

eGovernment Services in a Mobile Environment

Gunn Olaussen
Kirsti Nordseth Torgersen

Master of Science in Computer Science
Submission date: June 2006
Supervisor: Torbjørn Skramstad, IDI
Co-supervisor: Lillian Røstad, IDI
Gunnar Nordseth, Kantega

Problem Description

The objectives of the diploma are:

- To describe relevant standards concerning the topic and provide an assessment and recommendation of which standards should be used.
- To survey and describe available technologies for strong authentication on a mobile phone. The survey should include the initiative for open authentication (www.openauthentication.org) which also develops reference architecture and specifications for strong authentication on mobile phones
- To survey and describe how a mobile phone can be supported in a Liberty-type Circle of Trust
- To describe how eGovernment Services can be made available in a mobile environment in a secure way, using Liberty ID-WSF
- To develop a proof-of-concept (PoC).

Requirements for the proof-of-concept:

- The PoC shall be a working prototype demonstrating the use of a mobile phone for accessing an eGovernment register service
- The PoC shall demonstrate the interaction between a Liberty-enabled Web Service Provider, a Liberty-enabled Web Service Consumer (the mobile phone) and a Liberty-enabled Identity Provider.
- The register service "Lånekassen" (State Education Loan Fund) will be used for demonstration purposes
- As the information is non-sensitive, there will be no need for strong authentication. However, the thesis shall also include a design of how strong authentication can be supported by the mobile phone. The recommended solution shall be described at a high architectural level.
- If the proof-of-concept cannot be implemented due to lack of time or for other reasons, the thesis should include a technical design which shows how a proof-of-concept could be implemented
- If there are technical considerations which make it impossible to implement the proof-of-concept, these considerations should be documented and discussed in the thesis

Assignment given: 18. January 2006

Supervisor: Torbjørn Skramstad, IDI

Summary

This report was written as part of our thesis based on an assignment provided by Kantega. It deals with the use of mobile phones to access eGovernment services using the Liberty Identity Web Services Framework (ID-WSF). Additionally, it explores the use of strong authentication mechanisms on mobile phones while using the phone as the terminal to access such services.

The first part of the report describes the project and its goals. In this process, it defines three research questions that will be answered in the course of the thesis. Furthermore, it outlines how these questions should be answered. This part also includes a presentation of the prototype that was developed later in the project.

The second part of the report concentrates on the theoretical basis of the assignment. Existing standards and technologies for strong authentication and Liberty-enabled products are explored and evaluated. The focus of the evaluation is upon whether the technologies could be used in the prototype.

The third part of the report contains the requirements specification, design, implementation and testing documentation for the prototype. This part aims to describe all aspects of the prototype development and enables us to show that it is a valid proof-of-concept. Requirements and design incorporating strong authentication into the prototype are also provided, although this functionality was not implemented as specified in the assignment.

The last part of the report evaluates the results obtained in the course of the thesis and especially the resulting prototype. The prototype fulfills our requirements well, but there were some reservations on the security in the strong authentication design. This part also looks at what can be done in the future to further explore the topic and improve the results. Finally, it shows how the report has answered the research questions we defined in the beginning of the thesis by completing a prototype that accesses eGovernment services using Liberty ID-WSF.

Preface

This report is the master thesis of Gunn Olaussen and Kirsti N. Torgersen at the Department of Computer and Information Science (IDI) at the Norwegian University of Science and Technology (NTNU) in the spring of 2006. The thesis was written based on an assignment given by Kantega.

We would like to thank our main supervisor, Gunnar Nordseth at Kantega, for creating such an inspiring assignment. He has together with Svein Otto Solem and Jens Erik Torgersen from Kantega given us excellent feedback and assistance on our work. Kurt Näslund from Kantega has also been very helpful and we would like to thank him for the help with providing us access to a server and installing several of the needed components for us. Finally, we would like to thank our supervisor at IDI, Lillian Røstad, for guidance with the requirements of academic writing.

Trondheim, 14th June 2006

Gunn Olaussen

Kirsti N. Torgersen

Contents

I	Introduction	1
1	Report content	3
1.1	Introduction	3
1.2	Definitions	3
1.3	Report outline	5
2	Project description	7
2.1	Introduction	7
2.2	Research questions	7
2.3	Approach	8
2.4	Prerequisites	10
2.5	The prototype	11
II	Preliminary study	13
3	Background and technology	15
3.1	Introduction	15
3.2	Abstract security concepts	15
3.3	SOA - Standards and technology	18
3.3.1	XML	20
3.3.2	SOAP	20
3.3.3	WSDL	21
3.3.4	SAML	21
3.3.5	X.509 digital certificate	22
3.3.6	XML-Signature and XML-Encryption	22
3.3.7	HTTP	23
3.3.8	Secure transport protocols	23
3.4	Mobile phone technology	25
3.4.1	SIM	25
3.4.2	J2ME	25
4	eGovernment services	29
4.1	Introduction	29
4.2	Today's situation	29
4.3	Service types	30
4.4	Regulations	31
5	Strong authentication on mobile phones	33
5.1	Introduction	33
5.2	Definition	33
5.3	Authentication methods	35
5.4	Authentication tokens	36
5.5	Strong authentication standards	39

5.5.1	ID-SAFE	39
5.5.2	OATH	39
5.6	Existing products	44
5.7	Evaluation	47
5.7.1	Main goals	47
5.7.2	Choice of method	48
6	Liberty ID-WSF on mobile phones	49
6.1	Introduction	49
6.2	Liberty ID-WSF	49
6.2.1	Liberty ID-WSF SOAP binding	49
6.2.2	Mobile phone in a Liberty Circle of Trust	50
6.3	Related work	56
III	Prototype	59
7	Requirements specification	61
7.1	Introduction	61
7.2	Overall description	63
7.3	Specific requirements	66
7.3.1	Functional requirements	66
7.3.2	Non-functional requirements	70
7.3.3	Requirements for strong authentication	71
8	Design	75
8.1	Introduction	75
8.2	Design issues	75
8.3	Mobile application	77
8.3.1	System description	77
8.3.1.1	State diagram	77
8.3.1.2	Package diagram	78
8.3.1.3	Class diagram	79
8.3.1.4	Sequence diagram	83
8.3.2	Major design entities	86
8.3.2.1	XML Schema	86
8.3.2.2	SOAP client	88
8.3.2.3	Liberty messages	89
8.3.2.4	GUI	89
8.4	Identity provider	95
8.4.1	System description	95
8.4.2	Major design entities	98
8.5	Web service provider	99
8.5.1	System description	99
8.5.2	Major design entities	101
8.6	Strong authentication	102
8.6.1	New and altered design issues	102
8.6.2	New design	106

9	Implementation	109
9.1	Introduction	109
9.2	Implementation challenges	109
9.3	Changes from design	113
9.4	Code samples	119
10	Testing	123
10.1	Introduction	123
10.2	Testing strategy	123
10.3	Tests specifications	127
10.3.1	Check lists	127
10.3.2	System test	129
10.4	Test results	137
10.4.1	Check lists	137
10.4.2	System test	140
10.5	Summary	143
IV	Evaluation and Conclusion	145
11	Evaluation	147
11.1	Introduction	147
11.2	Evaluation criteria	147
11.3	The prototype	148
11.4	Strong authentication design	152
11.5	Summary	154
12	Further work	155
12.1	Extending the prototype	155
12.2	Deploying the application	156
13	Conclusion	157
	References	159
V	Appendices	165
A	Assignment text	167
B	Notation	169
B.1	State diagram	169
B.2	Package diagram	170
B.3	Class diagram	170
B.4	Sequence diagram	172

C XML Schemas	173
C.1 Mobile Request	173
C.2 WSP Response	173
C.3 WSP fault	174
D WSDL	175
E Screen texts	179
F Test and requirement connection	183
G Templates	185
H Results of first system test	187
I Abbreviations	189

List of Figures

3.1	Liberty's underlying standards and technology	19
3.2	An example of SOAP and XML	20
3.3	The protocols of SSL together with HTTP	24
3.4	The elements of J2ME for mobile phones	26
5.1	OATH authentication architecture	41
6.1	The message exchange of a mobile phone accessing a CoT	51
6.2	Decomposition of step 1 in the mobile's message exchange	52
7.1	System overview	63
8.1	State diagram for the prototype	78
8.2	Package diagram for the prototype	79
8.3	Class diagram for the prototype (controller package)	81
8.4	Class diagram for the prototype (communication package)	82
8.5	Sequence diagram for the prototype (part 1)	84
8.6	Sequence diagram for the prototype (part 2)	85
8.7	Class diagram for the authentication service	96
8.8	Class diagram for the discovery service	97
8.9	Class diagram for the WSP	100
8.10	State diagram for strong authentication	106
9.1	Java MIDP 2.0 performance	111
9.2	Updated class diagram (controller package)	115
9.3	Updated class diagram (communication package)	118

List of Tables

6.1	Summary of Liberty conformance	57
10.1	Requirements that will not be adequately tested by the system test . . .	125
10.2	Check list for the controller package	127
10.3	Check list for the communication package	128
10.4	Check list for the authentication package	128
10.5	Check list for the discovery package	128
10.6	Check list for the loanfund package	129
10.7	ST1 Normal execution	130
10.8	ST2 Test of error handling	131
10.9	ST3 Test of help and user abort during login	132
10.10	ST4 Test of WSP	133
10.11	ST5 Test of log-in and user abort during DS query	134
10.12	ST6 WSP error message	135
10.13	ST7 Test of message format	136
10.14	Check list result for the controller package	137
10.15	Check list result for the communication package	138
10.16	Check list result for the authentication package	138
10.17	Check list result for the discovery package	139
10.18	Check list result for the loanfund package	139
10.19	Test result for ST1	140
10.20	Test result for ST2	140
10.21	Test result for ST3	141
10.22	Test result for ST4	141
10.23	Test result for ST5	141
10.24	Test result for ST6	142
10.25	Test result for ST7	142
B.1	Notation for the state diagram	169
B.2	Notation for the package diagram	170
B.3	Notation for the class diagram	171
B.4	Notation for the sequence diagram	172
E.1	Label texts	179
E.2	Service texts and messages	180
E.3	Error messages from the WSP	180
E.4	Error messages from the application	181
E.5	Help screen text	182
F.1	Test and requirement connection	183
G.1	Check list template	185
G.2	Test specification template	185
G.3	Test result template	185
H.1	Test result for ST1 - first attempt	187
H.2	Test result for ST2 - first attempt	187
H.3	Test result for ST3 - first attempt	188
I.1	Abbreviations A-I	189
I.2	Abbreviations J-W	190

Part I

Introduction

1 Report content

1.1 Introduction

This report will try to answer the questions stated in the assignment text for the thesis. This text is included in appendix A. The report will deal with the topic of Liberty-enabled mobile phones which are trying to enter a Liberty Circle of Trust where they can access eGovernment services. To do this, the assignment states that the topics of mobiles, Liberty and strong authentication shall be surveyed and that a proof-of-concept shall be implemented.

This chapter introduces the concepts that are used in the assignment text and start of the report. That section will only serve as an introduction to those unfamiliar with the topic. The chapter also provides information about how the report is structured by identifying the main parts and chapters.

1.2 Definitions

In this section we describe some concepts that are important and commonly used when discussing the topic of this report. This is in no way a complete definition of all the concepts, but most of the items below will thoroughly defined when we address them later.

Circle of Trust (CoT) is a Liberty concept that according to their definition is "a federation of service providers and identity providers that have business relationships based on Liberty architecture and operational agreements and with whom users can transact business in a secure and apparently seamless environment.", *Liberty Technical Glossary* [1] Once the user has been identified by one of the identity providers in the CoT he can be recognised and participate in the CoT.

eGovernment is used to describe services offered by the government through electronic media like the Internet and mobiles, in addition to the traditional public administration offices.

Java 2 Platform, Micro Edition (J2ME) is a programming language used for developing applications for use on mobile phones.

Kantega is one of Norway's leading consulting firms on the topic of digital identity and electronic security.

Liberty Alliance is a consortium which is developing a set of specifications that form an open standard for federated network identity.

Liberty ID-WSF is the Liberty Identity Web Services Framework specification. It contains a set of specifications that build upon the ID-FF specifications for federated network identity. We will be using only ID-WSF to provide identity-based web services. This means that web services can be discovered and offered based on a user's identity.

"MinSide"¹ is a part of the Norwegian eGovernment initiative. It offers services to the citizens through a web portal.

Non-sensitive We use this concept to describe information or services that will not require strong authentication. This definition was chosen because it is used in the assignment text, but the information may still be considered sensitive by some public entities.

Proof-of-concept (PoC) is used to describe the prototype we are going to make, which will show that it is possible to implement the concept of accessing a Liberty Circle of Trust with a mobile phone.

Register service is a variety of an eGovernment service offered in Norway where information about a citizen is provided by a web service provider that does not offer its own user interface.

Service provider is an entity that offers electronic services through the Internet.

Service-oriented architecture (SOA) is a scheme where services are offered by providers that are loosely connected. This means that the entities can communicate across platforms and without strong dependencies on each other.

Strong authentication is when the user can provide at least two factors for proving his identity. This is often something he has and something he knows.

Web service is usually a service which is accessible by the web and can be found in registers, where the necessary information to connect is provided in the service offering.

¹English: MyPage

1.3 Report outline

The report consists of four major parts and a fifth which is a collection of all the attachments. Each chapter of the report is briefly described below.

Part I: Introduction

1. **Report content:** This chapter introduces the topic of the report and defines important concepts.
2. **Project description:** This chapter describes the goal, approach and prerequisites of the project. It also introduces the prototype which will be developed as part of the project.

Part II: Preliminary study

3. **Background and technology:** This is a descriptive chapter of the underlying technologies that some of the work will be based on.
4. **eGovernment services:** This chapter introduces eGovernment around the world and especially in Norway. The Norwegian initiatives are described along with some regulations which must be considered in the development.
5. **Strong authentication on mobile phones:** This chapter presents strong authentication in general and how it works on mobiles. It also contains a survey of existing products.
6. **Liberty ID-WSF on mobile phones:** This chapter explains the communication when using a Liberty-enabled mobile. It also explores existing Liberty products.

Part III: Development

7. **Requirement specification:** This chapter holds all the requirements for the prototype with and without strong authentication.
8. **Design:** This chapter contains the design of the mobile application, supporting systems and the design of the application with strong authentication.
9. **Implementation:** This chapter is used to document the work of implementing the prototype and the changes to the design.
10. **Testing:** This chapter contains test strategy, specifications and results.

Part IV: Evaluation and conclusion

- 11. *Evaluation:*** This chapter consists of the evaluation criteria, prototype evaluation and evaluation of the strong authentication design.
- 12. *Further work:*** This chapter covers our ideas for extensions to the prototype and potential deployment.
- 13. *Conclusion:*** This chapter summarises how well we have answered the research questions.

Part V: Appendices

2 Project description

2.1 Introduction

This chapter describes the foundation of the project and expresses what we want to achieve. Specifically, we want to state the objective, the approach we are taking to achieve this goal and the prerequisites that we base our work on. There is also a section describing the prototype we are going to implement to prove the concept of a mobile phone connecting to a Liberty Circle of Trust.

2.2 Research questions

The purpose of this thesis is to find and discuss available technologies for using Liberty ID-WSF on a mobile phone as well as to survey how strong authentication is possible in such a setting. To find out how and if this is possible we have formulated three research questions, which are quoted below. These are based on our interpretation of the assignment text and state what we think is important to answer in the thesis.

1. How can a mobile phone be supported in a Circle of Trust using Liberty ID-WSF?
2. How can eGovernment services be made available in a secure way using Liberty ID-WSF?
3. How developed are standards/technologies for strong authentication on mobile phones?

The first two questions will lead to the design and implementation of a prototype. Together with the discussions prior to the design, the evaluation of the prototype will help answer these questions. The last research question implies that a survey of existing standards, technologies and products must be undertaken. Note that we concentrate on authentication on the mobile phone while accessing some protected resource from the phone. In other words, we will not use the phone as a tool to authenticate while accessing the resource from a computer.

2.3 Approach

To answer the research questions, the work process will include three main parts. These are numbered according to their position in the report and below we will describe the main aspects that will be considered.

II PRELIMINARY STUDY

The objective of this part is to gain a thorough understanding of what standards and technology exist, how these are used and which solutions are available. All this should provide us with the necessary basis to make the correct choices when designing and implementing the system.

Important aspects which should be included are:

- Exploration of the challenges that must be overcome before we can use Liberty to access an eGovernment service.
- Complete a survey of existing frameworks that are Liberty compliant: How do they resolve the challenges and can any of these be used in our assignment? We must take into consideration that we probably will not be allowed to access all information about the frameworks. Can we adopt parts or ideas in our solution?
- Find a valid definition of strong authentication that complies with what is required in eGovernment services.
- Exploration of any existing standards/technology that can be used in the development of the prototype. Can we use any of the standards as they are, improve them or should we start from scratch to implement this functionality in our application?

III DEVELOPMENT

This phase is the actual system development and we will go through the stages of system analysis (to elicit requirements for the system), system design, coding and testing to ensure that the system is of a good quality. We also need to find one or more standards for documenting the results.

Important aspects which should be included are:

- Determine criteria after which the prototype should be evaluated.
- Find out how to structure the system - how many components should we have?
- Determine system requirements for the prototype with and without strong authentication
- Design the system with and without strong authentication.

- Create test cases and testing routines.
- Implementation of the prototype; demonstrating use of a mobile phone connecting to a non-sensitive eGovernment web service. Strong authentication shall not be implemented.
- Document the work as we implement and update the design when the prototype is finished.

What is also important to note is what to do if, for some reason, the proof-of-concept cannot be made. According to the thesis assignment text, the following should happen if there are unexpected difficulties:

- If the proof-of-concept cannot be implemented due to lack of time or for other reasons, the thesis should include a technical design which shows how a proof-of-concept could be implemented.
- If there are technical considerations which make it impossible to implement the proof-of-concept, these considerations should be documented and discussed in the thesis.

IV EVALUATION AND CONCLUSION

This part includes an assessment of the development, which means to examine the criteria, research questions and requirements we made in the earlier phases. These will allow us to assess the quality of the result and from that make suggestions on further development or use.

Important aspects which should be included are:

- Evaluation of the prototype.
- Evaluation of the strong authentication design. Did we have to change anything during the implementation of the prototype? Do we still think it is possible to implement strong authentication by following our design? If not, what will have to be changed?
- Further work. What possibilities exist for improvement?
- Conclusion. Did we manage to answer the questions raised at the start of the project?

2.4 Prerequisites

This section describes what components or resources we will need throughout the project. Furthermore it gives a short description of each item. A more thorough description will be provided in the following chapters.

LIBERTY

In our prototype we will only use the Liberty Identity Web Services Framework (ID-WSF). We will need entities that are able to act as one or more of the following roles as they are specified in the *Liberty Technical Glossary* [1]:

Identity Provider (IdP): The identity provider is the entity that authenticates the user and asserts his identity to the service providers. The service in charge of the authentication is called the Authentication Service (AS), and the result of a successful authentication with the AS is usually a reference and token to the discovery service.

Discovery Service (DS): The DS is a directory where service providers may register their services and service consumers may search for the services they need. It may be implemented as part of the IdP.

LUAD-WSC: In our case, the mobile phone will be a LUAD-WSC. This means that it is a LUAD that does the job of a WSC. Below we explain more precisely what being a LUAD and being a WSC means.

Liberty-enabled User Agent or Device (LUAD): A LUAD is a user agent or device that specifically complies to at least one of the profiles in Liberty. To be a LUAD, an entity must be more sophisticated than a common browser. It has to be able to send and receive messages that are constructed according to Liberty's specifications, and it has to understand the contents of these messages and act according to their specifications. To indicate that they are LUADs, such entities are denoted by adding the prefix "LUAD-" to their abbreviations.

Web Service Consumer (WSC): This is an entity that requests a web service according to the rules specified in Liberty ID-WSF. Our mobile will be a WSC, because it will use Liberty ID-WSF to request the services offered through the eGovernment system.

Web Service Provider (WSP): This is an entity that provides a web service according to the rules specified in Liberty ID-WSF.

MOBILE PHONE WITH J2ME

One of the demands of this assignment is to develop a prototype that may connect to a service offered on the Internet. Since most mobile phones today support J2ME, we will use this programming language to develop the application. Furthermore, the mobile phone will need to be able to connect to the Internet. However, this is not a problem as most mobiles today come with this ability, given that the necessary subsystem is provided by the mobile network operator.

The newest version of J2ME for mobile phones is MIDP 2.0. It includes support for secure network protocols, which was only optional in MIDP 1.0. Furthermore, as it is the newest version, most phones that are produced today support it. For these reasons we will base our implementation on the use of MIDP 2.0.

2.5 The prototype

This project will include the development of a prototype. The prototype will be a small application that can be run on mobile phones and provide access to eGovernment services. Potentially, the application could be made to access any such service, but we have chosen a set of services offered by the Norwegian State Educational Loan Fund (NSELF). The services are:

- Information about debt, last payment and next instalment
- Application status for grants and loans

The application should be able to authenticate a person that enters his username and password. It should then retrieve the service offerings for this person and present the names of the providers to the user. We will only make one service provider so that any others will only be dummy services². The user will then be able to chose the name of our provider and the application will request services. The provider will return all the services for this user and the application will present the list. The user will then be able to look at one or more of these without more communication with the provider.

As described in the scenario above, we will employ username/password authentication in the prototype since the information is non-sensitive. This means that the strong authentication design will not be implemented. It is however of great importance that the prototype is secure, since the content of the eGovernment services require identification by providing personal information. Additionally, the usability of the application is essential. The services that are offered are also available by using a computer to access the Internet, which is cheaper and faster. Therefore, the main motivation for using the application is the convenience of being able to access the services from anywhere.

²A dummy service will just contain the name of a provider and a non-existing address. If the user tries to choose one of these he should be told that the address does not exist.

Part II

Preliminary study

3 Background and technology

3.1 Introduction

In the following sections, the basics of the main technologies and concepts are explored. Some of these were introduced in chapter 1, but will be given a more thorough description here. This entire chapter serves as an introduction for readers that are not familiar with information security, service-oriented architecture and mobile development.

3.2 Abstract security concepts

In this section we provide some definitions and explanations of important concepts that we will use to describe and discuss topics throughout the report. To clarify the use of the following concepts, which are used in many different settings, we will provide one common definition for each and explain how we interpret and use these.

AUTHENTICATION

”Authentication is the process of confirming an asserted identity with a specified, or understood, level of confidence”, *Trust in Cyberspace* [2]. This means that authentication is to check that somebody or something is who or what it claims, with a specific amount of certainty that is dependent on amongst other things the method of confirmation. If the identity can be confirmed with a high level of confidence it is called strong authentication.

AUTHORISATION

Network Security: The Complete Reference [3] states that ”authentication establishes who the user is; authorization specifies what the user can do” In other words, to authorise an entity to access a resource means to confirm that the entity complies with the rules for accessing that resource. This usually happens after authentication so that the authority responsible for authorisation may use the identity information acquired to decide whether to grant access or not.

ENCRYPTION

”Encryption is any procedure used in cryptography to convert plaintext into ciphertext to prevent anyone but the intended recipient from reading the data.”, *Trust in Cyberspace* [2] In other words, to encrypt something electronically is to obscure it in a way that enables the rightful recipients to read it, but prevents unauthorised entities

to access the contents. To achieve this, an encryption algorithm must be used together with a key. To read the contents the recipient must have a matching algorithm and key. The scheme should not depend on the algorithm being secret, because it would be too costly to develop for each pair of communicating entities. Therefore, the keys are the only parts that have to be protected. The encryption and decryption can be done in two ways; symmetric or asymmetric. Both these options will be discussed below.

SYMMETRIC AND ASYMMETRIC CRYPTOGRAPHY

As explained in *Network Security Essentials - Applications and Standards* [4] there are several ways of encrypting and signing data. In the end, they all fall into one of two categories; symmetric or asymmetric. In symmetric cryptography both the sender and the recipient share a common key which is known only to them. The problem with this scheme is apparent when we consider that the secret key has to be made known to both parties, but to no one else. If they both operate in the same geographically confined environment one could imagine that the key may be delivered manually, but on a scale suitable for electronic communication, this is a very inefficient method.

To counter the problem of delivering the secret key to the other party, asymmetric cryptography has been invented. This kind of cryptography is also called public key cryptography. The basic concept is that each entity has a private and a public key. The private key is secret and known only to the entity to which it belongs, but the public key can and should be distributed widely amongst the entity's communication partners. The algorithms and keys are chosen in a way that enables messages to an entity to be encrypted by use of this entity's public key and decrypted by its private key. This way anybody can send confidential messages to the entity, but only the correct recipient can read them.

DIGITAL SIGNATURES

According to *Network Security Essentials - Applications and Standards* [4] digital signatures are "data appended to, or a cryptographic transformation of, a data unit to prove the source and integrity of the data unit and protect against forgery (e.g., by the recipient)". This means that, contrary to conventional signatures³, digital signatures are closely connected to the data unit they are applied to. With a digital signature it is possible to verify that it has not been moved to another document and that the data unit it signs has not been changed in any way. This would not be possible with a conventional signature.

If public key cryptography is in use, an entity can sign a data unit with its private key and any other entity with access to the signer's public key can verify the signature. However, if symmetric cryptography is used instead, only the entities that know the shared secret will be able to verify the signature, but these will also be able to create identical signatures. Thus digital signatures as replacements for conventional signatures will only make sense if public key cryptography is used. Because of this, digital signa-

³We use conventional signature to describe the act of signing your name on a piece of paper.

tures based on symmetric cryptography are usually called Message Authentication Code (MAC) instead. This is explained in *Building Secure Software: How to Avoid Security Problems the Right Way* [5].

Before the signature can be calculated from the data unit, a message digest has to be created. In *Building Secure Software: How to Avoid Security Problems the Right Way* message digests are defined as the output of a hash function. Hash functions are explained as one-way functions, which means that it is impossible, or at least infeasible, to calculate the input from the output.

CREDENTIALS

”A set of Credentials is information used to verify the identity of a user”, *Definition of Credentials* [6]. Often the credentials consist of a username and password. However, they may take other forms depending on the strength of the authentication. Examples of other possible credentials are personal identification number (PIN), username and one-time password (OTP), and digital certificate together with a signature made by the corresponding secret key.

PIN is the most common type of credential used for mobile phones and credit cards. It consists of a short sequence of numbers intended to be easy to learn by heart. When it comes to OTP, the passwords are different for each authentication. Each time the entity authenticates, a new password is calculated or chosen from a predefined set of passwords. Therefore, each OTP can only be used once, and the guessing or capturing of a password would usually not result in much damage.

INTEGRITY

Building Secure Software: How to Avoid Security Problems the Right Way [5] argues that ”when used in a security context, integrity refers to staying the same”. In other words electronic integrity is the property possessed by data that has not been altered or deleted on its way between two communicating parties. As explained in *It’s All About Authentication* [7] integrity has a strong connection to authorisation, since the main threat to electronic integrity is unauthorised entities being able to perform operations on the data. On a lower level, faulty communication links may affect integrity by causing the data to be distorted or lost, but this should be taken care of by the communication protocols.

IDENTITY FEDERATION

Identity federation is the act of creating ”associations between a given system entity’s identifiers or accounts”, *Liberty Technical Glossary* [1]. This means that if an entity is known by two or more different identifiers at different nodes in the system, the identities may be connected in a way that allows the entity to be known uniquely throughout the system.

TRUST

Network Security: The Complete Reference [3] defines trustworthy as "having reliable, appropriate, and validated levels of security". This means that if an entity trusts another entity, it is convinced that the other entity has these qualities. Consequently, it will assume that any claims that can be verified to originate from that entity are valid.

SINGLE SIGN-ON (SSO)

"Single sign-on encompasses the capability to authenticate with some system entity - in the Liberty context, an Identity Provider - and have that authentication honoured by other system entities, termed Service Providers in the Liberty context", *Liberty Technical Glossary* [1]. Thus, SSO allows an entity to only authenticate once and then be allowed access to several different nodes in a network of other entities.

One of the most important aspects of SSO is trust. In order to let the entity access other nodes after authenticating to one of the nodes, it is necessary for the nodes to be able to pass the authentication information between them. To do this they need to trust the entity that performed the authentication or else the information passed to them would be worthless.

3.3 SOA - Standards and technology

The article *webservices.xml.com: What is Service-Oriented Architecture* [8] defines service-oriented architecture (SOA) as "an architectural style whose goal is to achieve loose coupling among interacting software agents". The "loose couplings" mean that although two different systems may rely on each other for services, they should not depend on being built on compatible platforms or programming languages. In SOA, the inner workings of each software entity is its own business as long as it can offer or receive the services through an interface that is compatible with the rest of the SOA.

There are several standards and technologies that are suitable for use in SOA. These include both low-level ones that are developed to perform simple tasks and higher-level ones that can use the low-level standards and technologies to provide a wide range of complex functionality. One of these higher-level standards is Liberty. Other major standards are the family of web service related standards called WS-* and SAML.

Figure 3.1 shows the standards and technologies that Liberty ID-WSF depend on and how these relate to each other. Since using Liberty is one of our prerequisites, we will not describe the other higher-level standards. However, SAML will be briefly explained as Liberty is based on parts of SAML. Liberty will be described in chapter 6, but to provide a foundation for a better understanding of Liberty we explain the underlying technology in the following sections.

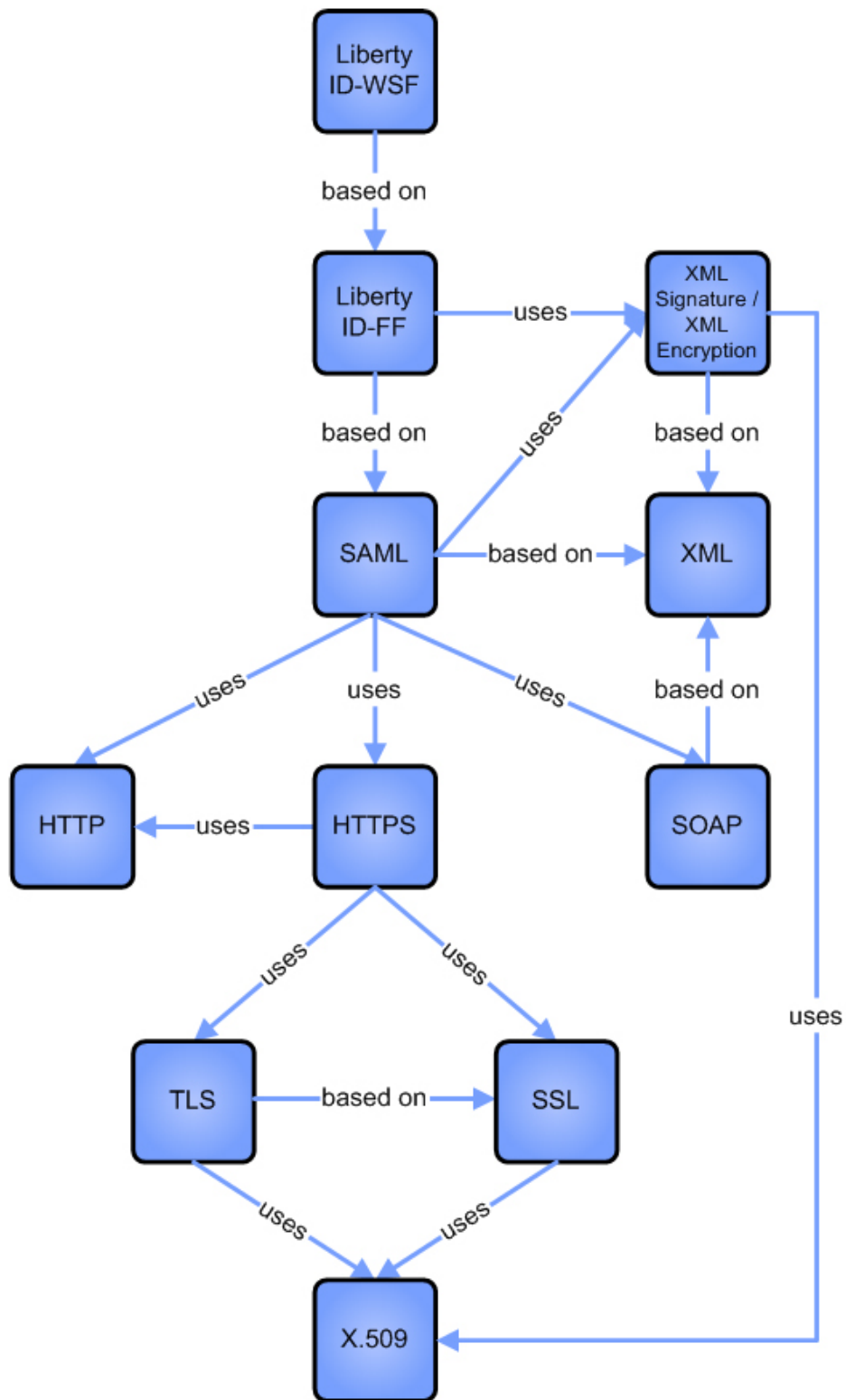


Figure 3.1: Liberty's underlying standards and technology

3.3.1 XML

The Extensible Markup Language (XML) [9] is managed by the World Wide Web Consortium (W3C) and is a language that can be supported on any platform. The reason for this is that it is text-based; leaving the interpretation and visual representation to the program that reads it. As shown in figure 3.2, an XML document is built up by enclosing the data with so-called tags that provide information about what the data represent. Data enclosed by tags is called an element. By choosing comprehensible tag names, the XML document can be made quite human-readable. Additionally, the elements may include attributes to further specify the information.

Because each programmer can construct his own XML documents, the need for a method of defining the structure emerges. This is needed both to help other entities understand it and help them create compatible XML documents. There are two common methods of doing this, Document Type Definition (DTD) and XML Schema. In this project we will only use XML Schemas, which are often called schemas for short. The schemas are XML documents and therefore they follow the rules of XML. Throughout the project we will use them to define the messages that flow between the entities in our prototype system.

```
<?xml version="1.0">
<env:Envelope xmlns:env="http://www.w3.org/2001/09/soap-envelope">
  <env:Body>
    <m:GetLastTradePrice xmlns:m="some-URI">
      <m:symbol>IBM</m:symbol>
    </m:GetLastTradePrice>
  </env:Body>
</env:Envelope>
```

Figure 3.2: An example of SOAP and XML
(adapted from [10])

3.3.2 SOAP

SOAP [11] is the most common language used for communication between web services. It is XML-based, which is one of the facts that makes it so suitable for this purpose. A SOAP message consists of a SOAP envelope with an optional SOAP header and a mandatory SOAP body inside. An example of such a message is shown in figure 3.2. The envelope encapsulates the message and will connect any headers to the body. The header is meant for including information that is not "application payload", *SOAP Version 1.2 Part 0: Primer* [12]. This could for instance be information about how the message should be handled. The body consists of the actual message to the application. It can contain any information that follows the rules of the XML standard.

3.3.3 WSDL

The Web Services Definition Language (WSDL) [13] is an XML-based structure that can describe the interface of web services. Each service is represented by a document containing descriptions of the messages sent to and from the service, together with information about where and how to contact the service. The standard uses XML Schemas to describe the messages.

It is possible to create abstract WSDL documents that are common for several services. In this case, the WSDL is made to describe the service type instead of the services themselves. Therefore it includes only the fields that are common for all the services. Typically the abstract WSDLs lack information about the URLs of the services. Consequently, each service needs to specify this information in some other way.

The WSDLs can help other entities make use of services they have never encountered before. By including them in a searchable registry, it would be possible for another entity to search for specific WSDLs to find the service it needs. However, it would also be possible to search for services by type, which in other words would mean to search for services that implement the same abstract WSDL.

3.3.4 SAML

Security Assertion Markup Language (SAML) is an open standard that is managed by the Organization for the Advancement of Structured Information Standards (OASIS). It is intended to give security and single sign-on capabilities to web services. SAML is also XML-based and may be used with SOAP messages.

The main aspects of SAML are described below. These explanations are based on the *Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0* [14]:

Assertions: An assertion is a claim made by an entity in the system. If the assertion recipient trusts the assertion issuer, it can assume that the claims contained within the assertion are true. There are three types of assertions, which are named after the type of information they carry. These are authentication, attribute and authorisation assertions.

Protocols: The SAML protocols define how the messages sent between entities of the system must be structured and what they can contain.

Bindings: The bindings of SAML contain information on how to link SAML messages to messages belonging to another protocol, so that they may be transmitted by use of this protocol. For instance, one could send SAML messages inside SOAP messages.

Profiles: Each profile contains a set of rules that define how to achieve a specific goal. For example the "SSO Profiles of SAML" specifies how to provide single sign-on.

Artifacts: SAML artifacts are used when the entire message is not to be sent using the primary communication link. For instance, this could be the case in situations where the message will be redirected via the user agent to the recipient. The artifact acts as a reference to the actual message, so that the recipient can contact the sender via another communication link and receive its message.

3.3.5 X.509 digital certificate

X.509 is a recommendation developed by the International Telecommunications Union (ITU). According to *Architecture and design for central authentication and authorization in an on-demand utility environment patent* [15], it is "the most common digital certificate format". Therefore, we will use only this standard when discussing digital certificates.

The intention of the X.509 certificate is to tie a public key to a subject, which is the owner of the certificate. The information in the certificate identifies the subject and is signed by a certificate authority (CA) that confirms the validity of the information. Alternatively, the certificate may be self-signed, meaning that the subject has signed the certificate himself. Either way, the certificate needs to be signed by somebody that is trusted by the recipient.

Without certificates the concept of public key cryptography would not be so useful. Then the connection between an entity and a public key could not be verified by anyone. This means that anybody that wishes to use an entity's public key must know in advance that the key really belongs to that entity. On a large scale this would be very inefficient.

The *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile* [16] describes how X.509 certificates can be used together with certificate revocation lists to provide better security. CRLs are a means of checking that the certificate presented by an entity has not been revoked. When a certificate should no longer be used, for instance if it has been stolen, it is added to the list. This will make it invalid, although it may not yet be past its expiration date.

The use of digital certificates and public key cryptography together is often referred to as Public Key Infrastructure (PKI). Authentication schemes based on these principles are usually denoted PKI-based. However, PKI is actually a quite wide term that includes most of the technology concerning this topic.

3.3.6 XML-Signature and XML-Encryption

XML-Signature [17] and XML-Encryption [18] are standards that define how to sign and encrypt data in a way that makes it possible to include it in an XML document. XML-Signature helps maintain integrity and authentication of the documents and the entity that signed them, while XML-Encryption is concerned with the confidentiality of

the documents. Both standards may be used together with several different algorithms and are therefore quite adaptable to the needs of each system.

3.3.7 HTTP

Hypertext Transfer Protocol (HTTP) is currently the most common protocol for accessing contents on the Internet. According to its specification, *Hypertext Transfer Protocol – HTTP/1.1* [19], HTTP is based on request/response messages being sent between communicating parties.

HTTP is basically a stateless protocol, meaning that it does not "remember" any information from one request to another. However, it is possible to add state management to the protocol, for example by using HTTP Cookies. These are small files that contain the information needed for the session.

3.3.8 Secure transport protocols

The Secure Sockets Layer (SSL) was originally developed by Netscape. Transport Layer Security (TLS) was based on version 3 of the SSL protocol and is managed by the Internet Engineering Task Force (IETF). SSL and TLS are very similar and, as argued in *Network Security Essentials - Applications and Standards* [4], TLS version 1.0 may be considered to be an SSL version 3.1. Consequently, at our level of detail the description of SSL may also apply to TLS.

Connections and sessions are important words in the SSL terminology. A connection is a communication link between a client and a server. Each connection is associated with a session that holds the security information negotiated between the two. This is done to avoid having to negotiate these anew with each connection.

Figure 3.3 shows the different protocols of SSL together with HTTP. An explanation to each of the protocols based on their presentation in *Network Security Essentials - Applications and Standards* [4] follows:

SSL Record Protocol: The main task of the SSL Record Protocol is to provide confidentiality and message integrity. This is done by both encrypting the messages and calculating a message authentication code (MAC) according to a specific algorithm. The MAC can then be used by the receiver to check that the message has not changed after the MAC was added.

SSL Handshake Protocol: The Handshake Protocol is used by the client and the server to authenticate each other and to negotiate a session between them. The server does not have to authenticate the client unless explicitly specified, but the client must authenticate the server.

SSL Change Cipher Spec Protocol: This protocol has only one defined message and is used for changing states.

SSL Alert Protocol: The Alert Protocol is designed for delivering information about SSL-associated warnings or errors to the other communicating party.

HTTP: HTTP was described in the previous section. The use of HTTP together with either TLS or SSL as depicted in the figure is commonly referred to as HTTPS. This is done to indicate that it is more secure than plain HTTP. Another difference from plain HTTP is that, through the session management of SSL, HTTPS is no longer a stateless protocol.

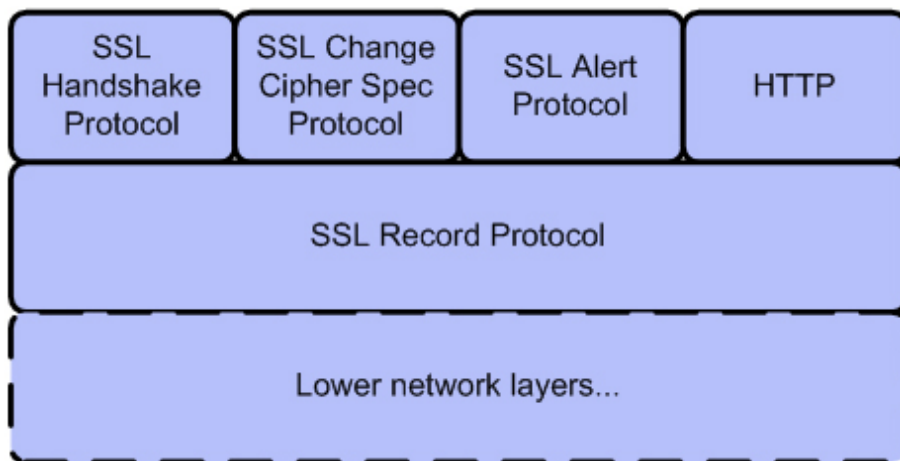


Figure 3.3: The protocols of SSL together with HTTP
(adapted from [4])

3.4 Mobile phone technology

This section introduces two important aspects of modern mobile phones which relate to application development on such devices. The Subscriber Identity Module will be used in the discussions regarding strong authentication. J2ME is the programming language we will use in the implementation and is presented here to introduce the possibilities and limitations of mobile programming.

3.4.1 SIM

According to *What is a Subscriber Identity Module (SIM)?* [20], the Subscriber Identity Module (SIM) is a type of smart card⁴ that contains the identity of the mobile network subscriber. It is needed to enable the mobile phone to connect to the network. Furthermore, it is the item that associates a specific telephone number with the phone. If the SIM is moved to another phone, the telephone number will be associated with the new phone instead.

Most SIM cards have the capability of performing cryptographic operations. This includes key generation and secure storage of private keys. The SIM cards' greatest advantage is that they are tamper-resistant⁵ and that access to them is restricted. In other words, the only way to access the content of the secure storage is through the interface provided by the phone and by entering a correct PIN. Even then, the private key will not leave the SIM as the cryptographic operations are performed directly on the card and only the results are returned.

3.4.2 J2ME

Java is one of the most popular programming languages today. The Java 2 Platform, Micro Edition (J2ME) programming language is a version of Java that is especially designed to work well on devices with limited capacities when it comes to performance, memory and storage. This makes it ideal for use with mobile phones.

The entities that comprise J2ME are shown in figure 3.4. These are configurations, profiles and optional profiles. The following sections will describe the entities that will be needed in the rest of the project. Note that, although there exist other versions of J2ME intended for other types of devices, we only describe the configuration of J2ME that is suitable for use on mobile phones.

⁴As explained in *Network Security: The Complete Reference* [3], a smart card is a plastic card containing a microchip. It is usually credit card sized, but can also be smaller. It can be put to various applications, including cryptographic functions.

⁵As defined in *Webster's encyclopedic unabridged dictionary for the English language* [21], to tamper is "to meddle, especially for the purpose of altering, damaging, or misusing". It also defined tamper-resistant as "difficult to tamper with".

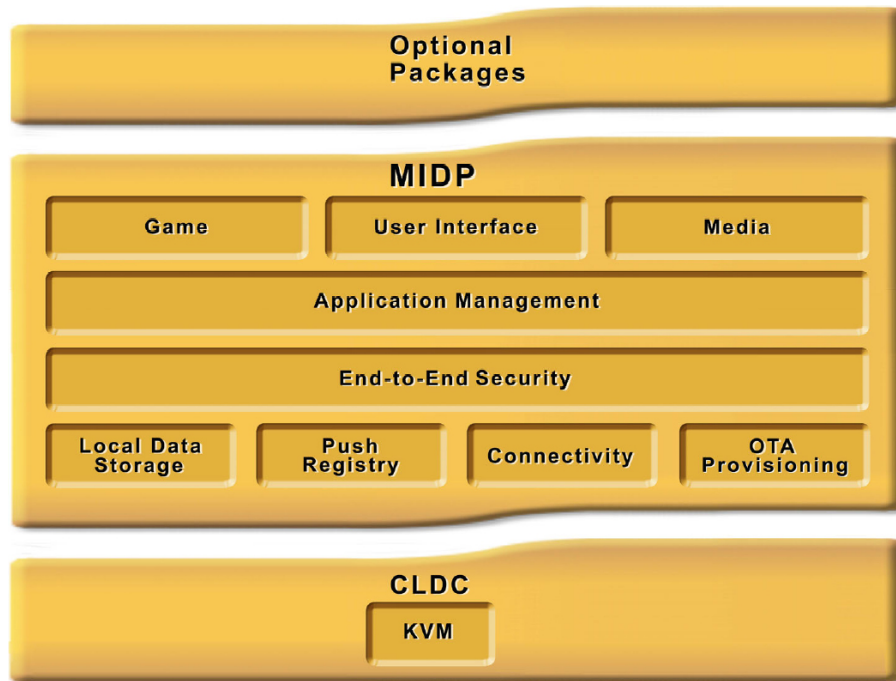


Figure 3.4: The elements of J2ME for mobile phones
(taken from [22])

CONNECTED LIMITED DEVICE CONFIGURATION (CLDC)

The CLDC contains a limited set of class libraries and the K Virtual Machine⁶ (KVM). All standard Java applications use the Java Virtual Machine (JVM) as an intermediate to the computer on which they are executing. This enables Java applications to run on different operating systems and platforms than they were compiled on. The K Virtual Machine is a part of the CLDC and is essentially a JVM that has been fitted to work well with mobile phones. According to *K Virtual Machine* [23], this means that it takes up less space, requires less memory and can run more effectively on slow processors than a standard JVM.

MOBILE INFORMATION DEVICE PROFILE (MIDP)

Together with CLDC, MIDP creates a fully operational Java Runtime Environment (JRE)⁷ that works well on devices with limited capabilities. According to the datasheet *Java 2 Platform, Micro Edition* [24] the MIDP adds the abilities to create user interfaces, network connections, local data storage and application management.

⁶The K in K Virtual Machine stands for kilobyte and is a prefix commonly used to indicate that something is developed for devices with limited capabilities, such as mobile phones.

⁷The JRE is the application that translates the Java classes to a form the operating system of the computer can understand.

An application written for MIDP is called a MIDlet. It is possible to create a collection of MIDlets that can share resources amongst them. This is then called a MIDlet suite. When a MIDlet needs to store persistent data, it does not use a file like other types of applications would do. Instead, it uses a record store that is a kind of indexed list of records; each holding some textually represented data.

There are currently two versions of the profile; MIDP 1.0 and MIDP 2.0. Being the least sophisticated version, MIDP 1.0 is supported by all Java-enabled mobile phones. This is not the case for MIDP 2.0, which only a limited set of devices can use, but most likely more phones will come with this capability eventually. In fact, according to the article *Gjør det enkelt å lage nye mobiltjenester*⁸ [25], about 65 percent of mobile phones in Norway today support MIDP 2.0, and it is estimated that the number will have reached 85 percent in 2008.

MIDP 2.0 introduces quite a few improvements compared to the previous version. Amongst other features described in the article *What's new in MIDP 2.0?* [26] are the improved support for end-to-end security and expanded connectivity. Both features include mandatory support for HTTPS, which was optional in MIDP 1.0.

However, as discussed in *Wireless Java Security - Understanding the issues* [27], the default random generator of J2ME uses only the time of instantiation in milliseconds as the random seed. This makes a poor source of randomness for the generator and when it is used for generating keys for an HTTPS session it makes the protocols less secure. When the random number generator is used for other purposes it is possible to create the source of randomness manually, but when it is used for HTTPS this is impossible.

Another problem with the HTTPS support in MIDP is the fact that the phones do not usually check certificate revocation lists during the validation process. This means that if the certificate being examined has been revoked due to a theft of the corresponding keys, the mobile phone will not know of it and may end up trusting the thief.

OPTIONAL PACKAGES

In addition to the basic Java RE for mobile phones it is possible to add optional packages. These may offer specific functionality that not all devices need. This way, J2ME reduces its size and increases its flexibility, but unfortunately it also results in great differences from device to device when it comes to Java support. The following packages are relevant for the topic of this thesis:

JSR-172 J2ME Web Services Specification: As described in the datasheet *Java™ 2 Platform, Micro Edition (J2ME™) Web Services Specification* [28], the J2ME Web Services Specification consists of two optional packages; the XML Processing APIs and a package that provides "RPC-based access to Web Services". The XML Processing APIs provide the functionality needed to be able to do basic manipulations of XML structures.

⁸English: Enables easy creation of new mobile services

RPC stands for Remote Procedure Call and in the world of Web Services this is a technology that allows for a client to invoke procedures at a server by sending XML-based messages. The advantage of this scheme is that the client and server are independent of each other as long as they both understand XML. In other words they may be designed quite differently, but are able to communicate because of their XML processing capabilities.

JSR-177 Security and Trust Services APIs (SATSA): The datasheet *Security and Trust Services APIs For the JavaTM2 Platform, Micro Edition* [29] lists the following capabilities of SATSA: "Application-level digital signatures, basic user credential management, cryptographic operations, and smart card communication". This means that mobile devices that include SATSA may use these capabilities to run applications that, amongst other things, implement user authentication, digital signatures and device authentication. Furthermore, it means that cryptographic keys may be stored in the phone's SIM, where they will be safer than in the phone's memory. Only a few phone models are delivered with SATSA and these are mostly smartphones.

A lot of the functionality achieved by the optional packages may also be achieved by implementing it together with the application that intends to use it. Consequently, the application would take up more space, but it would also be possible to use it on phones that do not include any of the optional packages. Thus, the decision whether to use the optional packages or not depends on which is most important; targeting more devices or requiring less storage space.

4 eGovernment services

4.1 Introduction

Electronic government (eGovernment) is the government's attempt to provide public administration through the use of information technology. Such projects are mainly targeted towards the citizens, where the goal is to reduce the paperwork needed to communicate with public authorities. This is called the G2C (Government-to-Consumer or Citizen) model and it is the focus area of our project. The G2C service is a direct communication between each citizen and the government, providing services from various departments. There are also two other models which target businesses (G2B) and other governments (G2G).

A subsection of eGovernment is the use of mobile devices to access the eGovernment services. This is sometimes called mobile government or mGovernment. The goal of this is of course to provide availability of information and services. It is important to note that this is merely a supplement to the existing services and cannot replace them.

The rest of this chapter should provide a better understanding of eGovernment through information about the status of various projects, an explanation of the technology used. It also looks at some requirements that must be fulfilled before governments may deploy electronic services.

4.2 Today's situation

Providing electronic services require a change of current practice in government administration and this has proved to be the main hold-up of these kinds of projects. However, there are some parts of the world that have progressed considerably in the past few years. The monitoring of the various nations' progress is done both locally and globally. Two examples of the latter are found in the *eEurope 2005 action plan* [30] and the "Office of E-Government and Information Technology" in the US.

A good survey among the UN member states is also provided by the *UN Global E-government Readiness Report* [31]. This shows both the readiness and the level of availability for the entire population. What is important to note is that Europe and the US are high up on the readiness scale, whereas Africa is far behind. Norway is ranked number 7 in this survey, which will give us an idea of Norway's position as we now switch focus to examine the Norwegian eGovernment projects.

The Norwegian government has formulated a strategy called *eNorge 2009*⁹ [32] which states the goal for the use of information technology in Norway. The vision presented in this plan is to simplify the lives of citizens and to help promote growth through innovation in the business sector. To start with, the access to Internet and computers must be ensured, then the next step is to make electronic services available. In the end, they want to offer all types of governmental services entirely through electronic systems and gather everything into one location. To achieve their goal, they focus on the need for better cooperation between the departments and a centralised plan for all major IT projects.

Presently, the only site offering electronic services in Norway is provided by *AltInn* [33], which allows persons and businesses to find, fill out and deliver forms to government departments. However, the development of a citizen portal called "MinSide" will be the next step towards electronic access. The plan is for this to deliver both register, transaction, message and calendar services in a G2C model. The work is supervised by the Ministry of Government Administration and Reform, but the launch of this portal has been postponed several times as they struggle to get the framework finished. The first version of the technical solution is ready, but connecting this with the new security portal (in charge of authenticating the users) and the providers (various departments) has proved to require a more carefully thought-out infrastructure.

4.3 Service types

Web portal technology is both well suited and often used in eGovernment services. Upon logging in to the portal, the user is shown a list of available governmental services. The user can then access these services by following their links without having to reauthenticate. In the Norwegian eGovernment project, the services are divided into transaction and register services which is explained later.

To enable the users to access the services without having to reauthenticate, all the authentication is based on the security portal, which is trusted by all the service providers. The security portal is able to send along tokens that are accepted by the service providers as proof that the user has authenticated successfully. This functionality makes the use of the system more convenient for the user, as he only has to remember one set of credentials and only has to authenticate once every time he uses the system.

A transaction service is a service that offers its own user interface. This means that, upon authenticating and selecting the service, the user will be transferred to the service provider to make use of the functionality offered there. An example of a transaction service is applying for a driver's licence. The user has to enter information into a form, but the information may be sensitive. Thus, the service provider does not want other entities to handle the information and chooses to provide the interface locally.

⁹English: eNorway 2009

Register services are services without their own user interfaces or services that offer information it trusts the portal to handle correctly. If the user chooses a register service, he stays put at the web portal while the portal requests the service on the user's behalf. The results are then presented through the interface of the web portal.

4.4 Regulations

When dealing with eGovernment services there are always governmental guidelines, laws or requirements to respect. So far only a few countries have prepared specific legal measures for providing eGovernment services. In Norway, the *Requirements specification for PKI for the public sector* [34] is an example of a fairly new directive that determines the rules for use of certificates and asymmetric key pairs in eGovernment services. It deals with nearly all aspects of PKI, ranging from key properties to requirements set for the issuer of certificates and private keys.

Additionally, *Lov om elektronisk signatur (esignaturloven)*¹⁰ [35] governs the use of PKI for digital signatures. It too has demands for nearly all aspects of PKI, but emphasises the parts related to digital signatures and is only valid for certificates and keys intended for this purpose.

The certificates spoken of by the *Requirements specification for PKI for the public sector* are divided into classes. The certificate classes are specified in *Leveranse oppgave 1 Anbefalte sertifikatprofiler for personsertifikater og virksomhetssertifikater*¹¹ [36]. This document specifies what kind of information is supposed to be included in each field of the certificate.

Last but not least, the Norwegian Data Inspectorate govern the use and protection of personal information. They too have their say in what kind of information the eGovernment services are allowed to handle and how this information should be stored in a secure way. They act according to *Lov om behandling av personopplysninger (personopplysningsloven)*¹² [37].

In addition to all the laws and requirements, there is a document called *Implementasjonsguide Registerinformasjon Min Side*¹³ [38]. This document is a guide to help developers make their service providers able to offer register services through the web portal called "MinSide". It too defines requirements, but they are directed towards the interface of the services and are only necessary to follow for those who intend to use "MinSide". In other cases the implementation guide can serve as a source of tips for how to make a specific service interoperable with those linked to "MinSide".

¹⁰English: Electronic signature law

¹¹English: Delivery task 1 Recommended certificate profiles for personal and enterprise certificates

¹²English: Law of personal information management

¹³English: Implementation guide for register information MyPage

5 Strong authentication on mobile phones

5.1 Introduction

This chapter explores the possibilities of strong authentication on mobile phones. There are two possible scenarios; the mobile can be used to access protected resources which require strong authentication, and the mobile can be used as an authentication token when accessing the resource from the Internet. We will only look at the first alternative.

The first section explores the theoretical basis and provides a definition. This will build on some of the aspects presented in chapter 3 and so an understanding of these should be achieved before continuing. Then, we will look at some of the work that has already been done in this area by presenting standards and existing products. Finally, we will make a decision on how we will achieve strong authentication in the part of the design that requires this functionality.

5.2 Definition

As mentioned earlier, authentication is the process of verifying that an entity has the identity it claims to have. In theory this entity may be anything from the user to an application or device, but in our case the focus is on authentication of the user. This is because we work with the perspective of identity federation and eGovernment services, which requires strong user authentication more than authentication of application or devices.

As explained in *It's All About Authentication* [7], a common definition of strong authentication is authentication that requires at least two factors. An authentication factor is a type of identity proof that is given during an authentication process. This means that the user must provide at least two different types of proof of his identity and this is the definition we will use throughout this project. There are three distinct authentication factors:

- Something the user knows, like a PIN code or password
- Something the user has, like a hardware token or a smart card
- Something the user is, like a fingerprint

Liberty defines a set of authentication classes that can be used to divide different authentication methods into categories. These may be used to specify the strength and type

of authentication used or required. The classes relevant for mobile phones are defined according to two distinct properties; the number of authentication factors and whether the user's mobile subscription profile is registered at the identity provider or not. As described in the *Liberty ID-FF Authentication Context Specification* [39], this results in the following authentication classes:

MobileOneFactorUnregistered: In this case, the user may not be known to the identity provider in advance, or if he is known there is no link between his identity at the identity provider and his mobile subscription. The only factor used is the mobile phone itself, and therefore it is the only thing that is authenticated. From our point of view, authentication methods in this class cannot be deemed strong.

MobileTwoFactorUnregistered: This class offers no link between the user's identity at the identity provider and his mobile subscription profile, but it does require a second factor other than the phone. This way, the identity provider may authenticate the mobile phone and the mobile phone may assert its user's identity to the identity provider.

The identity provider will not be able to check the correspondence of the phone user's identity with the registered user of the phone, but if the phone data was captured in the enrolment process it may be used as an authentication token. Alternatively, the identity provider may issue its own token that has to be used in the authentication. Providing that there is a connection between the user's identity and either the phone or some token that is presented by the phone, the class would provide strong authentication.

MobileOneFactorContract: Just as in *MobileOneFactorUnregistered*, this class will only authenticate the phone and not the user. In this class there is a link between the identity of the user at the identity provider and the mobile subscription profile of the user. Therefore, the identity provider may assume that the owner of the mobile phone is also its user. This relation is not sufficient to provide strong authentication as it cannot be considered a second factor because it is not given by the user at the time of the authentication.

MobileTwoFactorContract: This class provides both device and user authentication and an assured link between the user and his phone. Then, the phone will be something the user has, and the PIN code or biometrics the user provided as a second factor will be something the user knows or is. In other words, the authentication would be strong.

Of the four authentication classes for mobile phones described by Liberty, only two provide strong authentication from our point of view. However, the *MobileTwoFactorContract* would require an agreement with the user's mobile phone operator and this kind of agreement may cost money to arrange or make use of. For some applications this arrangement would work just fine, but in other cases it may limit the amount of potential users. This is because not all users have the same operator and to make an agreement with all of them would be a too expensive and/or cumbersome.

5.3 Authentication methods

In strong authentication there are three major methods which form the basis of the many authentication protocols that exist. These are one-time password (OTP), public key infrastructure (PKI) and biometrics.

ONE-TIME PASSWORD

What we normally refer to as one-time password is when a common shared secret is used to calculate a value which is identical at both the client and server. An example of this is when a bank customer has a code calculator which is used to calculate a number. This number can then be sent to the bank's application to prove that the customer is in possession of the token.

Another approach is the challenge/response protocol where the server generates a challenge value which is sent to the client, who uses a shared secret to generate a response value. This value is then sent back to the server, which can verify its correctness. The advantage of this approach is that the server can verify the user's identity, but neither party have to reveal the shared secret.

However, it is also possible to list all the passwords on a sheet of paper or a plastic card. The user will then be asked to enter one of these when he wishes to authenticate. Each of the passwords will only be used once, and the user will be given a new set of passwords when he has used all of them.

BIOMETRICS

Biometrics is a way to authenticate by use of the unique physical properties of the body. It must be done using a physical property which is hard to change and which it is known that only the correct person possesses. There are a number of different types currently in use and some examples include:

- Fingerprint scan
- Facial recognition
- Signature recognition
- Retinal or iris scan
- Voiceprint
- DNA fingerprint

The most common method is currently fingerprint recognition. There are several reasons, but foremost is the quality of unique identification. Also, fingerprint readers are inexpensive to make and can easily be integrated into various devices. This provides an easy way of accessing data compared to long and cumbersome passwords.

PUBLIC KEY INFRASTRUCTURE

When using PKI to authenticate, the user signs a challenge value with his private key. The signature is then returned to the challenger. The certificate could be sent along with the signature or obtained by the challenger in some other way. By validating the signature with the public key included in the certificate the challenger can be certain that the user is in possession of the private key corresponding to the certificate. Then the challenger must validate the signature on the certificate to be sure of the connection between the key pair and the person represented by the certificate. If everything goes well he can put the two together and the user will be authenticated.

5.4 Authentication tokens

Authentication tokens address the "what the user has" aspect of authentication. One of the advantages of tokens is that it is easy to revoke or disable them if they should fall into the wrong hands. However, the effectiveness of this scheme is dependent on the user notifying the token issuer or similar if he loses his token or suspects abuse. As users sometimes do not fully understand the risks, they may delay this action, thus increasing the possibilities of an attack. Therefore, the token alone does not necessarily provide more security than a password, but together with another aspect of authentication it may be very secure.

The rest of this section describes different kinds of authentication tokens. The first two parts look at hardware and software tokens respectively. These parts consider the token types generally, without special attention to mobile phones. The last part discusses how the token types previously mentioned would be suited for use on mobile phones.

HARDWARE TOKEN

A hardware token is a small security device which is given to a user to utilise in authentication. As described in the presentation *Public Key Infrastructure* [40], the defining characteristics of such a token are that it is one of a kind and that the cryptographic information never leaves the token. Usually this is accomplished by performing the cryptographic operations directly on the device and then returning the result.

The fact that the information never leaves the token means that it cannot be copied to another token. Consequently, they must all be unique, although they may look alike on the outside. These characteristics of hardware tokens are part of what make them so secure. Considering that each token is unique, the users will not have more than one of each token lying around his office or at home. Then he will take more care of the one token he has and he will notice a theft much quicker.

The hardware tokens may take on one of two forms; one-time password (OTP) generator or tamper-resistant cryptographic device. In the first case the token may look similar to a calculator, with a small alpha-numerical display and one or two buttons. If the token requires a PIN it usually has numbered buttons as well. Depending on the

physical layout of the token, the OTP generator is activated in one of the following ways: a button is pressed, a PIN is entered or the internal clock reaches a preset time interval. When the token is activated it displays a number that can be used as a one-time password for access to a system.

The tamper-resistant cryptographic device may also look different depending on how it is meant to be used, but often it is in the form of a smart card. A smart card is a card that is the size of a credit card and contains a microchip. This microchip contains amongst other things the cryptographic secret that is used for authenticating the user. It is capable of performing cryptographic operations. A system that wishes to authenticate the card holder may send a challenge to the card, which signs the challenge digitally and returns it. Then, the system may verify that the signature is correct.

Another way of using a smart card is to have it function the same way as an OTP generator. The system that wishes to authenticate the user may send a request to the card, so that it generates a new OTP. Then, the system can check whether this OTP corresponds with the OTP pattern that is registered on the user.

However, any use of smart cards need a dedicated smart card reader so that the user agent may communicate with them. Some computers come with smart card readers already installed, but most will have to connect an external reader to their device. Alternatively, the smart card chip may be put in a USB dongle or in the case of the smart card functioning as an OTP generator, a small reader with an alpha-numerical display may be used. However, in the last case the user will have to enter his OTP himself and he will have to carry around an extra device.

SOFTWARE TOKEN

A software token is a file that is downloaded to the PC or device that intends to use it. As mentioned in the presentation *Public Key Infrastructure* [40], software tokens are stored on traditional storage media, such as for instance hard drives, servers and USB flash drives. Common for all these storage media is the fact that they are meant for both reading and writing. In other words, the entire token may be copied and each user may have several copies. Although this arrangement is convenient for the user, the possibilities for an attack increases with each copy. Furthermore, the fact that the software token is so easily copied, an attack may go by unnoticed by the user, because he is still able to use his original token. To counter this problem, some sort of copy protection needs to be applied to the token. For instance, one could require a password to operate the token, and all the token copies could be disabled by the server if the wrong password was entered too many times.

The software tokens function in much the same way as the hardware tokens. The main difference in functionality is that the software tokens usually have more storage and processing capacity at hand. Consequently, they may perform more challenging cryptographic operations, either by calculating a more complex OTP or by signing a larger challenge with a more demanding cryptographic secret.

AUTHENTICATION TOKENS ON MOBILE PHONES

Mobile phones today have limited capabilities when it comes to use of authentication tokens. Most phones come with some internal storage, but with a few exceptions they have very limited capacity. Furthermore, this storage is often used for games, applications, messages, pictures and music, which leaves very little space to store a software token.

Few mobile users think twice before they download free mobile games that are offered through the Internet. The same users would probably be a bit cautious before doing the same on their PC. Traditionally, the threat of malicious content on mobile phones has not been significant, but as we use the mobile device in new ways, the reward of a successful attack increases. This means that if the software token is available in the common storage of the mobile phone, malicious applications may trick the user into giving up the token and maybe even the password.

To counter these problems MIDP 2.0 allows for applications to be digitally signed. This way unsigned applications or applications that are signed by untrusted parties are given restricted permissions. Consequently, the software tokens may be stored in a place where they are out of the reach of untrusted applications. However, a standard MIDP 2.0 implementation does not provide the means for using secure storage in the phone's internal memory. As described in *MIDP Application Security 3: Authentication in MIDP* [41], the phone will have to implement the "JSR-177 Security and Trust Services API" for J2ME to accomplish this.

When it comes to hardware tokens on mobile phones, one token type differentiates itself. The SIM card is a type of smart card that is already present on the mobile phone. Most SIM cards today have cryptographic capabilities that only need to be activated in order for it to function as a hardware token. Furthermore, the SIM card is only available to applications that are signed by the operator and is therefore quite safe from malicious applications. But the fact that the SIM is under the control of the operator is also its downside. This enables the operator to charge for both the use of the SIM and for signing the applications that need access.

To avoid having to pay the operator to use the hardware token, other possibilities than the SIM must be considered. However, most hardware tokens need some kind of a reader or USB connection. For instance, a smart card needs a smart card reader, and although the mobile phone already has a reader for smart cards it is occupied by the SIM. That means that the use of another hardware token than the SIM probably would require changes in the phone's hardware.

As opposed to PCs, mobile phones usually only belong to one person and they are carried around everywhere. These facts enable the mobile phone itself to function as an authentication token. It would then be a cross between a hardware and a software token, because the phone is a physical device, but the method for authenticating the phone may be implemented by use of software token solutions.

However, it could be pointed out that the phone should not be used as a proof of possession when it is also used as the terminal. An argument that supports this statement is that the phone may be stolen. By using a code as a second factor to limit the access to the token, it is possible to disable the token if the wrong code is entered too many times. This will be equally secure as for instance a web banking system using hardware tokens to authenticate the user. In both these cases, if the tokens are stolen, the second factor ensures that the thief will be left with a useless token after it has been disabled.

5.5 Strong authentication standards

Both the Liberty Alliance and the Initiative for Open Authentication (OATH) are currently working on standards intended to enable interoperable strong authentication. None of these are finished, but the OATH architecture is available in a draft version. Consequently, we will briefly summarise the information we have on the future Liberty standard and then present the main points of OATH. Finally, we will try to draw a conclusion on how suitable OATH would be for our application.

5.5.1 ID-SAFE

Liberty's Identity Strong Authentication Framework (ID-SAFE) is a framework currently being developed by the Liberty Strong Authentication Expert Group (SAEG). Its purpose is to provide strong authentication by supporting a number of different types and methods. According to *Liberty Alliance Project - Frequently Asked Questions* [42] an important quality is that it is based on open standards to promote interoperability between various suppliers of strong authentication devices. It should then be possible to use all types of software/hardware tokens, smart cards (for instance SIM cards) and biometrics. Since their goal is "universal strong authentication"¹⁴, any device can be used. This standard has not yet been released, but as mentioned in the FAQ, a draft version is expected in the middle of 2006.

5.5.2 OATH

The OATH Reference Architecture is described in *OATH Reference Architecture Release 1.0* [43]. It is an approach for providing strong authentication, which is meant as a way of realising their vision of universal strong authentication. To find out whether we can use this architecture in our project we will look at the background for why the architecture was created, the parts of the reference architecture that have been developed so far and one important algorithm that has been developed as part of this initiative.

¹⁴Universal strong authentication is when all users, all devices and all networks are strongly authenticated so that every step of the authentication process is covered.

We will not try to describe all aspects of the architecture here. Instead we present the goals and explain the main components. As always, we work from the mobile phone perspective and try to focus on what is important for strong authentication on mobile phones. Finally, we discuss how suitable the architecture is to apply in our project.

OBJECTIVES

As presented in the *OATH Reference Architecture Release 1.0* [43] there are four aspects which are important during the development of the architecture. These are the goals that the architecture strive to fulfil when it is completed. They are as follows:

Open and free to use: To ensure that the specification is open and free to use, existing standards should be used or adapted whenever possible. Otherwise, standardisation should be attempted using existing standardisation organisations.

Device development and embedding in existing mobile devices: The building blocks must be clearly specified so that it will become cheaper and easier to make authentication devices and possible to turn existing portable devices into authentication devices.

Native platform support: Applications and frameworks should have built-in support for strong authentication, possibly through the use of platform connectors.

Interoperable modules: The framework should support modularity, meaning that it should be possible to construct an authentication solution from existing modules. This will enable companies to compose authentication solutions using components from their preferred suppliers, which will usually be the best products available.

In addition to the architecture, OATH has also released a roadmap to help guide the development process towards universal strong authentication. The proposed OATH roadmap, *An Industry Roadmap for Open Strong Authentication* [44], has three main areas of interest, which are listed below. For each area, we provide a summary of the roadmap's suggestions.

Credentials and security devices This describes the authentication methods SIM, PKI and OTP and suggest that all these methods are supported and work together since they are all used for certain purposes. Additionally, OATH encourage the development of "All-in-one security devices". These are devices that are capable of handling many or all the authentication methods simultaneously.

Authentication protocols framework Some applications require a specific protocol for authentication, but the OATH framework suggest the following: the use of 802.1x¹⁵ for network applications and "embedded 802.1x clients" for devices.

¹⁵802.1x was created by the Institute of Electrical and Electronics Engineers (IEEE). According to *802.1X Offers Authentication and Key Management* [45] it is a standard for authentication over a network that supports several different authentication methods.

Credential provisioning and validation Because of the need for a unified way of handling the secrets (meaning issuing and managing the credentials), OATH suggests a solution where an RSA¹⁶ key pair is used to encrypt any shared secrets. The key pair is therefore superior to the secrets and existing PKI solutions and protocols can be used.

REFERENCE ARCHITECTURE

The OATH Reference Architecture 1.0¹⁷ consists of four main focus areas, which are the "Client Framework", "Validation Framework", "Provisioning and Management Framework" and "Common Data Model". The first three are part of the high level illustration of an "open authentication system" as shown in figure 5.1. The main components of the figure are explained below.

The "Common Data Model" describes how OATH envision providing interoperability in new and existing systems. To achieve this, OATH propose to standardise existing schemas that are used on systems to support strong authentication. This will allow a company to use any solution for strong authentication that follows this standard. They also suggest that token suppliers should attach meta-data following a standard format so that the token can be used with modules made by other suppliers.

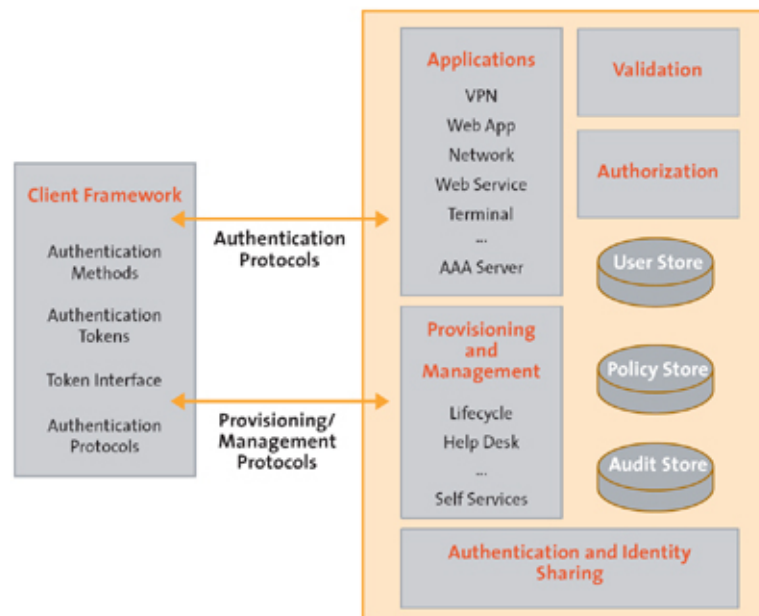


Figure 5.1: OATH authentication architecture
(taken from [43])

¹⁶RSA is a public key algorithm developed by Ron Rivest, Adi Shamir, and Leonard Adleman.

¹⁷Version 2.0 of this architecture will be released some time later this year and this will be the complete version.

Applications

This contains the applications that require strong authentication of their users. They use an authentication protocol based on a common standard to talk to the users and then a validation protocol to communicate with the validation module.

Authentication and Identity Sharing

In today's world of interconnected applications, more and more organisations have to share their user information to provide more desirable services to the end users. However, not all organisations wish to share everything both for security reasons and business related considerations. This module's task is to provide the technology that allow for organisations to share only the users' identity, only the users' authentication or both, depending on each organisation's needs and desires.

Authorisation

This module is a way to check an authenticated user before allowing him access to a needed resource. Authorisation is distinguished as a separate task, because a user that is authenticated successfully may still lack the privileges to access the resource in question. Thus, the authorisation may fail independently of the authentication.

User/Policy/Audit Store

As the names suggest, these are repositories that contain user profiles, policies and information about audit events. The first is a collection of personal information, attributes and possibly credentials.

Client Framework

OATH recommend that as many authentication methods as possible are supported, that both existing and new token standards can be used and that the standard protocols are supported. The aspects from the *OATH Reference Architecture* [43] that are important for our case are the possibilities of :

- Having an embedded token in a device
- Using the token for multiple authentication methods (for instance both as a OTP calculator and a storage for a PKI certificate)
- Having multiple keys (for use with different service providers) on one token
- Using their HMAC-based OTP (HOTP) algorithm for development of OTP calculator on the device

Validation

The Validation Framework allows for the attachment of several different handlers, so that the resulting solution may provide support for more than one protocol and more than one validation method. Furthermore, it provides support for simultaneous validation of several clients.

OATH's main intention with this framework is "to promote the development of appropriate interfaces (for both protocol and validation handlers)", *OATH Reference Architecture* [43]. This means that they want to provide the basic resources that vendors may need to use their own handlers with the framework. Furthermore, a number of existing validation protocols and interfaces may be deployed in the framework.

Provisioning and Management

Provisioning in this context is the process of securely delivering credentials and software to the correct entities, while management is concerned with the rest of the life cycle of these credentials and software. The challenge in these tasks lies in the fact that the information that is to be delivered may be destined for any of a wide range of devices that are potentially very different.

One of the main focus areas of OATH in this protocol is to allow vendors and customers the freedom to choose when it comes to the implementation of their provisioning protocols. Both open standards and proprietary protocols may be supported. Furthermore, OATH intend to "evaluate the requirements" that have to be in place to provide a better service than the existing protocols do for devices with limited capabilities. However, the way it is presented in the *OATH Reference Architecture* [43] indicates that this work has not finished or provided any results yet.

HOTP ALGORITHM

The HMAC OTP algorithm can be implemented in either software or hardware by any manufacturer or developer. This flexibility is what makes it ideal for use in universal strong authentication. Furthermore, the current version of the algorithm is counter-based, making it independent of carefully synchronised clocks on server and device. However, the HOTP algorithm is under constant development and future extensions may include a time-based algorithm. The current version of the HOTP has been submitted to the Internet Engineering Task Force (IETF) as a Request For Comments (RFC). This means that anyone can look at it and give their feedback.

RELEVANCE OF OATH

Although OATH are well on their way, the reference architecture is still in an early stage. Being cautious not to be dependent on something that is only available as a draft, we do not wish to base our work on OATH. However, they have some interesting ideas, especially concerning universal strong authentication.

The idea of allowing users to authenticate anywhere using the same methods is intriguing as it would mean a lot of advantages, both for the user and the service providers. For the user it would mean fewer codes to remember and fewer security devices to handle, in short it would be more convenient. For the service provider it would mean that they would not have to create their own mechanisms, but could share those of others.

For these reasons, we will try to think about this idea during the design of the strong authentication mechanism. However, as the reference architecture is not finished yet, we will not use it.

5.6 Existing products

Below we describe some of the products we found in our survey of strong authentication technology on mobile phones. The first three are PKI-based, while the rest are based on OTP.

ZEBSIGN MOBILE SOLUTION

ZebSign Mobile Solution provides ZebSign PersonID certificates for use on mobile phones. ZebSign PersonID certificates are qualified certificates for digital signatures according to Norwegian law. The solution is PKI-based and utilises the cryptographic capabilities the SIM cards already present in the mobile phones. According to the *ZebSign PersonID Certificate Policy* [46] the generation of both the private and public key is done in the SIM card by use of an activation code and a PIN code provided by the user. Then the private key is kept in the SIM and protected by the PIN, while the public key is sent to the CA to be associated with the user's certificate. When it is time to renew the keys, a new activation code must be acquired by the user and the process must be repeated.

Minimum key size of this solution is set to 1024 bits RSA for both user keys and CA keys. The certificates contain a unique user identifier that can be used to get the user's national identity number, if the service that requests this number is authorised to have access to such information.

TELENOR MOBIL PKI

Telenor is one of the largest mobile network operators in Norway. They have devised a solution they call "Mobil PKI"¹⁸. As explained in *Telenor - MobilHandel*¹⁹ [47] their solution supports both authentication and digital signatures and can be used for several purposes.

¹⁸English: Mobile PKI

¹⁹English: Telenor - MobileCommerce

Telenor Mobil PKI is SMS-based, which means that it works by attaching the user's digital signature to SMS messages. These can then be sent back to the server, which can verify the signature with the user's public key. Like the ZebSign Mobile Solution it works by utilising the cryptographic functionality of the SIM cards and it is also ZebSign that has the responsibility of governing the public certificates used in Telenor Mobil PKI. These similarities to ZebSign and the fact that ZebSign list Telenor under the heading *ZebSign Partnere / Kanaler*²⁰ on their homepage [48], make us suspect that the two solutions are one and the same. However, we did not find conclusive evidence of this, thus we describe them both separately.

Another reason for describing Telenor Mobil PKI specifically is the fact that, according to *Brønnøysundregistrene*²¹ [49], SPAMA's eFactory Security Server supports "Mobil-PKI". As SPAMA's eFactory Security Server is the software used in the security portal²², this means that it too supports "Mobil-PKI". Although "Mobil-PKI" could mean anything the way it is mentioned in the web page, John Ø. Kårikstad²³ confirmed by email that the security portal supports Telenor Mobil PKI, but that this functionality has not been activated.

BUYPASS MOBIL-ID

Buypass is one of Norway's leading vendors of smart cards. The Buypass smart cards are used both for authentication and for digital signatures, as well as virtual money²⁴. Amongst other things, they have provided Norsk Tipping²⁵ with the ability to offer their games over the Internet in a secure fashion.

One of Buypass's latest products is Buypass Mobil-ID, which is a software token that is downloaded to the mobile phone together with the Java application that intends to use it. The token is operator independent and offers the same functionality as the smart cards. According to the company's own description of the product, *Sikker identifisering og betaling via mobil*²⁶ [50], Buypass Mobil-ID is currently only available in Norsk Tipping's mobile gaming solution, but is easily adaptable for use with other applications. However, due to the fact that Buypass has a pending patent application on this product, we were not able to find any more information about it.

DIVERSINET'S MOBISECURE SOFT TOKENS

This product is an open platform for two-factor authentication on mobile devices based on the OATH reference architecture. Diversinet's approach simply involves support for OATH's OTP algorithm in their applications. The MobiSoft product suite contains several software tokens for use on mobile phones. A short description of each of the relevant

²⁰English: ZebSign Partners / Channels

²¹English: The Brønnøysund Register Centre

²²The security portal will be discussed later in the report. For the purpose of this chapter it is enough to say that it is the login service used in the Norwegian eGovernment projects.

²³Head of research and development, Sem & Steinersen Prokom AS

²⁴Virtual money are values that are registered in a computer system and used for electronic payments.

²⁵English: Norwegian Lottery

²⁶English: Secure Identification and Payment via Mobile Phones

technologies as they are presented in the *MobiSecure Soft Token Product Overview* [51] is provided below. The MobiSecure soft token has also been licensed by VeriSign for use in their solution called "Unified Authentication".

MobiSecure Token This is a software application that uses a credential stored on the phone to provide OTP values.

MobiSecure Multi-Token The user can store and administer several credentials on the mobile phone, intended for different service providers.

MobiSecure Browser Plug-In Token A micro browser on the mobile phone is used to request the service which provides strong authentication (based on OTP) of the user.

MobiSecure API Token Security in an application running on the device can be provided through OTP-based authentication, directly accessed via an API.

VASCO'S DIGIPASS GO 10

This product allows both strong authentication and digital signature from an ordinary mobile phone by using a software version of Digipass which is integrated on the SIM card. This product acts as a calculator that delivers OTPs that can be used to access a remote system and it makes Message Authentication Codes (MAC) which can be used to protect the integrity of message content. To achieve this it uses the DES algorithm²⁷ and some secret keys which are saved in the Digipass from the beginning and then never revealed. Two-factor authentication is covered since you have the token itself and you need to know the PIN to activate it.

PORTWISE MOBILE ID SOFT TOKEN

PortWise Mobile ID Soft Token is an OTP-based strong authentication token that can be used on a wide range of devices. As described in *PortWise 4.5 Strong authentication from PortWise* [52] PortWise's solution can be used in two different modes; synchronised and challenge. In the synchronised mode the server and client has a counter that determines which password to use next. The challenge mode depends on a server generated challenge to create the OTP. Both modes require the user to enter a PIN to generate the OTP. According to the whitepaper, the PortWise platform is OATH compliant, but it does not seem like they use the HOTP algorithm.

²⁷"DES, once the encryption standard of the United States government, is a block cipher that uses 16 rounds of activity against a 64-bit block of data, with each round adding a layer of complexity." [3]

5.7 Evaluation

As can be seen from the previous sections, there is a lot of work in progress on the topic of strong authentication with mobile phones. Mobile technology is advancing, and people are starting to realise the potential value of offering this functionality.

Some of the products we have surveyed may be possible to use for our application, but it is difficult to find enough information about them, as most of the companies that offer these products are reluctant to give out what they consider to be business secrets. The most promising of these products may be Telenor Mobil PKI. As it is already supported by the security portal, it would require less work to deploy. However, this will only work with phones that use Telenor, and we do not like this apparent dependence on one particular mobile operator.

Thus, Bypass Mobil-ID would be better as it is operator independent, but the lack of information about this product means that we are neither able nor willing to devise a solution that uses it. Maybe we would be allowed more information if we could convince them that we were going to deploy a solution based on their product, but as this is not the case we will create our own scheme of authentication instead. Some parts may be inspired by the products above and it would probably be possible to swap this scheme for Bypass if this was found to be the best solution at the time of implementation.

5.7.1 Main goals

To set some of the criteria the method of authentication has to fulfil, we give a list of the goals we want to achieve with the method we choose.

Security: Our first and most important goal is that the method should be secure enough for use with eGovernment services. This goal will mostly be fulfilled through the design, but it is mentioned here anyway because it is important that the chosen method does not show obvious signs of not being able to give adequate security.

Convenience: The motivation for using the mobile phone instead of a computer is convenience. The mobile phone is carried around and will usually be at hand the very instant the user decides he wants to access some protected resource. Therefore, it is important that the only thing he needs to carry to authenticate is the mobile phone itself.

Non-redundancy: As the mobile phone is a device with limited storage capacity, the method should not require large amounts of redundant data to be stored. Consequently, it would also be preferable to be able to use the same authentication for several purposes. Then, it would not be necessary to store a set of credentials for each authentication service the user has dealings with. This goal also coincides with the goal of OATH, as by fulfilling it we would take a step along the way to universal strong authentication.

5.7.2 Choice of method

When it comes to strong authentication, the methods to choose from are symmetric keys, PKI, one-time passwords and biometrics. As one of our goals is to perform the authentication without any extra devices, the OTP method cannot be based on code calculators or code sheets, which leaves us with the option of implementing the OTP generator as an application on the phone. Regardless of which method is chosen, it will have to be accompanied by a password to achieve the second factor of the authentication. As we are dealing with mobile phones, biometrics can be ruled out straight away. This method would require changes in the phones' hardware, as support for biometrics in such devices is currently very rare.

Symmetric keys require the user and the authentication service to share a set of common keys. In other words, it is only possible to authenticate at one authentication service with one set of keys. If the user wishes to authenticate with another service, using the same method, he will need to get another key. This will also be true for OTP authentication when the OTP generator uses a shared secret to calculate the next password. If the generator is instead dependent on a counter value to find the correct OTP, it has to be synchronised with the authentication service to find the correct code. Consequently, neither of these are suitable for use with several services.

In authentication methods that require a shared secret to operate, there is also the problem of provisioning. The challenge is to deliver the shared secret to the phone securely. There are various methods that work on ordinary computers which may be tailored to work with mobile phones, but the best would probably be to avoid the problem all together. This suggests that PKI would be a suitable approach.

The advantage of PKI is that the user is the only one who needs access to the secret key. To prove the user's identity, it is sufficient to use the secret key to digitally sign some value and then send the signature together with the certificate and public key to the authentication service. Consequently, the keys can be generated at the mobile phone and the secret key will never need to be sent over the network. The access to the secret key can be protected by a password and the service will know that the user has the password when he is able to access the secret key. Furthermore, with the use of signed digital certificates it is possible to authenticate anywhere, as long as the authentication service trusts the entity that signed the certificate.

However, this approach also introduces some new challenges. After the key generation, the user will have to get his certificate signed somehow. Consequently, he will need to send it to a certificate signing service at a commonly trusted certificate authority. The actual transmission of the certificate does not pose a threat to the strength of the authentication method, as the secret key can remain hidden on the phone. But the certificate and keys have no certified link to the user's identity yet, which means that the certificate authority will have to check their validity with other means. To solve this problem it is possible to use an OTP to authenticate the user prior to the signing of the certificate. As this task is done only once in a while, we are willing to tolerate an external source for the OTP.

6 Liberty ID-WSF on mobile phones

6.1 Introduction

In this chapter we first explain how Liberty ID-WSF can be used on mobile phones. To do this we look at what kind of messages are passed between the entities of the system to enable the phone to receive services from a web service provider. Then, we explore some of the products available today that implement all or some of the Liberty ID-WSF specification. This part of the chapter ends with a brief summary where we examine if any of the products can be used in the development of the proof-of-concept.

6.2 Liberty ID-WSF

Liberty ID-WSF aims to help developers make interoperable identity-based web services. Identity-based web services refer to web services that utilise the user's identity to be able to offer him better services. Liberty emphasise that the services should be interoperable, meaning that they should have the ability to communicate with each other across different platforms.

6.2.1 Liberty ID-WSF SOAP binding

The basic and most common binding of Liberty ID-WSF is the *Liberty ID-WSF SOAP Binding Specification* [53]. It specifies how to send the SOAP messages that are required by Liberty between Liberty-enabled entities. Furthermore, the specification defines a set of header blocks that may be attached to SOAP messages to convey context specific information. An example of a header block is the `<Correlation>` header block. It is used to provide a connection between request and response messages and to prevent replay attacks²⁸.

Another important header block is the `<Security>` header. It was originally defined by WS-Security [54], but has been adopted by Liberty to convey various kinds of security tokens. These tokens can be used as tickets for accessing services, and can be protected from tampering by use of digital signatures.

²⁸According to *Network Security Essentials* [4], a replay attack is "an attack in which a service already authorized and completed is forged by another 'duplicate request' in an attempt to repeat authorized commands".

The actual messages are to be placed in the <Body> element of the SOAP envelope. The <Body> element can contain anything from service provider specific information to Liberty specified protocol messages, as long as they are structured as XML. Common for all these messages is that they contain the instructions for providing the service or data that is the result of a service.

To protect the integrity and confidentiality of the communication over this binding, Liberty specify the following security considerations that should be covered by implementations:

Protection of the header blocks: To protect the integrity of the header blocks they should be signed and the receiver should check that the signature is correct.

Verification of sender identity claims: To make sure that the sender's identity claims are valid, the *Liberty Metadata Description and Discovery Specification* [55] should be used as much as possible. This specification defines how to communicate about the attributes that may help the receiving entity accept or reject an identity claim.

Prevention of replay attacks: Every entity that receives messages over the SOAP binding should keep a cache of these messages so that replays can be more easily detected. Providing that the <Correlation> header is used, it may be sufficient to cache the message ID and check that a message with the same ID is not received again for some period of time. The *Liberty ID-WSF SOAP Binding Specification* [53] states that "the probability of two randomly chosen identifiers being identical MUST be less than 2^{-128} and SHOULD be less than 2^{-160} ". Assuming that this requirement is fulfilled, this means that the probability of two message identifiers, occurring within a reasonable period of time, accidentally being equal is fairly small. Thus, the risk of falsely detecting replay is also quite insignificant.

6.2.2 Mobile phone in a Liberty Circle of Trust

This section describes how Liberty can be used to communicate with a mobile phone accessing a Circle of Trust. The focus will be on using the mobile phone as a LUAD-WSC and therefore some of the possibilities of Liberty ID-WSF, that are meant for other uses, may be omitted. As the mobile application is meant to be a supplement to an existing web-based system, we assume that the user has already given his consent through the web-based system to federate his identity at the WSP and IdP and to the use of the mobile application.

The description will treat the system entities as black boxes and concentrate on the communication flows between them. In other words, it will not describe the internal processes in detail. Figure 6.1 shows the main messages that are sent to and from the different entities, and the message flows are explained in the following sections. All these messages must adhere to the Liberty ID-WSF SOAP binding.

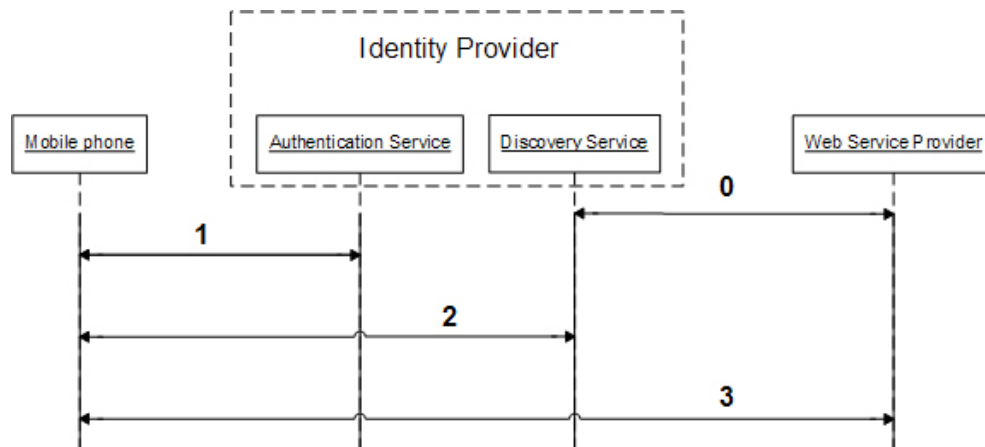


Figure 6.1: The message exchange of a mobile phone accessing a CoT

STEP 0: DiscoveryUpdate

This step happens before all the other steps and the time it takes from this step is completed until the next step is initiated cannot be known in advance. Strictly speaking, it is independent of the mobile phone, but we have included the step in this description to provide a better understanding of the information that lies within the discovery service.

The DiscoveryUpdate operation is defined in the *Liberty ID-WSF Discovery Service Specification* [56]. Its intention is to allow services to publish or delete their service descriptions at the Discovery Service. The DS may be hosted independently of the Identity Provider, but sometimes it is most convenient to include it in the IdP as figure 6.1 shows. Updates of service descriptions are accomplished by first deleting the old description and then publishing the new in a single operation.

The DS presents the service descriptions as `<ServiceInstance>` elements. Commonly, one `<ServiceInstance>` element is used for each service provider and may be connected to several resources within this provider. Each resource is presented as a `<ResourceOffering>` element and can be either identity related data or services. The `<ResourceOffering>` is identified by its `<ResourceID>` element, which may also provide a connection to the identity for which the resource is intended. For privacy reasons, the resource identifiers may need to be encrypted. In which case, the `<ResourceID>` element is replaced by an `<EncryptedResourceID>` element.

The description of the actual services are contained in the `<Description>` element of the `<ServiceInstance>` and may for instance include a reference to an external WSDL and information about what security mechanisms the service instance supports. This is represented as a prioritised list, so that clients may choose the one that is most suitable for both parties. Additionally, `<Description>` may contain a `<CredentialRef>` element containing a reference to a set of credentials or tokens meant to be used to access the service.

The `<Description>` element also specifies which type of service it is. This is done by providing a link to an abstract WSDL which defines the service type. If the concrete WSDL can be deduced from the abstract one by use of logic, it will be redundant. In such cases it is sufficient to include an `<Endpoint>` element instead of the complete WSDL, which merely states where to find the service.

STEP 1: Authentication

The communication flow of this step is regulated by the ID-WSF Authentication Protocol²⁹ as it is presented in *Liberty ID-WSF Authentication Service and Single Sign-On Service Specification* [57]. The specification recommends that the communication in this step is protected by SSL/TLS and that the client authenticates the server by use of the methods defined in SSL/TLS.

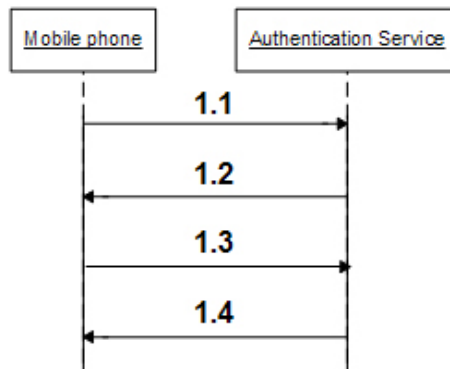


Figure 6.2: Decomposition of step 1 in the mobile's message exchange

As depicted in figure 6.2, this step may be decomposed into the following message flows:

STEP 1.1 Authentication procedure initiation: The authentication procedure must be initiated by the mobile phone sending a `<SASLRequest>` to the Authentication Service (AS) at the IdP. The message must contain identifiers of at least one authentication mechanism that the phone is prepared to use in the actual authentication. If only one such mechanism identifier is contained in the message, it is sometimes possible to reduce the message exchange by including the client response in this step instead of waiting until step 1.3.

In addition to the mechanism, identifiers the request may include a `<RequestAuthnContext>` element that further specifies the requirements to the authentication. The request should also specify the `authzID` that the phone wishes to use as an identifier of the user in the authorisation process. Furthermore, the

²⁹The ID-WSF Authentication Protocol is based on SASL. SASL stands for Simple Authentication and Security Layer and is a message exchange protocol especially designed for authentication over the Internet.

request may include an `advisoryAuthnID` that gives the AS a clue of what identity the mobile phone will try to authenticate the user as. Then, the IdP may save resources by refusing the authentication request at an early stage if it knows that it cannot go through with it anyway.

STEP 1.2 Server challenge: If the AS disapproves of the `<SASLRequest>` from the mobile phone, it has to send a `<SASLResponse>` that has a `<Status>` element containing the value "sa:abort" in its `code` attribute. This message cannot contain a `serverMechanism` attribute or any `<PasswordTransforms>`, `<Data>`, `<ResourceOffering>` or `<Credentials>` elements.

If the AS approves of at least one of the authentication mechanisms suggested and the other information provided, it sends a `<SASLResponse>` message to the mobile phone. The response has to include a `serverMechanism` attribute that specifies which SASL mechanism was chosen by the server. Furthermore, it has to specify either "sa:continue" or "sa:OK" as the value of the `<Status>` element.

If the status code is "sa:continue", it is an indication that the server needs more information to perform the authentication. In this case, the server may have included a challenge in the `<Data>` element. The challenge is specific to the SASL mechanism that was chosen, but may for instance be some data which the user has to sign digitally to indicate his possession of his private key. Furthermore, the response may include a `<PasswordTransforms>` element.

The `<PasswordTransforms>` specifies operations that the mobile phone has to perform on any passwords that are entered by the user. This may include transforming letters to lower case, removal of unwanted characters and/or other transformations.

If the status code is "sa:OK" it means that the authentication has completed successfully and step 1.3 may be omitted. In this case, please refer to the explanation of step 1.4 for a detailed description of the message.

STEP 1.3 Client response: If the status code of the authentication service's response is "sa:continue", the mobile phone has to respond by sending another `<SASLRequest>` message to the AS. The message should specify a `mechanism` attribute that corresponds to the `serverMechanism` attribute it received in the previous step, but if the mobile phone or the user chooses to abort the authentication attempt, the `mechanism` attribute may be empty or contain a value that is different from the `serverMechanism` specified.

Providing that the phone or user has not decided to abort, the request should include a response to the challenge made by the AS. The response is contained in the `<Data>` element. However, when the `<SASLRequest>` is used for this step of the process, it must not include the `<RequestAuthnContext>` element or the attributes `authzID` and `advisoryAuthnID`.

STEP 1.4 Server OK: If the user or phone has not aborted the authentication, the AS has to process the <SASLRequest> that contains the response. It may be that the AS is not yet satisfied with the response or that the chosen SASL mechanism requires further message exchanges. In which case steps 1.2 and 1.3 need to be repeated one or more times. However, on subsequent executions of step 1.2, the `serverMechanism` attribute and the <PasswordTransforms> element must be omitted.

If the AS is satisfied with the results, it returns a <SASLResponse> message with status code "sa:OK". This means that the user has been authenticated. Then the message may include a <Credentials> element and one or more <ResourceOffering> elements, in addition to the content previously described in this kind of messages. The <ResourceID> of the <ResourceOffering> elements should point to a resource that is intended for the newly authenticated identity.

STEP 2: Service lookup

The goal of the service lookup is to obtain a list of available services that fit certain criteria. Additionally, appropriate credentials for the services need to be acquired. In order to do this, the following steps must be successfully performed:

Step 2.1 Query: The <Query> element must at least contain a <ResourceID> element that links the <Query> to the discovery service for the user's identity. If no other criteria are specified, it means that all available entries are requested. To further specify the search, a number of <RequestedServiceType> elements may be included. These indicate which type of service the mobile phone is looking for. The DS may require the mobile phone to include some kind of credentials that it has obtained from the AS. Usually these credentials will be an authentication assertion that is signed by the IdP. However, it is also possible to send the assertion by reference as will be explained in Step 3, together with the reason why we do not address this option more thoroughly in this description.

STEP 2.2 QueryResponse: Upon receiving the <Query> the discovery service must search through its entries to find those that match the desired service types. The matching <ResourceOffering> elements are returned in a <QueryResponse> element. A <QueryResponse> is returned even if no matching services were found, but in that case there are no <ResourceOffering> elements in the message.

The status of the query is also indicated in the <QueryResponse>. If an error occurred during the processing, the status "Failed" must be returned to let the phone know that something wrong has happened. Otherwise the status is returned as "OK".

For each <ResourceOffering> returned, a <Credentials> element may be included. This element contains specific credentials or tokens for the resource. It

must be possible to refer to the credentials by an identifier included in the element. This way, the <ResourceOffering> element that needs the credentials may refer to them by this identifier instead of including them in the actual <ResourceOffering> element.

STEP 3: Interaction with the web service provider

How the interaction with the WSP is done is dependent on how the WSP is implemented. However, Liberty pose some requirements on the capabilities of service providers that are allowed to call themselves ID-WSF WSPs. One such requirement is that the WSP must support the "WSP Common" profile defined in *Liberty ID-WSF 1.1 Static Conformance Requirements* [58]. This means, that it must be able to communicate over the Liberty ID-WSF SOAP Binding and support both message authentication and authentication of the client by use of TLS.

Step 3.1 Service request: Regardless of how the WSP is implemented, the mobile phone tries to send requests on behalf of the user according to the rules defined in the service description. Additionally, it includes the credentials it received from the DS. The nature of these credentials depend on the service implementation, but SAML assertions are often used. As was the case for step 2, these may be transferred in one of two ways; either the assertion itself is sent or only a reference to it is presented.

In the latter case the assertion is kept at the discovery service until the WSP claims it by presenting the reference to the DS. This kind of reference is called an artifact. When the artifact is used the burden of retrieving the complete assertion is on the service provider. Considering that we are focusing on the web service consumer and on the possibilities for running such an application on mobile phones, we have chosen not to use this method of presenting tokens. We want to show that our application is able to handle the entire assertion, without help from the other entities.

Step 3.2 Provide service: Liberty require the WSP to support service invocations that are done according to the rules specified in its service description and that include valid credentials from the DS. This means that if the WSP accepts the credentials presented and is able to fulfil the request, it has to provide the service to the user. How this latter task is performed is not in the scope of Liberty, but in our case the provided service will be in the form of a SOAP message that is returned to the phone. This message will contain the information the phone requested from the service provider.

6.3 Related work

Currently, the interest of offering services on mobile phones is fairly high. As the Liberty ID-WSF specification can help achieve this, there is already a lot of work concerning the subject of using Liberty on Mobile phones going on. *Liberty Alliance - Conformant Products* [59] lists the products that have passed the conformance tests designed by Liberty, but not all work falls into this category. Below we describe the products and frameworks we found when we searched for relevant projects. Then we give a short evaluation of these, with emphasis on whether they are able to fulfil the roles we presented in chapter 2.

EPOK TRUSTED DATA EXCHANGE (TDX) SERVER

The TDX server is a platform to control identities and provide secure access to resources in an environment where the trust domains crosses business relationships. According to *Liberty Alliance - Conformant Products* [59], the TDX is Liberty conformant for ID-WSF with implementations of DS, ID-WSF IdP, LUAD-WSC and LUAD-WSP. However, Liberty ID-WSF is only provided through an add-on module.

NOKIA WEB SERVICES FRAMEWORK

The Nokia Web Services Framework consists of several APIs that are included in many of Nokia's smart phones. These APIs provide the phones with the capabilities needed to interact with web services offered through the Internet, but they need applications to utilise these capabilities.

According to *Introduction To Web Services In Nokia Devices* [60] the framework allows for the creation of both LUAD-WSCs and LUAD-WSPs on mobile phones. As listed in *Liberty Alliance - Conformant Products* [59], only a LUAD-WSC implementation based on this framework has been submitted for testing and found conformant by Liberty.

SUN JAVA SYSTEM ACCESS MANAGER

The Sun Java System Access Manager is a software suite consisting of both server applications and client APIs. According to *Liberty Alliance - Conformant Products* [59] the Access Manager is conformant with all the profiles of Liberty ID-WSF. This means that it can be used to implement any of the roles that are defined in Liberty ID-WSF. Amongst these are the LUAD-WSC, IdP, DS and WSP that we plan to use in our prototype.

The Access Manager is mostly used in the role as identity provider and discovery service. This way of using it is illustrated in *Deploying Mobile Web Services using Liberty Alliance's Identity Web Services Framework (ID-WSF)* [61], where it is combined with the Nokia Web Services Framework.

OMA WEB SERVICES NETWORK IDENTITY

The Open Mobile Alliance (OMA) have developed a specification on how to enable mobile devices to interact with web services. As an addition to this specification, they have developed the OMA Web Services Network Identity Specification (OWSER NI), which is based on Liberty's ID-FF and ID-WSF specifications. The OWSER NI is not an actual product, but rather an alternative specification. However, as it is relevant to the topic of Liberty on mobile phones, we give a short description of it anyway.

The OWSER NI specification and Liberty's specifications are very similar. The main difference is the fact that OMA's specification is only targeted towards mobile phones, while Liberty's specifications are meant for use with a broader range of devices. This way OWSER NI may be easier to relate to for a mobile application developer than Liberty. However, according to OMA's specification page on OWSER NI [62] it is only a candidate, meaning that it is at an early stage and may be subject to change.

SUMMARY

Table 6.1 summarises how the products we found during our search are conformant with the roles we require. Of these, the Sun Java System Access Manager comes closest to fulfilling our needs. It is also the most complete product, as it covers all profiles of Liberty ID-WSF. However, its documentation is enormous and we found it quite difficult to get an overview of how it should be used.

	LUAD- WSC	IdP	DS	WSP
EPOK TDX Server	x	x	x	
Nokia Web Services Framework	x			
Sun Java System Access Manager	x	x	x	x
OMA OWSER NI				

Table 6.1: Summary of Liberty conformance

Another interesting product is the Nokia Web Services Framework. The inclusion of Liberty support in the mobile phones' firmware means that developers will not have to create the Liberty specific message handling classes themselves. Then they can focus on the application development and hopefully this will result in more and better Liberty-enabled applications. However, as the Nokia Web Services Framework is developed by Nokia and only included in some of their smart phones, it has a long way to go before it can show its true potential.

The OMA OWSER NI could probably be used to fulfil our needed roles. Still, it cannot claim to support any of the Liberty roles since it has not been tested by Liberty. Our assignment text specifically specifies the use of Liberty, but it should be noted that this could be a potential alternative when working with mobile phones.

Part III

Prototype

7 Requirements specification

7.1 Introduction

This Software Requirements Specification (SRS) uses the *IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications* [63] as a guideline and includes the sections that are relevant for this project. This SRS is, in relation to the rest of this document, redundant for some parts so that it can be read independently of the entire master thesis. This was done to help those who plan to further develop the application so that they can read only the parts concerning the development.

PURPOSE

The purpose of this document is to state the requirements for the prototype which is a mobile application that provides access to governmental services. This specification aims to describe the structure of the system parts, define the elements that affects this development and list specific demands. The intended audience of this document is primarily Kantega employees that are interested in the subject. This is because they have given us this assignment and asked us to explore it. Since this report is confidential for one year after completion, Kantega will be able to restrict access to the discoveries in that period. After that, anybody wanting to develop an mGovernment application will be potential readers.

SCOPE

Our main product is a mobile application called MyMobileSite, but this will also need two supporting systems. These are the server with the identity provider (which we have called "mSecure Identity") and the web service provider (which is called "Educational Loan Fund"). The supporting system parts will be set up to help us prove the validity of our main application and are therefore only used as a mean to achieve this goal. Since they are important for proving our point, these entities will also be treated as part of the system in this SRS.

As defined in the assignment text, we should have time to both identify requirements and make a design for the application with strong authentication. However, we are not required to implement this part of the system. Therefore, we have chosen to divide the requirements into sections with and without strong authentication. This is to ensure that implementation and testing are done correctly for the part we are required to implement.

The objective of the project is to show that we can implement a mobile phone application using the Liberty framework which is secure enough for dealing with the personal information presented in a governmental service. The goal of the project is therefore to

provide an actual proof-of-concept, which means a realisation of the project objective. The benefit of this application is the easy and convenient service access it can provide to its users.

DEFINITIONS AND REFERENCES

This paragraph will provide the reader with an understanding of how we define and will use the most essential concepts or entities in this SRS. Any references in this document will point to the reference list of the complete report.

AS: This is the part of the IdP that provides authentication of the user.

DS: This is the part of the IdP that provides lookup of services connected to the user's identity.

IdP: This is the server which authenticates the user and provides lookup of services connected to the user's identity.

GUI: In our case the term GUI will be used when we talk about the user interface presented in our mobile application. If any of the other entities have GUIs these will not be used and therefore not discussed here.

LUAD-WSC: This is the mobile phone in our scenario.

Mobile application / MIDlet: These are all terms we use to talk about the program or application which is running on the mobile phone.

Product: The term product will be used interchangeably with the term system.

Prototype: When we use the word prototype we talk of the mobile application which is the most important part and what we are expected to make.

System: When we use the term system we think of the application, WSP and IdP working together to produce the expected result for the user.

WSP: This is the server which provides a personal service based on a token which confirms the user's identity.

OVERVIEW

The SRS contains two sections after this introduction. The first is an overall description of the prototype. This gives some information on the background which influences the requirements. The other section contains the complete list of specific requirements for the system, which is divided into functional, non-functional and those describing what is required to extend the prototype to include strong authentication.

7.2 Overall description

This section provides a brief description of the product for which we will specify requirements. This includes examining how the product relates to other systems, the functions it will provide, the future/potential users, the constraints that influence the development and a discussion of requirements that will be delayed until a future version.

PRODUCT PERSPECTIVE

The product is an independent system which is not developed to be used with the Norwegian government's other electronic services, but its purpose is to illustrate how to add mobile services to existing technologies like "MinSide". What is important to note is that the first version of "MinSide" does not support Liberty and consequently our application could not be used together with the providers in this system. The next step for "MinSide" will be to implement Liberty and when this happens our prototype (which conforms to the Liberty ID-WSF LUAD-WSC profile) can easily be supported. Therefore, we consider their requirements during development so that our application is valid for use with only minor modifications.

PRODUCT FUNCTIONS

The main function of the system is to provide citizens access to information about themselves that is stored and offered by the government. This must be done in a secure way to respect the privacy of the individuals. To accomplish this goal, the system is divided into three separate components, as shown in figure 7.1. Below, we describe the task that each component performs in order to allow the complete system to offer its main function.

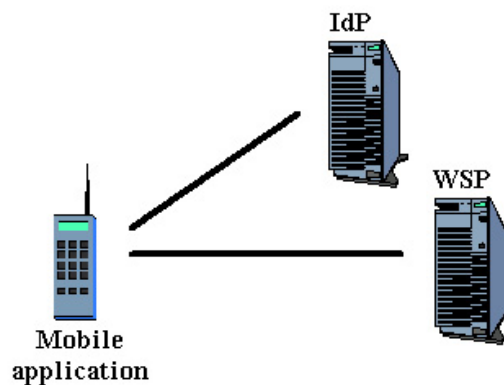


Figure 7.1: System overview

Identity provider: The IdP is responsible for checking that the user's login credentials are correct and finding providers. Then, it must provide the calling application with appropriate proof of the user's identity that can be presented to the WSP. Furthermore, it is responsible for storing the contact information for the WSP.

This information may only be delivered to applications that are able to present a valid user login.

Web service provider: The WSP is the component that stores the information which the user wants. To protect the user's privacy, no application can access it without having appropriate proof from the IdP that it acts on behalf of the user.

Mobile application: The mobile application is the only part of the system that will be in direct contact with the user. It is responsible for contacting the other components in the appropriate order and fashion, and presenting the results to the user.

USER CHARACTERISTICS

This requirements specification specifies the demands for the prototype to serve as a proof-of-concept, while still being realistic. This means, it should not require a lot of work and changes to use in a real setting a long with other governmental electronic services. Since the prototype is fairly close to what it would be like in use, we have chosen to provide a description of people which will use the deployed system.

- The user will be a Norwegian citizen or someone with a Norwegian residential permit.
- The user will have access to a mobile phone.
- The users' experience with electronic systems will range from novice to expert level.

The fact that people with so many different levels of experience, knowledge and confidence of using technical equipment will be potential users of this kind of system makes the development harder. Although most people today have mobile phones - most of them with advanced capabilities like Java - it is still a barrier for people to start using the advanced features. Therefore, we can assume some knowledge of the users, but we should also simplify the transition by providing a system that anyone can use.

As the application is to be used not only by Norwegian citizens, but also persons that have a residential permit, we cannot assume that all users will understand Norwegian. Most of the users will be able to understand English, but there are some people who do not understand any of these languages. These are usually dependent on interpreters in all their dealings with the government today. Therefore, making an application which can easily be extended to support more languages would be preferable.

CONSTRAINTS

There are some aspects of this field that limit our options when developing this prototype. These are given below.

- For this system to be valid for use by the government, the parts of it that use PKI must follow the rules given in *Requirements specification for PKI for the public sector* [34] and *Leveranse oppgave 1 Anbefalte sertifikatprofiler for personsertifikater og virksomhetssertifikater*³⁰ [36].
- To reduce the work needed to adapt existing systems so they can be used with the mobile application *Implementasjonsguide Registerinformasjon Min Side*³¹ [38] should be respected, especially when designing the interface between the application and WSP.
- The mobile industry has a lot of different standards making it very difficult to find a solution that works on all mobile phones. Therefore, we have to formulate the requirements so they allow us to support as many different phones as possible, while not compromising the other requirements.
- Mobile phones have limited storage, memory, screen size and input capabilities. This means that we have to make requirements for the the size and content of the text that is presented to the user as well as the size of the application.
- The procedure of user consent should be done through "MinSide" before using the mobile application. Thus, we will not address this issue.

APPORTIONING OF REQUIREMENTS

There are two features that we will present below which will be delayed until future implementations of this application.

Strong authentication: Strong authentication is a feature that does not need to be implemented according to the assignment text, but we will still elicit requirements for this functionality.

Single sign-on: We will not identify requirements for enabling single sign-on between different WSPs. This is because this functionality is not necessary for our prototype to function. Thus, we intend to focus on the more important aspects, such as being able to use the basic functionality of our WSP. Specifically, not supporting SSO means that we will not address issues of credentials time-out and caching of messages.

³⁰English: Delivery task 1 Recommended certificate profiles for personal and enterprise certificates

³¹English: Implementation guide for register information MyPage

7.3 Specific requirements

This section contains the requirements for the entire system. Each requirement has a unique identification number which allows us to refer to it through the rest of the report. All requirements are formulated to make them as unambiguous and complete as possible and to ensure that they are testable.

7.3.1 Functional requirements

This section contains all of the functional requirements which will describe what the system will do. The requirements have been divided according to the three system entities. All the requirements are equally important in order to have a well functioning system. Thus, we will only use the verb shall as the modal auxiliary in the requirement texts.

Each of the system parts must fulfil a set of requirements defined in *Liberty ID-WSF Static Conformance Requirements* [58]. For the IdP this means that the message exchange is predefined for both the AS and DS. However, the mobile application and WSP are not that affected by the requirements. Mostly they are required to put Liberty defined contents in the headers of the messages, but when the mobile application communicates with the IdP it too has its message exchange specified by Liberty.

IdP

- F-1 The identity provider shall be the host of both the authentication service and the discovery service.
- F-2 The discovery service shall conform to the DS profile of *Liberty ID-WSF Static Conformance Requirements* [58].
- F-3 The identity provider shall conform to the ID-WSF IdP profile of *Liberty ID-WSF Static Conformance Requirements* [58].
- F-4 The authentication service shall be able to authenticate the user by use of a username and password.
- F-5 The discovery service shall be able to issue an authorisation token based on the user's federated identity between the WSP and IdP.
- F-6 The identity provider shall offer its services through HTTPS and identify itself by use of an X.509v3 server certificate with a key based on RSA and a key length of 1024 bits.

The most important requirements to the IdP are those that concern conformance to the Liberty profiles (F-2 and F-3). This is to ensure that the system, in which we use our mobile application, provides a realistic setting. Otherwise, the proof of concept would not accomplish the goals we have set.

F-6 requires the identity provider to be reachable through HTTPS. Basically, this means that we want to secure the channel between the IdP and the mobile phone. This is done to keep the messages safe, while not having to apply message encryption in our application. As this is an eGovernment IdP, it has to respect the requirements set by the *Requirements specification for PKI for the public sector* [34]. Thus, the key is based on RSA and its length is set to 1024 bits.

WSP

- F-7 The web service provider shall conform to the ID-FF SP WSP profile of *Liberty ID-WSF Static Conformance Requirements* [58].
- F-8 The web service provider shall be able to accept assertions coming from the identity provider.
- F-9 The web service provider shall offer register services over HTTPS and identify itself by use of an X.509v3 server certificate with a key based on RSA and a key length of 1024 bits.
- F-10 The web service provider shall be able to answer requests that conform to the specification given in its WSDL.
- F-11 The web service provider shall structure its results according to our XML Schema.
- F-12 The web service provider shall use our XML Schema to structure error messages.
- F-13 The web service provider shall accept requests structured according to our XML Schema.
- F-14 The web service provider shall return all the register information affiliated with the user identified in the request.
- F-15 The web service provider shall if possible return information in the preferred language specified in the request.
- F-16 The web service provider shall use a time-out value and return an error message to the requester if it is exceeded.
- F-17 The web service provider shall limit the length of each service's aggregated values to 200 characters.

As was the case for the IdP, F-7 is required with the purpose of providing a realistic setting for the mobile application. However, the Liberty requirements to the WSP are not as extensive as those to the IdP. The most important is the one stating that the WSP must be able to send and receive messages over the Liberty SOAP Binding. The profile used in F-7 is combined for SP and WSP, but we will only use the ID-WSF (WSP) part of the profile. F-9 is also included for the same reason as in the IdP list of requirements.

Requirements F-11, F-12 and F-13 concern the interface for service invocation at the WSP. The schemas mentioned are created to achieve a well defined communication protocol between the mobile phone and the WSP. This will both ease the construction of requests and presentation of responses. Furthermore, it will make it easier to add more WSPs later on, as all they have to do is to "speak" the same language as our WSP. The requirements are all inspired by *Implementasjonsguide Registerinformasjon Min Side* [38], but as the results are to be presented on a mobile phone instead of a web browser, the schemas will be somewhat different.

Requirement F-14 is included to avoid having to perform several requests to the same provider. This will reduce the amount of data sent and received by the mobile phone and consequently the cost for the user. The reason for this is that the request and response elements needed will be smaller than the header element of the SOAP messages. Therefore, the extra amount of data received by including all the information in one message will be smaller than the extra amount of data sent and received if the user needed more information than what was contained in the first response. This theory is based on the assumption that the information returned by the WSP is quite small as required by F-17. This, in turn, is added to make sure it is easy to read on a small screen.

Requirements F-15 and F-16 are also inspired by the implementation guide. However, these are a lot more similar to the ones in the guide than the previously discussed. We included these requirements to reduce the extent and amount of alterations needed to existing services in order to offer them to mobile phones.

MOBILE APPLICATION

F-18 The mobile phone application shall conform to the LUAD-WSC profile of *Liberty ID-WSF Static Conformance Requirements* [58].

F-19 The application shall be able to communicate with the WSP with the information that is contained in the service offering obtained from the DS.

F-20 The application shall use an HTTPS connection to communicate with the WSP and IdP.

F-21 The application shall send the authentication credentials in the first authentication request.

F-22 The application shall trust the root certificate that has been used to sign the IdP's certificate.

F-23 The application shall trust the root certificate that has been used to sign the WSP's certificate.

F-24 The application shall allow the user to store his username which can be changed.

F-25 The application shall allow the user to change language settings.

- F-26 The application shall show a relevant error message when receiving a message that is structured according to our XML Schema.
- F-27 The application shall present the result by following the structure of our XML Schema.
- F-28 The application shall structure its requests to the WSP according to our XML Schema.
- F-29 The application shall attempt to parse XML documents without validating them first.
- F-30 The application shall display an error message if an XML document cannot be parsed.
- F-31 The application shall display an error message if it receives a message that indicates a fault.
- F-32 The application shall give the user the option to try again or exit the application when displaying an error message.
- F-33 The application shall be able to access other services providers which fulfil our requirements to the WSP.
- F-34 The application shall show a screen with relevant text informing the user of what is happening while waiting for response.
- F-35 The application shall require username and password to be filled in before passing values on to the IdP.

Requirement F-21 is set to reduce the amount of messages transmitted between the IdP and the mobile application. Since the mobile application will only support one type of authentication, the credentials might as well be included in the first message. Requirements F-22 and F-23 are essential to allow the phone to establish secure connections to the other system entities. In a realistic setting, the certificates would be signed by a recognised certificate authority (CA). This CA would most likely already be included in the phone's list of trusted CAs, but if it is not it has to be downloaded to the phone manually. This will also be the case during the testing of our application.

Requirement F-24 is mostly included for the convenience of the user. If he does not have to enter the username manually each time, the application will be much easier to use. As the mobile phone only belongs to one person, there is no need for storing more than one username. However, if the user wants to lend his phone to a friend or for some reason has changed his credentials, requirement F-25 allows the stored username to be changed. Additionally, F-25 requires the application to allow the user to change the language used in the application. This is done to provide a flexible application with a larger group of potential users.

Requirements F-26 to F-29 and F-33 all concern the interface of the mobile application and how it will process the messages it receives from the IdP and WSP. It is important

that this is well defined before the implementation starts, so that the different parts of the system are able to communicate. F-29 was introduced to save the mobile application a lot of work. By skipping the validation of the SOAP messages the application may fail half way through the parsing of a SOAP message if it is invalid, but since we assume that most messages received will be valid this approach is preferable.

Requirements F-30, F-31, F-32, F-34 and F-35 are the rules that define the functionality of the user interface. These are all set to ensure that the application give the user relevant feedback and the necessary options at all times. F-35 will also help keep redundant network traffic down to a minimum. If the user was allowed to start the login without filling in the username and password, the application would have to send a SASL request that was destined to be rejected, as the username and password was missing.

7.3.2 Non-functional requirements

This section lists the requirements that do not directly address the function of the system, but instead concern the qualities which the system should possess.

NF-1 The application shall be developed in J2ME using MIDP 2.0.

NF-2 The application shall not need any optional packages.

NF-3 The application shall not exceed 150kb in size.

NF-4 The application shall support both Norwegian and English.

NF-5 The application shall use the phone language as default and English if this is not possible.

NF-6 The Java classes shall be commented with Javadoc.

NF-7 The amount of work required to enable the application to support a new language shall not exceed 8 man-hours.

NF-8 The application shall have a consistent placement of selection keys on all screens.

NF-9 The application shall minimise the input needed from the user.

As we have said from the beginning, we are going to develop a Java application that requires MIDP 2.0, as specified in NF-1. Some mobiles have built-in support for additional packages, but since we want the majority of citizens to have access to this service we will do it without using any packages (NF-2). It should be noted that in the future it is likely that most phones will come equipped with a large selection of the available packages.

NF-5 is not a very important requirement for the prototype, but is included since we want it to be easy to add new languages. This requirement is a feature adding usability when several languages are added, but it requires little effort to make and is therefore included now.

It should be noted that NF-9 is not possible to test, but has been added to ensure that minimal user input is given a great deal of thought during the design. It is not possible to clarify it at present time since there are some uncertainties as to how things are done.

7.3.3 Requirements for strong authentication

As was decided upon in chapter 5, the strong authentication mechanism will consist of PKI-based. The actual authentication will be done by the following four steps:

1. The server sends a challenge to the mobile phone.
2. By entering his code, the user authorises the use of the secret key to sign the challenge.
3. The mobile phone sends the signature and the digital certificate to the server.
4. The server validates the signature and the certificate and if this goes well it trusts that the user has the identity asserted by the certificate.

This section describes what additional requirements we have to make to add support for strong authentication on the mobile phone. As the strong authentication is meant as an extension to the previously defined system with weak authentication, all requirements to that system will have to be fulfilled as well.

There is only one exception to this rule. Requirement F-21, will not be possible to fulfil in all cases. This comes from the fact that the authentication mechanism chosen for strong authentication is dependent on a challenge from the authentication service. The challenge is supposed to be signed and sent back to the service so that it may determine whether the signature is correct. In some protocols, the mobile phone may be allowed to make up its own challenge and then send it to the server along with the signature, but as the actual protocol to be used will be decided during the design, we must take the precaution of relaxing the requirement.

- SA-1 The digital certificate shall be revocable.
- SA-2 The digital certificate shall follow the recommendations of *Leveranse oppgave 1 Anbefalte sertifikatprofiler for personsertifikater og virksomhetsertifikater* [36].
- SA-3 The secret key used in the authentication shall be based on RSA with a key length of at least 1024 bits.
- SA-4 The secret key used in the authentication shall be generated on the phone itself and never be sent over a network.
- SA-5 The secret key used in the authentication shall not be stored in plaintext.
- SA-6 The access to and use of the secret key shall require authentication by two factors, one of which shall be proof of possession.
- SA-7 The secret key shall become useless if the authentication to access it fails three times in a row.
- SA-8 If the authentication to access the secret key fails, the user shall be notified of how many tries he has left.
- SA-9 The web service provider shall only return results that are appropriate to the specified authentication strength.
- SA-10 The digital certificate shall be signed by a widely trusted certificate authority by use of a certificate signing service.
- SA-11 The certificate signing service shall verify the user's existence in the Norwegian Central Register of Persons and that the national identity number corresponds with the name before signing the certificate.
- SA-12 Certificates of the certificate class "High" shall only be signed after personal attendance and authentication to appropriate authorities.
- SA-13 Certificates of the certificate class "Standard" shall be signed by a method giving equivalent certainty of the user's identity as if it was sent to him by mail to his registered address.

As was the case with weak authentication, the fact that the application is meant for use with eGovernment services means extra requirements. Because the authentication mechanism is based on PKI, we need to respect the *Requirements specification for PKI for the public sector* [34]. However, not all of the requirements in the *Requirements specification for PKI for the public sector* are relevant for our case. The requirements based on the requirements specification are SA-1 to SA-6 and SA-11 to SA-13.

The *Requirements specification for PKI for the public sector* was written with ordinary computers in mind and does not take into consideration the fact that mobile phones are personal devices. When it comes to requirement SA-6, this means that the fact that the

user has the phone in his hand will be sufficient proof of possession. The requirement has been kept close to its original wording anyway to show that it has not been weakened.

Requirement SA-7 is included in case the mobile phone is stolen. Its intention is to prevent misuse if the phone is stolen. If the user himself enters the wrong code accidentally, requirement SA-8 ensures that he is aware of the number of tries he has left.

Requirement SA-10 ensures that, although the certificate is generated on the phone, it will be activated by a signature from a widely trusted CA, so that the identity provider will trust its contents's validity. A "widely trusted CA" is a certificate authority trusted by many systems, including the identity provider.

Leveranse oppgave 1 Anbefalte sertifikatprofiler for personsertifikater og virksomhetssertifikater defines two different classes of personal certificates; "High" and "Standard". Certificates of the class "High" give more privileges to the user than certificates of the class "Standard". Because of this the requirements to the process of acquiring certificates of the class high are stricter. Consequently, both SA-12 and SA-13 are needed to cover both certificate classes.

Requirements SA-11 through SA-13 are based on the *Requirements specification for PKI for the public sector*, but they are a bit different from the ones included there. The reason for this is that our certificates are generated on the phone and then sent to a certificate signing service to be signed, instead of being issued by a certificate authority with the signature already included.

8 Design

8.1 Introduction

Since we rely on three different entities (mobile, IdP and WSP) for this prototype to work, we have chosen to separate these when presenting the design. The design of the mobile application is the most important, and it is therefore solved first with a higher level of detail than the rest. Furthermore, as both the IdP and the WSP are only intended to help the mobile application to do its work, they are designed with a simple implementation in mind. Thus, they will not feature functionality beyond that needed by the mobile.

8.2 Design issues

During the design there are a number of issues that have to be taken into account. These mostly concern the mobile application, but may also be relevant to the design of the other system entities. Below we present the issues and give a description of the problems involved.

DATA PRESENTATION

To present data in a meaningful way on a small screen is a challenge. The width of the screen on a mobile phone is limited, and mobiles have quite slow and cumbersome scrolling functionality. Thus, any data presented on such screens should be as compact as possible without losing any of the information it represents.

LANGUAGE SUPPORT

To make the application easy to use for as many people as possible we decided to support both Norwegian and English. However, we must find a way to do this that will allow for new languages to be added easily. Furthermore, we must have the performance in mind, making sure that the support of different languages does not slow the application down with an endless array of if-clauses.

TIME DEVIATIONS

When the mobile application sends SOAP messages it has to include a timestamp that will be checked by the receiving party. This means that, if the time setting on the phone deviates too much from that of the systems with which it communicates, errors may occur. However, as the mobile phone is a portable device, people usually bring them

along when they travel. This means that as the user enters different time zones he has to change the time setting on his phone. If he then tries to use the mobile application, he may be denied access to the services.

Although most phones today support different time zones, not all users know how to use this functionality. Most users will be content if the phone shows the correct time, and may not care about the time zone being wrong. However, when Java compares date and time values, time zones are taken into account. This may result in that the user will receive error messages, even though the clock on his phone is seemingly correct. There is not much we can do about this problem, other than allowing for a reasonable amount of time deviations and displaying descriptive error messages if time comparisons fail.

SECURE RANDOM

As mentioned in chapter 3, there are some problems with the default random number generator of MIDP. These problems only concern the mobile phone, as it is the only system entity that is based on MIDP. The random number generator uses only the system's time in milliseconds at the time of instantiation as the random seed, which makes it too easy to guess.

In the case where the system only features weak authentication, the random number generator will only be used for two things; message identifiers and generating session keys for the HTTPS connection. As we will explain in the next section, we cannot do anything about the random number generator used in the HTTPS connection. Furthermore, the message identifiers are not so critical to the overall security. Therefore, we will not create a more secure random number generator for now, but we will revisit the issue during our incorporation of strong authentication in the design.

SECURE TRANSPORT

Chapter 3 also mentioned that the HTTPS support in MIDP is a bit too weak. This is both due to the fact that it is based on the poor random number generator of MIDP and that it does not check whether the server certificate has been revoked. The MIDP classes that handle the HTTPS connection does not have any methods that allow specifying the key pair manually or improving the random number generator in any way. Therefore, we will just have to live with this part of the problem, as long as we are unwilling to encrypt the messages before sending them. However, there is no point in creating a stronger protection for the messages when the data that are transmitted only require weak authentication anyway.

The second problem of the HTTPS connection can be dealt with by use of a method suggested by the *Wireless Application Protocol Public Key Infrastructure Definition* [64]. This is called the short-lived certificate model and the idea is that, although the key pair used by the server will have a long lifetime of for instance a year, the certificates will have a much shorter lifetime of for instance 24 hours. The server will then have to download a new certificate each day, but in return it is able to stop downloading certificates for a given key pair if it suspects that the key pair has been compromised. Then, any stolen

keys will only be valid for a few hours. To be certain that everybody gets the message, the certificates can be revoked as well.

8.3 Mobile application

To show how the mobile application should work we start by constructing UML package, class and sequence diagrams [65] of the system. By doing this, we aim to determine the smaller entities which the program should be comprised of and what each entity's responsibilities are. The sequence diagram is used to show the interaction between parts within the application. The state diagram introduces a normal execution of the program and the results the application should produce. After the main system description, we look at especially important entities and concepts.

8.3.1 System description

This section contains the diagrams defining the application's structure. There are four different types of diagrams which together should give a complete picture. A description of the notation for all diagrams in this section can be found in appendix B.

8.3.1.1 State diagram

The diagram in figure 8.1 shows what the user will see on the screen during the use of the prototype. Error handling has been removed from the diagram to make it easier to follow and will be described in the next paragraph. By showing the states of the program from the user's perspective we try to show the path which the user must pass through before accessing the wanted service. We have also specified for each window what the possible user actions are and where errors might occur.

In this aspect of the program, the error handling deals with what the user can do after an error occurs and the current state of processing. The user has two options which are to either exit the program or try again. For the user to try again, the program must decide how far it must go back. This implies checking for the type of error and from there making a decision of whether the user must enter information again or the program can simply return to the previous state and send the message again. A specific example is when an error happens during the password verification. If the error is the result of a message timing out, the program will go back to verifying password waiting screen and send the message again. If the error instead is due to an unknown password, the user will be sent back to the login window.

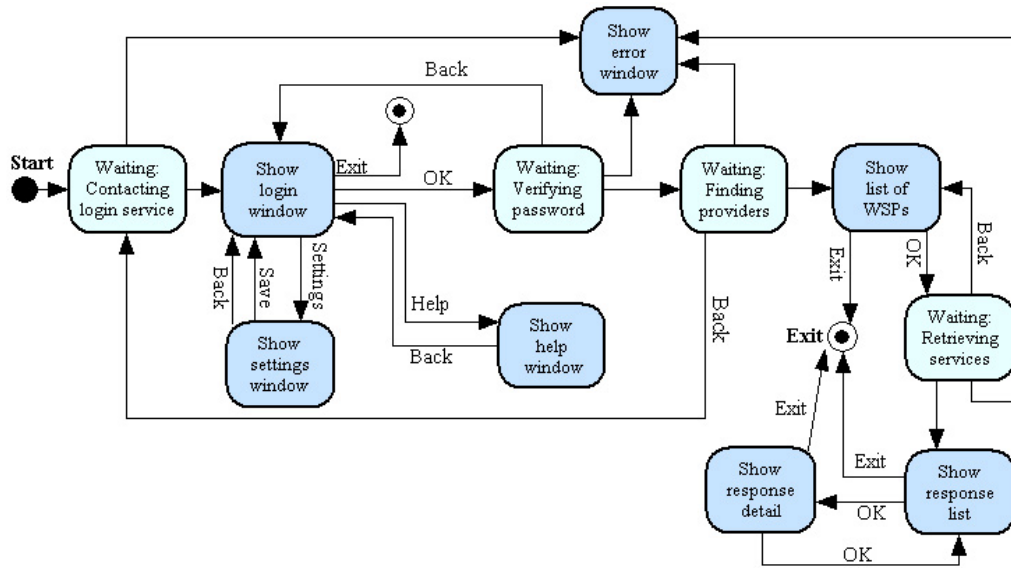


Figure 8.1: State diagram for the prototype

8.3.1.2 Package diagram

As shown in figure 8.2, the package diagram consists of two main packages which have all the classes. These separate between those we will develop ourselves, which are in the package called `mymobilesite`, and the rest which have `org` as the top package.

ORG

The `org` package contains all the open source libraries we will use in the application. These are various tools that will be used to parse XML and to create/send SOAP messages. We will use the package names as they are provided by the developers and we will not change anything within them.

MYMOBILESITE

This package contains all the classes which we will develop. These are then divided into two subpackages which are the controller and communication. The controller have all the classes that handle control information and user presentation. The communication package holds all the classes that are necessary for creating messages, transporting them and handling the results.

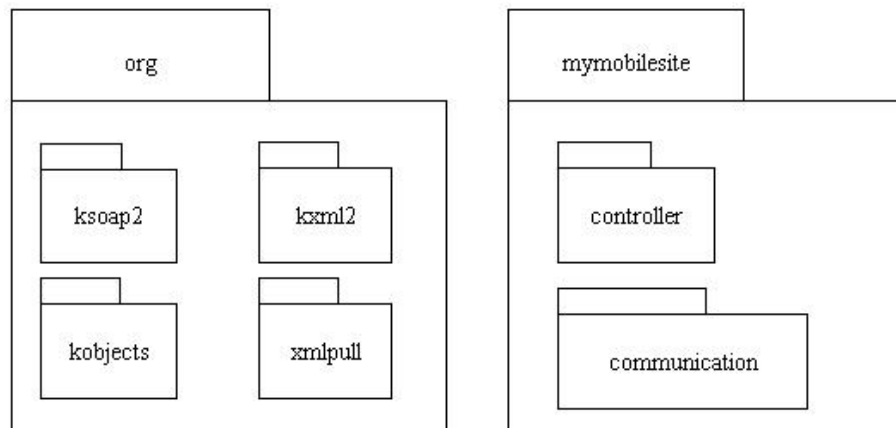


Figure 8.2: Package diagram for the prototype

8.3.1.3 Class diagram

The class diagrams in figure 8.3 (the controller package) and figure 8.4 (the communication package) show the structure of the mobile application. We had to split the classes into two diagrams to fit them into the page, but the classes in each diagram do not depend much on each other. The relationships between classes in separate diagrams are described in the explanation text for each diagram.

CONTROLLER PACKAGE

The main class in the entire mobile application is the `Controller` class which runs in the main thread. This class calls the GUI when it wants to change what the user sees and it calls `LibertyProtocol` to communicate with the services. It is notified by the GUI class when user events occur and it implements the `LibertyProtocolListener` to find out when results are ready to be displayed.

The abstract `Language` class is only used by the GUI for retrieving labels and texts in the correct language. It is the superclass of two specific language classes, one for each language we support. These will set the various variables in `Language` to the correct value according to their respective languages.

`Configurations` holds all the static variables that are accessed from several places in the code. `Service` is a simple class for storing results and is used by the GUI which displays them after the results have been retrieved by the `LibertyProtocolListener`.

COMMUNICATION PACKAGE

The main class in this part of the diagram is the `LibertyProtocol` which handles all communication by making instances of the correct `SOAPEnvelope` to send. The authentication service requires a special HTTPS connection which is made in the class `AuthTransport`. The messages are sent using SOAP messages and the creation of these is done by the messaging classes. The `SaslMessage` class is used for communication with the authentication service, the `DiscoveryQuery` class makes a request for the discovery service and the `WspRequest` class makes the request for the web service provider.

The resource offering gained from successful authentication and the offerings for services are stored in `ResourceOffering` objects. When the services are returned successfully, a `Service` object is created for each service and the results are returned to the controller.

If an error occurs somewhere within one of these classes, the class where it happened will create a `MessageException` using an error message in the correct language (obtained by using a static method in the `Language` class). The error will then be passed on by the `LibertyProtocol` to the `Controller`.

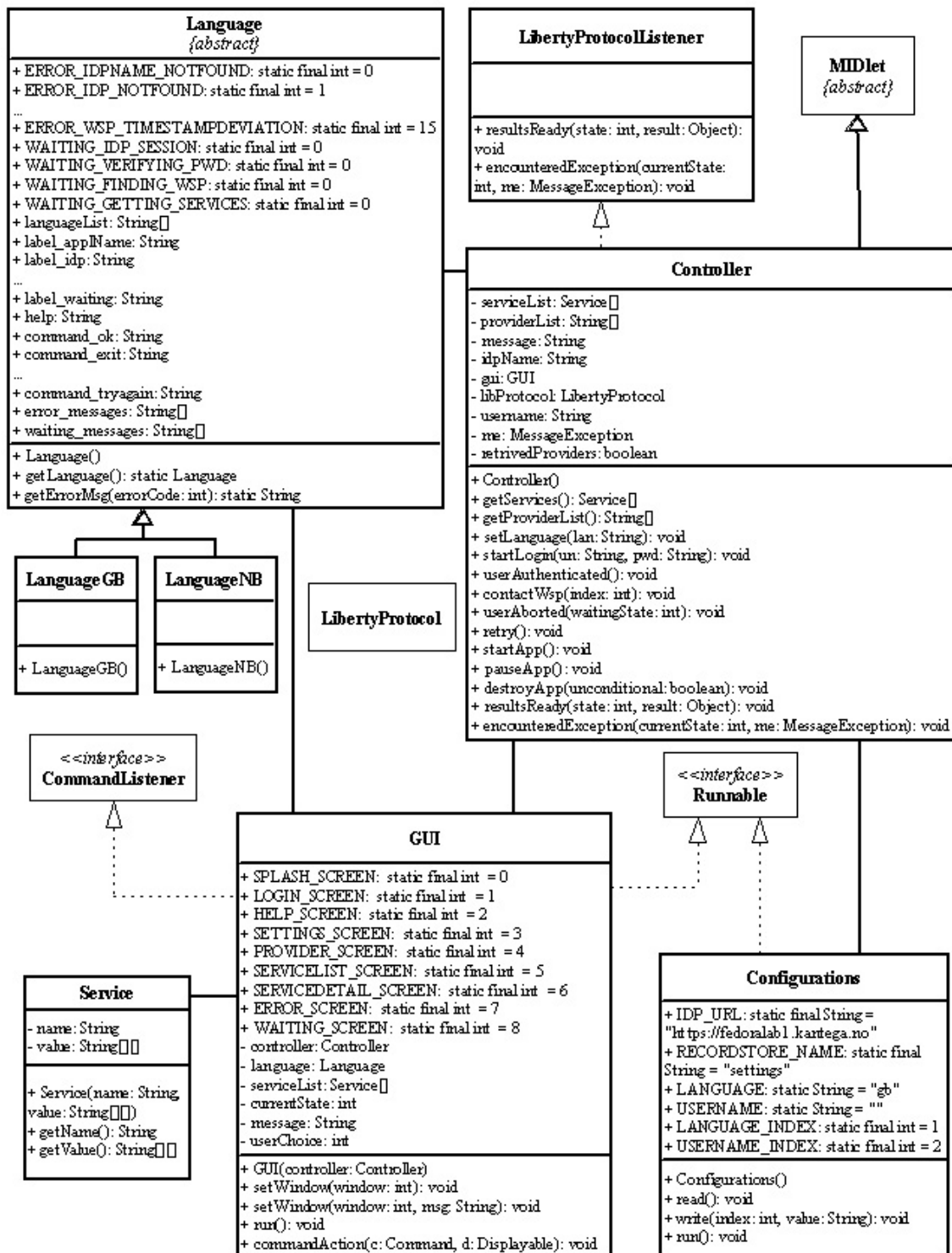


Figure 8.3: Class diagram for the prototype (controller package)

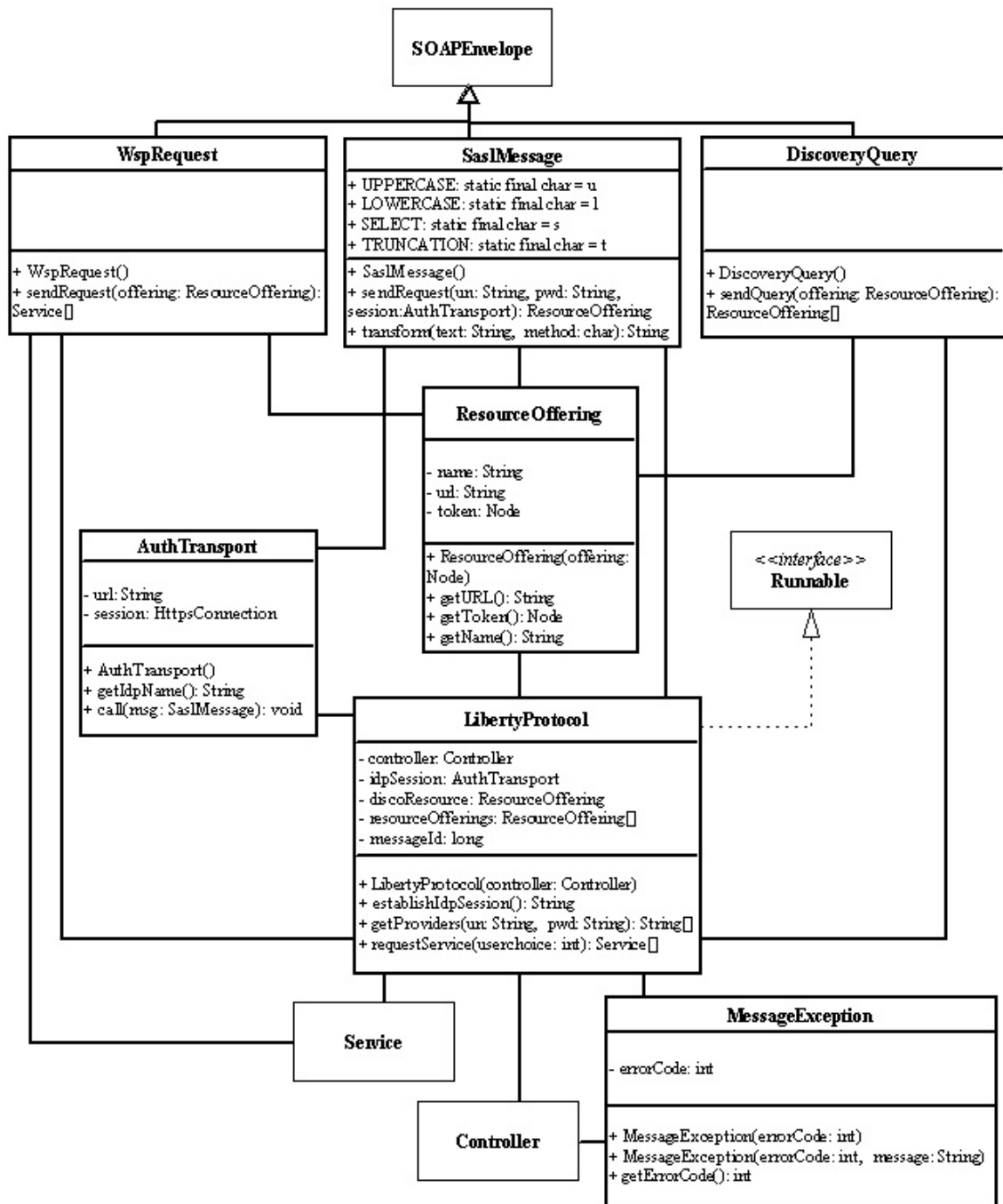


Figure 8.4: Class diagram for the prototype (communication package)

8.3.1.4 Sequence diagram

The sequence diagram³² in figure 8.5 and 8.6 shows how the classes cooperate in a normal execution, where errors and user settings are left out. Since the names of the methods that are called should be fairly self-explanatory we will not comment on the entire execution, but only point out states where the user can choose to perform other events or errors might occur.

A decision we made to simplify the diagram is to exclude internal method calls. This reduces the diagram size considerably, but should not affect the readability as these can be found in the class diagram. The internal method calls also include the `commandAction(Command, Displayable)` method which indicates the user action, but these actions are considered covered by the state diagram in section 8.3.1.1.

However, towards the end of the execution we included the `setWindow(servicedetail)` call, which is internal to the GUI class. This was done because the execution has reached a point where it only shows the information in the service results. At this stage all the method calls are either within the GUI class or between the GUI and Service classes. The inclusion of this method call in the diagram shows why the GUI retrieves the service values.

There are a several places in the diagram where the execution can differ from the standard path. Each of these are listed below with a description of what happens in each state:

GUI showing the login screen: The user could choose another item from the menu. If settings are changed we must update the file storing the language. If help is chosen, we show a simple window with a help text.

GUI showing the service detail screen: The user can go back and choose another service from the menu, in which case we must get the values for this service instead.

Any waiting window: An error can occur, and the user will get the option to try again or exit. The user can also choose to go back.

Any main GUI window: The user can choose exit and the program will end earlier than in the diagram.

³²To make the diagram fit into this report it had to be split in two. This means that, figure 8.5 shows the first part of the execution and figure 8.6 shows the second part, thus figure 8.5 should be examined first of the two. Each part only contains the classes that are active in that part, because the figures would be too tall for the page otherwise.

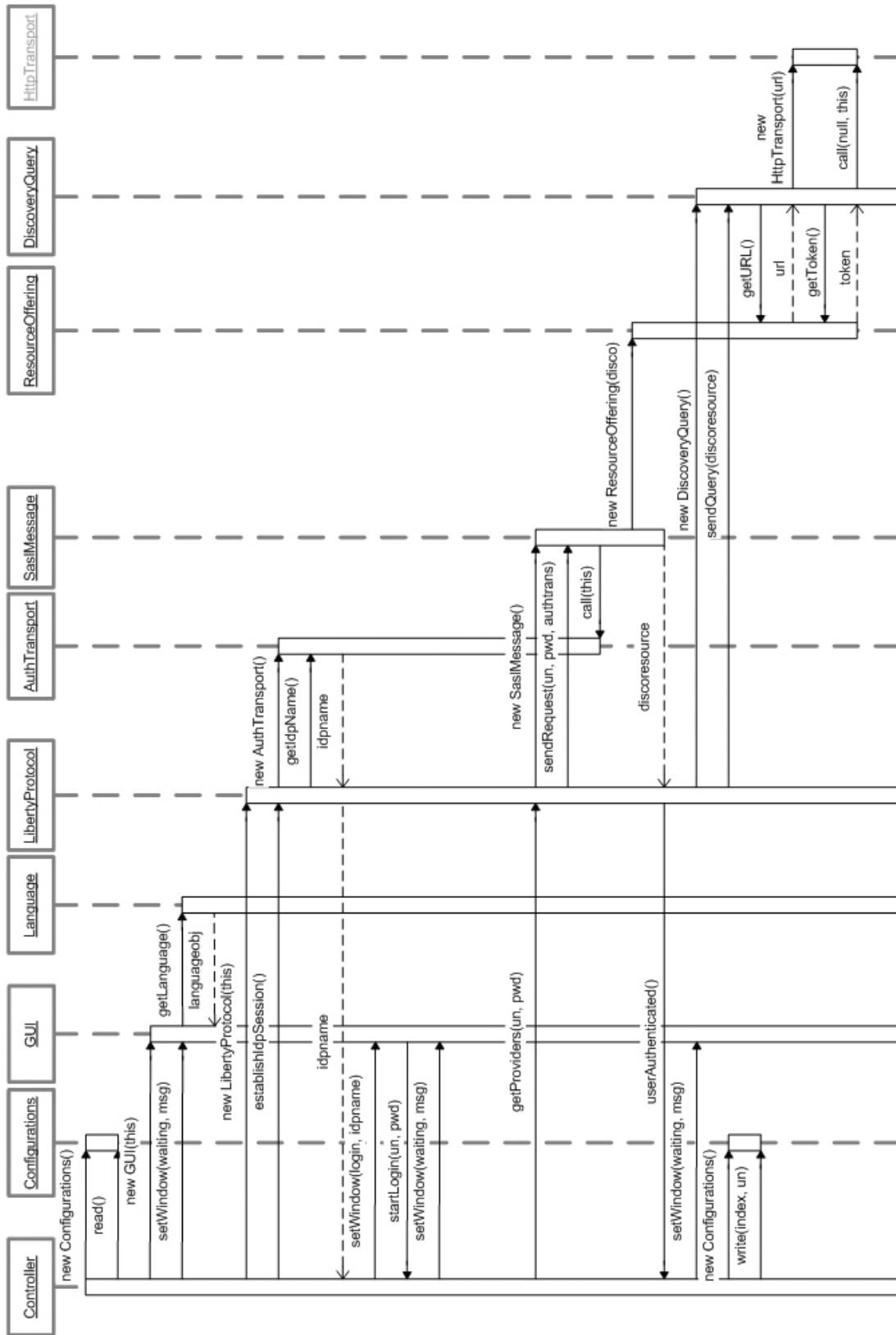


Figure 8.5: Sequence diagram for the prototype (part 1)

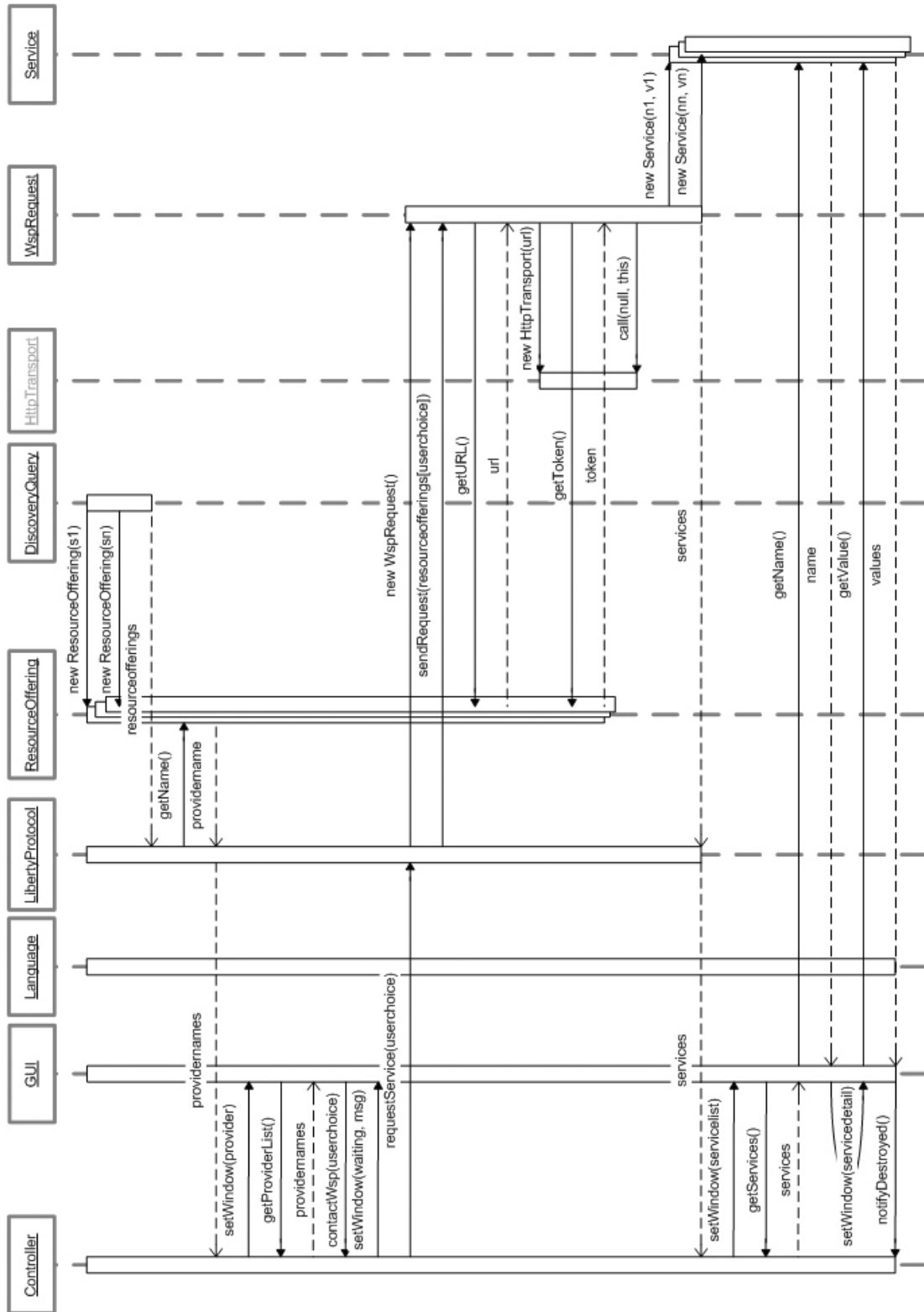


Figure 8.6: Sequence diagram for the prototype (part 2)

8.3.2 Major design entities

Not all parts of the system can be described good enough by use of the diagrams in the previous sections. Some design entities are in need of more thorough explanations. Most of these are loosely connected to the design issues introduced in section 8.2.

8.3.2.1 XML Schema

The communication between the system entities is defined through use of XML Schemas. Most of these are predefined by Liberty and SOAP, but some of them have to be designed by us. In these cases, the messages we design are placed in the `<body>` element of the SOAP messages, while the `<header>` element contains Liberty specified header information. The schemas we have designed ourselves can be found in appendix C.

The XML Schemas used in the communication between the mobile application and the WSP are designed with two main goals. The first goal is that the guidelines specified in *Implementasjonsguide Registerinformasjon Min Side* [38] should be followed as close as possible. This will ensure that the application is able to communicate with the existing services offered through the citizen portal, with only minimal modifications to the service providers. The second goal is to ease the computational burden on the mobile phone as much as possible. This implies that more data processing must be done by the service provider than was originally intended during the development of "MinSide".

To help achieve the first goal, we have chosen to use the Venetian Blind pattern³³ in the schema design, as specified in the implementation guide. Furthermore, we use `tns` as the namespace prefix for the `targetNamespace` and `xsd` as the namespace prefix for XML Schema. Additionally, the `targetNamespace` points to the document itself.

As specified in the implementation guide, we have designed three schemas for use with requests, responses and faults. However, contrary to the implementation guide we have chosen to use only one schema per function, regardless of which service provider the application is communicating with. This generalisation is done to help us achieve our second goal, namely to reduce the amount of work required by the phone. The following sections describe each of the schemas more thoroughly, together with more information about where and why they differ from the implementation guide recommendations.

THE REQUEST SCHEMA

In the original "MinSide" schema the requests contain the user's identification number, the user's preferred language and information about the strength of the authentication. However, considering that the mobile application will use Liberty, it will only be necessary to include the preferred language in the request. The rest of the information will be conveyed in the security token contained in the header.

³³The Venetian Blind pattern is an XML schema design pattern where the elements' content types are defined above the elements themselves.

THE RESPONSE SCHEMA

The response schema needs to be available to the WSP so that the WSP may use it to structure its responses. Originally, the providers that deliver services to the citizen portal specify their own response schemas accompanied by a provider specific style sheet³⁴. The style sheet is intended to transform the XML-based responses to HTML-based presentations of the data in order to present it to the user. This means that, the citizen portal can retrieve data from any WSP without having to modify its representations each time it has to present a new type of information.

However, as we mentioned earlier we intend to ease the computational burden on the mobile device. Considering that style sheet transformations can be a bit heavy for some devices, we decided to skip this in our design. The web service providers may still use style sheet transformations if they wish to "translate" from their original response schema to our simplified schema, but then it would be up to them to do the actual transformation.

The main element of the schema is the `<ServiceList>` element. It contains a list of the services that the provider has to offer the user. Each service is contained in a `<Service>` element and the `<ServiceList>` may contain as many `<Service>` elements as necessary. The `<Service>` element has an attribute called `name` that contains the name of the service. This name is intended to be displayed to the user when he is browsing the list of services returned by the WSP. Thus, the name should be descriptive of the information returned by that service.

Each `<Service>` element can have as many `<ValueElement>` tags as it likes. The `<ValueElement>` specifies the information that is to be presented the user when he chooses a service in the list. Each `<ValueElement>` has the attributes `label` and `value`. The `label` attribute is optional and is only intended to provide metadata, while the `value` attribute is obligatory and is supposed to contain the actual information required by the user. For instance if the value contains a number the label may contain the text "Sum". Then "Sum" will be displayed in front of the value, formatted in a way that distinguishes it from the value itself.

The distinction between label and value will also help the application to break the lines in the right places. If there is not enough room to place the entire value in line with the label, it usually looks better to break the line before the value than in the middle of the value, even if the value contains white spaces.

The `label` attribute is optional because it will not be needed to present all values. For instance, one could add an explanatory text beneath the values returned and then the label would be redundant.

³⁴A style sheet defines how different XML tags and their data should be presented the user.

THE FAULT SCHEMA

If an error occurs during the processing of the request at the WSP it will have to inform the mobile application about it. Therefore, we have constructed a separate fault schema, based on the one used for the citizen portal. The main difference between the schemas is the fact that the `<errorDetails>` element does not occur in our schema. Originally this element was intended to provide technical information that the citizen portal may use to find out what caused the error, but as the mobile application does not log anything, this element would not be of any use. However, if the `<errorDetails>` element was included anyway, it would not affect the processing of the fault message as the schema is not properly validated at the phone. Consequently, the phone will also accept error messages structured according to the fault schema for "MinSide".

As stated in the implementation guide the error codes for "MinSide" are "unknownId", "requestTimedOut", "serviceUnavailable", "notAuthorized" and "internalError". We have decided to keep these codes, and in section 8.5.2, we will explain how they are used.

8.3.2.2 SOAP client

To be able to talk to the other entities in the system the mobile application will need to implement a SOAP client that can construct and send SOAP messages. Consequently, it will also need XML parsing capabilities. One solution to this could be to make use of the optional package JSR-172 J2ME Web Services Specification. It would provide both SOAP and XML support, but then the application would only work on phones that included this package. Although most phones released so far this year has the JSR-172 package installed, it would mean that many will have to buy a new phone to use the application.

To avoid this problem we decided to use an open source library for SOAP and XML processing. When it comes to XML processing libraries, there are several to choose from. However, there is only one SOAP processing library that is widely used. It is called kSOAP [66] and the current implementation makes use of the XML processing library kXML [67]. Consequently, we will also use the this library.

Both kSOAP and kXML are available in their first and second versions. The first versions are officially deprecated, but are still available for download. The second versions are under constant development and may still contain undiscovered bugs. However, the second versions have much less restrictive licenses, stating that the developer can do whatever he likes with the code as long as the copyright information is kept intact. As a contrast, the first versions require the source code to be publicly available if the libraries are used. In the end, the licenses were the reason why we chose to use kSOAP2 and kXML2 instead of their first versions.

The most important classes of kSOAP are the `SoapEnvelope` and the `HttpTransport`. The `SoapEnvelope` is responsible for both constructing the outgoing SOAP requests and receiving the incoming SOAP responses. The `HttpTransport` is used to send the

requests and receive the responses over an HTTP connection. To secure the communication the application will use an `HttpsConnection` instead of an `HttpConnection`. Luckily, the `HttpsConnection` extends the `HttpConnection` and consequently we do not have to change anything in the `HttpTransport` class to use HTTPS instead.

The only aspect we could find that counted against the use of kSOAP was the fact that many of its classes has a lot of public variables. This could cause an inconsistency problem if a lot of different classes and threads used the same objects. However, as our own classes are the only ones that will use the various kSOAP objects, we will just have to be extra careful not to cause such problems.

8.3.2.3 Liberty messages

To be able to construct Liberty enabled messages, the application will need to add something to the generic SOAP messages constructed by kSOAP and tailor them to suit each type of message. Basically, this means that our message holding classes will be subclasses of the `SoapEnvelope` class. They will use the basic SOAP envelopes, but will expand them so that they also include the Liberty specific header and body elements as well as the requests that will be used to contact the WSP.

Liberty ID-WSF Security and Privacy Overview [68] states that "the authenticated identity of an identity provider must be presented to a user before the user presents personal authentication data to that identity provider." Consequently, it is not enough to present the URL of the identity provider to the user before he enters his username and password. The phone needs to be able to establish an HTTPS session with the IdP before the application presents its identity to the user. Then it is possible to use the information in the server certificate to show the identity of the IdP.

The original `HttpTransport` class from kSOAP does not support the possibility of establishing the session before it is used. This means that we will have to create our own transport class for use in the communication with the IdP. This will allow the application to establish the session with the IdP, show the authenticated identity of the IdP to the user, acquire credentials and send an authentication request.

8.3.2.4 GUI

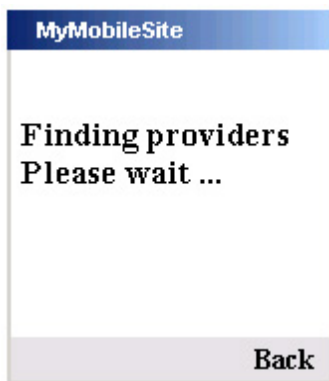
This section features a proposition for the appearance of each of the screens in the application. The figures show approximately what the windows will look like and the text describes our design decisions. There are nine different application windows which are described in more detail below. Below we have used English on all texts, but we will also enable setting Norwegian as the application language. Therefore all the texts and messages are translated and can be found in appendix E. Since the screen size of mobile phones are small we will try to use short texts without compromising on comprehension.

Another concern for the GUI comes from allowing mobile phones from various suppliers. They all have their own standards which will affect us since we must deal with different sets of keys and ways of displaying content. The former means that while the primary selection key is on the left on a Sony Ericsson, there might exist brands where the primary selection key is on the right. Thus, if a Sony Ericsson user wants to confirm several actions in a row he only needs to press the left button several times, whereas this may be different for users of other mobile brands. Therefore we will not specify that the key confirming an action should be on the right, but rather say that we should put it in the primary selection key position. However, in the figures of this section we have drawn the screens according to the standard of Sony Ericsson.

SPLASH SCREEN

The splash screen will be the opening screen of the application and should show an image with the application logo. The splash screen will only be used as a waiting screen before the language has been read from the mobile's record store. After that, a proper text will be provided to tell the user what is happening.

WAITING SCREEN



The waiting screen on the left is used to keep the user informed of the progress while the mobile is communicating with the identity provider or the web service provider. We have chosen not to use a progress bar to avoid tying up processing resources. Another problem is that we do not know how long it will take. Only the message creation and processing happens on the mobile and we cannot know how long the message transport and response generation will take. Instead we will give the user an informative text on what is happening and a message asking him to wait.

The only user option is the "Back" button, which takes him back to the last screen with user input. This button is placed on the secondary selection key for two reasons; in accordance with our standard and to ensure that the user does not accidentally push the button twice in the previous screen (which has a confirm button) and therefore unintentionally aborts the action.

LOGIN SCREEN

The login screen on the left will feature the name of the identity provider with which the application has established an HTTPS connection. Then the user can decide if this is the same provider that it has an account with. If so, the user can enter the username and password below. Since this takes more time when using a mobile as the input device, the username is stored in the mobile memory to save time on the next login. This choice was made since a mobile is a device closely connected with its owner, thus it is likely that only one person will use it.

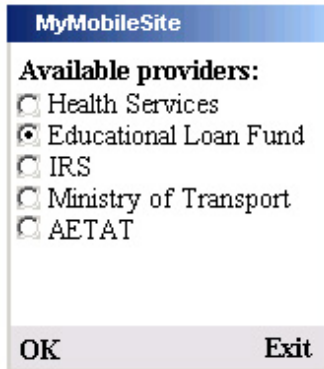
However, to add flexibility we permit username editing so that others can use the application if the user lends the phone to someone he trusts. We consider this to be acceptable in terms of security because, in the case of theft, the criminal will still need to guess the password which gives the user enough time to close/change the account.

The "OK" button is used to send the information, but before sending the GUI will perform a check of the two input fields to ensure that they are not empty. The "Menu" selection will have different names on phones by different suppliers. It will still feature the same content which are the options "Settings", "Help" and "Exit", where the first two will go to the respective screens and the last will close the program.

SETTINGS SCREEN

The figure to the left is a basic window allowing the user to specify the preferred language. The supported languages in this application are English (gb) and Norwegian bokmål (nb). The language can be chosen from a combo box and the button "Save" allows the user to confirm his choice and go back to the login screen. We have also added a button "Back" which takes the user back without saving. The name of the language should be in its original language, but to clarify extra we have also chosen to add the language code in brackets behind the name.

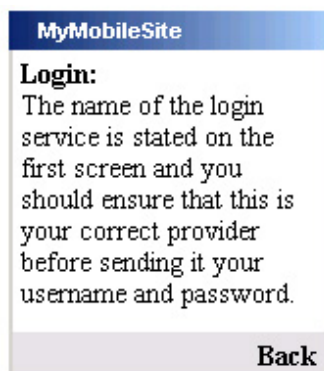
LIST OF WEB SERVICE PROVIDERS



This screen will list all the available web service providers that was returned from the discovery service. The figure on the left illustrates what this would look like with several providers, even though we just have the one. The list will allow the user to choose one of the providers and confirm with the "OK" button. At this point, the user's only other option will be to quit the program. It is not likely that the user will go back at this point since he usually cannot have several profiles. It is also unlikely that he lends the phone to someone else at this point. Therefore, this option is not necessary.

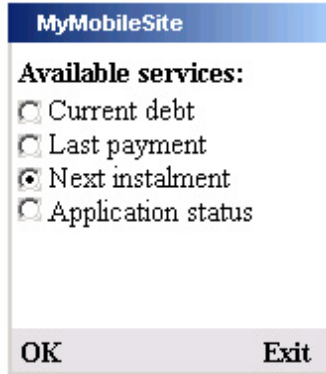
For selecting a provider we will use one of the built-in selection lists, so that the user can use the navigation keys on his phone to choose from the list. It is important that the list is exclusive, which means that it should only be possible to choose one provider at a time. Since we only have one WSP in our system, the selection marker should be set on this by default. This way the user can progress right away by clicking "OK". If there are several providers in the list, the marker should be set on the topmost one.

HELP SCREEN



The figure to the left shows how the help window consists of a plain text describing the basics of this application and a button labeled "Back" which takes the user back to the login window. The text is by no means a complete user manual, but will provide the user with some helpful hints. This is because we had to make the text short to make it easier to read on a small screen. We have decided to insert some linebreaks into the text to ensure readability because when the help text is so short it is important to make the included text good enough for the user to want to read it.

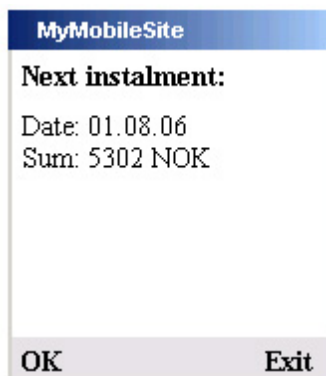
SERVICE LIST



The service list screen, which will look like the figure to the left, shows a list of all the available services that was returned from the chosen provider. The list will allow the user to choose one and to see details by using the "OK" button. For this screen we have the same situation as for the list of providers and to achieve a conformant design, the same solution for the list will be used. As was the case for the list of web service providers, the user's only alternate option to "OK" will be to choose "Exit" and close the application.

At this point we did not want to let the user go back to the provider list for two reasons. Firstly, letting the user go back would mean that we had to deal with SSO and getting new tokens when the old ones are expired. As explained in section 7.2 this will not be implemented in this version. Since there is only one implemented provider, it makes sense to leave this functionality out now. Secondly, a typical user will only require one service each time he uses the application, thus the option to go back would not often be used.

SERVICE DETAILS



This screen will feature the information for the selected service. The figure to the left shows that the name of the service will be at the top of the screen and the actual information will be displayed underneath. Both these fields come directly from the message received from the service provider since we have specified in the MobileRegisterResponse XML Schema that it should deliver a title field, label field and a value field. When finished viewing the information the user can select "OK" to go back to the list of services or "Exit" if he wants to close the program.

ERROR SCREEN



The figure to the left shows an example of the error screen with one of the possible error messages. This screen will inform the user that an error has occurred and then provide a more informative text which is specific to the type of error. In the case where the WSP returns a fault message this error text will be retrieved directly from the message, since we know that this will follow the MobileRegisterFault XML Schema. Otherwise, the error text will be created locally and adapted to the type of error that occurred. There are a few preset error messages that will be used, and these should give the user enough info to figure out what is wrong.

The two user options on this screen are "Exit" and "Try again". By selecting to try again, the program's action will depend on the type of error. The errors shown to the user come from a finite list, which disguises the errors sent between the three entities. The error texts are constructed to give the user an understandable text which gives information about what he should do. For a user not well acquainted with such systems, this should help him to decide if it is his fault or the program's. In the latter case he should realise either that he should try again or that there is an error on the server so he must contact the provider.

The mobile application will not create a log of error messages, as this could take up too much space on the phone. Instead the user manual will encourage users to report errors by phoning some kind of support line. We could have enabled the application to send a message to one of the other entities of the system whenever it encounters an error. However, we do not think this functionality will be necessary in the prototype.

8.4 Identity provider

When given the assignment, Kantega indicated that they would give us access to a finished identity provider, but as we approached the development phase it came clear that this would not happen. We therefore decided to use the Sun Java System Access Manager which came out as the best choice in our survey of Liberty-enabled products in chapter 6.

This software suite is a fairly large product which had more functions than we needed which complicated the installation. Since the scope of the assignment widened when we had to do this ourselves, we found after some effort that the installation of this was more time-consuming than the project could bear. Therefore, we made the decision to design our own identity provider. This will of course be somewhat simpler, but still have the necessary functions.

Our identity provider in reality consists of two different services, which is the authentication service that authenticates the user and the discovery service that finds resource offerings for the user. Our design documentation of this is split into two sections where the first gives a complete description of the system structure and the second has more details about certain areas where the decisions need to be documented more thoroughly.

8.4.1 System description

To describe the identity provider, we have decided that a class diagram is sufficient since this is only a small system which should aid our main application. Figure 8.7 shows the design of the authentication service and figure 8.8 shows the design of the discovery service. A description of the notation for the class diagrams in this section can be found in appendix E.

AUTHENTICATION SERVICE

The main class here is the `AuthenticationService` which is called by the JSP file that the mobile opens. This class retrieves the request and afterwards it uses the `SaslRequest` class to parse the message. To authenticate the user identified in the request, the class `UserDb` reads the information about registered users from an XML file and checks the credentials from the request to see if it matches that in the database. If so, the offering for the discovery service is stored in a `ResourceOffering`. The `SaslResponse` class constructs the response message containing the `ResourceOffering` which the `AuthenticationService` returns to the mobile application. The class `Log` is used by all classes where errors might occur (when in debug mode).

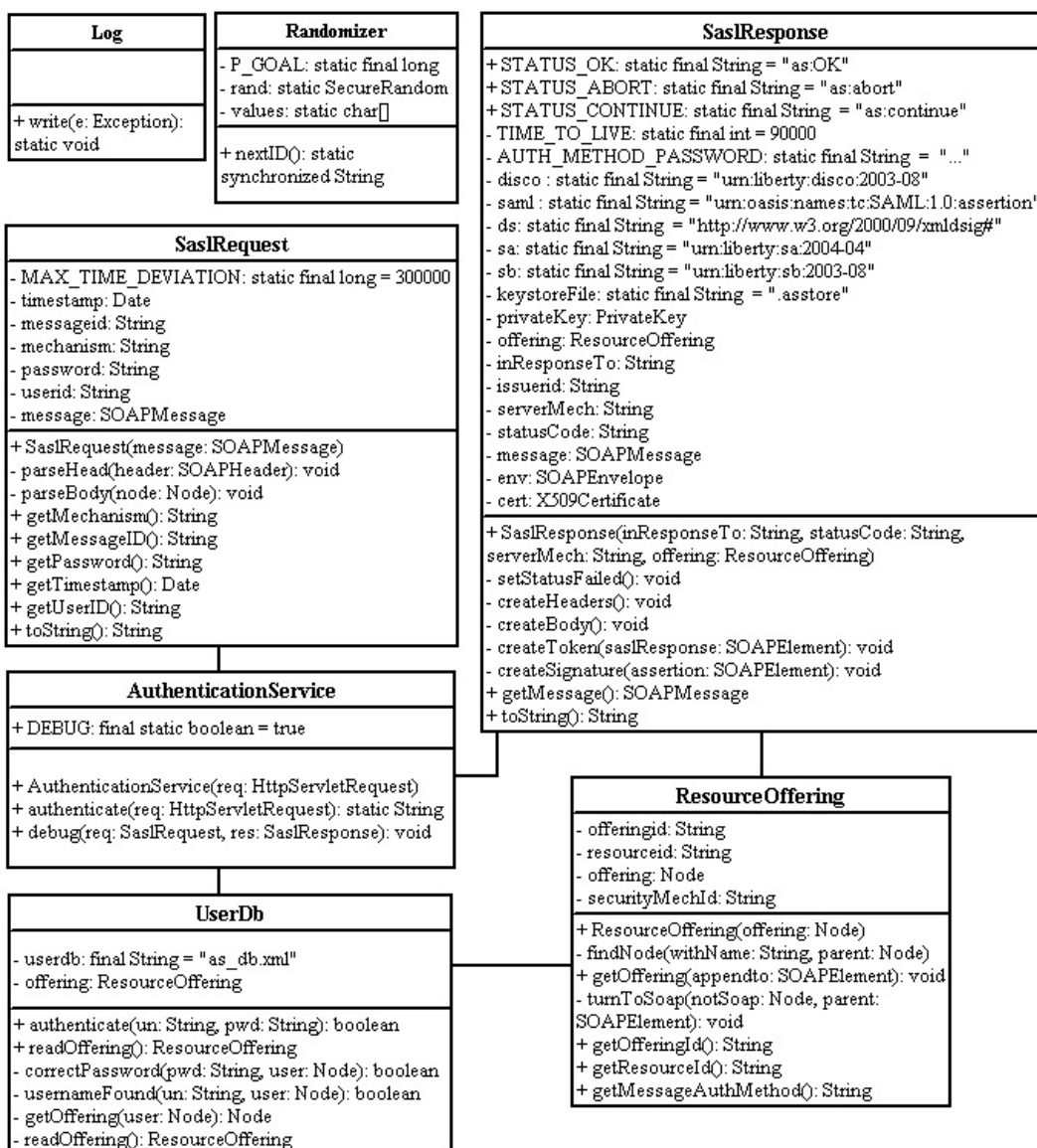


Figure 8.7: Class diagram for the authentication service

DISCOVERY SERVICE

The main class which is controlling the events in this service is the `DiscoveryService`. It is called from the JSP file and is used to get the request. `DiscoQuery` is responsible for parsing the SOAP message to get the content of the request. `DiscoveryService` then instantiates the `ServiceOfferings` class that looks through an XML file with users. If the user is found and has service offerings, this class creates `ResourceOffering` objects. These are used by the `DiscoResponse` when creating the response message. This message is returned through the `DiscoveryService` to the mobile application. The class `Log` is used by all classes where errors might occur (when in debug mode).

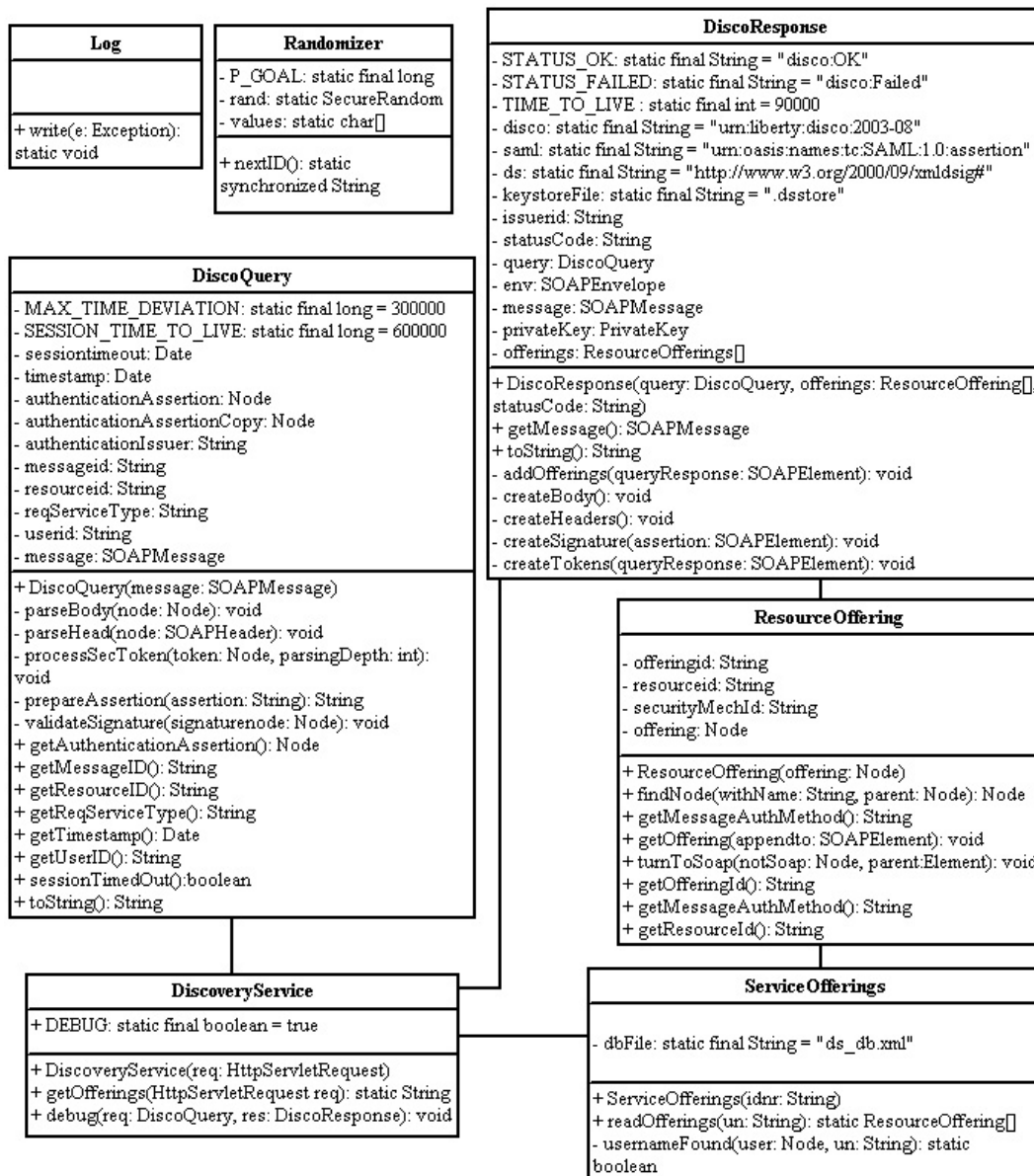


Figure 8.8: Class diagram for the discovery service

8.4.2 Major design entities

As was the case for the mobile application, not everything can be explained properly by use of diagrams. These design entities are included here to enable us to give better descriptions of them. As this is the identity provider with the discovery service included they mostly concern how the authentication of the user is transferred to other system entities.

ASSERTIONS

Both the authentication service and the discovery service at the identity provider has the need to convey authentication and authorisation decisions to other parts of the system. To do this they will use SAML assertions, but the assertions will be dissimilar for the different purposes.

As the AS is in charge of authenticating the user, it uses an authentication assertion to inform the DS of the authentication. This assertion contains an `<AuthenticationStatement>` element with information about the method and time of authentication. The content of the assertion is then signed and the signature included with the assertion.

Upon receiving a signed assertion, the DS can verify the signature and decide whether to trust the AS's authentication or not. If it trusts it, the DS will try to find the resource offerings the user asks for. Some or all of the resource offerings may require that the user presents a token to access them. In these cases, the DS will have to issue a new assertion, but this time it is an authorisation assertion. This is because it is not the DS that performed the authentication of the user. Thus, it has no other means of knowing the user's identity than the trust it has in the authentication assertion.

Consequently, the DS issues an authorisation assertion instead. This assertion contains an `<AuthorizationDecisionStatement>` element, which states what the user has been authorised to do and why. As evidence of why the DS has authorised the user, it presents the authentication assertion from the AS, with the signature intact. The authorisation assertion is also signed so that the web service providers may check the validity of both the authorisation assertion and the authentication assertion within.

IDENTITY FEDERATION

As the identity provider is meant to be able to serve several different web service providers, it may need to "translate" between different domains of identifiers for the users. However, since our identity provider will be used as a means to test the mobile application and has only to support one WSP, we will use each user's national identity number for identification on all nodes in the system.

Still, we will make some adaptations that would theoretically allow for the support of different identity domains. Most importantly we will use the `<ResourceID>` element to convey the identity of the user from one node to another. As this element is present

in the resource offerings, the discovery service will tie the potentially different identity domains to that of the IdP.

In some cases, the IdP may need to encrypt the identity of the user. It would then use the `<EncryptedResourceID>` element instead of the `<ResourceID>` element. Our implementation of the IdP and WSP will not support this, but the mobile application will be able to handle the resource identifier both with and without encryption.

REPLAY DETECTION

Ideally the IdP should maintain a cache of received messages to prevent replay. However, as this is not a prioritised part of the system and as the functionality would require some extra work to implement we have decided to skip it. The IdP will still try to check that the incoming message timestamps does not deviate too much from the moment they were received. This will ensure that an eavesdropper who has captured a valid message cannot replay it later. Furthermore, it will help us make sure that the timestamps calculated by the mobile application are valid.

8.5 Web service provider

The web service provider is a simple service that just delivers data about a user. Our description of the design is split into two sections; the first gives a complete description of the system structure and the second has more details about some system elements that are vital for the operation of the service.

8.5.1 System description

Our WSP is a simple provider that does nothing more than to check the signature of the assertion it receives. If this is correct and the user is found in its database, it creates a response message containing the information about the user. Figure 8.9 shows the design of the web service provider. A description of the notation for the class diagram in this section can be found in appendix E.

8.5.2 Major design entities

As the schemas used in the WSP are the same as the ones used in the mobile application, they are already described in section 8.3.2.1. However, there are still some aspects of the WSP we feel need to be explained further. These are mostly concerned with the interface of the WSP, as it is the part of the WSP that is in contact with the mobile application. Consequently, it is also the most important part.

WSDL

The WSDL for the web service provider can be found in appendix D. It is an abstract WSDL and its intention is to be used in the discovery service to define the service type for the service providers that are able to communicate with the mobile application. To be as generic as possible, the WSDL does not contain any service element that specifies the actual endpoint of the service. This information will instead be transferred to the mobile application outside the WSDL.

When the abstract WSDL is combined with a `<Description>` element included in the service offering at the DS, the concrete WSDL can be logically computed. The `<Description>` element contains both the endpoint of the web service provider and the SOAP action to be used. With this information and the WSDL the mobile application will know all it needs to know if it wants to contact the service.

ERROR MANAGEMENT

If the WSP encounters an internal error or the processing of a request fails, it will return an error message to the mobile phone. The error messages will follow the fault schema and as mentioned earlier they will contain an error code and a message from the WSP to the user. Below we list the possible error codes and the reasons why they occur. The corresponding error messages are listed in table E.3 in appendix E.

unknownId: If the user is unknown to the WSP it will return an error message with this error code. It basically means that the WSP could not find any information about the user in its database.

requestTimedOut: If the WSP uses too much time to answer the user's requests or the timestamp of the request deviates too much from the current time the WSP will return an error message with this code.

serviceUnavailable: This code will not be used by our implementation, but may be used by other WSPs.

notAuthorized: If the validation of the security token fails the WSP will return an error message with this code.

internalError: If some unexpected error occurred during the WSP's attempt to fulfil the request it will try to return an error message with this error code. However, depending on the nature of the internal error the WSP may not be able to return anything.

In addition to the error messages sent to the user, the WSP will use a simple log to track the exceptions that occur. This will be of great help to us when we try to find the cause of the errors, but the logs will not be used for anything else.

ACCESS CONTROL

Immediately upon receiving a request, the WSP will extract the security token and start to process the assertion. If the assertion is rejected, the WSP will return an error message, but otherwise it will look up the requested information and send it in the response to the user.

When the WSP checks the assertion it will process both the authorisation statement and the authentication assertion provided as evidence. Most importantly it will check the signature of both assertions to be certain of their origin and authenticity. If either assertion fails the verification, the entire token will be rejected.

REPLAY DETECTION

As was the case for the identity provider and for the same reasons we will not implement caching of received messages on the WSP. However, the WSP will check the timestamp of incoming messages and return an error message if it deviates too much.

8.6 Strong authentication

Adding strong authentication to the mobile application would mean changes to the existing design, as well as the introduction of new issues. Therefore we have divided this section into two separate parts, where the first will discuss the new considerations that come with the incorporation of strong authentication.

8.6.1 New and altered design issues

The introduction of strong authentication to the system requires both new functions and increased security from the mobile application. As the services that require strong authentication are of a more sensitive nature, it is important that both the user and the WSP can be confident that they are handled properly by the mobile phone.

As was decided upon in chapter 5, the strong authentication mechanism should be implemented by use of a PKI-based solution. This implies that the application will need to have cryptographic functionality, as well as a form of secure storage. Furthermore, this new authentication mechanism will not be compatible with the "PLAIN" SASL mechanism used so far. Consequently, we will have to find another SASL mechanism that will suit the needs introduced by the new authentication mechanism.

CRYPTOGRAPHIC API

When implementing PKI authentication in a J2ME MIDlet it would probably be best to use the Security and Trust Services API (SATSA) optional package. This package offers amongst other things tools for communication with the SIM card, and it would then be possible to utilise the cryptographic functions of the SIM. Additionally, we could use the SIM to store the private key. However, there are not many phones available today that implement SATSA. In fact, the only ones we could find was Nokia phones in the S60 3rd edition series.

As we prefer that our solution would be available to as many phone models as possible, we will try to avoid being dependent on SATSA. Consequently, we need something that will replace the cryptographic functions that plain MIDP implementations lack. Fortunately, the Bouncy Castle Lightweight API [69] does just that. Bouncy Castle is an open source package that provides methods for encryption, signing and key generation for some of the most common asymmetric and symmetric algorithms. Unfortunately it will not let the MIDlet communicate with the SIM and therefore the MIDlet will need to implement its own version of secure storage. We discuss how to do this in the next section.

SECURE STORAGE

Most of the requirements for strong authentication are directed towards the storage and handling of the secret key. This of course is a natural consequence of the fact that the secret key is very important. It is dangerous if the key falls into the wrong hands. Therefore, we have to protect the key as best we can from unauthorised access, both from other applications and in case the mobile phone is stolen.

To protect the key and fulfil requirement SA-5, the key needs to be encrypted before it is placed in the record store. To do this the cryptographic functionality of the Bouncy Castle API can be used, but some kind of key is needed for the encryption and decryption. However, as required by SA-7, an attacker will only have three tries to guess the decryption key before the private key becomes useless. Therefore, it is sufficient to use a code as the encryption key. This can then be used as the second factor for accessing and using the private key, as required by SA-6.

However, the problem with this approach is that the phone will not be able to check whether the password given by the user unlocked the key correctly. As the key is not stored anywhere in plaintext it cannot compare the decrypted value with the real key. One solution could be to use the decrypted key to sign something and check whether the signature was correct with the public key, but doing this would mean a lot of extra work for the phone.

A better method would be to try to authenticate with the decrypted key without knowing whether it is correct or not. If the password was incorrect, the authentication service would reject the authentication and notify the mobile phone of this event. Then, the authentication service would be able to count the number of tries the user has left and revoke the certificate if the authentication fails too many times, thus fulfilling SA-7.

SASL MECHANISM

As requirement SA-3 specifies, the keys shall be based on RSA. This means that we have to find a SASL mechanism that supports RSA. According to *ISO/IEC 9798-3 Authentication SASL Mechanism* [70] the "9798-U-RSA-SHA1-ENC" mechanism does just that. The "U" in "9798-U-RSA-SHA1-ENC" stands for unilateral, which means that this mechanism only authenticates the user. The mechanism is available with server authentication as well, but in our case this is not necessary as it will be done during the SSL handshake.

With this SASL mechanism the phone will not be able to include the authentication data in the first request. The server has to generate a random challenge value and send it to the phone in the first response. Then the phone can generate its own random value and create a token containing the phone's random value, the user's certificate and a signature made by the user's private key over both the server's and the phone's random value. The token can then be sent to the server in the <Data> element of the second SASL request.

Upon receiving the token, the server will verify that it is correct. It will combine the random value it created earlier with the one it received and check that the signature is valid. Furthermore, it will validate the user's certificate. If the token passes the tests, the user will be authenticated and the server can return the information needed to contact the discovery service.

SECURE RANDOM

As mentioned in chapter 3 the default random number generator used in MIDP is less than secure. With the poorly created random seed it is too easy to guess what the next number will be. When the strong authentication is introduced, the random number generator will be used for several new purposes. For instance, it will be used in the process of generating key pairs and creating values to be signed in the authentication. The Bouncy Castle Lightweight API includes a `SecureRandom` class intended to cope with the problem. However, it leaves it to the MIDlet that uses it to collect the randomness.

There are several possible sources for randomness, but on a mobile phone the best source available to a MIDlet will usually be the user himself. The time interval between each time he presses a button may be recorded and added to the random seed. However, this would require an extra process running on an already cramped device. It would be better to do as suggested in *Wireless Java Security - Understanding the issues* [27] and gather a certain amount of randomness at start up, while displaying a gauge that shows how many times the user will have to press the buttons before it is enough.

To avoid annoying the user too much, the gathered randomness will be stored in the MIDlet's record store. Then he will only have to "press buttons" the first time he starts the application. To avoid using the same random seed each time, the system will mix in the current time on every start up. However, the randomness gathering window will be available at all times from the settings screen, so that the user can create some more randomness whenever he wishes.

SECURE TRANSPORT

Section 8.2 discussed the problems with the HTTPS support of MIDP. The results of that discussion are still valid when the authentication has become stronger, but the reason for not employing extra encryption of the messages is different. Now that the authentication is stronger, the protection of the messages should be equally stronger, but as encryption and decryption are demanding processes it would seriously affect the performance of the application.

This claim may sound strange as the messages are already encrypted by the HTTPS connection. However, this encryption is done with the symmetric HTTPS session keys. As the phone has to communicate with a range of unknown web service providers, it is infeasible to have keys for each of them. Therefore, the extra encryption would need to be done with asymmetric keys, which is a much slower method. Additionally, it would have to download the public keys before the communication could start.

When J2ME performance increases, it may be possible to use extra encryption, but before then the approach may cause the usability to decrease to such an extent that nobody wants to use the application. However, as we are not going to implement this part of the system ourselves, anybody who wishes to implement it in the future should reconsider this issue in the light of new technological developments. He or she may find that the problems with the HTTPS support are already sorted out by the phone manufacturers, or that the performance of the phones is improved, so that the extra encryption approach has become feasible.

THE CERTIFICATE

The user's certificate is required to follow the recommendations of *Leveranse oppgave 1 Anbefalte sertifikatprofiler for personsertifikater og virksomhetssertifikater* [36]. This means that it will be in the X.509v3 format and that its fields will be given values as specified in the document.

As requirement SA-10 specifies, the user has to get his certificate signed by some kind of certificate signing service before it can be used. As mentioned in chapter 5, it is possible to authenticate the user to this service by use of OTP. However, we need to distinguish between the two classes of certificates. While the "Standard" class can allow the user to authenticate by use of a OTP he has received by mail, the "High" class will require personal attendance at a public office.

During his visit to the public office, the user must allow the clerk to authenticate himself to the certificate signing service through the phone by entering his own ID and OTP. By doing this, the clerk certifies that the user has authenticated himself personally by use of a valid identity proof. If certificates of this class are revoked, the user will have to repeat his visit to the public office to get a new one. If the same thing happened to a certificate of the "Standard" class, the user can generate a new key himself and get the certificate signed by use of a new OTP.

8.6.2 New design

With the introduction of strong authentication, there will be some changes to several parts of the design. Most of these changes will concern the parts of the system dealing with authentication of the user. Figure 8.10³⁵ shows how the state diagram for the mobile application will be altered with the introduction of strong authentication. We will use this diagram to illustrate the new features of the system and to describe how the authentication will differ from when the system only had weak authentication.

The notation is the same as before, with the boxes indicating what the user will see on his screen. To keep the diagram as simple as possible, both the help and error management have been left out. However, the help screen will be available from the states labeled "Verify personal info", "Please enter OTP number", "Certificate signed. Choose password", "Choose auth. mech." and "Unlock keys with password", and will display relevant text for each screen. Below we describe what happens as the system moves through its states.

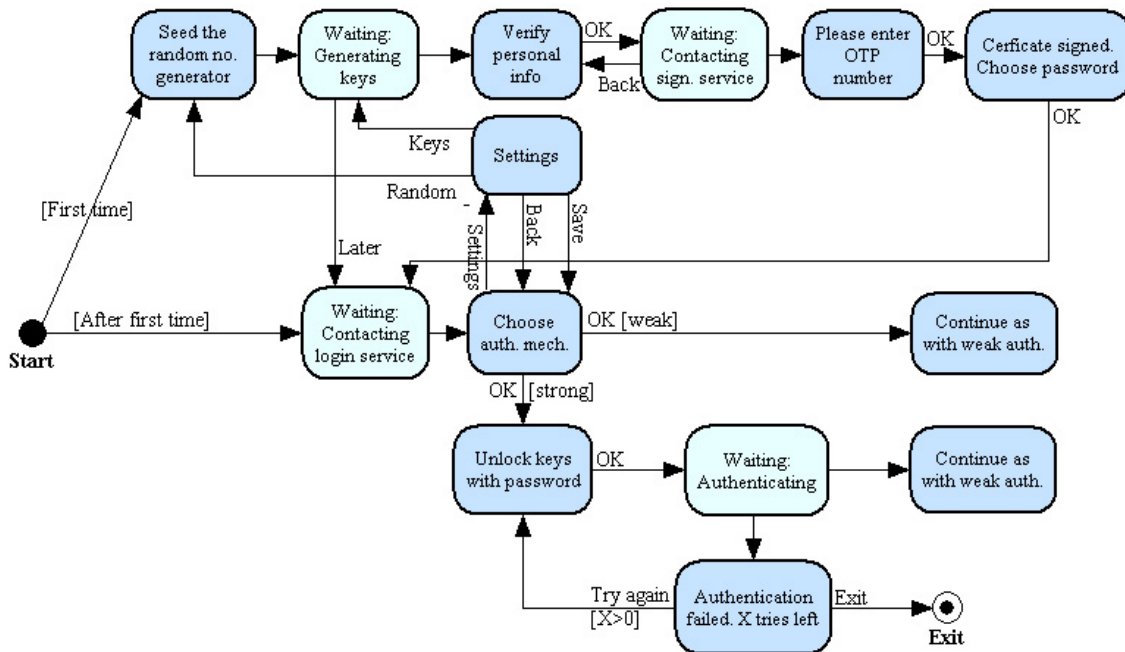


Figure 8.10: State diagram for strong authentication

³⁵The figure uses the text "Continue as with weak authentication" to depict that the execution continues as in figure 8.1. When choosing the weak authentication mechanism, the user will be shown the log-in window and advance from there. When choosing strong authentication, a user who has successfully authenticated will be shown the waiting screen "Finding providers".

SEEDING THE RANDOM NUMBER GENERATOR

First time the user opens the application he will be required to generate randomness for the random number generator. This will be indicated by a screen showing an explanation and a gauge that increases when he press the buttons. After the application has gathered enough randomness, the secure random number generator provided by Bouncy Castle will be initiated with the random seed. Then the seed will be stored in the MIDlet record store.

When the user opens the application again, he will not be required to follow this procedure unless he chooses to do so by opening the randomness generator in the settings menu. Instead, the random seed is read from the `RecordStore`, and the current time in milliseconds is mixed in to keep the seed from being the same each time. The new seed, with the time mixed in, is then stored in the record store and used to initiate the random number generator.

KEY PAIR AND CERTIFICATE GENERATION

After seeding the random number generator, the user will move on to generating a key pair for use in the strong authentication mechanism. He can abort the operation if he wishes, but otherwise the keys will be generated and the user will be required to check that the information in the certificate is correct. If there are any empty fields, he will be asked to fill them in.

When the user is satisfied with the content of the certificate he presses "OK" and the application attempts to contact the certificate signing service. The signing service will be presented with the certificate and a request to sign it. Upon examining that the information contained in the certificate is in accordance with the requirements, the server will ask for the OTP. If the certificate is of class "High", the clerk working at the public office will enter his special OTP, but otherwise the user will enter his OTP in the screen showed by the application.

If the certificate signing service accepts the OTP, it appends its signature to the certificate and sends it back to the application. The application will then store the certificate in its record store. The user will be prompted for a password to use as an encryption key for the private key. He is asked to enter the password twice and the application makes sure that the two passwords are identical. Then the application encrypts the private key and stores it in the record store. If there is a certificate or private key of the same certificate class in the record store already, it will be deleted and replaced with the new certificate or private key.

During the attempt to get the certificate signed by the certificate signing service, there are a couple of new errors that may occur. All of them result in the certificate signing service refusing to sign the certificate and a subsequent error message displayed on the user's screen. Such errors will occur if the OTP is not valid. Another cause is if the certificate signing service cannot verify the user's existence in the Norwegian Central Register of Persons and that the national identity number corresponds with the name as required by SA-11.

AUTHENTICATION

After the user has finished with the key and certificate generation, the application will try to establish the identity provider session and authenticate the identity provider like it did with only weak authentication enabled. If this is successful it will show a screen with the identity of the identity provider and a list of the authentication methods the application supports. If the user chooses a weak authentication method, he will continue with the authentication as described when the system only supported weak authentication.

If the user chooses to use a strong authentication method, the application will send a request to the identity provider, indicating that it will use the "9798-U-RSA-SHA1-ENC" SASL mechanism. Providing that the mechanism is supported by the IdP, it will return the challenge to be signed by the user in the response. Then the user will be shown a screen prompting for the password to the key store. After the user has entered his password, the application decrypts the private key, generates a random value and signs the random value with the challenge from the IdP. Then a token containing the certificate, the random value generated on the phone and the signature is sent to the IdP.

Providing that the password entered by the user was correct and that everything else was successful, the application will continue as with weak authentication after the user is authenticated. However, if the password was incorrect the authentication will fail. This will cause the IdP to increment its counter of how many times the user has failed to authenticate and return a message indicating the failure to the phone. The application will inform the user that the authentication failed and how many tries he has left. The user will then be given the option to try again or exit.

If the user fails to authenticate with the strong authentication mechanism three times in a row, the message in the screen labeled "The authentication failed. X tries left" will change to "The authentication failed too many times. The certificate might have been revoked". The user will then have only one option, which is to exit the application. When he wishes to use the strong authentication again, he must create a new key pair and certificate and get his certificate signed once more.

9 Implementation

9.1 Introduction

This chapter describes some aspects of how the proof-of-concept was implemented. We present the challenges we encountered during the implementation and describe the changes that were made in the design as a consequence of the problems. During the coding the standard *Code Conventions for the Java™ Programming Language* [71] was used to improve code quality, but we did not follow it strictly. The article *How to Write Doc Comments for the Javadoc Tool* [72] was used as a guide when writing Javadoc comments to make a good documentation for those who will use the proof-of-concept when creating a similar system later.

9.2 Implementation challenges

This section describes the areas that we struggled with when implementing the prototype. Problems were either caused by components functioning differently from what we imagined or things that we had not thought of. For each of the areas, we explain the problem, how it affected us and what we did to avoid or solve the problem. Some of the problems resulted in a change of the design and those are further covered by the next section.

CERTIFICATES AND SIGNING

To enable the server containing the IdP and WSP to communicate using HTTPS as required by F-6 and F-9, we needed to create a certificate that the server could use to authenticate itself. In a real setting, this certificate would be signed by a widely trusted certificate authority, but in our case we made both the CA certificate and the server certificate ourselves. Therefore, any mobile that wishes to use the application has to import the CA certificate manually in order to satisfy F-20, F-22 and F-23.

As some mobile phones never ask the user before rejecting certificates, it is important to be precise when determining the values in the certificate fields. For instance, the Common Name (CN) within the distinguished name field of the server certificate should always point to the server's URL.

F-5 requires the DS to be able to issue authorisation tokens. To do this, it has to receive an authentication token from the AS, asserting the user's identity. To ensure the integrity and authenticity of these assertions they need to be digitally signed. The creation of the certificates and private keys used for these signatures was fairly simple.

They were self-signed, meaning that we could create them directly within the keystore and keep them there. In a real setting, these certificates would be signed by a widely trusted CA, but in our case the use of self-signed certificates was sufficient.

When we wanted to sign the assertions we could simply read the certificate and private key directly from the keystore. The certificate was then included in the `<Signature>` element, so that the receiving party could use it to verify the signature. However, as it turned out the function that sent the SOAP message to the mobile phone added some extra line breaks and spaces to the message to make it more readable. This caused the signature verification to fail, since the assertion that was signed did not contain these extra characters. Additionally, the authentication assertion used as evidence in the authorisation assertion had all its `saml` namespace prefixes replaced by `ns0`. This also caused the verification to fail.

To avoid these problems, we processed the incoming assertion strings before verification. During this operation, we removed the `<Signature>` element of the assertion we were trying to verify the signature of. This was done since we knew that this element would not have been present in the assertion at the time of signing.

This approach of processing the string representation of the assertion until it fits may not be the most interoperable approach. It means that the WSP and DS may have trouble verifying signatures that are done by other systems than those we developed. However, this interoperability problem does not affect the mobile application since it only has to pass on the signed assertions and does not have to verify them itself.

Another flaw in the signature verification lies in the processing of the certificate. The verifying party does not check that it trusts the certificate's issuer and whether the certificate has been revoked. This functionality would need to be included if the WSP and DS were to be used as real applications. However, this is a deliberate resolution to limit the work needed on the supporting systems for the proof-of-concept.

USER ABORT

After some consideration, we found that the easiest way of implementing the functionality allowing the user to abort network operations from the phone was to simply cut off the network connection whenever the user chose to abort. This approach was inspired by *Networking, User Experience, and Threads* [73] and causes the sending and receiving of messages to fail and throw exceptions. In the calling classes, we then catch these exceptions and check whether they were caused by an abort or not before we process them. If the exceptions were caused by an abort, they are not reported and no action is taken, otherwise appropriate actions are taken depending on the exception.

This approach may sound a bit crude, but it is effective and there is no need for making things more difficult than they need to be. Another advantage is that the network traffic is cut down to a minimum. If the user aborts, he will not have to pay the cost of continuing to send or receive the messages he wanted to stop.

OPTIMISING THE CODE

During the implementation we thought a lot about how we could make the application use as little resources on the mobile phone as possible. This helped increase the speed of the application and in the end we were fairly pleased with the result. However, we have only tried the application on a Sony Ericsson K700i, which is a couple of years old, and we expect that the performance will be better on a newer model. This will be due to increased network speed and better Java performance. As illustrated in figure 9.1, especially the latter has improved dramatically over the last two years³⁶. Compared to our test model, SE K700i, the new phones are often two to three times faster.

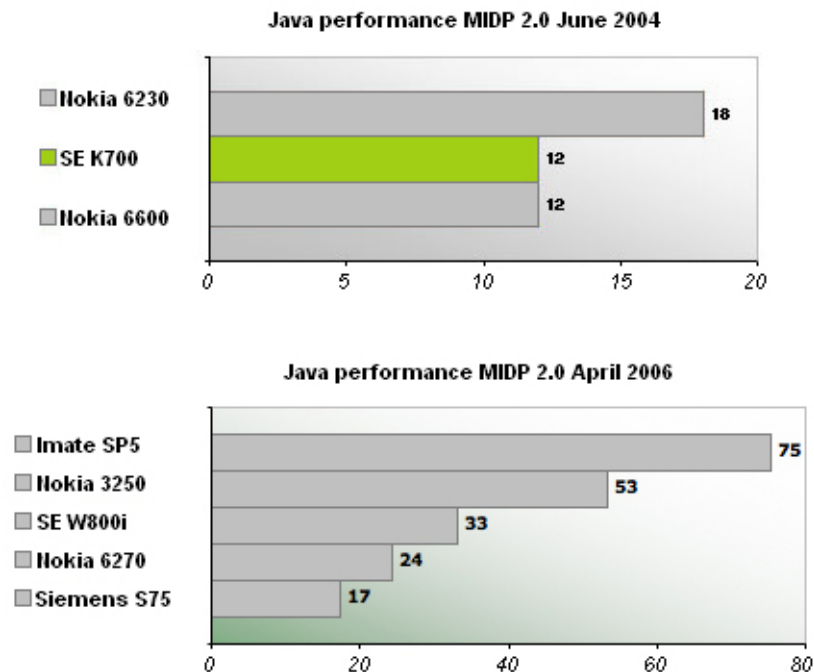


Figure 9.1: Java MIDP 2.0 performance
(adapted from [74] and [75])

One of the measures we took to ensure a modest use of resources was to set the object variables to `null` as soon as we knew we would not need them any more. This way they can be collected by the garbage collector and will not take up space in the phone's memory.

To decrease the overall size of the application and satisfy NF-3 we also introduced the abstract class `LibertyMessage`. This was done because we saw that the three SOAP message classes had a lot of methods in common. Thus there would be a large amount of redundant code if this measure had not been taken.

³⁶The figure is based on tests performed by use of JBenchmark testing and published in *Amobil :: Test - Nokia 3250* [74] and *Amobil :: Test - Sony Ericsson K700i* [75]. In this kind of test, an application measures the time required to perform a standard set of operations on the phone.

THREADS

As explained in *Networking, User Experience, and Threads* [73], in MIDP programming "the only threads that belong to you are the ones you create yourself". This means that every operation that may take some time to complete should be implemented in a separate thread. If this is not done, the application may lock up because the system thread is hijacked. Thus, we had to think a lot about this during the implementation and we ended up having more threads than we originally intended.

In our case the time-consuming operations are XML parsing, networking, writing to the record store and changing the screen. As XML parsing and networking never happen at the same time, these two could be put in the same thread. Both these operations are governed by the `LibertyProtocol` class, which means that it was convenient to have this class in charge of the thread. The other two operations need to be performed in parallel with other tasks. Therefore, these had to be put in separate threads, which are governed by the `Configurations` and `GUI` class respectively.

USE OF OPEN SOURCE CODE

We wanted to change as little as possible in the open source code we used. Therefore, the only alteration we made to these classes was in the `org.ksoap2.transport.ServiceConnection` class. This class is meant for use with both J2SE and J2ME, thus it imported a package and threw an exception that did not exist in J2ME. The only thing we did was to comment out these parts to avoid the errors that occurred because the package and exception could not be found.

In the method `call(SaslMessage)` of the `AuthTransport` class we have copied most of the code from the `call(String, SoapEnvelope)` method in `org.ksoap2.transport.HttpTransport`. This was done because much of the functionality in `call(SaslMessage)` resembled that of `call(String, SoapEnvelope)`. However, we did some alterations to this code afterwards so that the two methods are no longer identical.

GUI

What we quickly found out when we started programming the MIDlet interface was that every mobile brand has its own implementation of how form elements are placed. As we already had a test mobile at hand, we arranged the soft buttons in to fit the design description for our Sony Ericsson, thus fulfilling NF-8 for this phone. This meant that the emulator got a wrong arrangement of the keys which made it more inconvenient to use. If the proof-of-concept application should be released as a market product, this should be fixed and the easiest way to do this would be to set up a scheme which maps the buttons onto the right place for each mobile brand. This change will be restricted to the method `createButtons()` unless the developer also wants to change the button listeners. In which case, some code would have to be moved from `run()` into separate methods.

Another thing noticeable when running the application on a Sony Ericsson is the appearance of the lists, which diverge from the design. This is because we have used an

exclusive `ChoiceGroup` which is part of the MIDP implementation. Instead of giving a list to choose from, Sony Ericsson shows the top element and a select button. By pressing this another list opens and this can be used to select one item. This means that a couple of extra clicks are required before selecting the wanted option.

One way to solve this would be to replace the two `ChoiceGroup` instances with a `List`, which would achieve what we want. The reason why we chose not to do this was that a `List` cannot be added to a `Form` so then the appearance of the list screen would be different than the rest of the application. Although too time-consuming for the prototype, it would be feasible to change the implementation to use `List` instead.

WSP NAMES

In the list of service providers we wanted to show a descriptive name for each WSP. However, none of the fields in the resource offering are defined as being meant to contain the name of the WSP. Only the `<Abstract>` element can contain a human readable description of the service and naturally we chose this element to contain the service names.

As the application supports different languages we needed to distinguish between several names of each service. For instance, the WSP we implemented was called "Lånkassen" in Norwegian, but "Edu. Loan Fund" in English. This was solved by introducing a special character sequence in front of the name in each language. Then we could search for this character sequence and extract the name in the correct language from the rest of the text. For our WSP this meant that the `<Abstract>` element of its resource offering was set to ";;;gb;;;Edu. Loan Fund;;;nb;;;Lånkassen". The sequence ";;;" was chosen because it is unlikely to occur within the name.

9.3 Changes from design

The changes we made during the implementation only had a noticeable effect on the class diagrams for the mobile phone. The IdP and WSP was actually designed after the implementation of the mobile had started. This meant that we had more insight into what was needed, and the diagrams created for these have hardly changed at all. The sequence diagram showing the interaction of the mobile classes has only been altered with respect to method names and other trifles. Thus, the general idea in this has not changed at all, meaning that there is no reason to repeat the diagram here. The other diagrams are unaltered.

In figure 9.2 and figure 9.3 are the class diagrams which were updated after implementation was finished. The classes that have changes since the design are commented to explain the reason for the alterations. There are also some minor changes which are merely corrections of the design diagrams and these are not described. Included in this category are spelling errors, wrong visibility modifiers and parameter name changes.

CONTROLLER:

Configurations The changes here include removing the constructor since it did not do anything and adding more static constants which are used several places in the application. This gathers the constants in one class and prevents having to update more than one place if the values change. Some values are too long to show in the diagram.

Controller The addition of `getIdpName()` and the changes to `contactWsp()` and `userAborted()` are all consequences of introducing a better way to deal with user abort. Allowing the user to go back, we had to store some of the values in variables and remove them as parameters so the methods could be called several times.

GUI The method `setLanguage(String)` was moved from `Controller` because the GUI stores the language object and is the only place the language object is used. Therefore, it felt more natural to place it here. The introduction of the variable `previousState` with get method is caused by user abort, where we have decided where the user ends up when pressing "Back". The extra `commandAction(Command, Item)` is used on the login screen since we want the name of the identity provider to be focusable to give easy access to the "OK" button. Having two `commandAction` methods, we use the `doCommand(Command)` to check the event and avoid writing the same code twice. Also, we had to create new buttons when changing the language so this piece of code was moved into the method `createButtons()`.

Language The only change was the addition of the method `getWaitingMsg(int)` which gives the waiting messages in exactly the same way as for the error messages.

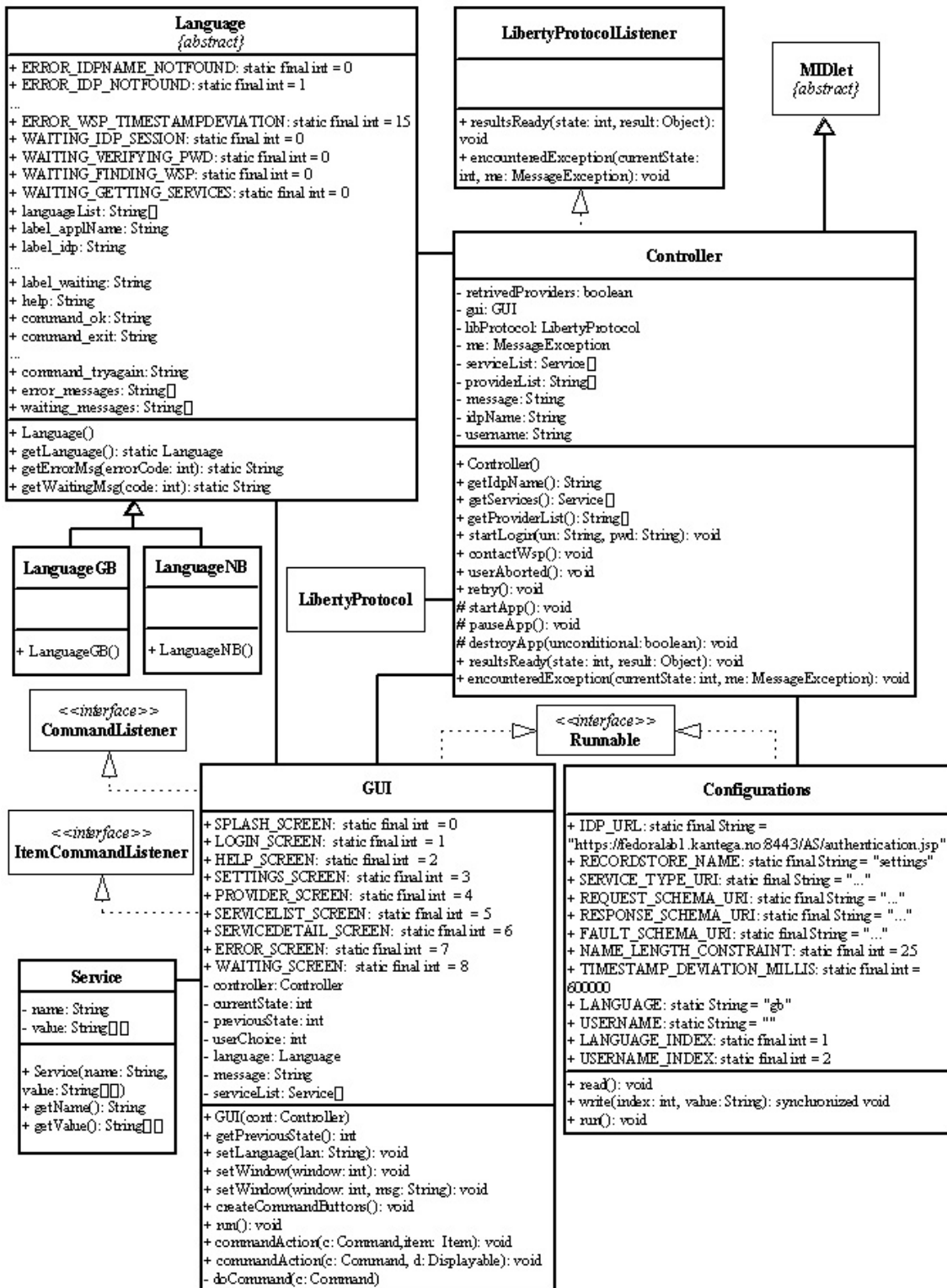


Figure 9.2: Updated class diagram (controller package)

COMMUNICATIONS:

AuthTransport The method `getName(String)` is used to get the value of the IdP subject from the certificate which can be showed to the user. The method `reset()` is used during an abort. It keeps the session, but closes the input/output streams on which we send/receive messages. Since we introduced the option to try again we cannot automatically close the session, so instead we have a `disconnect()` method which is called when we know that the session is no longer needed.

DiscoveryQuery In the constructor the parameter `ResourceOffering` was added so that the `DiscoQuery` class could use the resource offering of the discovery service in its communication with this service. The introduction of the `session` variable was caused by the method we used to implement the user abort functionality. This meant that we needed to store the `HttpTransport` so that we could get it from the `LibertyProtocol` class. The `createBody()` method was created to make the `sendQuery(ResourceOffering)` method more tidy.

LibertyMessage The `LibertyMessage` class was as explained earlier in this chapter introduced to reduce the level of redundancy in the code. It basically contains the methods, constants and variables that are common for two or more of the `DiscoveryQuery`, `SaslMessage` and `WspRequest` class. The constants `sb`, `sa`, `disco` and `wsse` were introduced to make the use of XML namespaces more convenient.

LibertyProtocol Since we have changed procedure for user abort we have added a new method `abort()` and introduced new variables that stores the sessions and a boolean `abort` value. There are two new methods for retry which are used for resending messages. The values that we are sending again also needs to be stored in variables.

Randomizer This class was introduced since we needed random values to use as message identifiers in several parts of the system. It creates a random sequence of characters and numbers, which fulfils our defined probability goal.

ResourceOffering Originally, we thought that the credentials element was inside the offering, but as it turned out it is next to the offering so the constructor also needs to get an `Element` containing the credentials. This credentials element contains credentials for all the offerings in the received message. When processing the offering, we need to store the `credentialReference` to find the correct credentials when using `processCredentials(Element)`. The code of the two processing methods were originally inside the constructor, but was moved to improve code readability. We have also introduced other variables. The `resourceID` was decided on as a mean to recognise the user instead of the username. Thus, we need to store it here so we can use it when creating request messages. We have not used the `soapAction`, but it might be necessary when implementing other WSPs and DSs.

SaslMessage We created the variable `transformations` when we realised that the SASL response could contain more than one transformation that should be used

on the password before sending it back to the server. We found that it was easiest to store the list of transformations and do all the transformations after the processing of the response was finished. The `needAnotherRequest` variable was introduced to enable the mobile application to send another request message if the IdP returns a response to the first message that indicates that this is necessary. This variable is then set to `true` and the `sendRequest(String,String)` will know that it should try again, usually after performing some password transformations. As was the case for the `createBody()` method of `DiscoveryQuery`, the `createRequest(String,boolean)` method was introduced to make the code more tidy.

WspRequest This class has undergone the same changes as the `DiscoveryQuery` class, as the two classes are very similar and most of the differences are in the content of the SOAP messages.

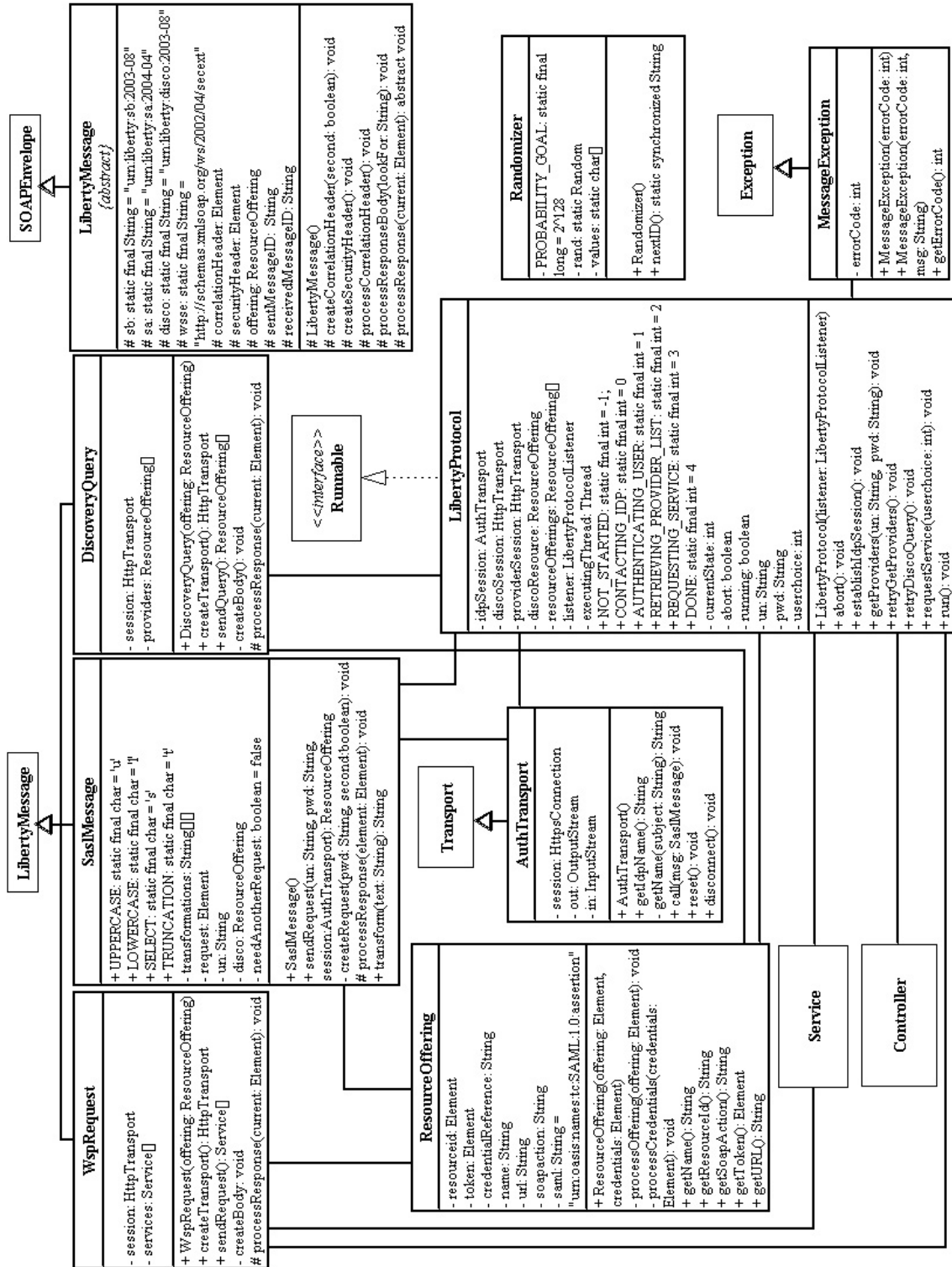


Figure 9.3: Updated class diagram (communication package)

9.4 Code samples

The purpose of this section is to try to summarise the programming effort and present the essence of what we have made. As we have tried to point out, what matters most in the code is how we use the Liberty standard to communicate between the participants in the chain to provide a service. An important factor in this is to create proper request and response messages. This is done several times, but what we want to stress is that all the messages have their own implementation on top of the common SOAP message structure. There is obviously a lot more code that deals with other aspects of the application, but we have chosen to present three methods here which treat message handling. We feel these are good examples of some of the issues we had to figure out - namely how to structure the messages, what to include and how to process them.

MAKING A HEADER

The following method is used by all the classes that extend the `LibertyMessage` class. It shows how we create one of the required header elements. These are important parts of the messages, although they do not directly concern the requests. They are mostly created for security reasons. For instance, the correlation header helps prevent replay attacks by creating a connection between the request and response messages. This is done by giving each message a unique identifier.

```
protected void createCorrelationHeader(boolean second) {
    correlationHeader = new Element();
    correlationHeader.setPrefix("sb", sb);
    correlationHeader.setNamespace(sb);
    correlationHeader.setName("Correlation");
    correlationHeader.setAttribute(env, "mustUnderstand", "1");
    sentMessageID = Randomizer.nextID();
    correlationHeader.setAttribute(sb, "messageID", sentMessageID);

    if (second) {
        correlationHeader.setAttribute(sb, "refToMessageID",
            receivedMessageID);
    }
    correlationHeader.setAttribute(sb, "timestamp", IsoDate.dateToString(
        new Date(System.currentTimeMillis()), IsoDate.DATE_TIME));
}
```

MAKING A REQUEST

The code sample below shows a typical example of how the application creates a SOAP message containing a request. As can be seen in the code, the request body is created as a hierarchy of XML elements. The top element is then put in the `bodyOut` variable inherited from `SoapEnvelope`, via the abstract `LibertyMessage` class. Since this variable is of the type `Object` the elements could not be created directly in the `bodyOut` variable, but had to be put there after their creation.

Upon completing the request body, the headers are created. In this method from the `SaslRequest` class, only the correlation header is needed, but in the `DiscoveryQuery` and `WspRequest` classes a security header is created as well. In all cases the correlation header is created last to keep the timestamp as fresh as possible. The finished headers are put in the `headerOut` variable that was inherited the same way as the `bodyOut` variable. It is of the type `Element []` so that it can hold several different header elements.

```
private void createRequest(String pwd, boolean second) {
    request = new Element().createElement(sa, "SASLRequest");
    request.setPrefix("sa", sa);
    request.setNamespace(sa);
    request.setName("SASLRequest");
    request.setAttribute(sa, "mechanism", "PLAIN");
    request.setAttribute(sa, "authzID", un);
    Element data = request.createElement(sa, "Data");
    data.addChild(Element.TEXT, "%x00" + un + "%x00" + pwd);
    request.addChild(Element.ELEMENT, data);
    bodyOut = request;

    createCorrelationHeader(second);
    headerOut = new Element[] { correlationHeader };
}
```

PROCESSING A MESSAGE

The following code sample is an example of how the application processes the responses it receives from the other entities in the system. It is taken from the class `DiscoveryQuery` and retrieves the status code, resource offerings and credentials from the DS response. As can be seen in the sample, the mobile application will look for the parts of the message that it needs and throw an exception if one of these parts could not be found.

In most cases the application does not check the namespace of the elements, thus it relies only on their local name to find them. We created it this way because we knew that there would not be no ambiguous element names in the messages. If elements with ambiguous names are introduced in the future, namespace checking must be included when looking for these elements.

```
protected void processResponse(Element current) throws MessageException {
    Element child;
    String statusCode = null;
    Element credentials = null;
    try {
        credentials = current.getElement(disco, "Credentials");
    } catch (Exception e) {}

    Vector offerings = new Vector(current.getChildCount());
    boolean hasStatus = false;

    for (int i = 0; i < current.getChildCount(); i++) {
        child = current.getElement(i);
        if (child == null) {
            /* Do nothing */
        } else if (child.getName() == null) {
            /* Do nothing */
        } else if (child.getName().equals("Status")) {
            hasStatus = true;
            statusCode = child.getText(0);
            if (statusCode != null) {
                if (statusCode.toLowerCase().indexOf("failed") > -1) {
                    throw new MessageException(Language.ERROR_DS_NORESULTS);
                }
            }
        } else if (child.getName().equals("ResourceOffering")) {
            try {
                offerings.addElement(new ResourceOffering(child, credentials));
            } catch (MessageException me) {
                /* This resource offering is not valid, skip it. */
            }
        }
    }
    providers = new ResourceOffering[offerings.size()];
    offerings.copyInto(providers);
    if (providers.length == 0) {
        throw new MessageException(Language.ERROR_DS_NORESULTS);
    } else if (!hasStatus || statusCode == null) {
        throw new MessageException(Language.ERROR_INVALID_MESSAGE);
    }
}
```

10 Testing

10.1 Introduction

This chapter includes everything that concerns the testing of the prototype. Its purpose is to document the work done to test the completed application and to ensure that all requirements are fulfilled in a satisfactory way. We also hope to uncover any programming errors in the application so that they can be corrected.

The testing effort will be focused on the developed mobile application, MyMobileSite. The IdP and WSP will be manipulated to test how the application responds to unexpected input, but will not be the subject of thorough testing themselves.

The test chapter has three parts, which is the development of a test strategy, the actual test specifications and the test results with a summary of what happened. The first two parts were written during the design of the application, but were slightly modified to accommodate the changes made in the implementation. The last part was written after the implementation was completed.

10.2 Testing strategy

Making a testing strategy ensures that we stick to an overall goal for all the testing effort. Therefore, we have to decide what kind of tests are most suitable for this application and the approach we will use. Furthermore, we have to clearly define the purpose of each part of the testing.

TEST ENVIRONMENT

There are two methods for testing a MIDlet; on the emulator and on the actual mobile phone. If the testing is done on the emulator provided by Sun, the results will be generic and not dependent on the phone model. The upside is that, the emulator has the possibility of monitoring the network traffic and memory usage, which is impossible to do on an actual phone. However, there are some differences between the MIDP implementation on the emulator and various phone models.

For these reasons, we intend to test our application on both the emulator and a mobile phone. It is most important that the application works well on the mobile phone, as it provides a realistic test environment. Therefore, we will put emphasis on the tests performed on the phone and use the emulator as a reference for how well the application will work on other phone models.

However, it would be foolish not to utilise the extra control over the test environment offered by the emulator. Thus, we will disable all optional packages during the emulator tests. That will enable us to test requirement NF-2, which reads "The application shall not need any optional packages". During the emulator tests we will also enable the network monitor, which will allow us to check that all messages are sent using HTTPS and that they contain the correct fields.

TESTING PURPOSE AND METHOD

As the mobile application is the most important part of the system, all the tests will be angled towards it. However, it is impossible to test it properly without the identity provider and web service provider. Similarly, the IdP and WSP are useless on their own. Consequently, all functional tests will be done on the system as a whole, but with the intention of discovering faults in the mobile application.

This means that most of the requirements will be tested in a functional system test. The requirements that cannot be tested functionally will if possible be controlled during code review. Table 10.1 shows a list of the requirements that will not be adequately tested during the functional test, together with the reason why. Some of these will be covered by the code review, but not all. We will not do any separate performance tests, but we will measure the time used by the system test cases to get an idea of the prototype's performance.

The possibility exists to complement the code review with a set of JUnit tests³⁷. However, as the classes are very dependent on each other, we find it hard to separate them in a way that does not require too much work on the JUnit tests. Furthermore, the JUnit test would not be able to test anything that we cannot test with the ordinary functional test, as most results are clearly visible through the GUI or in the network monitor of the emulator.

Yet another possibility would be to use security testing to ensure the quality of the security measures in the application, but according to *Amobil :: Hele www i lomma*³⁸ [76], "the mobile operators use advanced firewalls to stop computers on the Internet gaining access to your phone"³⁹. This means that the phone is not addressable from the Internet, and an attacker would either have to steal the phone or use some kind of packet sniffer to attack the communication link. Since no other personal information than the username is stored on the phone by the application, the only approach to an attack would be to guess the user's password. The task of ensuring that this is not a worthwhile activity is the IdP's responsibility, thus we will not test it. Furthermore, we already know of the weakness in the HTTPS support, meaning that we would not discover anything new by testing it.

³⁷JUnit tests are automatic tests written in Java intended to test Java code. This is done by use of various assert-methods that verify the output given a specific input.

³⁸English: Amobil :: The entire www in your pocket

³⁹Translated from Norwegian

Req	Reason
F-1	The testing of this requirement through functional tests or code review would be pointless, as its fulfillment can be verified by observing whether the two services are located at the same server or not.
F-2 F-3 F-7 F-18	The functionality of the system will not be dependent on all the Liberty conformance requirements to be fulfilled. Thus, it is difficult to cover these in a functional system test. Some of the requirements will be automatically tested with the testing of other requirements, but most of them will be controlled during the code review.
F-16	Whether the WSP uses a time-out value will only be tested during the code review as it is a requirement which when fulfilled will only affect the functionality of the WSP.
F-17	As our WSP does not require more than 200 characters, this requirement is already fulfilled by it. Thus there is no need to check it further.
F-29	This requirement will be checked during the code review as there is no way of telling through the GUI whether the XML documents are validated before parsing.
F-33	Due to the fact that we only have one WSP, it is impossible to check whether the phone is able to access other WSPs. However, we have considered the requirement during the design in chapter 8 and because of the design of the schemas and WSDL we are confident that the application is able to fulfil it. Additionally we will check that the application is able to display more than one WSP to choose from in one of the functional requirements.
NF-1	Requirement NF-1, which reads "The application shall be developed in J2ME using MIDP 2.0", is fulfilled simply by checking that the application works on the phone (since we know that it has support for MIDP 2.0). Thus it is only indirectly tested through the system tests, but if the tests are approved it means that the requirement is fulfilled.
NF-2	This requirement will be tested in a similar way as NF-1 by disabling all optional packages on the emulator and then running the application on it.
NF-3	The size of the application is difficult to test functionally, but can be easily checked through the operating system of the computer or phone.
NF-6	Whether the code is commented with Javadoc or not will only be visible in the code. Therefore, it can only be tested during code review.
NF-7	As we will not expand the application to include new languages after its completion, we cannot test this requirement in this project. However, we have considered it in the design and it is theoretically possible.
NF-8	The placement of the selection keys is highly dependent on the phone manufacturer. As we only have access to one phone model, we will have to check that the variables determining their placement indicate that the requirement is fulfilled during the code review.
NF-9	It is difficult to know what a minimal amount of input from the user is. However, in the discussion of the GUI in chapter 8 it is argued that the requirement is fulfilled. Thus there is no need to test it here.

Table 10.1: Requirements that will not be adequately tested by the system test

CODE REVIEW

The code review will be done after each class is finished. We will try to divide the classes so that each of us only reviews classes we have not been responsible for during the implementation. In the review we will have a list of requirements that will be especially checked for. The intention will be to indicate for each requirement whether it is fulfilled or not, thus making sure that no requirement will be overlooked. During the code review we will also look for programming errors, remove unused code and check that the comments are correct.

SYSTEM TEST

The system test will be divided into test cases, all of them concentrating on some requirements. We will try to make extensive test cases instead of many test cases. This is because most of the requirements can be tested during a single execution, without influencing each other. Additionally, the log-in procedure has to be performed with each execution, thus making the option of testing only a few requirements each run quite cumbersome and unnecessarily time consuming.

All test cases will be performed on a version of the system deemed finished from our point of view. There will not be any changes to the code during testing. If an error that will lead to a test failing is found, the testing will be aborted. The tests will then be executed again after the error corrections.

The debugging possibilities in MIDlets are limited and it would require some sort of logging functionality to trace requirements that would otherwise not be testable. However, the log would not be accessible from anything other than a MIDlet - meaning that the approach would require a separate MIDlet to read the logs as well. We think this is a bit too time consuming in relation to its importance and it would probably slow the application down during the system test. Therefore, we will not use this approach.

During the testing the IdP and WSP will be provisioned with test data. These will be fictitious data, but we will try to keep them as realistic as possible. As we only have one WSP, we will have to include an imaginary service offering in the discovery service in one of the test cases. The service offering will be valid in the sense that the phone will be able to extract all the information it needs, but its endpoint URL will only point to a non-existent server, causing the transaction to fail if the phone tries to contact it. There are some requirements that are similar for the IdP and the WSP, but these will only be tested on one of them. An example of this is the message format which will only be tested on the WSP.

10.3 Tests specifications

This section will present the check lists which will be used for code review and the test cases for the system test. For the latter we have also created a test/requirement connection matrix in appendix F. This provides a visual representation to help ensure that we have covered all the requirements, except those listed in table 10.1. The structure of the test specifications follows that of the templates in appendix G.

10.3.1 Check lists

Tables 10.2 and 10.3 show the check lists that are to be used in the code review of the mobile application. The specifications in tables 10.4, 10.5 and 10.6 are for the AS, DS and WSP. These contain the list of requirements to be checked during the review of each package. Even though the focus of the review is to check that these requirements are fulfilled, we will also be looking for other types of errors. If any errors are discovered they will be described in the comments field, regardless of whether they are connected to one of the requirements or not.

Package:	controller
Time of check:	[the date of code review]
Reviewed by:	[person]
Approved:	[Yes/No]
Comments:	[description]

ReqID	Requirement text	Result
NF-6	The Java classes shall be commented with Javadoc.	[OK/Failed]
NF-8	The application shall have a consistent placement of selection keys on all screens.	[OK/Failed]

Table 10.2: Check list for the controller package

Package:	communication
Time of check:	[the date of code review]
Reviewed by:	[person]
Approved:	[Yes/No]
Comments:	[description]

ReqID	Requirement text	Result
F-18	The mobile phone application shall conform to the LUAD-WSC profile of <i>Liberty ID-WSF Static Conformance Requirements</i> [58].	[OK/Failed]
F-29	The application shall attempt to parse XML documents without validating them first.	[OK/Failed]
NF-6	The Java classes shall be commented with Javadoc.	[OK/Failed]

Table 10.3: Check list for the communication package

Package:	authentication
Time of check:	[the date of code review]
Reviewed by:	[person]
Approved:	[Yes/No]
Comments:	[description]

ReqID	Requirement text	Result
F-3	The identity provider shall conform to the ID-WSF IdP profile of <i>Liberty ID-WSF Static Conformance Requirements</i> [58].	[OK/Failed]
NF-6	The Java classes shall be commented with Javadoc.	[OK/Failed]

Table 10.4: Check list for the authentication package

Package:	discovery
Time of check:	[the date of code review]
Reviewed by:	[person]
Approved:	[Yes/No]
Comments:	[description]

ReqID	Requirement text	Result
F-2	The discovery service shall conform to the DS profile of <i>Liberty ID-WSF Static Conformance Requirements</i> [58].	[OK/Failed]
NF-6	The Java classes shall be commented with Javadoc.	[OK/Failed]

Table 10.5: Check list for the discovery package

Package:	loanfund
Time of check:	[the date of code review]
Reviewed by:	[person]
Approved:	[Yes/No]
Comments:	[description]

ReqID	Requirement text	Result
F-7	The web service provider shall conform to the ID-FF SP WSP profile of <i>Liberty ID-WSF Static Conformance Requirements</i> [58].	[OK/Failed]
F-16	The web service provider shall use a time-out value and return an error message to the requester if it is exceeded.	[OK/Failed]
NF-6	The Java classes shall be commented with Javadoc.	[OK/Failed]

Table 10.6: Check list for the loanfund package

10.3.2 System test

The system test consists of 7 test cases that are specified in tables 10.7 - 10.13. For these tests we have defined 4 users that have different combinations of services and offerings. The users with their properties are listed below:

- Nina Olsen (17038492834) has one service offering with three services.
- Anne Hansen (09097873628) has no service offerings.
- Henry Johnsen (13125193312) has two service offerings; one is from our provider where he has two services, and the other is a dummy.
- Adam Strand (07067139184) has one service offering, but no profile at the issuing provider.

Identification TID:	ST1
Responsible:	Gunn Olaussen
Premises:	User: Nina Olsen (17038492834) Password: Thur2930. The phone language should be set to Norwegian.
Test operations:	<ol style="list-style-type: none"> 1. Start application 2. Fill in correct username and password. Select "OK". 3. Choose "Lånekassen". Select "OK". 4. Choose "Samlet gjeld". Select "OK". 5. In the service detail screen, select "OK". 6. Choose "Neste avdrag". Select "OK". 7. In the service detail screen, select "OK". 8. Choose "Siste avdrag". Select "OK". 9. In the service detail screen, select "OK". 10. Select "Avslutt".
Expected outcome:	<ol style="list-style-type: none"> 1. Splash screen with waiting message and then log-in screen. 2. First the waiting message "Sjekker passord", then "Søker etter tilbydere". There should be 1 provider in the list. 3. Waiting message "Henter tjenester". Then service list with 3 services. 4. The sum should be 250000. 5. Service list with 3 services. 6. The screen should show the sum 4171 and date 15.08.2006. 7. Service list with 3 services. 6. The screen should show the sum 4171 and date 15.05.2006. 7. Service list with 3 services. 10. The application closes. <p>For the entire application: the language on both labels and results should be Norwegian.</p>
Criteria of approval:	The application should display the expected output and the entire test should take less than 2 minutes.

Table 10.7: ST1 Normal execution

Identification TID:	ST2
Responsible:	Gunn Olaussen
Premises:	User: Anne Hansen (09097873628) Password: Ellif120. The application used in ST1 must be deleted and the application downloaded again. The phone language should be set to German.
Test operations:	<ol style="list-style-type: none"> 1. Start the application. 2. Fill inn the username and an incorrect password. Select "OK". 3. In the error screen choose "Try again". 4. Fill inn the username and the correct password. Select "OK". 5. In the error screen choose "Exit".
Expected outcome:	<ol style="list-style-type: none"> 1. Splash screen with waiting message and then log-in screen. 2. First the waiting message "Verifying password". Then the error screen with a message indicating incorrect password. 3. The log-in screen opens. 4. First the waiting message "Verifying password", then "Finding providers". Then an error message should indicate that no providers where found for this user. 5. The application closes. <p>For the entire application: the language on both labels and results should be English.</p>
Criteria of approval:	The application should display the expected output and the entire test should take less than 2 minutes.

Table 10.8: ST2 Test of error handling

Identification TID:	ST3
Responsible:	Gunn Olaussen
Premises:	User: Nina Olsen (17038492834) Password: Thur2930. Set the phone language back to English.
Test operations:	<ol style="list-style-type: none"> 1. Start the application. 2. Select "Help". 3. Select "Back". 4. Fill inn the username and password. Select "OK". 5. In the waiting screen "Verifying password", select "Back". 6. Fill inn the username and password again. Select "OK". 7. In the provider list screen choose "Exit".
Expected outcome:	<ol style="list-style-type: none"> 1. Splash screen with waiting message and then log-in screen. Check that the username from ST2 is displayed as default. 2. The help screen displays the entire help text (scroll to the bottom). 3. The log-in screen opens. 4. The waiting screen "Verifying password" opens. 5. The log-in screen opens. Check that the username from ST2 still is there as default. 6. The provider list opens with 1 provider. 7. The application closes. <p>For the entire application: the language on both labels and results should be English.</p>
Criteria of approval:	The application should display the expected output and the entire test should take less than 2 minutes.

Table 10.9: ST3 Test of help and user abort during login

Identification TID:	ST4
Responsible:	Gunn Olaussen
Time to test:	21.05.06
Premises:	User: Henry Johnsen (13125193312) Password: Fire83iw. IdP: A dummy service offering for the "Register of Persons" must be added to the discovery service.
Test operations:	<ol style="list-style-type: none"> 1. Start the application. 2. Fill inn the username and password. Select "OK". 3. Choose "Register of Persons". Select "OK". 4. In the error screen, choose "Try Again". 5. Choose "Edu. Loan Fund". Select "OK". 6. In the service list screen, select "Exit".
Expected outcome:	<ol style="list-style-type: none"> 1. Splash screen with waiting message and then log-in screen. Check that the username from ST3 is displayed as default. 2. First the waiting message "Verifying password", then "Finding providers". There should be 2 providers in the list. 3. The error screen should have a message indicating that the provider could not be contacted. 4. The provider list opens with the same 2 providers. 5. The service list opens with 2 services. 6. The application closes. <p>For the entire application: the language on both labels and results should be English.</p>
Criteria of approval:	The application should display the expected output and the entire test should take less than 2 minutes.

Table 10.10: ST4 Test of WSP

Identification TID:	ST5
Responsible:	Gunn Olaussen
Premises:	User: Nina Olsen (17038492834) Password: Thur2930.
Test operations:	<ol style="list-style-type: none"> 1. Start the application. 2. Try selecting "OK" without entering a password. 3. Remove the username and enter the correct password. Select "OK". 4. Fill in correct username and password. Select "OK". 5. When the waiting screen "Finding providers" opens, select "Back". 6. In the log-in screen, select "Exit".
Expected outcome:	<ol style="list-style-type: none"> 1. Splash screen with waiting message and then log-in screen. Check that the username from ST4 is displayed as default. 2. Nothing happens. 3. Nothing happens. 4. First the waiting message "Verifying password", then "Finding providers". 5. Then the log-in screen opens. 6. The application closes. <p>For the entire application: the language on both labels and results should be English.</p>
Criteria of approval:	The application should display the expected output and the entire test should take less than 2 minutes.

Table 10.11: ST5 Test of log-in and user abort during DS query

Identification TID:	ST6
Responsible:	Gunn Olaussen
Premises:	User: Adam Strand (07067139184) Password: 048hih840. He does not have a profile in the WSP even though a service offering has been made (emulating an internal error in the WSP).
Test operations:	<ol style="list-style-type: none"> 1. Start the application. 2. Fill in username and password. Select "OK". 3. Choose "Educational Loan Fund". Select "OK". 4. In the error screen, select "Exit".
Expected outcome:	<ol style="list-style-type: none"> 1. Splash screen with waiting message and then log-in screen. Check that the username from ST5 is displayed as default. 2. First the waiting message "Verifying password", then "Finding providers". Then provider list with 1 provider. 3. First the waiting screen "Retrieving services", then an error screen indicating what type of error (should be internal in the WSP). 4. The application closes. <p>For the entire application: the language on both labels and results should be English.</p>
Criteria of approval:	The application should display the expected output and the entire test should take less than 2 minutes.

Table 10.12: ST6 WSP error message

Identification TID:	ST7
Responsible:	Gunn Olaussen
Premises:	User: Nina Olsen (17038492834) Password: Thur2930. WSP: it must be modified so it sends the response message (containing the service list) in the wrong format.
Test operations:	<ol style="list-style-type: none"> 1. Start the application. 2. Select "Settings". 3. Change language to "Norsk (nb)". Select "Save". 4. Fill inn the username and password. Select "OK". 5. Choose "Lånekassen". Select "OK". 6. In the error screen choose "Avslutt".
Expected outcome:	<ol style="list-style-type: none"> 1. Splash screen with waiting message and then log-in screen. Check that the username from ST6 is displayed as default. 2. Settings screen which has a drop-down box with the choices "Norsk (nb)" and "English (gb)". 3. The log-in screen opens. 4. First the waiting message "Sjekker passord", then "Søker etter tilbydere". There should be 1 provider in the list. 5. An error message indicating wrong message format. 6. The application closes. <p>For the entire application: the language on both labels and results should be Norwegian after step 3.</p>
Criteria of approval:	The application should display the expected output and the entire test should take less than 2 minutes.

Table 10.13: ST7 Test of message format

10.4 Test results

This section presents the results of the testing as directed by the templates. It explains events that happened during testing in cases where the result diverged from what was expected.

10.4.1 Check lists

The code review was performed the 21th of May, before the system test. As no severe errors were detected during the initial review, the result was deemed successful. The results from the review can be found in tables 10.14 to 10.18. Although there was an error corrected after the first run of the system test, the alteration of the code was so insignificant that we did not do another review after the correction.

Package:	controller
Time of check:	21.05.06
Reviewed by:	Kirsti N. Torgersen
Approved:	Yes
Comments:	The placement of selection keys is not consistent on all phones, since the application should check for type of phone and place keys according to a predetermined schema for that phone to get an optimal solution. The current code will ensure a consistent placement on our test phone, which we deem good enough for the prototype.

ReqID	Requirement text	Result
NF-6	The Java classes shall be commented with Javadoc.	OK
NF-8	The application shall have a consistent placement of selection keys on all screens.	Failed

Table 10.14: Check list result for the controller package

Package:	communication
Time of check:	21.05.06
Reviewed by:	Gunn Olaussen
Approved:	Yes
Comments:	<p>F-18: For luad-wsc-authsvc-002, we only support "PLAIN" and not "CRAM-MD5" since that would require cryptographic support (making the application much larger).</p> <p>For, wsc-soapbind-002, we lack support of the ServiceInstance-Update header since the services we use to test the prototype does not move.</p> <p>For wsc-secmech-001/002, we do not support "urn:liberty:security:2003-08:null:null", "urn:liberty:security:2005-05:null:Bearer" and "urn:liberty:security:2003-08:TLS:null". The first two are omitted since we would have use message encryption to support these. The third is omitted since we need a way of passing on the authentication and authorisation assertions. However, we do support the "urn:liberty:security:2005-05:TLS:Bearer" which is recommended in the LUAD-WSC profile.</p>

ReqID	Requirement text	Result
F-18	The mobile phone application shall conform to the LUAD-WSC profile of <i>Liberty ID-WSF Static Conformance Requirements</i> [58].	OK
F-29	The application shall attempt to parse XML documents without validating them first.	OK
NF-6	The Java classes shall be commented with Javadoc.	OK

Table 10.15: Check list result for the communication package

Package:	authentication
Time of check:	21.05.06
Reviewed by:	Gunn Olaussen
Approved:	Yes
Comments:	<p>F-3: For idwsfidp-authsvc-002, we only support "PLAIN" and not "CRAM-MD5" since we that would require cryptographic support (making the application much larger).</p>

ReqID	Requirement text	Result
F-3	The identity provider shall conform to the ID-WSF IdP profile of <i>Liberty ID-WSF Static Conformance Requirements</i> [58].	OK
NF-6	The Java classes shall be commented with Javadoc.	OK

Table 10.16: Check list result for the authentication package

Package:	discovery
Time of check:	21.05.06
Reviewed by:	Kirsti N. Torgersen
Approved:	Yes
Comments:	F-2: For wsp-secmec-001, we do not support null as the peer entity authentication mechanism since the mobile does support it either and then it is worthless. For ds-registration-001/002/003, we do not supported automatic update of the DS content since this is not important for the function of the mobile application.

ReqID	Requirement text	Result
F-2	The discovery service shall conform to the DS profile of <i>Liberty ID-WSF Static Conformance Requirements</i> [58].	OK
NF-6	The Java classes shall be commented with Javadoc.	OK

Table 10.17: Check list result for the discovery package

Package:	loanfund
Time to check:	21.05.06
Reviewed by:	Kirsti N. Torgersen
Approved:	Yes
Comments:	F-7: For wsp-secmec-001, we do not support null as the peer entity authentication mechanism since the mobile does support it either and then it is worthless. For wsp-interact-001, we do not support the UserInteraction header since we have already assumed that the user have given consent through a web site.

ReqID	Requirement text	Result
F-7	The web service provider shall conform to the ID-FF SP WSP profile of <i>Liberty ID-WSF Static Conformance Requirements</i> [58].	OK
F-16	The web service provider shall use a time-out value and return an error message to the requester if it is exceeded.	OK
NF-6	The Java classes shall be commented with Javadoc.	OK

Table 10.18: Check list result for the loanfund package

10.4.2 System test

The code was finished on the 20th of May and the system test was executed the next day. The first two tests were successful, but the third one failed. Therefore, the test was aborted and we fixed the error which is identified and discussed in the test result documentation. The results of this test are in appendix H. After fixing the error, the test was rescheduled for the 22th of May. This test ran smoothly and the documentation of the results can be found in tables 10.19 to 10.25.

Identification TID:	ST1
Test executed by:	Gunn Olaussen
Time of test:	22.05.06
Actual output:	Emulator: Test ran according to specification in 58 seconds. Mobile: Test ran according to specification in 1.38 minutes.
Testing comments:	A substantial amount of the time was spent entering the username and password.
Type of error:	None
Test approved:	Yes

Table 10.19: Test result for ST1

Identification TID:	ST2
Test executed by:	Gunn Olaussen
Time of test:	22.05.06
Actual output:	Emulator: Test ran according to specification in 42 seconds. Mobile: Test ran according to specification in 1.41 minutes.
Testing comments:	Used password "123". Test phone did not have German as an option so we had to use Danish. A substantial amount of the time was spent entering the username and password.
Type of error:	None
Test approved:	Yes

Table 10.20: Test result for ST2

Identification TID:	ST3
Test executed by:	Gunn Olaussen
Time of test:	22.05.06
Actual output:	Emulator: Test ran according to specification in 1.01 minutes. Mobile: Test ran according to specification in 1.50 minutes.
Testing comments:	A substantial amount of the time was spent entering the username and password.
Type of error:	None
Test approved:	Yes

Table 10.21: Test result for ST3

Identification TID:	ST4
Test executed by:	Gunn Olaussen
Time of test:	22.05.06
Actual output:	Emulator: Test ran according to specification in 56 seconds. Mobile: Test ran according to specification in 1.43 minutes.
Testing comments:	A fairly large amount of the time was spent entering the username and password.
Type of error:	None
Test approved:	Yes

Table 10.22: Test result for ST4

Identification TID:	ST5
Test executed by:	Gunn Olaussen
Time of test:	22.05.06
Actual output:	Emulator: Test ran according to specification in 40 seconds. Mobile: Test ran according to specification in 1.09 minutes.
Testing comments:	A fairly large amount of the time was spent entering the username and password.
Type of error:	None
Test approved:	Yes

Table 10.23: Test result for ST5

Identification TID:	ST6
Test executed by:	Gunn Olaussen
Time of test:	22.05.06
Actual output:	Emulator: Test ran according to specification in 58 seconds. Mobile: Test ran according to specification in 1.25 minutes.
Testing comments:	A fairly large amount of the time was spent entering the username and password.
Type of error:	None
Test approved:	Yes

Table 10.24: Test result for ST6

Identification TID:	ST7
Test executed by:	Gunn Olaussen
Time of test:	22.05.06
Actual output:	Emulator: Test ran according to specification in 58 seconds. Mobile: Test ran according to specification in 1.41 minutes.
Testing comments:	WSP was modified by replacing "Service" with "Servic" so that the mobile could not interpret the message. When running on the emulator we got a NumberFormatException with the stack trace in the console window. However, this did not affect the operation of the application as the error was caused by the emulator.
Type of error:	None
Test approved:	Yes

Table 10.25: Test result for ST7

10.5 Summary

During the code review we found that most of the requirements were fulfilled. There are, however, some of the requirements concerning Liberty conformance that are not completely covered, but as we have explained in the comments on the code review results these requirements are not relevant to the functionality of the prototype.

Another problem was the placement of the commands on the selection keys. We found that this varied depending on the MIDP implementation. Thus, we would have to make a separate code version for each type of mobile to achieve optimal placement of the commands on all devices. As our application is only a prototype we do not think this is necessary, but future implementations should consider it.

All in all, we were quite pleased with the results of the system test. All test cases were completed successfully within the time limit of two minutes. However, we think the time to complete a service request would be shorter in real life. This is because the test cases required a lot of input of different usernames and passwords, which was very time consuming. In a real setting, there will usually be only one user per mobile. Thus, he will only need to enter his password as his username is automatically stored at the first successful login and retrieved on start-up.

Although all the test cases were approved, we encountered an error that was not corrected. It occurred only during the testing with the emulator, and only if the WSP returned an invalid response causing the mobile to retry the service request. In this case the emulator would print a stack trace in the console indicating a `NumberFormatException` and the network monitor would not show the outgoing message.

However, this did not affect the execution in any way as the results were returned normally and the return message was shown correctly in the network monitor. It may be that the error was caused by a bug in the emulator, an assumption that was strengthened by the fact that the stack trace did not contain a reference to any of the application classes. Thus, the issue could be resolved by asking Sun to investigate whether they have such a bug in their emulator. We have not done this as we do not think we would receive an answer in time to include it in this report, but we recommend that it is done before a potential deployment of the application.

Part IV

Evaluation and Conclusion

11 Evaluation

11.1 Introduction

This evaluation only covers the development phase of the thesis since we have already had several smaller discussions in the preliminary study. When examining the theoretical basis, we needed to discuss the alternatives and make decisions before entering the development phase.

In this chapter, we will start by presenting the evaluation criteria which were made while making the project description. These have been moved here, since they are only used in this chapter. The criteria are used to evaluate the prototype and strong authentication design in the next two sections. Finally, we will summarise the development effort.

11.2 Evaluation criteria

In this section, we describe the five criteria which we will use to evaluate the prototype. These sum up what goals we had for the prototype, and the fulfillment of these will determine how successful we deem the application. We have given the criteria certain priorities to reflect their importance when considering their use in eGovernment and that we are using a mobile as the terminal.

- 1. *Secure*** When examining how secure the application is, the criterion will be whether it would get governmental approval. The application will be used when the government provides access to identity-related information for the citizens. There are very strict regulations for access to and management of this information.
- 2. *Convenient to use*** The application shall be practical to use so that users will prefer the mobile over the web. In most contexts, users will still prefer the web when both options are available, but the goal of the prototype is to show that in the future this will be a viable alternative to the web.
- 3. *Interoperable*** The application should easily be able to connect to other identity providers and web service providers which support the Liberty profiles. Furthermore, the application interface should be similar to that of existing web service providers so adding mobile support would go smoother.

4. **Compatible with most mobiles** The goal is for the application to work on all newer mobiles. The user group for a governmental application is very large and diverse so it is important to consider that the mobiles will probably range from state-of-the-art to outdated.
5. **Modifiable** Given that this is just a prototype, we find it important to consider the ease of which the application can be extended to further test its possibilities.

11.3 The prototype

The most important evaluation criteria for this kind of mobile application are convenience and security. Both these determine whether the application will be used or not. If the application is not convenient to use, the user will not use it, and if the application is not secure, the government will not let anybody use it.

Although not as essential as the first two, the last three criteria are also important when determining whether the government would deploy a system based on this application or not. These all affect how many citizens will be able to use the application, what they will be able to use the application for and how extensive modifications will have to be done to existing systems to start using the application. Below, we will evaluate for each criterion how well the application covers it and to what extent we feel we have succeeded in fulfilling each criterion.

SECURE

As was introduced in chapter 3 and further discussed in chapter 8, the HTTPS support in mobile phones is far from perfect. It may be that these issues will be resolved in future models, but until then developers need to be aware of these problems and possibly take other precautions.

In this part of the design, we decided to employ short-lived server certificates to lessen the problem and did not use message encryption. The message encryption was omitted because we felt that it would affect the performance too much in relation to the security level of the information involved. We still think that this was the correct decision and that the short-lived server certificates will work well, although we did not use this approach during the testing.

Considering this and the fact that we used the *Requirements specification for PKI for the public sector* [34] where applicable, we believe that the application features adequate security to be approved for use by the government. Thus, this criterion is fulfilled for the prototype.

CONVENIENT TO USE

When examining the convenience of the application, the goal is to figure out what aspects will affect the user's decision on whether he will use this application again. Getting the information he wants is the most important, but for today's user the speed is most certainly an issue. For it to be a viable alternative, it must be just as quick as using the web, or it will be of no interest to the user. Of course, there is also the scenario where the user does not have web access. Then, the issue will be whether the application is so slow that the user feels he can do without the information rather than waiting.

Performance

The aspect of application performance has major implications for the usability. A significant amount of the time before getting results is spent waiting while sending messages to the supporting systems and parsing the results. By omitting message encryption, we feel we have kept the security measures to a minimum of what was possible without compromising the security. The test results from ST1, where we ran through a normal execution, gives a fairly good estimate of how long it takes to get the wanted information. The test ran in 1.38 minutes and this included the log-in, choice of provider and a look at the three available services. We think this is a highly acceptable time for a mobile application. For a normal user, the application would be even faster after the first time because the username is saved.

If extended to include more providers and services, the messages would be larger and it might take longer to receive and process the messages then. However, there is a lot of processing which must be done regardless of how much data is sent, so even if we doubled the amount of payload, the processing would be nowhere near double. Also, as we discussed in the implementation chapter and showed in the referenced benchmark tests, newer mobiles have more processing power. These tests indicated an increased performance of 2-3 times compared to the test phone, which would probably speed up the application.

The test phone communicates via the GSM network⁴⁰, using GPRS⁴¹, whereas many new phones use EDGE⁴¹ or UMTS⁴¹. According to Telenor, "UMTS is a brand new mobile network, which allows data to be transferred 8-10 times faster than in the GSM network"⁴² [79]. Our largest message in the prototype is 7,5 KB⁴³, which is considerable when sent over GPRS, but acceptable when using UMTS. Therefore, it is feasible to transfer larger messages when using a faster network. The only issue then would be whether the mobile could process these messages, but as we have mentioned, it is possible to increase the payload without much increase in processing.

⁴⁰"GSM is an open, digital cellular technology used for transmitting mobile voice and data services." [77]

⁴¹GPRS, EDGE and UMTS are different ways of transferring data in a mobile network. *Data Capabilities: GPRS to HSDPA* [78] lists the "average user throughputs for file downloads" for the General Packet Radio Service (GPRS) as 30-40 kbps, for the Enhanced Data rates for GSM Evolution (EDGE) as 100-130 kbps and for the Universal Mobile Telecommunications System (UMTS) as 220-320 kbps.

⁴²Translated from Norwegian.

⁴³This is the size of a response from the DS to the mobile for a user with one service offering.

Usability

The only thing we are not pleased with is how the buttons are placed. It now works the way we want on the Sony Ericsson test mobile, but not on the emulator. We know exactly how the button priorities should be set in order to fix it on the emulator, but what should have been done, if it had higher priority, is to fix it for all mobiles. This should be solved either by checking for the type of mobile at runtime or making one JAR file for each major brand of mobiles. If it was done at runtime, we could have specified the pattern to use for each brand so that the buttons were placed according to each phone's implementation of the GUI elements. This was given a low priority due to its low importance for proving the concept, and the fact that we would need to borrow one phone of each brand if we wanted to support them all.

In order for us to claim that our prototype features high usability, we should perform a usability test where we invite a group of potential users to try the application. This has not been done since we feel that the proof-of-concept does not have sufficient functionality to justify the time-consumption such a test would require. We would, however, strongly recommend such a test before a deployment can be considered.

Given that the mobile is something most people keep with them at all times, it is probable that many would prefer using the mobile to access the services even if they have access to a computer. We have tried to keep the user input to a minimum and to keep the processing time down, which we feel resulted in an acceptable solution. Hence, we will say that the criterion of convenience is adequately fulfilled for the prototype.

INTEROPERABLE

Interoperability is one of the main goals of Liberty. It was also one of the things we had in mind when we designed the WSP request/response schemas. Thus, the mobile application should be able to communicate with any WSP that uses these schemas and any IdP or DS that conform to their respective profiles in *Liberty ID-WSF Static Conformance Requirements* [58]. However, it should be noted that a few modifications may be necessary if the other IdP or DS require some of the Liberty requirements that were omitted in this version. The omitted requirements are listed and explained in the results of the code review in section 10.4.1.

A negative aspect of the proof-of-concept is that we did not show that the application actually works with an existing IdP or WSP. As we mentioned in section 8.4 when designing the identity provider, we tried to set up Sun Java System Access Manager, but eventually we had to make our own. However, the design of the application and WSP was made while waiting for Kantega to help us with the Access Manager. This means that every part of the design was made with the aim of using it with the Access Manager and we have not done any changes after deciding to make our own.

It was the plan all along to make the supporting systems fairly simple, but quite a lot of the functionality of a proper system was necessary to make it work. Rather than just making pre-set messages that were sent back no matter what came in - meaning no parsing, verification etc. - we designed a proper system, but with a few "shortcuts". One

example of such a simplification is the storing of user profiles in XML files, but this is an internal function of the supporting system and has no effect on the mobile application.

Despite of a few changes needed before deploying a proper version of this application, we think that the current level of interoperability is good enough to show that the prototype fulfils its purpose. Hence, this criterion is covered.

COMPATIBLE WITH MOST MOBILES

By not requiring any optional packages for J2ME, we believe that the application will be compatible with all mobile phones that support MIDP 2.0. As mentioned in chapter 3, this version of MIDP is already supported by the majority of mobile phones in Norway, and the number is rising. However, due to the nature of this project, our budget is quite small. Hence, we could only test the application on one phone model and the emulator. This means that, we cannot say with absolute certainty that the application will work on all phones with MIDP 2.0, but we strongly believe that it will.

There is another factor than the MIDP support which must be taken into account. The size of the application will also affect its compatibility, as the storage capacity of mobile phones vary between models. Our mobile application is about 150 KB, which is not a daunting size when we consider that many modern mobiles have support for expanding their storage space with memory cards. These often have a capacity of 1Gb, but the size of the application may still be considerable for some older models.

If message encryption and/or signing is introduced in a later version, it will require additional third party libraries that will increase the size of the application. In this case, it is necessary to use an obfuscator⁴⁴ to process the finished JAR file and remove as much redundancy as possible. This will shrink the JAR considerably and may make the difference between being able to use the application on a particular phone model and not being able to do so. In hindsight, we believe that it would be wise to use the obfuscator on the prototype as well.

Providing that our belief about being able to run the application on other phone models is true, this criterion should be fulfilled. However, we would have preferred to test this properly and recommend that this is done before a deployment of the application.

MODIFIABLE

During the design, we mostly thought about the modifiability when we constructed the language support. This decision was based on the assumption that at least one other language would be added in the case of a deployment. As Norway has two official languages, the eGovernment systems are required to support both. We chose differently because this report is written in English. Thus, it was more important to support this language. To compensate, we therefore isolated the language classes as much as possible so that the addition of another language will take a minimal amount of time.

⁴⁴An obfuscator, such as ProGuard [80], will remove code that is not used and make names that are short and incomprehensible. This saves space and makes it harder to recreate the code.

The application is modifiable in other ways as well. For instance, it would not require too much work to alter the request/response schema used by the WSP. As the message handling is contained in classes according to the entity with which it communicates, such modifications will be limited to one class only. Another possibility would be to add support for single sign-on between different WSPs. This may require the addition of another messaging class that could handle the obtaining of new tokens when the old ones had expired, but otherwise the alterations would be insignificant.

Furthermore, the addition of new service types would be possible. For instance, the application could support services where the user was allowed to send information back to the WSP. This would require changes to a few of the classes, but these changes would not be very time consuming. However, the new services may need additional security such as message encryption. This could mean that the performance of the application would deteriorate, thus reducing the convenience. This would then have to be considered in light of the increased convenience of being able to use the same application for more purposes.

Although only the language support was designed with this in mind, we are fairly pleased with the modifiability of the rest of the application as well. Therefore, this criterion is well fulfilled.

11.4 Strong authentication design

Although the strong authentication part of the design has not been implemented, it will undergo the same evaluation as the rest of the application. The difference will be that, rather than evaluating how the implemented system works, we will evaluate how we think the design will work with the rest of the prototype, based on the experiences we have obtained during the implementation of the current version. Modifiability is excluded from this evaluation since it is a quality that is considered when creating a detailed design - which we have not done. This is due to the assignment text which only required a high-level architectural description.

SECURE

As we mentioned in chapter 5, there is some doubt whether the mobile phone is sufficient proof of possession when it is used as the terminal. We consider the mobile to be a very personal item which means there is little chance of a malicious person using the phone. Even if a person should gain access to a mobile that is already turned on, the possibility of using the application is slim since this person will not have the other factor which is the code. The only chance for the malicious user to gain access is guessing the code, but the identity provider will block the private key if the wrong code is entered too many times. Then, the mobile would be useless as a token.

With this type of authentication, the user will be able to access content with a higher security level. Therefore, we have to look at the HTTPS support of mobile phones again and ask if it is sufficient to get governmental approval. This time a cryptographic API has already been imported in the application and the threshold of applying message encryption has therefore been lowered. However, with the previously mentioned performance issues, this may be yet another hold up in the execution.

As opposed to in the weak authentication design, we have devised a solution to the secure random problem in this design. This will not help with the poor HTTPS support, but will be necessary when the user's key pair is generated. Therefore, this measure helps provide better security in the application, although it is not enough to cover all aspects.

Like we concluded in chapter 8, the issue of HTTPS support will have to be reconsidered before an implementation of the solution. It may be that this issue will be solved by the phone manufacturers, but for now it is doubtful whether this solution will be secure enough for use by the government, and this criterion can therefore not be considered completely fulfilled.

CONVENIENT TO USE

When designing the strong authentication part of the application, one of our goals was that the user should only need the phone to authenticate. This would make the authentication as convenient as possible, since the user would have everything he needed with him when he wanted to use the application. In the solution we designed, the key and certificate generation process is the only thing that cannot be done without anything more than the phone itself. We accepted this in our solution, as it will probably be done just once a year or if the user forgets his password.

Unfortunately, we suspect that the strong authentication will be yet another drag on the application's performance, thus reducing the convenience of using the application. The key and certificate generation will probably be a bit slow, but that is acceptable since it is only done when the certificate needs to be renewed. A more serious concern is the fact that the authentication may be quite slow as well. This makes it important to introduce single sign-on, so that the user will get more results from each authentication. It may also be that the design will work well on a new phone, since these have much better performance than the one we used during testing.

Except from the added user input during key and certificate generation, the only extra user input will be the random seed gatherer. This may annoy some of the users, but as it will only have to be done the first time the application runs, we do not think it will deteriorate the convenience of using the application. It is important that the user understands that he only has to do this the first time. Otherwise, he may be so annoyed that he refuses to start the application again, and he would not discover that he will not be asked to do this the second time.

The performance may be a problem, especially if more security measures have to be introduced as well. As long as we have not tested the design, we cannot be certain, thus this should be tried out on the prototype before a large-scale implementation can be considered. If we disregard the performance issue, we are well pleased with the convenience of this solution.

INTEROPERABLE

The strong authentication method can be used with any IdP that supports the SASL mechanism and trusts the certificate authority which signed the user's certificate. The certificate signing has to be done by a provider that has the correct interface and knows the user's OTP pattern, but there are no other requirements to this provider's inner workings. Thus, the solution introduces no new interoperability issues and the criterion is well fulfilled.

COMPATIBLE WITH MOST MOBILES

The solution still requires no optional packages. As we explained previously, this should mean that it would work on any phone model. However, the size of the application will be an even bigger issue as the inclusion of a cryptographic API and more classes will increase its required storage space. The obfuscator will then be even more important to use, but may not be sufficient in all cases. Most mobiles will have enough space to store the application, but the user may be reluctant to give up the space he otherwise could have used for games, music or images - unless the application proves very useful and convenient to him. Therefore, we arrive at the same conclusion as before; the criterion is fulfilled, but with a few reservations.

11.5 Summary

As we have shown in the discussions of this chapter, the prototype fulfils most of the evaluation criteria in a satisfactory way. This means that we are fairly certain that it would be received well by both the government and the users if it was to be deployed, possibly with a few modifications to refine it and to add support for the other Norwegian language. Therefore we conclude that it has served its purpose as a proof-of-concept.

When it comes to the strong authentication design there is room for a few improvements. By implementing a prototype incorporating this design, some of the questions regarding its fulfillment of the criteria may be answered. Its greatest weakness is the HTTPS support, which will have to be carefully considered in the event of an implementation. Other than that, we think the design will work well and that most of the criteria will be fulfilled.

12 Further work

Throughout this project the focus has been on proving the concept that a mobile can connect to a Liberty Circle of Trust. This has resulted in a prototype where several features had to be left out to narrow the scope of the assignment. The amount of work required to describe important background technology and discuss the decisions we had to make, meant that making a complete application would not be possible. This chapter therefore describes some of the areas where the prototype could be extended and gives some thoughts about what should be implemented before the prototype could be deployed.

12.1 Extending the prototype

The finished prototype is a simple application, which shows how to access an eGovernment service on a mobile phone. The solution covers most of the requirements in the Liberty profiles and is ready to be tried with one service. In order to get someone interested in deploying this application, the next step will be to add more functionality so that the prototype transforms from just proving a concept into an proper example application. Below are three ideas which we think will be the next logical steps in the prototype's further development.

More providers To complete the prototype a few more service providers should be added. As we showed in the testing, the application can handle more than one provider so it should not be too time-consuming to copy the code of the WSP. Then, we would have more providers, and these could offer other service types.

Other services To make the prototype more appealing it would be smart to show that this application can support more than one type of service. In addition to retrieving information, it is also possible to allow the user to send information to the providers. This can for instance be to change his address with the national registration office. There are some restrictions since it is more cumbersome to write on a mobile, so some services will just not be suitable.

Strong authentication Before this could be implemented an even more extensive consideration of the HTTPS support must be performed. When moving into the topic of strong authentication, more sensitive services will be introduced and then the HTTPS support might not be good enough for governmental approval. When this issue has been scrutinised, the implementation of strong authentication will involve making the random seed generator, the key and certificate generation and the signing functionality. Naturally, if the prototype should be used with our IdP and WSP, these must also be updated slightly.

12.2 Deploying the application

The next step after adding necessary extensions will be to find someone interested in deploying the application. Potential developers will need to do some work to adapt existing services into a format suitable for a mobile, but few changes are required for the application. The proper identity provider would probably not need changes, but the WSP would need an add-on module if it does not support mobiles. Our WSP code could potentially be used for message creation and parsing.

Government This application has throughout the report been considered as an alternative that the government in Norway could use together with "MinSide". The governmental portal has considered using Liberty in the second version, which means that our application would be supported.

Private enterprise There is probably a better chance of the private sector implementing and releasing something like our application because they can profit from this type of service. They will most certainly charge people for using it, but a likely scenario would be implementing it with some free services to tempt the users into using the services that cost money. These will probably come at the same rate as today's SMS services, but it depends on the service content and utility value.

Collaboration If the application could have multiple functions, the cost of deploying the application could potentially be divided onto several entities. Our application can use most authentication and discovery services and communicate with all web service providers that uses an adapted schema. Therefore, it is possible to have two identity providers where one is used for eGovernment services and the other is used in the private sector. This would benefit the user, but it is not likely that the government would want to cooperate with private companies.

13 Conclusion

To summarise the findings of this thesis, we will now look back at the research questions in chapter 2. The main focus of this project has been to demonstrate how it is possible to use a mobile phone in a Liberty Circle of Trust. We have looked at some of the work done in other projects concerning this topic, but the most important part was the development of a proof-of-concept. This prototype demonstrated how non-sensitive eGovernment services could be made available to a mobile phone if the phone was Liberty-enabled. The solution will need some refinement before it can be used in real life, but it has proved that it is possible to access services this way.

Another important aspect has been to find out how eGovernment services could be made available in a secure way by using Liberty ID-WSF. Due to the problems with the HTTPS support of MIDP, we had doubts whether the prototype would be secure enough for use with eGovernment services. This was discussed in the evaluation, and we concluded that it would be sufficiently secure for the non-sensitive services.

Additionally, the project has explored the possibilities of strong authentication using only mobile phones. It evaluated some of the existing products offering this type of functionality, but although two of the products looked promising we concluded that to devise our own solution would be better. The reason for this is that one was operator-dependent and the other had no publicly available information. The latter was caused by a pending patent application. A part of the design was then devoted to our solution, and the results were evaluated, even though this functionality was excluded from the implementation. We found the design to be good, except the HTTPS support which may be inadequate.

References

- [1] *Liberty Technical Glossary*. Liberty Alliance, Edition 2.0-3 (accessed January 2006).
<http://www.projectliberty.org/specs/draft-liberty-glossary-v2.0-03.pdf>
- [2] *Trust in Cyberspace*. Schneider F.B. (editor), National Academy Press, 1999, ISBN 0-309-06558-5
- [3] *Network Security: The Complete Reference*. Bragg R., Rhodes-Ousley M. & Strassberg K., 1. edition, McGraw-Hill/Osborne, 2004, ISBN 0-07-222697-8
- [4] *Network Security Essentials - Applications and Standards*. Stallings W., 2. international edition, Prentice Hall, 2003, ISBN 0-13-120271-5
- [5] *Building Secure Software: How to Avoid Security Problems the Right Way*. Viega J. & McGraw G., 1. edition, Addison-Wesley, 2002, ISBN 0-201-72152-X
- [6] *Definition of Credentials*. M-Tech (accessed January 2006).
<http://www.mtechit.com/concepts/credentials.html>
- [7] *It's All About Authentication*. Graham D., SANS March 2005 (accessed January 2006).
<http://www.sans.org/rr/whitepapers/authentication/1070.php>
- [8] *webservices.xml.com: What Is Service-Oriented Architecture*. He H., September 2003 (accessed January 2006).
<http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html>
- [9] *Extensible Markup Language (XML)*. W3C, April 2006 (accessed January 2006).
<http://www.w3.org/XML/>
- [10] *SOAP Version 1.2 Usage Scenarios*. W3C, Edition 1.2, July 2003 (accessed January 2006).
<http://www.w3.org/TR/xmlp-scenarios/>
- [11] *SOAP Specifications*. W3C, June 2003 (accessed January 2006).
<http://www.w3.org/TR/soap/>
- [12] *SOAP Version 1.2 Part 0: Primer*. W3C, Edition 1.2, June 2003 (accessed January 2006).
<http://www.w3.org/TR/2003/REC-soap12-part0-20030624/>
- [13] *Web Service Definition Language (WSDL)*. W3C, March 2001 (accessed January 2006).
<http://www.w3.org/TR/wsdl>
- [14] *Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS, March 2005 (accessed January 2006).
<http://docs.oasis-open.org/security/saml/v2.0/saml-glossary-2.0-os.pdf>

- [15] *Architecture and design for central authentication and authorization in an on-demand utility environment patent*. IBM Corporation, August 2005 (accessed January 2006).
<http://www.freshpatents.com/Architecture-and-design-for-central-authentication-and-authorization-in-an-on-demand-utility-environment-dt20050825ptan20050188420.php>
- [16] *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. IETF, April 2002 (accessed January 2006).
<http://www.ietf.org/rfc/rfc3280.txt>
- [17] *XML-Signature Syntax and Processing*. W3C, February 2002 (accessed January 2006).
<http://www.w3.org/TR/xmlsig-core/>
- [18] *XML Encryption Syntax and Processing*. W3C, December 2002 (accessed January 2006).
<http://www.w3.org/TR/xmlenc-core/>
- [19] *Hypertext Transfer Protocol – HTTP/1.1*. Network Working Group, Edition 1.1, June 1999 (accessed January 2006).
<http://www.ietf.org/rfc/rfc2616.txt>
- [20] *What is a Subscriber Identity Module (SIM)?*. GSM Security (accessed January 2006).
<http://www.gsm-security.net/faq/subscriber-identity-module-sim.shtml>
- [21] *Webster's encyclopedic unabridged dictionary for the English language*. Random House Value Publishing, 1996, ISBN 0-517-15026-3
- [22] *Mobile Information Device Profile*. Sun Microsystems (accessed January 2006).
<http://java.sun.com/products/midp/midp-ds.pdf>
- [23] *The K Virtual Machine*. Sun Microsystems (accessed January 2006).
<http://java.sun.com/products/cldc/ds/>
- [24] *Java 2 Platform, Micro Edition*. Sun Microsystems (accessed January 2006).
<http://java.sun.com/j2me/j2me-ds.pdf>
- [25] *Gjør det enkelt å lage nye mobiltjenester*. Digi.no, Ernes A.K.B., March 2006 (accessed April 2006).
<http://www.digi.no/php/art.php?id=297964>
- [26] *What's New in MIDP 2.0*. Sun Microsystems (accessed January 2006).
<http://java.sun.com/products/midp/whatsnew.html>
- [27] *Wireless Java Security - Understanding the issues*. Povey D., SYS-CON Media, (accessed April 2006).
<http://www2.sys-con.com/ITSG/virtualcd/Java/archives/0802/povey/index.html>

- [28] *Java™ 2 Platform, Micro Edition (J2ME™) Web Services Specification*. Sun Microsystems (accessed March 2006).
http://java.sun.com/j2me/docs/j2me_jsr172.pdf
- [29] *Security and Trust Services APIs For the Java™ 2 Platform, Micro Edition*. Sun Microsystems (accessed March 2006).
http://java.sun.com/j2me/docs/sec_trust-177.pdf
- [30] *European Commission - Information Society - eEurope 2005*. (accessed April 2006).
http://ec.europa.eu/information_society/europe/2005/index_en.htm
- [31] *UNPAN 2004 E-government - Global Survey of E-government*. (accessed April 2006).
<http://www.unpan.org/egovernment3.asp>
- [32] *FAD - IT-politikk / eNorge*. (accessed April 2006).
<http://www.enorge.org>
- [33] *Altinn - Startsiden*. (accessed April 2006).
<http://www.altinn.no>
- [34] *Requirements specification for PKI for the public sector*. Ministry of Modernisation (Norway), Edition 1.02, January 2005 (accessed April 2006).
<http://odin.dep.no/filarkiv/250615/Kravspekk-engelsk-versjon.pdf>
- [35] *Lov om elektronisk signatur (esignaturloven)*. Ministry of Trade and Industry (Norway), July 2005 (accessed April 2006).
<http://www.lovdatab.no/cgi-wift/wiftldles?doc=/usr/www/lovdatab/all/nl-20010615-081.html&dep=alle&kort+,+titt=lov+om+elektronisk+signatur&>
- [36] *Leveranse oppgave 1 Anbefalte sertifikatprofiler for personsertifikater og virksomhetssertifikater*. SEID-prosjektet, Edition 1.02, February 2005 (accessed April 2006).
http://odin.dep.no/filarkiv/265358/SEID_Leveranse_1_-_v1.02.pdf
- [37] *Lov om behandling av personopplysninger (personopplysningsloven)*. Ministry of Justice and the Police (Norway), January 2001 (accessed April 2006).
<http://www.lovdatab.no/all/nl-20000414-031.html>
- [38] *Implementasjonsguide Registerinformasjon Min Side*. Teepo A.A. & Kværnø O., Moderniseringsdirektoratet og skattedirektoratet, August 2005 (confidential - not publicly accessible).
- [39] *Liberty ID-FF Authentication Context Specification*. Liberty Alliance, Edition 2.0-01 (accessed March 2006).
<http://www.projectliberty.org/specs/draft-liberty-authentication-context-v2.0-01.pdf>
- [40] *Public Key Infrastructure*. U.S. Air Force, August 2000 (accessed March 2006).
<http://www.e-publishing.af.mil/slides/PKI-Forms%20Conference.ppt>

- [41] ***MIDP Application Security 3: Authentication in MIDP***. Knudsen J., Sun Microsystems, December 2002 (accessed March 2006).
<http://developers.sun.com/techttopics/mobility/midp/articles/security3/>
- [42] ***Liberty Alliance Project - Frequently Asked Questions***. Liberty Alliance (accessed May 2006).
<http://www.projectliberty.org/about/faq.php>
- [43] ***OATH Reference Architecture Release 1.0***. Initiative for Open Authentication, Edition 1.0, 2005.
- [44] ***An Industry Roadmap for Open Strong Authentication***. Initiative for Open Authentication (accessed March 2006).
- [45] ***802.1X Offers Authentication and Key Management***. Geier J., Wi-Fi Planet, May 2002 (accessed March 2006).
<http://www.wi-fiplanet.com/tutorials/article.php/1041171>
- [46] ***ZebSign PersonID Certificate Policy***. ZebSign, Edition 1.11, November 2005 (accessed March 2006).
[http://www.zesign.no/upload/Dokumenter/Produkter%20og%20tjenester/Certificate%20Policy%20\(CP\)/ZebSign%20Person%20ID%20Policy%20v1.11.pdf](http://www.zesign.no/upload/Dokumenter/Produkter%20og%20tjenester/Certificate%20Policy%20(CP)/ZebSign%20Person%20ID%20Policy%20v1.11.pdf)
- [47] ***Telenor - MobilHandel***. Telenor ASA (accessed March 2006).
<http://telenormobil.no/mobilhandel/tjenesten/sikkerhet.do>
- [48] ***ZebSign AS***. ZebSign AS (accessed March 2006).
http://www.zesign.no/templates/Page____141.aspx
- [49] ***Brønnøysundregistrene***. Brønnøysundregistrene (accessed March 2006).
http://www.brreg.no/sikkerhetsportal/teknologi_spm.html
- [50] ***Sikker identifisering og betaling via mobil***. Buypass, December 2005 (accessed March 2006).
<http://www.buypass.no/Buypass%20mobile%20v1.pdf>
- [51] ***MobiSecure Soft Token Product Overview***. Diversinet (accessed March 2006).
http://www.diversinet.com/perspectives/MobiSecure_Product_Overview_Perspective.pdf
- [52] ***PortWise 4.5 Strong authentication from PortWise***. Hägerö P., 2005 (accessed March 2006).
<http://www.portwise.com/downloadcenter/docs/whitepapers/PortWise%204%205%20Strong%20Authentication.pdf>
- [53] ***Liberty ID-WSF SOAP Binding Specification***. Liberty Alliance, Edition 1.2 (accessed March 2006).
<http://www.projectliberty.org/specs/liberty-idwsf-soap-binding-v1.2.pdf>
- [54] ***Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)***. OASIS, February 2006 (accessed March 2006).
<http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>

- [55] *Liberty Metadata Description and Discovery Specification*. Liberty Alliance, Edition 1.1 (accessed March 2006).
<http://www.projectliberty.org/specs/liberty-metadata-v1.1.pdf>
- [56] *Liberty ID-WSF Discovery Service Specification*. Liberty Alliance, Edition 1.2 (accessed March 2006).
<https://www.projectliberty.org/specs/liberty-idwsf-disco-svc-v1.2.pdf>
- [57] *Liberty ID-WSF Authentication Service and Single Sign-On Service Specification*. Liberty Alliance, Edition 1.1 (accessed March 2006).
<http://www.projectliberty.org/specs/liberty-idwsf-authn-svc-v1.1.pdf>
- [58] *Liberty ID-WSF 1.1 Static Conformance Requirements*. Liberty Alliance, Edition 1.0 (accessed March 2006).
<http://www.projectliberty.org/specs/liberty-idwsf-1.1-scr-v1.0.pdf>
- [59] *Liberty Alliance Project - Conformant Products*. Liberty Alliance (accessed March 2006).
http://www.projectliberty.org/activities/conformant_products.php
- [60] *Introduction To Web Services In Nokia Devices*. Nokia, Edition 1.0, June 2004 (accessed March 2006).
http://www.forum.nokia.com/info/sw.nokia.com/id/6fc34af5-877d-4877-bfef-c9144c023a7c/Nokia_Web_Services_v1.0_en.pdf.html
- [61] *Deploying Mobile Web Services using Liberty Alliance's Identity Web Services Framework (ID-WSF)*. Nokia and Sun Microsystems (accessed March 2006).
http://www.sun.com/software/products/identity/sun_nokia_id-wsf_deployment_paper.pdf
- [62] *OMA Enabler Release and Specifications - OMA Web Services Network Identity*. Open Mobile Alliance, December 2005 (accessed March 2006).
http://www.openmobilealliance.org/release_program/owser_ni_v1.0.html
- [63] *IEEE Std 830-1998, Recommended Practice for Software Requirements Specifications*. IEEE, June 1998 (accessed March 2006).
<http://ieeexplore.ieee.org/xpl/tocresult.jsp?isNumber=15571>
- [64] *Wireless Application Protocol Public Key Infrastructure Definition*. Open Mobile Alliance, Edition 24-Apr-2001 (accessed April 2006).
http://www.openmobilealliance.org/release_program/docs/WPKI/WAP-217-WPKI-20010424-a.pdf
- [65] *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Fowler M. & Scott K., 2. edition, Addison-Wesley, 1999, ISBN 0-201-65783-X
- [66] *kSOAP 2*. Open source software (accessed April 2006).
<http://ksoap2.sourceforge.net>

- [67] **kXML 2**. Open source software (accessed April 2006).
<http://kxml.sourceforge.net/kxml2>
- [68] **Liberty ID-WSF Security and Privacy Overview**. Liberty Alliance, Edition 1.0 (accessed April 2006).
<http://www.projectliberty.org/specs/liberty-idwsf-security-privacy-overview-v1.0.pdf>
- [69] **bouncycastle.org**. Open source software (accessed April 2006).
<http://www.bouncycastle.org/specifications.html>
- [70] **ISO/IEC 9798-3 Authentication SASL Mechanism**. Zuccherato R. & Nystrom M., August 2001 (accessed April 2006).
<http://www.rfc-archive.org/getrfc.php?rfc=3163>
- [71] **Code Conventions for the Java™ Programming Language**. Sun Microsystems, April 1999 (accessed March 2006).
<http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html>
- [72] **How to Write Doc Comments for the Javadoc Tool**. Sun Microsystems (accessed March 2006).
<http://java.sun.com/j2se/javadoc/writingdoccomments/index.html>
- [73] **Networking, User Experience, and Threads**. Knudsen J., Sun Microsystems January 2002 (accessed April 2006).
<http://developers.sun.com/techtoc/mobility/midp/articles/threading/>
- [74] **Amobil :: Test - Nokia 3250**. Amobil.no, Valle M., April 2006 (accessed May 2006).
http://www.amobil.no/tester/mobiltelefoner/nokia_3250/25426/2
- [75] **Amobil :: Test - Sony Ericsson K700i**. Amobil.no, Øystein W Høie, June 2004 (accessed May 2006).
http://www.amobil.no/tester/mobiltelefoner/sony_ericsson_k700i/8314/3
- [76] **Amobil :: Hele www i lomma**. Amobil.no, Valle M., June 2006 (accessed June 2006).
http://www.amobil.no/nyheter/mobil-programvare/hele_www_i_lomma/26764
- [77] **GSM World - GSM Technology**. GSM Association (accessed June 2006).
<http://www.gsmworld.com/technology/what.shtml>
- [78] **Data Capabilities: GPRS to HSDPA**. Rysavy P., September 2004 (accessed June 2006).
http://www.3gamericas.org/pdfs/rysavy_data_sept2004.pdf
- [79] **Telenor - Mer om UMTS og EDGE**. Telenor (accessed June 2006).
<http://telenormobil.no/tjenester/3g/merom.do>
- [80] **ProGuard**. Open source software (accessed May 2006).
<http://proguard.sourceforge.net>

Part V

Appendices

A Assignment text

This appendix features the exact assignment text as given by Kantega.

TOPIC

The Liberty specifications (www.projectliberty.org) describe how user agents like web browsers and mobile phones can be used for accessing Circles of Trust.

As mobile data bandwidth is increasing, it becomes more desirable to offer content and services formatted for mobile phones. The adaption of Liberty standards for mobile phones makes it possible to retain the advantages of single sign-on, account linking etc in a mobile environment.

Some of the services which are considered require strong authentication. Many mobile phones now support Java. This makes it possible to develop one-time password (OTP) algorithms for executing in the mobile phone's Java VM. In effect, this turns the mobile phone into an OTP token device.

Other ways to support strong authentication would be to use the mobile phone as storage device for key pairs and digital certificates. The SIM card of the phone is a candidate for storing the keys and certificates. Access to the private key has to be secured by some mechanism.

Supporting strong authentication and Liberty on a mobile phone opens up for a raft of new services. Some of the services which can be considered are: financial transactions, online payment, personal health records etc.

Web Services is rapidly becoming the preferred method of distributing services. Liberty Alliance has developed the Web Services Framework (ID-WSF), which describes the interaction between Web Service Consumers (for instance portals), Web Service Providers and Identity Providers. The goal for ID-WSF is to make it easier to develop secure identity-based Web Services.

In Norway, Web Services and the related paradigm of service-oriented architecture (SOA) is used for developing new eGovernment services. A central concept is register services, which are developed by government agencies for presentation in the Norwegian citizen portal "MinSide" (MyPage).

OBJECTIVE

The objectives of the thesis are:

- To provide a description of relevant standards concerning the topic and an assessment and recommendation of which standards should be used.
- To survey and describe available technologies for strong authentication on a mobile phone.
 - The survey should include the initiative for open authentication (www.openauthentication.org) which also develops reference architecture and specifications for strong authentication on mobile phones.
- To survey and describe how a mobile phone can be supported in a Liberty-type Circle of Trust.
- To describe how eGovernment services can be made available in a mobile environment in a secure way, using Liberty ID-WSF.
- To develop a proof-of-concept (PoC):
 - The PoC shall be a working prototype demonstrating the use of a mobile phone for accessing an eGovernment register service.
 - The PoC shall demonstrate the interaction between a Liberty-enabled Web Service Provider, a Liberty-enabled Web Service Consumer (the mobile phone) and a Liberty-enabled Identity Provider.
 - The register service "Lånkassen" (State Education Loan Fund) will be used for demonstration purposes.
 - As the information is non-sensitive, there will be no need for strong authentication to prove the concept. However, the thesis shall include a design of how strong authentication can be supported by the mobile phone. The recommended solution shall be described at a high architectural level.
 - If the proof-of-concept cannot be implemented due to lack of time or for other reasons, the thesis should include a technical design which shows how a proof-of-concept could be implemented.
 - If there are technical considerations which make it impossible to implement the proof-of-concept, these considerations should be documented and discussed in the thesis.

B Notation

B.1 State diagram

The state diagrams are used to show all possible states of a system and how events results in change of state. Table B.1 gives an explanation of the symbols used.

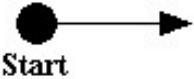
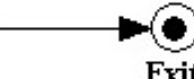



Symbol	Description
 <p>Start</p>	<p>Start</p> <p>This is the start-up state which is entered when the application opens. The following state will be entered automatically when the program begins.</p>
 <p>Exit</p>	<p>Exit</p> <p>This state is the final step which is reached when the application is closed. The symbol can be duplicated if necessary since it can be reached from several states.</p>
	<p>Main state</p> <p>This symbolises a state where the user can perform some action which is vital for the operation of the program. To move to another state from this requires user action (user pressing a selection key).</p>
	<p>Waiting state</p> <p>This is a state where all the user can do is wait for some process to finish. It is also possible to abort the operation and go back.</p>
	<p>Arrow</p> <p>The arrow symbolises a transition from one state to another. If the transition is due to a user action, a label will contain the title of the selection key. There could also be a condition which is symbolised by a text in square brackets. This holds the prerequisite which must be fulfilled before the transition.</p>

Table B.1: Notation for the state diagram

B.2 Package diagram

The package diagram is used to show how the classes are divided into packages, or groups according to functionality. The two different types of packages are illustrated in table B.2.

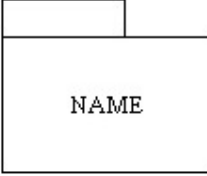
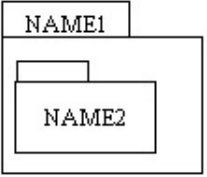
Symbol	Description
	<p>Package</p> <p>The package is a collection of Java classes which have similar functionality. The name of the package should say something about the main task of the package.</p>
	<p>Including package</p> <p>The including package is an upper level package in a hierarchy where one package contains one or several others.</p>

Table B.2: Notation for the package diagram

B.3 Class diagram

The class diagrams contains all the objects in the system and the properties and operations of these. It also shows the relationship between the objects. The main concepts of the diagram type are described in table B.3. There are some simplification which have been made to the diagrams in cases where it enhances readability. This includes omitting relationships, cardinality and long values. The values which are too long to have in the class diagram have been replaced with three dots ("...").

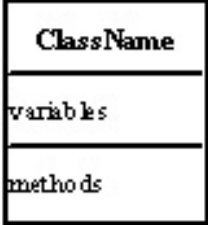
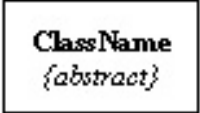
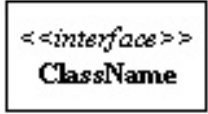



Symbol	Description
	<p>Class</p> <p>A class representation is a box consisting of three parts, where the top one must contain the name of the class, the one in the middle will contain any variables the class has and the bottom one has the methods.</p> <p>For the variables and constants we have chosen to include the following information: <i>[visibility] [name]: [data type]</i>.</p> <p>For the methods the chosen syntax is: <i>[visibility] [name]([list of parameters]): [return type]</i>, where the list of parameters is a comma-separated list of <i>[parameter name]:[date value]</i>.</p>
	<p>Abstract</p> <p>An abstract class is used when the object is an abstract concept and we will not make an instance of the class. It is used by making subclasses which inherit the properties.</p>
	<p>Interface</p> <p>An interface is an "empty" class which only defines the operations without implementing them. Thus, it consists of methods without any content in the method body.</p>
	<p>Association</p> <p>This relationship is drawn between two classes where one is using the other. We have here decided not to include multiplicity (number of objects participating in this relation) to remove unnecessary complexity.</p>
	<p>Generalisation</p> <p>This arrow is used when classes inherit properties from a superclass. All instances of the class where the arrow originates are also instances of the superclass and have all variables, methods and associations of the superclass. The exception is private methods and variables of the superclass which are not inherited.</p>
	<p>Realization</p> <p>This is used when a class implements a interface (which is the box on the side with the arrow head).</p>

Table B.3: Notation for the class diagram

B.4 Sequence diagram

The sequence diagram shows how the classes of the application communicate during its lifetime. Table B.4 explains the notation used in this type of diagrams.

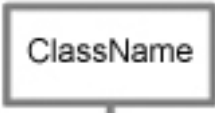
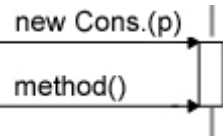
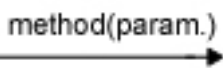
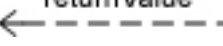
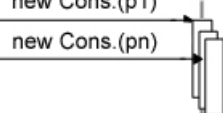
Symbol	Description
	<p>Class</p> <p>A class is represented by a box with the name of the class written within. Each class has a vertical dotted line beneath it which indicates the timeline of the application, with the most recent events at the bottom. When events that affect the class occur, their placement along this line gives an idea of the order in which they occur.</p>
	<p>Instance lifetime</p> <p>Whenever a new instance of a class is created, this will be indicated by a call to its constructor with the keyword new in front and the appearance of an empty white box on that class' timeline. The box illustrates the life of the instance and will continue down the timeline to the point where the instance has completed its work. This will be the point where the last method call to that class or the return of the last method call to that class is drawn.</p>
	<p>Method call</p> <p>This arrow indicates a method call on the instance in the end of the arrow by the instance in the start of the arrow. The method call is written as the name of the method followed by a set of parenthesis with a comma separated list of parameters within them. Each parameter is indicated with a keyword that give a clue to what the data contains. Whenever this is used as a parameter, it means that the instance which calls the method includes itself as a parameter in the method call.</p>
	<p>Method return</p> <p>In some cases the method calls will return the results to the caller. Then this arrow is used to indicate what kind of data that are passed back. The text above the arrow will give a clue to what these data contain.</p>
	<p>Several simultaneous instances</p> <p>In some cases it is necessary to create several simultaneous instances of one class. This is denoted as three white boxes on the class' timeline and two calls to its constructor with 1 and n respectively as the postfix to their parameters.</p>

Table B.4: Notation for the sequence diagram

C XML Schemas

C.1 Mobile Request

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  xmlns:tns=
    "http://fedoralab1.kantega.no:8080/requests/mymobilesite-request.xsd"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:disco="urn:liberty:disco:2003-08"
  targetNamespace=
    "http://fedoralab1.kantega.no:8080/requests/mymobilesite-request.xsd">

  <xsd:complexType name="InformationRequestType">
    <xsd:sequence>
      <xsd:group ref="disco:ResourceIDGroup"/>
      <xsd:element name="language" type="xsd:string" nillable="false"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="InformationRequest" type="InformationRequestType"/>
</xsd:schema>
```

C.2 WSP Response

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  xmlns:tns=
    "http://fedoralab1.kantega.no:8080/responses/mymobilesite-response.xsd"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace=
    "http://fedoralab1.kantega.no:8080/responses/mymobilesite-response.xsd">

  <xsd:complexType name="ServiceListType">
    <xsd:sequence>
      <xsd:element name="Service" type="ServiceType"/>
    </xsd:sequence>
  </xsd:complexType>
```

```

<xsd:complexType name="ServiceType">
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:sequence>
    <xsd:element name="ValueElement" type="ValueElementType"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ValueElementType">
  <xsd:attribute name="label" type="xsd:string" default=""/>
  <xsd:attribute name="value" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:element name="ServiceList" type="ServiceListType"/>
</xsd:schema>

```

C.3 WSP fault

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  xmlns:tns="http://fedoralab1.kantega.no:8080/faults/mymobilesite-faults.xsd"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://fedoralab1.kantega.no:8080/faults/mymobilesite-faults.xsd">

  <xsd:simpleType name="ErrorCodeType">
    <xsd:restriction base="xs:string">
      <xsd:enumeration value="unknownId"/>
      <xsd:enumeration value="requestTimedOut"/>
      <xsd:enumeration value="serviceUnavailable"/>
      <xsd:enumeration value="notAuthorized"/>
      <xsd:enumeration value="internalError"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:complexType name="MobileRegisterFaultType">
    <xsd:sequence>
      <xsd:element name="errorCode" type="ErrorCodeType"/>
      <xsd:element name="errorDescription" type="xsd:string" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:element name="MobileRegisterFault" type="tns:MobileRegisterFaultType"/>
</xsd:schema>

```

D WSDL

```

<?xml version="1.0" encoding="UTF-8"?>
<!--
=====
=== 1) Service specific name spaces. ===
=====
-->

<definitions
  xmlns:tns="http://fedoralab1.kantega.no:8080/abstract-mymobilesite-wsp/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:ns0="http://fedoralab1.kantega.no/requests/"
  xmlns:ns1="http://fedoralab1.kantega.no/responses/"
  xmlns:ns2="http://fedoralab1.kantega.no/faults/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:sb="urn:liberty:sb:2003-08"
  xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext"
  targetNamespace="http://fedoralab1.kantega.no:8080/abstract-mymobilesite-wsp/"
  name="abstract-mymobilesite-wsp">

<!--
=====
=== 2) Data types transmitted between the mobile phone and the WSP. ===
=====
-->

<types>
  <schema>

    <!-- Correlation header -->
    <xsd:import namespace="urn:liberty:sb:2003-08"
      schemaLocation=
        "http://www.projectliberty.org/specs/liberty-idwsf-soap-binding-v1.2.xsd"
      id="sb"/>

    <!-- WS-Security header -->
    <xsd:import namespace="http://schemas.xmlsoap.org/ws/2002/04/secext"
      schemaLocation=
        "http://schemas.xmlsoap.org/ws/2002/04/secext/secext.xsd"
      id="wsse"/>

    <!-- Request: -->
    <xsd:import namespace="http://fedoralab1.kantega.no/requests/"

```

```

    schemaLocation=
      "http://fedoralab1.kantega.no/requests/mymobilesite-request.xsd"
    id="ns0"/>

<!-- Response: -->
<xsd:import namespace="http://fedoralab1.kantega.no/responses/"
  schemaLocation=
    "http://fedoralab1.kantega.no/responses/mymobilesite-response.xsd"
  id="ns1"/>

<!-- Faults: -->
<xsd:import namespace="http://fedoralab1.kantega.no/faults/"
  schemaLocation=
    "http://fedoralab1.kantega.no/faults/mymobilesite-faults.xsd"
  id="ns2"/>

  </schema>
</types>

<!--
=====
=== 3) Messages transmitted between the mobile phone and the WSP. ===
=====
-->

<message name="GetRegisterInformationRequest">
  <part name="request" element="ns0:InformationRequest"/>
</message>
<message name="GetRegisterInformationResponse">
  <part name="response" element="ns1:ServiceList"/>
</message>
<message name="MobileRegisterFaultMessage">
  <part name="error" element="ns2:MobileRegisterFault"/>
</message>
<message name="CorrelationHeader">
  <part name="Correlation" element="sb:Correlation"/>
</message>
<message name="SecurityHeader">
  <part name="securityHeader" element="wsse:Security"/>
</message>

<!--
=====
=== 4) Interface and operations. ===
=====
-->

```

```

<portType name="MobileRegisterPortType">
  <operation name="getMobileRegisterInformation">
    <input name="inputMessage"
      message="tns:GetRegisterInformationRequest"/>
    <output name="outputMessage"
      message="tns:GetRegisterInformationResponse"/>
    <fault name="MobileRegisterFault"
      message="tns:MobileRegisterFaultMessage"/>
  </operation>
</portType>

<!--
=====
=== 5) How the service will be implemented.          ===
=====
-->

<binding name="MobileRegisterBinding" type="tns:MobileRegisterPortType">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http"/>

  <operation name="getMobileRegisterInformation">
    <soap:operation/>
    <input message="tns:GetRegisterInformationRequest">
      <soap:header message="tns:CorrelationHeader"
        part="correlationHeader" use="literal"/>
      <soap:header message="tns:SecurityHeader"
        part="securityHeader" use="literal"/>
      <soap:body parts="request" use="literal"/>
    </input>

    <output message="tns:GetRegisterInformationResponse">
      <soap:header message="tns:CorrelationHeader"
        part="correlationHeader" use="literal"/>
      <soap:body parts="request" use="literal"/>
    </output>

    <fault message="tns:MobileRegisterFaultMessage">
      <soap:fault name="MobileRegisterFault" use="literal"/>
    </fault>
  </operation>

</binding>
</definitions>

```


E Screen texts

Table E.1 contains the English and Norwegian translations of the texts used on labels and selection keys in the application. As can be seen in the table, the language names exist in only one language each. This is because we choose to display the name of each language in that language when the user is viewing the settings screen.

English	Norwegian
An error occurred	Det skjedde en feil
Available services	Tilgjengelige tjenester
Back	Tilbake
Exit	Avslutt
Help	Hjelp
Language	Språk
Login service	Innloggingstjeneste
Menu	Meny
MyMobileSite	MinMobilside
OK	OK
Password	Passord
Please wait	Vent litt
Save	Lagre
Service providers	Tjenestetilbydere
Settings	Innstillinger
Supported languages	Tilgjengelige språk
Username	Brukernavn
English (gb)	
	Norsk (nb)

Table E.1: Label texts

Table E.2 shows the English and Norwegian versions of service names, service texts and waiting messages. Although we only have one WSP, the table includes the name of another one as well since this text will be displayed during testing. However, as "Register of Persons" is only a dummy resource offering, it has no defined labels and values.

English	Norwegian
Register of Persons	Personregisteret
Edu. Loan Fund	Lånekassen
Application status	Status på søknad
Current debt	Samlet gjeld
Last payment	Siste innbetaling
Next instalment	Neste terminbeløp
Contacting log-in service	Kontakter innloggingstjeneste
Finding providers	Søker etter tilbydere
Retrieving services	Henter tjenester
Verifying password	Sjekker passord

Table E.2: Service texts and messages

Table E.3 shows the English and Norwegian versions of the error messages from the WSP. The first column is the error codes used in the WSP and the other two columns are the messages presented to the user to explain the error.

Code	English	Norwegian
unknownId	The username is unknown to this service.	Brukernavnet er ukjent for denne tjenesten.
requestTimedOut	Time-out before the service could create a response.	Tidsutkobling før tjenesten kunne lage et svar.
serviceUnavailable	The service is temporarily unavailable.	Tjenesten er for øyeblikket utilgjengelig.
notAuthorized	Access not authorized due to faulty info from log-in.	Tilgang ikke godkjent grunnet feil info fra innlogging.
internalError	An internal error occurred in the Educational Loan Fund system.	Det skjedde en feil internt i Lånekassens systemer.

Table E.3: Error messages from the WSP

Table E.4 shows the English and Norwegian versions of the error messages that are presented to the user in the application window when an `Exception` occurs in the application or if the IdP rejects a request.

English	Norwegian
The name of the log-in service was not found	Fant ikke navnet på innloggingstjeneste
The log-in service was not found	Fant ikke innloggingstjeneste
The log-in service could not be trusted	Kunne ikke stole på innloggingstjeneste
The connection with the log-in service was terminated	Koblingen til innloggingstjeneste ble avbrutt
The log-in service returned a fault	Innloggingstjenesten returnerte en feil
Wrong time settings on mobile or log-in service	Feil klokkeinnstilling på mobil eller innloggingstjeneste
The connection with the provider directory was terminated	Koblingen til tilbyderlisten ble avbrutt
No providers found	Fant ingen tilbydere
Wrong time settings on mobile or provider directory	Feil klokkeinnstilling på mobil eller tilbyderlisten
The connection with the service provider was terminated	Koblingen til tjenestetilbyderen ble avbrutt
No services found	Fant ingen tjenester
Wrong time settings on mobile or service provider	Feil klokkeinnstilling på mobil eller tjenestetilbyder
Message replay: Invalid message received	Ugyldig felt i mottatt melding
Received message with invalid format	Mottok melding med ugyldig format
The server rejected the log-in	Tjeneren avviste innloggingen

Table E.4: Error messages from the application

Table E.5 contains the English and Norwegian translations of the text in the help screen. In this text, the log-in window is explained to remind the user to check the name of the login-in service. This is placed first to increase the probability that new users will read it. Under the heading "Application" we try to give a brief summary of how the program should be used.

English	Norwegian
<p>Login: The name of the log-in service is stated on the login screen and you should ensure that this is your correct provider before sending your username and password.</p> <p>Language: The language can be changed by choosing Settings in the main menu.</p> <p>Application: After log-in the application will try to find the providers offering services for your user. After selecting a provider, a list of services is found and you can select one of these at a time to look at.</p>	<p>Innlogging: Navnet på innloggingstjenesten står på innloggingsskjermen og du bør sjekke at dette er din riktige tjeneste før du sender inn brukernavn og passord.</p> <p>Språk: Språket kan endres ved å velge Innstillinger i hovedmenyen.</p> <p>Applikasjon: Etter innlogging vil applikasjonen prøve å finne tilbydere som har tjenester for din bruker. Etter å ha valgt en tilbyder hentes det en liste med tjenester og du kan velge en om gangen av disse for å se på.</p>

Table E.5: Help screen text

F Test and requirement connection

Table F.1 shows which requirements are covered by which tests. This helps to check that all these requirements are covered by the system test.

	Requirement title	ST1	ST2	ST3	ST4	ST5	ST6	ST7
F-4	UN/PWD authentication	X	X	X	X	X		X
F-5	Token issuance	X			X			
F-6	IdP using HTTPS	X	X	X	X	X		X
F-8	WSP accept assertions	X			X			
F-9	WSP using HTTPS	X			X			X
F-10	WSP conformant to WSDL	X			X			X
F-11	WSP structuring results	X			X			
F-12	WSP structure error msg							X
F-13	Request schema	X			X			X
F-14	Return all info about user	X						
F-15	Preferred language	X			X		X	
F-19	Service offering content	X			X		X	X
F-20	HTTPS to WSP and IdP	X			X		X	X
F-21	Credentials in first msg	X	X	X	X	X	X	X
F-22	Trust IdP root cert	X	X	X	X	X	X	X
F-23	Trust WSP root cert	X			X		X	X
F-24	Store username			X	X	X	X	X
F-25	Change language							X
F-26	Error based on schema						X	
F-27	Result schema	X			X			
F-28	Request schema	X			X		X	X
F-30	No message validation							X
F-31	Msg indicating fault		X				X	
F-32	Options on error screen		X		X		X	X
F-33	Access to other providers				X			
F-34	Show waiting screen	X	X	X	X	X	X	X
F-35	Check UN/PWD with format	X	X	X	X	X	X	X
NF-4	Support of NB&GB	X	X	X	X	X	X	X
NF-5	Phone lang or GB as default	X	X					

Table F.1: Test and requirement connection

G Templates

This appendix contains the templates we have created for the specification and result reporting during testing. Table G.1 shows what the check lists must contain. The specification of each test case in the system test should follow the template given in table G.2. For each of the test specifications there should be documentation of the results, which follows the template in table G.3.

Package:	[the subject of review]
Time of check:	[the date of code review]
Reviewed by:	[person]
Approved:	[Yes/No]
Comments:	[description]

ReqID	Requirement text	Result
[ReqID]	[Requirement text]	[OK/Failed]

Table G.1: Check list template

Identification TID:	[number]
Responsible:	[person]
Premises:	[test conditions]
Test operations:	[description of operations that will be performed]
Expected outcome:	[data description]
Criteria of approval:	[description]

Table G.2: Test specification template

Identification TID:	[number]
Test executed by:	[person]
Time of test:	[date]
Actual output:	[data description]
Testing comments:	[description]
Type of error:	[none, noncritical, severe]
Test approved:	[data description]

Table G.3: Test result template

H Results of first system test

Tables H.1 to H.3 gives the results from the first execution of the system test which was aborted after ST3.

Identification TID:	ST1
Test executed by:	Gunn Olaussen
Time of test:	21.05.06
Actual output:	Emulator: Test ran according to specification in 51 seconds. Mobile: Test ran according to specification in 1.41 minutes.
Testing comments:	A substantial amount of the time was spent entering the username and password.
Type of error:	None
Test approved:	Yes

Table H.1: Test result for ST1 - first attempt

Identification TID:	ST2
Test executed by:	Gunn Olaussen
Time of test:	21.05.06
Actual output:	Emulator: Test ran according to specification in 59 seconds. Mobile: Test ran according to specification in 1.53 minutes.
Testing comments:	Used password "123". Test phone did not have German as an option so we had to use Danish. A substantial amount of the time was spent entering the username and password.
Type of error:	None
Test approved:	Yes

Table H.2: Test result for ST2 - first attempt

Identification TID:	ST3
Test executed by:	Gunn Olaussen
Time of test:	21.05.06
Actual output:	Emulator: While on step 7 the provider list suddenly appeared since we had not aborted properly. The reason was that we pressed back at a time where the answer was already received, causing the mobile to continue processing the response. Mobile: Test not executed.
Testing comments:	The cause of the error is known and it has earlier been fixed between the DS and WSP, but we forgot to do it for the AS. We need to check if the abort variable has been set before continuing to process the received message. The entire system test will be executed again 22.05.06.
Type of error:	Severe
Test approved:	No

Table H.3: Test result for ST3 - first attempt

I Abbreviations

This appendix contains a list of abbreviations that have been used in this report. These are presented in tables I.1 and I.2.

ABB	COMPLETE FORM
API	Application Programming Interface
AS	Authentication Service
CA	Certificate Authority
CLDC	Connected Limited Device Configuration
CN	Common Name
CRL	Certificate Revocation List
DES	Data Encryption Standard
DS	Discovery Service
EDGE	Enhanced Data rates for GSM Evolution
FAQ	Frequently Asked Questions
G2B	Government-to-Business
G2C	Government-to-Consumer or Government-to-Citizen
G2G	Government-to-Government
GPRS	General Packet Radio Service
GUI	Graphical User Interface
HMAC	Keyed-Hash Message Authentication Code
HOTP	HMAC-based OTP
HTML	HyperText Markup Language
HTTP	HyperText Transport Protocol
HTTPS	HyperText Transport Protocol Secure
ID	Identifier
IDI	Department of Computer and Information Science
ID-FF	Identity Federation Framework
ID-SAFE	Identity Strong Authentication Framework
ID-WSF	Identity Web Services Framework
IdP	Identity Provider
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
ITU	International Telecommunications Union

Table I.1: Abbreviations A-I

J2ME	Java 2 Platform, Micro Edition
J2SE	Java 2 Platform, Standard Edition
JAR	Java Archive
JRE	Java Runtime Environment
JSP	Java Server Pages
JSR	Java Specification Request
JVM	Java Virtual Machine
KB	Kilobyte
KBPS	KiloBits Per Second
KVM	K Virtual Machine
LUAD	Liberty-enabled User Agent or Device
MAC	Message Authentication Code
MIDP	Mobile Information Device Profile
NTNU	Norwegian University of Science and Technology
OASIS	Organization for the Advancement of Structured Information Standards
OATH	Initiative for Open Authentication
OTP	One-Time Password
PIN	Personal Identification Number
PKI	Public Key Infrastructure
RFC	Request For Comments
RPC	Remote Procedure Call
RSA	Algorithm by Rivest, Shamir and Adleman
SAEG	Strong Authentication Expert Group
SAML	Security Assertion Markup Language
SATSA	Security and Trust Services APIs
SASL	Simple Authentication and Security Layer
SIM	Subscriber Identity Module
SOA	Service-Oriented Architecture
SRS	Software Requirements Specification
SSL	Secure Sockets Layer
SSO	Single Sign-On
SP	Service Provider
SRS	Software Requirements Specification
TLS	Transport Layer Security
UML	Unified Modeling Language
UMTS	Universal Mobile Telecommunications System
UN/PWD	Username and Password
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
XML	Extensible Markup Language
WS	Web Services
WSC	Web Service Client/Consumer
WSDL	Web Service Definition Language
WSP	Web Service Provider

Table I.2: Abbreviations J-W