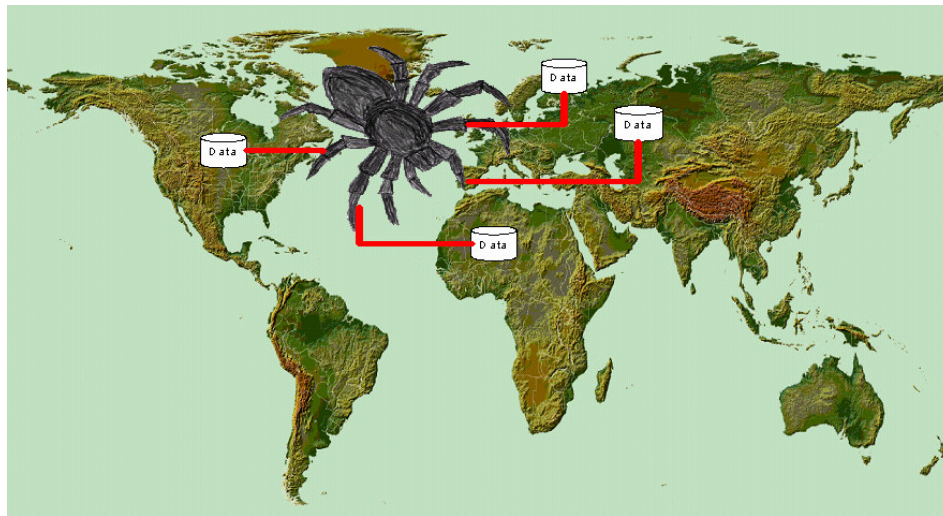


Secure XML file sharing in a JXTA P2P network for inter-organizational industrial collaboration



Muhammed Mostafa
d00muham@dtek.chalmers.se

Peter Wigren
d00peter@dtek.chalmers.se

Master's Thesis

Computer Science and Engineering Program

CHALMERS UNIVERSITY OF TECHNOLOGY

Department of Computer Engineering

Gothenburg 2004

All rights reserved. This publication is protected by law in accordance with “Lagen om Upphovsrätt, 1960:729”. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior permission of the author.

© Muhamed Mostafa and Peter Wigren, Gothenburg December 2004

Abstract

This report is a result of a project performed at the department of Industrial Environmental Informatics (IMI) at Chalmers University of Technology in order to solve the problem of securely share data in a P2P network. The main goal is to improve inter-organizational industrial collaboration for constructing cheaper and more reliable life cycle assessments by sharing data files in XML format. The solution must be sufficiently secure and reliable to let the companies feel they can fully trust it.

The solution features a P2P network in a Java environment, using the JXTA platform for a high level P2P programming and implementing security by utilizing RSA and AES. The solution known as Spider that is supposed to provide a secure, platform independent xml-file sharing environment for the companies.

Preface

This thesis describes the result of our Master's project that is the final part in our Master degree education in Computer Science and Engineering, at Chalmers University of Technology in Gothenburg, Sweden. This project has been performed at the department of industrial environmental informatics (IMI) at Chalmers University of Technology during the autumn of 2004.

We would first like to thank our examiner Tomas Olovsson, our local supervisors here at IMI Markus and Ulf for their time and invaluable support. Special thanks should also be noted to the JXTA community who answered some of our more complicated questions about the platform. For their courtesy of proving graphics and the SAXON libraries we also want to thank Max Rudberg [1] and Michael H. Kay [2]. Finally, we would like to thank all the people who have been supportive in different ways during our work.

Table of contents

1	<i>Introduction</i>	1
1.1	Background	1
1.2	Problem description	3
1.3	Requirement specification	4
2	<i>Method</i>	6
2.1	Possible solutions	6
2.1.1	The Java Applet solution	6
2.1.2	The Google solution	7
2.1.3	The P2P protocol implementation solution	7
2.2	Our solution	8
3	<i>Technology used for the implementation</i>	9
3.1	XML	9
3.2	Search engine based on XPath	10
3.3	JXTA (version 2.3.1)	11
3.3.1	JXTA concepts	11
3.3.2	JXTA configuration panel	17
3.4	Security	19
3.4.1	Public key encryption (RSA)	19
3.4.2	Symmetric key encryption (AES)	20
3.4.3	Digital signatures	22
3.4.4	Digital Certificates	23
3.5	Graphics	24
3.6	Logging method	25
4	<i>Implementation of the Spider application</i>	25
4.1	Solution overview	25
4.2	Code structure	32
4.3	Key management and certificates	35
4.4	Problems encountered during the work and their impact	37
4.4.1	Poorly documented platform	37
4.4.2	Continuously developing platform	37
4.4.3	JXTA configuration panel	38
4.4.4	Poor possibilities for large scale testing	38
4.5	Presentation of the application	38
5	<i>Analysis of the program</i>	44
5.1	Security assessment	44
5.1.1	Certificate attacks	44
5.1.2	Sending trash	45
5.1.3	System attacks	45
5.2	Network impact	46
5.3	Future improvements	47
6	<i>Conclusion and discussion</i>	48
7	<i>References</i>	49

Appendix A	I
8 Appendix B (licences)	III
8.1 The Sun Project JXTA(TM) Software License (Based on the Apache Software License Version 1.1).....	III
8.2 License bouncy castle	IV
8.3 Skin Look And Feel 1.2.10 License	IV
8.4 License of Log4j.....	V
8.5 Licence nanoXML	VI
8.6 Licence and conditions of use for SAXON package	VI
9 Appendix C.....	XIV

List of abbreviations

JXTA	Juxtapose as in side by side
IMI	Industrial environMental Informatics (Industriell Miljö Informatik)
HTTP	Hyper Text Transfer Protocol
P2P	Peer to Peer
LCA	Life Cycle Analysis
API	Application Program Interface
XML	eXtensible Markup Language
MetaL	Meta-programming Language
DTD	Document Type Definitions
XSD	XML Schema Definition
XPath	XML Path Language
PGP	Pretty Good Privacy
CA	Certificate Authority
SCer	Spider Certificate
TCP	Transmission Control Protocol
IP	Internet Protocol
NAT	Network Address Translation

1 Introduction

This report deals with the requirements, the design, and the construction of a system for secure P2P file-sharing of industrial environmental data, by using the JXTA [10] platform.

Due to increasing population and technological inventions, the environment today is far more exposed to complex threats than it was a few hundred years ago. This is mainly due to the complexity and the massive material and energy use of industrial production, of the waste production and water and air emissions resulting from inefficient production management, but also due to how produced products or services effect the environment during use. Since the concern of the environment often results in some kind of action from those that are affected, there is a need for a tool that can be used to deal with these complex systems [3]. To prioritize among the possible actions to improve the environmental performance of industry, there is a need for methods and tools that facilitate the overview of large and complex systems. Lifecycle assessment (LCA) [20] is one such tool, which is both accepted and used in industry and business. LCA provides ways to show the entire life span of a product and thus provide a way that takes into consideration the previously mentioned complexity.

However, data acquisition of data for LCA still holds a high cost, which holds back the companies from performing LCAs. Much has been done to improve this, such as development of common data formats [21, 22, 5, 23, 24], establishing agreements on common nomenclatures [25, 26, 27], build-up of common databases [26, 28, 29] etc. Many of these works are currently ongoing at both national and international scales, but there are also reasons to introduce and test new technologies and techniques to solve the problems. One such introduction and test, is to apply P2P technology for file sharing, to increase the availability of environmental data for LCA studies. This is what this Master thesis deals with.

By providing a platform for P2P sharing of data in a specified (ISO/TS 14048) XML format it may be possible to make data sharing easier and cheaper, hence making it easier and cheaper for companies to produce LCA studies.

1.1 Background

In order to put our work into a context we intend to provide some background information and explain several concepts that are the core of the solution. To do this we will first introduce the department where we have performed our work with their own words and then we intend move on to these concepts.

IMI (Industrial Environmental Informatics)

“Industrial Environmental Informatics works on different aspects of the information needed for environmental management in the industrial society.

We work with development of information models for different environmental databases. We specialize on methodologies for acquiring, documenting and quality assuring data. We work with developing environmental information management routines, methods, tools and systems, and we follow them into implementation in industrial environments. We maintain databases for different user groups, such as the Swedish competence center CPM. Our vision and our aim is that it should be affordable for any decision-maker anywhere in the industrial society to acquire good environmental information for their environmental decisions.” [4].

LCA

LCA is an abbreviation for *life cycle assessment* and is a vital tool for showing the environmental impact for a product or service over its entire lifecycle. The idea is to follow the item from its raw material extraction, over the construction, through its usage until it is finally scrapped. During this period the item will interact with the environment in one or several ways, direct or implicitly. The entire life cycle chain of interaction is described by the LCA methodology, which produces a summation of the environmental impact for an item during its life cycle. The partial events in the chain are referred to as processes [5], and they are the main building blocks for the LCA life cycle model. Figure 1.1 shows the outline for such a typical LCA, including also the natural environment with which a product or service interacts.

LCA: (Environmental) Life Cycle Assessment

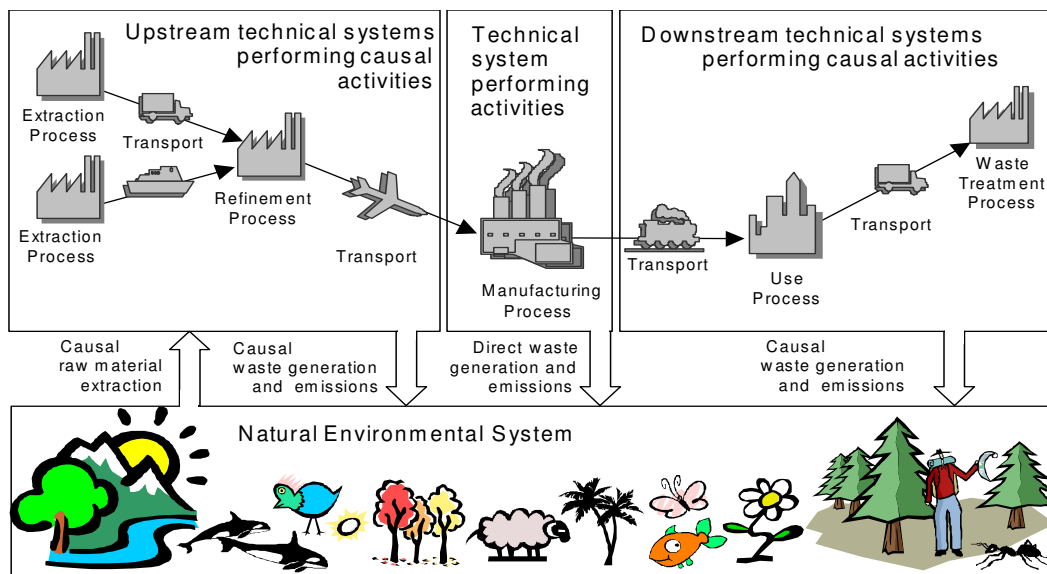


Figure 1.1¹

Process

As mentioned above, in the ISO technical specification ISO/TS 14048 [5] each module of the life cycle chain is referred to as a process, and ISO/TS 14048 describe the format for documenting and describing the environmentally relevant data about a process and its inputs (energy, resource, raw material, etc) and outputs (products, waste, emissions, etc.). ISO/TS 14048 is not yet well-established, and this fact is generating problems that will be addressed later.

A typical example of a process is the production of electricity. To produce electricity e.g. coal may be the input and electricity is acquired from its combustion. The combustion produces ashes as output, as well as e.g. CO₂ emissions. This process can then be used in the modelling of an life cycle flow chart.

Process standardization

¹ Used with permission. Copyright Raul Carlson and Ann-Christin Pålsson, CPM, Chalmers University of Technology, 1998

As mentioned above, ISO/TS 14048 is not the only way to format process data documentation, which results in a problem since the companies will have difficulties in understanding each other. For example certain data fields, concepts and key words are different in different companies. If the companies can use a common standard they will improve their interaction since it greatly enhances the interpretation of the results from any other company. This is vital since an LCA often consists of processes from several actors for a certain item or product.

The standard is referred to as ISO/TS 14048 and it defines the concept and structures needed when documenting LCA processes. The ISO/TS 14048 documentation format has been implemented as a data exchange format using XML by the department (IMI) where we have performed this master thesis. The main reason for using XML is that it provides platform independence, allows type checking and also constitutes a solid ground for data exportation to other formats among other things. It is also very easy to maintain an exact structure by using functionality discussed under the XML section.

The problems with LCAs

An LCA for a product or service requires a large amount of information concerning the natural resources are obtained, how they were transported, refined into raw material, put together into components etc. Sufficiently good data may exist, but they may not be known to the user, and hence the user may inadvertently have to reproduce costly work. The processes that describe these are often very expensive to produce and are likewise the components used for creating the item distributed around the world.

Since there are no established ways of sharing the data in any ordered form there will be a problem when the companies try to obtain the needed processes for the construction of an LCA. In the worst case they recreate a process that already exists because they weren't aware of its existence or can interpret already existing documentation. This way of working is both expensive and a waste of resources and it will discourage companies from ever start producing LCAs.

1.2 Problem description

The main problem encountered during the construction of an LCA is as mentioned above to obtain the processes it should consist of. Since they are expensive to produce and distributed around the world it is not an easy task to compose an accurate and cheap LCA. In order to establish a platform for inter-organizational industrial collaboration and encourage sharing the above standard emerged. Now companies can understand each other but lack the infrastructure to really share data. This is where our Master thesis comes in.

The core problem is to create a system that allows users to share their already created processes so that other companies can benefit from them. We also need to implement a search function to allow the users to search for a specific content of a file rather than the name. To provide a way of adapting the program to any change in the standard it is also desired that the search function is dynamic in the sense that it can be updated to new standards without requiring a reinstallation of the program.

To ensure future improvements such as payment for documents (see 5.3: Future improvements) the solution must be based on robust security. The security will also be needed to allow users to feel safe while sharing delicate information over the Internet.

1.3 Requirement specification

As a result of the discussion above an application needs to be created that can be used by different users to share documents. The solution, which we refer to as *Spider*, has to fulfil the following requirement specification which is provided by our employer (IMI). Statements that contain *shall* and *must* shall be included in the solution while *should* denotes optional implementation.

General specifications

Here we present important requirements for the development that doesn't fit under a specific header:

- The solution must be able to operate through firewalls and NAT servers (see section 3.4.1) since the users most likely will be using the program in this kind of environment.
- Access to the shared files must be controlled by the user.
- All members in a group² must be able to find each other

Technology specification

This section is about the technology we should use to implement the solution:

- The programming language for development shall be Java and it must work correctly under version 1.4.2.
- The application shall have user-friendly error handling.
- The code shall be documented according to IMI-standard.
- System documentation for how to use the application including figures and examples shall be available.

Security specification

This section is about the security aspects for the solution:

- It shall be possible for new qualified users to register and use the application without the intervention of the administrator.
- It shall be possible for user and administrator to uniquely identify another user in the network when a connection is initiated.
- All communication including file transfers shall be done in a secure way i.e. all communication between users must be encrypted.
- Different groups should be available for different purposes.
- There should be a way to control the members in a specific group.

Search specification

This section is about the search functions for the solution:

- The application shall only be used to search for files in XML-format
- A user shall be able to share different files in different groups.

² A group is a set of users in the JXTA network that have agreed on a certain set of services that enables them to communicate in a private setting. (For more information about JXTA and groups see the JXTA section under Technology used for the implementation)

- It should be possible for each user to search for files in different groups at the same time.
- It should be possible for the user to search for any kind of XML-file or only for files that support the ISO/TS 14048 XML format.
- There shall be a dynamic way of controlling how the search is performed in the XML documents that support the ISO/TS 14048 format

File transferring specification

This section is about how the file transfers should be done:

- File transfers shall be encrypted.
- The application shall place the downloaded files in a specific map that is selected by the user.
- The application shall not overwrite existing files.

Installation and running specification

At last this section is about installation and running the application:

- The application must be easy to install and run by the user.
- Documentation for the application must be available.

2 Method

The method we have used can be described as first generating several ideas that solve the problem and then weighing these against each other to come up with the best possible solution. These solutions will be presented in this section in comparison with each other and the reasons they have been rejected. After deciding on a solution, in close contact with our employers, the development started. The idea will be reviewed here but the final result will be presented in section 4.

2.1 Possible solutions

During our research we came across several ways of solving the problem. These solutions will be presented here along with the reason they have been rejected. This is done in a brief outline to emphasize on the solution rather than the technological implementation.

2.1.1 The Java Applet solution

Java applets are java code located on a webpage that is executed locally on a computer by using a browser as the execution platform. The idea is to create an applet on a web page that allows the user to share his or her files. Since the code is executed locally the user can add his or her files locally and then let the applet set up a communication link with the server. Other users that run this program can then ask the server for a certain file and the server can in its turn ask all the connected applets. If a user has a matching file he or she could send a reply to the requesting node, which contains information about the file and why it matches the query.

The first node can then decide whether or not he or she wants to download the files. The download could go over the server or directly between the nodes if IP numbers and such can be obtained from the server.

Pros

- Easy to maintain:
The applet code just needs to be replaced on the server and by doing so all new users will automatically upgrade their applications when revisiting the site.
- Easy to get started with:
A user only needs to enter the webpage in order to start the program
- Platform independent solution

Cons

- One single point of failure:
Since all the requests are sent over the server it is vital for the system that it never crashes. This is not good since the entire system will be dependent on one single computer.
- Java 1.4.2 will be required:
Since browsers only support java version 1.08 natively the browser needs to obtain a later version. The reason for a later version is to ensure secure transmissions due to the lack of support for cryptography in java 1.08. To upgrade the browser the user needs to download a plug-in that ensures later java version capability. This is not always nice since it's normally presented as a security hazard by most browsers. Java 1.4.2 was

also mentioned in the requirement specification.

- Unable to connect to other computers than the server that provided the applet:
A socket created by an applet throws exceptions when connecting to other computers than the provider. This can also be solved but we are uncertain how different browsers will handle this exception.
- Firewalls:
In order to communicate over firewalls the applet needs to implement the HTTP³ protocol which isn't trivial to accomplish.

This solution was rejected because it would take too much time to implement the HTTP protocol and because the program will require too much from the user the first time it is executed.

2.1.2 The Google solution

Since this solution is similar to the Google search mechanism we will refer to it as the Google solution. Here we want to keep a central database where a user can add his or her files and search for others. The interface could be a secure web page that allows logged in users to search for files.

Pros

- Easy to use and maintain
- It is possible to traverse NAT and firewalls by using existing browser technology.

Cons

- All searches will be performed by one server⁴. It has to search all the files that are shared for a single match and that might take a while with limited server capacity.
- No users want to give away their files and loose control of them to an external server. (See requirement specification: General specification)

This solution was rejected mainly because our employer didn't approve due to the reason here presented as the second con.

2.1.3 The P2P protocol implementation solution

There are currently several programs available that feature P2P technology. Among the most known are perhaps Direct Connect and KaZaA. The idea is to create a solution that implements one of these protocols and allows the users to share files directly by using P2P connections.

Pros

- The search can be done locally upon receiving a search string
- The network will be robust in the meaning that if one node dies the others won't be affected as much as if the server in one of the above solutions would die.

³ The reason the application needs to implement HTTP is that firewall usually keeps port 80 open to allow access to the Internet. To use port 80 we will need to implement the HTTP protocol. (For more information about firewalls, see the JXTA section under Technology used for the implementation.)

⁴ The department where we work will only have one server available.

- The users can have direct control of the files they are sharing
- The network will need a server only for look-up functionality and does not have to depend on it.

Cons

- The HTTP protocol needs to be implemented in order to penetrate firewalls.
- It will take too long time to implement a protocol from scratch, even if we have access to the exact specification.

This idea was rejected due to the reason that all the above pros can be obtained by using a platform that is known as JXTA. Not only does it provide these functionalities, but it also solves the firewall and NAT problem and provides a high level programming interface for program construction.

2.2 Our solution

This solution is referred to as the JXTA solution and features a program that uses the JXTA platform for P2P communication. The idea is to create JXTA peers that are connected in a private virtual network. One server/super-node will be needed as a central point of entry and then client/edge peers can connect to it creating what could be referred to as a community. In the network, search and file transfers can then be performed. These communities can operate independently of each other but to create a larger network these can be merged by connecting their super nodes. This is a scalable solution for the network growth and figure 2.1 shows how this expansion is possible.

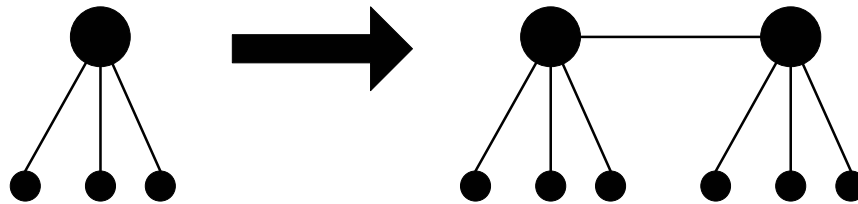


Figure 2.1

Pros:

- Allows high level P2P programming which will improve the development speed of the solution.
- The platform is independent of operating system since it is developed in java
- Traverse firewalls (the platform implements HTTP)
- The solution will be scalable

Cons:

- The platform is unknown to us and it will be impossible to construct an exact code structure for the program without testing

3 Technology used for the implementation

Since we want to focus on the main functionality for the solution (security for example) we have used as much ready technology as possible in accordance with our budget. This section is intended to present the used technology with a detailed description of how it works.

3.1 XML

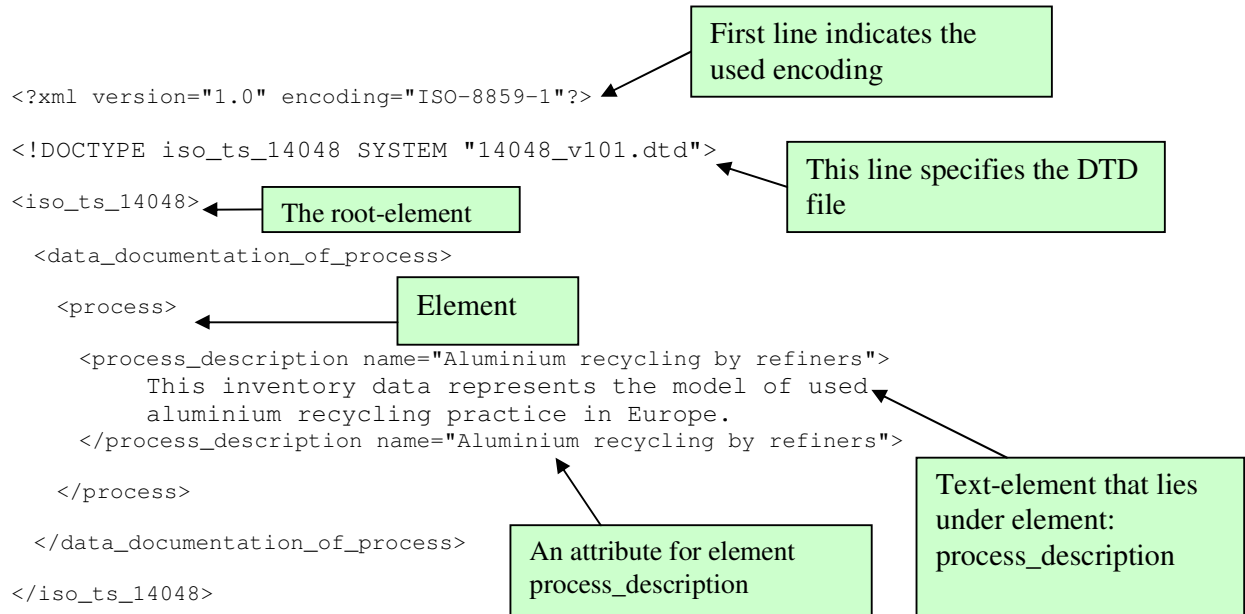
Definition

XML stands for *eXtensible Markup Language* and it is a *Meta-programming Language* (MetaL), which is used to record and handle information in a very simple and flexible text format. XML is like HTML in structure, however the main difference is that XML is extensible and it is possible to define own customized markup languages for limitless different types of documents. It differs from HTML, which works with a fixed format, and can be viewed as a part of XML. In XML the representation of the data is system independent and the user can define own sets of rules for the representation. More information about XML can be found at the official W3C home page [6].

XML-structure

The information in an xml-file is sorted in a hierarchical tree structure where the nodes are referred to as elements. The main element, which is called root-element, contains all other elements and the entire content of the document. Additionally any xml-document can point to a structure specification file, a-so-called DTD file which stands for *Document Type Definitions*. This defines the rules of how the content and the name of the elements in the xml-file should be interpreted. It also specifies which elements that are allowed and how they should appear in the xml-document whether they might be in other elements or directly under the root. Figure 3.1 shows what an xml-document might look like.

To improve an XML document to be even more precise a more advanced control can be obtained by using what is known as XSD files. This stands for *XML Schema Definition* and can be seen as an extension of DTD. It contains the functionality mentioned above but also includes a way of defining the types of the elements, whether they are float or integer for example. Moreover it also includes a way of specifying an exact structure in the sense that it can specify the number of occurrences of a certain element as well. The use of XSD files provides the possibility to control the exact structure of an XML file when it is created and thus the possibility of creating powerful search functions.

**Figure 3.1**

The benefits of XML

Here are some of the benefits that XML offers:

- XML allows the use of the ISO-standard ISO10646 (Unicode), which can represent an unlimited number of characters (i.e. Japanese) [7].
- Simplicity: It is easy to use by both programmers and document authors.
- Extensibility: It allows the creation of an own set of rules to represent the structure of the documents.
- Interoperability: It is platform independent.

3.2 Search engine based on XPath

The simplest way of finding a word in a text-document is to search through all the words in the text and compare them with the reference word. The case is different when we want to search in an xml-document since the document contains more information than the text and is ordered in the previously mentioned tree structure. Due to this reason our application's search operation must include necessary functions to go through all the text-elements in the document. It needs to have a powerful search engine to extract documents that satisfy specific criteria or contain data that a user might be looking for.

XPath

XPath, which stands for *XML Path Language*, is a language⁵ that is used for addressing parts of an XML document i.e. elements. By implementing XPath it is possible to search under one element rather than the brute force method of iterating over all in order to find a certain word or phrase. This way of addressing XML documents with a known structure can greatly increase the performance of the search engine.

⁵A typical XPath syntax which addresses the process_description element in figure 3.1 is shown below:
 iso_ts_14048/data_documentation_of_process/process/process_description

As mentioned in the XML section it is possible to build an XML document in a very structured way according to a standard and this can be used to enhance the search functionality of the program. The ISO/TS 14048 format provides a perfect way of utilizing the fast search capabilities provided by XPath since the structure is known to the application. For more information about XPath look at W3C home page [8]

Unfortunately java doesn't implement XPath in versions older than jdk1.5.0. Our program is however built on the java jdk1.4.2 platform and because of this we'll have to import the XPath implementation. To do so we have imported the SAXON package version 7.9.1, which is a collection of tools for processing XML documents. The package includes XPath functionality and thus we don't need to use a later Java version. SAXON is an open source product which is free to use [2].

3.3 JXTA (version 2.3.1)

"JXTA is a set of open, generalized peer-to-peer (P2P) protocols that allow any connected device on the network — from cell phone to PDA, from PC to server — to communicate and collaborate as peers. The JXTA protocols are independent of any programming language, and multiple implementations (called bindings in Project JXTA) exist" [9].

The JXTA protocols have been implemented in a Java environment by the people at project JXTA [10] and the platform is provided as a jar⁶ file java package. This package provides an infrastructure that allows the developers to create a network without having to think about how the information physically is being transported or how the peers are connected. When we refer to JXTA we refer to this java platform that has implemented all the JXTA protocols. Figure 3.2 shows how a virtual JXTA network is created from its physical counterpart.

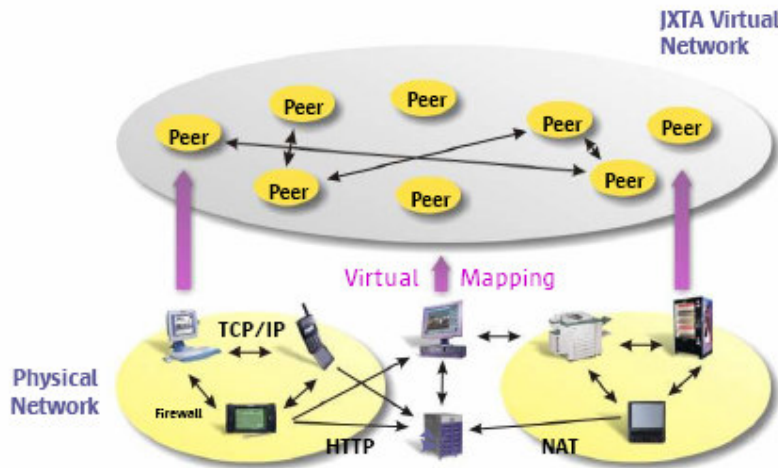


Figure 3.2⁷

3.3.1 JXTA concepts

In order to understand how the application works we need to describe some of the concepts that are used from the JXTA platform. Since understanding only can be achieved by a detailed walkthrough we intend to do so in this section.

⁶ A jar file is a package of compiled java classes

⁷ Used with the permission from Juan Carlos author of: *Project JXTA: An Open P2P Applications Platform*

Peer

A peer in the JXTA network is a networked device that implements one or more of the JXTA protocols. Since JXTA is thought to be platform independent this statement implies that a peer can be everything from a cell phone to a supercomputer. (This assumes that it fulfils the JXTA protocols.) In order to make the network robust and fault tolerant, peers operate independently and asynchronously from each other. As a benefit from this, the network won't suffer from failing peers. To make every peer unique in a sense that it can be identified in the network, it is given an ID number on the following format:

```
urn:jxta:uuid-
59616261646162614A787461503250338F400A9004F64173B23EE07CF55892E803
```

Furthermore a peer can have a certain purpose. A peer that is *rendezvous* will act as a super peer which means that it keeps a table of all other peers that are using this *rendezvous peer*. If a normal peer connects to a *rendezvous peer* it can obtain connections to other peers known by the *rendezvous peer*. Moreover a peer can also be *relay* which means it will operate as a router in the sense that it keeps information about the routes to peers. To send a message to a peer that is separated from the network by for example a firewall, *relay peers* can be used to forward this message. *Relay peers* will be further explained in the following section.

Firewalls and NAT

This section briefly describes what firewalls and NAT (Network Addressing Translation) servers are and how they are handled by the JXTA platform.

Firewall

The most basic explanation of a firewall is to say that it is a restriction in the communication with the world that lies inside the local network and the outside world. This is accomplished by closing down selected ports in order to prevent any traffic from using these to enter or leave the protected network. This becomes a problem when a contact is initiated from the outside since the sent package will be dropped by the firewall if it is using a closed port.

NAT

A problem with the current version of IP v. 4 (which is used today) is the rapidly decreasing amount of valid numbers. Several solutions exist to decrease this rate and one of them are to use local IP numbers in private networks. This allows the computers in this network to communicate without having to use IP addresses that are valid on the Internet. However, this creates a problem when the computers tries to communicate with a host on the Internet since the IP addresses used in the private network are invalid there. Routers simply will drop packages that designate from invalid IP addresses.

To solve this problem, NAT servers were invented. They operate according to the following scheme:

- A computer in the local network sends a package to a host over the Internet
- This package is intercepted by the NAT server which is connected between the internal network and the Internet.
- The local network source IP address is replaced with one that is valid on the Internet e.g. the servers IP address.
- The server records the address mapping in a table which will be used later to identify the local computer.
- When a package is sent back from the remote host, the server will use the table to replace its IP address in the packet and forward it to the correct computer.

A NAT will have the same effect as a firewall in the sense that an outside host cannot connect to a local host in the network due to the invalid IP address. The only connections that are allowed are those that are initiated by the computer in the local network since the NAT server won't be able to send the packages to the right computer if the connection isn't recorded in the table.

The JXTA solution

Since the use of firewalls and NAT servers are rather common these days, a P2P platform needs to solve the problems where certain nodes are unreachable. In JXTA this problem is conquered by the use of the relay peers. These peers maintain information about the routes to other peers and routes messages to and from peers behind these obstacles. By allowing this service a relay node can be used as an intermediate medium of transportation between two nodes that are separated by NAT. However this alone doesn't solve the firewall problem.

To communicate through firewalls this idea must be extended to include a way to send information to the relay node. The idea that is used here is to send information through the one port that must be open to allow access to the Internet: port 80. This port is often open to allow access to the Internet. However, many firewalls do not allow all kinds of traffic through this port but only for applications that implement the HTTP protocol. By using this protocol the JXTA platform can utilize the same idea mentioned above with the add-on functionality of traversing firewalls. Relay peers can be viewed as a post office where nodes can check if they have received any mail and post messages to other nodes. They simply connect to this relay peer and check their mailbox to see if they have received a message. If they have, they will bring it back home and if they have anything to deliver, they will post it there.

Figure 3.3 illustrates this concept.

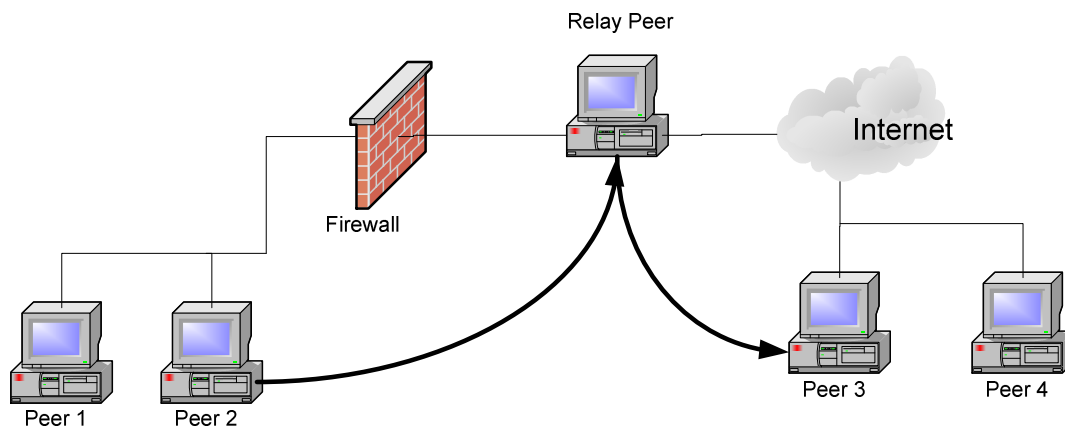


Figure 3.3

Groups

Groups or peer groups are a collection of peers that have agreed upon a common set of services and in this way created an isolated environment. A group is uniquely identified in the JXTA network through an ID number and is created from an advertisement (see advertisements) that exactly specifies it in the network. The peers themselves choose when to become a member of a particular group and are not in any way limited to be a member of only one. The groups are organized hierarchically which means that peers need to be members of the parent group in order to join a subgroup (see figure 3.4).

Since the JXTA network is organized in this way all nodes need to be a member of a super group in order to create and join other groups. This super group is known as the Net Peer Group in the JXTA network and a peer will automatically become a member of it when the JXTA platform is launched. This is true regardless of whether the peer is a member of a larger network with other JXTA nodes or isolated on the moon. The meaning of this is that peers can be members of the same group even if they are located in two separate networks without any connection and this applies to all groups.

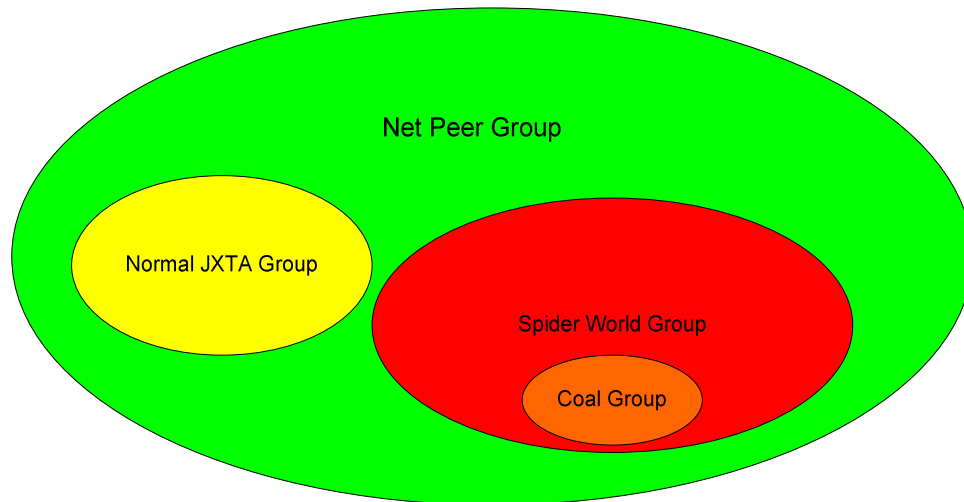


Figure 3.4

In order to restrict the membership in groups the creator can decide whether or not he or she wants to implement security. If a peer wants to become a member of a secure group it needs to provide a login and a password to be able to access the services that define the group. Since the resources are restricted in a group this implies that a private environment can be created for peers which a particular interest.

Advertisements

An advertisement in the JXTA network is a piece of information that is presented in a language neutral metadata structure as XML documents. They are used for representing JXTA network resources in a unique way to be advertised or discovered by other peers in the network.

There are several types of advertisements in JXTA and since the platform implements them as classes there may exist an almost infinite number of types due to the subclass functionality in Java. This makes it hard to categorize them all, however in the *JXTA programmer's guide* [9]; advertisements are known to be in one of the following subgroups.

- **Peer Advertisements**
It describes the peer resource i.e. a peer in the JXTA network. The information that can be found in the advertisement is the peer name, ID and available endpoints.
- **Peer Group Advertisements**
To create a separate environment, groups can be created from advertisements that contain group ID, name, description and (if the group is meant to be secure) login and

password. In order to prevent the login and password to be read in clear-text they can be encrypted.

- **Pipe Advertisements**
Pipes are created from pipe advertisements. To connect pipes they need to be created from the same advertisement which makes the platform aware that two peers have a connection and can send messages to each other using this abstraction. A pipe advertisement contains ID, type and a description.
- **Module Class Advertisements**
describes a module class. Its primary purpose is to formally document the existence of a module class. It includes a name, description, and a unique ID.
- **Module Spec Advertisements**
defines a module specification. Its main purpose is to provide references to the documentation needed in order to create conforming implementations of that specification. A secondary use is, optionally, to make running instances usable remotely, by publishing information such as a pipe advertisement. It includes name, description, unique ID (ModuleSpecID), pipe advertisement, and parameter field containing arbitrary parameters to be interpreted by each implementation.
- **Module Impl Advertisements**
defines an implementation of a given module specification. It includes name, associated ModuleSpecID, as well as code, package, and parameter fields which enable a peer to retrieve data necessary to execute the implementation.
- **Rendezvous Advertisements**
describes a peer that acts as a rendezvous peer for a given peer group.
- **Peer Info Advertisements**
describes the peer info resource. The primary use of this advertisement is to hold specific information about the current state of a peer, such as uptime, inbound and outbound message count, time last message received, and time last message sent.

Discovery service

In order to share information in the network a discovery service is needed. It allows a peer to publish advertisements for different resources that can be obtained by other peers. This can be done in two ways: locally or remote. When a local publish is performed, the advertisement for the resource is placed in the local cache where other peers can find it and when a remote publish is executed, the advertisement is broadcasted across the network. A peer that receives a remote publish e.g. an advertisement, it will store it locally in its cache.

To make this way of publishing information over the network scalable the travel of the advertisement is restrained by a hop count. The difference between this hop count and the one encountered in IP is that the advertisement isn't discarded in JXTA since it will be stored in a cache and if the procedure is performed many times by all the peers that receive this advertisement there is no actual limit for how far it can travel. This is referred to as crawling discovery because the information is travelling small steps and is virtually crawling over the network. The reason for this is to restrain the communication in the virtual network and avoid flooding it.

Finding advertisements is almost the same as publishing them in the sense that there are two ways of locating one. The first way is to scan the local cache to see if anything has been stored there and the other way is to send a request to the other peers in the network to search their caches and send whatever they find that matches the included search string. When a match is found it is simply sent to the requesting node but in order to receive the reply the peer must connect a listener to the request.

Listeners

Due to delays in the network the answers will need to be dealt with in an asynchronous manner (if one not has the luxury of waiting indefinitely for answers). Java has a nice way of solving this problem by providing what is known as call-back and it is implemented in JXTA as listeners.

In JXTA, a listener is an interface that can be implemented in a class which is called when something occurs in the network. To connect a listener to a certain event that needs monitoring, JXTA provides the possibility to tell the platform which class supposed to be called when a something has occurred. JXTA also provides a possibility to add a listener for reply handling when a certain method is called. Since every class that implements the listener interface can be registered as listener classes in JXTA private classes can also be used for achieving this functionality which is quite useful. When using the discovery service method – `getRemoteAdvertisements` for instance it is possible to give a listener as argument in order to let JXTA know how to deal with the responses.

Messages

A message is a document in XML format that is can be sent between peers in a JXTA network. The inherited properties of XML allow the message to contain elements that in their turn can contain data. When a message is created it only contains the structure that specifies it as a message but to store information, elements needs to be attached. These elements can be of three different types:

- `ByteArrayMessageElement`
The element consists of a byte array
- `TextMessageElement`
The element contains a string
- `InputStreamMessageElement`
Java `InputStreams` can be added directly into the element which makes it ideal for file transfers.

When storing these elements they are given a name in order to let the user clarify their contents and on the receiving end these tags can be used for obtaining a specific element. If the name of the element is unknown all elements can be obtained by calling an appropriate method in the `Message` class.

Pipes

In JXTA, pipes are known as a unidirectional message transporting mechanism. The endpoints of a pipe can be either input our output which also is the programming abstraction for a pipe. The input pipe is used for receiving information and the output pipe is used for

sending information and together they form a pipe. There are three different categories for pipes that can be either input or output with the following names and characteristics:

- **Point to point**
The pipes constructed from a pipe advertisement where the type is defined as point to point will only be able to communicate with one other node. There can only be one input pipe for every output pipe and this is described in figure 3.5



Figure 3.5

- **Propagate**
Pipes constructed from a pipe advertisement where the type is set to propagate allows many pipes of one sort (input or output) to connect to one pipe of the other sort. By doing so the JXTA network will handle the traffic between them without having to concern the developer or user. The concept is described in figure 3.6

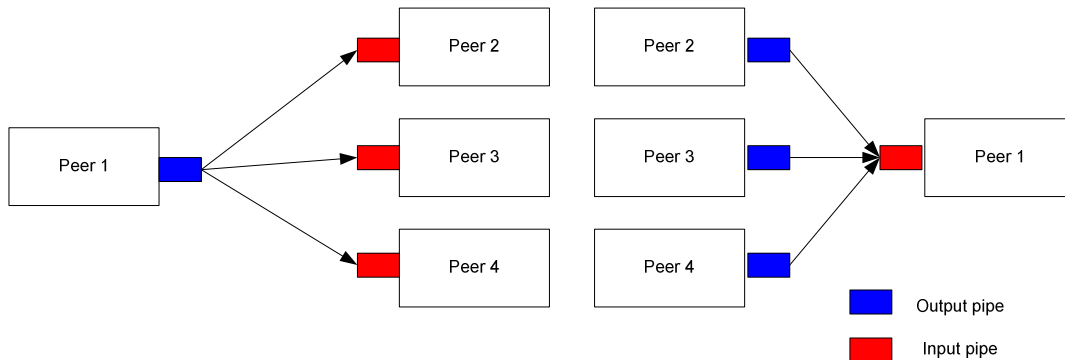


Figure 3.6

- **Secure**
Secure pipes are point to point pipes with one important difference: the messages will be encrypted using JXTA's default encryption: RC4.

3.3.2 JXTA configuration panel

When a JXTA application is run for the first time a configuration panel is displayed to configure necessary functions for the network environment. This panel is used to create username and password, to enter peer name, to select a rendezvous and relay peers and to configure TCP/IP and HTTP transports.

The following is a brief description of each sub-panel:

- **Basic:** This sub-panel is used to specify a name for the peer.
- **Advanced:** This sub-panel is used to set all necessary functions for TCP/IP and HTTP transports.
- **Rendezvous/Relay:** This sub-panel is used to setup rendezvous and relay if the peer is behind a firewall or NAT. It can be also used to make the peer acts as relay or rendezvous.

- Security: This sub-panel is used to enter a username and password.

All configuration information that has been chosen by the user will be stored in a file called *PlatformConfig* and it will be stored under JXTA home folder. Security information like username and password is stored under subdirectory called *pse* to JXTA home folder. There is another subdirectory under JXTA home folder called *cm* which is used to store advertisements that are used by this peer.

This information in JXTA home folder is used again to setup JXTA platform when the application is run next time. The only thing that the user has to do this time is to enter a correct username and password.

Unfortunately the only documentation that was available about this configuration is *Project JXTA v2.0: Java Programmer's Guide* [9], which only applies for JXTA version 2.0 not for JXTA 2.3.1 the version that we use. This lack of documentation led to misunderstandings regarding the Advanced- and Rendezvous/Relay-panels and how/what to select or remove in these panels. Look at 4.5.3 JXTA configuration panel to see the problems we encountered during this configuration. We overcame these problems by hard-coding all configurations and hid this configuration-panel from the user.

The left hand side in figure 3.7 shows the Advanced-panel with all possible options and the right hand side in the same figure show the Rendezvous/Relay-panel with all possible options.

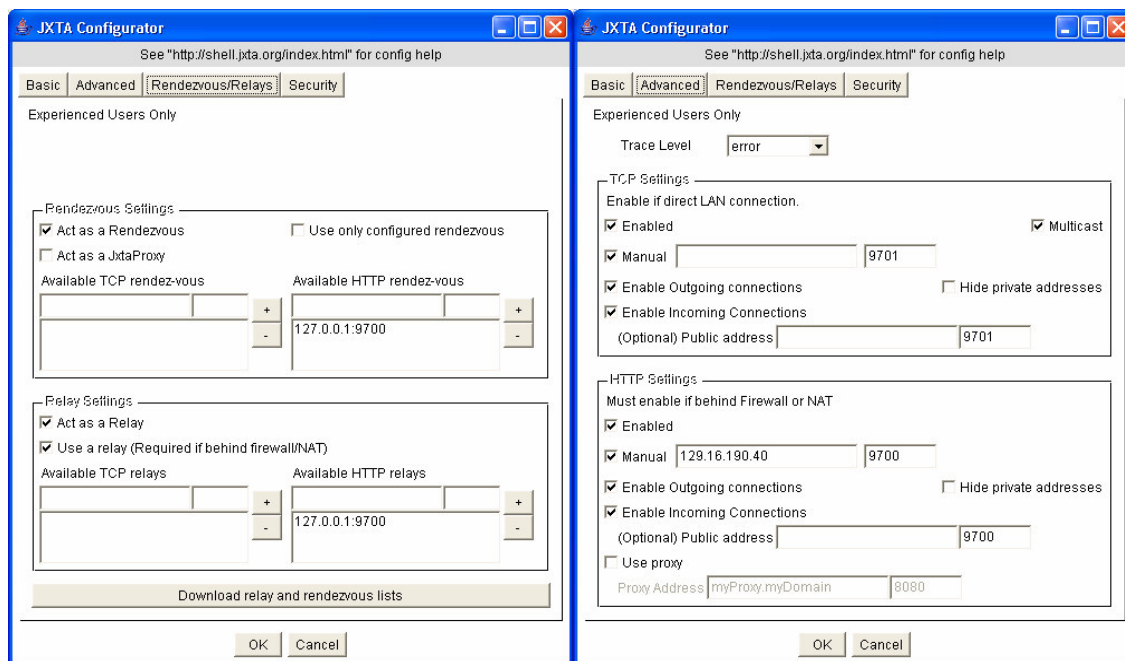


Figure 3.7

3.4 Security

3.4.1 Public key encryption (RSA)

RSA is the most widely used asymmetric crypto system and the letters stand for the names of the inventors: Rivest, Shamir and Adleman.

Short introduction

Public key encryption is based on the idea of two mathematically connected keys. One key is private while the other one is public. The public key is available to everyone while the private key is kept secret only to be known by the owner. The public key can be used to encrypt a message to the owner but only he can decrypt it since access to the private key is necessary. Figure 3.8 illustrates this concept.

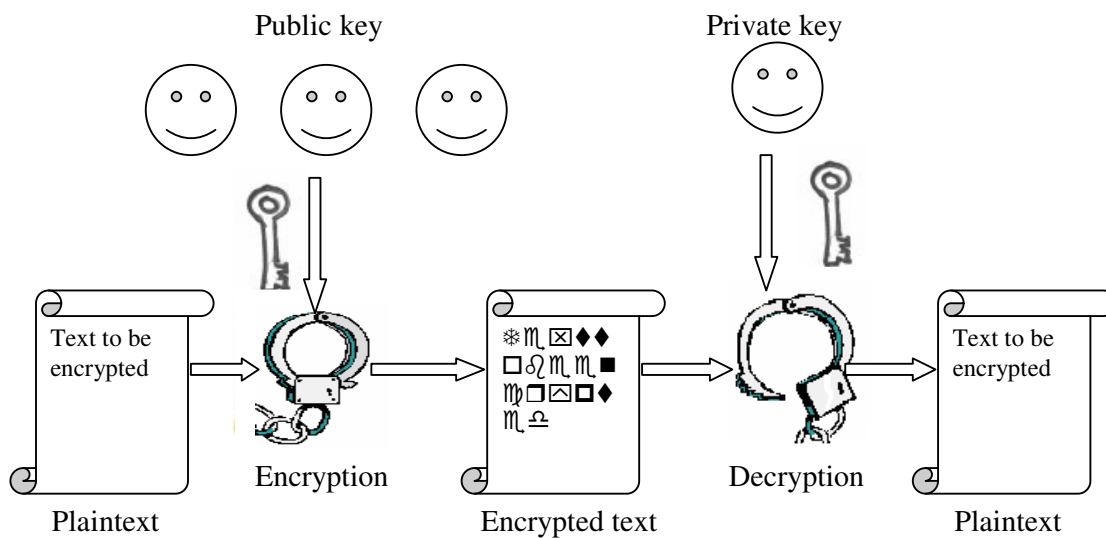


Figure 3.8

This way of encrypting messages is referred to as asymmetric encryption since the encryption and decryption is performed using two different keys. To see why RSA is safe please refer to appendix A.

Since RSA isn't included in Java we had to import a provider that provides this functionality and Bouncy Castle JCE version jdk14-124 [11] is the name of a freeware package we have used for this.

3.4.2 Symmetric key encryption (AES)

RSA is too slow however to be used in a normal application for encrypting and decrypting text messages. To increase the performance we also use a fast symmetric-key encryption system known as AES. Since symmetric-key encryption uses the same key for encryption and decryption, the key needs to be transported to the recipient in order to make it possible for him/her to decrypt the message. This can be achieved with the combination of RSA and will be explained later.

A simple way of explaining how the AES encryption algorithm works is to say that it performs substitution, transposition and modulo operations repeatedly according to a key. This will make the plaintext unintelligible and unreadable for those that don't have the key to decrypt the gibberish. For more information please refer to this Internet site [12].

Why AES is secure?

In the world of cryptography a cipher that has been on the market and hasn't revealed any flaws is considered to be safe. To backup this statement we intend to present a report from the CNSS [13] which quotes the NSA (National Security Agency) on the use of AES. According to the 6th paragraph AES should be used accordingly:

- AES with 128, 192 and 256 bits encryption is sufficient for protecting classified information up to the SECRET level
- AES with 192 and 256 bits encryption is sufficient for protecting TOP SECRET classified information

If the NSA considers AES to be this secure we believe that it is sufficient for our application as well.

Creating a fast crypto system with RSA and AES

As mentioned before AES can be used together with RSA to create a fast cryptosystem. First the information to be sent is encrypted by using a completely random AES key. This random key is referred to as the session key since it will only last one session. To allow the receiver to decrypt the message the session key is encrypted using the receiver's public key and bundled into a package that is sent over the network. The receiver can then extract the session key by using his private key to decrypt it and then decrypt the rest using the decrypted session key. Figures 3.9 and 3.10 describe the chain of events.

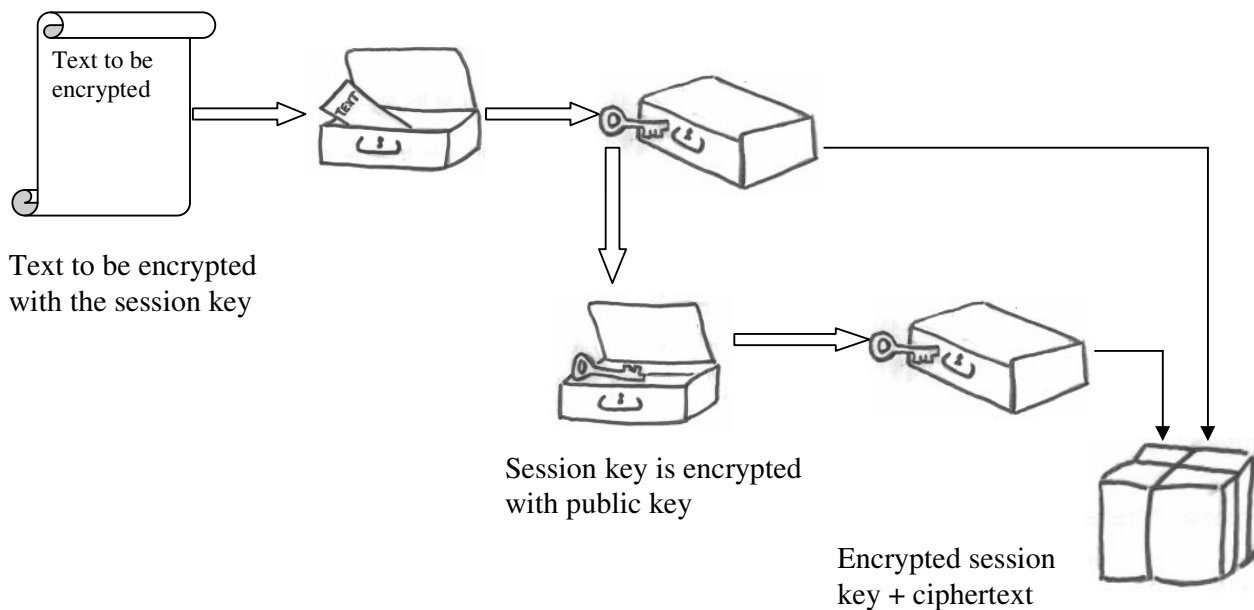


Figure 3.9

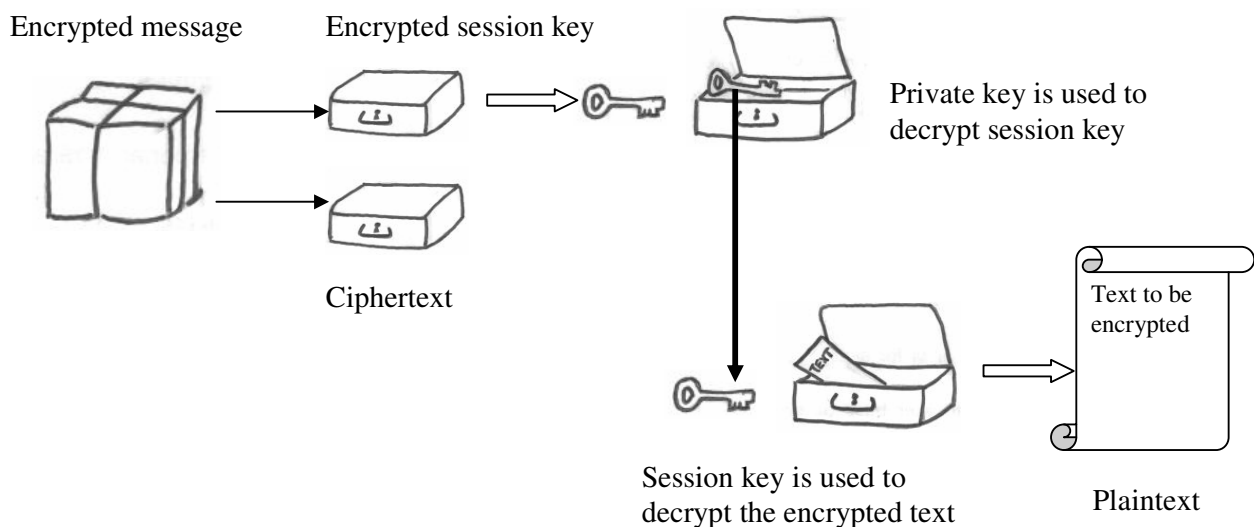


Figure 3.10

3.4.3 Digital signatures

A very important function that can be obtained from public key cryptography is to sign digital texts, which is referred to as digital signatures. It has the same purpose as a handwritten signature which is to verify the content of a text (to guarantee the content to be accurate). Digital signatures are used to preserve the information integrity of a document and provide a means of discovering if it has been tampered with (someone has changed the original text).

The way to sign texts is similar to common asymmetric encryption but instead of encrypting the text with someone else's public key we use our private key. To verify signatures the recipient can use our public key and check if the signature matches the supposedly signed text by decrypting the signature. If the decrypted signature matches the text, it proves it hasn't been tampered with. Furthermore this also proves that the person who signed the text is the owner of the private key connected to the public. Figure 3.11 illustrates how digital signatures are used.

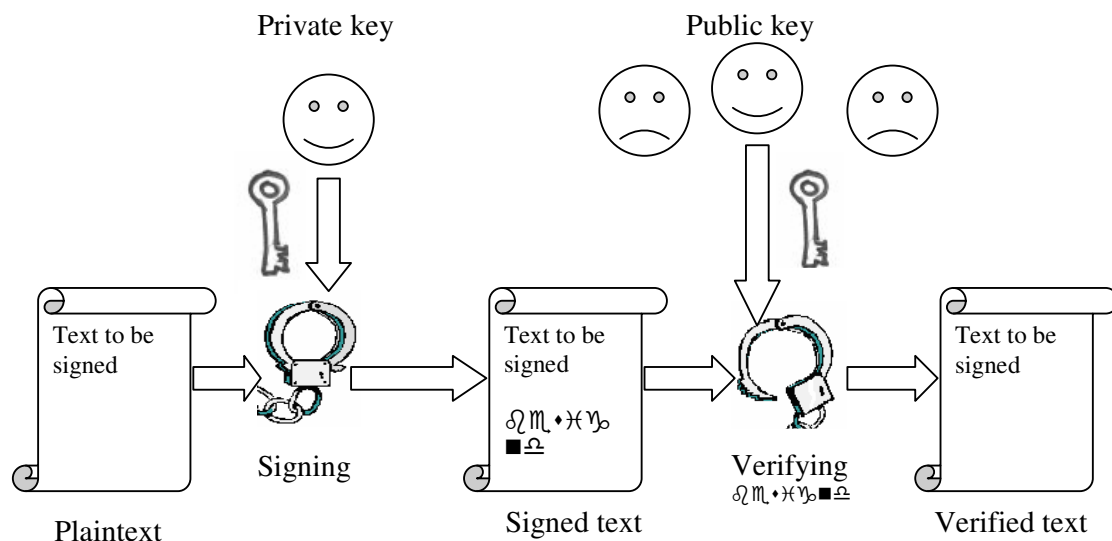


Figure 3.11

Hash functions

The problem with digital signatures is to sign large pieces of data since this will slow the signing and verification operations. A technique called one-way hash function can be added to improve the performance and solve this problem.

Hash functions can take any length of data and generate a fixed-length of output which is referred to as a message digest. If the data is changed even by one bit then the hash function will generate a completely different output.

The message digest can be signed by the private key to generate a signature and this signature can then be added to the message. These two can then be sent together to a recipient to make certain that no one changes anything in the message. Again the recipient can now verify the signature by taking a hash value of the received text and verify it against the received signature by using sender's public key. Figure 3.12 illustrates how hash function works

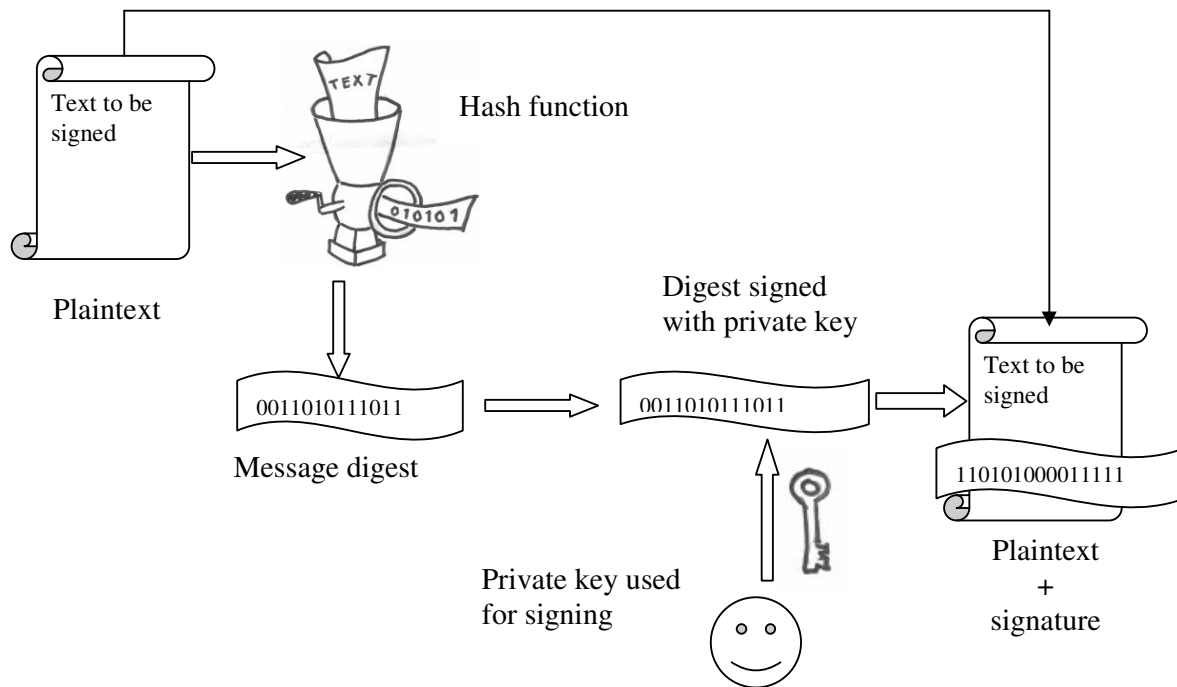


Figure 3.12

3.4.4 Digital Certificates

One problem with public key cryptography is to distribute the public keys in a trustworthy manner. There is no problem if the key is handed physically, but when the users can't meet and exchange keys this becomes an issue. Since we use public key cryptography in our application we solve this problem by using certificates and the rest of this section outlines the basic concepts of general certificates and their appliance.

Since anyone can send a public key to someone and pretend that he or she is the right person to exchange data with there must be a way to prove the senders identity. An attacker can for instance place himself between sender and receiver and by switching the public keys that are sent intercept the traffic. This is a kind of attack is known as the man-in-the-middle attack. It is therefore very important to know that the public key we just received belongs to the right owner and not to someone else.

Digital certificates are used to ensure that a public key belongs to the right person we want to exchange data with. A digital certificate contains a public key along with enough information about the identity of the owner to uniquely identify him or her. Since this information can be tampered with, a third party, whom users trust needs to guarantee the authenticity of the information. To do this all the information in the certificate along with the public key is hashed and signed. By adding this signature to the certificate the third party, the certificate authority (CA), can guarantee the correctness of it and prevent anyone from modifying the information.

A digital certificate contains at least three things:

- A public key.
- Certificate information, like identity and information about the user.
- One or more digital signatures.

Certificate format

There are different formats of certificates; the most popular are PGP and X.509 certificates.

The first one is PGP certificate and it stands for *Pretty Good Privacy* [14]. This format includes the following information but it is not limited to this and it can contain more:

- PGP version number.
- The certificate holder's public key.
- The digital signature of the certificate owner.
- The certificate's validity period
- The preferred symmetric encryption algorithm for the key

The second one is X.509 certificate and is very common format. All X.509 certificates comply with the ITU-T X.509 international standard according to [14]. Unlike PGP this kind of certificate must be signed by someone known as CA (Certificate Authority). This format contains the following information:

- The X.509 version number.
- The certificate holder's public key.
- The serial number of the certificate.
- The certificate holder's unique identifier.
- The certificate's validity period.
- The unique name of the certificate issuer.
- The digital signature of the issuer.
- The signature algorithm identifier

3.5 Graphics

Java supports change of the appearance of frames and other visual components by the use of what is known as look and feel. To select a particular appearance this method can be called with ready themes available in Java. However these standard themes are hideously ugly by reasons only known by their inventors.

To solve this, a package known as Skin Look and Feel or SkinLF can be used [15]. This provides the possibility to load a theme package of own drawn pictures rather than the Java default. By using this package it is possible to load a theme and tell Java to use the pictures instead of the standard according to a theme description file which is included in the theme pack.

However now when we have the means of adding a theme to the application we need some cool graphics. Since we are rather poor at drawing ourselves we contacted a person that has created a theme he calls Alluminum Alloy / Toxic and his name is Max Rudberg. With his approval we can now use this theme package in our application to remove the native Java look and feel. For more information please refer to Max's home page [1].

3.6 Logging method

To understand what is happening in program when debugging there is a need for visualizing the events. This is normally done by adding print statements directly to the *standard out*. However, when the application is out on the market it is not possible to solve problems in this way since a normal user wouldn't like this kind of behaviour from any program. To still be able to monitor the program and find bugs the program has a logging function⁸ that simply writes print statements to a file. This file can then be analyzed remotely to find the particular problem. More information about log4j can be found at the official log4j home page [16].

4 Implementation of the Spider application

This section will present a detailed presentation of the solution and what technology we have used for the different parts and finally show what the application looks like

4.1 Solution overview

This section gives an overview of the entire system that will be referred to later on in a more technical manner. The solution consists of three parts:

1. The server
2. The certificate generator
3. The client program

The entire system is described figure 4.1:

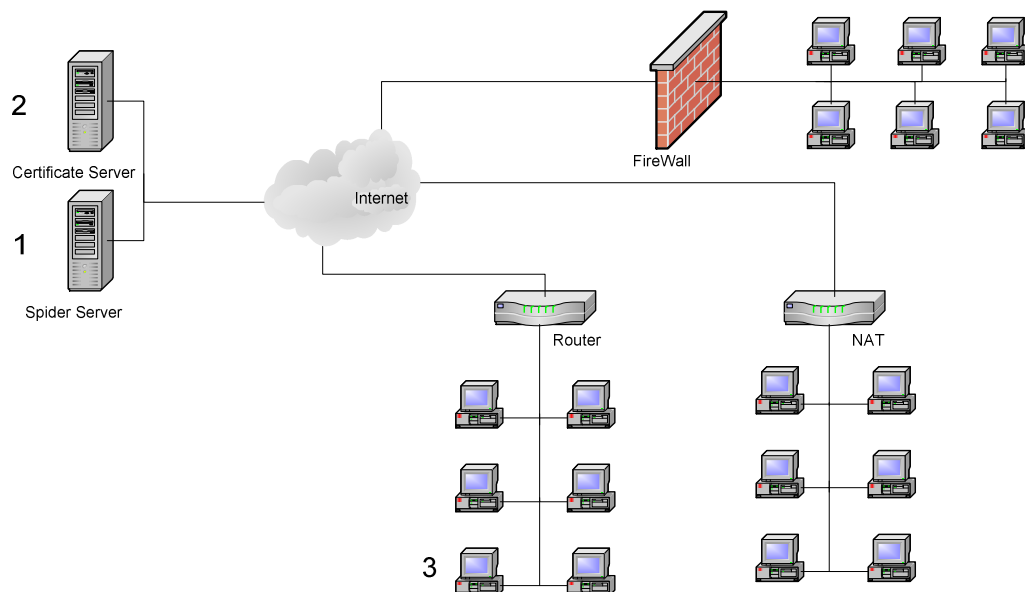


Figure 4.1

⁸ log4j version 1.2.9

Server

In order to let the clients connect to the private JXTA network we must have a central point of entry and that is what we refer to as the *spider server* or simply the server. The simplest description of the server is to say that it is a JXTA rendezvous and relay node with lookup and distribution functionality. When the server is started it launches the JXTA platform and joins a secure group known as “spider world” by reading a static advertisement for it from the hard drive. Then the server just sits idle and waits for connections

Client

The client program is simply a JXTA edge peer that must connect to the server in order to connect to the JXTA network. To start communicate with the server and other peers the client program also reads a static advertisement for the spider world group. Now when the client is in the right group it loads a static pipe advertisement from the hard drive and connects an output pipe to it. The other part of the pipe is located at the server which will be waiting for calls from the connected clients. To communicate with other nodes the client will use a propagated⁹ output pipe which other peers can connect an input pipe to in order to listen to this peer. The name of this particular output pipe is known as the broadcast pipe and the advertisement for it will be referred to as the broadcast advertisement. More information will be found under client – client interaction.

Certificate Server

To let the users obtain a certificate without having to call or contact the certificate authority a secure web page (figure 4.2) will be used for certificate generation. This webpage will require a login and a password for the user who wants to create a certificate. The required information the user must enter will be the main components for the certificate. When all information has been entered and the submit button is pressed the server will generate a private and public key as well as information of the validity of the certificate. All information except the private key will be hashed and signed by the server’s private key. The user’s private key will be encrypted with the password entered and the certificate will then be sent to the entered mail address.

The screenshot shows a web browser window titled "LCA@CPM portal :: ISO/TS 14048 data documentation tools and more - Microsoft Internet Explorer". The address bar shows "http://kattmynta:8080/nukes/index.html?module=spiderCertificateServer&op=createCertificate". The page features the "LCA@CPM" logo and a "CPM" logo. A left sidebar contains a "Main Menu" with links like Home, Modules, Project management, Manage groups, SPIDER, Manage.html, LCI Data Input, Change your info, Select theme, Logout/Exit, LCA@CPM, My Account, and LCI Store. Below the menu, it says "Who's Online" with "Currently, 0 guest and 1 member are online." and "This site is developed by Industrial Environmental Informatics CHALMERS". The main content area is titled "SPIDER" and "Generate a certificate". It asks the user to "Please enter the following information which we need to generate a certificate for you:". Below this is a form with fields for "Full name", "Company/organization", "Country", "Email", "Password", and "Repeat password", followed by a "Submit" button.

Figure 4.2

⁹ See pipes under section 3.3

Client – Server interaction

When both the client and the server have entered the spider world group and read the pipe advertisement mentioned above they can start communicating. However, now the client also needs to get in touch with other peers and this can be done in two ways.

1. The client can use the discovery service which is provided in the newly joined spider world group.
2. The peers can obtain the broadcast pipe advertisements from a central point of distribution i.e. the server.

In theory the first idea would be possible to implement by for example publishing the broadcast advertisement locally and then let other peers perform `getRemoteAdvertisement`. This would be a nice solution if it worked in practice, but it doesn't. To achieve the above stated goal to obtain advertisements from other peers, the following must be performed:

First a listener must be connected to the group's discovery service and then the peers start to perform remote publish of their broadcast pipe advertisement. This must be done at regular intervals, for example every 10th second. When other nodes are receiving this, their listener will connect an input pipe to the received broadcast pipe advertisement and the peers are connected. This will generate a large amount of traffic which will slow the network and does not guarantee that all nodes will connect to one another which is a requirement for the solution (see requirement specification 4.3: General specification).

The second alternative is to let the peers obtain broadcast pipe advertisements by acquiring them from the server or any other fixed location. The advantage of distributing the advertisements in this way is that we can control the traffic flow in the network that deals with advertisement discoveries. We will also be able to control membership in groups if we implement a way of checking against a database before we send advertisements to the clients (see future improvements)

We have used the second alternative to our solution mostly because we want to guarantee that all members in a group should be able to communicate with one another. The distribution is solved in the following way:

When a client connects to the server it will send a request to start downloading broadcast advertisements. In this request the broadcast advertisement for the requesting node is also included to let other nodes download it. To avoid traffic bursts the server will react by adding the node to what might be referred to as a subscription list. When the node is registered it will start receiving broadcast advertisements that belongs to other nodes, one every 5th second. This will stop when there are no more advertisements for the server to send. Every node will in this way receive the advertisement that is obtained from its own index in the advertisement array. The index will move when the nodes have received a new advertisement and when there are no more left, they will stop receiving until a new node adds its advertisement. Figure 4.3 describes how this is done.

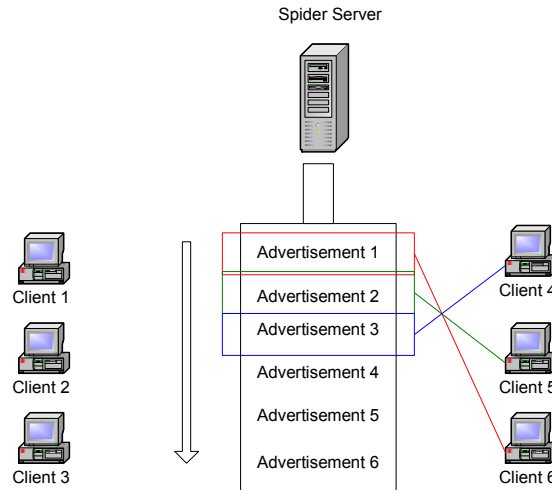


Figure 4.3

Client – Client interaction

As stated in the technology section, there are three different sorts of pipes. The pipes we use in the application are almost always propagate since they provide a nice way of sending information. The main advantage of propagate pipes is the ability to send messages to several nodes by just using one out pipe. This should be compared with the option where every pipe is point to point and to send a message to all the other peers the program will have to iterate through the output pipes that are connected to the respective node. Propagate pipes also makes it easier to handle the connections to all other nodes since only one pipe is needed for all outgoing communication.

In our application we have used propagate pipes in the way described in figure 4.4. Every node is equipped with a propagate output pipe that is constructed from an advertisement. When other nodes get hold of this advertisement they can connect an input pipe to it. The result will be an output pipe that can be used by the node to send messages to all other nodes connected to it. This particular output pipe will be referred to as the broadcast pipe.

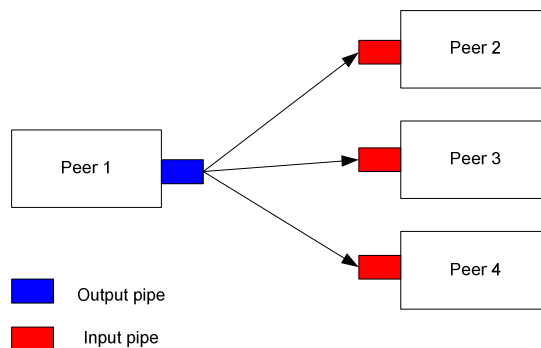


Figure 4.4

Now when we have declared the setup for the communication we intend to go into detail of what is sent over these connections and how we solve the security issues. There are two sets of sessions we intend to explain. The first is the file query and the second one is the file request.

File query

In order to ask other nodes for files we simply broadcast a search string through the broadcast pipe which the JXTA network will distribute to the other connected nodes. The procedure is known as the file query and the interaction between the nodes is presented in the following figures. Please note that the first message isn't encrypted due to the problem of key distribution.

File query phase one

In order to create ask for files the following items must be added to the message:

- *Search string*
A search string is a piece of information created by the user that will be used for pattern matching on the receiving end.
- *Reference*
Since the search performed needs to be unique to the node that sent the file query a unique number is added to the message.
- *Advertisement*
The advertisement that is added to the message is created for a temporary input pipe that is ready to receive any sent replies.
- *Certificate*
To let the other nodes be able to encrypt responses a certificate for the asking node is added to the message.

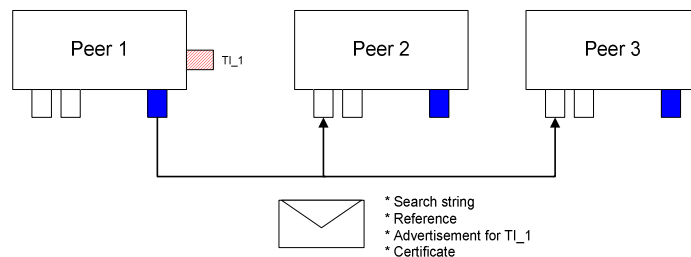


Figure 4.5

File query phase 2

In this phase the peers that have received the search string have searched their shared files for a match. Since an advertisement for the receiving input pipe in peer 1 was included in the first message the reply can be encrypted. The encryption we use for the message is 128-bits AES and this session-key is encrypted with the recipient's public RSA key (taken from the certificate) and is added to the reply. To send the reply the node simply extracts the provided advertisement from the message and connects an output pipe to it. Due to the propagate properties of the temporary input pipe in peer 1 many output pipes can be connected to it. The reply that concludes the search query contains the following information:

- *Matching file names*
This is simply a string that contains the file names of the matching files

- **Search string**
This is included to conveniently set the name of the reply tab (se program presentation) It is the same that was firstly transmitted
- **Reference**
The reference that was sent is stored in the reply.
- **Advertisement for I_1**
To let the first peer know where the files can be obtained the answering nodes also include an advertisement for an input pipe where they can be requested from.
- **Certificate**
The replying node add its certificate to the message in case peer 1 wants to send a file request later on using encryption. The certificate is also used to present peer one with information of the file name provider (peer 2) when the file name is selected in the table
- **Encrypted session key**
This is the encrypted AES key that has been used for encrypting the reply

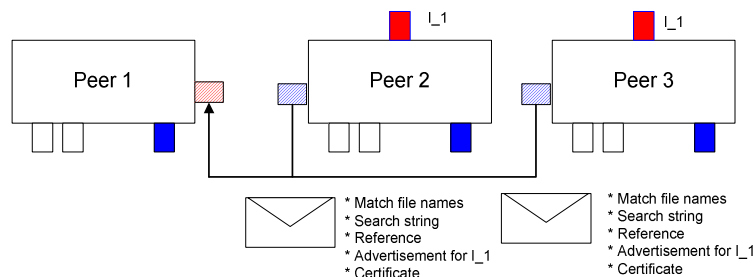


Figure 4.6

File request

When peer 1 has received the matching filenames these are displayed for the user in a tab and he or she can decide whether to download any of these files or not. This is also performed in two steps.

File request phase one

The file query starts when a node wants to download a file from the resulting file names acquired during the file query. Since all the file names that have been received by peer 1 is stored together with the advertisement for I_1 and the certificate for peer 2, peer 1 can initiate an encrypted request session directly with peer 2. The request is sent from an output pipe created from the advertisement for I_1 that was provided from peer2 and contains the following information:

- **File name**
This string is the requested path and the file name to make it unique if two files have the same name.

- Advertisement for TI_2
To be able to receive the file from peer 2 if everything works out, peer 1 includes an advertisement for a temporary input pipe. This pipe is point-to-point and thus can only be connected to one output pipe.
- Certificate
To encrypt the file that is to be sent back we also include the certificate for peer 1
- Encrypted session key
This is the encrypted AES key that has been used for encrypting the request

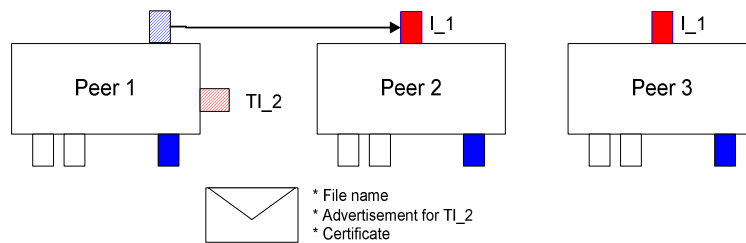


Figure 4.7

File request phase two

When peer 2 has received the file request message, it will check if the file is still present and if it is, send the file to peer 1. The reply contains the following:

- File name
In order to store the file to the hard drive a name must be provided for the file
- File data
The content of the file is located under this tag in the message
- Certificate
Placed in the message to identify the sender of the file
- Encrypted session key
This is the encrypted AES key that has been used for encrypting the reply

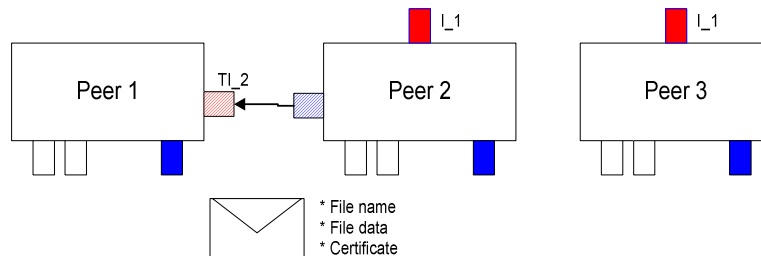


Figure 4.8

Comment on the abundant use of pipes and certificates

As the reader might have noticed, many pipes are used to pass messages between the different peers and the reason is to take advantage of the listener functionality. When constructing an input pipe, a listener can be attached and if a specific listener is attached to a specific input pipe the need for declaring the type of message will be redundant. For example the message sent in file request phase two can easily be separated from the message that is sent in file query phase two since the program is using two different listeners for these pipes. Since it is possible to let the JXTA platform takes care of the message distribution rather than having one large input pipe receiving all the messages and distribute them according to type this proves to be a nice solution.

The added certificates to the messages every time they are sent are today redundant in some cases but in a future version where message signatures have been implemented they will play a vital role (see future improvements).

4.2 Code structure

In this section the code structure of the program is reviewed. The Java classes used in the application and their corresponding use are visualized in figure 4.9. To understand their purpose a short description for each one is provided below.

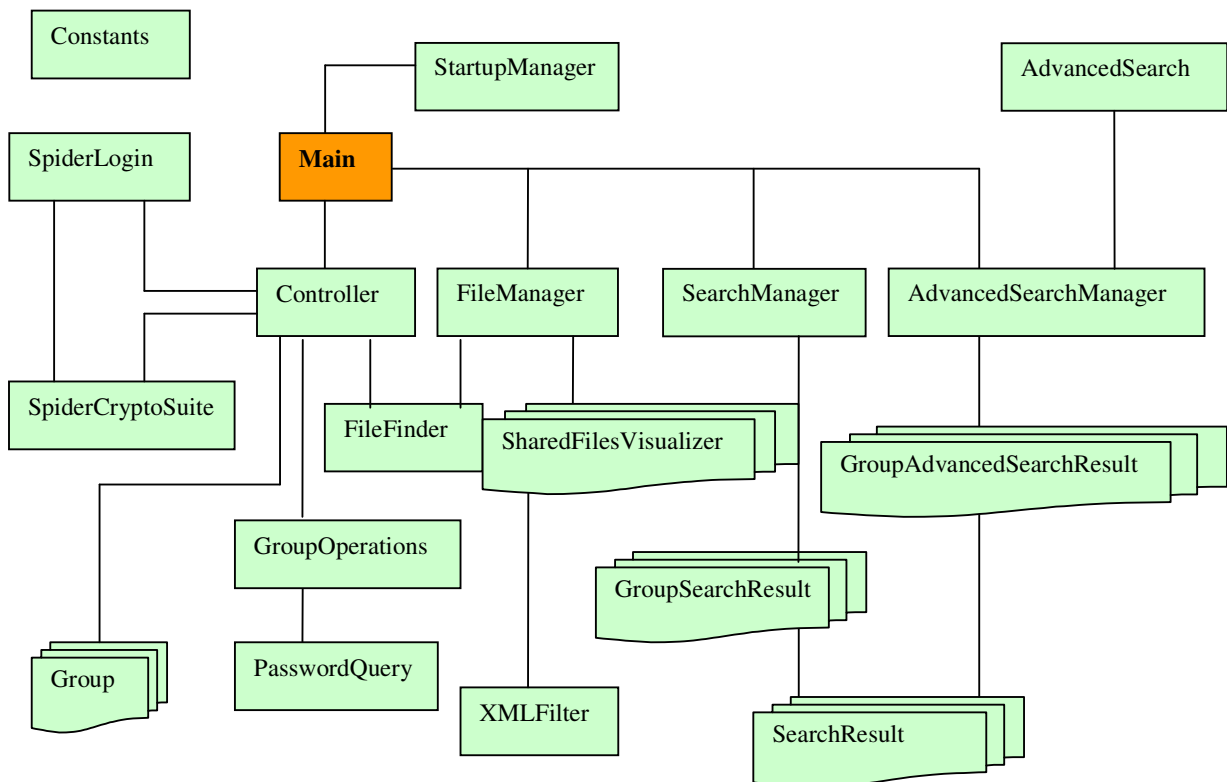


Figure 4.9

The program is started by invoking the main method in the Main class. This method starts by reading some parameters like server's IP address and download folder's name from a property file. And then an instance of Main class will be created to start the application.

This package called com.imi.spider and includes the following classes:

- **AdministrationTool:**
This class is used for creating group advertisements that can be read and joined by the client programs
- **AdvancedSearch:**
This class is used to interpret the advanced search specification file to a friendly user interface for the advanced search functionality (see section Advanced search panel under 4.5).
- **AdvancedSearchManager:**
This class is used to manage advanced search queries. This class creates an instance of AdvancedSearch from a file provided from the server and then displays this as a panel in this class. The class also contains a tabbed pane for handling the groups that have been joined which can be selected in this class by the user to perform an advanced search for that particular group.
- **AdvertisementStorage:**
This class is used by the server to keep track of what broadcast advertisements it has sent to a particular client. One AdvertisementStorage is created for every client that connects to the server.
- **CertificateAuthority:**
This class is called by the homepage where the users can obtain their certificates in order to create the certificate file that will be sent to the user.
- **Constants:**
This class contains almost all constants that are used by this package.
- **Controller:**
The controller class is the heart of the client program. It is the class that starts the JXTA platform and it also handles everything that occurs in the default spider world group including joining new groups.
- **FileFinder:**
This class is used for administrating groups and the files that can be shared for each group. The function for both normal and advanced search is performed here.
- **FileManager:**
This class mainly consists of a tabbed pane where all the joined groups are displayed. It allows the user to select a group which will show the corresponding instance of SharedFilesVisualizer where the user can change the shared files for the selected group.

- **Group:**
This class is used for handling all the events that occur in a joined group. A new Group is instantiated for every new group that has been joined except for the spider world group which is handled from Controller.
- **GroupAdvancedSearchResult:**
This class is used to manage and show the search results for advanced searches in a specific group. This class creates an instance of SearchResult to view the search results for every joined group.
- **GroupLoginPanel:**
Used for displaying the reply from the server which contains the groups that can be joined when sending such a request.
- **GroupOperations:**
This is a convenience class that provides all operations that can be associated with JXTA groups. For example does it provide a function that will take an advertisement as input and return a group.
- **GroupSearchResult:**
This class consists of a tabbed pane for displaying the search results in a particular group selected by the user. This is done by displaying the corresponding instance of SearchResult for that particular group.
- **Main:**
This class is used to start the spider application and contains and manages all user interfaces.
- **PasswordQuery:**
This class is used for obtaining a login and a password from the user to login to a particular group.
- **PasswordReader:**
This class is used to prompt the administrator for a password from *system in* that will be masked.
- **SearchManager:**
This class consists of a tabbed pane where all the GroupSearchResults are stored in order to let the user select a particular group to check the search results in them.
- **SearchResult:**
This class displays all the matching file names that have been received from a search query. The file names are stored in a table where they can be selected and downloaded from the providing peer. Since it is used both for advanced and normal search result visualizing it will look differently depending on which class that made the instantiation.

- **Server:**
This is a JXTA relay rendezvous/super node. It provides a central entry into the network for all the client peers as well as certain services. Once connected, the server will provide the peer with broadcast advertisements for other peers and if the peer wants to join other groups, their advertisements will be obtained from the server.
- **SharedFilesVisualizer:**
This class shows the files the user have decided to share in this particular group but it also allows the user to change this.
- **SpiderCryptoSuite:**
This class is used to perform all cryptography operations, which are generating keys (RSA and AES), encryption, decryption, signing and verifying. All methods are static and can be used without creating any instance of this class.
- **SpiderLogin:**
This class is used to create a user interface that asks the user to choose a certificate file and to enter a correct password to log on to the application. This class uses methods in class SpiderCryptoSuite.
- **SpiderPGPCertificate:**
This class is the programming digital counterpart of the certificate that is stored on the hard drive. It is capable of holding this information as well being created from different type of sources whether they are files or sent as a byte streams.
- **StartupManager:**
Displays the animated spider when the application starts.
- **XMLFilter:**
When connected to the file chooser class in java, only folders and XML files will be visible for selection.

4.3 Key management and certificates

The main key to start Spider application is a valid certificate. This lets you come in to the Spider world and begin to communicate with other peers in this world. All peers must have certificates that are signed by the same CA in order to verify each other's certificate.

We have mentioned earlier in section 3.5.4 different formats of certificates with their contents that we might use, but there are some problems with using them without modifications:

The X.509 format lacks some fields like the unique identifier of the certificate's holder since this identifier is intended to be unique across the Internet [14].

If we use PGP format then it doesn't contain root-signature which can be used to verify other certificates.

After discussion with our employer and the way they want to distribute these certificates we had to define a new format that is more suitable to our application. This format is a hybrid of the formats mentioned above and it fulfils the security requirements for this application. The following sections mention briefly its contents, how to distribute it and some pros and cons.

Construction

This certificate contains the most fields from both the X.509 and PGP format and they are:

- **Unique number:** Each certificate has a unique number that is generated only for this certificate.
- **Valid from:** Date when the certificate was created.
- **Valid to:** Date when the certificate is expired.
- **User name:** User name (full name).
- **Company name:** Name of the company where the user is working at.
- **User email address:** Email address of the user.
- **Encryption algorithm info:** Information about the symmetric-key encryption algorithm that is used to encrypt both texts and private keys.
- **Public key algorithm info:** Information about the public key encryption algorithm that is used by this certificate.
- **Signature algorithm info:** Information about signature algorithm that is used to sign the content of this certificate.
- **Public key:** The public 1024 bits RSA key for this user.
- **Private key¹⁰:** The corresponding private key. The key is encrypted by the user's password.
- **Signature:** The CA's signature of all the content of this certificate except the private key.
- **CA's public key:** The public key of CA to be used by the certificate's holder to validate other certificates.

This file has xml-format and the extension of the file is scer, which stands for Spider Certificate. The above mentioned fields are used as elements which contain their corresponding contents.

Distribution

How do the users get their certificates? How can they trust the certificate provider? Several questions must be answered in order to find a good way of distributing these certificates under high security.

Our employer has requested that users should be able to create their own certificates therefore we think that a web-server for this purpose is the best solution. Each user may have a password to log onto this web-server and start creating her/his own certificate. Each certificate is protected by a password that is chosen by the user and the user's private key is encrypted with this password. The result is sent to user's email address.

By server we mean the CA and it has a public and private key. Each user certificate will be signed by the CA's private key and each certificate contains a copy of CA's public key, as mentioned above, which is used later by the user to verify other certificates.

¹⁰ Please note that the private key is not included in the certificate when the peers communicate.

Pros

- Easy to create and distribute certificate since we use a web-server.
- One server contains all necessary information for this application.
- The user doesn't need to think about how to verify other certificates since the user's own certificate contains the CA's public key and verification of other certificates can be done transparently by the application.
- It is flexible because each user can use its certificate any where on any machine.

Cons

- A certificate is valid until it expires. Another person can use this certificate if he/she obtains the corresponding password. See section 5.3 future improvements to see how to overcome this problem.
- A web-server may be vulnerable for hacker. All servers are.
- If CA changes keys (private and public) then all certificate become invalid and all users must obtain new certificates.
- If the user has chosen an easy password then it will be easy for hacker to crack this password. The only restriction on the password selection is the length must be 8 characters and the rest is entirely defined by the user (see 5.1 Security assessment).

To see an example of what a certificate looks like please refer to appendix A

4.4 Problems encountered during the work and their impact

Under this section we will present the problems we have encountered and their impact on the resulting solution. This is meant to provide a deeper insight into why our solution turned out the way it did.

4.4.1 Poorly documented platform

Description

The JXTA platform provides the above stated functionality but in order to use it i.e. put it into code, examples and well-documented APIs are needed. This is not provided in any form from the JXTA community. The most helpful code we could find is by the character of "Hello World". The API that is provided with the platform does not offer anything more than the exact return types and input values.

Impact

Due to the absence of usable information the construction took unproportionally long time. We were also unable to define a program structure at the beginning of the project since we didn't know how the platform was operated. This also led to a programming style best described as trial and error.

4.4.2 Continuously developing platform

Description

The JXTA platform is subject to continuous development. This results in some of the existing example-code to be outdated due to newer platform versions. For books written on the subject of teaching, this is particularly true since they have a tendency of being outdated when they hit the bookstores. Another problem that arises when code is under development is the numbers of undocumented bugs that hasn't been reported yet.

Impact

When we first started to work with the JXTA platform we used the book: Mastering JXTA [17] as our reference. However the book doesn't state what version of the platform it uses for creating the examples. The results from running such code are deprecated methods and error generation among other things. This resulted in distrust for everything that didn't state what JXTA version it supported and this slowed our development rate since we had to test every piece of code we came across.

4.4.3 JXTA configuration panel**Description**

When the JXTA platform is started the first time it will produce a configuration panel used for configuring connections to rendezvous and relay peers among other things. This is needed for setting up the network and can be replaced by an own configuration. However the first configuration panel is incredible unintuitive, undocumented or described anywhere it deserves a private section in the report. Due to its poor interface we never created a private network i.e. connected to a computer in the local network but instead connected to a public JXTA rendezvous computer.

Impact

Since we accidentally connected to the wrong rendezvous and relay peer due to the bad configuration interface we could only find peers that were members of the local network. The reason for this is that JXTA uses multicast in local networks and thus is able to quickly find other peers on this particular network. The peers that are outside must first be resolved before they can be discovered and that takes several minutes. Due to this we not only lost a lot of time but we also started to disbelieve the platforms capabilities. We have now solved any future incidents of this type by constructing our own hard-coded configuration that replaces the visual default.

4.4.4 Poor possibilities for large scale testing**Description**

Since we only have access to a single network with a limited number of computers we have been unable to test our solution in a large scale. Bugs are relatively common in programs under development and to find most of them before the solution is released, the application needs to be tested by a number of people as well.

Impact

Due to this problem the program is not properly tested and since we can only solve bugs that we discover ourselves, we cannot guarantee that the program will operate properly when it is released.

4.5 Presentation of the application

Under this section we intend to put it all together and show how the application works.

Getting started with the application

In order to start using the application a user must first obtain a copy of the software. This can be accomplished by either downloading it from a web server [18] or obtaining a copy from any type of storage medium. When the application is installed on the hard drive, the user must obtain a certificate to start using it by logging in to a secure webpage and obtaining a certificate by entering the desired information as mentioned above (also see future improvements).

Selecting and unlocking the certificate

When the user has a copy of the client application and a certificate, the program can be started. The first screen that hits the user when starting is the certificate selection screen which allows the user to specify his or her certificate. To unlock/decrypt the private key in the certificate the user must enter his or her private password. The program will then test if the correct password has been entered by encrypting something with the user's own public key and then try to decrypt it with the private key. If successful the program will start the JXTA platform.

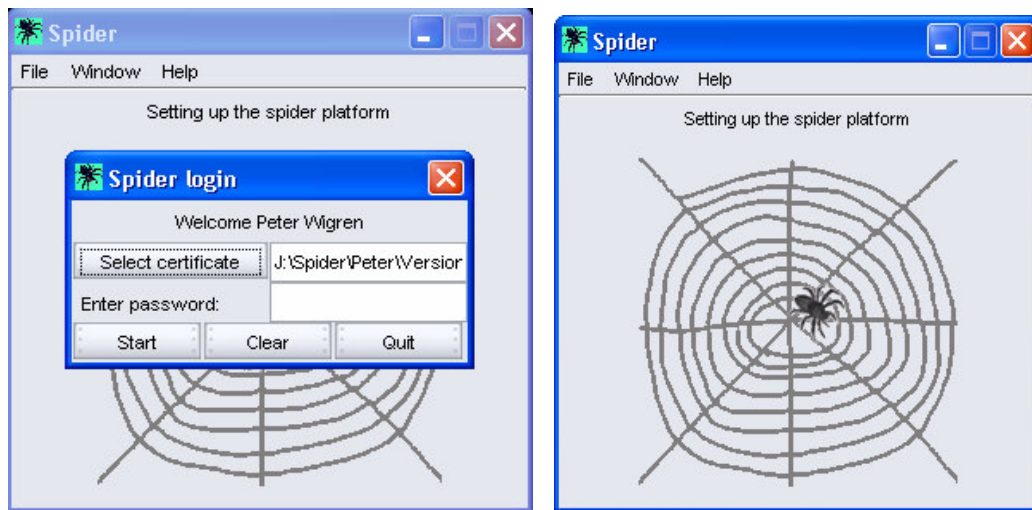


Figure 4.10

Logging onto the network

To create a private setting and let the users of the application be separated from other JXTA peers in the *Net Peer Group*¹¹ the users need to login to a secure group known as *Spider world*. To start the application the login and the password must be entered correctly or the program will shut down since it is configured to work under this particular group.

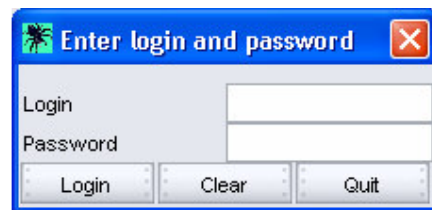


Figure 4.11

¹¹ See figure 3.4

Using the application

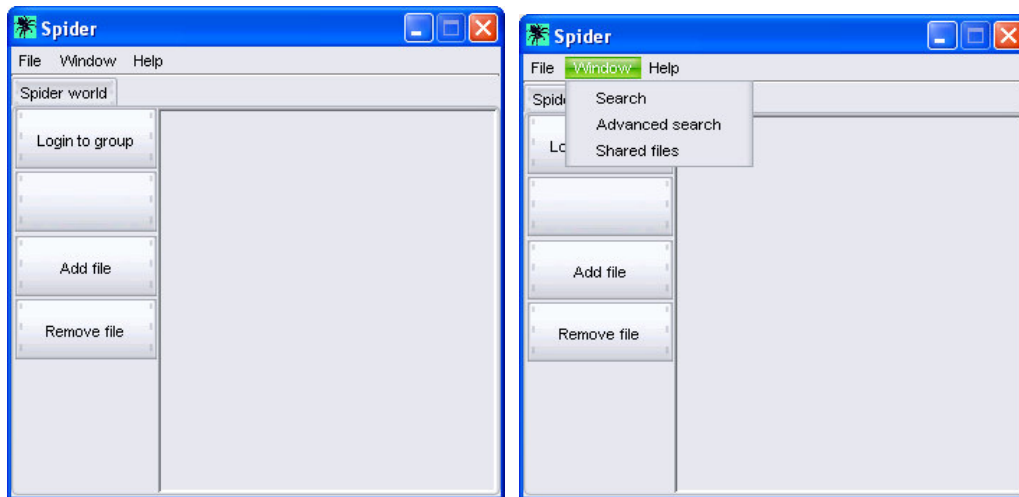


Figure 4.12

If the login is successful, the program that consists of three main panels is ready to use. By default, the first panel that welcomes the user is the administration panel known as *shared files* in the menu (see figure 4.12). It provides the following functionality:

- Allowing the user to log in to subgroups that are located under the *spider world* group and leaving them.
- Adding and removing shared files shared XML files for the selected group (in this case *Spider world*)

The other two panels under the window menu provides basic search functionality and a more advanced search function for XML documents in the ISO/TS 14048 format which was mentioned in the background section. Now we intend to in detail explain how every panel is operated.

Shared files panel

Under this panel the user can select a group he or she has joined and decide which files to share with this group. This is simply done by selecting the group and pressing the add files button. The files will then be available for other users in that group to be searched and downloaded (see figure 4.13). To restrict the user to only share xml files we perform a check when they are added to prevent this application from becoming a KaZaA clone.

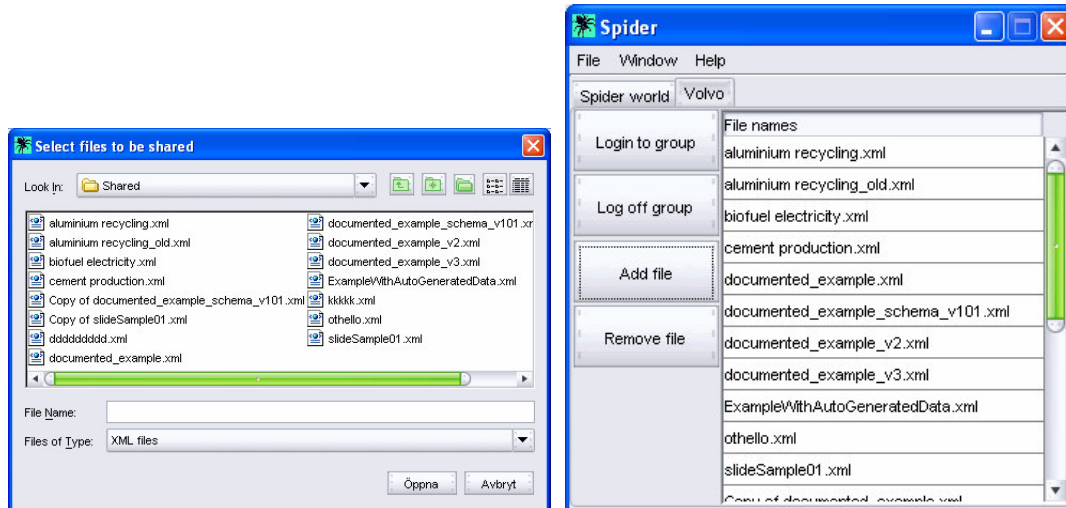


Figure 4.13

As mentioned before this panel is also used to log onto other groups and leaving them. To do so the program will send a request to the server which will respond with a list of possible groups to join (figure 4.14 shows this). When a group is selected the login button can be pressed which produces yet another group login screen (see figure 4.11) the result of this is a new tab named according to the group. Under this tab new files can be shared for this group entirely independently of other groups.



Figure 4.14

Search panel

This is the basic search function which can be used to search for more than files that support the ISO/TS 14048. The main purpose of this panel is to provide a quick search mechanism that also includes files that doesn't support the ISO/TS 14048 format.

To execute a search a word or phrase must be entered into the text field depicted in figure 4.15 which is then broadcasted to all other nodes connected to this particular node. When receiving this they will search the shared files under the group they received the query from and if any matches are found the filenames will be returned and displayed according to figure 4.15. In order to obtain information about the owner/provider of the file name the name in the table can be pressed which triggers the program to view a selected part of the certificate of the owner. To download the file the user just presses the download button which will result in a stored copy of the file in the received files directory.

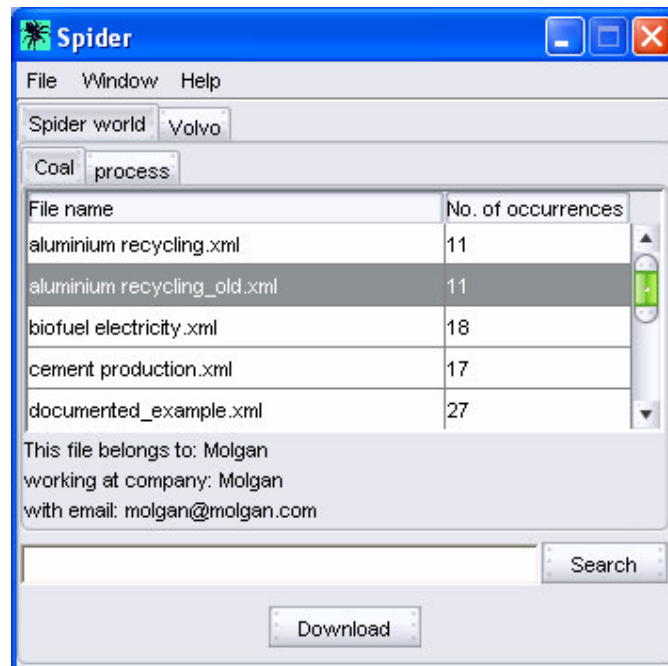


Figure 4.15

Advanced search panel

The second function is used only to search for ISO/TS 14048 documents and is referred to as advanced search. The user has to have some knowledge about ISO/TS 14048 format to be able to use this functionality. Advanced search is used when the user wants to be more specific and search for an ISO/TS 14048 document. To make an advanced search query the user must specify the search string according to the right panel seen in figure 4.16 A file known as the advanced search specification file, which is obtained from the server when the client first connects, specifies this panel and it specifies the fields that can be searched for according to the standard. The mechanism makes the program compatible with future versions of the ISO/TS 14048 standard.

The advanced search specification file that is loaded in figure 4.16 specifies six fields which can be searched for a specific content. There are two types of fields and the first is the ordinary free text field and it allows the user to enter a content that will be matched against the corresponding field in the xml file. The matching is done according to the combo box to the right of the field where the user can specify whether the matching should be exact, check for a substring with the entered value or check if the text in this element starts with the desired value.

The other type of field is the selected box field (in the figure: the lowest to the right) and it allows the user to select a predefined value under a certain element. This is possible since only a number of text strings are allowed under these elements.

To download the file the same procedure as mentioned for the search panel is performed and this concludes the presentation of the application.

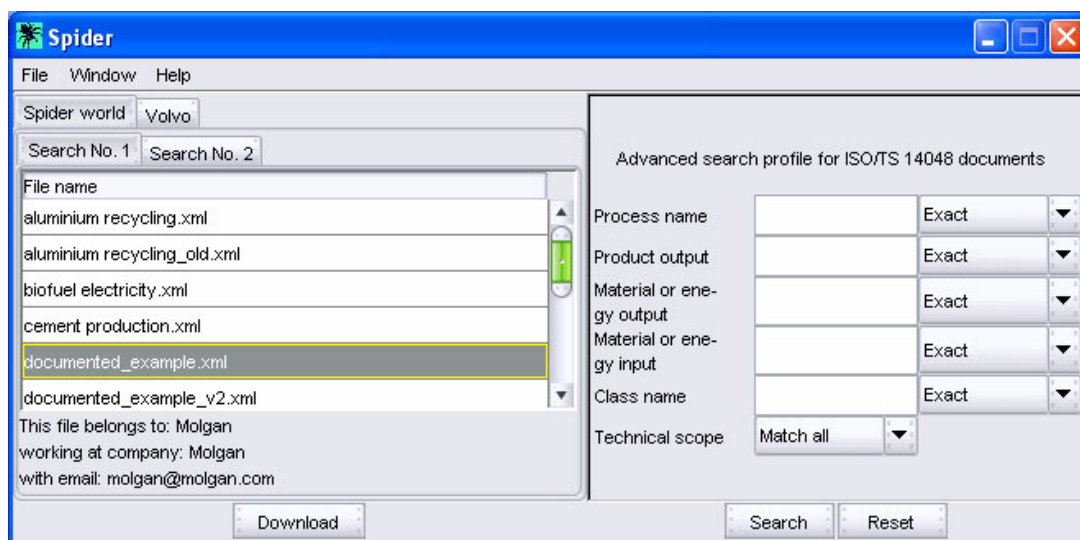


Figure 4.16

5 Analysis of the program

Under this section we intend to address issues such as security, how the generated traffic will affect the network and improvements of the application for future releases.

5.1 Security assessment

Since security is never stronger than the weakest link in the chain, we intend to focus on the security in this program and try to locate the weak links. Under this section we will present a number of attacks that we think are possible due to our way of writing the application and how likely they are to succeed.

The primary threat that faces a user of our application is that someone outside the community enters and tries to obtain data that he is not entitled to. There are currently several known ways of doing this and we will try to state how an attack can be performed and how likely it is to succeed.

5.1.1 Certificate attacks

Since the certificates are vital to the communication between the peers someone might obtain one and use it in the purpose of gaining entry to the network. By using a certificate the attacker can do a number of things to disrupt and intercept traffic between the peers and this section deals with the possibilities of gaining access to a certificate.

Stealing the certificate during transfer

When a node performs a file query his or her certificate is sent unencrypted. If someone intercepts this message the potential attacker can obtain a copy of the transferred certificate. However when we send the certificate in this way the private key is not included which renders the certificate useless. The attacker can use the certificate to perform file queries, but the responses are encrypted with the public key and the attacker cannot decrypt the response. Moreover it isn't possible for the attacker to switch the public key into something he has the private counterpart to since the certificate is signed.

Cracking the certificate

This scenario deals with the idea that an attacker has gained access to a certificate that contains everything. To make use of it the attacker must decrypt the private key since he otherwise cannot decrypt the received answers. The easiest way of doing so is to try hashing different passwords and decrypt the private key with this. Since the encryption key cannot have more combinations than the number of combinations for the characters in the password, the password is the weakest link. To get the desired information about the key length and the password length the attacker can simply obtain a copy of the program and decompile it.

We will present a test we have performed to show the time that is consumed to crack a private key that has been encrypted with a password. In our test we suppose the attacker is using brute force and try all possible keys for the security level of the password. The time it takes for testing every possible key is 0.030 seconds which also is the time it takes for our program to verify a key on a 1 GHz Intel Pentium 3. To try a key we simply encrypt something with the public key and decrypt it with the possible private key. If the plain text and the decrypted math we have found the private key. The result is shown in table 5.1

Security level	Password type	No. possible combinations	Total time to test all combinations
1	AbDratxF	$52^8 = 5.3 * 10^{13}$	$0.03 * 52^8 = 50855$ years
2	3b4rAtx6	$62^8 = 2.1 * 10^{14}$	$0.03 * 62^8 = 207705$ years
3	3b&rAtx%	$96^8 = 7.2 * 10^{15}$	$0.03 * 96^8 = 6862534$ years

Table 5.1

The security level is better the higher number and we assume that the passwords are completely random. The first security level password just contains a combination of small and capital letters. The second level also contains numbers while the third and most secure type of password also includes alphanumeric characters which in this test are the following:

!"#\$%&'()*+,-./:;<=>?@[\\]^_`{|}~ (plus the space character). To make this test more generic we only use the English alphabet.

This test gives a fair estimate of the security of our approach and it also shows that the least secure password is sufficient to protect the private key. However we believe there are faster ways of testing passwords but even if it is possible to optimize with a more mathematical approach the RSA algorithm still weighs heavy on the processor. Even if it was possible to enhance the testing by a factor thousand and use parallel testing on different computers the highest security is still too much for any attacker.

5.1.2 Sending trash

This kind of attack involves sending trash to the peers in order to make their life miserable. To send trash to a peer the attacker must only gain access to a valid certificate which can be done in the first step in the file query phase. When this is done he can use this to send anything he wants to any peer in the network. Currently our application has no protection against this kind of threat but we have a solution. In the future implementation section we discuss how signed messages can be implemented by using the certificates and this also solve this problem since an attacker doesn't have access to a valid certificate with decrypted private key.

5.1.3 System attacks

When we are referring to a system in this section we mean the system the program is running on and the computer that runs the program.

Hacking

There are number of ways of gaining access to a computer. We will not discuss how this can be done but rather the effect for our program if this occurs. As we see it there is no protection from this threat. If an attacker gains access to the computer he or she can monitor every keystroke done by the user and thus the password for the certificate as well as the certificate itself.

Social engineering

The easiest way of gaining access to something is to ask for it. This can be done by anyone pretending to be someone else. A practical example of this is the attacker who pretended to be the system administrator and over the phone asked user for their passwords. This can be applied to our application as well since the principles are the same and the only protection for this kind of threat is security aware users.

5.2 Network impact

In this section we intend to address how the messages are sent over the network and how large impact the communication will have on the network.

Messages

To view how a possible attacker can watch the conversation we have decided to use Ethereal [19] to monitor the network. To exemplify what is sent during a session we have decided to include a small part of the file query message. In this file query we have been looking for the word COAL which is highlighted in the extract. It should also be noted that the message is entire unencrypted and that every part is indexed with tags we have decided to give the elements.

```
....
.content-type..application/x-jxta-msg.content-length.....l.jxmg.....jxel...
search_tag..text/plain;charset=UTF-8....COALjxel...
reference_tag..text/plain;charset=UTF-8....2jxel...
certificate_tag...n&...57083398050789957617090795070299057125....Monday, November
22, 2004....Tuesday, February 22,
2005....Molgan....Chalmers....Molgan@Molgan.com+...AES: Advanced Encryption Standard,
128 Bits,...RSA: The RSA encryption algorithm, 1024 BitsP...SHA1withRSA: The signature
algorithm with SHA-1 and the RSA encryption algorithm.#
```

The reply however is encrypted is viewed in the following extract. Please notice that the tags aren't encrypted and this makes it a whole lot easier for us to extract the elements from the message.

```
...
encryptedSessionKey_tag....E.~=$.....OO...(^.6&..C9...H5.k...a.....BOJ.4.8.....o....a.Z.h....
...A...qk.t_z.R.k.#.s....W.)...V.....,.....jxel....reply_tag...P6.=+,...N~2Pj.&.....h.G
.~ ...Q/.~o.g.T...N,...Ix.....}.A.p.....[o87.....a..^'..sI.
N....5qR...c....2..mR..9..7..a.[jxel...
search_tag...../..v(.yz..jxel...
reference_tag.....h.).&!)yjxel...
certificate_tag.....\Y....N?.....&=.....qm.Z...i....D.C.GL.?.....)E.....Hd.;...-
..YC..X....?9.(MH.....>.T.A0..uM.....j..1..B....}'!>....]'!G.3.:@..G.^..>T....~'...w
.....x....f.{.o>MY.....}...,O].y...@.R{.Fb./}....5...8.....Kp.fd(=6u..(.....
.g....Q...W.K...X.....^.....Wm-..v$+-!..s..!jv.l.."1..nK..?..W....t.P..F?
..&.S..O.7.]>..d.....1....l.B.9#gy....4x...ZyQ,d,Ah....O..zm..\....3+.
....."+...l.dm.93.-.L.....R...@.-T.....K4..O..D7...$s?.i.....D../[.&.....l.P. C!...-
...
```

Impact on the network

As noticed from this the messages are rather large and from what we have seen the network traffic can be described as quick bursts of information. The amount of sent information might be an issue when we present the system to the public, but we are uncertain of this since we haven't had the possibility to test it on a large scale.

5.3 *Future improvements*

There are several things in the application that can be improved to improve the functionality and security. We describe some of these in the following list:

- **Declare invalid certificates:**
The possibility to declare a particular certificate invalid before it expires if one has ended up in wrong hands will be vital as the system grows. Connecting a database to the server that contains a list of all the invalid certificates can do this. When a node connects to the server it will receive a copy of this list and will be able to drop messages from peers with invalid certificates.
- **Possibility to charge money for certain documents:**
To make the application more appealing for financial gain, the possibility to let users register some documents as purchasable can be added. In this solution we have assumed that all documents should be shared and used as much as possible but in reality the shared documents cost money to produce. Some users might not share expensive information because they don't want to give away something that is too valuable. The idea is to add some extra information saying the document isn't available for free download since the owner demands a certain sum of money to allow this.
- **Increase the community:**
Today there is only one super-node in the system that all other peers connect to. In the future this community can be connected to other spider communities by connecting the super peers (servers). This hasn't been tested yet but according to the JXTA platform guide it is possible to do so.
- **Decrease the information flow:**
As stated previously there is much information being sent between the peers in the application. To make the communication more efficient storing information about peers locally could eliminate some of the redundant information.
- **Add signatures to the messages:**
As mentioned in the security assessment section it is possible to attack the system by sending trash to a certain peer. To solve this all the information in the message could be signed by the sending peer's private key. However this is not enough since an attacker can repeat the sent message and send it a hundred times again. To prevent this, the message should contain a time stamp and a signature based on the entire message content. This signature would prevent any peer from accepting anything that is too old and not properly signed.

6 Conclusion and discussion

This section is intended to address the question whether the program is a sufficiently good solution to solve the problem stated in the beginning of this report. An evaluation of the JXTA platform will also be included as well as user opinions of the program.

How well does the application solve the problem?

To know this there are two approaches. The first is to strictly follow the requirement specification and see if the ready system is missing something that is denoted with *shall*. From this point of view the program is a complete success since it implements everything with the above notation. However a system does not have to be perfect because it completely follows a list of requirements. There also exist a factor which easiest can be defined as user appeal. The second approach is focusing on this way of examining the program. To let a user judge whether the application is a success or not several users have tested it in a minimal system mainly consisting of one server and two nodes.

The results were rather positive and the main response is that the application is very easy to use and performs as could be expected. The interface is intuitive and easy to learn but one problem has been located when the program is started. Since it is required that the user must enter two passwords where the other one must be combined with a login a slight confusion has been noted. This is due to the fact the program requires this to start and this does not seem intuitive to a user since the knowledge of groups might be inadequate. However since the same procedure will be performed every time the application starts it is considered the confusion will decrease with time.

How well does the used technology perform?

The technology used in this solution works almost flawlessly for one big exception: the JXTA platform. As mentioned in section 4.4: *problems encountered during the implementation* it is clear the platform has been the most time-consuming part in the project. Despite this, the spider application is a proof of the potential power that lies in it. However to make it more appealing to developers all documentation must be improved. If better APIs were created along with some more advanced tutorials, the JXTA platform would be a great tool for future high-level P2P solutions.

Final comment

With this application it is proved that it is possible to create an application with high level P2P programming provided by the JXTA platform with sufficient security. The solution is not perfect as noted in the future improvements section since there exists some functionality that can be improved. The lack of testing on a large scale could have impacts later on but only the future can decide whether the application will stand the test of time.

7 References

- [1] Official homepage of Max Rudberg www.MaxThemes.com
- [2] "The Saxon XSLT Processor from Michael Kay", <http://saxon.sourceforge.net/>
- [3] Henrikke Bauman, Anne-Marie Tillman (2004) "*The Hitch Hiker's Guide to LCA*" ISBN 91-44-02364-2 Page 21
- [4] Official homepage of Industrial Environmental Informatics – IMI <http://www.imi.chalmers.se/>
- [5] Information resources for *ISO/TS 14048* http://www.imi.chalmers.se/Projects/ISO_TS_Info_Resources.htm
- [6] About Extensible Markup Language (XML) at official homepage of *World Wide Web Consortium* (W3C) <http://www.w3.org/XML/>
- [7] STATSKONTORET. (2002) "*Vad är XML?*" Stockholm, ISBN 91-7220-427-3 <http://www.statskontoret.se/upload/Publikationer/2000/200030.pdf>
- [8] About XML Path Language (XPath) at official homepage of *World Wide Web Consortium* (W3C) <http://www.w3.org/TR/xpath>
- [9] Sun Microsystems, Inc. (2003) "*Project JXTA v2.0: Java™ Programmer's Guide*" http://www.jxta.org/docs/JxtaProgGuide_v2.pdf
- [10] Official homepage of *JXTA™ technology* <http://www.jxta.org>
- [11] Home of the Legion of the Bouncy Castle <http://www.bouncycastle.org/index.html>
- [12] John J. G. Savard. (1999) "*A Cryptographic Compendium*" <http://home.ecn.ab.ca/~jsavard/crypto/co040401.htm>
- [13] CNSS Policy No. 15, (2003) *Fact Sheet No.1* http://www.nstissc.gov/Assets/pdf/cnssp_15_fs.pdf
- [14] Network Associates, Inc. and its Affiliated Companies (1999), "*An introduction to cryptography*", PGP 6.5 documentation <http://www.pgpi.org/doc/pgpintro/>
- [15] Home of Skin Look And Feel <http://www.L2FProd.com/>
- [16] Logging Services Project @ Apache, documentation of log4j project <http://logging.apache.org/log4j/docs/index.html>
- [17] Joseph D. Gradecki (2002) "*Mastering JXTA*" ISBN 0-471-25084-8
- [18] Official Spider homepage http://www.imi.chalmers.se/spider/spider_info.htm
- [19] Ethereum homepage www.ethereal.com/

- [20] ISO 14040:1997 *Environmental management – Life cycle assessment – Principles and framework*.
- [21] Carlson R., Löfgren G., Steen B. (1995) "*SPINE, A Relation Database Structure for Life Cycle Assessment*"; Göteborg; IVL-REPORT
- [22] Weidema B P. (1999) "*The SPOLD file format '99*", <http://lca-net.com/spold/publ>
- [23] Official homepage for the *CASCADE project* <http://192.107.71.126/cascade/>
- [24] Official homepage for *COST Action 530*
http://www.empa.ch/plugin/template/empa/*/5600/---/l=1
- [25] Official home page of *SETAC* <http://www.setac.org/>
- [26] Official homepage or *LCA@CPM* <http://kakapo.imi.chalmers.se/nukes/index.html>
- [27] Official home page of *UNEP/SETAC Life Cycle Initiative*
<http://www.uneptie.org/pc/sustain/lcinitiative/home.htm>
- [28] Official home page of *EcoInvent* <http://www.ecoinvent.ch/>
- [29] Official home page of *JEMAI* <http://www.jemai.or.jp/english/index.cfm>

Appendix A

Spider certificate example

```
<?xml version="1.0" encoding="UTF-8" ?>
<Spider_certificate>
  <!-- This is Spider certificate which is hybrid of PGP and X.509 certificate formats -->
  <Unique_number>74679355638685380587403599708769509051</Unique_number>
  <Valid_from>Wednesday, October 27, 2004</Valid_from>
  <Valid_to>Thursday, January 27, 2005</Valid_to>
  <User_name>Muhamed Mostafa</User_name>
  <Company_name>IMI, Chalmers, Gothenburg, Sweden</Company_name>
  <User_email_address>muhamedm@imi.chalmers.se</User_email_address>
  <Encryption_algorithm_info>AES: Advanced Encryption Standard, 128 Bits</Encryption_algorithm_info>
  <Public_key_algorithm_info>RSA: The RSA encryption algorithm, 1024 Bits</Public_key_algorithm_info>
  <Signature_algorithm_info>SHA1withRSA: The signature algorithm with SHA-1 and the RSA encryption
    algorithm</Signature_algorithm_info>
  <Public_key>48,-127,-97,48,13,6,9,42,-122,72,-122,-9,13,1,1,1,5,0,3,-127,-115,0,48,-127,-119,2,-127,-127,0,-116,59,4,-77,-
    95,70,26,45,117,-8,109,-74,111,-116,29,-96,-28,-14,-65,-101,46,89,78,84,17,-78,70,26,-59,96,-6,-50,28,96,1,-33,-
    121,107,-77,-20,69,110,-87,-89,-13,-7,41,-5,103,-79,54,19,-125,-33,-31,-87,14,-116,-34,-91,-38,91,-17,48,99,16,-54,-
    76,-92,-28,47,-17,-103,5,17,61,9,-66,10,-68,56,-22,38,74,24,-118,-118,-52,-73,-128,-29,-73,-53,69,71,82,118,30,21,-
    66,68,-5,82,9,-31,122,-67,-32,66,48,93,-65,-99,26,30,100,-95,23,11,105,-78,-95,20,-20,65,-41,-
    6,81,2,3,1,0,1</Public_key>
  <Private_key>-52,82,-14,113,-15,32,-24,-75,-78,-66,-9,-98,-1,-115,-19,-85,49,-72,84,38,-56,-27,-80,6,-78,-103,-13,-75,36,-
    112,105,106,-72,57,82,48,-19,60,32,15,75,121,35,-50,17,-38,5,81,22,-42,126,-40,-31,66,-13,-99,38,-83,-126,0,-
    42,125,50,-39,-73,4,-86,73,22,-29,-22,-8,51,77,-45,63,115,-106,89,-126,-67,96,-97,-105,-104,52,-7,126,-110,63,-
    12,17,14,25,97,122,-57,-88,-49,108,63,-17,62,-1,94,91,-93,-99,79,84,-33,2,-106,-74,-72,30,-107,1,-13,-102,-107,107,-
    42,-21,57,63,-48,74,-76,-85,-34,-62,37,7,-57,6,93,13,-90,90,42,23,90,105,-39,-2,116,121,-18,-96,27,-37,-9,-128,-4,83,-
    70,22,51,-47,-79,-69,28,9,-39,-61,-91,-21,-113,-82,-124,90,-99,-12,-10,71,-45,-34,48,41,-25,-53,23,-77,-67,-
    101,124,75,16,86,-19,-89,97,-23,45,112,-106,96,53,-15,-5,-10,31,31,11,-70,-61,-125,-90,19,61,87,-118,93,-
    53,122,59,56,122,93,111,-6,88,64,-39,89,-89,99,-20,-106,-29,50,-52,-79,57,-17,-23,3,82,89,-42,-65,-96,-16</Private_key>
  <Signature>49,40,-127,75,117,63,-88,115,55,-54,-83,76,90,46,-36,-113,48,-39,104,-84,101,-105,71,121,54,-101,-74,46,-91,-
    53,124,-92,97,43,-61,-117,107,3,18,70,-103,89,-32,-112,-11,-77,-34,75,74,0,-48,-108,-75,-86,52,110,62,-66,96,-
    33,0,56,-20,-109,-47,112,-57,-48,51,-71,-8,-45,-15,61,-79,30,16,19,-21,-67,-16,120,-61,-74,49,-31,68,59,35,-47,44,15,-
    99,63,-24,-93,-71,81,-3,28,14,-4,109,47,32,73,61,-46,-108,60,58,-95,-38,20,50,-23,-37,32,102,82,-115,12,-9,100,-
    60,112,-120,-105</Signature>
  <CA_public_key>48,-127,-97,48,13,6,9,42,-122,72,-122,-9,13,1,1,1,5,0,3,-127,-115,0,48,-127,-119,2,-127,-127,0,-
    89,91,104,117,-46,-113,-30,77,-79,54,-33,44,117,25,9,29,-2,56,-111,104,122,22,11,-5,-16,-16,92,80,-77,-82,-104,65,-
    109,-31,65,84,67,55,-42,73,18,123,94,79,-121,12,-119,34,-30,108,49,-17,32,20,-76,-74,-86,28,-80,14,41,68,-104,26,34,-
    38,-109,59,30,-92,-106,1,93,22,93,56,-83,113,-124,-37,86,117,-124,8,72,-40,-69,105,71,64,-124,19,71,51,95,-87,-4,-71,-
    116,-23,110,-4,34,57,109,-55,-94,-99,96,-98,116,-121,-54,76,29,-63,-108,-101,-119,-103,-11,2,89,119,-110,76,-37,-
    125,2,3,1,0,1</CA_public_key>
</Spider_certificate>
```

Why is RSA secure?

In order to understand why RSA is secure we need to explain the mathematics that it relies upon.

To produce the keys that are used for encryption the following procedure needs to be performed:

Generate two large random primes, p and q , of approximately equal size such that their product $n = pq$ is of the required bit length, e.g. 1024 bits.

1. Compute $n = pq$ and $\phi = (p-1)(q-1)$.
2. Choose an integer e , $1 < e < \phi$, such that $\gcd(e, \phi) = 1$. (\gcd = greatest common divisor)
3. Compute the secret exponent d , $1 < d < \phi$, such that $ed \equiv 1 \pmod{\phi}$.
4. The public key is (n, e) and the private key is (n, d) . The values of p , q , and ϕ should also be kept secret.

- n is known as the *modulus*.
- e is known as the *public exponent* or *encryption exponent*.
- d is known as the *secret exponent* or *decryption exponent*.

When this is done user A can encrypt a message to B by using his public key in the following manner:

1. Obtain the recipient B's public key (n, e) .
2. Represent the plaintext message as a positive integer m .
3. Compute the cipher text $c = m^e \pmod{n}$.
4. Send the cipher text c to B.

When B has received the message he can decrypt it using his private key in the following way:

1. B uses his private key (n, d) to compute $m = c^d \pmod{n}$.
2. Then B extracts the plaintext from the integer representative m .

To crack this cryptosystem someone needs to find p and q in order to compute ϕ and then ultimately calculate d . Since n is given in the public key the attacker first needs to factorize it into the two prime numbers it was created from (p and q). However if they are sufficiently big this is not as trivial as it may seem. In fact it will take millions of years for a single computer to decrypt an encrypted message using 1024 bits key [12].

8 Appendix B (licences)

8.1 The Sun Project JXTA(TM) Software License (Based on the Apache Software License Version 1.1)

Copyright (c) 2001-2004 Sun Microsystems, Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by Sun Microsystems, Inc. for JXTA(TM) technology." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.
4. The names "Sun", "Sun Microsystems, Inc.", "JXTA" and "Project JXTA" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact Project JXTA at <http://www.jxta.org>.
5. Products derived from this software may not be called "JXTA", nor may "JXTA" appear in their name, without prior written permission of Sun.

THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL SUN MICROSYSTEMS OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

JXTA is a registered trademark of Sun Microsystems, Inc. in the United States and other countries.

8.2 License bouncy castle

Copyright (c) 2000 - 2004 The Legion Of The Bouncy Castle (<http://www.bouncycastle.org>)

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

8.3 Skin Look And Feel 1.2.10 License

```
/* =====
 *
 * Skin Look And Feel 1.2.10 License.
 *
 * Copyright (c) 2000-2004 L2FProd.com. All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 * notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 * notice, this list of conditions and the following disclaimer in
 * the documentation and/or other materials provided with the
 * distribution.
 *
 * 3. The end-user documentation included with the redistribution, if
 * any, must include the following acknowledgement:
 * "This product includes software developed by L2FProd.com
 * (http://www.L2FProd.com/)."info@L2FProd.com.
 *
 * 5. Products derived from this software may not be called "SkinLF"
 * nor may "SkinLF" appear in their names without prior written
 * permission of L2FProd.com.
 *
 * THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED
 * WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
 * OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
```

```

* DISCLAIMED. IN NO EVENT SHALL L2FPROD.COM OR ITS CONTRIBUTORS BE
* LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
* CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
* SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
* BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
* WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
* OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
* EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
* =====
*/

```

8.4 License of Log4j

```

/*
* =====
*                               The Apache Software License, Version 1.1
* =====
*
*   Copyright (C) 1999 The Apache Software Foundation. All rights reserved.
*
*   Redistribution and use in source and binary forms, with or without modifica-
*   tion, are permitted provided that the following conditions are met:
*
*   1. Redistributions of source code must retain the above copyright notice,
*      this list of conditions and the following disclaimer.
*
*   2. Redistributions in binary form must reproduce the above copyright notice,
*      this list of conditions and the following disclaimer in the documentation
*      and/or other materials provided with the distribution.
*
*   3. The end-user documentation included with the redistribution, if any, must
*      include the following acknowledgment: "This product includes software
*      developed by the Apache Software Foundation (http://www.apache.org/)."
*      Alternately, this acknowledgment may appear in the software itself, if
*      and wherever such third-party acknowledgments normally appear.
*
*   4. The names "log4j" and "Apache Software Foundation" must not be used to
*      endorse or promote products derived from this software without prior
*      written permission. For written permission, please contact
*      apache@apache.org.
*
*   5. Products derived from this software may not be called "Apache", nor may
*      "Apache" appear in their name, without prior written permission of the
*      Apache Software Foundation.
*
*   THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES,
*   INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
*   FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
*   APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
*   INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLU-
*   DING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
*   OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
*   ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
*   (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
*   THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*
*   This software consists of voluntary contributions made by many individuals
*   on behalf of the Apache Software Foundation. For more information on the
*   Apache Software Foundation, please see <http://www.apache.org/>.
*/

```

8.5 Licence nanoXML

This is the license for nanoXML,
used by Skin Look And Feel for Theme Pack definitions

nanoXML is available at <http://nanoXML.sourceforge.net>

\$Id: LICENSE_nanoxml,v 1.2 2003/08/01 19:34:35 l2fprod Exp \$

Copyright (C) 2000 Marc De Scheemaecker, All Rights Reserved.

This software is provided 'as-is', without any express or implied warranty.
In
no event will the authors be held liable for any damages arising from the
use
of this software.

Permission is granted to anyone to use this software for any purpose,
including
commercial applications, and to alter it and redistribute it freely,
subject to
the following restrictions:

1. The origin of this software must not be misrepresented; you must not
claim
that you wrote the original software. If you use this software in a
product, an acknowledgment in the product documentation would be
appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not
be
misrepresented as being the original software.
3. This notice may not be removed or altered from any source
distribution.

8.6 Licence and conditions of use for SAXON package

Saxon

The contents of the downloaded file (saxon.zip) are subject to the Mozilla Public License
Version 1.0 (the "License"); you may not use this file except in compliance with the License.

You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT
WARRANTY OF ANY KIND, either express or implied. See the License for the specific
language governing rights and limitations under the License.

The Original Code of Saxon comprises all those components which are not explicitly
attributed to other parties.

The Initial Developer of the Original Code was Michael Kay of International Computers
Limited. Individual modules identified as being created by other individuals include separate
IPR notices. Other contributors are acknowledged individually in comments attached to the
relevant code modules. All Rights Reserved.

*Since February 2001 I (Michael Kay) have been an employee of Software AG, and I continue
to develop Saxon with the support and sponsorship of that company. Technically, therefore,
the "Initial Developer" no longer exists (the form of words was deliberately chosen to reflect*

the joint involvement of myself and ICL), so I now work as a "Contributor". International Computers Limited has since been absorbed into Fujitsu Limited, and trades as Fujitsu Services.

If you produce a product that includes or requires Saxon, please refer to it as "The Saxon XSLT Processor from Michael Kay", and include the URL of the home page, which is at <http://saxon.sourceforge.net/>.

There is no guarantee of technical support, though I am usually able to answer enquiries within a few days. Please subscribe to the mailing list available at <http://lists.sourceforge.net/lists/listinfo/saxon-help> and raise any enquiries there. Also check the Saxon project pages on source forge for details of known errors; all bugs are listed there as soon as I have sufficient evidence to describe the nature of the problem.

Ælfred

Earlier versions of the Saxon distribution included a modified version of the Ælfred XML parser from Microstar. Since Saxon now uses JDK 1.4, which includes an XML parser, there is no longer any need to supply an XML parser with Saxon.

JAXP

Earlier versions of the Saxon distribution included a copy of the JAXP 1.1 library (Java API for XML). Since this is now supplied as part of JDK 1.4, it is no longer distributed with Saxon.

Other components

Saxon includes the `org.apache.xerces.util.XMLChar` module from the Apache Xerces product, renamed as `net.sf.saxon.om.XMLChar` but otherwise unchanged. See the copyright statements included in the source code of this module for conditions of use.

Michael H. Kay
5 August 2003

Mozilla

MOZILLA PUBLIC LICENSE Version 1.0

1. Definitions.

1.1. ``Contributor'' means each entity that creates or contributes to the creation of Modifications.

1.2. ``Contributor Version'' means the combination of the Original Code, prior Modifications used by a Contributor, and the Modifications made by that particular Contributor.

1.3. ``Covered Code'' means the Original Code or Modifications or the combination of the Original Code and Modifications, in each case including portions thereof.

1.4. ``Electronic Distribution Mechanism'' means a mechanism generally accepted in the software development community for the electronic transfer of data.

1.5. ``Executable'' means Covered Code in any form other than Source Code.

1.6. ``Initial Developer'' means the individual or entity identified as the Initial Developer in the Source Code notice required by **Exhibit A**.

1.7. ``Larger Work'' means a work which combines Covered Code or portions thereof with code not governed by the terms of this License.

1.8. ``License'' means this document.

1.9. ``Modifications'' means any addition to or deletion from the substance or structure of either the Original Code or any previous Modifications. When Covered Code is released as a series of files, a Modification is:

A. Any addition to or deletion from the contents of a file containing Original Code or previous Modifications.

B. Any new file that contains any part of the Original Code or previous Modifications.

1.10. ``Original Code'' means Source Code of computer software code which is described in the Source Code notice required by **Exhibit A** as Original Code, and which, at the time of its release under this License is not already Covered Code governed by this License.

1.11. ``Source Code'' means the preferred form of the Covered Code for making modifications to it, including all modules it contains, plus any associated interface definition files, scripts used to control compilation and installation of an Executable, or a list of source code differential comparisons against either the Original Code or another well known, available Covered Code of the Contributor's choice. The Source Code can be in a compressed or archival form, provided the appropriate decompression or de-archiving software is widely available for no charge.

1.12. ``You'' means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License or a future version of this License issued under Section 6.1. For legal entities, ``You" includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, ``control" means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of fifty percent (50%) or more of the outstanding shares or beneficial ownership of such entity.

2. Source Code License.

2.1. The Initial Developer Grant.

The Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license, subject to third party intellectual property claims:

(a) to use, reproduce, modify, display, perform, sublicense and distribute the Original Code (or portions thereof) with or without Modifications, or as part of a Larger Work; and

(b) under patents now or hereafter owned or controlled by Initial Developer, to make, have made, use and sell ("Utilize") the Original Code (or portions thereof), but solely to the extent that any such patent is reasonably necessary to enable You to Utilize the Original Code (or portions thereof) and not to any greater extent that may be necessary to Utilize further Modifications or combinations.

2.2. Contributor Grant.

Each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license, subject to third party intellectual property claims:

(a) to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof) either on an unmodified basis, with other Modifications, as Covered Code or as part of a Larger Work; and

(b) under patents now or hereafter owned or controlled by Contributor, to Utilize the Contributor Version (or portions thereof), but solely to the extent that any such patent is reasonably necessary to enable You to Utilize the Contributor Version (or portions thereof), and not to any greater extent that may be necessary to Utilize further Modifications or combinations.

3. Distribution Obligations.

3.1. Application of License.

The Modifications which You create or to which You contribute are governed by the terms of this License, including without limitation Section 2.2. The Source Code version of Covered Code may be distributed only under the terms of this License or a future version of this License released under Section 6.1, and You must include a copy of this License with every copy of the Source Code You distribute. You may not offer or impose any terms on any Source Code version that alters or restricts the applicable version of this License or the recipients' rights hereunder. However, You may include an additional document offering the additional rights described in Section 3.5.

3.2. Availability of Source Code.

Any Modification which You create or to which You contribute must be made available in Source Code form under the terms of this License either on the same media as an Executable version or via an accepted Electronic Distribution Mechanism to anyone to whom you made an Executable version available; and if made available via Electronic Distribution Mechanism, must remain available for at least twelve (12) months after the date it initially became available, or at least six (6) months after a subsequent version of that particular Modification has been made available to such recipients. You are responsible for ensuring that the Source Code version remains available even if the Electronic Distribution Mechanism is maintained by a third party.

3.3. Description of Modifications.

You must cause all Covered Code to which you contribute to contain a file documenting the changes You made to create that Covered Code and the date of any change. You must include a prominent statement that the Modification is derived, directly or indirectly, from Original Code provided by the Initial Developer and including the name of the Initial Developer in (a) the Source Code, and (b) in any

notice in an Executable version or related documentation in which You describe the origin or ownership of the Covered Code.

3.4. Intellectual Property Matters

(a) Third Party Claims.

If You have knowledge that a party claims an intellectual property right in particular functionality or code (or its utilization under this License), you must include a text file with the source code distribution titled ``LEGAL" which describes the claim and the party making the claim in sufficient detail that a recipient will know whom to contact. If you obtain such knowledge after You make Your Modification available as described in Section **3.2**, You shall promptly modify the LEGAL file in all copies You make available thereafter and shall take other steps (such as notifying appropriate mailing lists or newsgroups) reasonably calculated to inform those who received the Covered Code that new knowledge has been obtained.

(b) Contributor APIs.

If Your Modification is an application programming interface and You own or control patents which are reasonably necessary to implement that API, you must also include this information in the LEGAL file.

3.5. Required Notices.

You must duplicate the notice in **Exhibit A** in each file of the Source Code, and this License in any documentation for the Source Code, where You describe recipients' rights relating to Covered Code. If You created one or more Modification(s), You may add your name as a Contributor to the notice described in **Exhibit A**. If it is not possible to put such notice in a particular Source Code file due to its structure, then you must include such notice in a location (such as a relevant directory file) where a user would be likely to look for such a notice. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Code. However, You may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear that any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

3.6. Distribution of Executable Versions.

You may distribute Covered Code in Executable form only if the requirements of Section **3.1-3.5** have been met for that Covered Code, and if You include a notice stating that the Source Code version of the Covered Code is available under the terms of this License, including a description of how and where You have fulfilled the obligations of Section **3.2**. The notice must be conspicuously included in any notice in an Executable version, related documentation or collateral in which You describe recipients' rights relating to the Covered Code. You may distribute the Executable version of Covered Code under a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable version does not attempt to limit or alter the recipient's rights in the Source Code version from the rights set forth in this

License. If You distribute the Executable version under a different license You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or any Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

3.7. Larger Works.

You may create a Larger Work by combining Covered Code with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Code.

4. Inability to Comply Due to Statute or Regulation.

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Code due to statute or regulation then You must: (a) comply with the terms of this License to the maximum extent possible; and (b) describe the limitations and the code they affect. Such description must be included in the LEGAL file described in Section 3.4 and must be included with all distributions of the Source Code. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

5. Application of this License.

This License applies to code to which the Initial Developer has attached the notice in **Exhibit A**, and to related Covered Code.

6. Versions of the License.

6.1. New Versions.

Netscape Communications Corporation ("Netscape") may publish revised and/or new versions of the License from time to time. Each version will be given a distinguishing version number.

6.2. Effect of New Versions.

Once Covered Code has been published under a particular version of the License, You may always continue to use it under the terms of that version. You may also choose to use such Covered Code under the terms of any subsequent version of the License published by Netscape. No one other than Netscape has the right to modify the terms applicable to Covered Code created under this License.

6.3. Derivative Works.

If you create or use a modified version of this License (which you may only do in order to apply it to code which is not already Covered Code governed by this License), you must (a) rename Your license so that the phrases "Mozilla", "MOZILLAPL", "MOZPL", "Netscape", "NPL" or any confusingly similar phrase do not appear anywhere in your license and (b) otherwise make it clear that your version of the license contains terms which differ from the Mozilla Public License and Netscape Public License. (Filling in the name of the Initial Developer, Original Code or Contributor in the notice described in **Exhibit A** shall not of themselves be deemed to be modifications of this License.)

7. DISCLAIMER OF WARRANTY.

COVERED CODE IS PROVIDED UNDER THIS LICENSE ON AN ``AS IS" BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED CODE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED CODE IS WITH YOU. SHOULD ANY COVERED CODE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED CODE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

8. TERMINATION.

This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. All sublicenses to the Covered Code which are properly granted shall survive any termination of this License. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

9. LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED CODE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO YOU OR ANY OTHER PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTY'S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THAT EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

10. U.S. GOVERNMENT END USERS.

The Covered Code is a ``commercial item," as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of ``commercial computer software" and ``commercial computer software documentation," as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Code with only those rights set forth herein.

11. MISCELLANEOUS.

This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by California law provisions (except to the extent applicable law, if any,

provides otherwise), excluding its conflict-of-law provisions. With respect to disputes in which at least one party is a citizen of, or an entity chartered or registered to do business in, the United States of America: (a) unless otherwise agreed in writing, all disputes relating to this License (excepting any dispute relating to intellectual property rights) shall be subject to final and binding arbitration, with the losing party paying all costs of arbitration; (b) any arbitration relating to this Agreement shall be held in Santa Clara County, California, under the auspices of JAMS/EndDispute; and (c) any litigation relating to this Agreement shall be subject to the jurisdiction of the Federal Courts of the Northern District of California, with venue lying in Santa Clara County, California, with the losing party responsible for costs, including without limitation, court costs and reasonable attorneys fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License.

12. RESPONSIBILITY FOR CLAIMS.

Except in cases where another Contributor has failed to comply with Section 3.4, You are responsible for damages arising, directly or indirectly, out of Your utilization of rights under this License, based on the number of copies of Covered Code you made available, the revenues you received from utilizing such rights, and other relevant factors. You agree to work with affected parties to distribute responsibility on an equitable basis.

EXHIBIT A.

``The contents of this file are subject to the Mozilla Public License Version 1.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is _____.

The Initial Developer of the Original Code is _____. Portions created by _____ are Copyright (C) _____
_____. All Rights Reserved.

Contributor(s): _____."

9 *Appendix C*

The code for all the project.