HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering
Degree Programme in Computer Science and Engineering

Design and Implementation of an Authentication and Authorization Module for Service Access in Ad Hoc Networks

Master's Thesis

Jimmy Kurian

Telecommunications Software and Multimedia Laboratory
Espoo 2005

HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Computer Science and Engineering
Degree Programme in Computer Science and Engineering

ABSTRACT OF MASTER'S THESIS

Author:	Jimmy Kurian
Title of thesis:	Design and Implementation of an Authentication and Authorization Module for Service Access in Ad Hoc Networks
Date:	February 17 2005
Pages:	13 + 77
Professorship:	Telecommunications Software
Code:	T-110
Supervisor:	Professor Antti Ylä-Jääski
Instructor:	M.Sc. (Tech.) Timo Kiravuo, M.Sc. (Tech.) Linda Källström

An ad hoc wireless network is a highly dynamic, self-organizing and adaptive network formed spontaneously, without the help of any infrastructure, between two or more devices equipped with wireless communications and networking capability. The users in ad hoc networks could provide services to each other. This introduces the requirement of a solution to control with whom a service access or request should be done. Traditional authentication and authorization solutions are not feasible in an infrastructure-less ad hoc network.

This thesis consists of the requirement analysis, design, and implementation of an architecture, which provides authentication and authorization, for service access in ad hoc networks. The architecture is based on a de-centralized approach using identity certificates along with local access control policies which is combined with certificate extensions, user identification with public keys, and hierarchical structuring of keys. In this approach, users are authenticated rather than devices. The proposed architecture was proved to be feasible with a successful implementation. This thesis was done as part of the SESSI research project at Telecommunications Software and Multimedia Laboratory of Helsinki University of Technology.

Keywords: ad hoc networks, service access, security, AA, authentication, authorization, access control
Language: English

To my wonderful parents
for their everlasting love and encouragement,
which I relied and enjoyed all through my life.

Acknowledgements

Firstly, I would like to express my deep-felt gratitude to my supervisor, Prof. Antti Ylä-Jääski, for the help and support he gave me during this thesis. I also thank him for providing me the wonderful opportunity to work in this research project.

This thesis would not have reached this state without the support and encouragement of my instructors, Linda Källström and Timo Kiravuo. I am delighted to thank Timo for his clear explanations of the concepts and valuable comments, which helped to shape this thesis. His guidance was invaluable to the completion of this work. A special thank goes to Linda whose suggestions, comments, and guidance made this thesis a success. I really thank her constant support, enduring patience and enthusiasm, which she bestowed all through my thesis.

I thank Sanna Liimatainen for her suggestions and timely advice, which kept me focused in my research. I would also like to thank the rest of the SESSI team for a great project and a memorable learning experience.

I would like to thank my parents, my sister, and my brother for their endless love, encouragement, and support they gave me all through my life. I thank, with all my heart, my fiancee for her patience, understanding, and encouragement for finishing this thesis. I also want to thank all my great friends for making this world a better place.

Last, but not least, I thank God almighty for being supportive in all my endeavors.

Espoo, February 17th 2005

Jimmy Kurian

Abbreviations and Acronyms

3G	3rd Generation
AA	Authentication and Authorization
AAA	Authentication, Authorization, and Accounting
ACL	Access Control List
AES	Advanced Encryption Standard
CA	Certification Authority
CHAP	Challenge Handshake Authentication Protocol
CRL	Certificate Revocation List
DES	Data Encryption Standard
DNS	Domain Name System
DSA	Digital Signature Algorithm
IP	Internet Protocol
IPsec	IP security
MD5	Message Digest, Version 5
NAS	Network Access Server
OCSP	Online Certificate Status Protocol
PDA	Personal Digital Assistant
PEM	Privacy Enhanced Mail
PKI	Public Key Infrastructure
PPP	Point to Point Protocol
RADIUS	Remote Authentication Dial-In User Service
RSA	Rivest-Shamir-Adelman
SCTP	Stream Control Transmission Protocol
SHA	Secure Hash Algorithm
SIP	Session Initiation Protocol
SLP	Service Location Protocol
SPKI	Simple Public Key Infrastructure
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UI	User Interface

Glossary of Terms

This section provides the glossary of terms used in this document. Those terms, which are defined as part of this thesis, are indicated in the glossary by an asterisk.

AA module* The authentication and authorization module developed as part of the SESSI framework. This thesis consists of the requirement analysis, design, and implementation of this module.

Access control The mechanism for controlling access to resources.

Access control rule The rule or information for controlling the access to resources.

Access Type* Denotes whether the service is accessed or provided. The AA module provides two-way access control.

Accounting The act of collecting information on the usage of resources in such a way that each entity is held responsible for their actions.

Application Certificate* The cryptographic keys required by a service. An application certificate could be an asymmetric certificate or a symmetric key. A user or group will have several application certificates used for different services.

Authentication The mechanism for confirming the identity of an entity, or the integrity and authenticity of an information.

Authorization The act of granting access for a particular entity to a particular resource by checking whether that entity has access rights to that resource.

Availability The mechanism for assuring that a resource is available to an authorized entity when required.

Base Certificate* The identity certificate which binds the name of the user or group with the base public key. Thus, the base certificate provides the base information for identifying the user or group. The base certificate of a user could be a CA signed or self-signed certificate. The base certificate of a group would be signed by the administrator of the group.

CertApplInfo* The identifier for the application certificates. Since a user has multiple application certificates, the CertApplInfo is required to specify which application key should be used for a particular service access.

Channel A means of transferring information between entities. A secure channel is a channel in which an unauthorized entity cannot read or alter information. An insecure channel is a channel which could be accessed by all entities, including authorized and unauthorized entities.

Confidentiality A mechanism to ensure that the data is not revealed to anyone who is not authorized to see it.

Crypto Service module* The cryptographic module developed as part of the SESSI framework. The module is a wrapper to the *OpenSSL* cryptographic library. The Crypto Service module provides a convenience interface to common cryptographic functions.

Data integrity A mechanism to control alteration of data by unauthorized entities.

Denial of Service (DoS) A form of attack in which a malicious entity prevents the access of services by a legitimate entity.

Device The communicating instrument which the user uses for communication. Devices in ad hoc networks could be PDAs, laptops, mobile phones etc.

Device User* The person who owns the device, when talked from a device's point of view.

Eavesdropping A form of attack in which a malicious entity listens to the data communicated between two entities. The malicious entity does not try to modify the data.

Entity Someone who receives, sends or alters information. An entity could be a user or device.

Entity authentication A mechanism which gives the guarantee to one entity that the other entity is exactly the same entity as it claims to be.

Group* A group mechanism introduced by the AA module. A group resides globally across the devices and will act just like a user in an ad hoc network. Each group will have an *administrator* who creates the group and manages its members.

Group ID* The group ID identifies the group globally across the devices. The group ID is generated from the base certificate of the group.

Impersonation A form of attack in which a malicious entity acts like an authorized entity, and access the information or services available to that authorized entity.

Message authentication A mechanism to guarantee the integrity and authenticity of the data delivered over a channel. Same as data origin authentication.

Message modification A form of attack in which a malicious entity alters the message. The alteration of messages could be by deletion, modification or insertion of messages.

Non-repudiation A mechanism which ensures that an entity cannot deny a previous communication or action done.

Receiver The entity who receives the message or data.

Replay A form of attack in which a malicious entity listens to the communication between legitimate entities, and re-transmits those messages later.

Role* The identities in which the device user is able to appear in the network. The device user can appear as the user herself, or as one of the groups in which she is a member.

SD module* The Service Discovery module developed as part of the SESSI framework.

Security level* The level of security which should be applied for a particular service access. The SESSI framework specifies three levels of security (see Section 2.4.1).

Sender The entity who sends the message or data.

Service* Any service running in a device. It could be a generic service or an infra-structural service or a connectivity service.

Service ID* The unique string which identifies a service globally across different users. In the AA module, service IDs are created by using a naming scheme similar to the Java package-naming scheme.

SESSI* SESSI stands for *Seamless Service Interworking in Heterogeneous Mobile and Ad Hoc Networks*. The goal of the project is to create technological premisses for seamless interworking of services and applications in third generation (3G) cellular networks and in wireless short-range networks.

SIP SIP stands for Session Initiation Protocol[19]. SIP is an application layer signaling protocol for establishing, modifying and terminating network sessions.

SLP SLP stands for Service Location Protocol[13]. SLP provides a decentralized, lightweight, scalable, and extensible framework for service discovery and selection.

SM module* The Session Management module developed as part of the SESSI framework.

Traffic Analysis A form of attack in which a malicious entity monitors the data traffic, and gets the pattern of the communication which could be used later.

User* A user is a person in the network, who is using a device for communication.

User ID* The user ID identifies the user globally across the devices. The user ID is generated from the base certificate of the user.

Verified User* The user who is verified for any role of the particular device user.

X.509 X.509 is an ITU-T recommendation. It specifies a widely used digital certificate format.

Contents

List of Figures

Chapter 1

Introduction

Wireless network communications have shown tremendous growth in the past decade. It introduced the widespread usage of personal devices such as PDAs, mobile phones, and laptops with various wireless communication interfaces. Data services which were earlier provided through fixed wireline networks have started shifting to wireless networks. 3G networks were designed to provide resources and functionalities required for the provisioning of sophisticated wide-band data services to mobile users. However, all these networks and services depend upon network infrastructures.

Another networking paradigm, which is emerging in parallel, is ad hoc networks, where devices self-organize to form a network and communicate with each other, without the help of any network infrastructure or centralized administration. The network could be formed instantaneously wherever a wireless network is required with the help of various wireless access technologies like Bluetooth, IEEE 802.11, or Hiperlan2. The users of the devices could provide services to each other. The services offered could range from a simple chat to military grade services used in a battlefield. Users could join the network, and access or provide services from or to other users. This functionality brings up the requirement that there should be some means for a user to control from whom a service should be accessed or to whom a service should be provided. Further, the lack of infrastructure and the shared nature of the wireless medium introduce several security requirements in ad hoc networks for service access. Not much research has been done in this area of ad hoc networks. Most of the research activities were done for routing technologies and the associated security issues. This thesis focuses on the security issues related to service access and proposes an authentication and authorization (AA) architecture for service access in ad hoc networks.

1.1 Background

An ad hoc wireless network is formed by a group of two or more devices equipped with wireless communications and networking capability. Devices that are within each other's access range communicate directly via wireless links, while those devices which are far apart rely on intermediate devices to forward the messages from the source towards the destination. Ad hoc devices are capable of detecting the presence of other such devices and perform necessary handshaking to allow communications and the sharing of information and services. An ad hoc wireless network does not rely on any fixed network infrastructure and is self-organizing and adaptive. This means that a formed network can be re-formed on the fly without the need for any system administration. The devices in the network are free to move arbitrarily, joining or leaving the network at any point of time, resulting in a highly dynamic network topology.[2]

The ability of an ad hoc mobile device to act as a server, providing a service, will depend on its computational power, memory, storage, and battery capacity. Devices in an ad hoc network can be of different types such as PDA, laptop, mobile phone etc., with varying power levels and functionalities. The heterogeneity of these devices implies that some devices are more powerful than others, and some can act as servers providing some service while others can only be clients.[2]

The users in the ad hoc networks could provide many services, which could be accessed by other users in the network. Further, the users could be both service providers as well as service requesters. There should be some mechanism to control the access of these services by only authorized users. It should be possible to verify the identity of the users to assure that the communicating user is the one that it claims to be. Further, there should be some mechanism, which makes sure that the communication is not altered by or disclosed to an unwanted user.

Security is an important issue in ad hoc networks. As said in [18], security in ad hoc networks is difficult to achieve, mainly because of the vulnerability of wireless links, the limited physical protection, the sporadic nature of connectivity, the dynamic topology and membership, the absence of a certification authority, and the lack of a centralized monitoring authority.

The security requirements for different services will vary in an ad hoc network. They mainly depend upon the purpose for which a particular service is used, and upon the environment in which it is used. There might be some services, which need not be protected at all and there might be some other services, which handle very sensitive information. Thus, the nature of the service and the degree of hostility of the ad hoc environment determines what level of security should be applied.

1.2 Research Problem

This thesis was done during the period February 2004 - December 2004 as part of the SESSI project[30, 29, 27, 28, 26] at Helsinki University of Technology. SESSI stands for *Seamless Service Interworking in Heterogeneous Mobile and Ad Hoc Networks*. The goal of the SESSI project is to provide technological premisses for seamless interworking of services and applications in third generation (3G) cellular networks, wireless ad hoc networks, and heterogeneous networks that expand 3G network with wireless short range networks. The project mainly concentrates on providing functionalities for service discovery, session management, and authentication and authorization in different network architectures.

The SESSI networking environment mainly consists of the following three network architectures:

1. **Ad Hoc Networks**: In ad hoc networks, there are no infrastructures available. Some or all nodes may provide services. Nodes join and leave the network at any point of time. This network does not have any servers for providing IP addresses or answering name queries. All nodes are considered as equal, and are together responsible for providing the required infrastructural services.

2. **3G Networks**: The 3G networks provide different kinds of services and centralized servers are part of the network. These will have connection to the global internet.

3. **Heterogeneous Networks**: The heterogeneous network consists of both infrastructured networks and ad hoc networks. The nodes in the ad hoc networks might have some connection points, known as access points, to the infrastructured networks. Through these access points, nodes in the infrastructured networks and ad hoc networks can access services from each other. The infrastructured network could be for example a 3G network, the global internet, or a company intranet.

The SESSI project has a span of two years (January 2004 to December 2005). The development cycle of SESSI was divided into two phases with each phase consisting of one year. This thesis was done as part of the first phase of the SESSI project. The goal of the first phase of SESSI project is to enable provisioning of data services over link-local ad hoc networks. This requires modification of existing services for use in a distributed manner and without centralized servers. Support for service discovery should be provided, since a user in the network needs to find out which all services are available. Session management is required for initiating a session between users. An authentication and authorization structure is required to handle several security issues in the network without centralized servers. Trust relationships need to be built among the peers. Both the service discovery and session management would rely on this authentication and authorization structure to provide the required security for service access in ad hoc networks. Thus, the SESSI project investigates the following three main research areas: service discovery, session management, and authentication and authorization.

1.2.1 Objective of this Thesis

The main objective of this thesis is to develop a security architecture, which provides authentication and authorization for service access in ad hoc networks. This becomes an interesting research topic since not much research is already done in this field of ad hoc networks. Even though there are different solutions available for infrastructured networks, none were found to adequately solve the requirements identified for ad hoc networks. Thus a specialized module needs to be developed which adequately solves the requirements. The development cycle includes the requirements analysis and the subsequent design of the module. Feasibility of the proposed design needs to be verified with successful implementation.

1.2.2 An Example Scenario

In this section, we will see an example scenario, which would benefit from the authentication and authorization module developed by this thesis. Alice, Bob, and Carol, who works in the same company, are attending a business conference. Each of them has their own laptops, and forms an ad hoc network. During the conference, Alice invites Bob and Carol for a private chat between them. Bob and Carol see Alice's invitation and they join for the chat. They exchange information and ideas between them through chat messages. Eve, who is working in another company, is also there for the conference. She would like to know what is going on in the chat. She tries to join for the chat, but her request is rejected. She tries to understand the chat messages between the other three, but is not able to do that also. She even tries to intercept the chat by altering some messages, and sending some messages as if Carol sends them. However, all those messages are rejected. Thus, Eve fails despite all these attempts.

The example scenario given above suggests the various security measures, which needs to be taken. We will look at them from Alice's point of view. Alice has to be sure that the other two users with whom she is chatting are Bob and Carol, and not anyone else. It should not be Eve acting as Bob or Carol. No other users other than Bob and Carol should be able to join for this private chat. Since this is a private chat, no other users listening to the data traffic should be able to understand the conversation. Further, it is also required that no other user should be able to modify or insert messages in such a way that it seem to be originated from Bob or Carol.

1.3 Scope of this Thesis

This thesis mainly concentrates on the issues related to service access in ad hoc networks. It proposes a security architecture, which provides authentication and authorization in ad hoc networks for service access. This thesis will not address solutions to provide accounting, nor the issues related to routing. Routing security is intensively researched and have many solutions proposed. This thesis will rely on some publicly available cryptographic and database libraries for the implementation.

1.4 Outline of this Thesis

This chapter serves as an introduction to the entire thesis. The rest of the thesis is organized as follows. Chapter 2 presents the concepts and technologies that provide the basis for the techniques used in this thesis. The requirements analysis for the design of the architecture, proposed by this thesis, is done in Chapter 3. Chapter 4 presents the proposed architecture, and Chapter 5 explains the implementation of that design. Chapter 6 evaluates and discusses the design and implementation done in this work. Finally Chapter 7 presents the conclusions from this research work.

Chapter 2

Background

This chapter outlines the concepts and technologies that provide the basis for the techniques used in this thesis. Section 2.1 presents various scenarios where ad hoc networks are used. It also discusses the inherent characteristics of ad hoc networks, which makes them vulnerable to various attacks, and the various security goals to be achieved in ad hoc networks.

The next two sections, Section 2.2 and 2.3, introduce two architectures which provide security services in infrastructural networks. The functionalities they provide need to be understood, and analysed to check whether they could be used in ad hoc networks, where centralized authorities do not exist, to provide authentication and authorization. Section 2.2 presents the Public Key Infrastructure (PKI) which enables secure communication, in an unsecure network, through the use of public key cryptography. It explains Certification Authorities (CAs) and their role, the structure of an X.509 digital certificate, and certificate revocation. Section 2.3 introduces AAA servers, which provide AAA services in infrastructural networks. The analysis of these architectures will be done in the next chapter.

In the last section, the SESSI framework is explained briefly. Since the authentication and authorization solution should be developed as part of the SESSI framework, the basic understanding of the framework is essential for the proper discussion of requirements. The section introduces the *service layers* and the *security levels* defined by SESSI. This section also presents the *Authentication and Authorization (AA)* module which is developed by this thesis for providing authentication and authorization. The AA module is developed as one of the infrastructural service components of the SESSI framework. This section also introduces the other infrastructural service components, namely *Service Discovery (SD)* module and *Session Management*

(SM) module.

2.1 Ad Hoc Networks

Ad hoc networks are desirable wherever a dynamic network for communication between different devices is required. An introduction of ad hoc networks was given in the Section 1.1. The following are some of the scenarios benefiting from an ad hoc network which provides automatic network establishment and service access[7, 21, 15]:

- **Conference**: A group of people attending a meeting or conference, or in a classroom wants a network to exchange data and provide services to each other.
- **Playing games**: People traveling together in a train want to play games or share music.
- **Sensor networks**: Sensor nodes used for data collection needs to communicate to each other.
- **Military operations**: Military forces or soldiers want to communicate to each other in a hostile environment.
- **Emergency services**: People involved in a rescue or emergency operation in an area where communication infrastructure is not available and wants a network for immediate communication.
- **Personal use**: A group of people want to communicate and exchange data without using infrastructural networks, which are either unavailable or expensive.

2.1.1 Characteristics of Ad Hoc Networks

The inherent characteristics of ad hoc networks makes them vulnerable to various attacks such as impersonation, eavesdropping, message modification, traffic analysis, replay, and denial of service. This section presents the main characteristics of ad hoc networks, and the resulting challenges posed by them[20, 31].

Vulnerable Wireless Link
The wireless link in ad hoc networks makes them susceptible to passive and

active attacks. Unlike wired networks, which have several lines of defense like firewalls and gateways, attacks on a wireless ad hoc network can come from any direction and can target any device. Passive attacks, like eavesdropping, violate confidentiality, giving access to secret information. Active attacks ranges from message modification and replay to impersonation, which violates authentication, integrity, availability, and non-repudiation. This means that every device should be prepared for protecting itself from these attacks.

Multi-hop Communications
In ad hoc networks, distant devices rely on multi-hop routes to communicate with each other. It is not usually possible to establish direct communication between devices due to the limited range of the wireless access technologies used. The data packets travel through other peer devices in the network before reaching the intended recipient. Further, due to the lack of infrastructure, routing algorithms are used to establish the routes and most of these algorithms are co-operative in nature[10]. This poses several attacks resulting from introducing false routing information or making all information routed through an adversary device.

Lack of Centralized Authority
Malicious nodes can take advantage of the situation that lacks a centralized authority. There are no authorized devices, which constantly monitors the network for malicious activity or which controls the adversary devices. All the networking functions are distributed among the devices in the network and rely on co-operative algorithms for the proper functioning of the network. Adversaries can design attacks, which breaks these cooperative algorithms.

Hostile Environment
The devices in an ad hoc network are autonomous units, which roam around in a hostile environment. Devices lack physical protection and are always under the threat of being captured and compromised. Further a device is always under the threat of being attacked by other malicious or compromised devices in the network. This means that every device should be prepared to operate in a mode that trusts no peer. Security solutions in a cooperative manner are always under risk. Every device should be capable of making its own security decisions without trusting other peer devices.

Dynamic Topology and Membership
An ad hoc network is very dynamic because of frequent changes in both its topology and membership. Devices join and leave the network at any time. Due to this, selecting certain devices to become administrative authority does not always work because they can always leave the network at any time. Further, there can be situations where compromised devices are

elected for becoming the administrative authority or the administrative devices becoming compromised. There are different solutions proposed in ad hoc networks, like a distributed architecture, where a group of devices in the network function together in a co-operative manner to provide server like functionalities. However, all these solutions seem too complicated and assume the presence of a minimum number of devices in the network, which is another serious limitation.

Limited Resources
The devices used in ad hoc networks are mostly personal hand-held devices, and hence have limitations in the resources such as battery life, network bandwidth, and processing power. Even though these resources tend to increase with advancement in technology, they are very less when compared to those of fixed wired network devices. The solutions proposed for ad hoc networks should be designed for optimized resource consumption. Thus complex mechanisms, which are usually resource hungry, are not feasible in ad hoc networks.

2.1.2 Security Goals in Ad Hoc Networks

This section introduces the various security requirements and the related concepts, which form the goals to be achieved for securing an ad hoc network.

Authentication

Authentication is a mechanism for confirming the authenticity of an entity or message. It provides some means for guaranteeing that a specific entity is who it claims to be or a specific message is exactly the same as it was sent by the entity who originated the message. Thus it involves entity authentication and message authentication (data origin authentication).

Entity Authentication and Identification: Entity authentication gives the guarantee to one entity that the other entity is exactly the same entity who it claims to be. Entity authentication involves identification, which is recognizing the identity of the other entity. Authentication is the process of verifying this identity. In ad hoc networks, since entities join and leave the network anytime, there is more chance for adversary entities to impersonate a legitimate user. The lack of infrastructure and centralized authority further elevates this problem.

Message Authentication and Data Integrity: Message authentication, or data origin authentication, refers to the integrity and authenticity of the data delivered over a channel. It provides the assurance to the receiver about the identity of the sender. Data integrity provides assurance to the receiver that the message received is exactly as sent by an authorized entity, and was not altered by any unauthorized entities. Alteration of data could be accidental alteration or malicious alteration, and involves activities such as insertion, deletion, or substitution of data. In ad hoc networks, malicious users might send messages as if they were originated from some legitimate users. During multi-hop routing in ad hoc networks, data passes from the sender to the receiver through several devices. This introduces the risk that an adversary in the route can alter the data prior to forwarding it further.

Authorization and Access Control Rules

Authorization is the act of granting access for a particular entity to a particular resource by checking whether that entity has access rights to that resource. Authorization decisions are based on access control rules defined for that resource. The rules are based on various properties of the resource, or on the credentials of the requesting entities. These rules are checked before granting authorization. The authorization granted could be, for example, network access or usage of a particular service. In an ad hoc network, since different users could be running various services, an authorization mechanism is required to control access to services.

Confidentiality

Confidentiality ensures that the data is not revealed to anyone who is not authorized to see it. Some security sensitive application might contain confidential information, which needs to be protected. In ad hoc networks, data transfer occurs through air interface, and is thus more susceptible to passive attacks such as eavesdropping or monitoring of data. Further, the need for multi-hopping in ad hoc networks requires that the data go through different devices before reaching the user to whom it was intended.

Non-repudiation

Non-repudiation ensures that an entity cannot deny a previous communication or action done by that entity. In case of a dispute, there should be some means to prove whether or not a particular entity was involved in a communication or action. In ad hoc networks, a user should be held responsible for the actions performed. Since there are no infrastructural authorities to control the actions in an ad hoc network, there should be some means to prove, at a later point in time, that a particular action took place.

Availability

Availability is a mechanism for assuring that a resource is available to an authorized entity when required. It ensures that a malicious entity cannot withhold the resources and prevent authorized entities from having access to those resources. Thus, the major concern of availability is to prevent denial of service attacks. Availability could be a major concern in ad hoc networks. An adversary could request for a service all the time making the user's device busy, thus forming a denial of service attack. Availability becomes a major issue when it comes to routing in ad hoc networks.

2.2 Public Key Infrastructure (PKI)

A *Public Key Infrastructure (PKI)* is a combination of technologies, software, and services which provides a total security architecture with the help of public key cryptography. A PKI makes secure communication possible in an unsecure network by providing authentication, integrity, confidentiality, and non-repudiation. It involves different components like digital certificates, methods which facilitate the creation of asymmetric key pairs and issuance of digital certificates, certificate authorities who issue the digital certificates, the registration authorities who check the validity of a certificate request, different certificate publishing methods and repositories for retrieving certificates, tools for certificate management like suspending, renewing or revoking certificates, methods for evaluating a chain of certificates, and various other related services and support.

2.2.1 Certification Authority (CA)

In public key cryptography, even though the key distribution is easier with public keys, it is very important to get the correct public key of the person with whom the communication is to be done. A solution to this problem is provided with the help of a trusted authority called *Certification Authority (CA)*. CA issues digital certificate that binds the name and the public key of a user. The certificate is issued based on the *certificate request* submitted by the user along with her credentials. The CA takes the responsibility for checking the trustworthiness of the identity and the credentials provided by the user. There are different classes of certificates issued based on the level of checking performed. The certificates are signed with CAs private key, and thus could be verified by any person who possesses the CA's public key. Since the certificates are signed, it cannot be modified without being detected. An adversary cannot create a certificate and impersonate the user. The main advantage of this system is that the user can store the certificate in an insecure location, like a directory service, or can provide her certificate directly to others when required. Further, the CA need not be online for others to verify a user.

2.2.2 X.509 Digital Certificates

X.509 digital certificates[24] are the most widely used standard for digital certificates. X.509 digital certificates are part of the X.509 framework[17]. The certificates are created and signed by a trusted Certification Authority (CA) and it binds a user with her public key. The person is identified with a globally unique and meaningful X.501 type[16] name. The latest version of X.509 certificates is version 3, whose structure is shown in Figure 2.1.

The following are the fields of the X.509 certificate:

Version identifies the version of the certificate. Version 3 is given when extensions are present. Version 2 is used when no extensions are present, but the unique identifier fields for Issuer and Subject are present. Version 1 is the default version in which no extensions and unique identifier fields are present.

Certificate Serial Number is an integer value assigned for the certificate by the issuing CA and is unique within the certificates issued by that CA.

Signature Algorithm Identifier The identifier for the algorithm used to sign the certificate together with any necessary parameters.

Issuer Name The X.501 type name of the CA who signed and issued this certificate.

Validity Period A pair of date-time values which shows the period during which the certificate is valid.

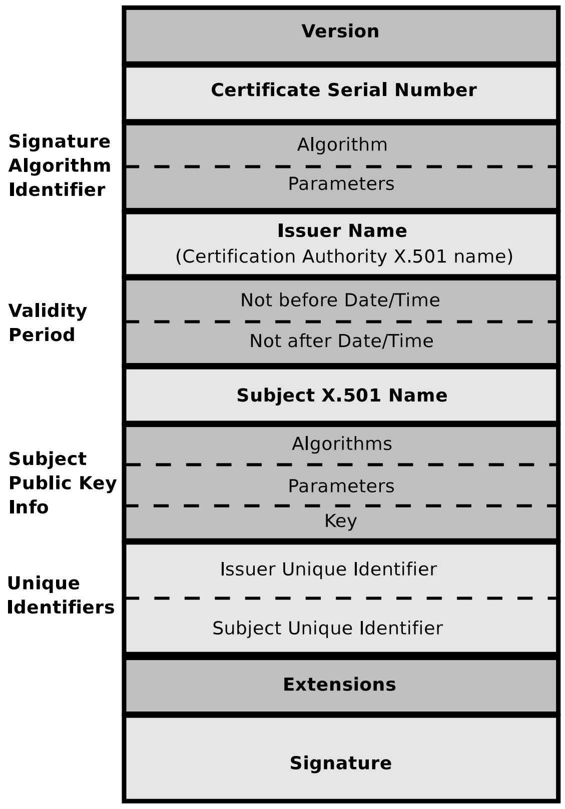

Figure 2.1: X.509 Certificate structure

Subject Name The X.501 type name of the user to whom the certificate is issued.

Subject Public Key Info The information about the public key of the user which includes the public key, the identifier for the algorithm used and any necessary parameters.

Unique Identifiers The unique identifiers for the issuing CA and the user. This allows the reuse of issuer and subject names over time. These fields are present from certificate version 2 onwards.

Extensions A set of extension fields which provide additional information. This field is present only in version 3 certificates.

Signature The digital signature of the CA for the hash value of all other fields. The signature algorithm identifier information is repeated again in this field.

2.2.3 Certificate Revocation

Even though certificates have a period of validity, it is desirable under various circumstances that a certificate must be revoked prior to its expiration date. The private key of a person or CA getting compromised, or a CA does not wish to certify a person anymore are examples of such situations. Certificate revocation is done with the help of *Certificate Revocation Lists (CRLs)* [14], which contain information about revoked certificates, and are signed and published by CAs periodically. Each CA thus maintains a list of revoked, but not expired, certificates issued by that CA. Each entry for the revoked certificate contains the serial number and the revocation date of that certificate. In lieu of or as a supplement to checking against a periodic CRL, it may be necessary in security-sensitive services, to obtain timely information regarding the revocation status of a certificate. This could be achieved with the help of *Online Certificate Status Protocol (OCSP)* [22] which provides the certificate status in real time. However, this requires network connection to the centralized servers running the certificate validation protocol.

2.3 AAA Servers

Authentication, Authorization and Accounting (AAA) servers are capable of handling requests for resource access and provide AAA services. AAA servers can play the role of authenticating users onto the network, authorizing the appropriate level of service to them, and providing accounting.

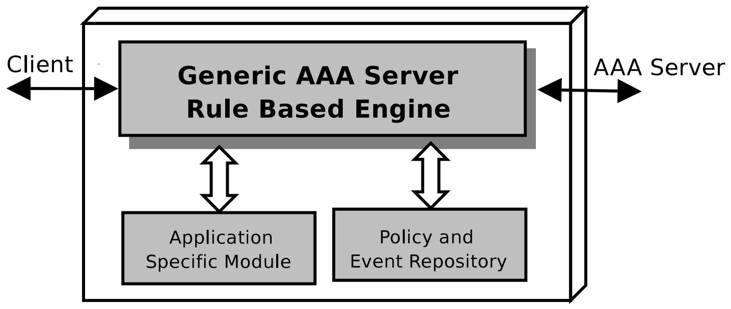

Figure 2.2: Generic AAA Architecture

Generic AAA architecture as described in [8] is shown in Figure. 2.2. The architecture is mainly divided into the following components:

Generic AAA Server defines the AAA functionality common to all applications.

Application Specific Module (ASM) defines application specific functionality. The ASM manages resources and configures the service equipment to provide the authorized service. It also takes part in making the authorization decision because it has the required application specific knowledge.

Event Log stores the events which occur in the AAA server. With the aid of certificates, this database could support non-repudiation.

Policy Repository is a database which contains information and policy rules for making authorization decisions on available services and resources.

As explained in [8], an authorization goes through the following steps:

1. The user or another AAA server submits a well-formatted request to the AAA server.
2. The AAA server inspects the contents of the request, determine what authorization is requested, retrieve the policy rules from the repository, perform various local functions, and then choose one of the following options for further processing of each component of the request:
   (a) Let the component be evaluated by an attached ASM.
   (b) Query the authorization event log or the policy repository for the answer.
   (c) Forward the component(s) to another AAA server for evaluation.

2.3.1 AAA Protocols

Two main AAA protocols are *Remote Authentication Dial-In User Service (RADIUS)* [25] and *Diameter* [4]. RADIUS is widely deployed and is pretty lightweight when compared to Diameter. Diameter, on the other hand, has more desirable features, but at the same time is more complex.

RADIUS

RADIUS is a pure client-server model protocol in which *Network Access Server (NAS)* acts as the client. The client passes user information to designated RADIUS servers. The RADIUS servers are responsible for receiving user connection requests, authenticating user, and then returning all the configuration information necessary for the client to deliver service to the user. Originally, RADIUS was designed to support dial-up PPP and terminal server access, but nowadays it is being used in many varying scenarios. The authentication mechanism in RADIUS is flexible and can support PPP PAP or CHAP, UNIX login, and other authentication mechanisms. Transactions between the client and the RADIUS server are authenticated through the use of a shared secret, which is never sent over the network. All the transactions comprise of variable length Attribute-Length-Value 3-tuples. The RADIUS protocol is extensible in such way that new attribute values can be added without disturbing existing implementations of the protocol.

Diameter

Diameter is defined as the successor of RADIUS and it removes known flaws of the RADIUS protocol. It is a peer-to-peer protocol in the sense that any node can initiate a request. Diameter is designed to be extensible. The basic concept behind Diameter is to provide a base protocol that could be extended in order to provide AAA services to new access technologies, and it focuses mainly on satisfying the requirements of network access [6]. The base protocol is used for accounting. Authentication and authorization are provided by application specific extensions. Different extensions to the base protocol could be made, which facilitates the usage of different access technologies. Failover behavior is well defined in Diameter and it supports application-layer acknowledgements, and defines failover algorithms and the associated state machine. It supports IPsec and TLS and it is a much more secure protocol than RADIUS. Diameter runs over more reliable transport mechanisms such as TCP and SCTP. This helps in congestion control as compared to RADIUS, which runs over UDP. Further, Diameter supports server-initiated messages and error messages. Diameter enables dynamic discovery of peers through DNS, and provides secure and scalable roaming. Diameter is much more advanced in compatibility issues and provides backward compatibility with RADIUS such that the two protocols may be deployed in the same network.

2.4 SESSI Framework

The module for providing authentication and authorization services should be developed as part of the SESSI framework. As explained earlier in Chapter

1, in the first phase the SESSI project concentrates to enable provisioning of data services over link-local ad hoc networks, modifying the existing services for use in a distributed manner and without centralized servers. The SESSI framework is shown in the Figure 2.3.

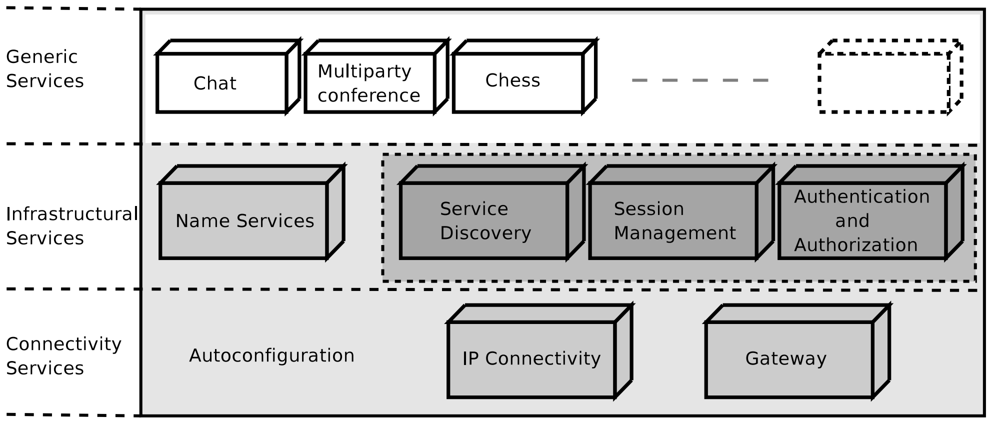

Figure 2.3: The SESSI framework

The SESSI service framework consists of three service layers: *connectivity services layer*, *infrastructural services layer* and *generic services layer*. Each layer is further divided into modules. The modules on each layer can use the functionality of those modules on the same layer or under it.

Connectivity Services Layer This is the lowest layer in the SESSI service framework. The modules in this layer provide link-local or external network connectivity for other parts of the SESSI framework. IP connectivity and gateway functionality are the main responsibilities of this layer.

Infrastructural Services Layer This layer forms the core of the SESSI framework. The modules in this layer provide an abstraction layer which hides any changes in the connectivity, thus providing a connection-independent seamless way for locating, setting up, and accessing services.

Generic Services Layer These are the actual services offered by a user to other users. This layer is the top layer of the SESSI service framework. There could be multiple services in this layer. Examples of services in this layer are chat service, games, multiparty conference service etc.

The framework relies on IP connectivity and thus requires an IP address to be assigned before it could be used. It uses *Apple Rendezvous*[1] for host autoconfiguration on IPv4 and for name services on both IPv4 and IPv6. Standard IPv6 neighbour discovery messages will be used for IPv6 address configuration. Although these are not part of the main SESSI focus area, some functionality would need to be developed, to support the whole infrastructure.

The infrastructural services form the core of the SESSI framework. The module, which provides authentication and authorization, is part of this services layer. The following three components should be developed as part of the first phase of the SESSI framework, to provide the basic infrastructural services.

Service Discovery (SD) module The Service Discovery (SD) module is to be developed to provide the service discovery functionality. *Service Location Protocol (SLP)*[13] is chosen as the base protocol for implementing the SD module. The main goal of the SD module is to facilitate service discovery using SLP protocol in the defined SESSI network without centralized servers. The SD module should also contain a service discovery API that is protocol independent. The SD module should conform to the SESSI security levels explained in Section 2.4.1.

Session Management (SM) module The Session Management (SM) module is to be developed to provide session management functionality. The *Session Initiation Protocol (SIP)*[19] is selected as the base protocol for implementing the SM module. The main objective of the SM module is to make SIP work in a decentralized fashion without any centralized servers.

Authentication and Authorization (AA) module The Authentication and Authorization (AA) module is to be developed to provide authentication and authorization services. The AA module could be accessed from all parts of the SESSI framework. The development cycle of this module constitutes my thesis. From this point onwards, we will focus mainly on the AA module. The next chapter discuss in detail about the requirements identified for this module.

2.4.1 SESSI Security Levels

The security requirements for different services will vary in an ad hoc network. The level of security required is determined by the security sensitiveness of the service, and the degree of hostility of the ad hoc environment. The SESSI framework specifies the following three levels of security. All SESSI-compliant services should try to support these three levels. If any level is not feasible to support, then service access on that level is denied.

Security Level: None This is the base security level. In this level no security is provided. It is used for clear text communication between users. This level is preferred when doing a non-sensitive communication or when usage of the device resources is to be kept minimum.

Security Level: Authentication In this level the other user is authenticated and authorized by verifying the user's right to use the service.

Security Level: Confidentiality In this level, all the communication is protected against eavesdropping by encryption. In addition to that, the communicating parties are authenticated and authorized.

Chapter 3

Requirements Analysis

The development cycle of this thesis constituted a requirements analysis phase. The goal of this chapter is to present the requirements identified as a result of that analysis phase. The design of the AA module will be done, based on these requirements presented in this chapter.

Section 3.1 lists the high level goals identified for the AA module. The next section presents the identified set of requirements. It presents the functional, design and interface level requirements. Section 3.3 explains the constraints and assumptions made as part of the requirement analysis. Section 3.4 concludes the chapter by selecting an approach for the basic design of the AA module.

3.1 High Level Goals

The following are the high level goals identified for the AA module. Even though the primary goal of the AA module is to provide authentication and authorization, it is required that it would also function as a security architecture providing other security functionalities, introduced in Section 2.1.2, if possible.

Goal-1: The solution should be able to cope with the dynamic topology and membership of ad hoc networks, as explained in Section 2.1.1.

Goal-2: The solution should not introduce any changes in the underlying protocols and security mechanisms already used by services.

Goal-3: It should be possible to refer to the same service across users in the network (service identification).

Goal-4: The users should be able to make sure that they are communicating with the intended user, and a message received is without modification and its source is as claimed (authentication).

Goal-5: Only authorized users are permitted to use the service (authorization and access control).

Goal-6: The solution should support grouping of users.

Goal-7: Users should be able to use the services without revealing their identity.

Goal-8: The solution should support storage of credentials of different users.

Goal-9: Sensitive information in a device should be protected against disclosure.

3.2 Identified requirements

These are the requirements identified for the design of the AA module. The requirements are identified based on discussions and analysis on the high level goals presented in the previous section.

3.2.1 Functional Requirements

REQ-1: Authentication of users
As stated by *Goal-4*, the AA module should store various credentials and provide various methods which the services could use to authenticate users. The AA module should not implement any authentication protocol as per *Goal-2*. The services, which use the AA module, should be able to use any authentication protocols of their choice.

REQ-2: Message authentication and integrity
According to *Goal-4*, the AA module should provide methods to check the authenticity and integrity of a message. The services, which use the AA module, should be able to verify whether the source of the message received is exactly as claimed and the message received is, exactly as sent from that authentic source, without any alteration.

REQ-3: Authorization and access control
Only authorized users should be able to access or provide a service, as stated by *Goal-5*. The AA module should store various access control rules defined by the user, and should provide various methods for authorization. The

access control rules should be fine-grained and flexible to provide enough control over access decisions.

REQ-4: Security level decision
The AA module should be able to provide the security level decision for a particular service access. Since different services provided by different users will have different security requirements, there should be provision for the user to define the security level required for a service usage. The security levels are explained in Section 2.4.1.

REQ-5: Storage of credentials and information
The AA module should provide methods for the storage of credentials of users, as stated by *Goal-8*. The AA module should also provide storage of related information, as required. These credentials and information will be used for the various functionalities provided by the AA module.

3.2.2 Design Requirements

REQ-6: De-centralized design
The design should follow a de-centralized approach, as per *Goal-1*. The idea of a centralized structure, for providing authentication and authorization services, is not feasible in ad hoc networks. This concludes that the AA module should be designed in a de-centralized manner, where each device has its own AA module, which provides the required authentication and authorization functionalities.

REQ-7: Global identification of users
The AA module should provide some means for the identification of users in the network, as per *Goal-4*. These identification information or IDs provided should be global such that they do not collide in an ad hoc network. Since a malicious user might try to use some other user's identity, it should be possible for a user to prove her identity, if required, to avoid impersonation. Allotting an ID to a user assuring uniqueness only in the current ad hoc network is not a good solution, since smaller ad hoc networks can always dynamically merge to form a bigger network or new users could join the network, which might create collision with another user's ID. Further, a globally unique ID for a user, which is permanent, provides the easiness for allocating access rights to the user.

REQ-8: Global identification of services
As stated by *Goal-3*, the AA module should provide some means for the identification of services in the network. These identification information or IDs provided should be globally unique such that a particular service ID refers to

the same service to all the users in the network. Further, different developers and organizations will have different versions of services developed. However, there is no need for a service to prove that the ID belongs to that service itself. So a unique naming style for services should be devised.

REQ-9: Support for groups
As per *Goal-6*, the AA module should support grouping of users, where a number of users act with the same credentials and identity. Since this thesis focuses on service access, many interesting features could be provided with the help of groups of users such as group communication, providing service access to a specific group etc.

REQ-10: Privacy of users
Privacy of users should be provided, as per *Goal-7*. They should be able to access services without revealing their real identity.

REQ-11: Internal security
As required by *Goal-9*, the AA module should protect sensitive information and keys from being exposed to other processes. These information and keys should be encrypted and stored. The encryption should be based on the information provided by the user of the device, and this information should not be stored in the device.

REQ-12: Support for multiple cryptographic algorithms and keys
As per *Goal-2*, the AA module should not restrict the services to use a particular algorithm. Also, it should not assume the same credentials to be used for all services. Thus, the user should be able to use different cryptographic algorithms and credentials for different services. The AA module should have support for symmetric as well as asymmetric cryptography. Further, the AA module should support various cryptographic algorithms and should be designed to support more in the future.

REQ-13: Scalability of design
The AA module should not depend on the availability of other users in the network. There might be situations where there are no other users, or only a few users in the network. On the other hand, sometimes the number of users in the network may even scale up to thousands. Thus, the AA module should be scalable enough to handle these situations, and should not introduce any restrictions on the number of users in the network, as per *Goal-1*.

REQ-14: Extensibility of design
The design should be adaptable to future requirements. Thus, the architecture devised should be able to implement additional features, without disturbing the existing base functionality.

3.2.3 External Interface Requirements

REQ-15: The AA module should be provided as an API
The AA module should be provided as a C-language API. Services can use this API to get the functionalities provided by the module.

REQ-16: Provide functionalities without sharing sensitive information
The AA module should not give out sensitive information, such as keys, to services. It should provide the required cryptographic functionalities using the sensitive information, but not revealing the sensitive information as such.

REQ-17: Provide transparency of algorithms used
The AA module should support multiple algorithms and credentials, as stated in *REQ-12.*. While using the AA module, to obtain the security functionalities, the services should have transparency of the underlying algorithms and credentials used.

REQ-18: External libraries and dependency
The AA module can use external libraries for obtaining required functionalities. However, while using a library, wrappers should be developed to minimize the dependency. This is required to switch to other libraries, if needed, in the future.

3.3 Constraints and Assumptions

This section explains the constraints and assumptions made as part of the requirement analysis. Some of the security goals, explained in Section 2.1.2, were found not to confine to the goals of this thesis.

Trusted Components
Even though, the AA module does not give out sensitive information to services, as stated in *REQ-15*, some components which are part of the SESSI framework will have access to this information. This is required since these components provide some functionality, using this information, to the AA module. These components are called *trusted components* by the AA module. When required, the AA module requests these trusted components for the services they offer, rather than these trusted components ask the AA module to provide sensitive information.

Relying on infrastructures or online-servers
The proposed architecture should function without relying on any online-servers or any other devices. However, there could be some features provided

which relies on online-servers. The architecture should not require frequent contact with these servers for its proper working. The availability of these servers should not in any way affect the proper functioning of the solution. Thus, the architecture may rely on online-servers once in a while or once in a lifetime for providing different functionalities. This is analogous to the example that a wireless device should be charged once in a while with the help of a power socket, but the power socket is not required during the working of the wireless device. Just like that, the AA module can expect that a device in ad hoc networks can come in contact with infrastructural servers, once in a while, to retrieve some information, but the AA module should not rely on these servers for providing the AA functionality during service access in ad hoc networks.

Limitations of Power
The devices in the ad hoc network could be heterogeneous. Some devices might be more powerful than the others. The factors like battery and computational power of devices always tends to grow year after year. However, the solution should not rely on the presence of powerful devices in the network or rely on the assumption that a certain ratio of powerful devices will be present in the network all the time. The user of the device should be able to select a security mode, which requires lesser power or opt for higher levels, according to her requirements. However, the devices can be assumed to satisfy the power requirements for cryptographic functionalities.

Availability
This thesis will not solve the availability issues for service access. It does not confine to the primary goal of the AA module. Further, the protection against various denial of service (DoS) attacks, thus assuring the availability of services to legitimate users is difficult to achieve. For example, a malicious user could send multiple requests, which make the device busy with various security checks. The attack becomes even more severe when it is a distributed denial of service (DDoS) attack.

Accounting
As stated in the scope, in Section 1.3, this thesis will not address solutions to provide accounting. Accounting requirement varies between services and could be implemented by the services, if needed. However, the AA module should use the logging functionality provided by the SESSI framework to keep a log of various activities.

Non-repudiation
Even though cryptographic algorithms could provide the proof of origin of a message, non-repudiation also involves some legal aspects. The AA module

will not address these issues.

3.4 Design Approach

In this section, we will discuss the design approach and technologies, which should be used for the design of the AA module. Various approaches were selected and analyzed. This section briefly goes through those approaches, discussing the strengths and weaknesses of each, and selects the most suitable one.

3.4.1 Approach Alternatives

Based on the requirements stated above, the following three approaches were identified for the basic design. All the three approaches have their strengths and weaknesses of their own.

1. **Design Based on the AAA Protocols**
   Authentication, Authorization and Accounting (AAA) servers are capable of handling requests for resource access, and provide AAA services. The AA architecture demands a centralized server. Infrastructural networks, such as 3G and heterogeneous networks, are capable of providing a centralized server which the AAA architecture demands. However, it is not feasible when it comes to ad hoc networks, which are formed without any infrastructure. One solution is that each device acts as a server as well as a client. Every device has a minimal AAA server installed and acts as a server providing access to itself or acts as a client while trying to access other devices. For Diameter, a separate application might be required, just like [5] and [3], which runs on top of the Diameter base protocol.

2. **Design Based on Authorization Certificates**
   The approach using authorization certificates is based on *Simple Public Key Infrastructure (SPKI)*[12, 11]. The authorization certificates are used to grant access rights to users. The SPKI approach identifies the users by their public keys and is suited well for a de-centralized scenario. The access right is issued in the certificate itself with various attributes, and the person who issues the certificate signs the whole certificate. Delegation of access right is provided using one attribute, which states whether the person to whom this certificate is issued can

delegate that access right to others. This approach seems to suit well to the ad hoc network scenario and the de-centralized approach we need.

3. **Design Based on Identity Certificates and Local Access Control Rules**
   The approach, using identity certificates and local access control rules, also seems to suit well to the ad hoc network scenario and the de-centralized approach we need. The identity certificates are used for the identification and authentication of the user. Access control rules are stored locally on each user's device, and are used to provide authorization. However, this approach lacks features like delegation of access rights.

3.4.2 Design Approach - Evaluation and Decision

This section will briefly evaluate the three identified approaches and decide the most suited approach. The server-based AAA architecture is designed to provide AAA services in an infrastructural network. Feasibility study was conducted for using AAA servers in the proposed de-centralized manner. It was not found to be an efficient solution to use the AAA protocols in the ad hoc network scenario, for providing service access. Implementing an AAA server in each device was not practical in the study. Thus this approach was rejected.

The other two approaches really suits for the solution we require since they support a de-centralized architecture. We will further look into those two approaches.

The approach involving authorization certificates provides various desirable features, such as delegation of access rights. However, it was found not suitable to our solution, due to the following reasons:

- Authorization certificates serve best in a solution where resource access permissions are granted at one place and the resource access is performed at some other place (or places). It has the advantage that the access permissions need not be replicated in all those places. For our solution, this advantage will not help since each user is responsible for giving access permissions to just her device.

- The approach is not flexible enough, if the user needs to change his access permissions with immediate effect. With authorization certificates, the certificates already issued need to be revoked and new certificates should be granted. We require a system, which has flexibility to modify the access rights with immediate effects, and without much effort from the user.

- Our solution targets ad hoc networks with personal devices. It would be useful for the user to actually see what all permissions are given to which all users in her device. With authorization certificates issued, the user would face difficulty in knowing which users have access permissions in her device. Even though logging could solve this, maintenance of the logs introduces additional overhead.

- A certificate is issued to the user, for each access permission she got in other devices. Thus, a user has to deal with several certificates. Storage of large number of certificates and searching through them would affect device performance.

- Each access permission forms a certificate. Thus, each service access involves the overhead of sending the certificate, verifying the certificate and validating the access permissions.

The design based on identity certificates and local access control policies was found to be a much simpler approach. Since certificates are required only for identification and authentication, this approach needs fewer certificates to be handled when compared to the authorization certificate approach. Authorization process is much faster, since the access rules required for authorization are stored locally. Further, it is easier for the user to manage the authorization permissions locally. This approach could be combined with certificate extensions, user identification with public keys, and hierarchical structuring of keys (explained in the next chapter) to achieve several required features.

To summarize, the design based on identity certificates and local access control policies was selected for the basic design of the AA module. This approach was found to be simple and more suitable for our solution.

Chapter 4

Design

This chapter presents the design of the AA module, which provides authentication and authorization in ad hoc networks for service access. The design is based on the requirements identified in the previous chapter. The feasibility of this design will be verified with an implementation, as stated in 1.2.1. The implementation will be described in the next chapter.

An overview of the design is presented in Section 4.1. Section 4.2 explains the various elements of the design like *users*, *services*, *trusted CAs*, *groups*, and *roles*. The application specific information and the access control rules are presented in the next two sections. Section 4.5 lists the information and credentials stored and managed by the AA module. The next two sections present how certificates should be verified and how security level decision is made. Section 4.8 explains the AA module architecture along with its interaction with other modules, and presents its internal security features.

4.1 Design Overview

The AA module is designed in a de-centralized manner, as stated by *REQ-6*. Each device in the network has its own local AA module, which provides the required authentication and authorization functionalities. The design avoids the use of co-operative algorithms, and does not introduce any changes on the protocols and security mechanisms, already used in different services. The AA module could be used by various services, such as generic services, infrastructural services and connectivity services, in the device. The *users* and *services* form the basis of the AA module design. The AA module authenticates and authorizes users, rather than devices. The services in the device rely on the AA module for authorization and security level decisions. The user is able to use multiple credentials for different services, as stated by *REQ-12*. The AA module solves the issue of securely verifying all these credentials by a hierarchical structure of keys. Each user will have one *base key* which identifies the user. All the other credentials of the user will be bound to this base key. Thus, to verify the user, verification of only the base key is required. As per *REQ-9*, the AA module supports *groups*, where a number of users act with the same credentials and identity. It provides various facilities for group creation and management. Figure 4.1 presents the basic structure of the proposed design. The elements of the design, presented in the figure, will be described in the subsequent sections.

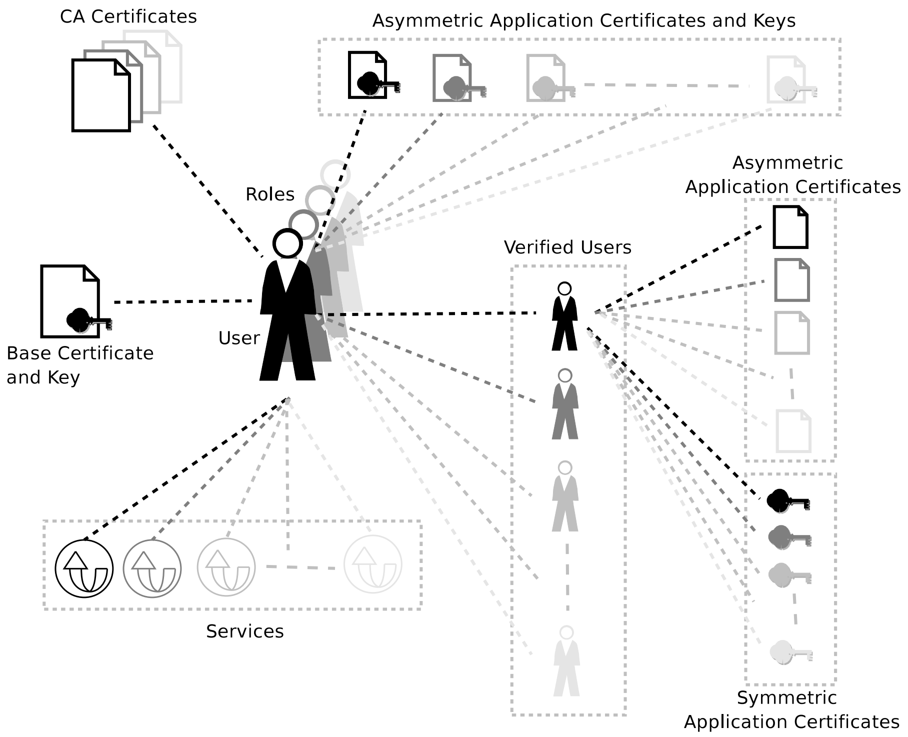
CA Certificates
Asymmetric Application Certificates and Keys
Roles
Asymmetric Application Certificates
Verified Users
Base Certificate and Key
User
Services
Symmetric Application Certificates

Figure 4.1: AA module design elements

4.2 Basic Elements of AA Module

This section presents the basic elements of the AA module such as users, services, trusted CAs and groups.

4.2.1 Users

A *user* is any person who owns a device in the ad hoc network. When talking from one device's point of view, the user who owns the device is known as the *device user*. Each device user will have a set of other users who are verified to the device user. These users are known as the *verified users* of the device user. Identification of users in the network is required, as per *REQ-7*.

Each user will have an asymmetric key pair, known as the user's *base key pair*. The user will have an identity certificate, known as the *base certificate*, which binds the name of the user with her base public key. Thus the base certificate provides the base information for identifying the user. The base certificate could be a Certification Authority (CA) signed or a self-signed certificate.

The AA module relies on asymmetric cryptography for identifying users globally. It is based on the assumption that a public key, or the hash of a public key having proper length, will be a globally unique identifier for the corresponding private key, or for the user who possess it. The AA module will generate a *user ID* from the base certificate of the user. The user ID identifies the user globally across the devices. The user ID is calculated as `HASH[base public key, base user name]`. The common hash algorithm used for this purpose, across all devices, will be SHA-1. Since the user ID is calculated from the base certificate of the user and the user can prove the ownership of the public key, the user can prove the ownership of the user ID also, if required.

4.2.2 Services

The term service is meant to denote any service running in a device. It could be a generic service or an infrastructural service or a connectivity service. A device user can provide several services, which could be accessed by her verified users. The device user could also access the services provided by her verified users. A service, which the device user wants to access or provide, should be first registered in the AA module.

Identification of services in the network is required. They should be uniquely identified globally such that it is possible to refer to the same service across users in the network, as per *REQ-8*. It is achieved by devising a service-naming scheme similar to the Java package-naming scheme. A service is identified by the *service ID*, which identifies the service globally across different users. The service ID is provided by the unique string of the format `Owner:Service:Version` which identifies the service. The `Owner` identifies the organization or individual who created the service. The `Service` gives the name of the service. The `Version` is required to keep various non-compatible versions of a service separate. Examples of service IDs could be `com.apple:quicktime:v1` and `fi.hut:sessichat:v3`.

Non-SESSI aware services are identified using their *service type string*, as defined in [9]. The service type string of a service is formed using its *abstract type*, *naming authority*, and *concrete type*. The abstract type describes the type of the service. The naming authority is the name of the organization that named the service. The concrete type is a sub-type of the abstract type.

4.2.3 Trusted CAs

A user may have a set of Certification Authorities (CA), whom she trusts. The user's AA module will have information and certificates of those CAs. The CA information will be used for the verification of base certificates of other users. By default, the base certificate of a user is a CA signed identity certificate.

4.2.4 Groups

The AA module supports *groups*, where a number of users act with the same credentials and identity, as stated by *REQ-9*. Just like a user, a group will have a base key pair and base certificate. Each group will have an *administrator*, who creates the group. When the group is created, the AA module of the administrator generates the key pair, and creates and signs the base certificate of the group. The design proposes a certificate extension, which binds the base certificate of a group with its administrator. The extension provides the additional information `SESSI:GROUP:BASE:admin user ID`. A group in the AA module resides globally across the devices. A group will act just like a user in the network. The member users can appear in the identity of the group in the network. Thus for non-member users, it will be just like another user.

Just like a user, a group is identified by its *group ID* which is unique globally across the devices. The AA module will generate a group ID from the base certificate of the group. The group ID is calculated as `HASH[base public key, base group name]`. The common hash algorithm used for this purpose, across all devices, will be SHA-1.

The group facility in the AA module introduces the concept of *role*. Since a group acts similar to a user in the network, the member of a group can appear in the network with group as her identity. Thus, the role of a device user defines how the device user appears to other verified users in the network. The device user can act as one of the groups in which she is a member, or act as the user herself. Thus, the notion of role provides multiple identities to a device user.

The administrator of a group decides the verified users for the group. The information about these verified users would be provided to the members of the group, if not already present in their AA module. Thus, the AA module should be capable of distinguishing the set of verified users for a particular role of the device user.

Group Administration

For group administration, the following three approaches were identified.

1. The administrator is solely responsible for managing the members and the information related to the group. The administrator adds, deletes, or modifies the members, and other key materials associated with the group. The administrator has the entire list of members. The members of a group might not know each other.

2. The members are having the rights to manage the information related to the group. The members in the group know each other and communicate whenever a change is made.

3. There will be some privileged members who have the rights for managing the group. These members will have information about the other members in the group. The other members will be like members in the first approach.

For the AA module, we selected the first approach for the design and implementation. However, if needed, the design could be altered without much effort to adapt to the second or the third approach. For the rest of the document we will be talking about group with respect to the first approach.

The group is created and managed by the group administrator. The administrator first creates the group by providing the *name* and other details of the group. The AA module in the administrator's device creates the group and subsequently creates the base key pair and the base certificate for the group. The administrator then adds the members of the group from her list of verified users. Each member of the group is allotted one *member ID* which is unique within the group. The administrator will always be allotted the member ID as `1`. In this approach, only the group administrator knows all the members. A group member does not know the other members of the group.

The administrator of the group is responsible for the decision of the verified users for the group. She decides the access rules for the group with the group's verified users. She is also responsible for the distribution of all the group's information among its members.

Group Management Service
The *group management service* running in the devices helps for the group management. The administrator maintains the list of the members joined in the group and periodically sends the group related information to those members. This information could be shared keys or certificates or CertApplInfo (see Section 4.3.3). The information is unicast to each member in the list by encrypting with the corresponding public key of the member. The members can also request the information from the administrator. When the members receive new information, they update their AA module. These shared keys and certificates are used till they receive new shared keys and certificates from the administrator. If in case a member was not present when the shared keys and certificates were send, they could be requested from the administrator at a later point in time. Sometimes new members need to be added or existing members need to be removed from the list. Then new shared keys and certificates would be sent to all the members of the updated group.

4.3 Application Specific Information

The AA module supports symmetric as well as asymmetric cryptography. An *application certificate* in the AA module identifies the cryptographic keys used for a service access. An application certificate could be an asymmetric certificate or a symmetric key. Thus, a particular service in the AA module might use symmetric or asymmetric keys. A user or group will have several application certificates used for different services. This conforms to *REQ-12*.

4.3.1 Asymmetric Certificates

The asymmetric application certificates are identity certificates created with an asymmetric key pair. The corresponding base key of the user or group will sign these certificates.

In the case of users, the user herself generates the asymmetric application keys. The asymmetric application certificate is signed by the user's base private key. The design proposes an extension for the certificate, which provides additional information `SESSI:SELF:APP:user ID`. This helps in identifying a certificate as an application certificate of the user. Thus, other users can automatically detect the user's application certificate and associate it with the corresponding base certificate of the user. The User Interface (UI) could be set such that it prompts a user whether to accept a certificate or not before adding it automatically.

The group administrator creates the asymmetric application keys for the group. The application certificate will be signed by the group's base private key. The design proposes an extension for the certificate, which provides additional information `SESSI:SELF:APP:group ID`. This will help to associate the certificate with the base information of the group.

4.3.2 Symmetric Keys

In addition to asymmetric application certificates, a user or group might have a set of symmetric keys. A symmetric key is shared between two users or groups or between a user and a group. It could also be shared within a group between all the members. The AA module will not be responsible for the creation of symmetric keys. Any key creation or key agreement protocol could be used for generating or deciding symmetric keys. The information is stored in the AA module along with the information with whom the key is shared.

In the case of users, the user might have symmetric keys shared with another user or group. For groups, the administrator is responsible for the decision of symmetric keys for the group with another user or group. She distributes this key among the group members. A member could use this key while acting in the role of the group. A group could have another set of symmetric keys, which is used internally between the members of the group for different services. This key is created by the administrator and is distributed among the members of the group. It could be used by services for multicasting messages, which needs to be understood only by the members of the group.

4.3.3 Application Key Identifiers

Since a user has multiple application certificates, an identifier is required to specify which application key should be used, for a particular service access. This identifier, termed as *CertApplInfo*, is used to retrieve the key materials for a specific service. There could be more than one CertApplInfo for one application certificate for different services. One particular service may also have different CertApplInfos binding it with different application certificates. The CertApplInfo value could be provided by the service itself. The AA module will provide a convenience function, which creates a unique value for a particular application certificate, such that a service could use it, if required. However, the AA module does not impose any restriction or have any control over the uniqueness of the value provided for CertApplInfo. It is the responsibility of the service to provide the value of the CertApplInfo and to assure the uniqueness of the value such that it corresponds to only one application certificate. In case of collisions, the AA module gives out a list of application certificates and it is the responsibility of the service to choose the required application certificate.

4.4 Access Control Rules

The AA module provides the facility to define and manage access control rules, as per *REQ-3*. The rules provide authorization for a verified user or group to access or provide a service while the device user acting in a specified role. An access control rule in the AA module has the following fields:

Service (S): The service for which the access permission is to be given. This could be set for `ALL` services also.

Role (R): The role of the device user. The entries with role as `SELF`, will be set by the device user herself. The group administrator sets the entries with role as group. Thus, this field will have specific entries for different roles and will not have an entry `ALL`, which is set for all the roles.

Verified User (U): The user or group to whom the access permission is to be given. This could be set for `ALL` verified users and groups also.

Access Type (A): Specifies whether the permission is for accessing or providing the service.

Security Level (SL): Specifies the security level for this service access. This could also be a `default` level, if no security level needs to be specified for this access rule.

Validity (V): Specifies the time period for which the access permission is valid.

4.5 Storage of Credentials and Information

The AA module stores and manage various credentials and information, as per *REQ-5*. The following list presents the data stored in the AA module.

- The information regarding trusted Certification Authorities (CA) and the corresponding CA certificates.
- The information regarding the services which are registered by the device user.
- Base asymmetric key pair and the corresponding base certificate of the device user, and of each group in which the device user is a member. Base certificate of each verified users and groups.
- The roles for which the verified users and groups are verified.
- Sets of asymmetric key pairs, the corresponding asymmetric application certificates, and CertApplInfos of the device user, and of each group in which the device user is a member. Sets of asymmetric application certificates and CertApplInfos of each verified user and group.
- Sets of symmetric keys and CertApplInfos which could be used for internal communication with other members of each group in which the device user is a member. Sets of symmetric keys and CertApplInfos, which are shared with verified users and groups, for each role of the device user.
- Access control rules which are set for verified users and groups.

4.6 Verification of Certificates

This section explains how the base certificates and asymmetric application certificates are to be verified by the AA module.

4.6.1 Base Certificates

The AA module identifies the following kinds of base certificates:

CA-signed certificates: This is the normal base certificate of a user. The AA module checks whether the CA who issued the certificate is a trusted CA of the device user. If it is a trusted CA, then the signature is verified and the certificate is accepted. The user of the certificate becomes a verified user of the device user.

AA module extended certificates: This is the base certificate of a group. An administrator who is a verified user of the device user should sign this certificate. If then, the signature is verified and the certificate is accepted.

Self-signed certificates: The AA module can also accept self-signed certificates instead of the default CA signed certificates. This option completely eliminates the need for a trusted central authority and provides privacy of the users. The AA module accepts these certificates only after getting the permission of the device user through the User Interface (UI).

4.6.2 Asymmetric Application Certificates

The AA module recognizes the following kinds of asymmetric application certificates:

User signed certificates: This is the normal asymmetric application certificate of a user or a group. The AA module checks whether the user or group is a verified user or a role of the device user. If then, the signature is verified and the certificate is accepted.

CA signed or self-signed certificates: These are the generic application certificates which are not created by AA module. Verification of these certificates requires permission through User Interface (UI) from the device user. Since these are not AA module specific certificates, the corresponding base information would be missing in them. The signature of the certificate should be verified and a verified user should be created according to the information provided by the certificate. The application-name specified in the certificate will be taken as the base user-name. The user ID will also be calculated based on this certificate. Then, the certificate is also added as the application certificate of the base user. Here the same certificate serves as both the base certificate and the application certificate. In this case, the base user-name and the application-name would be the same.

4.7 Security Level Decision

The AA module decides which security level should be used for a particular service access, as per *REQ-4*. However, the AA module does not ensure that a service adheres to the security level decision provided. This section explains how the AA module makes a security level decision. The security levels are explained in Section 2.4.1. The three security levels defined by SESSI are None, Authentication and Confidentiality. None is treated as the lowest level and Confidentiality is treated as the highest level. The security level decision is controlled by the following two settings in the AA module.

Device Security Level: The *Device Security Level* is the minimum security level with which service access is possible with the device. This level is useful when a user switches from a non-hostile environment to a hostile environment, in which the user just selects a higher security level. By default, this level is None. For example, if the device user sets the device security level as Authentication, it means that the minimum security level with which other users could access the device for services will be Authentication.

Access Rule Security Level: The user is able to specify the security level required, for a particular service access, along with the corresponding access control rule. Thus the security level could be specified to a finer level of service, role, user, access type, and validity. The access control rule is explained in Section 4.4.

Since default (`ALL`) settings could be used for the service and verified user fields, there could be more than one access control rule which satisfies a particular service access situation. The rule which satisfies the service access situation better should be chosen. The AA module uses the following criteria for rule selection: the service field is preferred to user field, and a specific rule is preferred to a default rule. Table 4.1 shows an example where several rules satisfy.

In this example, the effective security level, from access control rules, is Authentication, since second rule is the most specific one.

	Service(S)	**Verified User(U)**	**Access Type(A)**	**Security level**
1	ALL	Alice	access	None
2	chat123	Alice	access	Authentication
3	chat123	ALL	access	Confidentiality

Table 4.1: an example where several access control rules satisfy

The AA module provides the security level decision by taking the highest level from the device security level and the effective access rule security level.

4.8 AA Module Architecture

The AA module architecture proposed by this design is shown below. The AA module should be a part of every device in the network.

4.8.1 Architecture

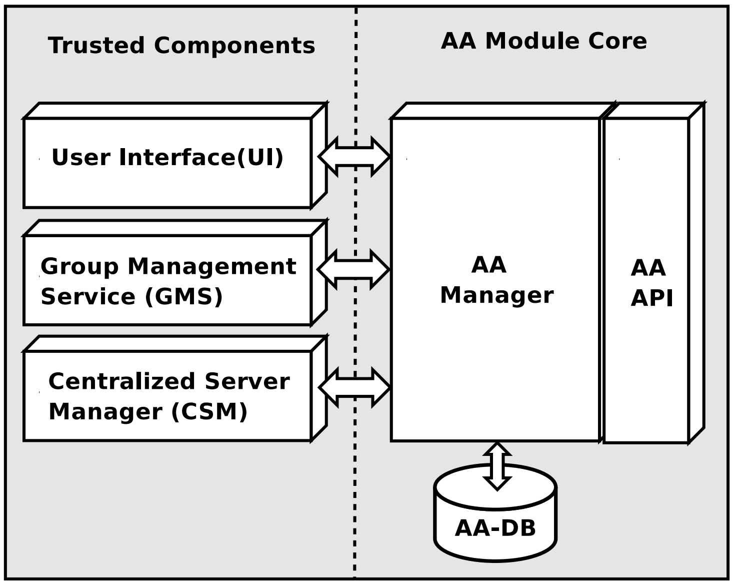

Figure 4.2: AA module architecture

Figure 4.2 presents the internal architecture of the AA module. It consists mainly of two parts, the *core AA module* and the *trusted components*.

Core AA Module

The core part of the AA module provides the basic functionality of the AA module as proposed by this design. The following are the components of the Core AA module:

AA-API
AA-API is the AA module API which provides the interface to the AA module, as stated by *REQ-15*. Services can use this API to interact with the AA module.

AA Manager
The AA Manager is the main component in the AA module. It handles all the core functionalities of the AA module. It is also responsible for managing the information in the AA-DB.

AA-DB
The AA-DB stores all the information required for the AA module, as per *REQ-5*. It keeps all the data and credentials required for users, services, trusted CAs, and groups. The list of information stored by the AA module is given in Section 4.5.

Trusted Components

The trusted components of the AA module introduce several features for the management of the module, thus extending the AA module functionality. The following are the trusted components of the AA module.

User Interface (UI)
The *user interface* could be used by the device user for local management of information and to interact with the AA module. It is used for minor modifications to manage the information stored. The user interface should be able to handle different user inputs. It should have provision for different user interface functionalities which calls the corresponding APIs in the AA module like adding users or groups, creating groups, selecting various roles, setting the device security level etc. Functionalities should be provided for the device user interaction while adding users having self-signed base certificates or users with non-AA module specific application certificates. The user interface should also have interface functionalities for handling the group management

service and contacting the centralized server manager.

Centralized Server Manager
The *centralized server manager* could be used to input or synchronize data in the AA module with a centralized server. A centralized server could be a huge database keeping credentials of various users. This could be a preferred method of input if the data to be handled is very large. For example, consider a situation where a new employee joins an organization. The administrative group of the organization will be handling various groups for different projects, a directory with public keys and certificates of various employees and various other data. Thus there will be a large amount of data to be inserted. In this situation, it is better that the employee just contacts a centralized server and downloads the information thus updating his device. The same applies when synchronizing the data for a group among various employees in a project. Thus updating the AA module through a centralized server is a preferred way when the data to be inserted is large enough such that manual input of data by the device user is a very tedious process. The centralized server manager is responsible for handling the data, which it gets from the centralized server.

Group Management Service
The *group management service* helps a device user to manage the groups across several devices. The following components were identified for the group management service.

- **Group membership management**: This part should handle own groups in which the device user is a member. It should be able to make requests to a group's administrator for the corresponding certificates and keys. It should periodically check the validity period of various credentials of each group. It is responsible for receiving various certificates and keys for different groups and updating the credentials in the AA module.

- **Group administration**: This part should maintain the groups that the device user administers. It should be able to handle requests from different group members for the corresponding certificates and keys. It is responsible for renewing keys and certificates periodically according to the requirements set for each group by the device user. If one member is added to an existing group, this process is responsible for sending the corresponding certificates and keys to this new member user. Further, if one member is revoked, then this process is responsible for creating a fresh set of certificates and keys and transferring them to the remaining member users.

4.8.2 Interaction with Other Modules

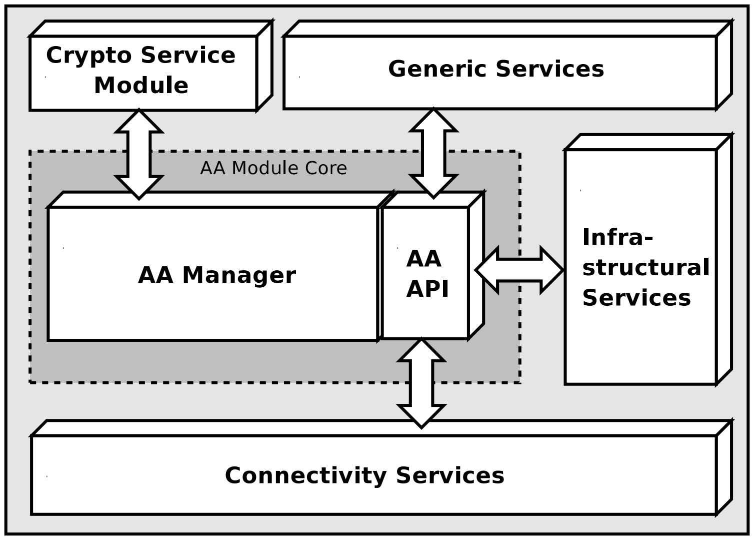

Figure 4.3: AA module interaction with other modules

Figure 4.3 describes the relation of the AA module with other modules. For simplicity, only AA Manager and AA-API are shown as part of the AA module. The AA Manager calls the Crypto Service module APIs for cryptographic purposes. The various services, such as generic services, infrastructural services and connectivity services, talk to AA-API for communicating with the AA module.

4.8.3 Internal Security

The AA module introduces several security features to protect the credentials stored, adhering to *REQ-11*. The devices are treated as single-user devices, rather than multiple-user devices. Thus, we will not be addressing the security issues related to multiple-user devices. The following are the various internal security features of the AA module.

- **Sensitive information never goes out**: The AA module does not give out sensitive information, such as private keys or shared secrets, to services. Apart from the trusted components of the AA module, this sensitive information is given only to the Crypto Service module for various kinds of cryptographic purposes.

- **Access through API**: The services access the AA module only through the AA-API provided. Only the trusted components and the Crypto Service module interact with the AA Manager.
- **Connection levels**: There will be different levels of connection to the AA module. Even though the services, which run inside a device, are supposed to be trusted, there should be a mechanism for controlling the extent to which the API functions are called. For interacting with the AA module, a service has to make the connection to the AA module requesting for a specific connection level. The level of connection determines the level of the API functions that could be accessed by the service. The lowest level does not allow for any kind of modification of the information in the AA module.
- **Protection of sensitive data**: The sensitive data in the AA module are encrypted and stored in the AA-DB so that no other process can read those data directly from disk. The encryption should be done by a device user provided password, and it should not be stored anywhere in the device's storage space. The password from the device user should be asked through the user interface when the AA module is accessed for the first time, after the device boot up.

Chapter 5

Implementation

This chapter explains the prototype implementation of the Authentication and Authorization (AA) module for ad hoc networks. The implementation is based on the design, which was presented in the previous chapter. Section 5.1 gives an overview of the implementation. It presents the various components of the implementation and their corresponding functionalities. The next section explains the external libraries used to support the implementation. Section 5.3 presents the various connection levels and the design of APIs. The next section explains how internal security is implemented in the AA module. A brief note about the testing and demonstration performed for the prototype implementation is presented in the last section.

5.1 Implementation Overview

The prototype implementation of the AA module, according to the design specified in the previous chapter, was developed on Linux platform and was written completely in C. The implementation focused on the core part of the AA module. The development of the trusted components will be left for future work. The entire source code consisted of around 15,000 lines of code. The AA-API is provided as a C interface, which other services will use to interact with the AA module, as required by *REQ-15*. The implementation used X.509v3 certificates for base certificates and asymmetric application certificates. Certificate extensions are used for binding an AA module specific certificate to its corresponding user's base certificate. The AA module used the logging functionality provided by the SESSI framework, to create a log of activities. Test code was written to check the various interface functions under different scenarios. A demonstration code was also developed, which creates the demonstration environment of a group of users using the AA module.

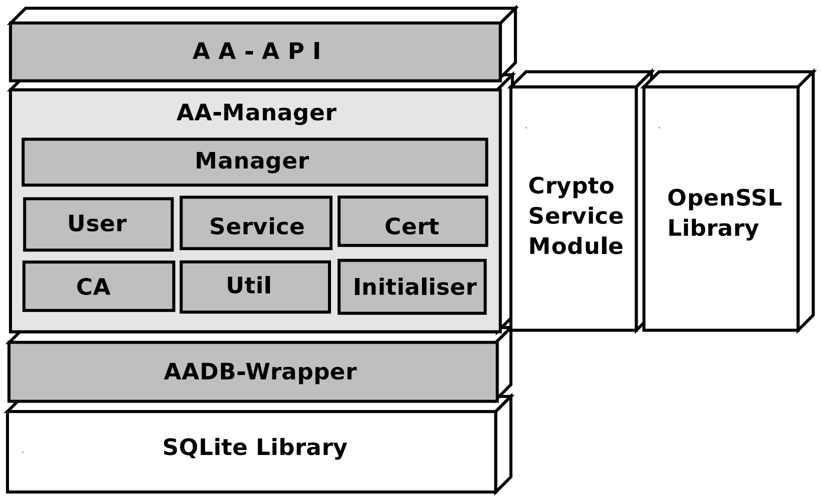

Figure 5.1: Overview of the functional parts of AA module

The Figure 5.1 shows the implementation details of the AA module. It presents the various functional parts of the AA module implementation (shown in gray boxes). It also shows the modules and libraries used to support the implementation (shown in white boxes). The AA module implementation could be divided mainly into three parts; the *AA-API*, the *AA-Manager* and the *AADB-Wrapper*. How the different functionalities are divided among the parts is described below.

AA-API: This contains the C interface functions of the AA module which would be used by other services to interact with the AA module. This part interacts with the AA-Manager part for providing the various functionalities to the services.

AA-Manager: This is the main part of the AA module. As shown in the figure, the AA-Manager is a combination of seven parts, which interact with each other. All these parts interact with the AADB-Wrapper and the *Crypto Service* module for storage and cryptographic functions respectively. The different parts of the AA-Manager are:

1. **Manager**: provides the central functionalities of the AA module like setting the base security level, storing the access control rules, getting

authorization and security level decision, getting the AA connection, providing internal security etc.

2. **User**: provides the user and group specific functionalities like creating groups, managing verified user and group data, retrieving verified user specific details, cryptographic functionalities like sign, verify, encrypt, and decrypt with base certificates etc.

3. **Service**: provides service specific functionalities like getting authorized services, register and unregister services, getting service details etc.

4. **Cert**: provides application-specific functionalities like creating and storing symmetric and asymmetric application certificates and keys, cryptographic functionalities like sign, verify, encrypt, and decrypt with application certificates, creating default CertApplInfo etc.

5. **CA**: provides functionalities specific to CAs like storing trusted CA data, retrieving CA details, getting the list of trusted CAs etc.

6. **Util**: provides miscellaneous utility functions used internally such as validating the various inputs, parsing the certificate extension field, returning the error string etc.

7. **Initializer**: provides functionalities used by the trusted components to initialise the AA module structure.

AADB-Wrapper: provides general purpose wrapper functions to the *SQLite library.* The AA-Manager parts use these internal functions, thus limiting the dependency of the AA module on the SQLite database library. This part interacts with the SQLite C API to provide storage functionality.

5.2 Supporting Libraries

The AA module relies on external libraries for cryptographic and storage functionalities. These libraries are:

5.2.1 Crypto Service Module

The AA module depends on the *SESSI Crypto Service* module for cryptographic services. The Crypto Service module was developed as part of the SESSI framework and is a wrapper for the *OpenSSL* cryptographic library[23].

The Crypto Service module provides a convenience interface to common cryptographic functions. All the other modules in the SESSI framework use Crypto Service module, thus limiting the dependency of the SESSI framework on the OpenSSL cryptographic library.

5.2.2 SQLite

SQLite is a small C library that implements a fast, small and reliable SQL database engine. It has a very small code footprint with less than 30K lines of C code, which is in the public domain. It is self-contained with no external dependencies and is a lot faster when compared to other popular database engines, for most common operations. The complete database is stored in a single disk file. Contrary to other popular database systems, like MySql, SQLite doesn't require a daemon. It communicates directly with the file system or memory where the data is stored. All these features made SQLite a clear choice for implementing AA-DB.

A wrapper for the SQLite library, known as AADB-Wrapper, was created as part of the AA module, as stated by *REQ-18*. The AA module uses this wrapper to communicate with AA-DB, thus limiting the dependency on the SQLite API.

5.3 Connection Levels and Design of APIs

Connection levels, introduced in Section 4.8.3, differentiates the connection made by different services to the AA module. A service, which needs to use the AA module, should first request for a connection with specific connection level, from the AA module. The connection is granted if the service, which requested the connection, has the authority to act in the specified permission.

It was identified that services, which interact with AA module, could be classified into four categories, and this lead to the design of four levels of connections. The same division was followed for the design of AA module APIs. The interface functions were designed such that it goes along with the connection levels. The functions could be used by a service according to the level of connection granted. The AA module does not give out sensitive information to services. The API provides the required functionalities using the sensitive information, as stated by *REQ-16*. Further, the services need not care about the type of algorithm or key used, as per *REQ-17*. The four levels of connection are explained below.

5.3.1 Level1 - Access

This type of connection is just for requesting the information and access decisions from the AA module. This is the most restricted kind of connection and is not permitted to make any changes to the AA module. At this level a service can:

1. Verify base certificates and asymmetric application certificates.
2. Get base security level and security level decision.
3. Get authorization decision.
4. Get the list of CertApplInfos for an application certificate, or the list of application certificates for a CertApplInfo.
5. Get the list of authorized services, or all services.
6. Get the list of authorized users, or all verified users.
7. Get own roles and default role.
8. Get service details.
9. Get base certificate, and application certificate details.
10. Get own member ID of groups in which device user is a member, or member ID of different members in which device user is an administrator.
11. Get trusted CA details.
12. Create a default CertApplInfo for an application certificate.
13. Get the owner of an application certificate.
14. Encrypt and decrypt with base certificates and application certificates.
15. Sign and verify with base certificates and asymmetric application certificates.

5.3.2 Level2 - Service

This connection is made by services. This connection can request information and access decisions just like Level1, and it can make some changes, which are allowed to be done by services. At this level, in addition to Level1 functionalities, a service can:

1. Register and unregister a service.
2. Change the service access type.

5.3.3 Level3 - Change

This connection is permitted for making changes to the AA module. This level is used by those services, which are defined to be trusted by the device user or by the trusted components of the AA module. At this level, in addition to Level2 functionalities, a service can:

1. Store verified base data, and asymmetric application data.
2. Create and store own asymmetric application certificate.
3. Store trusted CA data.
4. Create a group and add members for a group.
5. Set the default role.
6. Store own group base data.
7. Make a user verified for a role.
8. Store the CertApplInfo for an application certificate.
9. Store symmetric application data defined with a verified user or within own group.
10. Set the base security level.
11. Set the access control rules.

5.3.4 Level4 - Admin

This connection is permitted with full powers to change the AA module. This level is used only by the trusted components of the AA module. At this level, in addition to all what could be done by other type of connections, a service can:

1. Initialize the AA module.
2. Store own base data.
3. Write out base data and application data for verified users and own group members.

5.4 Protection of Sensitive Data

The AA module encrypts and stores sensitive data, such as private keys and shared secrets, in the AA-DB so that no other process can read those data directly from disk, as stated by *REQ-11*. The encryption is done with the help of an *internal-secret key (ISkey)*, which is created by the AA module during the initial phases of its configuration. This symmetric key does not go out of the AA module, and is totally transparent to other modules and services, which use the AA module. This internal-secret key is encrypted and stored with the help of a device user supplied password, which is termed as *passkey*. The passkey is not stored anywhere in the AA module's storage space. The idea behind using two symmetric keys is that the user provided passkey might change frequently. If all the sensitive information is encrypted with the passkey, then changing the passkey would result in re-encrypting all the sensitive information with the new passkey. This might not be convenient if the data to be re-encrypted is large. Further, a user-supplied passkey would be only a few (4-6) bytes long. Encryption of a number of different messages with a smaller length symmetric key leads to more attacks. Thus, the AA module creates an ISkey for the encryption of all the sensitive information and just the ISkey is encrypted and stored using the user supplied passkey.

5.4.1 Initial Phase

The AA module creates the internal-secret key (ISkey) during the initial stages of the AA module configuration, while the device user stores own

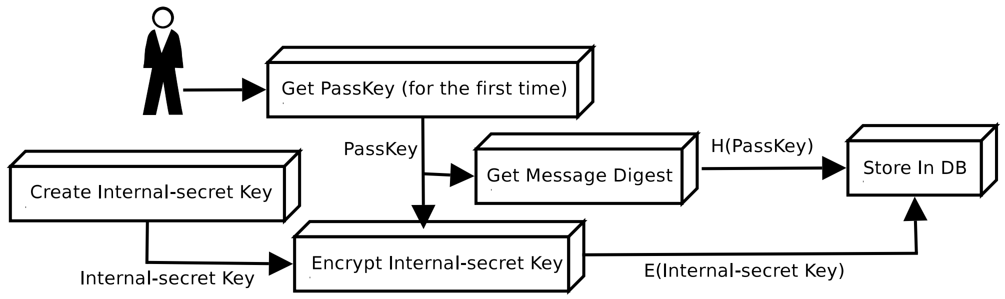

Figure 5.2: Configuring AA module with internal-secret key

base data. The device user's passkey is asked through the User Interface (UI). Then the AA module creates the ISkey, which is a symmetric key of default type and length (default is AES with 128 bit length). This ISkey is then encrypted and stored using the passkey. The hash of the passkey is calculated using the default message digest algorithm (default is SHA1) and stored in the AA-DB. Thus the AA module does not store the passkey of the user in the AA-DB. Instead it stores the hash of the passkey, which makes it possible at a later point of time to compare whether the passkey supplied by the user is the same. Figure 5.2 shows the process explained above. This process happens very rarely.

5.4.2 Retrieval of ISkey

The internal-secret key (ISkey) should be retrieved to encrypt or decrypt sensitive information. After the device boot up, the passkey from the user is asked through the UI, when the AA module is accessed for the first time. The message digest of the passkey is calculated and compared with the hash value stored in the AA-DB. If it matches, then the passkey is used to decrypt the ISkey, which is encrypted and stored in the AA-DB. This ISkey is kept in the memory of the AA module and could be used to encrypt or decrypt sensitive information. Figure 5.3 shows the steps explained in this process. This process is repeated every time the device boots up.

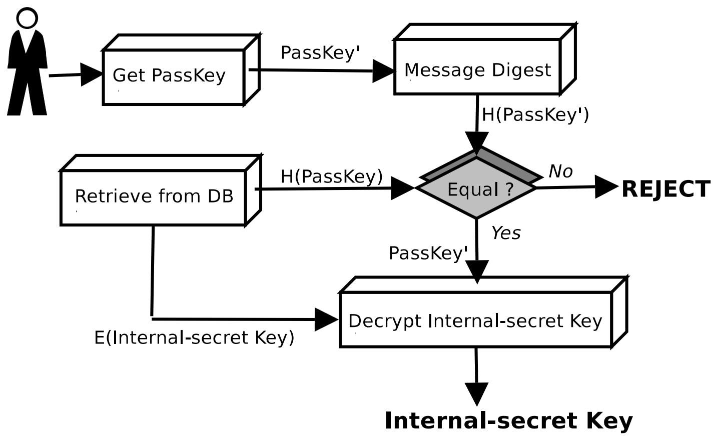

Figure 5.3: Retrieval of internal-secret key

5.4.3 Encryption and Decryption of Sensitive Data

Whenever there is a need to encrypt or decrypt sensitive information while data is stored or retrieved from the AA-DB, the corresponding internal function is called, which requests the ISkey and encrypts or decrypts the sensitive information. Figure 5.4 shows the process of the encryption function aa_encryptAAData() and the decryption function aa_decryptAAdata().

5.5 Prototype Testing and Demonstration

The AA module functionality for all the four levels of connection were tested using stand alone test functions. A demonstration code was developed, to test the module. It created the environment of a set of six users using the AA module. The demonstration code constitutes of a setup code and a demonstration application. The demonstration setup code created the AA module for the set of users. It involved testing various functionalities of higher connection levels such as initializing the AA module, creating own base information, adding trusted CAs, creating application specific information, adding verified users, adding base and application specific information of verified users, creating groups, adding group members, and setting access control rules. The demonstration application uses a level-1 connection for demonstrating various level-1 functionalities.

The AA module was developed in conformance with the SESSI framework. A

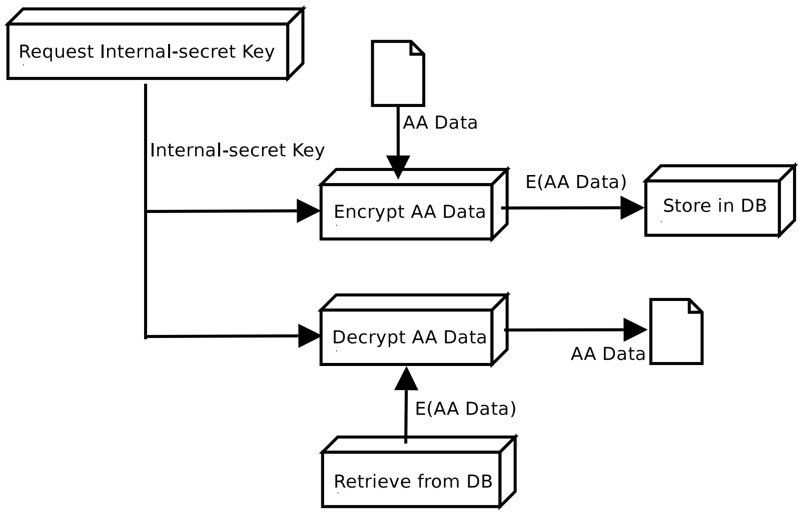

Figure 5.4: Encryption and decryption of sensitive data

subset of the functionality was successfully tested and used by the SD module implementation. The entire functionality would be used only when the trusted components will be developed in the future. The AA module was successfully demonstrated to the project funders. In the demonstration, the keys and certificates used for the SD module were pre-distributed among trusted users. When a user wanted to look for a service, it queried the AA module providing the current role, and asking to which all users the advertisement should be sent if she is going to access the service. While communicating with specific users, the SD module queried for the security level and for the functionalities required to protect and verify messages of the communication. The messages created by the SD module for communication were passed to the AA module for signing. The AA module could successfully resolve the mapping of CertApplInfo to the base user, thus providing authorization and security level decisions.

Chapter 6

Evaluation and Discussion

The Authentication and Authorization (AA) module was designed and successfully implemented as explained in Chapters 4 and 5. The objective of this chapter is to evaluate the architecture and functionalities of the AA module based on the requirements identified in Chapter 3. Even though the prototype implementation is discussed, the evaluation concentrates mainly on the proposed architecture and the design of the AA module.

6.1 Functional Features

The AA module provides various functional features as per the requirements presented in Section 3.2.1. Even though the primary goal of the AA module is to provide authentication and authorization, it also functions as a security architecture providing other security functionalities such as message confidentiality and security level decision. For obtaining most of the functional features, it is required to retrieve the appropriate application certificate and identify the user. In the AA module, a CertApplInfo identifies the application certificate required for the particular service access. The application certificate is in turn bound to the base certificate of the user. The base certificate identifies the user in the AA module, which helps for authorization and security level decisions. Thus, with the help of the CertApplInfo, it is possible to identify the required application certificate and the user. The AA module provides several convenient methods to retrieve the appropriate application certificate and identify the user. The methods implemented in the AA module are explained in Appendix B. We will go through the functional features offered by the AA module and see how they fulfill the requirements.

Authentication

The AA module provides various cryptographic functions along with the required credentials for performing authentication of users, as stated in *REQ-1*. The services, which use the AA module, can use their own authentication protocol for obtaining authentication functionality. The AA module does not implement any authentication protocol. For message authentication and integrity (*REQ-2*), the AA module provides signing and verification services with the required asymmetric keys retrieved from storage. For obtaining authentication, the base key pair or the service specific application certificate of the user could be used.

Authorization and Access Control

According to *REQ-3*, the AA module provides various methods for making authorization decision. The authorization decision is based on the specified access control rules. The AA module allows the device user to define fine-grained access control rules. Default settings are allowed in the rules to provide flexibility according to the security requirements and convenience of the device user (see Section 4.4). The AA module defines two-way access control such that a device user is able to define separate access control rules for both accessing and providing a service. Thus, the device user can specify the services, which would be provided by her and the verified users who can access them. Similarly, the device user can also specify the services, which she needs to access and the verified users who can provide them.

Security Level Decision

As per *REQ-4*, the AA module provides the security level decision for a particular service access. The security level could be specified along with a particular access control rule. Due to the default values in the rules, for service and user fields, there could be more than one access control rule, which satisfies a particular service access situation. Further, the AA module introduces the device security level which guarantees the minimum security level with which the service usage is possible with the device. This feature, which helps the device user to switch between security levels according to the hostility of her surroundings, is very useful in ad hoc networks. The AA module provides the security level decision based on the above two settings (see Section 4.7).

Storage of Credentials and Information

As stated by *REQ-5*, the AA module stores various credentials which helps secure usage of different services (see Section 4.5). It stores the device user's base and application keys and certificates, and other required information. It also stores the various certificates and credentials of verified users. The hierarchical binding of credentials helps in determining to which user it belongs. Services can provide identifiers (see Section 4.3.3) binding them with credentials and thus helping in retrieving the required credentials at a later stage.

Message Confidentiality

The AA module helps to provide confidentiality of messages by providing encryption and decryption methods with the required credentials. The base key pair or the service specific application certificates of the user could be used for providing confidentiality. The services do not need to care about the algorithms used to provide encryption.

6.2 Design Features

The architecture does not rely on infrastructures while providing various functionalities during service access in ad hoc networks. The solution depends on external authorities only while getting own base certificate or while getting the list of trusted CA certificates. A user contacts a CA for obtaining her base certificate signed. This could be done before initializing the AA module. The user keeps a list of trusted CA certificates for verifying the base certificates of other users. The AA module also supports the usage of self-signed certificates, which completely eliminates the dependency on CAs. Thus, with the list of trusted CA certificates, or by manually verifying self-signed certificates, the AA module allows the device user for adding verified users at any time in the ad hoc network, without depending on external servers. With the base certificates installed, the AA module supports creating any number of application certificates, or verifying and adding any number of verified user's application certificates. We will go through the design features offered by the AA module and see how they fulfill the requirements.

De-centralized Design

As stated by *REQ-6*, the AA module follows a de-centralized design. It is part of every device in the ad hoc network. The design does not rely on co-operative algorithms, which involves decisions from other devices in the network. The AA module in each device is capable of making its own decision without relying on peer devices. Further, the design does not introduce any changes in the underlying protocols and security mechanisms already used by services. However, the services need to be modified to use the module API for various functionalities and credential storage.

User and Service Identification

The AA module introduces methods for providing globally unique identifiers for users and services, as stated by *REQ-7* and *REQ-8*. For users, it relies on the hash value generated from the base public key of the user (see Section 4.2.1). For services, it uses a naming scheme which is similar to the Java package naming scheme (see Section 4.2.2). These methods could be replaced later with any other method, which creates globally unique identifiers for users and services. Even though in the current design, a service need not authenticate itself, it might not be the same if we start to consider that the services inside own device are not trustworthy enough. In that situation, it could be that in the future, services might also need to be authenticated. However, this will not disturb the current design of the AA module, since connection levels already introduce the required structure.

Support for Groups

As per *REQ-9*, the AA module supports groups. A group behaves just like a user in the network. The members of the group act with same identity (role) of the group using the same authentication credentials. The AA module supports application certificates which could be used for internal communication of the group. These credentials could be used for multicast based group communication, where the receiver is a single group of users rather than a list of separate users. Roles help a device user to appear in multiple identities in the network. The user can appear as user herself in the network or can appear as one of the groups in which she is a member.

The group mechanism introduced by the AA module also allows for the delegation of access rights of a service. When a device user gives access permissions to a group, she does not know who all are the members of the group. The device user just knows the administrator of the group and the group's credentials. The administrator manages the membership of the group. Thus, a device user giving access permissions to a group is delegating the access right, for that particular service, to the administrator of the group.

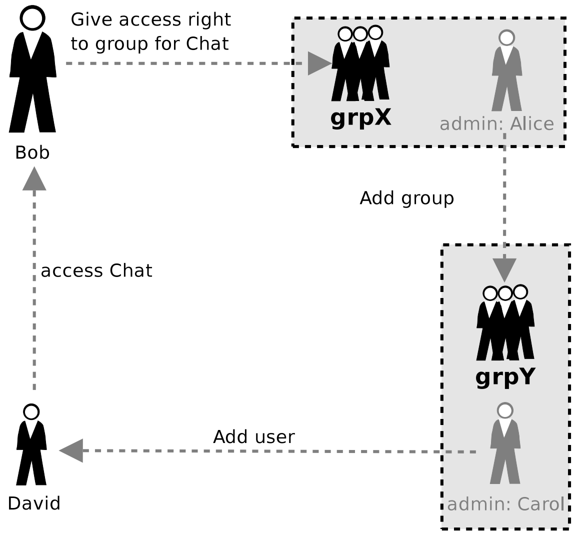

Figure 6.1: Delegation of access rights of a service

An example of delegation is shown in Figure 6.1. Here the device user Bob gives access permission to group *grpX* for accessing the service *Chat*. The administrator of the group is Alice. Thus, Bob delegates the access right of the service *Chat* to Alice, by adding *grpX* as a verified user. Now, Alice can add any number of users to *grpX*, thus giving them access to the service *Chat*. Alice delegates the right to Carol by adding the group *grpY*, for which Carol is the administrator. Now, Carol adds David thereby giving him the right to access the service *Chat*. However, the access right is not delegated to David. In this situation, David is completely unknown to Bob. Yet, David can access the service *Chat*, in the role *grpX*, from Bob's device.

Thus we can see that the group gives the facility for the delegation of access rights. The revocation of these rights is also very easy since the AA module uses local access control rules. If Bob just removes the access permission provided to *grpX*, then the whole branch of users who obtained access rights through Alice is revoked from accessing the service *Chat* from Bob's device.

Protection of Privacy

Using a certificate signed by a CA might be a source for privacy problems. There might be situations where users would like to protect their privacy while accessing some services. The AA module satisfies *REQ-10* by providing mechanisms to protect the privacy of the users in the network with the help of groups and self-signed certificates. Using these mechanisms, users can access the services without revealing their real identity.

Group Mechanisms: The administrator of a group decides who all are the members of that group. The administrator decides on the verified users also. The verified user just knows the information regarding the administrator and the group. The verified user does not know the identity of the members. Members access or provide a service on behalf of the group. The member will be authenticated and authorized as the group. Only the administrator, if required, can reveal the real identity of the member.

Self-signed Certificates: The AA module's default base certificates are CA signed certificates. However, if permitted by the device user, the AA module could also allow self-signed certificates from a user. Thus it is easy for a user to create as many self-signed certificates as needed and use them as different identities in the network. For example, a user in the network will have one permanent certificate which is signed by a CA and will have several self-signed certificates, which are used only for temporary purposes and are periodically replaced with new certificates. Yet it will be up to a device user to accept only those users who have CA certified base certificates as verified users or to also accept users with self-signed base certificates.

Internal Security

As required by *REQ-11*, the AA module introduces several security features for protecting the credentials stored in AA-DB. The sensitive information in the AA module is protected using an internal key and the device user supplied passkey, as explained in Section 5.4. This protects sensitive information from being retrieved from the storage by other processes in the device or by other users if the device gets into their hands. Further, the services access the AA module only through the AA-API provided. The AA module does not reveal sensitive information to services. This information is passed only to the trusted components and to the SESSI Crypto Service module. The AA module also introduces connection levels to control the activities of services in one's own device. It helps in differentiating the connection made by different services with the AA module and limits the extent to which the services can act.

Support for Multiple Cryptographic Algorithms and Keys

As required by *REQ-12*, the AA module supports multiple credentials and keys for a user. It does not restrict the services to use a particular algorithm. A user can have any number of application certificates, and they could be created with the help of the AA module. The AA module supports asymmetric and symmetric cryptography. Application certificate identifiers (CertApplInfo) are used to specify which application certificate is required, for a particular service access.

The keys form a hierarchical structure in the AA module as presented in Figure 6.2. Each user has a base key pair, and a base certificate which is signed by a Certification Authority (CA) or by the user herself. The base key pair is used to sign the user's asymmetric application certificates. When a user creates a group, the group's base certificate is signed with the user's base key. The base key of the group is used to sign the asymmetric application certificates of the group. Thus the keys form a hierarchy where the base key of the user forms the root of the hierarchy and the leaf nodes are the keys used for different services.

The hierarchical key structure helps to minimize the dependability on centralized authorities like CA. Only the base certificate of the user is required to be signed by a CA. The AA module creates all the application certificates and the group base certificates, which hold the extension identifying the parent key in the hierarchy. This binding helps in identifying the various credentials of a user. The hierarchical key structure also helps in creating any number of certificates for services without CA interaction. Further, the services can use any type of user names, for instance strings with a specific syntax, hashes or public keys. There is no restriction that the user names should be in the X.500 format. The hierarchical key structure seems to be an efficient means for user authentication and authorization, since the root key in the hierarchy identifies the user. This helps in access control, since the permissions can be given to the user rather than giving to various service specific names.

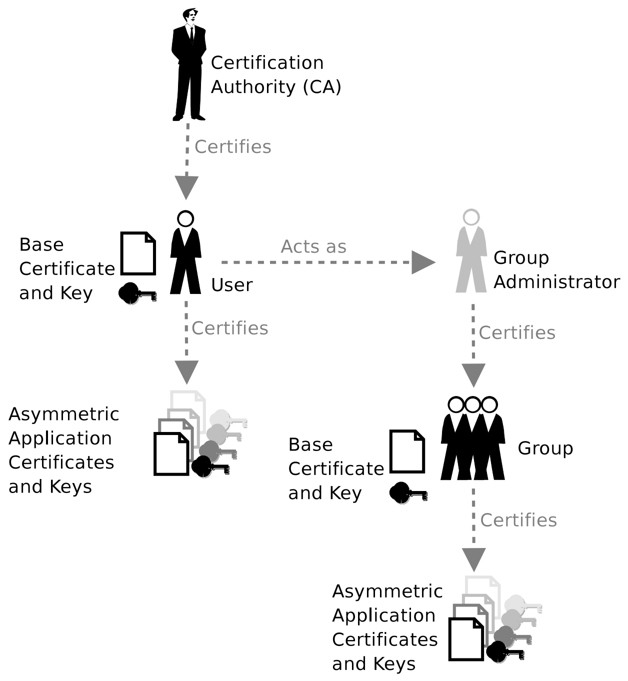

Figure 6.2: Hierarchical key structure

Scalability of Design

The design of the AA module does not impose any restrictions on the maximum or minimum number of users in the network for its proper functioning. It does not depend on the availability of other users in the network. The number of users it can handle just depends upon the capacity of the device used. Thus the AA module satisfies *REQ-13* since it is scalable enough to handle any number of users in the network.

Extensibility of Design

The design of the AA module is quite extensible, as per *REQ-14*, and can adapt to future requirements. The AA module works with globally unique IDs for differentiating users and services. The user IDs are created with the help of cryptographic functions. The structure of the AA module supports any kind of unique IDs for users and services. Thus, the system could be changed to work along with other technologies which work with unique IDs, for example HIP. The AA module provides functionalities and credentials required to easily implement any user authentication or key agreement protocol. The dependencies on the cryptographic and database libraries are minimized and they could be replaced easily with other libraries of similar functionality. The AA module uses local access control rules. The current access control rules could be replaced with more fine-grained rules, if required. For example, a device user Alice could implement a more fine-grained access control rule saying that Bob can access her chat service only on Mondays between 7:00 and 15:00. Further, different users could have different kinds of access control rules implemented in their devices to suit their needs.

6.3 Implementation

The implementation of the AA module is written completely in C language and the module functionalities are provided for use by other services through C API, as stated by *REQ-15*. The AA module does not give out sensitive information, but at the same time provides cryptographic functionalities using those sensitive information, as required by *REQ-16*. Further, the AA-API provides transparency of cryptographic algorithms used. As per *REQ-17*, the services need not care about the type of algorithms used or the keys used to provide the functionalities. The implementation supports various types of symmetric and asymmetric algorithms as provided by the cryptographic library. Even though the implementation uses the OpenSSL and SQLite libraries, it uses wrappers which were built on top of these libraries, which makes it possible to switch to other similar libraries easily (*REQ-18*). Thus, any cryptographic and database libraries could be used to provide the necessary functionalities. The implementation used X.509v3 certificates for the base and asymmetric application certificates. It used certificate extensions for binding AA module specific certificates to the corresponding user's base certificate. The implementation was successful and proved the feasibility of the architecture.

Chapter 7

Conclusions

The main goal of this thesis was to provide an authentication and authorization architecture for service access in ad hoc networks. This thesis involved requirement analysis, design, and implementation of a de-centralized architecture, which will be a part of every device in the network.

The design focused on the approach using identity certificates and local access control rules for user authentication and authorization. This design approach was combined with certificate extensions, user identification with public keys, and hierarchical key structure to achieve several desirable features. In this approach, users are authenticated rather than devices.

The hierarchical key structure introduced by this thesis seems to be an efficient means for user authentication and authorization. A user in the network will have one base key and multiple application keys. The base key helps to identify the user, and the application keys are used by different services. The base keys or the application keys could be used for authentication. The application keys, which are signed by the base key, are bound to the user with the help of certificate extensions. Thus, in the hierarchical structure, the root key in the hierarchy identifies the user and the leaf keys are used by various services. This helps in access control, since the permissions can be given to the user rather than giving to various service specific names. The design introduces the application key identifiers, which helps to identify the application keys required for a particular service access. Since the application keys are bound to the user, the user could be identified, and the authorization and security level decision regarding the user could be obtained.

The group mechanism introduced by this thesis provides various facilities. It allows for multiple identities and provides privacy of users. It also allows for delegation of access rights. The architecture minimizes the dependency on centralized authorities, such as CAs, with the help of hierarchical key structure and self-signed certificates. CAs are used only to certify the base certificate of a user, and the trusted CA certificates are used to verify the base certificates of other users. The application certificates are created and signed by the user herself. The architecture is also able to avoid CAs completely with the help of self-signed certificates.

Services can use the architecture without changes to the underlying protocols and security mechanisms. However, the services need to be modified to use the architecture for various functionalities and credential storage.

There are many possibilities for future work. The current implementation focused on the core part of the architecture. The trusted components, such as group management service, centralized server manager, and user interface should be developed to take advantage of the architecture's full functionality. The required basic APIs for these components are provided as part of the core implementation. Even though most of the functionalities are provided, there might be requirement for new methods according to the design of the trusted components.

The current implementation provides only a base structure for the connection levels. The authenticity of a service needs to be checked before granting connection. Each function checks whether the service, which calls it, has the proper valid connection granted to access that function. This checking is also implemented at a very base level and could be changed later with a more powerful checking mechanism.

To conclude, this thesis proposed a de-centralized security architecture, which provides authentication and authorization for service access in ad hoc networks. The architecture successfully satisfied the requirements and met the goals of this thesis. The proposed architecture was proved to be feasible with implementation and demonstration was done with other services using the architecture capability.

Appendix A

An Example Scenario

This chapter presents an example scenario, which shows a step-by-step process where one user prepares his AA module by creating and gathering information. Let us consider an example where Bob wants to create his own credentials and gather credentials from other users. Bob also wants to create a group of his verified users.

Creating Own Credentials

First Bob has to create or gather his own information for communicating with other users in the network. Bob needs his base certificate, application certificates and list of trusted CAs. He does the following steps:

Create base information: Bob creates his base asymmetric key pair. He also gets his base certificate, which is signed by a CA. The certificate and the key information are stored in his AA module. Bob sets his base security level through the AA module UI. Now Bob has his base information in his device.

Add trusted CAs: A user needs a list of trusted CAs to verify other users. The list of trusted CAs could be provided by Bob's organization or Bob himself can choose the list of CAs whom he can trust. Alternatively the list could be downloaded from a centralized server also. The list contains the information regarding various CAs along with their certificates. This information is updated into Bob's AA module.

Add application-specific information: Now Bob wants to install some applications. He downloads the application into his device and installs it.

The application registers with the AA module with parameters such as service name, service ID, access type etc. The application when run has an option for the creation of application certificates. Bob takes that option and the application prompts Bob for the role and the application-specific user-name. Bob enters the user-name and chooses the role as `SELF`. The application makes a request to the AA module for the creation of an application certificate for the role as `SELF`. The parameters provided are user-name, application-name etc. The AA module creates the asymmetric key pairs, creates the certificate according to the parameters and signs the certificate with the base private key of Bob. The AA module installs the certificate and gives back a handle, which identifies the certificate. The application can request to get the certificate with that handle. The application should give the CertApplInfo (along with service ID and the handle) back to the AA module. The AA module adds the information of the CertApplInfo and binds it with the certificate. The application can have its own function to create a CertApplInfo or it can use the default function, which is provided by the AA module.

Adding Verified Users

At this point, Bob has his own base certificates and application certificates. Now, Bob wants to add Alice as a verified user. Bob does the following steps:

Add base information: Bob gets Alice's base certificate. He gets it from some trusted source like a centralized server or directly from Alice through a trusted medium. He verifies the certificate with his trusted list of CAs. If the certificate could be verified successfully, then the base certificate is accepted. If the base certificate could not be verified successfully, the UI prompts Bob to decide whether to accept the base certificate or not. Once accepted, the certificate and the base information of Alice are added to Bob's AA module. Now Alice is a verified user of Bob.

Add application-specific information: Alice sends her application certificates to Bob or Bob gets it from a centralized server. Bob verifies the certificates with Alice's base public key. The application certificates are added along with the CertApplInfo and the service ID.

Thus at this point, Bob has Alice as a verified user and has Alice's application-specific information. In return, Bob sends Alice his base certificate and application-specific information. Thus Bob becomes a verified user of Alice also. Bob adds Carol, Dave and Ted in the similar way.

Adding Groups

Now Bob wants to create a group with Alice, Carol and Dave. Since, Bob creates the group, he becomes the administrator of the group. He does the following steps:

Create base information: In the UI of his device, Bob selects the group creation option. He enters the group name. The AA module creates the public key pair, and creates and signs a certificate binding the group name and the public key. The information is added into the AA module of Bob with admin ID = `SELF` and member ID = `1`.

Create application-specific information: Now Bob wants to create some application certificates for the group. The application is run and is made to request the AA module for the creation of the application-specific certificate for the role as the group. The parameters provided are username, application-name etc. The rest is the same as creation of own application-specific certificates. Now, Bob has created the group, has become the administrator of the group, and created the application certificates.

Adding members: Bob wants to add members into the group. From the list of his verified users Bob selects Alice, Carol and Dave. Bob can optionally set the time till each person is a member. The member ID is allotted to each member automatically. The AA module calls the corresponding function, for updating the members, in the group management service. The group management service, running in Bob's device, sends the base and application certificates and key pairs to the group management service running in each member's device. When the group management service in the member device receives the information, it updates the AA module (admin ID=`ID of Bob`, member ID= `allotted member ID`).

Adding other user's information: Now Ted is a non-member of the group created above. Ted has Bob as a verified user and has Bob's base certificate. Bob sends to Ted, the group's base and application certificates along with CertApplInfo and service ID information. Ted checks the signature of the base certificate with Bob's base public key. To Ted, the group acts just like another user. Ted also sends his base and application certificates and other information to Bob. Since Ted is already a verified user of Bob, Bob need not update the base and application-specific certificates of Ted. However, Bob updates his AA module with the information that Ted is a verified user of the group. This helps in updating the group members at a later stage. Bob calls his group management service for updating the group members. The group members update Ted's information as a verified user for the role as the group. Thus for Alice, Carol and Dave, Ted is a verified user for the role as the group created by Bob. Thus, they trust the administrator of the group to add another user as verified user only for the role as the group.

Creating Symmetric Keys

A user in a role (acting as `SELF` or a group member) can have any number of symmetric keys with each identified user or group.

Between users: Bob wants to create a symmetric key with Alice for a specific service. The users decide on the key and create it with the help of the Crypto Service module. The key is added to both devices along with CertApplInfo and service.

Between a user and group: As explained earlier, Ted is a non-member of the group created by Bob. There could be a set of symmetric keys for Ted with the group. To Ted, the group is just like another user. Bob (administrator) and Ted decide on the key and create the symmetric keys with the help of the Crypto Service module. The keys are added to both devices along with CertApplInfo. Bob then passes these keys to Alice, Carol and Dave who are the members of the group.

Between group members: The group members can have a set of symmetric keys for internal communication of the group for different services. Bob (administrator) creates the symmetric keys and CertApplInfo and passes them to Alice, Carol and Dave who are the members of the group.

Setting Access Control Rules

At this point, Bob has base information as well as application-specific information about other users. Bob can give them access rights to services. Access rights could be allotted to groups also. If needed, Bob can also set the security level required for communication with each access right setting.

Access Control and Security Level Decision

Now at a later point of time, Alice's device contacts Bob's device for accessing a service. It provides the service ID of the service to be accessed and the user-name (the application-specific user-name) or the CertApplInfo. A request comes to Bob's AA module to identify the user and the application certificate associated with the given CertApplInfo (or the given user-name). The AA module identifies the corresponding application certificate(s) and gives back an array of handles of the application certificates and a handle of the user to the application. Ideally only one element should be there since only one certificate should be identified with a given CertApplInfo. However, as explained earlier in Section 4.3.3, since the AA module cannot impose the uniqueness of the CertApplInfo provided, there can be situations where more than one application certificate matches the requested criteria. In that case, it is the responsibility of the application to identify, which user and application certificate to be used. If no certificates were identified, then a corresponding error will be returned.

At this point in time, the application got the handles for the user and application certificate. It can request various functionalities with the application certificates like encryption, decryption, signing etc. When it comes to encryption and decryption, the application doesn't even have to care whether it is symmetric or asymmetric. The application can ask for an authorization decision from the AA module for the user for accessing the service. The authorization function requires other parameters like the role in which Bob acts, the access type which indicates whether Alice accesses the service or provides it etc. The application could also ask for the required security level decision from the AA module.

Appendix B

Usage of AA Module

This chapter gives a brief explanation of how the AA module could be used by a service. The AA module implementation defines the following structures.

- *AA_ Con* - defines the connection with the AA module.
- *AA_ User* - identifies a user or group. The objects identifying roles, user and verified user are of this type.
- *AA_ Service* - identifies a service.
- *AA_ Cert* - identifies an application certificate.

The main objective of a service while using the AA module would be to obtain the required application certificate structure (*AA_ Cert*) or the user structure (*AA_ User*) such that it can perform various functionalities as explained in the Section 5.3. How a service obtains these objects is explained in the following three steps.

Step 1

Get the AA module connection. Every service, which needs to access the AA module APIs, should first secure an AA module connection. This could be done with the following function.

- *AA_getAACon()* - The service requests the required connection level by giving its identification along with the request. The connection object obtained is passed along with other function requests. Each function checks whether the service, which calls it, has the proper valid connection granted to access that function.

Step 2

Use the following functions to obtain the required service, role or verified user objects.

- *AA_getAAService()* - Gets the *AA_Service* structure for the service specified by the service ID.
- *AA_getAuthorizedServices()* - Gets the list of services for which the specified verified user or group is authorized to access or provide, while the device user acting in the specified role.
- *AA_getAllServices()* - Used to get a list of all registered services.
- *AA_getAAUser()* - Gets the *AA_ User* structure for the user or group specified by the user ID or group ID. The ID specified could be `AA_USER_SELF` for creating the *AA_ User* structure for `SELF`.
- *AA_getOwnRoles()* - Gives back a list of roles for the device user.
- *AA_getDefaultRole()* - Used to get the default role in which the device user will act.
- *AA_getVerifiedUsers()* - Gives the list of verified users and groups for the specified role. If the role specified is `NULL` then it would be treated as for `ALL` roles.
- *AA_getAuthorizedVerifiedUsers()* - Gets the list of verified users who are authorized to access or provide the specified service, while the device user acting in the specified role.

Step 3

If the previous step did not provide the required user or application certificate object, it would be possible to get them by using the appropriate parsing functions as listed below.

- *AA_getCertApplInfoList()* - Gets the list of CertApplInfos for the specified service and user. There could be more than one CertApplInfo for a service for a specific user.

- *AA_getApplCertificatesForCertApplInfo()* - Gets the list of application certificates which are linked with the given CertApplInfo and service. The service could be `NULL` if needed, which would fetch all the application certificates which are associated with the given CertApplInfo.

- *AA_getOwnerOfApplCertificate()* - Gets the owner of the application-specific certificate. If it is an asymmetric application certificate, then the corresponding user or group who is the owner of the certificate is returned. If it is a group symmetric certificate then the group is returned. If the application certificate is a verified symmetric certificate, then the corresponding role of the device user and the verified user are returned.

- *AA_getCertApplInfoListForApplCertificate()* - Gives the list of CertApplInfos and the services associated to a given application certificate.

- *AA_getApplCertificatesForUser()* - Gives the list of application certificates for a particular user defined for a particular service. If the service specified is `NULL`, then the list will contain all the application certificates of the specified user.

By calling the required functions as stated in the three (or two) steps above, a service will have the required objects for calling the functions, which provide the required functionalities like checking and setting authorization or access control for a verified user, performing encryption and decryption, creating own application certificate etc. The steps are shown in Figure B.1.

Even though it is not possible to present and explain all the interface functions in this document, we will consider the most common scenario where the AA module will be used by a service. Mostly for a Level 1 access connection, the objective would be to get the corresponding verified user such that the authorization and security level decision could be made. It might also be required that a particular application certificate of the verified user could be obtained such that encryption of the communication could be done. The service obtains either the *CertApplInfo* (see Section 4.3.3) or the *user ID* from the verified user before starting the service access. In the first case, the service searches and finds the corresponding application certificate by using *AA_getApplCertificatesForCertApplInfo()*. The service finds the owner of the application certificate using *AA_getOwnerOfApplCertificate()*. Then it

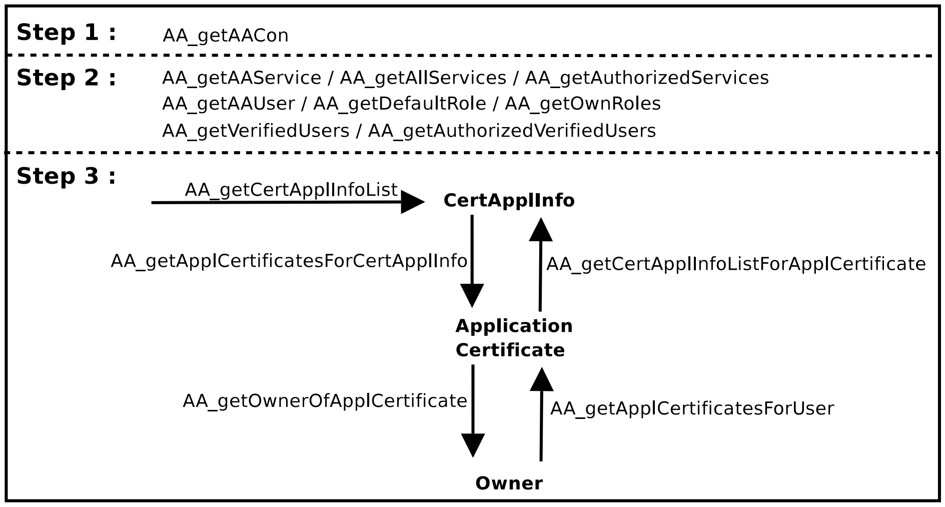
Step 1 :
AA_getAACon
Step 2 :
AA_getAAService / AA_getAllServices / AA_getAuthorizedServices
AA_getAAUser / AA_getDefaultRole / AA_getOwnRoles
AA_getVerifiedUsers / AA_getAuthorizedVerifiedUsers
Step 3 :
AA_getCertApplInfoList
CertApplInfo
AA_getApplCertificatesForCertApplInfo
AA_getCertApplInfoListForApplCertificate
Application Certificate
AA_getOwnerOfApplCertificate
AA_getApplCertificatesForUser
Owner

Figure B.1: AA module API usage by a service

checks the authorization permission and obtains the security level decision for the communication. If required, encryption of the communication is done using the application certificate obtained. In the second case where the *user ID* of the verified user is given, the application certificate for the service is obtained through the *AA_ getApplCertificateForUser()* function.

Bibliography

[1] Mac OS X v10.2 Technologies: Rendezvous, October 2002.

[2] C-K Toh. *Ad Hoc Mobile Wireless Networks: Protocols and Systems.* 2002.

[3] Calhoun, P. R., Johansson, T., Perkins, C. E., and Hiller, T. Diameter Mobile IPv4 Application. Internet draft, IETF, February 2004. Expires July 2004.

[4] Calhoun, P. R., Loughney, J., Guttman, E., Zorn, G., and Arkko, J. Diameter Base Protocol. RFC 3588, IETF, September 2003.

[5] Calhoun, P. R., Spence, D., and Mitton, D. Diameter Network Access Server Application. Internet draft, IETF, February 2004. Expires Aug 2004.

[6] Calhoun, P. R., Zorn, G., Pan, P., and Akhtar, H. Diameter Framework Document. Internet draft, IETF, February 2001. Expires July 2001.

[7] Charles E. Perkins. *Ad Hoc Networking.* Addison-Wesley, 2001.

[8] de Laat, C., Gross, G., Gommans, L., Vollbrecht, J., and Spence, D. Generic AAA Architecture. RFC 2903, IETF, August 2000.

[9] E. Guttman and C. Perkins and J. Kempf. Service Templates and Service: Schemes. RFC 2609, IETF, June 1999.

[10] Elizabeth M. Royer and C.K. Toh. A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks. IEEE Personal Communications, pages 46-55, April 1999.

[11] Ellison, C. SPKI Requirements. RFC 2692, IETF, September 1999.

[12] Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., and Ylonen, T. SPKI Certificate Theory. RFC 2693, IETF, September 1999.

[13] Guttman E. and Perkins C. and Veizades J. and Day M. Service Location Protocol, Version 2. RFC 2608, The Internet Engineering Task Force, June 1999.

[14] Housley, R., Polk, W., Ford, W., and Solo., D. Internet X.509 Public Key Infrastructure, Certificate and Certificate Revocation List (CRL) Profile. RFC 3280, IETF, April 2002.

[15] I.F. Akyildiz and W. Su and Y. Sankarasubramaniam and E. Cayirci. Wireless Sensor Networks: A Survey. IEEE Computer, vol. 38, no. 4, pages 393-422, Mar. 2002.

[16] ITU-T Recommendation X.501. Information Technology - Open Systems Interconnection - The Directory: Models, 1993.

[17] ITU-T Recommendation X.509 (1997 E). Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, June 1997.

[18] Jean-Pierre Hubaux and Levente Buttyan and Srdan Capkun. The Quest for Security in Mobile Ad Hoc Networks. In Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC 2001).

[19] Jonathan Rosenberg and Henning Schulzrinne and Gonzalo Camarillo and Alan Johnston and Jon Peterson and Robert Sparks and Mark Handley and Eve Schooler. SIP: Session Initiation Protocol. RFC 3261, The Internet Engineering Task Force, June 2002.

[20] Lidong Zhou and Z.J. Haas. Securing Ad Hoc Networks. *IEEE Network 13*, 6 (Nov/Dec 1999), 24–30.

[21] Magnus Frodigh and Per Johansson and Peter Larsson. Wireless ad hoc networking - The art of networking without a network. Ericsson Review No. 4, 2000.

[22] Myers, M., Ankney, R., Malpani, A., Galperin, S., and Adams, C. X.509 Internet Public Key Infrastructure, Online Certificate Status Protocol - OCSP. RFC 2560, IETF, June 1999.

[23] project, T. O. OpenSSL: The open source toolkit for SSL/TLS. Available online: http://www.openssl.org/, 2004.

[24] R. Housley and W. Ford and W. Polk and D. Solo. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. RFC 2459, IETF, January 1999.

[25] Rigney, C., Willens, S., Rubens, A., and Simpson, W. Remote Authentication Dial In User Service (RADIUS). RFC 2865, IETF, June 2000.

[26] University of Helsinki (UHE) and Helsinki University of Technology (HUT) and Tampere University of Technology (TUT). SESSI Design of Experimentation 1. Tech. rep., September 2004.

[27] University of Helsinki (UHE) and Helsinki University of Technology (HUT) and Tampere University of Technology (TUT). SESSI Interface and Functional Specification. Tech. rep., July 2004.

[28] University of Helsinki (UHE) and Helsinki University of Technology (HUT) and Tampere University of Technology (TUT). SESSI Software Architecture. Tech. rep., June 2004.

[29] University of Helsinki (UHE) and Helsinki University of Technology (HUT) and Tampere University of Technology (TUT). SESSI State of the Art Analysis. Tech. rep., May 2004.

[30] University of Helsinki (UHE) and Helsinki University of Technology (HUT) and Tampere University of Technology (TUT). SESSI Use Cases. Tech. rep., March 2004.

[31] Yongguang Zhang and Wenke Lee. Intrusion Detection in Wireless Ad-Hoc Networks. In Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom 2000), August 2000, Boston, Massachussetts.