Tel: (301) 587–3000 **Fax:** (301) 587-7877 **http:**//www.setecs.com **E-mail:** info@setecs.com

Security Technologies for Open Networking Environments (STONE)

DoE SBIR Phase I Project

The Final Report

Date: November 2005

Contact Information:

Sead Muftic, Ph.D.
President/CEO (Prime Investigator)
Tel: (301) 587–3000, E–mail: sead.muftic@setecs.com
SSIC, 8070 Georgia Avenue, Silver Spring, MD 20910

Prepared for The United States
Department of Energy
High-Capacity Network Research Program
Office of Science

Work Performed under Award #DE–FG02–04ER84072

Security Technologies for Open Networking Environments (STONE)

Table of Contents

Project Results

During this project, SETECS performed research, created the design, and the initial prototype of three groups of security technologies: *(a) middleware security platform*, *(b) Web services security*, and *(c) group security system*. As discussed below, the results of the project indicate that the three types of security technologies can be used either individually or in combination, which enables effective and rapid deployment of a number of secure applications in open networking environments.

1.1 Middleware Security Platform

The *middleware security platform* represents a set of object-oriented security components providing various functions to handle basic cryptography, X.509 certificates, S/MIME and PKCS#7 encapsulation formats, secure communication protocols, and smart cards. In this project the platform has been designed in the form of *security engines*, i.e., software components that manipulate *secure objects* (software entities with specific security data) and perform various security functions.

In particular, the following four security engines have been designed:

- A *Registration Engine*, which manipulates registration objects, containing registration and identification data for users, servers and Web applications. Registration and Identification objects are stored in a Security database, in the LDAP/X500 directory, and in smart cards;
- A *Certification Engine*, which performs various X.509 certificate functions – creation, verification, distribution, storage and revocation of certificates and Certificate Revocation Lists (CRLs). Certificates and CRLs are handled in the form of secure objects, stored in LDAP/X500 directory and in smart cards;
- An *Authorization Engine*, which creates, uses and processes various authorization policies and credentials, both for Web based applications and for GSAKMP policies. User authorization credentials are handled in the form of authorization objects, which will be stored in Security database and in smart cards;
- A *Secure Group Applications Engine*, which creates and verifies secure group documents – performing digital signatures, encryption, and access control. Documents are XML encoded and thus suitable for Web manipulation and processing.

Security services provided by these engines are critical to enable secure Web and group applications. The security platform is currently available through high–level functional APIs and as methods exported by generic security objects.

By creating a middleware security platform consisting of multiple independent components (i.e., security engines and secure objects) several advantages have been achieved:

- *Object-oriented*: data and functions are available to application developers transparently through a set of exported reusable services, without the need for repetitive implementation and testing of such modules;

- *Modularity*: each component can be replaced by alternative modules without impacting the overall system architecture and performance;
- *Simplified development and testing*: each module can be developed and tested independently, resulting in simpler development with lower risks;
- *Portability*: since each module can be developed in different programming languages (e.g., Java, C++) and adapted to different environments, the resulting middleware security platform can be ported more easily to different IT platforms;
- *Simplified extensions*: the middleware security platform can be extended by creating and adding new objects leaving other components unchanged, thus reducing development and implementation efforts.

During the project, the middleware security platform has been fully designed and a preliminary Java-based prototype has been created for the *Microsoft Windows* operating system.

1.2 Web Services Security

The *Web services security* system, designed in the project, consists of technologies and applications that provide authentication (i.e., single sign–on), authorization, and federation of identities in an open networking environment. The system is based on OASIS SAML and XACML standards for secure Web services.

Figure 1.1 shows the Web services security system's design and architecture that was created during the project.

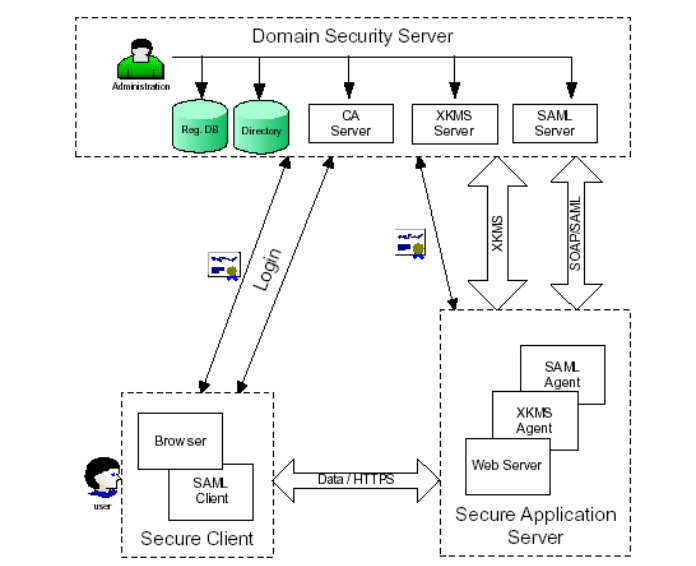

Figure 1.1: Components of SETECS' Web Services Security System

Its topology comprises three major components:

- *Domain Security Server (DSS)* is the main building block of the system. This server handles logins, SAML assertions, authentication of the users, and certificate functions.

The DSS uses different databases containing users' and servers' data. Important components of the DSS are a Certificate Authority (CA) server for certificates, XKMS server for session keys and SAML server for assertions.

- *Secure Application Server (SAS)*, which as the name implies, is the server running an application and SAML Agent module needed for handling the security (verification and authentication of users).

Secure Client (i.e., user), which accesses a Web service using a browser in a secure way.

Domain Security Server

As shown in Figure 1.2, in addition to the SAML server, the Domain Security Server consists of several components, including the database server, the CA server, the Web server as well as the LDAP server.

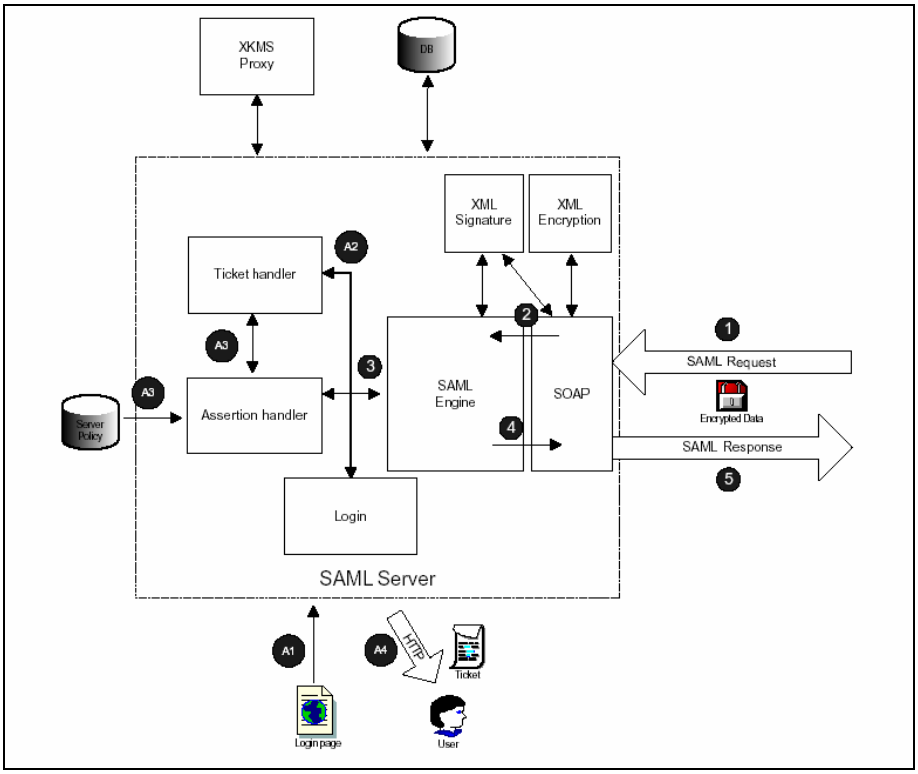

Figure 1.2: Transactions with the Domain Security Server

The SAML server has two roles:

- *Handling of the login sequence*: At the beginning of each session, the user has to login into the system, which is accomplished by accessing the DSS login page using a standard web browser (step A1). Once the user's credentials are verified against the registration database, two processes are initiated. First, a SAML ticket is created for the user (step A2) and, second, SAML attribute and authentication assertions will be created based on the server policy (step A3). Both the SAML ticket and the assertions will then be sent back to the user (step A4).

- *Processing of SAML Request/Response messages*: the SAML Request is received from Secure Applications Servers through a SOAP protocol (transaction ❶), extracted from the SOAP message and decrypted by the XML encryption. Before the SAML Request message is processed by the SAML module (step ❷), the messages digest and the signatures are verified using XML Signature. The SAML Engine processes the message according to SAML protocol and creates a SAML Response message. The message is signed using the XML Signature, encrypted with XML Encryption before being packed into a SOAP message and sent back to the requester (SAS) (transaction ❺).

Authorization System

In addition to the SAML and XACML engines, the authorization system consists of two sets of components:

- *An Authorization Administration System:* authorization administration is performed at the Domain Security Server as part of the overall domain security administration. For this purpose, the DSS is extended with additional components, to maintain components of the authorization configuration file (rules, policies, and policy sets (Figure 1.3). Administration of the authorization system performs functions to create user roles, to specify authorization rules, policies and policy sets.
- *An Authorization Enforcement System:* this system comprises two components: *Policy Decision Point (PDP)*, located at the DSS, making decisions, and *Policy Enforcement Point (PEP)*, located at the secure application servers, requesting authorization decisions.

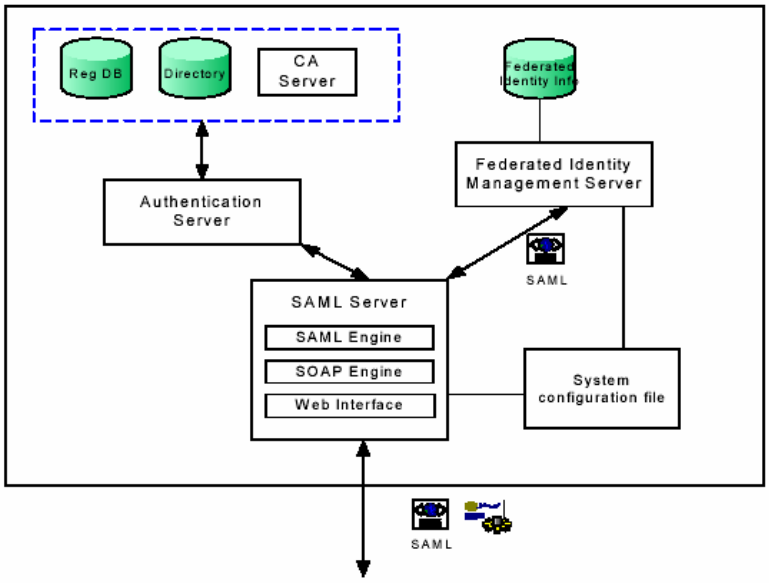

Figure 1.3: Authorization Components at the Domain Security Server

Federation of Identities

Federation of identities in multi-domain scenarios is supported by a set of security engines that represent the core of the Federated Identities Management Server, which is also an extension of the Domain Security Server.

The Federated Identity Management server allows users to federate their identities or terminate the federation between the *service provider* and the *identity provider*. At the service provider web site, the users are offered a list of identity providers to which they can choose to federate their identities. After users federate their identity, they can perform Single Sign-On protocol in an environment of federated domains.

Identity federation scenario is shown in Figure 1.4 and prototype Web pages for this scenario are shown in Figures 1.5 and 1.6.

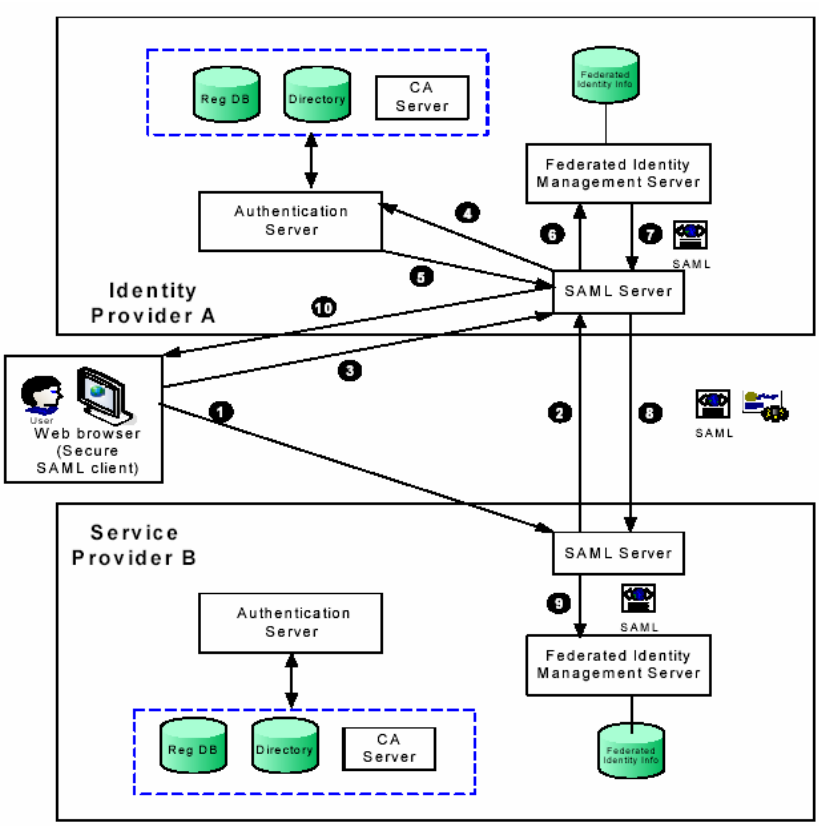

Figure 1.4: Identity Federation Scenario

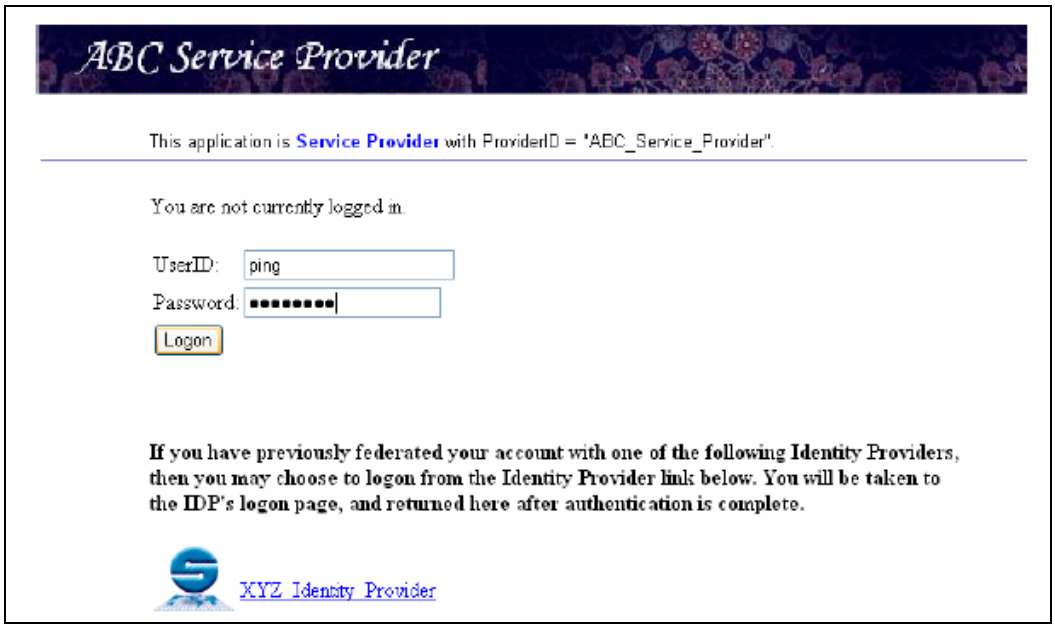

Figure 1.5: Web Page of the Service Provider Federated with XYZ Identity Provider

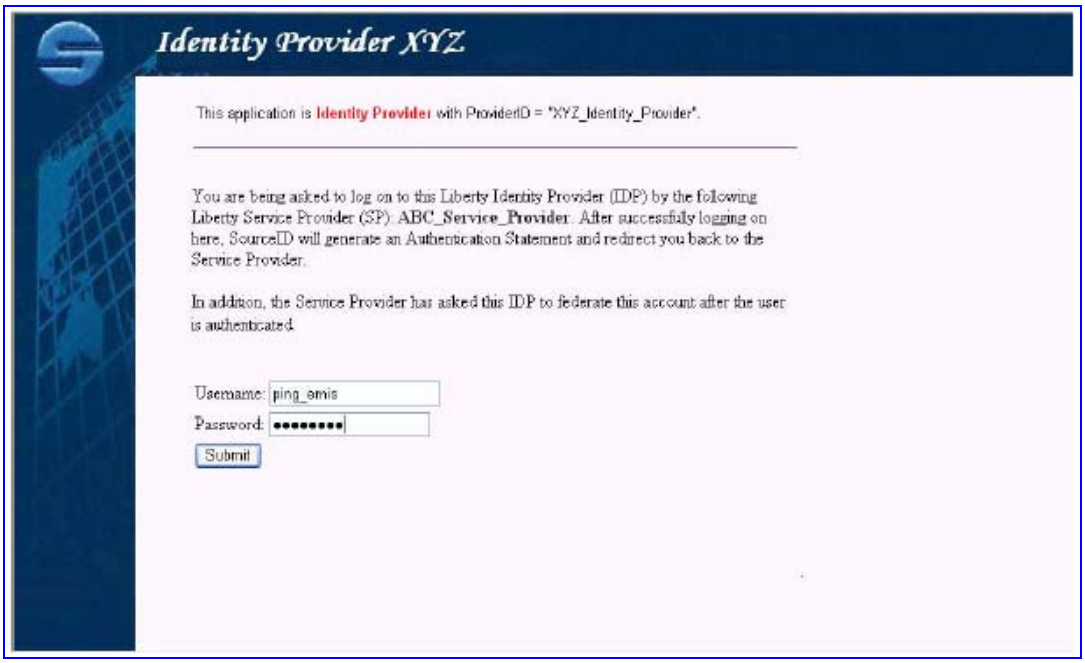

Figure 1.6: Web Page of the XYZ Identity Provider

During the project, the Web services security architecture has been fully designed and preliminary prototyping has been carried out for some of its components.

Authentication Example (Single and Federated Domains)

Figure 1.7 illustrates two application scenarios for the Web services security system: authentication in a single domain and authentication in federated domains (in this case two domains).

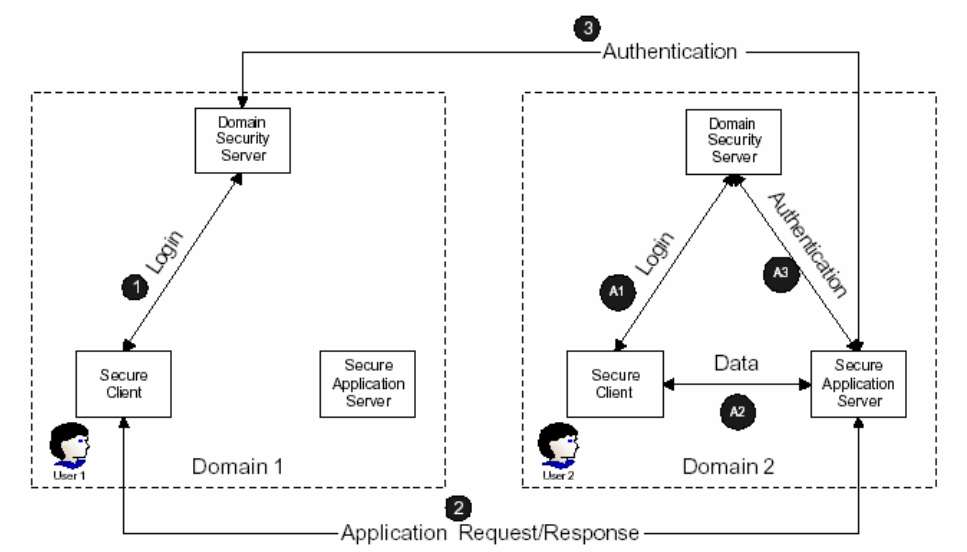

Figure 1.7: Single and Federated Domain Architecture

In both scenarios, a user performs an initial *login* through his/her home DSS, shown as arrow (1) and (A1). A user that belongs to Domain 1 and tries to access the application server that belongs to Domain 2, performs an application request (2), presenting a security token received in the initial login (1). For the user belonging to Domain 2, access to the application is illustrated by (A2). Before access is granted, user *authentication* is performed. This security function of the Web services security system is carried out by contacting the user's home DSS, (3) if the user belongs to Domain 1 or (A3) if the user belongs to Domain 2. In addition to authentication, an *authorization* function may be required, depending on the application server's security policy. If the user's authentication (and authorization if needed) is successful, the application data will be accessible to the user. It is important to note that, with this architecture, the user is required to login only once to his home DSS, which then issues and sends a security token back to the user. This token is subsequently used within the system (and transparently to the user) every time the user requests access to a SAS.

1.3 Group Security System

The *group security system* consists of a number of security technologies under a unified architecture, which supports creation of secure groups and execution of secure group transactions and applications in an open networking environment. The system is based on extensions of the GSAKMP standard for group key distribution and management.

Figure 1.8 shows the architecture of the group security system designed during the project, consisting of three major layers:

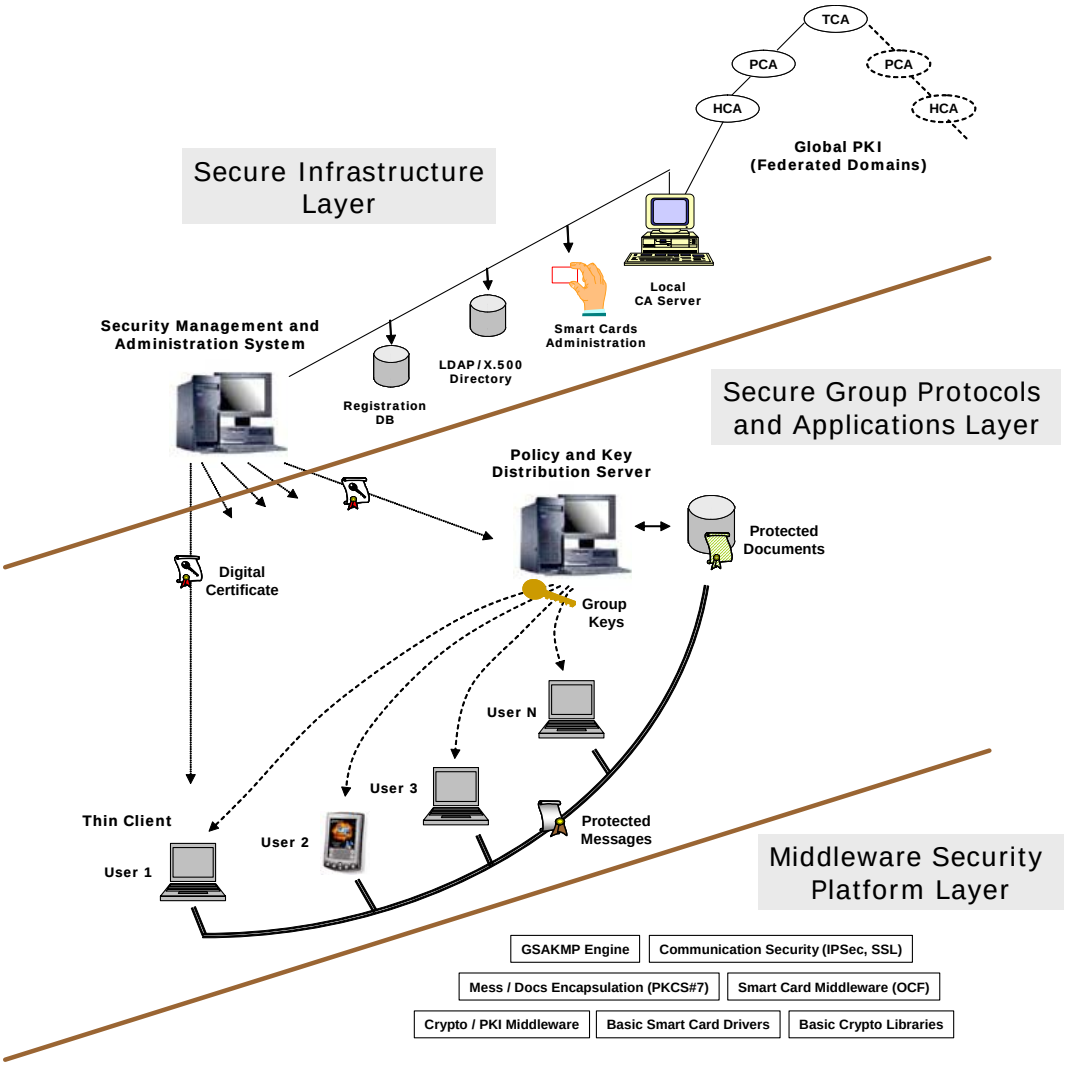

Figure 1.8: Components and Layers of the Group Security System

- The *Top layer* is the Security Infrastructure with the *Security Management and Administration System* components and protocols that provide security functions common to all secure network applications: registration, identification and certification of all entities, their strong authentication, management of smart cards (if used), handling of audit logs, etc. ;
- The *Middle layer* is the Secure Group Protocols and Applications layer, consisting of the *Policy and Group Key Distribution Server and Web-based (thin) Client.* Within this layer, two types of messages are exchanged, all compliant to the GSAKMP standard: communication messages exchanged between the Group Members, and administration messages, exchanged between Server Administrators, Group Officers and Group Members. All these messages are based on a request/response protocol, i.e., a client sends a request to the server indicating the function to execute and including required data. After the server executes the function, the result (success or error code) and

corresponding data is sent back to the client. All these components are described in more details below;

- The *Bottom layer* is the supporting *Middleware Security Platform*, the cryptographic platform already described in section 1.1.

The group security system is designed to perform the functions necessary to create secure groups and enable secure group applications. Specifically, the system can manage group roles, create and disseminate a group's security policy, perform authentication and authorization (i.e., access control) of users using PKI certificates and Web services security, generate group keys, and recover from compromises. In accordance with the GSAKMP standard, the group security system must perform all the required group life-cycle functions: (a) group definition, (b) group establishment, (c) group maintenance, and (d) group removal.

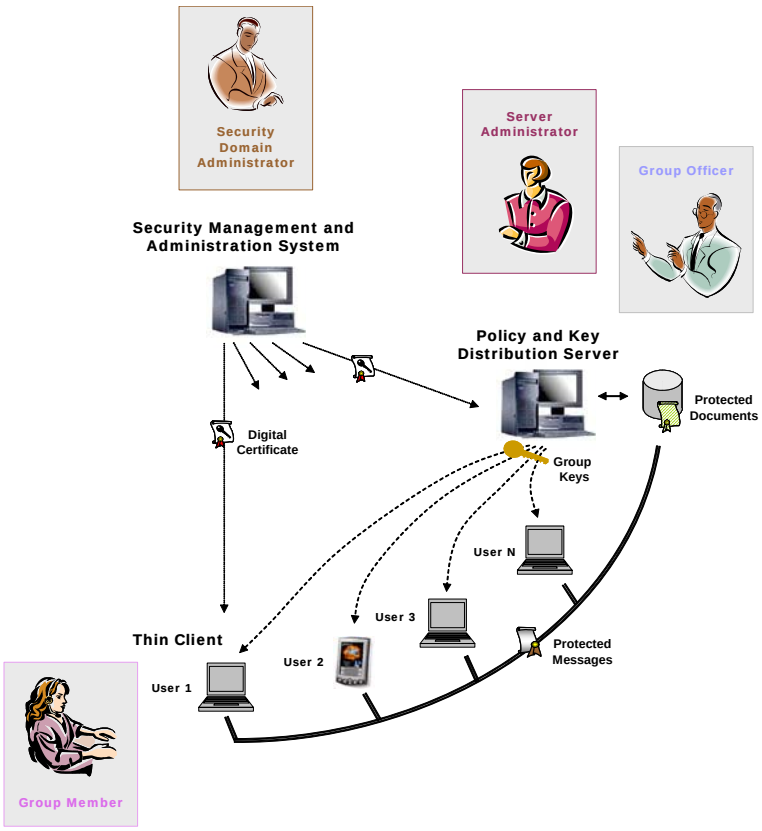

Figure 1.9: Roles supported by the Group Security System

As shown in Figure 1.9, the group security system has been designed to support four roles:

- *The Security Domain Administrator* is responsible for providing security functions defined in the top layer;
- *The Server Administrator***.** The central component of the group security system is the Policy and Group Key Distribution Server. The role of the server administrator is to configure, maintain and administer the multiple servers that comprise a typical

configuration. In addition, the server administrators register and authorize group officers at individual servers, enabling them to create multiple groups;

- *The Group Officer* (GO) authorizes the creation of groups at a specific Policy and Group Key Distribution Server. The group officer specifies the group's security policy by generating a group policy token, with parameters such as algorithms and protocols used to protect group messages. This token contains a list of all authorized group members, so that the server can later perform authorization (i.e., access control) and key distribution based on the membership list specified by the group officer;
- *The Group Member* (user) is any entity that participates in group transactions. The entity is most often a user, but can also be a program as in the case of Grid architecture described in section 5. A group member gains access to the group key by contacting the Policy and Group Key Distribution Server and requesting group membership. The server allows access to the group key according to whether the user is specified as a group member in the policy token.

All the described security functions will be performed automatically and transparently using simple and easy-to-use interfaces. The prototype interface for the Server Administrator is shown in Figure 1.10.

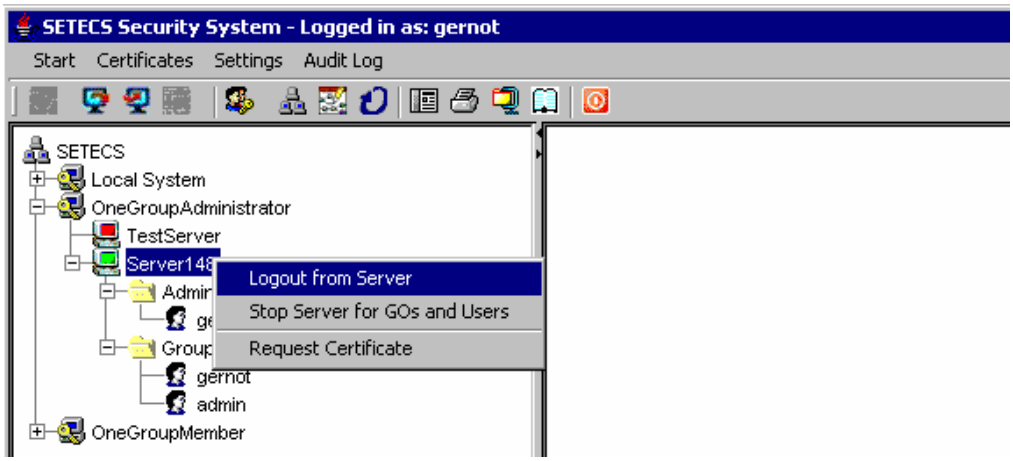

Figure 1.10: Prototype Interface for Secure Group Server Administrator

Secure Group Applications

The group security system has been designed to support four secure group applications:

- A *Secure Instant Messaging:* with the Secure Instant Messaging application, a group of pre-registered and authorized users may securely exchange short instant messages. The messages are encrypted using the group key. Whenever a new user joins the group or some member leaves the group, a new group key is generated and immediately distributed to all the members in the group. This key management mode supports "forward" and "backward" confidentiality of group messages, i.e., new members joining a group are not in the position to read previously exchanged messages and those who leave a group are not in the position to read future group messages. When a group session terminates, all the exchanged messages are not accessible by the users.

- A *Secure Whiteboard:* the Secure Whiteboard application allows messages to be exchanged within a forum while, at the same time, being deposited on the "whiteboard" for long-term accessibility by the authorized members. The protected messages in the Whiteboard application may also be significantly longer than the typical one- or two-line instant message.
- A *Secure Document Sharing:* the Secure Document Sharing application enables a group member (or user) to create protected documents/files, encapsulate for multiple recipients, and deposit them to a common repository/server. Authorized member of the group can download these documents to their local machines, verify and review them before further use.
- A *Secure Document Archiving:* with the Secure Document Archiving application, documents are deposited on the server for long-term accessibility. The period of time the documents can be archived can be defined according to the specific customer requirement.

During the project, the group security system architecture was fully designed and preliminary prototyping was carried out for some of its components.