.J. Borking and P. Verhaar, pages 33–54.

[19] D. Boneh, G. Di-Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 506–522, 2004.

[20] G. Brassard, C. Crépeau, and J-M. Robert. All-or-nothing disclosure of secrets. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 234–238. Springer, 1986.

[21] J. Brickell, D.E. Porter, V. Shmatikov, and E. Witchel. Privacy-preserving remote diagnostics. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 498–507, New York, NY, USA, 2007. ACM.

[22] J. Brickell and V. Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In *Advances in Cryptology - ASIACRYPT 2005, 11th Interna-*

tional Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings, pages 236–252, 2005.

[23] R. Brinkman, J. Doumen, and W. Jonker. Using secret sharing for searching in encrypted data. In *Secure Data Management, VLDB 2004 Workshop, SDM 2004, Toronto, Canada, August 30, 2004, Proceedings*, pages 18–27, 2004.

[24] C. Cachin. Efficient private bidding and auctions with an oblivious third party. In *CCS '99: Proceedings of the 6th ACM conference on Computer and communications security*, pages 120–127, New York, NY, USA, 1999. ACM Press.

[25] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. *Lecture Notes in Computer Science*, 1592:402–414, 1999.

[26] J. Camenisch, M. Dubovitskaya, and G. Neven. Oblivious transfer with access control. In *CCS '09: Proceedings of the 16th ACM conference on Computer and communications security*, pages 131–140, New York, NY, USA, 2009. ACM.

[27] J. Camenisch, G. Neven, and A. Shelat. Simulatable adaptive oblivious transfer. In *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings*, pages 573–590, 2007.

[28] R. Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.

[29] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 639–648, New York, NY, USA, 1996. ACM.

[30] K. Cartrysse. *Private Computing and Mobile Code Systems*. PhD thesis, Delft University of Technology, November 2005. ISBN 90-90199-53-5.

[31] K. Cartrysse and J.C.A. van der Lubbe. Privacy in mobile agents. *First IEEE Symposium on Multi-Agent Security and Survivability*, August 2004.

[32] D. Catalano, R. Gennaro, N. Howgrave-Graham, and P. Q. Nguyen. Paillier's cryptosystem revisited. In *ACM Conference on Computer and Communications Security*, pages 206–214, 2001.

[33] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.

[34] D. Chaum. Blind signatures for untraceable payments. In *CRYPTO*, pages 199–203, 1982.

[35] D. Chaum. Security without identification: transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.

[36] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (abstract). In *CRYPTO '87: A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology*, page 462, London, UK, 1988. Springer-Verlag.

[37] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Advances in Cryptology - Crypto'88*, pages 319–327, 1990.

[38] M. Chen, S. Gonzalez, and V.C.M. Leung. Applications and design issues of mobile agents in wireless sensor networks. *IEEE Wireless Communications*, 14(6):20–26, December 2007.

[39] M. Chen, T. Kwon, Y. Yuan, and V.C.M. Leung. Mawsn: Mobile agent based wireless sensor networks. *Journal of Computers*, 1(1):14–21, Apr 2006.

[40] W. Chen and Y. Zhang. A multi-constrained routing algorithm based on mobile agent for manet networks. In *JCAI '09: Proceedings of the 2009 International Joint Conference on Artificial Intelligence*, pages 16–19, Washington, DC, USA, 2009. IEEE Computer Society.

[41] B. Chor and N. Gilboa. Computationally private information retrieval (extended abstract). In *STOC*, pages 304–313, 1997.

[42] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *FOCS '95: Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, page 41, Washington, DC, USA, 1995. IEEE Computer Society.

[43] S. Chow, P. Eisen, H. Johnson, and P.C. van Oorschot. White-box cryptography and an aes implementation. In *Proceedings of the Ninth Workshop on Selected Areas in Cryptography (SAC 2002)*, pages 250–270. Springer LNCS 2595 (2003), 2002.

[44] S. S. M. Chow, J-H. Lee, and L. Subramanian. Two-party computation model for privacy-preserving queries over distributed databases. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2009, San Diego, California, USA, 8th February - 11th February 2009*, 2009.

[45] C.K. Chu and W.G. Tzeng. Efficient -out-of- oblivious transfer schemes with adaptive and non-adaptive queries. In S. Vaudenay, editor, *Public Key Cryptography*, volume 3386 of *Lecture Notes in Computer Science*, pages 172–183. Springer-Verlag, 2005.

[46] J. Claessens, B. Preneel, and J. Vandewalle. (how) can mobile agents do secure electronic transactions on untrusted hosts? a survey of the security issues and the currenct solutions. *ACM Transactions on Internet Technology*, 3(1):28–48, February 2003.

[47] European Community. Compete ceo: Isps sell clickstreams for $5 a month, 2007.

[48] European Community. The data retention (ec directive) regulations 2009, 2009.

[49] Cymphonix Corporation. How to prevent secure web traffic (https) from crippling your content filter - a cymphonix white paper, 2007.

[50] R. Cramer, I. Damgård, and J.B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *EUROCRYPT '01: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, pages 280–299, London, UK, 2001. Springer-Verlag.

[51] I. Damgård and M. Jurik. Efficient protocols based on probabilistic encryption using composite degree residue classes, 2000.

[52] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *PKC '01: Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography*, pages 119–136. Springer-Verlag, 2001.

[53] Y. Desmedt. Society and group oriented cryptography: A new concept. In *CRYPTO '87: A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology*, pages 120–127, London, UK, 1988. Springer-Verlag.

[54] Y. Desmedt and Y. Frankel. Shared generation of authenticators and signatures (extended abstract). In *CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, pages 457–469, London, UK, 1992. Springer-Verlag.

[55] Y.G. Desmedt and Y. Frankel. Perfect homomorphic zero-knowledge threshold schemes over any finite abelian group. *SIAM J. Discret. Math.*, 7(4):667–679, 1994.

[56] A. Dey, S. Lederer, and J. Mankoff. Towards a deconstruction of the privacy space. In *In Proc. Workshop on Ubicomp Communities: Privacy as Boundary Negotiation*, 2003.

[57] G. Di-Crescenzo, T. Malkin, and R. Ostrovsky. Single database private information retrieval implies oblivious transfer. In *EUROCRYPT*, pages 122–138, 2000.

[58] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[59] J. Domingo-Ferrer. A new privacy homomorphism and applications. *Inf. Process. Lett.*, 60(5):277–282, 1996.

[60] J. Domingo-Ferrer and Y. Saygin. Recent progress in database privacy. *Data Knowl. Eng.*, 68(11):1157–1159, 2009.

[61] W. Du and M.J. Atallah. Secure multi-party computation problems and their applications: A review and open problems. In *In New Security Paradigms Workshop*, pages 11–20, 2001.

[62] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 10–18. Springer-Verlag New York, Inc., 1984.

[63] Mininova Labs Erik. Passclicks - visual passwords, June 2010.

[64] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft. Privacy-preserving face recognition. In *PETS '09: Proceedings of the 9th International Symposium on Privacy Enhancing Technologies*, pages 235–253, Berlin, Heidelberg, 2009. Springer-Verlag.

[65] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.

[66] H. Federrath, A. Jerichow, D. Kesdogan, and A. Pfitzmann. Security in public mobile communication networks. In *IFIP TC 6 International Workshop on Personal Wireless Communications*, pages 105–116, 1995.

[67] C. Fontaine and F. Galand. A survey of homomorphic encryption for nonspecialists. *EURASIP J. Inf. Secur.*, 2007:1–15, 2007.

[68] M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology — EUROCRYPT 2004*, 2004.

[69] K. B. Frikken, J. Li, and M.J. Atallah. Trust negotiation with hidden credentials, hidden policies, and policy cycles. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2006, San Diego, California, USA*, 2006.

[70] Keith B. Frikken. Practical private dna string searching and matching through efficient oblivious automata evaluation. In *Proceedings of the 23rd Annual IFIP WG 11.3 Working Conference on Data and Applications Security XXIII*, pages 81–94, Berlin, Heidelberg, 2009. Springer-Verlag.

[71] J.A. Garcia-Macias and J. Gomez-Castellanos. *Sensor Networks and Configuration: Fundamentals, Standards, Platforms, and Applications.*, chapter MANET versus WSN. Springer-Verlag, Germany, 2006.

[72] B. Gedrojc, K. Cartrysse, and J.C.A. van der Lubbe. Private bidding for mobile agents. In *SECRYPT 2006, Proceedings of the International Conference on Security and Cryptography, Setúbal, Portugal, August 7-10, 2006, SECRYPT is part of ICETE - The International Joint Conference on e-Business and Telecommunications*, pages 277–282. INSTICC Press, 2006.

[73] B. Gedrojc, M. van Hensbergen, and J.C.A. van der Lubbe. Private computing with beehive organized agents. In *SECRYPT 2007, Proceedings of the International Conference on Security and Cryptography, Barcelona, Spain, July 28-31,*

2007, SECRYPT is part of ICETE - The International Joint Conference on e-Business and Telecommunications. INSTICC Press, 2007.

[74] B. Gedrojc and J.C.A. van der Lubbe. Private matching of noisy data". In *Proceedings of the 27th Symposium on Information Theory in the Benelux*, pages 275–282, June 2006.

[75] R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. Cryptology ePrint Archive, Report 2009/547, 2009. http://eprint.iacr.org/.

[76] E-J. Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216, 2003. http://eprint.iacr.org/.

[77] O Goldreich. Secure multi-party computation. Final (incomplete) Draft, Version 1.4, October 27 2002.

[78] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC '87: Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 218–229, New York, NY, USA, 1987. ACM.

[79] D. Goldschlag, M. Reed, and P. Syverson. Onion routing. *Commun. ACM*, 42(2):39–41, 1999.

[80] S. Goldwasser. Multi party computations: past and present. In *PODC '97: Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, pages 1–6, New York, NY, USA, 1997. ACM.

[81] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[82] Object Management Group; Agent Platform Special Interest Group. Omg document agent/00-09-01, September 2000.

[83] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of First International Conference on Mobile Systems, Applications, and Services (MobiSys'03)*, pages 31–42, 2003.

[84] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra. Executing sql over encrypted data in the database-service-provider model. In *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 216–227, New York, NY, USA, 2002. ACM.

[85] H. Hacigümüs, B. Iyer, and S. Mehrotra. Efficient execution of aggregation queries over encrypted relational databases. In *Database Systems for Advances Applications, 9th International Conference, DASFAA 2004, Jeju Island, Korea, March 17-19, 2004, Proceedings*, pages 125–136, 2004.

[86] J. M. Hellerstein, Y. Li, and J. D. Tygar. Private matching. In *In Computer Security in the 21st Century*, pages 25–50. Springer, 2004.

[87] M. van Hensbergen, B. Gedrojc, and J.C.A. van der Lubbe. A beehive approach to e-commerce mobile agents. In *Proceedings of the 28th Symposium on Information Theory in the Benelux*, May 2007.

[88] M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In *Proceedings of EuroCrypt 2000, LNCS series*, pages 539–556. Springer-Verlag, 2000.

[89] Fox-IT Experts in Security. Small white paper – fort fox data diode. *http://www.datadiode.eu/uploads/whitepaper/foxit_ffdd_whitepapersmall.pdf*, 2008.

[90] M. Jakobsson, A. Juels, and R.L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proceedings of the 11th USENIX Security Symposium*, pages 339–353, Berkeley, CA, USA, 2002. USENIX Association.

[91] S. Jarecki and V. Shmatikov. Efficient two-party secure computation on committed inputs. In *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings*, pages 97–114, 2007.

[92] S. Jha, L. Kruger, and V. Shmatikov. Towards practical privacy for genomic computation. In *SP '08: Proceedings of the 2008 IEEE Symposium on Security and Privacy*, pages 216–230, Washington, DC, USA, 2008. IEEE Computer Society.

[93] M. Joye. On white-box cryptography. In *Security of Information and Networks, Eds. A. Elci, S.B. Ors, and B. Preneel*, pages 7–12, 2009.

[94] Y. T. Kalai. Smooth projective hashing and two-message oblivious transfer. In R. Cramer, editor, *Advances on Cryptology — EUROCRYPT 2005, Aarhus, Denmark*, Lecture Notes in Computer Science, pages 78–95. Springer-Verlag, 22–26 May 2005.

[95] G. Karjoth, N. Asokan, and G. Gülcü. Protecting the computation results of free-roaming agents. *Mobile agents '98, Lecture Notes in Computer Science*, pages 195–207, 1998.

[96] D. Kesdogan, M. Borning, and M. Schmeink. Unobservable surfing on the world wide web: is private information retrieval an alternative to the mix based approach? In *PET'02: Proceedings of the 2nd international conference on Privacy enhancing technologies*, pages 224–238, Berlin, Heidelberg, 2003. Springer-Verlag.

[97] B. Kleineidam. Webcleaner, June 2010.

[98] M. Kohlweiss, S. Faust, L. Fritsch, B. Gedrojc, and B. Preneel. Efficient oblivious augmented maps: Location-based services with a payment broker. In N. Borisov and P. Golle, editors, *Privacy Enhancing Technologies, 7th International Symposium, PET 2007 Ottawa, Canada, June 20-22, 2007, Revised*

Selected Papers, volume 4776 of *Lecture Notes in Computer Science*, pages 77–94. Springer, 2007.

[99] M. Kohlweiss and B. Gedrojc. Privacy friendly location based service protocols using efficient oblivious transfer. *Kryptowochenende 2006 - Workshop uber Kryptographie, pp. 29 - 32, July*, 2006.

[100] T. Kölsch, L. Fritsch, M. Kohlweiss, and D. Kesdogan. Privacy for profitable location based services. In D. Hutter and M. Ullmann, editors, *Security in Pervasive Computing, Second International Conference, SPC 2005, Boppard, Germany, April 6-8, 2005, Proceedings*, volume 3450 of *Lecture Notes in Computer Science*, pages 164–178. Springer, 2005.

[101] E. Kushilevitz and R. Ostrovsky. Replication is not needed: single database, computationally-private information retrieval. In *FOCS '97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, page 364, Washington, DC, USA, 1997. IEEE Computer Society.

[102] C. Lai, L. Gong, L. Koved, A. Nadalin, and R. Schemers. User authentication and authorization in the Java platform. In *15th Annual Computer Security Applications Conference*, pages 285–290. IEEE Computer Society Press, 1999.

[103] L. Lamport. Password authentication with insecure communication. *Commun. ACM*, 24:770–772, November 1981.

[104] H.Y. Lin and W.G. Tzeng. An efficient solution to the millionaires' problem based on homomorphic encryption. In *In ACNS 2005, volume 3531 of Lecture*, pages 456–466, 2005.

[105] Y. Lindell and B. Pinkas. Privacy preserving data mining. *J. Cryptology*, 15(3):177–206, 2002.

[106] Y. Lindell and B. Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *EUROCRYPT*, pages 52–78, 2007.

[107] Y Lindell and B Pinkas. Secure multiparty computation for privacy-preserving data mining. *Journal of Privacy and Confidentiality*, 1(1), 2009.

[108] H.E. Link and W.D. Neumann. Clarifying obfuscation: Improving the security of white-box des. *Information Technology: Coding and Computing, International Conference on*, 1:679–684, 2005.

[109] H. Lipmaa. Verifiable homomorphic oblivious transfer and private equality test. In Chi Sung Laih, editor, *Advances on Cryptology — ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 416–433, Taipei, Taiwan, November 30–December 4 2003. Springer-Verlag.

[110] S. Loureiro, R. Molva, and A. Pannetrat. Secure data collection with updates. *Electronic Commerce Research*, 1(1-2):119–130, 2001.

[111] A. Lysyanskaya and N. Triandopoulos. Rationality and adversarial behavior in multi-party computation. In *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, pages 180–197, 2006.

[112] Q. Ma and P. Deng. Secure multi-party protocols for privacy preserving data mining. In *WASA '08: Proceedings of the Third International Conference on Wireless Algorithms, Systems, and Applications*, pages 526–537, Berlin, Heidelberg, 2008. Springer-Verlag.

[113] K. Mehta, D. Liu, and M Wright. Location privacy in sensor networks against a global eavesdropper. *Network Protocols, IEEE International Conference on*, 0:314–323, 2007.

[114] K. Mehta, D. Liu, and M Wright. Protecting location privacy in sensor networks against a global eavesdropper. *IEEE Transactions on Mobile Computing*, 99(PrePrints), 2011.

[115] A. J. Menezes, S. A. Vanstone, and P. C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1996.

[116] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *STOC '85: Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 245–254, Atlanta, Georgia, USA, 1–4 May 1999. ACM.

[117] M. Naor and B. Pinkas. Oblivious transfer with adaptive queries. In *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 573–590, London, UK, 1999. Springer-Verlag.

[118] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *SODA '01: Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 448–457, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.

[119] C. Nicol. Ict policy: A beginner's handbook. *http://rights.apc.org/handbook/*, 2003.

[120] J.B. Nielsen and C. Orlandi. Lego for two-party secure computation. In *TCC '09: Proceedings of the 6th Theory of Cryptography Conference on Theory of Cryptography*, pages 368–386, Berlin, Heidelberg, 2009. Springer-Verlag.

[121] W. Ogata and K. Kurosawa. Oblivious keyword search. *Journal of Complexity*, 20(2-3):356–371, 2004.

[122] T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In *EUROCRYPT*, pages 308–318, 1998.

[123] R. Ostrovsky and W.E. Skeith, III. Private searching on streaming data. In *CRYPTO 2005*, pages 223–240, 2005.

[124] Open Wep Application Security Project (OWASP). Man-in-the-browser attack, April 2009.

[125] P. Paillier. Public-key cryptosystem based on composite degree residuosity classes. In J. Stern, editor, *Advances on Cryptology — EUROCRYPT 1999*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–228, Prague, Czech Republic, 2–6 May 1999. Springer-Verlag.

[126] P. Paillier and D. Pointcheval. Efficient public-key cryptosystems provably secure against active adversaries. In K.Y. Lam, E. Okamoto, and C. Xing, editors, *Advances on Cryptology — ASIACRYPT 1999*, volume 1716 of *Lecture Notes in Computer Science*, pages 165–179, Singapore, 14–18 november 1999. Springer-Verlag.

[127] C. Prins. When personal data, behavior and virtual identities become a commodity: Would a property rights approach matter?, volume 3, issue 4, SCRIPTed. *http://www.law.ed.ac.uk/ahrc/script-ed/vol3-4/prins.asp/*, 2006.

[128] M. O. Rabin. How to exchange secrets with oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.

[129] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *STOC '89: Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 73–85, New York, NY, USA, 1989. ACM.

[130] C.Y.C. Richard, R. Han, C.S. Li, and J. R. Smith. Secure transcoding of internet content. In *in Proc. Int. Workshop Intelligent Multimedia Computing and Networking (IMMCN*, pages 940–943, 2005.

[131] R. L. Rivest, A. A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 26(1):96–99, 1983.

[132] C. Roger. Introduction to Dataveilliance and Information Privacy, and Definition of Terms. *http://www.anu.edu.au/people/Roger.Clarke/DV/Intro.html/*, 1997.

[133] V. Roth. On the robustness of some cryptographic protocols for mobile agent protection. In *MA '01: Proceedings of the 5th International Conference on Mobile Agents*, pages 1–14, London, UK, 2002. Springer-Verlag.

[134] A-R. Sadeghi, T. Schneider, and I. Wehrenberg. Efficient privacy-preserving face recognition. Cryptology ePrint Archive, Report 2009/507, 2009. http://eprint.iacr.org/.

[135] K. Saleh. Understanding total system assurance: the case of mobile agent-based wireless sensor network systems. *IJCA Special Issue on Wireless Information Networks and Business Information System*, pages 26–32, 2011. Published by Foundation of Computer Science.

[136] T. Sander and Tschudin. C. F. On software protection via function hiding. *Information hiding, Lecture Notes in Computer Science 1525*, pages 111–23, 1998.

[137] T. Sander and Tschudin. C. F. Protecting mobile agents against malicious hosts. *Mobile agents and security, Lecture Notes in Computer Science*, pages 44–60, 1998.

[138] T. Sander and Tschudin. C. F. Towards mobile cryptography. *Proceedings 1998 IEEE symposium on security and privacy*, pages 215–224, 1998.

[139] T. Sander, Y. Young, and M. Yung. Non-interactive cryptocomputing for nc_1. *40th Annual Symposium on Foundations of Computer Science, IEEE*, pages 554–66, 1999.

[140] J. Scourias. Gsm history: Handover, 2006. http://www.privateline.com/mt_gsmhistory/2006/01/handover.html.

[141] GSM Association: Location Based Services. Permanent reference document. Technical report, SE.23, 2003.

[142] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[143] D. Singelée and B. Preneel. Secure e-commerce using mobile agents on untrusted hosts. Technical report, COSIC Internal Report, 2004.

[144] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *SP '00: Proceedings of the 2000 IEEE Symposium on Security and Privacy*, page 44, Washington, DC, USA, 2000. IEEE Computer Society.

[145] H. K. Tan and L. Moreau. Certificates for mobile code security. In *SAC '02: Proceedings of the 2002 ACM symposium on Applied computing*, pages 76–81, New York, NY, USA, 2002. ACM Press.

[146] J.R. Troncoso-Pastoriza, S Katzenbeisser, and M Celik. Privacy preserving error resilient dna searching through oblivious automata. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 519–528, New York, NY, USA, 2007. ACM.

[147] Y. Tseng, S. Kuo, H. Lee, and C. Huang. Location tracking in a wireless sensor network by mobile agents and its data fusion strategies. *The Computer Journal*, 47(4):448–460, 2004. Location tracking.

[148] W.G. Tzeng. Efficient 1-out-of-n oblivious transfer schemes with universally usable parameters. *IEEE Trans. Comput.*, 53(2):232–240, 2004.

[149] Infosecurity Magazine UK. Google docs leaks out private data, 2009.

[150] O. Ünay and T. I. Gündem. A survey on querying encrypted xml documents for databases as a service. *SIGMOD Rec.*, 37(1):12–20, 2008.

[151] G. Vigna. Protecting mobile agents through tracing. In *Proceedings of the Third International Workshop on Mobile Object Systems, in conjunction with the 11th European Conference on Object-Oriented Programming (ECOOP '97)*, Jyvaskyla, Finland, June 1997. Online Proceedings.

[152] G. Vigna. Cryptographic traces for mobile agents. In *Mobile Agents and Security*, volume 1419 of *LNCS State-of-the-Art Survey*, pages 137–153. Springer-Verlag, June 1998.

[153] J.P. Walters, Z. Liang, W. Shi, and V. Chaudhary. Wireless sensor network security: A survey, in book chapter of security. In *in Distributed, Grid, and Pervasive Computing, Yang Xiao (Eds*, pages 0–849. CRC Press, 2007.

[154] L.C. Washington and W. Trappe. *Introduction to Cryptography: With Coding Theory*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2002.

[155] B.R. Waters, D. Balfanz, G. Durfee, and D.K. Smetters. Building an encrypted and searchable audit log. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2004, San Diego, California, USA*, 2004.

[156] A.F. Westin. Privacy and freedom. *Atheneum Publishers*, 1967.

[157] Z. Yang, S. Zhong, and R. N. Wright. Privacy-preserving queries on encrypted data. In *Computer Security - ESORICS 2006, 11th European Symposium on Research in Computer Security, Hamburg, Germany, September 18-20, 2006, Proceedings*, pages 479–495, 2006.

[158] A.C. Yao. Protocols for secure computations. In M. S. Carberry, editor, *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science, Chicago IL*, pages 160–164. IEEE Computer Society Press, 1982.

[159] D-F. Ye, K-Y. Lam, and Z-D. Dai. Cryptanalysis of "2 r" schemes. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 315–325, London, UK, 1999. Springer-Verlag.

[160] Bennet S. Yee. A sanctuary for mobile agents. *Secure Internet programming: security issues for mobile and distributed objects*, pages 261–273, 1999.

[161] A. Young and M. Yung. Sliding encryption: A cryptographic tool for mobile agents. In *FSE '97: Proceedings of the 4th International Workshop on Fast Software Encryption*, pages 230–241, London, UK, 1997. Springer-Verlag.

Samenvatting

Dit proefschrift heeft als doelstelling het geven van oplossingen om de privacy te waarborgen in het geval dat niet vertrouwde proxies gebruikt worden voor communicatie en uitvoering van gevoelige informatie d.w.z. private computing. Een voorbeeld van private computing is een toegangscontrolesysteem (proxy) welke toegang geeft (of niet) aan gebruikers gebaseerd op vingerafdrukken. Uit privacy overwegingen wil de gebruiker zijn vingerafdrukken niet aan het systeem prijsgeven. The systeem gebruikt een mechanisme om vingerafdrukken te vergelijken met al eerder verzamelde vingerafdrukken om zo de identiteit van de gebruiker te kunnen verifiëren. De uitdaging is dat vingerafdrukken, zelfs van dezelfde gebruiker, nooit gelijk zijn zoals wachtwoorden dat wel zijn. Dit maakt het zo lastig om vingerafdrukken te vergelijken, zeker als het systeem ook niets over de vingerafdrukken mag leren behalve of ze gelijk zijn of niet.

Dit proefschrift beschrijft twee problemen binnen private computing. Het eerste probleem is om een niet-vertrouwde proxy informatie te laten verzamelen van verschillende bronnen. De uitdaging is om de geselecteerde informatie door de niet-vertrouwde proxy te laten verzamelen, waarbij de confidentialiteit van de alle informatie wordt gegarandeerd. Het tweede probleem is om een niet-vertrouwde proxy de verzamelde informatie met elkaar te laten vergelijken. De uitdaging is de niet-vertrouwde proxy een vergelijkingsfunctie uit te laten voeren zonder informatie prijst te geven over de inputs, maar wel de mogelijkheid te bieden de uitkomst te leren. Het probleem is vergelijkbaar met het Multi-Party Computation miljonairsprobleem, waarbij in het private computing probleem de niet-vertrouwde proxy de uitkomst leert van de berekening zonder daarbij eerst contact te moeten zoeken met de gebruikers.

Voor het selectie en collectie probleem worden er twee benaderingen uiteengezet. Ten eerste wordt het parallelle selectie en collectie probleem behandeld, waarbij een niet-vertrouwde proxy gelijktijdig informatie verzameld van verschillende bronnen zonder de privacy van de gebruiker aan te tasten. Het probleem wordt gepresenteerd in een Locatie-Based Service (LBS) scenario met het doel om private locatie informatie te beschermen. De oplossing gebruikt twee verschillende Oblivious Tranfer (OT) protocollen en homomorpfe encryptie. Ten tweede wordt het sequentiele selectie en collectie probleem behandeld, waarbij informatie wordt opgehaald bij verschillende bronnen op basis van een vast pad voordat het terugkeerd naar de proxy. De oplossing maakt

gebruik van threshold digitale handtekeningen en hash chaining. Vervolgens wordt er gebruik gemaakt van een hulp mechanisme welke ervoor zorgt dat het te volgen pad compleet wordt doorlopen ook als een van de bronnen niet beschikbaar is.

Twee benaderingen worden er uiteengezet voor het vergelijkingsprobleem. Ten eerste wordt er een enkelvoudige vergelijking uitgevoerd waarbij de niet-vertrouwde proxy de ongelijkheidsfunctie berekend. De oplossing is om een bestaand bitsgewijs vergelijkingsprotocol te reconstrueren op een zodanige manier dat de proxy één bit aan informatie lekt (het resultaat van de vergelijking) maar niets meer. De reconstructie van het nieuwe protocol is gebaseerd op meerdere homomorfe encrypties en decrypties gebruikmakend van ElGamal. Ten tweede wordt er een meervoudige vergelijking uitgevoerd welke kan worden toegepast op het vergelijken van vingerafdrukken probleem zoals hierboven beschreven. De uitdaging is om een niet-vertrouwde proxy meervoudige ongelijkheidsfuncties uit te laten voeren waarbij de proxy wederom alleen één bit aan informatie leert m.a.w. dat alle functies voldaan hebben aan de ongelijkheidscondities of dat sommige functies hebben gefaald, maar dat de proxy niet leert welke dat waren. De oplossing is gebaseerd op hetzelfde bitsgewijze vergelijkingsprotocol als bij de enkelvoudige vergelijking, met het verschil dat een ander homomorf encryptie schema wordt gebruikt en de hiding functie wordt uitgebreid.

Dit proefschrift demonstreert dat private computing protocollen ontwikkeld kunnen worden om de privacy van de gebruiker te beschermen terwijl de functionaliteit in het cryptografische domein wordt vergroot. Bovendien kunnen de gepresenteerde protocollen ook toegepast worden op andere applicaties waar niet-vertrouwde proxies onvermijdelijk zijn.

Summary

The objective of this thesis is to preserve privacy for the user while untrustworthy proxies are involved in the communication and computation i.e. private computing. A basic example of private computing is an access control system (proxy) which grants access (or not) to users based on fingerprints. For privacy reasons the user does not want to reveal his fingerprint to the system, since he does not trust the system in storing his fingerprint securely. The system uses a mechanism to compare a new fingerprint with previously collected fingerprints, in order to verify the identity of the user. The challenge is that fingerprints, even if they are from the same user, are never exactly equal like passwords are. This makes fingerprints hard to compare, especially when the system should not learn anything from these fingerprints other than if they are equal or not.

This thesis addresses two problems within private computing. First, the problem of letting an untrustworthy proxy collect private information from various sources is investigated. The challenge is to let the untrustworthy proxy perform the collection of the selected information, while guaranteeing confidentiality of the inputs and outputs. Second, the problem of letting an untrustworthy proxy compare the collected private information is addressed. The challenge is to let the untrustworthy proxy compute a comparison function without being able to learn the actual inputs, but being allowed to learn the outcome of the function. The problem is similar to the Millionaires' problem known from Multi-Party Computation, however in the private computing case the untrustworthy proxy learns the outcome of the computation without having to inform the users.

For the selection and collection problem two approaches are addressed. First, the parallel selection and collection approach is considered whereby an untrustworthy proxy collects information simultaneously from various sources without loosing the users privacy. The problem is presented within a location-based services (LBS) scenario with the goal to protect private location data. The solution is based on two distinct oblivious transfers and the usage of homomorphic encryption. Second, the sequential selection and collection approach is considered where information is collected from various sources based on a fixed itinerary before returning with the results to the proxy. The solution is provided using threshold signature schemes and hash chaining. Furthermore, a mechanism is constructed which ensures that the itinerary is completed

even if one of the sources is unavailable.

Two approaches are addressed for the comparison problem. First, a single comparison is undertaken, where the untrustworthy proxy computes one inequality function. The solution is to use a bit-wise comparison protocol and reconstruct it in such a way that the proxy leaks one bit of information (the result of the comparison) but nothing else. The reconstruction of the protocol is based on multiple homomorphic encryptions and decryptions using ElGamal. Finally, the multiple comparison problem is addressed which can be applied to the fingerprint matching problem as described above. The challenge is to let an untrustworthy proxy compare multiple inequality functions, learning only if all off the functions satisfied the comparison conditions or that some failed while letting the proxy remain oblivious to which conditions failed. The output of the function also only leaks one bit of information. The solution is based on the same bit-wise comparison protocol as the single comparison but it is reconstructed using a different homomorphic encryption scheme and extending the hiding function used for comparison.

This thesis demonstrats that private computing protocols can be designed to protect the privacy of the users while providing functionality in the cryptographic domain. Moreover, the presented protocols can also be applied within other applications where untrustworthy proxies are unavoidable.

Acknowledgements

I would like to thank my supervisor Jan van der Lubbe for his support he has given me during all these years, in good and bad times. Thank you for the nice discussions we had in your office or at the IKEA, just between work. I thank Inald Lagendijk for being my promotor, for helping me to improve my thesis and for providing an outstanding research environment within the Information and Communication Theory (ICT) group. I would like to thank all the participants of the PAW project.

I would like to thank everybody who has been part of the ICT group during the five years I was there. Dick, Emile, Jeroen[2], Richard[2], Ronald, Wouter, Jan[2], Pavel, Ginneke, Jenneke, Pien, Jesper, Pim, Eva, David, Jacco, Maarten, Marco, Theo, Umut thanks for all the fun we had during my research. A special thanks goes to Kathy who started as my MSc. thesis supervisor and later became a colleague in the group. We had many discussions on cryptography and privacy, but together with Kosmas we also discussed various interesting cultural aspects of life. I always enjoyed the conversations with Ivo during my first years as researcher. During my last years in the group, my roommates Zekeriya and Alper where of great support. Ben, Hans, Robbert, Anja, Annette, Saskia were a great help for all administrative and computer problems related tasks. Arjan, Daniela, Kai-Fan, Maarten, Martin, Maurice, Michiel, Paul, Jeroen and Vincent I would like to thank for all the discussions we had on your theses.

Ronald, Peter-Jan, Omar, Eugene, Kathy en Mark, we should not forget to have at least one dinner per year together. I want to thank Lejla and Bart for making it possible to collaborate with various international researches at the COSIC group, Katholieke Universiteit Leuven, Belgium. Claudia, George, Mina, Gregory, Dave, Péla, Brecht, Dries, Elke, thanks for showing me the way around the group. A special thanks goes to Markulf who helped me a lot with the research.

Bart and Rene we had a amazing trip together during my research period. Thanks for this unforgettable period in my life. Remember to Arrive Alive.

My roommates at the Balthasar van der Polweg where of great help. Mark and Nanning, thanks for the fun we had. Robert, I hope I didn't drive you too much crazy.

In time of stress, it is good to relaxt. I want to thank everybody at the AV Koplopers and especially everybody in group 5. Cor, keep up the good work and I hope you stay with us as a trainer for a long time.

Research was amazing but writing took longer than expected. Writing during work is hard, but I got a lot of support from my colleagues at Fox-IT. I want to thank you all. Special thanks goes to my roommates Ronald, Jeff and Nadeem. I was lucky to find another great company which supports me greatly in this adventure, Riscure. Thanks for all the support and looking forward to the future working together.

None of this would have been possible without the support from my family. I thank the Koehlers, Kitty, Cees, Raymond, for their interest and for all their compassion. Family Visser, Mieke, Paul, Isanne and Adriaan thank you for your support.

To my parents, it is you who have made this thesis possible, with all the love, opportunities and advices you have given me. To my brother, Philip, thanks for all the support.

Anouk, Willem-Jan, Daniel, Selma, Kemo, Dennis, Mandy, Michael, Eefke, Femke, Goos, Tamara, Ellen, Rosanne, Julien, thanks. I know I have forgotten to mention people who have contributed in some way to this thesis. Please allow me to make it up to you in the future.

And finally, I owe a lot of gratitude to Quirina. I spend many hours writing my thesis in the evenings, the weekends and during holiday times. Claiming one hour of work usually ended in writing the whole evening. I want to apologize for my bad time management and I know we will have a great future together.

Curriculum Vitae

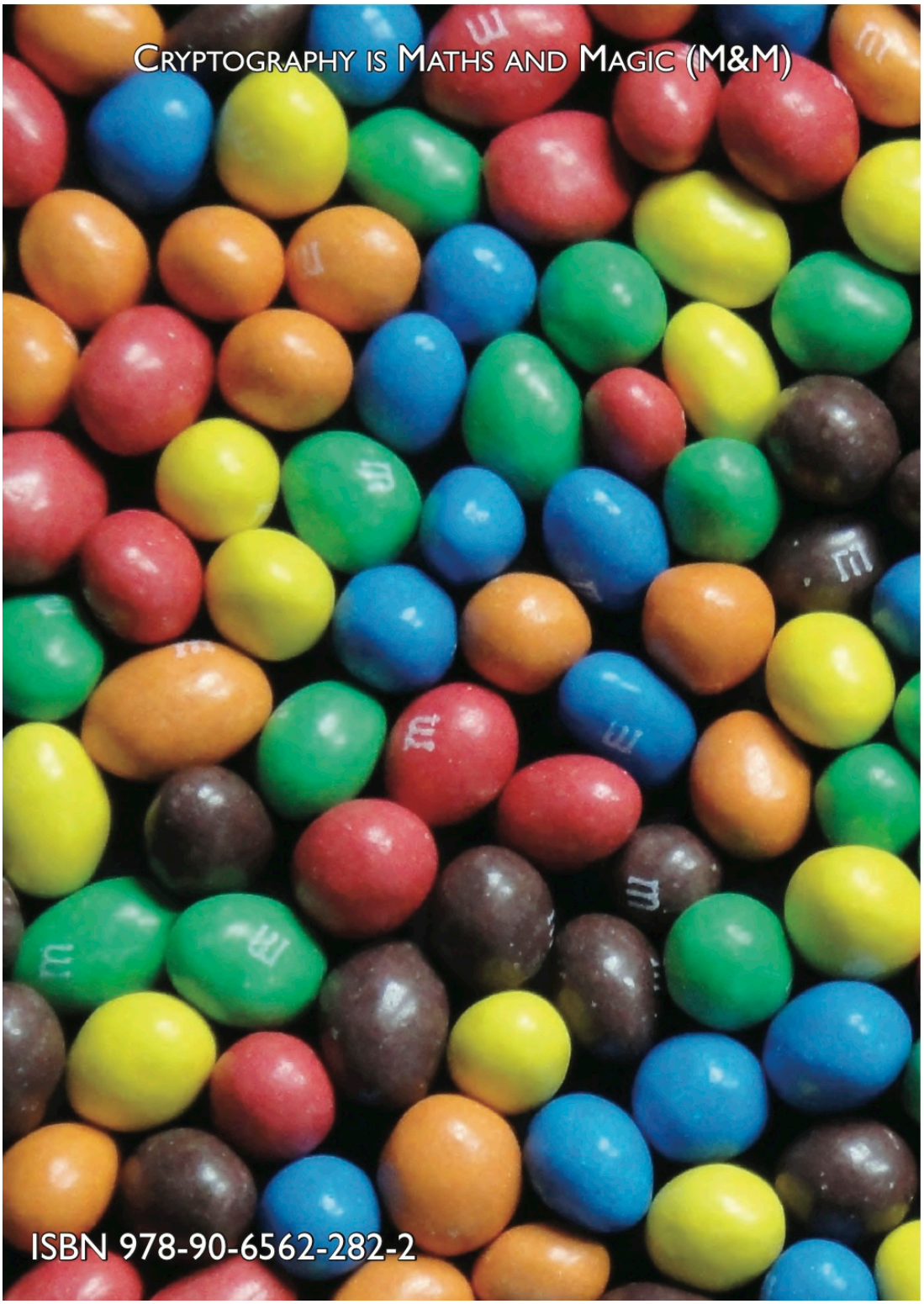
Bartlomiej (Bartek) Gedrojc was born on 28th of November 1975 in Poznan (Poland). After moving to the Netherlands in 1981, he received in 1992, his MAVO-diploma, in 1994, his HAVO-diploma and in 1996, his VWO-diploma, from the R.S.G. Prof. Zeeman in Zierikzee.

Thereafter, he studied Electrical Engineering at Delft University of Technology. During these studies he was an exchange student at the National University of Singapore (NUS). In 2004, he graduated at the Information and Communication Theory group with a MSc degree (In Dutch: Ir. diploma) and a master thesis entitled "Bitwise decision making within an Malicious host using an Oblivious Third Party".

In 2004 he started as a PhD researcher at the Information and Communication Theory group at Delft University of Technology in the research area of cryptography and privacy. He worked on the Privacy in an Ambient World (PAW) project funded by the IOP GenCom program. In 2006 he made an academic visit to the Katholieke Universiteit Leuven, Belgium at the COSIC group for a period of three months.

In 2008 he worked as a security consultant and product manager in the crypto department at Fox-IT in Delft. Here, among other things, he audited an Internet voting system (RIES) and conducted a Common Criteria evaluation (EAL7+).

Since 2011, he has been working as a sales executive at Riscure, Delft. Here, his work focusses on the commercial aspects of Side Channel Analysis.



ISBN 978-90-6562-282-2