Security in the Ambient Computational Environment

By

S.Vidyaraman

Submitted to the department of Electrical Engineering and Computer Science and the Faculty of the Graduate School of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science.

_____

Dr. Joseph B. Evans (Chair)

_____

Dr. Gary J. Minden (Committee Member)

_____

Dr. Arvin Agah (Committee Member)

_____

Date thesis accepted

Acknowledgements

This thesis is the result of the help and support of many people.

I would like to thank my committee - Dr. Evans, Dr Minden and Dr Agah for providing me with their valuable suggestions and insight. All their inputs came in at the right time and their help continued even after ACE was unfortunately closed down. Thanks are also due to our Senior Research Engineer Leon Searl without whom ACE would not have been designed or implemented.

Abstract

An Ambient Computational Environment (ACE) is a set of Devices and services that allow any user to operate in the Environment without the usage of too many cumbersome devices attached to the users person. Security in such an infrastructure is a must for apparent reasons. This thesis addresses the design and implementation issues related to securing an ACE, namely identification of a user and a service in the ACE and ensuring proper encrypted communications between users and services alike. A limited form of Public Key Infrastructure has been put in place in the ACE. Identification through X.509 digital certificates and RSA Key pairs also ensures compatibility with other applications.

Table of Contents

Table of Figures

List of Tables

1 Introduction

1.1 ACE and the concept behind it!

ACE stands for the Ambient Computational Environment, a ubiquitous networking environment where all conventional devices are embedded in the work area. ACE has been conceptualized with the aim of giving users the mobility they carve for in today's world while also relieving users the burden of cumbersome devices. A mobile user in today's world needs to have a pager, cell phone and a laptop with wireless connectivity to truly stay mobile. He faces the usual list of problems whenever he wants to hook up a presentation, even from his office desktop to the conference room machine. ACE solves all these problems. In fact, the user need not even carry any mobile device with him to be mobile.

In an Ambient Computation Environment, all devices are embedded into the environment itself! Cameras, Projectors, Video screens, Speakers, microphones etc are present in the environment, be they in the office or the conference room or even the hallway! And the user can pop up their workspace from anywhere in the ACE domain. He need not carry a laptop to be mobile. All he would need in the future would be a cell phone or an advanced futuristic communication device to control his environment.

There are a number of services in the ACE, which are in communication with each other. All theses services subscribe to a central "server service" which is the ASD (ACE Service Directory). The ASD also stores information about each device / service as far as its context and state are concerned.

In a nutshell, Ambient Computational Environments is the integration of computational resources into a robust, secure, and pervasive network where users can easily access and co-opt devices, services, and applications via spoken, graphical, or gesture commands. ACE is ubiquitous and accessing information and computational

processing power is easy and fast, independent of the user's location within the environment.

1.2 Entities in the ACE

The main actors in the ACE are the daemons (services) and the users of the system. Security design and implementation revolves around these two entities.

1.2.1 Daemons / Services

A daemon is a piece of software: it provides services, which can be accessed inside the ACE's infrastructure. Typical daemons in the ACE environment are Audio Capture and Play, Fingerprint Identification Unit control daemon, Camera and Projector control daemon etc. These daemons directly or indirectly control physical devices. Some of them do not have anything to do with devices at all. Most, if not all of the physical devices do not have the capability of having a hardware specific key associated with them. Access to such devices in the ACE Infrastructure is protected through the daemons.

1.2.2 Device

A device implies any physical device. This may include a host computer on the ACE Infrastructure, a projector, a camera etc. They may be wired or wireless. There may be devices that have the capability to store a key internally. There may also be devices that just need to be controlled by a daemon. Most devices do not come with a provision for storing an internal key within the device itself. All devices are of use in providing a service of some sort. All devices in the ACE are controlled through a device specific daemon.

1.2.3 Users

Users in the ACE domain form the main entity other than the daemons around which security design and implementation revolve. All users in the ACE

infrastructure have various forms (and hence levels) of identification / authentications schemes. They consist of user name / password pairs, IButtons, and fingerprint identification. Electronic identification consists of a RSA Key pair along with which X.509 Digital certificates are issued to each user by the ACE central certification Authority.

1.3 Security issues in the ACE

Security in the ACE is an issue of paramount importance. The issue ranges from providing secure communications to providing proper access control mechanisms to users. As mentioned above, the important entities around which (whom) the security of the ACE revolves are the daemons and the users themselves. Users are usually the weakest links in the entire security scheme. Given the security schemes that have been implemented in the ACE, it finally rests on the user not to choose a bad password and compromise the system. Daemons communicate with each other through their command interfaces. Some daemons shuttle around streams of data (Audio & Video) between themselves. All these communications need to be secured. Users need to be properly identified. This is done by means of user name / password pairs, IButtons, and fingerprint identification. Electronic identification consists of a RSA Key pair along with which X.509 Digital certificates are issued to each user by the ACE central certification Authority. When such user identification occurs, problems ranging from issuing the digital certificate to ensuring that the communication of the identification form (password / fingerprint etc) is properly secured are addressed. Herein comes the concept of issuing daemons a cryptographic key through a common key manager. Communication of the concerned data within an ACE between daemons has to be secured. If the user accesses the ACE from outside the domain, a proper protocol has to be put into place to prevent an attacker from knowing the password. Other concerns in the ACE relate to PKI issues like issuing certificates and distributing them. Certificates are issued to Daemons and Users the first time they startup or are registered. Prior to the creation of certificates, RSA key

pairs are created for each user and daemon. A daemon is supposed to be up and running all the time. If a daemon shuts down and comes up again, it is reissued the same RSA Key Pair and X509 Certificate from the ACE Certificate Authority. The daemon is determined to be the "same" based on its name and the address from where it operates. There is a dependence on the OS security at this point.

1.4 Organization of the Thesis

- Chapter 2 deals with the background work on the protocols and key exchange / management mechanisms. This chapter deals with the requirements of the protocols and key exchange mechanisms on a generic security level (and hence applicable to ALL architectures) and a few ACE specific requirements.
- Chapter 3 deals with the ACE Security Architecture Overview. It lists a few services that are illustrative of security scenarios in the ACE. Common security pitfalls and their design solutions are also discussed here.
- Chapter 4 describes the daemons that have been implemented in the ACE as part of the security solutions. This chapter also describes the limited form of PKI solutions that have been implemented in the ACE.
- Chapter 5 makes a pointed analysis at the ACE security design. It proceeds through a 3-step process that reveals whether the ACE security design should be the way it is or should the track be different in terms of the approach taken.
- Finally, chapter 6 marks a conclusion and the areas where future work in the ACE has scope.

2 Background Work

This section of the thesis deals with the background and work relating to the security issues in the ACE. We start by describing the scenario in the ACE, which requires the deployment of the mentioned security component (like a security protocol or a key distribution system or a PKI). Then we describe a few of the schemes of the security component that were studied / taken into consideration. Based on the specific requirements of ACE, we choose a particular scheme for the security solution under consideration.

2.1 Security Protocols

2.1.1 Introduction and Scenario

An ACE domain, in its barest form, is defined as the set of all the host machines on which an ACE daemon is running. All these host machines may operate as a desktop machine or act as a controller to various devices like a camera or a projector. In addition, the desktop machines may have devices connected to them as part of user authentication mechanisms. (Fingerprint scanner, IButton receptacle) Whenever a user logs into an ACE domain, he is authenticated by a daemon into the ACE domain. This is possible when the user logs into the ACE domain from within the domain itself. By "within the ACE domain", we mean that the user logs into the ACE through one of the hosts that are part of the ACE domain and hence have a local authentication daemon running on them (like the IDMonitor). When the user wants to use the ACE resources from outside the ACE domain, he needs to log into the ACE domain from outside. The difference in this case is that the authentication parameters for the user have to be sent from outside the ACE domain across the network, a public and unprotected channel and hence the user has to use a standalone application to log into the ACE domain. When the stand-alone application tries to transmit the authentication parameters to the ACE domain, it needs to operate a protocol (on the

application layer for any ACE) so that the transmitted parameters on inspection cannot reveal the authentication parameters. What has just been mentioned above is a requirement a security protocol operating in ACE should satisfy. In this section, we shall lay down the requirements of a security protocol with reference to ACE and also present a few protocols that could be used for ACE in the context of remote login. This section is intended to set the requirements for a generic security protocol by means of which two parties can communicate with each other securely and privately, ensuring the integrity of the messages passed. The ground for the scenario is described and hence the capabilities of the attacker.

In particular, the context in which the protocol shall be deployed can also assume a scenario in which there are a number of wireless devices in the network under consideration. Most of these devices are initially assumed to be Laptop devices were wireless Ethernet cards are expected to provide connectivity. The utility of the protocol is to provide the user with a secure channel to communicate to another location, be it another wireless device or the main hub itself. The extension comes about by addition of wireless devices that aren't prone to direct user interaction. Examples of such wireless devices are wireless cameras, projectors and other resources. The protocol requirements take no notice of such distinctions: it shall in fact encompass all possible devices: wired and wireless.

The intention of the protocol is its usage in lieu of wireless networks for secure mutual authentication. However, it is expected that the protocol, when designed / modified from an existing one, shall be as useful in the wired domain as in the wireless domain. The reason for mentioning the purpose of the protocol is that certain aspects of the protocol should be specifically tailored for the wireless domain.

2.1.2 Requirements

We present here the requirements of a generic security protocol. Most of the requirements, as mentioned before are generic to any security protocol [7]. The

requirements shall aim at achieving these security Properties, namely confidentiality, integrity, which is closely related to authenticity and availability

Apart from the above two requirements of confidentiality and integrity that are generic and the most common properties, availability is also made a "property" in the wireless domain. With "Wireless Devices Implication" in mind, this simply means that the wireless device shall be in full utility with the protocol in place as it would have been without. On the design scale, the implication is that the protocol shall not impose a load on the device so much as to reduce its functionality.

2.1.2.1 Constraints

The section below now represents the situation in which the protocol has to operate. These situations are the constraints under which the protocol has to operate adequately. The single line statement of the protocol requirements is "To ensure the satisfaction of all the security properties in face of such a situation".

- Communication Model

The following communication model is assumed for the underlying layer. The two parties communicate over an insecure channel that, for all practical purposes shall be controlled by the attacker. Note that the term attacker here includes both an active adversary and a passive attacker.

- Channel control

The attacker can read all the messages that pass through the channel (Messages in the sense of the underlying bits or the information that is actually traversing the channel)

- Message modification / Delay

The attacker can modify the messages traversing the channel or delay them by any time scale as is needed to subvert any clock granularity that is in use.

- Message generation

The attacker can generate and send any message to the two communicating parties as and when she wishes. The primary requirement of the protocol shall be to ensure secure and reliable communication in spite of the attacker having all the controls described above.

2.1.2.2 State of the two communicating parties

- Zero Knowledge Systems

The two communicating parties shall have no knowledge of each other initially. This is not to say that "Alice and Bob" do not know each other. The implication is that the two parties do not have a shared secret key. (Note that this state is true only of 2 parties trying to communicate with each other in the presence of a third part: this scenario mostly applies to two wireless users communication with each other without knowing each others credentials)

- Parallel Sessions

The two communicating parties are host computers that are open to the network. The attacker shall be able to open multiple sessions with both Alice and Bob simultaneously. Also, these multiple sessions cannot communicate with each other. This is to say that there shall be no inter-process communication between any two sessions on a computer. The adversary shall be able to engage them into any protocol conversation, with the possibility of using the information obtained in one session in another. (Note that the actual protocol may not allow this, but it is required of the protocol design that despite a possibility of multiple sessions, security concerns are met)

2.1.2.3 Stand Alone Protocol Requirements

- Other Protocol Interactions

The attacker shall be able to convince Alice and Bob to engage in other stand Alone safe protocols, in as many sessions as is possible/required. This raises the possibility of a chosen protocol attack. It is expected of the protocol to withstand such attacks: A reference is given for the design principles for avoiding the "chosen protocol attack".

- Perfect Forward Secrecy

This simply means that the disclosure of a password does not in any way compromise or reveals prior recorded conversations. This is a requirement in the extreme case of an inadvertent password disclosure that may occur beyond the protocol interactions. Hence the protocol is expected to ensure forward secrecy i.e. the disclosure of a password shall not reveal previous conversations.

- Password guessing Attacks

The protocol shall protect itself from any password guessing attacks (Plain cipher text attacks and offline guessing attacks). It shall also hopefully protect itself from database exposure attacks (as in Lamports hash Algorithm), although it should be recognized that protection from database exposure isn't actually the function of the protocol itself strictly.

- Transitivity of Trust

The protocol shall not allow any transitivity of Trust. By Transitivity of Trust, it is meant that a user can engage in a session with another user, but a third party has to go through the normal process of a protocol interaction. The third party, on the basis of an interaction with one of the two parties currently in a session shall not automatically engage into a session with the other party in the established session.

- Message Exchanges

This is more in the context of the wireless devices mentioned before. It is required of the protocol to enable all the security properties when the devices / users are intermittently on and off. This is more in the context of saving power and recognizing the fact that the CPU power on the devices is limited. This assumes significance as the protocol can't impose a 10-message exchange for mutual authentication and later have the session expire in 10 seconds.

2.1.2.4 Wireless Devices Implication

Stated below are the implications of having wireless devices of varying CPU and battery power in the network [8]. There are quite a few differences between the wired and the wireless domain.

- CPU power

Most of the Wireless devices are expected to have a very low CPU computing power. The implication of this is that these devices can't be expected to perform huge RSA type calculations very fast. The protocol design has to ensure the availability and proper usage of the wireless device. This runs in tune with satisfying the "Availability" Security property. Hence, the protocol may, for instance mandate a certain strong cryptographic scheme for a certain device and a relatively weak one for another device.

Example: Here strong and weak have implications in the sense of usage and time constraint. A Laptop may engage in a session that is a few hours long and a wireless camera in a session that is a few minutes long. Hence, the "weak" cryptographic scheme may have key reusability duration of an hour or so, which is however safe enough for the wireless camera.

- Battery Power

The battery power of these devices is also limited: hence the devices are expected to be off and on intermittently (sleeping mode). This is closely related to

the requirement with CPU power in the context of the security scheme providing a differing message exchange scheme and/or cryptographic scheme (amongst other things) dependent on the device.

2.2 Protocols Considered

The security protocols for ACE are used for authenticating a remote user logging into the ACE domain from outside. At the end of the execution of such a protocol, it is required that the remote user and the ACE Remote connection Manager daemon that runs in the ACE domain perform mutual authentication. Besides that, they should also establish a session key capable of encrypting further communications. The following section explains 3 such protocols that form the basis for most of the security protocols today [4,7]. We also present reasons as to why some protocols were rejected. As is most often the case, many protocols were short-listed and only one was used.

2.2.1 Needham Schroeder Protocol

Table 2-1: Needham Schroeder Protocol Message Exchange

1.	A → S:	A, B, R_A
2.	S → A:	$\{R_A, B, K_{AB}, \{K_{AB}, A\}_{KBS}\}_{KAS}$
3.	A → B:	$\{K_{AB}, A\}_{KBS}$
4.	B → A:	$\{R_B\}_{KAB}$
5.	A → B:	$\{R_B - 1\}_{KAB}$

The table above summarizes the Needham Schroeder protocol in its original form. This protocol is flawed due to certain reasons that will be explained. However, even if the flaw were to be corrected, it's not suitable for ACE's use. The protocol requires the use of a Server S besides the two authenticating parties. It also

requires that the Server S have predefined shared keys with each of the entities. All messages between the 2 entities are encrypted with their shared keys. Hence, the party A has to contact the server with a request to connect to B. A also sends a nonce with his request. Subsequent messages are self-explanatory where the server sends the shared key to A encrypted with its shared key with A and with B. The flaw in the protocol lies in step 4. Notice that B has never been sent the nonce before. An active attacker could hijack the session at this point and try to reverse engineer the session key. However, the reason why this protocol is not suited to ACE is that the number of exchanges is far too much. Besides that, it requires a dedicated server storing a symmetric key for each user in the domain. In the case of the Remote Connection Manager, a separate server is not an efficient way to authenticate the outside party.

2.2.2 Otway-Rees Protocol

Here we shall present a modified Otway-Rees Protocol, which is very much the same as the original one except that the modification doesn't include the message parts that are present in the original one for practicalities sake. The basic security properties are preserved in our presentation of the modified protocol.

Table 2-2: Otway-Rees Protocol Message Exchange

1.	A → B:	A, R_A
2.	B → S:	A, B, R_A, R_B
3.	S → B:	$\{A, B, R_B, K_{AB}\} K_{BS}, \{A, B, R_A, K_{AB}\} K_{AS}$
4.	B → A:	$\{A, B, R_A, K_{AB}\} K_{AS}$

Otway-Rees protocol is very similar to the Needham Schroeder protocol except that it uses two nonce's to resolve the flaw in the Needham Schroeder protocol. Once again, we note two factors in the protocol that aren't desirable to ACE

implementation. One is the presence of a server, a third entity that isn't desirable. The second issue is the key establishment phase takes 4 steps (which is in fact the result of a server presence)

2.2.3 Diffe-Hellmann Protocol

This well-known protocol is described underneath. While the Diffe-Hellmann protocol is not used in its original form in the ACE, a modified version of it called the SPEKE protocol is used to authenticate a remote user into the ACE domain. The protocol hinges around two publicly known values p (a prime number) and g (a generator less than p).

- A chooses a random value x uniformly modulo p-1 and calculates $R_A = g^x \bmod p$ and sends it to B
- B chooses a random value y uniformly modulo p-1 and calculates $R_B = g^y \bmod p$ and sends it to A. Now B also calculates the session key $K_{AB} = R_A^y \bmod p$ which reduces to $g^{xy} \bmod p$.
- A calculates the session key $K_{AB} = R_B^y \bmod p$ which reduces to $g^{xy} \bmod p$.

A lot of protocols exist that are dependent on the Diffe-Hellmann protocol. Other protocols also exist that may satisfy ACE requirements. The protocol implemented in ACE is SPEKE.

2.3 Key Distribution

2.3.1 Introduction and Scenario

This section is intended to set the requirements for a generic key distribution mechanism by means of which two parties can obtain cryptographic keys

and use them to encrypt communications between them through any cryptographic scheme. The keys can be used as temporary session keys for exchanging data in a session or for communication between the two parties to ensure the integrity of the messages passed. The section only sets down the requirements of such a desired robust distribution mechanism: it does not describe any such scheme itself. In essence, all the different scenarios that occur in the ACE environment are described. Standard security properties of any generic key exchange protocol ARE NOT described here. This is a follow up on the previous section on Security Protocol Requirements. It is to be noted that in the ACE Architecture, all the daemons operate inside the ACE domain. In such a scenario, all communications between the daemons are encrypted by means of SSL. Hence, the initial key for secure communications is defined through the startup script. All keys for further communications (between the daemons through their command interfaces and for transferring their data streams) are issued through a Key manager. The present implementation consists of a single key manager which hands out keys as requested by daemons. The requirements below are the precursor for a design structure that consists of a number of key managers that jointly issue a key to a number of users / daemons for conferencing purposes etc [5].

2.3.2 Key Requirements

There is a daemon associated with each device. There are a few fundamental requirements of any key distribution that are not described here. The requirements and scenarios that are specific to the ACE Infrastructure are described here.

2.3.2.1 Keys Distribution Model

- It is expected that the keys that are distributed shall be used for different cryptographic algorithms. The key exchange shall ensure this by providing keys of varying lengths.
- The Model used for the key exchange is not a fixed one. There may be static or dynamic as explained below.

Although the ACE Infrastructure contains the ASD akin to a central server, this may not always be the case. The ASD by itself is classified as a service, which may come up or go down. Hence, in most cases, the initial setup of a device or service is expected to be in a configuration file. After the initial setup, each device / service can have the option of requesting its private / public key pair from the central database (the ACE Certificate Authority). In such cases, Encrypted key exchange is required. This falls in the realm of daemon-to-daemon communication. It is expected that the initial key for daemon-to-daemon communication shall come through a startup configuration file. Hence, when the daemons communicate between themselves, their communications are encrypted with SSL.

In other cases where the service is active, it may require a session key for purposes of a message / control to be passed on. In such cases, Authenticated key exchange is required. Authentication shall be done in this case through digital certificates that are issued to every daemon by the ACE Certificate Authority.

- In a scenario where it is required that in a conference, a user has access higher than he is normally allowed, there must be a Key Distribution System so that session keys are appropriately generated for a particular time limit.

  Example:

  A conference may be initiated in a room where certain resources need to be controlled by a user who is otherwise not allowed to do so. The resource may be a camera or a projector in that room. In such a situation, the user must be in a position to acquire a key with a time limit that allows a time limited access to those resources. This falls more in the realm of Role based Access control and is beyond the scope of this thesis.
- In scenarios where the use of a resource shall demand agreement from all k users, the appropriate form of keys shall be used. This is a case where k users

can perform an action, but k-1 can't do so. In essence, the key center shall be able to distribute private pieces of information to different (k) users so that they are able to collectively compute a shared (session) key. Note that the present implementation has a single Key Manager and hence demands a separate service to ensure that private pieces of information are distributed to k authorized users who need to compute a session key.

2.3.2.2 Wireless devices

All the above requirements are desired properties of any key exchange / distribution mechanism. Listed below are a few specific to wireless devices in the ACE environment:

Wireless devices have the limitation of bandwidth, power (computational AND battery if applicable). These are with respect to devices like wireless cameras, projectors etc. They do NOT apply to Laptops, though if the desired security level could be achieved, the key exchange / distribution scheme might well be applied here too.

- Minimum number of passes

This is the key statement that covers bandwidth and power limitations. The number of passes that are required of a (session) key exchange shall be kept as low as possible. In addition to that, the number of bits transmitted shall also be kept as low as possible. This is for bandwidth conservation. It shall also be attempted to keep the online computation of the device as low as possible to reduce the latency time. This is apart from satisfying the security properties. Hence, reduction in effective computation should not reduce security strengths.

2.3.3 Key Center requirements

These are the information centers that hold all the information on all the key distribution schemes and the actual keys (configured or generated). Listed below are the desired properties of such centers (inside a domain).

2.3.3.1 Key Storage

The keys that are “permanent” to each service / resource are expected to be stored on a database / key store. It is hoped that there will be a scheme (similar to Lamports Hash) that protects the entire system from eavesdropping and server disclosure attacks. This is an extension from the previous sections “Perfect Forward Secrecy” requirement.

2.3.3.2 Number of centers

If there is only one center, then the usual case is that the center knows all communication and if it goes down, all key distribution processes stop. Hence, it is a single point of failure.

There shall be a scheme that entails the power of the key center to m new centers with the following properties [5].

- There shall be *l* centers that are capable of providing the same functionality of the key center as before.
- Even if (*l*-1) centers are compromised, they shall not have information on any common key.
- Even if (*l*-1) centers AND n users are compromised, they shall not have any information that those n users should not know.

3 The ACE Security Infrastructure Overview

This section gives an overview of the architecture of an ACE and the underlying security components in it.

3.1 The ACE Architecture

The ACE concept revolves around and central "server" called the ASD: ACE Service Directory and a number of "clients" that connect to the ASD, which are called services / daemons in ACE terminology. All these services have a specific function to perform. They perform a narrow set of functions: that is they provide services in the ACE as such. The purpose of these services is to enhance the net feeling of the ACE so that an ACE user can work without the usual burdens of today's systems. The ACE envisages a user without any cumbersome devices that he has to carry around.

3.1.1 Daemons: Modes of Communication

All the daemons communicate with the ASD primarily and also provide a command interface so that other daemons can communicate with them. The purpose of the command interface (communication) is to avail of the services that these daemons provide. These communications are encrypted within an ACE using SSL.

The ASD is used for mainly registering the daemons that start up in the ACE. It also considers itself as a daemon too. The significance of this will be apparent in the fact that in one of the databases relevant to the ACE, the ASD information is also stored as if it were to a service. The ASD maintains a database that has information about the services (Machine name / IP Address, its location in the building (Room name), its class etc). Information about the ASD itself can be obtained from the relevant database. Services can obtain information from the ASD about other services.

Here we note that there are 3 "types" of communications:

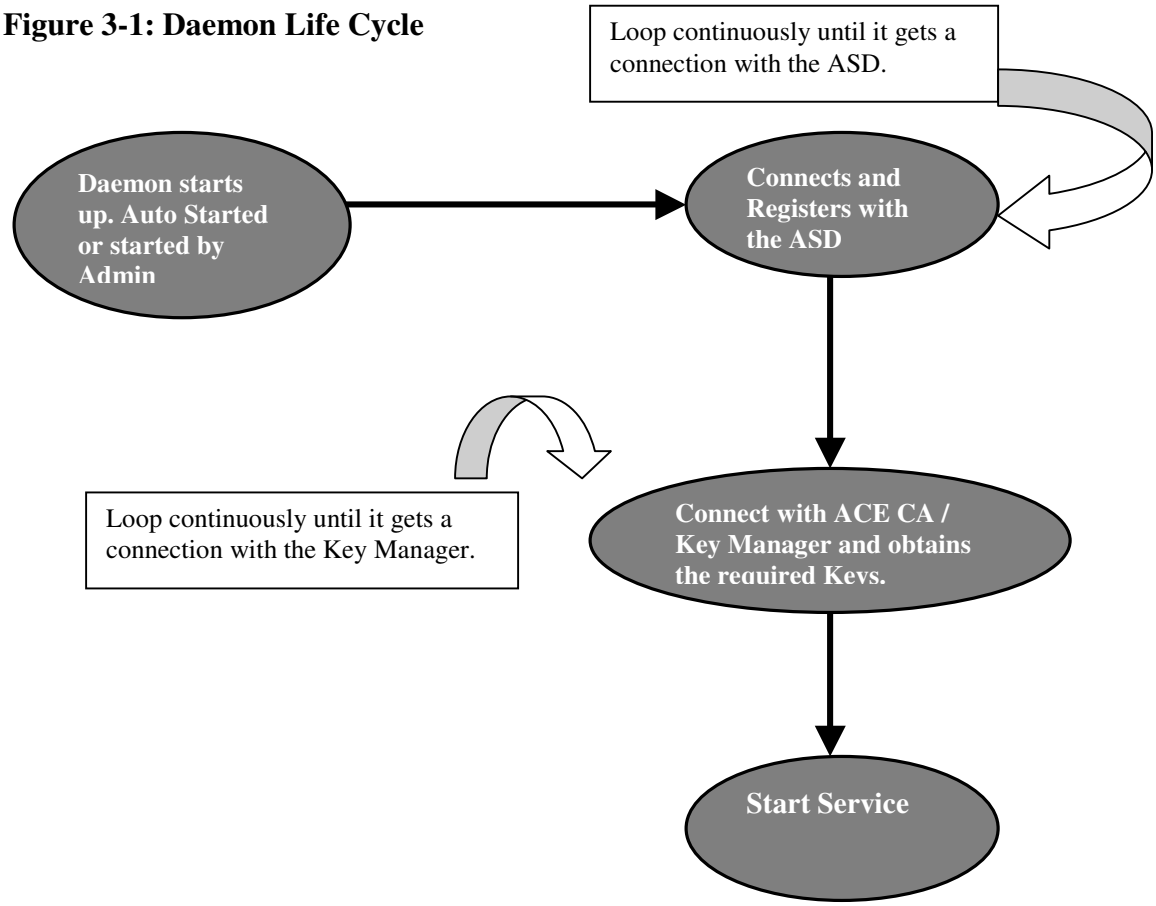
1. Between the daemons and the ASD
2. Between two or more daemons themselves
    a. Through the daemons command interfaces (this is the same as 1)
    b. For purposes of communicating data streams
3. Between two ACE domains

The first two types of communications are encrypted by SSL (not including the data stream communication). The third type of communication has not yet been given any thought about. Besides these communications, there are the network commands that are passed through between the command interface and the implementation thread that each daemon has. These communications are also (to be) encrypted through SSL.

It should also be noted that the daemons (services) also have a port open (their address where other daemons contact them) on the machine that they startup and run. It is possible to telnet into these ports and issue raw commands to the service, the commands being listed in the specifications documents of the services. These ports have communications encrypted through SSL. They are not equipped with an authentication procedure.

We describe below the life cycle of a daemon and the basic connections it makes. It should be noted that this just relates a daemon connecting to the security related services. All the services have different initialization schemes depending on their function.

Figure 3-1: Daemon Life Cycle



Life Cycle:

1. The service starts up: It is either auto started on a machine that is added to the ACE domain or started by the Admin
2. The daemon loops until it can connect and register with the ASD. It knows how to find the ASD through a configuration file (the preferred way) or the address of the ASD and the Key Manager can be hard-coded or found by DNS or a similar mechanism (not attractive).
3. After getting through with the ASD, the service connects with the Key Manager and obtains a key pair for identification purposes and other keys as per its requirement.
4. The daemon starts service.

Note that this is a generic life cycle for a daemon. Almost all daemons differ from this pattern in that they connect to various other services also and perform their function oriented initialization tasks.

3.1.2 Daemons: Related issues

This section covers the present situation and security threats posed by the current setup. It also describes the various streams of data flowing through the network in the ACE and identifies these streams as the ones that need to be encrypted for security.

At the start up of each ACE domain (which is technically initiated by the startup of a number of services starting with the ASD), the key for secure communications is presently specified through a configuration file, which in itself is a not a desirable practice. It must be mentioned at this point that the security of any ACE domain relies on HEAVILY on the OS security model. Any user with too much permission on the file system can very seriously hamper any ACE's Security. The code for ACE is presumably open source and all it would take is permission to read a file (the ACE configuration File for instance) on part of any user to compromise the security of any ACE.

The other types of network traffic that flow in the ACE is relating to the traffic related to VNC and the audio and video streams. All these are equally important. The audio and video streams are produced by two services that essentially provide connection between two systems so that the two systems can do a simple audio/video transfer between them. The VNC streams are much more fundamentally important. They form the essence of the mobility of any ACE. Whenever a user is registered in the ACE, he gets a default workspace created on a robust machine where he runs all his applications. When a user tries to log into the ACE domain, he gets into a VNC viewer that shows his applications that are running. Hence, the VNC stream that runs between the server and the viewer on the local machine has to be

protected / encrypted. There isn't any provision for this yet. It must be noted that the VNC stream is the main stream to all the applications that the user runs. Hence, once the VNC stream is compromised, it can be effectively deemed that the entire ACE security has been compromised from the viewpoint of that particular user.

Below is a listing of the ***some of the services*** that are present in the ACE. The function of each is described in a few words. The purpose of mentioning these services is to bring out the security issues in the ACE. The related security aspect of these services are discussed as well as the measures that are to be taken to ensure the security of the services by part and in whole as an entire domain. These services are representative of the entire ACE domain in terms of representing security issues.

Table 3-1: Daemons and Security Issues

Daemon	**Description of service provided and associated security issues**
ACE Service Directory	This isn't exactly a service although it registers itself as a service when it starts up. It does provide services to other connecting services in the sense that it provides information about the other services. It is also by far the critical "server" of the ACE.
Audio Capture	This service (also called Daemon from now on) simply captures data from the microphone from the local machine on which it is running and transmits it another service (the Audio Play Daemon) that takes in the audio data and plays it on its local machine.

Audio Play	This service is at the receiving end of Audio Capture. It receives the data stream that the Audio Capture service sends and plays the same on the local machine.
Video Capture & play	These two services are the same as their Audio counterparts except that they transfer Video data instead of Audio Data

The security issues involved with the Capture and Play daemons are apparent. The streams that these daemons send (audio and video) need to be encrypted before sending them on the network. These daemons shall be required to obtain cryptographic keys from a key manager (or negotiate a session key between themselves) and agree on a cryptographic scheme before sending data between themselves.

System Resource Monitor: (SRM)	The System Resource Monitor is the service that holds information about the entire ACE resources. This includes information such as the number of CPUs on each machine, the load on each machine etc. It receives all the information it has from the host resource monitors that are running on every machine on the ACE domain. The information transfer from the HRM's (Host Resource Monitor) to the SRM occurs in two parts: The SRM first gets information from the HRM upon initializing through the HRM's command interface. Later, whenever the HRM has any relevant information to pass on to the SRM, it does through means of notifications.

	These notifications are similar to the network commands that pass through the ACE and are hence encrypted through SSL in the same manner as the other network commands are.
Host Resource Manager: (HRM)	This service shares a many to one relationship with the System Resource Monitor. This service runs on every host machine on the ACE Domain. In fact, one of the definitions of an ACE domain is that each host on the ACE Domain has to have an HRM running on it. This service provides information to the SRM about the local systems load, number of CPUs etc. As mentioned before, it communicates with the SRM by means of its command interface and notifications.
IButton	The IButton service is part of the services that provide ways to log into an ACE domain. It falls under the category of Authentication. This daemon essentially polls the serial IButton Interface and decides whether an IButton has been depressed into the IButton slot. It tries to match the user by checking whether the IButton Serial Number is already there is the user Database. The problem with this is the classical database disclosure one. Once the database in compromised, the IButtons are useless. It is even more acute in this case because once the IButton Serial Number is revealed; there is no use of the IButton itself. This is too much of an infrastructure waste. Further more, the IButton, if lost or stolen is more a menace than a use when not stolen. Hence, one of the design issues include placing the "IButton User" in

	a domain with restricted privileges and have the" actual user" as identified by user name / password in a domain with full privileges. It is however an administrative decision on whom to place where. Hence, one of the design requirements shall be to provide for 2 domains (Domains as in User Authorization Domains), one with significantly lesser permissions than the other.
ID Monitor	This is the Daemon that takes in notifications from the IButton etc and authenticates the user. It then tries to display the VNC desktop of the user on the machine where the user has logged in.
Finger Print Identification Unit	As a service, this is similar to the IButton. It differs only in the fact that it polls the FIU instead of the IButton. As before with the IButton, the Fingerprint data has to be protected else the entire system simply collapses again. This service sends notifications to the Requested service that listens for such notifications.
Network Logger	This Service provides a logging facility for all the daemons. Whenever an event occurs that any daemon thinks worthy of logging, it simply invokes the network logger services. The network logger performs the function of accounting in the ACE.

3.2 Security Services

The three fundamental tenets of Security Authentication, Authorization and Accounting are addressed here.

Any ACE domain has two fundamental players: The ***Services*** (Daemons) and the ***user***. Currently, there is no authentication between the daemons themselves. Any Service, after registering itself with the ASD can request and get the command interface of any other service. The first issue to be addressed with the Daemons is authentication within them. The preferred means of identification if the daemons can be through an RSA Key pair (as is the same for users too in addition to fingerprints and IButtons). There shall be a number of daemons that start along with the ASD. These daemons shall have the following functions:

1. They shall manage the Keystore for the daemons (Note that Keystore in this case refers to the same Keystore as created by the java keytool. This keystore is protected through a password.)
2. They shall be responsible for distributing dynamically generated keys to users and daemons alike for
    a. Identification
    b. Conferencing
3. The only authentication for the daemons for the ASD and the Key Management Service shall be the address of the ASD, which shall be known through the configuration files. It should be noted that this again places reliance on the OS security model. Any change in the configuration file compromises the service. Essentially, it affects the working of the ACE domain by depriving it of the services of a daemon.

These daemons shall issue each daemon a key pair and a digital (X509) Certificate upon startup. The key lifetime shall be the lifetime of the daemon or a specified time limit. The only authentication of the daemon upon startup shall be the fact that

1. The administrator is starting them up though a script (that can be executed only by the administrator)

OR

2. The fact that they are "Auto Started" on a host machine with the host machines key for reference.

Again, this is an instance where there is a reliance on the OS Security model. It is assumed that only the administrator can add machines on the ACE domain.

These are the authentication schemes for daemons within themselves. The other major player in the ACE: the user can log in to the ACE by three mechanisms:

1. User / Password Combination
2. IButton
3. Fingerprint Identification.

Any user can log in from “inside” the ACE domain or from “outside” the ACE domain. An ACE domain is defined as the set of all host machines that are registered in the ACE database as part of the ACE domain. These hosts mostly have a set of services running on them, with the Host Resource monitor being one that almost certainly runs on the host machine. As mentioned before, the authentication mechanism with the IButton and Fingerprint methods are fraught with the danger that unlike passwords, the essential secret (the Fingerprint data or the IButton Serial Number) cannot be changed (easily or at all) once it is compromised.

The present mode of ***remote*** authentication is done through a service (The Remote Connection Manager), which authenticates the user through the SPEKE protocol. Each user, after logging into the ACE is presented with a set of his workspaces (which are nothing but VNC sessions) and the set of services that he is allowed to operate within ACE. This shall be in GUI form with the services. Accounting is taken care of by the Network Logger Service.

4 Security Services implemented in ACE

4.1 Remote Connection Manager

This service comes into play when the user logs into the ACE domain from outside the domain (from his house for example). There are two primary issues involved in this process. The first one is to authenticate the user. The second one is to see what operations the user is allowed to perform in the ACE domain where he has logged. In our thesis, we deal only with the first issue, namely Authentication.

4.1.1 Authentication via the RCM

This issue is one of verifying that the user credentials exist in the database of authorized users. User credentials may be of any kind. They may be a user name / password pair, a user name / fingerprint id, a user name / IButton Id or a digital X509 Certificate. The Remote connection manager waits for a connection on a well-known address and then authenticates the incoming user. The authentication procedure follows the SPEKE protocol. (Simple Password-authenticated Exponential Key Exchange) The SPEKE protocol is an ideal Zero Knowledge Password Proof protocol. The situation in the ACE with the Remote Connection Manager is somewhat similar. The Remote Connection Manager has no knowledge of the incoming connection. Furthermore, the communication is assumed to be over an insecure channel with all the restrictions that are mandated in the previous chapter. SPEKE overcomes all these problems. The SPEKE protocol is described below.

4.1.2 The SPEKE Protocol

The SPEKE protocol is a variation of the Diffie-Hellman session Key Establishment process [2]. The DH Key establishment process consists of a prime p and a generator g. The SPEKE protocol is a variant in that it does not presuppose that the generator g be exchanged on the network. Instead the generator is chosen as a hashed function of the password, and is then squared to keep the exponential in the

prime-order subgroup. The protocol has two stages, a session key establishment stage and a verification stage where the two parties (the Remote Connection Manager daemon and the standalone application that connects to the daemon) prove that their session keys are the same. Simply exchanging the double and single hash of the session key itself does verification.

Notation:

S	A small password shared by Alice & Bob
P	A large prime, where (p-1)/2 is also prime
R_A	Secret random number chosen by Alice
R_B	Secret random number chosen by Bob
h(x)	One-way hash of x, like SHA1(x) or MD5(x)

Table 4-1: SPEKE Protocol Exchange

	Alice		Bob
Key Exchange			
	$Q_A = S^{(2 R_A)}$	→	
		←	$Q_B = S^{(2 R_B)}$
	$K = Q_B^{(2 R_A)}$		$K = Q_A^{(2 R_B)}$
	Abort if K< 2		Abort if K< 2
Verification			
		←	$V_1 = h(h(K))$
	$V_2 = h(K)$	→	
	Abort if V_1 != h(h(K))		Abort if V_2 != h(K)

After Alice and Bob verify they have the same value for K, they know they used the same password S. K can now be used as a mutually authenticated session key.

The magic is that even a very **small S** can generate an arbitrarily **large K**.

Note: The two messages Q_B and V_1 can be combined in one reply from Bob. This results in a minimal 3-message mutual authentication protocol.

The advantage with SPEKE is that the Remote Connection Manager and the standalone application can very easily be modified to accept the IButton or the fingerprint data instead of a password. The other implicit advantage of the fact that offline dictionary attacks cannot be carried out on the password is that the fingerprint data and the IButton Serial Number are also not compromised by the protocol. This assumes a lot of significance because once those data are compromised; replacement is difficult if not impossible. Furthermore, the prime generation is done at the Remote connection manager end thereby assuring that the host at the other end is not computationally affected except for generating the session key itself. The session key, which is generated, can now be used for any cryptographic scheme [3]. ***SPEKE thus provides strong password authentication without the need for a strong password.***

SPEKE is used in the ACE domain only for a connection from a reasonably computationally powerful remote host into the ACE domain. It should be noted that SPEKE has the barest minimum of three message exchanges. This fact is ideal and very well suited for a wireless device with low bandwidth and power (computational and power). But the session key calculation is definitely not an easy one for a device with low computational power.

4.2 Key Manager

An ideal Key Managing system in the ACE would be to have a multi-center Key Manager (as mentioned in the previous section) so that there is no central point of failure. In the present implementation, there is a single ACE Key Manager that is supposed to run on a persistent store machine. The Key Manager is a very simple daemon that issues keys for specific cryptographic algorithms. After issuing the key, it also stores the same in a protected keystore and triggers a notification to the Network Logger that logs the key type issued, to whom it was issued and when.

Presently, the key manager supports the following cryptographic schemes:

1. Generating RSA Public Key Pairs
2. Generation of Symmetric Keys for the following algorithms:
    1. DES
    2. Triple DES
    3. Blowfish
    4. AES
    5. CAST5 & CAST6
    6. IDEA
    7. RC2, RC4, RC5, RC6
    8. Skipjack
    9. Twofish
    10. Serpent

It must be realized that generating keys, which are a few bits long, is not the issue while generating keys here. What is implied by support for these algorithms is that the Java security provider (Bouncy Castle in this case) is capable of supporting cipher engines for performing encryption / decryption processes according to the algorithm specifications.

4.2.1 Which Key to use where?

The current recommendations on Key sizes vary depending on the applications in use. Important considerations while using keys of varying sizes are with regard to the cryptographic algorithm, the computational power (and hence the time) required and the application. With regard to Public Keys, doubling the key size roughly corresponds to a six-times speed slowdown in software. This would not matter with offline applications, but would matter a lot in the ACE Infrastructure with all daemons being on the network and with time being a crucial factor. Comparison between public keys and symmetric keys are not worthwhile simply because the applications for which they are used for vary very widely. Given below is a table that

gives the recommendations for Key sizes by the Industry, Corporation and the Government.

Table 4-2: Key Sizes Recommendation

Year	Industry	Corporation	Government
1995	768	1280	1536
2000	1024	1280	1536
2005	1280	1536	2048
2010	1280	1536	2048
2015	1536	2048	2048

The key points in the choice of a key are the Life Span & the required Security Margin.

4.2.1.1 Life Span

The life span can be approximately judged from the table above. In most situations, the decision to change the key size (and in some cases the algorithm itself) is determined by any recent breakthroughs and administrative confidence levels. The above table serves as a guide for the administrators. However, implementing such decisions are also very tedious. In the limited PKI structure that we have in the ACE, decision to change from a RSA key size of 1024 to 2048 has

enormous and far-reaching implications in terms of implementing it. The Certificate Authority keys have to be generated and a new CA certificate has to be generated. All the Old keys for all users and the daemons have to be generated and new certificates have to be generated and signed by the ACE Certificate Authority. It's almost the same situation as though the Certificate Authorities keys had been compromised. Hence, the decision to change keys / algorithms has to be taken keeping in view the administrative overhead also apart from pure technical decisions.

4.2.1.2 Security Margin

The security margin of the key with the corresponding algorithm relates to how easy / hard it is to break the algorithm with the given key size. In this case, the straightforward method is to choose the longest key, which would take a few thousand years to break. Unfortunately, such a simple solution is only applicable to most offline non real-time operations / architectures. For instance, generating a digital signature for an E Mail is an offline non-critical operation. If a user feels a 2048 bit key would be more secure, there would be no adverse affect on performance. However, the same is not true in the ACE. A daemon that takes more time to perform the required operation is not effective. As mentioned before, doubling the public key results in a slowdown in performance by six times, something that is intolerable in real time applications. The ACE solution has to be much more sophisticated, with an analysis for the requirements for each situation. The key manager is hence designed to issue keys of all sizes for almost all the algorithms. An instance of an ACE situation is cited here. Envisage a scenario where a wireless camera with limited processing power is present in the ACE. Instructions are to be given to it so that it may turn around 60 degrees. Such instructions need to be encrypted while being transmitted. Among the host of algorithms and key sizes, we need to choose one that does not impose a computational strain on the wireless camera and achieves the objective (the command being processed) in near about the same time as it would when the command sent to it is not encrypted. With a choice between 128-bit SSL

and 40-bit RC4, which one would we choose? The answer in this scenario could very well be the 40-bit RC4 key. The constraint being that the command be in air for a very short time and the key be discarded after a one-time use. Hence a 40-bit RC4 key makes sense in terms of available computational power and time. This is so despite the fact that breaking a 40-bit key would take about 18 minutes!!!

It should be noted that real time in this context means real time service to the ACE users. It is not used in the same context as real time operations like controlling an airplane.

4.3 PKI in ACE

4.3.1 Introduction

In this section we shall present the limited PKI services that have been deployed in the ACE. We'll first examine the need for a PKI in the ACE. The central issue with PKI is that of identity management. Users can authenticate themselves with the help of an IDMonitor daemon inside the ACE domain or the Remote Connection Manager from Outside the Domain. A user-password pair or an IButton Serial Number or a fingerprint ID before does authentication, as mentioned. Digital (X509) certificates can also be used for authentication although the use of digital certificates for authentication is presently not implemented in the ACE. Digital certificates serve a two-fold purpose. Whenever a user who is already logged in needs to obtain authorization to access a daemon, he can present the certificate as his credential. There is no need for a new authentication handshake. The PKI also helps the users of an ACE domain to operate on the Internet and hence acts as a gateway to the outside non-ACE world. A registered ACE domain can act as an identification mechanism to the outside world (if the ACE Root certificate is a trusted one). The other advantage of using a PKI in the ACE structure is one of hierarchy. A university may have multiple ACE domains connected by a master ASD. In such a situation, scalability issues are naturally addressed by means of hierarchy. Trust management in

such a hierarchical structure is best handled by a PKI. In a situation where a few million nodes are involved, the magnitude of the number of users is likely to be of the same order. It is scarcely possible for an issue of Authentication across domains to be handled by means of fingerprint IDs or IButton IDs. A certificate based authentication and hence role based access control is ideal. The issue of authentication and identification in a hierarchical structure reduces to determining a chain with a few certificates at best. The other alternative would be to search among a million user name / password / IButton / Fingerprint Ids. Such a solution would be costly in terms of time and also in terms of having such a huge database online. We shall now proceed to the certificate authority in the ACE and describe its functions and limitations.

4.3.2 PKI Components

The basic components of a PKI are described here. A PKI essentially consists of the following components:

4.3.2.1 Security Policy

A Security policy of the PKI describes the policy of the organization towards issuing and managing its certificates. It describes the business practices of the organization and the manner in which relegates authority to issue certificates, manage keys etc. Disaster Recovery Plans are also mentioned in the Security Policy. The ACE does not have a Security Policy as of now.

4.3.2.2 Registration Manager

A Registration Manager is an interface between the user and the Certificate Authority. It authenticates the user and determines the level of trust that may be placed on the user depending on the methodology of authentication and the

Security Policy. The registration authority also determines if a certificate can be issued to the user for the specific purpose that the user may want. For instance, in the ACE, the user logging in from outside the domain could be granted a certificate (temporary i.e. for a limited time) for the purpose of a transaction like hearing the audio stream in a conference, but may not be granted a certificate that could be used for a strong authentication key negotiation. In the ACE, inside a single domain, the administrator adds users manually the first time. Hence, in this situation, the administrator plays the role of a Registration Authority. There is no daemon as such that separately authenticates users for a certificate. The Remote Connection manager and the ID Monitor handle authentication. All users by default have a RSA Key pair and a certificate created for them the first time they are registered into the ACE.

4.3.2.3 Certificate Authority

The Certificate Authority is the main daemon in ACE that issues the digital X509 Certificates. It takes care of generating a RSA Key Pair for the user and then generates a X509 Certificate for the user. The generated certificate is an all-purpose certificate for the user. The ACE Certificate Manager upon startup first checks if there is already a key store. If there is one, it attempts to load it with the password provided. If the Keystore doesn't load, the daemon terminates operation with an Error Signal. If the Keystore loads up, the Certificate Manager checks for the existence of the ACE Master Certificate and then starts its operational loop. In its operational loop, the Certificate Authority daemon waits for an incoming request to issue a certificate. Upon such a request, it issues a RSA Key pair and a certificate to the entity (user or daemon) in question. The Key Pair and the certificate are also stored locally onto the disk. It is this Keystore that the daemon attempts to load in case it crashes and starts up again. The daemon also supports methodologies to revoke a certificate and create a certificate revocation list.

The basic X.509 v3 format was completed by ISO/IEC and ANSI X9, which is described below in ASN.1:

```
Certificate ::= SEQUENCE {
    tbsCertificate       TBSCertificate,
    signatureAlgorithm   AlgorithmIdentifier,
    signature            BIT STRING  }
```

The ASN.1 definition of tbsCertificate is:

```
TBSCertificate ::= SEQUENCE {
    version         [0]  EXPLICIT Version DEFAULT v1,
    serialNumber         CertificateSerialNumber,
    signature            AlgorithmIdentifier,
    issuer               Name,
    validity             Validity,
    subject              Name,
    subjectPublicKeyInfo SubjectPublicKeyInfo,
    issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
                         -- If present, version must be v2 or v3
    subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                         -- If present, version must be v2 or v3
    extensions      [3]  EXPLICIT Extensions OPTIONAL
                         -- If present, version must be v3
    }
```

Given below are a sample X509 certificate and the details of the same. The details below correspond to the ACE Master Certificate.

Table 4-3: ACE Master Certificate (X509 V3) details

Issuer Name	***OU = Research & Development*** ***CN = ACE: ITTC Domain*** ***T = Certificate Authority*** ***C = USA*** ***L = Lawrence*** ***E = ace@ittc.ku.edu*** ***O = University of Kansas***
Subject Name	OU = Research & Development CN = ACE: ITTC Domain T = Certificate Authority C = USA L = Lawrence E = ace@ittc.ku.edu O = University of Kansas
The issuer and the Subject name are the same here since this is the ACE Master Certificate.	
Signature Algorithm	Md5RSA

Public Key: RSA (2048 bits)	This is a simple bag of bits
Thumbprint Algorithm	Sha1
Thumbprint	20EC 6221 2AEF C381 96C5 59B1 8FBF E631 D88C 6CFB

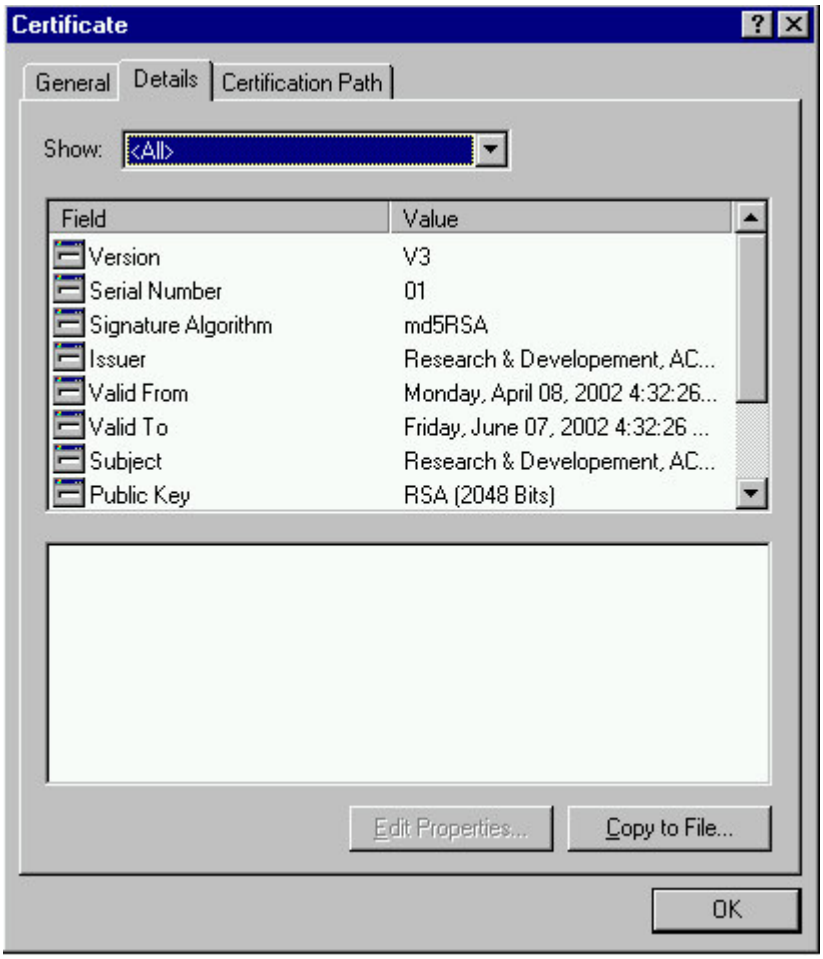


Figure 4-1: ACE Master Certificate

The certificates issued to the users are typically the same except for the change in the Subject Name.

4.3.2.4 Certificate Distribution System

The Certificate Distribution System is responsible for making available the certificates of all the users in a publicly available repository. It is essential in this context that the Certificate Distribution system place the certificates in a standard directory service (preferably in something as common as a LDAP server) so that outside parties (outside the ACE domain) can also access the certificate list. The certificate distribution system also has one other very important function, which is to publish the Certificate Revocation List. The Certificate Revocation list tells the outside world that a certificate, which appears to be all right on all fronts, is actually a faulty one because it has been revoked for some reason (mostly the user would have left the ACE domain or his private key would have been compromised). The Certificate Revocation List is simply a list of the serial Numbers of the certificates that have been revoked. The ACE Certificate Authority signs it. Since the Serial Numbers are unique across the ACE Domain, it uniquely identifies the certificate that has been revoked. It should be noted that in the ACE domain, the daemon that goes under the name of the ACE Certificate Authority is also capable of answering queries regarding the revocation of a certificate. Hence, it also acts as the Certificate Distribution system inside the ACE Domain.

4.4 Remote Authentication Scenario

This section describes the typical scenario in local and remote authentication processes in the ACE. The Remote Connection Manager manages the remote authentication procedure. This service waits at a known address (host/port). A standalone application from a remote host makes a connection to the RCM. As

mentioned before, they talk over the SPEKE protocol. In this particular case, the message exchange goes as below:

- The RCM service sends through a list of protocols it supports to the standalone application
- The application replies with the protocol that it can engage over (at present, they both support only SPEKE), the user name and the protocol specific parameters, which in this case are:
    - The Prime Number P
    - The calculated value $Q_A = S^{(2 R_A)}$ (refer Table 4.1)
- In this case, the application would have also calculated the session key by this stage
- The RCM calculates the Key and replies with the double hash of the key
- The application verifies the double hash of the key and replies with the single hash of the key

We note a few issues here. Since the ACE code is written in Java, the RCM service is safer than most applications from typical buffer overflow problems. There is also a design issue is this case where the standalone application sends the Prime Number P. Ensuring the number sent is a prime with the desired properties is an issue here. The RCM will have to ensure that it does not fall prey to a poorly sent prime number. Java is also susceptible to numerical overflows and underflows. Strict checking has to be ensured at the implementation level. We'll also present an alternate scenario. We could have had the RCM send the Prime Number signed with its private key. The connecting application would have to connect to the LDAP service, obtain the services (RCM) certificate, verify the prime number P and proceed with the remaining steps as usual. But then, an active attacker with the capability to spoof the RCM could very well spoof the LDAP service also. Hence this approach does not offer any significant advantage in terms of added security.

5 Analysis of ACE Security

5.1 The Three-step process

This section looks at ACE security in a broad perspective and examines objectively its security needs and the solutions offered. The evaluation is conducted in a three-step procedure [11]. We begin by examining the security needs of ACE. We look at the situation from the end users and the system architects viewpoint and hence make a definitive statement on the issue at hand. We then look at the solution that has been offered in the ACE domain. The thesis analyses how well the solution offered actually solves the problem. In deciding "how well" the solution offered solves the problem, we take into account issues like timing, scalability and flexibility into issue. The third part of the section looks at the new problems (if any) that are added on due to these solutions. It examines the burden on the domain in terms of decrease in efficiency (if any) and makes suggestions for relief in terms of the parameters (time, computation etc). It is up to the implementer to modify/change any existing scheme to fit into these recommendations.

5.1.1 What problem are we attempting to solve?

The concept behind ACE, as described in the introduction section is one a all pervasive networking environment where users have their computing needs solved and available "in the environment" itself. To facilitate this, the Ambient Computational Environment has computational devices and other accessories throughout the Environment. Hence the ACE is populated with computers with display screens, IButton receptacles, Fingerprint monitors, cameras, projectors, speakers, microphones etc. A user by definition can access his working files from anywhere in the ACE domain, can play music, chat, conference and do a host of other operations. To facilitate this, the ACE architecture is built to have a number of daemons running on all these computers (essentially desktop items). All these daemons perform a predefined function. They provide a set of services in the ACE.

As part of providing these services, they are required to access certain information from the computer and are required to control devices that are attached to the computer. All these daemons communicate with each other over the ACE network. Providing security in the entire context is the essential problem statement of this thesis. On a finer scale, the problem can be divided into the three AAA parts. The first part is that of a user who access the computer. The user has to be authenticated into the ACE domain. How do we accomplish this authentication? Where from does the user log into the ACE domain? Is it from "inside" the ACE domain or from "outside"? How do we identify the daemons? These are the issues that are addressed in this thesis. The issue of user identity in the ACE domain and across multiple ACE domains is also addressed. The problem of access control after the user has been logged in is not addressed in this thesis. The second part is one of ensuring secure communications in the ACE network. As mentioned before, all the daemons are in constant communication with each other. Furthermore, whenever a user access his desktop / workspace, he is presented with a VNC session. All communications between the daemons and the VNS server and client need to be encrypted. For this purpose, we need a daemon that can issue keys for encrypting communications between any two parties in the ACE. That is the second part of the problem we try to solve.

5.1.2 How well does the solution offered solve the problem?

The solution offered in the ACE in terms of the above problems is as follows. The first issue broadly is user authentication. ACE solves this problem by three methodologies. The first one is the normal and ubiquitous user name password pair. Authentication is performed through a security protocol embedded in between the daemon in the ACE taking in connections and the stand alone application attempting to make the connections. This type of authentication is for a user logging in from outside the ACE domain. All users inside the ACE domain have an IDMonitor daemon running in the background on the host where they wish to log in. A

fingerprint ID or an IButton performs authentication here. Here, communications involve transfer of data between the different daemons in the ACE. These communications are encrypted by means of SSL, the keys for which shall be distributed by the Key Manager. This is part of the solution to encrypting communications between two parties. The Key Manager is a service in the ACE that responds to services requesting for keys of different lengths (different cryptographic algorithms and hence different purposes. For instance, a 2048 bit RSA Key pair is most likely to be used for a X509 certificate whereas a 128 bit symmetric key might be used for encryption of messages between two or more nodes/entities in the ACE. In addition to these services (the Remote connection manager and the Key Manager), which take care of Remote Authentication and key distribution in the ACE, there is a Certificate Authority in the ACE that issues digital X509 certificates to users and daemons in the ACE domain. This Certificate Authority forms part of the PKI solutions offered in the ACE. This forms part of the user / daemon identity management solution. Authentication could also be done by means of these X509 certificates. They allow the creation and easy user management in a system with thousands of ACE domains and a few million users. Furthermore, they allow ACE users to interact and use a common identity with the outside world. The Public Key Infrastructure in ACE hence allows the users to also allow a common interface (digital certificates) to identify themselves with the outside world AND with the ACE domain (remotely too if need be).

On a concluding note, we may say that the measures in ACE solve the problem of user authentication and identification very well, satisfying the ACE requirements very well. The IButton and Fingerprint Ids are an overkill with BOTH of them being incorporated, but they probably will have some addition administrative uses in the future.

5.1.3 What new problems does it add?

In this section, we examine the new problems (if any) added due to the incorporation of the above security measures in ACE.

Authentication

ACE resolves the issue of user Authentication by these methodologies

- IButton
- Fingerprint ID
- Passwords
- X509 Digital Certificates

5.1.3.1 IButton Problems

The problems associated with IButtons are apparent and have already been mentioned before. IButtons have a unique Serial number etched into them. Only an IButton receptacle can read this serial number. While this provides adequate protection against duplicating the IButton, it doesn't help if the IButton is stolen from the user. A "stolen IButton" is a social engineering problem rather than a technical problem. However the ramifications on the technical side are more disastrous. If a user identifying himself with an IButton is granted full access control as his status allows, the users entire workspace is compromised. This is a very real problem in the real world as the IButtons are designed to fit in the Key chain or more exotically in digital jewelry (IButton rings). Such items are pretty much easy to steal or be compromised by means of social engineering. The technical solution to these problems of social engineering is effective only to a small extent. One solution is to put users logging into the ACE domain from OUTSIDE the domain in a restricted access space. These users will have their normal permissions with some critical ones cut off. They may, for example have permissions to create and Edit a file, but not

permanently destroy one. Access over the ACE resources should be similarly restricted. The problem introduced due to this lies in the domain of access control. Additional implementation has to be done to ensure that other "domains" of users are also available for access control where the users are physically one and the same, but are digitally different!

5.1.3.2 Fingerprint Woes

Biometric systems in general are prone to many problems as the technology that has evolved is not time tested or error proof. The biometric authentication scheme used in the ACE is a fingerprint scanner. Fingerprint woes are slightly different. According to the latest information, they can very easily be bypassed with a small amount of social engineering, which is not even perceptible as stealing an IButton would be [10]. Methods for bypassing them range from the very simple to the more sophisticated ones. One approach is to record the fingerprint data and try and replay it in the ACE authentication mechanism. Fingerprints especially are very easy to tap into and making an artificial copy in some cases do not take more than 24 hours. Such exploits have been provably demonstrated too. Another approach (a very simple one) relies on the fact that the fingerprint scanners use the skin as a capacitive layer to detect the fact that a finger is indeed pressed on the scanner. When a user logs in with a finger impression, he leaves an impression on the fingerprint scanner. Such a latent image can be used to dupe the system by simply breathing slowly on the scanner. The scanner verily recognizes the "fingerprint". A third approach entails sniffing the port on which the sensor system (Fingerprint Monitor) has been connected to. The sensor system can then be effectively bypassed and artificial data can be replayed into the ACE daemon that awaits the sensor data. Such an exploit would however be applicable to IButtons also and would also require the user to have appropriate permissions on the computer system where the sensor has been connected. These issues demand a technical solution from the manufacturers side rather than the programmer's side. On the whole, it mandates that a fingerprint-

authenticated user also be placed in a restricted access domain than a user authenticated by a X509 certificate or a user name password. This restriction may be even more than that of an IButton authenticated user.

5.1.3.3 Other Issues

User Name Password pairs and X509 certificates do not impose any new problem technically. It should be noted that as part of the final goal, ACE should integrate with the OS itself in all aspects. Hence User name / password combinations are the same as that of the OS parameters. Hence, only other methods of authentication such as the IButton, fingerprint ID etc need to be stored in a database specific to the ACE and apart from the OS. All the protocols in ACE are designed so that even a weak password provides a strong password based authentication (SPEKE protocol). X509 certificates do impose the issue of a safe storage of the private key, but it is expected that a person coming into the ACE domain with a certificate / RSA Key pair would log in from inside the ACE domain or from outside the ACE domain with his own computer. Hence the private key would definitely be in a protected place. However, there are a few extraneous issues that are discussed in section 5.2 with regard to X509 and other implementations. Passwords still remain the Achilles heel of almost all the security systems in the world [9]. The administrative solution to these problems is user education on choosing good passwords. That said all security mechanisms are usually a fine balance between user convenience and strong security.

5.2 Extraneous Considerations

One of the major considerations in Security today is the implementation issue. A good security protocol is considered good on paper if the math behind it is unbreakable. Implementation issues bring up security holes ranging from serious buffer overflow problems to degradation in performance due to various factors. One of them is the effect of API's. In this section we shall how

implementation issues of security schemes affect the performance of systems in general.

5.2.1 Effect of APIs

We mentioned that one of the parameters that was considered in choosing a key size (and hence the cryptographic algorithm) were the speed of computation (and hence lower time taken for the operation). In real world applications however, the implementation details also bring about a significant impact on performance. In the case of the Remote Connection Manager, after a session key has been derived, the API used for the encryption / decryption process matters. Any key schedule (in the API), for instance should contain a reference to the key material, not the actual key itself. A call that accesses the key itself can take a significant amount of time. In cases where dynamic negotiation of the cryptographic algorithm itself is addressed, a call to the encrypt routine itself is under consideration. It is still not clear how such issues will be resolved with java, the language used in ACE development.

5.2.2 Miscellaneous Issues

One other extraneous security issue worth mentioning that has recently cropped up is related to X509 certificates. These digital certificates and a wider range of data on the net are encoded in ASN.1 format (Abstract Syntax Notation) ASN.1 has now been diagnosed to have a fault in the way it has been implemented. “There were people who knew there were problems with the parse, but they weren't security people, so they didn't know it was a security problem.” Says Steve Bellovin. The practical ramifications still remain unclear and much more on how ACE with its PKI and Certificate Authority would be affected by it. The flaw seems to occur with a malformed incoming message. The ACE daemons shuttle data across the network among themselves and the Remote Connection Manager is the only daemon that is open to the outside world. While no accurate assessment can be made at this juncture on the effect on ACE, it is an issue definitely worth monitoring. It also bears

significance on the debate of using proprietary APIs to developing ACE's own raw code for critical operations. On a broader scale, commenting on Security itself, it must be realized that it is a process, not a product. There is no single point of in the software development cycle of ACE where we can say "The ACE is completely secure. Nothing more needs to be done". Security cannot be delivered once and for all as a product (Bruce Schiner). As ACE development goes on, new measures at securing the Environment should be investigated and implemented. Older security schemes should be upgraded to thwart the latest attacks. That will make ACE truly secure in the long run.

6 Conclusions and Future Work

This thesis presents the security scenario in an Ambient Computational Environment. It describes our effort to implement Authentication and secure communications in the ACE Infrastructure. It also describes the limited set of PKI implementations in the ACE. In this prototype, we have successfully implemented the following.

- Certificate Authority
- Key Manager
- Remote Connection Manager

While these services have been implemented, (unfortunately) the majority of the daemons have not been changed to take advantage of these services yet. Hopefully if ACE is resurrected, something could be done in that regard.

We suggest some future works that are yet to be implemented to make ACE a completely secure environment.

1. ACE needs a proper user interface that can manage all the communications to the different daemons that run in the background. These applications shall be required to assist the user to completely use the ACE resources making the authentication and coordination procedure transparent.

2. PKI aware applications need to be implemented so that ACE users can interface with the outside world and also build in support for future devices such as smart cards etc so that functions like retrieving private keys from the ACE keystore can be automated and made much more secure. These applications must fuse tightly with the existing security features in the OS (like PKI in Windows 2000)

3. Our implementation of the Key Server/ Manager is a single point of failure. One probable future work would be to implement a multiple key manager solution keeping in mind the requirements in section 2.2.3.

References

[1] Kaufman, Perlman & Speciner “Network Security: Private Communication in a Public World”, Prentice Hall ISBN 0-13-061466-1

[2] David P. Jablon, "Strong Password-Only Authenticated Key Exchange" Computer Communication Review, October 1996, Volume 26, Number 5

[3] P. MacKenzie, “On the Security of the SPEKE Password-Authenticated Key Exchange Protocol” Bell Laboratories, Lucent Technologies

[4] Kelsey, Schneier and Wagner “Protocol Interactions and the Chosen Protocol Attack” Security Protocols Workshop ’97, number 1361 in Lecture Notes in Computer Science, pp. 91-103.

[5] Kurosawa, Okada, Sakano, “Security of the Center in Key Distribution Schemes” Advances in Cryptology | ASIACRYPT ’94, number 917 in Lecture Notes in Computer Science, pp. 333 - 341.

[6] Michael Roe, “Performance of Protocols” Security Protocols Workshop ’99, number 1796 in Lecture Notes in Computer Science, pp. 140-152.

[7] C. Boyd and W. Mao, “Design and Analysis of Key Exchange Protocols via Secure Channel Identification” Advances in Cryptology | ASIACRYPT ’94, number 917 in Lecture Notes in Computer Science, pp. 171 - 181.

[8] F. Stajano and R. Anderson, “The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks” Security Protocols Workshop ’99, number 1796 in Lecture Notes in Computer Science, pp. 172-194.

[9] R. Morris, K. Thompson, “Password Security: A Case History”, Communications of the ACM, Vol. 22, Nov. 1979, pp. 594-597.

[10] Bruce Schneier, “Fun with Fingerprint Readers” Cryptogram Newsletter, May 15 2002. http://www.counterpane.com/crypto-gram-0205.html#5

[11] Bruce Schneier, “How to think about Security” Cryptogram Newsletter, April 15 2002. http://www.counterpane.com/crypto-gram-0204.html#1

Appendix A: Code Samples

ACE Daemon Configuration File

```
asd=lemondrop.ittc.ku.edu:10500
roomname=nowhere
buildingname=nowhere
port=/dev/ttyS1
javax.net.ssl.keyStore=$(TopDir)/etc/endeavour.key
javax.net.ssl.keyStorePassword=passphrase
javax.net.ssl.trustStore=$(TopDir)/etc/cacerts
javax.net.ssl.trustStorePassword=passphrase
edu.ku.ittc.ACE.ACEConnection.keytype=DES
edu.ku.ittc.ACE.ACEConnection.cipher=DES/CBC/PKCS5Padding
edu.ku.ittc.ACE.ACEDaemon.debug=false
edu.ku.ittc.ACE.ACEDaemon.UseSSL=false
edu.ku.ittc.ACE.ACEDaemon.PolicyAssertions=$(TopDir)/etc/policyassert
edu.ku.ittc.ACE.ACEDaemon.Authorizer=$(TopDir)/etc/authorizer
edu.ku.ittc.ACE.ACEDaemon.PrivateKey=
edu.ku.ittc.ACE.ACELibrary.debug=false
edu.ku.ittc.ACE.ACERoom.ConfigDirectory=$(TopDir)/etc
edu.ku.ittc.ACE.Implementation.ACENetworkLogger.logfile=/tmp/ace/logFile.txt
edu.ku.ittc.ACE.Implementation.ACENetworkLogger.logserver=pigpen:10000
edu.ku.ittc.ACE.Implementation.ACECertificateAuthority.MasterKeyStoreLocation=/users/krsna/.ace/CertificateAuthority/ACEKeyStore
edu.ku.ittc.ACE.Implementation.ACECertificateAuthority.RootCertificateLocation=/users/krsna/.ace/CertificateAuthority/ACEMasterCertificate.cer
edu.ku.ittc.ACE.Implementation.ACECertificateAuthority.UserDaemonRSAKeyStoreLocation=/users/krsna/.ace/CertificateAuthority/UserDaemonRS
AKeyStore
edu.ku.ittc.ACE.Implementation.ACECertificateAuthority.UserDaemonCertificateStoreLocation=/users/krsna/.ace/CertificateAuthority/UserDaemonCe
rtificateStore
edu.ku.ittc.ACE.Implementation.ACECertificateAuthority.CertificateRevocationListLocation=/users/krsna/.ace/CertificateAuthority/ACECertificateRe
vocationList.crl
edu.ku.ittc.ACE.Implementation.ACECertificateAuthority.UserCertificatesLocation=/users/krsna/.ace/CertificateAuthority/User/
edu.ku.ittc.ACE.Implementation.ACECertificateAuthority.DaemonCertificatesLocation=/users/krsna/.ace/CertificateAuthority/Daemon/
```

```
edu.ku.ittc.ACE.Implementation.ACECertificateAuthority.KeyStorePassword=ACE
edu.ku.ittc.ACE.Implementation.ACECertificateAuthority.SerialNumberLocation=/users/krsna/.ace/CertificateAuthority/SerialNumber.txt
edu.ku.ittc.ACE.Implementation.ACECertificateDistributionSystem.LoginDN=cn=root,dc=ku,dc=edu
edu.ku.ittc.ACE.Implementation.ACECertificateDistributionSystem.LoginPassword=secret
edu.ku.ittc.ACE.Implementation.ACECertificateDistributionSystem.Container=dc=ku,dc=edu
edu.ku.ittc.ACE.Implementation.ACECertificateDistributionSystem.Host=129.237.127.131
edu.ku.ittc.ACE.Implementation.ACECertificateDistributionSystem.Codebase=/users/krsna/.ace/webpages
edu.ku.ittc.ACE.Implementation.ACEKeyManager.UserDaemonKeyStoreLocation=/users/krsna/.ace/KeyManager/UserDaemonKeyStore
edu.ku.ittc.ACE.Implementation.ACEKeyManager.Password=ACE
edu.ku.ittc.ACE.Implementation.PostgresDatabase.JDBC_Driver=org.gjt.mm.mysql.Driver
edu.ku.ittc.ACE.Implementation.RoomDatabase.DatabaseName=jdbc:mysql://localhost/roomdb
edu.ku.ittc.ACE.Implementation.ACEServiceDirectory.DatabaseName=jdbc:mysql://localhost/asd
edu.ku.ittc.ACE.Implementation.AuthorizationDatabase.DatabaseName=jdbc:mysql://localhost/authdb
edu.ku.ittc.ACE.Implementation.ACEUserDatabase.DatabaseName=jdbc:mysql://localhost/userdb
edu.ku.ittc.ACE.Implementation.ACEWorkspaceServer.DatabaseName=jdbc:mysql://localhost/workspacedb
edu.ku.ittc.ACE.Implementation.DatabaseUser=ace
edu.ku.ittc.ACE.Implementation.DatabasePassword=43gyFqzYa
edu.ku.ittc.ACE.Implementation.ReserveConnections=3
edu.ku.ittc.ACE.Implementation.ACESystemResourceMonitor.appResourcesNeeded=$(TopDir)/etc/ACESRMFile.csv
```

Sample A: Generating a Key of the Specified Algorithm

```
if (KeyParameters.equalsIgnoreCase("AES"))
{
        try
        {
                KeyGenerator keygen = KeyGenerator.getInstance("Blowfish");
                if ( (keySize !=0) && (keySize % 8 == 0) )
                {
                        keygen.init(keySize,new SecureRandom());
                }
                else
                {
                        // Default Key Size
                        keygen.init(new SecureRandom());
                }
```

```
                // Use it to generate a key
                k = keygen.generateKey();
                byte[] rawkey = k.getEncoded();
                SymmetricKey = new String(rawkey);
        }
        catch(Exception e)
        {
                System.out.println(e);
                System.exit(-1);
        }
}
```

Sample B: Generating a X.509 Digital Certificate

```
private X509Certificate generateX509Certificate(String AliasName,Hashtable subjectDN,int notBefore, int notAfter)
{

        RSAPrivateKey   privKey = null;
  RSAPublicKey    pubKey = null;
  try
  {
                // First Generate the RSA Key Pair !
          KeyPair         pair = generateRSAKeyPair(2048);
          privKey = (RSAPrivateKey)pair.getPrivate();
          pubKey = (RSAPublicKey)pair.getPublic();

          X509V3CertificateGenerator  certGen = new X509V3CertificateGenerator();
                // Recall that the serialNumber in the
                // generateACEMainCertificate Function was set to 2
                // So the first time this function is called, the serialNumber shall have a value of 2
          certGen.setSerialNumber(serialNumber);
          // Increment the serialNumber
          serialNumber = serialNumber.add(BigInteger.valueOf(1));
                String serialNumberLocation = ACEConfiguration.getConfigValue( SERIAL_NUMBER_LOCATION );
```

```
        FileOutputStream serialNumberFile = new FileOutputStream(serialNumberLocation);
// This is written unencrypted !
// Encrypt this Later
// This is done so that the next time the CA loads, it can figure out the
// last Serial Number it had assigned !
serialNumberFile.write(serialNumber.toByteArray());
certGen.setIssuerDN(new X509Principal(issuerDN));
if (notBefore == 0)
{
        certGen.setNotBefore(new Date());
}
else
{
        certGen.setNotBefore(new Date(System.currentTimeMillis() - 1000L * 60 * 60 * 24 * notBefore));
}
certGen.setNotAfter(new Date(System.currentTimeMillis() + 1000L * 60 * 60 * 24 * notAfter));
certGen.setSubjectDN(new X509Principal(subjectDN));
certGen.setPublicKey(pubKey);
certGen.setSignatureAlgorithm("MD5WithRSAEncryption");
X509Certificate cert = certGen.generateX509Certificate(aceMasterprivKey);
// The Certificate has been generated
// Store the certificate and the RSA Key Pairs
// Set the Key Entries ( Both Public and Private )
// Also set the Certificate Entry
        Certificate[] chain = new Certificate[2];
        chain[1] = masterKeyStore.getCertificate("ACE Master Certificate");
chain[0] = cert;
        // Store the Keys First
        UserDaemonKeyStore.setKeyEntry(AliasName + " Public Key", pubKey,passwd,chain);
        UserDaemonKeyStore.setKeyEntry(AliasName + " Private Key", privKey,passwd,chain);
        String UserDaemonKeyStoreLocation = ACEConfiguration.getConfigValue( USER_DAEMON_RSA_KEYSTORE_LOCATION );
        FileOutputStream UserDaemonKeyStoreFOS = new FileOutputStream(UserDaemonKeyStoreLocation);
        UserDaemonKeyStore.store(UserDaemonKeyStoreFOS, passwd);

        // Store the Certificate
        UserDaemonCertificateStore.setCertificateEntry(AliasName + " Certificate",cert);
```

```
                String UserDaemonCertificateStoreLocation =
ACEConfiguration.getConfigValue(USER_DAEMON_CERTIFICATE_STORE_LOCATION );
                FileOutputStream UserDaemonCertificateStoreFOS = new FileOutputStream(UserDaemonCertificateStoreLocation);
                UserDaemonCertificateStore.store(UserDaemonCertificateStoreFOS, passwd);

                // Now write the Certificate just as a plain file
                // Check if the Alias Name corrosponds to  a User or a Daemon
                String CertificateLocation = null;
                if (AliasName.indexOf(":") == -1)
                {
                        CertificateLocation = ACEConfiguration.getConfigValue( USER_CERTIFICATES_LOCATION );
                }
                else
                {
                        CertificateLocation = ACEConfiguration.getConfigValue( DAEMON_CERTIFICATES_LOCATION );
                }
                CertificateLocation = CertificateLocation+AliasName+".cer";

                FileOutputStream fOut = new FileOutputStream(CertificateLocation);
                fOut.write(cert.getEncoded());
                // This is the Private Key  that is ACTUALLY
                // going back to the service that requested the certificate
                // Note that this is a HUGE design issue !
                // This is necessiated because the RSA keys are generated by the CA
                // We could also have the services generate their own RSA key pairs
                // request for a certificate ..... but that leads to a whole new set of
                // problems !
                tempPrivateKey = privKey;
          return cert;
        }
        catch (Exception E)
        {
                // Somethings Wrong !
                // You cant generate the Certificate
                E.printStackTrace();
                return null;
```

```
        }
}
```

Sample C: Certificate publishing in LDAP service

```
public void GeneratedCertificateCmdNotify( String GeneratedCertificate, String Type)
{
   try
   {
                String theContainer =
                ACEConfiguration.getConfigValue( LDAP_CONTAINER );
                System.out.println(GeneratedCertificate);
                System.out.println(Type);
                ByteArrayInputStream inStream = new ByteArrayInputStream(decode(GeneratedCertificate.toCharArray()));
                CertificateFactory cf = CertificateFactory.getInstance("X.509");
                X509Certificate cert = (X509Certificate)cf.generateCertificate(inStream);
                LDAPAttribute attribute = null;
                LDAPAttributeSet attributeSet = new LDAPAttributeSet();
                java.security.Principal subjectDN = cert.getSubjectDN();
                String subName = subjectDN.getName();
                String title =  subName.substring(subName.indexOf("T=")+2,subName.indexOf(",",subName.indexOf("T="))) ;
                String title[] = {title};
                attribute = new LDAPAttribute( "title", title );
          attributeSet.add( attribute );
          // Get the Daemon Address later
          String[] ip = { "Contact ASD at lemondrop.ittc.ku.edu:10500"};
          if (Type.equalsIgnoreCase("user"))
                {
             String objectclass_values[] = { "ACEUser" };
             attribute = new LDAPAttribute( "objectClass", objectclass_values );
             attributeSet.add( attribute );
             attribute = new LDAPAttribute( "userCertificate;binary", encode(cert.getEncoded()));
                   attributeSet.add( attribute );
```

```
                String dn = "dc="+x.substring(x.indexOf("CN=")+3,x.indexOf(",",x.indexOf("CN=")))+"," +theContainer;
                        LDAPEntry newEntry = new LDAPEntry( dn, attributeSet );
                lc.add( newEntry );
            }
            else
            {
                String objectclass_values[] = { "ACEService" };
                attribute = new LDAPAttribute( "objectClass", objectclass_values );
                attributeSet.add( attribute );
               attribute = new LDAPAttribute( "userCertificate;binary", encode(cert.getEncoded()));
              attributeSet.add( attribute );
              String dn = "dc="+x.substring(x.indexOf("CN=")+3,x.indexOf(",",x.indexOf("CN=")))+"," +theContainer;
                        attribute = new LDAPAttribute( "ip", ip );
              attributeSet.add( attribute );
              LDAPEntry newEntry = new LDAPEntry( dn, attributeSet );
              lc.add( newEntry );
            }
        }
        catch (LDAPException e)
        {
                e.printStackTrace();
                if (e.getLDAPResultCode() == 68)
                {
                        // Whats happening ? The certificate was already there !
                        // Somethings badly worng !
                        // The CA is issuing 2 certificates with the same DN
                        System.exit(-1);
           }
        }
        catch (Exception E)
        {
                E.printStackTrace();
                System.exit(-1);
        }
}
```