Persona Concept:
System Architecture and Prototypes

By
MAHESH SUBRAMANIUM

Course: CS 506 Project

Major Professor: Dr. Kal Toth
School of Electrical Engineering and Computer Science
Oregon State University
April 5, 2012

Abstract

This project explores a security framework, called the Persona Concept, aimed at giving the user greater control over their private data in cyberspace, in particular, their electronic credentials. The background for the Persona Concept, along with the requirements identified can be found in three (3) technical publications produced by Kal Toth and myself in 2003 (see list of references [1], [2] and [3]).

The Persona Concept uses a Persona Server as the secure storehouse for persona data such as credit card and bank account identifiers as well electronic credentials such as X.509 certificates. The data is secured using public key encryption. The server component interacts with the user via a Persona Client which obtains X-509 Certificates on behalf of the user from a Certification Authority (CA) service. Interaction among the various components (Persona Server, Persona Client and Certification Authority) is enabled by the use of Web Services.

This project focused on managing the user data across these components including the creation, secure storage and retrieval of Persona data to and from the Persona Server using Simple Object Application Protocol (SOAP) and under the control of public key encryption schemes. SOAP is an open XML-based application-level Internet protocol for implementing distributed access to object-oriented services deployed on the Web. SOAP is designed to allow applications running on a range of platforms to access remote objects and applications on servers across a TCP/IP network. In addition to this, the project built upon the work by Ike Chen (OSU M.S. graduate project in 2004) [4] by enabling X-509 Certificate file transfer between the client and Certification Authority using SOAP (improvements were made to the mechanisms for transferring files, particularly in support of X.509 certificates). A Java-based prototype client called the Persona Client (PC) was created and used in conjunction with a small local password-based security scheme to protect the private key data on the user device.

My project first implemented the upgrade to the Certification Authority services and detailed the various policies surrounding storage of data. I then implemented the Persona Client which enabled me to test the persona services in place. I then implemented the encryption schemes and the server components enabling the secure storage and transfer of Persona Data. Components to create, edit and delete data were packaged and published as services to be used by the Persona Client.

Future extensions of this project could include exploring custom credential generation based on Security Assertion Markup Language (SAML) and Extensible Access Control Markup Language (XACML) which together can be used to specify XML-based credentials and access control policies for an enterprise. Other extensions include the implementation of discretionary and mandatory security policies using XACML to allow users to conduct transactions across various domains.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Motivation behind the project

Currently, an individual's private and personal consumer data is scattered across government, company and web service provider databases and applications. Opportunities for identity theft and misuse of privacy information are increasing exponentially due to the globalization of e-business and the advent of wireless and other information access technologies [5], [6]. In addition to this, implementations of authentication systems, which are meant to serve both users and web service providers (WSPs), have not always respected the privacy needs and rights of the individual consumer. The dangers of hacking, viruses and mistakes make even well known web service providers less trustworthy. We find the exposure in direct correlation to the number of web service providers a consumer uses.

As part of this project, we investigated some of the existing and emerging security standards, technologies and architectures. The aim was to establish a sound understanding of their capabilities and shortcomings as part of a process to discover mechanisms and protocols that might be adapted to support the Persona concept. The following sub-sections summarize the findings to date with respect to some of these technologies.

1.1.1 Authentication mechanisms

The existence of a wide variety of authentication systems with their own user IDs and passwords led to the development of "Single Sign-On" (SSO) systems. These systems eliminated the need for different passwords and IDs and simplified web access by requiring only a single user ID and password when a user attempts to access many or all her web services. The SSO system intermediates between the user and each WSP accessed by the user. It does so by authenticating the user and sharing the user's ID, credentials and other personal data (e.g. the user's credit card) with the WSPs as required.

More recently, SSO (Single Sign-On) solutions like Microsoft Passport [7] have been attempting to address some of the problems. In addition, a consortium of companies has come together to develop SAML (Secure Assertion Markup Language) [8] to support an open interoperability standard.

SSO is meant to simplify web access by requiring only a single user ID and password. Kerberos is perhaps the first SSO model. Microsoft Passport and the Liberty Alliance Project [9] are highly popular SSO initiatives. Instead of requiring users to establish separate identities and authentication mechanisms for every web service they may use, users register with an SSO authentication server using a single user ID and password.

1.1.1.1 Microsoft Passport

Microsoft Passport makes a shared authentication service available to affiliate web service providers (WSPs). When a user begins to execute sensitive business transactions with the WSP, the WSP re-directs to MS Passport's "Authentication Server". The Authentication Server authenticates the user and then re-directs back to the WSP - indicating authentication success or failure. This approach maintains a single point of authentication and relies on trusted connections between WSP affiliates and the Authentication Server.

Although this approach simplifies password management for the user, there is lot of arenas for certain types of misuse and abuse and indeed there are published experiments pointing to this [10], [11]. For example:

- Maintaining a very large number of user accounts and passwords by a single company on a relatively small number of Authentication Servers makes for a single point of attack and exposes large volumes of private date to hackers and insider threats within Microsoft;
- Connections between the WSP and the Authentication Server are secured by a symmetric key encryption strategy. Those who have investigated Passport contend that this does not provide strong enough security. Frequent rotation of keys to enhance security would be too cumbersome and would not scale up for a large number of participating WSPs;
- Unattended login sessions can be hijacked by unauthorized lurkers;
- The amount and nature of sensitive data that could be left in cached areas like downloaded pages and cookies makes public Internet café usage a particularly risky prospect.

Just the sheer volume of fixes and sizeable code changes and version support and upgrade would make fixing these problems from the browser end cumbersome.

1.1.1.2 Liberty Alliance

Liberty Alliance Project was developed by a consortium of companies varying from Airliners to Software firms and it seems to not have some of the shortcomings of MS Passport – in particular, it avoids the single caretaker model and consequently removes the single point of attack.

Liberty Alliance has been formulated on the concept of collaborating domains associated with each other through "trust circles". "Identity Providers" manage authentication data within their own domains, collaborate with each other through secure channels, and trust each others' authentication of identities.

Although, Liberty Alliance model removes a single point of attack, the mechanisms for trusted interoperability between identity servers is not yet well defined. The unattended browser sessions and caching vulnerabilities are also essentially the same as for MS Passport.

1.1.1.3 Public Key Infrastructure

Public Key Infrastructure (PKI) as implemented by Asymmetric key (a.k.a. "public key") technologies has made large positive contributions to security over the web. Most notably, Secure Sockets Layer (SSL) [12] has become a critical mechanism for providing levels of trust that have enabled web-banking and on-line credit-card processing over the last few years.

PKI, and the SSL as implemented by most web browsers, is designed to support 2-way authentication. The required knowledge and relative complexity for a user to obtain a digital certificate has resulted in most web sites supporting only 1-way authentication.

We therefore decided to explore possible strategies whereby the Persona Concept could be leveraged to enable the pros of above-mentioned technologies and combine it with the strength of 2-way SSL while maintaining a relatively simple, easy-to-manage interface for the user.

2 OVERVIEW OF THE PERSONA CONCEPT

"Persona" has been used to represent a user's profile in cyber space. It has also been used to denote a software agent or assistant interacting with the user to provide advice while brokering web transactions. In our usage, Persona denotes the entire range of a user's identity, authentication data, credentials, and electronic wallet contents like credit cards, bank account numbers and other personal information, which could potentially be used in a networked environment. Persona and Persona data denote the same thing and in this report, you can find them used interchangeably. The Persona Concept explores new concepts

and strategies aimed at better serving the privacy and authentication needs of the consumer. The Persona Concept aims to offer better control and accountability over the web user's Persona.

The Persona can be likened to an identity card. Potentially, the Persona could be implemented on a smart card, in a handheld device, in a protected file maintained on a desktop computer, on a diskette, or on a controlled web server. The Persona Concept tries to overcome the shortcomings of current authentication schemes. It is designed with ease of deployment in mind. The successful implementation of the Persona Concept involves the interaction between the Persona Client (PC), which acts as the interface to the user, the Persona Server (PS) which acts as the secure store-house of Persona data and the Certification Authority (CA) which supplies authentication certificates to requesting clients.

We consider the persona to be an active software agent that encapsulates private and personal data and performs a range of authentication and personalization services on behalf of its owner. Most broadly, the persona carries out tasks ranging from user authentication and provisioning of credentials to Web Service Providers (WSPs), through to personalization and customization activities such as channel selection, user interface adaptation, auto web form filling and automated searching and bidding. For the most part, the Persona Concept project has restricted discussions and investigations to authentication, credential handling and accountability mechanisms for the most part. Fig. 1 illustrates the persona architecture. [3]

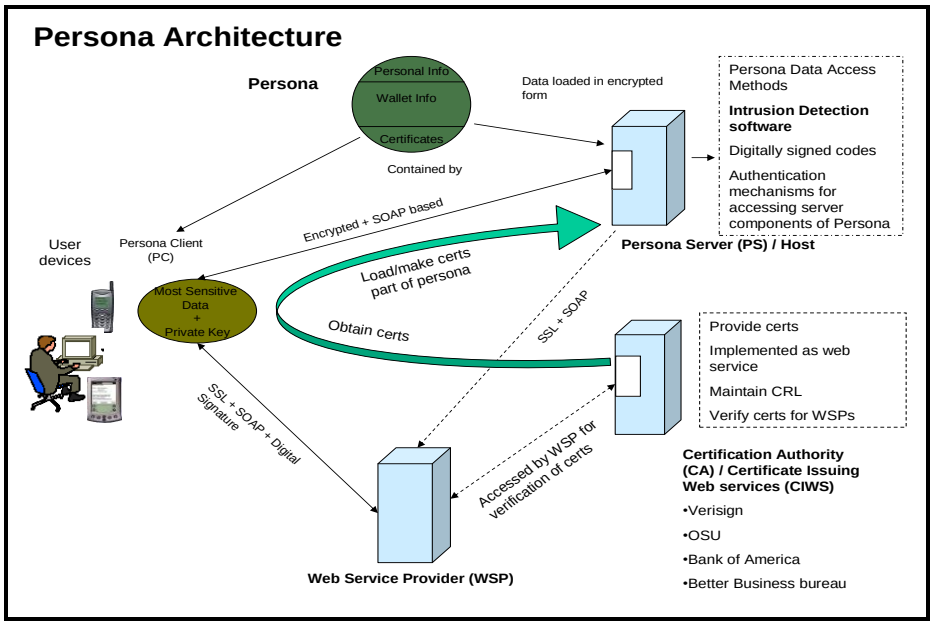


Figure 1: Persona Conceptual Framework

3 PROJECT OVERVIEW

The following list summarizes the work conducted to achieve the results documented in this report:

Over the period of this project, I investigated both commercial and identity management technologies like Microsoft Passport, Liberty Alliance and Public Key Infrastructure. This enabled me to identify and synthesize various ideas that could feasibly be supported by the Persona Concept. I investigated the area of Web Services and SOAP As a member of the Smart Agents Research Group in Oregon State University; I configured the systems and workstations to support the project. I, along with Ike Chen (now an MS Graduate from Oregon State University) and Richard Wu (from the University of Alberta, Canada) started to research the various areas of Persona Concept under the guidance of Dr. Kal Toth. Ike concentrated on building the Certification Authority in support of the Persona Concept, Richard

concentrated on the integration of the Persona Client with the browser and I concentrated on the configuration of the Persona Server and the development of Server side components. Sidharth Anand (now an MS graduate from Oregon State University) handled the area of generation and usage of custom credentials using Secure Assertion Markup Language (SAML) [13].

During the next phase of my work I upgraded the Web Service based Certification Authority to support the transfer of X-509 files. I also identified the various policies surrounding the storage and usage of data, the usage scenarios, and the encryption schemes to be employed to store Persona data etc. I implemented a Persona Client (PC), which enabled me to test various server-side Web Services that were implemented to support the Persona Concept.

In the next phase, I began implementing the screens and forms for user interaction and the routines supporting the encryption of data. I used the security providers from Bouncy Castle, an open source solution, for my encryption purposes. I then experimented with Web Services and SOAP to study open XML-based application-level Internet protocols designed to allow applications to access remote objects. This eventually enabled me to build the server-side components to securely store, retrieve and modify Persona data.

During the final period of my project, I ran a number of demonstration scenarios and documented the results in this project report.

4 PERSONA CONCEPT REQUIREMENTS

4.1 Overview

This document addresses the various requirements for this project, which will involve creating a Persona Client and Persona Server web services that can be called from Persona clients and web service providers.

4.2 Conceptual Requirements for the Persona Concept

The Persona Concept is modeled in object-oriented terms and the key use cases are identified. We will begin with an overview of the requirements as we see them. Mandatory and desirable requirements are distinguished with “shall” and “should” respectively, and the requirements are listed below.

4.2.1 Persona Data

The Persona shall contain data items such as:

- Identity: owner’s legal name, aliases etc.
- Authentication Data: Passwords, Certificates, Credentials
- Profile Data: contact info, e-mail, fax # etc.
- E-Wallet Information: credit card #, debit card #, bank accounts
- Other: favorites, air mile accounts, and memberships.

In the implementation of this project, I have captured few of the data items mentioned above as a way to illustrate the usage.

4.2.2 Identity, Authentication and Authorizations

A given Persona instance for a user should contain a unique and widely accepted identifier associated with the persona owner – normally the full legal name. Private authentication code(s) bound to the persona identifier, such as passwords or private keys, shall be supported. The Persona shall also be

capable of supporting the authentication of web services by obtaining and maintaining related authentication data such as digital certificates.

4.2.3 Multiple Device Support

The persona owner shall be free to transact with any WSP on the web. The user shall be able to employ PCs, laptops, cell phones and/or PDAs at home, at work or at some other location such as an Internet café.

These devices may or may not be capable of supporting all persona data and capabilities. Furthermore, there may be scenarios where persona data may not be stored on the device being used because of security and capacity reasons. The system is to be designed under the assumption that the user device shall be able to hold only the private key and a bare minimum of libraries to support the Persona Client.

4.3 Derived Set of Requirements for the Persona Concept

We arrived at the following set of derived requirements after analyzing the conceptual requirements:

- The Persona is to be implemented such that it can be embedded in a physical device, such as a smart card, SIM or USB drive – or more practically and at the bare minimum, the private key can be stored in the physical device in the users possession;
- A Single Sign-On (SSO) effect is to be implemented for the user of the Persona using a secret password or password-phrase;
- A Persona is to be able to contain electronic credentials (potentially specified in SAML).
- To support 2-way SSL, the Persona should be able to contain X.509 digital certificates and corresponding private keys.
- A Credential Issuing Authority (or CA for short) is required to issue and digitally sign X.509 certificates.

4.4 Operational Requirements for the Persona Concept

The operational requirements can be expressed in terms of use cases where the principle actors are the Persona owner acting through the Persona Client, the Persona Server and the Certification Authority. We are ignoring administrators and other possible actors at this stage.

We are also assuming that the system design will need to incorporate multiple persona copies and/or fragments to support availability and reliability requirements, off-line transactions, and transactions from cell phones and PDA's.

The following list identifies (only) the key use cases with brief clarifying comments:

- Create a Persona Instance: owner configures and populates the persona.
- Maintain an Instance: owner updates and re-configures the persona.
- Deploy Instances: owner deploys persona components to host(s).

5 PERSONA CONCEPT USAGE SCENARIOS

The principle "actors" supporting the Persona Concept are:

- Users and
- Administrators (web services and credential issuing).

The key use cases will be supported by two types of web services:

- Persona Server Services
- Certification Authority Services

Most services will be responding to requests from users. A Persona Agent can be modeled in object-oriented terms for specifying the Persona Concept use cases. A Persona is modeled as a class composed of encapsulated data with methods. The encapsulated data will comprise the Persona as defined towards the start of the report - they include user identities, profile data, personal preferences, electronic credentials, and other private data. Each exposed method can be modeled as a use case invoked by a user. Below you can find two principal usage scenarios for the Persona Concept.

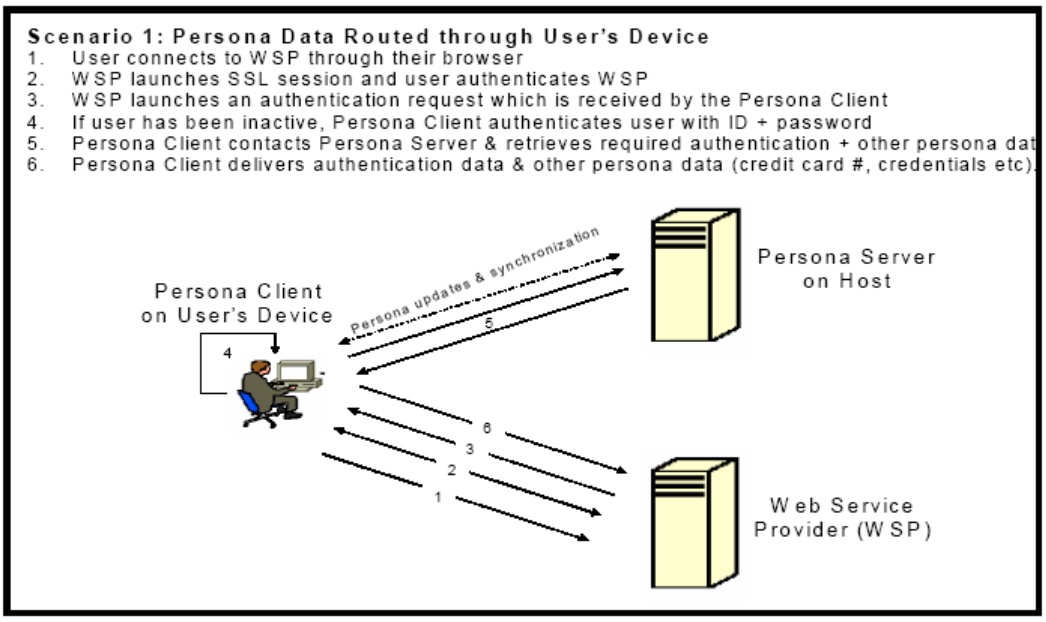


Figure 2: Persona Client interacting with a WSP with data from the Persona Server routed through the PC

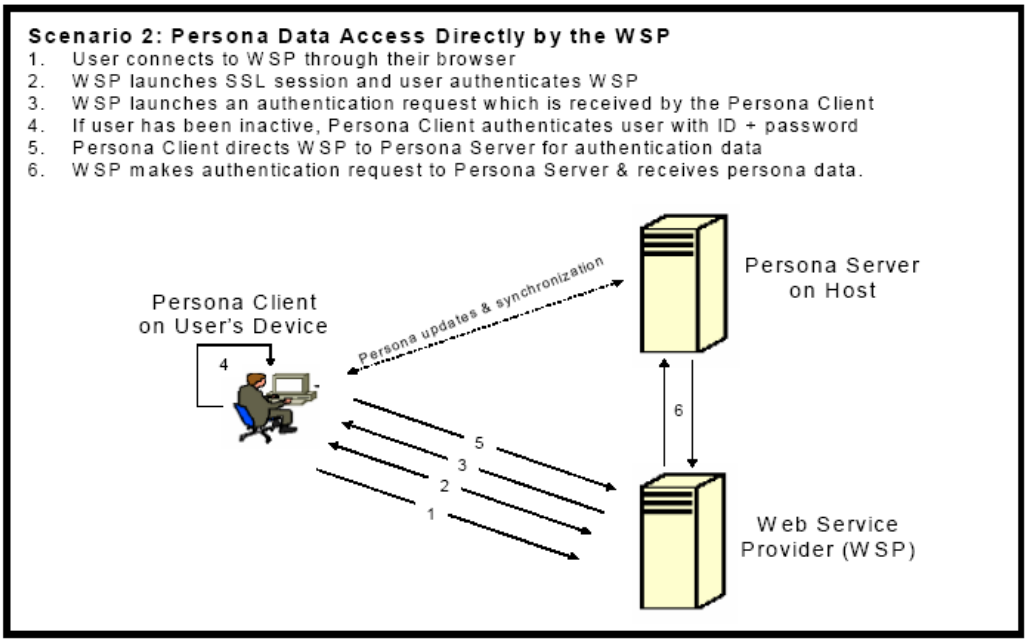


Figure 3: Persona Client interacting with a WSP with WSP getting data directly from the Persona Server

The principal use cases to be supported by the Persona Concept are listed below:

5.1 Persona Data Management

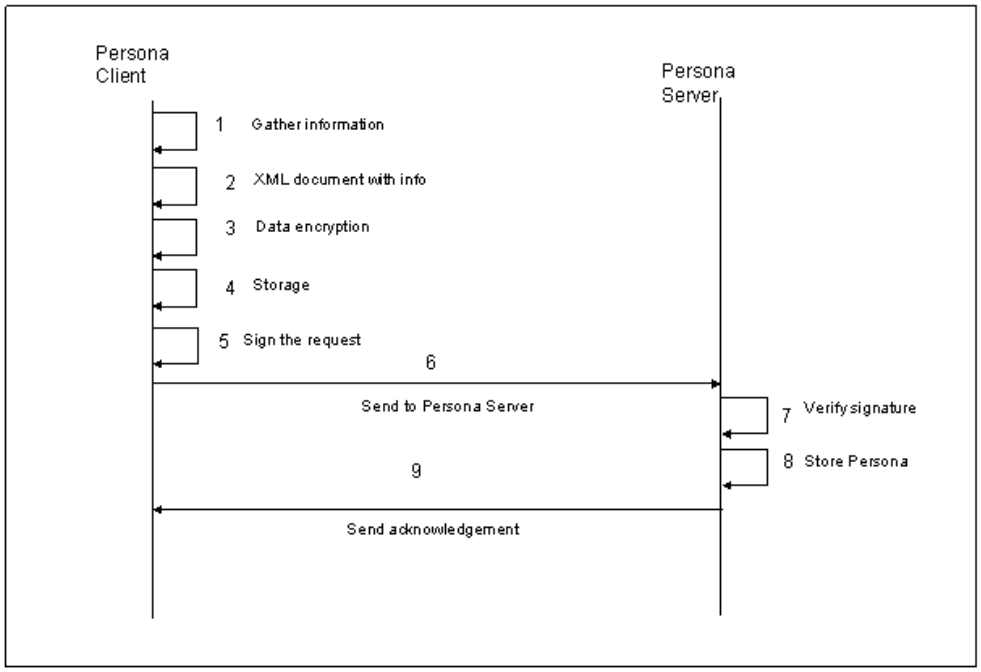
The major sequence of events surrounding the management of Persona data are given below

5.1.1 Create/Add data to the Persona

The sequence of events surrounding this process are given below

1. Persona Client gathers the required information from the user
2. Persona Client creates an XML based document with the labels being the principal fields like user name, credit card number etc.
3. Persona Client encrypts the data – default encryption scheme is Public key encryption
4. Persona Client stores the encrypted version of what the user entered for the principal fields.
5. Persona Client signs the request
6. Persona Client sends the XML file to the Persona Server along with the user name
7. Persona Server verifies the signature and finds the user Persona to associate it with
8. Once the user Persona to associate with is found, the Persona file is stored on the Persona Server
9. Persona Server sends an acknowledgement back with the results of the operation

Figure 4: Sequence diagram illustrating creating a Persona

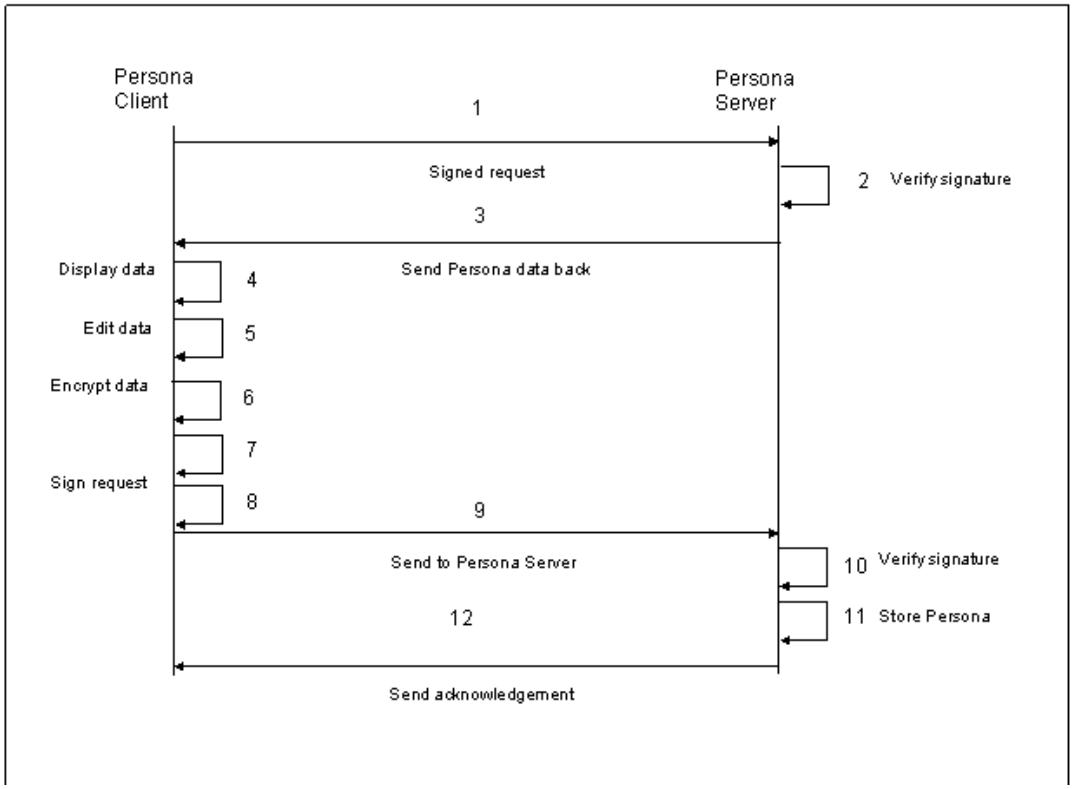


5.1.2 Modify data in the Persona

The sequence of events surrounding this process is given below

1. Persona Client sends a signed request to retrieve the Persona from the Persona Server
2. Persona Server verifies the signature
3. Upon success, Persona Server retrieves the Persona and sends it to Persona Client
4. Persona Client displays the data in the Persona to the user
5. User edits the data and clicks the button to upload the Persona
6. Persona Client encrypts the data – default encryption scheme is Public key encryption
7. Persona Client stores the encrypted version of what the user entered for the principal fields in the XML document.
8. Persona Client signs the XML file
9. Persona Client sends the XML file to the Persona Server along with the user name
10. Persona Server verifies the signature and finds the user Persona to associate it with
11. Once the user Persona to associate with is found, the Persona file is stored on the Persona Server
12. Persona Server sends an acknowledgement back with the results of the operation

Figure 5: Sequence diagram illustrating modifying Persona Data



5.2 Certificates / Credential Management

The principal use-cases under this are:

- Request for a new X509 Certificate from the CA
- Store the certificate as part of the Persona
- Store the private key password protected in a location of the user's choice

The sequence of events/actions surrounding requesting for a new X509 Certificate are illustrated below:

1. CA creates self signed certificate and also sets up certification revocation lists, folders to store signed certificates etc (only for the first time set up)
2. Persona Client captures the data to be included in the certificate
3. Persona Client Creates public and private key
4. Persona Client stores the private key securely (password protected scheme) in a location of the user's choice
5. Persona Client Creates unsigned certificate and the certificate signing request
6. Persona Client sends over the unsigned certificate to the CA server for signing (possible SSL connection)
7. CA verifies the information given in the Certificate
8. CA signs the certificate and stores the certificate in its folders
9. CA sends the signed certificate back
10. Persona Client stores the certificate in local database

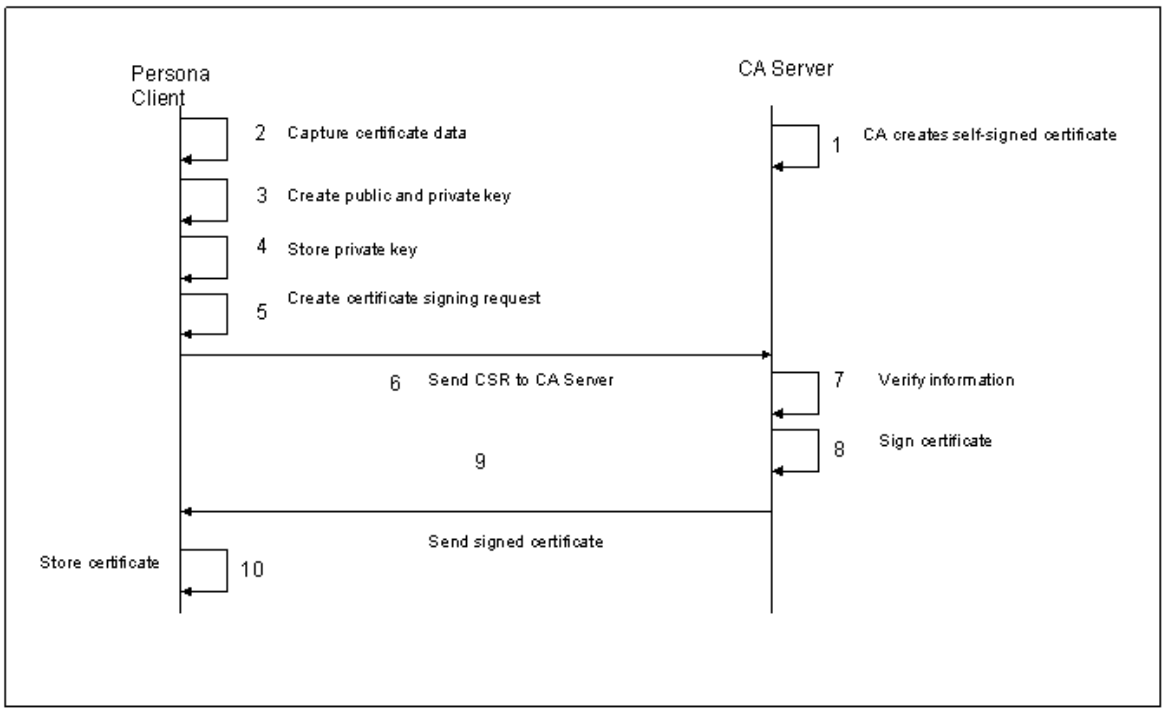


Figure 6: Persona Client requesting and obtaining certificates from CA

6 SYSTEM DESIGN MODEL

6.1 Architecture-First

Architecture-first is one of the highest priorities when applying modern software engineering principles [13]. This is because the chosen software architecture has direct influences on technical, business, and social decisions made during implementation [15]. Identifying and managing those influences earlier and throughout a software development project is critically needed to build a quality software system. Web services will be a very important part of the technical influences and impacts on how a software project should or can be managed simply because the Web services-based computing potentially brings in a lot more project stakeholders and interfaces than more heterogeneous applications.

6.2 System Engineering

To meet the project's goals we made a few important engineering choices which are highlighted in the sections that follow.

6.2.1 Process Model

The process model was informal but the methods and techniques of the Rational Unified Process (RUP) and Extreme Programming (XP) were applied wherever possible. For example, the concepts of use cases, architecture-first, and iterations were applied across requirements specifications, design, and prototyping phases.

6.2.2 Architectural Design/Evaluation Methods

Our architectural design approach was based on Kruchten (2001), which presents a 4+1 multi-view model, logical, process, physical, and development architectural views centered on use cases. However, we deviated a little to meet the development needs. First, our logical architectural view is not an object model but rather a component model of the design: each component has its corresponding interface, a service interface. Secondly, the deployment architectural view was used instead of physical architectural view. The architectural views were modeled in UML to maintain semantic and syntactic consistency. Software Architecture Analysis Method (SAAM) [16] was applied to express the Persona Concept's system architecture by following the three stages and eight steps suggested in SAAM. First, a number of non-functional requirements were developed from which a set of architectural quality attributes were derived, such as reliability and security. Then, a corresponding set of scenarios against those attributes was determined and, finally, followed by a review and evaluation of the architecture.

6.2.3 Development Platforms

All the server-side components for this project were developed to run on a Linux platform with the client-side components on the Windows 2000 platforms where Microsoft .NET Framework and Compact Framework are available.

Both open source software, such as Apache and commercial software tools like Visual Studio.NET, Metrowerks Code warrior etc were used to develop Persona Server and Client parts. All the server-side components were developed in Java. Java was used to develop client-side components.

6.2.4 Conceptual System Model

Figure 4 illustrates the Persona Concept. Users interact with Web services providers and servers hosting a Persona Server, which manages their Persona data, and a Credential Issuing Authority. The user may employ a variety of devices to get connected. The Persona Client manages the user's local data and provides an interface to connect to the services. The Client is often integrated with a Web browser context. Users have physical control over the devices. The local data can be protected with a single password.

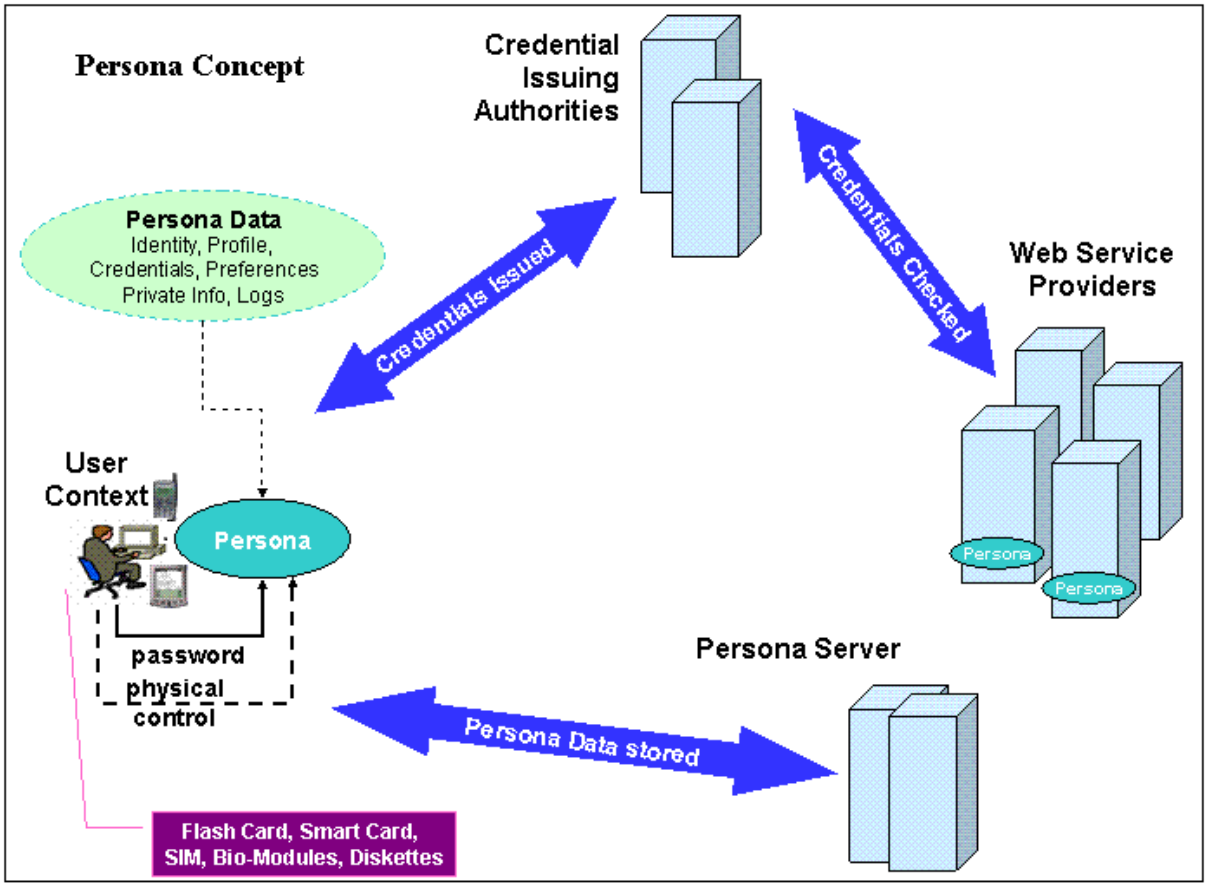


Figure 7: Persona Concept – System Model

Since the Persona Client is the conduit for connecting to the servers hosting the Persona data and credentials, the use of a single password gives the effect of a virtual Single Sign-on experience. Users use the credentials issued to them by credential issuing authorities as part of their Persona in order to gain access to various domains. The domains that the Persona is associated with have the capability to verify these credentials (depending on their security policy) with the credential issuing authorities. The Web services interface built for this purpose makes this a secure and straightforward process.

6.3 Persona System Development

The development mainly involves architectural design and prototype implementation. The architectural design attempts to address a wide range of architectural issues with a focus on their width. The prototyping work is directed to a narrower set of functional partitions of the Persona System. This strategy makes it possible to explore a broad set of architectural issues and current technologies needed in building Persona Concept within time and resource constraints.

6.3.1 Design

6.3.1.1 Architecture Overview

Architecturally, the Persona Server, Persona Client and the Certificate Issuing Authority form the core of Persona Concept. Persona Server is entrusted with the storage and protection of user data. It is deployed on a server in which the user possesses some disk space and has access to it. Potentially, the user could be

afforded a range of encryption schemes to choose from but at this stage the default mechanism is a public key encryption scheme. The keys are obtained from certificate issuing authorities. Our initial development approach assumes that the Persona Server is hosted on a trusted computing platform. Distributed Persona Clients and multiple Persona Servers for redundancy with intrusion detection mechanisms are left for potential exploration in the future.

Persona Server is mainly built with components based on J2EE and Web Services (i.e. SOAP). These components support interoperability across a range of computing platforms. Persona Client is built with Java technology. To integrate with browsers, a .NET version of the Persona Client was built by Richard Wu (Master's graduate, University of Alberta, Canada). J2ME will find a potential use if the client is to be used in Mobile devices. The system is designed to be scalable, aiming for a distributed architecture. Based on the amount of data and considering the need for redundant data, Persona Concept is to be distributed and stored on multiple servers.

Persona Data is encrypted using public key encryption schemes with only authorized users being able to decrypt and access their own data. We believe that this makes the system relatively secure, even in the eventuality of the hosting server getting compromised. Assessing the security risks of this approach is left for further study. The potential for incorporating intrusion detection mechanisms is also an area outside the scope of this project but an interesting area of study. Colin Van Dyke, a PhD student under Dr. Cetin Koc of Oregon State University is conducting studies in this field.

6.3.1.2 Persona Server Design

Figure 5 illustrates in broad terms the usage of Persona Server. Users obtain signed credentials and other certificates from certification issuing authorities, which are a part of their Persona along with other personal data, and stored in the trusted/semi-trusted host. The interaction with the Persona proceeds with the help of Persona Client that can be integrated with a Web browser context. Both the CA and Persona Server will be hosted using SOAP/Web services to provide open but protected access. Persona Client is responsible for coordinating the interactions with Web services providers, getting the data from Persona Server, and providing them to Web services providers. The back-end storage of Persona data of users can be implemented through commercial database solutions like SQL Server, or more preferably (as is done in this project), through interoperable solutions like XML format documents.

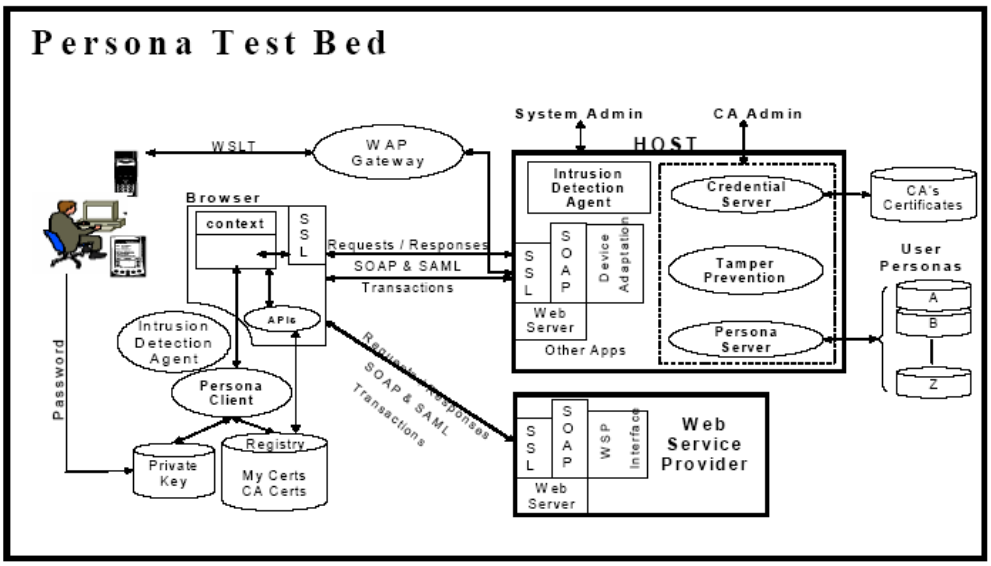


Figure 8: Persona Server design

The published set of services, the server components and the architecture can be represented as in Figure 7 below.

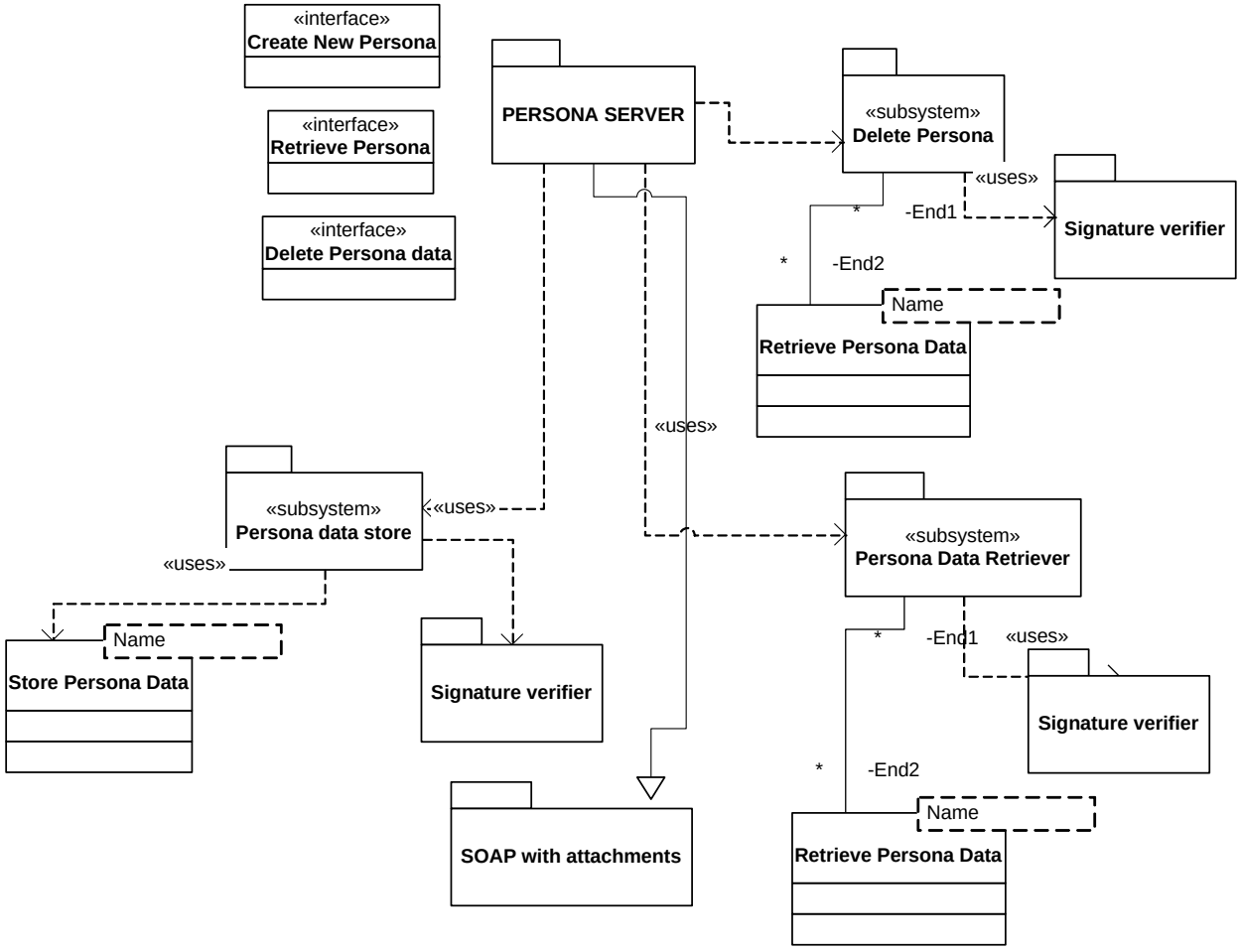


Figure 9: Persona Server interfaces

6.3.1.3 Persona Client Design

The Persona Client is the principal interface for the user to her Persona Data and other services. An integration of the Persona Client with the browser was implemented by Richard Wu (University of Alberta student who completed his master's degree under Prof. Toth in 2003). We welcome the reader to go through [17] for more information on how this was accomplished. Our implementation of the Persona Client was targeted at implementing the functionalities to generate Certificates and the Persona data access mechanisms. The principal components of the Persona Client can be represented by Figure 8 below:

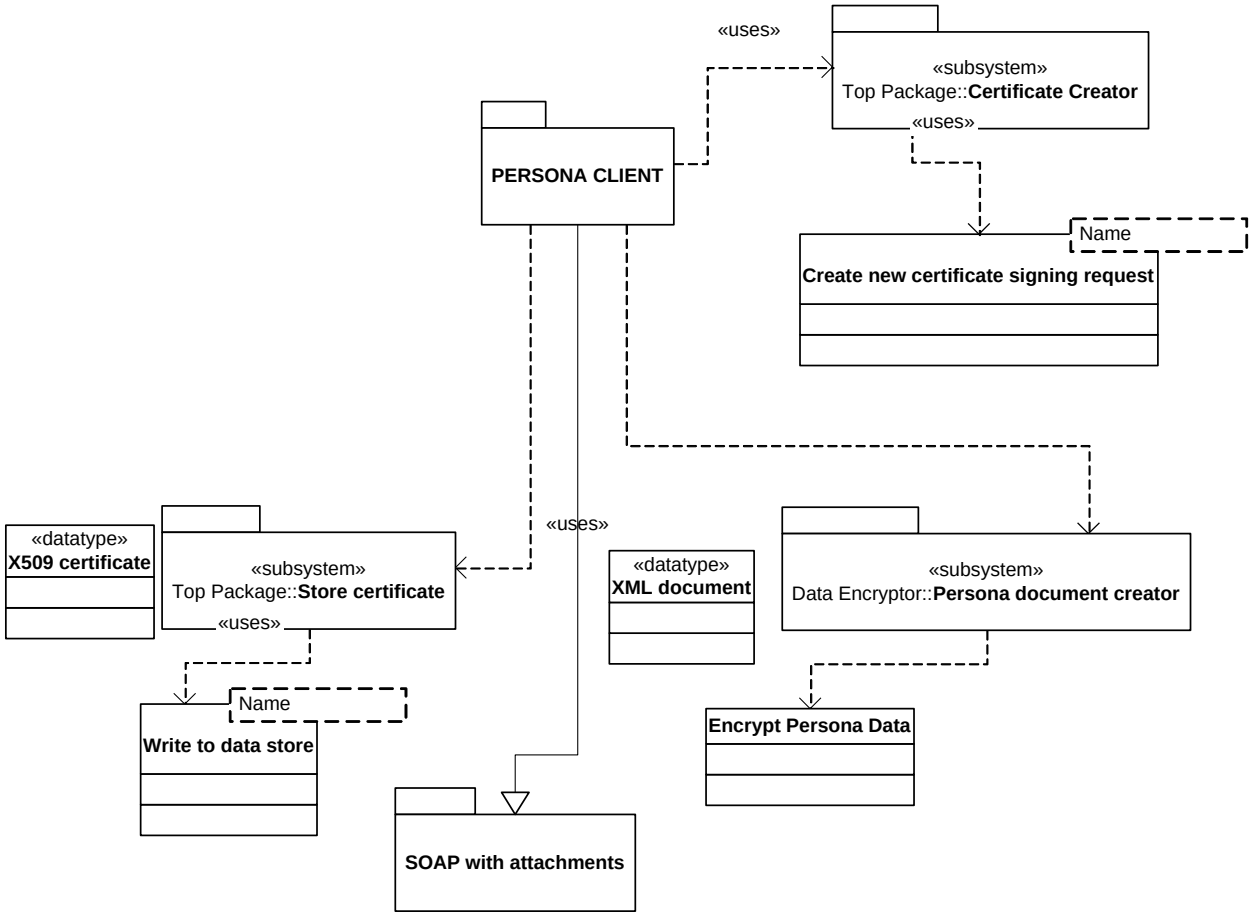


Figure 10: Persona Client design

6.3.1.4 Certificate Authority Upgrade

An upgrade was put in place to the Certification Authority, which enabled it to send and receive X509 Certificates in the form of files. The published set of services and the CA architecture can be represented by Figure 9. There is a Certificate Creator service, just in case for devices that may not have the computing power to create certificates. In this case, the private key will be known to the CA and have to be communicated to the user by some offline means.

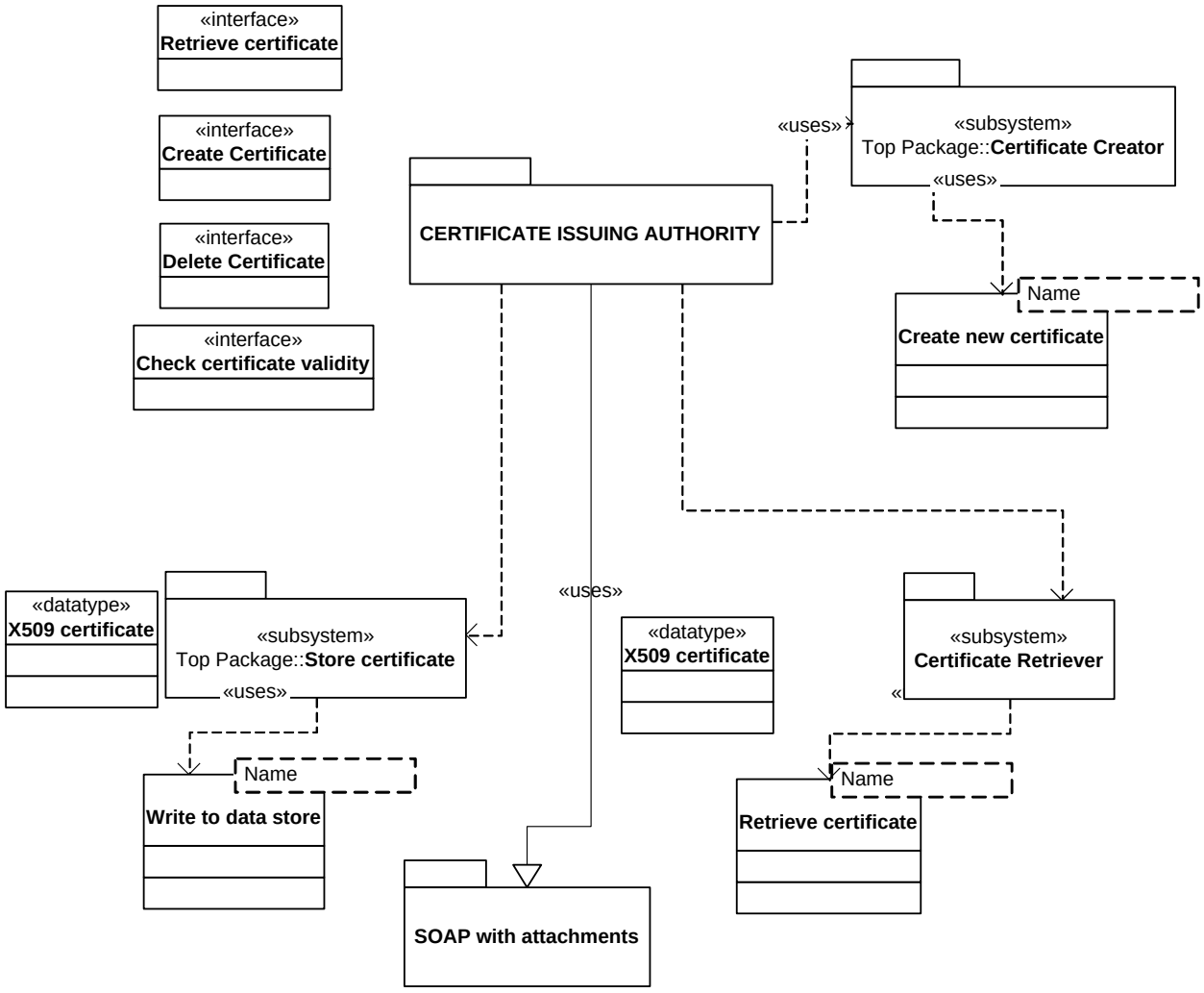


Figure 11: Certificate Server interfaces

7 SOFTWARE PROTOTYPING

7.1 Persona Client

The Persona Client is a SOAP-aware module sitting in the user machine and is capable of generating and sending SOAP requests to a Persona Server over HTTP. A SOAP request is a type of SOAP message; normally there are only two types of SOAP messages: a *SOAP request* is what a SOAP client sends to a SOAP server and a *SOAP response* is what a SOAP server sends to a SOAP client in response. Figure 10 illustrates a simple SOAP request

```
<?xml version=”1.0” encoding=”UTF-8”?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <signmycrt soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/”>
      <op1 xsl:type=”xsd:string”>Mahesh Subramanium</op1>
    </signmycrt>
```

```
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 12: Sample SOAP request

The Persona Client presents a user interface for the user to add/edit Persona contents. The data is collected and organized under principal field names like name, credit card number, bank account number etc. Validity checks for the individual fields have not been built into this solution and can be part of future development work.

7.2 Persona Server

The Persona server is also a SOAP-aware machine and has the capability to accept requests from Persona Client and author appropriate responses. These encoded responses go back to the requesting Persona Client. Inside the Persona Server, there are three entities:

- Service manager
- Deployed services list
- XML translator

The service manager is responsible for managing services against requests. It will read name of the SOAP service that the SOAP client wants to invoke and check whether the required service actually resides on this SOAP server. To that end, it will consult the deployed services list, a list of all services that the SOAP server hosts. If yes, the server manager will pass on the SOAP request to the XML translator. The XML translator is then responsible for converting the XML structure of the SOAP request to that of a programming language (for example, Java programming language) that programmers use to implement the actual service. It is also responsible for converting the response from the actual service back to the XML structure of the SOAP response.

The prototype was developed on Red Hat Linux 8.0 (running on a Pentium processor based Dell workstation). The Java code is based on AXIS Beta 2, Java JDK 1.4, and org.bouncycastle API. Apache's Jakarta Tomcat 4.0.3 was used as the Web server and servlet engine.

7.2.1 Install Tomcat

Go to the Apache Web site (http://www.apache.org/tomcat) and download Tomcat. At a minimum, we need these Java packages: jakarta-regexp-1.2, servlet-2.3, xerces-j 1.4.4, Tomcat 4.0.3 itself, and the Tomcat 4.0.3 default Web applications.

Once we've downloaded the files, use ***rpm*** to install them. After running the installs, we should have several new directories and files. On Red Hat, the Tomcat server is installed by default into */var/tomcat4.* The *init.d* script is installed to */etc/init.d/tomcat4*. The startup configuration script can be found in */etc/tomcat4/conf/tomcat4 .conf.*

7.2.2 Configure the Tomcat script

Before we launch Tomcat, we must edit the Tomcat start-up script to point to the installation of the JDK. Go to the directory */etc/tomcat4/conf* and edit the file *tomcat4.conf*. Change the `JAVA_HOME` variable to point to your copy of JDK 1.4. On the workstation, the line was `JAVA_HOME=/usr/java/j2sdk1.4.2_07`. The other script variables, `CATALINA_HOME`, `JASPER_HOME`, and `CATALINA_TMPDIR` should already be set correctly.

Finally, we *must* add the following line to the end of the tomcat4.conf script or AXIS will not work.

```
JAVA_ENDORSED_DIRS="$CATALINA_HOME"/bin:
```

```
"$CATALINA_HOME"/common/lib:
"$CATALINA_HOME"/webapps/axis/WEB-INF/lib
```

7.2.3 Start and Test Tomcat

As root, use the *init.d* script to launch the server:

```
# /etc/init.d/tomcat4 start
Starting tomcat4:  [  OK  ]
```

To test that Tomcat is running, point the Web browser to http://localhost:8080/. (The port on your system is set in the file TOMCAT_HOME/ *conf/server.xml*. If we see the default Tomcat web page, the Tomcat server is working.

7.2.4 Install AXIS

The AXIS toolkit is distributed as a collection of *jar* files. To install AXIS on your server, go to http://xml.apache.org/axis/index.html and download the Beta 2 version. The version will be in a *tar* file named *xml-axis.tar.gz*. Unzip and untar the file into a temporary directory. We'll refer to this temporary directory as AXIS_HOME.

Copy the entire AXIS_HOME/*webapps/axis* directory to the directory TOMCAT_HOME/*webapps*. Next, copy the Xerces XML parser located at TOMCAT_HOME/*common/lib/xerces.jar* to TOMCAT_HOME/*webapps/axis/WEB-INF/lib*. The following commands show what to copy:

```
# cp -pr $AXIS_HOME/webapps/axis \ $TOMCAT_HOME/webapps
# cp $TOMCAT_HOME/common/lib/xerces.jar \ $TOMCAT_HOME/webapps/axis/WEB-INF/lib
```

7.2.5 Test Axis

At this point, AXIS is installed and ready to use on the server. To test AXIS, point your browser to http://localhost:8080/axis/. You should see a page like the one shown in Figure 13 below.

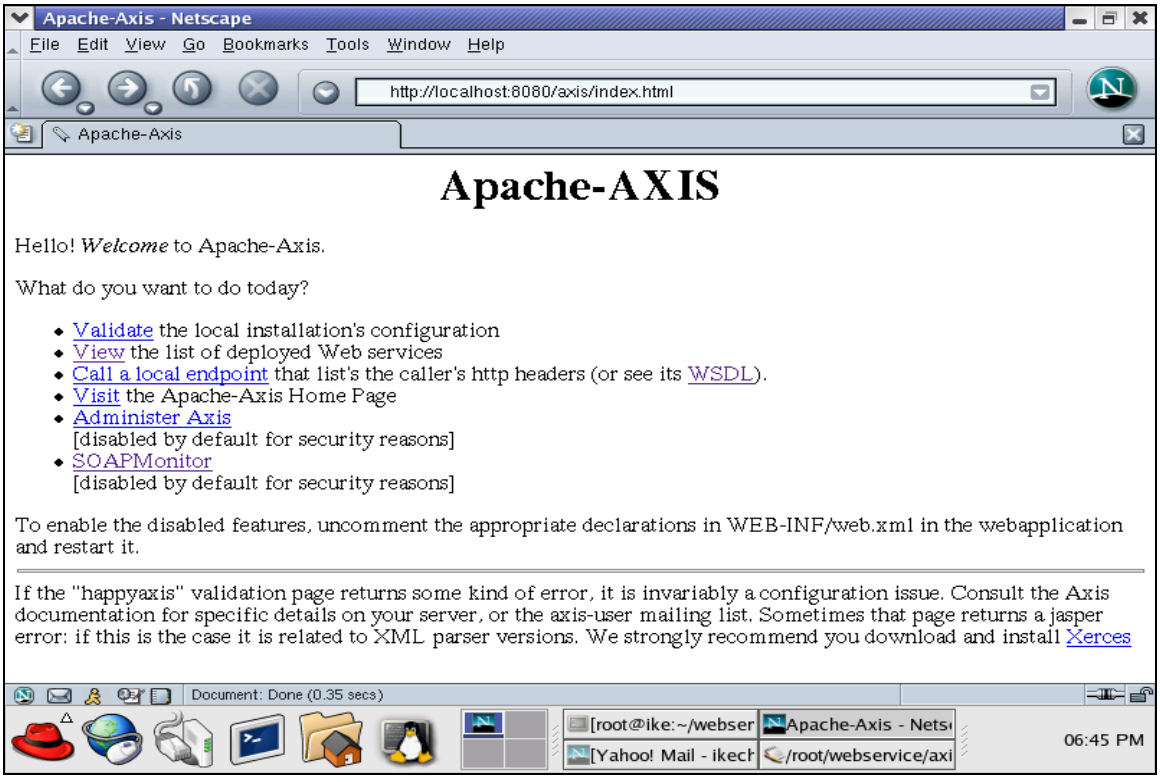


Figure 13: Testing Axis

7.2.6 Set up the development environment

To develop clients with AXIS, we will need to add several of the AXIS *jar* files to the Java CLASSPATH. If we've defined the AXIS_HOME environment variable in your shell, then the following command will add everything we need.

```
% export CLASSPATH=\
$AXIS_HOME/src/axis/lib/axis.jar:\
$AXIS_HOME/src/axis/lib/jaxrpc.jar:\
$AXIS_HOME/src/axis/lib/commons-logging.jar:\
$AXIS_HOME/src/axis/lib/tt-bytecode.jar:\
$AXIS_HOME/src/axis/lib/wsdl4j.jar:\
$TOMCAT_HOME/common/lib/xerces.jar:\
$AXIS_HOME/src/axis:\
$CLASSPATH
```

7.2.7 Installing the Persona Web Services on Axis

7.2.7.1 CreatePersona

The Persona Client uses this service to store the user Persona on the Persona Server. I am using Axis JWS (Java Web Service) Files - Instant Deployment to publish this web service.

Make a java file called **CreatePersona.java** that will be called on the Axis server side. Please refer to the **CreatePersona.java** source code in Appendix B.

Copy **CreatePersona.java** file into the webapps directory, and rename it "CreatePersona.jws".

- Axis automatically locates the file, compiles the class, and converts SOAP calls correctly into Java invocations of the service class.
- The Persona Client on the client side that will call the method: **CreatePersona** in the web service **CreatePersona** running on the Axis server side. Please refer to the **Persona Client (pClient.java)** source code in Appendix.

This service in turn calls another method called the WritePersona to actually store the Persona data on the Persona Server.

7.2.7.2 DeletePersona

The Persona Client uses this service to delete the complete Persona from the Persona Server. Extreme care needs to be exercised before calling this method. The Persona Client currently asks only for confirmation before executing this step. Future development work could include some more robust confirmation mechanisms.

The implementation of the Web Service is similar to CreatePersona service detailed above.

7.2.7.3 RetrievePersona

The Persona Client uses this service to retrieve Persona data. The Persona Client presents the Persona Server with a signed request and the name of the user of which it wishes to access the data. The Persona Server utilizes the Signature Verifier method to verify the signature of the requesting user, retrieves the

appropriate Persona, and sends it back. The Persona is sent back as a XML file using Soap with Attachments package.

8 POTENTIAL FUTURE WORK

Future extensions of this project could include exploring custom credential generation based on Security Assertion Markup Language (SAML) and Extensible Access Control Markup Language (XACML) which together can be used to specify XML-based credentials and access control policies for an enterprise. Other extensions include the implementation of discretionary and mandatory security policies using XACML to allow users to conduct transactions across various domains.

Other extensions include the implementation of discretionary and mandatory security policies to allow users to conduct transactions across various domains. A very interesting aspect of custom credentials is the ability to entrust or authorize users with specific capabilities (for a specified period of time), which we think would find application in authorizing users to conduct transactions on behalf of organizations. SAML ties in very nicely with this requirement. SAML provides signed authorizations to entities to perform certain activities on behalf of another entity. A real world example to this would be a firm authorizing an employee to conduct transactions (like buying products from a dealer) on its behalf. Now the credentials should have sufficient data for verifying the nature and validity of the data contained in it. Also, SAML introduces the semantics of conveying authentication and authorization information which different organizations can agree upon. A simple extension of the project is to allow off-line access to selected Persona Data to authorized WSP's to enable transactions like offline bidding. Basically, this involves, making a section of the Persona shareable to trusted Web Service Providers, say encrypt the credit card number with the public key of the Web Service Provider and hosted on the Persona with interfaces to access it. The WSP can access this information and conduct transactions like automatic bill payment and electronic bidding (to a limit which can be set by the user). This is in contrast to the credit card number stored in the WSP for the same purpose.

As more and more data is shared over the Internet, hosting data securely for easy access from a variety of devices is becoming an attractive option. In addition, users would like to save a lot of personalized information to be accessible from different systems. To protect this information, intrusion detection mechanisms becomes a perfect fit for use with the Persona Concept. Already a healthy amount of research has been done on this and it is an area which promises great results in the coming days.

9 PROJECT SUMMARY

In the implementation of this project, I had chance to learn about various emerging technologies and techniques. I have also increased my adherence to best practices in project development, documentation and testing. As part of the implementation of the project, I undertook responsibilities of system installation, configuration and maintenance and gave me fresh hands on experience on a wide variety of environments. . The following list summarizes the key areas and what I learned in each area:

- Security Mechanisms: PKI, CA, X.509, SSL, Digital Signature, SAML, XACML, Intrusion detection, Access Control Lists.
- Web Services: SOAP/XML, WSDL, UDDI, AXIS, Web Service Toolkit (WSTK).
- Tools/environments: Red hat Linux, J2EE, Bouncy castle API, Apache Web Server, Tomcat Web Server, AXIS, WSTK, JDK, SOAP Monitor, Metrowerks Codewarrior.

- Programming and Scripting Languages: Java, shell scripts, XML.

10 REFERENCES

[1] K.C. Toth, M.Subramanium, Requirements for the Persona Concept, Requirements for High Assurance Systems (RHAS'03) workshop, Monterey, CA, September 9, 2003

[2] K.C. Toth, M. Subramanium, The Persona Concept: A Consumer-Centered Identity Model, MobEA (Emerging Applications for Wireless and Mobile Access), Budapest, Hungary, May 2003.

[3] K.C. Toth, M. Subramanium, Persona Concept for Privacy and Authentication, International Business & Economics Research Journal, June 2003.

[4] Jiandong (Ike) Chen, "Certification Authority Services For the Persona Concept", CS506 Project, Computer Science, Oregon State University, October 2003

[5] Associated Press. "Feds Charge 3 in Massive Credit Fraud Scheme", CNN.com. November 26, 2002, http://www.cnn.com/2002/LAW/11/26/ID.theft.ap/index.html

[6] J. Leyden. "Feds Break Massive Identity Fraud", The Register, http://online.securityfocus.com/news/1718

[7] Microsoft, www.microsoft.com/netservices/passport/passport.asp, March 2002

[8] Documents on SAML found at OASIS website, www.oasis-open.org/committees/security/#documents

[9] Liberty Alliance Specification, www.projectliberty.org

[10] David P. Kormann, Ariel D.Rubin, "Risks of the Passport Single Sign On Protocol", Computer Networks, Elsevier Science Press, Volume 33, pages 51-58, 2000.

[11] Marc Slemco, "Microsoft Passport to Trouble", http://online.securityfocus.com/library/3632

[12] Logan Bodia "Real World SSL Benchmarking", www.ciscoworldmagazine.com/webpapers/2002/05_rainbow.shtml

[13] Siddarth Anand, "SAML" (a summary of SAML basics and sample use cases), CS505 Readings and Conference report, Computer Science, Oregon State University, March 2003

[14] Royce, Walker, Software Project Management: A Unified Framework, Addison Wesley, 1998, ISBN 0-201-30958-0

[15] L. Bass, P. Clements, and R. Kazman. Software Architecture in Practice. Addison Wesley, Reading, Massachusetts, 1998.

[16] Kazman, P., Bass, L., Abowd, G. and Webb, S.M. (1994) SAAM: A Method for Analyzing the Properties of Software Architectures, in Proceedings of the International Conference on Software Engineering (ICSE 16), 81-90.

[17] Richard Wu, "Mobile Persona Client Agent: Missing Pieces in On-line User-Centric Security Solutions", Project Report, Master of Software Technology, Computer Science, University of Alberta, May 2003

11 APPENDIX

11.1 Installation Instructions

This section briefly describes the steps to undertake to set up the environment for Persona usage. Choices exist on the packages themselves, but the required components are identified herewith.

11.1.1 Persona Server

For the Persona Server, the prototype uses Red Hat Linux release 9 (Shrike) release 2.4.20-31.9. However, since Red Hat has stopped support for its desktop version of Linux, any other Linux versions should be equally compatible. A recommended one is Mandrake Linux at www.mandrakelinux.com

The Persona Server uses Jakarta Tomcat from Apache Software Foundation for the implementation of the servlet components. You can get Tomcat from here: http://jakarta.apache.org/tomcat/index.html. The prototype uses Jakarta Tomcat v4.1.30. At a minimum, we need these Java packages: jakarta-regexp-1.2, servlet-2.3, xerces-j 1.4.4, Tomcat 4.0.3 itself, and the Tomcat 4.0.3 default Web applications.

Once we've downloaded the files, use ***rpm*** to install them. After running the installs, we should have several new directories and files. On Red Hat, the Tomcat server is installed by default into */var/tomcat4.* The *init.d* script is installed to */etc/init.d/tomcat4*. The startup configuration script can be found in */etc/tomcat4/conf/tomcat4 .conf.*

Install Java Development Kit (JDK) to support the Web Services. The prototype uses JDK version 1.4.2_04. Obtain the latest version of JDK and install it in the Server. Before we launch Tomcat, we must edit the Tomcat start-up script to point to the installation of the JDK. Go to the directory /etc/tomcat4/conf and edit the file tomcat4.conf. Change the JAVA_HOME variable to point to your copy of JDK. The other script variables, CATALINA_HOME, JASPER_HOME, and CATALINA_TMPDIR should already be set correctly.

Install Apache AXIS which is an implementation of the Simple Object Access Protocol (SOAP) submission to W3C. AXIS is the container for the Web Services and resides on top of Tomcat HTTP Server. The AXIS toolkit is distributed as a collection of *jar* files. To install AXIS on your server, go to http://xml.apache.org/axis/index.html and download the latest version. The prototype uses AXIS version 1.1. The version will be in a *tar* file named *xml-axis.tar.gz.* Unzip and untar the file into a temporary directory. We'll refer to this temporary directory as AXIS_HOME.

Copy the entire AXIS_HOME/*webapps/axis* directory to the directory TOMCAT_HOME/*webapps*. Next, copy the Xerces XML parser located at TOMCAT_HOME/*common/lib/xerces.jar* to TOMCAT_HOME/*webapps/axis/WEB-INF/lib.*

Finally, we must add the following line to the end of the tomcat4.conf script or AXIS will not work.

```
JAVA_ENDORSED_DIRS="$CATALINA_HOME"/bin:
  "$CATALINA_HOME"/common/lib:
  "$CATALINA_HOME"/webapps/axis/WEB-INF/lib
```

To develop clients with AXIS, we will need to add several of the AXIS *jar* files to the Java CLASSPATH. If we've defined the AXIS_HOME environment variable in your shell, then a command like the following will add everything we need.

% export CLASSPATH=\

$AXIS_HOME/src/axis/lib/axis.jar:\

$AXIS_HOME/src/axis/lib/jaxrpc.jar:\

$AXIS_HOME/src/axis/lib/commons-logging.jar:\

$AXIS_HOME/src/axis/lib/tt-bytecode.jar:\

$AXIS_HOME/src/axis/lib/wsdl4j.jar:\

$TOMCAT_HOME/common/lib/xerces.jar:\

$AXIS_HOME/src/axis:\

$CLASSPATH

Install the Security Providers and Java Cryptographic Extensions (JCE) from Bouncy Castle. Another option is the providers and JCE from Cryptix. The prototype uses Bouncy Castle. These can be obtained from www.bouncycastle.org.

We need these because in Java API terminology, cryptographic services are programming abstractions to carry out or facilitate cryptographic operations. Most often, these services are represented as Java classes with names conveying the intent of the service. An instance of a service is always associated with one of many algorithm or types. The algorithm determines the specific sequence of steps to be carried out for a specific operation. Similarly the type determines the format to encode or store information with specific semantics. The entity or organization implementing an algorithm is called the Provider. A Provider need not implement all the services specified in the JCE.

The Security provider is coupled with the hosted Web Services to provide functionalities involving cryptographic operations. An example would be the service for signing a Certificate where the X509 Certificate is generated on the user device and a Certificate Signing Request (CSR) is sent to the Certificate Authority where the CA signs the request.

Once these steps have been completed, the Persona Server is ready to accept requests from the Persona Client.

11.1.2 Persona Client

The Persona Client requires some basic set up to enable it to implement functionalities as defined by the Persona Concept. The underlying operating system can be Windows or Linux, virtually any Operating System for which a Java Virtual Machine (JVM) exist. The Persona Client for the prototype is done in Java using JDK version 1.4.2_07. The underlying operating system is Windows XP Home edition. But the same could be implemented using .NET.

Install the latest version of JDK to support the client code. Also, install the Security Providers and JCE as done for the Server. This is needed because some cryptographic functionality like generation of the CSR, encryption of data using public key require the presence of the Provider. Please do not forget to add the Provider (Bouncy Castle or Cryptix) and a corresponding priority value to the java.security file located at JDK_HOME\jre\lib\security where JDK_HOME refers to the path of the directory for the Java installation. This is the line that you have to add (for Bouncy Castle):

```
security.provider.6=org.bouncycastle.jce.provider.BouncyCastleProvider
```

The Persona Client also requires the presence of XML parsers and APIs to support the transfer of files to and from the Persona Server. Also, since the Persona data is defined in XML files, the presence of parsers makes it easier to parse and extract Persona contents.

11.2 Screen Shots

Below some of the important screens during the execution of the prototype has been captured.

Figure 14: Persona Start Screen

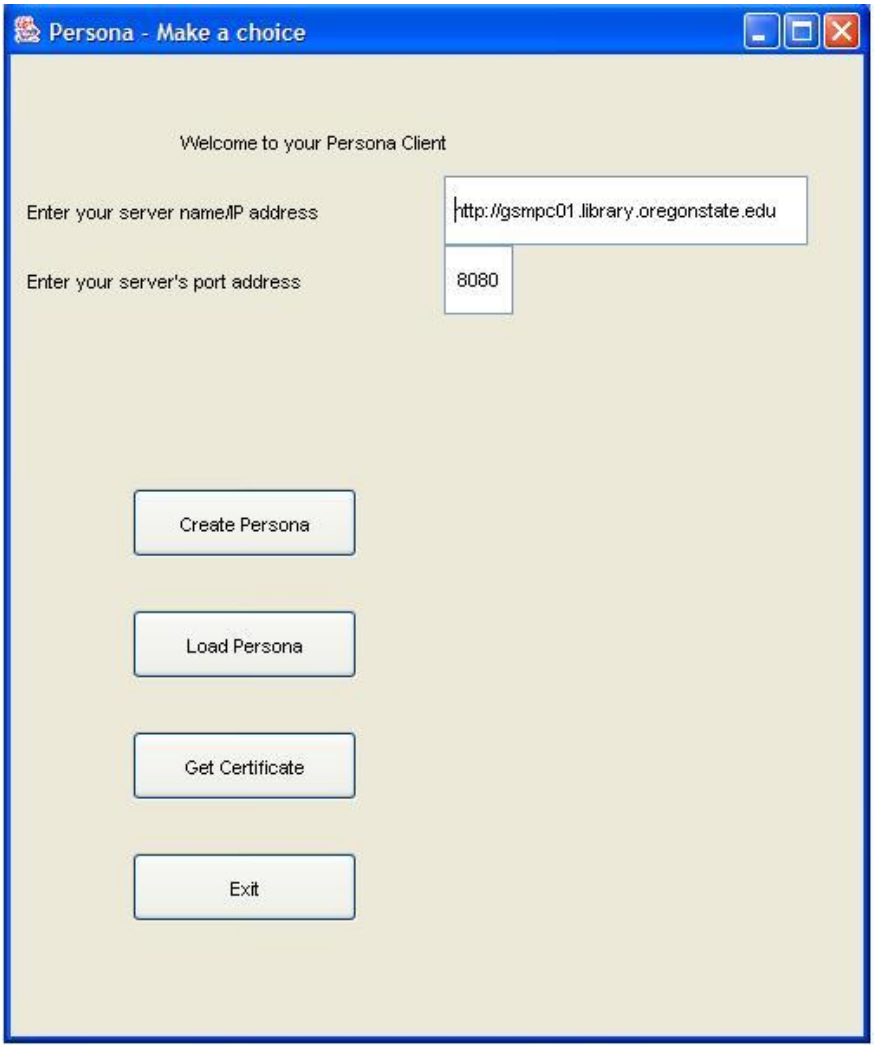


Figure 15: Persona – Certificate Request Screen

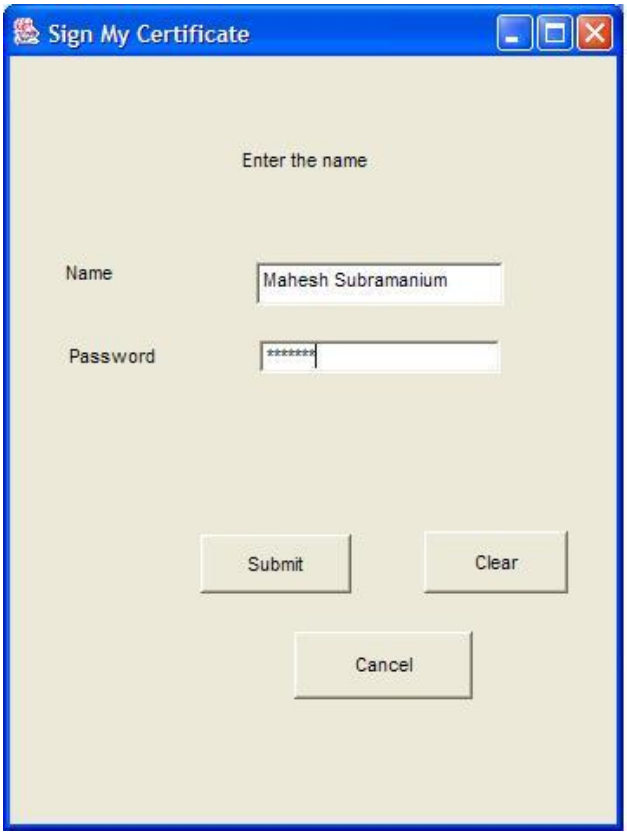


Figure 16: Persona – Certificate Generation Successful

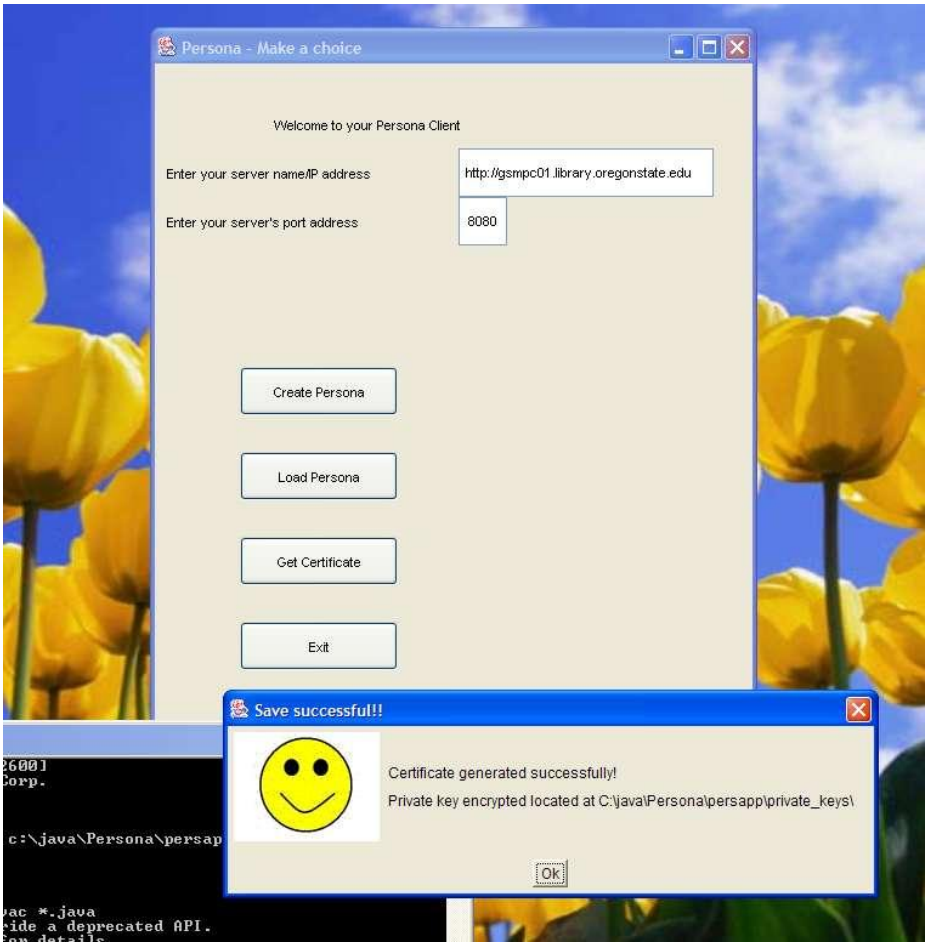


Figure 17: Persona – Generated Certificate

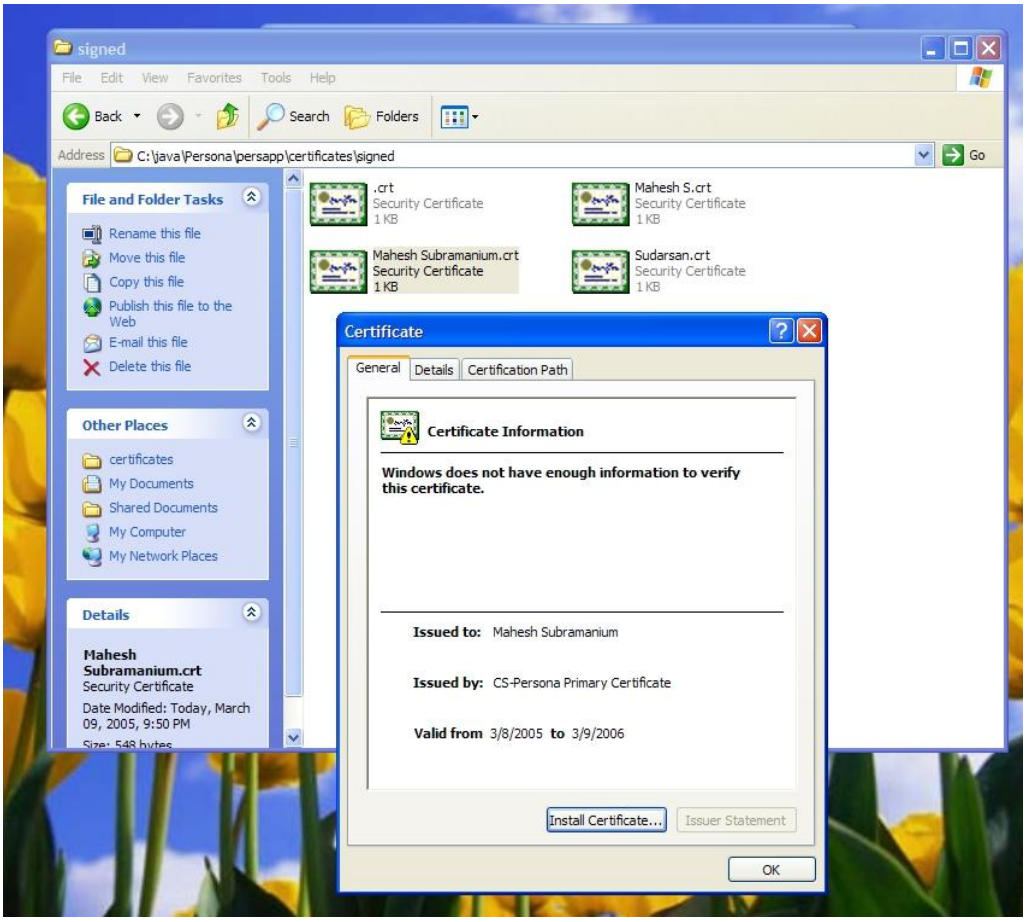


Figure 18: Persona – Generated Private Key

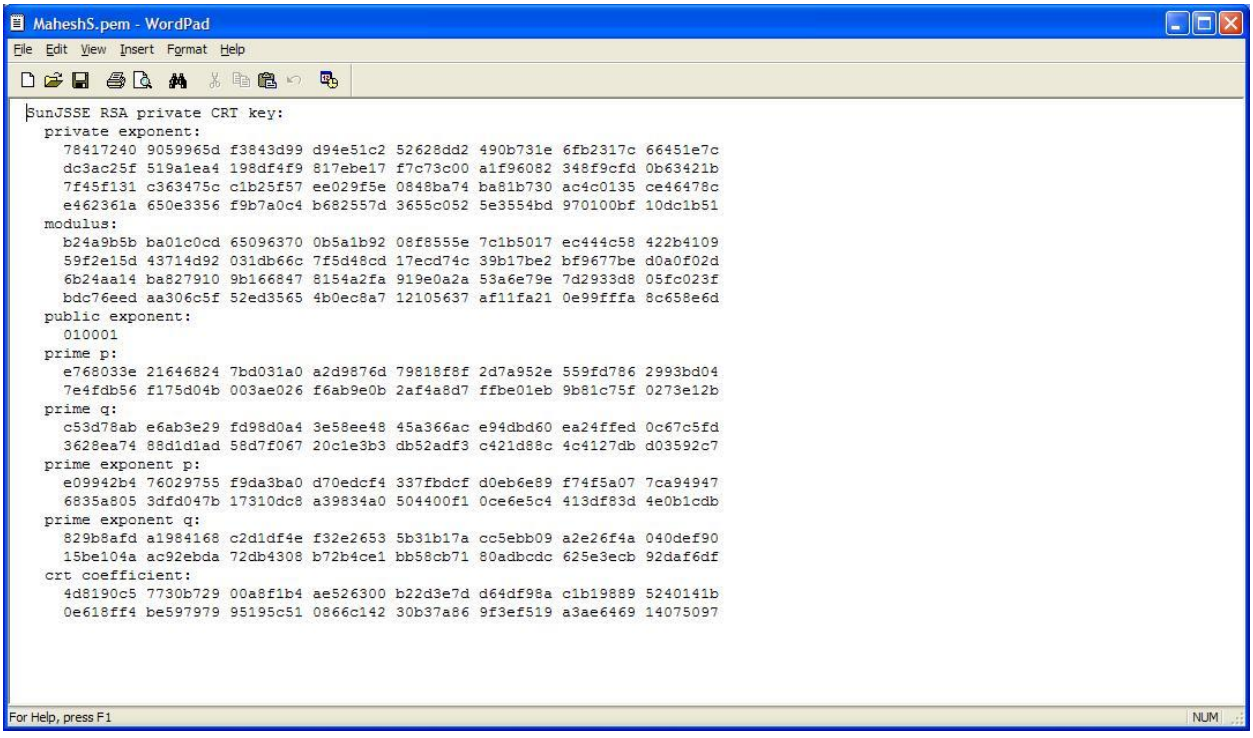


Figure 19: Persona – Password encrypted Private Key

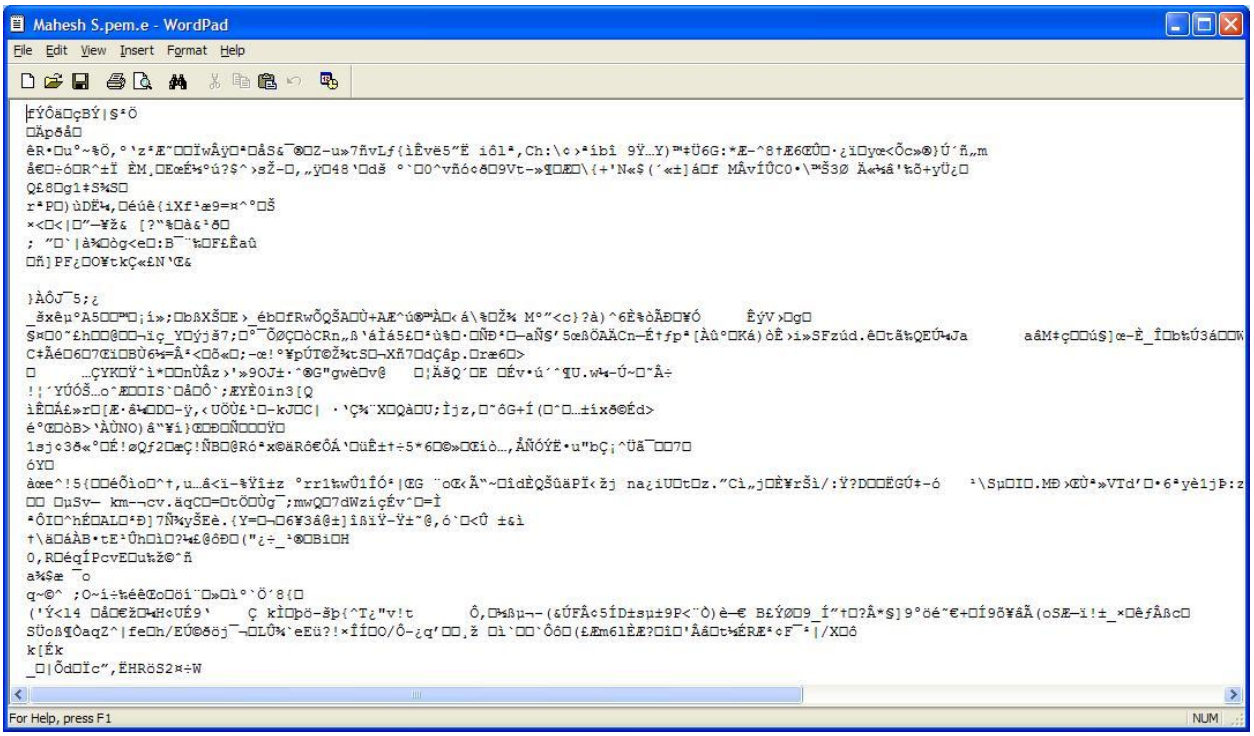


Figure 20: Persona – XML document with encrypted data between tags

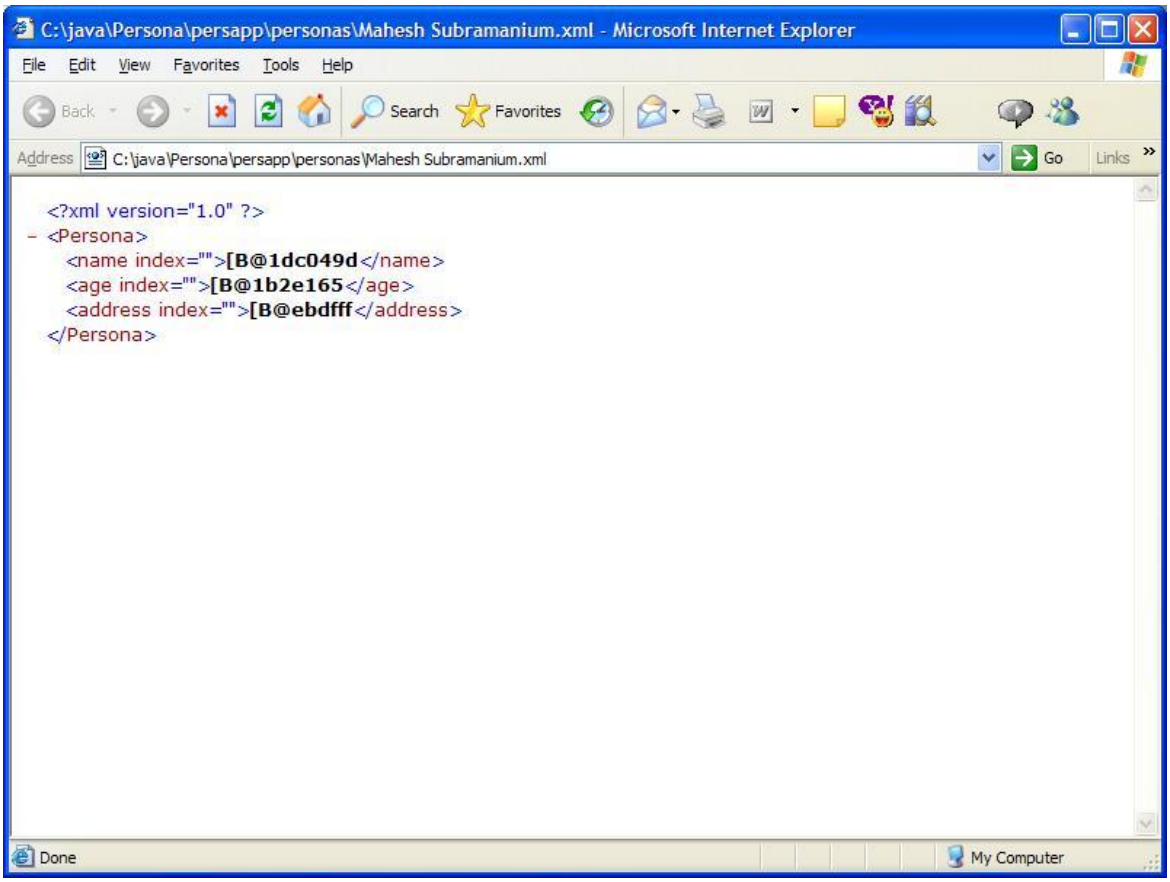


Figure 21: Persona – Persona asking for Password to decrypt file containing Private Key

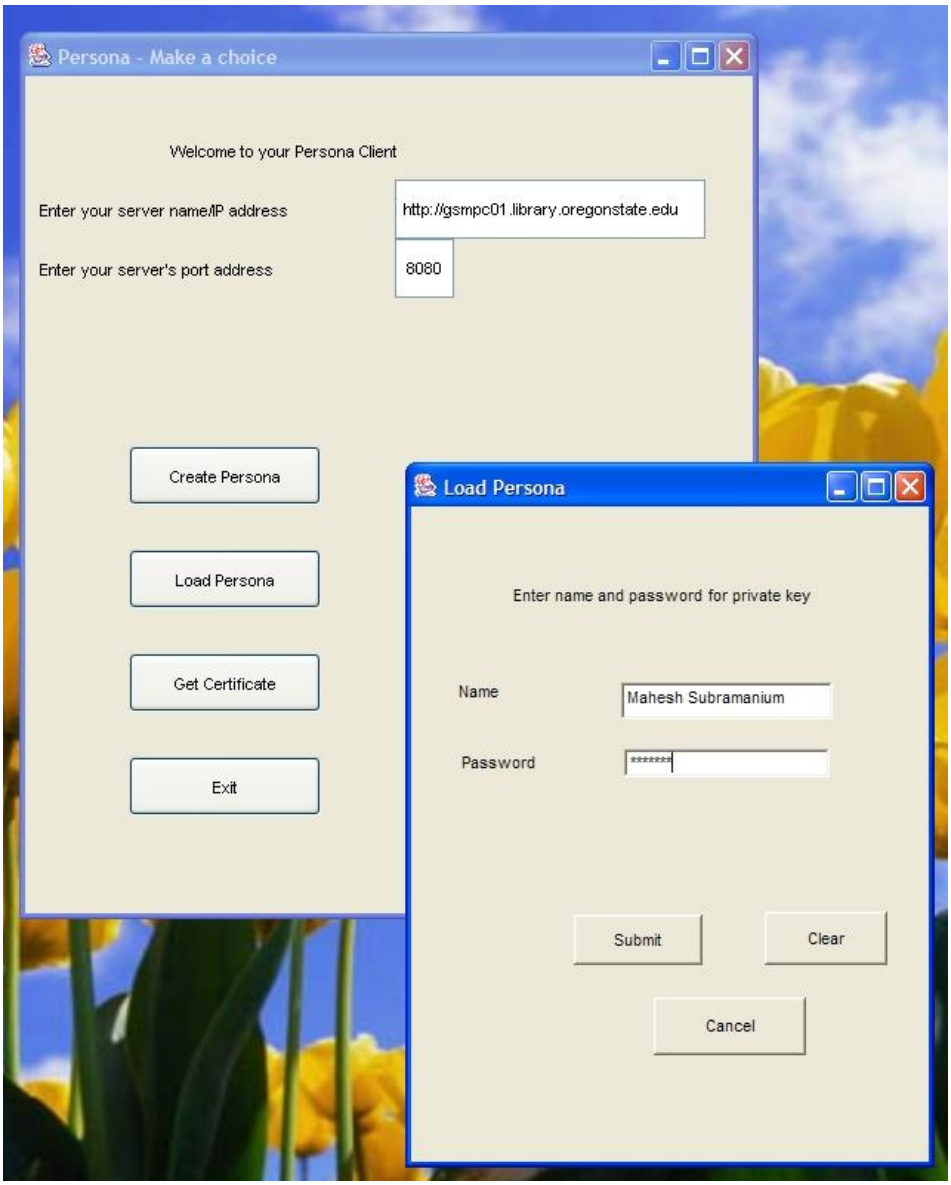
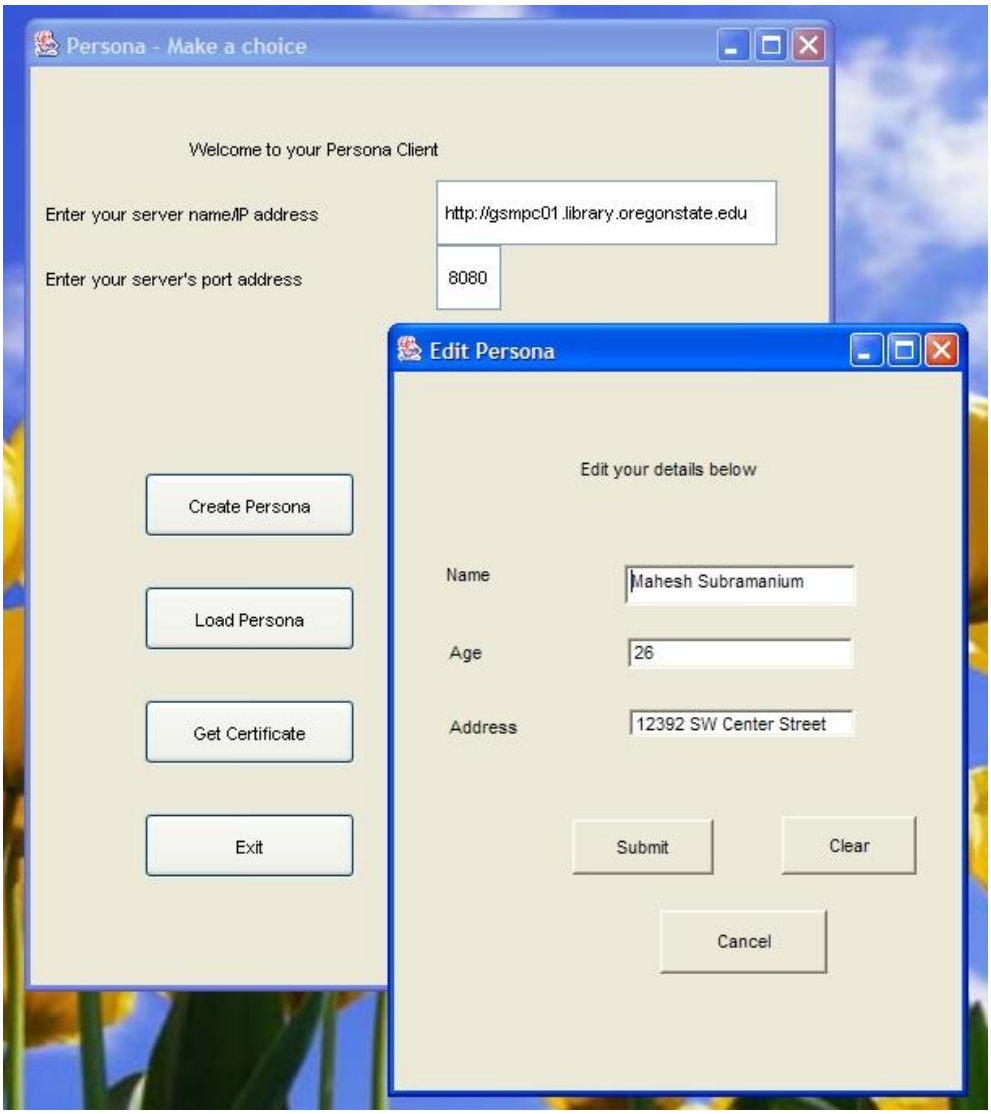


Figure 22: Persona –Sample Error Message

Figure 23: Edit Persona



11.3 Source Code

11.3.1 pClient.java

```
/**
 * pClient.java
 *
 * Description:
 * @author                      subramma
 * @version
 */

package persapp;
import javax.swing.*;

public class pClient {

        public pClient() {
                try {
                        // For native Look and Feel, uncomment the following code.

                        try {

        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
                        }
                        catch (Exception e) {
                        }

                        startup_Frame frame = new startup_Frame();
                        frame.initComponents();
                        frame.setVisible(true);
                }
                catch (Exception e) {
                        e.printStackTrace();
                }
```

```
        }

        // Main entry point
        static public void main(String[] args) {
                new pClient();
        }

}
```

11.3.2 Startup_Frame.java

```
/**
 * startup_Frame.java
 *
 * Description:
 * @author                      subramma
 * @version
 */

package persapp;

import java.awt.*;
import java.lang.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;

public class startup_Frame extends javax.swing.JFrame {

// IMPORTANT: Source code between BEGIN/END comment pair will be regenerated
// every time the form is saved. All manual changes will be overwritten.
// BEGIN GENERATED CODE
        // member declarations
        javax.swing.JLabel startup_screen_label = new javax.swing.JLabel();
        javax.swing.JButton createPersona_button = new javax.swing.JButton();
        javax.swing.JButton loadPersona_button = new javax.swing.JButton();
```

```
        javax.swing.JButton getCertificate_button = new javax.swing.JButton();
        javax.swing.JButton exit_button = new javax.swing.JButton();

        //defining the server addres
        javax.swing.JLabel serveraddress_label = new javax.swing.JLabel();
        static javax.swing.JTextField serveraddress_textfield = new javax.swing.JTextField();

        //defining the port
        javax.swing.JLabel serverport_label = new javax.swing.JLabel();
        static javax.swing.JTextField serverport_textfield = new javax.swing.JTextField();

        //public String host;


// END GENERATED CODE

        public startup_Frame() {
        }

        public void initComponents() throws Exception {
// IMPORTANT: Source code between BEGIN/END comment pair will be regenerated
// every time the form is saved. All manual changes will be overwritten.
// BEGIN GENERATED CODE
                // the following code sets the frame's initial state
                startup_screen_label.setSize(new java.awt.Dimension(290, 40));
                startup_screen_label.setLocation(new java.awt.Point(30, 30));
                startup_screen_label.setVisible(true);
                startup_screen_label.setText("Welcome to your Persona Client");
                startup_screen_label.setHorizontalAlignment(javax.swing.JLabel.CENTER);
                createPersona_button.setVisible(true);
                createPersona_button.setSize(new java.awt.Dimension(130, 40));
                createPersona_button.setText("Create Persona");
                createPersona_button.setLocation(new java.awt.Point(70, 250));
                loadPersona_button.setVisible(true);
                loadPersona_button.setSize(new java.awt.Dimension(130, 40));
```

```
loadPersona_button.setText("Load Persona");
loadPersona_button.setLocation(new java.awt.Point(70, 320));
getCertificate_button.setVisible(true);
getCertificate_button.setSize(new java.awt.Dimension(130,40));
getCertificate_button.setText("Get Certificate");
getCertificate_button.setLocation(new java.awt.Point(70,390));
exit_button.setVisible(true);
exit_button.setSize(new java.awt.Dimension(130, 40));
exit_button.setText("Exit");
exit_button.setLocation(new java.awt.Point(70, 460));
setLocation(new java.awt.Point(0, 0));
getContentPane().setLayout(null);
setTitle("Persona - Make a choice");


serveraddress_label.setSize(new java.awt.Dimension(290, 40));
serveraddress_label.setLocation(new java.awt.Point(10, 70));
serveraddress_label.setVisible(true);
serveraddress_label.setText("Enter your server name/IP address");
serveraddress_label.setHorizontalAlignment(javax.swing.JLabel.LEFT);

serveraddress_textfield.setSize(new java.awt.Dimension(210, 40));
serveraddress_textfield.setLocation(new java.awt.Point(250, 70));
serveraddress_textfield.setVisible(true);
serveraddress_textfield.setText("http://gsmpc01.library.oregonstate.edu");
serveraddress_textfield.setHorizontalAlignment(javax.swing.JLabel.LEFT);

serverport_label.setSize(new java.awt.Dimension(290, 40));
serverport_label.setLocation(new java.awt.Point(10, 110));
serverport_label.setVisible(true);
serverport_label.setText("Enter your server's port address");
serverport_label.setHorizontalAlignment(javax.swing.JLabel.LEFT);

serverport_textfield.setSize(new java.awt.Dimension(40, 40));
serverport_textfield.setLocation(new java.awt.Point(250, 110));
```

```
serverport_textfield.setVisible(true);
serverport_textfield.setText("8080");
serverport_textfield.setHorizontalAlignment(javax.swing.JLabel.CENTER);



getContentPane().add(startup_screen_label);
getContentPane().add(serveraddress_label);
getContentPane().add(serveraddress_textfield);
getContentPane().add(serverport_label);
getContentPane().add(serverport_textfield);

getContentPane().add(createPersona_button);
getContentPane().add(loadPersona_button);
getContentPane().add(getCertificate_button);
getContentPane().add(exit_button);

setSize(new java.awt.Dimension(500, 600));

// event handling
createPersona_button.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent e) {
                createPersona_buttonActionPerformed(e);
        }
});
loadPersona_button.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent e) {
                loadPersona_buttonActionPerformed(e);
        }
});
exit_button.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent e) {
                exit_buttonActionPerformed(e);
        }
});
```

```
                getCertificate_button.addActionListener(new java.awt.event.ActionListener() {
                        public void actionPerformed(java.awt.event.ActionEvent e) {
                                getCertificate_buttonActionPerformed(e);
                        }
                });

                addWindowListener(new java.awt.event.WindowAdapter() {
                        public void windowClosing(java.awt.event.WindowEvent e) {
                                thisWindowClosing(e);
                        }
                });

// END GENERATED CODE
        }

        private boolean mShown = false;

        public void addNotify() {
                super.addNotify();

                if (mShown)
                        return;

                // resize frame to account for menubar
                JMenuBar jMenuBar = getJMenuBar();
                if (jMenuBar != null) {
                        int jMenuBarHeight = jMenuBar.getPreferredSize().height;
                        Dimension dimension = getSize();
                        dimension.height += jMenuBarHeight;
                        setSize(dimension);

                        // move down components in layered pane
                        Component[] components =
getLayeredPane().getComponentsInLayer(JLayeredPane.DEFAULT_LAYER.intValue());
```

```
                    for (int i = 0; i < components.length; i++) {
                        Point location = components[i].getLocation();
                        location.move(location.x, location.y + jMenuBarHeight);
                        components[i].setLocation(location);
                    }
                }

                mShown = true;
            }

            // Close the window when the close box is clicked
            void thisWindowClosing(java.awt.event.WindowEvent e) {
                setVisible(false);
                dispose();
                System.exit(0);
            }

            public void exit_buttonActionPerformed(java.awt.event.ActionEvent e) {
                setVisible(false);
                dispose();
                System.exit(0);
            }

            public void createPersona_buttonActionPerformed(java.awt.event.ActionEvent e) {
                try{
                    createPersona_Frame frame = new createPersona_Frame();
                    frame.initComponents();
                    frame.setVisible(true);
                }catch (Exception ex) {
                    ex.printStackTrace();
                }
            }

            public void loadPersona_buttonActionPerformed(java.awt.event.ActionEvent e) {
/*              try{
```

```
                    JFileChooser d;
                    JFrame parent;

                    d = new JFileChooser();
                    d.setCurrentDirectory(new File("."));
                    int result = d.showOpenDialog(frame);
                    String filename = d.getSelectedFile().getName();
                    System.out.println(filename);
                    //frame.initComponents();
                    //frame.setVisible(true);
                }catch (Exception ex) {
                    ex.printStackTrace();
                }
*/

                try{
                    loadPersona_Frame frame = new loadPersona_Frame();
                    frame.initComponents();
                    frame.setVisible(true);
                }catch(Exception ex){
                    ex.printStackTrace();
                }

        }

        public void getCertificate_buttonActionPerformed(java.awt.event.ActionEvent e) {
                try{
                    signMyCertificate_Frame frame = new signMyCertificate_Frame();
                    frame.initComponents();
                    frame.setVisible(true);
                }catch (Exception ex) {
                    ex.printStackTrace();
                }
        }
```

}

11.3.3 signMyCertificate_Frame.java

```
/**
 * signMyCertificate_Frame.java
 *
 * Description:
 * @author                              subramma
 * @version
 */

package persapp;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import org.apache.axis.encoding.XMLType;
import org.apache.axis.utils.Options;
import javax.xml.rpc.ParameterMode ;
import javax.xml.namespace.QName;


import java.io.*;
import java.util.*;
import java.security.*;
import java.security.interfaces.*;
import java.math.*;
import org.bouncycastle.jce.*;
import java.security.cert.*;

import org.bouncycastle.asn1.pkcs.*;


public class signMyCertificate_Frame extends javax.swing.JFrame {

        // member declarations
```

```
java.awt.Label enterDetail_label = new java.awt.Label();
java.awt.Label name_label = new java.awt.Label();
java.awt.Label age_label = new java.awt.Label();
java.awt.Label address_label = new java.awt.Label();
static java.awt.TextField name_textfield = new java.awt.TextField();
java.awt.TextField age_textfield = new java.awt.TextField();
java.awt.TextField address_textfield = new java.awt.TextField();
java.awt.Label password_label = new java.awt.Label();
java.awt.TextField password_textfield = new java.awt.TextField();
java.awt.Button submit_button = new java.awt.Button();
java.awt.Button clear_button = new java.awt.Button();
java.awt.Button cancel_button = new java.awt.Button();

public String host =startup_Frame.serveraddress_textfield.getText() + ":";
String servicepath = "/axis/ArchiveService.jws";

String port = startup_Frame.serverport_textfield.getText();

String endpoint = host + port + servicepath;
String method = "signmycertificate";
String ret = null;
String crtfile = null;
String user_name = null;
String filecontent = null;
String filename = null;
String message = null;



public void signMyCertificate_Frame() {

}

public void initComponents() throws Exception {
        // the following code sets the frame's initial state
        enterDetail_label.setVisible(true);
        enterDetail_label.setAlignment(java.awt.Label.CENTER);
        enterDetail_label.setLocation(new java.awt.Point(60, 40));
        enterDetail_label.setText("Enter the name");
        enterDetail_label.setSize(new java.awt.Dimension(220, 40));
```

```
name_label.setVisible(true);
name_label.setLocation(new java.awt.Point(30, 110));
name_label.setText("Name");
name_label.setSize(new java.awt.Dimension(90, 30));
password_label.setVisible(true);
password_label.setLocation(new java.awt.Point(32, 164));
password_label.setText("Password");
password_label.setSize(new java.awt.Dimension(52, 18));

name_textfield.setLocation(new java.awt.Point(142, 119));
name_textfield.setVisible(true);
name_textfield.setSize(new java.awt.Dimension(143, 25));
password_textfield.setLocation(new java.awt.Point(144, 164));
password_textfield.setVisible(true);
password_textfield.setSize(new java.awt.Dimension(139, 19));


submit_button.setVisible(true);
submit_button.setLabel("Submit");
submit_button.setLocation(new java.awt.Point(110, 276));
submit_button.setSize(new java.awt.Dimension(87, 34));
clear_button.setVisible(true);
clear_button.setLabel("Clear");
clear_button.setLocation(new java.awt.Point(239, 274));
clear_button.setSize(new java.awt.Dimension(83, 36));
cancel_button.setVisible(true);
cancel_button.setLabel("Cancel");
cancel_button.setLocation(new java.awt.Point(164, 332));
cancel_button.setSize(new java.awt.Dimension(103, 39));
setLocation(new java.awt.Point(0, 0));
getContentPane().setLayout(null);
setTitle("Sign My Certificate");

getContentPane().add(enterDetail_label);
getContentPane().add(name_label);
getContentPane().add(name_textfield);
getContentPane().add(password_label);
getContentPane().add(password_textfield);
getContentPane().add(submit_button);
getContentPane().add(clear_button);
```

```
		getContentPane().add(cancel_button);

		//to mask the characters typed in as password
		password_textfield.setEchoCharacter('*');

		setSize(new java.awt.Dimension(358, 477));


		// event handling
		clear_button.addActionListener(new java.awt.event.ActionListener() {
			public void actionPerformed(java.awt.event.ActionEvent e) {
				clear_buttonActionPerformed(e);
			}
		});
		cancel_button.addActionListener(new java.awt.event.ActionListener() {
			public void actionPerformed(java.awt.event.ActionEvent e) {
				cancel_buttonActionPerformed(e);
			}
		});
		addWindowListener(new java.awt.event.WindowAdapter() {
			public void windowClosing(java.awt.event.WindowEvent e) {
				thisWindowClosing(e);
			}
		});

		submit_button.addActionListener(new java.awt.event.ActionListener() {
			public void actionPerformed(java.awt.event.ActionEvent e) {
				submit_buttonActionPerformed(e);
			}
		});

	}

	private boolean mShown = false;

	public void addNotify() {
		super.addNotify();

		if (mShown)
			return;
```

```
                // resize frame to account for menubar
                JMenuBar jMenuBar = getJMenuBar();
                if (jMenuBar != null) {
                        int jMenuBarHeight = jMenuBar.getPreferredSize().height;
                        Dimension dimension = getSize();
                        dimension.height += jMenuBarHeight;
                        setSize(dimension);

                        // move down components in layered pane
                        Component[] components = getLayeredPane().getComponentsInLayer(JLayeredPane.DEFAULT_LAYER.intValue());
                        for (int i = 0; i < components.length; i++) {
                                Point location = components[i].getLocation();
                                location.move(location.x, location.y + jMenuBarHeight);
                                components[i].setLocation(location);
                        }
                }

                mShown = true;
        }

        // Close the window when the close box is clicked
        void thisWindowClosing(java.awt.event.WindowEvent e) {
                setVisible(false);
                dispose();
                System.exit(0);
        }

        public void cancel_buttonActionPerformed(java.awt.event.ActionEvent e) {
                setVisible(false);
                dispose();
        }

        public void clear_buttonActionPerformed(java.awt.event.ActionEvent e) {
                name_textfield.setText("");
                age_textfield.setText("");
                address_textfield.setText("");
        }
```

```
        public void submit_buttonActionPerformed(java.awt.event.ActionEvent e) {

            if (name_textfield.getText().equals(""))
            {
                                        message = "Error! Name field cannot be blank";

                                           MessageBox box = new MessageBox();
                                        box.setTitle("Error!!");
                                           box.addChoice("Ok");
                                           box.setCloseWindowCommand("Close");
                                        box.ask(message);
                                        return;

                    }

    if (password_textfield.getText().equals(""))
    {
                                        message = "Error! Password field cannot be blank";

                                           MessageBox box = new MessageBox();
                                        box.setTitle("Error!!");
                                           box.addChoice("Ok");
                                           box.setCloseWindowCommand("Close");
                                        box.ask(message);
                                        return;

                    }



                    CreateMyCertificate cmc = new CreateMyCertificate();
                    user_name = name_textfield.getText();
                    try{
                                crtfile = cmc.createmycrt(user_name);
                    }catch(Exception ex){
                                ex.printStackTrace();

                    }

                    ArchiveClient ac = new ArchiveClient();
```

```
ac.storeFile(crtfile,"certificates");

try{
        /*below is the routine to encrypt the private key with a user entered password*/

        String password = "";
        String keystore_name = "C:\\java\\Persona\\persapp\\certificates\\persona.ks";

        KeyStore ks = KeyStore.getInstance( KeyStore.getDefaultType());

        ks.load( null, password.toCharArray());

        BufferedReader reader = new BufferedReader( new InputStreamReader( System.in));
        ks = cmc.loadKeyStore( keystore_name, password);

        //get the private key corresponding to user_name - see KeyPairRetrieve.java
        KeyPairRetrieve kpr = new KeyPairRetrieve();

        KeyPair kp = kpr.getPrivateKey(ks,user_name,password.toCharArray());
        //System.out.println(kp.getPrivate().toString());

        //write the private key to a file temporarily
        String filepath = "C:\\java\\Persona\\persapp\\private_keys\\";
        FileOutputStream prvkeyFOS = new FileOutputStream(filepath + user_name + ".pem");
        OutputStreamWriter prvkeyOSW = new OutputStreamWriter( prvkeyFOS );
        prvkeyOSW.write(kp.getPrivate().toString());
        prvkeyOSW.close();

        //encrypt the file by user entered password

        PasswordEncryptOrDecrypt(filepath+user_name + ".pem","e",password_textfield.getText());

        message = "Certificate generated successfully!"+"\n"+
                                                "Private key encrypted located at " +
                                                        filepath;

   MessageBox box = new MessageBox();
box.setTitle("Save successful!!");
        box.useImageCanvas("Happy_Face.gif");
   box.addChoice("Ok");
```

```
                box.setCloseWindowCommand("Close");
            box.ask(message);

                    boolean success = (new File(filepath + user_name + ".pem")).delete();
                if (!success) {
                        // Deletion failed
                }


            }catch(Exception ex){
                        ex.printStackTrace();
            }

            setVisible(false);
            dispose();

    }

     public static void PasswordEncryptOrDecrypt(String filename,String mode,String user_password ) {

              try {

                        FileInputStream fis = new FileInputStream( filename );
                        DataInputStream dis = new DataInputStream( fis );

                        FileOutputStream fos = new FileOutputStream( filename + "." + mode );
                        DataOutputStream dos = new DataOutputStream( fos );

                        PasswordBasedEncryptor pbe = new PasswordBasedEncryptor( user_password );

                        if ( mode.equals("e") )
                                pbe.encrypt( dis, dos );
                        else if ( mode.equals("d") )
                                pbe.decrypt( dis, dos );
                        else
                                System.out.println("Error, '" + mode + "' is invalid");

                        dos.flush();
                        fos.flush();
                        fos.close();
```

```
                    fis.close();
                } catch (Exception ex) {
                    ex.printStackTrace();
                }

        }




}


//this class and its methods generates the certificate
class CreateMyCertificate
{

        String filename = null;
        String keystore_name = null;
        String signed_filename = null;
        public static KeyPairGenerator kg;


        public void CreateMyCertificate(){}


        public String createmycrt(String yourName) throws Exception {


                Security.addProvider( new org.bouncycastle.jce.provider.BouncyCastleProvider());
                SecureRandom random = SecureRandom.getInstance("SHA1PRNG");

                String password = "";
                keystore_name = "C:\\java\\Persona\\persapp\\certificates\\persona.ks";

                KeyStore ks = KeyStore.getInstance( KeyStore.getDefaultType());

                        ks.load( null, password.toCharArray());
```

```
BufferedReader reader = new BufferedReader( new InputStreamReader( System.in));

ks = loadKeyStore( keystore_name, password);

KeyPair kp = createKeyPair();

java.security.cert.Certificate cert = createCertificate( "C=US, O=OSU, OU=CS-Persona Primary Certificate", kp.getPrivate(),

"C=US, O=OSU, OU=CS-Persona Primary Certificate", kp.getPublic());

java.security.cert.Certificate[]certsca = new java.security.cert.Certificate[1];
certsca[0] = cert;

addKeyEntry( ks, "ca", kp.getPrivate(), password, certsca);

PrivateKey caPrivKey = (PrivateKey)ks.getKey("ca",password.toCharArray());
java.security.cert.X509Certificate cacert = (java.security.cert.X509Certificate)ks.getCertificate("ca");

KeyPair owner = createKeyPair();
java.security.cert.Certificate certu = createCertificate( (X509Principal)cacert.getSubjectDN(), caPrivKey,

"CN=" + yourName + ", C=US, O=OSU, OU=CS-Persona Primary Certificate", owner.getPublic());

java.security.cert.Certificate[]certs = new java.security.cert.Certificate[1];
certs[0] = certu;

addKeyEntry( ks, yourName, caPrivKey, password, certs);

saveKeyStore( ks, keystore_name, password);
```

```
                                java.security.cert.X509Certificate certc = (X509Certificate)ks.getCertificate( yourName);

                                signed_filename = "C:\\java\\Persona\\persapp\\certificates\\signed\\" + yourName +
".crt";

                                FileOutputStream fos = new FileOutputStream(signed_filename);

                                saveCertificate( certc, fos);

                                fos.flush();
                                fos.close();

                                //System.out.println("Mahesh generated private key: " + owner.getPrivate().toString());



            //System.out.println( ks.getCertificate(yourName));
         //return ks.getCertificate(yourName).toString();
        return signed_filename;
}


         /**
          * Load Keystore
          */
         public static KeyStore loadKeyStore( String fname, String pwd) throws Exception {
                  KeyStore ks = KeyStore.getInstance( KeyStore.getDefaultType());
                  FileInputStream fis = new FileInputStream( fname);
                  ks.load( fis, pwd.toCharArray());
                  fis.close();

                  return ks;
         }

         /**
          * Store current keystore
          */
         private static void saveKeyStore( KeyStore ks, String fname, String pwd) throws Exception  {
                  FileOutputStream fos = new FileOutputStream( fname);
```

```
                ks.store( fos, pwd.toCharArray());
                fos.close();
        }

        /**
         * Create key pair
         */
        public static KeyPair createKeyPair() throws Exception  {
                kg = KeyPairGenerator.getInstance("RSA");
                return kg.generateKeyPair();
        }

        /**
         * Create self signed certificate
         */
        private static java.security.cert.Certificate createCertificate( X509Principal issuer, PrivateKey issuer_priv,
                                                                                                                        String
ownerDN, PublicKey owner_pub) throws Exception  {
                SecureRandom random = SecureRandom.getInstance("SHA1PRNG");

                X509V3CertificateGenerator certGen = new X509V3CertificateGenerator();
                certGen.setSerialNumber( BigInteger.valueOf( random.nextLong()));
                certGen.setIssuerDN( issuer);
                certGen.setNotBefore( new Date( System.currentTimeMillis() - 1000L * 60 * 60 * 24));
                certGen.setNotAfter( new Date( System.currentTimeMillis() + (1000L * 60 * 60 * 24 * 365)));

                certGen.setSubjectDN( new X509Principal( ownerDN));
                certGen.setPublicKey( owner_pub);
                certGen.setSignatureAlgorithm( "MD5WithRSAEncryption");
                X509Certificate cert = certGen.generateX509Certificate( issuer_priv);

                return cert;
        }

        /**
         * Create/sign certificate
         */
        private static java.security.cert.Certificate createCertificate( String issuerDN, PrivateKey issuer_priv,
                                                                                                                        String
ownerDN, PublicKey owner_pub) throws Exception  {
```

```
                return createCertificate( new X509Principal( issuerDN), issuer_priv,
                                                                ownerDN, owner_pub);
        }

        /**
         * save private / public key
         */
        private static void addKeyEntry( KeyStore ks, String alias, PrivateKey priv, String pwd, java.security.cert.Certificate[]
certs) throws Exception  {
                ks.setKeyEntry( alias, priv, pwd.toCharArray(), certs);
        }
        /**
         * Save Certificate
         */
        private static void saveCertificate( java.security.cert.Certificate cert, OutputStream out) throws Exception  {
                out.write( cert.getEncoded());
                out.flush();
        }
}
```

11.3.4 createPersona_Frame.java

```
/**
 * createPersona_Frame.java
 *
 * Description:
 * @author                          subramma
 * @version
 */

package persapp;

import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import org.apache.axis.encoding.XMLType;
import org.apache.axis.utils.Options;
import javax.xml.rpc.ParameterMode ;
```

```
import javax.xml.namespace.QName;

import java.io.*;
import java.util.*;
import java.security.*;
import java.security.interfaces.*;
import java.math.*;
import org.bouncycastle.jce.*;
import java.security.cert.*;

import org.bouncycastle.asn1.pkcs.*;

/*import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.BadPaddingException;
import java.security.SignatureException;
import java.security.NoSuchAlgorithmException;
import java.security.InvalidKeyException;
import java.security.InvalidAlgorithmParameterException;*/



public class createPersona_Frame extends javax.swing.JFrame {

        // member declarations
        java.awt.Label enterDetail_label = new java.awt.Label();
        java.awt.Label name_label = new java.awt.Label();
        java.awt.Label age_label = new java.awt.Label();
        java.awt.Label address_label = new java.awt.Label();
        java.awt.TextField name_textfield = new java.awt.TextField();
        java.awt.TextField age_textfield = new java.awt.TextField();
        java.awt.TextField address_textfield = new java.awt.TextField();
        java.awt.Button submit_button = new java.awt.Button();
        java.awt.Button clear_button = new java.awt.Button();
        java.awt.Button cancel_button = new java.awt.Button();

        public String host =startup_Frame.serveraddress_textfield.getText() + ":";
        String servicepath = "/axis/WritePersona.jws";
```

```
String port = startup_Frame.serverport_textfield.getText();

String endpoint = host + port + servicepath;
String method = "writepersona";
String ret = null;
String op1 = null;
String filepath = null;
String filecopypath = null;
String filename = null;
String filecopyname = null;
String xform = "RSA/NONE/PKCS1PADDING";
String message;

PublicKey pubk;



public createPersona_Frame() {

}

public void initComponents() throws Exception {
        // the following code sets the frame's initial state
        enterDetail_label.setVisible(true);
        enterDetail_label.setAlignment(java.awt.Label.CENTER);
        enterDetail_label.setLocation(new java.awt.Point(60, 40));
        enterDetail_label.setText("Enter your details");
        enterDetail_label.setSize(new java.awt.Dimension(220, 40));
        name_label.setVisible(true);
        name_label.setLocation(new java.awt.Point(30, 110));
        name_label.setText("Name");
        name_label.setSize(new java.awt.Dimension(90, 30));
        age_label.setVisible(true);
        age_label.setLocation(new java.awt.Point(32, 164));
        age_label.setText("Age");
        age_label.setSize(new java.awt.Dimension(52, 18));
        address_label.setVisible(true);
        address_label.setLocation(new java.awt.Point(32, 205));
        address_label.setText("Address");
        address_label.setSize(new java.awt.Dimension(76, 29));
        name_textfield.setLocation(new java.awt.Point(142, 119));
```

```
name_textfield.setVisible(true);
name_textfield.setSize(new java.awt.Dimension(143, 25));
age_textfield.setLocation(new java.awt.Point(144, 164));
age_textfield.setVisible(true);
age_textfield.setSize(new java.awt.Dimension(139, 19));
address_textfield.setLocation(new java.awt.Point(145, 208));
address_textfield.setVisible(true);
address_textfield.setSize(new java.awt.Dimension(139, 17));
submit_button.setVisible(true);
submit_button.setLabel("Submit");
submit_button.setLocation(new java.awt.Point(110, 276));
submit_button.setSize(new java.awt.Dimension(87, 34));
clear_button.setVisible(true);
clear_button.setLabel("Clear");
clear_button.setLocation(new java.awt.Point(239, 274));
clear_button.setSize(new java.awt.Dimension(83, 36));
cancel_button.setVisible(true);
cancel_button.setLabel("Cancel");
cancel_button.setLocation(new java.awt.Point(164, 332));
cancel_button.setSize(new java.awt.Dimension(103, 39));
setLocation(new java.awt.Point(0, 0));
getContentPane().setLayout(null);
setTitle("persapp.createPersona_Frame");

getContentPane().add(enterDetail_label);
getContentPane().add(name_label);
getContentPane().add(age_label);
getContentPane().add(address_label);
getContentPane().add(name_textfield);
getContentPane().add(age_textfield);
getContentPane().add(address_textfield);
getContentPane().add(submit_button);
getContentPane().add(clear_button);
getContentPane().add(cancel_button);

setSize(new java.awt.Dimension(358, 477));



// event handling
clear_button.addActionListener(new java.awt.event.ActionListener() {
```

```
                public void actionPerformed(java.awt.event.ActionEvent e) {
                        clear_buttonActionPerformed(e);
                }
        });
        cancel_button.addActionListener(new java.awt.event.ActionListener() {
                public void actionPerformed(java.awt.event.ActionEvent e) {
                        cancel_buttonActionPerformed(e);
                }
        });
        addWindowListener(new java.awt.event.WindowAdapter() {
                public void windowClosing(java.awt.event.WindowEvent e) {
                        thisWindowClosing(e);
                }
        });

        submit_button.addActionListener(new java.awt.event.ActionListener() {
                public void actionPerformed(java.awt.event.ActionEvent e) {
                        submit_buttonActionPerformed(e);
                }
        });

}

private boolean mShown = false;

public void addNotify() {
        super.addNotify();

        if (mShown)
                return;

        // resize frame to account for menubar
        JMenuBar jMenuBar = getJMenuBar();
        if (jMenuBar != null) {
                int jMenuBarHeight = jMenuBar.getPreferredSize().height;
                Dimension dimension = getSize();
                dimension.height += jMenuBarHeight;
                setSize(dimension);

                // move down components in layered pane
```

```
                    Component[] components = getLayeredPane().getComponentsInLayer(JLayeredPane.DEFAULT_LAYER.intValue());
                    for (int i = 0; i < components.length; i++) {
                        Point location = components[i].getLocation();
                        location.move(location.x, location.y + jMenuBarHeight);
                        components[i].setLocation(location);
                    }
                }

                mShown = true;
            }

            // Close the window when the close box is clicked
            void thisWindowClosing(java.awt.event.WindowEvent e) {
                setVisible(false);
                dispose();
                System.exit(0);
            }

            public void cancel_buttonActionPerformed(java.awt.event.ActionEvent e) {
                setVisible(false);
                dispose();
            }

            public void clear_buttonActionPerformed(java.awt.event.ActionEvent e) {
                name_textfield.setText("");
                age_textfield.setText("");
                address_textfield.setText("");
            }

            public void submit_buttonActionPerformed(java.awt.event.ActionEvent e) {
/*              try{

                    Service service = new Service();
                    Call call = (Call)service.createCall();
                    call.setTargetEndpointAddress(new java.net.URL(endpoint));
                    call.setOperationName(method);
                    op1 = name_textfield.getText();
                    filecontent = name_textfield.getText() + "$" + age_textfield.getText() + "$" + address_textfield.getText();
```

```
    call.addParameter("op1", XMLType.XSD_STRING, ParameterMode.IN);
    call.addParameter("filecontent",XMLType.XSD_STRING, ParameterMode.IN);


call.setReturnType(XMLType.XSD_STRING);
    ret = (String) call.invoke(new Object [] {op1,filecontent});

    System.out.println(ret);  */

    if (name_textfield.getText().equals(""))
    {
                        message = "Error! Name field cannot be blank";

                          MessageBox box = new MessageBox();
                        box.setTitle("Error!!");
                          box.addChoice("Ok");
                          box.setCloseWindowCommand("Close");
                        box.ask(message);
                        return;

                }


                KeyPairRetrieve kpr = new KeyPairRetrieve();//retrieve public & private keys of user
                CreateMyCertificate cmc = new CreateMyCertificate(); //to access the keystore
                AsymmetricCipherGenerator acg = new AsymmetricCipherGenerator();

                String password = "";
                String keystore_name = "C:\\java\\Persona\\persapp\\certificates\\persona.ks";

                try{
                        KeyStore ks = KeyStore.getInstance( KeyStore.getDefaultType());
                        KeyPair kp = cmc.createKeyPair() ;

                        ks.load( null, password.toCharArray());

                        BufferedReader reader = new BufferedReader( new InputStreamReader( System.in));
                        ks = cmc.loadKeyStore( keystore_name, password);

                        kp = kpr.getPrivateKey(ks,name_textfield.getText(),password.toCharArray());
```

```
                pubk = kp.getPublic();

        filepath = "C:\\java\\Persona\\persapp\\personas\\";

                filename = name_textfield.getText() + ".xml";
                FileOutputStream fout = new FileOutputStream(filepath+ filename);
                OutputStreamWriter out = new OutputStreamWriter(fout);



                out.write("<?xml version=\"1.0\"?>\r\n");
                out.write("<Persona>\r\n");
                out.write("  <name index = \"" + ' ' + "\">");
                out.write(acg.encrypt(name_textfield.getText().getBytes(),pubk,xform).toString());
                out.write("</name>\r\n");

                out.write("  <age index = \"" + ' ' + "\">");
                out.write(acg.encrypt(age_textfield.getText().getBytes(),pubk,xform).toString());
                out.write("</age>\r\n");

                out.write("  <address index = \"" + ' ' + "\">");
                out.write(acg.encrypt(address_textfield.getText().getBytes(),pubk,xform).toString());
                out.write("</address>\r\n");
                out.write("</Persona>");

                out.close();


        filecopypath = "C:\\java\\Persona\\persapp\\pcopy\\";

                filecopyname = name_textfield.getText() + ".txt";
                FileOutputStream fout2 = new FileOutputStream(filecopypath+ filecopyname);
                OutputStreamWriter out2 = new OutputStreamWriter(fout2);

                //out2.write("<?xml version=\"1.0\"?>\r\n");
                //out2.write("<Persona>\r\n");
                //out2.write("  <name index = \"" + ' ' + "\">");
                out2.write(name_textfield.getText());
                out2.write("\n");
                //out2.write("</name>\r\n");
```

```
                    //out2.write("  <age index = \"" + ' ' + "\">");
                    out2.write(age_textfield.getText());
                    out2.write("\n");
                    //out2.write("</age>\r\n");

                    //out2.write("  <address index = \"" + ' ' + "\">");
                    out2.write(address_textfield.getText());

                    //out2.write("</address>\r\n");
                    //out2.write("</Persona>");

                    out2.close();


                }catch(Exception ex){}



        ArchiveClient ac = new ArchiveClient();
                    ac.storeFile(filepath + filename,"personas");

                    ac.storeFile(filecopypath + filecopyname,"pcopy");

                    setVisible(false);
                    dispose();


                    message = "Persona saved successfully!";

                MessageBox box = new MessageBox();
            box.setTitle("Save successful!!");
                    box.useImageCanvas("Happy_Face.gif");
                box.addChoice("Ok");
                box.setCloseWindowCommand("Close");
            box.ask(message);

    }
```

```
}
```

11.3.5 editPersona_Frame.java

```
/**
 * createPersona_Frame.java
 *
 * Description:
 * @author                              subramma
 * @version
 */

package persapp;

import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import org.apache.axis.encoding.XMLType;
import org.apache.axis.utils.Options;
import javax.xml.rpc.ParameterMode ;
import javax.xml.namespace.QName;

import java.io.*;
import java.util.*;
import java.security.*;
import java.security.interfaces.*;
import java.math.*;
import org.bouncycastle.jce.*;
import java.security.cert.*;

import org.bouncycastle.asn1.pkcs.*;

/*import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.BadPaddingException;
import java.security.SignatureException;
import java.security.NoSuchAlgorithmException;
import java.security.InvalidKeyException;
```

```
import java.security.InvalidAlgorithmParameterException;*/



public class editPersona_Frame extends javax.swing.JFrame {

        // member declarations
        java.awt.Label enterDetail_label = new java.awt.Label();
        java.awt.Label name_label = new java.awt.Label();
        java.awt.Label age_label = new java.awt.Label();
        java.awt.Label address_label = new java.awt.Label();
        java.awt.TextField name_textfield = new java.awt.TextField();
        java.awt.TextField age_textfield = new java.awt.TextField();
        java.awt.TextField address_textfield = new java.awt.TextField();
        java.awt.Button submit_button = new java.awt.Button();
        java.awt.Button clear_button = new java.awt.Button();
        java.awt.Button cancel_button = new java.awt.Button();

        public String host =startup_Frame.serveraddress_textfield.getText() + ":";
        String servicepath = "/axis/WritePersona.jws";

        String port = startup_Frame.serverport_textfield.getText();

        String endpoint = host + port + servicepath;
        String method = "writepersona";
        String ret = null;
        String op1 = null;
        String filepath = null;
        String filecopypath = null;
        String filename = null;
        String filecopyname = null;
        String xform = "RSA/NONE/PKCS1PADDING";

        String message;

        PublicKey pubk;

        String user_name,user_age,user_address;
```

```
public editPersona_Frame(String name,String age,String address) {
        user_name = name;
        user_age = age;
        user_address = address;

}

public void initComponents() throws Exception {
        // the following code sets the frame's initial state
        enterDetail_label.setVisible(true);
        enterDetail_label.setAlignment(java.awt.Label.CENTER);
        enterDetail_label.setLocation(new java.awt.Point(60, 40));
        enterDetail_label.setText("Edit your details below");
        enterDetail_label.setSize(new java.awt.Dimension(220, 40));
        name_label.setVisible(true);
        name_label.setLocation(new java.awt.Point(30, 110));
        name_label.setText("Name");
        name_label.setSize(new java.awt.Dimension(90, 30));
        age_label.setVisible(true);
        age_label.setLocation(new java.awt.Point(32, 164));
        age_label.setText("Age");
        age_label.setSize(new java.awt.Dimension(52, 18));
        address_label.setVisible(true);
        address_label.setLocation(new java.awt.Point(32, 205));
        address_label.setText("Address");
        address_label.setSize(new java.awt.Dimension(76, 29));
        name_textfield.setLocation(new java.awt.Point(142, 119));
        name_textfield.setVisible(true);
        name_textfield.setSize(new java.awt.Dimension(143, 25));
        name_textfield.setText(user_name);
        age_textfield.setLocation(new java.awt.Point(144, 164));
        age_textfield.setVisible(true);
        age_textfield.setSize(new java.awt.Dimension(139, 19));
        age_textfield.setText(user_age);
        address_textfield.setLocation(new java.awt.Point(145, 208));
        address_textfield.setVisible(true);
        address_textfield.setSize(new java.awt.Dimension(139, 17));
        address_textfield.setText(user_address);
        submit_button.setVisible(true);
```

```
submit_button.setLabel("Submit");
submit_button.setLocation(new java.awt.Point(110, 276));
submit_button.setSize(new java.awt.Dimension(87, 34));
clear_button.setVisible(true);
clear_button.setLabel("Clear");
clear_button.setLocation(new java.awt.Point(239, 274));
clear_button.setSize(new java.awt.Dimension(83, 36));
cancel_button.setVisible(true);
cancel_button.setLabel("Cancel");
cancel_button.setLocation(new java.awt.Point(164, 332));
cancel_button.setSize(new java.awt.Dimension(103, 39));
setLocation(new java.awt.Point(0, 0));
getContentPane().setLayout(null);
setTitle("Edit Persona");

getContentPane().add(enterDetail_label);
getContentPane().add(name_label);
getContentPane().add(age_label);
getContentPane().add(address_label);
getContentPane().add(name_textfield);
getContentPane().add(age_textfield);
getContentPane().add(address_textfield);
getContentPane().add(submit_button);
getContentPane().add(clear_button);
getContentPane().add(cancel_button);

setSize(new java.awt.Dimension(358, 477));



// event handling
clear_button.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent e) {
                clear_buttonActionPerformed(e);
        }
});
cancel_button.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent e) {
                cancel_buttonActionPerformed(e);
        }
});
```

```
		addWindowListener(new java.awt.event.WindowAdapter() {
			public void windowClosing(java.awt.event.WindowEvent e) {
				thisWindowClosing(e);
			}
		});

		submit_button.addActionListener(new java.awt.event.ActionListener() {
			public void actionPerformed(java.awt.event.ActionEvent e) {
				submit_buttonActionPerformed(e);
			}
		});

	}

	private boolean mShown = false;

	public void addNotify() {
		super.addNotify();

		if (mShown)
			return;

		// resize frame to account for menubar
		JMenuBar jMenuBar = getJMenuBar();
		if (jMenuBar != null) {
			int jMenuBarHeight = jMenuBar.getPreferredSize().height;
			Dimension dimension = getSize();
			dimension.height += jMenuBarHeight;
			setSize(dimension);

			// move down components in layered pane
			Component[] components = getLayeredPane().getComponentsInLayer(JLayeredPane.DEFAULT_LAYER.intValue());
			for (int i = 0; i < components.length; i++) {
				Point location = components[i].getLocation();
				location.move(location.x, location.y + jMenuBarHeight);
				components[i].setLocation(location);
			}
		}
```

```
                mShown = true;
        }

        // Close the window when the close box is clicked
        void thisWindowClosing(java.awt.event.WindowEvent e) {
                setVisible(false);
                dispose();
                System.exit(0);
        }

        public void cancel_buttonActionPerformed(java.awt.event.ActionEvent e) {
                setVisible(false);
                dispose();
        }

        public void clear_buttonActionPerformed(java.awt.event.ActionEvent e) {
                name_textfield.setText("");
                age_textfield.setText("");
                address_textfield.setText("");
        }

        public void submit_buttonActionPerformed(java.awt.event.ActionEvent e) {
/*              try{

                        Service service = new Service();
                        Call call = (Call)service.createCall();
                        call.setTargetEndpointAddress(new java.net.URL(endpoint));
                        call.setOperationName(method);
                        op1 = name_textfield.getText();
                        filecontent = name_textfield.getText() + "$" + age_textfield.getText() + "$" +
address_textfield.getText();
            call.addParameter("op1", XMLType.XSD_STRING, ParameterMode.IN);
            call.addParameter("filecontent",XMLType.XSD_STRING, ParameterMode.IN);


           call.setReturnType(XMLType.XSD_STRING);
            ret = (String) call.invoke(new Object [] {op1,filecontent});

            System.out.println(ret);  */
```

```
if (name_textfield.getText().equals(""))
{
                    message = "Error! Name field cannot be blank";

                      MessageBox box = new MessageBox();
                    box.setTitle("Error!!");
                      box.addChoice("Ok");
                      box.setCloseWindowCommand("Close");
                    box.ask(message);
                    return;

          }


          KeyPairRetrieve kpr = new KeyPairRetrieve();//retrieve public & private keys of user
          CreateMyCertificate cmc = new CreateMyCertificate(); //to access the keystore
          AsymmetricCipherGenerator acg = new AsymmetricCipherGenerator();

          String password = "";
          String keystore_name = "C:\\java\\Persona\\persapp\\certificates\\persona.ks";

          try{
                    KeyStore ks = KeyStore.getInstance( KeyStore.getDefaultType());
                    KeyPair kp = cmc.createKeyPair() ;

                    ks.load( null, password.toCharArray());

                    BufferedReader reader = new BufferedReader( new InputStreamReader( System.in));
                    ks = cmc.loadKeyStore( keystore_name, password);

                    kp = kpr.getPrivateKey(ks,name_textfield.getText(),password.toCharArray());
                    pubk = kp.getPublic();



          filepath = "C:\\java\\Persona\\persapp\\personas\\";

                    filename = name_textfield.getText() + ".xml";
                    FileOutputStream fout = new FileOutputStream(filepath+ filename);
                    OutputStreamWriter out = new OutputStreamWriter(fout);
```

```
out.write("<?xml version=\"1.0\"?>\r\n");
out.write("<Persona>\r\n");
out.write("  <name index = \"" + ' ' + "\">");
out.write(acg.encrypt(name_textfield.getText().getBytes(),pubk,xform).toString());
out.write("</name>\r\n");

out.write("  <age index = \"" + ' ' + "\">");
out.write(acg.encrypt(age_textfield.getText().getBytes(),pubk,xform).toString());
out.write("</age>\r\n");

out.write("  <address index = \"" + ' ' + "\">");
out.write(acg.encrypt(address_textfield.getText().getBytes(),pubk,xform).toString());
out.write("</address>\r\n");
out.write("</Persona>");

out.close();


filecopypath = "C:\\java\\Persona\\persapp\\pcopy\\";

filecopyname = name_textfield.getText() + ".txt";
FileOutputStream fout2 = new FileOutputStream(filecopypath+ filecopyname);
OutputStreamWriter out2 = new OutputStreamWriter(fout2);


//out2.write("<?xml version=\"1.0\"?>\r\n");
//out2.write("<Persona>\r\n");
//out2.write("  <name index = \"" + ' ' + "\">");
out2.write(name_textfield.getText());
out2.write("\n");
//out2.write("</name>\r\n");

//out2.write("  <age index = \"" + ' ' + "\">");
out2.write(age_textfield.getText());
out2.write("\n");
//out2.write("</age>\r\n");

//out2.write("  <address index = \"" + ' ' + "\">");
```

```
                                        out2.write(address_textfield.getText());

                                        //out2.write("</address>\r\n");
                                        //out2.write("</Persona>");

                                        out2.close();

                        }catch(Exception ex){}


                ArchiveClient ac = new ArchiveClient();
                                ac.storeFile(filepath + filename,"personas");


                                ac.storeFile(filecopypath+filecopyname,"pcopy");

                                setVisible(false);
                                dispose();


                                message = "Persona saved successfully!";

                        MessageBox box = new MessageBox();
                    box.setTitle("Save successful!!");
                                box.useImageCanvas("Happy_Face.gif");
                        box.addChoice("Ok");
                        box.setCloseWindowCommand("Close");
                    box.ask(message);


        }

}
```

11.3.6 ArchiveClient.java

```
package persapp;

 import java.io.*;
 import java.util.*;
 import java.net.*;
```

```
import org.apache.soap.*;
import org.apache.soap.util.mime.*;
import org.apache.soap.rpc.*;
import javax.activation.*;
import javax.mail.BodyPart;
import javax.mail.internet.MimeBodyPart;

class ArchiveClient {

  URL url;
  Call call = new Call();
  String urn = "urn:ArchiveService";

 public void ArchiveClient(){}

 public void storeFile(String filename,String path) {


  try {
    url = new URL("http://gsmpc01.library.oregonstate.edu:8080/axis/ArchiveService.jws");
    call.setTargetObjectURI(urn);
    call.setMethodName("StoreFile");
    call.setEncodingStyleURI(Constants.NS_URI_SOAP_ENC );
    Vector params = new Vector();
    File file = new File(filename);
    DataSource ds = new ByteArrayDataSource(new File(filename), null);
    DataHandler dh = new DataHandler(ds);
    params.addElement(new Parameter("filename", String.class, file.getName(), null));
    params.addElement(new Parameter("path",String.class,path,null));
   params.addElement(new Parameter("dh", javax.activation.DataHandler.class, dh, null));
    call.setParams(params);
    Response response = call.invoke(url, "");


  //FileOutputStream os3 = new FileOutputStream("C:\\java\\Persona\\persapp\\certificates\\signed\\"+filename);

          //  MimeBodyPart mbp = new MimeBodyPart();
    //mbp = (MimeBodyPart)response.getBodyPart(1);
    //DataHandler dh2 = mbp.getDataHandler();
    //System.out.println(dh2.getContentType());
```

```
     //Object o = dh2.getContent();
     //dh2.writeTo(os3);
     //os3.close();
     //dh2= call.invoke(url, "");

    /* if (!response.generatedFault()) {
       Parameter p = response.getReturnValue();
       System.out.println("Response = " + p.getValue());
     } else {
       Fault f = response.getFault();
       System.out.println("Fault: " + f.getFaultString());
     }     */
   } catch (Exception e) {
     System.out.println("Exception: " + e.getMessage());
   }
 }


}
```

11.3.7 AsymmetricCipherGenerator.java

```
/*
* author Mahesh
 */

package persapp;

import java.security.KeyPairGenerator;
import java.security.KeyPair;
import java.security.PublicKey;
import java.security.PrivateKey;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.BadPaddingException;
import java.security.SignatureException;
import java.security.NoSuchAlgorithmException;
import java.security.InvalidKeyException;
import java.security.InvalidAlgorithmParameterException;
```

```
import java.io.*;

import org.bouncycastle.jce.provider.BouncyCastleProvider;

public class AsymmetricCipherGenerator {
	public static byte[] encrypt(byte[] inpBytes, PublicKey key, String xform) throws
				NoSuchAlgorithmException, InvalidKeyException, IllegalBlockSizeException,
				NoSuchPaddingException, BadPaddingException, InvalidAlgorithmParameterException {
		Cipher cipher = Cipher.getInstance(xform);
		cipher.init(Cipher.ENCRYPT_MODE, key);
		return cipher.doFinal(inpBytes);
	}
	public static byte[] decrypt(byte[] inpBytes, PrivateKey key, String xform) throws
				NoSuchAlgorithmException, InvalidKeyException, IllegalBlockSizeException,
				NoSuchPaddingException, BadPaddingException, InvalidAlgorithmParameterException{
		Cipher cipher = Cipher.getInstance(xform);
		cipher.init(Cipher.DECRYPT_MODE, key);
		return cipher.doFinal(inpBytes);
	}

}
```

11.3.8 BytesToString.java

```
package persapp;

import java.io.*;



public class BytesToString {

	/**
		Converts the array of bytes to a string. Non-ascii characters
		are converted to ".".

		@param b The array of bytes to convert.
		@return The bytes converted into a string.
	*/
	public static String conv(byte[] b) {
		return conv(b, b.length);
	}
```

```
	/**
		Converts the array of bytes to a string. Non-ascii characters
		are converted to ".". Only the first len bytes are used.

		@param b The array of bytes to convert.
		@param len Only uses bytes b[0]..b[len-1].
		@return The bytes converted into a string.
	*/
	public static String conv(byte[] b, int len) {
		byte[] c = new byte[len];
		for(int i=0; i<len; i++) {
			byte n = b[i];
			if ((n < 32) || (n > 128))
				n = 65;
			c[i] = n;
		}
		return new String(c);
	}
}
```

11.3.9 PasswordBasedEncryptor.java

```
package persapp;

import java.io.*;
import java.security.*;
import javax.crypto.*;
import javax.crypto.spec.*;


public class PasswordBasedEncryptor {

	byte[] salt = { (byte)0xd4, (byte)0xa3, (byte)0xff, (byte)0x9e,
					(byte)0x12, (byte)0xc7, (byte)0xd0, (byte)0x84 };

	Cipher c;
	PBEParameterSpec paramSpec;
	SecretKey passwordKey;
```

```
	public PasswordBasedEncryptor(String password) {
		try {
			paramSpec = new PBEParameterSpec( salt, 20 );
			PBEKeySpec keySpec = new PBEKeySpec( password.toCharArray() );
			SecretKeyFactory kf = SecretKeyFactory.getInstance("PBEWithMD5AndDES");
			passwordKey = kf.generateSecret( keySpec );
			c = Cipher.getInstance("PBEWithMD5AndDES");
		} catch (Exception e) {
			System.out.println("PBEncryptor() caught exception '" + e +
'' with message '' + e.getMessage() + ''.");
			e.printStackTrace();
		}
	}

	public void encrypt(DataInputStream dis, DataOutputStream dos)
		throws Exception {

		c.init(Cipher.ENCRYPT_MODE, passwordKey, paramSpec);
		CipherInputStream cis = new CipherInputStream( dis, c );

		int i;
		while ( (i = cis.read()) >= 0 ) {
			dos.write(i);
		}
	}

	public void decrypt(DataInputStream dis, DataOutputStream dos)
		throws Exception {

		c.init(Cipher.DECRYPT_MODE, passwordKey, paramSpec);
		CipherInputStream cis = new CipherInputStream( dis, c );

		int i;
		while ( (i = cis.read()) >= 0 ) {
			dos.write(i);
		}

	}
```

```
}
```

11.3.10 MessageBox.java

```
package persapp;

import java.awt.*;
import java.awt.event.*;

//import ImageCanvas;



/**
 * Provides a reusable window that presents a message and
 * choice buttons to the user. A modal dialog is used.
 * Since a thread is used to set the dialog to visible,
 * when the client calls ask() it will not block.
 *
 *
 * @author Mahesh
 */
public class MessageBox implements Runnable,
    ActionListener, WindowListener, KeyListener {

//---------- Private Fields -----------------------------
private ActionListener listener;
private Dialog  dialog;
private String  closeWindowCommand = "CloseRequested";
private String  title;
private Frame   frame;
private boolean frameNotProvided;
private Panel   buttonPanel = new Panel();
private Canvas  imageCanvas;

//---------- Initialization -----------------------------
/**
 * This convenience constructor is used to delare the
 * listener that will be notified when a button is clicked.
 * The listener must implement ActionListener.
 */
```

```
public MessageBox(ActionListener listener) {
    this();
    this.listener = listener;
}
/**
 * This constructor is used for no listener, such as for
 * a simple okay dialog.
 */
public MessageBox() {
}
// Unit test. Shows only simple features.
public static void main(String args[]) {
    MessageBox box = new MessageBox();
    box.setTitle("Test MessageBox");
    box.useImageCanvas("LightBulb.gif");
    box.askYesNo("Tell me now.\nDo you like Java?");
}
//---------- Runnable Implementation ---------------------
/**
 * This prevents the caller from blocking on ask(), which
 * if this class is used on an awt event thread would
 * cause a deadlock.
 */
public void run() {
    dialog.setVisible(true);
}
//---------- ActionListener Implementation ---------------
public void actionPerformed(ActionEvent evt) {
    String command = evt.getActionCommand();
    dialog.setVisible(false);
    dialog.dispose();
    if (frameNotProvided) frame.dispose();
    if (listener != null) {
        listener.actionPerformed(evt);
    }
}
//---------- WindowListener Implementatons ---------------
public void windowClosing(WindowEvent evt) {
    // User clicked on X or chose Close selection
    fireCloseRequested();
```

```
}
public void windowClosed(WindowEvent evt) { }
public void windowDeiconified(WindowEvent evt) { }
public void windowIconified(WindowEvent evt) { }
public void windowOpened(WindowEvent evt) { }
public void windowActivated(WindowEvent evt) {  }
public void windowDeactivated(WindowEvent evt) { }

//---------- KeyListener Implementation ------------------
public void keyTyped(KeyEvent evt) { }
public void keyPressed(KeyEvent evt) {
   if (evt.getKeyCode() == KeyEvent.VK_ESCAPE) {
      fireCloseRequested();
   }
}
public void keyReleased(KeyEvent evt) { }

private void fireCloseRequested() {
   ActionEvent event = new ActionEvent(this,
      ActionEvent.ACTION_PERFORMED, closeWindowCommand);
   actionPerformed(event);
}
//---------- Public Methods -------------------------------
/**
 * This set the listener to be notified of button clicks
 * and WindowClosing events.
 */
public void setActionListener(ActionListener listener) {
   this.listener = listener;
}
public void setTitle(String title) {
   this.title = title;
}
/**
 * If a Frame is provided then it is used to instantiate
 * the modal Dialog. Otherwise a temporary Frame is used.
 * Providing a Frame will have the effect of putting the
 * focus back on that Frame when the MessageBox is closed
 * or a button is clicked.
 */
```

```
public void setFrame(Frame frame) { // Optional
    this.frame = frame;
}
/**
 * Sets the ActionCommand used in the ActionEvent when the
 * user attempts to close the window. The window may be
 * closed by clicking on "X", choosing Close from the
 * window menu, or pressing the Escape key. The default
 * command is "CloseRequested", which is just what a Close
 * choice button would probably have as a command.
 */
public void setCloseWindowCommand(String command) {
    closeWindowCommand = command;
}
/**
 * This is handy for providing a small image that will be
 * displayed to the left of the message.
 */
public void useImageCanvas(Canvas imageCanvas) {
    this.imageCanvas = imageCanvas;
}
/**
 * This loads the image from the specified @param fileName,
 * which must be in the same directory as this class.
 * For example @param fileName might be "LightBulb.gif".
 */
public void useImageCanvas(String fileName) {
    try {
        ImageCanvas imageCanvas = new ImageCanvas(MessageBox.class, fileName);
        useImageCanvas(imageCanvas);
    } catch(Exception ex) {
        print("MessageBox.helpfulHint() - Cannot load image " + fileName);
        ex.printStackTrace();
    }
}
/**
 * The @param label will be used for the button and the
 * @param command will be returned to the listener.
 */
public void addChoice(String label, String command) {
```

```
    Button button = new Button(label);
    button.setActionCommand(command);
    button.addActionListener(this);
    button.addKeyListener(this);
    buttonPanel.add(button);
}
/**
 * A convenience method that assumes the command is the
 * same as the label.
 */
public void addChoice(String label) {
    addChoice(label, label);
}
/**
 * One of the "ask" methods must be the last call when
 * using a MessageBox. This is the simplest "ask" method.
 * It presents the provided @param message.
 */
public void ask(String message) {
    if (frame == null) {
        frame = new Frame();
        frameNotProvided = true;
    } else {
        frameNotProvided = false;
    }
    dialog = new Dialog(frame, true); // Modal
    dialog.addWindowListener(this);
    dialog.addKeyListener(this);
    dialog.setTitle(title);
    dialog.setLayout(new BorderLayout(5, 5));

    Panel messagePanel = createMultiLinePanel(message);
    if (imageCanvas == null) {
        dialog.add("Center", messagePanel);
    } else {
        Panel centerPanel = new Panel();
        centerPanel.add(imageCanvas);
        centerPanel.add(messagePanel);
        dialog.add("Center", centerPanel);
    }
```

```
    dialog.add("South", buttonPanel);
    dialog.pack();
    enforceMinimumSize(dialog, 200, 100);
    centerWindow(dialog);
    Toolkit.getDefaultToolkit().beep();

    // Start a new thread to show the dialog
    Thread thread = new Thread(this);
    thread.start();
}
/**
 * Same as ask(String message) except adds an "Okay" button.
 */
public void askOkay(String message) {
    addChoice("Okay");
    ask(message);
}
/**
 * Same as ask(String message) except adds "Yes" and "No"
 * buttons.
 */
public void askYesNo(String message) {
    addChoice("Yes");
    addChoice("No");
    ask(message);
}
//---------- Private Methods -----------------------------
private Panel createMultiLinePanel(String message) {
    Panel mainPanel = new Panel();
    GridBagLayout gbLayout = new GridBagLayout();
    mainPanel.setLayout(gbLayout);
    addMultilineString(message, mainPanel);
    return mainPanel;
}
// There are a variaty of ways to do this....
private void addMultilineString(String message,
        Container container) {

    GridBagConstraints constraints = getDefaultConstraints();
    constraints.gridwidth = GridBagConstraints.REMAINDER;
```

```
        // Insets() args are top, left, bottom, right
    constraints.insets   = new Insets(0,0,0,0);
    GridBagLayout gbLayout = (GridBagLayout)container.getLayout();

    while (message.length() > 0) {
        int newLineIndex = message.indexOf('\n');
        String line;
        if (newLineIndex >= 0) {
            line = message.substring(0, newLineIndex);
            message = message.substring(newLineIndex + 1);
        } else {
            line = message;
            message = "";
        }
        Label label = new Label(line);
        gbLayout.setConstraints(label, constraints);
        container.add(label);
    }
}
private GridBagConstraints getDefaultConstraints() {
    GridBagConstraints constraints = new GridBagConstraints();
    constraints.weightx  = 1.0;
    constraints.weighty  = 1.0;
    constraints.gridheight = 1; // One row high
        // Insets() args are top, left, bottom, right
    constraints.insets   = new Insets(4,4,4,4);
        // fill of NONE means do not change size
    constraints.fill = GridBagConstraints.NONE;
        // WEST means align left
    constraints.anchor = GridBagConstraints.WEST;

    return constraints;
}
private void centerWindow(Window win) {
    Dimension screenDim = Toolkit.getDefaultToolkit().getScreenSize();
    // If larger than screen, reduce window width or height
    if (screenDim.width < win.getSize().width) {
        win.setSize(screenDim.width, win.getSize().height);
    }
    if (screenDim.height < win.getSize().height) {
```

```
        win.setSize(win.getSize().width, screenDim.height);
    }
    // Center Frame, Dialogue or Window on screen
    int x = (screenDim.width - win.getSize().width) / 2;
    int y = (screenDim.height - win.getSize().height) / 2;
    win.setLocation(x, y);
}
private void enforceMinimumSize(Component comp,
        int minWidth, int minHeight) {
    if (comp.getSize().width < minWidth) {
        comp.setSize(minWidth, comp.getSize().height);
    }
    if (comp.getSize().height < minHeight) {
        comp.setSize(comp.getSize().width, minHeight);
    }
}
//--- Std
private static void print(String text) {
    System.out.println(text);
}

} // End class
```

11.3.11 ImageCanvas.java

```
package persapp;

import java.awt.*;
import java.awt.event.*;

//import ImageCanvas;


/**
 * Provides a reusable window that presents a message and
 * choice buttons to the user. A modal dialog is used.
 * Since a thread is used to set the dialog to visible,
 * when the client calls ask() it will not block.
 *
```

```
 *
 * @author Mahesh
 */
public class MessageBox implements Runnable,
    ActionListener, WindowListener, KeyListener {

//---------- Private Fields ------------------------------
private ActionListener listener;
private Dialog  dialog;
private String  closeWindowCommand = "CloseRequested";
private String  title;
private Frame   frame;
private boolean frameNotProvided;
private Panel   buttonPanel = new Panel();
private Canvas  imageCanvas;

//---------- Initialization ------------------------------
/**
 * This convenience constructor is used to delare the
 * listener that will be notified when a button is clicked.
 * The listener must implement ActionListener.
 */
public MessageBox(ActionListener listener) {
    this();
    this.listener = listener;
}
/**
 * This constructor is used for no listener, such as for
 * a simple okay dialog.
 */
public MessageBox() {
}
// Unit test. Shows only simple features.
public static void main(String args[]) {
    MessageBox box = new MessageBox();
    box.setTitle("Test MessageBox");
    box.useImageCanvas("LightBulb.gif");
    box.askYesNo("Tell me now.\nDo you like Java?");
}
//---------- Runnable Implementation ---------------------
```

```
/**
 * This prevents the caller from blocking on ask(), which
 * if this class is used on an awt event thread would
 * cause a deadlock.
 */
public void run() {
    dialog.setVisible(true);
}
//---------- ActionListener Implementation ---------------
public void actionPerformed(ActionEvent evt) {
    String command = evt.getActionCommand();
    dialog.setVisible(false);
    dialog.dispose();
    if (frameNotProvided) frame.dispose();
    if (listener != null) {
        listener.actionPerformed(evt);
    }
}
//---------- WindowListener Implementatons ---------------
public void windowClosing(WindowEvent evt) {
    // User clicked on X or chose Close selection
    fireCloseRequested();
}
public void windowClosed(WindowEvent evt) { }
public void windowDeiconified(WindowEvent evt) { }
public void windowIconified(WindowEvent evt) { }
public void windowOpened(WindowEvent evt) { }
public void windowActivated(WindowEvent evt) {  }
public void windowDeactivated(WindowEvent evt) { }

//---------- KeyListener Implementation ------------------
public void keyTyped(KeyEvent evt) { }
public void keyPressed(KeyEvent evt) {
    if (evt.getKeyCode() == KeyEvent.VK_ESCAPE) {
        fireCloseRequested();
    }
}
public void keyReleased(KeyEvent evt) { }

private void fireCloseRequested() {
```

```
    ActionEvent event = new ActionEvent(this,
        ActionEvent.ACTION_PERFORMED, closeWindowCommand);
    actionPerformed(event);
}
//---------- Public Methods -------------------------------
/**
 * This set the listener to be notified of button clicks
 * and WindowClosing events.
 */
public void setActionListener(ActionListener listener) {
    this.listener = listener;
}
public void setTitle(String title) {
    this.title = title;
}
/**
 * If a Frame is provided then it is used to instantiate
 * the modal Dialog. Otherwise a temporary Frame is used.
 * Providing a Frame will have the effect of putting the
 * focus back on that Frame when the MessageBox is closed
 * or a button is clicked.
 */
public void setFrame(Frame frame) { // Optional
    this.frame = frame;
}
/**
 * Sets the ActionCommand used in the ActionEvent when the
 * user attempts to close the window. The window may be
 * closed by clicking on "X", choosing Close from the
 * window menu, or pressing the Escape key. The default
 * command is "CloseRequested", which is just what a Close
 * choice button would probably have as a command.
 */
public void setCloseWindowCommand(String command) {
    closeWindowCommand = command;
}
/**
 * This is handy for providing a small image that will be
 * displayed to the left of the message.
 */
```

```
public void useImageCanvas(Canvas imageCanvas) {
    this.imageCanvas = imageCanvas;
}
/**
 * This loads the image from the specified @param fileName,
 * which must be in the same directory as this class.
 * For example @param fileName might be "LightBulb.gif".
 */
public void useImageCanvas(String fileName) {
    try {
        ImageCanvas imageCanvas = new ImageCanvas(MessageBox.class, fileName);
        useImageCanvas(imageCanvas);
    } catch(Exception ex) {
        print("MessageBox.helpfulHint() - Cannot load image " + fileName);
        ex.printStackTrace();
    }
}
/**
 * The @param label will be used for the button and the
 * @param command will be returned to the listener.
 */
public void addChoice(String label, String command) {
    Button button = new Button(label);
    button.setActionCommand(command);
    button.addActionListener(this);
    button.addKeyListener(this);
    buttonPanel.add(button);
}
/**
 * A convenience method that assumes the command is the
 * same as the label.
 */
public void addChoice(String label) {
    addChoice(label, label);
}
/**
 * One of the "ask" methods must be the last call when
 * using a MessageBox. This is the simplest "ask" method.
 * It presents the provided @param message.
 */
```

```
public void ask(String message) {
   if (frame == null) {
      frame = new Frame();
      frameNotProvided = true;
   } else {
      frameNotProvided = false;
   }
   dialog = new Dialog(frame, true); // Modal
   dialog.addWindowListener(this);
   dialog.addKeyListener(this);
   dialog.setTitle(title);
   dialog.setLayout(new BorderLayout(5, 5));

   Panel messagePanel = createMultiLinePanel(message);
   if (imageCanvas == null) {
      dialog.add("Center", messagePanel);
   } else {
      Panel centerPanel = new Panel();
      centerPanel.add(imageCanvas);
      centerPanel.add(messagePanel);
      dialog.add("Center", centerPanel);
   }
   dialog.add("South", buttonPanel);
   dialog.pack();
   enforceMinimumSize(dialog, 200, 100);
   centerWindow(dialog);
   Toolkit.getDefaultToolkit().beep();

   // Start a new thread to show the dialog
   Thread thread = new Thread(this);
   thread.start();
}
/**
 * Same as ask(String message) except adds an "Okay" button.
 */
public void askOkay(String message) {
   addChoice("Okay");
   ask(message);
}
/**
```

```
 * Same as ask(String message) except adds "Yes" and "No"
 * buttons.
 */
public void askYesNo(String message) {
   addChoice("Yes");
   addChoice("No");
   ask(message);
}
//---------- Private Methods ----------------------------
private Panel createMultiLinePanel(String message) {
   Panel mainPanel = new Panel();
   GridBagLayout gbLayout = new GridBagLayout();
   mainPanel.setLayout(gbLayout);
   addMultilineString(message, mainPanel);
   return mainPanel;
}
// There are a variaty of ways to do this....
private void addMultilineString(String message,
      Container container) {

   GridBagConstraints constraints = getDefaultConstraints();
   constraints.gridwidth = GridBagConstraints.REMAINDER;
      // Insets() args are top, left, bottom, right
   constraints.insets   = new Insets(0,0,0,0);
   GridBagLayout gbLayout = (GridBagLayout)container.getLayout();

   while (message.length() > 0) {
      int newLineIndex = message.indexOf('\n');
      String line;
      if (newLineIndex >= 0) {
         line = message.substring(0, newLineIndex);
         message = message.substring(newLineIndex + 1);
      } else {
         line = message;
         message = "";
      }
      Label label = new Label(line);
      gbLayout.setConstraints(label, constraints);
      container.add(label);
   }
```

```
}
private GridBagConstraints getDefaultConstraints() {
   GridBagConstraints constraints = new GridBagConstraints();
   constraints.weightx  = 1.0;
   constraints.weighty  = 1.0;
   constraints.gridheight = 1; // One row high
      // Insets() args are top, left, bottom, right
   constraints.insets   = new Insets(4,4,4,4);
      // fill of NONE means do not change size
   constraints.fill = GridBagConstraints.NONE;
      // WEST means align left
   constraints.anchor = GridBagConstraints.WEST;

   return constraints;
}
private void centerWindow(Window win) {
   Dimension screenDim = Toolkit.getDefaultToolkit().getScreenSize();
   // If larger than screen, reduce window width or height
   if (screenDim.width < win.getSize().width) {
      win.setSize(screenDim.width, win.getSize().height);
   }
   if (screenDim.height < win.getSize().height) {
      win.setSize(win.getSize().width, screenDim.height);
   }
   // Center Frame, Dialogue or Window on screen
   int x = (screenDim.width - win.getSize().width) / 2;
   int y = (screenDim.height - win.getSize().height) / 2;
   win.setLocation(x, y);
}
private void enforceMinimumSize(Component comp,
      int minWidth, int minHeight) {
   if (comp.getSize().width < minWidth) {
      comp.setSize(minWidth, comp.getSize().height);
   }
   if (comp.getSize().height < minHeight) {
      comp.setSize(comp.getSize().width, minHeight);
   }
}
//--- Std
private static void print(String text) {
```

```
    System.out.println(text);
}

} // End class
```