Marcel Heupel

Porting and evaluating the performance of IDEMIX and TOR anonymity on modern smartphones

Portierung und Bewertung der Laufzeit von IDEMIX und TOR-Anonymität auf modernen Smart-phones

Prüfer: Prof. Dr. Dogan Kesdogan
Betreuer: Dipl.-Inform. Mohamed Bourimi

November 30, 2010

Der Speicherung meiner Diplomarbeit zum Zweck der Plagiatsprüfung stimme ich zu. Ich versichere, dass die elektronische Version mit der gedruckten inhaltlich übereinstimmt.

Siegen, November 30, 2010

Marcel Heupel
Zusammenfassung


Von besonderem Interesse ist in diesem Szenario die Möglichkeit dass Nutzer sich gegenseitig diese Beweise erbringen können, da verwandte Arbeiten bisher immer einen Server als verifizierende Instanz (z.B. im Rahmen von Zugangs-Kontrollmechanismen) verwendeten.

## Contents

1 Introduction  
1.1 Motivation .............................................................. 1  
1.2 Objectives of this Thesis ............................................. 3  
1.3 Outline ........................................................................ 4  

2 Basic Concepts and Technologies  
2.1 Identity Management Systems ........................................ 5  
2.2 Anonymous Credential Systems ..................................... 6  
  2.2.1 Credentials .............................................................. 6  
  2.2.2 Digital Credentials .................................................. 6  
  2.2.3 Anonymous Credentials .......................................... 7  
2.3 Anonymity .................................................................... 7  
  2.3.1 Anonymity Networks ................................................ 8  
  2.3.2 Tor ...................................................................... 8  
  2.3.3 JonDo ................................................................... 9  

3 Idemix - IBM’s Identity Mixer  
3.1 Idemix in a Nutshell ....................................................... 10  
3.2 Main Components of Idemix ........................................... 12  
  3.2.1 System Setup .......................................................... 12  
  3.2.2 Attributes .............................................................. 12  
  3.2.3 Credentials .............................................................. 13  
3.3 Main Idemix Protocols .................................................... 13  
  3.3.1 The Issue Protocol .................................................. 13  
  3.3.2 The Prove Protocol .................................................. 14  

4 Requirement Analysis  
4.1 Di Ме Background Information ....................................... 18  
4.2 The Di Ме Scenario ....................................................... 19
Contents

4.3 Identifying Concrete Requirements .................................................. 19
  4.3.1 Concrete Requirement Related to O1 and O2: Porting Idemix and
         Developing a Prototype Reference Implementation ......................... 19
  4.3.2 Concrete Requirement Related to O3 and O4: P2P Business Scen-
         nario Implementation Supporting TOR-Anonymity .......................... 20
  4.3.3 General Development Related Requirements ................................. 21
  4.3.4 Additional Requirements for the User Interface ............................. 22

4.4 Summary of Requirements .......................................................... 22

5 Related Work
  5.1 Idemix Related Work (R1, R2.x, R4-R6) .......................................... 24
  5.2 Related Work to Usability of Access Control in Collaborative
         Mobile Settings (R3.x, R7) ......................................................... 25
  5.3 Summary ....................................................................................... 26

6 Approach
  6.1 R1: Porting of the Idemix Library .................................................. 27
  6.2 R2: Development of the Idemix Module .......................................... 28
     6.2.1 R2.1: Obtain Credential .......................................................... 29
     6.2.2 R2.2 and R2.3: Show Proof ...................................................... 29
  6.3 R3: Prototypic Implementation of the Business Conference
         Scenario .......................................................................................... 29
     6.3.1 R3.1: Client-Server Communication ........................................... 30
     6.3.2 R3.2: Client-Client Communication ............................................ 31
  6.4 R4: Development of the Certificate Authority Server ....................... 32
  6.5 R5: Anonymous Network Communication ....................................... 33
  6.6 R7: Providing a User Interface Prototype ........................................ 34

7 Implementation
  7.1 Adoptions of the Idemix library ..................................................... 36
  7.2 Implementation of the Core Module ................................................. 37
     7.2.1 Implementation of the Credential Issuance ................................... 37
     7.2.2 Implementation of the Proof Protocol ......................................... 39
  7.3 Implementation of the Network Interface ........................................ 40
     7.3.1 The XML-RPC Service .............................................................. 41
     7.3.2 The XMPP Service .................................................................. 41
  7.4 Implementation of the Certificate Authority ...................................... 41
  7.5 Enabling Anonymous Network Communication with Tor ................... 41
  7.6 Additional Implementation Details .................................................. 42
     7.6.1 Eclipse and the Android SDK .................................................... 43
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.6.2</td>
<td>Android</td>
<td>43</td>
</tr>
<tr>
<td>7.6.3</td>
<td>The Android Development Tools</td>
<td>44</td>
</tr>
<tr>
<td>7.6.4</td>
<td>Communication protocols</td>
<td>44</td>
</tr>
<tr>
<td>7.6.4.1</td>
<td>XML-RPC</td>
<td>44</td>
</tr>
<tr>
<td>7.6.4.2</td>
<td>XMPP</td>
<td>45</td>
</tr>
<tr>
<td>7.6.4.3</td>
<td>The eJabberd Server</td>
<td>45</td>
</tr>
<tr>
<td>7.6.5</td>
<td>Used Libraries</td>
<td>46</td>
</tr>
<tr>
<td>7.6.5.1</td>
<td>Idemix</td>
<td>46</td>
</tr>
<tr>
<td>7.6.5.2</td>
<td>Apache XML-RPC</td>
<td>46</td>
</tr>
<tr>
<td>7.6.5.3</td>
<td>Smack</td>
<td>46</td>
</tr>
<tr>
<td>7.6.5.4</td>
<td>The Orbot Tor-Client for Android</td>
<td>46</td>
</tr>
<tr>
<td>7.6.5.5</td>
<td>XML-RPC for Android</td>
<td>47</td>
</tr>
<tr>
<td>7.6.5.6</td>
<td>aSmack</td>
<td>47</td>
</tr>
<tr>
<td>8</td>
<td>Performance Evaluation and Results</td>
<td></td>
</tr>
<tr>
<td>8.1</td>
<td>General Aspects of Performance Evaluation</td>
<td>48</td>
</tr>
<tr>
<td>8.1.1</td>
<td>General Definition of Performance Evaluation</td>
<td>48</td>
</tr>
<tr>
<td>8.1.2</td>
<td>General Objectives of Performance Evaluation</td>
<td>49</td>
</tr>
<tr>
<td>8.2</td>
<td>Evaluation Setting</td>
<td>49</td>
</tr>
<tr>
<td>8.2.1</td>
<td>Variation of Key Length</td>
<td>50</td>
</tr>
<tr>
<td>8.2.2</td>
<td>Variation of Supported Attributes</td>
<td>50</td>
</tr>
<tr>
<td>8.2.3</td>
<td>Variation of Attributes Issued in a Credential</td>
<td>50</td>
</tr>
<tr>
<td>8.2.4</td>
<td>Variation of Predicates</td>
<td>50</td>
</tr>
<tr>
<td>8.2.5</td>
<td>Variation of Devices</td>
<td>51</td>
</tr>
<tr>
<td>8.2.6</td>
<td>Variation of Communication Channels</td>
<td>51</td>
</tr>
<tr>
<td>8.3</td>
<td>Preparations</td>
<td>51</td>
</tr>
<tr>
<td>8.4</td>
<td>Carrying Out the Measurement</td>
<td>52</td>
</tr>
<tr>
<td>8.5</td>
<td>Discussion of Results</td>
<td>52</td>
</tr>
<tr>
<td>8.6</td>
<td>Summary</td>
<td>57</td>
</tr>
</tbody>
</table>

9 Summary and Conclusion

10 Acknowledgements
List of Figures

2.3.1 A message with 3 encryption layers ........................................ 9

3.1.1 Use-case diagram ........................................................................ 11
3.2.2 Examples of a SystemParameters file (A) and a GroupParam-
eters file (B) .................................................................................. 12
3.3.3 Example of a credential structure file ........................................ 14
3.3.4 Sequence diagram of the issuance protocol ................................ 15
3.3.5 Sequence diagram of the proof protocol ........................................ 16
3.3.6 Example of a proof specification file ........................................... 17

6.2.1 Simplified class diagram of the client application ....................... 28
6.3.2 Diagram illustrating the implemented scenario ............................. 30
6.3.3 Sequence diagram of the broadcast use-case ............................... 32
6.3.4 Simplified class diagram of the server application ....................... 33
6.6.5 Selected GUI masks of the prototype ........................................... 34
6.6.6 Selected GUI masks of the prototype to build a proof ................ 35
6.6.7 Additional GUI masks of the prototype ......................................... 35

7.6.1 The Android system architecture [Google Inc., 2010] ............ 44

8.5.1 Average response times of the issuance and proof protocol ....... 52
8.5.2 Response times of the issuance .................................................. 53
8.5.3 Response times of the proof ....................................................... 53
8.5.4 Relative growth of the proof computation time ........................... 54
8.5.5 Complete computation time of the One-Of-Each-Proof ............... 54
8.5.6 Percentage computation times of the sub proofs ......................... 55
8.5.7 Percentage computation times of the build proof method .......... 55
8.5.8 Pie diagram of the computation time of the proof ....................... 56
List of Figures

8.5.9 Results for the Samsung smartphone ........................................... 56
8.5.10 Traceview showing visualising the method calls during the proof protocol ................................................................. 57
Code listings

1. Example of a Credential Issuance Part 1 .................................. 38
2. Example of a Credential Issuance Part 2 .................................. 39
3. Example of a Proof Generation .................................................. 40
4. Example of a XMPP message ..................................................... 42
5. The Idemix Interface ............................................................... 43
6. XML-RPC method call ............................................................. 45
Abbreviations

API ............... Application Programming Interface
CSCL ............. Computer Supported Cooperative Learning
CSCW ............. Computer Supported Cooperative Work
GUI ............... Graphical User Interface
HCI ............... Human-Computer Interaction
HTML ............. Hyper Text Markup Language
HTTP ............. Hyper Text Transfer Protocol
Idemix ............. Identity Mixer
IdM ............... Identity Management System
IP ................. Internet Protocol
MVC .............. Model-View-Controller
OS ................ Operating System
P2P ............... Peer-to-Peer
PET ............... Privacy enhancing technologies
PICOS ............ Privacy and Identity Management for Community Services
PRIME ............ Privacy and Identity Management for Europe
RPC ............... Remote Procedure Call
Tor ................ The Onion Router
UI ................ User Interface
URL ............... Unified Resource Locator
WWW ................ World Wide Web
XML .............. eXtensible Markup Language
XMPP ............. eXtensible Messaging and Presence Protocol
1 Introduction

1.1 Motivation

The use and disclosure of personal information for private and business life is a major trend in our information society. Also, most people have a social nature and interact with others in order to feel good [Slagter, 2004]. Security is more and more of importance in the everyday life when people are taking part in social or business networks because there, it is necessary to disclose some personal information. When people are participating in the modern online life, they are leaving traces in every place they are visiting in the virtual world. Even though they are aware of some of these traces they are leaving behind (e.g. registering and creating a profile on a social website like Facebook), many other traces remain unrecognized (e.g. gathered information about their used browser or operation system by the providers in an implicit way). Another dimension consists of losing control over the disclosed information, once it is published.

Due to the rapid evolution of computing systems and sinking costs of connecting them to the Internet, mobile devices are increasingly used in different sectors of our leisure and professional life. Ericsson estimates mobile subscriptions have hit 5 billion and the Wi-Fi Alliance and Wakefield Research estimate that only in this year 216 million devices will be sold from which 82 million provide Wi-Fi functionality [Wi-Fi Alliance, 2010]. Thereby, these devices offer new possibilities of situated interaction and are therefore also used in social settings. Such mobility (mostly with continuous connectivity) brings new challenges for security and privacy in ubiquitous and pervasive computing. The recent massive propagation of mobile devices and mobile applications gains strength from leveraging efficient, secure privacy-respecting interaction and communication patterns between individuals and communities as well as seamlessly supported interaction with these devices in term of enjoyable user experience. With respect to the different capabilities and restrictions of modern mobile devices (e.g. smartphones and tablet PCs etc.), addressing security and usability aspects becomes crucial. Experts from various research communities believe that there are inherent trade-offs between security and usability to be considered [Cranor and Garfinkel, 2005] [Shneiderman et al., 2009] [Boyle et al., 2008].

Indeed, a good example for trade-offs between privacy (security) and usability or maybe for good design is Apple’s iPhone. Even tough it is less secure than RIM’s BlackBerry or devices using Microsoft’s Windows Mobile [eWeek.com,
customers were still switching to it and Yahoo announced to focus in its mobile program on the support of iPhone and abandoned its BlackBerry smartphone application [TechCrunch, 2009]. However, preserving the users’ security and privacy in social settings is the most often-cited point of critique of mobile and ubiquitous computing [Hong and Landay, 2004]. Even though anonymization and data minimization mechanisms (i.e., through removing sensitive data like names, addresses, etc) are provided, the users can be re-identified across distinct high popular social networks like Facebook, Flickr, MySpace or Twitter with an error rate of just 12% like recently shown in [Narayanan and Shmatikov, 2009]. Linkability and users’ profiles based on social interaction could be the starting point for potential man-in-the-middle attacks. Enhancing security or its main part privacy, when using the Internet or its famous segment the World Wide Web (WWW; Web) for social or collaborative interactions, lies in following a combination of different approaches and good practices. This is for instance the usage of privacy-enhancing technologies (PET). This helps to achieve security goals like anonymity, data minimization and many other established techniques for providing confidentiality as well as integrity of sensitive data (e.g., identity of users, communication data etc.). However, usability is a prerequisite for security and privacy. Therefore, it is part of a major effort to balance and improve security and privacy design of mobile applications by considering usability aspects especially due to the mobile devices limitations and capabilities (e.g., screen size, limited memory, computation capabilities and ease of localization).

One of the most overlooked and critical topics of computer security is a vast understanding of the interplay between usability and security [Cranor and Garfinkel, 2005]. In social and collaborative interaction settings advantages such as enhancing social contacts, personalizing services and products compromise with notable security and privacy risks arising from the user’s loss of control over their personal data and digital footprints. From the usability perspective, large amounts of scattered personal data lead to information overload, disorientation, loss of efficiency. This often results in not using security options offered by the application. Regarding security and privacy with respect to the anonymity of identities, it is common practice to use different pseudonyms in relation to a subset of attributes (e.g. nickname: surfer123, likes spicy food, loud music and is 21 years old) for different contexts. Most people are of the opinion that, using this pseudonym is "being anonymous" and they are not aware, that everything they are doing, could be easily monitored and linked to build comprehensive profiles. It is not necessary to know the real name of a person, if you know everything else, in some cases even more than the person herself. Because of the vast amount of different contexts (communities, business, ..) one is confronted to, it is reasonable to use some sort of identity management solution, to handle all the multiple identities one possesses.

One of the means for enhancing privacy in communication to individuals and services is to allow for the usage of partial identities or digital faces, i.e. user data selected to be disclosed for a particular purpose and context. Technical
systems and applications supporting privacy-enhancing collaborative users’ activities sometimes have to allow for user-controlled identity management (IdM) as required in Di.Me. Furthermore, such IdM system has to be deployable on mobile platforms by providing good performance in terms of response time as a usability quality of service factor [Shneiderman et al., 2009]. Pure response times lead to end-user frustration and affects the usage of the applications especially when no adequate help or feedback are provided. These general requirements are based on the needs of the EU FP7 project Digitalize.Me (Di.Me) and defines the objectives of this thesis as described in the next Section.

1.2 Objectives of this Thesis

The main objective of this thesis is to provide a prototypic implementation of an IdM module for smart-phones by using the IBM Identity Mixer called Idemix. The first analysis of Idemix has shown, that it is satisfying the needs of Di.Me for the support of anonymous credentials. Since Idemix is already available in form of a Java based open source library, it is possible to use this library on modern mobile devices supporting Java such as done in [Armac et al., 2009]. However, new trends shows that Android as well as iOS based platforms gain a wide acceptance. Because of this it is one of the primary goals of this work to port Idemix on one of these platforms (O1), in order to build a flexible module (O2) which can be used to fulfill the needs from the representative Di.Me Scenarios. This module will act as an intermediate layer (API) supporting Idemix integration of all intended scenarios.

Because response times are an important factor of overall usability and availability requirement from the security perspective\(^1\), the the prototypic implementation has to be evaluated with respect to the performance of the resulting module for the different needs of the respective Di.Me scenarios (O3). This is especially of importance because the acceptance of Java in the world of mobile development is also suffering under the success of the Android and iOS platforms. These platforms provide strong capabilities related to usability and interaction design such as powerfullness of user interface elements or support of seamless interactivity etc. One of the requirements of building an anonymous credential system for Di.Me is also to provide anonymity at the network level. Because Idemix is only providing the cryptographic mechanisms which allow for anonymity at the application level, the prototypic implementation, as well as the performance evaluation, need to reflect and consider the possibility of enabling anonymity at the network level (O4). For this thesis, the network anonymity solution TOR was chosen.

---

\(^1\)The CIA security triangle classifies security from the perspective of Confidentiality, Integrity, and Availability. Latter is concerned with the provision of correct data in acceptable ranges addressing so the performance aspect.
1.3 Outline

This thesis is structured as follows: At first, the chapter 2 gives an overview about the technologies being used for developing the prototype and for performing the measurements. Many of the concepts are also of importance in the requirements analysis. We will also give an short introduction to the basic concepts of identity management and anonymity networks, followed by describing the concrete realization we used in our implementation.

Due to the fact that Idemix is the core theme in this work, chapter 2 is dedicated to deal with the Identity Mixer developed by IBM. The two main protocols and functional principles are explained in detail, as well as the most important components, like for instance credentials and attributes.

In chapter 4 is described how the requirements for implementing the prototype are gathered from the scenario and as well derived from previously, related researches. Therefore, a short introduction to the IBM Identity Mixer (Idemix) is given besides some background information about the Di.Me project, on which work package descriptions we base our scenario. After that short introductory part, the requirements are posed and discussed in detail.

Chapter 5 will outline the related work in the most overlapping research areas, like anonymous credentials and identity management. Also the work from the EU project PRIME and PrimeLife will be elucidated, because they made a big contribution to this work by the development and use of the Identity Mixer library (idemix).

In chapter 6 we describe the concrete realization of the requirements, derived from the scenario. Additionally the evaluation setting is explained in detail.

Chapter 7 focuses on the implementation details of the approach. The two different supported communication protocols, the client-server architecture and the P2P architecture and also details about the implementation will be expounded. We will examine the concrete implementation of the idemix protocols in the Android client as well as the Java server and depict exemplary code examples.

Finally, in chapter 8 the proceeding of the measurement as well as the results and evaluation of performance will be discussed, before we conclude in chapter 9 with the summary and an overview about future research fields.
2 Basic Concepts and Technologies

In this chapter we are providing information about basic concepts and technologies we used in our approach. We first introduce the Android platform and then provide some basic understanding about key concepts like identity management and anonymity networks.

2.1 Identity Management Systems

The term Identity Management can be understood in a couple of different ways. It is often seen as a combination of authentication, authorization and accounting of these processes, which is a general concept in almost every application, supporting multiple users. All this mechanisms, concerning the access control to a computer system or a database, applying different role concepts and also the administrative processes in this context, come under the heading of server-centric identity management. Here the main goal is the management of the rights and restrictions of the users, as well as the logging of their activities [Hansen et al., 2008].

In contrast to sever-centric IdM, there is also user-centric IdM, which is dealing with the management of one or more different identities of a user. In user-centric IdM the user should have full control over her data and can determine, which private data is disclosed in which context. If there is no possibility to chose the revealed data, like in many government related processes, the user should at least be able to transparently see which data is used by whom and in what context. Therefore a user-centric IdM should also be able to log the transactions, so that the user can review them, at a later time. In contrast to the server-centric solution, the logging is happening on the users own device, which is beneficial for his privacy [Hansen et al., 2008], [Clauss et al., 2006].

One of the main objectives of an user-centric IdM system is to provide functionalities to manage all the different identities, a user possesses, in different contexts. This so-called partial identities are in most cases a subsets of the real identity of a user, bound to a certain pseudonym. In summary, a good IdM system has to allow for mechanisms to provide pseudonymity as well as anonymity to achieve the following main goals according to [Clauss et al., 2006]:
1. Controlled user pseudonymity, ideally it is not possible to link real identity and pseudonym or different pseudonyms.

2. Controlled accountability: a pseudonym can be used for secure authentication and in terms of misbehavior the user could be called into account.

2.2 Anonymous Credential Systems

In this section we will provide a general description about anonymous credentials and the differences to regular credentials. The explanations should be “common sense understandable”, and so only main features and functionists are described, without the mathematical and cryptographic details.

2.2.1 Credentials

A credential is a certificate, a person can use to convince another person about the facts (e.g. a set attributes from the user) attested by it. This certificate is issued by a trusted authority (e.g. the government). The American Heritage Dictionary of the English Language defines a certificate as “a document testifying to the truth of something” [Ahde, 2006]. To enhance the integrity of the certified information, there are many approaches to prevent forgery of such documents. This can be for example watermarks, special paper etc. An example for an traditional, paper based credential (or certificate) is the german drivers license card. Anyone who is in possession of such a card, can show this to another person to proof that one is allowed to drive a car. And also the name and birthdate are certified by this card.

2.2.2 Digital Credentials

In the past few years, the use of digital certificates has been steadily increasing, and still is. Some countries have even launched approaches of bringing a digital version of the personal identity card to the marked (in Germany for example the new ePersonalausweis allows for digital authentication in E-business or E-government processes [BMI, 2010]. We focus in this thesis on digital credentials and illustrate the differences to “offline” credentials. In general, digital credentials have the same features like the certificates described above, because they are by definition, a digital equivalent to the paper based credentials. But besides this, they have some beneficial qualities, that should be mentioned in the following.

How a user can obtain a credential is already described in detail in chapter 3 in the context of Idemix. In principal the user has to convince an issuer, that he is in possession of a certain set of attributes, and the issuer certifies, that this is true. This certified list of attributes is called credential, and can be used by the user in order to convince other entities that the containing values are true. In practice, in order to receive a first root-credential, which can be used in order
authorize to certificate authorities to get other credentials, one has to show a paper-based credential to a certificate authority [Brands, 2001]. In conventional credential systems, the user would now just send the whole credential to another party (in the drivers license example, this would e.g. mean to give the real card or a photocopy of it to an car rental company). This would be ok for the verifier, but there are two important disadvantages for the user. On the one hand, the user is revealing all the attributes of the credential at once, even if only one value is needed (e.g. birthdate etc. is also revealed, but only the vehicle category is needed). On the other hand, the verifier could just take that credential to impersonate herself as the user [Brands, 2001].

2.2.3 Anonymous Credentials

When using anonymous credentials, the two issues mentioned in the previous section can be avoided. In an anonymous credential system (e.g. Idemix), a credentials itself is never transmitted. The user is only sending a cryptographic proof, that she is in possession of the credential and that certain selected attributes fulfill the claimed peculiarities. Thereby only the necessary attribute is used, while the other stay hidden. Additionally, there is also the possibility, only to proof that a complex predicate over a selected set of attributes holds, without revealing the concrete value of the attribute [Neven, 2008].

2.3 Anonymity

Concerning anonymity the society has changed a lot in the last 15-20 years. In the past, people were moving much more anonymously through life than it is the case today [Köpsell and Pfitzmann, 2003]. Back then everyone was using cash money, public telephone booths and public places were not under video surveillance, which is all relatively anonymous. In contrast to this, when people are more and more using EC or credit cards for shopping (Sweden is even considering to completely abandon cash money [Giedroyc, 2010], mobile phones can easily localized and communication over the internet can be reconstructed even after years, it becomes more and more difficult, to life without being observed all the time. Also the opinion of the society has changed in the last 15 years, since back then anonymity was seen much more negatively than today [Köpsell and Pfitzmann, 2003]. In the following we will provide a short introduction in the term anonymity, since it is an important prerequisite to successfully establish an anonymous credential system. Therefore we are also introducing some examples of methods to disguise communication in the internet.

In the literature (e.g. [Pfitzmann and Hansen, 2010] or [Köhntopp and Pfitzmann, 2001]) the term Anonymity is defined as follows:

Anonymity is describing the condition, when it is not possible to identify a single person, because she is hidden in a large group of other people. Also it is impossible to determine which person is communicating with whom. Thereby
is is obvious, that the possibility to identify a single person, depends on the number of members in this group. Also should all members behave similar to not be apparent. Also it is important to mention, that in reality anonymity is never absolute. It is only an approximation of being identified in the group of similar behaving people. Because not everyone is exactly acting alike, the degree of anonymity one enjoys may vary [Köpsell and Pfitzmann, 2003]. Additionally, when anonymity is established, two other strong security goals could be achieved, to enhance the privacy of the members in this group. The first is *unlinkability* of transactions or communication. This means, that even if an (possible) attacker is observing the whole communication, he/she is not possible to link different transactions (or messages) to a single person. The second, even stronger security goal is to provide *unobservability* of transactions. By doing this, an attacker is not even possible to determine if any transaction happened or not.

### 2.3.1 Anonymity Networks

When people are using the internet, independent of what they are actually doing, even when using PETs like encryption, the IP address of the recipient is always visible in the header of the data package. By monitoring and tracking the IP address, a possible attacker could get clues about the real identity (or at least his home town) and also who is communicating with whom. There are several approaches to disguise the IP address in order to achieve anonymity the network traffic. In this subsection we will provide a short introduction in the general functionality of anonymity networks by describing the functionality of the two anonymity networks Tor and JonDo.

### 2.3.2 Tor

The principle of the Tor network, can be described as *Onion-Routing*, where data packages are routed through constantly changing nodes. If a user likes to send a message to another user, this is encrypted several times, with the public key of a Tor-node. These nodes are called Onion router, due to the fact, that multiple encryption layers are wrapped around the original message. Figure 2.3.1 illustrates an encoded message. In the middle is the original message, surrounded by 3 encryption layers. This is how the message leaves the client. Before sending the client randomly choses a circuit of three nodes in the network of Tor nodes, and encrypts the message with the public key of the nodes. The message header is containing the IP address of the first Tor node. After receiving the message, the node decrypts it and receives another encrypted message. Now the IP address of the second layer is visible, and the message can be sent to the second node and so on. Each node is *peeling* a layer from the message until the last node decrypts it and sends the original message it to the recipient. In Tor, in general a path of three different node- servers is selected randomly from the client application and switching in different intervals.
The Tor network consists of a very large number of nodes, which are each having a relatively small bandwidth. In principle every user with an DSL internet connection can configure his own Tor node and add it to the public network.

### 2.3.3 JonDo

For the sake of completeness we will here describe the second important anonymity network, the JonDo (also known as AN.ON) network. In contrast to the Tor network, where onion routers are used to disguise the identity, the JonDo network is using a different approach. In their solution, the nodes in the network are *mixes*. This is a really different concept. This approach goes back to David Chaum (see [Chaum, 1981]) In a Chaum-Mix all participants are sending messages of the same length (even dummy messages) to a server, called *mix*. This mix then changes the message (with encryption mechanisms) and sends it to the next destination. This makes it impossible to observe who is communicating with whom. The Tor network uses many Tor nodes with relatively small bandwidth. Contrary to that, the JonDo network uses a relatively small number of nodes, which therefore need a high bandwidth, to satisfy the users. JonDo servers are often maintained by companies. In order to use the bandwidth, users need to buy data volume [JonDos GmbH, 2010].

![Figure 2.3.1: A message with 3 encryption layers](image)
3 Idemix - IBM’s Identity Mixer

In this chapter we introduce the IBM’s Identity Mixer, Idemix, because it is an integral part of the solution proposed for the problem to be solved in the context of Di.Me. Therefore, we provide this introduction, before we address concrete requirements in chapter 4. The main components and protocols are explained.

3.1 Idemix in a Nutshell

Idemix is an anonymous credential system, developed by the research group led by Jan Camenisch at IBM Research Zurich [IBM Research, 2010]. It enables to perform anonymous authentication between users and/or service providers and as well supports accountability of transactions [Camenisch and Van Herreweghen, 2002]. An Idemix credential is obtained from an issuing authority, attesting to the users attributes such as birth date or access rights and allows for various protocols and mechanisms cited in standard literature (i.e. property proofs, usage limitation, revocation of credentials, verifiable encryption). The main protocols performed, are the credential issuance and the show proof protocol which are using the Camenisch-Lysyanskaya signature scheme [Camenisch and Lysyanskaya, 2001], [Camenisch and Lysyanskaya, 2003]. Prerequisites for successfully performing those protocols are that all participants share the same group and system parameters, which define for example the size of the RSA modulus, or the domain of a hash function [Report, 2010]. During the issuance protocol the user and a certificate authority (CA) interactively create a credential. This credential is signed by the issuing CA with its private key, so it can easily be verified using the issuers public key. It also contains the users pseudonym, to bind the users master secret to the credential [Report, 2010]. Figure 3.1.1 illustrates a use-case diagram of the main protocols of Idemix.

In contrast to PETs sending (pseudonym) certificates to a given verifier, Idemix based solutions only send proofs (such as "I am older than 18" or "working in the automotive industry"). It allows for un-linkable, selective disclosure of such attested credential attributes while not revealing others. When the user shows this credential to another entity (another user or a service provider), the credential itself is never revealed. Indeed, several zero-knowledge proofs are performed,
to convince the other entity of the possession of a signed statement about an attribute. A zero-knowledge proof means that the disclosed these attributes were certified by the issuer. Due to the nature of zero-knowledge proofs, the user can show the same credential over and over, without risking the linking of information by the other entity [Report, 2010]. Since Idemix enables for involving the user in the personal data disclosure process, the user can decide which attributes of his digital face can be disclosed or not [Bichsel and Camenisch, 2010]. An example for the usage of Idemix is for instance, one could prove to Amazon being over 18 when buying games having such restriction without disclosing the accurate birthday. Furthermore, one can pay and receive the respective game and only the delivery service (e.g. DHL or UPS) will be able to see his/her address etc. All these strengths make it perfect for realizing an user-controlled IdM supporting digital faces in Di.Me for both; collaboration of users with each other as well as collaboration of users with a given service.

Figure 3.1.1: Use-case diagram
3.2 Main Components of Idemix

The anonymous credential system Idemix consists of several components. The most important ones should be explained in detail, to provide a vast understanding about how Idemix works.

3.2.1 System Setup

In order to be able to perform the cryptographic protocols, all participants have to use the same system parameters, that are defined in two connected files, the SystemParameters file and the GroupParameters file. They are defining for instance the key length of the RSA modulo, the size of an hashed String or prime probabilities. An example of the two files is depicted in Figure 3.2.2.

![Figure 3.2.2: Examples of a SystemParameters file (A) and a GroupParameters file (B)](image)

Because all participants need the same parameters, in order to perform the cryptographic protocols, it is good practice to store this file could be an URL, where all participants can obtain it. It is also necessary, that all participants (issuer, users, trusted third parties) generate and publish their own public keys.

3.2.2 Attributes

Attributes in Idemix could be name, surname, affiliation etc. and are formally understood as a tuple consisting of name, value and type (a=n,v,t). Each tuple, represented by the name, has to be unique in its scope (e.g. the XML-namespace of the CredentialStructure file). At the moment allowed attribute types are: Integer, String, Date and Enumerations. Dates are represented as seconds relative to the origin date, 1.1.1900. String attributes need to be encoded, in order to be used in a group G of the order n. This is done by computing a Hash of the length n and only the hashed value is used in the credential. Integer and Date values need not to be encoded (unless they are longer than n).
3.2.3 Credentials

A credential in Idemix is defined as a set of attributes, together with some cryptographic information. It is obtained from a certificate authority by performing the issuance protocol. Basically it can be seen as some kind of certificate about a set of attributes someone has possession of. The credential allows to compute statements about the value of the containing attributes in order to prove them to another entity. The important difference to regular certificates is the fact, that a credential itself is never shown to another instance.

3.3 Main Idemix Protocols

In this Section, we address the two main protocols of Idemix, namely, the "Issue Protocol" and "Prove Protocol". There are other protocols like the "Credential Updates Protocol" which is similar to the Issue Protocol and that allows for just updating parts of existing credentials. For accuracy, we only address the two main protocols in the following.

3.3.1 The Issue Protocol

The credential issuance protocol is interactively performed between the recipient (e.g. the user), who will obtain the credential and the issuer, the Certificate Authority (CA) (e.g. a trusted company or the government). In order to obtain a credential, the recipient needs a credential structure definition, that defines the attribute structure of the credential to be issued. In general, this definition is formulated in XML. Figure 3.3.3 illustrates an example of such a file. It is necessary that both parties have the same definition, in order to perform the protocol.

With this specification, it is also possible that both parties (issuer and recipient) compute a string, called context, that is a list of all public parameters, used to bind the zero-knowledge-proof to the current context. This context helps the CA to difference between different users that are performing the issuance protocol at the same time and could later be used for credential updates.

The protocol itself consists of three steps, taking place between 4 rounds, in which the recipient and the issuer create the cryptographic part of the credential. In the sequence diagram 3.3.4 these 4 rounds are depicted in detail. At first, the recipient initiates the issuance protocol, by sending a request to the issuer. After this, the following protocol is performed:

Round 0 After the issuer has finished initializing, (by loading the attribute structures etc.) he computes a random value, called nonce, and sends it to the recipient.

Round 1 The recipient receives the nonce, and uses it to compute (see annex for detailed mathematical description) a cryptographic message with the attributes he likes to be included to the credential.
Round 2 The issuer receives the message and uses it to create the cryptographic part of the credential (by e.g. signing the attributes with his secret key) and replies by sending this to the recipient. For later accountability the issuer stores the used user pseudonym and the context.

Round 3 The recipient can now use the received message from the issuer to create and save the credential. He also saves the context for later credential updates etc.

3.3.2 The Prove Protocol

The prove protocol can be used to create a proof of possession of a certain credential. It is performed between two entities, namely the prover and the verifier. During this, the prover uses his credential(s) to create a cryptographic object, the so called called proof. This can be send to the verifier to convince him of the ownership of a credential. Some extensions also allow to include verifiable encryptions and enumerated attributes. Due to the fact, that there is little interaction in the protocol, one could also describe the two processes separately.

The first step is the creation of the proof by the prover. The prover defines a proof specification, where all attributes, used credentials and additional subproofs
are defined. An example of a proof specification is provided in Figure 3.3.6. In this example the possession of three different credentials is proven, and also implicitly the equality of some attributes in different credentials, by using the same identifier. With this specification, the values that should be included and a random value (beforehand requested from the verifier) the prover can compute the proof object. The second main step of the protocol is the verification of the statements in this object by the verifying entity. An example of a statement is the attribute age is > 18 etc. Therefore the prover sends the proof, together with the specification to the verifier. The latter now computes several verifying protocols to check if the formulated statements. After this, the verifier can send an response to the prover, depending on the result of the verification.

Every proof contains a general proof of possession of the CL-signature (named after the Camenisch-Lysyanskaya signature scheme). This is, that the prover is in possession of a credential sigend by the CA. Additional it is possible to formulate a broad range of different statements about the attributes to be included to the proof. Some of these statements should be mentioned here:

**Equality** This can be used to prove equality of attributes in different credentials. For example to prove that the name in the government ID credential is the same like the name in the drivers license credential. This is implicitly done
by including multiple credentials in the proof and using the same identifier for an attribute.

**Inequality** Can be used to formulate inequality statements. For instance that a number lies in a certain range (this is used to prove that the age of a person is > 18)

**Set membership** Enables to prove that a attribute value lies in a certain set of values or not.

**Verifiable Encryptions** A user specifies a public key (e.g. from a trusted third party) to allow for verification of an encrypted attribute value included in the proof.
Figure 3.3.6: Example of a proof specification file
4 Requirement Analysis

To meet the objectives of this thesis, further concrete requirements have to be derived in order to ease the realization of solution for our problem. Thus, we have chosen a representative Di.Me scenario covering the essential aspects from our problem (anonymity, un-linkability etc.). In the following, we first introduce the IBM Identity Mixer, before we provide Di.Me background information and describe the scenario chosen. Then, we address the requirements gathering in detail and summarize the identified requirements at the end of the chapter.

4.1 Di.Me Background Information

Since personal information gains more and more importance for the digital personal identity of users, the usage and disclosure is increasing [Harper et al., 2008]. Di.Me aims at integrating all personal data in a personal sphere by a single, user-controlled point of access: the Di.Me user-ware. This tool will run on the user’s devices, and intend to rely on scaleable peer-to-peer (P2P) capabilities in order to avoid external storage of personal data as far as possible and to enhance data portability. The Di Ме user-ware only envisages to store and processes data on the users devices unless the user explicitly initiates the disclosure to a third party. External services (e.g. web-communities, enterprise systems) will be integrated via gateways. Communication to individuals and services will make use of digital faces, i.e. partial identities bound to a given context and user data selected for a particular purpose and context. An open trust, privacy, and security infrastructure will enable the user to securely use personal data. Anonymous data disclosure, data withdrawal and policies will foster privacy and trust. For this, Di.Me targets to elaborate concepts such as digital faces and anonymity at the application level (anonymous user-controlled IdM) as well as at the network level.

Intelligent user interfaces also on mobile devices will intend to promote the intuitive usage of powerful PETs through intuitive and usable UIs. In addition, it will try to enable the user to monitor, control, and interpret personal data with the help of these UIs and PETs. This will be enhanced by the usage of semantic technologies like data-mining, intelligent analysis, mapping and others which support the users to maintain the large amount of collected data, communications and complex situations. The project implements a user-driven design process. Usability and uptake will be monitored and improved by large-scale
quantitative evaluations supporting a scaleable test-concept. Leading industrial partners will involve their customers for validating the project’s artifacts with consumers and professional users. Di.Me is targeting the involvement from 500 up to 10000 users. This show the need to address usability aspects with respect to security and privacy in the context of this thesis primarily from the performance point of view.

4.2 The Di.Me Scenario

The Di.Me user-ware tool is to be used in all digital life spheres, which includes business and private life, different communities and different contexts. Therefore, a couple of scenarios were selected to cover all different aspects, and also cover the case, where a user is attending in different spheres at the same time. To gather our realistic requirements, we oriented us on the "Business Conference Scenario" from the Di.Me project and adapted it, to make use of other ongoing and previous research projects. The choice of this scenario is due to its representative nature for the main security and privacy requirements (verifiable anonymous and un-linkable credentials) by considering those related to performance.

The setting of our scenario is a conference. The attendees have the option to publish some selected personal contact information e.g. on a conference website or any accessible place by the attendees. The other attendees can browse the information and could e.g. send contact requests. When publishing such information, it is possible to conceal some information and make them only fulfilling some authentication and authorization restrictions, e.g. available for people in possession of a particular attribute (e.g. the email address is only visible to people working in the automotive industry).

All attendees received a special credential, proving they are registered participants of the conference. This credential can be used to sign in to the conference website as well as to log in. Besides the public section of the website, where everybody can publish their contact information, there are several chat rooms and discussion forums. Attendees can e.g. create their own private chat room, provided with the same attribute requirement, as for personal information. If another attendee likes to join this chat room, he/she has to prove that e.g. his job is "automotive engineer".

4.3 Identifying Concrete Requirements

4.3.1 Concrete Requirement Related to O1 and O2: Porting Idemix and Developing a Prototype Reference Implementation

Actual business reports are stating that smartphone sales grew 96 percent from the third quarter last year, and smart-phones accounted for 19.3 percent of overall
mobile phone sales in the third quarter of 2010. Carolina Milanesi, research vice president at Gartners announced: "This quarter saw Apple and Android drive record smartphone sales. Apple’s share of the smartphone market surpassed Research In Motion (RIM) in North America to put it second behind Android while Android volumes also grew rapidly making it the No. 2 operating system worldwide". Because of this fact, we are favoring the Apple iPhone as well as one of the latest Android devices as target platform for our prototypic implementation. Due to a first analysis of the Idemix source code and due to previous experiences in developing iPhone applications (especially with the restrictions apple is enforcing to developers), we concluded that it would be more (time) expensive and difficult to port the Java Idemix library to Objective-C or C++, in order to build an iPhone prototype. In comparison the porting to Android platform, where applications are written in Java, would only consist of a few smaller adaptations to the source code. Another cause is that we have not to respect IBM’s license restrictions imposed for the usage of Idemix. IBM’s responsibles communicated to us, that they can only allow for the usage of their Java based code and that any translation to other programming languages has to be checked. In the case of Android, only minor changes and adaptations are expected so that they allowed for porting Idemix to Android. Furthermore, the Android platform independency and the design of the Android OS stack allows for more flexibility with respect to embed additional security layers if needed. This is in the case of Apple’s iOS very difficult if not possible according to the current development agreements. Consequently, we decided the implementation of the prototype for Android. Thus we formulate our concrete requirements related to O1 and O2 as follows:

1. Porting Idemix will target in this thesis the Android platform. (R1)
2. An interface module should act as a mediator for all intended Di.Me scenarios to the Idemix ported version. (R2)
3. The prototype implementation of "Di.Me’s Business Conference Scenario" also will be carried our on the Android platform. (R3)

4.3.2 Concrete Requirement Related to O3 and O4: P2P Business Scenario Implementation Supporting TOR-Anonymity

Developing a prototype reference implementation of an user-controlled IdM system for mobile end-user devices is one of the requirements of Di.Me. This implies that the IdM to be developed or integrated has to be deployable on mobile platforms with minimal costs in terms of computation consumption as well as memory footprints. Later will affect the performance (response times) and have crucial impact on usability acceptance. Indeed, Herranz et. al. states in [Herranz, 2009] that "A time-consuming credential generation or verification procedure may
be critical .. as regular user hardware (wether it is a smart-card or a handheld device) is not ready to support the newest algorithms which use shorter keys for an equivalent security level, like those based on elliptic and hyperelliptic curves.”. This shows, that implementations have to consider capabilities and limitations of a specific device and use realistic scenarios of the respective business domain by deploying them on the target device. In our case, we will evaluate Idemix for a potential integration as we formulated in R1 and R2. The resulting interfacing module (R2) has to be used in the prototypic implementation and because Di.Me wants to leverage P2P possibilities, a realistic P2P mobile collaborative scenario implementation (with a worst case effect on performance) has to be provided and evaluated on the target mobile platform. In this case, optimization with the help of session management mechanisms such as implemented in some related work for server-side interaction scenarios (e.g. [Armac et al., 2009]) will become not possible. According to this, the Di.Me "Business Conference Scenario" has to be implemented, so that it fulfills the previous requirements, namely, by providing a device-centric P2P implementation based on the resulting module from R2 with the possibility to enable TOR anonymity at the network level for our scenario. In order to issue credentials in our scenario we need a Certificate Authority. Adaptation of network security layer with end-to-end encryption and address anonymization to digital.me applications and devices in specific use cases. Empowering the user to use anonymization also at the level of the network connection should be provided in intended evaluations. For instance, a CA could be able to build profiles when contacted by users (at least for users’ location e.g. based on network localization mechanisms). Providing TOR anonymity in combination with user-controlled IdM in the mobile application should enhance the privacy. However, it will surely affects the performance of the application when the anonymity is enabled. For this, using an idemix CA or at least simulating it with realistic parameters has to be considered. Therefore we formulate our concrete requirements to meet O3 and O4 as follows:

1. The prototypic implementation required in R3 has to support P2P capabilities. (R3-Refined)
2. A realistic Idemix CA server implementation has to be provided. (R4)
3. TOR-Anonymity has to be supported, so that end-users could be able to enable or disable it on demand. (R5)
4. The performance Evaluation has to be carried our with the result of R3-Refined and R4 by considering R5 (also one time with enabled anonymity and one time without it). (R6)

4.3.3 General Development Related Requirements

Agile supporting of changing end-user requirements while building as well as evaluating developed prototypes and concepts is another main requirement in Di.Me.
4 Requirement Analysis

Last but not least and especially because Di.Me is targeting a large-scale end-user involvement many changes are expected while building the prototypic mobile applications, the integration as well as maintenance costs have to be considered as well. However, literature assess [Herranz, 2009] that in general, IdM authentication of Idemix-based systems ease the implementation work of developers. Furthermore, the Idemix designers are taken the ease usage of Idemix into account (also by developers without security background) [Bichsel and Camenisch, 2010]. For this, we have no explicit requirement in this work, but in order to be able to easily adapt and integrate the prototype in future works, we should also consider this.

4.3.4 Additional Requirements for the User Interface

Observations of previous work showed us that the usage of Idemix in general strongly depends on usability and interaction design. Also in combination with a low latency anonymity network the long response times could negatively affect the user trials since Di.Me targets the involvement of a very large testers community (from 500 up to 10000 as mentioned before). Since performance is seen as a quality of service requirement from the usability perspective and an availability requirement from the security point of view \(^1\), providing an usable (and intuitive) interface for generating digital faces and different kinds of proofs etc. is also crucial (R7).

4.4 Summary of Requirements

In summary, we identified the following concrete requirements:

R1: Porting of the Java library of Idemix to Android

R2: Development of interfacing module capable of performing the Idemix protocols

R2.1: Obtain credential from the CA

R2.2: Show Proof to Server

R2.3: Show Proof to other Client

R3: The prototypic implementation for the "Di.Me Business Conference Scenario"

R3.1 Enable Client-Server Communication

R3.2 Enable P2P communication between Clients

\(^1\) The reader may remember the security triangle: confidentiality, integrity, and availability. Availability signifies that all data is at anytime and from everywhere available
4 Requirement Analysis

**R4:** Development of a server application to act as prototypic Certificate Authority

**R5:** Enable anonymous network communication over the Tor network

**R6:** Measurement and evaluation of the performances with different configurations and the effect of TOR-Anonymity on the latency

**R7:** Prototypic provision of an usable (and intuitive) interface for generating digital faces and different kinds of proofs
In this work, Idemix represents the best possibility to fulfill Di.Me’s requirements without additional development costs and was therefore predefined as given for this work. Idemix is licensed for non-commercial usage in EU projects since it was and still is developed and used in projects funded by the EU (Prime and PrimeLife). Such proceeding of using results from previous EU funded projects is also strategically preferred by the EU. However, Idemix represents a cutting-edge framework in comparison to other solutions like U-Prove [Brands and Paquin, 2010]. We first present related work to Idemix in general and focus on those providing performance evaluations results (R1, R2.x, R4-R6). Then, we address the user-controlled access rights management (i.e. authentication and authorization) in collaborative mobile settings with respect to usability issues (R3.x, R7).

5.1 Idemix Related Work (R1, R2.x, R4-R6)

Idemix has been used in the PRIME FP6 and the PrimeLife FP7 EU projects to build prototypes. Idemix has been used in the PRIME FP6 and the PrimeLife FP7 EU projects to build prototypes. A competitive Microsoft implementation called U-Prove has been recently released [Brands and Paquin, 2010]. However, Idemix specifiers affirm that they provide with Idemix a much more extensive specification and more features in comparison to U-Prove [Bichsel and Camenisch, 2010]. All these strengths made it perfect for realizing an user-controlled IdM supporting digital faces in Di.Me for both; collaboration of users with each other as well as collaboration of users with a given service.

Other works such as [Bichsel et al., 2009], [Verslype et al., 2008], [Armac et al., 2009], and [Verslype et al., 2010] used idemix for different scenarios. However, our elaboration of published results showed us that none of these works fulfilled all our requirements on the same mobile client and thereby considered using (1) complex mobile collaborative scenarios, (2) by supporting P2P settings in addition to other broadcasting possibilities e.g. client/server, (3) by involving a realistic CA, and (4) by allowing for the option of anonymity at the network level (e.g. TOR integration).

Bichsel et. al [Bichsel et al., 2009] introduced their implementation of Idemix on a standard Java Card where they achieved transaction times of about 10 seconds with the severely restricted computation power of such a card. At least not
all at the same time since [Verslype et al., 2008] represents an exception in this respect. With their PetAnon system, they implemented a privacy-preserving e-petition system which supports participation in the democratic decision-making process. It has been used in the Belgian eID card (in a bootstrap procedure), after which users are allowed to sign petitions anonymously, while certain constraints may be imposed. Since this kind of scenarios has a collaborative nature and was deployed on a platform similar to mobile devices (smart card), the performance evaluation results presented there are still representative at a certain degree (even though it does not fulfil R3).

In [Verslype et al., 2010] Verslype et al. are introducing PriMan, a flexible middleware framework that should support developers in the design and implementation of privacy-preserving applications. However and again, none of these works used the forth complete implementation recently released this year which makes provided results obsolete especially for current mobile devices. The less contribution of this paper should be providing up-to-date performance evaluation results for the first complete idemix implementation on a mobile platform. Armac et al. confirm in [Armac et al., 2009] that the use of the idemix 2009 implementation is not as efficient as desired. By contacting the idemix group, later have confirmed that "the performance of their reference implementation is not optimised yet and future versions are expected to run efficiently on mobile devices such as PDAs."

5.2 Related Work to Usability of Access Control in Collaborative Mobile Settings (R3.x, R7)

In the following, the presentation of related work to usability in collaborative mobile settings is oriented to the state-of-the-art summarized by Bourimi et al. in [Bourimi et al., 2009] and [Heupel and Bourimi, 2011]. We focus here primary on main mechanisms of access control, namely, authentication and authorization.

Social interaction is mostly supported by different categories of collaborative systems and social software. Those systems have to fulfill multi-user requirements and are consequently characterized by complex scenarios supporting these requirements in the respective domain. Often, this complexity is reflected in the user interface (UI) which becomes crucial for mobile applications deployed on mobile devices with limitations in the screen space [Shneiderman et al., 2009] [Boyle et al., 2008]. According to Shneiderman et al., "an extrapolation of current trends leads to the suggestion that most computer-based tasks will become collaborative because just as most work environments have social aspects" [Shneiderman et al., 2009]. Thus, software systems and applications supporting social interaction are considered as socio-technical systems in the Computer-Supported Cooperative/ Collaborative Work (CSCW) as well as Human-Computer Interaction (HCI) research fields [Gross and Koch, 2007] [Shneiderman et al., 2009].

A fairly large number of research work is focused on making access rights
management usable for end-users. An overview on existing approaches for access rights management in collaborative environments has been provided by Bullock and Benford in [Bullock and Benford, 1999] as well as in Haake et al. in [Haake et al., 2004]. Latter work reviewed the state of the art especially by focusing on group formation and access control in shared workspace systems concluding that "in todays shared workspace systems access rights management by end-users is insufficiently supported, either due to too complex role models, access control parameters and user interfaces that end-users cannot easily understand, or due to insufficient functionality". They provided a combined approach by using the room and virtual keys metaphors allowing dynamic groups to be formed without privileged users. End-users (instructors, tutors, students etc.) are able to form groups (1) by key assignment, (2) by invitation, (3) with free enrollment, and (4) enrollment confirmed by the members of the respective groups [Haake et al., 2004]. Furthermore, they provided an intuitive UI for supporting their approach in [Schummer et al., 2005].

For both; authentication and authorization, cryptography is an established used mechanism for increasing confidentiality and integrity of exchanged data. However, a total security or privacy provision is an illusion [Hong and Landay, 2004] because current approaches are not able to avoid at least threats and attacks e.g. emerging from loosing devices or based on physical access to them [Dwivedi et al., 2010]. Approaches focus on hindering such attacks. Security and usability research for developing usable (psychologically acceptable) security mechanisms is a young research field which depends on the context in which those mechanisms are to be used [Cranor and Garfinkel, 2005]. Because of this and many facts cited above, Bourimi et al. argue in [Bourimi et al., 2009] and also in [Bourimi et al., 2010a] that security and privacy design by considering usability is specific to the project context (here Di.Me).

5.3 Summary

In summary, the main contribution of this thesis mainly consists of (1) providing current performance evaluation results when using Idemix on modern mobile devices (i.e. Android platforms), and (2) supporting thereby optional anonymity at the network level (predefined TOR network). In contrast to related work introduced above, we thereby use the latest implementation of Idemix recently released. Additionally we implement a server application to simulate an Idemix CA and involved it in the evaluation process. Since performance is strongly influencing the usability of applications, we also put focus on the usability of the resulting prototype and will provide an elaborate UI prototype in order to reduce the trade-off between usability and security.
6 Approach

In this chapter we provide a detailed description about the concrete fulfillment of the requirements gathered in chapter 4. Also decisions we made when designing the prototype application, like e.g. used communication protocols, should be explained. The requirement concerning the performance evaluation will not be mentioned in this chapter, since it is such an important part of this thesis that we dedicate a separate chapter for it (see chapter 8).

6.1 R1: Porting of the Idemix Library

One of the main objectives of this thesis it the evaluation of the performance of Idemix on a modern smartphone. In order to fulfill this and the other gathered requirements like the implementation of the prototypic application, we used the Java open source library of Idemix, which can be obtained at the website of the PrimeLife project [PrimeLife Team, 2010]. As mentioned in the requirement analysis, the targeted mobile platform for this approach is Android. Consequently the first requirement (R1) was the porting of the Idemix cryptographic library to the Android platform. Because the original Idemix library is written in Java, and the programming language for Android applications is also Java\(^1\), only a few adaptations were necessary, which are briefly summarized in the following.

The first adaption was to remove the dependencies from third party libraries, due to incompatibilities to the Android platform caused by differences in the Java APIs. For example, the external library, the original Idemix project uses for XML processing, is not compatible to Android, thus we needed to remove such dependencies. To achieve this we made several adaptions to the XML parsing routines, in order to use only the standard Java API provided by Google. The second dependency to an external library for logging purposes was relatively easy to get rid of. By doing so, it was possible to only use the standard Google API which made the porting process a lot easier. Some additional changes to the Idemix source code were necessary because of differences in the standard Java API of Oracle (previously Sun) and the one used by Google. After we successfully ported the Idemix library to Android, we again performed the Unit Tests that were provided with the original source code. The results showed that

\(^1\)As also explained in later in 7.6.2 it is also possible to use native C/C++ libraries in applications, but since the Idemix library is written in Java that was no option.
the algorithms are still correct by providing the same results as before.

6.2 R2: Development of the Idemix Module

The second requirement (R2) demands to provide an interfacing module to be able to use the Idemix of the protocols in many different contexts and to be easily adaptable in different settings. Therefore we encapsulate the Idemix library with a more usable and flexible interface. In this section we will focus on the explanation of this interface class, responsible for performing the main Idemix protocols on the client application.

Figure 6.2.1: Simplified class diagram of the client application

We developed a core component that can be seen as some kind of mediator between the GUI, the Idemix library, and the communication layer. We also developed another important helper component for setting up and configuring Idemix as well as to easily switch between different configurations. This allows on the one hand to easily adapt the application to different contexts and on the other hand it helps us to fulfill requirement R6 (see also chapter 8 for more details).

Figure 6.2.1 illustrates the designed application architecture. In the middle is the class *IdemixConnection* that is mediating between the other components for
communication, Idemix related cryptography and user interaction. In the following subsections the most important methods of the mediating module API, which are implementing the main Idemix protocols are explained in detail. We abstract here from the implementation details and will only describe the functionalities and general design decisions concerning those core components.

6.2.1 R2.1: Obtain Credential

In order to receive a credential, the client needs to perform the Issuance Protocol with a Certificate Authority. Therefore we implement an interface for initializing the Client and to perform the different steps of the protocol (see 3.3.1 for the protocol description).

6.2.2 R2.2 and R2.3: Show Proof

As defined in the requirement analysis the client needs to be able to formulate a predicate about the attributes of his/her credentials and proof the correctness to another client or a server. From the perspective of the mediating core component (see IdemixConnection in Figure 6.2.1) it makes no difference, if one is showing the proof to a server or to another client. The designed component performs the steps of the protocol (see 3.3.2 for the protocol description), like the loading of the proof specification and the computation of the different sub-proofs. The module can therefore use any class implementing the interface IdemixService in order to perform the protocols used to compute and send a proof.

6.3 R3: Prototypic Implementation of the Business Conference Scenario

The main part of the prototypic implementation was the fulfillment of the requirement to support the chosen Di.Me conference scenario (R3) so that the performance evaluation is as realistic as possible. To model the scenario we implemented the architecture depicted in Figure 6.3.2.

In order to provide the required functionality to implement the scenario we needed a flexible communication layer. The interface is therefore designed with the help of Remote Procedure Call (RPC) technology. The interface (Idemix-Interface) is also shown above in Figure 6.2.1. For the application it looks like a local method call, but it is actually a method call on another machine. This allows to abstract from the concrete methods used in order to communicate with another client or server. To achieve this abstraction we implement a general Interface IdemixService for the communication. This allows us to change the network communication layer, without touching the rest of the application, which is a general approach of software engineering\(^2\). Since the general capability of

\(^2\)This follows the concept of Separation-of-Concerns, see e.g. [Balzert, 2009]
the client application to perform the Idemix protocols was already addressed in section 6.2, we will focus in the following on the capabilities to communicate with different entities in order to perform the protocols. To support this we developed a server centric and also a P2P approach. This is necessary because to cover the scenario a flexible architecture is necessary. Both design decisions will also be discussed in the following in order to satisfy the requirements R3.1 and R3.2.

The clients are able to show proofs to each other over the XMPP server and to show them to a server implementing the developed interface. In our implementation the Certificate Authority and the other supported servers are running on the same physical machine, which is not contradictory to the scenario, since in our setting the conference organization who is providing the website is also issuing the “conference credential”.

6.3.1 R3.1: Client-Server Communication

In order to satisfy R3, the client application needs to be able to perform the Idemix protocols to support the Di.Me scenario. The cryptographic functionality is brought by importing the Idemix library and the performance of the protocols on the client was already described in section 6.3. In order to perform the credential issuance and the show proof protocol the client and the server need to
exchange data somehow. Therefore we need a communication mechanism that suits our requirements. Common practice is the use of XML based communication protocols, because of their capabilities to be platform and programming language independent. Although we know that XML processing is slightly influencing the performance of the transactions, we chose it anyway, because interoperability is very important. The use of mobile devices should not be restricted to the ones written in Java. Anyway, the minimal difference in performance due to the XML processing is negligibly small in relation to the computation time of the cryptographic algorithms.

The Certificate Authority (CA) is written in Java and implementing an XML-RPC server, so we can send XML-RPC requests to perform the credential issuance protocol. The server supports full Idemix capabilities, so it is also possible to show proofs to the server. This can be used, for example if an additional authentication is needed in order to sign a new credential. It also allows us to easily fulfill requirements (we are not focusing on in this thesis) like group awareness and real time communication. As mentioned before, users can create their own chat rooms and can define certain restrictions other users have to fulfill in order to join. At the moment there is just password protection, but in order to obtain the password, users can send an Idemix proof to the owner.

6.3.2 R3.2: Client-Client Communication

One of the requirements derived from the Di.Me conference scenario was that clients should be able to show proofs to each other. Because of the fact that it should not depend on the availability of a local Wi-Fi network and that mobile phones in Germany have no static IP address (unlike e.g. in Australia), XML-RPC is not suitable in this context. Based on good experiences with a satisfying approach in this context [Bourimi et al., 2007] and another previously implemented project [Bourimi et al., 2010b], we chose to use XMPP for our approach. It is also XML based and allows us easy to establish client-to-client communication. The XMPP network uses a decentralized client-server structure with no authority server like in other networks (AOL, ICQ etc.). Anyone may run their own XMPP server on their own domain. To avoid the need for a central server, which knows all worldwide given IDs, the ID is constructed like an email address (username@domain.xy). XMPP is known for its strengths, like e.g. the flexibility that allows adding custom functionality, the good security, the decentralization, and the fact that is is an open standard with a long history [Adams, 2001].

In order to show proofs to other users, the clients need to be connected to an XMPP server. By default, XMPP supports mainly functionality for communication and awareness purposes, but it can be extended and customized. This adaptability helped by implementing a custom XMPP extensions, allowing for sending customized Idemix-messages to be able to perform the Idemix protocols. Using XMPP is very beneficial for the approach of this thesis and ongoing work, because is enables to easily implement additional cooperation features like group
Figure 6.3.3: Sequence diagram of the broadcast use-case

6.4 R4: Development of the Certificate Authority Server

In order to be able to issue credentials for our prototypic scenario, we needed a certificate authority. This server application should also be able to verify proofs, sent from the client application. Since the server application is a native Java application, no adaptions to the Idemix library were necessary. To perform the
Idemix protocols we implement an interfacing module able to issue and to verify credentials. Also it has an integrated Web server, allowing clients to send XML-RPC requests. The general architecture is very similar to the client application. One of the main differences is that the server has no user interface. In order to maintain the issuance and proof verification all performed steps are logged to the console. With this server application we fulfill requirement R4.

As depicted in Figure 6.3.4 the general architecture is comparable to the architecture of the client application. The implemented interface, the class IdemixConnection, is one of the core components of the implementation and defines the for the RPC-Method calls. There is also a class SystemConfiguration similar to the one in the client which is a helper class for setting up an configuring Idemix and also to easily switch between different configurations.

6.5 R5: Anonymous Network Communication

To fulfill the requirement for anonymous communication using the Tor network, we installed the Orbot client [The Tor Project, 2010] on our testing devices and also on the emulator. This is a local proxy application allowing to route all
SOCKS\(^3\) connections over the Tor network. In order to work properly, the Orbot client has to be started before the application starts, so that the proxy service is running in the background. In the application it is now possible to activate the anonymization, so that the XML-RPC calls will be routed over the Tor-network. Thereby the user can decide by his own, if he likes to be anonymous when issuing the credential or not. The respective interface to do this is also described in the next section (6.6).

6.6 R7: Providing a User Interface Prototype

The last of the requirements was to provide a prototypic UI in order to dynamically build a proof and to view and edit digital faces. There we developed several GUI masks in order to fulfill requirement R7. In the first screenshot (Figure 6.6(a)) a GUI mask to define the attributes and the context of a digital face is illustrated.

![Figure 6.6.5: Selected GUI masks of the prototype](image)

(a) GUI mask to edit a digital face  
(b) GUI mask to create a proof face

Figure 6.6(b) depicts the first view in order to create a custom proof object. The user can select a credential and the containing attributes are listed below. Now it is possible to select the attributes that should be revealed in the proof. By clicking Continue the View, shown in Figure 6.7(a) will show up. Here the user can choose different predicates he/she likes to add to the proof. Screenshot 6.8(b) therefore illustrates the View to formulate an Inequality predicate.

\(^3\)SOCKS is an Internet protocol allowing the routing of network packets via a Proxy server. The name is an abbreviation for Sockets; Internet socket or network socket is an endpoint of a bidirectional inter-process communication flow across an IP-based computer network, e.g. the Internet.
Figure 6.6.6: Selected GUI masks of the prototype to build a proof

(a) GUI masks to add additional subproofs

(b) GUI mask to define an inequality proof

Figure 6.6.7: Additional GUI masks of the prototype

(a) GUI Mask for application settings

(b) GUI Mask for listing available digital faces
7 Implementation

To implement the prototype for Android and the server application and as well to be able to evaluate the performance and some usability aspects of Idemix and Tor in our scenario, we used the Java open source library of Idemix, which can be obtained at the website of the PrimeLife project [PrimeLife Team, 2010]. In this chapter we provide a more detailed view on the implementation details. Consequently we look into the concrete implementation of the main functionalities and explain them with the help of several selected code examples.

7.1 Adapations of the Idemix library

As already mentioned in the Approach (see chapter 6.1), we made a few adaptions to the original Idemix source code in order to use the library on Android. Here we provide a few implementation details to this adaptions. The first adaption was to remove the dependencies from third party libraries, due to incompatibilities to the Android platform. For example the external library for XML processing (Apache Xerces) is not compatible to Android, so we needed to remove the dependencies. We made some adaption to the class `Parser.java`, whose purpose is to parse XML files (e.g. Credential Structure files). By doing so, it was possible to only use the standard Google API for the XML processing. In order to get rid of the dependency to the `commons.logging` library we implemented a wrapper class in order to use the native Android API for logging (`android.util.Log`).

Some additional changes to the Idemix source code were necessary because of differences in the standard Java library of Oracle (previously Sun) and the one used by Google. For instance, in a few algorithms in the code, the class `java.util.TreeMap` is used. This class is available in the Google as well, but it has a different superclass. In the standard Java API the class TreeMap implements the Interface `java.util.NavigableMap` but in the Google API there this interface is not available. The same problem exists for example with the class `TreeSet` and the interface `NavigableSet`. Because those interfaces mainly provide some convenient methods for the access to the elements stored in a set, it was possible to change the algorithms, so that they are using the classes `java.util.Map` and `java.util.Set`. Several Unit Tests showed that the algorithms are still correct\(^1\). Finally the were

\(^1\)The corresponding Unit Tests are, together with the source code, on the CD attached to this thesis.
also a few adaptions necessary in order to save files to the local filesystem on Android. If files need to be saved to the applications private folder, the writing routine needs to know the current Application Context that, besides other information, holds the address of the applications folder\textsuperscript{2}. Because of the problem that the original methods, to save a credential by calling “credential.save(filename)”, have no access to the applications’ Context, we could not use them. To still be able to save data, we implemented a separate routine, allowing us to save issued credentials to the device.

7.2 Implementation of the Core Module

One of the most important classes of our prototypic Android application is the class \textit{IdemixConnection}. As also depicted previously in Figure 6.2.1 the implemented interface, the class \textit{IdemixConnection}, is one of the core components of the implementation. It can be seen as some kind of mediator between the GUI, the Idemix library, and the communication layer. There is also a class called \textit{SystemConfiguration} which is a helper class for setting up an configuring Idemix and also to easily switch between different configurations. Since the two protocols \textit{IssueCredential} and \textit{ShowProof} are two essential components of the whole application, they will be discussed in the following. Thus we provide representative code examples to illustrate the implementation in detail.

7.2.1 Implementation of the Credential Issuance

In order to receive a credential, the client needs to perform the Issuance Protocol with a Certificate Authority. Therefore we implement an interface for initializing the Client and to perform the different steps of the protocol (see 3 for the protocol description). The following code examples illustrate how the issuance of a credential is implemented in the prototype. As shown in Code Listing 1 the first step is the loading of the Issuance Specification from the file (\textit{CredStructGermanID.xml} in this example) and the \textit{SystemParameters}. As also mentioned before in chapter 3 the Credential Issuance Specification file defines which attributes to include and what issuance mode to use (Issuance modes are: Revealed, Unrevealed, Committed). In this example, all attributes should be issued as \textit{REVEALED}. This means that Issuer and Recipient use the same \textit{Values} (initiated in Line 15 in Code Listing 1) object and all issued values are visible to the Issuer, too.

The next Step is the loading of the actual attribute values to include to the credential. All values need to be added to the \textit{Values} object. In this example they are hard coded to keep it simple.

\textsuperscript{2}As defined on [Google Inc., 2010] the \textit{Context} is an interface to global information about the application environment. The feature of interest here is that it allows access to application specific resources, like the private application folder. This is a folder where the application can store private data with read/write restrictions for other applications.
When all values are prepared the Recipient is finally initialized (see Line 24 in Code Listing 1).

The next steps in the protocol are shown in Code Listing 2. This is where the actual Issuance protocol is performed, like it is described in section 3.3.1. The client initiates the issuance protocol by sending the name of the CredentialStructure file to use and the values that are revealed during the Issuance and the Issuer answers with a random value (nonce) when initialized. The client now computes the first message with the random value and sends it to the Issues, who creates the second message and sends it back to the Recipient. This message (IRMess-age2) is containing the data for the credential and additional data that allow the recipient to check if the Issuance was done correctly. After the correctness is ensured, the credential can be saved to the device and can to be used in the proof protocol.
7 Implementation

BigInteger nonce = IdemixService.initiateIssueCredential(
    valuesRecipient,
    SystemConfiguration.FILE_LOCATION +
    "CredStructGermanID.xml");
RIMessage1 msgToIssuer1 = recipient.round1(nonce);
IRMessage2 message2;
Creditial cred = null;
try{
    message2 = IdemixService.computeRound2(msgToIssuer1);
    cred = recipient.round3(message2);
} catch (Exception e) {
    return false;
}
// Recipient received answer, computes credential
if (cred == null) {
    System.out.println("Credential not created");
    return false;
}
// Save the credential
boolean result = saveCredential(cred, "credGermanID.bin");
return result;

7.2.2 Implementation of the Proof Protocol

In this section we have a close look at the implementation of the ShowProof protocol. To illustrate this, Code Listing 3 shows a simple example of the code to create a proof and send it to a verifier.

It starts similar to the credential issuance protocol, by loading the specification file, called ProofSpec in this case. After that, the credential(s) that should be included in the proof need to be loaded and stored in a HashMap. The identifier in this HashMap needs to be the same name that is defined in the ProofSpec file, otherwise the proof will not compile. The next important step, is to request the random value (nonce) from the verifier (see also chapter 3 for the protocol specification) that is done by calling IdemixService.initiateVerifier("parameterUrl"); (see Line 11 of Code Listing 3). The parameter "parameterUrl" is used in our prototype for defining the location where the configuration files (SystemParameter, GroupParameter etc.) that are currently used by the client, are stored. This
allows for easily switching different configurations and thereby eases the execution of the measurements (this will also be discussed in chapter 8). When the client received the nonce from the verifier, it is used in order to initialize the Prover with the clients’ master secret, the HashMap containing the credentials and the proof specification. In this simple example the Proof does not contain any additional predicates, such as commitments, verifiable encryptions, inequality proofs etc. If that was the case, they need also be precomputed and added as parameters to the initialization of the proof. If there are no predicates, the parameters can also be null (as depicted in Line 16 of Code Listing 3). After initializing, all that is left to do is to build the proof object and to send it to the verifier. If there are additional predicate, they need also be communicated to the verifier. But since the actual attribute values should remain hidden, this are not the same as used in the initialization, but special encrypted versions of the predicates.

7.3 Implementation of the Network Interface

We implemented a general Interface `IdemixService` for the communication. This allows us to change the network communication layer, without touching the rest of the application. In our approach this was very helpful when we added the feature
to be able to perform the protocols also over XMPP. This interface defining the API the Idemix module (described before in section 7.2) is using to perform the Idemix protocols with another entity. In the following we look look into the two implementations we provided, in order to use XML-RPC and also XMPP. Thereby we are supporting classic Client-Server communication and also P2P-Communication over XMPP.

7.3.1 The XML-RPC Service

To perform the XML-RPC requests we are using the Android-XML-RPC library and on server-side the XML-RPC implementation from Apache. Therefore we implement a simple web server application that supports the defined Idemix Interface. An example of that interface definition is shown in Code Listing 5. It allows clients to call methods on the server, in order to perform the protocols to issue a credential or to verify a proof.

7.3.2 The XMPP Service

Our prototypic implementation supports XML-RPC to perform issuance of credentials at a CA and as well XMPP for P2P proof protocols. The XMPPService is implementing the IdemixService interface, in order to be easily integratable to the prototype. In order to be able to send messages to other clients we also implemented the class XMPPClient that is responsible for connecting and logging on to the XMPP server. For implementing the XMPPClient we use the aSmack library [The BuddyCloud Team, 2010] which is a fork of the smack, a XMPP library for java [Ignite Realtime, 2010], optimized for Android. Since aSmack only provides basic XMPP functionalities, like chat, presence notification etc., we need to implement a custom extension in order to send the messages that allow for the protocols of Idemix. An example of a proof message is depicted in Code Listing 4.

7.4 Implementation of the Certificate Authority

The Certificate Authority is implemented very similar to the client application, but there was no adaption to the Idemix library necessary in order to use the Idemix protocols. We implement a Web interface for the XML-RPC calls that allows us to send XML-RPX requests to the server in order to perform the Idemix protocols. Code Listing 5 illustrates a small part of this interface definition.

7.5 Enabling Anonymous Network Communication with Tor

To fulfill the requirement for anonymity, we used Orbot [The Tor Project, 2010] an Android port of the TOR client. It provided a local proxy which allows us
to anonymize our TCP traffic on demand. The anonymization can be manually activated or deactivated by the user allowing him/her to hide his/her IP address at the price of lower latency. To achieve the routing we extended the XML-RPC client with the capability to use the local proxy when sending requests to a server. To activate the anonymization we make use of a parameterized constructor determining if the proxy should be used or not.

7.6 Additional Implementation Details

In this section we will mention the development environment, used library and provide a short summary of the targeted mobile platform.
7 Implementation

```java
public String getIssuerPublicKey();
public BigInteger initiateIssueCredential(
    Hashtable<String, BigInteger> values,
    String credStruct_fn);
public byte[] computeRound2 (byte[] message);
public BigInteger initiateVerifier();
public BigInteger initiateVerifier(String folder);
public boolean verifyProof(
    byte[] proofByteArray,
    byte[] specByteArray,
    byte[] commByteArray,
    byte[] repsByteArray,
    byte[] msgByteArray,
    byte[] verEncByteArray);
```

Code listing 5: The Idemix Interface

7.6.1 Eclipse and the Android SDK

The Eclipse platform, is a plugin oriented development environment, which can
for instance be used for application development in several different programming
languages. In this work we are using the Java Eclipse IDE 3.5 (Galileo) and the
Android Development Tools (ADT) plugin for Eclipse.

7.6.2 Android

Android is a software stack for mobile devices that includes a mobile operating
system, middleware and key applications. Android was initially developed by
Android Inc., which was purchased by Google in 2005. The Android operating
system is based upon a modified version of the Linux kernel. Android applications
are generally written in Java, making use of special Java libraries developed by
Google. It is also possible to develop native Android libraries written in C or
C++. Since October 2010 there is a special SDK available on the official Android
developer site (see [Google Inc., 2010]), called Native Development Kit (NDK).
But this is mainly only used to build special libraries for performance critical part
of the application. But nevertheless, the development language recommended by
Google is still Java. The complete source code of Android is open source and
published under an Apache license, what makes it also attractive for companies
developing special Android devices.

The Android operating system software stack consists of Java applications
running on a Java based object oriented application framework on top of Java
core libraries running on a Dalvik virtual machine. Since Android version 2.2 it
is also supporting JIT compilation.

The Android core libraries are written in C and include an SQLite relational
database management system, the OpenCore media framework and the WebKit layout engine, to just name a few. An overview about the Android system and the core libraries is provided by Figure 7.6.1

![Android system architecture](image)

**Figure 7.6.1:** The Android system architecture [Google Inc., 2010]

### 7.6.3 The Android Development Tools

The Android Development Tools (ADT) is a plugin for Eclipse and adds some useful extensions to the Eclipse IDE. It consists of several integrated tools which help to create and debug Android applications. So they help creating new Android projects, writing code and testing the code on an emulator.

### 7.6.4 Communication protocols

#### 7.6.4.1 XML-RPC

The *Extensible Markup Language Remote Procedure Call* protocol is a specification to call methods on distributed Systems. It was developed to easily exchange data between different Systems in different programming languages. Therefore it uses XML for the representation and the HTTP protocol for the data transfer. A basic example of an XML-RPC method call is provided in listing 6.
<?xml version="1.0"?>
<methodCall>
    <methodName>Idemix.methodname</methodName>
    <params>
        <param>
            <value><string>stringvalue</string></value>
        </param>
        <param>
            <value><int>10</int></value>
        </param>
    </params>
</methodCall>

7.6.4.2 XMPP

The Extensible Messaging and Presence Protocol (XMPP) is an open technology for XML routing, mainly used for instant messaging and presence information, but also for multi-user-chat (MUC), voice and video calls, collaboration, lightweight middleware and content syndication [Saint-Andre et al., 2009]. Unlike most instant messaging protocols, XMPP is an open standard. Like e-mail, it is an open system where anyone who has a domain name and a suitable Internet connection can run their own XMPP server and talk to users on other servers. The standard server implementations and many clients are also free and open source software. It is known for its strengths, like e.g. the flexibility that allows adding custom functionality, the good security, the decentralization, and the fact that is is an open standard with a long history [Adams, 2001]. The XMPP network uses a decentralized client-server structure. There is no authority server like in other networks (AOL, ICQ etc.). Anyone may run their own XMPP server on their own domain. To avoid the need for a central server, which knows all worldwide given IDs, the ID is constructed like an email address (username@domain.xy). This IDs is called JabberID or just short JID. If a user wants to use his JID from multiple locations (or devices), this is possible by using the so-called “resource”. It is appended to the JID after a “/”(i.e. for mobile and home username@domain.xy/mobile and username@domain.xy/home). Adding additional functionality to the core functionality can be done by implementing the XMPP Extension Protocols which are actively developed by the XMPP Standards foundation ([Foundation, 2010]).

7.6.4.3 The eJabberd Server

To support direct user communication as well as for several different awareness purposes we are using the eJabberd server [EJabberd, 2010]. It was also used in the the previous projects of Bourimi et. al. described in [Bourimi et al., 2007] and also in [Bourimi et al., 2010b]. In this work, we mainly use it, to perform the
client-to-client proof protocol. EJabberd is an XMPP instant messaging server, written in Erlang. It is licensed under GPLv2 so it is free to use and open source. The server is only of about 10MB in size and can be run under Microsoft Windows as well as under UNIX derived platforms. As a result of being build on top of the open source software Erlang, it has a database and a web server already included. The server supports clustering, so it is possible to connect multiple machines together, serving the same or several Jabber domain(s). The nodes in the cluster are fault-tolerant, which means, if one or more of the nodes crashes, the others will be able to continue working without serious adverse effects. This is a positive aspect concerning general security aspects by enhancing the availability of the server infrastructure. EJabberd claims to be fully XMPP compliant and furthermore supports a long list of XMPP extension protocols, as well as security mechanisms like SASL and STARTTLS.

7.6.5 Used Libraries

In our implementation we use a couple of third party libraries, in order to provide the required functionality. The most important ones should be shortly mentioned here.

7.6.5.1 Idemix

In order to implement the Idemix functionality we import the open source library, published on the PrimeLife website for non-commercial use. The detailed functionality was also addressed in an own chapter about Idemix. Necessary adoptions were also explained earlier in this chapter in section 7.1.

7.6.5.2 Apache XML-RPC

Apache XML-RPC [Apache, 2010] is an Java library to provide XML-RPC functionality. It comes with an API to easily implement a Server or Client application.

7.6.5.3 Smack

Smack is an open source XMPP library, developed by ignite realtime [Ignite Realtime, 2010]. It is a pure Java library and allows for full XMPP functionality like e.g. instant messaging and presence notifications.

7.6.5.4 The Orbot Tor-Client for Android

The Tor client for Android, called orbot [The Tor Project, 2010] Allowing to route all traffic over the TOR-Anonymity Network. In principle the Orbot client (and any other Tor client) is a locally installed proxy server, allowing applications to route SOCKS connections over the Tor network.
7.6.5.5 XML-RPC for Android

Android XML-RPC is an open source project for Android, which provides simple XML-RPC functionality for the Android platform. It is licensed under the Apache 2.0 license and can be obtained at the projects’ [android-xmlrpc Team, 2010] google code site.

7.6.5.6 aSmack

Since the Java library smack (also other XMPP libraries in Java) could not just be imported and used in an Android project, for a couple of reasons, the BuddyCloud team [The BuddyCloud Team, 2010] published a fork of the original smack library for the Android platform. It is also open source and can be obtained from their google code site.
8 Performance Evaluation and Results

Requirement 6 was to measure and evaluate the performance of the Idemix protocols in the given scenario. Because it is also one of the most important objectives of this work, and will also be an evaluation of the whole approach, this will be discussed in this chapter separately. In the following we will describe the preparations and execution of different tests we formulated in order to carry out the measurements of the performance. Each test is repeated for about 50 times in a row, to be able to get reliable results and to identify discordant values. After the presentation of the results, we will describe the evaluation and discussion of the results.

8.1 General Aspects of Performance Evaluation

8.1.1 General Definition of Performance Evaluation

In the center of the evaluation is the performance, which can be defined in several ways. The common way, is to look at computation times and memory consumption from an objective point of view. When a user looks at a system, he/she often describes the performance of a system subjectively (e.g. “this system works fast”), what is very inaccurate and allows no real evaluation. A good practice to evaluate performance is to look at the capabilities of a system to solve intended tasks in a certain amount of time [Mos and Murphy, 2002].

In terms of quantitative evaluation are two aspects of importance:

Response Time: the time until an transaction is completed and

Throughput: the number of transactions in a given time interval.

When we are talking about Web applications there is a special term for the response times, called Web Interaction Response Time. It defines the time until an answer to a request is completely received. It is also called Roundtrip time [Halter and Munroe, 2001]. This is of high relevance for this thesis, since the XMPP and also the XML-RPC requests are HTTP bound, and so we are always talking about “Roundtrip time” when e.g. a proof is sent to a verifier and the time until the response arrives is measured.
8.1.2 General Objectives of Performance Evaluation

The main objective of almost any performance evaluation is the optimization of critical aspects of a system. Not sufficient response times should be identified and the performance should be enhanced if possible. Thereby the performance tests can focus on different aspects of the system [Halter and Munroe, 2001]:

1. Analysis of non-functional aspects of the system. In general this is a so called stress test, carried out to evaluate the scalability and to identify the point where the system crashes.

2. Analysis of functional requirements.

3. Analysis and comparison of different infrastructures (e.g. databases, network technologies).

One on the main objectives of this thesis is the evaluation of the Idemix protocols on a representative modern smartphone. Therefore we will on the one hand, perform some kind of stress test, but not to identify the level when the system crashes. It is of more relevance to find an acceptable tradeoff between the level of security (main parameter for this is the length of the keys) and the usability of the application (defined by the Roundtrip times of Idemix related interactions). On the other hand an objective is to identify potentials for optimization of the implementation of the Idemix protocols, if possible.

8.2 Evaluation Setting

For carrying out a performance evaluation of the Idemix protocols, we implemented the Di.Me scenario like described above and deployed it on an Android device. In order to be able to easily perform different measurements, we implemented a special GUI, to start different prepared use-cases, which were processed multiple times. To be able to separately observe the influence of the anonymization with Tor, all measurements are performed with and without routing the network traffic over the Tor proxy.

In the evaluation we examine the two main protocols: issue credential and show proof (depicted in the use case diagram 3.1.1). We select a few key parameters to focus on during the measurement of the performance in our scenario. By doing so, we could determine the most crucial factors influencing the runtime of the protocols. Since previous works (e.g. [Verslype et al., 2008] or [Armac et al., 2009]) which evaluated the performance of Idemix identified the key length of the key used in the RSA\(^1\) modulo operations as the most crucial parameter, we also lie the focus on a variation of this and other system parameters. In addition to that we also look into the impact the anonymization network has on the overall

---

\(^{1}\)RSA is a asymmetric cryptography algorithm for encryption and also signing of data. One of the main steps of the encryption it the exponentiation of the data modulo n.
performance and will also include some other factors. The following subsections will provide detailed information about the parameters we changed and why we chose them.

8.2.1 Variation of Key Length
As it was pointed out in previous related works (see [Armac et al., 2009] and also [Verslype et al., 2008]), one of the most important variables, concerning the performance is the length of the key used for the modulo operation in the RSA algorithm. To change this parameter we have to change it in the SystemParameters file. Because of the fact that the parameters in this file are used in the whole library for many different key generations, we need to update several other files, too in order to uphold the compatibility. For example do we have to generate new group parameters since they are being generated using the parameters defined in the SystemParameters. As a consequence, the issuer needs to generate a new key pair too, because his/her keys depend on the group parameters. This all leads to the generation of a bunch of files, only because of a change of one parameter in the SystemParameters.

8.2.2 Variation of Supported Attributes
The maximum number of attributes that can be used for the issuance of a credential is determined by the IssuerPublicKey. In the key file there is for each attribute a different base key. Each of this keys can only be used once, so the number of keys is the maximal number of supported attributes. We varied this maximal supported number by generating different issuer keys and looked into the influence on the Issuance protocol performance.

8.2.3 Variation of Attributes Issued in a Credential
Besides the number of supported attributes we also changed the number of actually used attributes. Therefore we performed 5 different tests, with 3, 10, 30, 50 and 100 attributes. To change the attribute type is not necessary, because in the calculations they are all of the type BigInteger and this would make not difference.

8.2.4 Variation of Predicates
In the early evaluation and research of related work (see e.g. [Verslype et al., 2008]), we noticed that the different kinds of predicates used in the proof generation have significantly different impact on the computation time. So we tested a representative of each kind separately, with variation of the other parameters. This allows us to differ more detailed about the performance of different use-cases.
8.2.5 Variation of Devices

All tests were performed on a Samsung Galaxy S i9000 smartphone. It has a 1GHz Hummingbird processor, 512 MB RAM and is running Android 2.1. The up-to-date hardware (comparable to the hardware in the iPhone 4) makes it perfectly suitable for our purposes. In the beginning of November, Samsung published an update of the operating system of the Samsung Galaxy S i9000 to Android version 2.2. Because of this, we were able to also examine the difference of the Android version on the same device. Additionally to the real devices we tested the application on an Android emulator, running on the MacBook Pro 4,1 with MacOS X v10.6.

8.2.6 Variation of Communication Channels

In relation to the computation times, the time spend with communication over the Internet or a local network is negligibly small. So we did not look into differences in the performance of XML-RPC and XMPP. Both are XML based (which means messages in both protocols need some time to be parsed) and have an average latency of under 0.02 sec. In the context of communication channels we only focused to look into the effect of the anonymization through the Tor network and therefore performed two tests. In the first test, we only send an echo request to the server. The second test consists of two steps, the issuance and the proof protocol, which can also be performed independently. First we formulate a representative credential and then also a proof using this credential, using one of each predicate type.

8.3 Preparations

Since we are performing a lot of different measurements we need a way to do this automatically. Therefore we implemented a special GUI allowing us to start different testing routines. One of the most sophisticated tasks of the preparation of the measurements, was the variation of the parameters stored in the System-Parameters file (sp.xml). This is because of the design of the system that uses interconnected XML files for the definition of parameters and structural definitions of proofs and credentials. Like described before, when changing the SystemParameters, new GroupParameters and also issuer keys need to be created. In order to easily generate those files, we implemented a special routine on the server. By giving the SystemParameters as input, all dependent files are generated automatically. To support different configurations and to be able to switch easily between them, we store them in different folders, using the folder name as parameter. All that is now necessary in order to perform the measurements on a device, besides installing the prototypic application, is to copy the folder with the prepared XML files for the test to the SD-Card of the device.
8 Performance Evaluation and Results

8.4 Carrying Out the Measurement

In order to carry out the measurements, we copied the prepared Issuance and Proof specification files to the testing device. We prepared some special testing routines that can be started from the GUI. So we set up the application settings to perform the tests with a defined key length and start the routine. After this, each of the predefined Credential Issuances and Proof Computations is performed 50 times in a row, logging the response time for each single call to a file. After the testing routine finished, the files are extracted from the device in order to evaluate the measurements.

8.5 Discussion of Results

In order to evaluate the measurement results we gathered the values and created a synoptical table and also several diagrams illustrating the results graphically.

The table in Figure 8.5.1 shows the average response times of the issuance protocol and as well of the show proof protocol. The illustrated response times in this case include the whole time needed to perform the specific protocol. (The whole methods were also shown in detail in the previous chapter 7.2). As shown in table 8.5.1 the length of the key (modulo of the RSA algorithm) is very critical for computation time.

<table>
<thead>
<tr>
<th></th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>50</th>
<th>75</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>969.94</td>
<td>1021.34</td>
<td>1042.66</td>
<td>1091.9</td>
<td>1158.36</td>
<td>1276.82</td>
</tr>
<tr>
<td>1024</td>
<td>1122.28</td>
<td>1240.58</td>
<td>1323.54</td>
<td>1379.34</td>
<td>1465.3</td>
<td>1542.32</td>
</tr>
<tr>
<td>1536</td>
<td>1621.94</td>
<td>1709.76</td>
<td>1760.28</td>
<td>1801.78</td>
<td>1963.5</td>
<td>1954.12</td>
</tr>
<tr>
<td>2048</td>
<td>2278.24</td>
<td>2339.22</td>
<td>2403.52</td>
<td>2471.96</td>
<td>2595.94</td>
<td>2714.44</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Issue Credential</th>
<th>Show Proof</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1024</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1536</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2048</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 8.5.1: Average response times of the issuance and proof protocol

The gathered values are also used to generate several diagrams that illustrate the results more clearly and allow to interpret the results. The first diagram (8.5.2) depicts the increase of the computation time in relation to the number of issued attributes. Each line stands for a key length that is constant in the series of tests. It can also be seen that the computation time strongly depends on the length of the used keys. The variation of the attributes seems to have linear impact on the computation time. Furthermore the computation time seems to depend exponentially from the length of the key.

Diagram 8.5.3 depicts the same relation for the proof protocol. In this diagram we can see similar dependencies as in the previous example. An increase of proven attributes seems to have (almost linear, because of the small number
Figure 8.5.2: Response times of the issuance of measured points, random variances have more impact) linear influence on the computation time, while an increase of the key length has exponential impact.

To have a deeper into this dependency, Figure 8.5.4 illustrates the key length in relation to the computation time with the same measurement values. Each line has an constant number of proven attributes (10,50,75,100). As it can be seen, the graphs are performing a slight curve.

The next diagram depicts the results of a different series of tests we per-
formed. While the first series focused on the number of attributes in the issuance and proof protocol, the series evaluated in the following holds the number of attributes constant. The main goal of this test was to identify the influence of the different sub proof (e.g. additional inequality proofs, commitments, prime encoded attributes etc.) routines to the computation time of the whole proof. Therefore a representative proof was designed, containing one of each kind of predicate proofs. Diagram 8.5.5 illustrates the computation time of the whole proof protocol with this proof definition.

As we can see, the computation time is also increasing exponential if the key-length is increased. In order to identify the percental proportioning of the different sub proofs in the whole computation we used an embedded debugging routine of the Android SDK that allows to log all method calls during the exe-
8 Performance Evaluation and Results

cution of the application and store them to a file. An example of such a log file, (graphically represented by the tool TraceView of the Android SDK) is depicted in Figure 8.5.10. With the use of the TraceView tool, we were able to identify the dependency of each subroutine to the length of the used key. The results are shown in the table in Figure 8.5.6.

<table>
<thead>
<tr>
<th></th>
<th>512</th>
<th>1024</th>
<th>1536</th>
<th>2048</th>
</tr>
</thead>
<tbody>
<tr>
<td>parse Spec</td>
<td>35</td>
<td>36</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>load Cred</td>
<td>25</td>
<td>10</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>build Proof</td>
<td>14</td>
<td>7.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>verify proof</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>load key encrypt</td>
<td>5.5</td>
<td>4.5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>init Verifier</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3.5</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 8.5.6: Percentage computation times of the sub proofs

As we can see, for smaller keys the time used to parse and load the proof specification is very time consuming. For longer key sizes the time needed to compute the proof increases relatively strongly. To identify the reason for this, the table in Figure 8.5.7 shows the breakdown to the percental analysis of the proof computation alone.

<table>
<thead>
<tr>
<th></th>
<th>512</th>
<th>1024</th>
<th>1536</th>
<th>2048</th>
</tr>
</thead>
<tbody>
<tr>
<td>verEnc</td>
<td>53</td>
<td>34</td>
<td>21.5</td>
<td>13</td>
</tr>
<tr>
<td>Inequality</td>
<td>17.5</td>
<td>27</td>
<td>34</td>
<td>39</td>
</tr>
<tr>
<td>computeChal</td>
<td>6.6</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>primeEncode</td>
<td>8</td>
<td>13.5</td>
<td>17</td>
<td>21</td>
</tr>
<tr>
<td>pseudo</td>
<td>2.5</td>
<td>1.5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>domain</td>
<td>1.5</td>
<td>1</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>comm</td>
<td>1</td>
<td>2</td>
<td>3.5</td>
<td>4</td>
</tr>
<tr>
<td>repr</td>
<td>0.5</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>get random</td>
<td>1.5</td>
<td>1</td>
<td>0.5</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Figure 8.5.7: Percentage computation times of the build proof method

The shifting of the percental segmentation is illustrated by the following two pie diagrams, to make the influence of the key size more clear. Figure 8.9(a) shows the distribution with a key length of 512bit. The blue part represents the time to parse the proof specification. Only 10% are used for computing the proof. In Figure 8.9(b) 40% of the computation time is used to compute the proof and the parsing took also 40%.

\footnote{Because of the fact that the debug tool has only a limited cache size, and that this limit was reached before the whole protocol could be finished for the longer key sizes, there are only reliable results for the first two columns.}
The last point of the evaluation is the impact of Tor on the protocol performance. Since Tor is only affecting the network communication and not the computation of credentials or proofs. We can analyze this independently. The table in Figure 8.5.9 shows the results of the test to identify the influence of Tor on the performance. It can be seen that the optional use of the Tor network to disguise the network address is influencing the latency of transactions in a considerable way. In most cases the use of Tor anonymization averagely increases the latency of transactions for about 1.5 sec. But the values can vary in a large range, determined by the actual circuit the TOR client choses in the TOR network. We also measured latency over 20 seconds for a simple request.

<table>
<thead>
<tr>
<th></th>
<th>1024bit</th>
<th>2048bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>issue credential (7 Attr):</td>
<td>1,2</td>
<td>2,1</td>
</tr>
<tr>
<td>issue credential (30 Attr):</td>
<td>1,2</td>
<td>2,2</td>
</tr>
<tr>
<td>complete proof:</td>
<td>2,7</td>
<td>6,2</td>
</tr>
<tr>
<td>adult proof:</td>
<td>0,9</td>
<td>2,7</td>
</tr>
<tr>
<td>echo latency</td>
<td>0,01-0,1</td>
<td>0,028</td>
</tr>
<tr>
<td>- with tor</td>
<td>1,1 - 4</td>
<td>1,5</td>
</tr>
</tbody>
</table>

As we can see in figure 8.5.10 the computation of the proof, takes almost 50% of the time in the proof protocol. The reason for this is the fact that almost the whole time during the computation several modular exponentations are performed. This explains the big influence of the key length on the overall performance. While we experimented with several complex combinations of the different proof predicates we also figured out that the inequality predicates as well as added verifiable encrypted values strongly influence the computation time.
8 Performance Evaluation and Results

![calls of modular exponentiation](image)

**Figure 8.5.10:** Traceview showing visualising the method calls during the proof protocol

### 8.6 Summary

In this chapter we evaluated the performance of the Idemix protocols on an Android smartphone. We therefore defined several test cases, performed the measurements and presented the results. The results confirmed the statements of Armac et. al. [Armac et al., 2009] and Verslype et. al. [Verslype et al., 2008] that the computation time strongly depends on the key length. The influence of the number of issued attributes was also analyzed, and it was showed that the dependency to the computation time is nearly linear. Since the parsing of XML files is time consuming too, it is also an important factor of the overall computation time, especially if a relatively small key is used. Finally the analysis of Tor showed that in our use cases, the Tor anonymization increases each Roundtrip time for around 1-4 seconds, in the worst case (which is occurring randomly, when Tor nodes with small bandwidth are selected) even more.


9 Summary and Conclusion

The main objective of this thesis was to provide a prototypic implementation of an IdM module for smart-phones by using the IBM Identity Mixer called Idemix and to evaluate the performance of the Idemix protocols with this prototype. Also the influence of the anonymization of the network communication with Tor was analyzed in this context.

In order to achieve the objectives concrete requirements were gathered, derived from the Di.Me conference scenario and also from related work. Based on the requirements the prototypic Android application as well as the prototypic CA were designed and implemented. With the help of this prototypes it was then possible to perform many different measurements to identify crucial parameters for the performance of the Idemix protocols Issue Credential and Show Proof and as well analyze the impact of the use of the Tor anonymity network on the performance. So it was identified that the length of the key exponentially increases the computation time of a proof, as result of the numerous modular exponentiations being performed. Also the use of Tor anonymization has to be considered depending on the concrete use-case, since it leads to additional increased Roundtrip times. thereby is also very unstable concerning the performance due to the random selection of the circuit in the Tor network.

Because of the P2P setting of the evaluated scenario, this thesis had to deal with a worst case scenario for Idemix, which also goes beyond related work. Armac et. al describe in their work [Armac et al., 2009] that they were able to lessen the impact on the performance by using session keys when connected to their server. In the context of this thesis this is not possible.

Concluding, the main contribution of this work consist of (1) providing current performance evaluation results of a P2P setting when using Idemix on mobile devices, and (2) supporting thereby optional anonymity at the network level, as well as (3) providing some prototypes for an intuitive UI to maintain digital faces and to generate self defined proofs were provided. Thus, we used in contrast to related work, the latest Idemix implementation recently released and simulated an Idemix CA and involved it in the evaluation. The results showed us that the usage of Idemix in general strongly depends on usability and interaction design.

The developed prototype and as well the measured results build a good starting point for future work. Since Idemix is already available in form of a Java based open source library, it is possible to use this library on modern mobile devices supporting Java such as done in [Armac et al., 2009]. However, new trends
Summary and Conclusion

show that Android as well as iOS based platforms gain a wide acceptance. Current efforts concentrate on porting Idemix to C++ in order to cover a broader range of mobile platforms such as the mentioned iOS devices, Symbian OS, and the new QNX-based BlackBerry operating system. Conducted lab tests by using first GUIs also generated further requirements for future work concerned with conceptualization and design of intelligent UIs for the Di.Me userware. Like encouraged by the first GUI prototypes in this thesis it is also open to ease the use of Idemix for unexperienced users and also for developers with automation and optimization of a few processes. Besides improvements of the application the original source code has also much potential for enhancements. Armac et al. confirm in [Armac et al., 2009] that “the use of the Idemix implementation of 2009 is not as efficient as desired.” By contacting the idemix group, they have confirmed that “the performance of their reference implementation is not optimized yet and future versions are expected to run efficiently on mobile devices such as PDAs.”
10 Acknowledgements

I would like to thank Prof. Dr. Dogan Kesdogan, for giving me the opportunity to write this thesis in his department. I am very grateful to my supervisor Mohamed Bourimi for the care with which he reviewed this thesis and the countless remarks, suggestions and help he provided, even late at night. Also my thanks go to Dr. Thomas Barth, for taking the time to review the thesis and provide helpful suggestions.
Bibliography


[Bourimi et al., 2009] Bourimi, M., Kühnel, F., Haake, J. M., el Diehn I. Abou-
needs in real-identity collaborative systems. In CRIWG, pages 110–125.

aging visual tailoring and synchronous awareness in web-based collaborative 
systems. In Haake, J. M., Ochoa, S. F., and Cechich, A., editors, CRIWG, 
volume 4715 of Lecture Notes in Computer Science, pages 40–55. Springer.

[Bourimi et al., 2010b] Bourimi, M., Ueberschaer, B., Ganglbauer, E., Barth, T., 
Kesdogan, D., Dax, J., and Heupel, M. (2010b). Building usable and privacy-
preserving mobile collaborative applications for real-life communities: A case 
study based report. To appear in the IEEE iSociety Conference 2010 (IN 
PRESS).

vacy factors in video-based media spaces. In Harrision, S., editor, n Media 


digital certificates: building in privacy.

[Bullock and Benford, 1999] Bullock, A. and Benford, S. (1999). An access con-


