İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

DISTRIBUTED KEY MANAGEMENT SYSTEM IN AD HOC NETWORKS

M.Sc Thesis by

Elif AKTEN

Department: Computer Engineering

Programme: Computer Engineering

Supervisor: Prof. Dr. Bülent ÖRENCİK

December 2004

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

DISTRIBUTED KEY MANAGEMENT SYSTEM IN AD HOC NETWORKS

M.Sc. Thesis by
Elif AKTEN
(504011392)

Date of submission : 20 December 2004
Date of defence examination: 7 January 2005

Supervisor (Chairman): Prof. Dr. Bülent ÖRENCİK
Members of the Examining Committee Doç. Dr. Coşkun Sönmez
Prof.Dr. Ertuğrul ÇELEBİ

DECEMBER 2004

ACKNOWLEDMENTS

I have had a great honor to be supervised by Professor Bülent ÖRENCİK. I would like to take this opportunity to acknowledge his scientific guidance and support.

A person owes most of her/his knowledge to the business environment, and I feel very lucky to have such colleague, teaching me so much and making the office a great place. I thank all my friends and especially to our Project Manager for her tolerance.

I would like to thank my family, especially my father and my mother, for always being there for me when I needed their prayers.

Elif AKTEN December 2004

TABLE OF CONTENTS

ABBREVIATIONS

AODV : Ad hoc On Demand Distance Vector
CA **:** Certification Authority
CRL **:** Certificate Revocation List
CSMA/CA **:** Carrier Sense Multiple Access with Collision Avoidance
DH **:** Diffie-Hellman
DOS **:** Denial of Service
DSDV : Destination Sequence Distance Vector
DSR : Dynamique Source Routing
ICMP : Internet Control Message Protocol
IP : Internet Protocol
IPSEC : Internet Protocol Security
ISM : Industrial, Scientific, and Medicine
KDC : Key Distribution Center
KTC : Key Transaction Center
LAN : Local Area Network
LOS : Line Of Sight
MAC : Message Authentication Code
MANET : Mobile Ad hoc Network
MD4 : Message Digest Algorithm Version 4
MD5 : Message Digest Algorithm Version 5
OCSP : Online Certificate Status Protocol
PGP : Pretty Good Privacy
PK : Public Key
PKG : Private Key Generation
PKI : Public Key Infrastructure
RA : Registration Authority
RF : Radio Frequency
RSA : Rivest, Shamir, and Adleman
SAODV : Secure AODV
SHA-1 : Secure Hash Algorithm
SK : Secret Key
SPKI : Simple Public Key Management
SRP : Secure Routing Protocol
SUCV : Statistically Unique Cryptographically Verifiable
TCP : Transmission Control Protocol
TESLA : Timed Efficient Stream Loss-tolerant Authentication
TTP : Trusted Third Party
VSS : Verifiable Secret Sharing

LIST OF FIGURES

Page No

TASARISIZ AĞLARDA DAĞITIK ANAHTAR YÖNETİM SİSTEMİ

ÖZET

Tasarısız ağlar hareketli sunucular için yeni bir kablosuz dağıtık haberleşme paradigması oluşturmuşlardır. Geleneksel ağlarda terminaller birbirleriyle kablolar aracılığıyla bağlıdırlar, ayrıca ağ üzerinde özelleşmiş sunuculara ihtiyaç duyarlar(Örneğin paketleri yönlendiren yönlendiriciler).

Tasarısız ağlarda ise terminaller kablosuz arabirimler ile birbirlerine bağlıdırlar ve önceden ayarlanmış özel bir altyapıya ihtiyaç duymazlar. Mesela her terminal bir yönlendirici sorumluluğunu yerine getirerek paketleri diğer terminallerin yerine ulaştırırlar.

Tasarısız ağlar savaş alanlarında, askeri personelin merkezi ve kalıcı bir altyapıya ihtiyaç duymadan haberleşebilmesi amacıyla kullanılmak üzere geliştirildiler ve basit mobil cihazları destekleyecek şekilde tasarlandılar.

Mavi diş ve IEEE 802.11 gibi kablosuz teknolojilerinde görülen gelişmeler tasarısız ağları daha da cazip hale getirdi ve günümüzde bir çok değişik alanda kendine uygulama alanı bulmuştur.

Kablosuz ağlar güvenlik saldırılarına, paket taşıma işleminin hava ortamında yapılması sebebiyle, geleneksel ağlardan daha fazla açıktır.

Özellikle haberleşmeyi dinleme, ikinci kere tekrarlama ve araya girip değiştirme saldırılarına karşı tam bir hedef durumundadırlar. Dolayısıyla kablosuz ağlar bu tip saldırılara karşı koyacak mekanizmaları içinde barındıracak şekilde yapılandırılmalıdırlar.

Tasarısız ağlar için tasarlanan pek çok uygulama sağlam güvenlik yapıtaşlarına ve gizlilik koruma metotlarına ihtiyaç duymaktadırlar. Sabit yapıları ve kablolu terminalleri olan ağlar bu ihtiyaçları karşılayabilmek için merkezi güvenlik çözümleri kullandılar.(Örneğin güvenilir üçüncü partiler, ateş duvarları)

Ancak tasarısız ağların sabit bir yapıları olmadığı ve kaotik karakteristiklerinden dolayı bu tür mekanizmaların etkinliği önemli ölçüde kısıtlanmış oldu.

Bir iletişim sisteminin başarısı iletilen bilginin güvenliğinden emin olunmasıyla ölçülür. İletişim kuran her hangi iki birimin efektif olarak haberleşebilmeleri için karşılıklı güven kurmaları gerekmektedir. Doğrulama(yetkilendirme) böyle bir güven kurmanın temel yoludur. İletişimin herhangi iki düğüm arasında olabileceği büyük ve dinamik ağlarda her düğümün diğer düğümleri doğrulayabilmesini varsaymak imkansızdır. Bu yüzden doğrulama servisi ayrıca sağlanmalıdır. Bunun geleneksel yöntemlerinden biri de ortak güvenilir üçüncü tarafların devreye sokulmasıdır. Açık Anahtar Altyapılarındaki sertifika otoritesi böyle bir yaklaşımın en başarılı örneklerinden biridir.

Açık anahtar altyapısı iletişim kuran düğümlerin sertifika otoritesi aracılığıyla birbirlerini doğrulayabildikleri herhangi bir ağda kullanılabilir. Ancak böyle bir yapının tasarısız kablosuz ağlarda kullanılması doğası gereği oldukça güçtür. Açık anahtar altyapısı geleneksel ağlarda başarıyla uygulanmasına rağmen , kablosuz tasarısız ağlarda kabul görüp görmeyeceği belli değildir.

Bu tezde tasarısız ağların önündeki engelleri inceledik, gerekli çözümlerin isterlerini ortaya koyduk, güvenliğin ne kadar önemli olduğunun altını çizdik ve tasarısız ağlar için uygulanabilir bir açık anahtar altyapısı çözümü ortaya koyduk. Aynı zamanda tasarısız ağları bekleyen güvenlik saldırılarını inceledik ve karşı koymak için gerekli güvenlik adımlarını sunduk. Bu tezde önerdiğimiz sertifika otoritesi sağlayan pratik çözüm bir dizi değişik ağ konfigürasyonuna uygun ve tasarısız ağlarda kullanılan yönlendirme protokollerine kolay ve basit şekilde uyarlanabilir.

Yüksek güvenlikli ve kullanılabilir bir anahtar yönetim yapısı kurabilmek amacıyla, sorumluluğu birden fazla sunucuya yayan eşik kriptografisi kullanmayı öneriyoruz. Hatta tümüyle dağıtık sertifika otoritesi yöntemini benimsiyoruz, ki bu çözümde ağdaki tüm düğümler sertifika otoritesinin yeteneklerine sahiptir. Bu çalışma ölçeklenebilir, dağıtık bir anahtar yönetim servisinin tasarısız ağlar için formüle edilmesini amaçlamaktadır.

Prototip bir dağıtık anahtar yönetim servisi yapılabilirliğini ve performansını göstermek amacıyla geliştirilmiştir. Bu AAA(Açık Anahtar Altyapısı) servisi ile haberleşmek için verimli ve efektif bir iletişim sistemidir.

DISTRIBUTED KEY MANAGEMENT SYSTEM IN AD HOC NETWORKS

SUMMARY

Ad hoc networks are a new wireless, decentralized communication paradigm for mobile hosts. In traditional networks nodes are interconnected through wired links and there are specialized nodes, i.e. routers that handle packet forwarding. In Ad hoc networks mobile nodes are interconnected through wireless interfaces, and nodes do not rely on any fixed infrastructure. Instead every node in the network functions as a router as well as an application node and forwards packets on behalf of other nodes.

Ad hoc networks were initially developed explicitly for battlefield settings, where military personnel required a communication medium that did not rely on a centralized entity or wired infrastructure, and had the capability to support lightweight, mobile nodes. Developments in wireless technologies, such as Bluetooth and IEEE 802.11 made Ad hoc networks more attractive. Also civilian applications exploit the advantages of Ad hoc networks and although there is a trend to adopt Ad hoc networks for commercial uses due to their unique properties.

Wireless networks are more prone to security attacks as all transmissions are carried out using the air medium. They are especially susceptible to attacks of eavesdropping, replay and spoofing. These systems therefore need to have built-in features to withstand these attacks without compromising security in any way.

Many emergent applications designed for Ad hoc networks necessitate robust security primitives and privacy protection. Networks with fixed topologies and wired hosts have employed centralized security solutions (e.g., trusted third-parties, firewalls, etc) to fulfill these requirements. However, the lack of infrastructure and ephemeral nature of Ad hoc networks has limited the effectiveness of similar security mechanisms in server-less contexts.

The success of a communication system is limited by how much trust can be placed in the information being transmitted. For any two communicating entities, mutual trust must be established to support effective communication. Authentication is the fundamental basis for building such trust. In large-scale dynamic networks, where communication may occur between any two entities, it is not feasible to expect every entity to have the capability to authenticate all other entities with which it might communicate. To this end,

it is necessary to provide a framework for authentication. One of the traditional ways to support such authentication is to involve a commonly trusted third party. The Certificate Authority in the Public Key Infrastructure is one of the most popular and successful examples of such an approach. Public Key Infrastructure can be used in any networks where the communicating parties can authenticate each other via correspondence with the Certificate Authority. However, providing such an infrastructure in Ad hoc wireless networks is a challenging task due to their infrastructure-less nature. Although Public Key Infrastructure has been successfully deployed in wired and infrastructure-based networks, it is unclear if this is approach can be successfully extended to wireless Ad hoc networks.

In this thesis, we studied these challenges in detail, identified the requirements for such solutions, pointed out the importance of its security and proposed a practical Public Key Infrastructure service for Ad hoc networks. Also, we have analyzed the security threats an Ad hoc network faces and presented the security objectives that need to be achieved. Furthermore we presented a practical solution to provide Certificate Authority support that is also flexible to the varying network configurations simple to implement and easy to integrate with Ad hoc network routing protocols.

To build a highly available and highly secure key management service, we propose to use threshold cryptography to distribute trust among all servers. We also adopt the fully distributed certificate authority approach, which means the capabilities of the certificate authority are distributed to all nodes in the Ad hoc network. Moreover this work aims at providing a scalable, distributed key management services in Ad hoc networks.

A prototype of the distributed key management service has been implemented, which shows its feasibility and availability. Using this practical PKI service, we present an effective and efficient communication protocol for correspondence with certification services.

1. INTRODUCTION

A wireless Ad hoc network is a collection of wireless mobile hosts forming a temporary network without the aid of any established infrastructure or centralized computer such as base stations or mobile switching centers. Ad hoc networks rely on each other to keep the network connected rather than relying on any fixed infrastructure, unlike traditional mobile wireless networks. Battlefields, military tactical operations where geographical or terrestrial constraints demand a totally distributed network without any fixed infrastructures, are the main applications of Ad hoc networks. However, applications of Ad hoc networks range from emergency and disaster situations to commercial applications such as sensor networks and virtual classrooms. Ad hoc networks are an emerging research area with practical applications. Obviously, security is an important issue in such applications. However, they are vulnerable due to its fundamental characteristics [1].

The main challenge in the design of mobile Ad hoc networks is their vulnerability to security attacks. Like many distributed systems, security in Ad hoc networks widely relies on the use of key management mechanisms. Specific key management systems have to be developed to suit the characteristics of mobile Ad hoc networks because traditional key management systems are not appropriate for them.

1.1 Background

Ad hoc networking is a wireless networking paradigm for self-organizing networks that until recently has mainly been associated with military battlefield networks. However, with the availability of wireless technologies such as Bluetooth and IEEE 802.11 and the development of the next generation networks, civilian applications that exploit the advantages of Ad hoc networking are being envisioned. So far most of the research that

has been done on Ad hoc networking has focused on routing. Other issues such as security and network addressing have received considerably less attention and these issues need to be addressed before any successful applications will appear.

1.2 Purpose

Security is an important subject for Ad hoc networks, especially for the applications that are security-sensitive. To secure an Ad hoc network, we consider the following attributes:

1. Availability
2. Confidentiality
3. Integrity
4. Authentication
5. Non-repudiation.

We employ cryptographic schemes, such as digital signatures, to protect both routing information and data traffic. Also we used encryption scheme for confidentiality of routing information and network traffic [1].

Use of such schemes usually requires a key management service. We adopt a public key infrastructure because of its superiority in distributing keys and in achieving integrity, confidentiality and non-repudiation. Because of distributed nature of Ad hoc networks deploying public key infrastructure is not as easy as wired networks. Efficient secret key schemes are used to secure further communication after nodes authenticate each other and establish a shared secret session key [1].

2. THEORETICAL BACKGROUND

In this section we presented the background knowledge on Ad hoc Networks, standards, network security and cryptography.

2.1 Ad Hoc Networks

Mobile Ad hoc network is a collection of wireless mobile nodes dynamically forming a temporary network without the use of any existing network infrastructure or centralized administration. Due to the limited transmission range of wireless network interfaces, multiple hops may be needed for one node to exchange data with another across the network [1].

2.1.1 Ad hoc networks characteristics

There are a number of characteristics in mobile Ad hoc networks. One of them is that they have dynamic topologies. Nodes are free to move arbitrarily. Thus, the network topology is typically multi-hop, so may change randomly and rapidly at unpredictable times [2].

Another characteristic is bandwidth constraint. Wireless links will continue to have significantly lower capacity than their hardwired counterparts.

Also, there is energy-constraint in this type of networks. Some or all of the nodes in a mobile Ad hoc network may rely on batteries or other exhaustible means for their energy.

Finally, there is limited physical security. Mobile wireless networks are generally more prone to physical security threats than are fixed-cable nets. The increased possibility of eavesdropping, spoofing, and denial-of-service attacks should be carefully considered [2].

2.1.2 Ad hoc networks applications

Examples of potential practical use of mobile Ad hoc networks may be a group of people with laptop computers at a conference that may wish to exchange files and data without mediation of any additional infrastructure in-between. It may be used in home environment for communication between smart household appliances. Ad hoc networks are suitable to be used in areas where earthquake or other natural disasters have destroyed communication infrastructures. It perfectly satisfies military needs like battlefield survivability, operation without pre-placed infrastructure and connectivity beyond the line of sight. For monitoring and measuring purposes a large number of small computing devices could be spread over a hostile to form self-sustained Ad hoc networks.

Mobile Ad hoc networks have significant advantages above traditional communication networks. For example, use of Ad hoc networks could increase mobility and flexibility, as Ad hoc networks can be brought up and torn down in very short time. Ad hoc networks could be more economical in some cases as they eliminate fixed infrastructure costs and reduce power consumption at mobile nodes. They are more robust than conventional wireless networks because of their non-hierarchical distributed control and management mechanisms. Also, radio emission levels could be kept at low level because of short communication links (node-to-node instead of node to a central base station), this increases spectrum reuse possibility or possibility of using unlicensed bands. Moreover, communication beyond Line of Sight (LOS) is possible at high frequencies because of multi-hop support in Ad hoc networks.

Despite the mentioned advantages and potential application possibilities, Ad hoc networks are yet far from being deployed on large-scale commercial basis. Some fundamental Ad hoc networking problems remain unsolved or need optimized solutions. Although various routing protocols are suggested and tested for mobile Ad hoc networks, performance metrics like throughput, delay and protocol overhead in relation to successfully transmitted data need better optimization. This optimization would probably also depend on application type and desire to maximize the throughput or minimize the delay. One single protocol will probably not be able to work efficiently across entire range of design parameters and operating conditions.

2.1.3 Standards Used in Ad hoc Networks

In this part we will give some knowledge about wireless technologies and communication standards.

2.1.3.1 IEEE 802.11

IEEE 802.11 is a digital wireless data transmission standard in the 2.4 GHz Industrial, Scientific, and Medicine (ISM) band aimed at providing a wireless Local Area Network (LAN) between portable computers and a fixed network infrastructure. This standard defines a physical layer and a MAC layer. The most popular technology is the direct sequence spread spectrum and can offer a bit rate of up to 11 Mbps in the 2.4 GHz band, and in the future, up to 54Mbps in the 5GHz band. The basic access method in the IEEE 802.11 MAC protocol is the Distributed Coordination Function which is a Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) MAC protocol. However, the 802.11 standard cannot do multi-hop networking as it is. The development of a number of protocols is required. The maximum data rate of IEEE 802.11 is 11Mbps. Its range is 100 meters [3].

2.1.3.2 Bluetooth

Bluetooth is a digital wireless data transmission standard operating in the 2.4 GHz ISM band aimed at providing a short-range wireless link between laptops, cellular phones and other devices. This band defines 79 different Radio Frequency (RF) channels that are spaced of 1 MHz. The main aim of the Bluetooth specification is to guarantee the interoperability between different applications that may run over different protocol stacks.

However, in order to implement a wireless multi-hop network over Bluetooth, either or both a packet switch layer and a circuit switch layer need to be defined on top of the Bluetooth data link layer protocol.

The maximum data rate of Bluetooth is 1Mbps. Its range is 10 meters. Bluetooth has lower power consumption than IEEE 802.11. Also, Bluetooth support both voice and data packet types while IEEE 802.11 just support data packet type [3].

2.2 Network Security

When discussing network security, three aspects can be covered; the services required, the potential attacks and the security mechanisms. The security services aspect includes the functionality that is required to provide a secure networking environment while the security attacks cover the methods that could be employed to break these security services. Finally the security mechanisms are the basic building blocks used to provide the security services [5].

2.2.1 Security Services

In providing a secure networking environment some or all of the following services may be required [5]:

- Confidentiality
- Authentication
- Integrity
- Non-repudiation

2.2.1.1 Confidentiality

The confidentiality service ensures that certain information being transferred is never disclosed to unauthorized entities. Not only is the confidentiality of the transmitted data also the confidentiality of location (trace-ability) and traffic analysis important in the ubiquitous computing environment [5].

2.2.1.2 Authentication

The authentication service enables a node to ensure the identity of the peer node it is communicating with. Without authentication, an adversary could masquerade a node,

thus gaining unauthorized access to resource and sensitive information and interfering with the operation of other nodes [5].

2.2.1.3 Integrity

The integrity service guarantees that a message being transferred is never corrupted. A message could be corrupted because of benign failures, such as radio propagation impairment, or because of malicious attacks on the network. Integrity service ensures that only authorized parties are able to modify transmitted information. Modification includes writing, changing status, deleting, creating or replaying of transmitted messages [5].

2.2.1.4 Non-repudiation

The non-repudiation service ensures that parties can prove the transmission or reception of information by another party, i.e. a party cannot falsely deny having received or sent certain data [5].

2.3 Security Attacks

Security attacks can be classified in the following two categories [6] depending on the nature of the attacker:

- Passive Attacks
- Active Attacks

2.3.1 Passive Attacks

The attacker can only eavesdrop or monitor the network traffic. Typically this is the easiest form of attack and can be performed without difficulty in many networking environments, e.g. broadcast type networks such as Ethernet and wireless networks [5].

In a passive attack, the attacker does not disrupt the operation of a routing protocol but only attempts to discover valuable information by listening to the routing traffic. The

major advantage for the attacker in passive attacks is that in a wireless environment the attack is usually impossible to detect. This also makes defending against such attacks difficult. Furthermore, routing information can reveal relationships between nodes or disclose their Internet Protocol (IP) addresses. If a route to a particular node is requested more often than to other nodes, the attacker might expect that the node is important for the functioning of the network, and disabling it could bring the entire network down [5].

Other interesting information that is disclosed by routing data is the location of nodes. Even when it might not be possible to pinpoint the exact location of a node, one may be able to discover information about the network topology [5].

Depending on the attackers action passive attacks can be classified in the two categories.

2.3.1.1 Eavesdropping

This attack is used to gain knowledge of the transmitted data. This is a passive attack, which is easily performed, in many networking environments. However using an encryption scheme to protect the transmitted data can easily prevent this attack [5].

2.3.1.2 Traffic Analysis

The main goal of this attack is not to gain direct knowledge about the transmitted data, but to extract information from the characteristics of the transmission, e.g. amount of data transmitted, identity of the communicating nodes etc. This information may allow the attacker to deduce sensitive information, e. g the roles of the communicating nodes, their position etc. Unlike the eavesdropping attack this one is more difficult to prevent [6].

2.3.2 Active Attacks

The attacker is not only able to listen to the transmission but is also able to actively alter or obstruct it.

To perform an active attack the attacker must be able to inject arbitrary packets into the network. The goal may be to attract packets destined to other nodes to the attacker for

analysis or just to disable the network. A major difference in comparison with passive attacks is that an active attack can sometimes be detected. This makes active attacks a less inviting option for most attackers. Yet, it may still be a real alternative when a large amount of money is at stake such as in commercial or military environments [6].

Depending on the attackers action active attacks can be classified in the nine categories.

2.3.2.1 Impersonation

Here the attacker uses the identity of another node to gain unauthorized access to a resource or data. This attack is often used as a prerequisite to eavesdropping. By impersonating a legitimate node, the attacker can try to gain access to the encryption key used to protect the transmitted data. Once the attacker knows this key, it can successfully perform the eavesdropping attack [5].

2.3.2.2 Modification

This attack modifies data during the transmission between the communicating nodes, implying that the communicating nodes do not share the same view of the transmitted data [5].

2.3.2.3 Insertion

This attack involves an unauthorized party, who inserts new data claiming that it originates from a legitimate party. This attack is related to that of impersonation [5].

2.3.2.4 Replay

The attacker retransmits data previously transmitted by a legitimate node [5].

2.3.2.5 Denial of Service

This active attack aims at obstructing or limiting access to a certain resource. This resource could be a specific node or service or the whole network [5].

2.3.2.6 Black Hole

In this attack, a malicious node uses the routing protocol to advertise itself as having the shortest path to the node whose packets it wants to intercept.

In a flooding based protocol such as Ad hoc on Demand Distance Vector (AODV) the attacker listens to requests for routes. When the attacker receives a request for a route to the target node, the attacker creates a reply where an extremely short route is advertised. If the malicious reply reaches the requesting node before the reply from the actual node, a forged route has been created. Once the malicious device has been able to insert itself between the communicating nodes, it is able to do anything with the packets passing between them. It can choose to drop the packets to perform a denial-of-service attack, or alternatively use its place on the route as the first step in a man-in-the-middle attack [6].

2.3.2.7 Sleep Deprivation

Usually, this attack is practical only in Ad hoc networks, where battery life is a critical parameter. Battery powered devices try to conserve energy by transmitting only when absolutely necessary. An attacker can attempt to consume batteries by requesting routes, or by forwarding unnecessary packets to the node using, for example, a black hole attack.

This attack is especially suitable against devices that do offer some services to the network or offer services only to those who have some special credentials. Regardless of the properties of the services, a node must participate in the routing process unless it is willing to risk becoming unreachable to the network [6].

2.3.2.8 Location Disclosure

A location disclosure attack can reveal something about the locations of nodes or the structure of the network. The information gained might reveal which other nodes are adjacent to the target, or the physical location of a node. The attack can be as simple as using an equivalent of the trace route command on UNIX systems. Routing messages are sent with inadequate hop-limit values and the addresses of the devices sending the ICMP error-messages are recorded. In the end, the attacker knows which nodes are situated on

the route to the target node. If the locations of some of the intermediary nodes are known, one can gain information about the location of the target as well [6].

2.3.2.9 Routing Table Overflow

In a routing table overflow attack the attacker attempts to create routes to nonexistent nodes. The goal is to create enough routes to prevent new routes from being created or to overwhelm the protocol implementation.

Proactive routing algorithms attempt to discover routing information even before it is needed while a reactive algorithm creates a route only once it is needed. This property appears to make proactive algorithms more vulnerable to table overflow attacks. An attacker can simply send excessive route advertisements to the routers in a network. Reactive protocols, on the other hand, do not collect routing data in advance. For example in AODV, two or more malicious nodes would need to cooperate to create false data efficiently: The other node requests routes and the other one reply with forged addresses [5].

2.4 Security Mechanisms

Most of the security services previously mentioned can be provided using different cryptographic techniques. The following subsections give an overview of which techniques are used to provide each of the services [5].

2.4.1 Confidentiality

The confidentiality service requires protection from eavesdropping attacks and this can be provided using an encryption scheme. The stricter requirement implies that the service must also provide protection against traffic analysis. Such a service will typically require additional mechanisms along with some encryption scheme [5].

2.4.2 Integrity

The integrity service can be provided using cryptographic hash functions along with some form of encryption. When dealing with network security the integrity service is often provided implicitly by the authentication service [5].

2.4.3 Authentication

Authentication can be provided using encryption along with cryptographic hash functions [5].

2.4.4 Non-repudiation

Non-repudiation requires the use of public key cryptography to provide digital signatures. Along with digital signatures a trusted third party must be involved [5].

2.4.5 Availability

The availability is typically ensured by redundancy, physical protection and other non-cryptographic means, e.g. use of robust protocols [5].

2.5 Cryptographic Background

Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication.

The basic service provided by cryptography is the ability to send information between participants in a way that others cannot intercept or read it. Though there are many kinds of cryptography, the ones most commonly used are those based on representing information as numbers and mathematically manipulating those numbers. This kind of cryptography provides integrity checking and authentication. This section provides a general overview of various types of cryptographic algorithms that are commonly used in practice [8].

In traditional cryptography, a message in its original form is known as plain text or clear-text. The encrypted information is known as cipher-text and the process of producing this cipher-text is known as encryption. The reverse process of encryption is called as decryption. Cryptographic systems tend to involve an algorithm and a secret value. The secret value is known as the key. The reason for having a key in addition to an algorithm is that it is difficult to keep devising new algorithms that will allow reversible scrambling of information [8].

There are three-types of cryptographic functions:

- Symmetric Key Encryption
- Asymmetric Key Cryptography
- Hash Functions

2.5.1 Cryptographic Goals

The fundamental goal of cryptography is to address the confidentiality, data integrity, authentication, and non-repudiation in information security.

Confidentiality is a service used to keep the content of information from all but those authorized to have it. Data integrity is a service, which addresses the unauthorized alteration of data. Authentication is a service related to identification. Non-repudiation is a service, which prevents an entity from denying previous comments or actions. These services can be used to prevent and detect cheating and other malicious activities [8].

In the following subsections, we will present some popular cryptographic techniques, like symmetric-key encryption, asymmetric-key encryption, digital signatures, and digital certificates.

2.5.2 Symmetric Key Encryption

Symmetric key cryptography involves the use of a single key. Given a message (or a plain-text), encryption produces the cipher-text, which is about the same length of the

plain text and decryption retrieves back the plain text, using the same key that is used for encryption. This kind of encryption is also called as conventional or secret key cryptography. Symmetric encryption schemes can be used to provide confidentiality, integrity and authentication [8].

In Figure 2.1 it is shown that a plain text is encrypted with a shared key and produced cipher-text, and than cipher-text is decrypted using the same shared key used for encryption and plain text is obtained.

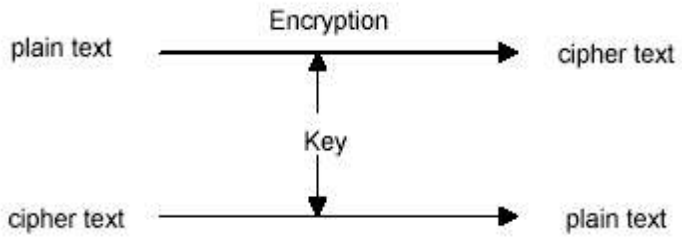

Figure 2.1 Secret Key Cryptographic System [5]

Symmetric key encryption is fast and secure. However, the shared key must be distributed over a secure communication channel.

Secret key systems provide strong authentication functionality. This implies that someone can prove knowledge of a secret without revealing it, a functionality that is essential for wireless systems.

Authentication is generally implemented using a challenge-response mechanism. For example, as shown in Figure 2.2, suppose A and B wish to communicate with each other and they decide upon a key K_{AB} to verify each other's identity. Each of them picks a random number, which is known as a challenge and send it to each other. The value of the random number, say x, encrypted with the key K_{AB} is known as the response to the challenge x. Thus, if A and B complete this exchange, they have proved to each other that they know K_{AB} without revealing it to an imposter or an eavesdropper. This kind of

challenge-response mechanism is used in GSM and the USDC systems (and infact in most of the mobile communication systems) for authenticating a mobile user. One apparent flaw in these kinds of systems is that an eavesdropper can form challenge-response pairs, since he/she can pose a challenge to either A or B and store the responses. To avoid this situation it is essential that the challenges be chosen from a large enough space, say 2^{128} values, so that there is no significant chance of using the same challenge twice. Also, note that the key K_{AB} can also represent an algorithm A_{AB}, which uses the random number x and produces an encrypted value. Only A and B know the algorithm [5].

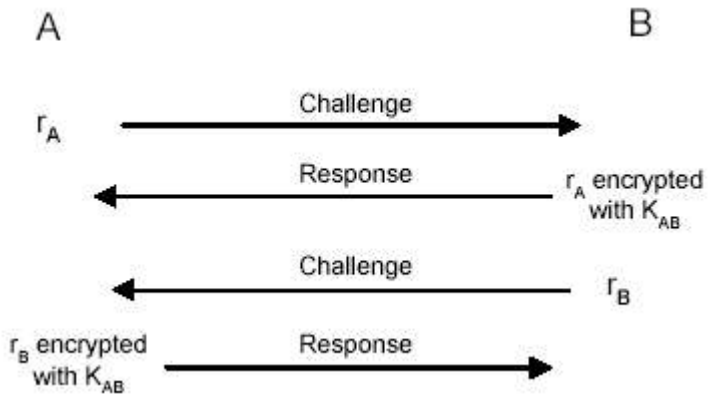

Figure 2.2 Secret Challenge-Response Mechanisms in Secret Key Systems [5]

2.5.3 Asymmetric Key Cryptography

Asymmetric key encryption depends on two different but mathematically related keys. In asymmetric key cryptography, the keys are not shared. Instead, each individual user has two keys: a private key (that is not revealed to anyone) and a public key (that is open to the public). This kind of cryptography is also commonly called as Public Key Cryptography and was invented by Diffie and Hellmann in 1975. In these systems, encryption is done using the public key and decryption is done using the private key.

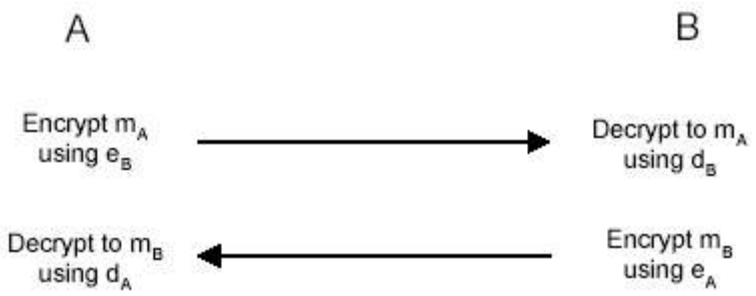

Figure 2.3 Information Transfer in a Public Key Cryptographic System

In Figure 2.3 it is shown that, two people, A and B, wishing to communicate over an insecure channel (say a wireless channel), suppose A's <Public Key, Private Key> pair is $<e_A; d_A>$ and B's pair is $<e_B; d_B>$. Assume that the public keys are known to both A and B. It's clear that each person encrypts the data using the other person's public key which can be decoded by the other person using his own private key. This kind of encryption/decryption is not much different from secret key systems, but the biggest benefit of public key systems over secret key systems comes from the authentication mechanism. In the case of authentication in secret key systems, if A and B want to communicate with each other, they have to share a secret (key K_{AB} or algorithm A_{AB}) among themselves. If one wants to communicate with a lot of entities he/she must remember a lot of secret keys each corresponding to every entity he wishes to communicate. Public key cryptography comes over this problem by the use of public keys. In this case, the entities that wish to communicate with each other have to remember only their private key. To communicate with another entity, they have to look-up the public key of the other entity (from a directory server) and use it encrypt the messages to be communicated to that entity.

In Figure 2.4 authentication mechanism is shown in asymmetric cryptography. For example, suppose A wants to verify (authenticate) B 's identity. A Chooses a random number r, encrypts it using B 's public key and sends the result to B. Now, B can prove his/her identity by decrypting the encrypted message (the challenge) using his/her private key and sending the decrypted random number r (the response) backs to A [8].

Figure 2.4 Authentication Mechanisms in a Public Key System

Though public key systems provide a highly efficient authentication mechanism, the down side of public key encryption is that it tends to be very slow and resource intensive. This makes it difficult to send large amounts of data using public key encryption. It is typically only used to encrypt small amount of data, like digital signatures.

In case of communication networks, these public key systems require excessive computations and transfer of large-number of bits along power/bandwidth-limited channels. So, these systems were not initially recommended for wireless/mobile communications where bandwidth and power (and hence battery life of portable devices) are at a premium. This is one of the main reasons that the 2nd Generation Systems like GSM and the USDC were primarily secret key systems. But as 3rd generation systems with higher capacities are introduced, private key systems will begin to play an important role in providing security, authentication and access control [8].

Public key cryptography also facilitates digital signatures, whereby a person can sign a plain text using his private key and anyone can verify the person's identity by using the public key of that person. Further, others cannot forge the signature of the person since it involves his private key.

2.5.2.1 Diffie-Hellman

The Diffie-Hellman (DH) algorithm was the first public key algorithm published. However, it is limited to securely exchanging keys that can subsequently be used to provide the security services mentioned above [5].

The DH algorithm shown in Figure 2.5 requires two public parameters, a prime p and a generator g of Z_P. A generator of Z_P is an integer g such that g, g^2... g^{p-1} (mod p) generate the values 1 through p - 1 in some order. To exchange a shared key user A and user B generate the random secrets x_A and x_B. User B sends $y_B = g^{x_B}$ to A and A sends $y_A = g^{x_A}$ to B. A and B can then generate the shared secret key k as:

$$k = y_A^{x_B} = y_B^{x_A} = g^{x_B \cdot x_A} \text{ (mod p)} \qquad (2.1)$$

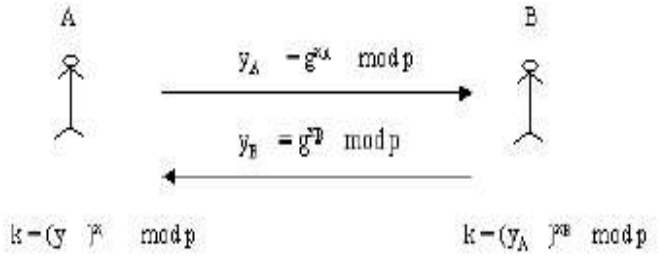

Figure 2.5 Diffie-Hellman key exchange.

2.5.2.2 RSA

RSA is the initials of the three creators: “Rivest, Shamir, and Ad leman”. It is the most widely used public-key cryptosystem. It is based on the following idea: It is very simple to multiply numbers together, especially with computers. But it can be very difficult to factor numbers. For example, if you multiply 34537 and 99991, it is a simple matter and the result is 3453389167. But the reverse problem is much harder. Suppose the number is 3453389167 then it is a hard problem finding the multiplying two integers.

RSA can be used to provide confidentiality, integrity, authentication and non-repudiation services. The public key $pk = \{e, n\}$ and the private key $sk = \{d, n\}$. To encrypt a message m or decrypt a cipher text c, the following calculations are performed:

- $c = m^e \pmod{n}$ (2.2)

- $m = c^d \pmod{n} = m^{ed} \pmod{n}$ (2.3)

If the algorithm is intended to use to provide confidentiality the values n and e are made publicly known while d is kept secret. For user A to encrypt a message intended for user B, B's public key pk_B is used for the encryption. Since only B has knowledge of the secret key sk_B it alone can decrypt the cipher text and recover the plain text,

$$m = d_{sk_B}(c) = c^d \pmod{n} \quad (2.4)$$

In the following example, suppose that person A wants to make a public key, and that person B wants to use that key to send A a message. In this example, we will suppose that the message A sends to B be just a number.

Here are the steps:

1. Person A selects two prime numbers. We will use p=23 and q=41 for this example, the real numbers should be much larger.

2. Person A multiplies p and q together to get $(p)\,.(q) = 943$. 943 is the "public key", which he/she tells to person B (and any one he/she wishes).

3. Person A also chooses another number e, which must be relatively prime to

$(p-1).(q-1)$. In this case, $(p-1)\,.(q-1) = (22)\,.(40) = 880$ so $e = 7$, e is also part of the public key, so B also is told the value of e.

4. Now B knows enough to encode a message to A. Suppose, for this example, that the message is the number M =35

5. B Calculates the value of $C = M^e \pmod{N}$ $=35^7 \pmod{943}$.

6. C = 545, the number 545 is the encoding that B sends to A.

7. Now A wants to decode 545. To do so, he needs to find a number d such that $e.d = 1 \pmod{(p-1).(q-1)}$, in this case, such that $7.d = 1 \pmod{880}$. A solution is $d = 503$.

8. To find the decoding, A must calculate $C^d \pmod{N} = 545^{503} \bmod 943$ so the decrypted message M = 35.

2.5.4 Hash Functions

One of the fundamental primitives in modern cryptography is the cryptographic hash function, often informally called a one-way hash function. Hash functions are also called as message-digests or one-way transformations. A hash function is a computationally efficient function mapping binary strings of arbitrary length to binary strings of some fixed length, called hash-values [8].

The basic idea of hashing or message digesting is to mangle the information so badly that the process cannot be reversed. A typical example of message digesting is password authentication in personal computer systems. For security reasons, the system does not store the actual (unencrypted) password, but a hashed or digested value of it. When a password is supplied, the system computes the hashed or digested value of the supplied password and compares it the stored hash value. Hashing can also be used for other functions such as message fingerprinting, digital signatures, message integrity checking etc [5].

In communication systems we shall often require evidence that a message that has been sent has not been subject to modification in any way. Typically this is carried out using a hash function. A hash function H when applied to a message M yields a value H (M) of specific length known as the hash value of that message. H (M) is often referred to as a

message digest. The mapping of messages to digests is one-way; given M and H (M) it should be computationally infeasible to and M ' such that

H (M ') =H (M). (2.5)

The digest is a form of reduced message calculated using a publicly known technique. A receiver of a message can check whether a message and a corresponding digest agree. Hash functions are largely intended for use in conjunction with cryptography to provide signatures. If M is a message then A can provide evidence to B that he created it, and that it has not been tampered with, by calculating E (K_{AB} : H (M)) and sending the message M together with the newly calculated encrypted hash value. On receiving the message, B can calculate H (M) and then E (K_{AB} : H (M)) and check whether the value agrees with the encrypted hash value received. Since the amount of encryption is small this is a quite efficient means to demonstrate authenticity (assuming A and B do not fake messages from the other) [5].

There are lots of messages digest algorithms; one most popular one is message digest algorithm version 5 (MD5). Ronald Rivest created MD5. This algorithm consists of a number of rounds that apply different bit wise functions to get to the resulting output. MD5 uses four rounds with each consisting of 16 steps. The output of this algorithm is a 128-bit hash value. As input a bit string of arbitrary length is accepted. Since the algorithms work with blocks of 512 bits the input must be padded. In addition, to overcome the security problem of finding different length messages with the same hash a binary length value is concatenated to the message [8].

The most common cryptographic uses of hash functions are with digital signatures and for data integrity. With digital signatures, a long message is usually hashed (using a publicly available hash function) and only the hash-value is signed. The party receiving the message then hashes the received message, and verifies that the received signature is correct for this hash-value. This saves both time and space compared to signing the message directly, which would typically involve splitting the message into appropriate-sized blocks and signing each block individually. Note here that the inability to find two

messages with the same hash-value is a security requirement, since otherwise, the signature on one message hash-value would be the same as that on another, allowing a signer to sign one message and at a later point in time claim to have signed another [8].

At the highest level, hash functions may be split into two classes:

- Un-keyed hash functions: this specification dictates a single input parameter (a message).
- Keyed hash functions: this specification dictates two distinct inputs, a message and a secret key.

2.5.4.1 Modification Detection Codes

Modification detection code (MDC) also known as manipulation detection codes, and less commonly as message integrity codes (MIC), is a subclass of un-keyed hash functions. The purpose of an MDC is (informally) to provide a representative image or hash of a message, the end goal is to facilitate, in conjunction with additional mechanisms data integrity assurances as required by specific applications [8].

2.5.4.2 Message Authentication Codes

Message Authentication Code (MAC) is a subclass of keyed hash functions. The purpose of MAC is (informally) to facilitate, without the use of any additional mechanisms, assurances regarding both the source of a message and its integrity MAC have two functionally distinct parameters, a message input and a secret key [8].

2.5.4.3 Secure Hash Algorithm

Secure Hash Algorithm version (SHA-1) is based upon the MD5 algorithm. However, since the output hash value is larger, 160 bits as opposed to 128 in MD5, it is more secure against brute force attacks. The internal processing also provides for added security thanks to the redundancy of input words used in an expansion step [5].

2.5.5 Digital Signatures

A digital signature is a data structure that provides proof of origin, i.e. authentication and integrity, and depending on how it is used, it can also provide non-repudiation. Digital signatures have many applications in information security, including authentication, data integrity, and non-repudiation. One of the most significant applications of digital signatures is the certification of public keys in large networks [8].

If user A wants to send a message to user B, and doesn't want it to be modified during transmission and B wants to be sure that the message really came from user A. What user A does is that he/she computes a hash digest of the message, which he/she encrypts with his private key sk_A. He/she then sends both the message and the encrypted digest, which is his/her signature. User B can then verify the signature by computing the hash digest of the message he received and comparing it with the digest he/she gets when decrypting the signature using A's public key pk_A. If the digests are equal user B knows that A sent the message and that it has not been modified since he/she signed it.

Digital Signature Generation:

If an entity A which has a public private key pair (pk_A, sk_A) wants to generate a digital signature for a message M:

1. Compute $\tilde{M} = \mathrm{H}(M)$ (2.6)

2. Compute $s^* = sk_A(\tilde{M})$ (2.7)

Message M and signature s^* are made available to entities, which may wish to verify the signature.

Digital Signature Verification:

If an entity B wishes to verify a message M and signature s^*:

1. Obtain A's public key pk_A

2. Compute $\tilde{M} = \mathrm{H}(M)$ (2.8)

3. Compute $\mathrm{u} = pk_A(\tilde{M}, s^*)$ (2.9)

4. Verify signature if u=true

Both digital signature generation and digital signature verification is shown in Figure 2.6

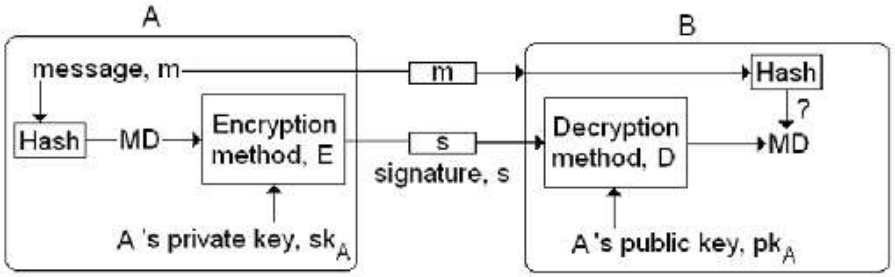

Figure 2.6 Digital Signatures

Digital signatures are easily transportable, cannot be imitated by someone else, and can be automatically time-stamped. The purpose of a digital signature is to provide a means for an entity to bind its identity to a piece of information held by the entity into a tag called a signature [8].

2.6 Threshold Cryptography

Threshold cryptography is distribution of trust. An (n, t + 1) threshold cryptography scheme allows n parties to share the ability to perform a cryptographic operation (e.g., creating a digital signature), so that any t + 1 parties can perform this operation jointly, whereas it is infeasible for at most t parties to do so, even by collusion. For the service to

tolerate t-compromised servers employ an (n, t+1) threshold cryptography scheme and divide the private key k of the service into n shares ($s_1, s_2, \ldots s_n$), assigning one share to each server. We call ($s_1, s_2, \ldots, s_n$) an (n, t + 1) sharing of k.

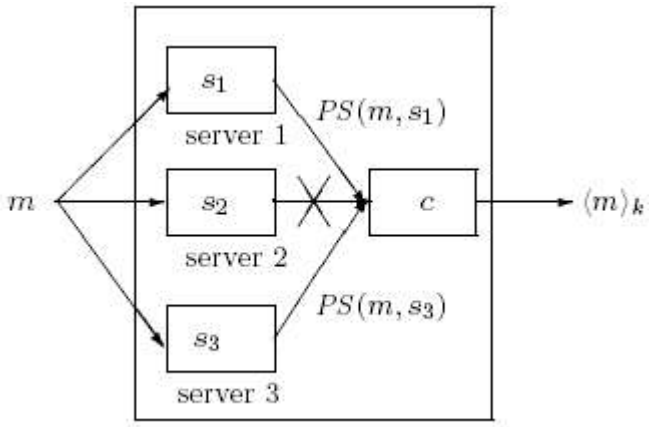

Figure 2.7 Threshold Cryptography

2.7 Secret Sharing

Secret sharing allows a secret to be shared among a group of users (share holders) in such a way that no single user can deduce the secret from his share alone. Only by combining a sufficient number of the shares can the secret be reconstructed. A secret sharing scheme where k out of n shareholders is needed to reconstruct the secret is referred to as a (n, k) threshold scheme.

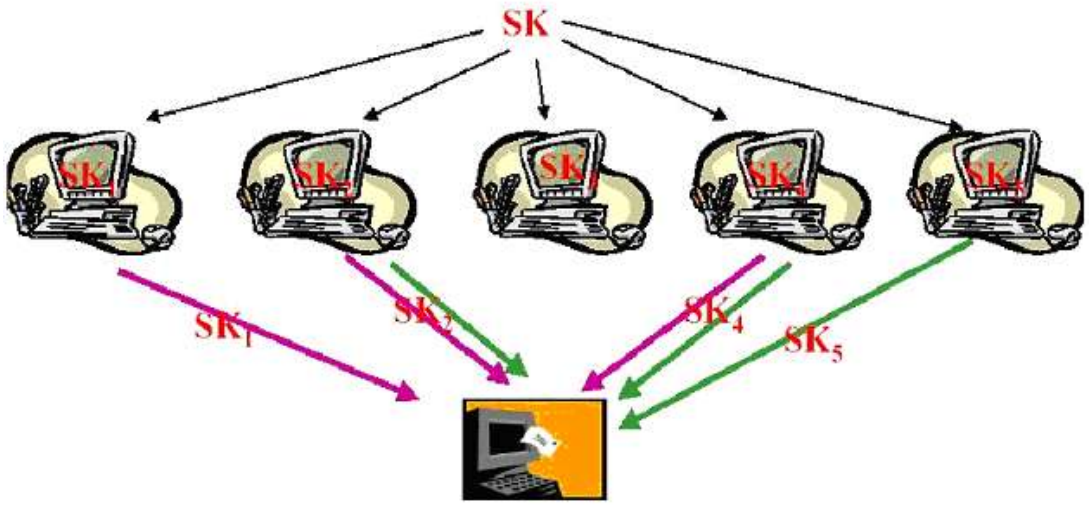

Figure 2.8 Secret Sharing

2.7.1 Shamir's Secret Sharing

This (n, k) threshold secret sharing scheme proposed by Adi Shamir [8] is based on polynomial interpolation and works as follows. The secret S is to be shared among the n shareholders identified by id_i, i = 1... n. The dealer who is the trusted party responsible for generating the secret and distributing it to the users performs the following steps:

1. A prime p is chosen such that p>max (S, n)

2. A polynomial is f (x) = $a_0+a_1x+a_2x^2+...+a_{k-1}x^{k-1}$ is generated where

$a_0=S$ (2.10)

and a_i i =1, 2.... k-1 is chosen randomly from Z_p .

3.The shares S_i i =1, 2…n are generated as S_i = f (id_i) (mod p) (2.11)

4. The shares are securely distributed to the respective shareholders

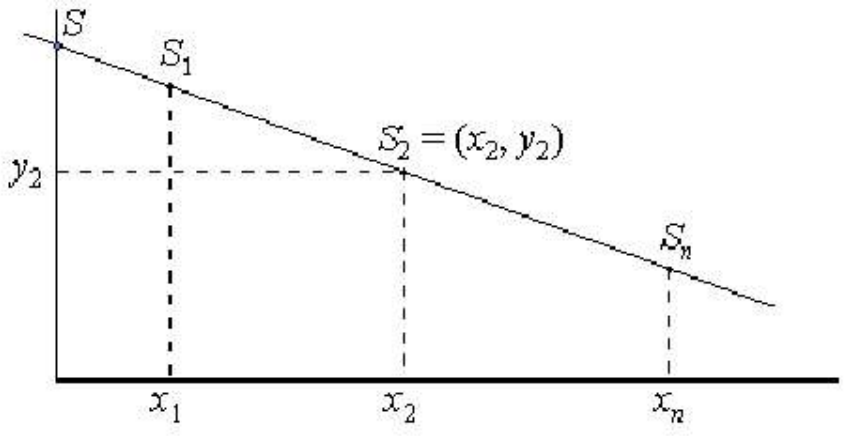

Figure 2.9 Shamir's Interpolating Scheme

To reconstruct the secret Lagrange interpolation is used. Within knowledge of a minimum of k shares the polynomial f (x) can be reconstructed and the secret recovered by calculating f (0). The Lagrange interpolation is described below:

$$f(x) = \sum_{i=1}^{k} S_i . I_{id_i}(x) (\mathrm{mod}\ p) \quad \text{where } \mathrm{I_{id}(x)} = \prod_{i=1\ i\neq j}^{k} {x - id_j}/{id_i - id_j} \tag{2.12}$$

Recovering the secret *S* from a set of shares $\{ S_1, S_2 \ldots S_K \}$ is equivalent to solving the following system of linear equations. Note that any subset of k out of the n values can be used.

$f(1) = S + a_1 1 + a_2 1^2 + \ldots + a_{k-1} 1^{k-1} = S_1$
$f(2) = S + a_1 2 + a_2 2^2 + \ldots + a_{k-1} 2^{k-1} = S_2$

▪
▪
▪
▪

$$F(k) = S + a_1 k + a_2 k^2 + \ldots + a_{k-1} k^{k-1} = S_K \quad (2.13)$$

Blakely's idea is based on Euclidean spaces. If the threshold scheme is (5, 3) for example the shared secret is in the regular three-dimensional space. The shared secret is a point in space. Each part or share is a plane containing the shared secret, or point. If enough number of planes exists, that is, we have at least k number of participants the shared secret is the point where all planes intersect. If some shares are missing we only get a plane or a line with the actual secret still undetermined. This can of course be generalized into any k-dimensional space [9].

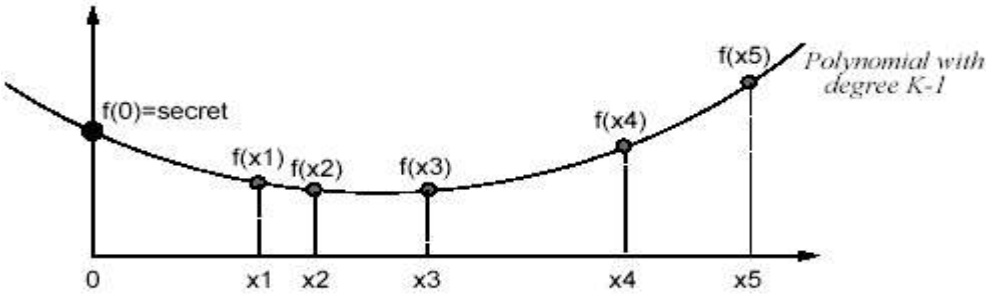

Figure 2.10 Lagrange Interpolation

It is important that no shareholder gains knowledge of any share other than his own. Otherwise he could potentially gain knowledge of k shares and then be able to reconstruct the secret himself. Therefore a trusted party is needed to perform the reconstruction of the secret, i.e. the shareholders provide their shares to the trusted party who performs the action requiring the secret, e. g the signing of certificates etc.

2.7.1.1 Advantages of Secret Sharing

- Size of each share ≤ size of the secret S.
- For a fixed k, S_i pieces can be dynamically added or deleted without affecting the other S_i pieces.

- Easy to change the S_i shares without changing the secret S (a new polynomial with the same free term).

- Allows for a hierarchical scheme in which the number of pieces needed to determine S depends on their importance.

- Provide tradeoff between security and reliability according to the choice of k and n.

2.7.1.2 Disadvantages of Secret Sharing

- An entity that computes the shares and distributes them needed: referred as dealer; the dealer is trusted.

- An entity that combines all the pieces in order to recover the key is needed.

- The combiner becomes a central point of trust and failure.

- Once the key is recovered an insider can potentially forge the group signature, or read messages individually.

- Need a system that provides decryption and signing without revealing the secret key.

2.8 Proactive Secret Sharing

In the secret sharing scheme described above the secret is protected by distributing it among several shareholders. In k+1 out of n threshold schemes, security is assured if throughout the entire lifetime of the secret the adversary compromises no more than k of the n location.

However, given sufficiently long time an attacker could compromise k shareholders and obtain their shares, thereby allowing him to reconstruct the secret. To defend against such attackers proactive secret sharing schemes update the shares on a regular basis. An

attacker must then compromise k shareholders between the updates since only k shares belonging to the same update period can be used to reconstruct the secret.

Proactive schemes are proposed as a countermeasure to mobile adversaries. A proactive threshold cryptography scheme uses share refreshing, which enables servers to compute new shares from old ones in collaboration without disclosing the service private key to any server. The new shares constitute a new (n,k) sharing of the service private key. After refreshing, servers remove the old shares and use the new ones to generate partial signatures. Because the new shares are independent of the old ones, the adversary cannot combine old shares with new shares to recover the private key of the service. Thus, the adversary is challenged to compromise k+1 server between periodic refreshing.

Share refreshing relies on the following homomorphism property.

If $(s_1^1,\ s_2^1 \ldots\ s_n^1)$ is a (n,k) sharing of S_1 and $(s_1^2, s_2^2 \ldots s_n^2)$ is a (n,k) sharing of S_2, then $(s_1^1,\ s_2^1 \ldots s_n^1)^{v}$ is a (n,k) sharing of S_1+S_2. If S_2 is 0, then we get a new (n, k) sharing of S_1.

Given n servers, Let $(s_1, s_2 \ldots s_n)$ be a (n,k) sharing of the private key S of the service, with server i having s_i. Assuming all servers are correct, share refreshing proceeds as follows first, each server randomly generates $s_{i1}, s_{i2} \ldots . s_{in}$ is an (n,k) sharing of 0. We call these newly generated s_{ij}'s sub shares. Then, every sub share s_{ij} is distributed to server j through a secure link. When server j gets the sub shares $s_{1j}, s_{2j} \ldots s_{nj}$ it can compute a new share from these sub shares and its old share

$$s'_j = s_i + \sum_{i=1}^{n} s_{ij} \qquad (2.14)$$

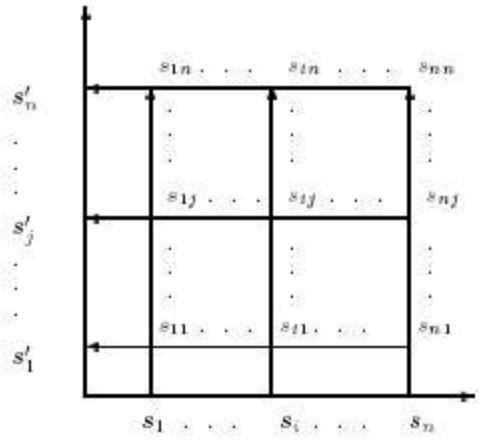

Figure 2.11 Proactive Share Refreshing

A variation of share refreshing also allows the key management service to change its configuration from (n,k) to (n',k'). This way, the key management service can adapt itself, on the fly, to changes in the network. If a compromised server is detected, the service should exclude the compromised server and refresh the exposed share; if a server is no longer available or if a new server is added, the service should change its configuration accordingly. For example, a key management service may start with the (7; 3) configuration. If, after some time, one server is detected to be compromised and another server is no longer available, then the service could change its setting to the (5; 2) configuration. If two new servers are added later, the service could change its configuration back to (7; 3) with the new set of servers.

The only difference is that now the original set of servers generate and distribute sub shares based on the new configuration of the service: for a set of k of the n old servers, each server i in this set computes an (n',k') sharing $(s_{i1}, s_{i2} \ldots s_{in})$ of its share s_i and distribute sub share s_{ij} secretly to the j_{th} server of the n new servers. Each new server can then compute the new share from these sub shares. These new shares will constitute a (n',k') sharing of the same service private key.

Note that share refreshing does not change the service key pair. Nodes in the network can still use the same service public key to verify signed certificates. This property makes share refreshing transparent to all nodes, hence scalable.

2.9 Verifiable Secret Sharing

If any shareholder wishes to prevent the reconstruction of the secret, he can provide an invalid share, e.g. a random value, to be used for the reconstruction. The Lagrange interpolation will then result in the reconstruction of a value, different from the secret S. Verifiable secret sharing mechanisms are used to prevent this type of denial of service attack. These schemes are based on hard to invert homomorphism functions and, in particular, on the hardness of computing discrete logarithms over Z_p, for prime p.

The mechanism works as follows:

Select f (x) over Z_p as in Shamir's

1. $f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{k-1} x^{k-1} \pmod p$ (2.15)

2. set up p, q (q divides p-1)

$$b \in Zp^*,\ g = b^{(p-1)/q} \pmod p \quad (2.16)$$

3. Witness generation (publicly known)

$$W_i = g^{a_i \pmod q} \pmod p \quad (2.17)$$

as witnesses of the coefficients of the sharing polynomial.

4. Secret share verification

Each node can then upon receiving its share verify it by checking that

$$g^{S_i} = g^{a_0}.(g^{a_1})^{id_i} \ldots\ldots\ldots (g^{a_{k-1}})^{id_i^{k-1}} \quad (2.18)$$

Such a scheme will allow multiple parties to renew proactively their shares, without changing the secret. This method limits the amount of damage if parties get compromised.

2.10 Threshold Signature

In the conventional digital signatures such as RSA [8] and ElGamal [2], a single signer is sufficient to sign a message and any verifier can verify the validity of the signature with the signer's public key. However, the responsibility of signing messages for many applications needs to be shared by a set of signers. It is a policy for some applications and occasions that at least k persons rather than one person generate cooperatively a signature. The multi-signature schemes, the (n,k) threshold signature schemes and the (n,k) threshold multi-signature schemes were proposed for slightly different concepts. In these schemes, it is required that several signers cooperate to generate a valid group signature for a message on behalf of the group.

Threshold signatures are digital signatures where signers can establish groups such that only certain subsets of the group can produce signatures in behalf of the group. The collection of subsets that are authorized to produce signatures is called the access structure of a threshold scheme. More particularly, a (n,k) threshold signature scheme is a digital signature scheme where any k or more signers of a group of n signers can produce in behalf of the group. In general, a threshold signature does not reveal the actual group members that have co-operated to produce it.

Multi signatures are threshold signatures with the additional feature that they reveal the identities of the group members who produced them. In multi signatures, the signing members are not anonymous at all. The special case of a (1, 1)-threshold signature scheme is an ordinary digital signature scheme.

The goal of a threshold signature scheme is to enforce dual control over the signing capability (choose k>1) or to eliminate single points of failure (choose n>1), or both.

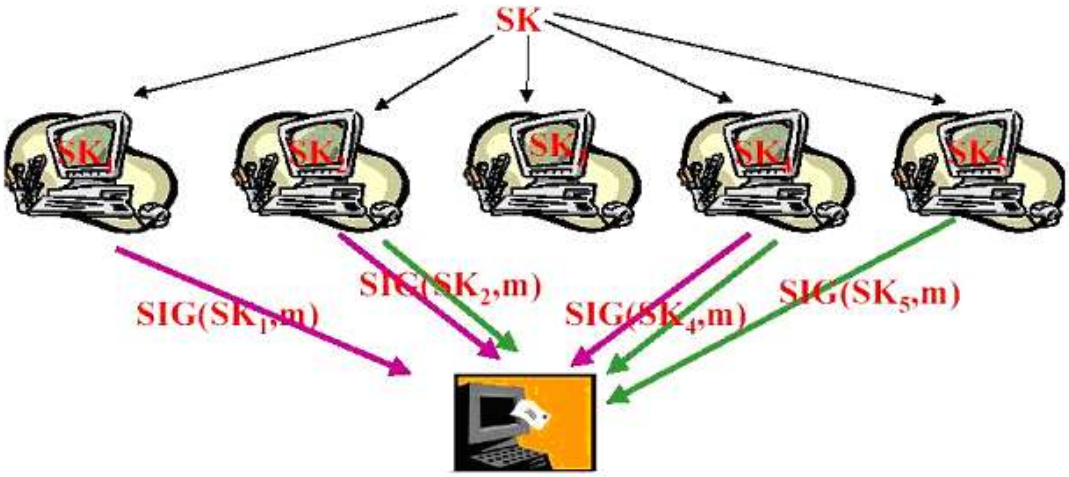

Figure 2.12 Threshold Signature

3. KEY MANAGEMENT

This chapter considers key management techniques for controlling the distribution, use, and update of cryptographic keys. Systems providing cryptographic services require techniques for initialization and key distribution as well as protocols to support on-line update of keying material, key backup/recovery, revocation, and for managing certificates in certificate-based systems. This chapter examines techniques related to these issues.

3.1 Key Management Objectives and Threats

Key management plays a fundamental role in cryptography as the basis for securing cryptographic techniques providing confidentiality, entity authentication, data origin authentication, data integrity, and digital signatures. The goal of a good cryptographic design is to reduce more complex problems to the proper management and safe-keeping of a small number of cryptographic keys, ultimately secured through trust in hardware or software by physical isolation or procedural controls. Reliance on physical and procedural security (e.g., secured rooms with isolated equipment), tamper-resistant hardware, and trust in a large number of individuals is minimized by concentrating trust in a small number of easily monitored, controlled, and trustworthy elements [8].

The purpose of key management is to:

1. Initialize system users within a domain.

2. Generate, distribute and install keying material.

3. Control the use of keying material.

4. Update, revoke and destroy keying material.

5. Store, backup/recover and archive keying material.

Keying relationships in a communications environment involve at least two parties (a sender and a receiver) in real-time. In a storage environment, there may be only a single party, which stores and retrieves data at distinct points in time.

The objective of key management is to maintain keying relationships and keying material in a manner, which counters relevant threats, such as:

• Compromise of the confidentiality of keys.

• Compromise of authenticity of secret or public keys. Authenticity requirements include knowledge or verifiability of the true identity of the party a key is shared or associated with.

• Unauthorized use of secret or public keys. Examples include using a key which is no longer valid, or for other than an intended purpose [8].

3.2 Security Policies and Key Management

Key management is usually provided within the context of a specific security policy. A security policy explicitly or implicitly defines the threats a system is intended to address. The policy may affect the stringency of cryptographic requirements, depending on the susceptibility of the environment in question to various types of attack. Security policies typically also specify:

1. Practices and procedures to be followed in carrying out technical and administrative aspects of key management, both automated and manual.

2. The responsibilities and accountability of each party involved.

3. The types of records (audit trail information) to be kept, to support subsequent reports or reviews of security-related events [8].

3.3 Simple Key Establishment Models

The following key distribution problem motivates more efficient key establishment models.

3.3.1 The n^2 Key Distribution Problem

In a system with n users involving symmetric-key techniques, if each pair of users may potentially need to communicate securely, then each pair must share a distinct secret key.

In this case, each party must have n - 1 secret keys; the overall number of keys in the system, which may need to be centrally backed up, is then n .(n-1)/2, or approximately n^2. As the size of a system increases, this number becomes unacceptably large [8].

3.3.2 Point-to-Point and Centralized Key Management

Point-to-point communications and centralized key management, using key distribution centers or key translation centers, are examples of simple key distribution (communications) models relevant to symmetric-key systems. Here “simple” implies involving at most one-third party. These are illustrated in Figure 3.1 and described below, where K_{XY} denotes a symmetric key shared by X and Y [8].

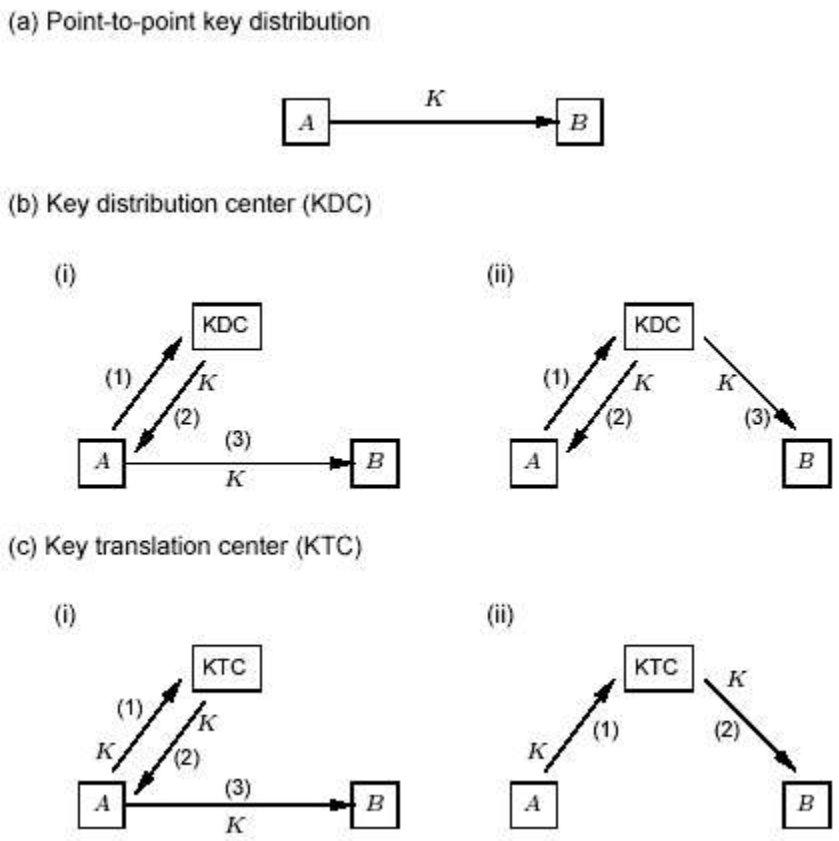

Figure 3.1 Simple key distribution model

1. Point-to-Point Mechanisms: These involve two parties communicating directly. This method is shown in Figure 3.1.

2. Key Distribution Centers (KDCs): KDCs are used to distribute keys between users, which share distinct keys with the KDC, but not with each other. A basic KDC protocol proceeds as follows.

    - Upon request from A to share a key with B, the KDC T generates a key K, then sends it encrypted under K_{AT} to A. This is shown in Figure 3.1.b.i.

    - KDC T also encrypts a copy of K (for B) under K_{BT}. Alternatively, T may communicate K (secured under K_{BT}) to B directly. This is shown in Figure 3.1.b.ii.

3. Key Translation Centers (KTCs): The assumptions and objectives of KTCs are as for KDCs above, but here one of the parties (e.g., A) supplies the session key rather than the trusted center. A basic KTC protocol proceeds as follows.

- A sends a key K to the KTC T encrypted under K_{AT}. This is shown in Figure 3.1.c.i.

- The KTC deciphers and re-enciphers K under K_{BT}, then returns this to A (to relay to B) or sends it to B directly. This is shown in Figure 3.1.c.ii.

KDC's provide centralized key generation, while KTC's allow distributed key generation. Both are centralized techniques in that they involve an on-line trusted server.

Point-to-point mechanisms require that A and B share a secret key a priori. Centralized key management involving a trusted party T requires that A and B each share a secret key with T. These shared long-term keys are initially established by non-cryptographic, out-of-band techniques providing confidentiality and authenticity (e.g., in person, or by trusted courier).

Centralized key management involving third parties (KDC's or KTC's) offers the advantage of key-storage efficiency: each party need maintain only one long-term secret key with the trusted third party (rather than one for each potential communications partner). Potential disadvantages include: vulnerability to loss of overall system security if the central node is compromised (providing an attractive target to adversaries); a performance bottleneck if the central node becomes overloaded; loss of service if the central node fails (a critical reliability point); and the requirement of an on-line trusted server.

3.4 Trusted Third Parties

The success of a communication system is limited by how much trust can be placed in the information being transmitted. For any two communicating entities, mutual trust must be established to support effective communication. Authentication is the fundamental basis

for building such trust. In large-scale dynamic networks, where communication may occur between any two entities, it is not feasible to expect every entity to have the capability to authenticate all other entities with which it might communicate. To this end, it is necessary to provide a framework for authentication. One of the traditional ways to support such authentication is to involve a commonly trusted third party. [8]

A trusted third party (TTP) is an entity trusted by all users of the system and is often used to provide the key management services mentioned above. Depending on the nature of their involvement they can be categorized as in-line, on-line or off-line.

3.4.1 Roles of Trusted Third Parties

Trusted Third Parties (TTP's) are classified based on their real-time interactions with other entities.

3.4.1.1 In-line TTP

An in-line TTP participates actively in-between the communication path of the two users. This is shown in Figure 3.2 [8].

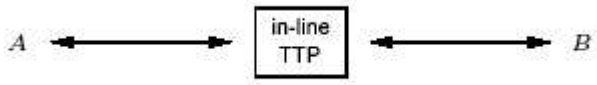

Figure 3.2 In-line TTP

3.4.1.2 On-line TTP

An on-line TTP is involved in real-time during each protocol instance (communicating with A or B or both), but A and B communicates directly rather than through TTP. An on-line TTP participates actively but only for management purposes, the actual communication between the users is direct. This is shown in Figure 3.3 [8].

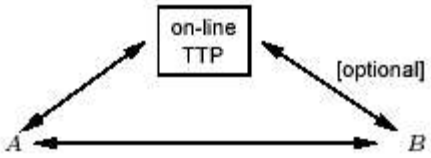

Figure 3.3 On-line TTP

3.4.1.3 Off-line TTP

An off-line TTP communicates with the users prior to their setting up a communication link. During the actual protocol run the off-line TTP is not active; in fact it does not even need to be connected to the network. Since every node in the network trusts the TTP, authentication provided by the TTP can be trusted with a high level of confidence, as long as the TTP remains secure and reliable. This is shown in Figure 3.4 [8].

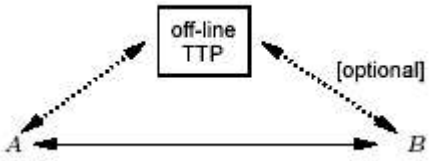

Figure 3.4 Off-line TTP

3.5 Certificate Authority

The Certification Authority (CA) is a trusted third party whose signature on the certificate vouches for the authenticity of the public key bound to the subject entity. The significance of this binding (e.g., what the key may be used for) must be provided by additional means, such as an attribute certificate or policy statement. Within the certificate, the string, which identifies the subject entity, must be a unique name within the system

(distinguished name), which the CA typically associates with a real-world entity. The CA requires its own signature key pair, the authentic public key of which is made available to each party upon registering as an authorized system user. This CA public key allows any system user, through certificate acquisition and verification, to transitively acquire trust in the authenticity of the public key in any certificate signed by that CA.

Certificates are a means for transferring trust, as opposed to establishing trust originally. The authenticity of the CA's public key may be originally provided by non-cryptographic means including personal acquisition, or through trusted couriers; authenticity is required, but not secrecy [8].

3.5.2 Digital Certificate

A digital certificate is a data structure consisting of a data part and a signature part. The data part contains clear text data including, as a minimum, a public key and a string identifying the party (subject entity) to be associated therewith. The signature part consists of the digital signature of a certification authority over the data part, thereby binding the subject entity's identity to the specified public key [8]. Also there is some additional information which the certificate data part might contain include:

1. A validity period of the public key
2. A serial number or key identifier identifying the certificate or key
3. Additional information about the subject entity (e.g., street or network address)
4. Additional information about the key (e.g., algorithm and intended use)
5. Quality measures related to the identification of the subject entity, the generation of the key pair, or other policy issues
6. Information facilitating verification of the signature (e.g., a signature algorithm identifier, and issuing CA's name)
7. The status of the public key.

All these fields in digital certificate are shown in Figure 3.5.

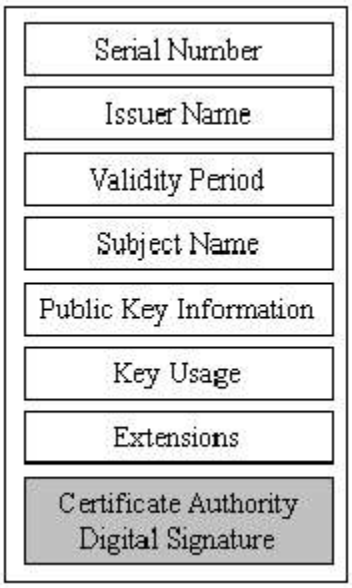

Figure 3.5 Digital Certificate Format.

The serial number is used to uniquely identify the certificate, and issuer name is the name of the trusted party who has issued the certificate. The validity field specifies how long the certificate is valid. The subject is the entity being identified by the certificate, i.e. the entity whose public key is being certified. The next two fields contain the public key being certified and information about what is certified will be to be used for (e.g. encryption, signatures etc.). The extensions field can be used to specify any additional information about the certificate. The signature field contains the certificates signature along with information about the hash algorithm used etc. [1].

If Alice wants to send a secret message to Bob, so she encrypts the message using Bobs public key pk_{Bob} that she has retrieved from a server. However the key that Alice retrieved actually belongs to an attacker. The secret message, which was intended for Bob, can now be decrypted and read by the attacker.

Digital certificates are used to prevent the type of attack described above. Basically a digital certificate is a statement issued by some trusted party saying that it verifies that the

public key pk_A in fact belongs to the user A. The trusted party digitally signs this statement and therefore anyone with the authentic public key of the trusted party can verify the certificate and thereafter use pk_A and be sufficiently sure that it actually belongs to node A.

3.6 Public Key Infrastructure

For any two communicating entities, mutual trust must be established to support effective communication. Authentication is the fundamental basis for building such trust. In large-scale dynamic networks, where communication may occur between any two entities, it is not feasible to expect every entity to have the capability to authenticate all other entities with which it might communicate. To this end, it is necessary to provide a framework for authentication [1].

The use of public key cryptography requires that the authenticity of the public keys can be established. A straightforward approach requires that any two users that wish to communicate must exchange their public keys in an authenticated manner and this would require the initial distribution of *n (n-1)* public keys. However, by having a trusted third party issue certificates to each of the users only the public key of the TTP needs to be distributed to each of the users. One of the traditional ways to support such authentication is to involve a commonly trusted third party.

A Public Key Infrastructure (PKI) provides the mechanisms needed to manage such certificates and consists of the components illustrated in Figure 3.6

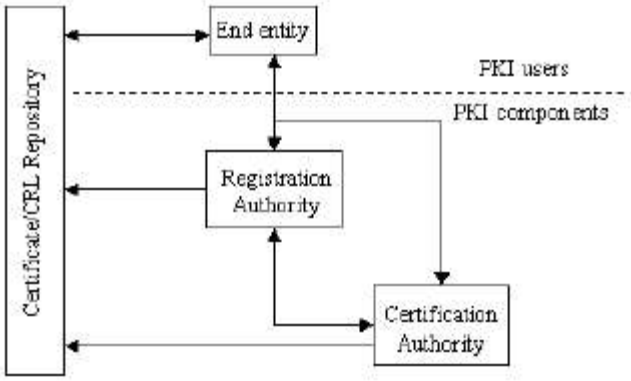

Figure 3.6 Main Components of a PKI

An end entity is either a user of a certificate or the subject to which a certificate has been issued. The certification authority (CA) is the component responsible for issuing and revoking certificates while the registration authority (RA) is responsible for establishing the identity of the subject of the certificate and the mapping between the subject and its public key. The CA can implement the registration function and therefore the RA is an optional component [1].

The following basic services should be provided by the PKI components described above:

1. Registration
2. Initialization
3. Certification
4. Key update
5. Revocation
6. Certificate and revocation notice distribution

Other services that may also be provided by the PKI include key recovery, key generation, and cross-certification, secure time stamping and non-repudiation.

The following subsections briefly describe each of the basic services mentioned above [1].

3.6.1 Registration

The registration service establishes the mapping between an end entity and its public key. This typically includes providing the public key to the Registration Authority (RA) along with any information that is required for the certificate, e.g. name, e-mail address, organization etc. The RA may also require that the end entity prove that it possesses the corresponding private key, e.g. by generating a digital signature. Next the RA needs to verify the information received from the end entity. Requiring that the end entity in person provide proof of identity such as a driver's license or ID card could e.g. do this. Finally when the RA has verified the identity of the end entity it contacts the CA and request the generation of the certificate [1].

3.6.2 Initialization

Before an end entity can use the services provided by the PKI it has to be initialized with certain information. The most important item required is the CA's certificate which contains the public key, which is needed to verify any certificates issued by the CA. Other information could be addresses of certificate repositories and other PKI components that the user may need to contact etc. The initialization also includes the generation of the end entities public/private key pair [1].

3.6.3 Certification

Upon receiving a certification request from the RA, the CA generates and signs the certificate. This process includes filling in the certificate form with the information provided by the RA and adding any additional information required, e.g. any extensions uses etc. The CA then digitally signs the certificate using its private key SK_{CA} [1].

3.6.4 Key Update

Key pairs are typically only valid for a limited time, ranging from days to years depending on the application. The key update service provides for the transition to a new key pair and the issuing of the corresponding certificate [1].

3.6.5 Revocation

The CA is responsible for maintaining the status of the certificates it has issued. E.g. if a certificate becomes invalid due to the compromise of the private key the CA needs to revoke the certificate. Certificates may also need to be revoked if e.g. any information in the certificate becomes invalid, e.g. subject name, organization etc. [1].

3.6.6 Certificate and Revocation Notice Distribution

After a certificate has been issued it needs to be made available to the owner and to other users that wish to use it. After the CA generates a certificate it can distribute it in a number of ways, e.g. by making it available on a publicly accessible server or by providing it to the certificate owner directly [1].

In case a certificate is revoked the PKI must provide a mechanism that informs certificate users of this. A common method used is that the CA on a regular basis publishes a certificate revocation list (CRL) that lists all the certificates that have been revoked. Certificate users can then use the CRL to check whether or not a certain certificate has been revoked [1].

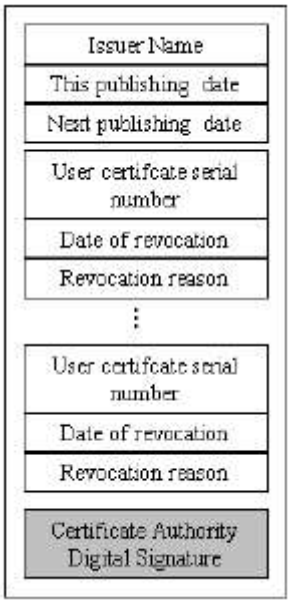

Figure 3.7 Example of CRL components.

A problem with CRL 's is that the time between the compromise and the notification of a certificate being revoked can be significant since CRL 's are only published at regular intervals. A different approach to revocation notification is the use of on-line revocation notification mechanisms. These allow the certificate users to query the CA in real-time for the status of a particular certificate. This method is called Online Certificate Status Protocol (OCSP) [8].

4. AD HOC NETWORKING

An Ad hoc network is a collection of nodes that do not need to rely on a predefined infrastructure to keep the network connected. Ad hoc networks can be formed, merged together or partitioned into separate networks on the fly, without necessarily relying on a fixed infrastructure to manage the operation. Nodes of Ad hoc networks are often mobile, which also implicates that they apply wireless communication to maintain the connectivity, in which case the networks are called as mobile Ad hoc networks (MANET). Mobility is not, however, a requirement for nodes in Ad hoc networks, in Ad hoc networks there may exists static and wired nodes, which may make use of services offered by fixed infrastructure [9].

In Ad hoc networks the communicating nodes do not necessarily rely on a fixed infrastructure, which sets new challenges for the necessary security architecture they apply. In addition, as Ad hoc networks are often designed for specific environments and may have to operate with full availability even in difficult conditions, security solutions applied in more traditional networks may not directly be suitable for protecting them [9].

4.1 Introduction

Mobile Ad hoc network is a collection of wireless mobile nodes dynamically forming a temporary network without the use of any existing network.

Due to the limited transmission range of wireless network interfaces, multiple hops may be needed for one node to exchange data with another across the network

Ad hoc networking is a networking paradigm for mobile, self-organizing networks. Mobile Ad hoc networking offers convenient infrastructure less communication over the shared wireless channel. Typically the network nodes are interconnected through wireless

interfaces and unlike traditional networks lack specialized nodes, i.e. routers that handle packet forwarding. Instead every node in the network functions as a router as well as an application node and forwards packets on behalf of other nodes. There are a lot of differences between mobile Ad hoc networks and traditional networks. Ad hoc networks do not rely on any fixed infrastructure. It relies on each other to keep the network connected. Also, the topology of Ad hoc networks is dynamically changing and its communication is based on wireless links. Due to the above characteristics, the main challenge in the design of mobile Ad hoc networks is their vulnerability to security attacks [5].

4.2 Area of Ad hoc Networks Applications

A lot of applications may be thought of using Ad hoc networking. It is especially useful in areas where the current fixed, infrastructure based solutions seem inflexible and over-scaled. The feasible applications may also include issues where Ad hoc networking technology could supersede the fixed architecture but there will be primarily cooperation instead of competition since the mobile and dynamic Ad hoc architecture has huge disadvantages - that are security, bandwidth and scalability

4.2.1 Military

The major challenges in the military field are that reliability, performance, security etc. are even more critical than in civilian applications. A lot of publications in the field of Ad hoc networking are actually published by military organizations.

Probably one of the hardest, largest and first applications is utilized in military environments. The first application of Ad hoc networking was in the military domain. Ad hoc networking enables battlefield units to communicate anywhere and anytime, without the requirement of any fixed infrastructure. The fact that every node forwards packets also provides for a robust network. The loss of any one unit will not disrupt the network since there will (hopefully) be other units that can still [10].

4.2.2 Emergency Services

Another huge public application may be in the area of emergency services. Firefighters, Police and others sometimes have to operate in areas where no information infrastructure is present and operations still need to be coordinated. Another scenario would be a situation after a great natural disaster, atomic reactor melt down or other catastrophes where an existing infrastructure is destroyed. Mobile units could help to build up emergency information structures [5].

4.2.3 Multimedia

Modern mobile telecommunication devices have become real wonders of miniaturization. Almost any mobile phone has got an integrated camera, microphone, MP3 player etc. Other multimedia applications will surely be added in the recent future. With more and more advanced cameras, movies, pictures etc. have to be transmitted to other devices - most likely PCs - to be able to view and edit movies on a more comfortable screen or upload your MP3s from the PC to the phone. This may (and partially is presently done) be done using Ad hoc technology [15].

4.2.4 Conferences

There is usually no fixed office network infrastructure when meeting with business or research partners. To be able to exchange data and support cooperative work there is a need for high-bandwidth networking technology that supports spontaneously created connections [15].

4.2.3 Personal Area Networks

Personal Area Networks (PAN) is used in an even smaller range than wide area networks (WLAN) s. They are intended to operate in a personal environment, connecting mobile phones with each other, or with PDAs, Laptops etc. There are a lot of scenarios of using

these networks, alongside with other WLAN based solutions. The IEEE tries to establish a standard for PANs. [15].

Another example could be when a person holding a PDA comes within communication range of a printer. If both the PDA and the printer were Ad hoc enabled the PDA could automatically get access to the printing services.

4.2.4 Home and Business

Throughout homes and businesses there is a huge amount of possible applications of Ad hoc technologies. There is not really the need to replace existing infrastructures - actually this could be quite a bad idea, given the disadvantages of Ad hoc technology concerning bandwidth and security - but to improve existing solutions by adding Ad hoc flexibility [15] .

4.2.5 Automation

Automation could definitely profit from Ad hoc technology. A lot of scenarios are possible where Ad hoc technology would help. Service robots will become more and more mobile in the next years, needing more advanced mobile communications techniques to even more increasing their mobility. These technologies shall help coordinating their work, increasing efficiency and making them independent of a fixed communications environment [15].

4.2.6 Sensor Networks

Sensor networks are Ad hoc networks consisting of communication enabled sensor nodes. Each such node contains one or more sensors, e.g. movement-, chemical- or heat sensors. When a sensor is activated it relays the obtained information trough the Ad hoc network to some central processing node where further analysis and actions can be performed. Such sensor networks may consist of hundreds or thousands of sensors and can be used in both military and non-military applications, e.g. surveillance, environmental monitoring etc. [1].

Sensor networks differ significantly from the other types of Ad hoc networks described in this section. The most significant difference is the small size, extremely limited power resources and processing power of the sensor nodes [1].

4.2.7 Collaborative Networking

This application of Ad hoc networking may be the most intuitive. The simplest example is when a group of people is attending a meeting and need to share information between their laptops or Personal Data Assistant (PDA)'s. If these devices were Ad hoc enabled they could dynamically set up a network consisting of the meeting participants and thus enable the sharing of the information. Without Ad hoc networking, a great deal of configuration and setup would be required to accomplish this task [1].

4.3 Characteristics

When designing protocols for Ad hoc networks, whether it is routing protocols or security protocols it is important to consider the characteristics of the network.

4.3.1 Dynamic Network Topology

The network nodes are mobile and thus the topology of the network may change frequently. Nodes are free to move arbitrarily. Thus, the network topology is typically multi-hop, so may change randomly and rapidly at unpredictable times Nodes may move around within the network but the network can also be partitioned into multiple smaller networks or be merged with other networks [1].

4.3.2 Inefficiencies and Bandwidth-Limitations

The use of wireless communication typically implies a lower bandwidth than that of traditional networks. This may limit the number and size of the messages sent during protocol execution [1].

There are matters to cope with in wireless communications that you do not have to deal with using hard-wired technology. Cables are shielded. Wireless communication links all share the same media (primarily air). This implies problems when two networks are very close to each other. Each network creates traffic. This means interferences to another network. These Interferences reduce the efficiency of the affected network [15].

4.3.3 Energy Constrained Nodes

Most mobile devices are battery powered. Routing calculations and sending control and data packages through the network drains energy. It is important to keep energy consumption as low as possible without increasing latency too much. Another aspect is that an energy source is often shared by multiple applications. If we take a laptop as an example, the battery provides among other things to the processor, the storage devices etc. and network adapter. It would be totally unacceptable if a network adapter were consuming lots of the battery power and therefore significantly reducing the overall working time of the entire device. The factor energy will become even more important in the future since the growing cycles of battery lifetime cannot keep up with the growth rates of bandwidth and processing power [15].

4.3.4 Limited Physical Security

The use of wireless communication and the exposure of the network nodes increase the possibility of attacks against the network. Due to the mobility of the nodes the risk of them being physically compromised by theft, loss or other means will probably be bigger than for traditional network nodes [1].

Wireless communication is much more vulnerable to security issues (denial-of-service attacks, bugging etc.) than hard-wired connections since the physical media cannot be protected from foreign access [15].

4.3.5 Network Origin

This aspect effects what assumptions or prerequisites that can be made on the nodes in the network. E.g. if it is a planned Ad hoc network, it can perhaps be assumed that the nodes can be supplied with some initial data structures such as certificates, passwords, user names etc. However, if the network is spontaneous no such assumptions can be made [1].

4.3.6 Network Range

Depending on the actual physical topology, Ad hoc network can be formed localized or distributed manner. In localized mode the network nodes are within physical range, e.g. the same room [1].

In distributed mode, the nodes are distributed over a large area without the possibility of physically interacting. Depending on the actual physical topology and distribution of the nodes in the Ad hoc network, certain techniques may not be applicable. The most obvious situation is a technique that requires physical interaction. Such a technique obviously cannot be used in a highly distributed Ad hoc network [1].

4.3.7 Network Capabilities

Depending on the nodes capabilities Ad hoc network can be formed uniform or, diverse.

In uniform mode all nodes have approximately the same capabilities in terms of power source, CPU performance and memory size etc.

In diverse mode the nodes' capabilities differ significantly, certain nodes may be high-end computers while other are embedded devices.

If the capabilities of the nodes in the network are diverse, certain techniques may not be directly applicable. A certain technique may be applicable to a subset of the nodes but completely unusable on the rest of the nodes. An example of this could be the use of public key cryptography; although this is not an issue for high-end CPU's it may not be feasible for embedded devices [1].

4.3.8 Network Transience

Depending on the longevity of Ad hoc network it may be constructed as short term or, long term.

In short termed mode, nodes come together once and create an Ad hoc network, when finished no knowledge is kept about the other nodes. These networks typically only persist during a relatively short time period, i.e. less than a few hours [1].

In long termed mode, the same nodes will probably be part of the same Ad hoc network multiple times and therefore save information about the other nodes for future use. These networks will persist during a longer time period. This also includes Ad hoc networks that may only persist during a short time period, but that instead are created frequently [1].

The longevity of the Ad hoc network may influence the allowed complexity of some initialization phase. E.g. for a network consisting of nodes that will frequently join in an Ad hoc network it may be tolerable with a more complex initialization phase than that of a network that will only last for a short time and will not reoccur [1].

4.4 Routing

The routing of data packets in computer networks is the process of moving information from a source to a given destination. The way to do this in the best way possible is the concern of the routing algorithm used. In the case of wired networks the main routing algorithms used are either distance vector routing or link state routing [16].

4.4.1 General Routing Principles

Routing has been going on in large networks, as the Internet, for quite some time and lots of different routing algorithms has been proposed and tested. The main categories that have survived and have been in large use are those described below.

These are performing well in wired networks, but some problems still exists [10].

4.4.1.1 Distance Vector Routing

A distance vector routing algorithm is a distributed algorithm that is quite simple to implement and has low computational demands. It basically means that each router has a way of knowing about its neighbors. The router knows the metric of each link to those neighbors. The metric can be of any type, for example delay or similar [10].

Also, each router has a distance vector, that is, a table of distances, to each destination available and which outgoing link to use. All the neighboring routers exchange information between each other based on what they know about their neighbors and what they told them. Each time a router receives an update from a neighboring router it updates its own vector with the minimum distances. If some values are changed the router in turn, sends out an update to all its neighbors [10].

Distance vector routing is commonly known to have problems with changing topology. Especially a problem known as the count-to-infinity problem which makes the routers construct an ever growing path to a node that has gone down or a link to that node being broken. The algorithm can also converge slowly on changing topology, and while converging, create routing loops [10].

4.4.1.2 Link State Routing

Instead of depending on every neighbor to gradually give information about how to get to all the destinations, the link state routing algorithm gets a complete picture of the entire network and locally calculates the shortest path to every destination [10].

This means that for each router to get information about every link all the routers must broadcast (flood) the information that they have to all others [10].

When all the information needed is collected the process of finding the shortest path can begin. This is accomplished using Dijkstra's or Prim's algorithm. To be able to compute the shortest path all nodes need to have a complete view of the network and then performs the computations locally. This makes these kinds of algorithm global routing

algorithms in comparison to the distributed or decentralized distance vector algorithms [10].

As has been stated this kind of algorithm demands a complete view over the network, which makes the number of messages, needed to be sent between all routers relatively high. Also, the computation of the shortest path problems using Dijkstra's algorithm is in the order of $O(n^2)$ [10].

4.4.2 Routing in Ad hoc networks

In the case of a mobile Ad hoc network the topology is highly dynamic. This leads to quickly changing link states. Some links get broken while other pairs create other links [10]. In Figure 4.1 the mobile host 1 (MH_1) is moving from the vicinity of MH_2. As it gets closer to MH_7 and MH_8 new links are established to these hosts. These characteristics are different from the one that appears in most wired networks. The routing algorithms used in the wired case have problems with topology changes, and if these happen often the problems is just getting worse [16].

Another problem that arises in wireless networks that is not as common in wired routing is the asymmetrical links. That is, one node can reach another but the return path is not the same.

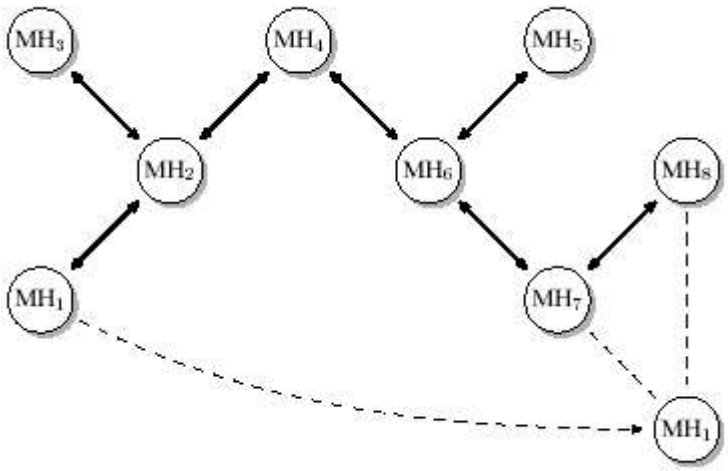

Figure 4.1 An Ad hoc network of mobile nodes [16].

4.4.2.1 Destination-Sequenced Distance Vector Protocol

Destination-Sequenced Distance Vector (DSDV) is an Ad hoc version of the commonly known distance vector algorithm. To overcome the problems with slow convergence in the ordinary distance vector algorithm and prevent routing loops in highly dynamic topologies this algorithm adds a sequence number to the routing table entries [16].

The destination nodes update the sequence numbers when new links to them are detected. Also, when a node detects a broken link it sends this out together with an updated sequence number. The receiving nodes check for higher or equal sequence numbers. If a route update packet is received with lower sequence number it is discarded. In the case of a sequence number that is equal to the one already held the metric is checked to see if it is better or worse [16].

To save bandwidth the protocol uses two types of route update messages. First one is a full dump that a node sends to all its neighbors. These messages contain the complete routing table. The other variant is an incremental update, which only updates the routes that has changed since the last full update. In this way it is possible to send only small packets, conserving bandwidth and transmission time for most cases. When these

incremental updates are getting too big the node can send a full dump instead in hope to be able to send smaller packets after that [16].

To get rid of the possibility of an oscillating system update messages are only sent out after a delay. After this delay the routing information has stabilized and is not that sensitive to oscillation. The delay is computed using a running, weighted average over the most recent updates [12].

Since the topology might change the route update messages are sent out at certain time intervals. However, there is no need for synchronizing the different nodes as the update events are handled asynchronous. The use of these repeating update messages keeps all the nodes busy when not in need for communication. However, when the needs arrive all nodes are ready to forward the data directly. The DSDV algorithm gets rid of the undesirable properties that the original algorithm possesses. It propagates the bad news of broken links fast and keeps the path updates stable. Also, using the sequence number rules for updating distance vector values it guarantees loop-free paths to each destinations at all times [10].

4.4.2.2 Dynamic Source Routing

The Dynamic Source Routing (DSR) protocol is completely demand based. It does not need any kind of periodic updates or node announcement messages. Instead, the protocol acquires the needed routing information on demand [13].

The routing protocol is divided into two parts. The first is the route discovery and the second is route maintenance. The discovery phase is initiated when a node needs to send information to another node that is not available in its current path cache. The node broadcasts a special discovery packet with the destination and a unique identification number. All nodes within the wireless transmission range receive the packet. They, if they are not the destination, add their node address to the path in the packet header and retransmit the discovery packet. A packet with the same identity as has already been seen is discarded. Also, if the node itself is mentioned in the path header the packet is

discarded. This technique efficiently cuts down on duplicate packets in the air. This is shown in Figure 4.2

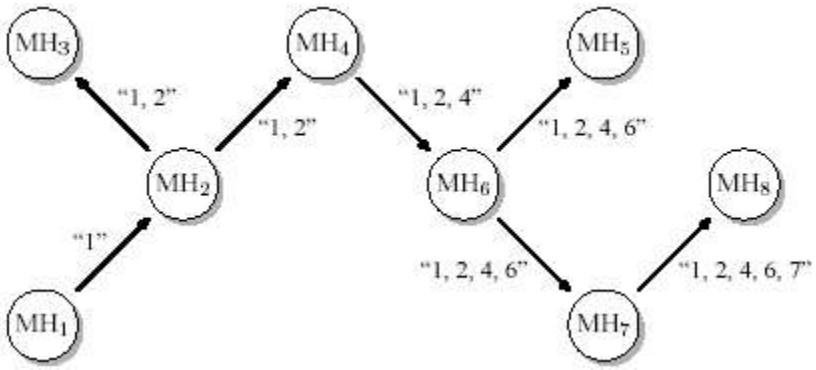

Figure 4.2 DSR route discovery examples with mobile host 1 as the initiator and 8 as the target.

When the packet finally finds its way to the destination, the destination node returns a route reply. The reply is sent using the routing cache if present. Otherwise a new route discovery is initiated but in this case with the route reply piggy backed on the discovery packet. If this piggy backing is not allowed there is a great risk of infinite looping of route discoveries. Another way is to reverse the source path collected by the route discovery; however, this takes for granted that all links are bi-directional which may not always be the case [12].

After a path has been discovered the second phase of the protocol is in use. This is the maintenance phase. During this phase all communications are done using the previously found paths. Each node on the path is responsible for resending packets that are not acknowledged by the next hop node. After a maximum limit number of retries a route error message is sent back to the source node indicating that the path is broken [12].

A number of different optimizations of the protocol are possible. For example, each node on a path that is not actually originating a route discovery can cache some part of the messages that they see go by. This can speed up route discovery but may also place some overhead on the software and the processing time of the processor [13].

4.4.2.3 Ad hoc On-Demand Distance Vector Routing

In contrast to the DSDV, the Ad hoc On Demand Distance Vector (AODV) is an on demand protocol. It borrows the idea of route discovery and maintenance from DSR while still using a distance vector approach to the routing [11].

Each node keeps a distance vector for each known destination and its next hop neighbors. If a destination that is needed is not in the list a route discovery is initiated. By issuing a broadcast packet to its neighbors containing source and destination addresses and a sequence number the node hopes to find a path to the destination.

If a node, that is not the actual destination, receives a request it checks its own routing table. If it can find the destination there it replies to the source node with this information. If, however, this information is not in the node's routing table it adds the source as a destination in its own table using appropriate hop count, increments the request packet's hop count and rebroadcasts it to its neighbors. This is shown is Figure 4.3 [13].

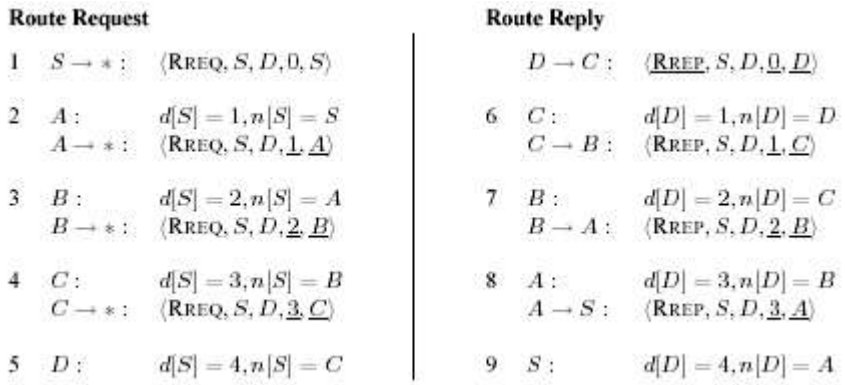

Figure 4.3 Simplified route discovery example in AODV. Host S is the initiator of the route request (RREQ) and the destination is host D.

This procedure is continued until the destination is reached and a route reply packet is sent back to the source. After this all data traffic is routed using the discovered path. If the nodes move and some links break a route error packet is sent out to tell the nodes that the path is no longer available. If this happens the source node initiates a new route request [12].

To eliminate the need of flooding a large network with route request packages the time to live field is used to limit the number of hops a route request can be sent. If a request fails this is gradually incremented until the destination is reached. This approach is called expanding ring search [13].

The AODV protocol can only handle symmetrical links but has the ability to also be used in multi cast groups. Additionally it supports a hello message, a variant of the route request message, which is used to detect the one-hop neighbors. This may not be used if the lower layers already support this kind of service [13].

4.4.2.4 Cluster-Based Networks

In some situations the network may be small, that is, consist of a small number of nodes. In this case the above-mentioned routing algorithms may be adequate.

However, if the network grows and the number of nodes are large there may be unnecessary overhead and the network diameter, the number of hops, may become too large to handle straight on [12].

In the case of wired networks area controllers are used to manage a limited set of computers and routing between these are done using some backbone infrastructure [12].

This makes the network manageable for each smaller cluster and the network diameter might become smaller using the backbone routers. However, administrative personnel do the setup of these area controllers often during the design of the network. This is clearly not something that can be done as easily in mobile Ad hoc networks. There are some ideas and techniques available that manages to accomplish these goals automatically using distributed algorithms and elections of cluster heads.

The idea of using clusters is beneficial in many ways. It can be used to group nodes into clusters that are separated to nearby clusters by transmission frequency or spread spectrum coding. This way the congestion can be lowered [12].

4.4.3 Secure Routing

The Secure Routing Protocol (SRP) proposed by Papadimitriou and Haas [12] is mainly focused on the route discovery process. The routing protocols mentioned in earlier section were not concerned at all with security and there are many attacks available to disrupt the network connectivity by targeting the routing protocol.

The SRP uses symmetric cryptography, which is efficient for message authentication codes between the peer nodes in a communication channel. The intermediate nodes need not do anything but pass on the routing request and data traffic.

This does, however, imply the availability of shared secrets between every two nodes that are to communicate with each other. The problem of key distribution is not addressed in [12] other than using key negotiation algorithms such as for example the Elliptic Curve Diffie-Hellman algorithm.

SRP can be added to a number of different routing protocols quite easily but in [12] Papadimitriou and Haas describes the setting using a DSR like protocol as base. The basic assumption is the knowledge of a shared key that can be used initially for authentication of the peer nodes and the correct routing information.

Later on during the actual information exchange the keys can be used for integrity and privacy of the data, thus creating a secure channel.

Many reactive routing protocols, including DSR, DSDV, and AODV uses monotonically increasing sequence numbers to identify requests and suppress duplicate packages flooding the network. The use of these sequence numbers enables attacks such as the rushing attack. The need for this ability is great and even SRP uses it. In addition a random request identifier is also added to the message. This way the rushing attack has a low probability of succeeding since the identifier used are hard to foresee. The use of this random number does involve the use of a secure pseudo random number generator in each node initiating a route request.

When the source node creates the route request message the non-mutable fields of the IP packet, the source address, the sequence number, the random identifier, and the shared key are all input to a hashing algorithm. The hash is also part of the SRP header. The request is flooded out, as is done in most reactive protocols, to all neighbors. Each neighbor adds itself to the list of nodes on the path and forwards the message. There is no need for each intermediate node to do any cryptographic or otherwise computationally expensive operation. When the target node receives the route request it verifies that the request indeed comes from the correct source, a source with which it shares a secret key. It then constructs a route reply and creates the MAC using the complete path information as input to the hash algorithm. It then sends the route reply back in the reverse path it came on. One-strength of Ad hoc networks is the redundancy, which is taken advantage of in SRP. The target will receive multiple route requests from different routes. It can reply to each of these giving the source node the ability to choose one of the many alternatives.

The target will have to limit the number of replies so that a malicious node cannot consume too much bandwidth or resources by forwarding spoofed route requests.

When the source node receives the route replies it can validate the correctness of the route by verifying the MAC. Since the reply was routed using the path information in the reply packet the source node can be certain that the route was actually followed and correct. So far so good, to ensure communication with the target the source can choose one or more of the routes that did get back. Even if an adversary drops all data traffic on a given route another route can be chosen.

This works as long as there are enough benign nodes in the network. To further take advantage of the redundancy the ability to use diversity coding can make the communications even more reliable. In diversity coding the techniques of error correcting codes and redundant information is used. The actual data is split up and sent along multiple different routes. When the target receives the pieces it can combine them to the complete message even though not all of the pieces arrived correctly or at all [19].

5. SECURITY IN AD HOC NETWORKS

Ad hoc networking has been a field of very active research in recent years. However, most of the research has been focused around various protocols for multi-hop routing, leaving the area of security most unexplored.

5.1 Needs for Security

The actual needed security will depend on the type of application and the value of the information that travels in the network. This is always a trade-off that must be considered for any type of security system, Is it worth the money to build or implement a secure system when there is no valuable information flowing around. For how long must the information be safe forever or is it just tactical real time information that is worthless in a few hours or so. These question and many more are needed to be considered when making a secure system [17].

Some ideas of the need for security in different types of applications are given below, ranging from low to extremely high security needs. What must be added is the need for making the systems practically usable, which often leads to in the form of human beings [17].

5.1.1 Home Networking

For a home network connecting the TV, stereo, VCR, DVD player etc. to a remote control the need for security might not be that high. Of course, the integrity aware people might not like the idea of neighbors being able to listen in on the control traffic and conclude what shows are being recorded and so on. However, these are more of a comfort

issue than a real security threat. Also, it should not be possible for a bypassing individual that bought a remote control of the same brand to control the appliances.

On the other hand, if the network is expanded to include control over the burglar alarm and carport some additional security measures must be taken. With such addition the value of the protected information, the house and all the appliances included, increases and so does the need for keeping the network safe [15].

5.1.2 Emergency Services

In the case of emergency services using an Ad hoc network for their communication the main need is the availability. For disaster areas there is probably no real need for securing the actual traffic since the most important thing is to be able to direct and control the rescue teams.

In the case of terrorist attacks the network should also protect the integrity and be safe against radio jamming since these attacks otherwise may lead to even worse situations. To go further along these lines, for a police squad trying to get control of a hostage situation the need for them to hide their communication from the kidnapper is vital. On the other hand, some communications might also need to be kept from the journalists until certain investigations are over [18].

5.1.3 Military Applications

The highest need for security is probably within the military tactics. An Ad hoc network in the battlefield can be very valuable to the enemy if they can get hold of a node. This is worth a lot and therefore the attacker might not stop at any costly attack method. This implies that the security of the network must be high enough that even the most advanced and expensive attack will not jeopardize the security of the network [1].

5.1.4 Physical Security

The need to secure information has mostly been addressed in the context of traffic flow. In the case of small mobile devices the aspect of physical security is even higher. A

stationary node can be locked in using locks and chains but this will not be a valid solution for a mobile device. This poses some extra security related problems on Ad hoc nodes. Since they cannot be physically protected in the same way as other equipment solutions based on tamper proofed chips are proposed [1].

Such solutions can provide for some safety of the data even in the case of theft of the device.

5.2 Key Management in Ad hoc Networks

Any successful key management framework for Ad hoc networks requires fault tolerance, security, and availability. These terms are sometimes used interchangeably, mainly because they are not independent of each other. To avoid confusion, we clearly define these terms and apply them to some existing approaches for evaluation.

5.2.1 Fault Tolerance

The main concern of fault tolerance is the capability to maintain correct operation in the presence of faulty nodes. We restrict the definition of faulty nodes to observable faults. If a node is malfunctioning and other nodes can observe such malfunctions, a certain level of recovery is possible. We employ intelligent replication using threshold cryptography to provide tolerance of faulty nodes [23].

5.2.2 Security

Acting as the trust anchor for the whole network, framework should be secure against malicious nodes or adversaries. While it may not be possible to be resistant to all levels of attacks, there should be a clear threshold of attacks a system can withstand while operating normally [23].

5.2.3 Availability

Traditionally, the term availability has been used in conjunction with fault tolerance. But in Ad hoc networks availability is also highly dependent on the connectivity of the network. In wired networks, if there are no faulty or compromised nodes, the system is by definition available for clients since connectivity is not a problem. In Ad hoc networks, even when there are no faulty or compromised nodes, clients may not be able to contact the desired services due to inconsistent connectivity [23].

The simplest approach to providing Certificate Authority (CA) functionality in an Ad hoc network is to assign a single node to be the CA. The success of this scheme depends on that single CA node. This approach is not fault tolerant, since failure of one node breaks the system. Similarly this approach is highly vulnerable, since an adversary needs only compromise one node to acquire the secret key. Finally, given the expected mobility and unpredictability of Ad hoc networks, it may be possible that nodes will not be able to reach the CA in a timely fashion, making availability very unpredictable [23].

Therefore, a single CA cannot effectively service a whole Ad hoc network.

Robustness, which is missing in the single CA scheme, can be achieved by replicating a fully functional CA on *r* different nodes. With *r* replicas, the system can withstand *(r - 1)* failures because the CA service is available as long as there is at least one operational CA.

Availability has also been improved since a client node has a better chance of reaching one of *r* CA's to get service. Unfortunately, the system has become more vulnerable. An adversary needs only compromise one of the *r CA* nodes to acquire the secret key and so compromise the whole system. Therefore, using replicated CA's is not a viable solution in Ad hoc networks. The problem of using replicated CA's stems from the fact that each replica has full knowledge of the system secret.

Zhou and Haas first proposed to use threshold cryptography to securely distribute the CA's private key over multiple nodes to form a collective CA service [4]. Using *k* out of *n* secret sharing can provide a good level of fault tolerance and security. They address the

problem clearly and present conceptual design issues, but do not address the problem of availability in their work. Again, connectivity between clients and the distributed CA's was not a concern.

Kong and others address availability by making all *M* nodes in the network share CA functionality [24]. A client need only contact *k* out of *M* nodes to get a certification service. Assuming there are *k* nodes in a client's one hop neighborhood, the client can get a certification service cheaply by using a one-hop broadcast for the request. While this solution addresses availability and fault tolerance, it compromises the security of the system. In general, the gap between *k* and *n* in secret sharing schemes defines the security of the system. *k* can be chosen between 1 and *n* in any secret sharing scheme. As *k* approaches *n*, thus closing the gap between *k* and *n*, the system becomes more secure because an adversary needs to compromise at least *k* nodes to collapse the system. But if k is too large, the system becomes less available to clients and also less tolerant to faults. When *k* approaches 1, making the gap larger, the effect is reversed and the system becomes more available but also less secure. Kong chose to keep *k* relatively small to address the availability problem and ended up with a vulnerable system where any adversary need only compromise a small number of nodes in the network to collapse the service.

Hubaux and others propose another notable scheme [12]. In their scheme, there is no concept of a CA. Every node acts as its own CA, similar to the PGP "Web of Trust" model. The main difference between PGP and their scheme is that there is no longer a well-known certificate directory where all certificates are stored. Rather, every user in the system carries a part of the certificate directory. In PGP, when two users wish to authenticate each other, they must search the certificate directory for a chain of certificates that links both users. In Hub ax's scheme, this problem is transformed into finding an intersecting point between the certificate chains carried by each user. Hubaux proposed a shortcut-hunter algorithm for this problem. While their approach is practical for the totally self-organizing networks they aimed at, it has the inherent problem of no definite trust anchor like the CA in other CA-based PKI approaches [12].

Therefore, it is not meaningful to evaluate this scheme using our framework.

5.3 Distributed Key Management

As noted many times before is the problem of usage of on line trusted parties, key servers, and CA's. To address this specific problem it has been suggested that the CA's are replicated and therefore available to a larger extent. Even though it is more available it is also more vulnerable. If an adversary can compromise one CA all the secrets kept in that, and all the others, are also compromised thus weakening the security in the system at large.

One solution is that of threshold cryptography that was introduced by Zhou and Haas [4] introduce a distributed key management service based on this. The service at large has a public key known to all nodes in the system. The nodes trust certificates signed by the service's private key. The nodes can query the service in the system to get the other nodes' public keys. The nodes can also submit update requests to update their own public keys.

The service consists of n special nodes in a threshold (n, k+1) configuration where

$n \geq 3k + 1$. These nodes are the servers of the service. The servers themselves also have public keys to the other servers and thus are able to construct secure channels between themselves. The threshold scheme allows for an adversary to compromise up to k servers in any period of time. However, the assumption is that the adversary lacks the computational power to break the cryptographic schemes that are used [20].

In this key management service the servers share the ability to sign certificates. When the service is requested each available server partially signs the certificate.

Any combiner node to retrieve the complete signature combines these partial signatures. In the case of compromised servers the combiner can check the validity using the public key of the service and if it is not valid the combiner node tries another set of k+ 1 signatures. The combiner itself need not know any secret and can thus be any node in the network.

The actual threshold scheme used can be any that fulfils the requirements set up. To be more robust the scheme employs share refreshing and allows for automatic adaptation of the configuration. Thus when non-compromised servers identify compromised one they can change the threshold scheme to exclude them.

The share refreshment allows the servers to compute new shares from the old one that they already hold. In addition, since the new shares are independent of the old shares a compromised server cannot use the old shares in any combination with the new shares to sign a certificate. They're by forcing the adversary to compromise another set of k + 1 server between each update [20].

The usage of threshold cryptography is very promising and fits nicely into the ideas of non-centralized services. However, there are large demands on signature validation, which might not fit small nodes with limited computational power that well [23].

5. 4 Self-Organized Public Key Infrastructure

One way to overcome the centralized issues in PKI using CA's was to let the users themselves provide the certificates as is done in Pretty Good Privacy (PGP). However, PGP itself still suffers from the problem of distributing the certificates using public key-servers, which are kind of centralized servers that need special treatment as such. To be really decentralized this storage must be accomplished without the need for any specially treated nodes. This is where the self-organized PKI comes in.

Using the same web of trust idea as PGP this approach puts the public certificates at each user. However, all users cannot know about every certificate and every other user. Instead each user hold a small number of certificates and when needed merge these certificates with some other node together getting enough certificates to enable the verification of a public key using a trust chain in the merged set of certificates.

In their paper [12] Hubaux, Buttyán, and İ.Capkun describe a graph algorithm in which each user, using only locally available information, calculates a sub graph from the complete trust graph. The algorithm, which they call "The Shortcut Hunter algorithm,"

proves to give quite good results when running it on actual PGP trust graphs. It gives a scalable, decentralized solution to the problem of PKI certificate distribution. They showed that the probability of two nodes' merged certificate lists holds a certificate chain between the two of them was high using a relatively small number of certificates in each node. Of course, this relates to the size of the complete trust graph and the assumption of it to be connected [12].

As is the case for all such trust graphs it is possible to extract information about that knows whom, or at least who trusts whom. This is the foundation of the decentralized certificate authority approach but may be used in malicious ways by adversaries.

6. PARTIALLY DISTRIBUTED CERTIFICATE AUTHORITY

This solution proposed by Zhou and Hass [12] uses a *(n, k)* threshold scheme to distribute the services of the certificate authority to a set of specialized server nodes. Each of these nodes is capable of generating a partial certificate using their share of the certificate signing key sk_{CA}, but only by combining *k* such partial certificates can a valid certificate be obtained.

The solution is suitable for planned, long-term Ad hoc networks. Since it is based on public key encryption it requires that the all the nodes are capable of performing the necessary computations.

Finally it assumes that a subset of the nodes is willing or able to take on the specialized server role.

6.1 System Overview

The system contains three types of nodes; client, server and combiner nodes. The client nodes are the normal users of the network while the server and combiner nodes are part of the certificate authority. The server nodes are responsible for generating partial certificates and storing certificates in a directory structure allowing client nodes to request for the certificates of other nodes. The combiner nodes, which are also server nodes, are responsible for combining the partial certificates into a valid certificate. Although not stated implicitly by the authors the system also has an administrative authority, which will be termed the dealer. The dealer is the only entity in the system that has knowledge of the complete certificate-signing key sk_{CA} [12].

Every node in the network has a public/private key pair and it is the responsibility of the dealer to issue the initial certificate for the nodes public key as well as distributing the public key pk_{CA} of the certificate authority, which is needed to verify the certificates [12].

The certificate authority as a whole has a public/private key pair, pk_{CA}/sk_{CA} of which the public key is known to all network nodes. The private key sk_{CA} is shared among the server nodes according to Shamir's secret sharing scheme [12].

Figure 6.1 illustrates the different components of the system.

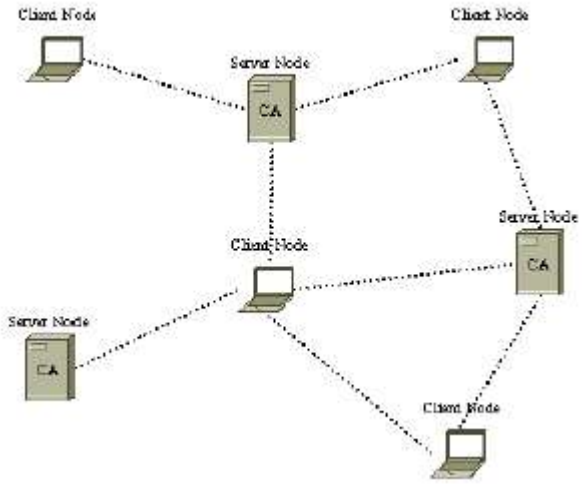

Figure 6.1 System architecture showing three server nodes of which one is also a combiner.

6.2 Certificate Issuing

Before any node may join the network it must first obtain a valid certificate from the dealer off-line. At the same time the node should be supplied with the CA's certificate along with any other parameters that are required [12].

6.3 Certificate Renewal

The certificates are only valid for a certain amount of time and therefore need to be renewed before they expire. When a node wishes to renew its certificate, it must request a certificate renewal from a minimum of *k* server nodes. If the request is granted, each of these *k* server nodes generates a partial certificate with a new expiration date. These partial certificates are then sent to a combiner, which could be one of the *k* servers, which then combines the partial certificates [12].

If any of the servers are compromised they may generate an invalid partial certificate, which they then send to the combiner. The certificate produced by the combiner will then also be invalid. Verifying the validity of the certificate before accepting it prevents this type of denial of service attack. If the combiner detects that the certificate is invalid it will request a new set of partial certificates until a valid certificate is obtained [12].

If a node changes its private/public key pair it will need to update its certificate with the new public key, this is accomplished in a similar way as the renewal.

6.4 Certificate Retrieval

The server nodes are responsible for storing the certificates of all nodes in the network. This allows any nodes requiring the public key of any other nodes to simply request the corresponding certificate from any of the server nodes.

This service requires that all nodes must register their certificate with the servers when they initially join the network. The servers must also have a mechanism of synchronizing their certificate directories in the case of updates and renewal [12].

6.5 System Maintenance

The maintenance of the certificate authority consists of two main parts, the issuing of the initial shares and the proactive update of the shares to protect against mobile adversaries. The share update can also allow the system to change its configuration, e.g. from a *(3, 8)*

to a *(2, 5)* threshold scheme. This could be useful e.g. if some of the servers have been compromised or for some other reason are unavailable [12].

During the bootstrapping of the network the dealer generates *n* shares of the CA's private key sk_{CA}.

At periodic intervals the servers update their shares of the CA's private key sk_{CA}. At the beginning of the update, each server generates a random *(n, k)* sharing of *0* and distributes a share to each of the other servers, each of these shares are called sub shares. Each server now has *n* sub shares from different servers, which are added to their old share giving them they're new updated share [12].

During the share update a malicious server can potentially launch a denial of service attack against the distributed CA by generating invalid sub shares. When the other servers use these to update their old shares the updated share will also be invalid, effectively preventing the correct generation of certificates. To prevent such an attack the authors propose using a verifiable secret sharing scheme. This allows the servers to validate each of the sub shares before using them [12].

Due to the highly dynamic topology of Ad hoc networks, all servers might not be connected during the share update. Therefore, mechanisms must be in place to handle such situations. An example would be if the network were segmented into two parts. The servers in each part may update their shares independently of each other. If the network later merges, the shares held by the servers will be inconsistent. The authors mention a mechanism that deals with this but no details about it are given [23].

7. FULLY DISTRIBUTED CERTIFICATE AUTHORITY

This solution is first described by Lou and Lu in [2] and later analyzed by Lou et al in [24]. It uses a *(n, k)* threshold scheme to distribute an RSA certificate-signing key to all nodes in the network. It also uses verifiable and proactive secret sharing mechanisms to protect against denial of service attacks and compromise of the certificate-signing key.

This solution is aimed towards planned, long-term Ad hoc networks with nodes capable of public key encryption. However, since the service is distributed among all the nodes when they join the network, there is no need to elect or choose any specialized server nodes.

7.1 System Overview

In this solution, the capabilities of the CA are distributed to all nodes in the Ad hoc network, see figure 7.1. Any operations requiring the CA's private key sk_{CA} can only be performed by a coalition of *k* or more nodes.

The services provided by the CA can be grouped as certificate related services and system maintenance services. The certificate related services include certificate renewal and revocation.

The system maintenance services include incorporating joining nodes into the CA, i.e. provide them with their share of the CA's private key sk_{CA}. This service is called share initialization [2].

The system maintenance also includes proactively updating the shares of the CA's private key to protect it from being compromised. This service is termed share update.

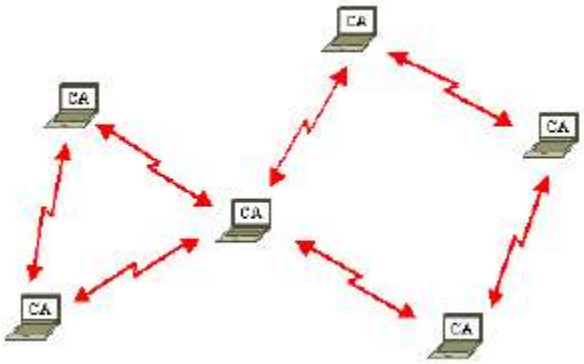

Figure 7.1 Fully distributed CA service where all nodes in the network are equal and each holds a share of the signing key.

The availability of the service is based on the assumption that every node will have a minimum of *k* one-hop neighbors and that the nodes are provided with a valid certificate prior to their joining the network. The system then provides services to maintain and update these initial certificates [2].

7.2 System Maintenance

This section describes the steps required to setup and maintain the distributed CA service. The maintenance is required to handle the joining of new nodes and to protect the service against attackers who try to compromise the CA service.

7.2.1 System Bootstrapping

During the system-bootstrapping phase, the administrative authority responsible for the Ad hoc network (known as the dealer) initializes the first *k* nodes. The initialization includes providing the nodes with their own certificates $cert_{id}$, the CA's certificate $cert_{CA}$

and their shares of the CA's secret key sk_{CA}. The bootstrapping phase is illustrated in Figure 7.2 where,

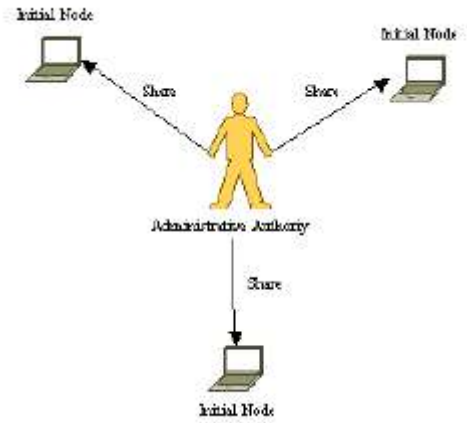

Figure 7.2 Share initialization during the bootstrapping phase. The dealer provides each of the initial nodes with their share.

the dealer is the only entity that has access to the certificate-signing key sk_{CA} and he/she can therefore issue the initial certificates mentioned above [2].

To initialize the first *k* nodes of the network the following steps are performed:

1. The dealer generates the sharing polynomial

$$f(x) = a_0 + a_1 x + \ldots + a_{k-1} x^{k-1} \qquad (7.1)$$

over Z_N where $a_0 = SK_{CA}$.

3. Each of the initial *k* nodes, identified by id_i, i = 1…k are securely supplied with their polynomial share

$$S_i = \mathrm{f}(id_i) \bmod N. \tag{7.2}$$

3. The dealer broadcasts k public witnesses of the sharing polynomial's coefficients $\{ g^{a_0} \ldots g^{a_{k-1}} \}$ after which it destroys the polynomial and quits.

4. Each node verifies that it received a valid share in step 2 above by checking

$$g^{S_i} = g^{a_0} \cdot (g^{a_1})^{id_i} \cdots (g^{a_{k-1}})^{id_i^{k-1}} \tag{7.3}$$

After the bootstrapping of the first *k* nodes, the dealer is only responsible for the registration, initialization and initial certification of any new nodes joining the network [2].

7.2.2 Share Initialization

Any new nodes joining the network are incorporated into the distributed CA by being provided with their own share of the CA certificate-signing key sk_{CA}. Since the dealer is no longer part of the network this share distribution mechanism needs to be handled by the nodes that have already been initialized as illustrated in Figure 7.3.

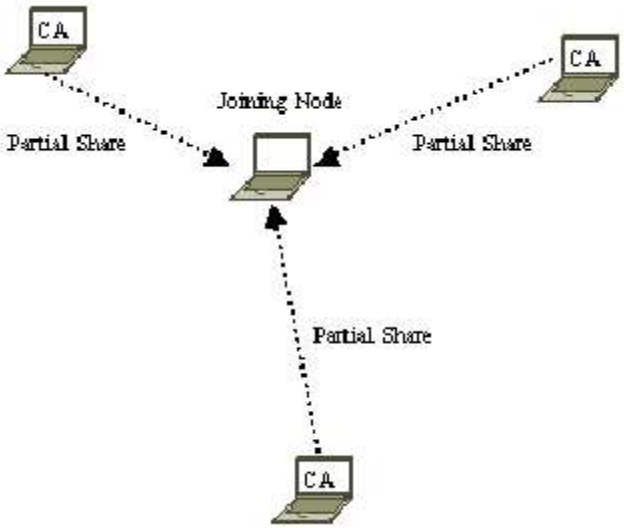

Figure 7.3 Share initialization during operational phase, due to joining node. Those nodes are already part of the CA service, which generate a new share from their shares and initialize the new node.

A node i already initialized can generate a partial share $S_{p,i} = S_i . I_{id_i}(id_p)$ for the joining node p, I_{id_i} (x) is the Lagrange term. By combining k such partial shares the complete share S_p for the joining node can be generated as following

$$S_p = \sum_{i=1}^{K} S_{p,i} = \sum_{i=1}^{K} S_i . I_{id_i}(id_p) = \mathrm{f}(id_p) \bmod \mathrm{N} \qquad (7.4)$$

However the joining node can only be allowed to know the value of the sum of the *k* sub shares, not the value of the sub shares themselves. The reason for this is that $I_{id_i}(id_p)$ is a publicly known value and therefore the S_i can be derived, thereby revealing the secret shares of the nodes in the coalition [2].

To protect the secrecy of the coalition nodes' secret shares, they shuffle their partial shares before sending them to the joining node. Figure 7.4 illustrates how the shuffling

scheme works in the case where the coalition consists of two nodes, node 1 and node 2. First node 1 and 2 secretly agree on a random shuffling factor $d_{1,2}$ where one of the nodes treats it as a positive number and the other node treats it as a negative number. The two nodes then add the shuffling factor to their respective sub shares and send the resulting shuffled sub shares to the joining node 3. Node 3 then adds up the shuffled sub shares resulting in its complete share S_3. In the case that *k* is larger than 2, i.e. the coalition consists of more than 2 nodes, each pair of nodes $\{i, j\}$ in the coalition agree on a shuffling factor $d_{i,j}$. Therefore the numbers of shuffling factors that need to be distributed are *k (k-1)/2*, where *k* is the size of the coalition.

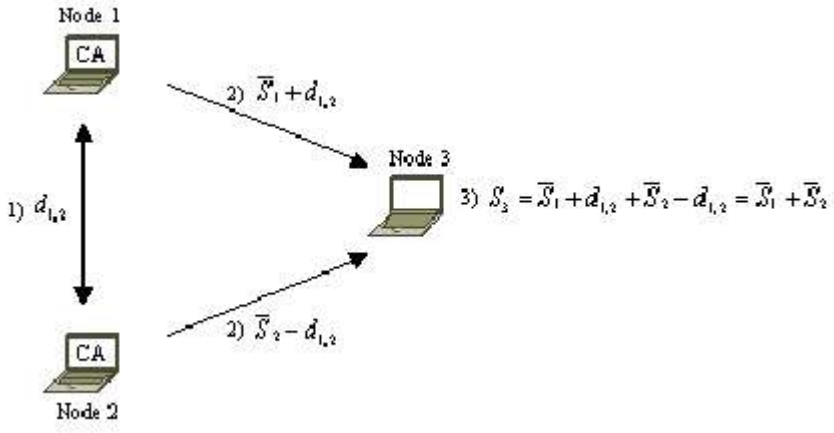

Figure 7.4 Complete shuffling scheme.

Below is a step by step description of the share initialization described above, where node *p* is the node that wishes to be initialized:

1. Node *p* locates a coalition *B* of *k* nodes, $B = \{id_1 \ldots id_k\}$ and broadcasts an initialization request.

2. Each node in the coalition verifies the certificate of node *p*, $cert_p$ and checks that it hasn't been revoked. If the verification fails the request is denied.

3. Each pair of nodes $\{i, j\}$ in the coalition now need to agree on a shuffling factor $d_{i,j}$. One of the nodes generates the shuffling factor and encrypts it with the public key of the other node and signs it. It also generates and signs a public witness $g^{d_{ikj}}$ of the shuffling factor. The witness is needed to be able to detect and identify any misbehaving coalition nodes that generate an invalid shuffled partial share. The shuffling factors and their witnesses are then sent to the requesting node *p*.

4. Node *p* distributes the shuffling factors and the witnesses received to all the coalition nodes.

5. Each coalition node *j* now generates the partial share $S_p^i = S_j . I_{id_j}(id_p)$ for node *p* and shuffles it using the shuffling factors received in the previous step. The shuffled partial share $\overline{S}_p^i$ is generated as follows:

$$\overline{S}_p^i = S_p^j + \sum_{i=1, i\neq j}^{K} [sign(id_i - id_j).d_{i,j}] \bmod N \tag{7.5}$$

Where sign (x) = $\begin{matrix} -1, x \leq 0 \\ 1, x > 0 \end{matrix}$

6. Each coalition node *j* sends its shuffled partial share $\overline{S}_p^i$ to node p.

7. Node *p* verifies each of the received shuffled partial shares $\overline{S}_p^i$ by checking that

$$g^{\overline{S}_p^j} = g^{S_p} \prod_{i=1, i\neq i}^{K} (g^{d_{i,j}})^{sign(id_i, id_j)} \tag{7.6}$$

Where $g^{S_p} = g^{a_0} . (g^{a_1})^{id_p} \ldots . (g^{a_{k-1}})^{id_p^{k-1}}$

If the verification fails node p drops the invalid shuffled share and issues a new initialization request from a new coalition excluding the misbehaving node that was detected. Node p also revokes the misbehaving nodes certificate and floods an accusation against it.

8. If all the shuffled partial shares where correct, node p can obtain its new share by adding the shuffled partial shares

$$S_P = \sum_{j=1}^{K} \ddot{S}_p^j \mod N \tag{7.7}$$

After being initialized with its share of the certificate signing key sk_{CA} the node has become part of the CA and can participate in the provision of the CA's service, including certificate renewal and revocation as well as initializing any joining nodes with their share of the certificate-signing key sk_{CA} [23].

7.2.3 Share Update

Proactive secret sharing is required to protect against attackers that (given enough time) can compromise k or more nodes and thereby be able to reconstruct the shared secret, in this case the certificate-signing key sk_{CA}. As illustrated in Figure 7.5 the lifetime of the network is divided up in time periods, where each time period consists of two phases, the operational phase and the share update phase. During the operational phase nodes can renew their certificate and request share initialization. During the share update phase all nodes that have been initialized update their shares in a distributed manner.

During the share update phase the following three steps are performed:

- Collaborative generation of the update polynomial $f_{update}(x)$.
- Distribution of the update polynomial to all network nodes.

- Distributed evaluation of the share update $\overline{S}_p = f_{update}(id_p)$ of all nodes p.

Figure 7.5 show different phases during the lifetime of the Ad hoc Network.

At the beginning of the share update phase each node initiates the update process with the probability $1/\hat{n}$ where $\hat{n}$ is a known estimate of the total number of nodes in the network. The node that decides to initiate the share update then locates a coalition of *k* nodes that generate the update polynomial $f_{update}(x) = b_1 x + b_2 x^2 + ... + b_k x^{k-1}$ mod N. Each of the polynomial's coefficients is then encrypted, signed, and flooded in the network. At this point every node in the network has received $\{E_{pk_{CA}}(b_1) \ldots 6\, E_{pk_{CA}}(b_{k-1})\}$, which it has authenticated by verifying the signatures. This prevents an attacker from being able to flood a false update polynomial. Each node *p* in the network can now generate its update share $\overline{S}_p = f_{update}(id_p)$; however since the polynomial is encrypted it must request the evaluation of its update share from a coalition of *k* nodes. Each of the nodes in the coalition returns a partial update share to the requesting node *p* that adds them to obtain its complete update share $\overline{S}_p$. Adding this update share to the nodes current share provides the updated share of sk_{CA} [23].

During the share update phase two different versions of shares can be present since not all nodes update their shares simultaneously. However, as long as the nodes in the coalition use the same version of their shares, the services required can be performed. Therefore the nodes keep the old share until the share update phase is complete. At this point they destroy the old share and thereafter only use the updated share until the next share update phase [22].

7.3 Certificate Issuing

This solution is based on the assumption that all nodes have been initialized, registered, and issued a valid certificate before they join the network. The distributed CA never issues new certificates; it only manages certificates once they have been initially created. The responsibility of initializing, registering, and certifying new nodes belongs to the administrative authority responsible for setting up the network, i.e. the dealer [22].

7.4 Certificate Renewal

Since certificates are only valid for a limited time period they must be renewed before they expire. When a node *p* wishes to renew its certificate *cert* it requests a certificate renewal from a coalition of *k* of its one-hop neighbors. Each node i in the coalition first checks if the old certificate has not already been expired and that it as not been revoked. If they agree to serve the request they each generate a new partial certificate $cert_i$ and return it to the requesting node *p*. Node *p* then combines the *k* partial certificates to obtain its updated certificate $cert_{update}$. The details of this process are described below:

1. Node *p,* that wishes to update its certificate, broadcasts a certificate renewal request to its one-hop neighborhood, see Figure 7.6.

2. Each of the nodes that receive the request, check that the requesting nodes current certificate *cert* has not expired or been revoked. If the certificate is still valid and the receiving node i decides to serve the request it generates the partial certificate $cert_i$ by using its share of CA's certificate signing key sk_{CA}. The partial certificate is generated as follows; $cert_i = (cert)^{S_i}$ modN where S_i is node i's sub share of sk_{CA}.

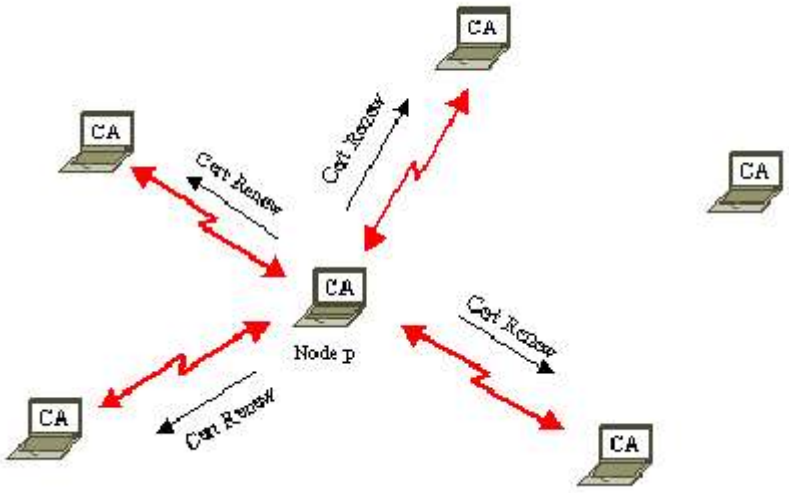

Figure 7.6 Node requesting for certificate renewal.

4. Each node i serving the request randomly generate a u and calculate $A_1 = g^u$ and $A_2 = cert^u$. It then calculates

$$c = \text{Hash}\,(g^{S_i}, cert_i, A_1, A_2) \tag{7.8}$$

$$r = u\text{-}c.S_i\,. \tag{7.9}$$

The values A_1, A_2, r are used to enable the verification of the generated partial certificate $cert'_p$

4. Each node i returns $cert_i$, A_1, A_2, r to the requesting node p.

5. Node p now chooses k of the partial certificates received and verifies each of them by checking that $g^r.(g^{S_i})^c = A$ and that $cert^r.(cert_i)^c = A_2$. Node p can generate c by

applying the same hash function used in step 3 above. A_1, A_2, r are known from step 4 and

$$g^{S_i} = g^{a_0} \cdot (g^{a_1})^{id_i} \cdots (g^{a_{k-1}})^{id_i^{k-1}} \quad (7.10)$$

can be calculated using the public witnesses of the sharing polynomial's coefficients.

6. If any of the partial certificates failed the verification, the certificate of the node that generated the invalid certificate is revoked.

A different partial certificate is then chosen from those received in step 4 and validated as described in step 5. If less than *k* valid certificates are received the certificate renewal fails.

7. Node *p* combines the *k* valid partial certificates to obtain a candidate certificate $cert_{updated}'$.

$$cert_{updated}' = \prod_{i \in B} (cert_i)^{I_{id_i}(0)}$$
$$= (cert^{\sum_{i \in B} S_i I_{id_i}(0)})$$
$$= (cert)^{t.N + sk_{CA}} \quad (7.11)$$

8. The updated certificate should be

$$cert_{updated} = cert^{sk_{CA}} \text{ [23].} \quad (7.12)$$

7.5 Certificate Revocation

The certificate revocation mechanism is based on the assumption that all nodes monitor the behavior of their one-hop neighbors and maintain their own certificate revocation lists. If a node discovers that one of its neighbors is misbehaving it adds its certificate to the CRL and floods an accusation against the node. Any node receiving such an accusation first checks its CRL to verify that the accusation didn't originate from a node whose certificate has been revoked. If the accuser's certificate has been revoked the accusation is ignored. However, if the accusation originated from a valid node, the accused node is marked as suspect. When thresholds of legitimate accusations, i.e. k accusations, against the same node are received the accused node's certificate is revoked [21].

Figure 7.7 illustrates the sequence of events when two nodes B and C detect a misbehaving node D, first nodes B and C revoke node D's certificate and thereafter they flood an accusation. Upon receiving the accusations, node A first validates them by checking its own CRL to make sure that the certificate of node B or C hasn't been revoked. Since a threshold of accusation has been received (in this example $k = 2$) node A also adds node D to its CRL. Node A then retransmits the accusations of node B and C. Node E and F receiving these accusations then perform the same steps as node A.

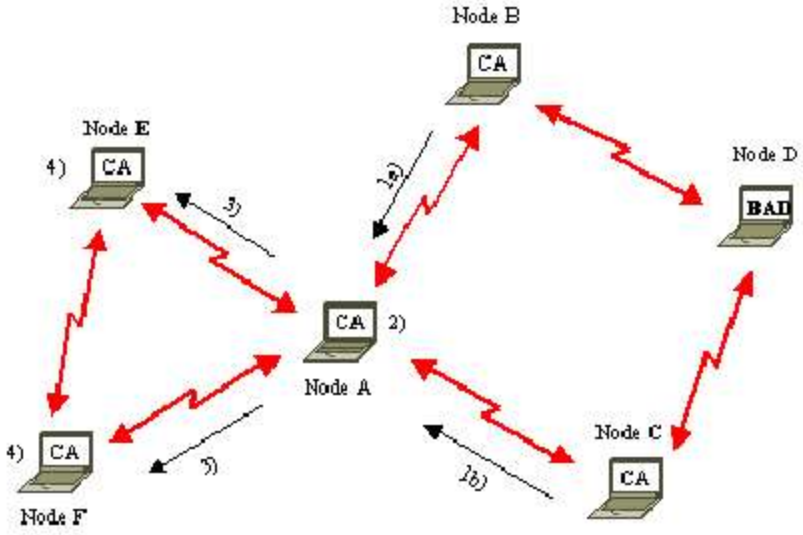

Figure 7.7 Certificate revocation and distributed maintenance of certificate revocation lists.

8. THE PROPOSED ARCHITECTURE

In this section, we present our overall architecture that provides ubiquitous, scalable and robust authentication services, specifically the certification services for mobile users in Ad hoc wireless networks. Our design employs the asymmetric cryptographic primitives, specifically standard RSA. We take the certificate based authentication mechanisms.

8.1 Design Rational

In this section we will explain design issues of proposed architecture, distributed key management system.

8.1.1 Certification Services

As identified, certification services include issuing/renewing certificates, revoking certificate, storing and retrieving certificates and certificate revocation lists (CRL). Each certificate is stamped with an expiration time. Nodes have to renew and get a new certificate before expiration. Certificates may become invalid before expiration. The revocation service has to put such information on the revoked certificates into CRL's for queries from nodes [5].

The focus of this report is to provide ubiquitous certification services to mobile nodes in an Ad hoc wireless network.

8.1.2 Certificate-Based Approach

There are five basic aspects that characterize a secure communication: Data integrity, which ensures that only authorized parties are able to modify transmitted information. Authentication is a mechanism, which the sender of a message can be correctly identified.

Message confidentiality, which ensures that transmitted data is accessible only for reading by, authorized entities. Non-repudiation requires that neither the sender, nor the receiver of a message be able to deny the transmission and Service availability, which demands that computer system assets be available to authorized entities when needed [5].

Certificate-based approaches readily provide workable solutions to the first four of the above-mentioned functions. We seek to solve the problem of service availability through the design of the ubiquitous certification protocols, specifically for mobile Ad hoc networks.

8.1.2.1 Authentication via Certificates

Authentication in the RSA context relies on two factors. For node v_i to authenticate itself to another node, it has to prove its knowledge of the private key sk_i and the association between itself and the advertised public key pk_i. A challenge/response protocol can be followed to prove the knowledge of the private key sk_i. A certificate proves the association.

Generally a certificate $CERT_i$ is a statement $cert_i$ that is signed by some trusted authority's private key SK. The statement $cert_i$ may read: “It is certified that the personal public key of node v_i until time t is pk_i ". In RSA, $CERT_i = (cert_i)_{SK}$. The certificate does not need to be private. It is actually made public by some directory service or being carried by node v_i. The authority's public key PK is assumed to be well known to every node so each node can verify the validity of v_i 's certificate $CERT_i$ by applying PK to check if $cert_i = (CERT_i)_{PK.}$ A valid certificate proves the association between node v_i and its public key pk_i [5].

8.1.2 Threshold Secret Sharing

Threshold secret sharing exhibits several desirable properties that fit well in Ad hoc networks:

(a) A key feature of Ad hoc networks is lack of centralized control. Distributing and sharing the control are consistent to the nature of Ad hoc networks.

(b) Ubiquitous services are enabled once the secret is fully distributed to each locality. Besides, intrusion detection is more practical and efficient if localized.

(c) The threshold K is the balance point between service availability and intrusion tolerance. An adversary group must destroy partial shares (N-K+1) holders to turn off certification services, whereas it must break in at least K shareholders to steal SK the system secret. We next compare our choice of threshold with two extreme cases in the entire solution space:

"1 out of N" scheme: the centralized solution is actually a special case of the threshold secret sharing. The threshold K=1 results in vulnerable security since SK is held by a single entity. The centralized server suffers from a single point of service denial and a single point of compromise. In the former case, no service can be provided to network entities. In the latter case, the system secret SK is revealed and the entire system is compromised [5].

"N out of N" scheme: The threshold results in maximal security but minimal fault tolerance. Unless the adversary breaks into every entity in the system, it cannot expose SK. However, since all N entities are required in providing the service, the system secret is lost once system failure occurs at any single entity. Besides, this approach is not scalable in a large network [5].

Therefore, our threshold security design seeks to provide flexible tradeoffs among security, availability, intrusion tolerance, and fault tolerance. By choosing $1< K < N$

1we avoid both the" single point of compromise" and the" single point of failure/DoS attack" problems. Though a certain number of entities may be intruded or faulty, the system secret SK is not exposed or lost.

8.1.3 Polynomial Secret Sharing

Our design makes extensive use of the polynomial secret sharing due to Shamir. A secret, specifically the certificate-signing key SK, is shared among all n nodes in the network. To distribute SK, a random coefficients polynomial of order $k-1$ is generated:

$$f(x) = a_0+a_1x+a_2x^2+...+a_{k-1}x^{k-1} \quad (8.1)$$

Node v_i gets its share $Pv_i = f(v_i)$ privately. Any coalition of k nodes can potentially recover SK by Lagrange interpolation:

$$f(x) = \sum_{i=1}^{k} S_i I_{id_i}(x) (\text{mod } p) \quad (8.2)$$

where $I_{id}(x) = \prod_{\substack{i=1 \\ i \neq j}}^{k} (x - id_j) / (id_i - id_j)$

No coalition up to k-1 nodes yields any information about SK. This mechanism is robust against the attacks of the adversaries.

8.1.4 Secret Share Updates

In threshold secret sharing, adversaries need to compromise at least K entities to expose the system secret SK. However, they have the entire system lifetime to mount these attacks. Gradual break-ins into K entities over a long period of time may be possible and therefore long-lived secret sharing is not sufficient. A naive make-up is to periodically change SK itself. [5]

However, in a wireless Ad hoc network without centralized control, it is an open issue who has the authority to annul the current SK and enforce the change. This motivates our

design choice to periodically update the shares instead of the *SK* itself. Further defend the privacy of the shared *SK* against the adversaries' there are a lot of candidate polynomials that can be applied to share *SK* in the polynomial secret sharing context. The mechanism is based on periodically updating the secret shares with different polynomials. We apply this technique with scalable algorithms to further improve the robustness against adversaries.

8.1.5 Certificate Revocation

In an Ad hoc network there may be some misbehaving nodes. Defend the network against adversaries we use a certificate revocation mechanism. In this revocation mechanism misbehaving nodes that are accused by some other nodes are added to certificate revocation list. When a node id is in certificate revocation list this node is not trusted and treated as an adversaries.

8.2 Primitives

At the bootstrapping phase of the network, a dealer has to send each networking node privately its share of the *SK* , according to a polynomial of order k -1. We denote this process the "initialization" of nodes. A large Ad hoc wireless network may contain hundreds or even thousands of networking nodes. Relying on the dealer for the initialization, as most works in threshold secret sharing do, is not scalable and may not be feasible. Moreover, new nodes may join over the lifetime of the network. Maintaining a dealer on line to handle future node joins would compromise the overall system robustness and security. The dealer would become the single point of failure and inviting target of DoS attacks. Since the dealer is the only entity that holds the complete *SK* , maintaining a dealer on line also increases the adversaries' chance to compromise the dealer and therefore *SK* , thus effectively turns down the whole services. We address these issues by a scalable initialization mechanism called "self-initialization". In our architecture, the dealer is only responsible to initialize the very first k nodes, no matter how large the network would be. The initialized nodes collaboratively initialize other

nodes, typically their neighboring nodes. Repeating this procedure, the network progressively "self-initializes" itself.

In our architecture, each networking node i with nonzero ID v_i is associated with a personal RSA key pair $< sk_i ; pk_i >$ sk_i denotes $v_i's$ private key for decryption and signing pk_i denotes $v_i's$ public key for encryption and verification. The private key sk_i is only possessed by node v_i for two typical usages. One is for v_i to decrypt messages that are encrypted by the corresponding public key pk_i. The other is for node v_i to sign some messages or statements to generate a signature. The public key pk_i is advertised to other nodes. It is used for other nodes to encrypt messages toward v_i, and to verify a signature that is supposedly signed by node v_i with its private key sk_i.

In our architecture, each node carries a certificate signed by SK (Secret Key). The corresponding PK (Public Key) is to be well known to each node in the network, so that certificates are globally verifiable. Nodes without valid certificates will be isolated, that is, the network will not forward their packets. Essentially these nodes without valid certificates are treated the same as adversaries. They are denied from access to any network resources. When a mobile node moves to a new location, it exchanges certificates with its new neighbors and goes through mutual authentication processes to build trust relationships. Neighboring nodes with such trust relationship help each other forward and route packets.

Each certificate is stamped with an expiration time. Nodes have to renew and be issued a new certificate upon the expiration of its old certificate. In the centralized authentication architecture, nodes have to contact a CA server for this service. In our architecture, we distribute the private key SK that is used to sign certificates into each node of the network by a polynomial of order k - 1. A node v_i with its expiring old certificate requests a new certificate from any coalition of k nodes, typically among its one-hop neighbors. A neighboring node checks its record on v_i that requests certification services.

If its record shows v_i a well-behaving legitimate node, it returns a “partial" certificate by applying its share of SK. Otherwise the request is dropped. By collecting k partial certificates, v_i combines them together to generate the full new certificate as if it were from a CA server. A misbehaving or broken node that is detected by its neighbors will be unable to get a new certificate. It will be cut out from the network at the expiration of its current certificate. Moreover, we propose explicit certificate revocation and distributed CRL storage/retrieval mechanisms to deal with the scenario when a node is detected misbehaving or broken well before its current certificate's expiration time.

By distributing the certification services into each node's one-hop neighborhood, we realize ubiquitous service availability for mobile nodes and robustness against DoS attacks. The communication is localized into one-hop range, which minimizes the exposure to wireless channel errors.

8.3 Algorithm

We adopt the polynomial secret sharing to share SK among the network. In our algorithms, node v_i's polynomial share $P v_i$ and its additive share $SK\ v_i$ in term of a specific coalition are defined over the ring Z_N. After the initialization of the network, each node with its ID v_i holds a polynomial share

Detailed algorithms are as follows:

Secret Sharing:

$$f(x) = a_0+a_1x+a_2x^2+...+a_{k-1}x^{k-1} \qquad (8.3)$$

Node v_i gets its share $P v_i = f(v_i)$ privately. Any coalition of k nodes can potentially recover SK by Lagrange interpolation defined as in (8.2)

Pv_i = f (v_i) mod N, where f (x) = $SK + f_1 x + ... + f_{k-1} x^{k-1}$ is the secret polynomial. Since the certificate verification key PK is to be well known, N is also well known as part of PK. Only the ID's and shares of the participating k nodes are involved.

Partial signature Generation: Node v_i generates a partial signature with his/her partial private key SK_{v_i} $CERT_{vi}$. $cert$ is a new nodes certificate request.

$$CERT_{vi} = (cert)^{SK_{vi}} \text{ modN} \tag{8.4}$$

Combining Partial Certificates: Node v_i sends $CERT_{vi}$ to the requesting node v_j. Upon receiving k partial certificates $\{CERT_{v1}, CERT_{v2}, \ldots, CERT_{vk}\}$ from the coalition B, node v_i combines them together by multiplication to generate a "candidate certificate" CERT'.

$$CERT' = \prod_{j=1}^{k} CERT_{vj} = (cert)^{\sum_{j=1}^{k} SK_{vj}} = (cert)^{t.N+SK} = \text{CERT}.(\text{cert})^{\text{t.N}} \bmod N \tag{8.5}$$

Inputs: CERT' the candidate certificate
Cert: statement of the certificate, to be signed

Output: CERT: Certificate

1: Z := $(cert)^{-N}$ mod N
2: j: = 0, Y: = 1
3: while j < k do
4: Y: = Y. Z mod N, j: = j + 1
5: if (cert = Y^{PK} mod N) then
6: break while
7: end if
8: end while
9: output Y = CERT

As long as k neighbors respond with partial certificates, other neighbors are free to move or fail, additional response messages may be lost. A neighboring node can move away freely after sending back the partial certificate.

Proactive Secret Share Update: Given n servers, Let $(s_1, s_2 \ldots s_n)$ be a (n,k) sharing of the private key S of the service, with server i having s_i. Assuming all servers are correct, share refreshing proceeds as follows first, each server randomly generates $s_{i1}, s_{i2} \ldots . s_{in}$ is a (n,k) sharing of 0. We call these newly generated s_{ij}'s sub shares. Then, every sub share s_{ij} is distributed to server j through a secure link. When server j gets the sub shares $s_{1j}, s_{2j} \ldots s_{nj}$ it can compute a new share from these sub shares and its old share

$$s'_j = s_i + \sum_{i=1}^{n} s_{ij} \ . \qquad (8.6)$$

The above operations require minimum requirements on the reliability of wireless communications.

8.4 Implementation

In this section we will explain details of implementation.

8.4.1 Self Initialization

Each well-behaving, certificate-holding entity v_x can also obtain a secret share P_{v_x}, which is used to derive SK_x, in providing certification services. We have devised a localized self-initialization algorithm to compute secret shares given an un-initialized entity v_x a local coalition of K shareholders, $\{v_{(x,1)}, v_{(x,2)}, \ldots, v_{(x,k)}\}$ can compute its secret share

P_{v_x} Can compute its secret share:

$$P_{v_x} = \sum_{i=1}^{k} S_i I_{id_i}(x) (\text{mod } p) \tag{8.7}$$

where $I_{id}(x) = \prod_{\substack{i=1 \\ i \neq j}}^{k} (x - id_j) \Big/ (id_i - id_j)$

Each coalition member $v_{(x,i)}$ computes a partial secret share $SS_{(x,j)}$ from its secret share $P_{(x,i)}$ then returns the partial secret share to the requester. The requester's secret share is the sum of the partial secret shares.

As Lagrange coefficients are publicly known, v_x can derive $P_{v_{(x,j)}}$ by knowing $SS_{(x,j)}$ directly. To keep $P_{v_{(x,j)}}$ as a secret only to its owner $v_{(x,i)}$, we employ a complete shuffling scheme. In the complete shuffling scheme, a random between any two members in the coalition is generated. The entity with larger ID treats the nonce as a positive number while the other side treats it as a negative number. Each member totally has K-1 such nonces. Before $v_{(x,i)}$ sends back the partial secret share, it sums the K-1 nonces and $SS_{(x,j)}$, then sends a shuffled partial secret share to v_x. It is easy to verify that v_x obtains the same value P_{v_x} nonce is exchanged.

Protocol requires four steps

1. The un-initialized node broadcasts the service request, along with local coalition information.

2. Each coalition member selects a random nonce for other members if its ID is lower than the other one. Each nonce is encrypted with the personal $< pk >$ of the intended receiver. The requester acts as the router and accepts all shuffling packages from K-1 members (The member with the highest ID needs not select nonces).

3. The requester routes encrypted nonce's to intended receivers.

4. Each member decrypts all nonce's, computes a shuffled partial secret share, and then sends it back to v_x.

8.4.2 Issuing of Certificates

A new certificate request is sent by a client who wants to obtain a new certificate on a public key is controls (it generates and stores).

Before client begin protocol is must be authenticated. First new node send a message includes modN which is used to authenticate user. After authentication new node, nodes send a message including threshold K value.

To start issuing a new certificate client sends the message (issue-request, cid, key, name (IP...),), and waits for response. Key is signature or encryption key.

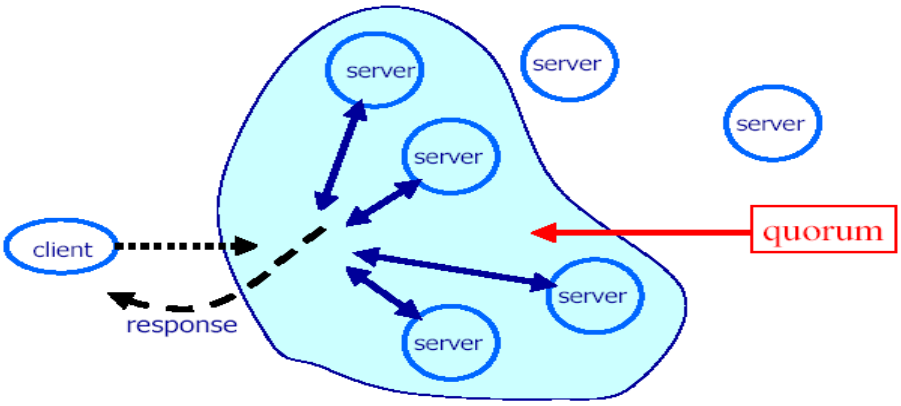

Figure 8.1 Certificate Request

Client will retrieve messages (issue-reply,cid,s) . After decrypting new node sends reply to threshold node broadcast answer (issue-answer,cid,d) decrypted. After verified decrypted value is same as value send nodes send sub share of new node encrypted with new nodes.

8.4.3 Distributing Own Public Key

After initialize phase new node send its certificate to all neighbors, if a node receiving a certificate may decide to store it or may discard it. If it store this new arriving certificate, it send certificate to the node that wants this certificate. This model also provides distributed certification service.

8.4.5 Giving Sub Share to New Arrival Node

After certification phase node must be initialized as a server node to give certification service. So it needs a sub share of CA private key SK. According to proactive share refreshing algorithm all nodes generate a value from their sub shares, which must be sending to the new arrival node in a secure way. It is encrypted nodes public key in my protocol. So only new node can decrypt these values and after combining all values it obtains its own sub share. This also provides update of all sub shares of nodes in Ad hoc network.

A local coalition of K shareholders, can compute its secret share P_{vx} by Lagrange interpolation:

$$P_{vx} \equiv f(v_x) \equiv \sum_{j=1}^{K} P_{v(x,j)} . l_{v(x,j)}(v_x) \equiv \sum_{j=1}^{K} SS_{(x,j)} (\text{mod } N) \qquad (8.8)$$

Each coalition member $v_{(x,j)}$ computes a partial secret share $SS_{(x,j)}$ from its secret share P_v then returns the partial secret share to the requester. The requester's sub share is the sum of K partial secret share.

8.5.6 Node Leave

Node may send node leave message or it may be decided by the nodes that node is unreachable. In that state the certificate of the node must be revoked.

If threshold value K is in critical it must be updated.

8.5.7 Key Update

To further enhance the robustness of our design, we periodically update all the secret shares to invalidate compromised secret shares. In the bootstrapping phase all secret shares are tagged with version number 1 and ID. Each secret share update will increase the version by 1, and there is a predefined minimal lapse time between two consecutive

updates. During the update transition period, secret shareholders with the same version tag can collaboratively function as the CA

We employ the proactive secret share update algorithms to create a community of K entities with new version of secret shares. Then the self-initialization protocol is employed to propagate the new version over the entire network. If there are multiple proactive updates concurrently going on, then we may have version conflicts. In our solution is to include the lowest ID of the original updated entities into the version tag. The version tag with the lowest ID wins in the case of version conflicts.

8.5.8 Certificate Revocation Scheme

The first duty of a node after entering a network is to broadcast its certificate to all the nodes, and simultaneously sends a request that the nodes send their profile tables. The profile table contains information about the behavior profile of each node in a network. The information in the profile tables is used to determine whether or not a given certificate should be revoked. Each node is required to compile and maintain a profile table. A profile table can be represented in the form of a packet of varied length depending on the number. It should be noted that a node is allowed to accuse a given node only once throughout the lifetime of a certificate. Therefore, when an accusation is broadcast, the nodes are required to check the data in their profile tables, and add the information regarding the new accusation (certificate serial number of the accuser and the node being accused, and the date), only if there is no prior record of the of accusation launched against the nodes.

Details of the fields and content of the profile table are as follows:

1-Owner's ID: This field is the first 32 bits of the profile table.

It contains an integer indicating the serial number of the owner's (the node that compiled the profile table) digital certificate.

2. Node count: this is a 16-bit field containing a short integer indicating the owner's perspective regarding the current number (N) of nodes the network.

3. Peer ID: This is a 32-bit field containing the certificate serial number of a node that is accused of misbehavior. This field also serves the purpose of a marker: if it contains zero, it indicates the end of the profile table.

4. Certificate status: This field contains a 1-bit flag; it is set if the certificate of peer i is revoked and unset otherwise.

5. Accusation info: This is a 64-bit field; the first 32 bits contains an integer indicating the certificate serial number of a node that accused peer i of misbehavior. The remaining 32 bits contains the date that the accusation was made.

If field 3 does not contain zero, the profile table continues with the certificate status and accusation info fields; and if there are more than one accuser, it continues with 97-bit blocks containing information about other accusers.

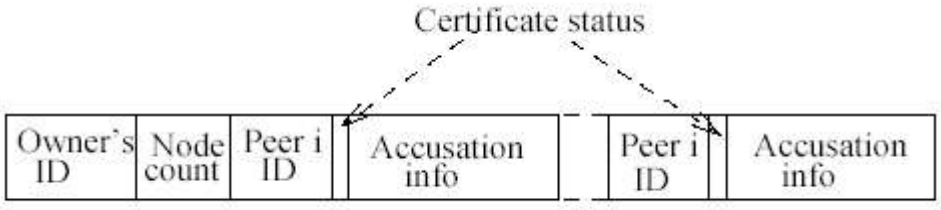

Figure 8.2 Fields of status table

The information regarding the number of accusations, the identity of the accusers, the nodes being accused and the date the accusation was made, should be consistent in all the profile tables. If the node requesting the profile tables, notices any inconsistency, it is expected to launch an accusation against the node(s) that sent the inconsistent data. Profile table data is assumed to be inconsistent if it differs from the data contained in the majority of the other profile tables. Finally, the node compiles its own profile table based on the data the majority of the profile tables contain.

It should be noted that a node is allowed to accuse a given node only once throughout the lifetime of a certificate. Therefore, when an accusation is broadcast, the nodes are required to check the data in their profile tables, and add the information regarding the new accusation (certificate serial number of the accuser and the node being accused, and the date), only if there is no prior record of the accuser accusing that particular node.

8.5.8.1 Stipulation for Certificate Revocation

In addition to a profile table and a certificate repository, each node is required to compile and maintain a status table. Initially, it is compiled from the data in the profile table, and updated simultaneously along with the latter when a new, pertinent accusation is received. The status table is used to ascertain the status of a certificate; it consists of the following info:

Number of accusations against node i (A_i): The total number of accusations- limited to one per node made against node i.

Number of accusation made by node i :(α_i) the total number of accusation limited to one per node made by node i -1

Behavior index of node i (β_i): The behavior index of a node i β_i is a number such that $0<\beta_i \leq 1$ It is a measure of the status of the node amongst its peers. The greater the value of β_i the higher the status of the given node i β_i is computed as follows:

$$\beta_i = 1-\lambda A_i \qquad (8.9)$$

where $\lambda = 1/2N-3$ N is the node count.

Weight of node i accusation (ω_i): The weight of a node accusation or potential accusation. It depends on the behavior index and the number of accusations it made.

$$\omega_i = \beta_i - \lambda \alpha_i \quad (8.10)$$

where $\lambda = 1/2N-3$

Revocation quotient (R_j): This number determines whether or not the certificate for node j should be revoked. It is computed as follows:

$$R_j = \sum_{i=1}^{N} \sigma_{ij} \omega_i \quad (8.11)$$

The revocation quotient threshold R_T is configurable parameter. Its value depends on the sensitivity of the security requirement. Typically R_t maybe N/2 where N is the number of nodes in network If $R_j > R_t$ certificate of node j is revoked [26].

9. EVALUATION OF IMPLEMENTATION

In this section we will briefly explain simulation and simulation environment.

9.1 Simulation

In this section we describe our architecture for ubiquitous security services in wireless Ad hoc networks. From the cryptographic perspective, our design is based on the concepts of threshold secret sharing and secret share updates.

From the system aspect, the architecture is fully distributed and localized.

9.1.1 Simulation Environment

-Programming Language: JAVA (Java Development Kit 1.4.0)

-Development Environment (IDE): Eclipse

-Development Platform: Windows

-Libraries Used: Java Crypto Environment (JCE), GNU Crypto

9.1.2 Features

There is a configuration file used in Distributed Key Management System simulation. This configuration file contains some initialization parameter, such as, node number in the network, threshold value etc.

The config.conf file:

1. Configuration

a. Threshold Value

b. Node Count

c. Certificate Validity Period

d. Key Length

2. Node

a. Range

b. Coordinate Constraint

c. Password Phase

9.1.3 Bootstrapping

In bootstrapping phase the network constructed according to configuration parameters; Node count, threshold value, key length etc.

In simulation one Ad hoc network simulated. That network has some nodes and a public/private key pairs ($< PK ; SK >$). Also all nodes have their own public/private key pairs.

Simulated Ad hoc network's private/public key pair is generated according to key length read from configuration file. Then a random coefficient polynomial is constructed. Polynomial f (x) of degree k -1, then any members of the community can recover the secret via Lagrange interpolation, while any less than K members of the community reveal no information of the secret. This coefficient's of polynomial is randomly selected between of 1-10.

Generated Public Key (PK) is converted to certificate structure by signing with private key SK . This network's certificate is distributed all off the nodes in Ad hoc network.

RSA secret key SK $\{d, n\}$ generated and randomly selected a polynomial

f (x) = $SK + a_1x + a_2x^2 + ... + a_{k-1}x^{k-1}$,such that the shared f (0) is SK . Each entity

v_i (i =1, 2, 3.... N) (N: Number of nodes in network) holds a secret share $P_{vi} = (f(v_i) \bmod N)$.

Each networking node i with nonzero ID v_i is associated with a personal RSA key pair $< sk_i ; pk_i >$. sk_i Denotes $v_i's$ private key for decryption and signing, pk_i denotes v_i's public key for encryption and verification. The private key sk_i is only possessed by node v_i for two typical usages. One is for v_i to decrypt messages that are encrypted by the corresponding public key pk_i. The other is for node v_i to sign some messages or statements to generate a signature. The public key pk_i is advertised to other nodes. It is used for other nodes to encrypt messages toward v_i and to verify a signature that is supposedly signed by node v_i with its private. Nodes pk_i is converted to certificate structure and signed with Ad hoc network signing key SK , and then all the nodes in the network can verify each other's certificate. Nodes without valid certificates will be isolated, that is, the network will not forward their packets. Essentially these nodes without valid certificates are treated the same as adversaries. They are denied from access to any network resources.

9.1.4 Packages

In our simulation we used package hierarchy

a. tr.edu.itu.adhoc

b. tr.edu.itu.adhoc.certificate

c. tr.edu.itu.adhoc.crypto

d. tr.edu.itu.adhoc.gui

e. tr.edu.itu.adhoc.messages

f. tr.edu.itu.adhoc.network

g. tr.edu.itu.adhoc.simulation

h. tr.edu.itu.adhoc.store

i. tr.edu.itu.adhoc.utils

9.1.5 Classes

In this section we give some details about classes that we used in simulation.

9.1.5.1 Package: tr.edu.itu.adhoc.certificate

In this section we give all classes and some details about these classes in tr.edu.itu.adhoc.certificate package.

NodeCertificate：This is a class that is used to represent each nodes certificate data type. Each node has an object in the type of NodeCertificate.

CertificateRevocationList：This is a class that is used to represent certificate revocation list in the Ad hoc network. Each node verifies other client's certificate with CertificateRevocationList. In the system there is only one instance of CertificateRevocationList object. Each node searches for fresh copy of CertificateRevocationList and stores it in its local store.

BlackList: This is a class that is used to store node accusations. If a node accurate another node this accusation is not directly accepted and the node certificate is not revoked .In our system it is put into a blacklist. This instance of BlackList object must be synchronized and this done in our system with a protocol.

9.1.5.2 Package: tr.edu.itu.adhoc.crypto

In this section we give all classes and some details about these classes in tr.edu.itu.adhoc.crypto package.

Crypto: This class is used for all cryptographic functions. One of them is to generate keys according to the given algorithm and key length. Another one is for encrypting given data with a given public key and also there is another function used for decryption of an encrypted data. Also there is a function that is used for verifying signature.

CyptoUtils: This class is used as a utility object. There are some constants used in simulation.

MyLagrange: This class is used in LaGrange operations.

9.1.5.3 Package: tr.edu.itu.adhoc.gui

In this section we give all classes and some details about these classes in tr.edu.itu.adhoc.gui package.

DrawNode: This class is used for displaying a node and its all details. For example, nodes location, nodes neighbors, nodes certificate.

DrawPath: This class is used to find a path between to node. In this class dijsktra algorithm is used for path finding. Also in this class the path between nodes are drawn in a frame.

LinkOutputToTextArea: This class is used for debugging. All the system messages are written into a text area. Also this class is used to give some messages to the user.

SimulationComponent: This class is the main class of the simulation. The project is run from this class and generates management interface.

9.1.5.4 Package: tr.edu.itu.adhoc.messages

In this section we give all classes and some details about these classes in tr.edu.itu.adhoc.messages package.

Message: This class provides an interface to all other message classes. In this class we defined all methods and properties that a message object should have.

AuthenticationMessage: This message class implements Message interface. In order to use this class, destination node and data should be set. This class is used for new joining nodes to authenticate themselves to the network. This message is send all the neighboring nodes. To verify the message integrity also messageMAC control is added. In this message source node shows the node that wants to join in to the adhoc network.

AuthenticationFailMessage: This message class implements Message interface. The joining node sends its neighbors AuthenticationMessage and put a data in to this message, if the data is incorrect authentication fails and nodes received AuthenticationMessage sends back AuthenticationFailMessage.

AuthenticationSuccess: This message class implements Message interface. The joining node send its neighbors AuthenticationMessage and put a data in to this message, if the data is correct authentication is succeeded and nodes received AuthenticationMessage sends back AuthenticationSuccessMessage.

CertificateAccusationMessage: This message class implements Message interface. This class is used when a nodes suspicious from another node. This node sends CertificateAccusationMessage to all nodes in its range. In this message node sets accused nodes id and reason of accusation.

CertificateAccusationResponseMessage: This message class implements Message interface. This type of message is sent when node receives a message in type CertificateAccusationMessage. First it checks its black list if same node accused by the same node and also checks certificate status from certificate revocation list. Then this

node calculates accusation rate of accusatory and suspicious node. Then node and this information to its blacklist and send it over all network.

CertificateRetriavalMessage: This message class implements Message interface. This type of message is sent when node wants to receive some other node certificate. Source and requesting node ids are set before sending this message to all over the network.

CertificateRetriavalRespMessage: This message class implements Message interface. This type of message is sent when node receives a message type of CertificateRetriavalMessage from some other node. If node has this certificate in its store it sends it to the requesting node. Else this node sends the message to its neighbors.

CertificateRevocationMessage: This message class implements Message interface. When a node receives a message in type of CertificateAccusationMessage, it first calculates behavior index of accusatory node and suspicion node. If it value exceeds a revocation range then this nodes certificate should be revoked and should be put into certificate revocation list. This type of message is sent to all over the network and gives other nodes information about revoked node.

Confirmation: This message class implements Message interface. When a new node wants to join to the network it starts joining protocol. If protocol is successfully applied at the end node sends Confirmation message and it sends its certificate to all over the network. This message also sends when a protocol successfully applied.

JoinInit1Message: This message class implements Message interface. When a new node wants to join to the network, it first send authentication message. If the response to this message is Authentication success message it send joint step 1 message. In this message there is new nodes public key and message integrity provided by Message Authentication Code (MAC).

Join1RespMessage: This message class implements Message interface. When new arrival node send JoinInitStep1 Message to its neighbors, all the nodes receiving this message sends a random data which is encrypted with new nodes public key. Message integrity is provided by Message Authentication Code (MAC).

JoinInit2Message: This message class implements Message interface. After sending joiningstep1 message new node to its neighbors, nodes received this message sends back a random data encrypted with new nodes public key, and nodes sends this messages back to new node as joinInitResponse message. After new node receives joinInitResponse it takes the encrypted data and decrypts it with its private key and sends back clear text to each node respectively. Message integrity is provided by Message Authentication Code (MAC).

LeaveMessage: This message class implements Message interface. LeaveMessage is sent when a node wants to leave the network. If a node wants to leave network it sends leave message, the nodes in its range takes this message and sends this messages to the nodes in their ranges. After receiving a leave message node updates its certificate store and updates its routing table. If the threshold value is critic it must be adjusted so the share update protocol dynamically starts.

ProactiveShareUpdateMessage: This message class implements Message interface. ProactiveShareUpdateMessage is sent when the threshold value should be updated. This protocol begins automatically when threshold value is critic. If the total node number is high in respect of threshold value or it is low, threshold value should be updated. An algorithm works after each node join or node leave.

SecureCommunication: This message class implements Message interface. If a node wants to communicate with a node securely it send a message encrypted with the destination nodes certificate and also it is signed with the source nodes private key.

9.1.5.5 Package: tr.edu.itu.adhoc.network

In this section we give all classes and some details about these classes in tr.edu.itu.adhoc.network package.

Network: This class provides an interface to Ad hoc network class. In this class we defined all methods and properties that an Ad hoc network object should have.

AdhocNetwork: This class implements Network interface. AdhocNetwork class is used in bootstrapping phase. In simulation environment we generate all nodes, system private key, public key and systems certificate. Also nodes key pairs are generated and according to threshold algorithm system private key is distributed to all nodes. In bootstrapping phase nodes certificates are signed with systems private key and system public key is loaded to all nodes. In simulation we also give random coordinates to all nodes.

Router: This class is used to find path between nodes. A node executes this algorithm when it needs to send a message to a node that it not its routing table. Also periodically it is executed and routing table is updated according to nodes new locations.

Node: Sending message to other nodes and processing received messages are done in this class. Each node has an incoming message queue and an outgoing message queue. Messages that will be sent are put to outgoing message queue and received message are in incoming message queue. There is a thread that watches this queues and process messages. Also certificate and certificate revocation list operations are done in this class.

9.1.5.6 Package: tr.edu.itu.adhoc.simulation

In this section we give all classes and some details about these classes in tr.edu.itu.adhoc.simulation package.

SimulateNetwork: This class is used to simulate ad hoc network.

9.1.5.7 Package: tr.edu.itu.adhoc.store

In this section we give all classes and some details about these classes in tr.edu.itu.adhoc.store package.

Queue: This class defines a data structure and some methods to manage nodes local certificate store, routing table, certificate revocation list.

MyQueue: This class extends queue class. This class defines a data structure and some methods to manage nodes local certificate store, routing table, certificate revocation list.

CertificateStore: This class extends MyQueue class. Each node in Ad hoc network simulator has a certificate store. If a node needs other node's certificate it checks its certificate store. If a new certificate is obtained it is stored in nodes certificate store. Also there is certificate revocation list in this store. In our simulation environment node store is limited and works as first in first out model.

9.1.5.8 Package: tr.edu.itu.adhoc.utils

In this section we give all classes and some details about these classes in tr.edu.itu.adhoc.utils package.

Basic: This class is a utility class that is used to increase readability of out simulation. We define some constants in this class.

iniFiles: This class is a utility class that is used to process configuration file that is used to configure the simulation environment.

Logger: This class is a utility class that is used for logging purpose. All the message of the simulation program is saved in a file.

9.1.6 Snapshots from Simulation

The window shown in Figure 9.1 is open when the simulation is started. In this window first we should construct the network. The network is constructed according to the properties in config.ini file.

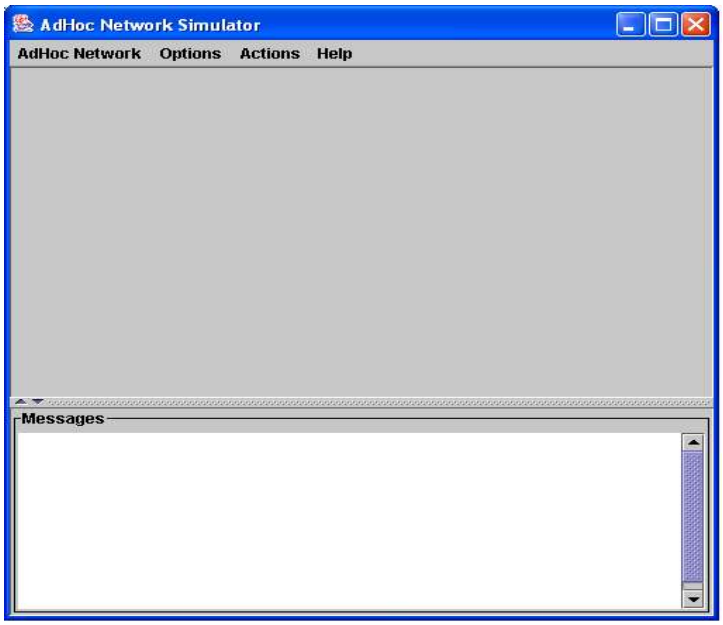

Figure 9.1 Ad hoc Network Simulator Main Frame

The window shown in Figure 9.2 is shown when a new ad hoc network is constructed. All the system key pairs and nodes key pairs are generated. Also partial shares of nodes are distributed to nodes.

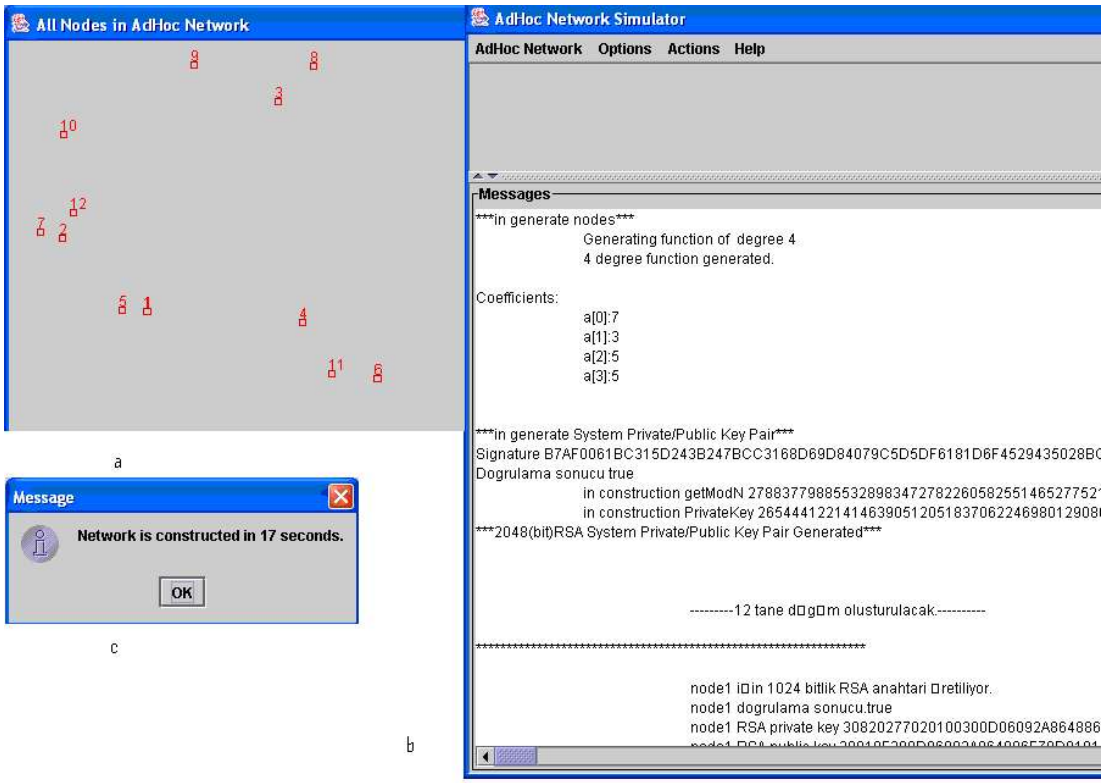

Figure 9.2 Construction of Ad hoc Network (a-c)

This Figure 9.3 shows Node generation time according to threshold value.

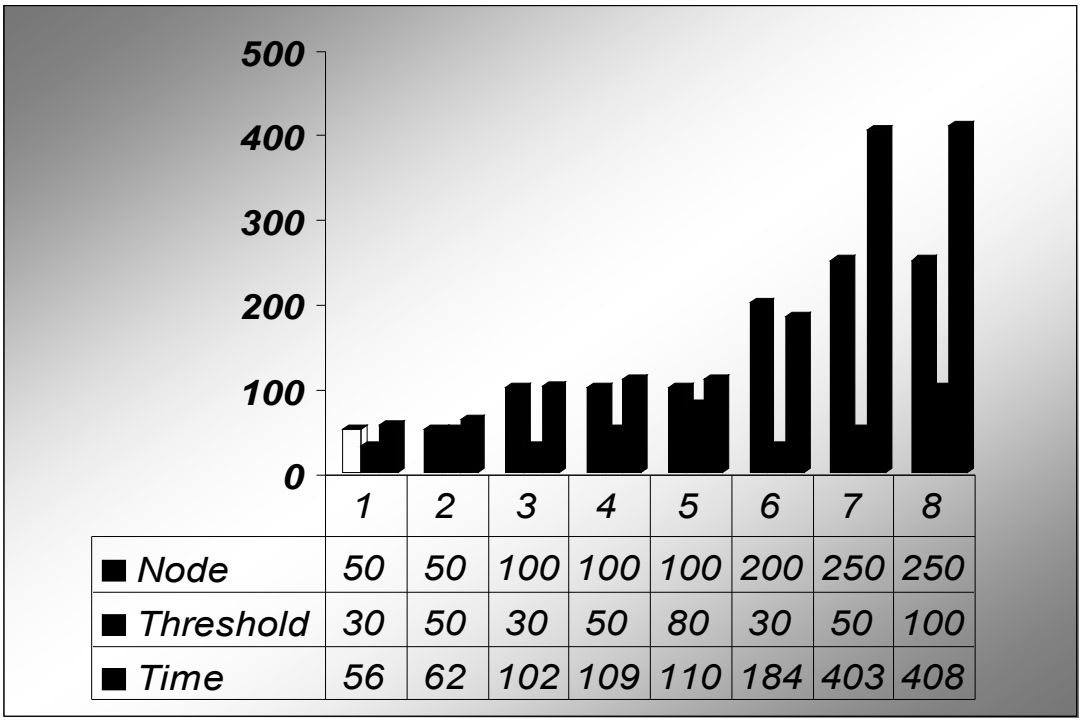

Figure 9.3 Node Generation Time

This menu is shown in Figure 9.4 is shown when user tries to find path from a source node to a destination node. This path is found according to Disjkstra shortest path algorithm.

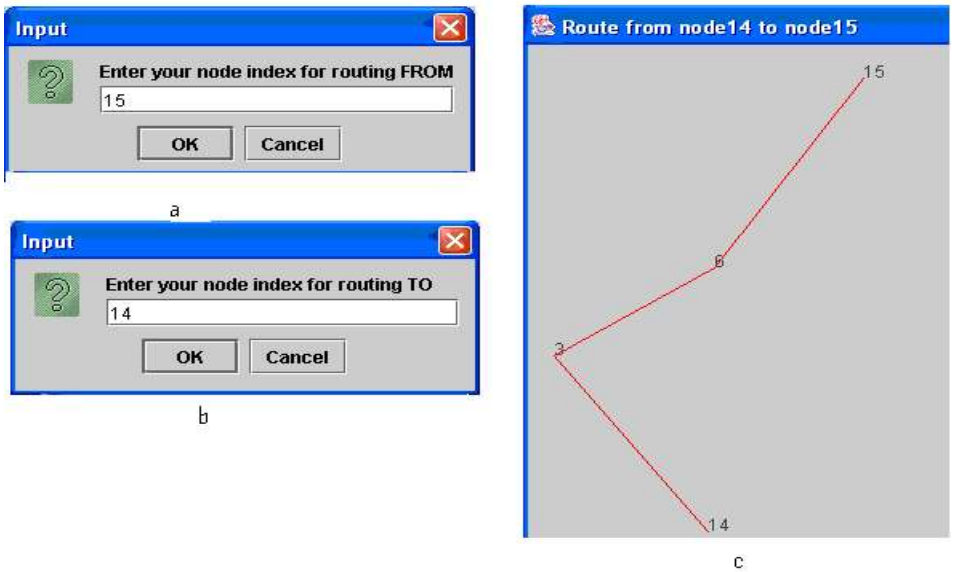

Figure 9.4 Paths Finding from a Node to another Node (a-c)

The window shown in Figure 9.5 is shown when a mobile node moves from a coordinate to another coordinate. Here destination and direction is parameter.

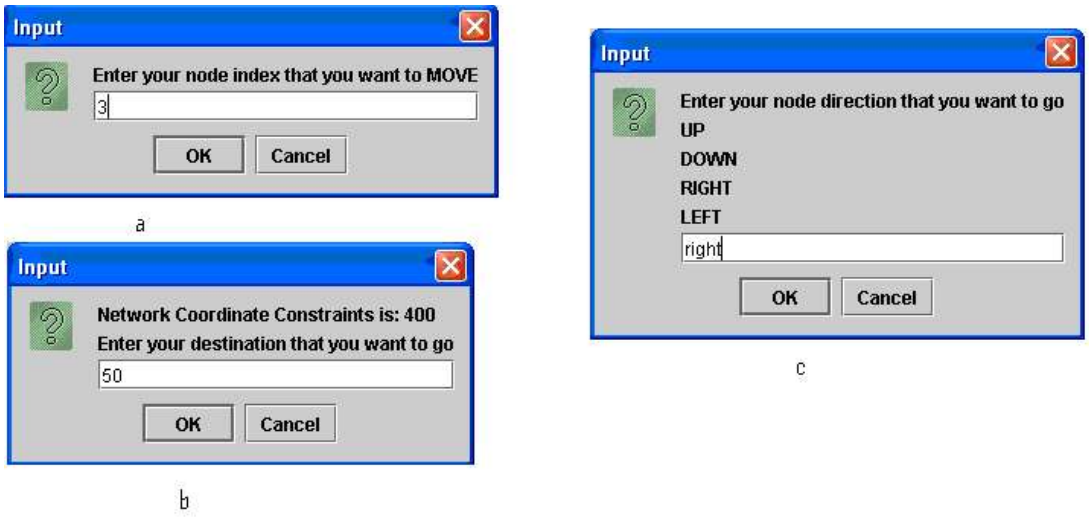

Figure 9.5 Moving a Node (a-c)

The figure in 9.6 shows all properties of a node. This nodes index is a parameter.

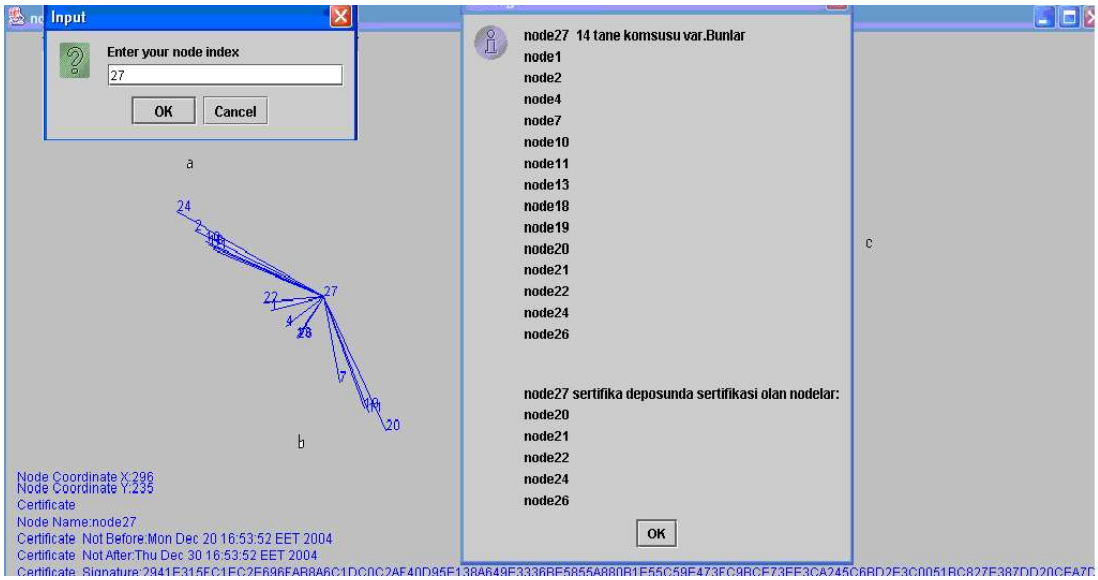

Figure 9.6 Properties of a Node (a-c)

The window shown in Figure 9.7 is shown when a new node tries to join in to the ad hoc network.

Figure 9.7 Adding a new Node to the Ad hoc Network (a-i)

In Figure 9.8 it is shown that, two nodes wants to communicate securely. Source node decides destination node and if its certificate is not in its local store it finds it. The message that will be encrypted is taken from source node and encrypted with destination nodes public key and send. After receiving encrypted message, destination node decrypts this message.

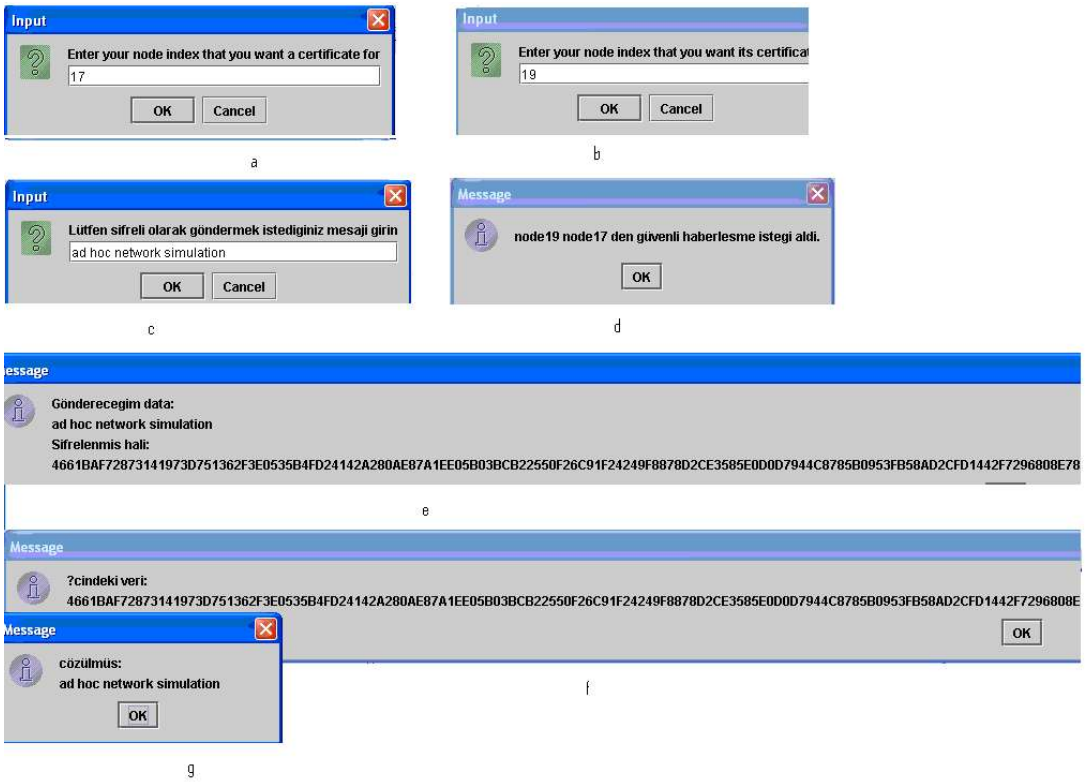

Figure 9.8 Secure Communication between nodes (a-g)

In figure 9.9 node accusation is shown.

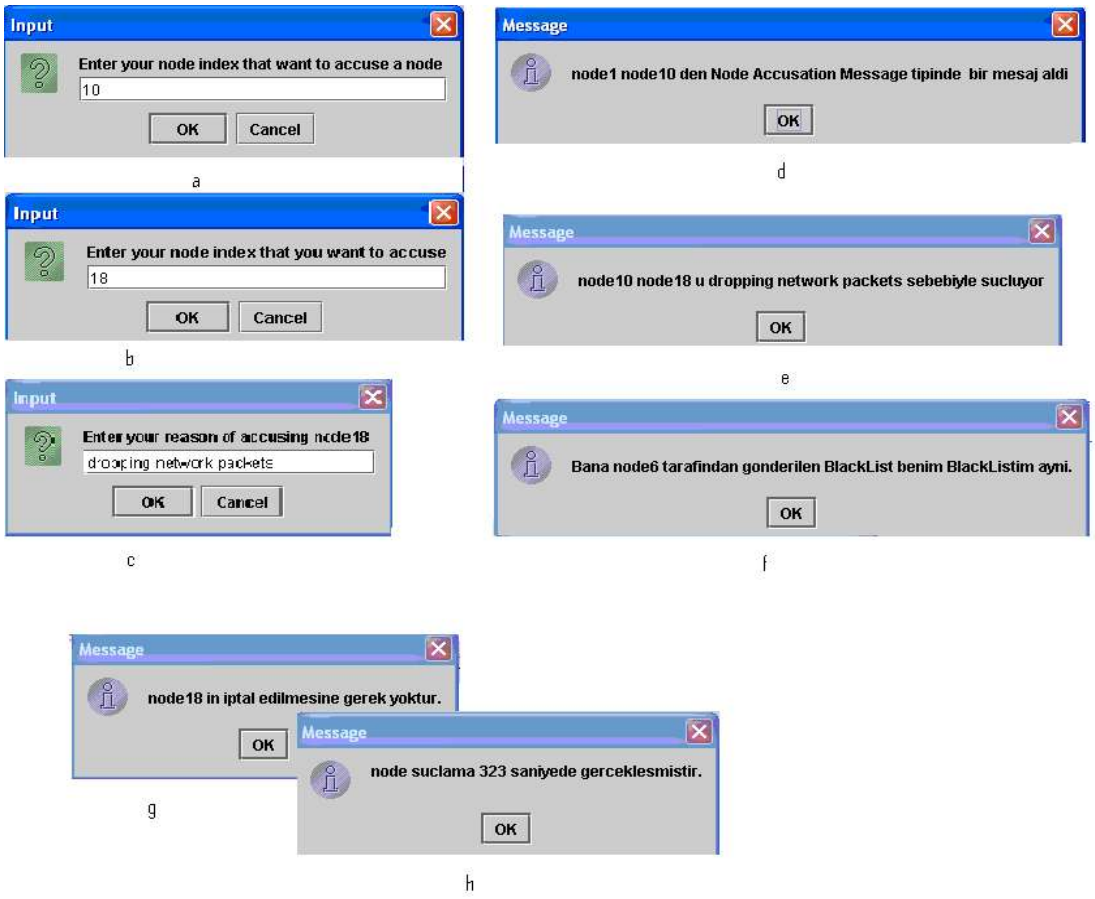

Figure 9.9 A node accusation process (a-h)

In figure 9.10 a node wants to leave the network. This process is shown in details.

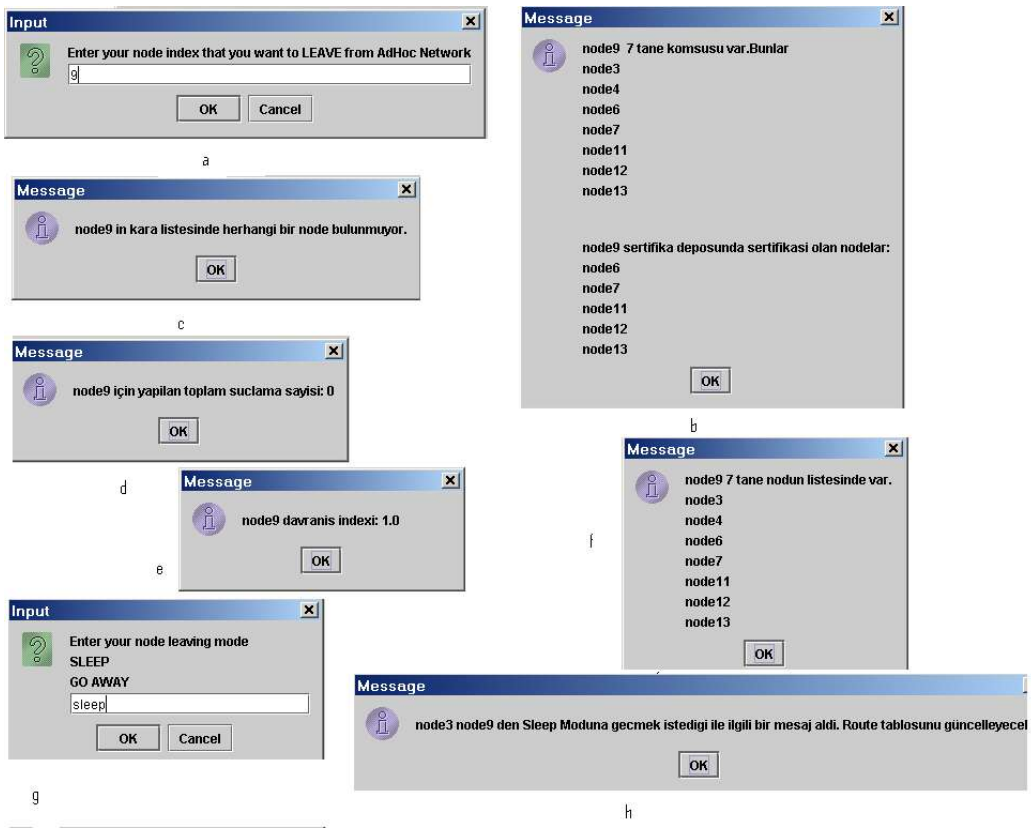

Figure 9.10 A node leaving process from ad hoc network (a-h)

10. CONCLUSION

The protocols presented in this paper provide an efficient way to achieve authentication for mobile nodes in Ad hoc network.

In this paper, we studied security challenge details in mobile Ad hoc networks. Also, we identified the requirements for security solutions, pointed out the importance of its security and propose a practical Public Key Infrastructure service for Ad hoc networks. Further, we have analyzed the security threats an Ad hoc network faces and presented the security objectives that need to be achieved. We also presented a practical solution to provide Certificate Authority support that is also flexible to the varying network configurations, simple to implement and easy to integrate with Ad hoc network routing protocols.

To build a highly available and highly secure key management service, we propose to use threshold cryptography to distribute trust among over all servers. We also adopted the fully distributed certificate authority approach, which means the capabilities of the certificate authority are distributed to all nodes in the Ad hoc network. This work aims at providing a scalable, distributed key management services in Ad hoc networks.

A prototype of the distributed key management service has been implemented, which shows its feasibility and availability. Using this practical PKI service, we present an effective and efficient communication protocol for correspondence with certification services.

This proposed protocol offers a ubiquitous authentication service for mobile nodes. Also confidentionaltiy, integrity and non-repudiation are guaranteed.

The security protocol proposed is based on public key cryptography and certificate-based approach. Also secret sharing and threshold cryptography algorithm is used. For this

security protocol a simulation environment is developed. In this simulation environment we implemented all steps offered. Bootstrapping phase, sharing of secret, joining to network, accusation of a node, providing mobility to a node and departure of a node steps are implemented.

In simulation environment, nodes are generated in a predefined area of random coordinates. This coordinates may be user specified parameter. Also mobility of a node is supplied by parameter and direction. This mobility can be provided by speed, time and direction. Also, in this simulation environment, there is no time synchronization. Because in certificate based authentication approach time is very important it should be implemented. Also proactive share update protocol should be optimized because this protocol takes long time. Also, threshold value may be optimized.

Consequently, our simulations we concluded that distributed key management system is a applicable architecture for mobile Ad hoc networks.

REFERENCES

[1] **Klas Fokine,** 2002. Key Management in Ad Hoc Networks

[2] **J. Macker S. Corson,** Jan 1999**.** Mobile Ad hoc networking (MANET): Routing protocol performance issues and evaluation considerations**.**

[3] **Silvia Giordano, Wiley**, 2000. Mobile ad-hoc networks. Handbook of Wireless Networks and Mobile Computing.

[4] **L. Zhou and Z. J. Haas,** IEEE Networks, Volume 13, Issue 6 1999. Securing Ad Hoc Networks**,** 1999.

[5] **Klas Fokine,** 2003. Key Management in Ad Hoc Networks.

[6] **W. Stallings**, 1999. Cryptography and Network Security: Principles and Practice, 2nd ed., Prentice-Hall.

[7] **Jaap Haartsen Mahmoud Naghshineh Jon Inouye,** 2002**.** Bluetooth: Vision, Goals, and Architecture.

[8] **A. Menezes, P. vanOorschot, and S. Vanstone**, 1996. Handbook of Applied Cryptography.

[9] **Vesa Kärpijoki**, Helsinki University of Technology Telecommunications Software and Multimedia Laboratory 2002**.**Security in Ad hoc Networks.

[10] **PERKINS, C. E., Ed**. 2001. Ad hoc networking.

[11]**Janne Lundberg,** 2002. Helsinki University of Technology Telecommunications Software and Multimedia Laboratory. Routing Security in Ad Hoc Networks.

[12] **HUBAUX, J.-P., BUTTYÁN, L., APKUN, S**. 2001. The Quest for Security in Mobile Ad hoc networks**.**

[13] **JOHNSON, D. B., MALTZ, D. A., AND BROCH, J**. 2000. DSR The Dynamic SourceRouting Protocol for Multihop Wireless Ad hoc networks.

[14] **L.Zhou and Z.J.Hass**, IEEE Networks, Volume13, Issue 6, 1999. Securing Ad hoc networks

[15 **C. Mitchell and F. Piper**,1988. Key Storage in Secure Networks. Discrete Applied Mathematics.

[16] **Ariadne ,** 2001**.** A Secure On-Demand Routing Protocol for Ad hoc Networks.

[17] **Konrad Wrona ,** 2002. Distributed Security in Ad hoc Networks.

[18] **F.Stajano and R. Anderson,** 2001**.** In Proceedings of the 7th International Aworkshop on Security Protocol. The resurrecting duckling: Security issues for ad-hoc wireless networks.

[19] **L.Zhou and Z.J.Hass**, IEEE Networks, Volume13, Issue 6, 1999**.** Securing Ad hoc networks.

[20] **Haiyun Luo, Songwu Lu,** Oct 2000**.** Ubiquitous and Robust Authentication Services for Ad hoc Wireless Networks.

[21] **J. Kong, P. Zerfos, H. Luo, S. Lu and L. Zhang**, IEEE ICNP 2001**.** Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks.

[22] **H. Luo, P. Zerfos, J. Kong, S. Lu and L. Zhang**, IEEE ISCC 2002. Self-securing Ad hoc Wireless Networks.

[23] **Seung Yi, Robin Kravets ,** 2002. Department of Computer Science University of Illinois at Urbana-Champaign**.** MOCA: Mobile Certificate Authority for Wireless Ad hoc Networks.

[24] **J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang,** In Proceedings of ICNP ’01.

Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks.

[25] **H. Luo and S. Lu**, 2000.Technical Report, UCLA Computer Science Department Ubiquitous and Robust Authentication Services for Ad hoc Wireless Networks.

[26] **Claude Cr´epeau_ and Carlton R. Davis**, 2000**.** A Certificate Revocation Scheme for Wireless Ad hoc networks

APPENDIX -A

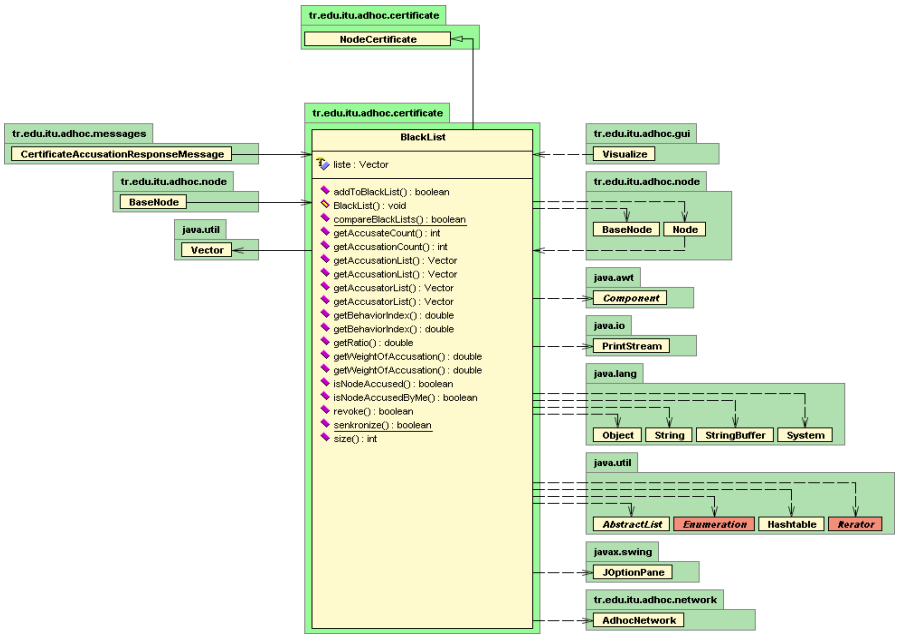

Figure A.1 Node Certificate Class

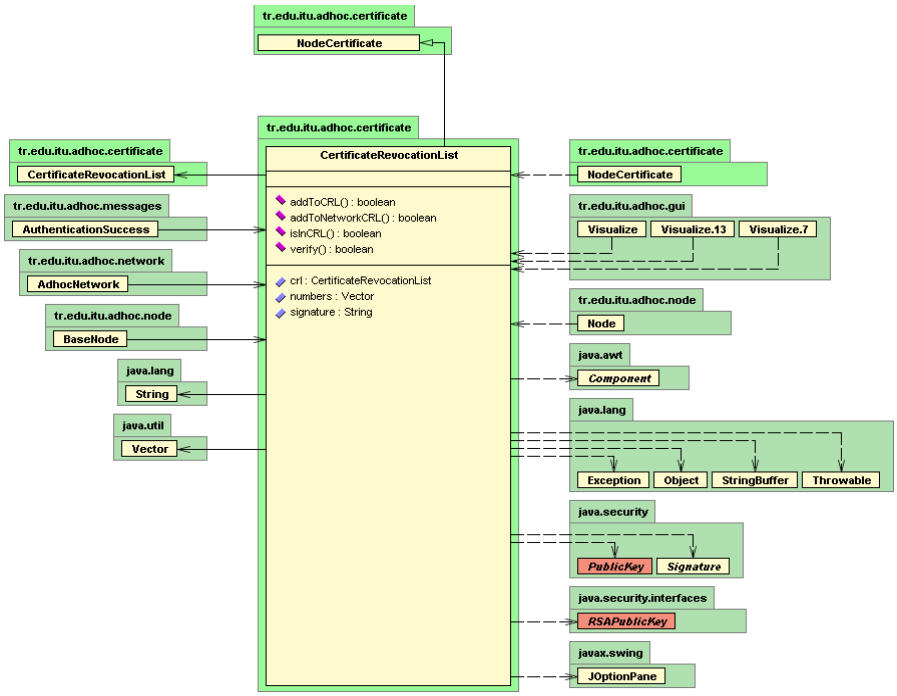

Figure A.2 Certificate Revocation Class

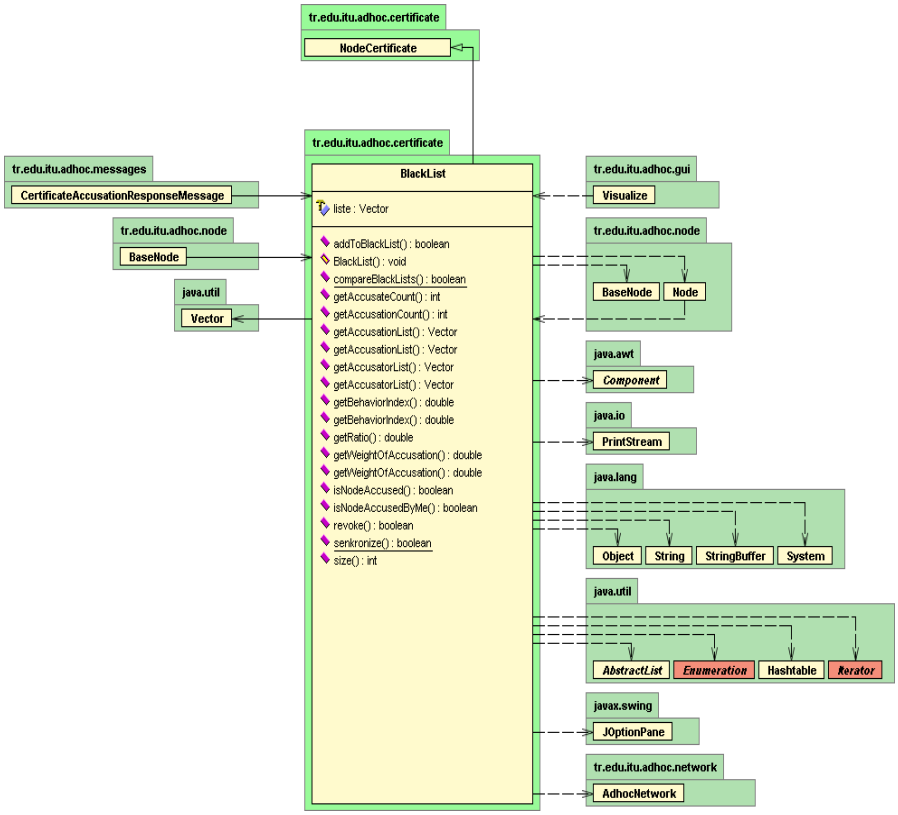

Figure A.3 Black List Class

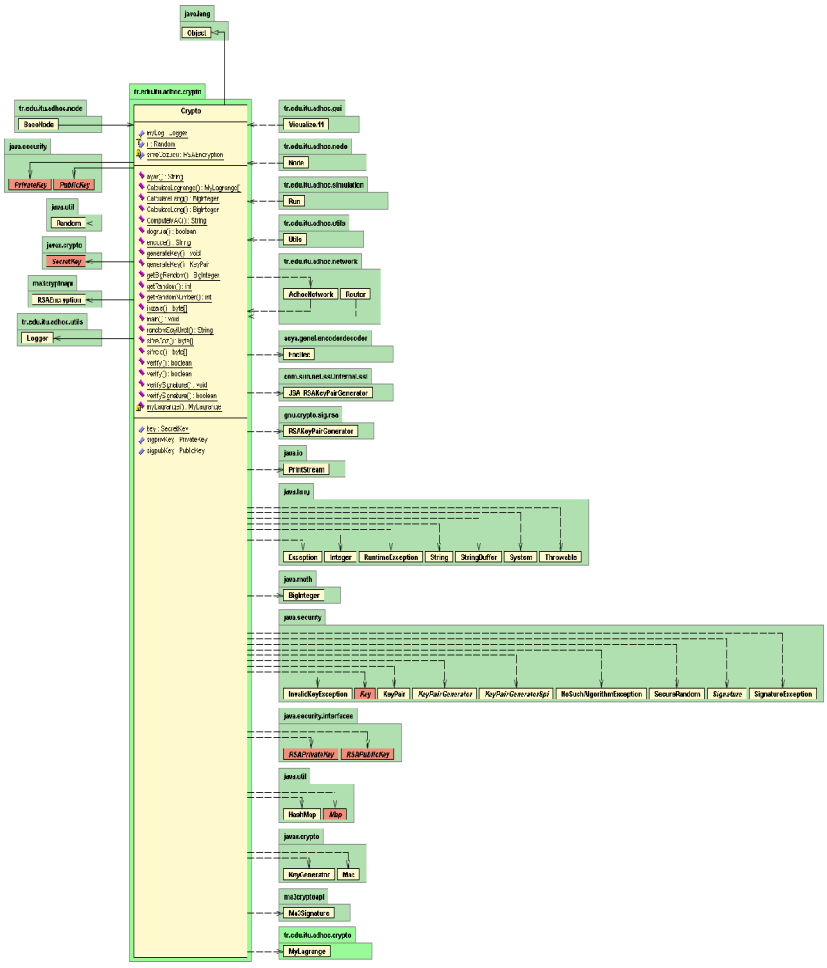

Figure A.4 Certificate Revocation Class

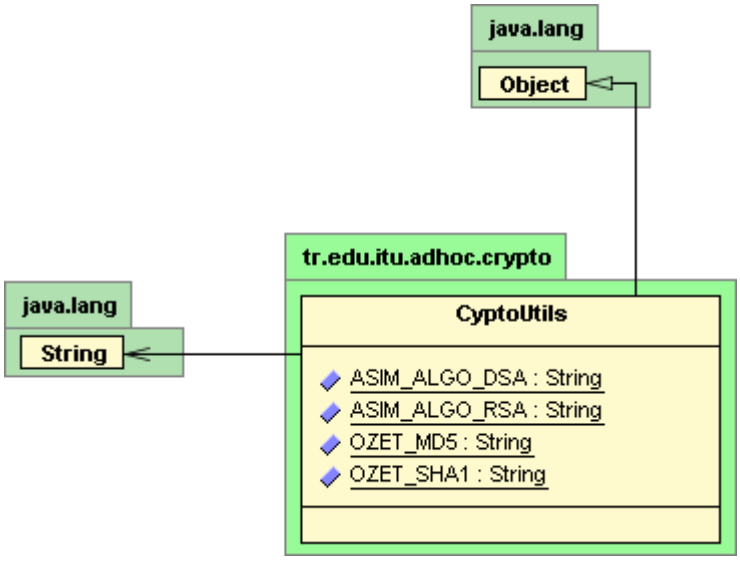

Figure A.5 Crypto Utils Class

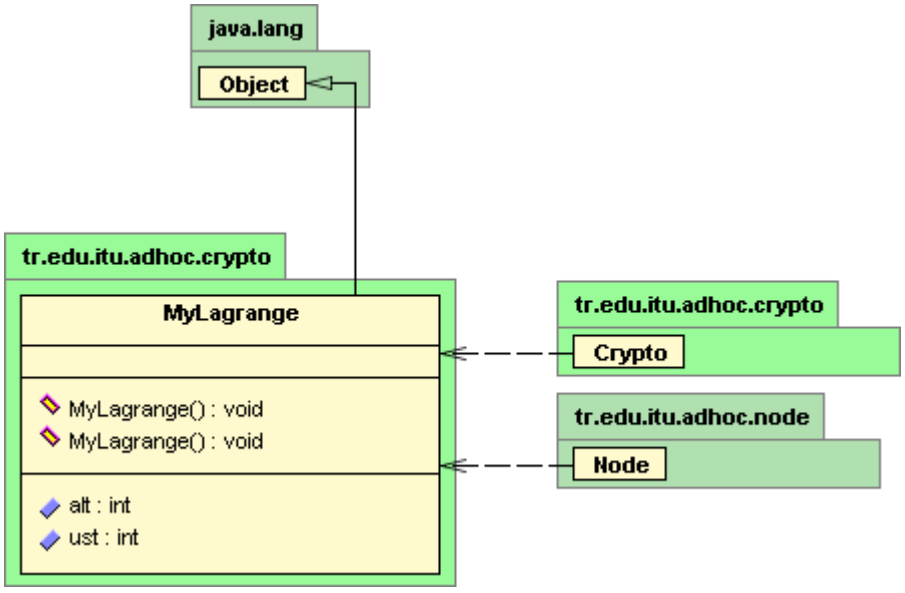

Figure A.6 MyLagrange Class

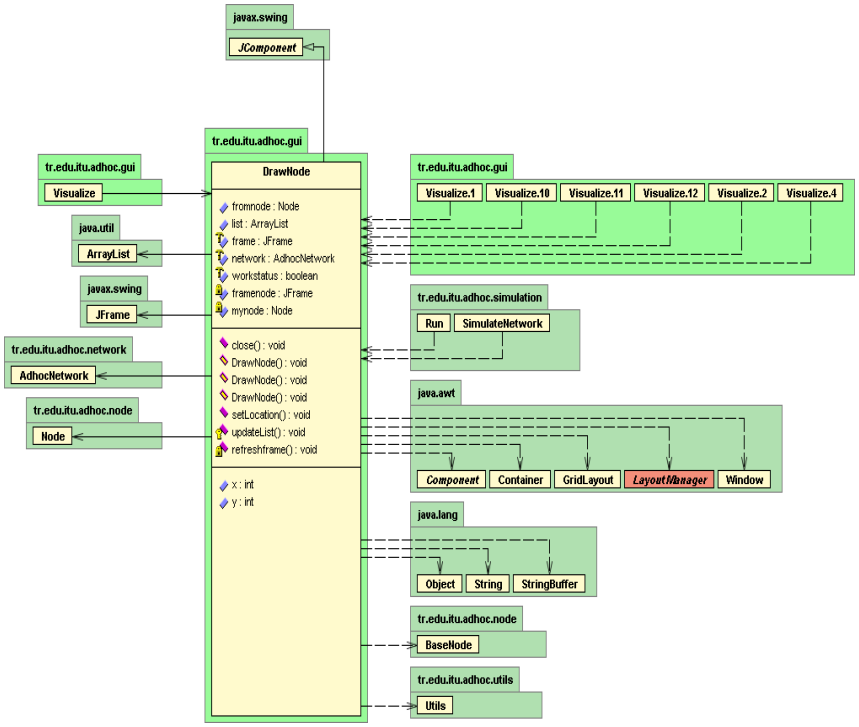

Figure A.7 Draw Node Class

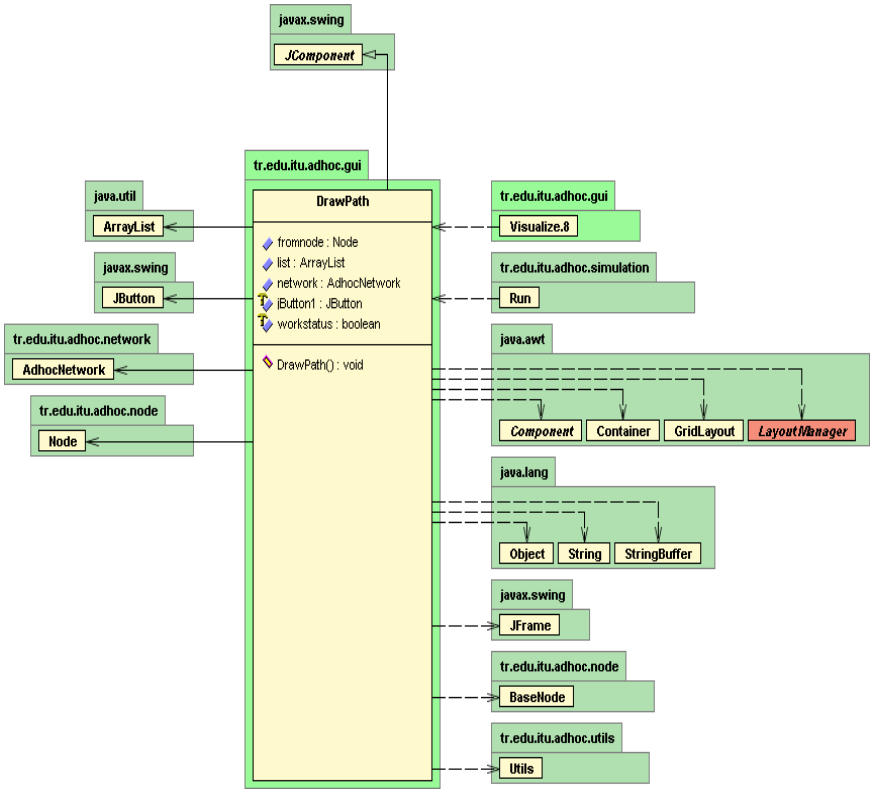

Figure A.8 Draw Path Class

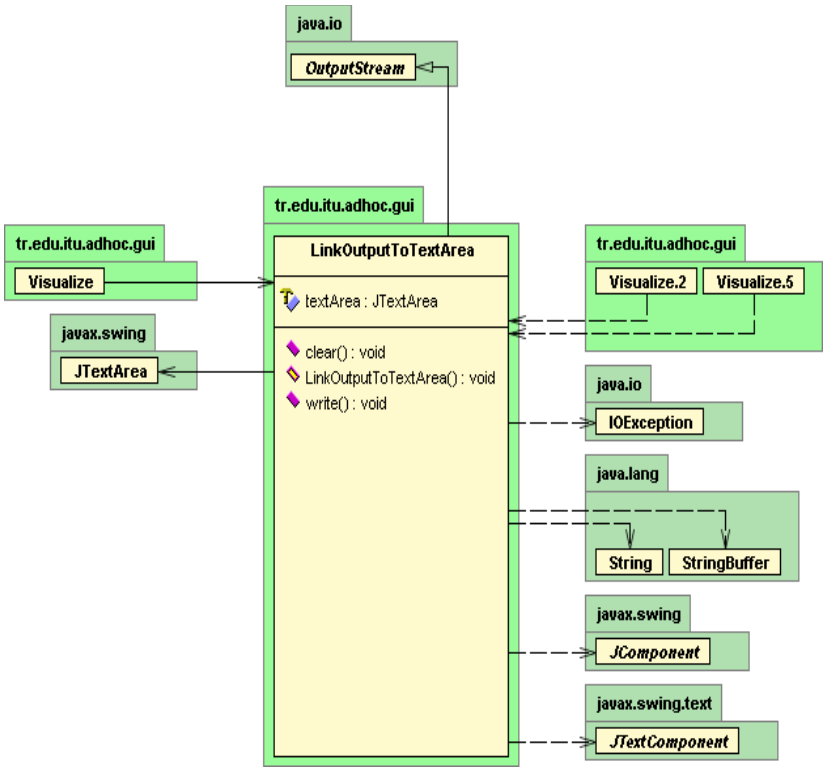

Figure A.9 Link Outputto TextArea Class

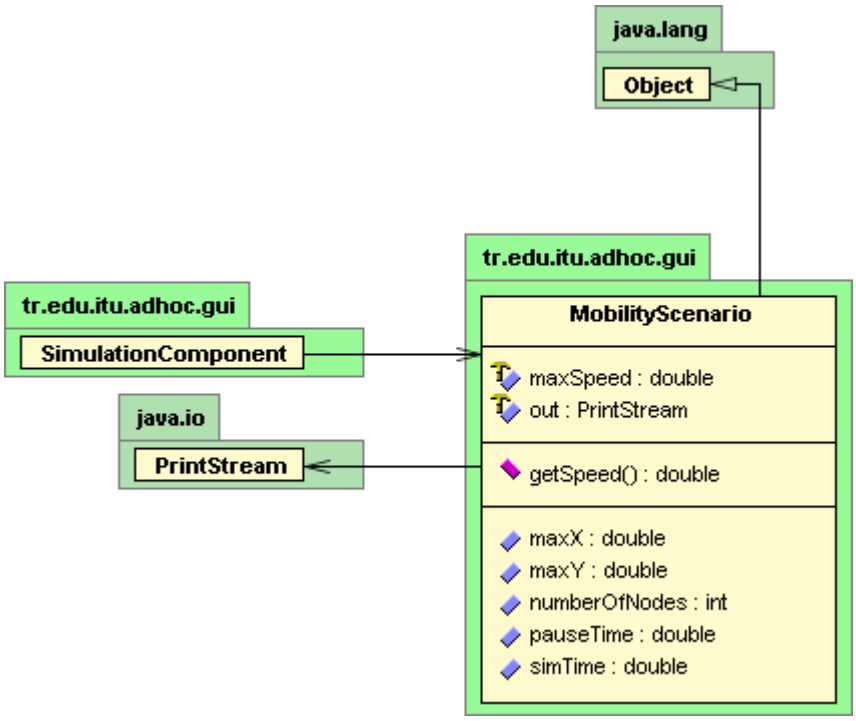

Figure A.10 Mobility Scenario Class

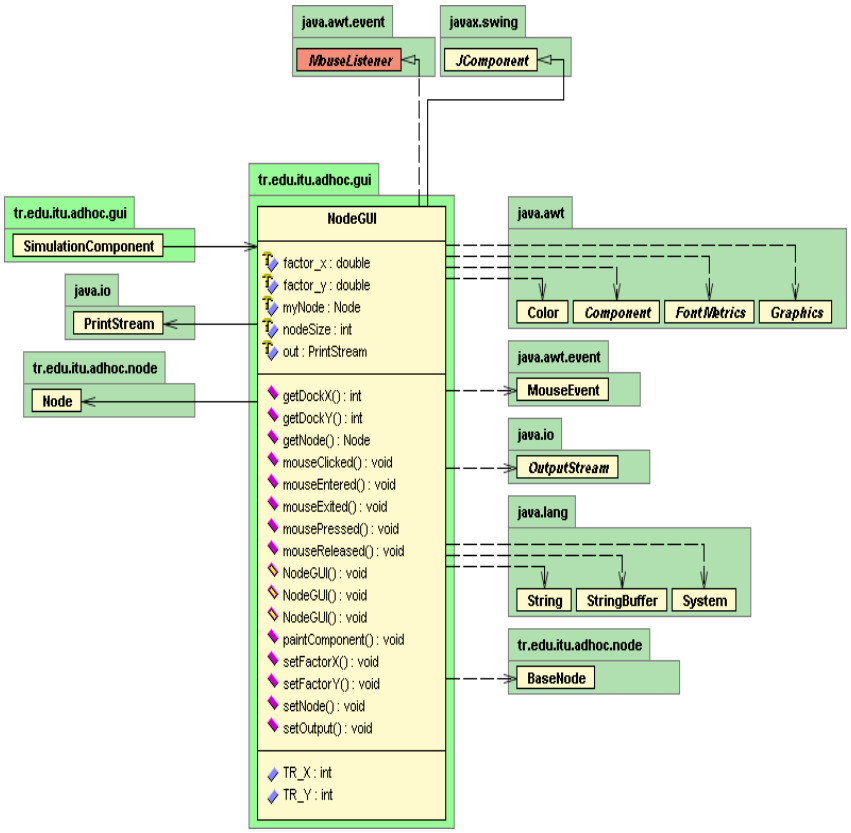

Figure A.11 Node GUI Class

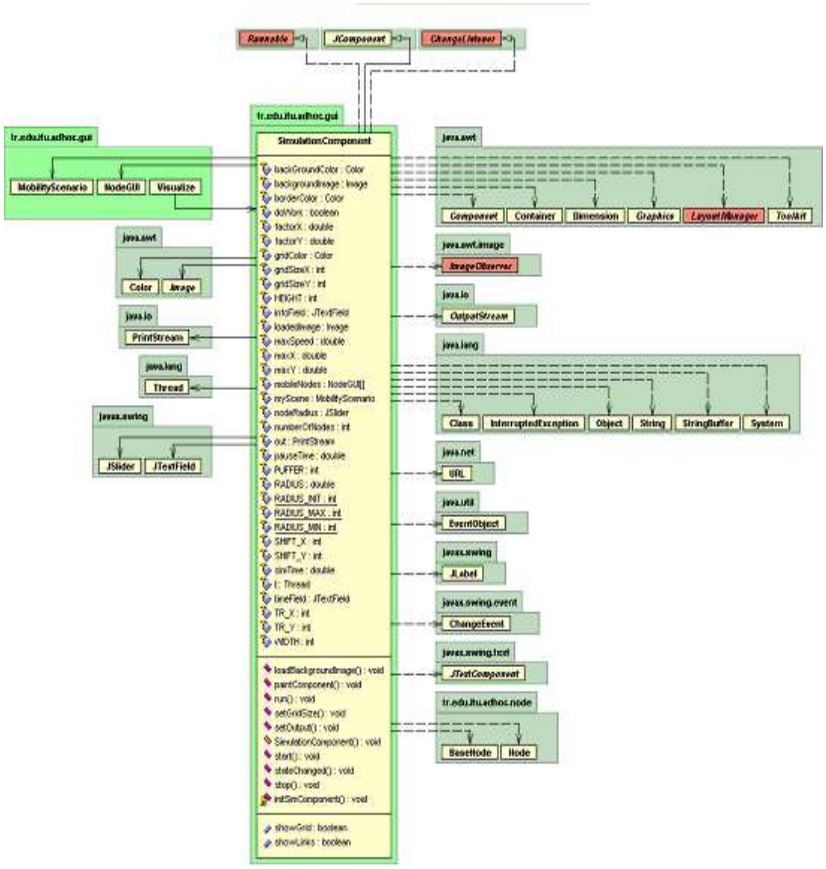

Figure A.12 Simulation Component Class

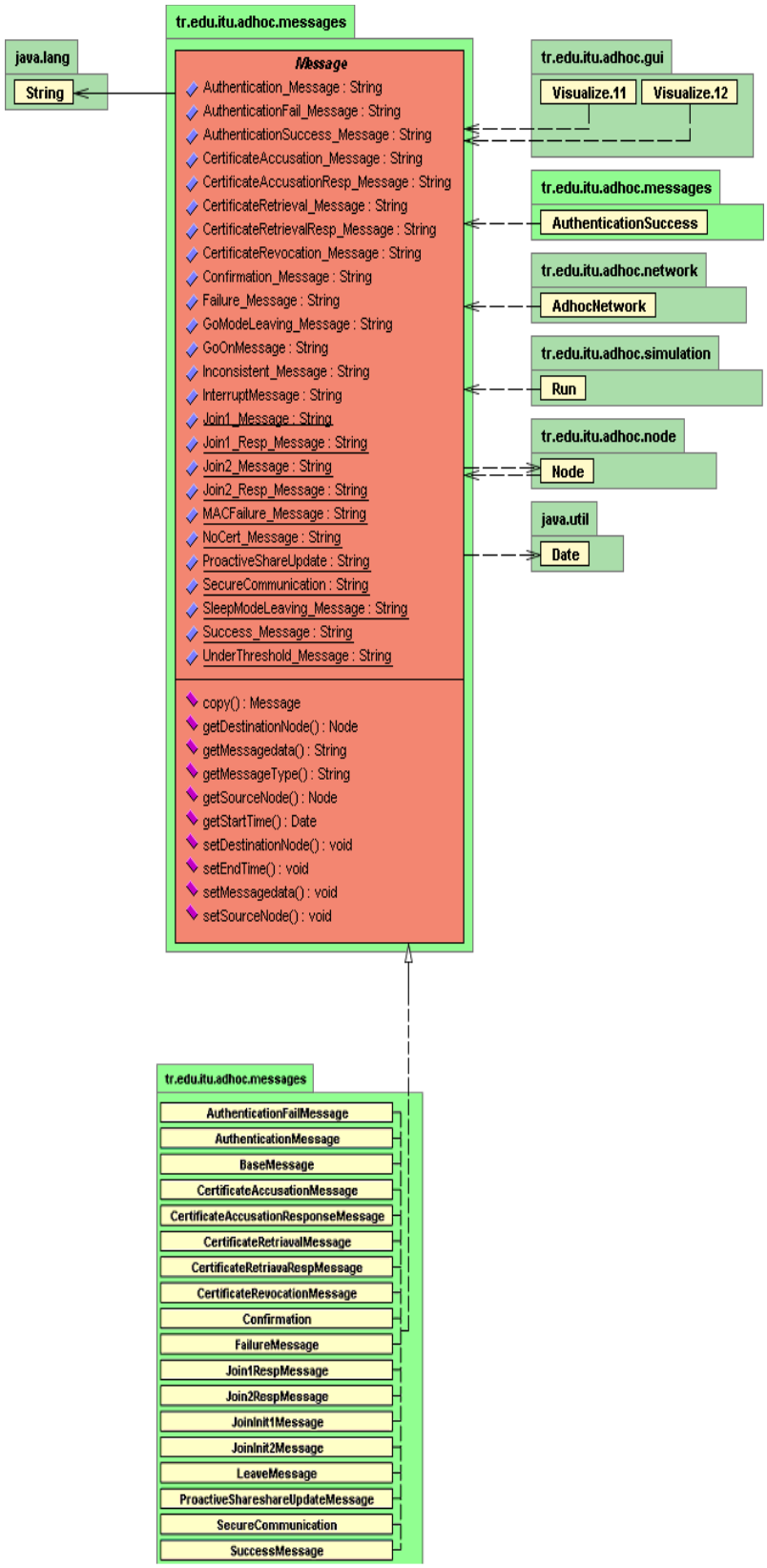
tr.edu.itu.adhoc.messages
java.lang
String
Message
Authentication_Message : String
AuthenticationFail_Message : String
AuthenticationSuccess_Message : String
CertificateAccusation_Message : String
CertificateAccusationResp_Message : String
CertificateRetrieval_Message : String
CertificateRetrievalResp_Message : String
CertificateRevocation_Message : String
Confirmation_Message : String
Failure_Message : String
GoModeLeaving_Message : String
GoOnMessage : String
Inconsistent_Message : String
InterruptMessage : String
Join1_Message : String
Join1_Resp_Message : String
Join2_Message : String
Join2_Resp_Message : String
MACFailure_Message : String
NoCert_Message : String
ProactiveShareUpdate : String
SecureCommunication : String
SleepModeLeaving_Message : String
Success_Message : String
UnderThreshold_Message : String
copy() : Message
getDestinationNode() : Node
getMessagedata() : String
getMessageType() : String
getSourceNode() : Node
getStartTime() : Date
setDestinationNode() : void
setEndTime() : void
setMessagedata() : void
setSourceNode() : void
tr.edu.itu.adhoc.gui
Visualize.11
Visualize.12
tr.edu.itu.adhoc.messages
AuthenticationSuccess
tr.edu.itu.adhoc.network
AdhocNetwork
tr.edu.itu.adhoc.simulation
Run
tr.edu.itu.adhoc.node
Node
java.util
Date
tr.edu.itu.adhoc.messages
AuthenticationFailMessage
AuthenticationMessage
BaseMessage
CertificateAccusationMessage
CertificateAccusationResponseMessage
CertificateRetriavalMessage
CertificateRetriavaRespMessage
CertificateRevocationMessage
Confirmation
FailureMessage
Join1RespMessage
Join2RespMessage
JoinInit1Message
JoinInit2Message
LeaveMessage
ProactiveSharesshareUpdateMessage
SecureCommunication
SuccessMessage

Figure A.13 Message Class

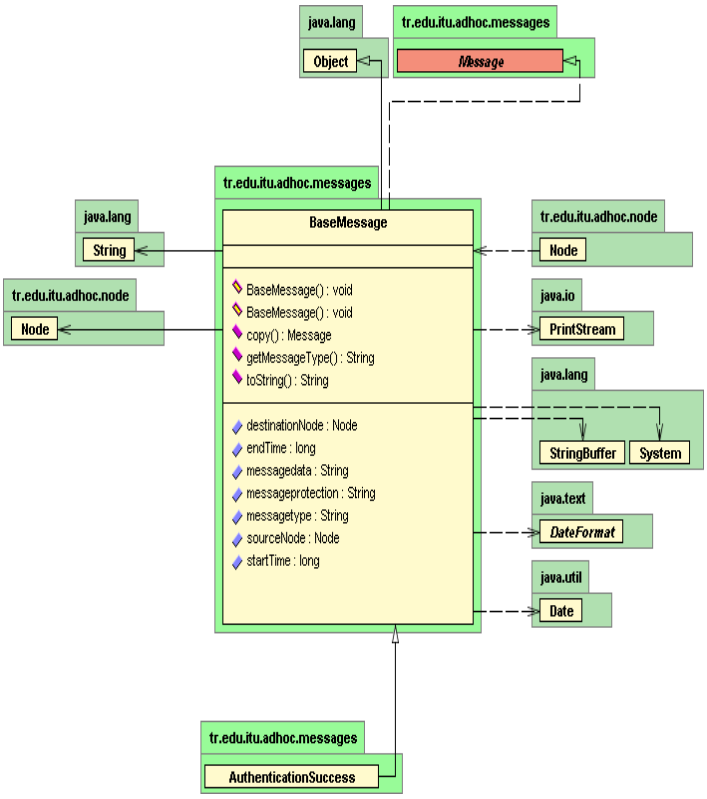

Figure A.14 Base Message Class

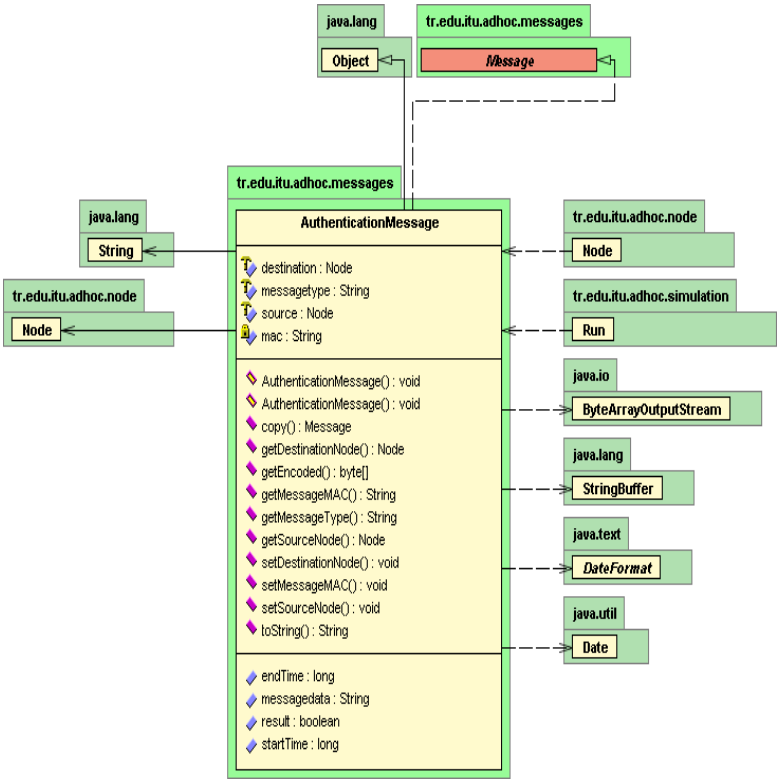

Figure A.15 Authentication Message Class

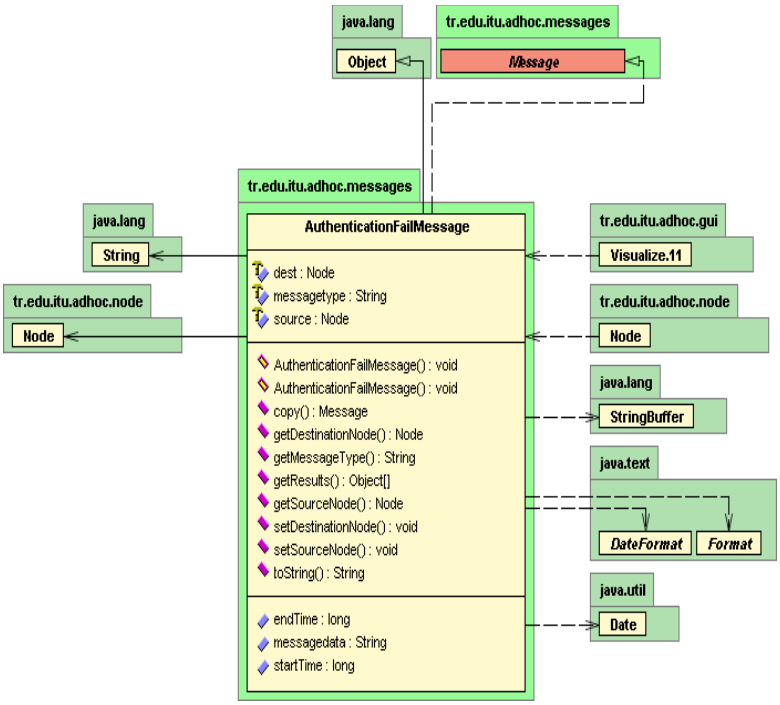

Figure A.16 Authentication Fail Message Class

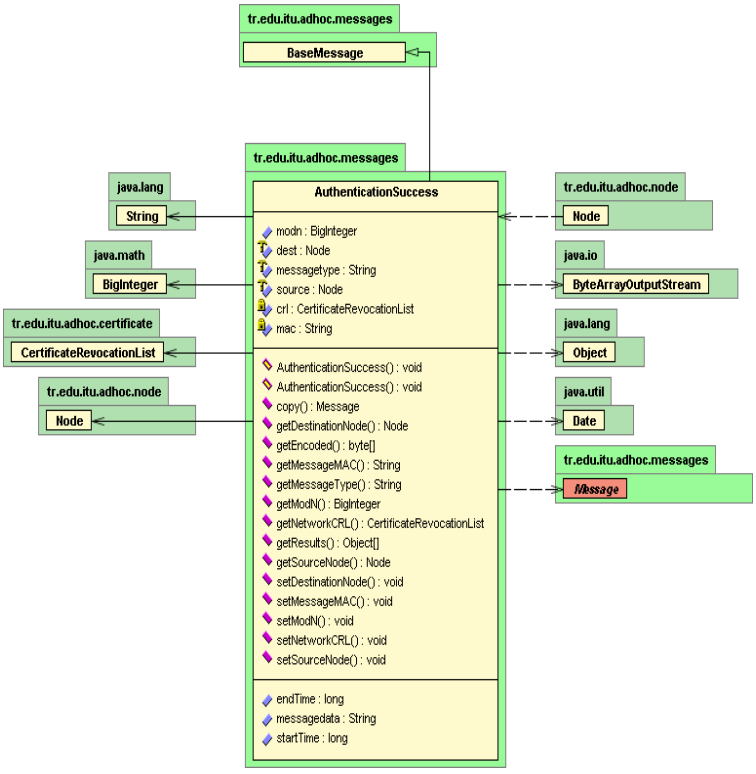

Figure A.17 Authentication Success Message Class

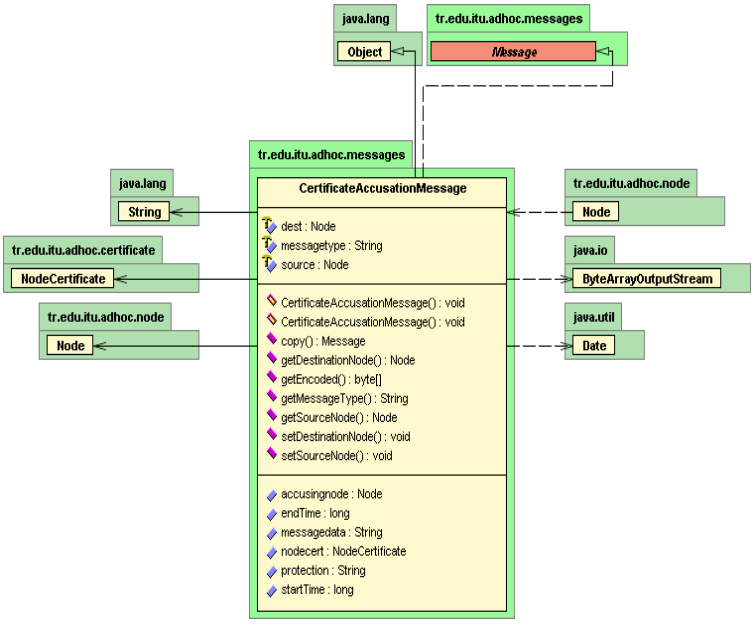

Figure A.18 Certificate Accusation Message Class

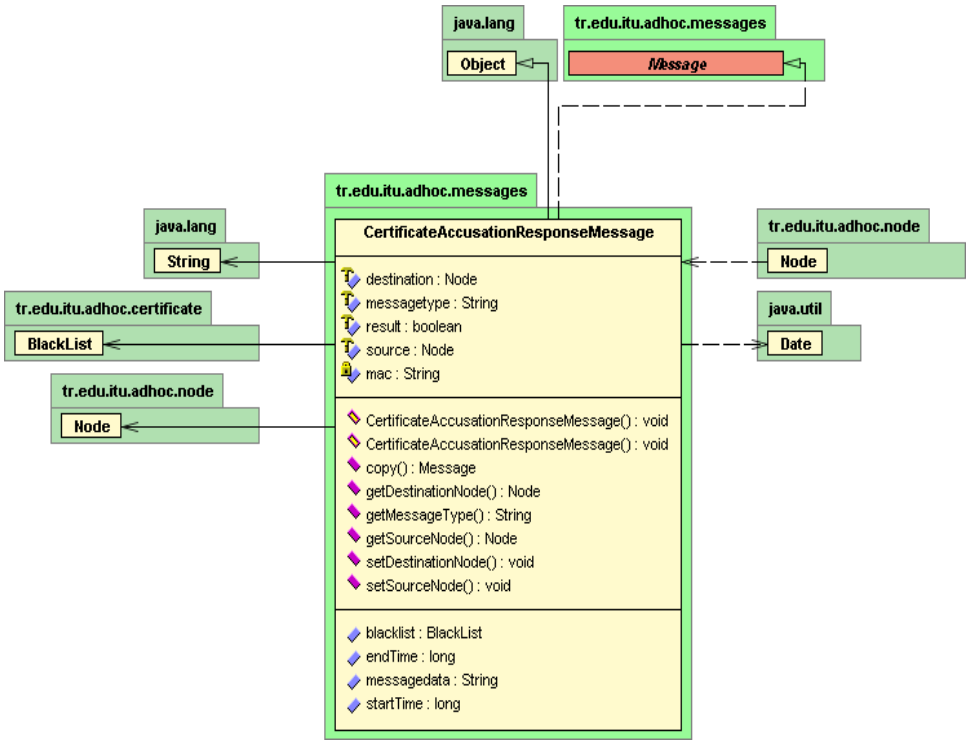

Figure A.19 Certificate Accusation Response Message Class

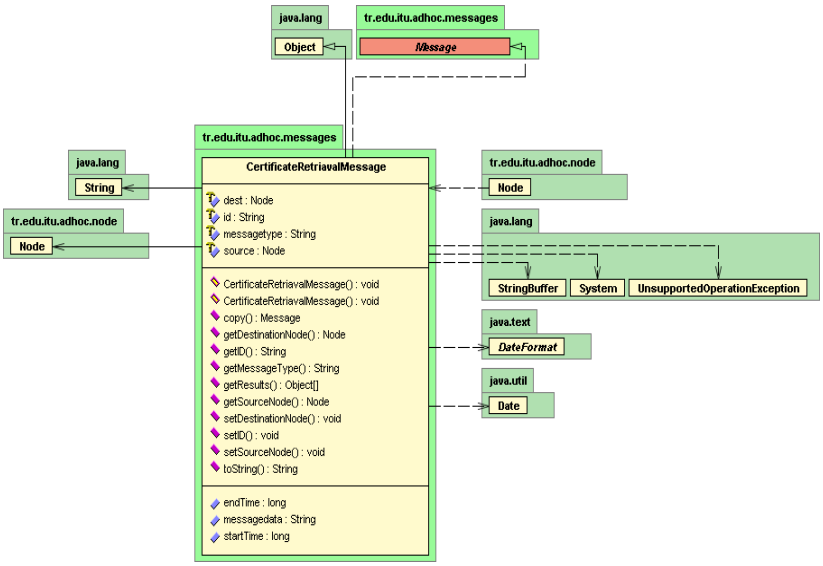

Figure A.20 Certificate Retrieval Message Class

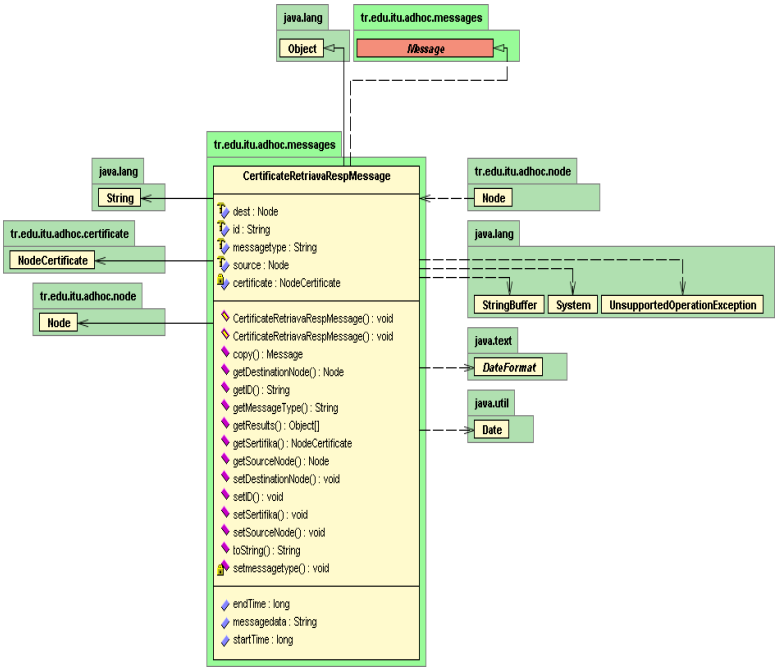

Figure A.21 Certificate Retrieval Response Message Class

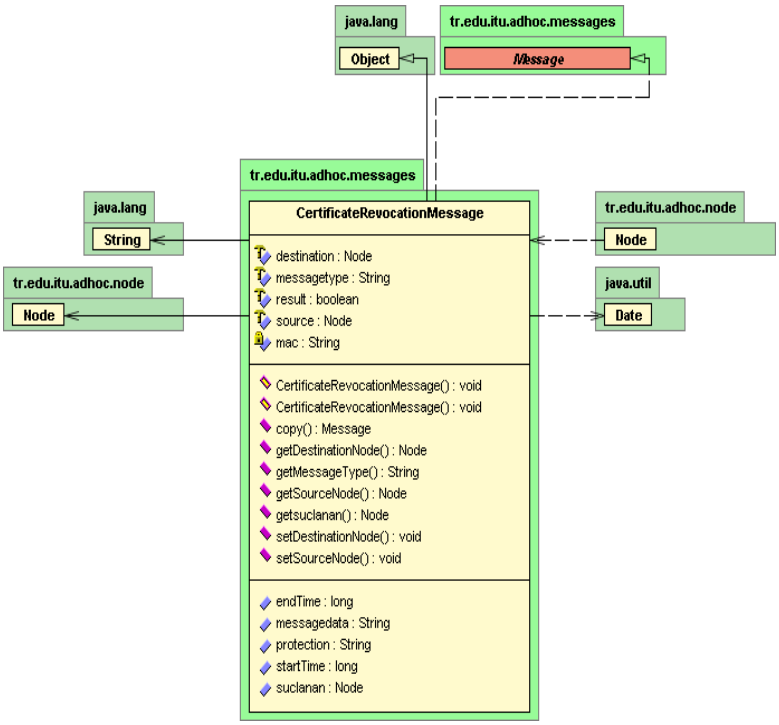

Figure A.22 Certificate Revocation Message Class

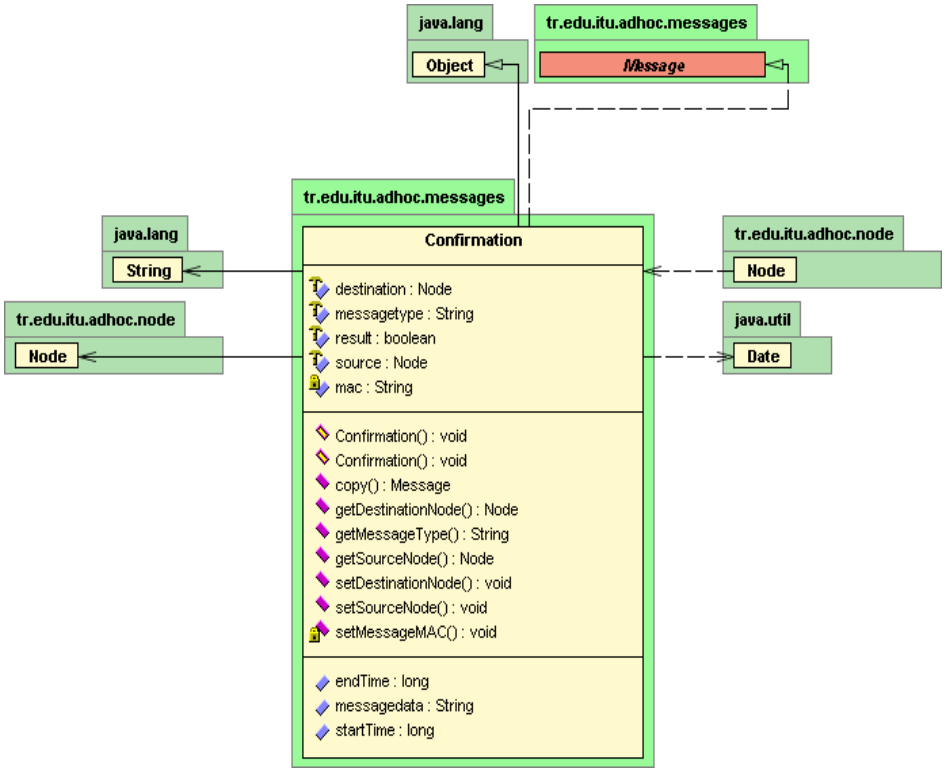

Figure A.23 Confirmation Class

AUTOBIOGRAPHY

I was born in Kırklareli at 27-January-1980. I graduated from Edirne Anatolia Teachers School at 1996 and started studying at İstanbul Technical University at same year. I graduated from the university at 2001 and I have been working in TUBITAK-UEKAE for 3 years.