# Secure Group Communications in Emergency Ad Hoc Networks

Raja Rai Singh Verma

A thesis submitted for the degree of
Doctor of Philosophy in Computer Science
University of Dublin, Trinity College
Department of Computer Science
December, 2005

# Declaration

I hereby declare that:

(a) This thesis has not been submitted as an exercise for a degree at this or any other University.

(b) This thesis is entirely the work of the author, except where otherwise stated.

(c) The Trinity College Library may lend and copy this thesis upon request.

Raja

Raja Rai Singh Verma

December, 2005

To my parents for all the years that
they have stood by me

# Acknowledgements

First and foremost I wish to express my sincere thanks and gratitude to my supervisor, Prof. Donal O'Mahony, Head of the Networks and Telecommunications Research Group (NTRG) here in Trinity College Dublin. Throughout my years of postgraduate research Donal ("Boss") has been a constant source of well-needed guidance, provocative discussions, and helpful advice. He always took the time, despite a busy schedule, to analyse my ideas and never failed to produce numerous suggestions for improvement. I must thank him for his patient reading of all the various versions of my write-ups and the resulting comments which have helped shape this dissertation. In addition, I am most grateful for the funding he arranged.

I would also like to thank the current and the previous members of the NTRG who offered me valuable support during my time with the group. In particular I would like to thank Hitesh Tewari who had the misfortune of having an office next door to mine, and whose whiteboard was used for the genesis of my ideas. I'm also grateful to my NTRG colleagues Stephen Toner and Patroklos G. Argyroudis for taking the pain of going through various drafts of this thesis. The efficient support and advice provided by the staff of the Department of Computer Science has been invaluable.

My landlady and her husband, Yvonne and Charles McQuigg were great in the hospitality extended to me during my stay with them. In particular they treated me like one of their own. Moreover I would also like to thank Dave and Mary Edmonds for the moral and transportation support when my motorcycle broke down at ungodly hours. I would also like to mention the Puri family who had the misfortune of me dropping in at odd hours to sample their excellent Indian cooking.

This acknowledgement is not complete without mention of Prof. G.V. Singh and Indu Singh without whose support I would not be where I'm today. And finally I would like to thank my family and friends for extending me never-ending support and encouragement.

# Abstract

Secure Group Communications in Emergency Ad Hoc Networks
Raja Rai Singh Verma
Supervisor: Prof. Donal O'Mahony
Networks and Telecommunications Research Group (NTRG)
Department of Computer Science

Ad Hoc networks are an ideal way to form interactions between mobile wireless nodes of emergency services from different domains (i.e. organizations or countries). However, due to the frequent topological and membership changes it is difficult to form trust relationships in an Ad Hoc environment. The existing approaches for building security relationships in Ad Hoc networks are either computationally intensive or require extensive initial coordination/pre-configuration of participating nodes. To address these shortcomings this thesis proposes the use of a flexible trust negotiation approach involving confidential and progressive exchange of digital certificates. This certificate exchange is governed by the local policies of the two participating nodes. A local policy on the service providing node maps the attributes from the certificates received into services it can provide to the remote node. A two-tier key formation scheme is also proposed to ensure the confidentiality of the trust negotiation process and to secure the communication paths.

The one-to-one trust negotiation concept is extended to provide access control for secure group collaboration formed between nodes of an Ad Hoc network deployed in an emergency situation. This collaboration focuses on the crucial issues of membership admission control and efficient group message propagation. A node seeking admission has to produce certificates which satisfy the admission policy. This policy is an expression of certificate attribute name/value pairs linked together using logical operators. Once a stranger meets the admission policy, it is given the current group key and attached to the group hierarchy. This group key is used by each member of the group to send encrypted messages to all other members. The nodes participating in the group are organized into a tree hierarchy where each node in the group is only aware of its parent or child, which need not be physically adjacent. It is the responsibility of the

underlying routing algorithm to find communication paths between nodes in the network. This separation of the logical structure of the group from the physical topology makes the group capable of withstanding frequent topological changes while still remaining scalable. Furthermore, the group messages propagate using the robust mesh topology instead of following the rigid group tree hierarchy.

# Related Publications

**Refereed Conferences**

[vot04] Verma, R., O'Mahony, D.& Tewari, H., Progressive Authentication in Ad Hoc Networks, in Proceedings of the Fifth European Wireless Conference, Barcelona, Spain, February 24-24, 2004, pp. 511-517.

[avo+04] Argyroudis, P., Verma, R., Tewari, H. & O'Mahony, D., Performance Analysis of Cryptographic Protocols on Handheld Devices, in Proceedings of 3rd IEEE International Symposium on Network Computing and Applications, Cambridge, MA, Aug 30 - Sept 1, pp 169-174, 2004.

**Workshops**

[vot03] Verma, R., Tewari, H., & O'Mahony, D., Wanderer: Secure Group Formation and Communication in Ad Hoc Networks, in Proceedings of the 2nd Joint IEI/IEE Telecommunications Systems Research Symposium, Dublin, May 6th 2003.

[vot01] Verma, R., O'Mahony, D. & Tewari, H., NTM- Progressive Trust Negotiation in Ad Hoc Networks, in Proceedings of the First Joint IEI/IEE Symposium on Telecommunications Systems Research, Dublin, November 27th, 2001, 8pp

# Table of Contents

**1 Introduction ........ 1**

1.1 Security in Ad Hoc Networks ........ 1

1.2 Motivation for the Research ........ 2

1.3 Trust Negotiation Approach ........ 2

1.4 Collaboration Scenarios ........ 4

1.5 Contributions ........ 7

1.5.1 Outline and Scope of the Thesis ........ 7

**2 Access Control and Key Establishment ........ 9**

2.1 Access Control ........ 9

2.1.1 Access Control in Convectional Networks ........ 10

2.1.1.1 Asymmetric Key Management ........ 12

2.1.1.2 Trust Management ........ 15

2.1.1.3 Trust Negotiation ........ 16

2.1.1.4 Conclusion of Access Control in Conventional Networks ........ 19

2.1.2 Access Control in Ad Hoc networks ........ 19

2.1.2.1 Threshold Certification Authorities ........ 19

2.1.2.2 Self-Organizing Public-Key Infrastructure ........ 21

2.1.2.3 Pre-arranged Shared Secret ........ 21

2.1.2.4 Hybrid Approaches ........ 22

2.1.2.5 Conclusion of Access Control in Ad Hoc networks ........ 23

2.2 Key Establishment Between Two Entities ........ 24

2.2.1 Key Agreement Protocols Between Two Entities ........ 24

2.2.1.1 General Requirements of a Key Agreement Protocol ........ 25

2.2.1.2 Performance Requirements of a Key Agreement Protocol ........ 27

2.2.1.3 Choice of Key Agreement Protocol ........ 28

2.2.1.4 Analysis of the Modified STS protocol ........ 34

2.2.2 Conclusion of the Key Establishment Between Two Entities ........ 39

**3 Secure Group Formation and Communication in Ad Hoc Networks ........ 40**

3.1 Secure Group Formation in Ad Hoc Networks ........ 40

3.1.1 Peer Group Admission Control ........ 41

3.1.2 Robust Membership Management for Ad-Hoc Groups ........ 42

3.1.3 Secure Multicasting in Conventional and Ad Hoc Networks ........ 43

3.1.3.1 Key Establishment Between Multiple Entities ........ 46

3.1.3.1.1 Diffie-Hellman Group Key Agreement ........ 47

3.1.3.1.2 Non Diffie-Hellman Group Key Agreement ........ 49

3.1.3.1.3 Group Key Transport Protocols ........ 50

3.1.4 Conclusion of Secure Group Formation in Ad Hoc networks ........ 51

3.2 Group Communications ........ 51

3.2.1 Overview of Multicasting in Conventional Networks ........ 51

3.2.2 Multicasting in Ad Hoc Networks ........ 53

3.2.2.1 Tree-based Multicasting Algorithms ........ 54

3.2.2.2 Mesh-based Multicasting Algorithms ........ 56

3.2.2.3 Flooding and Location based Multicasting ........ 58

3.2.2.3.1 Geocasting and Anycasting ........ 58

3.2.2.4 Reliable Multicasting Protocols ........ 60

3.2.2.4.1 Deterministic Protocols ........ 60

3.2.2.4.2 Probabilistic Protocols....61
3.2.2.5 Conclusion of Group Communications....62

**4 One-to-One Trust Negotiation....63**
4.1 Two-Tier Key Formation....63
4.2 Neighbour Trust Model (NTM) Layer....66
4.2.1 NTM Protocol....67
4.2.2 Timers in the NTM Layer....69
4.2.3 NTM Organization....69
4.2.4 State Machine of the NTM Protocol....71
4.2.5 NTM Layer Summary....73
4.3 Remote Trust Model (RTM) Layer....74
4.3.1 RTM Protocol....74
4.3.2 RTM Organization....78
4.3.3 State Machine of the Key Agreement in the RTM Protocol....80
4.3.4 Service Policy and Certificate Release Policy....83
4.3.5 Compliance Checker and Manual Intervention....85
4.3.6 RTM Layer Summary....87
4.4 Implementation....89
4.4.1 Layer Structure for One-to-One Trust Negotiation....89
4.4.2 Hardware Implementation....90
4.4.2.1 Timer Intervals in NTM Layer....94
4.4.2.1.1 Timer $_X$ Interval in NTM Layer....94
4.4.2.1.2 Timer $_{NT}$ Interval in NTM Layer....95
4.4.2.1.3 Timer $_{DT}$ interval in NTM layer....97
4.4.2.2 RTM Performance....98
4.4.2.2.1 Combined RTM and NTM Test....100
4.4.2.3 Encrypted Data Traffic Overheads....101
4.4.2.4 Key Storage Overheads....102
4.4.2.5 Implementation Summary....103
4.5 Conclusion of One-to-One Trust Negotiation....103

**5 Wanderer: Secure Group Formation....105**
5.1 Group Joining Protocol....107
5.2 Flexible Group Structure....112
5.3 Group Key Propagation....115
5.4 Group Message Propagation....117
5.5 Group Maintenance and Reorganization....119
5.5.1 Voluntary Group Leave....119
5.5.2 Involuntary Group Leave....121
5.5.3 Partitioning and Re-merger....123
5.5.4 Summary of the Wanderer Scheme....123
5.6 Implementation....124
5.6.1 Group Layer Structure....124
5.6.2 JEmu Emulator....125
5.6.3 Actual Implementation....128
5.6.3.1 Joint Implementation....130
5.6.4 Emulation....131
5.6.5 Implementation Summary....134
5.7 Conclusion....134

**6 Conclusions and Future Directions ........ 136**
6.1 Summary of Contributions ........ 136
6.2 Directions for Future Research ........ 138

**Appendix A Cryptographic Benchmarks ........ 140**
A.1 Test Environment ........ 140
A.2 RSA Key Generation ........ 140
A.3 RSA signature generation and verification ........ 141
A.4 Symmetric ciphers and message digests ........ 142
A.5 Related work ........ 144

**Appendix B Policy Language ........ 145**
B.1 Language Specification ........ 145
B.2 Usage of the policy language ........ 146
B.2.1 Service Policy ........ 146
B.2.2 Certificate Release Policy ........ 146
B.2.3 Membership Criteria ........ 146
B.3 Language Suitability ........ 146
B.4 Compliance checkers ........ 147
B.4.1 Compliance Checker used in the One-to-One Trust negotiation Scheme ........ 148
B.4.2 Compliance Checker used in the Wanderer Scheme ........ 149

**Appendix C Code Fragments ........ 151**
C.1 Adding and extracting custom attribute into a X.509 certificate. ........ 151
C.2 Creation and sending of an “Advertise” network message ........ 153
C.3 Receiving and deciphering of network messages by the group layer ........ 155

**Bibliography ........ 160**

**Table of Figures**

Figure 1 : Credentials Required for Trust Building in Diverse Scenarios........................3

Figure 2 : Scenario Involving One-to-one Collaborations...............................................5

Figure 3 : Access Graph of the Restricted Data ..............................................................5

Figure 4 : Group Interactions in a Multi-National Coalition ...........................................6

Figure 5 : Symmetric and Asymmetric Paradigms.........................................................11

Figure 6 : Certificate Chain in Hierarchical CA's...........................................................13

Figure 7 : PGP's Web of Trust........................................................................................14

Figure 8 : Architecture of the Keynote System..............................................................15

Figure 9 : Key Management using Threshold Cryptography.........................................20

Figure 10 : Classification of Key Establishment Protocols............................................24

Figure 11 : Comparison of Diffie-Hellman Key Agreement Protocols..........................29

Figure 12 : Station-to-Station (STS) Protocol................................................................31

Figure 13 : Modified STS Protocol................................................................................34

Figure 14 : State Diagram of the Modified STS protocol..............................................35

Figure 15 : Threat Tree of the Modified STS protocol..................................................37

Figure 16 : Key Distribution using a Tree Structure .....................................................44

Figure 17 : Perrig and Octopus Group Key Agreement Protocol ..................................47

Figure 18 : Performance Characterises of Generalized Diffie-Hellman based Protocols 48

Figure 19 : Boyd's Group Key Agreement .....................................................................50

Figure 20 : Flood and Pruning of Routes in Multicasting Algorithms ...........................52

Figure 21 : Classification of the Multicasting Algorithms in Ad Hoc networks.............54

Figure 22 : Geocasting ...................................................................................................59

Figure 23 : Two Tier-Key Formation ............................................................................64

Figure 24 : Example of the Two-Tier Key Formation....................................................65

Figure 25 : NTM Protocol..............................................................................................68

Figure 26 : NTM Organization.......................................................................................70

Figure 27 : State Diagram of the NTM Protocol............................................................71

Figure 28 : Relationship between the States and Data Structures in the NTM Layer ......73

Figure 29 : RTM Trust Negotiation Protocol..................................................................76

Figure 30 : Identity Shielding.........................................................................................78

Figure 31 : RTM Data Structure Organization................................................................79

Figure 32 : Data Structures in the Certificate Exchange Agent (CEA)...........................80

Figure 33 : State Machine of the RTM Protocol (Phase 1)........................................81

Figure 34 : Relationships between the States and Data Structures of CEA....................83

Figure 35 : Service Policy for Access to Services on the Drug Units Node...................84

Figure 36 : Compliance Checker for the One-to-One Interaction (Service Provider)......86

Figure 37 : Compliance Checker for the One-to-One interaction (Service Requester)....87

Figure 38 : Generalized NTRG Stack..............................................................................89

Figure 39 : Layer Structure for One-to-One Trust Negotiation .....................................90

Figure 40 : Performance Tests of the NTM Protocol .....................................................91

Figure 41 : OpenSSL Dump of the X.509 Address Certificate.......................................93

Figure 42 : Major Factors in Determination of the Timer Interval ................................94

Figure 43 : NTM Protocol Key Agreement Performance...............................................96

Figure 44 : Performance of Phase 1 of the NTM protocol (Dynamic Membership)........96

Figure 45 : NTM Protocol Phase 1 Performance (Incremental Dynamic Membership) ..97

Figure 46 : Sample RTM Trust Negotiation Protocol .....................................................99

Figure 47 : Screen Shot of the One-to-One Trust Negotiation Prototype........................99

Figure 48 : Performance of the One-to-One Trust Negotiation ....................................100

Figure 49 : Encrypted Data Traffic Overheads .............................................................101

Figure 50 : Overheads for Message Delivery.................................................................102

Figure 51 : Multiple Groups and One-to-One Interactions............................................105

Figure 52 : Group Membership Using Trust Negotiation...............................................106

Figure 53 : Group Joining Protocol ...............................................................................108

Figure 54 : Compliance Checker for Wanderer (Leader) ..............................................110

Figure 55 : Compliance Checker for Wanderer (Stranger).............................................111

Figure 56 : Different Types of Nodes in Wanderer........................................................112

Figure 57 : Sub-Leader Appointment Protocol .............................................................113

Figure 58 : Logical and Physical Structure of a Group .................................................114

Figure 59 : Non-member Relaying the Join Request......................................................114

Figure 60 : Group Key Propagation Using TEK's ..........................................................116

Figure 61 : Group Key Update Protocol.........................................................................117

Figure 62 : Group Message Propagation Using Mesh Topology ...................................118

Figure 63 : Group Messaging Protocol...........................................................................119

Figure 64 : Voluntary Leave of a Sub-Leader.................................................................120

Figure 65 : Voluntary Leave of the Leader.....................................................................121

Figure 66 : Wanderer Layer Structure ...........................................................................125

Figure 67 : JEmu Emulation Architecture ........................................................... 126

Figure 68 : JEmu Emulation Algorithm ............................................................... 127

Figure 69 : Sample Join Process in Wanderer ...................................................... 129

Figure 70 : Screen Shot of the Wanderer ............................................................ 129

Figure 71 : Performance of the Group Joining Protocol ......................................... 130

Figure 72 : Implementation of the Wanderer with the One-to-One Trust Negotaition .. 130

Figure 73 : Performance of the Joint Implementation ............................................ 131

Figure 74 : Logical Structure of the Emulated Group ............................................ 132

Figure 75 : Screen Shot at the Start of the Emulation ............................................ 132

Figure 76 : Emulation Results ............................................................................ 133

Figure 77 : RSA Key Generation Benchmarks ....................................................... 141

Figure 78 : RSA Signature Generation and Verification Benchmarks ....................... 142

Figure 79 : Benchmarks of Symmetric Ciphers and Message Digests ........................ 143

Figure 80 : AES Benchmarks ............................................................................. 143

Figure 81 : Comparison of Five Policy Languages ................................................. 147

Figure 82 : Windows Dump of the *Address* Certificate ......................................... 152

# 1 Introduction

## *1.1 Security in Ad Hoc Networks*

Mobile Ad hoc Networking (MANET) [manet] is a new paradigm allowing interaction between mobile wireless devices. Unlike conventional wireless networks, an Ad Hoc network has little or no dependence on fixed infrastructure to maintain connectivity between the participating nodes (i.e. the mobile devices). Instead the nodes rely on each other to keep the network connected despite topological and membership changes. Hence the deployment of an Ad Hoc network is simple and quick. This ease of deployment makes the use of Ad Hoc networks ideal in a civilian or military emergency scenario. Some of these scenarios may require security against attackers. However the characteristics of Ad Hoc networks which makes their usage ideal in emergency scenarios also gives rise to some unique security problems. The traditional security mechanisms for conventional wired and wireless networks generally rely on access to an online Trusted Third Party (TTP). This access to an online TTP cannot be guaranteed at all times in an Ad Hoc network due to its dynamic topology and membership. Furthermore, the use of wireless as a preferred medium of communications leaves the Ad Hoc network open to *active* and *passive* attacks [ao03, k00, sa99, zh99]. An active attacker interferes with communications either by impersonating a node, causing message modification, replaying messages or just denying access to genuine nodes. In contrast a passive attacker just eavesdrops and then tries to compromise the confidentiality of the communications. Moreover, the unreliability of the wireless medium makes it difficult to distinguish between a deliberate attack and an unintentional network disruption. Some traditional security schemes for conventional wired and wireless networks frequently use public-key cryptographic operations. Such schemes are not suitable for the CPU and battery constrained mobile wireless devices which tend to form the bulk of nodes participating in an Ad Hoc network. Thus traditional security mechanisms for conventional wired and wireless networks are unsuitable for used in an Ad Hoc network and have motivated the researchers to develop new security schemes [bh03, ddg+01, hbc01, klx+02, kmt03, mah00, zh99].

## *1.2 Motivation for the Research*

The security schemes developed for Ad Hoc networks take three main approaches. In the first approach, the participating nodes are pre-configured with an initial shared secret [bhr+01, cps03, eg02]. The second approach requires extensive coordination between the pre-configured participating nodes [ddg+01, kka03, klx+02, ldo03, lzk+02, yk03, zh99] for operation of the security scheme. In the security schemes using pre-configuration, the initial setup time of the Ad Hoc network is minimized. However, the pre-configuration limits scalability and also introduces a single weak point (i.e. the initial common shared secret) on whose compromise the entire security of the network can unravel. The drawback of schemes requiring extensive coordination is that they lack reliability in cases of link and route failures. Such failures can occur frequently in a large or mobile Ad hoc network. In the third approach, the security mechanism for the participating nodes is based on the frequent issue of public-key certificates [klx+02, kmt03, mah00]. This approach can result in big computational overheads (see Appendix A) and is unsuitable for the CPU and battery constrained mobile wireless devices which form the bulk of nodes in an Ad Hoc network. A common drawback of the security schemes using the above mentioned approaches is that they lack the flexibility to allow the security to vary from lax to strict, depending on the requirement of the scenario. For example, an Ad Hoc network setup for playing a multi-player fun game has lower security requirement compared to a military coalition scenario.

Therefore for an Ad Hoc network deployed in an emergency scenario there is a requirement for security solutions having minimal pre-configuration, little coordination and the ability to vary the security requirements. Moreover, the security solutions targeting this environment should make minimal use of computationally heavy cryptographic primitives. This motivated the development of security solutions in this thesis addressing these requirements utilizing the trust negotiation approach [trust, w03, wsj00] as the underlying method of building trust among the participating nodes.

## *1.3 Trust Negotiation Approach*

An example of the trust negotiation approach in the real world is where two strangers exchange paper credentials to build trust to perform some task or access some service/resource. The scenario shown in Figure 1, demonstrates the usage of paper credentials by a person to build trust in different situations. In this example, the person generally releases all his credentials eagerly but, in his home country the person may have a three step negotiation, with the traffic police also showing its credentials. The same paper credential plays different roles for the individual depending on the situation mainly due to the credential release policies of the credential owner and requester. These policies also ensure that each trust building process is different despite the use of the same paper credentials. The use of this approach in an Ad Hoc network enables the participating nodes to make their own decision on the trustworthiness of a remote node. The level of trustworthiness can then be used to determine the access permissions the remote node grants to its services/resources. In this approach the pre-configuration of nodes is minimal as a certificate can be used in different roles in diverse scenarios and areas of operation. Moreover since the trust negotiation between two nodes happens on demand, the coordination required to build trust is minimized. The use of policies gives the trust negotiations approach the flexibility to vary the security requirement from lax to strict, depending on the scenario.

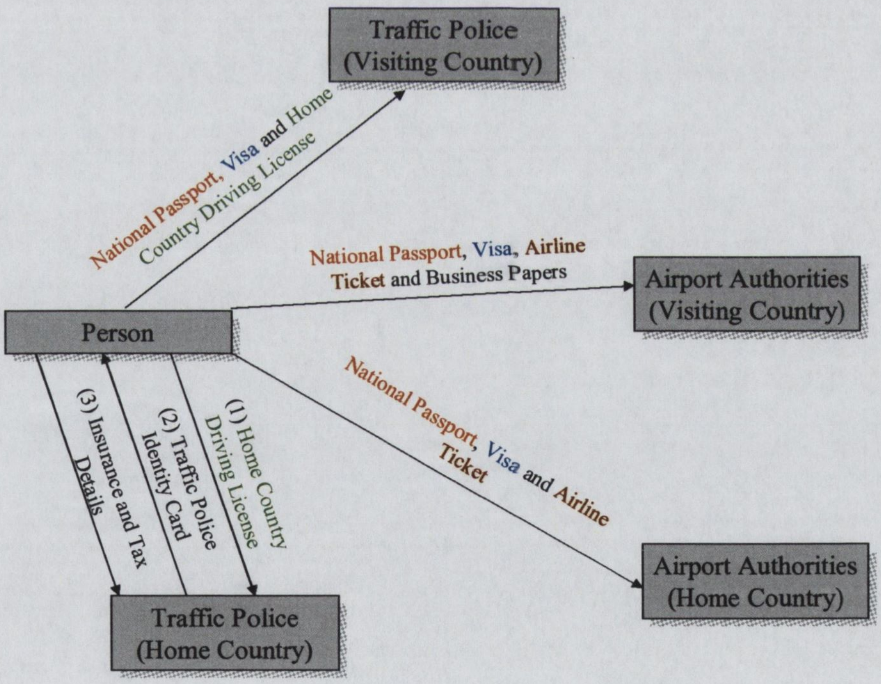

Traffic Police (Visiting Country)
National Passport, Visa and Home Country Driving License
National Passport, Visa, Airline Ticket and Business Papers
Airport Authorities (Visiting Country)
Person
(3) Insurance and Tax Details
(2) Traffic Police Identity Card
(1) Home Country Driving License
National Passport, Visa and Airline Ticket
Airport Authorities (Home Country)
Traffic Police (Home Country)

**Figure 1 : Credentials Required for Trust Building in Diverse Scenarios**

The use of trust negotiation process in conventional wired computer networks was first proposed by Winsborough et al [wsj00]. Their process involves the progressive interchange of digital certificates containing custom attribute name/value pair(s). The use of custom attributes has two advantages. Firstly, the TTP can certify the extra custom attributes embedded in the certificate. Secondly, the provision to embed custom data implies that the abstract paper credentials can be converted easily into digital certificates. A node taking part in a trust negotiation process releases the local certificates to the remote node based on the local policies and the remote certificates received. Any node can ask for a certificate from the remote node by referring to the certificate's embedded attribute name(s) or attribute name/value pair(s). A policy then translates the received and verified attribute name/value pair(s) into access to services provided by the node. The actual trust negotiation protocol [hjm+02] for conventional wired network is an extension of the Transport Layer Security (TLS) [da99] protocol. However, the trust negotiation between two conventional fixed wireless devices happens in a surrogate manner [hss+04]. In this surrogate technique instead of the two wireless nodes negotiating trust directly, their respective agent nodes on the fixed network negotiate trust and then convey the result to the wireless nodes.

In this thesis the trust negotiation approach is used to build secure collaborations between the participating entities of an Ad Hoc network. The outline of these secure collaborations is presented in the next section.

## *1.4 Collaboration Scenarios*

This thesis proposes two types of secure collaborations for nodes of an Ad Hoc network deployed in an emergency scenario. The first is a one-to-one trust negotiation scheme between two nodes. In this scheme, the new trust negotiation protocol (Chapter 4) is suited for CPU constrained mobile handheld wireless devices unlike the complex, time consuming and fixed-network centric design of well known trust negotiation protocols [hjm+02, hss+04]. The one-to-one interactions are particularly useful in the emergency scenarios like the one outlined in Figure 2. In this scenario several mobile nodes from different emergency services converge in a crisis situation. The first on the spot are the police patrol cars. They assess the situation and decide to call for medical assistance and specialist police units. These units arrive along with some from the media. These diverse units form a wireless Ad Hoc network to maximize connectivity at the crisis site. Some of the units/nodes may encrypt communications as they may have to route data through units that they do not trust (e.g. media unit).

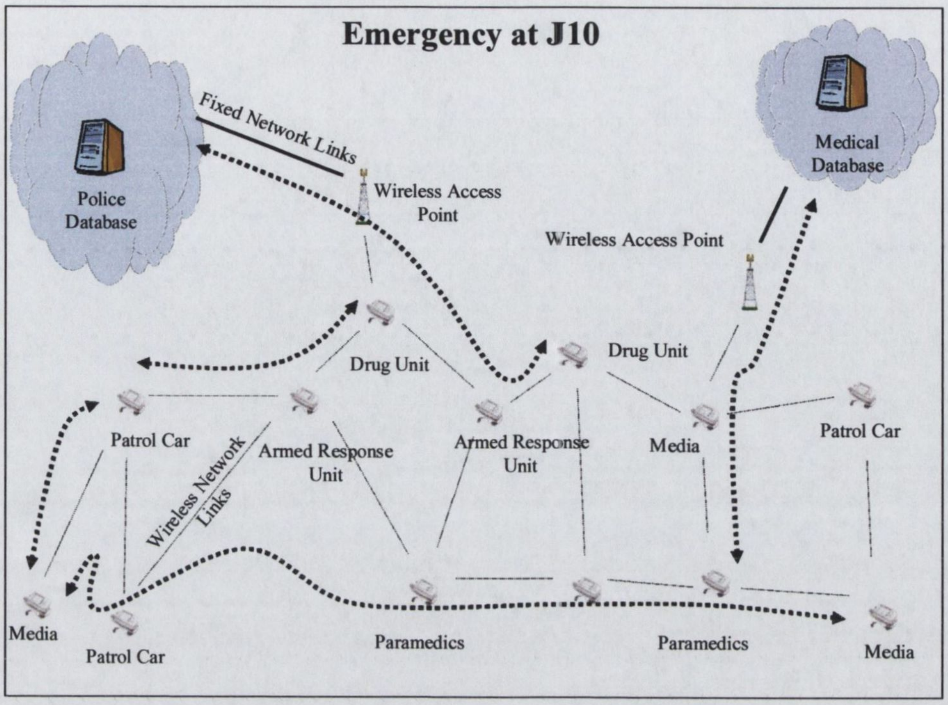
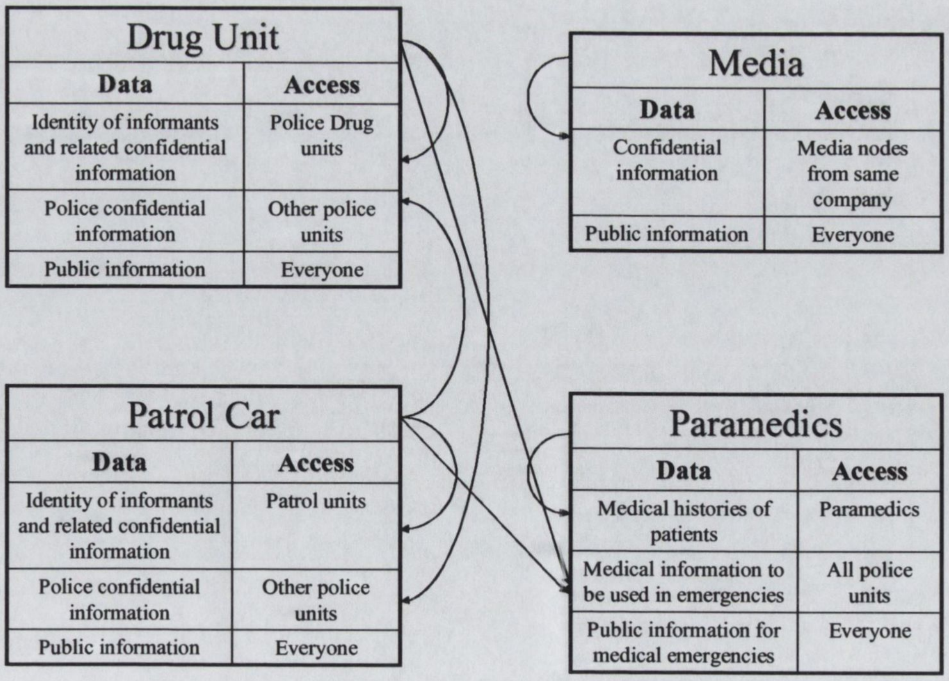


**Figure 2 : Scenario Involving One-to-one Collaborations**

In the scenario illustrated in Figure 2 it is best left to each node in the Ad Hoc network to decide what information it can share with the remote node after a trust negotiation. An illustrative mapping between the types of information on each node (in Figure 2) and the remote node allowed to access the information is shown in Figure 3.



Drug Unit

| Data | Access |
|---|---|
| Identity of informants and related confidential information | Police Drug units |
| Police confidential information | Other police units |
| Public information | Everyone |

Media

| Data | Access |
|---|---|
| Confidential information | Media nodes from same company |
| Public information | Everyone |

Patrol Car

| Data | Access |
|---|---|
| Identity of informants and related confidential information | Patrol units |
| Police confidential information | Other police units |
| Public information | Everyone |

Paramedics

| Data | Access |
|---|---|
| Medical histories of patients | Paramedics |
| Medical information to be used in emergencies | All police units |
| Public information for medical emergencies | Everyone |

**Figure 3 : Access Graph of the Restricted Data**

The second proposed secure collaboration is of the group variety. Such secure group collaborations are formed when a large number of nodes come together to perform some common task or share some resources. The lack of scalability of the one-to-one trust negotiation scheme prevents it from being used as is for secure group collaborations. For example if *n* is the number of nodes then *n*(n-1)/2* one-to-one interactions are required. Moreover each of the *n*(n-1)/2* interactions require cryptographic interchanges. Therefore on a CPU constrained mobile wireless device it would not be feasible to support secure group collaborations using the one-to-one trust negotiation scheme. A secure group collaboration involving a multi-national coalition scenario is illustrated in Figure 4. Each country in the coalition has their national group for coordination and application sharing. At the same time a participating node has to trust nodes from other countries for certain defined purposes. These initial four national groups will evolve into many more groups depending on the tasking of the nodes. A node may be part of many groups, ranging from the highest security national groups to medium security task groups to low security groups for general information exchange. The new entrant should have sufficient credentials to get it admitted to the intended group. At the same time the new entrant should be satisfied that it is not being duped into joining some untrustworthy group.

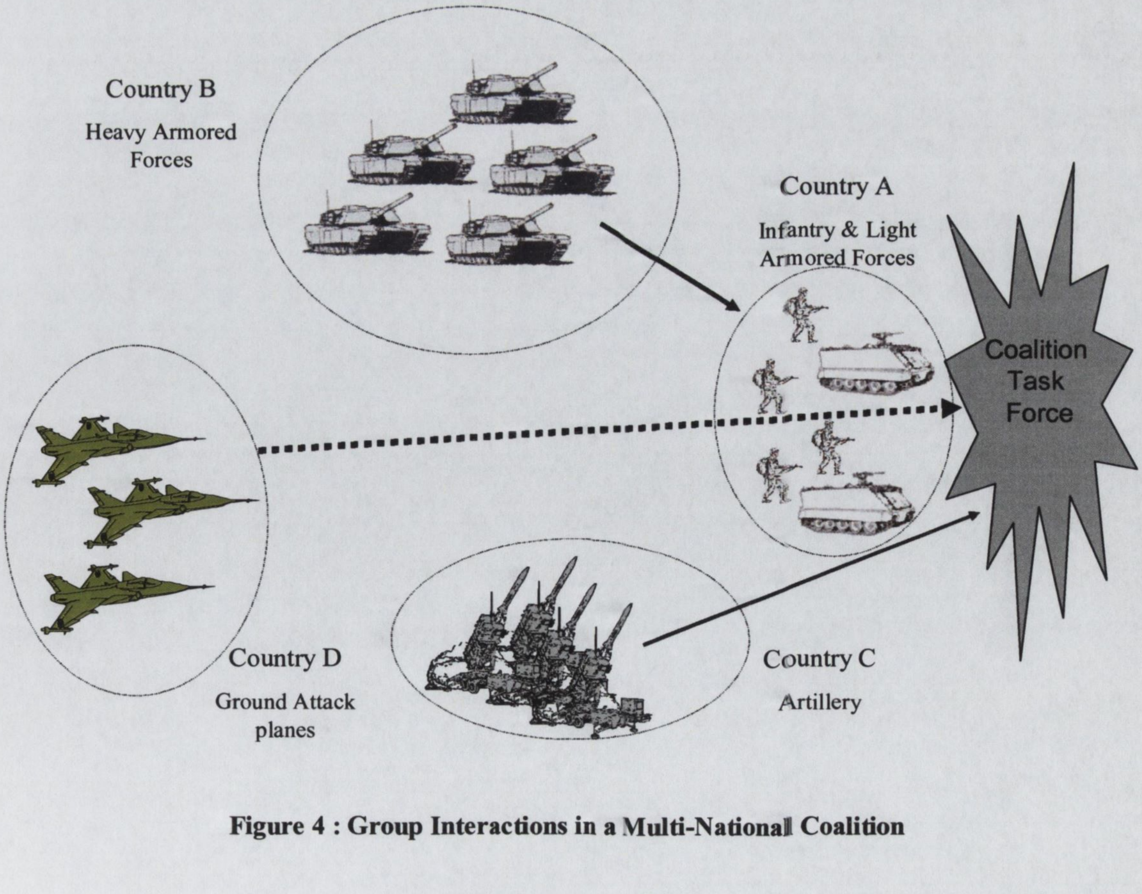


**Figure 4 : Group Interactions in a Multi-National Coalition**

## *1.5 Contributions*

The main contribution of this thesis is a flexible, scalable, and routing-protocol independent secure group collaboration scheme for Ad Hoc networks deployed in a civilian or military emergency scenario. Membership control in the secure group collaboration is achieved using a variation of the new trust negotiation protocol developed for the one-to-one interaction between two Ad Hoc nodes. A novel two-tier key formation provides confidentiality and integrity to the one-to-one trust negotiation protocol interchange and the communications between the authenticated members of the Ad Hoc network. The focus of the proposed one-to-one (Chapter 4) and group (Chapter 5) schemes is on the protocol and the architectural requirements as this allows the prototypes to be tested on real commodity mobile handheld wireless devices. The scope of the research involved in the development of the two schemes is presented in the next section.

### 1.5.1 Outline and Scope of the Thesis

Chapter 2 starts with a survey of the access control mechanisms in conventional and Ad Hoc networks. This survey concludes that the trust negotiation approach is best suited to provide access control for the Ad Hoc nodes participating in a one-to-one interaction. To provide confidentiality and integrity during the trust negotiation process, a common key has to be agreed between the two participating nodes. The choice of the key establishment protocol used to negotiate a common key has to be a compromise between the traditional security requirements and acceptable performance in face of the Ad Hoc network topological changes. At the end of Chapter 2, the compromise key establishment scheme selected is a modified version [wm99] of the Station-to-Station (STS) protocol [dow92].

In Chapter 3 a survey of secure group formation and communication in Ad Hoc networks is undertaken. This uncovered the inadequacies of the present schemes in terms of the scalability and the ease of deployment. Another related issue surveyed in this chapter is the establishment of a group key shared by many nodes. Existing group key agreement schemes were found to be inflexible and not fault tolerant for a large and dynamic Ad Hoc group. However, the existing group key transport mechanisms in general are more suited for a common key distribution in a large and dynamic Ad Hoc group. The lessons learnt from the surveys in Chapters 2 and 3 are used to design the one-to-one interaction and the group collaboration presented in Chapters 4 and 5 respectively.

Chapter 4 presents a new one-to-one trust negotiation scheme along with a novel two-tier key formation for Ad Hoc networks deployed in an emergency scenario. The trust negotiation scheme uses a policy to translate the attributes received in the remote node's certificate(s) to determine the access to services that the node provides. Other similar trust negotiation work [trust, w03, wsj00] has concentrated on the policy language aspects of the trust negotiation process, and less on the protocol and the architectural issues. Moreover, the protocol developed for wireless devices in the original scheme [hss+04] is complex and relies on access to a fixed network. In this thesis, the emphasis is on the development of a trust negotiation protocol which works on real handheld mobile wireless devices with reasonable performance. The use of two-tier key formation is also integrated into the scheme to provide confidentiality and integrity to the trust negotiation process. In order to test the prototype of the scheme, a simple policy language is used (see Appendix B).

The one-to-one trust negotiation process is extended to provide access control for secure group collaboration proposed in Chapter 5. A group is formed when many entities satisfying a common criteria, come together to share a resource, service or an application. The group has a flexible, scalable and fault-tolerant tree structure which aids in the robust distribution of the confidential group messages using a mesh topology. At the end of the chapter, the group collaboration protocols are tested on real handheld wireless nodes and some emulated nodes. The schemes presented in Chapter 4 and 5 are independent of the underlying routing algorithm to increase their portability.

Finally, in Chapter 6, the thesis concludes with a summary of the contributions. A discussion of possible future work arising from the schemes presented in this thesis is given. Initial results of this thesis have already been presented at the European Wireless'04 conference [vot04].

# 2 Access Control and Key Establishment

The advent of the multi-user networked computer systems stimulated the development of access control mechanisms. In such systems there is always a possibility of accidental or malicious disclosure of the confidential data to unauthorized users. This disclosure was initially prevented by using a simple password-based access control mechanism. However, overtime the access control mechanisms became more complex over time due to the evolution of new attacks, the desire for greater flexibility and the introduction of wireless as a networking medium. In the infrastructureless Ad Hoc networks, access control mechanisms are more complex as it is not possible to guarantee access to an online TTP. In the remainder of this chapter, the access control schemes for both conventional and Ad Hoc networks are discussed along with the feasibility of using these techniques in a large and mobile Ad Hoc network deployed in an emergency scenario. This chapter also discusses the related issue of key establishment techniques between two nodes in an Ad Hoc network setup by emergency services.

## *2.1 Access Control*

Access control is usually defined as a two stage process i.e. Authentication followed by Authorization. The authentication process verifies the identity of the entity requiring access. This is followed by the authorization process which delineates the access rights of the authenticated entity. However in newer access control systems the process of auditing is also incorporated which entails keeping a log of the access control activity. Such a log aids in the identification of malicious entities. Since the auditing process is a significant area of research in its own right, it will be largely ignored in this thesis.

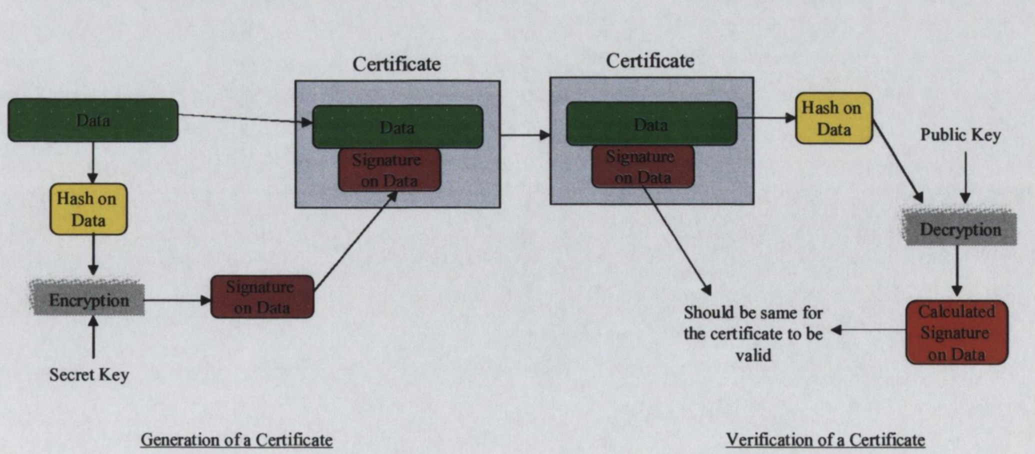
An example of a simple access control mechanism is remote telnet access into a UNIX system. The first step of the user authentication is carried out by verification of the user-id and password. Then next step of the authorization involves using an Access Control List (ACL) to determine the files and directories that the authenticated user can access. To simplify the access control mechanism, the steps of authentication and authorization are merged into one in the key-oriented access control systems. In such systems the access credentials have the access rights cryptographically embedded in a secure manner. Therefore an entity only has to produce the relevant credential(s) to have access. Common examples of such systems are the Simple Public Key Infrastructure (SPKI) [spki] and the Keynote [bfk98] systems.

### 2.1.1 Access Control in Convectional Networks

The first formal definition of an access control mechanism was by Lampson [171] and took the form of an access control matrix having two manifestations: the *Access Control Lists* (ACL) and the *Capability Lists*. In an ACL, a resource-wise record is maintained detailing the user access. However in the Capability Lists, records are kept for each user detailing what resource it can access. These generalized concepts are still being used in present-day access control systems. Lampson also defined the *confinement problem* [173] which stated that despite the best efforts of the designer of an access control system, confidential data could be leaked from the system.

To address the *confinement problem*, the Mandatory Access Control (MAC) [mac] technique sought to make it impossible for any access not permitted under the system-wide security policy to take place. This policy is dictated by the system manager(s) and the ordinary user has no say in it. Therefore the ordinary user cannot give access to its data other users unless permitted by the system-wide MAC. Moreover, the MAC mechanism has to be embedded into the appropriate system components (e.g. security kernel) and consequently has a central point of failure. In contrast to the MAC paradigm, the Discretionary Access Control (DAC) [dac] allows each user to grant access rights. Consequently, its major drawback is that in absence of any central control, a user can intentionally or accidentally distribute restricted data. This is particularly true in large organizations where determining the ownership of data is difficult. To address such situations a form of DAC called the Role-Based Access Control (RBAC) [fk92] was proposed. In RBAC, the role of the user in the organization determines the access control decisions. A user cannot pass on the access control permissions to others users unless it is permitted by the RBAC policy. Thus the RBAC is a restricted form of DAC and can be said to be a compromise between the MAC and DAC access control paradigms.

The earlier implementations of the MAC and DAC techniques had complex key management [bgs92, g89a] as they were mostly based on symmetric-key cryptography. This ensured that a plethora of keys had to be managed by an access control system. For example a key is needed for each server-client pair and each server-resource association. Moreover, the identity and associated keys had to be distributed securely. Thus the authentication and authorization process required a complex key management system. This was greatly simplified with the introduction of the public key cryptography (see Figure 5). In this paradigm only the *public key* of an entity needed to be distributed to others leading to easier distribution and management of the keys. Furthermore, the identity and key association management was simplified by the introduction of *certificates* which contained the identity and public key along with other information about an entity. The identity and public key along with other information is bound together in a way that it is cryptographically verifiable. This introduction of certificates led to new ways to manage the asymmetric keys. The two contrasting but widely used asymmetric key management schemes are the distributed Pretty Good Privacy (PGP) [z97] and the structured X.509 [x.509].



Symmetric Paradigm

Asymmetric Paradigm

Generation of a Certificate

Verification of a Certificate

**Figure 5 : Symmetric and Asymmetric Paradigms**

#### *2.1.1.1 Asymmetric Key Management*

The most widespread asymmetric key management system in use nowadays is the X.509 Public Key Infrastructure (PKI) framework [x.509]. The core of the system is the Certification Authority (CA) responsible for the issue of the certificates. When an entity trusts a CA it stores a self-signed CA certificate. This contains the public key of the CA which is used to verify all other certificates issued by the CA. Any certificate issued by the CA is valid for a fixed time interval and contains the identity, public key and other information cryptographically signed by the CA. In situations when a time-valid certificate has to be revoked, the CA (or other entities designated by the CA) includes a reference to the certificate in the periodic issue of the Certificate Revocation List (CRL) [x.509]. Therefore to check the up-to-date validity of a certificate, an entity also has to refer to the latest CRL issued by the corresponding CA. The periodicity of the CRL update needs to be carefully selected depending on the scenario. If the CRL update time is large, some untrustworthy certificates may be continued to be used after the compromise is known. Moreover, if the CRL update time is small it can lead to high distribution overheads. To mitigate the problems in deciding the periodicity of the CRL issue, a real-time certificate status checking mechanism is defined in the Online Certificate Status Protocol (OCSP) [x.509]. Under normal operational conditions the OCSP protocol returns a good, revoked or unknown status for a queried certificate. A good status returned implies that the certificate is valid (i.e. has not being revoked). However, the good status does not mean that the certificate is necessarily time-valid. A revoked status of the certificate entails that the CA has explicitly revoked the concerned certificate. If no information about the certificate in the query is available on the CA, then an unknown status is returned.

Another feature of the X.509 architecture is the provision to organize the CA's in a hierarchical tree structure. An entity places trust in the root CA of the tree hierarchy. Consequently the entity trusts the entire tree of CA's. The leaf CA's does most of the certification work while the intermediate level CA's extend the trust from the root CA to the leaf CA's. This leads to creation of a *certificate chain* starting with the root CA certificate (see Figure 6). The root CA certifies the certificate of the CA one level down in the tree hierarchy. This certificate chaining process is recursively carried down the tree hierarchy and terminates at the leaf level CA. When a leaf level CA issues a certificate to an entity, the *certificate chain* is also given to the entity. The chaining of certificates is a good concept, but creates problems when the non-leaf CA certificates expire or are revoked. Moreover, in a dynamic Ad Hoc network, due to link and route failures the CRL update may never reach the intended entities in time. The non- timely receipt of the CRL's can result in an entity making the wrong trust decision about a certificate.

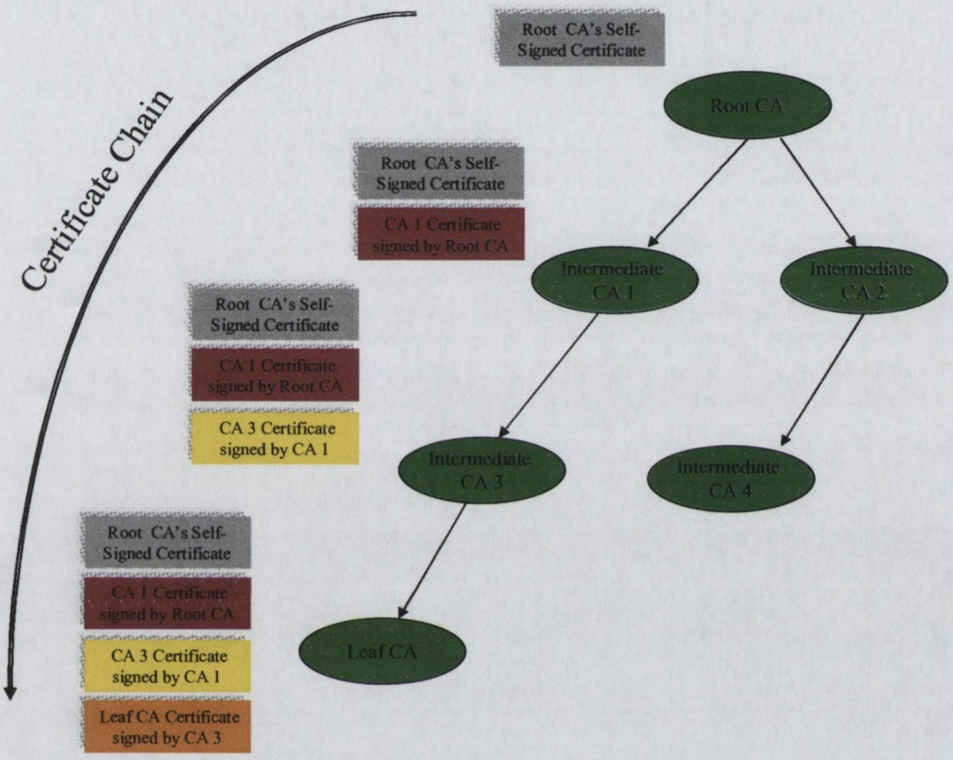

Certificate Chain
Root CA's Self-Signed Certificate
Root CA
Root CA's Self-Signed Certificate
CA 1 Certificate signed by Root CA
Intermediate CA 1
Intermediate CA 2
Root CA's Self-Signed Certificate
CA 1 Certificate signed by Root CA
CA 3 Certificate signed by CA 1
Intermediate CA 3
Intermediate CA 4
Root CA's Self-Signed Certificate
CA 1 Certificate signed by Root CA
CA 3 Certificate signed by CA 1
Leaf CA
Leaf CA Certificate signed by CA 3

**Figure 6 : Certificate Chain in Hierarchical CA's**

In contrast to the hierarchical structure of the X.509, PGP has a distributed structure with each participating entity responsible for maintaining its own certificate store. However there are some central repositories which are rarely used. Instead of a CA issuing a certificate, each participating entity in PGP generates its own self-signed certificate. To build trust in a self-signed certificate, the PGP uses the *web of trust* model. If an entity trusts a remote entity's certificate it signs the concerned certificate's public key using its own secret key. Thus the entity becomes an *introducer* of the public key it has signed. This process of signing can be repeated over several entities leading to creation of a "ring" of certificates. However, the revocation mechanism of certificates in a *ring* is difficult and cumbersome. After the *ring* is formed, "Bob" can decide that it does not trust "Alice" anymore and consequently revokes its signature on Alice's public key. This decision will have to be conveyed to "Charles" manually and so on down the chain (see Figure 7). Though PGP is fully distributed, it is not suitable for Ad Hoc networks, as self-signed certificates have little trust values in such environment. To trust a self-signed certificate, it must be signed by some trusted *introducer* which is not a reliable method to estimate the trustworthiness of a certificate (see Figure 7).

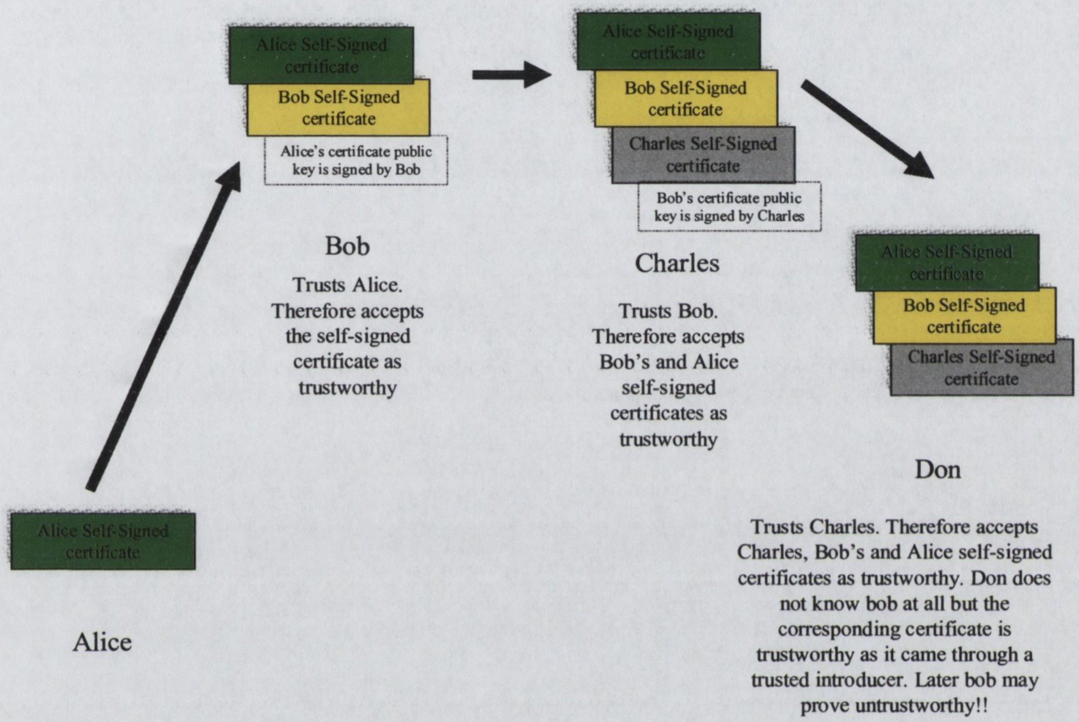

Alice Self-Signed certificate
Bob Self-Signed certificate
Alice's certificate public key is signed by Bob
Bob
Trusts Alice. Therefore accepts the self-signed certificate as trustworthy
Alice Self-Signed certificate
Bob Self-Signed certificate
Charles Self-Signed certificate
Bob's certificate public key is signed by Charles
Charles
Trusts Bob. Therefore accepts Bob's and Alice self-signed certificates as trustworthy
Alice Self-Signed certificate
Bob Self-Signed certificate
Charles Self-Signed certificate
Don
Trusts Charles. Therefore accepts Charles, Bob's and Alice self-signed certificates as trustworthy. Don does not know bob at all but the corresponding certificate is trustworthy as it came through a trusted introducer. Later bob may prove untrustworthy!!
Alice Self-Signed certificate
Alice

**Figure 7 : PGP's Web of Trust**

The Secure Socket Layer (SSL) [fkk96] is presently the most used example of a authentication scheme on the conventional networks that relies on asymmetric key management. The authentication of the entities by the SSL mechanism is followed by the authorization process in the access control systems. An additional advantage of using SSL is the establishment of a common key between the two participating entities (i.e. the client and server). This common key can be used to encrypt or verify their communications. In SSL, the server releases its certificate unilaterally even before it can authenticate the client. The client has no opportunity to ask for more server certificates than the ones released by the server. Therefore the access control systems which use SSL for authentication are inflexible. This inflexibility is addressed to an extent by trust management systems discussed in the next section.

#### *2.1.1.2 Trust Management*

Trust management systems like the KeyNote [bfk98] and the Simple Public Key Infrastructure (SPKI) [spki] amalgamate the two steps of authentication and authorization into one. The KeyNote system manages the delegation of access rights to a *principal* (i.e. user/client/node) to act on an *application*. The *principal* asks for an *action* (i.e. access) to be performed on the *application*. This *principal* also supplies the signed *credentials* (a public-key certificate is an example of a KeyNote credential) showing that it is authorized to perform the requested *action*. The *application* then asks the *KeyNote Trust Management* system if the *request* for the *action* asked for by the *principal* is valid. The *application* making a query to the KeyNote system also sends the *application policy*. This policy defines the access control a *principal* can have on the *application*. Thus a typical query by the *application* to the *KeyNote Trust Management System* contains the *principal's* identifier, *action* requested, *application policy* for the *application* and supporting *credential(s)* provided by the *principal*. This query to *KeyNote Trust Management System* returns a *Policy Compliance Value (PCV)* telling the *application* if the *action* requested by the *principal* is allowed or not. The result of the query also depends on the KeyNote system verifying the *credentials* from the trusted PKI's. A schematic diagram of the keynote system is presented in Figure 8.

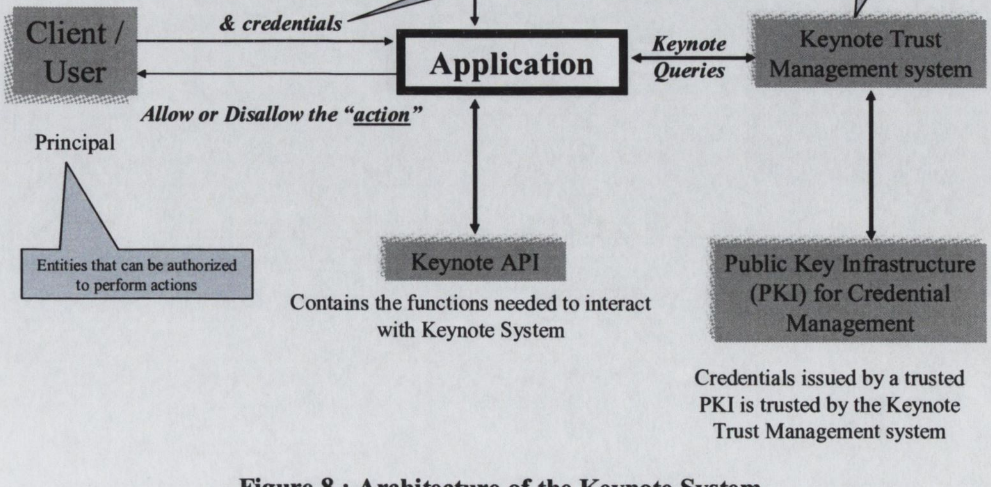

Application Policy
Govern the actions that principals are authorized to perform on the application
Tell the application if the action requested by principals is allowed or not, given a policy and the credentials
Operations with security consequences that are to be controlled by the system
Credentials primarily allow a principal to delegate authorization to other principals
Client Request for a "action"
& credentials
Client / User
Application
Keynote Queries
Keynote Trust Management system
Allow or Disallow the "action"
Principal
Entities that can be authorized to perform actions
Keynote API
Contains the functions needed to interact with Keynote System
Public Key Infrastructure (PKI) for Credential Management
Credentials issued by a trusted PKI is trusted by the Keynote Trust Management system

**Figure 8 : Architecture of the Keynote System**

The *credentials* in KeyNote are signed messages (similar to public key certificates) and are primarily concerned with delegation of the access authorizations. A *principal* has the power to delegate all or part of the *actions* granted to it to other *principals*. This delegation can be done transitively by the *principals* and lead to the formation of a delegation chain. However a delegation chain may be overridden by the local *application policy*. The Scalable Trust of Next Generation Management (STRONGMAN) [kig+03] extends the Keynote concept by embedding the access authorizations into the *credentials* (i.e. lazy policy instantiations). This idea of embedding the access rights into the certificates is taken one step further in the SPKI [spki]. Certificates in the SPKI framework contain the public key and access authorizations embedded in the form of an ACL along with optional inclusion of the identity of the certificate owner. The ACL can be implemented in any way the developer chooses. In addition to the CRL approach for revoking a time valid certificate, the SPKI also offers the revalidation method. This is similar to the OCSP certificate validity checking mechanism of the X.509 framework. An entity wishing to check the validity of a certificate(s) sends it to the designated server on the PKI. If the queried certificate(s) are valid then the certificate(s) themselves are returned. In case of the certificate(s) being invalid, an empty result is returned.

The KeyNote and SPKI systems manage pre-existing trust relationships (i.e. security associations) and are not able to dynamically form new ones. Thus they are trust management systems and not trust negotiation systems. The system to dynamically form trust relationships between two participating entities was first proposed by Winsborough et al [wsj00] as part of the Trust Establishment Project at the IBM Haifa Research Laboratory.

### *2.1.1.3 Trust Negotiation*

The trust negotiation scheme proposed by Winsborough et al [wsj00] was geared towards a client-server architecture and involved progressive interchange of certificates (containing custom embedded attributes) governed by the local policies of the involved entities. A server allows all access to it through a *Server Security Agent* regulated by the *Service Governing Policy*. The server certificates released to the clients is governed by the *Server Certificate Access Policies*. For the client there is a similar *Client Security*

*Agent* along with client certificates and the corresponding *Client Certificate Access Policies*. The respective *security agents* of the server and client request certificates from each other. A request for a certificate can be fulfilled by providing a certificate or another certificate request. The client/server releases the certificate asked for if the corresponding certificate release policy is satisfied. Otherwise the client/server asks for certificates that can fulfil the release policy by sending a certificate request to the remote entity asking for the local certificate. Therefore a *security agent* can ask the remote server/client for additional certificates before it releases its certificates basing such decisions on the local certificate release policies. The *Service Governing Policy* maps the attribute(s) from the valid received remote certificates into access control decision. In extreme cases there can be a cyclical exchange of the same certificate requests leading to a deadlock. This deadlock is reached as each participating entity in the negotiation is waiting for the other to release a certificate before it will release its local certificate to the remote entity. The two participating entities in the trust negotiation protocol recognize the deadlock if two consecutive and similar cyclic exchange of certificate request(s) takes place. Then the protocol interchange is suspended. To overcome the deadlock, the server can release the *Service Governing Policy* to an unfamiliar new client. Moreover, a client can store the *Accumulated Server Credentials* and all the prior incoming/outgoing certificate requests. This helps in fast renegotiation and deadlock avoidance on the client.

The work of the Trust Establishment Project was progressed by the TrustBuilder project [trust, w03] with refinements to the architecture and the definition of the first trust negotiation protocol [hjm+02]. The architecture became peer-to-peer based with both the participating entities having common data structures. This simplified the process of writing the policies for trust negotiation. The security agent could ask the remote entity for the *certificates* or *policies*. These *policies* contained the conditions for access to a service. Therefore it became simpler for an entity to know what certificates are required. In the TrustBuilder project, more protection was accorded to the sensitive certificates with better access control and provisions not to disclose the sensitive attribute name/value pair(s) (*selective disclosure*) contained in a certificate.

The success and failure of the trust negotiation process also depends significantly on the negotiation strategy [wsj00] used by the two participating entities. These negotiation strategies could be loosely classified as eager, parsimonious or hybrid. The eager strategy entails release of all certificates which are unlocked by the local policies to the remote entity. After a batch of the remote certificates is received, an entity computes (depending on the policies) which local certificates are free to be disclosed to the remote entity. Then these certificates are sent unsolicited to the remote entity. There is little or no role for certificate requests in such a negotiation. The use of an eager strategy can lead to disclosure of too much information to the remote entities. In the parsimonious strategy, the release of the unlocked certificates is dependent on the receipt of certificate requests from the remote entity. This approach has a higher probability of a deadlock in the trust negotiation. To mitigate the deficiencies in the previous two approaches towards negotiations, a hybrid type was proposed. The hybrid strategy is a two-step process with the entity following the eager strategy in the first stage. Then the results of the first stage are fed into the second parsimonious strategy stage. The hybrid strategy is the best for getting a positive result from the trust negotiation process with adequate security for the sensitive certificates.

The TrustBuilder approach initially looked attractive for Ad Hoc networks, but on closer examination it had several drawbacks. Firstly, the trust negotiation protocol for a conventional wired network had redundant messages due to enveloping of the negotiation protocol [hjm+02] into the Transport Layer Security (TLS) protocol [tls]. The additional messages in the TLS protocol meant a higher likelihood of interruption in the protracted protocol interchange due to topological changes. Secondly, a trust negotiation between two wireless devices happens in a surrogate manner [hss+04]. In this surrogate technique instead of the two wireless nodes negotiating trust directly, their respective agent nodes on the fixed network negotiate trust. The result of the trust negotiation carried out by the agent nodes is then conveyed securely to the wireless nodes. However this surrogate technique of trust negotiation has two disadvantages. Firstly, a second protocol is required to get the result of the trust negotiation to the wireless nodes securely from the fixed nodes. This increases the complexity of the solution. Secondly, in Ad Hoc networks it is difficult to guarantee routes between nodes. Therefore the results of the trust negotiation may not get back to the wireless nodes. Another minor drawback is that the TrustBuilder scheme has little or no avenues for human intervention if the trust negotiation process became deadlocked.

#### *2.1.1.4 Conclusion of Access Control in Conventional Networks*

The static access control mechanisms for the conventional networks are generally inflexible for use in a large and dynamic Ad Hoc network deployed in an emergency scenario. In Ad Hoc networks formed by emergency services it is best for each participating entity to make individual decisions about the trustworthiness of a remote entity. The trust negotiation approach efficiently provides this feature. This interim conclusion is arrived after examining the access control mechanisms in conventional networks. Therefore to arrive at a final conclusion, the next section examines the suitability of access control schemes in Ad Hoc networks for use by emergency services.

### 2.1.2 Access Control in Ad Hoc networks

The unique characteristics of Ad Hoc networks prompted the development of new schemes for access control. Access control schemes in conventional wired and wireless networks generally rely on access to an online TTP. This cannot be guaranteed in Ad Hoc networks due to frequent route changes and partitioning. Moreover, the relatively unreliable wireless medium used in the Ad Hoc networks ensures that key management schemes for conventional networks are not robust to withstand the frequent topological and membership changes. This resulted in cryptographic schemes specifically designed for Ad Hoc networks [bh03, bhr+01, ddg+01, hbc01, kka03, klx+02, ldo03, lzk+02, yk03, zh99] which can be classified into three major categories: namely, the threshold CA approach, self-organizing PKI approach and pre-arranged shared secret approach.

#### *2.1.2.1 Threshold Certification Authorities*

The use of the threshold cryptography [gjk+96] along with the public-key paradigm to make a distributed certification authority for authentication was first proposed by Zhou et al [zh99]. The scheme (see Figure 9) is based on the pre-distribution of parts of the secret key among some special entities designated as the *servers*. This distribution is done in such a way that to compromise the secret key of the distributed CA service an attacker(s) has to compromise a *threshold number t* of *n servers* where $t<n$. All the participating entities in the service know the public key of the service and so can verify

the validity of a message if it is signed by the distributed secret key. Since the secret key is distributed, any valid message has to be signed by more than the *threshold number* of *servers*. To prevent the progressive compromise of the *servers*, *share refreshing* is done periodically. This involves each *server* generating a new share of its part of the secret key in conjunction with other *servers*. Moreover, the *threshold number* of *servers* can be changed. The *share refreshing* and the allowance for change in *threshold number* is a useful feature to cope with network partition situations, as a new partition can generate a new distributed secret key with a different number of *servers*. However, *share refreshing* can have large communication and computation overheads. If the number of *servers* is *n*, then the *share refreshing* process involves O($n^2$) generations of numbers (for new share), O($n^2$) network transmissions and O(n) computations.

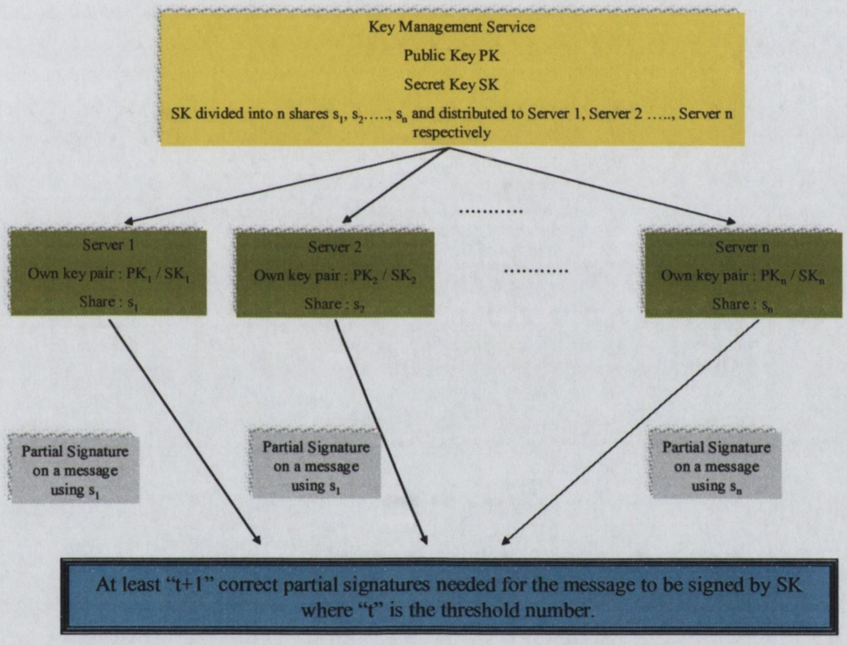

Key Management Service
Public Key PK
Secret Key SK
SK divided into n shares $s_1$, $s_2$....., $s_n$ and distributed to Server 1, Server 2 ....., Server n respectively
...........
Server 1
Own key pair : $PK_1$ / $SK_1$
Share : $s_1$
Server 2
Own key pair : $PK_2$ / $SK_2$
Share : $s_2$
...........
Server n
Own key pair : $PK_n$ / $SK_n$
Share : $s_n$
Partial Signature on a message using $s_1$
Partial Signature on a message using $s_1$
Partial Signature on a message using $s_n$
At least "t+1" correct partial signatures needed for the message to be signed by SK where "t" is the threshold number.

**Figure 9 : Key Management using Threshold Cryptography**

The operation of the distributed CA is a complex process. The initializing of the scheme requires that all the servers know each others public keys ($PK_i$ in Figure 9). This may result in big network and time overheads if the number of *servers* is large. The coordination required for collaborative generation of keys and initial setting up of the shares is time-intensive. Moreover some entities (namely the *servers*) will have to work more than other entities in Ad Hoc networks. This was rectified in a variation that was proposed by Luo et al [lzk+02] making it mandatory for all the nodes in an Ad Hoc network to participate in the distributed key management service. Another drawback of the distributed CA approach is the high-overheads incurred when the network partitions and re-merges. The partitions may have generated their own distributed secret keys. On re-merger a new distributed secret key has to be generated for the merged group with the individual *servers* having to generate fresh shares. Several variations of the distributed CA [kka03, ldo03] has been proposed including interesting work is by Seung Yi et al [yk03] which combines the distributed CA with concept of levels of confidence in a certificate chain.

#### *2.1.2.2 Self-Organizing Public-Key Infrastructure*

The previous approach of a distributed certification authority using threshold cryptography to ensure authentication in an Ad Hoc network involved pre-configuration and extensive coordination among the participating nodes. In contrast, the spontaneous mechanism proposed by Hubaux et al [hbc01] involved a public-key distribution scheme for Ad Hoc networks similar to the PGP *web of trust* concept. However the major difference of this self-organizing approach from the PGP is the way the certificates are used to build a chain. In this scheme, an entity selects a subset of certificates from its repository to be disclosed to the remote entity. An entity then merges the received certificates with their own certificates. This is followed by use of the Hunter Algorithm [hbc01] on the merged certificate repository to build certificate chain(s). A certificate trust chain should lead from a local certificate to the remote entity's certificate. The local entity then uses the public-key contained in the selected remote entity's certificate (from the valid certificate chain (if any)) to build a security association. The probability of finding such a certificate chain is high but not guaranteed. This decentralized scheme leads to disclosure of too much information about the participating entities as it releases several unnecessary certificates, which may not be needed in chain formation. Another major drawback of this scheme is that it makes no provision for certificate revocation.

#### *2.1.2.3 Pre-arranged Shared Secret*

This broad approach for key management for authentication is based on the existence of a shared secret among the nodes in the Ad Hoc network. There are many ways in which the pre-configured shared secret can be used for key management. A straightforward example is the Secure Pebblenets [bhr+01] in which all the participating nodes are pre-configured with a common key. The key update is done periodically and has three phases. In the first phase the network is divided into clusters of nodes. Each cluster elects a head node responsible for key distribution. In the next phase these cluster heads generate a traffic encryption key authenticated by the initial shared secret. This is followed by the operational phase in which the cluster heads agree to a new *global* traffic encryption key out of all the individual traffic encryption keys generated. Then this global traffic encryption key is used by the group of participating nodes to encrypt/decrypt messages.

Another pre-configured shared secret approach is the probabilistic key sharing [cps03, eg02] method. In this approach, the nodes wanting to participate in the Ad Hoc network select for themselves a subset of pre-generated keys. Each of the participating node then searches for other nodes with at least one similar common key. The probability of finding another node with similar key is high due to the way the superset of pre-generated keys was initially generated. This shared key between two nodes is used to encrypt mutual communications. A node wanting to communicate with another node with which it does not have a shared key will send the data to other nodes with which it has a shared key. These nodes in turn will send it to other nodes that they have a common shared key. This process continues till the data reaches the intended node. The probabilistic key management for authentication is preferred in sensor networks as it relies on computationally inexpensive symmetric encryption and also has a small memory requirement. The pre-arranged shared secret approach in general has a major drawback that if the initial shared secret is compromised then it is difficult to stop a progressive compromise of the security scheme. Moreover, the scalability is limited as the participating nodes have to be pre-configured with the shared secret.

#### *2.1.2.4 Hybrid Approaches*

In order to optimise the performance, some of the designers of key management schemes for authentication in Ad Hoc network decided to use the best features from the aforementioned approaches. One such scheme proposed by DeCleene et al [ddg+01] has a hierarchical framework with the division of the region of operation into areas. Each area in the hierarchy has a controller having keys agreed with all nodes in the area. These area controllers re-key a node when it moves between different areas using a handover mechanism. The area controllers are fixed entities essential to the operation of the scheme. However, in a dynamic Ad Hoc network access to the area controllers cannot be guaranteed. Another hybrid scheme designed for military scenarios was proposed by Kong et al [klx+02] and uses the threshold certification authority, the pre-arranged shared secret model along with a Public Key Infrastructure (PKI) based centralized model. Initially the scheme has an aerial node acting as the centralized PKI node for key distribution and verification in the network. If this aerial node is destroyed, the scheme uses threshold cryptography based on the pre-distributed secret sharing to form a distributed CA to replace the destroyed central aerial node.

#### *2.1.2.5 Conclusion of Access Control in Ad Hoc networks*

The threshold distributed certification approach requires extensive coordination among the participating nodes of the Ad Hoc network. Schemes using this approach are not fault tolerant in respect to link and route failures, which occur frequently in a large dynamic Ad hoc network. Moreover, pre-configuration is required among the participating nodes of a threshold distributed certification authority. The schemes using pre-configuration have limited scalability in emergency scenarios which requires a spontaneous setup of an Ad Hoc network. The Self-Organizing Public-Key Infrastructure approach is radically different from the other approaches of building security associations in Ad Hoc networks. However, this approach results in disclosure of too much information (in form of certificates) to the remote entity along with no provision for certificate revocation. Thus the existing paradigms for key management in Ad Hoc network are not viable to provide authentication for access control in a large and dynamic Ad Hoc network deployed in an emergency scenario.

To provide confidentiality and integrity to the access control process in Ad Hoc networks, a shared key needs to be established between the two participating entities. Moreover, the present key establishment techniques can also double in role as an authentication method.

## *2.2 Key Establishment Between Two Entities*

A common key can be established between two entities using two distinct categories of protocols (see Figure 10) differentiated in the way the participating entities contribute to the common key formation. In the key transport protocols, one entity generates a key and transports it securely to the other entity. However, in key agreement protocols both the entities have a say in the parameters used to generate the common key. This contributory aspect of the key agreement protocols ensures that the common key agreed is fresh, unlike the key transport protocols where a malicious entity can pass compromised or reused key to the other participating entity. A reused key increases the probability of key compromise by providing data for a brute force cryptanalysis attack. Therefore in this thesis the focus is on the key agreement protocols.

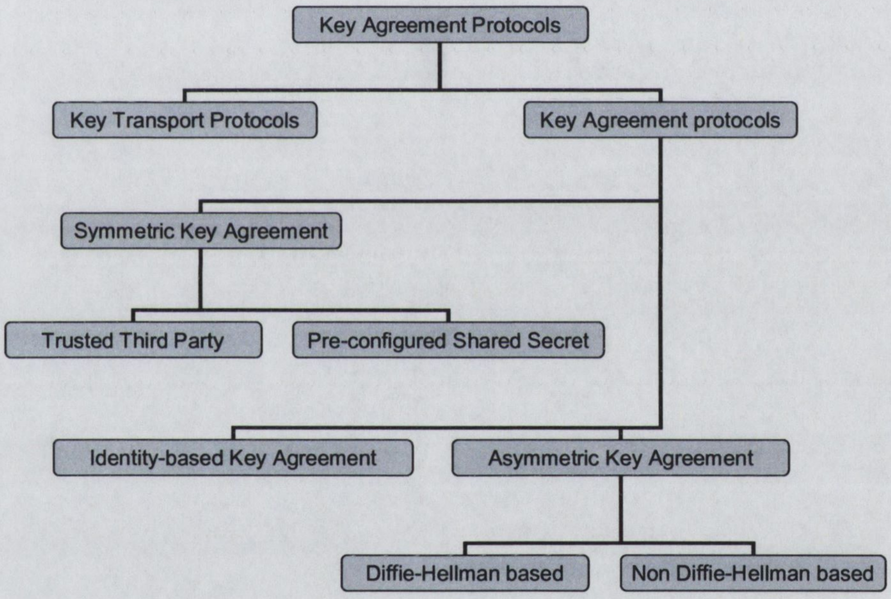


**Figure 10 : Classification of Key Establishment Protocols**

### 2.2.1 Key Agreement Protocols Between Two Entities

In Ad Hoc networks, key agreement protocols are more susceptible to the active and passive attackers due to the use of a wireless medium. Any key agreement scheme used in Ad Hoc networks must authenticate the two participating entities to guard against man-in-the-middle type of attacks. An additional benefit of the authentication in the key agreement protocols is that this information can be later used in the access control process. The search for an appropriate key agreement protocol starts with the class of protocols using the symmetric cryptography (see Figure 10). Such protocols either rely on a participating online TTP or use a pre-configured shared secret for authentication. The TTP approach implies the protocol interchange is between three active parties, the two participating entities and the TTP. However, in Ad Hoc networks it is not always possible to guarantee a route to the online TTP. Therefore the use of symmetric key agreement approach is not considered feasible. The second class of key agreement protocols (Figure 10) use identity-based encryption. In such protocols the identity information used for authentication is contained in the public key. However, identity based key agreement protocols suffer from the same drawbacks as symmetric key agreement protocols as their operation either requires pre-configuration or access to an online TTP to authenticate the identity of the participating entities. In contrast to identity or symmetric key agreement schemes, the asymmetric approach generally uses certificates to authenticate entities. To update certificate revocation information, the entity has to have periodic access to the CA. This is preferable to a participating online TTP or pre-configuration approaches. Therefore the focus of the search is on the asymmetric key agreement protocols. There are a plethora of such schemes [abb+04, bm03, ikev1, ikev2, mov96, s96, sk00] available providing various degrees of protection against attackers. The requirements for the key agreement protocols required in this thesis are specified in the next two sections. In the next section the general requirements for a key agreement protocol are enumerated. This is followed by a section on the Ad Hoc network specific requirements.

#### *2.2.1.1 General Requirements of a Key Agreement Protocol*

This section presents the requirements for any key agreement protocol providing authentication in conventional or Ad Hoc networks. A good attempt at enumerating the formal objectives for key agreement was by Syverson et al [so94, so96]. These formal objectives formed the basis for defining the general requirements.

- **Entity Authentication** – The entity involved in the agreement should be able to authenticate the other entity ensuring that the common key is with the intended party and not with a man-in-the-middle type of attacker. One of the best definitions of entity authentication is by Menezes et al [mov96] stating "*Entity authentication is the process whereby one party is assured (through acquisition of corroborative evidence) of the identity of a second party involved in a protocol, and that the second has actually participated (i.e. active at, or immediately prior to, the time the evidence is acquired)*". This definition was further extended to **strong entity authentication** and is quoted as "*strong entity authentication of A to B is provided if B has a fresh assurance that A has knowledge of B as her peer entity*"

- **Mutual Authentication** – Both the participating entities authenticate each other. This is a trivial extension of the entity authentication requirement.
- **Key Freshness** – A negotiated key is fresh if it can be guaranteed to be new, as opposed to the possibility of an old key being reused which an adversary might exploit.
- **Key Authentication** – Let the two entities involved in the key agreement process be A & B. This requirement applicable to the entity A implies that A is assured that the shared key is known to A & B. Also existence of key authentication from A to B does not imply that opposite is also assured. This requirement can be further refined into two more,
    - **Implicit Key Authentication** – The key agreement protocol is said to provide implicit key authentication from entity A to B, if A is assured that no one except B can learn the secret value of the mutual key.
    - **Explicit Key Authentication** – The key agreement protocol is said to provide explicit key authentication from entity A to B, if A is assured that B *has actually computed* the secret value of the mutual key.
- **Key Confirmation** – The protocol provides key confirmation if A is assured that B has the possession of the agreed shared key. Implicit key authentication along with key confirmation provides explicit key authentication [mov96, s96]. Also existence of this property from A to B does not imply that opposite is also assured.
- **Forward Secrecy** – If the long-term private keys of one or more of the entities involved in the protocol are compromised then it does not affect the secrecy of the previous keys agreed between the two entities.
- **Key Compromise Impersonation** – The compromise of the long-term keys of an entity A by an attacker C does not allow C≠A to impersonate A.

#### *2.2.1.2 Performance Requirements of a Key Agreement Protocol*

These are important in mobile Ad Hoc networks as most of the participating devices use wireless as the principal medium of communication and also may have limited CPU and battery power.

- **Minimum Pre-configuration** – In order to setup a spontaneous Ad Hoc network, the pre-configuration has to be kept to a minimum. Therefore protocols that assume that the two entities have already got each others public–key or have some pre-arranged secret are not suitable for use in a spontaneous Ad Hoc network. A CA based approach is more acceptable as it only involves getting a single public-key that can verify all the certificates issued by that CA.
- **Minimal Number of Messages Exchanged** – The dynamic topology of an Ad Hoc network dictates that the key agreement has to be done quickly with the least amount of messages exchanged. Since the wireless is a shared medium, the throughput of the network decreases if simultaneous neighbouring entities try to transmit in a small interval of time.
- **Low Communication Overhead** – Bigger messages take more time to transmit and therefore may not be desirable in a dynamic Ad Hoc network. Also fragments of bigger messages which need to be split for transmission can be withheld leading to attack that is difficult to distinguish from the normal network congestion. However pre-computation of data to be used in protocol messages can reduce the time an entity is engaged in a protocol interchange.
- **Low Computational Overhead** – Commodity handhelds with limited CPU and battery power form the bulk of the entities in an Ad Hoc network. The use of computationally time-intensive strong cryptographic primitives can be a drain on the limited CPU and battery power of such devices. Consequently the use of computationally heavy cryptographic primitives should be kept to a bare minimum in a key agreement scheme to be used in a dynamic Ad Hoc network deployed in an emergency scenario.
- **Non Reliance on Time Stamping** – As access to a network synchronized time is not typically available in an Ad Hoc network, the key agreement protocol should not rely on time stamping.

- **Resistance to Denial of Service (DoS) Attacks** – The protocol should provide adequate protection against attackers trying to prevent legitimate entities having a protocol interchange. This is the most difficult to achieve as new forms of DoS attacks are being constantly formulated. Moreover, as the wireless is a shared medium, it is more prone to such attacks than the conventional wired networks.
- **Minimal Negotiation of Capabilities in the Protocol** – There should be minimal negotiation of capabilities like hashing or signature algorithms. Such a negotiation can add messages or increase the length of messages in a key agreement process.

#### *2.2.1.3 Choice of Key Agreement Protocol*

The key agreement protocol chosen has to satisfy most of the general and performance requirements mentioned in the last two sections. This choice will be a compromise as some of the requirements contradict each other. A prominent example is the general requirement of the key agreement protocols to be resistant to the Denial of Service (DoS) attacks. This contradicts with the performance goals of having the minimum number of messages exchanged as the most general form of the DoS mitigation requires a stateless exchange of cookies (i.e. two extra messages in the protocol interchange). Another method of DoS mitigation suggested by Aura et al [an97] used *stateless connection*. To transform an existing key agreement protocol using the stateless approach results in additional overheads in the communications and computation requirements. Meadows [m99] suggested that each message in the protocol must be authenticated for DoS mitigation. However, this approach can result in big protocol message sizes and also an increase in number of protocol messages. Another approach towards DoS mitigation entails solving of *puzzles* [jb99] by the participating entities. The above mentioned DoS mitigation methods conflict with the performance goals of minimal computational and communication overheads. The search for the compromise asymmetric key agreement protocol is further narrowed down to the ones using the Diffie-Hellman paradigm. The major reason for this focus is that the Diffie-Hellman paradigm provides forward secrecy. The two most important columns in the comparison of such key agreement protocols (Figure 11) are the method used for entity authentication and the number of the CPU intensive public key operations at the initiator and responder respectively. As pre-configuration with a shared secret reduces the scalability, the protocols using the PKI for authentication are preferred.

| Key Agreement Protocol | # of messages exchanged | Implicit Key Authentication | Key Confirmation | Entity Authentication | Entity Authsentication using? | Public Key Operations on I, R |
|---|---|---|---|---|---|---|
| Static Diffie-Hellman | 0 | I↔R | no | no | PRE | 1,1 |
| ElGamal | 1 | I↔R | no | no | PRE | 2,1 |
| Nyberg-Rueppel | 1 | I↔R | I←R | I | PRE | 2,3 |
| Agnew-Mullin-Vanstone (AMV) | 1 | I↔R | I←R | I | PRE | 2,3 |
| Ephemeral Diffie-Hellman | 2 | no | no | no | - | 2,2 |
| Atenise-Steiner-Tsudik (AST) | 2 | I↔R | no | no | PRE | 3,3 |
| Matsumoto-Takashima-Imai (MTI) A(0) / B(0) | 2 | I↔R | no | no | PRE | 3,3 |
| Matsumoto-Takashima-Imai (MTI) C(0) / C(1) | 2 | I↔R | no | no | PRE | 2,2 |
| Goss | 2 | I↔R | no | no | PRE | 3,3 |
| Arazi's | 2 | I↔R | no | I,R | PRE | 3,3 |
| Yacobi | 2 | I↔R | no | no | PRE | 3,3 |
| Lee-Lim-Kim (LLK) | 2 | I↔R | no | no | PRE | 2,2 |
| Key Exchange Algorithm (KEA) | 2 | I↔R | no | no | PRE | 3,3 |
| Unified Model | 2 | I↔R | no | no | PRE | 3,3 |
| Menezes-Qu-Vanstone (MQV) | 2 | I↔R | no | no | PRE | 2.5,2.5 |
| Just-Vaudenay / Song-Kim | 2 | I↔R | no | no | PRE | 3,3 |
| Station-to-Station | 3 | I↔R | I↔R | I,R | PKI | 3,3 |
| SKEME basic mode | 3 | I↔R | no | no | PRE | 2,2 |
| Hirose-Yoshida | 3 | I↔R | no | I,R | PRE | 3,3 |
| Oakley aggressive mode | 3 | I↔R | I↔R | I,R | PKI/PRE | 3,3 |
| Just Fast Keying - JFKi variant | 4 | I↔R | I↔R | I,R | PKI/PRE | 4,3 |
| Just Fast Keying - JFKr variant | 4 | I↔R | I↔R | I,R | PKI/PRE | 3,3 |
| Internet Key Exchange Version 1 (IKEv1) Main Mode | 6 | I↔R | I↔R | I,R | PKI/PRE | 3,3 |
| Internet Key Exchange Version 2 (IKEv2) Phase 1 | 4 | I↔R | I↔R | I,R | PKI/PRE | 3,3 |

I : Initiator

R: Responder

I↔R : Property is provided by I and R respectively

I←R : Responder provides the property to Initiator

**Public Key Operations** are Diffie-Hellman **$a^b$ mod n** and the RSA verification operation.

**PRE** – Pre-exchange of public keys / shared secret

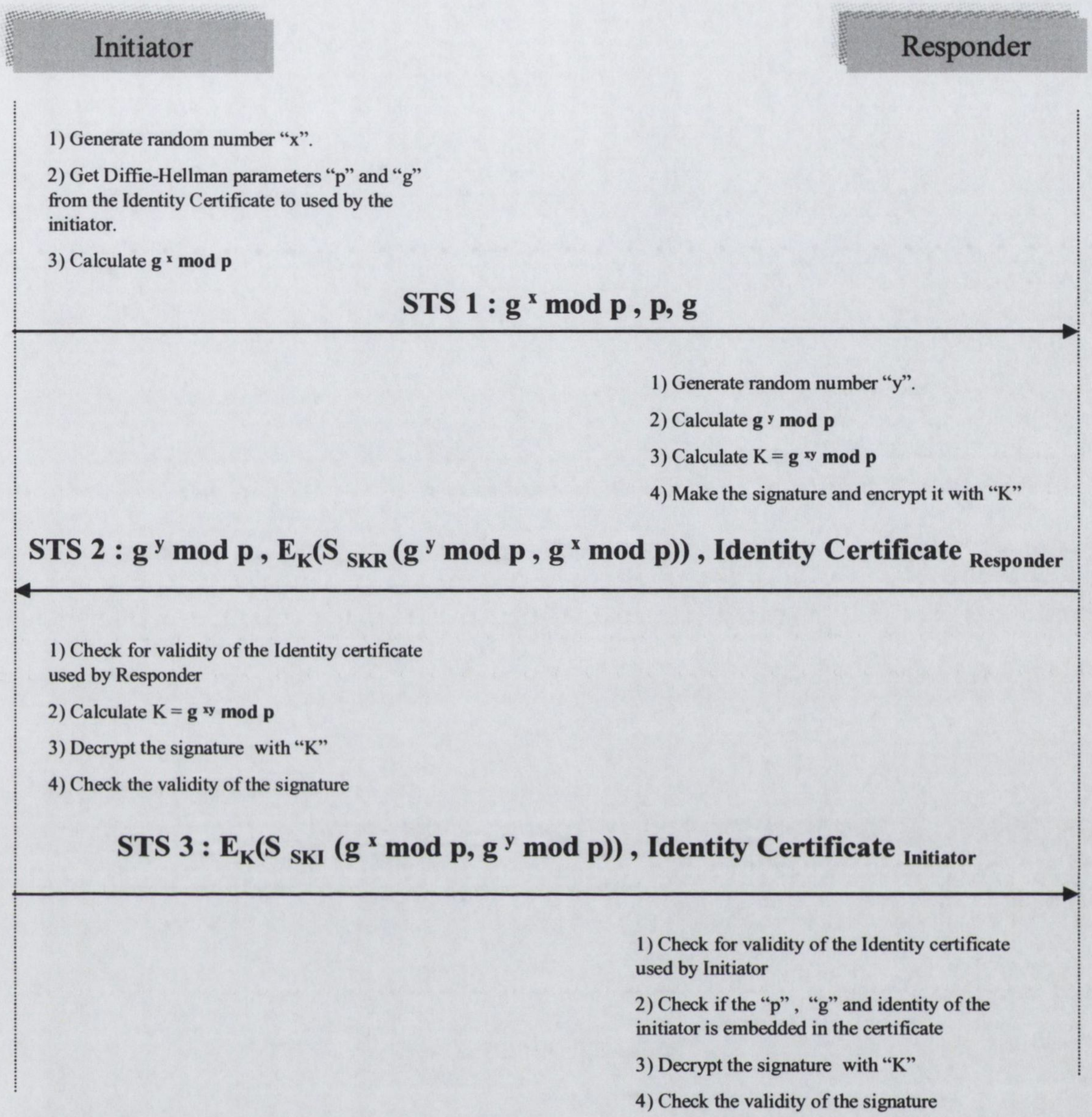
**PKI** – Uses certificates authenticated by Public Key Infrastructure

**Figure 11 : Comparison of Diffie-Hellman Key Agreement Protocols**

The asymmetric key agreement protocols that best satisfy the general requirements are the Internet Key Exchange version 2 (IKEv2) [ikev2] phase 1 and the Just Fast Keying (JFK) [abb+04] suite. These two protocols combine cryptographic primitives in a novel way to provide protection against the DoS type attacks. If, during the execution of the IKEv2 phase 1, a DoS attack is detected then an additional stateless exchange of cookies [o98] is done by using two protocol messages. Thus, the DoS mitigation in IKEv2 phase 1 increases the number of messages to six. This violates the performance goal of the minimum number of passes in the protocol interchange. Similarly, the DoS protection is provided by the Internet Key Exchange version 1 (IKEv1) protocols main mode of operation by using six message exchanges. However, the Just Fast Keying (JFK) protocol addresses the problem of DoS by proposing two variants each having four message exchanges. The JFKi variant protects the initiator of the protocol against DoS attack while the JFKr variant protects the responder. The low number of messages in JFK variants compared to the IKEv2 and IKEv1 is achieved by combining many cryptographic primitives in a single message. This increases the complexity of the key agreement process and results in large messages (specifically the third message) during the protocol interchange. A common characteristic of the IKEv2, IKEv1, JFKi and JFKr protocols is that they are based on the Station-To-Station (STS) [dow92] protocol. A drawback of the STS protocol (Figure 12) is its inability to defend against DoS attacks launched against the initiator and responder. However the STS protocol satisfies most of the other general and performance requirements. Therefore the STS protocol is the compromise key agreement scheme selected in this thesis. The STS protocol satisfies the following requirements:

- **General Requirements** – Entity authentication, Mutual authentication, Key freshness, Implicit key authentication, Key confirmation and Forward secrecy. The key compromise impersonation is provided if the certificate revocation information is up-to-date.
- **Performance Requirements** – Minimum pre-configuration, Minimal number of messages exchanged, Low communication overhead, Low computational overhead, Non reliance on time stamping and no negotiation of capabilities in the protocol.

The protocol exchange of the original STS protocol is presented in the Figure 12. This is followed by a discussion on the properties of the STS protocol.



Initiator

Responder

1) Generate random number “x”.

2) Get Diffie-Hellman parameters “p” and “g” from the Identity Certificate to used by the initiator.

3) Calculate **$g^{x}$ mod p**

**STS 1 : $g^{x}$ mod p , p, g**

1) Generate random number “y”.

2) Calculate **$g^{y}$ mod p**

3) Calculate K = **$g^{xy}$ mod p**

4) Make the signature and encrypt it with “K”

**STS 2 : $g^{y}$ mod p , $E_{K}$(S $_{SKR}$ ($g^{y}$ mod p , $g^{x}$ mod p)) , Identity Certificate $_{Responder}$**

1) Check for validity of the Identity certificate used by Responder

2) Calculate K = **$g^{xy}$ mod p**

3) Decrypt the signature with “K”

4) Check the validity of the signature

**STS 3 : $E_{K}$(S $_{SKI}$ ($g^{x}$ mod p, $g^{y}$ mod p)) , Identity Certificate $_{Initiator}$**

1) Check for validity of the Identity certificate used by Initiator

2) Check if the “p” , “g” and identity of the initiator is embedded in the certificate

3) Decrypt the signature with “K”

4) Check the validity of the signature

Notation Used:

p and g – Diffie- Hellman parameters generated by the initiator and embedded into the identity certificate of the initiator.

SKI and SKR– Private keys of the Identity Certificates used by the initiator/responder

x and y – secret Diffie-Hellman keys of initiator and responder respectively

K = $g^{xy}$ mod p = $(g^{y})^{x}$ mod p = $(g^{x})^{y}$ mod p is the secret key between the entities

E $_{Key}$ (message) – Symmetric encryption of the “message” using the Key

S $_{Key}$ (message) – Asymmetric signing using key on hash of the “message”.

Shared Information between the Initiator and the Responder:

Encryption / Signature / Certificate algorithms used

**Figure 12 : Station-to-Station (STS) Protocol**

The main premise in the STS protocol is that an attacker cannot compute the value of private $x$ from the public $g^x$ *mod p* despite knowing the values of the prime number $p$ and the generator $g$ (also may be termed as the Diffie-Hellman assumption). A protocol interchange is started with the initiator generating a random value $x$ and then computing the public value $g^x$ *mod p*. The values of $g$ and $p$ are pre-generated and embedded into the identity certificate used by the initiator. Then the public values $g^x$ *mod p*, $p$ and $g$ are sent to the responder using the STS1 message. These values of $p$ and $g$ have to be chosen [openssl, z00] carefully as an improper choice can lead to a small-subgroup cryptanalysis attack. The responder on the receipt of the STS 1 message generates the private $y$ and uses it to compute public value $g^y$ *mod p* and the common key $K = (g^x \text{ mod } p)^y = g^{xy}$ *mod p*. Then the signature on the public values $g^x$ *mod p* and $g^y$ *mod p* is encrypted using the key $K$. This ensures that the initiator can check that the STS 2 message has not being tampered with by a third party. The encryption of the signature in STS 2 ensures that the initiator gets an implicit confirmation that it has the correct key K. The STS 3 message confirms to the responder that the initiator has the right common key K. Moreover, the messages STS 2 & STS 3 ensure that the initiator and responder mutually authenticate each other. It is important to note that $p$ and $g$ along with the identity of the initiator are embedded in the certificate used by initiator for authentication. This prevents a man-in-the-middle attacker changing the values of the $p$ and $g$ and impersonating the initiator.

A major criticism of the STS protocol is that it only provides implicit key confirmation using encryption. This was mitigated in a variant which used Message Authentication Code (MAC) [kbc97] function in the second and third messages of the STS protocol to provide explicit key authentication (similar to the second protocol in Figure 13). This increases the length of the second and third messages. Over time, attacks on the STS protocol with the MAC variant were found which necessitated minor modifications to the protocol. In the unknown key-share (UKS) attack, [l96] an attacker can hijack the public key of the initiator or responder identity certificates used in the protocol interchange. The attacker gets a certificate incorporating the hijacked public key. This allows the attacker to impersonate the initiator or responder using the spurious certificate and mount a man-in-the-middle attack. Though the common key is not compromised in this attack, the belief of the initiator that it is communicating with the responder and vice-versa is inaccurate. The attacker simply relays messages between the initiator and responder. A simple solution to prevent the UKS attack is to include the identifier of the initiator ($I_{initiator}$) and responder ($I_{responder}$) in the signatures, providing the protocol with the strong entity authentication property. These identifiers have to be same as those embedded into the identity certificates. However this kind of public key substitution UKS attack can be prevented completely by making it mandatory for the CA issuing the certificate to ask for proof of possession of the secret key during the certification process. Most researchers nowadays tend to ignore this public key substitution attack as most CA's have some kind of proof of ownership of the secret key built into the certification process.

However another type of UKS attack can be mounted on the STS protocol MAC variant due to exposure of the signatures. This attack is based on the fact that it is possible to have the same signature verifiable using a different public/secret key pair. This *duplicate signature* [wm99] property is possible in most of the public key cryptosystems (including the widely used RSA). To perpetuate an attack of this type, an attacker has to generate the new public/secret key pair and get it certified in real time (during the run of the STS protocol interchange) from a CA. However this duplicate signature UKS attack is difficult against the STS variation using encryption (similar to the first protocol in Figure 13) since the signature is not exposed. Through this attack can be launched against the STS using the encryption variant with a low probability of success if the attacker has "*A complete specification of the underlying symmetric-key encryption and signature schemes, together with a statement of the security properties*". In such cases of complete knowledge of specifications the probability of the key compromise can be further decreased if the certificates used in the protocol are encrypted by the shared key along with provision of flow numbers in the message [wm99].

Some researchers preferred the use of a MAC as the export of strong encryption algorithms faced restrictions by some governments. This is no longer true as several versions of strong encryption are available freely which have reasonable performance on the commodity handheld device (see Appendix A). Therefore due to the small message size and easy availability of strong cryptographic mechanism, the STS variant using encryption is chosen. This protocol is shown in the first protocol of the Figure 13 and is analysed in the next section.

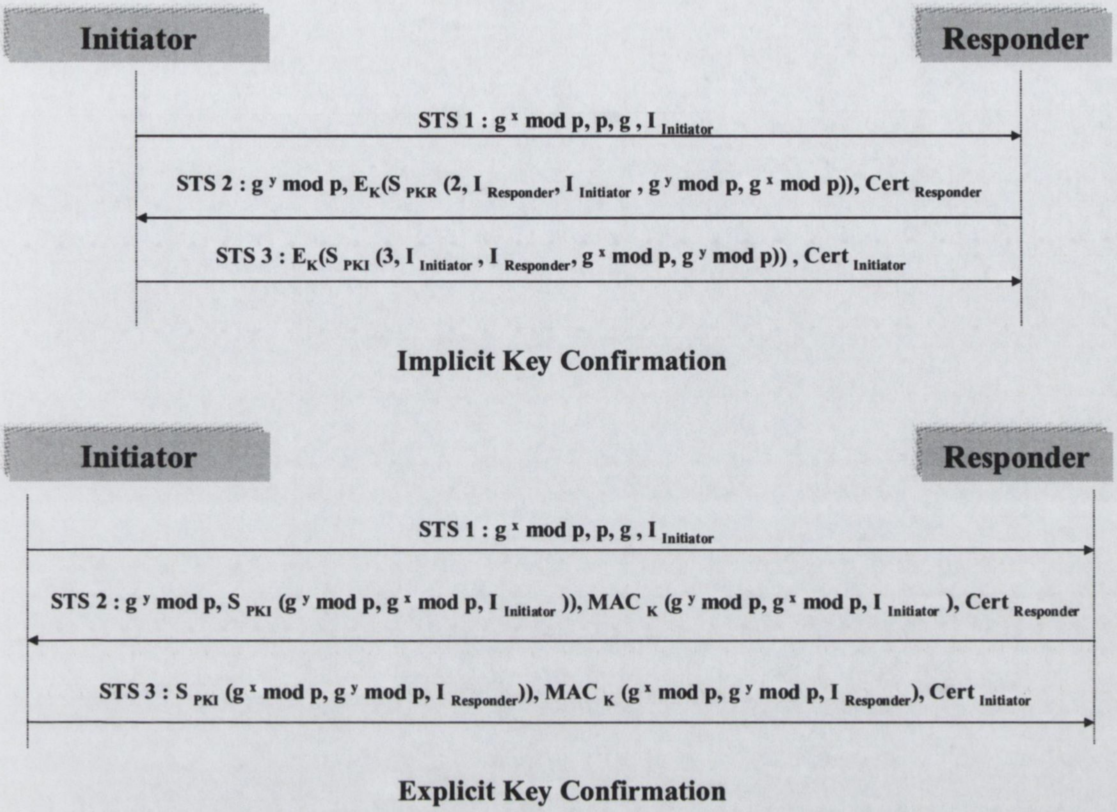

Initiator
Responder
STS 1 : $g^x$ mod p, p, g , $I_{Initiator}$
STS 2 : $g^y$ mod p, $E_K(S_{PKR}$ (2, $I_{Responder}$, $I_{Initiator}$ , $g^y$ mod p, $g^x$ mod p)), $Cert_{Responder}$
STS 3 : $E_K(S_{PKI}$ (3, $I_{Initiator}$ , $I_{Responder}$, $g^x$ mod p, $g^y$ mod p)) , $Cert_{Initiator}$
Implicit Key Confirmation
Initiator
Responder
STS 1 : $g^x$ mod p, p, g , $I_{Initiator}$
STS 2 : $g^y$ mod p, $S_{PKI}$ ($g^y$ mod p, $g^x$ mod p, $I_{Initiator}$ )), $MAC_K$ ($g^y$ mod p, $g^x$ mod p, $I_{Initiator}$ ), $Cert_{Responder}$
STS 3 : $S_{PKI}$ ($g^x$ mod p, $g^y$ mod p, $I_{Responder}$)), $MAC_K$ ($g^x$ mod p, $g^y$ mod p, $I_{Responder}$), $Cert_{Initiator}$
Explicit Key Confirmation

**Figure 13 : Modified STS Protocol**

### *2.2.1.4 Analysis of the Modified STS protocol*

The analysis starts with the presentation of the state machine (proposed by author of the thesis) for the Modified STS protocol in Figure 14. The start state for a protocol interchange is the "Idle" state on the Initiator/Responder. On the responder node the state machine for a STS interchange starts on receipt of a STS 1 message. If the responder finds the STS 1 message valid, then a STS 2 message is sent to the initiator and the state machine for that negotiation transits to the "Awaiting STS3" state. Otherwise if "p" or "g" or "$g^x$ mode p" is invalid [z00] then the state machine returns to the "Idle" state. The creation of the STS 2 message is time intensive as it requires a private key signature operation. Thus the CPU-Resource exhaustion DoS attack can be mounted on the "Received STS1" state. Moreover, if the device has limited memory capacity then Memory-Resource exhaustion DoS attack is also possible as the responder keeps a state entry for each valid STS 1 message received. The next state transition occurs from the "Awaiting STS3" state to the "Received STS3" state when a STS3 message is received by the responder. This transition can continue to the "Idle" state if the STS 3 message is invalid. Still, the responder has expended CPU cycles in answering the STS 1 message for an interchange which may turn out to be malicious. Moreover, the responder also has to perform a CPU intensive computation to verify the STS 3 message. Thus both the CPU and memory exhaustion DoS attack can be mounted on the "Received STS 3" state.

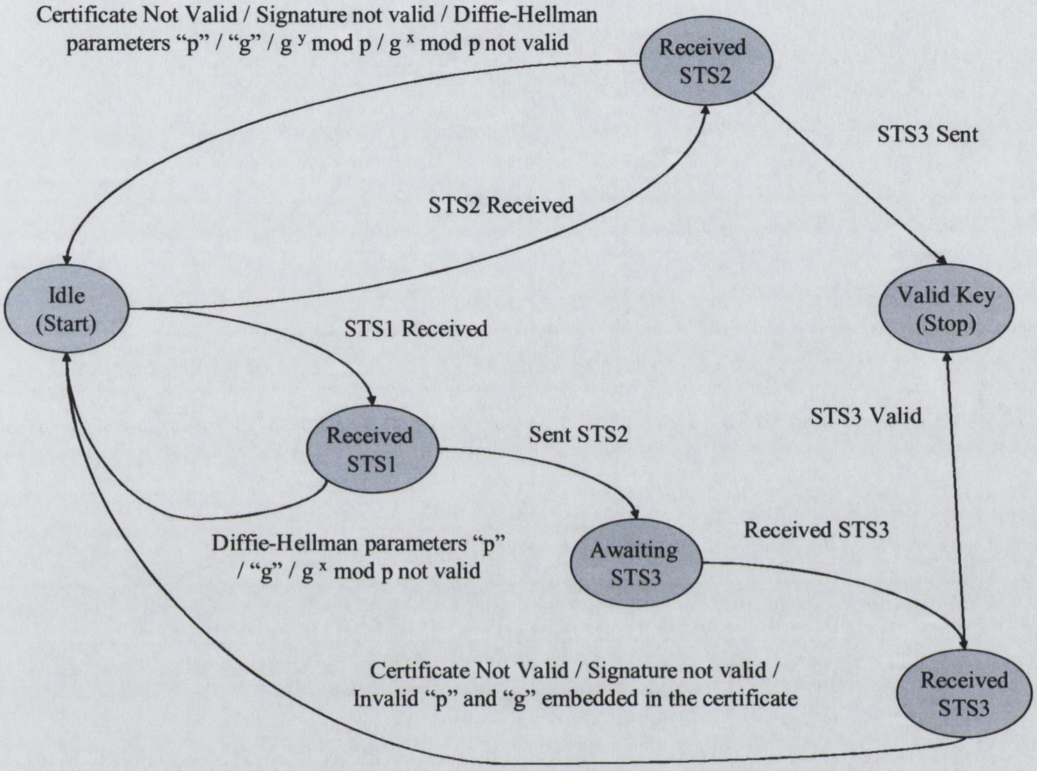

Certificate Not Valid / Signature not valid / Diffie-Hellman parameters "p" / "g" / g $^y$ mod p / g $^x$ mod p not valid
Received STS2
STS3 Sent
STS2 Received
Idle (Start)
Valid Key (Stop)
STS1 Received
STS3 Valid
Received STS1
Sent STS2
Received STS3
Diffie-Hellman parameters "p" / "g" / g $^x$ mod p not valid
Awaiting STS3
Certificate Not Valid / Signature not valid / Invalid "p" and "g" embedded in the certificate
Received STS3

**Figure 14 : State Diagram of the Modified STS protocol**

It is assumed in the state diagram that the initiator sent out the STS 1 message to the responder. Therefore, the state machine starts on the initiator after the receipt of a STS 2 message. A valid STS 2 message ensures that a STS 3 message is sent to the responder. Then the state machine transits to the state "Valid key" from the "Received STS2" state. However if the STS 2 message is invalid then the state machine transits back to the "Idle" state. As verification of the STS 2 message is computationally expensive, a CPU resource exhaustion DoS attack can be mounted against the "Received STS2" state.

An assumption in the aforementioned state machine analysis is that the authenticity of the identity certificates used in the protocol interchange can be checked at all times. This is not possible in Ad hoc networks, as up-to-date Certificate Revocation Lists (CRL) may not be available and an invalid certificate may be accepted for entity authentication.

There is also the possibility of the trusted Certificate Authority (CA) being compromised. Such a compromise can lead to unsuspecting entity to trust a malicious certificate being issued by the "cracked" CA. Another assumption in regard to the trusted CA is that it checks for the proof of procession of the private-key during the issue of an identity certificate.

The analysis of the modified STS protocol presented in this section is dependent on the fact that the cryptographic algorithms used in the protocol cannot be compromised. Specifically the Diffie-Hellman assumption, the RSA assumption (used by the identity certificates) and the assumption that symmetric key algorithm used in the protocol are not compromised. The Diffie-Hellman assumption can be compromised using the small sub-group attack [z00] if improper Diffie-Hellman variables are used in the protocol. In contrast, the published attacks against the RSA assumption have been able to factor up to a 576-bit modulus [rsa]. This is easy to remedy by choosing a larger modulus. In this thesis the Advanced Encryption Standard (AES) [dr01] algorithm is used for symmetric encryption against which there is no currently published workable attack.

Another drawback in the Modified STS protocol is the use of certificates in plaintext during the interchange. Thus an eavesdropping adversary knows the identities of the two participating entities. The solution suggested by Blake-Wilson et al [wm99] involving the encryption of certificates using the common negotiated key agreed is flawed. An adversary can easily obtain the identity certificates used by an entity by participating in a genuine key agreement. This makes it easy for an attacker to mount a known plaintext cryptanalysis attack against the common key if it is used to encrypt the identity certificates. Moreover, some recent attacks on the hashing algorithm make it possible with a low probability to generate meaningful colliding certificates. The term meaningful is important as the colliding certificate generated may not be useful for an attacker to perpetuate an attack. Another unlikely scenario not considered in the state machine analysis is the hijacking of an entity's hardware by attackers. This can lead to a host of attacks including identity theft and launching of DoS attacks against unsuspecting entities. The previously discussed strengths and weakness of the Modified STS protocol led to construction of a threat tree presented in Figure 15. The dotted lines in the tree connect interrelated branches of the threat tree together. This helps in condensing the threat tree of the Modified STS protocol.

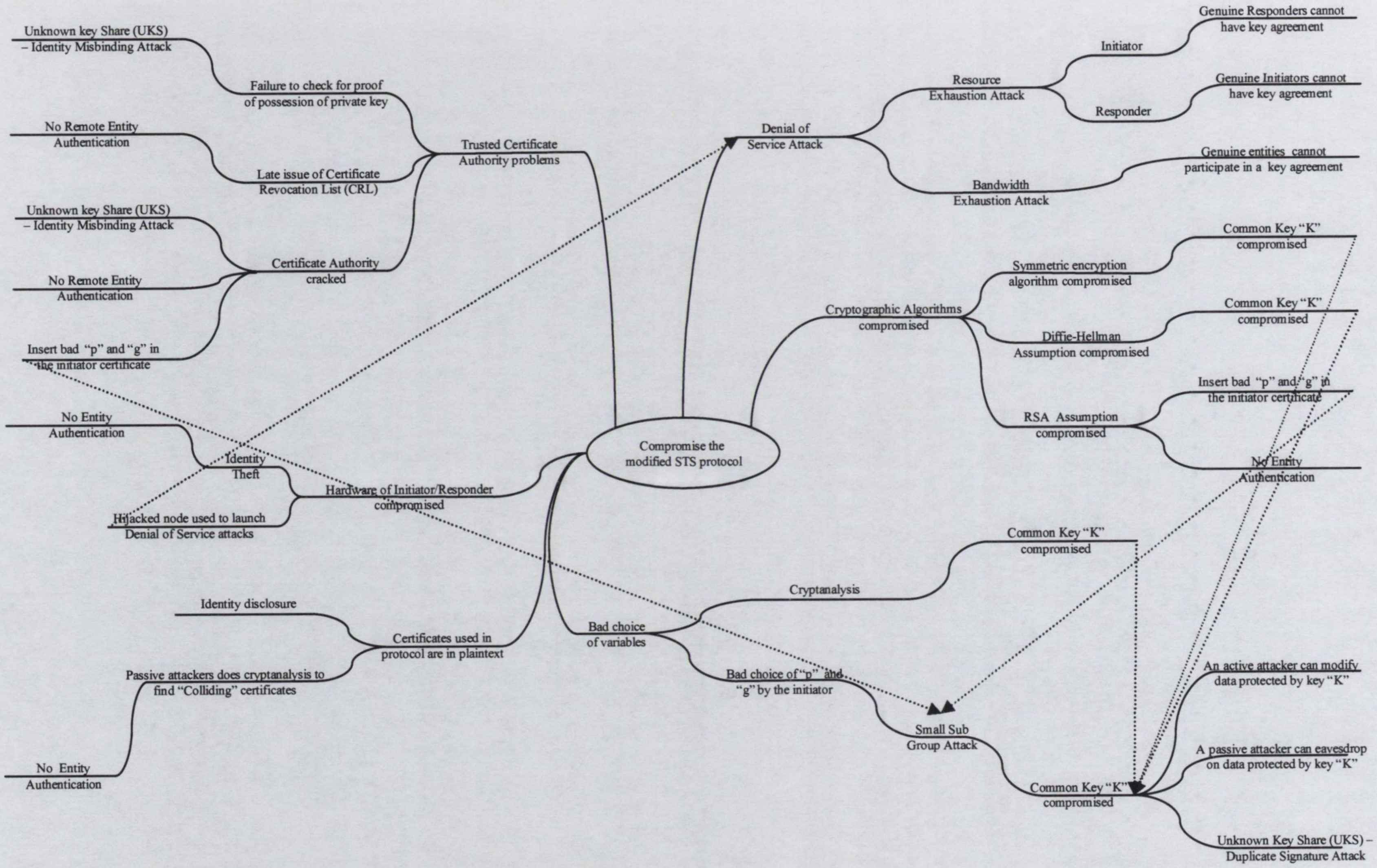

Compromise the modified STS protocol
Trusted Certificate Authority problems
Failure to check for proof of possession of private key
Unknown key Share (UKS) – Identity Misbinding Attack
Late issue of Certificate Revocation List (CRL)
No Remote Entity Authentication
Certificate Authority cracked
Unknown key Share (UKS) – Identity Misbinding Attack
No Remote Entity Authentication
Insert bad "p" and "g" in the initiator certificate
Hardware of Initiator/Responder compromised
Identity Theft
No Entity Authentication
Hijacked node used to launch Denial of Service attacks
Certificates used in protocol are in plaintext
Identity disclosure
Passive attackers does cryptanalysis to find "Colliding" certificates
No Entity Authentication
Denial of Service Attack
Resource Exhaustion Attack
Initiator
Genuine Responders cannot have key agreement
Responder
Genuine Initiators cannot have key agreement
Bandwidth Exhaustion Attack
Genuine entities cannot participate in a key agreement
Cryptographic Algorithms compromised
Symmetric encryption algorithm compromised
Common Key "K" compromised
Diffie-Hellman Assumption compromised
Common Key "K" compromised
RSA Assumption compromised
Insert bad "p" and "g" in the initiator certificate
No Entity Authentication
Bad choice of variables
Cryptanalysis
Common Key "K" compromised
Bad choice of "p" and "g" by the initiator
Small Sub Group Attack
Common Key "K" compromised
An active attacker can modify data protected by key "K"
A passive attacker can eavesdrop on data protected by key "K"
Unknown Key Share (UKS) – Duplicate Signature Attack

Figure 15 : Threat Tree of the Modified STS protocol

The diagram represents a simplified version of the threat tree where the branches for the active and passive attacker are not expanded further. An active attacker interferes with the protocol run while a passive attacker only eavesdrops on the communication. It is difficult to enumerate all the classes of attacks an active attacker can perpetuate. However, but a partial list is presented below,

- **Modification** – The attacker modifies the protocol primitives so that the integrity of the protocol is compromised. The man-in-the-middle types of attack are a special type of modification attack.
- **Replay** – The protocol messages are replayed maliciously afterwards by an attacker to compromise the integrity of the protocol. The reflection attacks are a special type of the replay attack in which an adversary replays the message in a protocol run immediately. The use of identifiers "2", "3" and identities of the participating entities in the primitives used in the Modified STS protocol interchange guards against this class of attacks.
- **Typing attack** – This type of attack is perpetuated when an attacker changes the fields in the protocol primitives. The typing attack against the Modified STS is unlikely as any change in the network primitives can invalidate the protocol interchange due to use of identifiers "2" and "3" in authenticated parts of the primitives.
- **Certificate Manipulation** – The certificates used in the protocol are modified or changed to compromise the integrity of the protocol.
- **Protocol Interaction** – Another protocol is used to attack the current protocol or data of the current protocol. There has been no such published attack of this type on the Modified STS protocol.
- **Denial of Service** – The protocol exchange is denied to bona-fide entities by malicious attacker(s). Attacks of this type are a strong possibility as demonstrated in the state machine analysis.

It is important to note that nearly all the active attacks require eavesdropping to succeed. In contrast to an active attacker, the passive attacker eavesdrops on the communication and then tries to apply cryptanalysis to find weaknesses in the protocol or tries to compromise keys to understand the encrypted traffic.

An improper threat analysis of the cryptographic primitives used in a protocol can lead to devastating effects. For example the widely used Wireless Encryption Protocol (WEP) [802.11] which relies on pre-distribution of a shared key between the wireless mobile station and access point has been shown to have major security flaws [fms01]. The access point sends a challenge in clear text to the wireless node. This plaintext challenge is encrypted by the wireless mobile station using the pre-arranged shared secret. The encrypted challenge is sent back to the access point. If after decryption the access point finds decrypted text that matches the initial challenge then the wireless mobile station is authenticated and is allowed to use the services. The same shared secret is also used to encrypt all transmissions between the wireless mobile node and access point. The plaintext and encrypted text of the initial challenge can be easily overheard by an passive attacker along with encrypted traffic. An attacker can then feed the plaintext challenge, the encrypted reply and encrypted communications into a brute force cracker and get the shared secret. The time of the cracking depends on the size of the shared secret. After breaking the shared secret an intruder can easily commit active or passive attacks [sir01].

### 2.2.2 Conclusion of the Key Establishment Between Two Entities

In an Ad Hoc network environment the authenticated key establishment process has to compromise between the general requirements and performance goals necessitated by the Ad Hoc environment. The major compromise made in this thesis is in balancing the need for DoS protection against the number of message exchange required during the protocol. After analysis, the Modified STS protocol is found to be the most suitable compromise candidate for authenticated key agreement between two entities in an Ad Hoc network deployed in an emergency scenario. The key agreed upon is used to ensure the confidentiality and integrity of protocols used in the One-to-One trust negotiation and the secure group collaboration schemes. Moreover, this key protocol also provides authentication information for the access control process on the two participating entities.

# 3 Secure Group Formation and Communication in Ad Hoc Networks

Secure group collaborations are formed in an Ad Hoc network when a number of nodes come together to perform certain tasks or share resources in a confidential manner. Such collaborations face two major challenges in Ad Hoc networks. The first challenge is to have an access control mechanism to determine membership without relying on access to a TTP. Secondly, a secure group message propagation mechanism which is robust and fault-tolerant for dynamic network conditions caused by membership and topological changes. Hence, this chapter starts with a discussion of secure group formation approaches in Ad Hoc networks. This is followed by an examination of the multicasting techniques for Ad Hoc networks, illustrating the issues involved in ensuring an efficient and fault-tolerant one-to-all group message delivery.

## *3.1 Secure Group Formation in Ad Hoc Networks*

Traditional approaches for secure group formation in wired conventional networks have focused on ensuring maximum availability and fault tolerance. This has resulted in diverse schemes like the ISIS [br94s], Horus [rbm96], Transis [adk92], Totem [mma96] and RMP [wmk94]. The improvement in network speed and reliability has shifted the emphasis to providing a robust membership access control in schemes like the Rampart [r94], SecureRing [kmm98], Horus/Ensemble [rbd01], Spread [as98] and Antigone [dp00]. An attempt at standardization was made with the IETF sponsored Group Secure Association Key Management Protocol (GSAKMP) [hch+00] framework. This framework defined a generalised architecture for secure multicast groups but left the definition of authentication (for access control) to the implementer. The aforementioned schemes are designed for a conventional wired network and rely on guaranteed access to an online TTP for authentication which may not be possible in Ad Hoc networks. Hence new techniques were developed for Ad Hoc networks, with the research concentrating on two different approaches. The first approach focused on the mechanics of admission control and group management while the other approach has concentrated on the distribution of content among the group members in a confidential manner (i.e. secure multicasting). The two major schemes which employ the first approach are the Peer Group Admission Control [kmt03] and Robust Membership Management for Ad-hoc Groups [mah00].

### 3.1.1 Peer Group Admission Control

The ongoing Peer Group Admission Control project [kmt03] at SCONCE, Department of Information and Computer Sciences, UC Irvine proposes a group admission control scheme for the peer-to-peer networks (i.e. networks with no central entities). This scheme has three sub-schemes, giving the network the ability to cope with diverse scenarios encountered in a peer-to-peer environment (similar to Ad Hoc networks). In the first sub-scheme the group is assumed to have a fairly static membership. An Access Control List (ACL) is defined at group creation time, and is cryptographically signed to ensure authenticity. This ACL is pre-distributed to all members and contains references to the identity certificates of all members. Any message signed by an entity whose identity certificate is referenced in the group ACL is deemed a valid group message. A new entity joining the group requires its identity to be added to the group ACL. This new group ACL is then re-distributed to all group members. However priming with an ACL is inflexible and restricts the scalability of the group. To improve the scalability of the group join process, the second sub-scheme relies on one or more online designated members trusted by all present and potential members. These designated members authenticate a new member and then issue it with a group membership certificate entitling the new member to join the group. Thus these designated nodes act as a certification authority and the resulting group hierarchy is flat. The flat hierarchy still restricts the scalability and some members have to bear the additional overhead of the admission control process. In the third sub-scheme the scalability is further increased with the work of admission control distributed among the existing members. This sub-scheme uses three different methodologies for collaboration among existing members to add a new member.

- In the first method a fixed threshold number of existing members should “sponsor” a new member to join the group.

- The second method involves a percentage of the existing members to sponsor the new member.
- The third method is a hybrid of the first two methods.

In all the methods of the third sub-scheme, a high degree of coordination among existing members is required. Moreover, a precise count of the existing members has to be maintained for the "sponsoring" process to work. The high degree of coordination and precise membership count is difficult to achieve in an Ad Hoc network with dynamic membership and topological conditions. After the admission control process is over a new member is able to send a message to all other group members in a confidential manner. This is achieved by all members of the group taking part in a key agreement process using the Group Diffie-Hellman protocol (GDH) suite [stw98] to agree to a group key. The forward security of the group key is maintained by negotiating a new key every time a group member leaves/joins. However renegotiating the group key using the GDH suite can cause big overheads in a large dynamic Ad Hoc group exhibiting frequent membership changes.

### 3.1.2 Robust Membership Management for Ad-Hoc Groups

A robust membership management scheme for Ad Hoc groups proposed by Mäki et al [mah00] from the Laboratory for Theoretical Computer Science, Helsinki University of Technology, Finland relies on certificates to define group membership. In this scheme a group leader acts as the CA and issues group membership certificate. This certificate entitles a member to join the group. To improve the scalability, a leader can appoint sub-leaders. These sub-leaders can admit new members and appoint further sub-leaders. This creates a tree hierarchy similar to an X.509 Public Key Infrastructure. However, the organization of the certification authorities in a tree hierarchy to certify membership has two major drawbacks in Ad Hoc networks. Firstly, certificate revocation is difficult and cumbersome. Secondly, frequent generation of strong public-private key pairs is CPU intensive on commodity mobile wireless devices. The authors of this scheme do not divulge their thoughts on the admission control and how the group messages will be propagated. Thus this scheme is a group membership management system and not a complete secure group communication scheme.

These two schemes concentrated on admission control and membership management aspects of the secure group formation problem. A further important aspect of secure group formation to be considered is the distribution of group content in a confidential manner. This was addressed by the secure multicasting community and their approaches are discussed in the next section.

### 3.1.3 Secure Multicasting in Conventional and Ad Hoc Networks

The problem of secure multicasting has been studied in depth over the last two decades. In conventional wired networks secure multicasting schemes could be broadly classified into two distinct approaches. The static key approach is based on the assumption that a group key is generated once during the lifetime of the group. Such an approach is used in schemes like the Group Key Management Protocol (GKMP) [hm97], Simple Key-Management for Internet Protocols (SKIP) [cla+96], Internet Security Association and Key Management Protocol (ISAKMP) [mss97], Oakley Key Determination Protocol [o98], and Scalable Multicast Key Distribution Scheme (SMKD) [b96]. In these schemes if the group key is compromised by an attacker, the group has to be reformed leading to big overheads.

In contrast to the static key approach, the dynamic approach uses a changing group key usually distributed securely by a central entity. The change of the group key may be prompted by members leaving or joining. In schemes like the Fiat-Naor Broadcast Encryption [fn93] and Secure Lock [cc89], the central entity generates and distributes the group key. This means that the central entity has a security association with each member for group key distribution. Thus, this approach can result in huge storage overheads on the central entity if the group is large. A variation of this centralized approach to decrease the storage requirements is the Iolus [m97] scheme. In this scheme the group is divided into sub-groups with each of these having a central entity. It is the responsibility of these designated sub-group central entities to coordinate among themselves to pass the group messages in a confidential manner. This concept of dividing the group key management among several entities is also used by Cliques [stw97]. However, in a dynamic Ad hoc network, extensive coordination among the designated entities is difficult to achieve.

A scheme which has variants ranging from a tightly controlled centralized approach to an Ad Hoc like situation with every member making its own decision, is the Versakey Framework [wcs99]. The first variant of this framework uses a centralized tree-based key management in which a hierarchy of key-encryption keys and group key is maintained. In Figure 16 the key arrangement is shown for a group with eight members ($M_1$ to $M_8$) having the group key $K_0$. The hierarchy of key encryption keys is used to distribute the group key to the members. In the example, each member has knowledge of four key-encryption keys and the group key. Therefore in the case of a member leaving or joining, only five keys need to be changed (i.e. the four key-encryption and group keys). This variant is good for scenarios with dynamic membership but places heavy load on the group controller member.

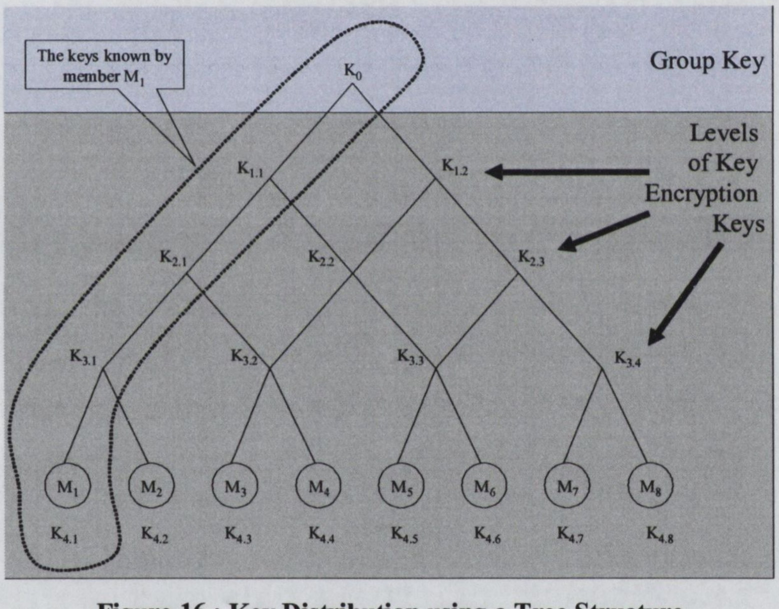


**Figure 16 : Key Distribution using a Tree Structure**

To ease this load, in the second variant the group controller having a key encryption key for each member. In this variant, the key distribution structure is centralised and flat. This decreases the scalability of the second variant. The third variant has a fully distributed structure with special members responsible for admission control along with the propagation of the group key and messages. In this flavour each member periodically maintains the neighbour node status through a "heartbeat" mechanism. This variant resembles an Ad Hoc network scenario as each member constructs its own view of the secure multicast group and does not rely on a central entity. However, the distributed structure makes it difficult for the group to cope with malicious entities. Another drawback of the third variant is its inability to handle multiple members leaving or joining the group in a small time interval.

The secure multicasting approaches described above are not suitable for use in Ad Hoc networks. Thus a number of new multicasting techniques for Ad Hoc networks has been developed. A scheme proposed by Lazos et al [lp03b] uses location information (mainly using the Global Positioning System) to build a tree for key distribution (similar to Figure 16). Initially the participating entities are distributed in interconnected clusters spread throughout the Ad Hoc network. Then the group controller (which may be a single or a distributed entity) builds an optimal key distribution tree using the location information. In another closely related scheme [lp03a] by the same authors, the criteria for the key distribution tree changes. Instead of location information, the new criterion is the energy expended by a node to build the key distribution. Another scheme using the key distribution tree (similar to Figure 16) in conjunction with directed diffusion [ige00] is the Logical Key Hierarchy for Wireless sensor networks (LKHW) [pml+03]. In this scheme the designated group member sends out an advertisement in the form of a broadcast looking for new members. The interested entities reply to this advertisement. If this reply contains valid authentication information, a join process occurs. At end of this interchange the new entity is allocated a place on the key distribution tree. This enables the new member to send encrypted messages only understood by other members of the multicast group. This approach of using the key distribution tree lacks scalability for a large and dynamic multicast group. Moreover, the extensive coordination required for operation of the key distribution may be difficult to achieve in an Ad Hoc network with dynamic topology.

A secure multicasting scheme designed for Ad Hoc networks using certificates for admission control mechanism was proposed by Lin et al [ln03]. In this scheme the group is structured as a tree and a new entity wishing to join the group broadcasts a request and may get back several replies from the pre-existing group members. The new entity then has the join process with the group member with the best aggregate path quality. The best aggregate path quality is calculated as a function of the hop count of the path, number of group multicast members in the path, path quality to the source and number of members already connected to the group member with which the join process is attempted. The join process involves mutual authentication, common key establishment and checks to determine if the new entity has a service-access certificate allowing it access to the multicast service. Although this service-access certificate is pre-issued, it can be revoked in the lifetime of the group by propagation of an appropriate CRL to the members. Another feature of this scheme aimed at mobility scenarios involving frequent link breakages is a K-out-of-N coding approach for link resiliency. This implies that a member attaches itself to N nodes in the tree and has to receive only K (<N) messages to correctly get the group key. The group key is used to encrypt/decrypt group messages for secure distribution among members. This K-out-of-N approach also increases the security, as a new entity has to authenticate itself with more than one existing member. In contrast to the complex join process the leave process of a member from the group is simple. The member leaving tells the members above and below in the tree hierarchy of the intended leave and then leaves the group. This leave process allows the members below and above to reconstruct the group tree structure.

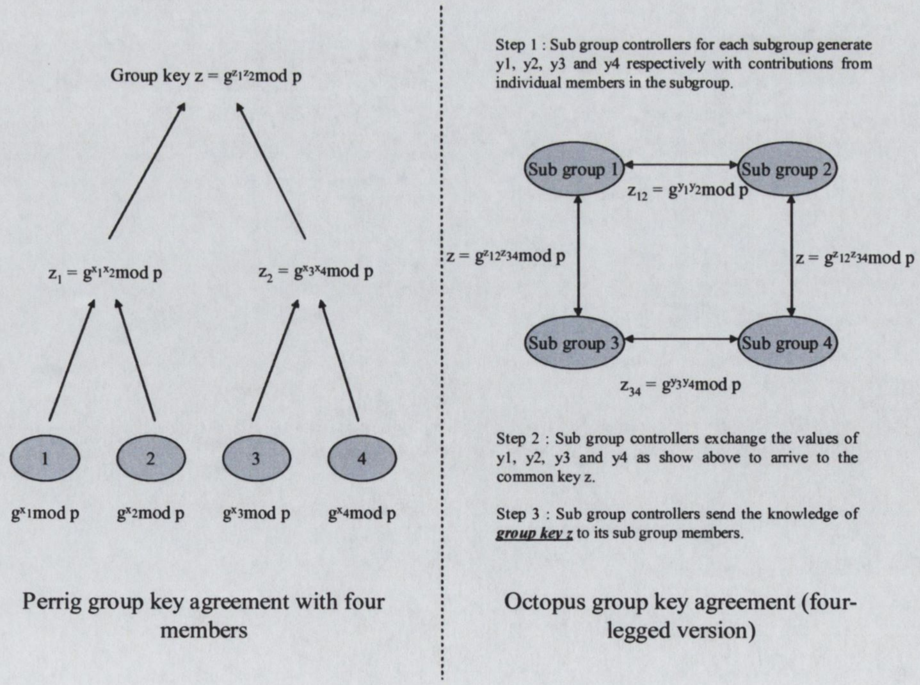
Tree based secure multicasting schemes in general face the problem of survival of the key distribution structure under unexpected topology or membership changes. These schemes can deal with expected group leaves but when a member leaves involuntarily due to link failures then the reconstruction of the tree structure by the group controller can be a time intensive process. This can lead to DoS to the genuine group members and late updating of the group key, making the multicast group susceptible to external attackers. Several other proposed schemes [yd02] concentrate on key agreement among group members. Instead of going into individual details of such schemes, the next section examines the bigger issue of key establishment between multiple entities in an Ad Hoc networks.

#### *3.1.3.1 Key Establishment Between Multiple Entities*

A common key agreed among members of a group allows for confidential one-to-all member communications. This common key can be established between multiple entities using two approaches. All the participating entities contribute to the common key in a group key agreement approach. This approach can be further categorised depending on the use of the Diffie-Hellman concept. However, in the group key transport approach, the group key is generated by one or more designated members and transported securely to all other members. In this section the issue of authentication is not considered, as a member is assumed to be already authenticated by the group admission control mechanism.

### 3.1.3.1.1 Diffie-Hellman Group Key Agreement

The entities participating in the group key agreement generate their own private and public Diffie-Hellman values (i.e. member $m_i$ generates $x_i$ and computes $g\,^{x}{}_{i}\ mod\ p$ for $i = 1\ to\ n$ where $n$ is the number of members). The public values (i.e. g $^{x}{}_{i}$ mod p) can be combined in a numbers of ways to get a group key. One method of combination organises the members in a ring structure with each member $m_i$ communicating with $m_{i+1}$ or $m_{i-1}$ ($m_n/m_1$ communicates with $m_1/m_n$ to complete the ring) for the purposes of group key agreement. Such a ring topology is used by group key agreement schemes like the Ingemarson-Tang-Wong (ITW) [itw82] protocol, Group Diffie-Hellman (GDH) [stw98] suite of protocols, Steer-Strawczynski-Diffie-Wiener (SSDW) [ssd90] , Burmester and Desmedt with broadcasts (BDB) [bd94] protocol and Burmester and Desmedt without broadcasts (BD) [bd94] protocol. The Perrig protocol [p99] uses a tree structure along with Diffie-Hellman concept to agree to a group key. In Figure 17 a representative Perrig key agreement between four members is presented.



Step 1 : Sub group controllers for each subgroup generate y1, y2, y3 and y4 respectively with contributions from individual members in the subgroup.

Step 2 : Sub group controllers exchange the values of y1, y2, y3 and y4 as show above to arrive to the common key z.

Step 3 : Sub group controllers send the knowledge of group key z to its sub group members.

Perrig group key agreement with four members

Octopus group key agreement (four-legged version)

**Figure 17 : Perrig and Octopus Group Key Agreement Protocol**

Another way of using the generalized Diffie-Hellman key agreement was suggested in the Octopus protocol [bw98] which divided the group into sub groups for ease of computations. A four-legged version of the Octopus protocol is shown in Figure 17. It is called four-legged as the members are divided into four sub groups containing nearly the same number of members. A more complex version of octopus protocol which divided the members into more than four sub groups was also proposed by Becker and Wille [bw98]. These group key agreement protocols using the Diffie-Hellman concept are analysed in the table shown in Figure 18 which is adapted from the book "Protocols for Authentication and Key Establishment" [bm03]. This table shows that the computational requirement on an individual node is minimized by the Octopus and BD protocols. But this performance optimization comes at the expense of large numbers of directed member-to-member network messages. These directed network messages were reduced by using broadcasts in the GDH.3 and BDB protocols. The trade-off in this case was an increase in the number of rounds of the protocol interchange. One protocol round consists of all messages that can be sent in parallel by the members. Another interesting compromise between the computational efficiency, number of messages and number of rounds was achieved by the Perrig protocol. This protocol combined the broadcasting concept with the idea of organization of the members into a tree structure. The table also shows that the best efficiency is achieved by the GDH.1 and GDH.2 protocols.

| Protocol | Number of exponentiations per node "$n_i$" | Messages sent by an node "$n_i$" | Total number of messages for key agreement | Number of broadcasts in key agreement | Number of rounds | Agreed Shared Secret |
|---|---|---|---|---|---|---|
| Ingemarson-Tang-Wong (ITW) | n | n-1 | n(n-1) | 0 | n-1 | $g^{x_1 x_2 x_3 \cdots\cdots x_n}$ |
| Group Diffie-Hellman version 1 (GDH.1) | i+1 | 2 | 2(n-1) | 0 | 2(n-1) | $g^{x_1 x_2 x_3 \cdots\cdots x_n}$ |
| Group Diffie-Hellman version 2 (GDH.2) | i+1 | 1 | n-1 | 1 | n | $g^{x_1 x_2 x_3 \cdots\cdots x_n}$ |
| Group Diffie-Hellman version 3 (GDH.3) | 3 for most nodes but n for last node "$n_n$" | 2 | 2n-3 | 2 | n+1 | $g^{x_1 x_2 x_3 \cdots\cdots x_n}$ |
| Steer-Strawczynski-Diffie-Wiener (SSDW) | n-i+2 | 2 | 2(n-1) | 0 | 2(n-1) | $g^{x_n (g^{x_{n-1} (g \ldots (g^{x_1 x_2})}})$ |
| Perrig | At least ⌈$\log_2 n$⌉ +1 | 1 | n | n-2 | ⌈$\log_2 n$⌉ | Nodes are organized in a tree structure to derive the shared secret |
| Octopus (four-legged version) | At least 4 | 3 | 3n-4 | 0 | 4 | Nodes are organized into four subgroups to derive the shared secret |
| Burmester and Desmedt with broadcasts (BDB) | At least 3 | 2 | 2n | n | 2 | $g^{x_1 x_2 + x_2 x_3 \cdots\cdots + x_n x_1}$ |
| Burmester and Desmedt without broadcasts (BD) | At least 3 | 4 | 4n-1 | 0 | 2n | $g^{x_1 x_2 + x_2 x_3 \cdots\cdots + x_n x_1}$ |

**Figure 18 : Performance Characterises of Generalized Diffie-Hellman based Protocols**

An interesting method of using Diffie-Hellman public values was proposed by Tzeng et al [tt00] in which the shared secret agreed after the protocol run is $g^{x_1+x_2+x_3+.....+x_n}$ instead of the normal $g^{x_1x_2x_3.....x_n}$. A common criticism of using the contributory Diffie-Hellman group key agreement mechanism presented in this section is that to achieve perfect forward security, the common key has to be re-negotiated whenever a member leaves or joins. Such a renegotiation can lead to large communication and memory overheads if the group is large. Moreover the ring or tree topology required for key agreement is difficult to maintain in a dynamic Ad Hoc network.

### 3.1.3.1.2 Non Diffie-Hellman Group Key Agreement

One notable scheme that does not use the generalized Diffie-Hellman concept was proposed by Pieprzyk et al [pl00]. The parts of the shared secret is pre-distributed to the members based on the threshold key distribution [gjk+96]. Each member only has a share of the global shared secret and any attacker has to compromise a *threshold number* of participating entities to gain access to the global shared secret. These individual shares of the shared secret are combined by the members to arrive at a group key. Thus the key agreement protocol can change the group key by changing the individual shares on the members. However this scheme has limited scalability as it requires pre-configuration and extensive coordination for operation.

Another prominent group key agreement protocol, proposed by Boyd [b97a], has some features common to a key transport protocol. This scheme relies on a central entity $m_1$ that all members $m_i$ $(i = 2 \ to \ n)$ trusts. The members $m_i$ $(i = 2 \ to \ n)$ have the public key of the member $m_1$. Moreover, $m_1$ has a key encryption key agreed with all other members $m_i$ $(i = 2 \ to \ n)$. Using this pre-configuration, the steps of the protocol are shown in Figure 19. The premise in this protocol is that without the knowledge of $r_1$, the group key cannot be calculated. Another similar scheme which replaces the MAC with a random oracle function was proposed by Boyd et al [bn03]. These schemes by Boyd and others do not provide forward security. This implies that if the long-term private keys of one or more of the entities involved in the protocol are compromised then the past/future/present group keys are compromised. These protocols have limited scalability as a central entity has to perform pre-distribution of key encryption keys in a secure way with other members. Moreover, if the central entity has an encryption key agreed with all members then it can simply transport a group key periodically. Thus the complicated protocol run using the MAC and random numbers are not required.

Step 1: a) $m_1$ generates random number $r_1$

b) $m_1$ broadcasts Set and $S_{SK\ m_1}$ (Set, $H(r_1)$) to all $m_i$ ($i = 2$ to n)

Step 2: $m_1$ broadcasts $E_{KEK12}(r_1)$, $E_{KEK12}(r_1)$, $E_{KEK12}(r_1)$, ..............$E_{KEK1n}(r_1)$.

Step 3: $m_i$ ($i = 2$ to n) generates random number $r_i$ broadcasts it to all $m_i$ ($i = 1$ to n) .

Step 4: $m_i$ ($i = 1$ to n) calculates group key = $MAC_{r_1}$( $(H(r_2))$ XOR $(H(r_3))$ XOR .......... XOR $(H(r_n))$ )

Notation used:

Members : $m_i$ ($i = 1$ to n) where $m_1$ is the central entity.

Set = Identity of all members $m_i$ ($i = 1$ to n)

$r_i$ is the random number generated by $m_i$ ($i = 2$ to n).

H (M)= One way hash function on M.

$MAC_K$ (M) = Message authentication code of M using the shared key K.

$S_K$ (M) = Signature on M using the secret key K.

$SK_{ml}$ = Secret key of the central entity $m_1$.

KEKij = Key encryption key agreed between members $m_i$ and $m_j$ .

**Figure 19 : Boyd's Group Key Agreement**

### 3.1.3.1.3 Group Key Transport Protocols

In group key transport protocols, a central entity generates the group key and sends it securely to all group members. To send the group key securely, the central entity and other members share a common key. Some schemes, using this philosophy, are the Burmester and Desmedt Star [bd94] protocol, Hirose-Yoshida key transport [hy98] protocol and Mayer and Yung family [my99] of protocols. Sharing a common key limits the scalability of the schemes in a large and dynamic Ad Hoc network. The central entity also has to keep track of membership, establish security associations with members and frequently update the group key. In a large dynamic Ad Hoc group keeping track of membership is a difficult and time consuming task. This also implies that the forward security is difficult to achieve in a group transport protocol as the group key needs to change with a membership change. Moreover, network traffic to and from the central entity can easily cause congestion.

### 3.1.4 Conclusion of Secure Group Formation in Ad Hoc networks

This section examined the secure group formation schemes for Ad Hoc networks and came to the conclusion that either these schemes have limited scalability or are infeasible to use on commodity handheld wireless devices with limited computational and battery power. Similarly, the secure multicasting schemes for Ad Hoc networks either lack good membership control mechanisms or are not robust enough to cope with dynamic membership and topology conditions. The related issue of group key establishment schemes for a large and mobile Ad Hoc network was also examined. The group key agreement schemes are not feasible as they can cause big overheads in a large or mobile group having frequent membership changes. The group key transport protocols, on the other hand, can result in big overheads for the central controlling entity(s). However, the group transport schemes require less coordination than the group key agreement schemes.

## *3.2 Group Communications*

Once a group is formed, the next important question is how the members will communicate with each other. This one-to-many multicasting problem has being studied in detail over the last two decades. Multicasting methods have improved considerably on wired conventional networks driven by the need to stream the same video and/or audio stream to more than one recipient at the same time. However new multicasting methods are required to cope with the dynamic nature of the Ad Hoc networks. In this section the evolution of multicasting in the conventional wired and Ad Hoc networks is presented. These multicasting schemes are also examined for their suitability in a large Ad Hoc networks deployed in an emergency scenario. It is important to note that in this section the discussion is limited to general multicasting techniques without security.

### 3.2.1 Overview of Multicasting in Conventional Networks

The multicasting in conventional wired networks has evolved considerably over last two decades. One of the most used multicasting schemes for conventional wired networks is the Internet Group Management Protocol (IGMP) [cdk+02]. In the IGMP scheme, a node registers its interest in a multicast data stream by notifying the local router. The local router also periodically checks for the presence of the receiver and senders nodes in its network. The routers in the IGMP scheme are also responsible to find the paths between the senders and receivers of the multicast group across the network. Another approach towards multicasting in conventional networks uses flooding. In the *flood and prune* method, a sender first floods the network to deliver the multicast group traffic. Appropriate routers in the network forward the traffic if they have valid receivers. The routers having no receivers send prune messages back to the sender to stop unnecessary traffic flowing. Thus the distribution of multicasting traffic is in form of a minimal pruned tree. This is called *reverse shortest path tree* and the algorithms of these type called the *flood and prune protocols* (Figure 20). The two most common examples of protocols using the *flood and prune* method are the Distance Vector Multicast Routing Protocol (DVMRP) [p03] and Protocol Independent Multicast Dense-Mode (PIM-DM) [fhh+04]. The DVMRP has its own unicast routing algorithm while the PIM-DM as the name suggest is independent of the underlying unicast protocol. These *flood and purne* protocols along with the IGMP protocol are not feasible for Ad Hoc networks as they rely on guaranteed online access to fixed central nodes (routers).

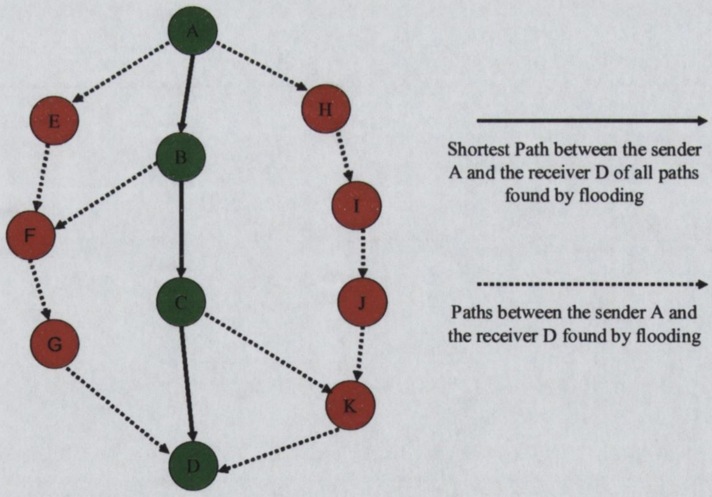

A
E
H
B
I
F
C
J
G
K
D
Shortest Path between the sender A and the receiver D of all paths found by flooding
Paths between the sender A and the receiver D found by flooding

**Figure 20 : Flood and Pruning of Routes in Multicasting Algorithms**

In contrast to the *flood and prune* method, the *center-based tree* based multicast algorithms rely on the routers explicitly build the multicast tree. The Core-Based Tree (CBT) [b97] algorithm is a scheme that uses the *center-based tree* approach. This scheme relies on a backbone of routers organized in a tree structure with the root of the tree being the core router. It is the responsibility of this tree to forward the multicast traffic to the senders and receivers. An entity wanting to join a multicast group attaches itself to the local CBT router which is on the tree hierarchy of routers. Therefore the multicast group has a tree structure with the core router as the root and senders/receivers as leafs. The CBT algorithm was further improved by the PIM Sparse-Mode (PIM-SM) protocol. Instead of the core in the CBT there is a similar rendezvous point (RP). The common feature of both the algorithms (i.e. CBT and PIM-SM) using the *center-based trees* approach is the maintenance of a distribution tree. This tree structure is difficult to maintain in a large and dynamic Ad Hoc network deployed in an emergency scenario.

The Multicast Open Shortest Route First (MOSPF) [m94] algorithm is another multicasting scheme in conventional network which uses flooding. In MOSPF, each router keeps a periodically updated map of the receivers and senders in the network by flooding of routing messages. Then using the Dijkstra algorithm [clr90] the router determines the shortest path to a sender or receiver. Due to use of flooding to keep the track of all senders and receivers of multicast traffic in MOSPF, this approach is not scalable and can have big routing overheads in a large and dynamic Ad Hoc network. The unsuitability of the conventional multicasting schemes, led to the researchers in the Ad Hoc community to develop new schemes.

### 3.2.2 Multicasting in Ad Hoc Networks

Multicasting schemes developed for Ad hoc networks have to cope with unreliable links, dynamic topology and ever-changing membership. This makes the task of route maintenance difficult. In this section multicasting schemes for Ad Hoc networks are presented along with their suitability for use in an emergency scenario. This suitability is examined by focusing the discussion on the scalability, mobility and reliable message delivery features of the schemes. The discussion starts with the multicast schemes classified (in Figure 21) depending on how the multicast group structure is maintained. The simplest of the multicasting techniques use the *flooding-based* approach [hot+99, kv00, kv98, mgl04, ni97, pkd02, wc02]. In this approach the participating entities use the broadcast mechanism to send data to its neighbours, who in turn pass the data on until the data reaches the intended recipient(s). In contrast, the organised *tree-based* approach [ltm99, oks99, rp99, ssb99, wt99] ensures that the participating entities are structured in the form of a tree which leads to minimal data traffic in the Ad Hoc network as the tree structure ensures only one path between two participating entities. This *tree-based* approach can be subdivided into two categories depending on whether the source or the receiver constructs the tree. The *mesh-based* approach [cgz98, lk00, lsg00, mf99] uses the fact that there can be redundant routes between any two participating entities of an Ad Hoc network. The redundant routes can be used to ensure a robust multicast delivery message mechanism despite dynamic topology and membership. Other classifications can be done on the basis that if the multicast algorithm uses *location* information, or if the nodes use caches to improve the *reliability* of multicast message delivery. Most of the classes of the multicasting algorithms (in Figure 21) can be further sub-categorized on the method of route maintenance. In multicast schemes the routes can be maintained proactively or reactively. The *proactive* approach relies on soft state maintenance by periodic update of routes and membership even if no multicast messages are being sent. In contrast, the *reactive* approach relies on a hard state maintenance. The routing and membership information is sought only when some member has some multicast content to send.

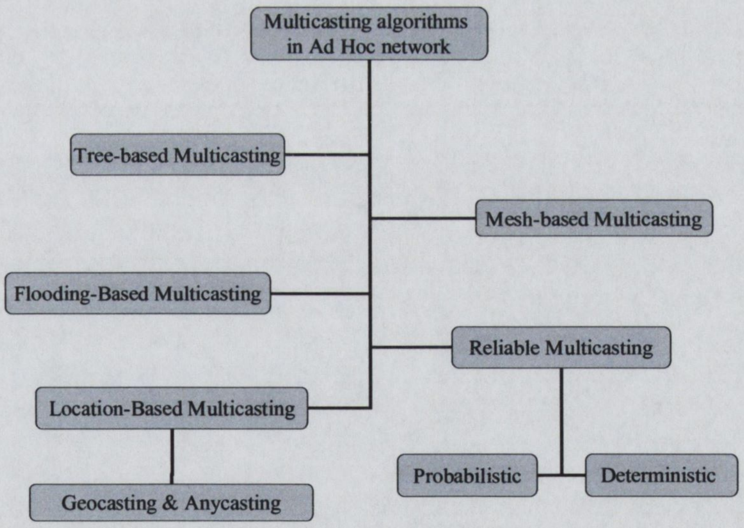


**Figure 21 : Classification of the Multicasting Algorithms in Ad Hoc networks**

### *3.2.2.1 Tree-based Multicasting Algorithms*

This section presents and compares some of the tree structure based multicasting schemes designed for Ad Hoc networks. The first scheme discussed in this section is the

Ad hoc Multicast Routing (AMRoute) [ltm99]. This is a proactive multicasting algorithm for Ad Hoc networks similar to the *center-based approach* (i.e. the CBT and PIM-SM schemes) of the conventional wired networks. The AMRoute builds a tree structure among the nodes of the Ad Hoc network interested in multicasting. Moreover, only the member nodes of the multicasting group keep the associated state information scheme while other non-members nodes in the Ad Hoc network are spared this unnecessary task. The multicast tree structure is used for group message propagation. However, this multicast group tree structure is independent of the network topology and connectivity in the tree structure is maintained by the underlying unicast routing algorithm. The tree structure copes with partitioning and re-merging of the network by using a *core resolution protocol* which runs periodically to reconstruct or maintain the tree structure. A node wanting to join the multicast tree uses limited Time-To-Live (TTL) broadcast flooding to find the existing group members. The AMRoute algorithm is suited for stable Ad Hoc networks due to its reliance on a tree structure for group message propagation. However, mobility can cause loop formation in the routing tables which may result in congestion and collisions due to frequent re-transmission of the same multicast message. A similar scheme that uses the concept of tree group structure along with the robust mesh based group message propagation is the Multicast Core-Extraction Distributed Ad Hoc routing (MCEDAR) [ssb99]. This makes the MCEDAR algorithm more tolerant to link breakages compared to the AMRoute scheme.

A novel *tree-based* multicasting algorithm is the proactive Ad hoc Multicast Routing utilizing Increasing id-numberS (AMRIS) protocol [wt99]. In this scheme each member of the multicast group has a dynamically assigned id-number. The tree is formed with the root member broadcasting its id-number. New members joining (directly with the root member) set their id-number to be greater than the root's id-number. This process of the member joining a pre-existing member and choosing an id-number greater that the pre-existing members id-number is repeated till a tree is formed. Thus the tree is formed with the property that the children member will have an id-number greater than its parent member. This implies that the root member has the smallest id-number. The tree with each ascending id-number is used for group message propagation. In this scheme each member periodically updates its neighbour status information to maintain the group tree structure. If due to mobility or a member leaving unexpectedly the multicast tree is affected. In such cases it is the responsibility of the members with the greater id-numbers (i.e. children member) to reconstruct the tree. The reconstruction involves limited broadcasting (using a TTL mechanism) to look for suitable member (i.e. having a smaller id-number) of the multicast group to rejoin. In contrast to the earlier AMRoute approach, in AMRIS the non-members can be forced to join the multicast group. Thus a more optimal tree is used by AMRIS for multicast message propagation. The AMRIS algorithm works well in stable Ad Hoc networks. However, the high mobility of members in AMRIS may cause multicast message loss due to network congestion.

The Bandwidth Efficient Multicast Protocol [oks99] using the tree structure in conjunction with an reactive algorithm to reduce the control message overheads. A member wanting to send the multicast traffic computes an optimal route to a forwarding member. These forwarding members have the responsibility of forwarding the multicast traffic to the neighbouring members. The number of forwarding nodes is kept to a minimum by this algorithm to minimize the forwarding of the multicast traffic, thus achieving greater multicast efficiency. A similar approach of discovering the routes reactively is used by the Multicast operation of the Ad hoc On-demand Distance Vector (MAODV) protocol [rp99]. However compared to the Bandwidth Efficient Multicast Protocol the MAODV has less efficiency in multicast packet delivery. The tree-based multicasting Ad Hoc algorithms are suitable for scenarios in which the participating nodes have low mobility. For the high mobility scenarios the mesh-based algorithms are more suitable since they use the redundant links to maintain connectivity among the multicast group members.

#### *3.2.2.2 Mesh-based Multicasting Algorithms*

The first *mesh-based* multicast scheme discussed in this section is the On Demand Multicast Routing Protocol (ODMRP) [lsg00]. This protocol uses a reactive multicasting algorithm based on flooding. At the start of the protocol interchange the *senders* of the multicast traffic look for the *receivers* by broadcasting out a Join-Query packet. The intermediate entities forward this Join-Query packet until it reaches the intended *receivers*. These intermediate nodes keep a track of the Join-Request packets to eliminate loops due to duplicate packets. On receipt of an appropriate Join-Request the *receivers* send back a Join-Reply packet back to the *senders*. This request/reply mechanism makes the intermediate nodes realise that they are functioning in a forwarding role for the multicast messages. These intermediate nodes are collectively known as the forwarding group (FG) and are responsible for propagation of the multicast group messages. The *senders* are also responsible for periodic refreshing of routes by sending out the Join-Query messages. The main benefit of the ODMRP protocol is that a node can leave and join with no control overheads. This protocol also has the ability to also act as an unicast routing algorithm. However, since the forwarding of multicast traffic is done using limited flooding it increases the control overheads and therefore reduces the scalability of the protocol. A similar protocol suffering from the same drawbacks as the ODMRP scheme is the proactive Forwarded Group Multicast Protocol (FGMP) [cgz98]. It differs from the ODMRP in the way the mesh structure is constructed. Instead of the sender initiated construction of the FG in ODMRP, both the senders and receivers can construct the FGMP mesh structure.

The Core Assisted Mesh Protocol (CAMP) [mf99] is another proactive mesh-based multicasting algorithm. This protocol borrows heavily from the CBT and uses the "Core" members to limit the control traffic needed when a receiver member joins/rejoins the group. The protocol classifies the participating entities into three types: simplex, duplex and non-members. The simplex members are used to create one way connections between the sender-only members and the rest of the multicast mesh. This implies that no multicast group traffic is sent from the multicast mesh to the sender-only entities by the simplex entities. In contrast the duplex members are full members (i.e. "core" members) of the multicast mesh responsible for receiving and forwarding of the multicast traffic. The non-members are the entities in the network that are not a part of the multicast delivery mesh. High mobility of nodes in the CAMP can result in big control overheads, since each member maintains the membership information using a periodic heartbeat broadcasting mechanism. The CAMP algorithm another drawback is its reliance on an underlying unicast routing algorithm to function.

The proactive Neighbour Supporting ad hoc Multicasting routing Protocol (NSMP) [lk00] is a mesh-based algorithm which improves the multicasting efficiency by localizing the multicast group maintenance process. This algorithm assumes that most of the link breakages are localized and repairing them frequently can result in efficient multicast message propagation. Therefore the broadcasting based *local route discovery* is used frequently for short distances. The less used periodic *flooding route discovery* is used to discover the long-distance routes. The localized link breakage assumption fails if the entire multicast group is highly mobile with a dynamic topology. This kind of frequent topological change can result in high control traffic overhead in the NSMP protocol.

### *3.2.2.3 Flooding and Location based Multicasting*

In a *flooding-based* multicasting approach, messages are delivered network-wide using broadcasting. A participating node rebroadcasts a multicast message it received to all its neighbours and only discards a message if it has seen it before. This method of multicasting is best suited for Ad Hoc networks with high mobility, as it ensures a robust message delivery mechanism. However, every participating node has to keep a history of messages received. This may require large storage if the number of participating nodes is large. Moreover, the flooding of the multicast messages may cause big overheads due to collision, retransmission and redundant retransmission. The various *flooding-based* multicasting techniques were classified by Mohapatra et al [mgl04] into four distinct categories. The first sub-category uses a simple flooding based approach [hot+99] leading to high reliability of multicast message delivery in a mobile Ad Hoc network. In the next sub-category, the flooded multicast message is re-broadcasted by an entity based on a preset probability [pkd02]. The third sub-category uses the knowledge of the neighbour's [wc02] (one/two hop nodes depending on the scheme) state to decide if the multicast message is to be re-broadcasted. The last sub-category (i.e. *location-based* multicasting) consists of multicast schemes for a large group spread over a big area. These schemes use the knowledge of area information (usually using GPS devices) to decide on the multicast message re-broadcast [kv98]. The next section will briefly examine the suitability of the two *location-based* multicasting techniques of Geocasting and Anycasting for deployment in a large Ad Hoc group.

#### 3.2.2.3.1 Geocasting and Anycasting

*Geocasting* is a multicasting routing technique (Figure 22) first proposed for the conventional Internet by Navas et al [ni97]. This algorithm ensured that a multicast message is delivered to all participating entities in a geographical region. Adaptations for Ad Hoc networks of the *geocasting* method was first proposed in the Location-Based Multicast (LBM) [kv98] scheme. The LBM scheme has two mechanisms to deliver the messages to all nodes in a specified *geocast region* [ni97]. In the first mechanism the flooding of the multicast messages is limited to a *forwarding* region until it reaches the *geocast region*. The second mechanism uses the *central point* of the *geocast region*. An intermediate node only forwards a multicast message, if the distance computed by the node is less than that computed by the other intermediate node requesting the forward of the message. In contrast to the LBM flooding approach the GeoTORA proposed by Ko et al [kv00] uses a route driven approach for geocasting. The geocasting schemes are of use only in large Ad Hoc networks with all the participating nodes having expensive GPS or similar devices to derive location information.

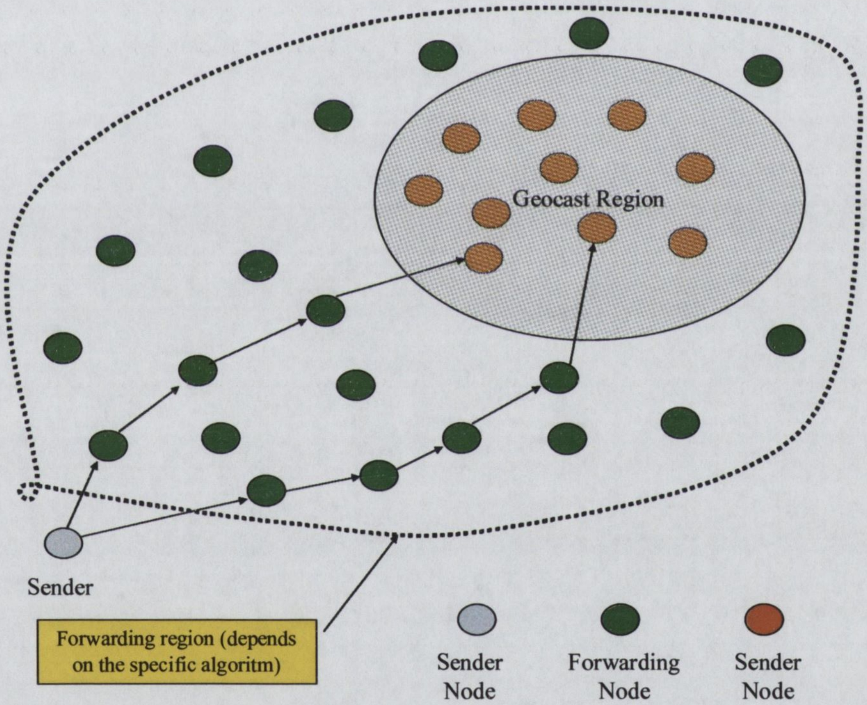


**Figure 22 : Geocasting**

The *Anycasting* routing, specified in the IPv6 specification [ipv6] relies on several servers supporting a common service having a common *anycast* address. A participating node communicates with the nearest server having the *anycast* address to join the service. This kind of distributed access service is useful in Ad Hoc networks deployed in disaster or battlefield scenarios. However a reliable backbone for content synchronization between the servers is difficult to achieve in an Ad Hoc network.

### *3.2.2.4 Reliable Multicasting Protocols*

In some secure group collaborations reliable delivery of the group messages may be required. This section examines the Ad Hoc multicasting protocols that provide a reliability guarantee beyond best effort. These protocols can be classified into two distinct types: deterministic and probabilistic [ve03]. The former provide a definite delivery guarantee while latter try to guarantee delivery of multicast message with a minimum probability.

#### 3.2.2.4.1 Deterministic Protocols

The first deterministic protocol designed for the medium mobility Ad Hoc networks was the Reliable Broadcast (RB) [pr97]. This protocol assumes that there is some underlying clustering algorithm and the participating entities in the network are divided into clusters with each of the clusters having a cluster head. These interconnected cluster heads are responsible for the multicast message propagation. A node wanting to send a multicast message sends it to its cluster head which then relays the message to other connected cluster heads. This forwarding process is repeated until the multicast message reaches all the cluster heads. The reliability is ensured by all cluster heads recursively sending back acknowledgements to the original cluster head for receiving the multicast message. If the group is large or highly mobile the receipt of the acknowledgements may be delayed. In such cases of high mobility the RB algorithm can switch to flooding of the acknowledgements back to the original cluster head. This increases the complexity of the algorithm to O ($n^2$) from O (n) [ve03] where "n" is the number of participating entities in the network. Instead of using cluster heads, the Adaptive Reliable Broadcast (ARB) [gs99] uses a tree structure to reliably propagate the group messages. A participating node sends a multicast message up the tree until it reaches all the other member nodes. The core member (root of the tree) receives acknowledgement of the message delivery from all participating nodes. To improve the efficiency of the protocol, acknowledgements can be aggregated by a node and sent piggyback with subsequent multicast group messages. In case, the tree is fragmented, due to mobility, participating nodes, in different fragments, send the multicast messages by broadcasting them to the *forwarding region*. Nodes in the *forwarding region* then relay the multicast messages to the appropriate fragments. This process is called "gluing together" of the multicast tree. However, mobility causes the ARB protocol to suffer the same performance degradation as in the RB protocol. The performance degradation is manifested as congestion and frequent retransmissions making the ARB and RB unsuitable for large and dynamic Ad Hoc network.

The two deterministic protocols designed for Ad Hoc networks assume that the multicast receivers are known to the sender beforehand are the Reliable Multicast Algorithm (RMA) Protocol [gsp+02] and Reliable Adaptive Lightweight Multicast (RALM) [tol+02] algorithm protocol. The main difference is that in RALM a single receiver collects all the acknowledgements for the delivery of a multicast message and then forwards them to the sender while in the RMA all the acknowledgements are received by the sender itself. Moreover the RALM uses a TCP-like window congestion control mechanism. The RMA and RALM protocols suffer from the same problems of performance degradation as in the RB and ARB protocols. Thus, the RMA and RALM are unsuitable for large and dynamic Ad Hoc groups.

#### 3.2.2.4.2 Probabilistic Protocols

The probabilistic protocols are designed for large multicast groups in Ad hoc networks and guarantee the delivery of a multicast message with a certain probability. The first protocol considered is the Anonymous Gossip (AG) [crb01] which is designed to increase the reliability of any reactive multicasting protocol. This protocol works in a two stage process. The first stage is when the multicast messages are sent to the group unreliably using the underlying reactive multicasting protocol. In the second stage two participating nodes in the AG periodically query each other and then recover any lost multicast messages. This process requires the nodes in the AG to maintain a message cache. The reliability of the message delivery increases considerably with this periodic recovery of messages. The same concept is used by the Route Driven Gossip (RDG) [leh03] scheme which unlike the Anonymous Gossip (AG) scheme does not require an underlying multicasting algorithm. Instead the periodic querying of the messages caches is used by participating nodes to propagate the multicast messages. Thus the periodic queries form the basis of the multicasting in the AG scheme.

Reliable deterministic multicasting algorithms are good for small static groups. This lack of scalability is due to the fact that reliability is ensured by acknowledgements of delivery which might clog up the network if the group is large. This lack of scalability was addressed by the probabilistic multicasting schemes. The reliable probabilistic approach requires storage of the multicast messages. This might result in huge storage requirements on individual entities if the size of the multicast group is large. Therefore the schemes mentioned in this section ensuring reliable delivery of multicast messages in an Ad Hoc network are not scalable.

#### *3.2.2.5 Conclusion of Group Communications*

Individual multicasting algorithms for Ad Hoc networks were developed for specific scenarios as one solution is not suitable for all scenarios. This section examined the multicasting schemes and found that for a large and dynamic Ad Hoc network deployed in an emergency scenario, the *mesh-based* approach is best suited. The *mesh-based* approach is a compromise between the *flooding-based* and *tree-based* approaches. The *mesh-based* approach is best suited to maintain connectivity among the multicast group members despite topological and membership changes. Even though the simple *flooding-based* approach has the lowest control overheads, connectivity among members of the multicasting group can decrease due to congestion caused by high volume traffic in a large and mobile Ad Hoc network. The hierarchical *tree-based* approach requires continuous tree maintenance is a difficult task if the Ad Hoc multicasting group which is large and mobile. The suitability of the mesh-based algorithms was demonstrated in the comparative simulation of Ad hoc wireless multicasting protocols carried out on GloMoSim [glomosim] simulator by Lee et al [lsh+00]. This section also examined the multicasting algorithms in Ad Hoc networks which provided a reliable multicast message delivery and found them unsuitable for a large and dynamic multicast group.

# 4 One-to-One Trust Negotiation

This chapter presents a one-to-one trust negotiation scheme suitable for an Ad Hoc network deployed in an emergency scenario. The trust negotiation process involves progressive exchange of certificates (with custom embedded attributes) governed by the local policies of the two participating entities. A policy translates the attributes from the received certificates into access control permissions on the service providing node. The exchange of the certificate is done using a new protocol which is protected against attackers by a novel two-tier key formation. This two-tier key formation ensures the confidentiality and integrity of all communications in the Ad Hoc network and is independent of the routing algorithm used. The chapter ends with a description of a prototype of the scheme running on real commodity handheld wireless devices. The detailed discussion of the one-to-one trust negotiation scheme starts with the two-tier key formation framework.

## *4.1 Two-Tier Key Formation*

The confidentiality and integrity of the trust negotiation process is ensured by key formation between the two participating entities. This key formation is independent of the routing algorithm. Thus the key formation is portable as no one routing algorithm operates efficiently in all the mobility and membership scenarios [hbt+03, hbt+04]. The independence from the routing algorithm is achieved by splitting the key formation framework into two tiers (Figure 23). The first tier involves a peer-to-peer key formation between neighbours (i.e. nodes within transmitting and receiving range). This peer-to-peer key is used to encrypt the communication between two neighbours. Since this key tier is below the routing level, it uses the Media Access Control (MAC) address to identify and authenticate the remote node. However, a Media Access Control (MAC) address is easy to impersonate by attackers. Therefore, authentication of the MAC address in the peer-to-peer key formation process is provided by an *address* certificate (signed by a CA). This certificate has a custom attribute containing the MAC address which the owner node is authorized to use. Another benefit of the peer-to-peer key tier is that it thwarts eavesdropping by external passive attackers. These attackers do not process an *address* certificate trusted by the existing nodes in the network. Consequently such attackers are excluded from routing in the Ad Hoc network. The second tier of the key formation is above the routing level. This end-to-end key provides confidentiality and integrity to the main trust negotiation and subsequent communications between the two nodes. These keys prevent *internal rogue* attackers from eavesdropping on the data traffic between two nodes. The *internal rouge* nodes are authenticated (at the MAC level) to join the network and take part in routing of packets. The authentication for an end-to-end key is provided by *identity* certificates (certified by a CA) containing embedded identification information. This two-tier key formation framework has two assumptions. Firstly, each participating node has an *address* and an *identity* certificate. Secondly, the network links are bi-directional to allow peer-to-peer key negotiation.

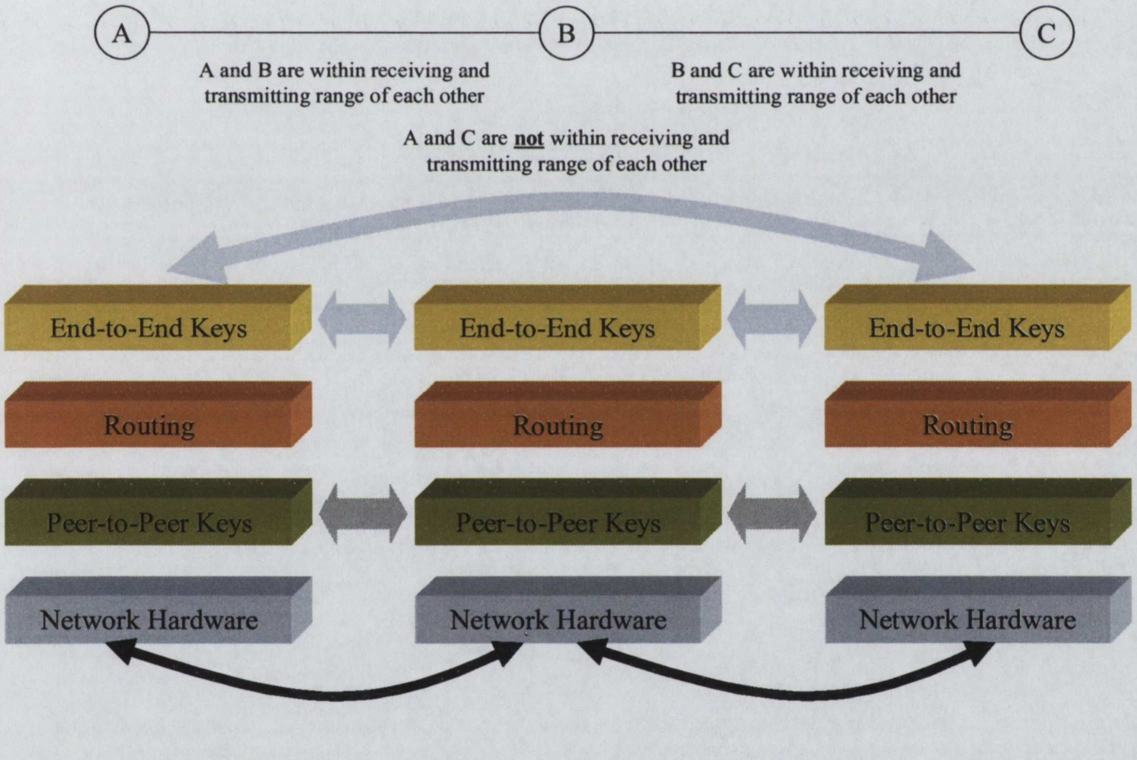

A
B
C
A and B are within receiving and transmitting range of each other
B and C are within receiving and transmitting range of each other
A and C are not within receiving and transmitting range of each other
End-to-End Keys
End-to-End Keys
End-to-End Keys
Routing
Routing
Routing
Peer-to-Peer Keys
Peer-to-Peer Keys
Peer-to-Peer Keys
Network Hardware
Network Hardware
Network Hardware

**Figure 23 : Two Tier-Key Formation**

An example of the two-tier key formation is presented in Figure 24. There are five wireless nodes (i.e. A, B, R, C and D) in the wireless Ad Hoc network along with a node $X$ which is not authenticated at the peer-to-peer level to join the network. The authenticated nodes form the peer-to-peer keys $K_1$ to $K_4$ and an end-to-end key $K$ between nodes $A$ and $D$. Thus an *external passive* attacker such as $X$ eavesdropping on the traffic in the network between nodes *A* and *D* has to compromise the key $K_1$ or $K_2$ followed by *K* to understand the eavesdropped traffic. Similarly an internal *rogue* user on node *R* has to compromise the key *K* to understand the traffic between nodes *A* and *D*. This two-tier key formation particularly makes it difficult for an *external passive* attacker to understand end-to-end communications as it has to compromise at least two independent keys.

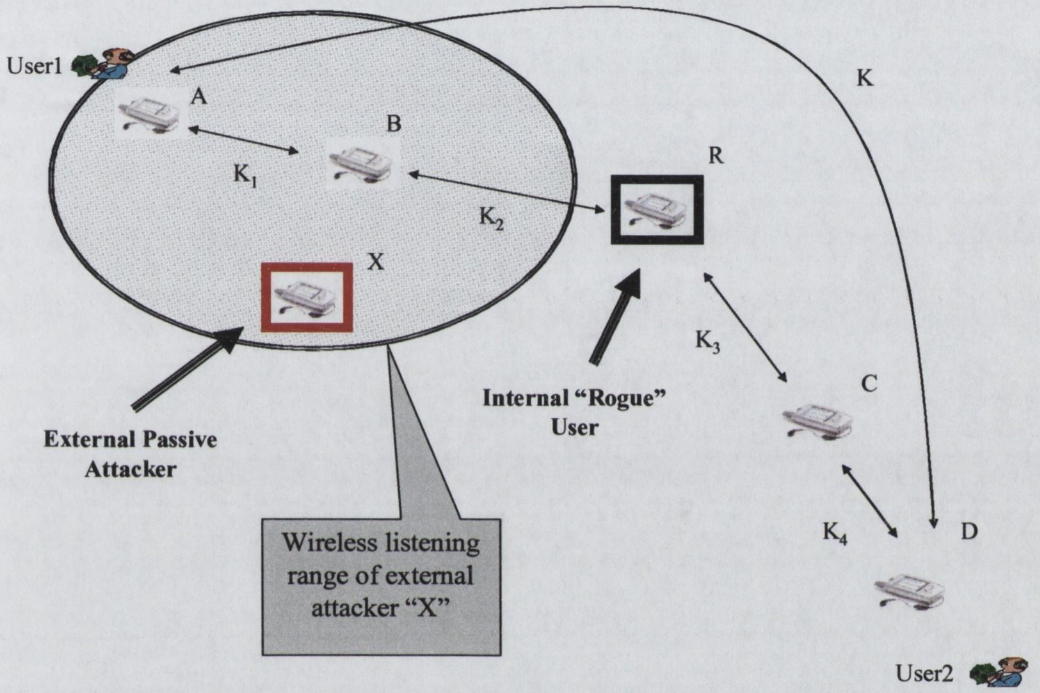

User1
A
B
K1
R
K
K2
X
K3
C
Internal "Rogue" User
External Passive Attacker
Wireless listening range of external attacker "X"
K4
D
User2

**Figure 24 : Example of the Two-Tier Key Formation**

The peer-to-peer key formation is formalized into the Neighbour Trust Model (NTM) layer situated below the routing level. This layer has the dual function of detecting neighbours and then forming keys with them. The other layer in the one-to-one trust negotiation scheme is the Remote Trust Model (RTM) layer situated above the routing layer. This layer is responsible for the end-to-end key formation along with the main trust negotiation process. The RTM layer also provides access control to the services provided by the node. These access control decisions depend on the outcome of the trust negotiation process. The first layer discussed in this chapter is the NTM layer.

## *4.2 Neighbour Trust Model (NTM) Layer*

This layer is responsible for negotiating peer-to-peer keys used to encrypt the communications between the two neighbouring nodes. To do this a node must first discover its neighbours. This can be done either in a proactive or a reactive fashion. In the former a node actively looks for its neighbours by periodically broadcasting a *Hello* message. Interested neighbours then reply to this message to register themselves with the original node. The use of the proactive approach can result in network congestion if the periodicity of the *Hello* message broadcast is small in a high node density wireless Ad Hoc network. In contrast, a reactive approach toward neighbour detection implies that a node only looks for neighbours when there is some data to be sent. Thus the data transmission waits till the neighbours are discovered. This latency can cause dropped packets if the neighbours are not aware of each other and consequently the node is unable to send high priority traffic [ipv6nd]. Therefore in the NTM protocol use of the proactive periodic broadcasting method is preferred.

Nodes employing the NTM scheme can increase their interval for broadcasts of *Hello* message if network congestion is detected. However, in an Ad Hoc network it is difficult to distinguish between an intentional attack and unintentional network congestion. Consequently, instead of employing an automatic algorithm to increase the periodicity of the broadcast, the user is asked to make this decision in the NTM layer. To aid such manual decisions, the user is provided with neighbour node density and the average packet transmission rates of the neighbours. The issue of automatic determination of the broadcast periodicity is a significant area of research in its own right and is beyond the scope of this thesis.

The NTM layer uses the first message of the modified STS protocol discussed in Chapter 2 as the "Hello" message to detect the neighbours. This message serves two purposes: neighbour detection and beginning of the peer-to-peer key agreement process. The authentication of the MAC level network address of the nodes is done using an *address* certificate signed by a CA and contains a custom attribute *mac_id*. Thus a node cannot participate in an Ad Hoc network, if the node's neighbours do not trust the CA issuing the node's *address certificate*. These *address* certificates are used in the NTM protocol interchange presented in the next section.

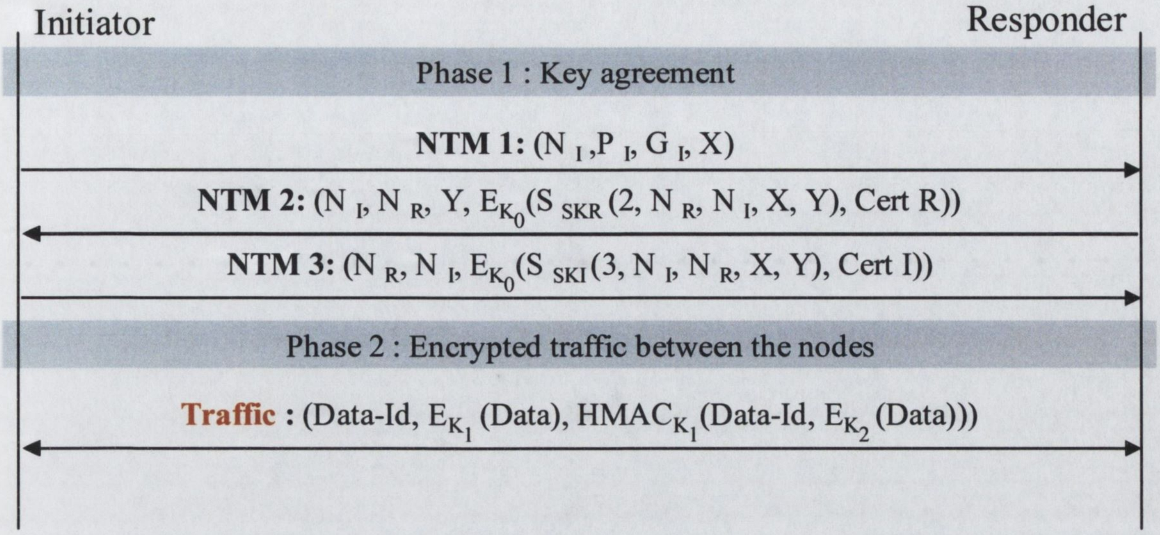
### 4.2.1 NTM Protocol

The protocol used by the NTM for key agreement presented in Figure 25 is based on the Modified STS protocol discussed in the Chapter 2. However, this protocol has two additions over the original protocol without affecting the cryptographic properties. Firstly, the *address* certificates used for authentication in the protocol interchange are encrypted. The encryption of the *address* certificate prevents the disclosure of the certificate and associated information contained in it to passive *external* attackers (see Figure 24) eavesdropping on the network traffic. This encryption of certificates partially mitigates the identity disclosure branch of the threat tree for the Modified STS protocol (see Chapter 2).

However, to prevent the known plaintext attack on the shared secret it is not directly used to encrypt the certificates used in the NTM key agreement interchange. Instead a key $K_0$ derived from the negotiated shared secret $K$ is used in the NTM protocol's phase 1. Similarly for use in phase 2 of the NTM protocol, the encryption key $K_1$ and the authentication key $K_2$ are derived from the negotiated shared secret $K$. These keys $K_1$ and $K_2$ are required as the phase 2 of the NTM protocol uses the encrypt-then-authenticate method [k01]. The use of different keys is necessary as a key should be used only for one purpose in a cryptographic protocol. This ensures that additional data is not available to an attacker analysing the traffic to compromise the mutual shared key. The three keys used in the NTM protocol are derived from the shared secret $K$ using the Keyed-Hashing for Message Authentication (HMAC) [kbc97] algorithm. In brief the derived keys used in the NTM protocol are,

- $K_0$ - Used in the key agreement phase of the NTM protocol
- $K_1$ - Encryption key for the phase 2 of the NTM protocol
- $K_2$ - Authentication key for phase 2 of the NTM protocol

The second modification in the NTM protocol over the Modified STS protocol discussed in Chapter 2 is the use of timers to partially mitigate the DoS attack. These timers are discussed after the presentation of the NTM protocol in Figure 25.



Initiator
Responder
Phase 1 : Key agreement
**NTM 1:** $(N_I, P_I, G_I, X)$
**NTM 2:** $(N_I, N_R, Y, E_{K_0}(S_{SKR}(2, N_R, N_I, X, Y), \text{Cert R}))$
**NTM 3:** $(N_R, N_I, E_{K_0}(S_{SKI}(3, N_I, N_R, X, Y), \text{Cert I}))$
Phase 2 : Encrypted traffic between the nodes
**Traffic** : $(\text{Data-Id}, E_{K_1}(\text{Data}), HMAC_{K_1}(\text{Data-Id}, E_{K_2}(\text{Data})))$

Notation Used:

$X = (G_I)^x \bmod P_I$ where small "x" is a random number generated by the initiator.

$Y = (G_I)^y \bmod P_I$ where small "y" is a random numbers generated by the responder. Generated fresh for each key agreement by the responder.

$P_I$, $G_I$ – Diffie –Hellman parameters used by the initiator and embedded in the certificate Cert I.

K – Shared Secret = $(G_I)^{xy} \bmod P_I$.

$N_I$ – The network address used by the initiator and embedded in the certificate Cert I.

$N_R$ – The network address used by the responder and embedded in the certificate Cert R.

SKI – Secret key of the certificate Cert I.

SKR– Secret key of the certificate Cert R.

$E_{Key}(M)$ – Encryption of M using the symmetric key "Key".

$S_{Key}(M)$ – Signing of M using the asymmetric secret key "Key".

$HMAC_{Key}(M)$ – Keyed-Hashing for Message Authentication applied on the plaintext M using the key "Key".

$K_0 = HMAC_K(X,Y,1)$ i.e. Encryption key used in the Phase 1

$K_1 = HMAC_K(X,Y,2)$ i.e. Encryption key used in the Phase 2

$K_2 = HMAC_K(X,Y,3)$ i.e. Authentication key used in Phase 2

Data-Id – Unique sequence number for each "Data" transmission

Data – Data transmission between the initiator and responder.

Shared Information between the Initiator and the Responder:

Encryption / Signature / Certificate / Hashing algorithms and Key Sizes

**Figure 25 : NTM Protocol**

### 4.2.2 Timers in the NTM Layer

The three types of timers used in the NTM layer to partially mitigate the DoS attacks are,

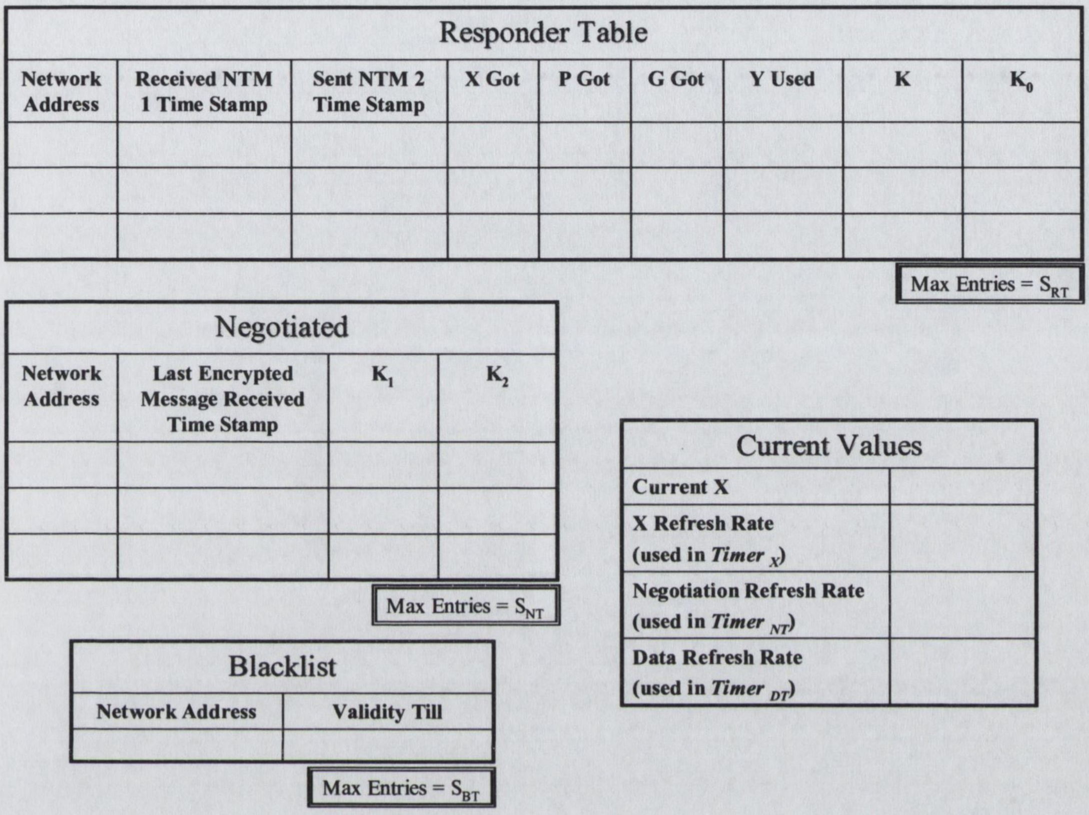
- *Timer* $_X$ - The expiry of this timer makes the node broadcast the NTM 1 message with a fresh Diffie-Hellman parameter *X* (after a fresh random number generation). This periodic refreshing deters launching of the small sub-group attacks [z00] against the initiator. Once a new *X* is computed, any reply to the NTM 1 message using a NTM 2 message containing the old *X* is discarded. The *X* in the subscript uniquely identifies this timer.
- *Timer* $_{NT}$ (*NT* stands for Negotiation Timer) - This timer is attached to each unfinished key negotiation by the responder. If an appropriate valid NTM 3 message is not received before the expiry of this timer the unfinished key negotiation is discarded. The *NT* in the subscript stands for Negotiation Timer
- *Timer* $_{DT}$ (*DT* stands for Data Timer) - This timer is attached by the initiator and responder to each key agreement which has finished successfully. If no encrypted data traffic is received from the appropriate remote node, the use of the negotiated shared secret (and the associated keys) is discarded.

This periodic clearing of the state information for a malicious key agreement by using the timers *Timer* $_{NT}$ and *Timer* $_{DT}$ helps in partial DoS mitigation. On each node, the NTM layer has one global timer *Timer* $_X$, but many instances of the timers *Timer* $_{NT}$ and *Timer* $_{DT}$ as they are attached to the individual key negotiations. These timers can be changed dynamically by the user if network congestion is detected. The timers act on the data structures storing the state information of the NTM protocol interchange.

### 4.2.3 NTM Organization

The NTM protocol interchange depends on the data structures presented in Figure 26. These data structures are used to store the timer values (i.e. *Current Values*), the state of the individual protocol interchanges (i.e. *Responder Table*) and the negotiated peer-to-peer keys (i.e. *Negotiated*). There is also a data structure dedicated to storing the MAC level network addresses of the nodes suspected to have malicious behaviour (i.e.

*Blacklist*). The NTM protocol does not negotiate a peer-to-peer key with the detected malevolent blacklisted nodes. However, the detection mechanism to identify the malicious users is outside the scope of this thesis.



Responder Table

| Network Address | Received NTM 1 Time Stamp | Sent NTM 2 Time Stamp | X Got | P Got | G Got | Y Used | K | $K_0$ |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |

Max Entries = $S_{RT}$

Negotiated

| Network Address | Last Encrypted Message Received Time Stamp | $K_1$ | $K_2$ |
|---|---|---|---|
| | | | |
| | | | |
| | | | |

Max Entries = $S_{NT}$

Current Values

| | |
|---|---|
| **Current X** | |
| **X Refresh Rate (used in *Timer* $_X$)** | |
| **Negotiation Refresh Rate (used in *Timer* $_{NT}$)** | |
| **Data Refresh Rate (used in *Timer* $_{DT}$)** | |

Blacklist

| Network Address | Validity Till |
|---|---|
| | |

Max Entries = $S_{BT}$

**Figure 26 : NTM Organization**

The *Responder Table* is used to store the status of the key negotiation on the responder nodes. Once the common shared secret has been agreed successfully with the remote node, the corresponding entry is deleted from the *Responder Table* and moved to the *Negotiated* data structure. The timers of type *Timer* $_{NT}$ acts on the *Responder Table* to weed out the key negotiations which have not been completed in the prescribed time. Similarly the timer of type *Timer* $_{DT}$ acts on the *Negotiated* data structure to weed out the peer-to-peer keys of neighbour's no longer in range. In the *Current Values* data structure the present value of the *X* and intervals for the timers are stored. It is important to note that the data structures have a finite size to conserve memory on the mobile handheld devices. Thus periodic refreshing of the state information is necessary to ensure there is space in data structures for storing states of key negotiations with genuine remote nodes. Another important fact is that no data structures are needed to store the state of the protocol interchange if the node is an initiator node. This is due to the way the state machine governing the NTM protocol interchange is constructed.

### 4.2.4 State Machine of the NTM Protocol

The state machine of the NTM protocol is presented in Figure 27. This state machine is based on the state machine of the Modified STS protocol presented in Chapter 2. Some additional features are added to the state machine to make the NTM protocol resistant to DoS attacks and network congestion (additions are in red).

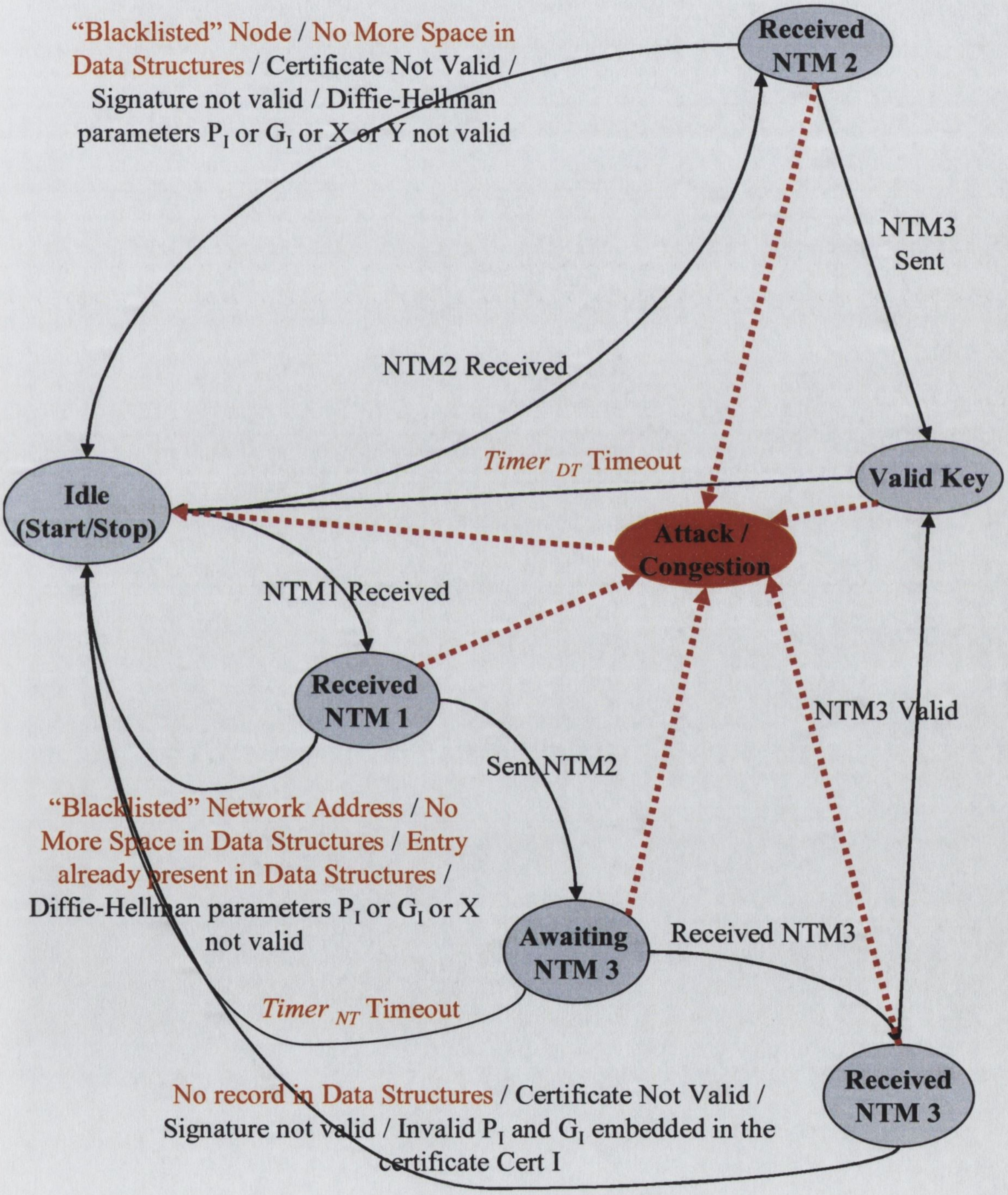
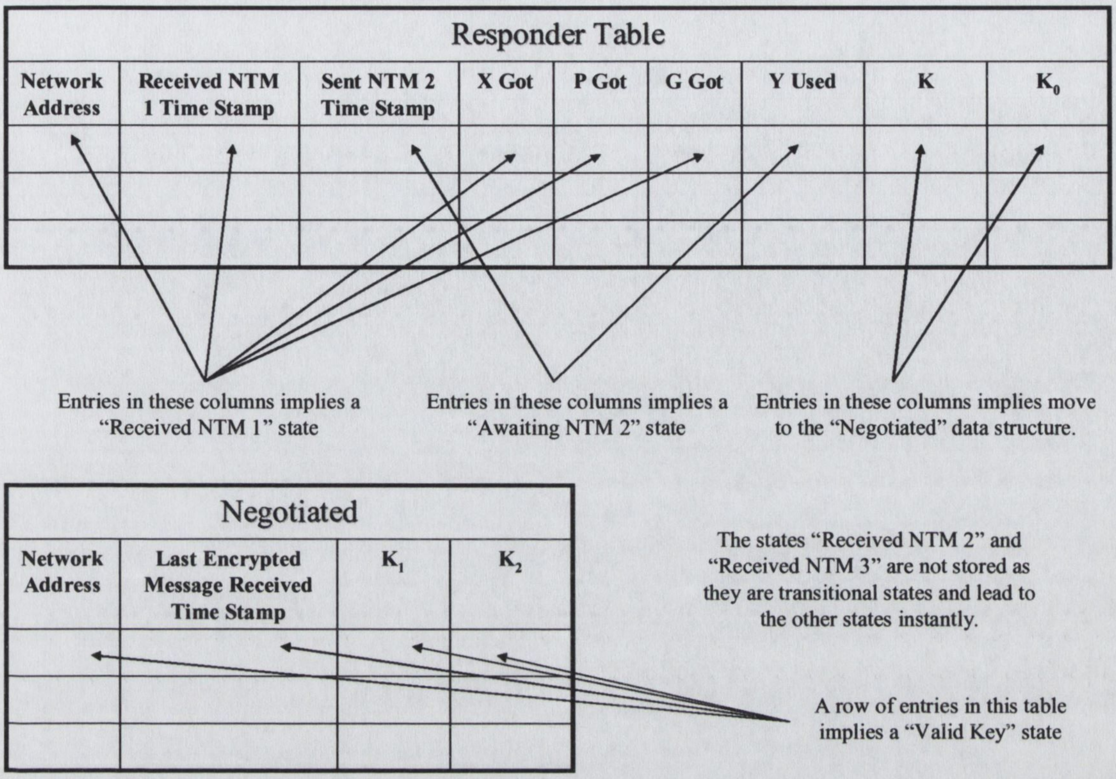

"Blacklisted" Node / No More Space in Data Structures / Certificate Not Valid / Signature not valid / Diffie-Hellman parameters $P_I$ or $G_I$ or X or Y not valid
Received NTM 2
NTM3 Sent
NTM2 Received
$Timer_{DT}$ Timeout
Valid Key
Idle (Start/Stop)
Attack / Congestion
NTM1 Received
Received NTM 1
NTM3 Valid
Sent NTM2
"Blacklisted" Network Address / No More Space in Data Structures / Entry already present in Data Structures / Diffie-Hellman parameters $P_I$ or $G_I$ or X not valid
Awaiting NTM 3
Received NTM3
$Timer_{NT}$ Timeout
No record in Data Structures / Certificate Not Valid / Signature not valid / Invalid $P_I$ and $G_I$ embedded in the certificate Cert I
Received NTM 3

**Figure 27 : State Diagram of the NTM Protocol**

The state machine for the NTM protocol has the start and stop states (i.e. *Idle State*) combined into one. The state machine can transit from the *Received NTM 1* and the *Received NTM 2* state to the *Idle* state, if the data structures available holding the states of a protocol interchange are full. These conditions can happen if the node density in the neighbourhood is high or there is a DoS attack launched against the node. The DoS attack involves frequent sending of malicious NTM 1 and NTM 2 messages to the nodes. The periodic cleanout of the un-progressed states *Awaiting NTM 3* and *Valid Key* by the timers of type $Timer_{NT}$ and $Timer_{DT}$, helps mitigates these DoS attacks.

In this state machine (compared to original in Chapter 2), a node ignores a new NTM 1 message from a remote node with which it has a valid negotiated shared secret or in process of negotiating one (i.e. entry in the *Responder Table* or *Negotiated* data structures). This happens frequently as the NTM 1 message is periodically broadcasted by a node to discover new neighbours. A state machine transition can also stop if the remote node uses an *address* which is blacklisted. Similarly, the *Received NTM 3* state can transit to the *Idle* state if an appropriate data structure entry in the *Responder Table* is missing. This may occur if the timer $Timer_{NT}$ expired for the protocol interchange after waiting for an appropriate NTM 3 network primitive.

The states *Received NTM 2* and *Received NTM 3* are transitional states leading to another state instantly after computations. Thus these states do not need to be stored on the respective nodes, decreasing the states against which a memory exhaustion type of DoS Attack can be mounted. An additional state incorporated into the state machine compared to the Modified STS protocol discussed in Chapter 2 is the *Attack/Congestion* state. This state is reached if the node detects excessive numbers of fake messages or frequent computations leading to unsuccessful key agreement. The user has to decide with the help of the data provided by the NTM layer if the *Attack/Congestion* state is caused by an intentional attack or use of inappropriate timer intervals leading to network congestion. Thus the user of the node can either adjust the timers or take further action against the remote node outside the context of this thesis. The selection of the timer time intervals is discussed in the implementation section at the end of this chapter. The mapping between the states of the state diagram and data structures of the NTM layer is given in Figure 28.



Responder Table

| Network Address | Received NTM 1 Time Stamp | Sent NTM 2 Time Stamp | X Got | P Got | G Got | Y Used | K | $K_0$ |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |

Entries in these columns implies a "Received NTM 1" state
Entries in these columns implies a "Awaiting NTM 2" state
Entries in these columns implies move to the "Negotiated" data structure.

Negotiated

| Network Address | Last Encrypted Message Received Time Stamp | $K_1$ | $K_2$ |
|---|---|---|---|
| | | | |
| | | | |
| | | | |

The states "Received NTM 2" and "Received NTM 3" are not stored as they are transitional states and lead to the other states instantly.
A row of entries in this table implies a "Valid Key" state

**Figure 28 : Relationship between the States and Data Structures in the NTM Layer**

The Modified STS protocol chosen in Chapter 2 which formed the basis for the NTM protocol sacrificed DoS protection to speed up the protocol interchange by reducing the number of messages exchanged. This problem is mitigated in the NTM protocol to an extent by use of timers which discards a state in protocol interchange if it is not completed in a reasonable time. Moreover, the incorporation of the *Attack/Congestion* state alerts the user if there is an intentional attack or some network congestion.

### 4.2.5 NTM Layer Summary

The NTM layer establishes peer-to-peer keys between neighbouring nodes to ensure the secrecy of the network communications against passive attackers who are not authenticated to join the Ad Hoc network. The authentication for the key formation uses an *address* certificate which entitles the owner node to use a Media Access Control address certified by a CA. Thus a node will only route packets for a neighbour, if it processes an *address* certificate trusted by the node. Moreover, the NTM layer is independent of the routing algorithm used by the node.

## *4.3 Remote Trust Model (RTM) Layer*

This layer is responsible for the end-to-end trust negotiation by progressive confidential exchange of certificates governed by the policies of the two participating nodes. The attributes embedded into the valid received certificates are used by a node providing service(s) to decide the access control permissions. The confidentiality and integrity of the trust negotiation and subsequent mutual data communications is protected by an end-to-end key negotiated by the two nodes. The RTM layer's operation requires no user intervention except when to resolve deadlock in the RTM protocol interchange. The RTM protocol is described in the next section.

### 4.3.1 RTM Protocol

The RTM protocol interchange has three phases (Figure 29). Phase 1, forms the shared secret based on the Modified STS protocol analysed in Chapter 2. This shared secret is used to ensure the confidentiality and integrity of the main trust negotiation in phase 2 and subsequent data communications in phase 3. However, phase 2 and 3 may not be required if the services sought on the remote node are not available. Thus, the key formation and service discovery are integrated into the phase 1 of the RTM protocol.

The RTM protocol interchange between two nodes can be triggered in two ways. The initiator node *A* can explicitly ask for trust negotiation (in phase 2) from a remote node by sending it a *Service_Request* message. This message also contains the service(s) the node initiating the process wants to access on the remote node *B*. The remote node *B* replies with a *Service_Reply* message containing the services that require negotiation along with those that are already unlocked. If the services(s) asked for are not available then it sends the *Service_Reply* message with a NULL service(s) offered field. To finish the RTM protocols phase 1 key formation, node *A* issues a *Service_Reply_Confirmation* message. This is depicted in phase 1 - option 1 of Figure 29. The node providing the service generates a random Unique Identifier (UID) to identify the protocol interchange. This is sent to the other node in a confidential manner by use of encryption in the *Service_Reply* message. An implicit RTM protocol interchange (phase 1 - option 2) can be initiated if a node *A* tries to access a locked service(s) on the remote node *B*. In this case the remote node *B* commences the RTM protocol by sending a *Negotiation_Required* message to the node *A* which requested the unlocked service(s). The response to this is a *Negotiation_Required_Reply* message from the node *A* requesting the service(s). This key negotiation phase ends with a *Negotiation_Required_Confirmation* message from the service provider *B* which also contains the encrypted UID.

Once phase 1 of the RTM protocol establishes a common shared secret between the two nodes, the main trust negotiation using the phase 2 starts using the *Certificate_Request* and the *Certificate_Reply* message pair. This trust negotiation is encrypted to prevent the disclosure of the certificates to eavesdropping nodes. Thus the confidentiality of the sensitive certificates owned by the two participating nodes is ensured. To match the *Certificate_Request* with the *Certificate_Reply* a randomly generated Request Identifier (RID) is used. The certificates are requested in the *Certificate_Request* message by the attribute name/value pair(s). It is not necessary for the reply to contain all the certificates requested. This ensures a give and take of certificates according to the certificate release policies. To prevent malicious node including certificates they don't own in the trust negotiation, the *Certificate_Reply* messages contains a proof of ownership of the certificate. Each node has to sign the RID with the private key of the certificate being sent so that the remote node can verify the ownership. Therefore it is imperative that a fresh RID is generated for each *Certificate_Request* message. An unlocked certificate sent to a remote node without any specific request using the *Certificate_Request* message uses the UID instead of the RID for the proof of ownership. A deadlock in trust negotiation process is detected when an identical *Certificate_Request* message is repeated. In the event of a deadlock in the certificate exchange, the node requesting the service can use the *Service_Policy_Request* message to ask for the service policy of the service providing node. If the disclosure policy of the service policy is satisfied then the node sends this using the *Service_Policy_Reply* message. The end of the negotiation is signalled using the *Negotiation_End* message that also contains the service(s) that are unlocked. The use of UID ensures that no fake *Negotiation_End* message is sent. If a negotiation is successful, subsequent data traffic is encrypted. The keys $K_0$, $K_1$ and $K_2$ in the RTM protocol are derived using the same method as in the NTM protocol. Moreover, the phase 3 of the RTM protocol is same as the NTM protocol's phase 2.

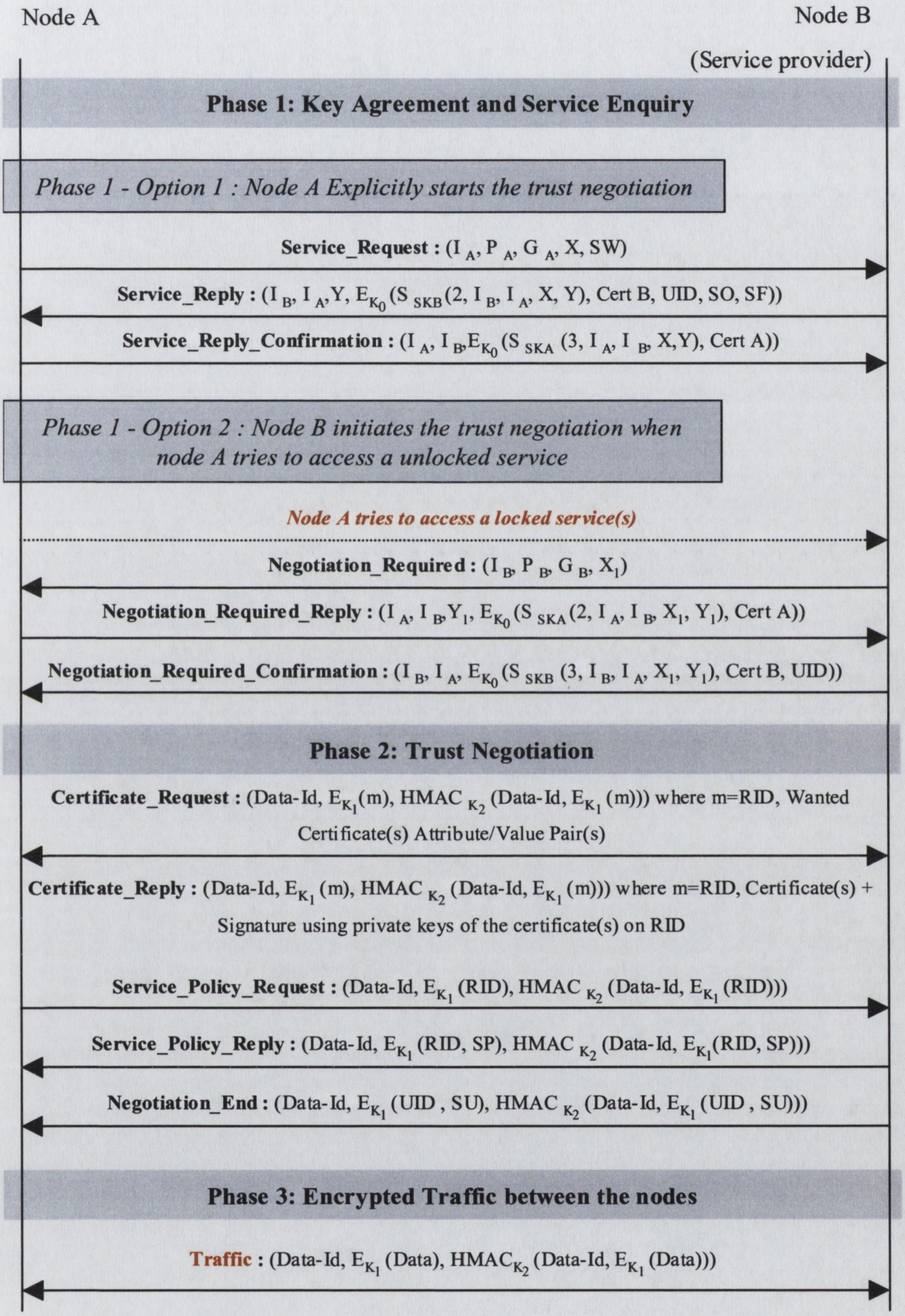

Node A
Node B
(Service provider)
Phase 1: Key Agreement and Service Enquiry
Phase 1 - Option 1 : Node A Explicitly starts the trust negotiation
Service_Request : $(I_A, P_A, G_A, X, SW)$
Service_Reply : $(I_B, I_A, Y, E_{K_0}(S_{SKB}(2, I_B, I_A, X, Y)$, Cert B, UID, SO, SF))
Service_Reply_Confirmation : $(I_A, I_B, E_{K_0}(S_{SKA}(3, I_A, I_B, X, Y)$, Cert A))
Phase 1 - Option 2 : Node B initiates the trust negotiation when node A tries to access a unlocked service
Node A tries to access a locked service(s)
Negotiation_Required : $(I_B, P_B, G_B, X_1)$
Negotiation_Required_Reply : $(I_A, I_B, Y_1, E_{K_0}(S_{SKA}(2, I_A, I_B, X_1, Y_1)$, Cert A))
Negotiation_Required_Confirmation : $(I_B, I_A, E_{K_0}(S_{SKB}(3, I_B, I_A, X_1, Y_1)$, Cert B, UID))
Phase 2: Trust Negotiation
Certificate_Request : (Data-Id, $E_{K_1}(m)$, HMAC $_{K_2}$ (Data-Id, $E_{K_1}(m)$)) where m=RID, Wanted Certificate(s) Attribute/Value Pair(s)
Certificate_Reply : (Data-Id, $E_{K_1}(m)$, HMAC $_{K_2}$ (Data-Id, $E_{K_1}(m)$)) where m=RID, Certificate(s) + Signature using private keys of the certificate(s) on RID
Service_Policy_Request : (Data-Id, $E_{K_1}$ (RID), HMAC $_{K_2}$ (Data-Id, $E_{K_1}$ (RID)))
Service_Policy_Reply : (Data-Id, $E_{K_1}$ (RID, SP), HMAC $_{K_2}$ (Data-Id, $E_{K_1}$(RID, SP)))
Negotiation_End : (Data-Id, $E_{K_1}$ (UID , SU), HMAC $_{K_2}$ (Data-Id, $E_{K_1}$ (UID , SU)))
Phase 3: Encrypted Traffic between the nodes
Traffic : (Data-Id, $E_{K_1}$ (Data), HMAC$_{K_2}$ (Data-Id, $E_{K_1}$ (Data)))

**Figure 29 : RTM Trust Negotiation Protocol**

Notation Used:

$P_A$, $G_A$ – Diffie –Hellman parameters used by Node A and embedded into the identity certificate Cert A used by node A.

$P_B$, $G_B$ – Diffie –Hellman parameters used by Node B and embedded into the identity certificate Cert B used by node B.

$X = (G_A)^x \bmod P_A$ where small "x" is a fresh random number generated by the Node A.

$Y = (G_A)^y \bmod P_A$ where small "y" is a fresh random number generated by the Node B.

$X_1 = (G_B)^{x_1} \bmod P_B$ where small "$x_1$" is a fresh random number generated by the Node B.

$Y_1 = (G_B)^{y_1} \bmod P_B$ where small "$y_1$" is a fresh random number generated by the Node A.

K – Shared secret agreed between the nodes A and B

$K = (G_A)^{xy} \bmod P_A$ (Option 1)

$K = (G_B)^{x_1 y_1} \bmod P_B$ (Option 2)

$I_A$, $I_B$ – The identity used by the nodes A and B in the trust negotiation interchange and embedded in the respective identity certificates (i.e. Cert A and Cert B)

SKA and SKB– Private keys of the certificates Cert A and Cert B respectively

$E_{Key}(M)$ – Encryption of M using the symmetric key "Key".

$S_{Key}(M)$ – Signing of M using the asymmetric secret key "Key".

$HMAC_{Key}(M)$ – Keyed-Hashing for Message Authentication applied to M using the key "Key" to ensure data integrity.

$K_0 = HMAC_K(X,Y,1)$ i.e. Encryption key used in the Phase 1 (Option 1).

$K_1 = HMAC_K(X,Y,2)$ i.e. Encryption key used in the Phase 2&3 (Option 1).

$K_2 = HMAC_K(X,Y,3)$ i.e. Authentication key used in Phase 2&3 (Option 1).

$K_0 = HMAC_K(X_1,Y_1,1)$ i.e. Encryption key used in the Phase 1 (Option 2).

$K_1 = HMAC_K(X_1,Y_1,2)$ i.e. Encryption key used in the Phase 2&3 (Option 2).

$K_2 = HMAC_K(X_1,Y_1,3)$ i.e. Authentication key used in Phase 2&3 (Option 2).

Data – Data transmission between the two participating nodes.

Data-Id – Unique sequence number for each data transmission in phase 2&3

SP – Service Policy of the node B

UID – Unique ID generated by the node B to identify the Trust Negotiation

RID – Request ID generated by the node to match the appropriate Certificate_Request with Certificate_Reply. Also used to match Service_Policy_Request with Service_Policy_Reply

SW – Service wanted by the node A from the Node B

SO – Service offered by the node B to the node A

SF – Service available without a trust negotiation on the node B

SU – Service unlocked by the node B after a trust negotiation for the node A

Shared Information between the Initiator and the Responder:

Encryption / Signature / Certificate / Hashing algorithms and Key Sizes

The identities ( $I_A$ and $I_B$) used in the RTM protocol phase 1 have to be embedded into the certificates used by the nodes *A* & *B*. This is achieved by embedding the attribute/value pair *ID_Public*= $I_A$ or *ID_Public*= $I_B$ in the identity certificates along with the Diffie-Hellman parameters *p* and *g*. However some nodes may want to mask their true identity so the $I_A$ may just be a pseudonym value for broadcast in plaintext (in the RTM protocol), while the real identity is embedded in the certificate using the *ID_Private* attribute. A node may have several pseudonyms each linked to the real identity with a certificate. Thus the real identity of a node is shielded from the passive eavesdropping attackers as the identity certificates used in the RTM protocol interchange are sent encrypted. An example of using this identity protection in a certificate is presented in Figure 30.

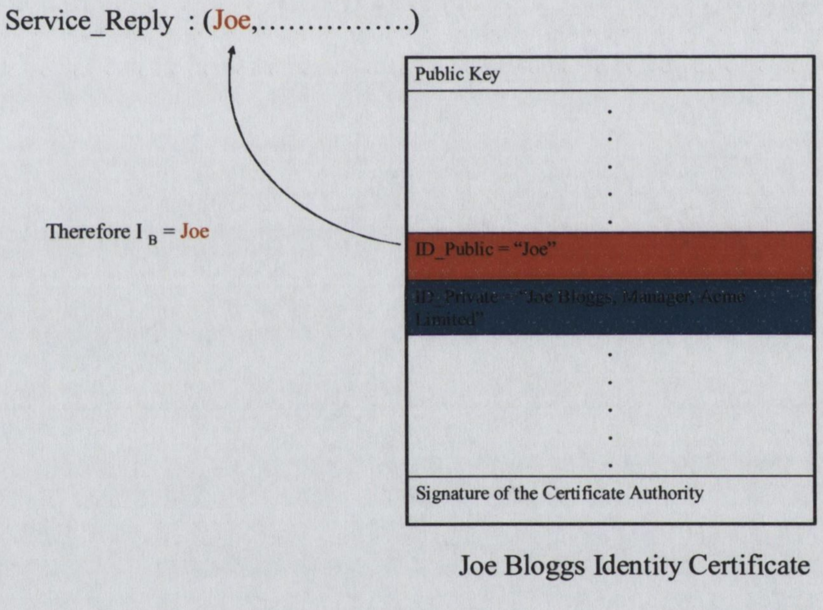

Service_Reply : (Joe,................)
Public Key
Therefore $I_B$ = Joe
ID_Public = "Joe"
ID_Private = "Joe Bloggs, Manager, Acme Limited"
Signature of the Certificate Authority
Joe Bloggs Identity Certificate

**Figure 30 : Identity Shielding**

## 4.3.2 RTM Organization

The RTM protocol interchange between the two participating nodes are governed by the *Certificate Exchange Agent* (CEA) which also controls the access to the services offered by the node. The strategy of progressive exchange of certificates is also decided by the CEA depending on inputs from the data structures in the RTM layer (Figure 31). The CEA also contains the key negotiation status data structures similar to the NTM layer. A *Local Certificate Store* (LCS) contains the local certificates to be used in negotiation along with the trusted certificate issuers (CA's) and their respective Certificate Revocation List (CRL's). The *Certificate Release Policy* (CRP) contains the release policies for each of the certificate in the LCS which may be used for trust negotiation. A Certificate Exchange History (CEH) keeps a record of the present and cache of old trust negotiations for ease of future and present negotiations. The mapping between the services allowed and the node to which it is allowed is kept in the *Service Access Table* (SAT). The Service Policy (SP) contains the association between the attribute name/value pair(s) and the associated services available on the node. The language for writing these policies is presented in the Appendix B.

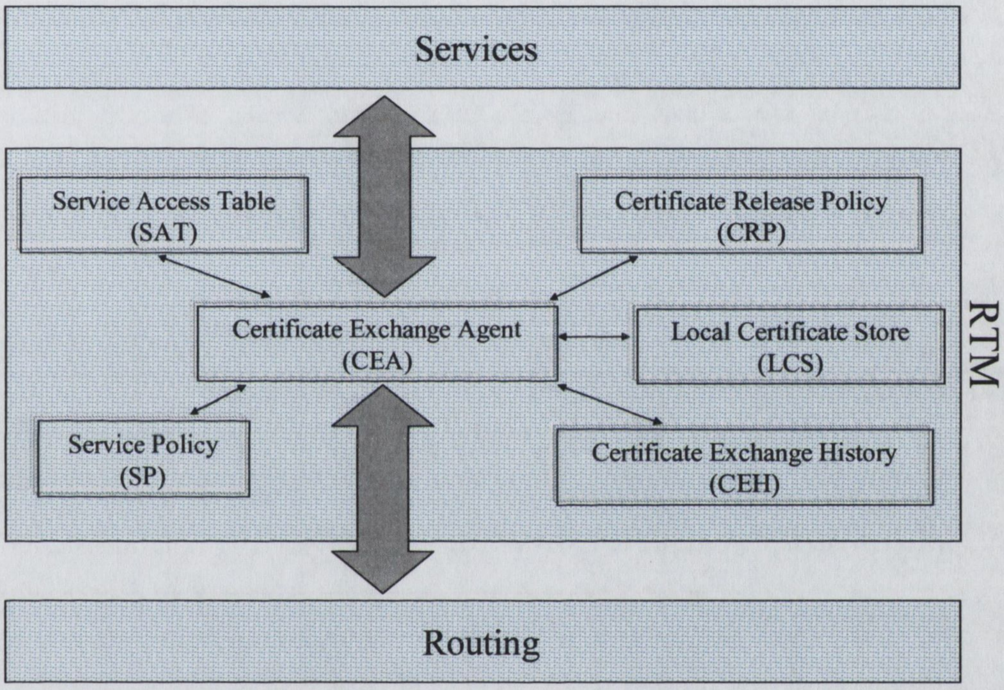
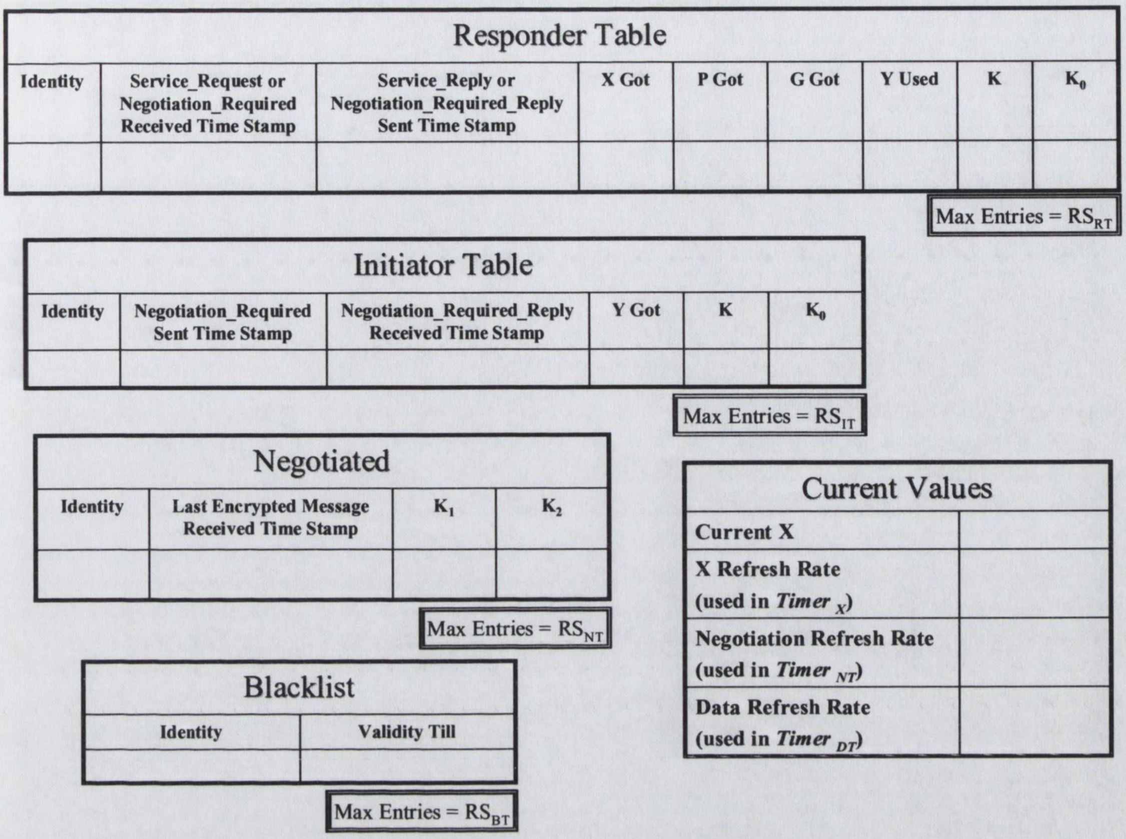


**Figure 31 : RTM Data Structure Organization**

The data structures contained in the *Certificate Exchange Agent* (CEA) are shown in the Figure 32. These data structures are similar to that used in the NTM layer and contain the states of the RTM protocol phase 1 interchange. Only major change is the addition of the *Initiator Table* to hold states when the node initiates a RTM protocol interchange (phase 1 - option 2). In these data structures of the RTM layer, the *identity* of the remote node is used to identify a protocol interchange whose state is being kept. The types of timers used in the RTM protocol are same as that of the NTM protocol. However, the intervals of the timers used in the RTM layer is larger than the ones used in the NTM layer to compensate for the multi-hop communications.



Responder Table

| Identity | Service_Request or Negotiation_Required Received Time Stamp | Service_Reply or Negotiation_Required_Reply Sent Time Stamp | X Got | P Got | G Got | Y Used | K | $K_0$ |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |

Max Entries = $RS_{RT}$

Initiator Table

| Identity | Negotiation_Required Sent Time Stamp | Negotiation_Required_Reply Received Time Stamp | Y Got | K | $K_0$ |
|---|---|---|---|---|---|
| | | | | | |

Max Entries = $RS_{IT}$

Negotiated

| Identity | Last Encrypted Message Received Time Stamp | $K_1$ | $K_2$ |
|---|---|---|---|
| | | | |

Max Entries = $RS_{NT}$

Blacklist

| Identity | Validity Till |
|---|---|
| | |

Max Entries = $RS_{BT}$

Current Values

| | |
|---|---|
| Current X | |
| X Refresh Rate (used in *Timer* $_X$) | |
| Negotiation Refresh Rate (used in *Timer* $_{NT}$) | |
| Data Refresh Rate (used in *Timer* $_{DT}$) | |

**Figure 32 : Data Structures in the Certificate Exchange Agent (CEA)**

It is difficult to construct a full state machine for the entire RTM protocol due to use of policies in the decision making process. Therefore the analysis of the RTM protocol is split into two parts. The first part presents the state machine for the phase 1 in which key agreement takes place. This is followed by the algorithm governing the phase 2 interchange in which the main trust negotiation takes place.

### 4.3.3 State Machine of the Key Agreement in the RTM Protocol

The initial three messages exchanges in the RTM protocol phase1, options 1 and 2 are similar to the NTM protocol. Two options in the phase 1 implies that there are six messages in the RTM protocol key agreement instead of the three in the NTM protocol. Moreover, the RTM protocol is a reactive protocol so there is no periodic broadcast to discover the remote node. This necessitated a new state machine depicted in Figure 33. The states for the responder are same as that of the NTM protocol (i.e. lower half consisting of the *Received 1*, the *Awaiting 3* and the *Received 3* states). Therefore for sake of brevity, the analysis of these states will not be repeated in this section.

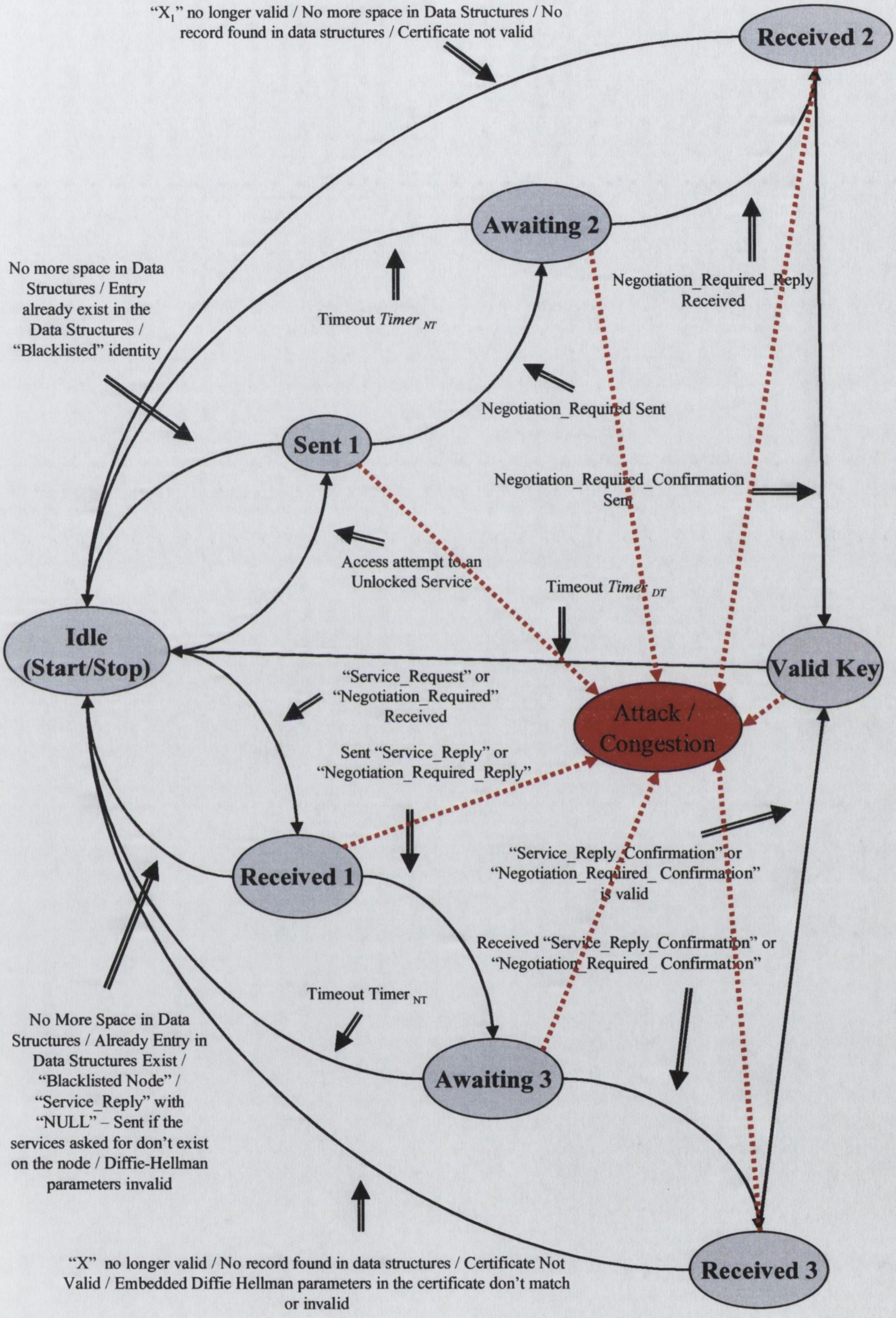
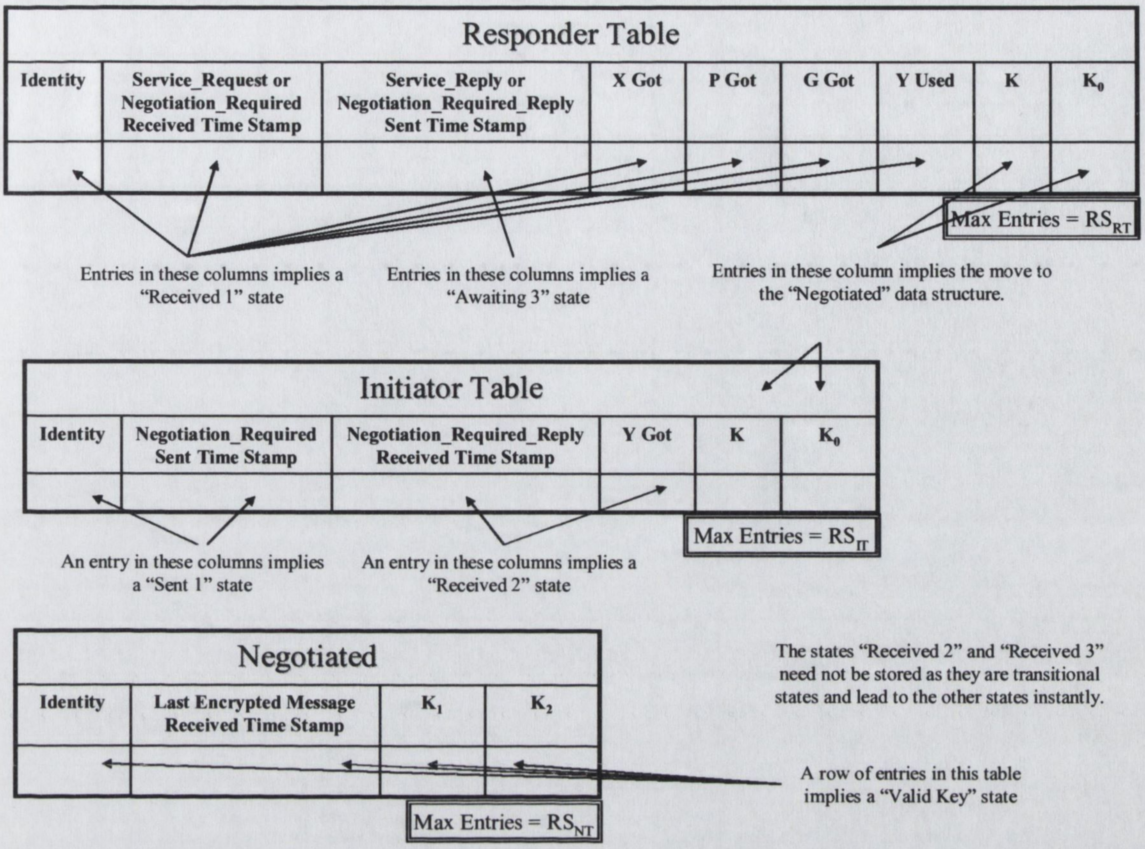

"X$_1$" no longer valid / No more space in Data Structures / No record found in data structures / Certificate not valid
Received 2
Awaiting 2
No more space in Data Structures / Entry already exist in the Data Structures / "Blacklisted" identity
Timeout *Timer $_{NT}$*
Negotiation_Required_Reply Received
Negotiation_Required Sent
Sent 1
Negotiation_Required_Confirmation Sent
Access attempt to an Unlocked Service
Timeout *Timer $_{DT}$*
Idle (Start/Stop)
Valid Key
"Service_Request" or "Negotiation_Required" Received
Attack / Congestion
Sent "Service_Reply" or "Negotiation_Required_Reply"
"Service_Reply_Confirmation" or "Negotiation_Required_ Confirmation" is valid
Received 1
Received "Service_Reply_Confirmation" or "Negotiation_Required_ Confirmation"
Timeout Timer $_{NT}$
No More Space in Data Structures / Already Entry in Data Structures Exist / "Blacklisted Node" / "Service_Reply" with "NULL" – Sent if the services asked for don't exist on the node / Diffie-Hellman parameters invalid
Awaiting 3
"X" no longer valid / No record found in data structures / Certificate Not Valid / Embedded Diffie Hellman parameters in the certificate don't match or invalid
Received 3

**Figure 33 : State Machine of the RTM Protocol (Phase 1)**

The initiator part of the state machine had to be modified as the RTM protocol is a reactive protocol unlike the proactive NTM protocol. A new state *Sent 1* is added because of the option 2 of the RTM protocols phase 1 interchange. An unauthorized access attempt by a remote node results in a *Negotiation_Required* message being sent. Then the node waits for a finite time (by use of timer *Timer* $_{NT}$) for receipt of an appropriate *Negotiation_Required_Reply* message. If the *Negotiation_Required_Reply* is received then the state machine transits to the *Received 2* state. Otherwise, the state machine returns to the *Idle* state on expiry of the timer *Timer* $_{NT}$. This periodic clearing of states mitigates to an extent the DoS attacks launched against the initiator. The *Received 2* state can transit to either the *Valid Key* state or the *Idle* state. This *Valid Key* state transition occurs if the *Negotiation_Required_Reply* message received is valid. Once the *Valid Key* state is reached the use of timer *Timer* $_{NT}$ is discarded and instead the timer *Timer* $_{DT}$ is attached to the agreed shared secret. The expiry of the timer *Timer* $_{DT}$ ensures that the common keys which haven't been used for a per-determined time is discarded. This might happen if the remote participating node in the RTM key agreement is unreachable. The RTM phase 1 has the same attacks and mitigation as the NTM protocol as both have the same cryptographic properties. Again the issues of intrusion detection and mitigation are outside the context of this thesis.

The state machine for the RTM protocol has two more similarities with the NTM protocol. Firstly, the non-storage of the transitional states *Received 2* and *Received 3* leading to decrease in states against which a memory exhaustion type of DoS attack can be mounted by an attacker. Secondly, incorporated into the state machine is the *Attack/Congestion* state which is reached when the node detects excessive numbers of fake messages or frequent computations leading to unsuccessful key agreement. The user of the node can either adjust the timers or take further action against the remote node outside the context of this thesis. The mapping between the states of the state diagram and the data structures in the Certificate Exchange Agent (CEA) is given in Figure 34.

The key agreement in the phase 1 clears the way for the actual trust negotiation to take place in a confidential manner in the phase 2. This trust negotiation is governed by the service policy and the certificate release policies of the two participating nodes. These are presented in the next section.



Responder Table

| Identity | Service_Request or Negotiation_Required Received Time Stamp | Service_Reply or Negotiation_Required_Reply Sent Time Stamp | X Got | P Got | G Got | Y Used | K | $K_0$ |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |

Max Entries = $RS_{RT}$

Entries in these columns implies a "Received 1" state

Entries in these columns implies a "Awaiting 3" state

Entries in these column implies the move to the "Negotiated" data structure.

Initiator Table

| Identity | Negotiation_Required Sent Time Stamp | Negotiation_Required_Reply Received Time Stamp | Y Got | K | $K_0$ |
|---|---|---|---|---|---|
| | | | | | |

Max Entries = $RS_{IT}$

An entry in these columns implies a "Sent 1" state

An entry in these columns implies a "Received 2" state

Negotiated

| Identity | Last Encrypted Message Received Time Stamp | $K_1$ | $K_2$ |
|---|---|---|---|
| | | | |

Max Entries = $RS_{NT}$

The states "Received 2" and "Received 3" need not be stored as they are transitional states and lead to the other states instantly.

A row of entries in this table implies a "Valid Key" state

**Figure 34 : Relationships between the States and Data Structures of CEA**

## 4.3.4 Service Policy and Certificate Release Policy

Local policies play a major role in determining the trust negotiation strategy. The two types of policy's used in the RTM are the *service policy* and *certificate release polices*. The service policy contains a mapping between service(s) offered by the node and the attributes in the valid certificates received from the remote node. Figure 35 gives a sample service policy for the scenario mentioned in the introduction chapter. In the example, the general service is open to all remote nodes. However access to *PoliceInfo* and *SecretAssets* services require valid certificates from the remote node with the appropriate attribute name/value pairs(s). Moreover, the *SecretAssets* service is a secret service and is not disclosed during the initial enumeration of services using the *Service_Reply* message. The conditions under which the service policy can be released to the remote node are contained in the disclosure policy. This service policy can also be released manually by the user of the service provider node, if a deadlock in negotiation is detected. Such a disclosure helps the node requesting the service to examine the service policy to find certificates required to satisfy the service policy.

| *Service* | *Certificates Required* |
|---|---|
| **General Service**<br>Public information | **Access to Everyone**<br>No attributes from remote certificates required |
| **PoliceInfo Service**<br>Police confidential information | **Other police units**<br>The valid remote certificate(s) should have the attributes name/value pairs confirming that the node is a police unit. |
| **SecretAssets Service : (Secret)**<br>Identity of informants and related confidential information | **Police Drug units**<br>The valid remote certificate(s) should have the attributes name/value pair(s) confirming that the node is a police drug unit with the rank above that of a "trainee". |
| ***Disclose Service Policy if,***<br>The valid remote certificate(s) should have the attributes name/value pair(s) confirming that the node is a police unit. | |

**Figure 35 : Service Policy for Access to Services on the Drug Units Node**

If the remote node had a trust negotiation in past with the service providing node and its CEH has a record of the SecretAssets service, then such remote nodes can produce the relevant certificates directly without waiting for the service policy disclosure. The above mentioned service policy written using the policy language defined in Appendix B is,

***General Service***: Free<br>
***PoliceInfo Service***: Organization=Police AND Certificate Issuer=Police<br>
***SecretAssets Service***: Secret: Organization=Police AND Unit=Drug AND Rank<> "trainee" AND Certificate Issuer=Police<br>
**Disclosure Policy:** Organization=Police AND Certificate Issuer=Police

The first field is the name of the service(s) followed by the optional secret field. At the end comes the expression of attribute name/value pair(s) required in certificates to satisfy access to the service(s). The service policy ends with the optional disclosure policy of the service policy. The certificate release policies are also formulated using the same policy language. Each local certificate in the Local Certificate Store (LCS) has an attached certificate release policy. This policy governs under what conditions the certificate can be released to the remote node. An example of such a policy for a drug unit's officer identification certificate is,

*(Organization=Police AND Certificate Issuer=Police) AND (Department=Drug AND Certificate Issuer=Police) AND ((Rank=Detective OR Rank=Sergeant) AND Duty=Patrolling AND Certificate Issuer=Personnel Department, Police)*

This certificate can be released to the node from the Police Drugs department. In addition the remote node has to prove that it is a detective or sergeant assigned to patrolling duties at night. The trusted certificate issuers are also specified. This implies if the attributes present in certificates are not signed by these certificate authorities then the above policy evaluation is false which in turns means that the local certificate will not be released. The policy conditions are evaluated for true or false using the compliance checker. The outputs of the compliance checker are used by the CEA to direct the phase 2 of the RTM protocol interchange.

### 4.3.5 Compliance Checker and Manual Intervention

The compliance checker for a service proving nodes takes inputs shown in Figure 36. The outputs are used by the *Certificate Exchange Agent* (CEA) to execute phase 2 of the RTM protocol. Every time the node receives a remote certificate(s) during the phase 2 of the RTM protocol, the compliance checker is invoked. Thus, the compliance checker is called iteratively over the course of the RTM protocol phase 2 trust negotiation exchange. To identify a trust negotiation the compliance checker uses the remote node's identity and UID.

The compliance checker can then return three status answers depending on the inputs provided. A yes status returned implies all the service(s) wanted by the remote node can be unlocked for access. Thus, the input *services wanted* and the output *services unlocked* are same. This leads to sending of the *Negotiation_End* message to successfully conclude the RTM phase 2 protocol interchange. A no or partial status returned implies that some additional certificates are required from the remote node for unlocking the wanted service(s). These statuses may also result from the fact that all or some of the service(s) asked for by the remote node are not available on the node. A partial status also returns the subset of the service(s) wanted which can be unlocked using the services unlocked output. In contrast, the no status returns no output in form of services unlocked. The *eager strategy* input allows the compliance checker to release all the certificates which are unlocked but not asked for by the remote node. This option can lead to disclosure of too much information about the node to others and should be only used as a last resort for automatic trust negotiation using the phase 2 of the RTM protocol.

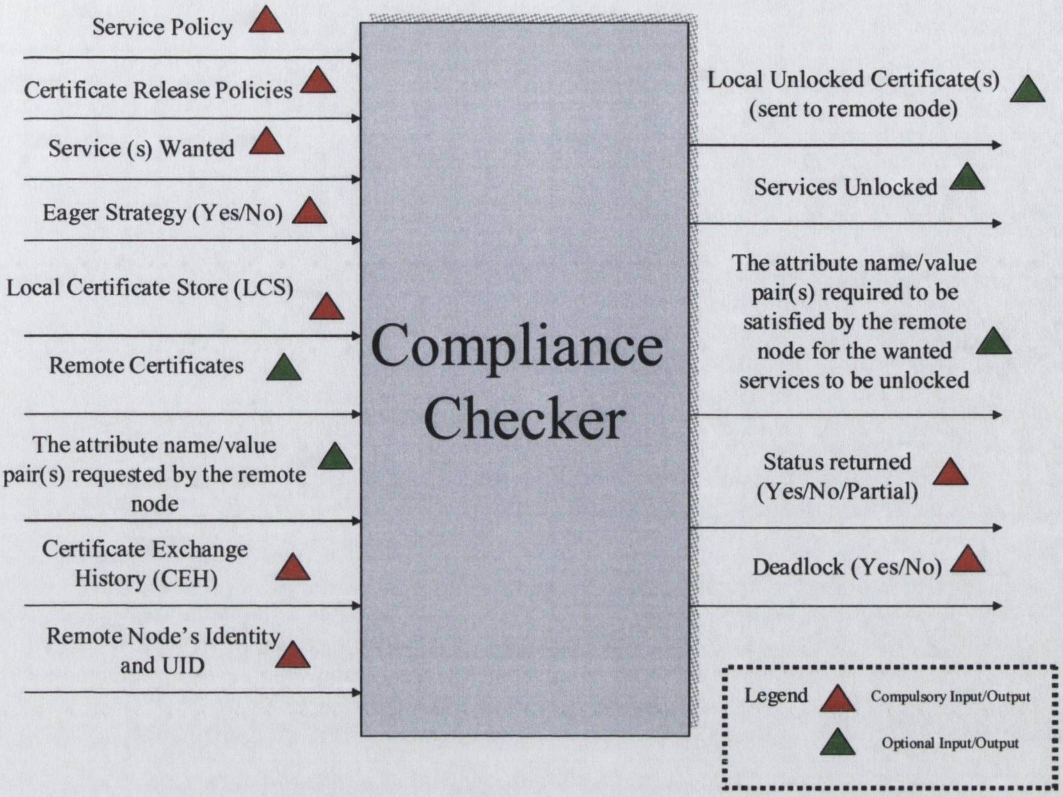

Service Policy
Certificate Release Policies
Service (s) Wanted
Eager Strategy (Yes/No)
Local Certificate Store (LCS)
Remote Certificates
The attribute name/value pair(s) requested by the remote node
Certificate Exchange History (CEH)
Remote Node's Identity and UID
Compliance Checker
Local Unlocked Certificate(s) (sent to remote node)
Services Unlocked
The attribute name/value pair(s) required to be satisfied by the remote node for the wanted services to be unlocked
Status returned (Yes/No/Partial)
Deadlock (Yes/No)
Legend
Compulsory Input/Output
Optional Input/Output

**Figure 36 : Compliance Checker for the One-to-One Interaction (Service Provider)**

The compliance checker also returns the attribute name/value pair(s) needed to satisfy the service policy when it returns a partial or a no status. These are used in the *Certificate_Request* messages to request for the appropriate certificates from the remote node. The no and partial status can also set the *deadlock* output to be true. This is the cue for human intervention since the automatic processes of trust negotiation has failed. If the node is a service requester, the change in the compulsory inputs and outputs are shown in Figure 37. The main change is that the *service policy* and the *status returned* are defunct making them optional during the operation of the compliance checker for a service requester node. If the compliance checker indicates a deadlock then the user is asked for intervention. To aid in the deadlock detection, the compliance checker has access to the negotiation history from the *Certificate Exchange History* (CEH). A user can manually release certificate(s) or the service policy to a remote node. If the deadlock still occurs frequently, the user can also edit the service policy and individual certificate release policies. Thus, the user intervention in the RTM layer is mainly provided for deadlock resolution in the trust negotiation performed by the phase 2 of the RTM protocol.

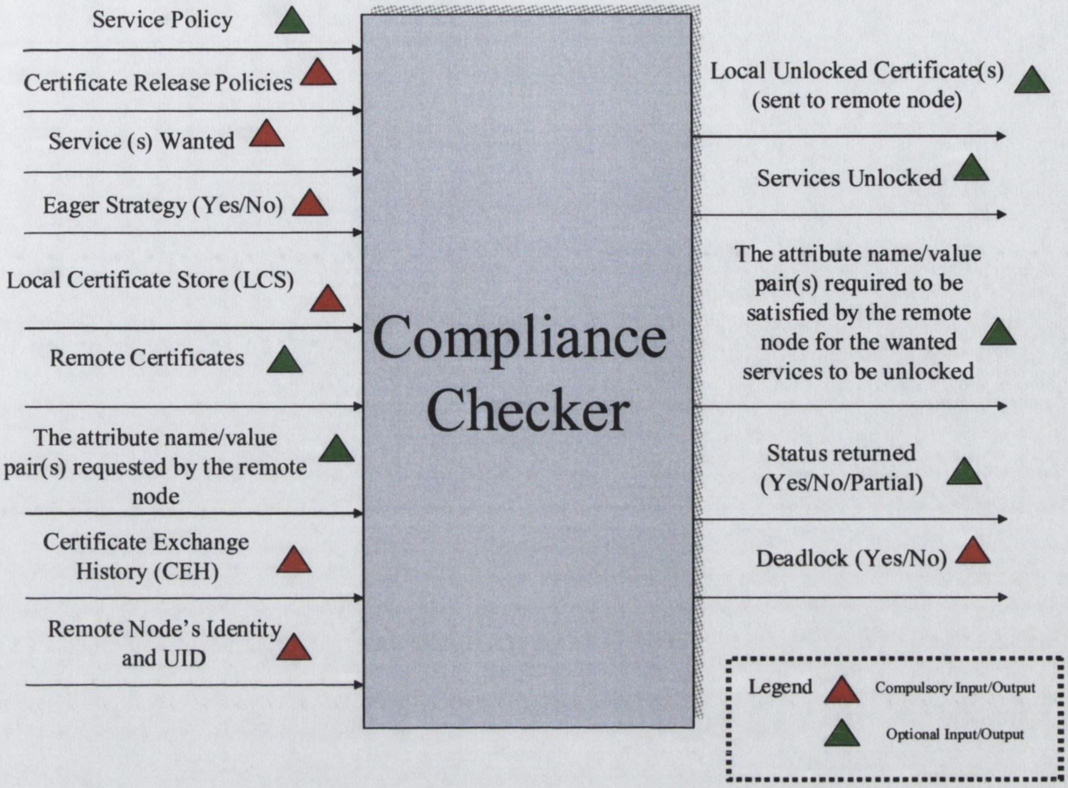

Service Policy
Certificate Release Policies
Service (s) Wanted
Eager Strategy (Yes/No)
Local Certificate Store (LCS)
Remote Certificates
The attribute name/value pair(s) requested by the remote node
Certificate Exchange History (CEH)
Remote Node's Identity and UID
Compliance Checker
Local Unlocked Certificate(s) (sent to remote node)
Services Unlocked
The attribute name/value pair(s) required to be satisfied by the remote node for the wanted services to be unlocked
Status returned (Yes/No/Partial)
Deadlock (Yes/No)
Legend
Compulsory Input/Output
Optional Input/Output

**Figure 37 : Compliance Checker for the One-to-One interaction (Service Requester)**

This thesis does not focus on the policy issues as the emphasis is on proposing a trust negotiation protocol and the associated framework that is deployable in a real Ad Hoc network. Due to paucity of time and the difference in implementation methods, the pre-developed trust negotiation languages [swy+02] for policy definition are not used in this thesis. Instead for testing of the protocols a rudimentary language for writing the policies is defined in Appendix B. This rudimentary language and the associated compliance checkers satisfy the requirements (see Appendix B) of a policy language [swy+02].

### 4.3.6 RTM Layer Summary

The one-to-one trust negotiation system proposed in this section is suitable for providing access control to services provided by a node in an Ad Hoc network. A node does not require a pre-configured common shared secret to authenticate a remote node for access control. Instead a node can dynamically build trust by having a progressive, confidential and role-free exchange of certificates based on the local policies of the two participating nodes. This implies that a node can deploy in a new area of operation with the same certificates. The node providing the service converts the attributes from the valid remote node's certificate(s) into access control permissions using a local service policy. This use of policies implies that the security requirements can be varied according to the deployment scenario. Moreover, little coordination is required to operate the trust negotiation scheme. Only the two participating nodes have to take part. This is unlike the other approaches for Ad Hoc security where several nodes are required to coordinate, even if two nodes want a one-to-one interaction. Moreover, the confidentiality and the integrity of all communications in the one-to-one scheme are protected by a novel two-tier key formation. This key formation is independent of the routing algorithm used by the nodes of the Ad Hoc network. In brief the one-to-one trust negotiation scheme presented in this chapter meets the following criteria of a good Ad Hoc cryptographic system [zh99].

**Authentication** – The scheme provides for mutual authentication of identities of the users of the nodes during the key formation. The services are only provided to the remote authenticated node which have negotiated trust and been found eligible according to the service policy.

**Confidentiality** – The communications in the network is made confidential by using a two-tier key formation. Moreover, certificates are encrypted during the protocol interchange, thus protecting their disclosure to passive attackers. The sensitive confidential certificates are protected against disclosure to malicious remote nodes by use of certificate release policies.

**Non-Repudiation** – The use of signatures and certificates ensures that a node cannot falsify the trust negotiation process.

**Availability** – The scheme through use of timers is able to partially mitigate the memory and CPU exhaustion type of DoS attacks.

**Integrity** – The encrypted communications in the network have their integrity protected by use of the HMAC algorithm.

One of the motivations for proposing a trust negotiation based approach for access control in Ad Hoc networks was that other existing Ad Hoc security schemes are CPU intensive for commodity wireless handheld devices. Therefore to test the performance of the one-to-one scheme, the next section presents a prototype implementation on a real Ad Hoc testbed consisting of commodity handheld wireless devices.

## *4.4 Implementation*

This section describes the implementation and testing of the one-to-one trust negotiation scheme on the actual mobile wireless devices The scheme is implemented using the NTRG mobile test-bed with its generic stack structure [md01a, md01b] (see Figure 38). This stack structure allows a LEGO like approach for the development and assembly of the networking protocols. In this architecture, the layers generally communicate with a layer above and below it using the messaging queue. The bottom layers of the stack interact with the networking hardware and top layers consisting of applications interact directly with the user. This generalized stack allowed for rapid prototype development of applications on Pocket PC's (Win CE) as well as fixed Workstations (Win32).

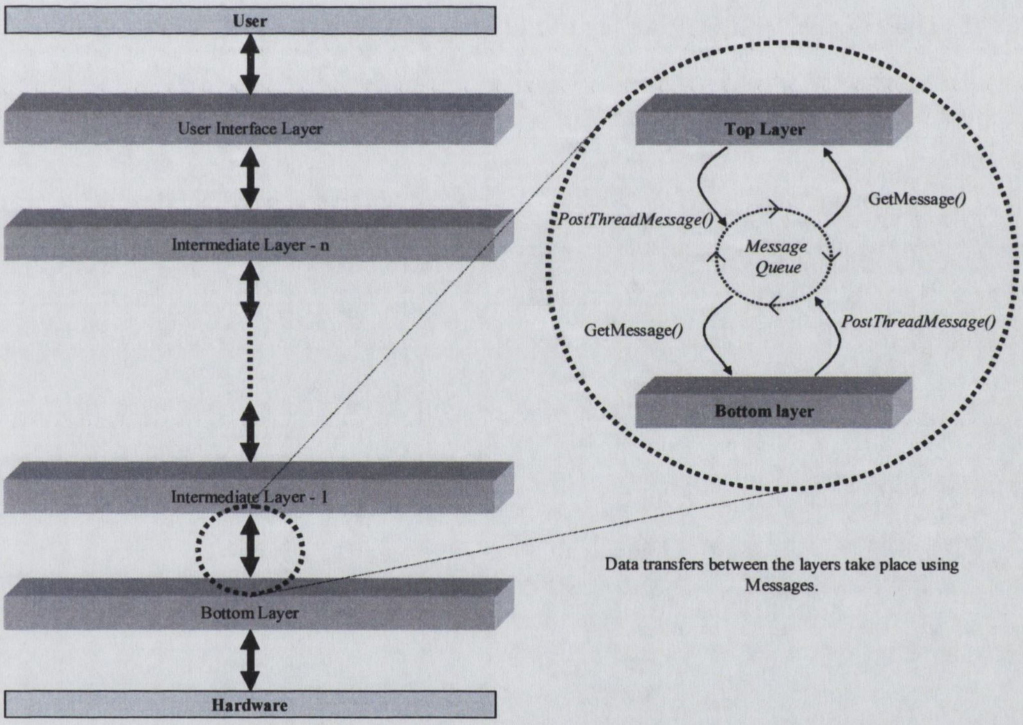


**Figure 38 : Generalized NTRG Stack**

### 4.4.1 Layer Structure for One-to-One Trust Negotiation

The protocol stack for the one-to-one trust negotiation scheme using the NTRG stack structure is shown in Figure 39. At the bottom are the 802.11 layer which is a radio interface to an IEEE 802.11b network card using broadcast IP and the Ethernet layer that is an interface to an Ethernet card. If the node acts as a bridge between the fixed network and the Ad Hoc network both the 802.11 and Ethernet layers are present. In case of mobile nodes only the 802.11 layer is present. On top of the layers interacting with the hardware is the implementation of the Neighbour Trust Model (NTM). The next layer in the stack implements the Dynamic Source Routing (DSR) [bjm04] Ad Hoc Networking Protocol. On top of which sits the Remote Trust Model (RTM) layer. This layer decides if the data coming from below can go up to a particular service by checking on the trust negotiation status. The two services provided currently are the person-to-person telephony application and the Instant Messaging like chat program in the audio and the chat layers respectively. The audio application provides a simple person-to-person telephony using the Session Initiation Protocol (SIP) [sip] as the signalling protocol. The Instant Messaging relays simple text messages between the users of the nodes participating in the Ad Hoc network.

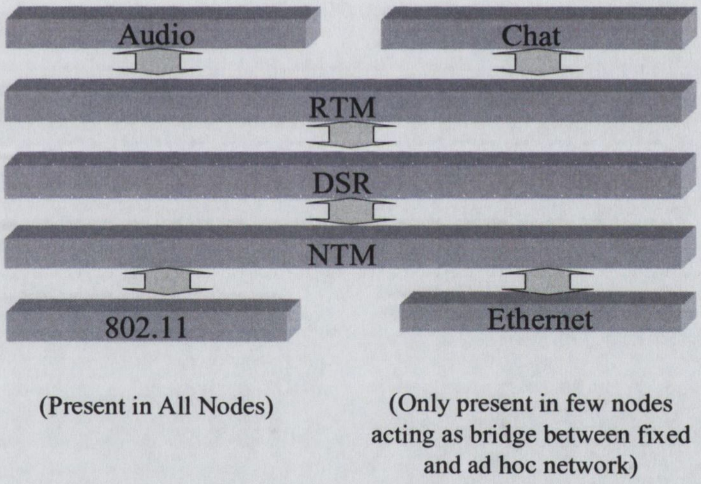
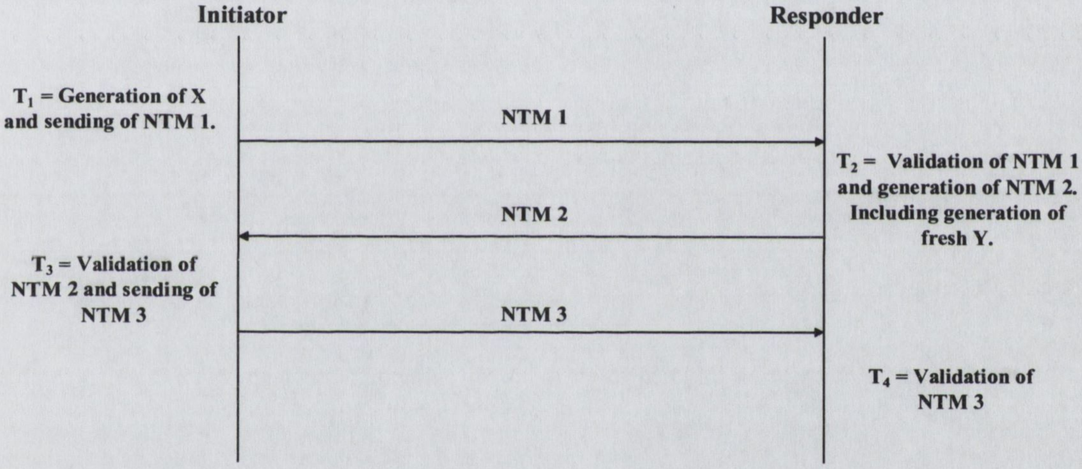

Audio
Chat
RTM
DSR
NTM
802.11
Ethernet
(Present in All Nodes)
(Only present in few nodes acting as bridge between fixed and ad hoc network)

**Figure 39 : Layer Structure for One-to-One Trust Negotiation**

### 4.4.2 Hardware Implementation

The NTRG testbed has six HP (Compaq) iPAQ H3630 Palmtop [ipaq] with a 206 MHz StrongARM processors and 32MB RAM (16MB ROM), running the Windows CE Pocket PC 2002 [wince] operating system. These devices use Orinoco Silver wireless cards [802.11] to provide the wireless interface. The performance of the implementation of the one-to-one trust negotiation scheme on these devices is tested initially by concentrating on the NTM protocol interchange involving just two wireless enabled handhelds. This test involves finding the average time for negotiation of an authenticated peer-to-peer shared secret in the NTM layer. The time required for the shared secret formation is crucial as a big time interval can mean noticeable latency before the actual data transmission occurs. This time is more crucial in scenarios where nodes have high mobility and don't spend more than a few seconds in contact with each other. Since in the NTM protocol phase 1 interchange there is no capability negotiation, the initial task is to fix the key sizes used in the NTM protocol. The size of the negotiated shared secret is fixed at 128-bit as it is the minimum recommended key length for the HMAC-MD5 algorithm used in the key derivations. Moreover the 1024-bit public key certificate is selected as its use makes the NTM phase 1 key agreement interchange about five times quicker than the 2048-bit ones (Figure 40).


Initiator
Responder
$T_1$ = Generation of X and sending of NTM 1.
NTM 1
$T_2$ = Validation of NTM 1 and generation of NTM 2. Including generation of fresh Y.
NTM 2
$T_3$ = Validation of NTM 2 and sending of NTM 3
NTM 3
$T_4$ = Validation of NTM 3

| Test Conditions | | Time in milliseconds (Average over 10000 tests) | | | | |
|---|---|---|---|---|---|---|
| Size of the Mutual Shared secret Agreed (bits) | Size of the Public Key of the Network Address Certificates (Bits) | $T_1$ | $T_2$ | $T_3$ | $T_4$ | Total Time for key agreement in NTM protocol |
| 128 | 512 | 7.6855 | 24.3710 | 26.0399 | 9.4134 | 67.5099 |
| *128* | *1024* | *7.5377* | *87.8972* | *101.6189* | *21.2467* | *218.3004* |
| 128 | 2048 | 7.7268 | 506.5320 | 560.1564 | 61.1251 | 1135.5403 |

**Figure 40 : Performance Tests of the NTM Protocol**

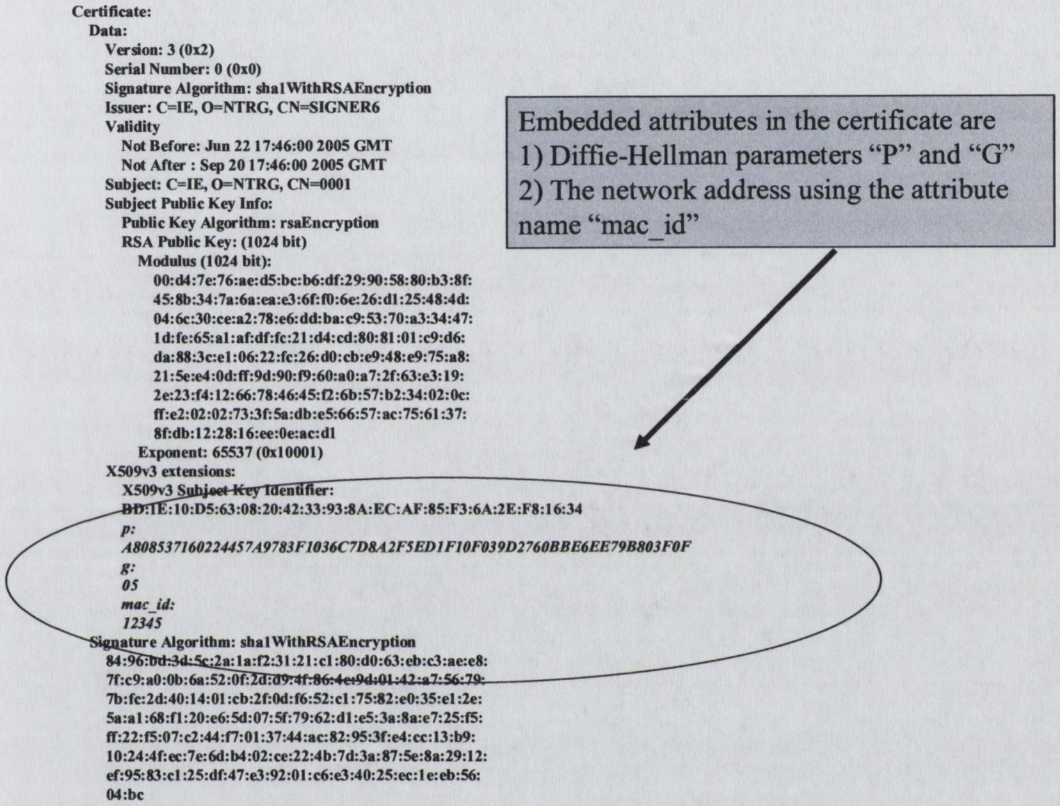
The conditions for the NTM test are,

- The test does not take into account the once-off initialization times of the various data structures as they are constant and less than 50ms.
- The X.509 framework [x.509] is used to implement (see Figure 41) the *address* certificate. This might add extra non–required parameters to certificates. The X.509

certificates used for implementation throughout this thesis has the RSA parameter "e" set to 65537 (recommended value) to prevent the Low Encryption Exponent Attack [b99]. However, this made the signing operation using the secret key considerably slower than the verification using the public key.

- To decrease the network latency time it was ensured that the two nodes are not in transmit/receive range of any other nodes. This ensured that the test executes in minimal time.
- The Windows CE port of the OpenSSL [openssl] cryptographic toolkit, version 0.9.7b is used for the implementation. This allowed the cryptographic code developed for the Win CE operating system to be used on the desktop machines (i.e. Win 32 based OS).
- Random number generators used to ensure the freshness of the Diffie-Hellman parameters needs to be properly seeded (i.e. have enough entropy). Thus in the implementation, windows events (i.e. RAND_event() function) along with the current content of the screen (i.e. RAND_screen() function) is used to seed the random number generator in OpenSSL. A check for enough entropy in the seed value is performed (i.e. RAND_status() function) to ensure a proper selection. On the nodes acting as bridge between the Ad Hoc and wired networks, the seed value is derived from the thermal noise in the hardware (i.e. also referred to as the Pentium Random Number Generator). This is essential as these types of nodes have an automated start-up and don't provide enough entropy to the OpenSSL random number seeding algorithm (i.e. the RAND_event() or RAND_screen functions).
- The data integrity of the encrypted traffic between the two participating mobile nodes is ensured using the Keyed-Hashing for Message Authentication (HMAC) algorithm [kbc97]. The specific sub-algorithm used is the HMAC-MD5 implying that the underlying hashing algorithm is the MD5 message digest. The minimal recommended key length for the HMAC-MD5 is 128-bits. The MD5 algorithm was selected over the SHA1 for use in the HMAC operation due to its better performance (Appendix A). It is important to note that the collision attacks on the underlying hashing algorithm are not considered as it is beyond the scope of this thesis.
- The derivation of the key $K_0$, $K_1$ and $K_2$ also uses the HMAC-MD5 algorithm. Thus the 128-bit key length is used for encryption using the Advanced Encryption Standard (AES) [dr01] symmetric cipher.

- The certificates used in the NTM protocol have an associated verification chain of length three (i.e. four certificates in the chain). This length of three was selected as it is the maximum certificates chain length on a standard windows installation (i.e. Win XP & Pocket PC 2002). Thus, the signature verification and the associated *address* certificate verification in the NTM protocol's key agreement test require four RSA verification operations.



```
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 0 (0x0)
    Signature Algorithm: sha1WithRSAEncryption
    Issuer: C=IE, O=NTRG, CN=SIGNER6
    Validity
      Not Before: Jun 22 17:46:00 2005 GMT
      Not After : Sep 20 17:46:00 2005 GMT
    Subject: C=IE, O=NTRG, CN=0001
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (1024 bit)
        Modulus (1024 bit):
          00:d4:7e:76:ae:d5:bc:b6:df:29:90:58:80:b3:8f:
          45:8b:34:7a:6a:ea:e3:6f:f0:6e:26:d1:25:48:4d:
          04:6c:30:ce:a2:78:e6:dd:ba:c9:53:70:a3:34:47:
          1d:fe:65:a1:af:df:fc:21:d4:cd:80:81:01:c9:d6:
          da:88:3c:e1:06:22:fc:26:d0:cb:e9:48:e9:75:a8:
          21:5e:e4:0d:ff:9d:90:f9:60:a0:a7:2f:63:e3:19:
          2e:23:f4:12:66:78:46:45:f2:6b:57:b2:34:02:0c:
          ff:e2:02:02:73:3f:5a:db:e5:66:57:ac:75:61:37:
          8f:db:12:28:16:ee:0e:ac:d1
        Exponent: 65537 (0x10001)
    X509v3 extensions:
      X509v3 Subject Key Identifier:
      BD:1E:10:D5:63:08:20:42:33:93:8A:EC:AF:85:F3:6A:2E:F8:16:34
      p:
      A808537160224457A9783F1036C7D8A2F5ED1F10F039D2760BBE6EE79B803F0F
      g:
      05
      mac_id:
      12345
  Signature Algorithm: sha1WithRSAEncryption
    84:96:bd:3d:5c:2a:1a:f2:31:21:c1:80:d0:63:eb:c3:ae:e8:
    7f:c9:a0:0b:6a:52:0f:2d:d9:4f:86:4e:9d:01:42:a7:56:79:
    7b:fc:2d:40:14:01:cb:2f:0d:f6:52:c1:75:82:e0:35:e1:2e:
    5a:a1:68:f1:20:e6:5d:07:5f:79:62:d1:e5:3a:8a:e7:25:f5:
    ff:22:f5:07:c2:44:f7:01:37:44:ac:82:95:3f:e4:cc:13:b9:
    10:24:4f:ec:7e:6d:b4:02:ce:22:4b:7d:3a:87:5e:8a:29:12:
    ef:95:83:c1:25:df:47:e3:92:01:c6:e3:40:25:ec:1e:eb:56:
    04:bc
```

Embedded attributes in the certificate are
1) Diffie-Hellman parameters "P" and "G"
2) The network address using the attribute name "mac_id"

**Figure 41 : OpenSSL Dump of the X.509 Address Certificate**

After determination of the key sizes for the shared secret negotiation the next task is to establish the timer intervals. These timer intervals are important as they provide partial DoS attack mitigation. An improper selection of the timer intervals can result in unsuccessful shared secret negotiation if the node density in the transmit/receive range is large. Thus the rationale for selecting the timer intervals for the testbed is presented in the next section. These timers are,

- *Timer $_X$* - The expiry of this timer makes the node broadcast the NTM 1 message with a fresh Diffie-Hellman parameter *X* (after a fresh random number generation). In the RTM layer expiry of this timer results only in generation of the parameter *X*.

- *Timer* $_{NT}$ – The responder attaches this timer to each unfinished key negotiation. The expiry of this timer implies that the unfinished key negotiation should be abandoned.
- *Timer* $_{DT}$ - If no encrypted data traffic is received from the appropriate remote node after the key negotiation phase, this timer expires making the node discard the negotiated shared secret (and the associated keys).

### *4.4.2.1 Timer Intervals in NTM Layer*

The selection of timer intervals in the NTM phase 1 key agreement is helped by the fact that there is no capability negotiation in the NTM protocol. This decreases the number of factors to consider. In this section the first timer to be examined is the *Timer* $_{X}$.

#### 4.4.2.1.1 Timer $_{X}$ Interval in NTM Layer

The "X" has to be valid for the minimal time required to successfully finish a NTM key agreement phase. This is dependent on the time required for the various cryptographic primitives, the network transmission times, the network conditions and the time required to process a network message by the software stack (listed in Figure 42). Out of these factors, barring the network congestion time delay the remaining factors can be predetermined by performing actual tests.

| **Serial Number** | **Description of the Operation** | **Number of operations** |
|---|---|---|
| 1 | $a^b$ mod n – Diffie-Hellman operations | 4 |
| 2 | RSA verify – Verification of the Certificates and the Signatures. | n1+n2+2 |
| 3 | RSA sign – Signatures required in the protocol run | 2 |
| 4 | Symmetric Encryptions – Formation of Encrypted Signatures | 4 |
| 5 | Symmetric Decryptions – Decryption of encrypted signatures | 4 |
| 6 | Hashing – Required for key generation | 2 |
| 7 | HMAC – Required for key generation | 2 |
| 8 | Network transmission time – Three network primitives | 3 |
| 9 | Maximum processing times – Maximum processing time required for each network primitive to reach the NTM layer | 4 |
| 10 | Allowance for network congestion | Variable |

n1= Number of certificates in the chain terminating with the certificate used by the responder

n2= Number of certificates in the chain terminating with the certificate used by the initiator

Assumption – All certificate use have the same length certificate on the initiator and responder

**Figure 42 : Major Factors in Determination of the Timer Interval**

Of all the factors the time allowance for network congestion is hard to estimate as accurate modeling of the wireless medium is difficult. This is illustrated in experiments by Cavin et al [css02] showing that for the same mobile Ad Hoc scenario the simulators [glomosim, ns2] produced diverse performance results. The difference between results on the simulator based testbeds is further illustrated in experiments by Haq et al [hk05].

Under ideal conditions the Diffie-Hellman value "X" has to be valid for just over 219 ms. This is obtained from the benchmark results (Figure 40) of the NTM phase 1 key agreement (i.e. $T_1+T_2+T_3+T_4=218.3004$ms). A further allowance of 2 ms (average) can be made for the message transmission and processing times. Therefore, the *Timer $_X$* interval for generation of the Diffie-Hellman "X" is fixed at 225 ms (slightly more to compensate for adverse network conditions) on the handheld mobile devices of the testbed. This interval of the timer *Timer $_X$* will be tested further in this section to show that it is effective under the maximum node density of the testbed.

#### 4.4.2.1.2 Timer $_{NT}$ Interval in NTM Layer

The interval for the timer *Timer $_{NT}$* has to be greater than of the timer *Timer $_X$* as it is essential to have additional time to complete the trust negotiation over the time required for the generation of the Diffie-Hellman public value "X". This is due to the possibility of network congestion caused by many nodes in the transmit/receive range. In the testbed the interval for the timer *Timer $_{NT}$* is found after testing several values in the worst case scenario. This is achieved by putting all the six mobile handheld wireless nodes of the testbed within the transmit/receive range of each other. Such a layout ensures the maximum probability of network congestion leading to frequent retransmissions. During the test all nodes are switched on simultaneously and the time required to form the peer-to-peer shared secrets between all the mobile nodes is recorded. Thus each participating node has five neighbours which resulted in overall concurrent formation of fifteen peer-to-peer shared secrets. The periodicity of the timer *Timer $_{NT}$* is varied by 25 ms and the average results for 100 tests are plotted in the Figure 43. The time values in the graph are total for the phase 1 of the NTM protocol to finish on all participating nodes. This graph only shows the major results among the many tested intervals the timer *Timer $_{NT}$*. These results clearly show that the interval of the timer *Timer $_{NT}$* affects the average time required for key agreement in phase 1 of the NTM protocol. The best performance in this test is obtained with the *Timer* $_X$ at 225 ms and *Timer* $_{NT}$ at 425 ms.

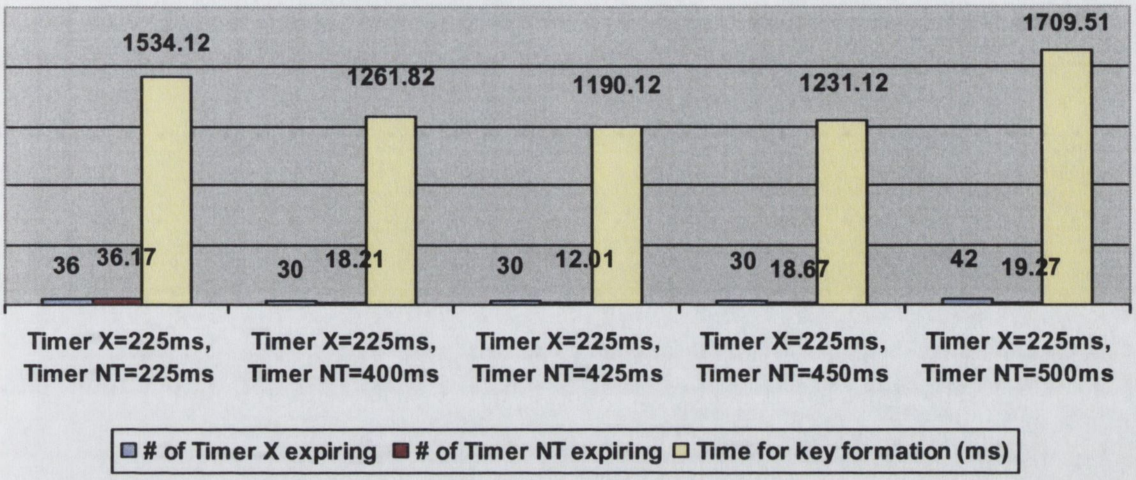


**Figure 43 : NTM Protocol Key Agreement Performance**

The optimal interval of the timer *Timer* $_{NT}$ obtained will vary if the number of nodes in the transmit/receive range increases. The next test verifies if the optimal interval for the timer *Timer* $_{NT}$ works reasonably if the node density is less than six. Thus keeping the periodicity of the timer *Timer* $_{NT}$ at 425 ms, the test was repeated 100 times with different node densities and the results graphed in Figure 44 . This graph shows that the optimal interval for the timer *Timer* $_{NT}$ worked reasonably in lesser node densities. Further work is required to determine the interval for the *Timer* $_{NT}$ for larger node densities, but the unavailability of inexpensive accurate simulators or actual hardware stymied the effort in this direction. However in emergency scenarios having comparable node density to the test conditions, the key formation in the NTM protocol will deliver a reasonable performance. It is also important to note that the maximum CPU usage on the handhelds during the running of these tests is about 25 percent.

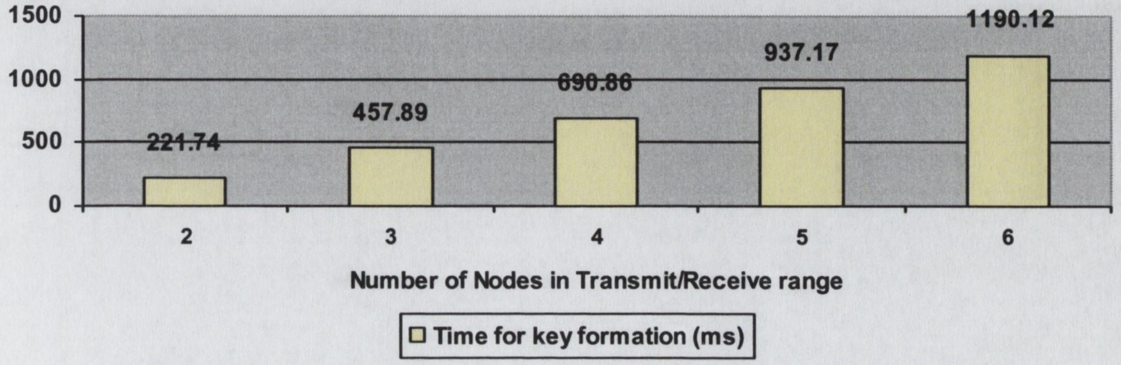


**Figure 44 : Performance of Phase 1 of the NTM protocol (Dynamic Membership)**

The tests above did not take account of the node mobility. To simulate mobility on the actual hardware with reproducible results is difficult. However, in the testbed the mobility of handheld nodes was simulated using the following method. Initially two nodes are switched on and the time required for the phase 1 NTM key agreement between them is recorded. Another node is then switched on and the time interval for this node to have NTM phase 1 key agreement with other two is noted. This is analogous to the situation of a mobile wireless node coming in range of a network and then joining the network by having authenticated peer-to-peer key formation with existing nodes. Similarly tests are carried on till number of nodes is six. The average results for 100 tests are tabulated in Figure 45 show a reasonable performance of the NTM phase 1 key agreement despite simulated mobility.

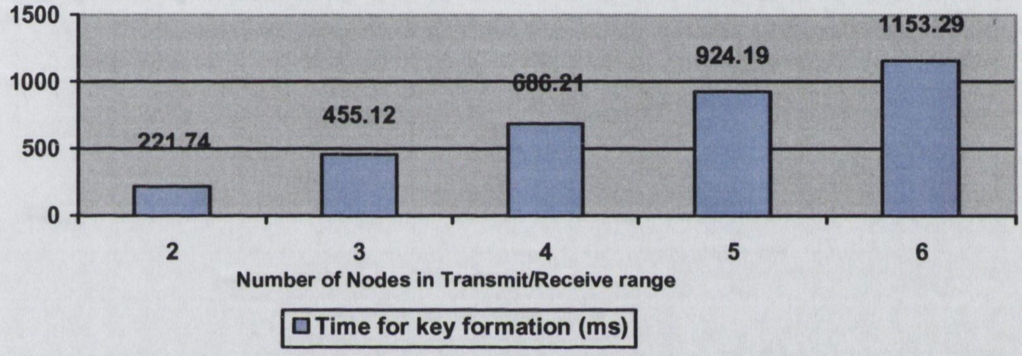


**Figure 45 : NTM Protocol Phase 1 Performance (Incremental Dynamic Membership)**

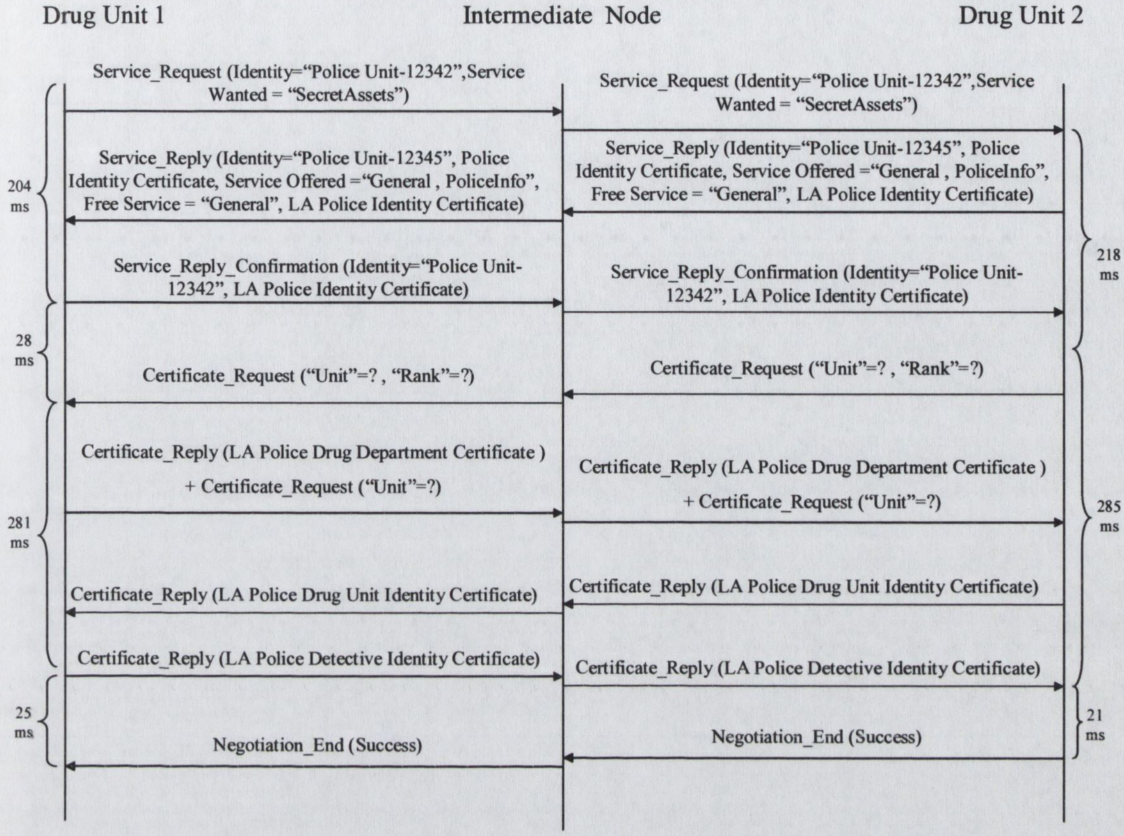
### 4.4.2.1.3 Timer $_{DT}$ interval in NTM layer

The interval of the timer *Timer* $_{DT}$ has to be significantly larger then than of timer *Timer* $_{NT}$ as constant traffic between the two nodes having a common shared secret cannot be assured. This timer periodicity mainly depends on the mobility of the node and the traffic patterns. Thus this timer has to be adjusted dynamically by the user during the operation. A small interval for the timer *Timer* $_{DT}$ can result in frequent NTM phase 1 key agreements. In contrast, a large interval can result in a node not being aware that a neighbour is not in range for a considerable time. In the prototype, the instances of the timer *Timer* $_{DT}$ in the NTM layer had the periodicity set at 2 seconds. This implies that a node cannot detect for 2 seconds if its neighbour with whom it had a peer-to-peer shared secret has left the transmit/receive range. The NTM layers tests in this section showed that the use of timers on the actual handheld mobile devices is feasible.

#### *4.4.2.2 RTM Performance*

Next performance tests where carried out on the RTM protocol. In these tests the emphasis is on the performance of the actual trust negotiation in the phase 2 of the RTM protocol. This change in focus is due to the fact that the performance of the phase 1 key agreement of the RTM protocol is similar to that of the NTM protocol (Both the protocols have similar cryptographic properties). The timers used in the RTM test were incremented by 20ms over the ones used in the NTM layer. This increase is necessary as the participating nodes in the RTM protocol interchange may be multi-hop (maximum of five as total number of nodes in the testbed is six). An average of 2 ms is required by a node to forward a message (i.e. time taken by the DSR routing algorithm) under maximum CPU conditions. Therefore it an average of 20 ms for a round trip of a message from one node to another five hops away. The rest of the implementation conditions for the RTM test is same as that for the NTM layer (i.e. hashing & encryption algorithms and the certificate chain length of three).

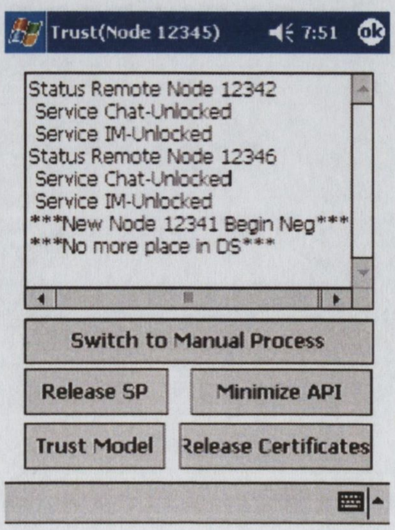
In the RTM protocol interchange for the test illustrated in Figure 46, three nodes were lined up in a straight topology (no other nodes in vicinity). The two nodes at the extremities could not communicate with each other except through the middle intermediate node. These nodes at the extremities perform a RTM protocol interchange (phase 1 and 2). The test used the trust negotiation scenario outlined before in this chapter and uses 1024-bit certificates to negotiate a 128-bit shared secret. The initiator of the RTM protocol test spends 204 ms on the phase 1 while the responder providing the service takes 218 ms. The main trust negotiation using the phase 2 of the RTM protocol takes 281 ms and 285 ms respectively on the two nodes. On the initiator, the entire process ends in 538 ms, while on the responder node providing the service it takes 524 ms. Thus the trust negotiation process between two nodes involving negotiation of a 128-bit shared secret and exchange of three 1024-bit modulus certificates is completed securely within a reasonable ½ second. The size of the X.509 1024-bit modulus certificates used in the RTM/NTM layer tests varied from 500 to 900 bytes with the variations in the size mainly due to the number of the embedded custom attribute name/value pairs. This ensured that the maximum sizes of the network messages in the NTM and RTM protocol is around 1300 bytes.



Drug Unit 1
Intermediate Node
Drug Unit 2
Service_Request (Identity="Police Unit-12342",Service Wanted = "SecretAssets")
Service_Request (Identity="Police Unit-12342",Service Wanted = "SecretAssets")
Service_Reply (Identity="Police Unit-12345", Police Identity Certificate, Service Offered ="General , PoliceInfo", Free Service = "General", LA Police Identity Certificate)
Service_Reply (Identity="Police Unit-12345", Police Identity Certificate, Service Offered ="General , PoliceInfo", Free Service = "General", LA Police Identity Certificate)
204 ms
218 ms
Service_Reply_Confirmation (Identity="Police Unit-12342", LA Police Identity Certificate)
Service_Reply_Confirmation (Identity="Police Unit-12342", LA Police Identity Certificate)
28 ms
Certificate_Request ("Unit"=? , "Rank"=?)
Certificate_Request ("Unit"=? , "Rank"=?)
Certificate_Reply (LA Police Drug Department Certificate ) + Certificate_Request ("Unit"=?)
Certificate_Reply (LA Police Drug Department Certificate ) + Certificate_Request ("Unit"=?)
281 ms
285 ms
Certificate_Reply (LA Police Drug Unit Identity Certificate)
Certificate_Reply (LA Police Drug Unit Identity Certificate)
Certificate_Reply (LA Police Detective Identity Certificate)
Certificate_Reply (LA Police Detective Identity Certificate)
25 ms
21 ms
Negotiation_End (Success)
Negotiation_End (Success)

**Figure 46 : Sample RTM Trust Negotiation Protocol**

The screenshot of the prototype on the Drug Unit 2 handheld is shown in the Figure 47. It shows a node 12345 allowing the nodes 12342 and 12346 access to its services. Also a failed negotiation with node 12341 due to the node 12345 reaching its internal limit (artificially limited for the test) of data structures (*Negotiated*) is shown.



```
Status Remote Node 12342
 Service Chat-Unlocked
 Service IM-Unlocked
Status Remote Node 12346
 Service Chat-Unlocked
 Service IM-Unlocked
***New Node 12341 Begin Neg***
***No more place in DS***
```

**Figure 47 : Screen Shot of the One-to-One Trust Negotiation Prototype**

#### 4.4.2.2.1 Combined RTM and NTM Test

This involved testing the performance of the RTM and NTM layers together with the mobile nodes within the transmit/receive distance of each other. The nodes are switched on simultaneously. Initially each node performs the NTM phase 1 interchange with others followed by the RTM phase 1 and 2. This RTM interchange follows the same interchange as illustrated in Figure 46 (without the intermediate node). In this test the timer intervals for the NTM and RTM layers are same as those used in the previous tests. This combined performance test is repeated 100 times for different number of nodes in the network and the average results are tabulated in the Figure 48. The time for the RTM phase 1 and 2 was recorded after the NTM component was over. At the end of the test, (n*(n+1))/2 peer-to-peer and similar number of end-to-end shared secrets are formed (n= number of nodes). Due to the increased CPU load the times for the RTM phase 1 slightly increased. However, the maximum CPU usage on the handhelds during the test peaked at about 50 percent. This implied there was enough CPU power left for other applications to run on the handhelds.

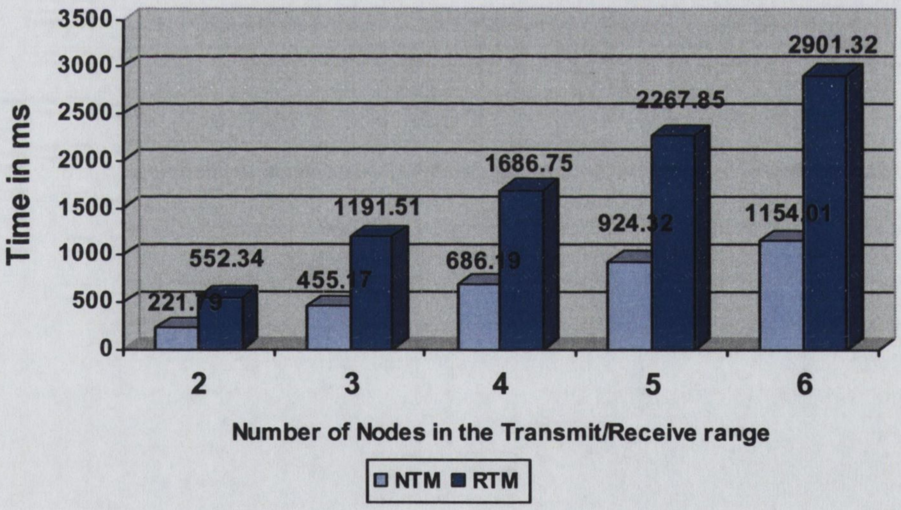


**Figure 48 : Performance of the One-to-One Trust Negotiation**

After the RTM key agreement phase and the trust negotiation, encrypted data is exchanged between the two nodes. The timer *Timer* $_{DT}$ is used to keep the status of an end-to-end (RTM) keys current. This timer's interval mainly depends on the mobility of the node and the traffic patterns. In the prototype implementation, the intevral for timer *Timer* $_{DT}$ in the RTM layer is set at 10 seconds. This implies that a node can wait at most for 10 seconds for encrypted traffic to be received before the concerned end-to-end key is discarded. The 10 seconds is the time the audio layer waits for an answer to an invite from a remote node. The timer *Timer* $_{DT}$ is used mainly in the NTM phase 2 and RTM phase 3 protocol interchanges whose overheads are discussed in the next section.

### *4.4.2.3 Encrypted Data Traffic Overheads*

The protocol components of the encrypted data interchanges (i.e. NTM phase 2 and RTM phase 3) use the symmetric encryption and hashing algorithms which are computationally inexpensive on handheld devices (see Appendix A). In construction of the encrypted primitives in the abovementioned protocol interchanges, the symmetric encryption is more computationally expensive than the hashing algorithms. To test the time to compute such network primitives, repeated tests (i.e. 100,000 iterations) where performed on several sizes of plaintext data. The average times for such computations are tabulated below,

| *Plaintext size in Bytes* | *Average time in ms to compute encrypted text and HMAC* |
|---|---|
| 100 | 0.3613 |
| 200 | 0.6115 |
| 300 | 0.8617 |
| 400 | 1.1119 |
| 500 | 1.4038 |
| 600 | 1.6540 |
| 700 | 1.9042 |
| 800 | 2.1544 |
| 900 | 2.4463 |
| 1000 | 2.6965 |

**Figure 49 : Encrypted Data Traffic Overheads**

In a typical end-to-end voice conservation (with GSM compression), 169 byte plaintext packets are sent at an average of 10 times a second. This entails an approximate CPU overhead of 5.81 ms for encryption/decryption on the sender and receiver nodes. These tests and the audio example show that the computational overhead for the construction of encrypted traffic primitives in the encrypted data transmission protocols (i.e. NTM phase 2 and RTM phase 3) is minimal.

In the one-to-one trust negotiation prototype, the NTM phase 2 and RTM phase 3 protocol interchanges use a 128-bit key on a 16 byte plaintext block. This plaintext block is automatically padded to a 16 byte boundary which adds 1 to 16 bytes overhead in the data transmission. This coupled with the use of HMAC-MD5 for message integrity adding 16 bytes, leads to the total overhead for each encrypted data message to vary from 17 bytes to 32 bytes (Figure 50). Therefore, the data transmission overhead for the encrypted data traffic using the NTM phase 2 and RTM phase 3 protocols is minimal.

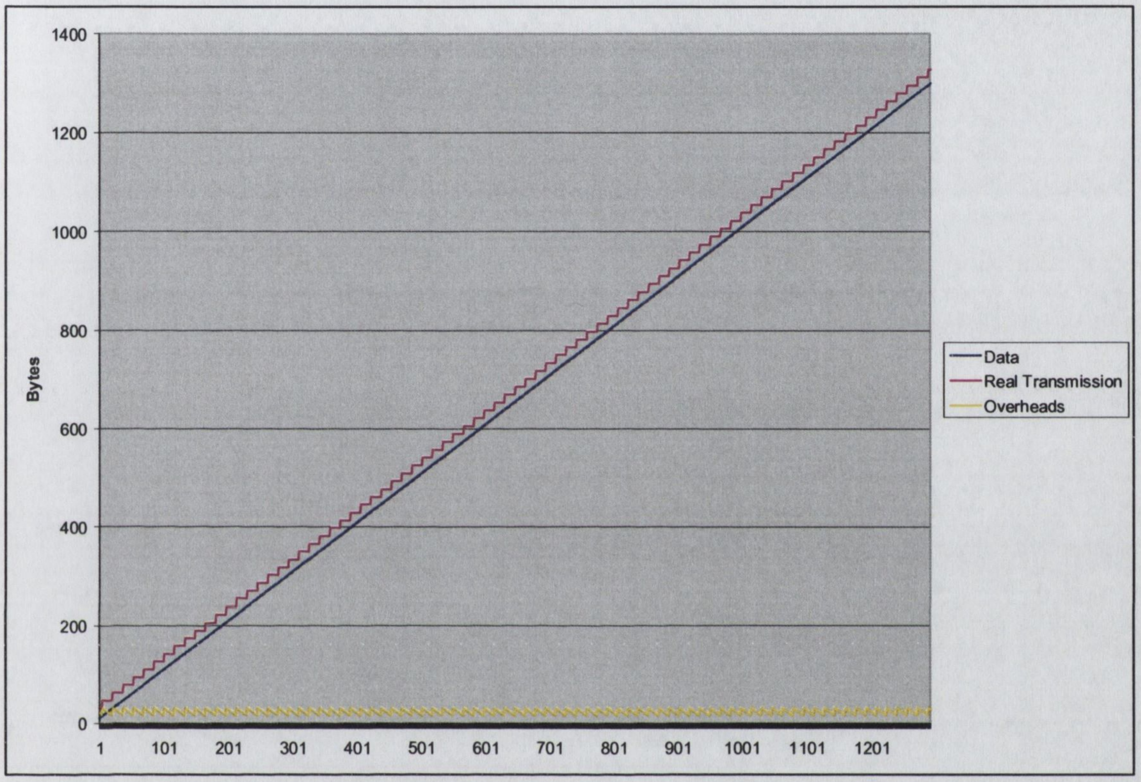


**Figure 50 : Overheads for Message Delivery**

### *4.4.2.4 Key Storage Overheads*

The "Negotiated" data structure in the NTM and RTM layer hold the keys to ensure confidential communications. The NTM layer requires the maximum of $S_{NT}$*(NL+2*KL+TS) bytes for the storage where NL = length of a node name (dependent on naming scheme), KL = key length of the mutual keys and TS = time stamp length (4 bytes on a windows system). Therefore for a 128-bit peer-to-peer shared secret using a 4 byte naming scheme, the storage requirement in the "Negotiated" data structure is $S_{NT}$*40 bytes. Similarly, the RTM layer requires a maximum of $RS_{NT}$*(NL+2*KL+TS) bytes for the storage. Therefore for a 128-bit end-to-end shared secret using a 16 byte naming scheme the storage requirement is $RS_{NT}$*52 bytes.

#### *4.4.2.5 Implementation Summary*

The prototype of the One-to-One Trust Negotiation was implemented and tested on testbed consisting of six iPAQ handheld devices with an 802.11b wireless interface. Experiments conducted on the testbed concluded that with the node density (i.e. nodes in the transmit/receive range) of six, the One-to-One trust negotiation scheme has a reasonable performance. The performance of the scheme depended on the user-changeable timer values used by the NTM and RTM components for the cryptographic protocol interchange. Optimal values of the timers where obtained for the maximum node density of six (maximum possible under the testbed). Economic restrictions prevented the testing for timer intervals on higher node densities. However, it is a trivial exercise to find timer values for higher node densities. Simulators where not used during the testing of the scheme as the focus of the thesis was to prove that the scheme could work on the commodity handheld devices. Porting the huge code base to simulators is time consuming and furthermore there is no guarantee that the simulators reliably model the wireless Ad Hoc environment.

In brief the implementation and the subsequent benchmarking tests led to the following conclusions,

- The key agreement phases of the NTM and RTM protocols have reasonable performance when implemented on real commodity handheld wireless devices. Special emphasis was placed on these phases as they are the most computationally expensive in the entire scheme
- The timer dependent key agreement phases works well in the maximum node density (i.e. six) of the testbed.
- Actual encrypted data exchange has negligible computational overheads on the real handheld devices.

## *4.5 Conclusion of One-to-One Trust Negotiation*

This chapter presented a multi-faceted trust negotiation scheme for Ad Hoc networks by progressively exchanging certificates based on the local policies of the two participating nodes. The custom embedded attribute name/value pair(s) in the certificate(s) is translated into access control permissions for the service(s) on the service providing node. This exchange of certificates is done by a new trust negotiation protocol is protected by a novel two-tier key formation framework. The One-to-One Trust Negotiation scheme satisfies the requirements for an Ad Hoc security solution deployable in an emergency scenario. These requirements tabulated below where originally enumerated in motivation section of the Introduction Chapter.

- **Minimal Pre-Configuration** – The role-free use of certificates in the scheme, lets the node function in a new area of operation with same certificates
- **Little Coordination** – No network wide coordination (except routing) among nodes is required in the operation of the scheme. Only the two participating nodes are involved in the trust negotiation process.
- **Vary the security requirements** – The use of policies lets the participating nodes vary the security requirements depending on the scenario.
- **Computationally feasible** – The prototype implementation on the scheme on the commodity handheld wireless device has a reasonable performance.

The scheme is also routing protocol independent, increasing its portability as the different routing algorithms proposed for Ad Hoc networks perform well under certain simulated network conditions [hbt+03, hbt+04].

# 5 Wanderer: Secure Group Formation

This chapter describes an extension of the one-to-one trust negotiation proposal to determine membership for a secure Ad Hoc group formed by emergency services. Such groups are usually formed when two or more nodes come together to perform some common task. For example in Figure 51, the nodes of an Ad Hoc network form groups based on the roles and the national allegiance of the nodes. The taskforce group consist of members from many countries. A small subset of the nodes forms the commander group consisting of the decision making nodes drawn from different countries. The members of the commander group are also simultaneously members of the national and the overall taskforce group. In this scenario, some individual nodes may also decide to have one-to-one interactions between themselves to aid the decision making process. Thus a node in this scenario may simultaneously receive data from several groups and individual nodes. This implies that the secure group collaboration scheme should coexist with the one-to-one interaction framework.

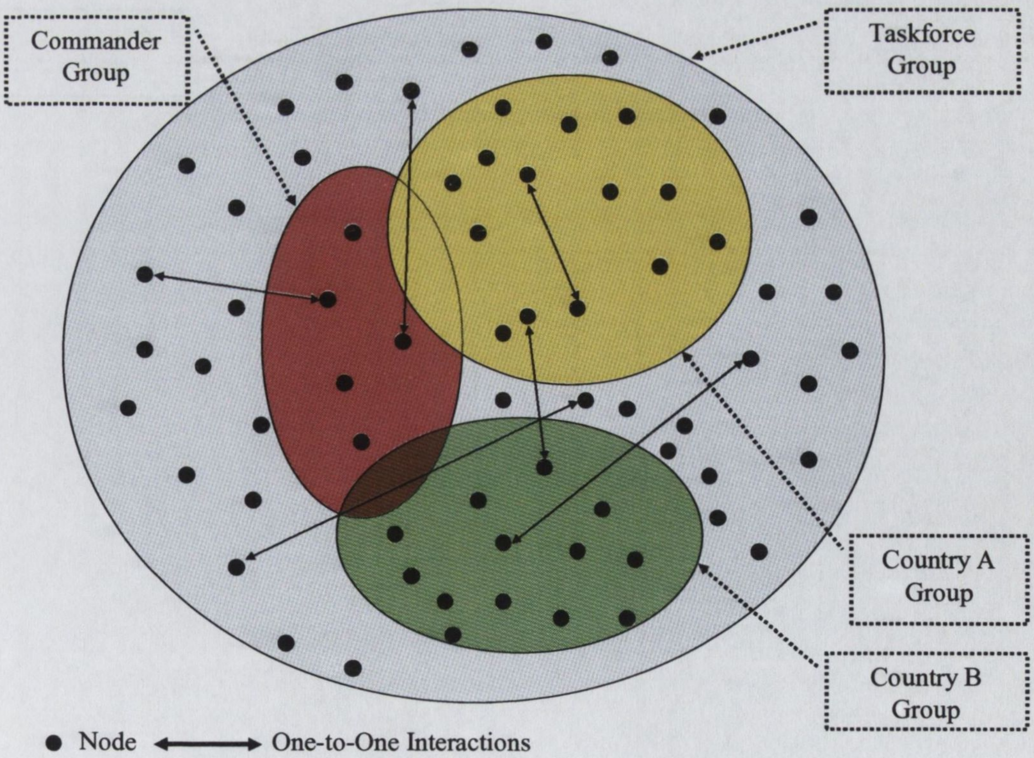
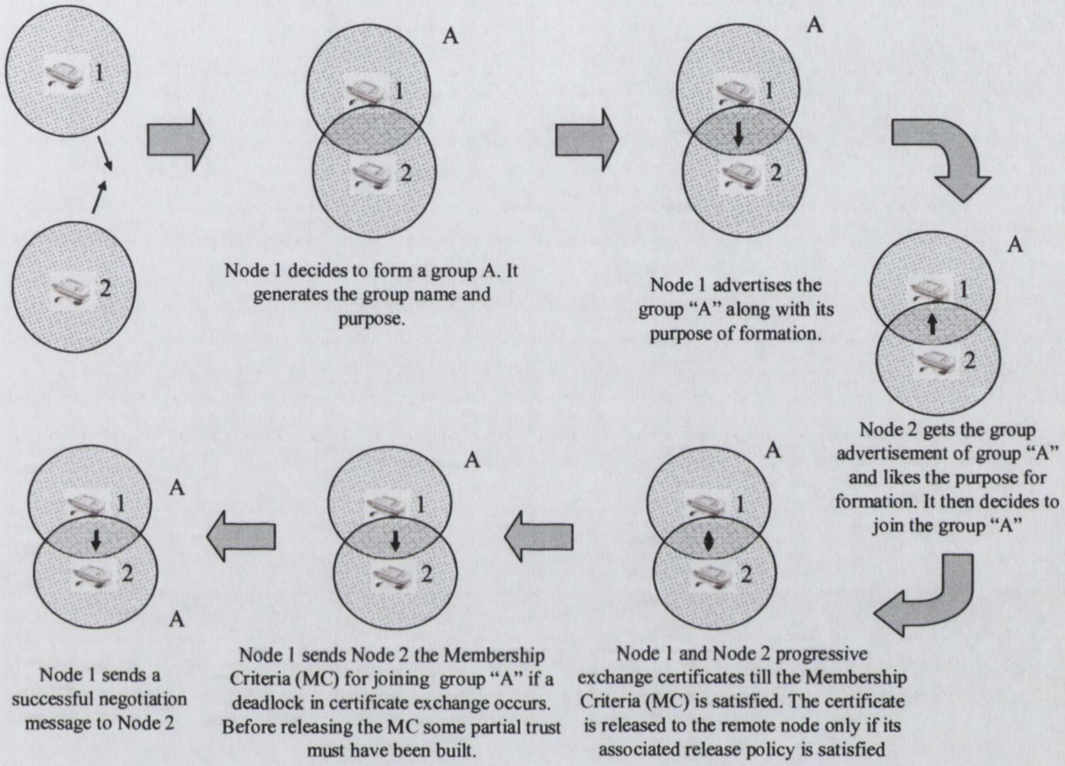


**Figure 51 : Multiple Groups and One-to-One Interactions**

In this chapter, the discussion starts with the process of a node joining a group. To illustrate the group membership process, initially consider the simplified case of two nodes coming together and then one of them deciding to form a group. This process of group formation is explained graphically in Figure 52 .



Node 1 decides to form a group A. It generates the group name and purpose.

Node 1 advertises the group "A" along with its purpose of formation.

Node 2 gets the group advertisement of group "A" and likes the purpose for formation. It then decides to join the group "A"

Node 1 and Node 2 progressive exchange certificates till the Membership Criteria (MC) is satisfied. The certificate is released to the remote node only if its associated release policy is satisfied

Node 1 sends Node 2 the Membership Criteria (MC) for joining group "A" if a deadlock in certificate exchange occurs. Before releasing the MC some partial trust must have been built.

Node 1 sends a successful negotiation message to Node 2

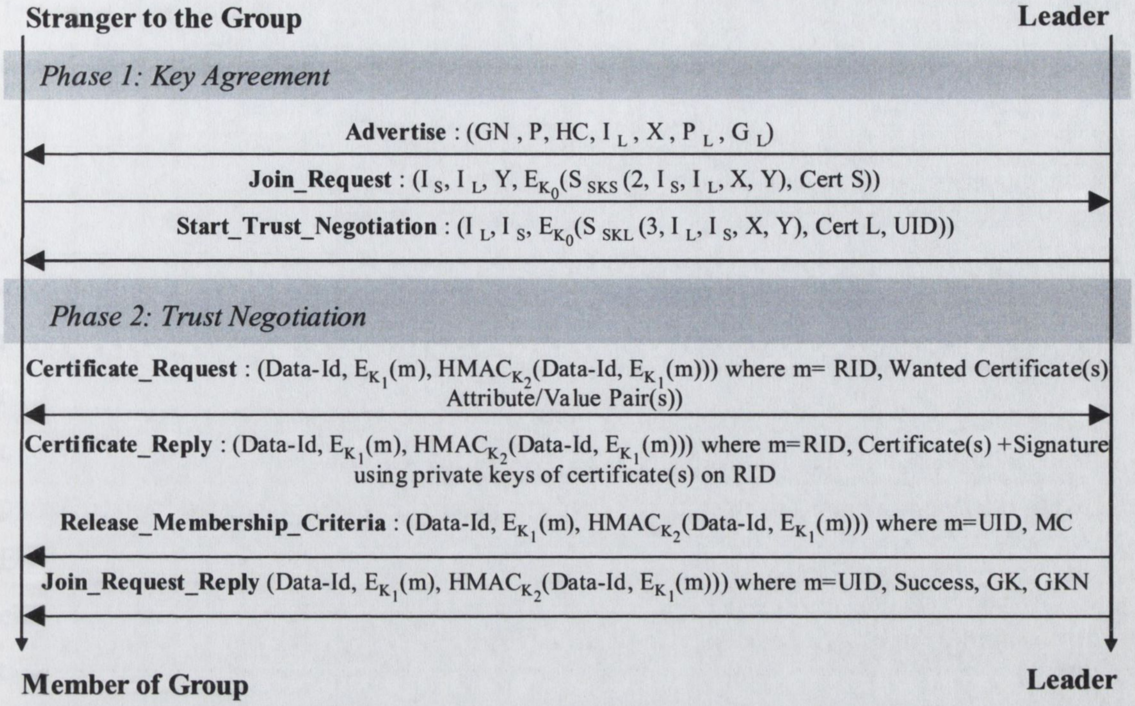
**Figure 52 : Group Membership Using Trust Negotiation**

Every group has a *unique identifier*, which is used by a node to identify a group. Moreover, the group has a *purpose*, which is the reason motivating its formation. The *unique identifier* also called the *group name* is used to differentiate the groups while, the *purpose* explains the motivation behind the group formation. Thus many groups with same *purpose* can form in an Ad Hoc network having different *group names*. The *group name* is a function of the purpose string, group creation date and time so that *group name* duplication is highly unlikely. To join a group the user of a node selects the "human readable" *purpose* and the associated *group name*. Then the underlying automatic join process uses the *group name* to initiate the group join process. It is desirable that the *purpose* of group formation be advertised to nodes in the intermediate area so that no other group is formed with the same *purpose*. In the wanderer scheme the purpose is expressed as a string like "Allied soldiers in sector D" or "Incident at J10". If the group *purpose* is a secret, this string can be a pre-arranged hash making it human-unreadable. This hash can be pre-distributed to all interested nodes through some secure channel. This *masking* of the *purpose* string ensures that the *purpose* for group formation is only known to the interested nodes.

Once a node wants to join a group by selecting the *group name* it initiates a join process with the designated group member. The admission to the group is conditional to the node satisfying the Membership Criterion (MC). This is an expression (see Appendix B) listing attribute name/value pair(s) required in valid certificate(s) to join the group. Therefore a node wanting to join the group will have to produce certificate(s) that can satisfy the MC. A typical MC may look like "((country id = A AND Issuer = Country A) OR (country id = B AND Issuer = Country B)) AND (authorized sector id = D AND issuer = coalition headquarters)". Thus a node wanting to join has to have ownership of a certificate issued by the *coalition* CA with the embedded attribute name/value pair (sector id, D). The other attributes wanted in the MC can be found in the certificates issued by *Country A* or *Country B*. If the node has ownership of certificate(s) issued by *Country A* then it should have the two embedded pairs (country id, A) and (Issuer, Country A). Otherwise if the certificates are issued by *Country B*, the embedded pairs should be (country id, B) and (Issuer, Country B) for a successful join to the group. In brief, the nodes wanting to join the group should be able to prove that it is from country *A* or *B* and have the authorization from the coalition to operate in *sector D*.

## *5.1 Group Joining Protocol*

The join process is formalised into a protocol depicted in Figure 53. This protocol is a proactive protocol with the leader looking for prospective members by periodically broadcasting an *Advertise* message. The node designated as stranger to the group replies to this message and starts the join process. The first phase of the join process is a key agreement between the two participating nodes. The resulting common secret is used to ensure the confidentiality and integrity of the second phase in which the stranger has a trust negotiation with the leader. The objective of the trust negotiation is to verify that the stranger fulfils the MC of the group. This trust negotiation is necessary as the stranger also has to be satisfied that the leader is trustworthy and is not an impostor trying to get the stranger node to disclose sensitive confidential certificates.


Stranger to the Group
Leader
*Phase 1: Key Agreement*
**Advertise** : (GN , P, HC, $I_L$, X , $P_L$, $G_L$)
**Join_Request** : ($I_S$, $I_L$, Y, $E_{K_0}(S_{SKS}(2, I_S, I_L, X, Y)$, Cert S))
**Start_Trust_Negotiation** : ($I_L$, $I_S$, $E_{K_0}(S_{SKL}(3, I_L, I_S, X, Y)$, Cert L, UID))
*Phase 2: Trust Negotiation*
**Certificate_Request** : (Data-Id, $E_{K_1}(m)$, $HMAC_{K_2}(\text{Data-Id}, E_{K_1}(m))$)) where m= RID, Wanted Certificate(s) Attribute/Value Pair(s))
**Certificate_Reply** : (Data-Id, $E_{K_1}(m)$, $HMAC_{K_2}(\text{Data-Id}, E_{K_1}(m))$)) where m=RID, Certificate(s) +Signature using private keys of certificate(s) on RID
**Release_Membership_Criteria** : (Data-Id, $E_{K_1}(m)$, $HMAC_{K_2}(\text{Data-Id}, E_{K_1}(m))$)) where m=UID, MC
**Join_Request_Reply** (Data-Id, $E_{K_1}(m)$, $HMAC_{K_2}(\text{Data-Id}, E_{K_1}(m))$)) where m=UID, Success, GK, GKN
Member of Group
Leader

Notation Used:

$P_L$, $G_L$ – Diffie –Hellman parameters used by the leader and embedded into the identity certificate Cert L used by the leader.

$X = (G_L)^x \bmod P_L$ where small "x" is a fresh random number generated by the leader.

$Y = (G_L)^y \bmod P_L$ where small "x" is a fresh random number generated by the stranger.

HC – Hop Count to prevent endless broadcast of the *Advertise* message.

GN – Group name to uniquely identify the group

P – Purpose for the group formation

$I_L$, $I_S$ – The identity used by the leader and the stranger in the trust negotiation interchange and embedded in the respective identity certificates

SKL and SKS– Private keys of the certificates Cert L and Cert S respectively

$E_{Key}(M)$ = Encryption of M using the symmetric key "Key".

$S_{Key}(M)$ = Signing of M using the asymmetric secret key "Key".

$HMAC_{Key}(M)$ = Keyed-Hashing for Message Authentication applied to M using the key "Key" to ensure data integrity.

K – Shared Secret agreed between the leader and stranger

$K = (G_L)^{xy} \bmod P_L$

$K_0 = HMAC_K(X,Y,1)$ i.e. key used in the phase 1 key agreement protocol.

$K_1 = HMAC_K(X,Y,2)$ i.e. Encryption key used in phase 2 of the join protocol.

$K_2 = HMAC_K(X,Y,3)$ i.e. Data integrity key used in phase 2 of the join protocol.

GK – Group wide common key generated periodically by the leader.

GKN – Sequential number identifying the group key.

Data-id – Unique sequence number identifying each data transmission.

Shared Information between the Initiator and the Responder:

Encryption / Signature / Certificate / Hashing algorithms and Key Sizes

**Figure 53 : Group Joining Protocol**

The key agreement phase of the group join protocol starts when an interested node receives an *Advertise* message from a group leader. This message contains a *hop count* which is decremented by one each time a node forwards the message. The use of hop count prevents endless flooding. If any node likes the purpose of the group it responds with a *Join_Request* message. This acceptance can be either automated or manual. In the automated response, the *Join_Request* message is only sent in response to an *Advertise* message whose purpose or group name strings are preconfigured into the node. On the receipt of the *Join_Request* message, the leader sends back the *Start_Trust_Negotiation* message. This message also contains a unique identifier (UID) to identify the negotiation process. The key $K_1$ derived from the shared secret $K$ also serves as the Traffic Encryption Key (TEK) for communication between the two nodes. This TEK is used to encrypt all messages (excluding the group messages) between the two nodes. Similarly the key $K_2$ also derived from $K$ serves as the Data Integrity Key (DIK). The identity authentication is done as in the one-to-one scheme using certificates with the appropriate embedded parameters (i.e. *Identity Certificate* of the RTM layer). This key agreement phase has the same cryptographic properties, data structures and state diagram as the NTM protocol's key agreement phase.

Phase 2 of the group join protocol is a trust negotiation involving confidential progressive exchange of certificates governed by the local policies of the participating nodes. This interchange in the group collaboration follows the same procedure as in the RTM phase 2 trust negotiation. One of the major changes is that the progressive exchange of certificates between the leader and stranger solely depends on the local certificate release polices and there is no role for the service policy. This use of certificate release policies provides protection to the sensitive confidential certificates of both the involved nodes. The leader or the stranger does not need to continue the join process if it finds the certificate from the remote node untrustworthy. The other major difference from the one-to-one schemes is that the process of trust negotiation is carried out till the stranger satisfies the MC (not the service policy). The stranger cannot ask for the MC but it is released by the leader unilaterally using the *Release_Membership_Criteria* message. This is done so as to eliminate the possibility of accidental disclosure of a sensitive and confidential MC. The leader can release the MC manually or automatically to resolve a deadlock in certificate exchange. The automatic disclosure of the MC depends on the disclosure policy of the MC (similar to the service policy of the one-to-one scheme). The disclosure of the MC gives the stranger the option to check if the leader it is joining itself satisfies the MC. Such a check can be performed during or after the execution of the trust negotiation phase of the join protocol.

The join process ends with a *Join_Request_Reply* message which tells the stranger if the join process has been a success or failure. In Figure 53 the protocol runs shows the case when the join process is a success. The *success* carrying *Join_Request_Reply* message contains the group key and associated group key number. This group key is used by members to encrypt/decrypt group messages. If the join process is a failure the *Join_Request_Reply* message will not contain any other information except the *failure* notification. The join protocol is modelled on the lines of the IETF sponsored Group Secure Association Key Management Protocol (GSAKMP) [hch+00]. The phase 2 of the join process requires a compliance checker to govern the progressive exchange of the certificates between the leader and stranger. This compliance checker for the leader with its inputs and outputs is shown in Figure 54. The checker is similar to one used in the RTM with a few modifications.

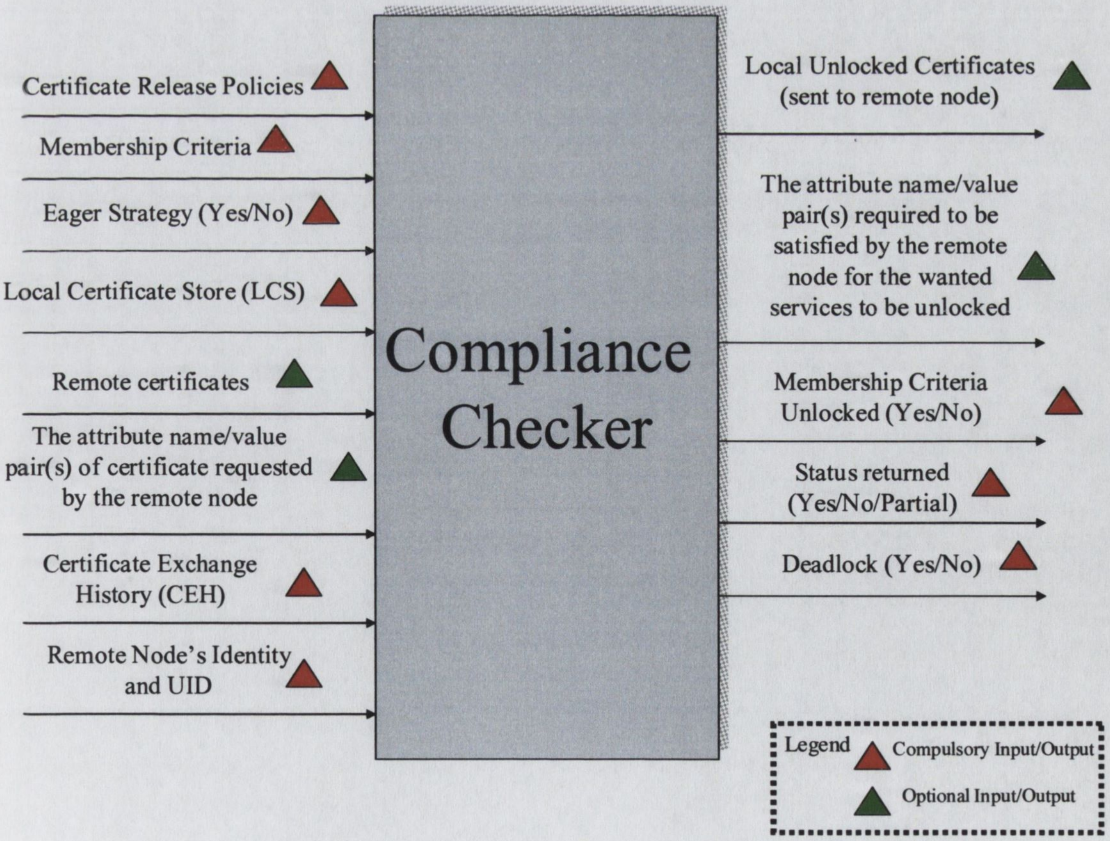
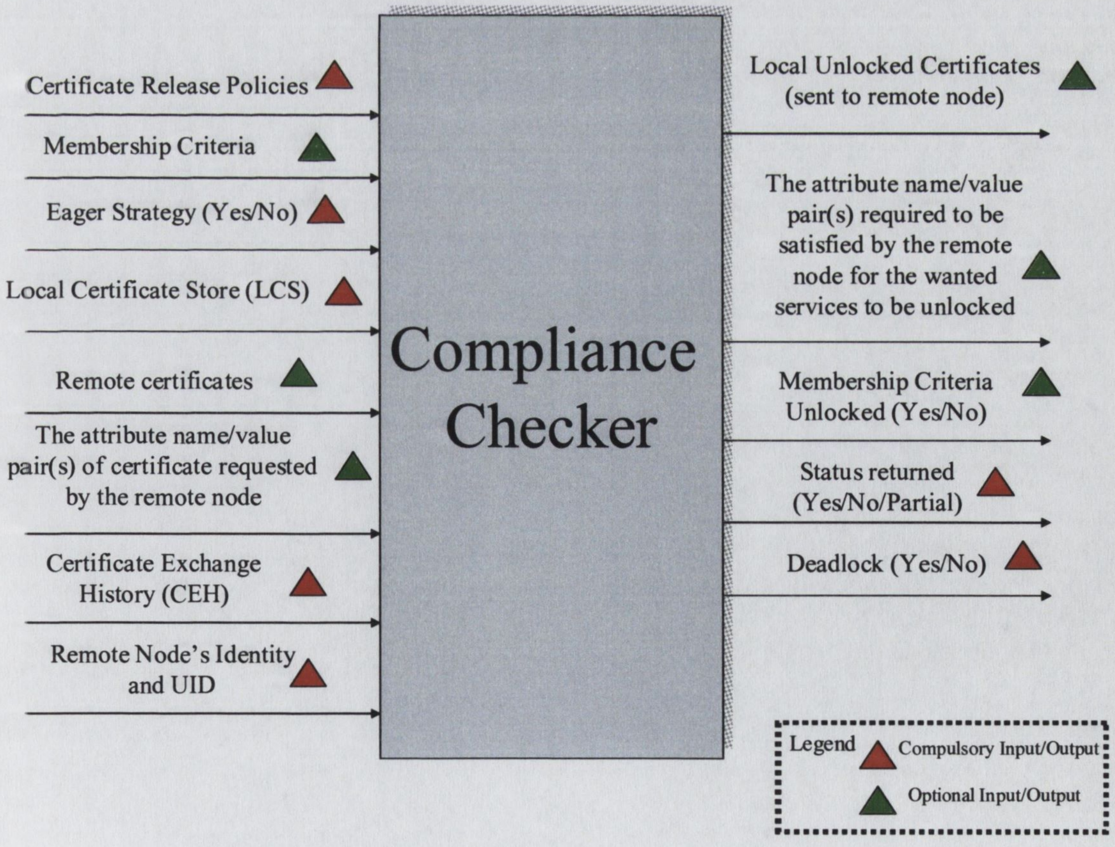

Certificate Release Policies
Membership Criteria
Eager Strategy (Yes/No)
Local Certificate Store (LCS)
Remote certificates
The attribute name/value pair(s) of certificate requested by the remote node
Certificate Exchange History (CEH)
Remote Node's Identity and UID
Compliance Checker
Local Unlocked Certificates (sent to remote node)
The attribute name/value pair(s) required to be satisfied by the remote node for the wanted services to be unlocked
Membership Criteria Unlocked (Yes/No)
Status returned (Yes/No/Partial)
Deadlock (Yes/No)
Legend
Compulsory Input/Output
Optional Input/Output

**Figure 54 : Compliance Checker for Wanderer (Leader)**

The first modification is that the *services wanted* and *service policy* inputs are discarded and the *MC* input introduced. The output for the *services unlocked* is not required in the compliance checker for the wanderer. It is replaced by the output telling if the MC is unlocked to be released to the remote node. This happens if the disclosure policy of the MC is satisfied. The interpretation of the output status by the node in the wanderer scheme is different compared to the one-to-one framework. A yes status implies that the stranger has by producing certificates satisfied the MC. Thus, the stranger is eligible to join the group. A no status implies that not even one attribute name/value pair of the MC has been satisfied by the valid remote certificates. The partial status reply implies that some of the attribute name/value pairs in the MC have been satisfied by the remote certificates. This implies more certificate exchange is required to satisfy the MC. This compliance checker is slightly different for the stranger node (Figure 55). In this case the input for MC and the corresponding output are optional. However on receipt of the MC the stranger can check if the leader also satisfies the MC. This enables the stranger to reverse check the trustworthiness of the leader.



Certificate Release Policies
Membership Criteria
Eager Strategy (Yes/No)
Local Certificate Store (LCS)
Remote certificates
The attribute name/value pair(s) of certificate requested by the remote node
Certificate Exchange History (CEH)
Remote Node's Identity and UID
Compliance Checker
Local Unlocked Certificates (sent to remote node)
The attribute name/value pair(s) required to be satisfied by the remote node for the wanted services to be unlocked
Membership Criteria Unlocked (Yes/No)
Status returned (Yes/No/Partial)
Deadlock (Yes/No)
Legend
Compulsory Input/Output
Optional Input/Output

**Figure 55 : Compliance Checker for Wanderer (Stranger)**

In case the deadlock output returns a yes, then the leader has two options. Either the leader automatically releases the MC if its disclosure policy has been met by the stranger or else manually releases it. Once a stranger satisfies the MC it joins the group with a star structure. The centre of the star structure is the leader. Such a structure has limited scalability as the leader has the capability of handling only a certain amount of members.

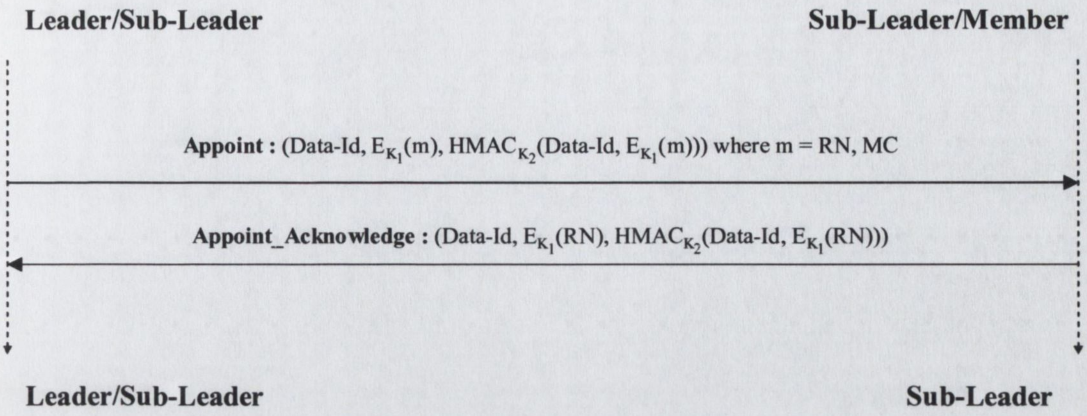
## *5.2 Flexible Group Structure*

A group with a single leader and a large number of members in a star structure is not scalable. Therefore to make the group scalable, the group leader can appoint sub-leader(s) out of the members attached to it. These sub-leader nodes have all the powers of the leader except that of issuing the group key. Moreover, the member being appointed the sub-leader cannot refuse the appointment. The properties of the three types of the nodes in the wanderer scheme are tabulated below:

| **Powers/Properties of different types of members of the group** | **Leader** | **Sub-leader** | **Ordinary Member** |
|---|---|---|---|
| **Group Key Generation** | Yes | No | No |
| **Appoint New Sub-leaders** | Yes | Yes | No |
| **Appoint New Members** | Yes | Yes | No |
| **Procession of the MC** | Yes | Yes | Maybe |
| **Send/Receive Group Messages** | Yes | Yes | Yes |

**Figure 56 : Different Types of Nodes in Wanderer**

A sub-leader can appoint further sub-leaders giving the group a tree hierarchy. This appointment is done using the protocol interchange illustrated in Figure 57. It is also important to note that during the sub-leader appointment, the MC is also released to the new sub-leader. However, determining the selection criteria for the sub-leader appointment is difficult in an Ad Hoc network. A sub-leader appointment criterion which looks good at a point of time may not be a good choice after some time due to topological and membership changes. In the wanderer scheme for sake of consistency, the first member to be attached to a leader/sub-leader is made the sub-leader first. Then the second member is appointed sub-leader and so on.


Leader/Sub-Leader
Sub-Leader/Member
Appoint : (Data-Id, $E_{K_1}$(m), $HMAC_{K_2}$(Data-Id, $E_{K_1}$(m))) where m = RN, MC
Appoint_Acknowledge : (Data-Id, $E_{K_1}$(RN), $HMAC_{K_2}$(Data-Id, $E_{K_1}$(RN)))
Leader/Sub-Leader
Sub-Leader

MC = Membership Criteria for the group

RN = Generated by the sender to ensure randomness in message and also ensure that the acknowledgement is from the right node.

$K_1$ = Traffic Encryption Key agreed between the two participating nodes.

$K_2$ = Data Integrity Key agreed between the two participating nodes.

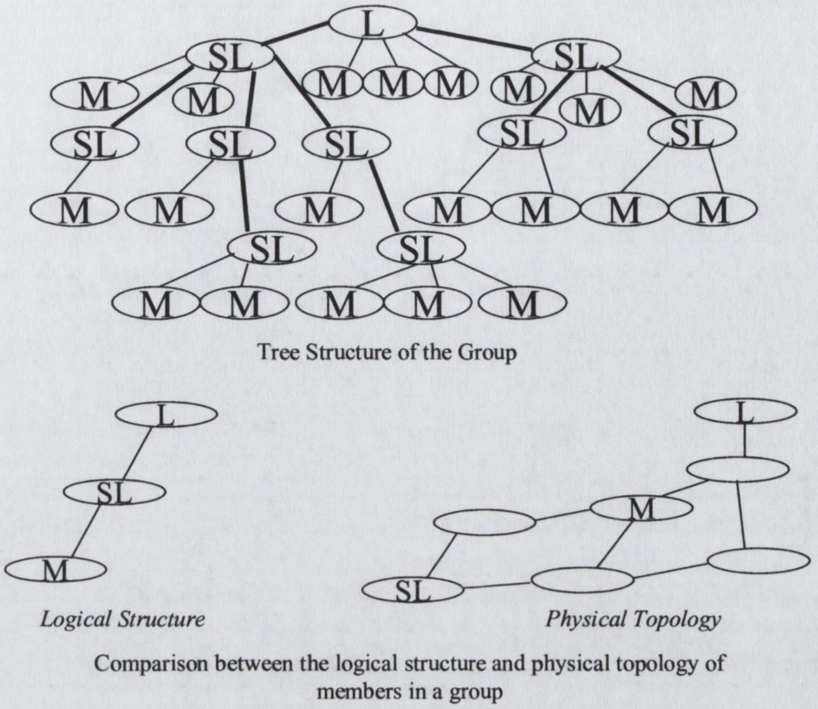
Data-Id – Unique sequence number identifying each data transmission

HMAC $_{Key}$ (M) - Keyed-Hashing for Message Authentication on the plaintext M using the key "key".

**Figure 57 : Sub-Leader Appointment Protocol**

The tree hierarchy resulting from a sub-leader appointing further sub-leaders is shown in Figure 58. The label "L" denotes the leader, "SL" denotes sub-leader and "M" the ordinary members. Further it is not a requirement that the group hierarchy be reflected in the physical topology of the network. As illustrated in Figure 58 the hierarchy may be L→SL→M but in the physical topology L may be closer to M. It is the underlying routing algorithm that maps the logical group structure to the physical topology. This allows the physical topology to change without any effect on the group structure, a situation analogous to overlay networks [afb+02]. Thus re-keying of the group is not required if the topology changes.

To increase the scalability, each participating node of the group only keeps track of its parent and children in hierarchy. This implies a participating node only has a localised view of the group. Such a localized view is desirable as it is it is difficult for any node to keep track of the membership of a large and dynamic group in an Ad Hoc network. Therefore the distributed tree group structure is easy to maintain by the individual nodes participating in the secure group collaboration.



Tree Structure of the Group

*Logical Structure*

*Physical Topology*

Comparison between the logical structure and physical topology of members in a group

**Figure 58 : Logical and Physical Structure of a Group**

The nodes which are not members of the group in the Ad Hoc network transparently relay the group messages. Figure 59 shows the scenario in which a node *X* wants to join a group *A*. It can be seen that the join request is routed through a node that is not a member of the group. These intermediate relaying nodes can only garner the *group name* and *purpose* from the join process as the rest is encrypted.

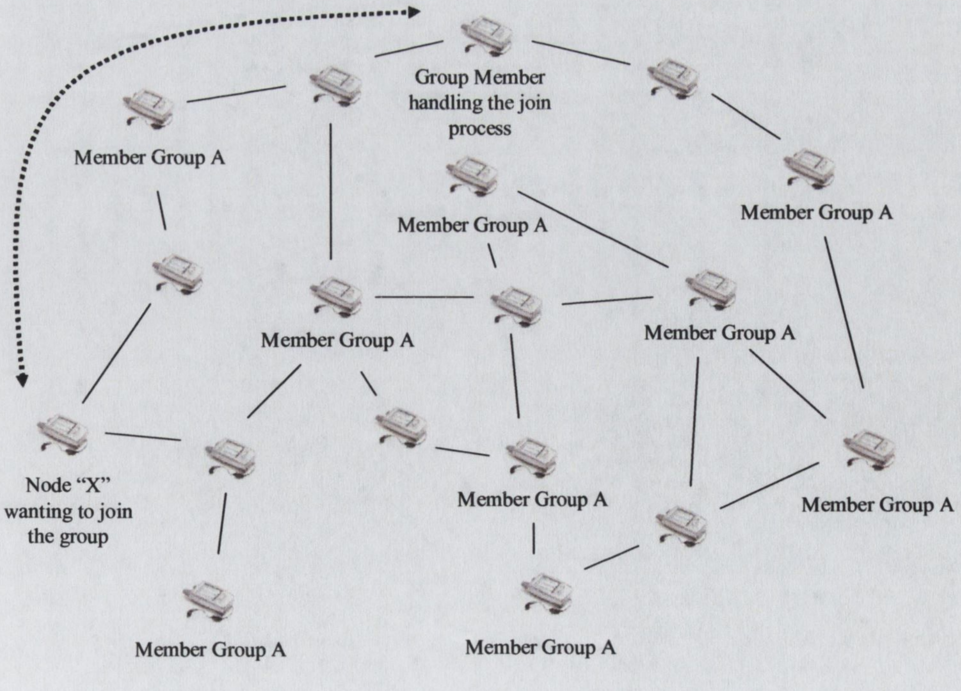


**Figure 59 : Non-member Relaying the Join Request**

To cater for a high mobility scenario a node may join at more than one place on the group tree hierarchy. This *multi-join* benefits the group structure and the node having multiple parents. The group structure is more robust as there are redundant links in the hierarchy. These redundant links are useful in maintaining cohesion of the group structure, if the nodes have high mobility or the network is experiencing localized congestion. As the group key update (explained in the next section) follows a top to down propagation pattern in the group tree, the node practicing the *multi-join* has more than one parent and consequently has a higher probability of the group key delivery. This *multi-join* decision is left to the individual nodes.

## *5.3 Group Key Propagation*

The group key is generated periodically by the leader of the group. This ensures that an external attacker will not gain access to all group messages by compromising one group key. Each group key has a sequence number (*group key number*) attached to it which tells which *group key* was used, simplifying the decryption. This also helps each member in the group organize a cache of group keys which allows them to decrypt a group message which uses an old group key. In Ad Hoc networks, the ordering of group message delivery is not guaranteed due to topology changes making the caching of the group keys a necessity. However, the cache size has to be kept small to prevent sending of fake group messages using old compromised group keys. If a received group message is encrypted by a group key not received yet, such a group message is kept in a message queue. As soon as the appropriate group key is received, the group messages in the queue can be retrieved and decoded. This feature of the use of caches is good for upper-level protocols in which the order of message delivery is not important. However, in cases of real-time traffic like audio and video streams, the caching of group message is not recommended.

The leader generates the group key periodically and sends it to all its child nodes in the hierarchy. However the group keys are not sent in plaintext but encrypted with the TEK's agreed with the children nodes. Other sub-leaders then pass the group key down the hierarchy similarly (Figure 60). Therefore the group key propagates top-to-bottom in the group tree hierarchy periodically.

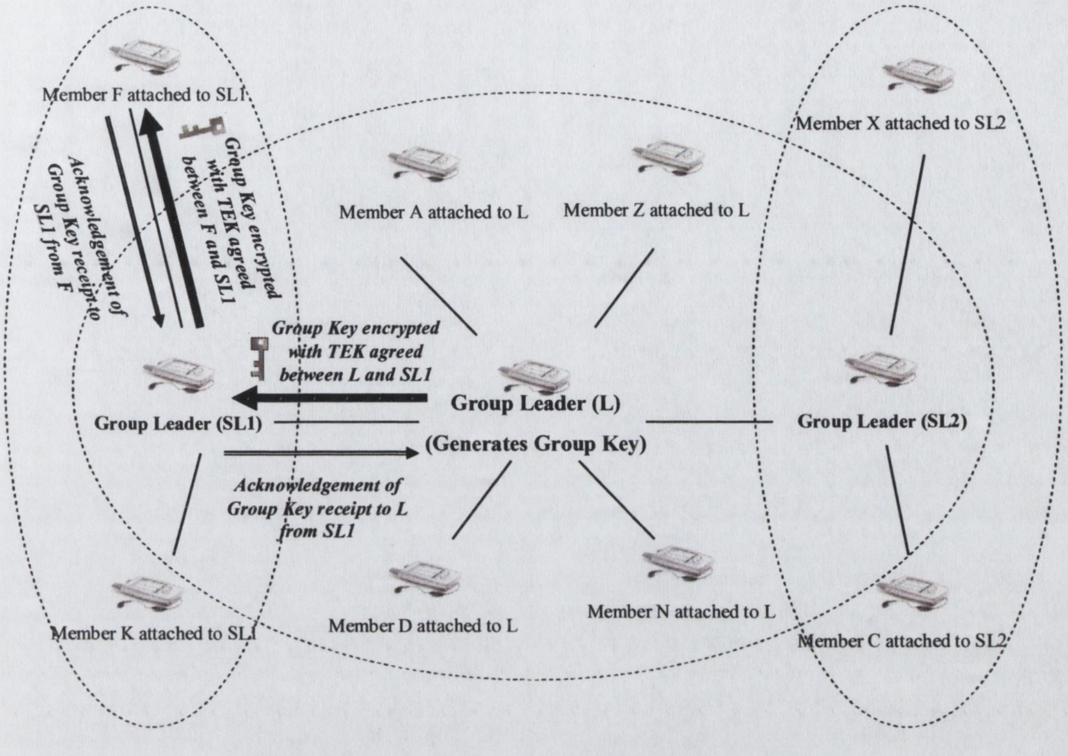
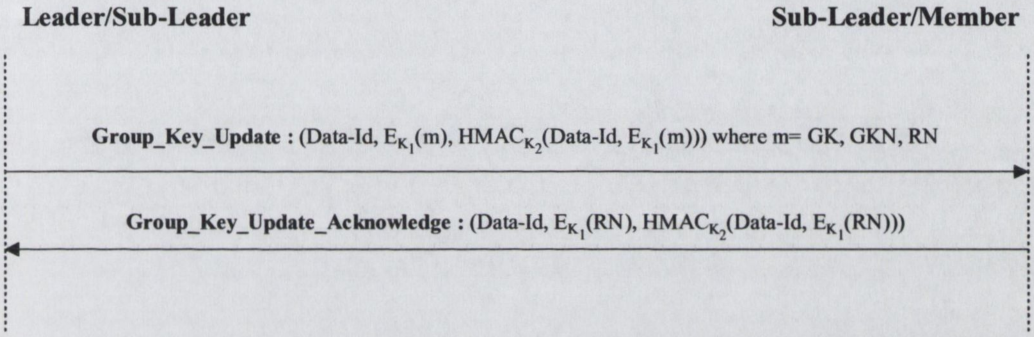


**Figure 60 : Group Key Propagation Using TEK's**

The method of group key propagation (Figure 60) may not be the most efficient way. However, this method ensures the propagation of group keys in a secure manner without sacrificing the scalability. It can be argued that the group key is not required if the messages can be propagated in the same method as the group key. But later in this chapter the tree-based group membership structure evolves into a mesh structure for group message propagation. This in line with the survey in Chapter 3 which concluded that in a large and mobile Ad Hoc group, mesh-based group propagation is the best method for ensuring robust and fault-tolerant group message propagation. The group key number will aid in this conversion.

The protocol interchange to perform the group key update is presented in Figure 61. This interchange has an acknowledgement mechanism for the group key update which keeps a node updated about its children node status. If a child node does not acknowledge several consecutive group key updates it is considered to have left the group. The sub-leader/leader does not send any more group key update to such child nodes. However, a member that left the group voluntarily/involuntarily in such a fashion can understand group traffic until the group key is updated. This is a necessary shortcoming as it is difficult for a central entity (i.e. leader) in a large and mobile Ad Hoc group to keep track of all the participating nodes. It must be also considered that the group node that left voluntarily was trusted some time back. In the case a node is forced to leave the group due to misbehaviour, it can understand some group messages until the group key is updated. It can be argued that in cases of forced leaves, the confidentiality of the group messages can be maintained by discarding the group wide key and instead use the individual TEK's to propagate the group messages. But a tree structure is not robust and scalable to propagate frequent group messages if the group is large and mobile. Thus, the tree structure is not used to propagate the group messages.


Leader/Sub-Leader
Sub-Leader/Member
Group_Key_Update : (Data-Id, $E_{K_1}$(m), $HMAC_{K_2}$(Data-Id, $E_{K_1}$(m))) where m= GK, GKN, RN
Group_Key_Update_Acknowledge : (Data-Id, $E_{K_1}$(RN), $HMAC_{K_2}$(Data-Id, $E_{K_1}$(RN)))

GK = Group Key Generated periodically by the Leader

GKN = Key number in sequence associated with each group key. Also generated by Leader.

RN = Generated by the sender to ensure randomness in message and also ensure that the acknowledgement is from the right node.

$K_1$ = Traffic Encryption Key agreed between the two participating nodes.

$K_2$ = Data Integrity Key agreed between the two participating nodes.

Data-Id – Unique sequence number identifying each data transmission

HMAC $_{Key}$ (M) - Keyed-Hashing for Message Authentication on the plaintext M using the key "key".

**Figure 61 : Group Key Update Protocol**

## *5.4 Group Message Propagation*

This section presents a method which converts the tree group structure into a robust mesh-based structure for secure group message propagation. In Ad Hoc networks the mesh structure is best for message propagation as it provides redundant links for communication. A participating node that wants to transmit data will send a group message to its parent and children nodes in the tree hierarchy. Those nodes receiving the message propagate the group message in the same way. However, it may happen that a participating node of the group and overhears a group message which is not destined for it yet. In the wanderer scheme, this node will propagate this overheard group message to its parent and children nodes. This lets the distributed tree group structure to evolve into a robust and fault-tolerant mesh-structure for group message propagation. This method is illustrated in Figure 62. To illustrate the effectiveness of mesh structure over tree structure for message passing lets assume that the group structure is a fully balanced binary tree with *n* nodes. The physical topology is same as the logical group tree structure. A group message propagation using the tree can take maximum of $2log_2n$ hops. Moreover, there can be network congestion at the leader of a large or mobile group as all group messages pass through it. However, if in physical topology there is a network between members then using the method outlined in Figure 62, the group message propagation can take at most $log_2n$ hops. Therefore mesh structure through redundant link halves the hop count in the theoretical case. The redundant links also help in decreasing the load on the leader making it less susceptible to congestion.

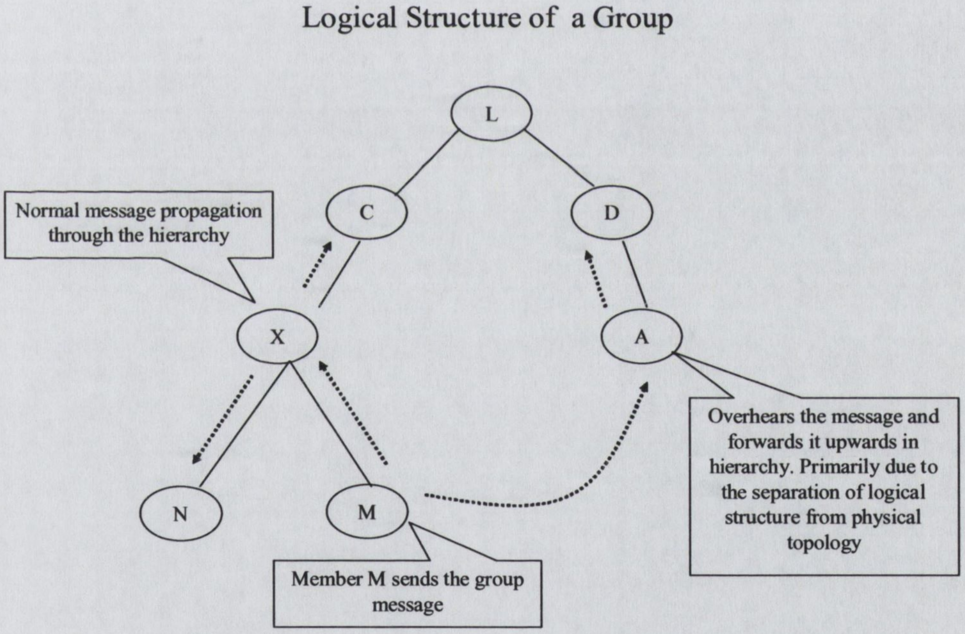
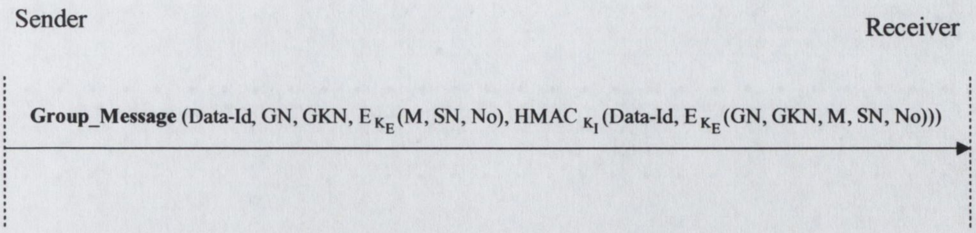

Logical Structure of a Group
L
C
D
X
A
N
M
Normal message propagation through the hierarchy
Overhears the message and forwards it upwards in hierarchy. Primarily due to the separation of logical structure from physical topology
Member M sends the group message

**Figure 62 : Group Message Propagation Using Mesh Topology**

This conversion of the tree structure into mesh based structure can be done if there is some identifier on the group message for the participating nodes to recognize. The best identifier known to all participating nodes is the *group name*. However, the possibility of delay in the group key propagation can mean that node may not have the current group key to decrypt group messages. Therefore the *group name* and *group key number* have to be in plaintext for ease of identification and decoding of the group message by a participating node. This group messaging protocol is shown graphically in Figure 63. The use of the HMAC in the group message prevents tampering with the group message by an attacker.


Sender
Receiver
**Group_Message** (Data-Id, GN, GKN, $E_{K_E}$(M, SN, No), HMAC $_{K_I}$(Data-Id, $E_{K_E}$(GN, GKN, M, SN, No)))

M – Group message sent by a participating node of the group.

GN – Unique group identifier.

GKN – Group key number of the group key used to encrypt the message.

SN – Identifier of the node sending the group message.

No - A sequence number attached to each message by the node sending the group message.

Data-Id – Unique sequence number for each "Data" transmission.

Group Key – Group key used to encrypt the message.

$K_E$ – HMAC $_{\text{Group Key}}$(GKN,0)

$K_I$ – HMAC $_{\text{Group Key}}$(GKN,1)

E $_{\text{Key}}$ (M) – Encryption of M using the symmetric key "Key".

HMAC $_{\text{Key}}$ (M) - Keyed-Hashing for Message Authentication on the plaintext M using the key "Key".

**Figure 63 : Group Messaging Protocol**

## *5.5 Group Maintenance and Reorganization*

The dynamic nature of the Ad Hoc networks implies that there can be sudden group membership and topological changes. Some of the membership changes can be voluntary while others can involuntary due to topological changes. There can also be an involuntary leave enforced from the group due to misbehaviour. Despite the changes in topology and membership the group structure should survive. In this section the various scenarios in which there can be changes to the membership and the group structure are addressed.

### 5.5.1 Voluntary Group Leave

A voluntary group leave can be performed by three types of participating nodes in a group: namely a leader, a sub-leader and a member. If the participating node leaving the group is a sub-leader it informs its parent in the hierarchy about the intended leave and provides the parent with the list of its children in the hierarchy. This message also contains the TEK/DIK keys the sub-leader negotiated with its children nodes. The parent node then sends the sub-leader that is leaving an acknowledgement. On receipt of the acknowledgement, the sub-leader leaving informs its children nodes in the hierarchy that it is leaving. This message also contains the identity of the parent of the sub-leader leaving. Then the sub-leader leaves and its former parent and children nodes build direct communications between them (Figure 64). If there is a transmission failure during this process the group becomes partitioned. The nodes in the tree below the sub-leader lose the group membership and such nodes will have to renegotiate to join the group. To decrease the probability of such situations each of the messages sent in the leave process by the sub-leader is sent repeatedly (some fixed number of times) till acknowledgements are received.

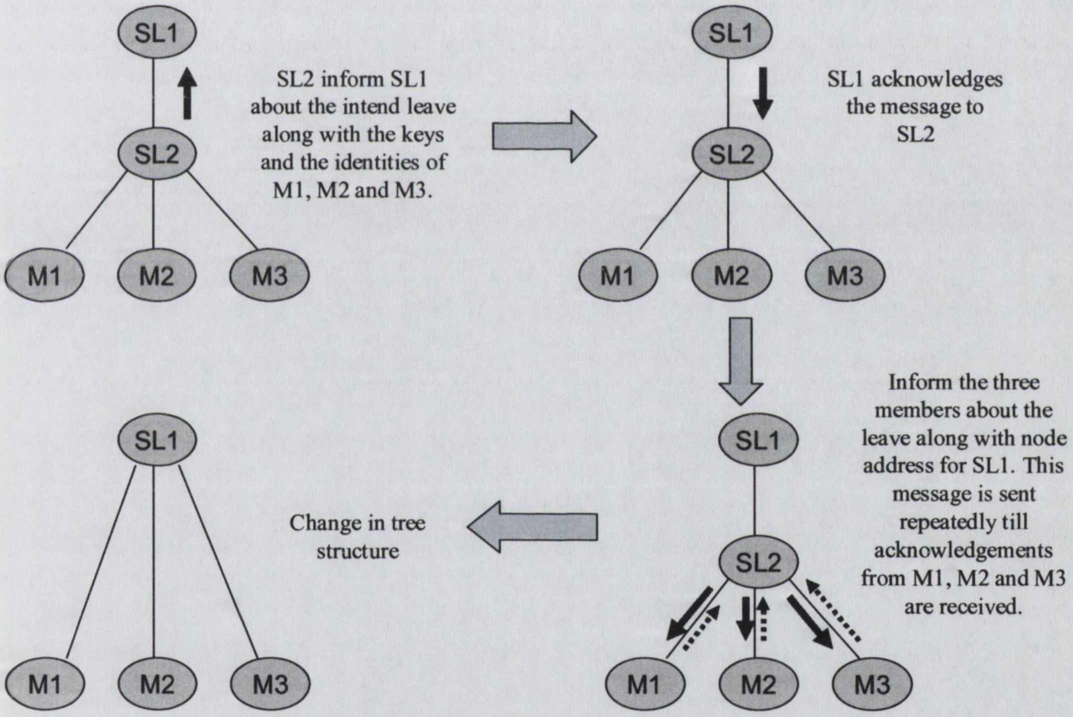
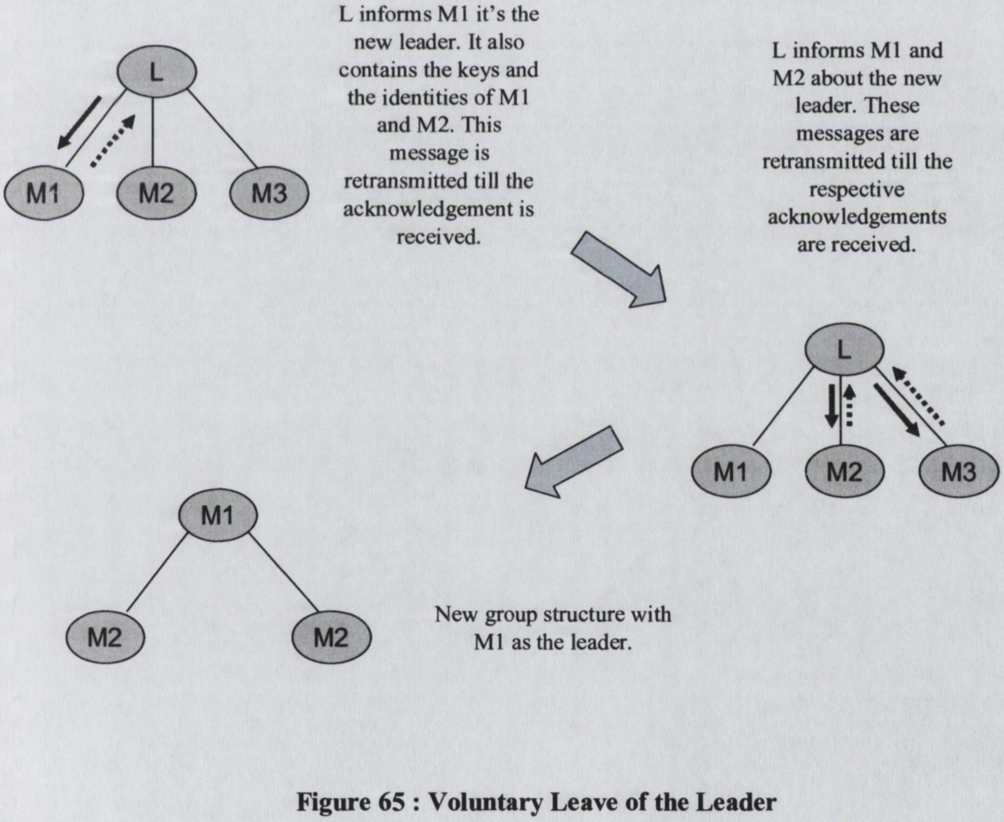

SL1
SL2
M1
M2
M3
SL2 inform SL1 about the intend leave along with the keys and the identities of M1, M2 and M3.
SL1 acknowledges the message to SL2
Inform the three members about the leave along with node address for SL1. This message is sent repeatedly till acknowledgements from M1, M2 and M3 are received.
Change in tree structure

**Figure 64 : Voluntary Leave of a Sub-Leader**

An ordinary member leaves the group by not acknowledging the group key update. The leader/sub-leader it is attached to automatically removes it from its list of children after a few group key updates are not acknowledged. In case, the leader of the group wants to leave it has two choices. Firstly, the leader can disband the group by sending a special DISBAND message down the hierarchy using the individual TEK's in the same fashion as the group key update. The use of DISBAND approach is recommended in the case the MC has very strict requirements for group membership. In case the MC is not strict this approach towards group disbanding should not be used as any malicious sub-leader can cause partial disbanding of group (the nodes under the malicious sub-leader in the tree structure). Therefore in case of lax MC, the second approach is used. In this approach, the leader can appoint one of the directly attached sub-leader as the new leader. The algorithm in the second choice followed is similar to the one for a sub-leader leaving the group and is depicted in Figure 65. In case there is some communication failure during the process the group will disband. It is important to note that all the messages in the actual protocols of the process in the Figure 64 and Figure 65 use TEK's and DIK's to ensure confidentiality and integrity by using the encrypt-then-authenticate method [k01]. Thus a malicious attacker from outside the group cannot initiate a false voluntary leave of the leader or a sub-leader.


L informs M1 it's the new leader. It also contains the keys and the identities of M1 and M2. This message is retransmitted till the acknowledgement is received.

L informs M1 and M2 about the new leader. These messages are retransmitted till the respective acknowledgements are received.

New group structure with M1 as the leader.

**Figure 65 : Voluntary Leave of the Leader**

### 5.5.2 Involuntary Group Leave

The involuntary leaving of a node from a group can be for two reasons. Firstly there is no route from the node to the group for some time. Such leaves are detected by the parent if a node does not acknowledge some group key updates. The threshold for number of acknowledgements not received for an automatic leave should be set taking into account the network congestion. If the node leaving involuntary in this fashion is a sub-leader, the nodes below it in tree hierarchy will have to rejoin the group. In case the group leader does not have a route to rest of the group, then the group key is not updated. This increases the possibility of group key compromise. A group can operate in this state for some time till the participating nodes notice that there is no periodic group key update. In such cases it is left to individual nodes if they want to leave the group.

The second reason for involuntary leave is that the node was misbehaving. The techniques used to detect misbehaviour of a node are beyond the scope of this thesis. There can be three different types of forced leaves:

- **Member Forced Leave** – The leader sends down a MISBEHAVIOUR message down the tree with the misbehaving member identity using the TEK's (same way as the group key update). This makes the leader/sub-leader(s) aware to which the member (to be expelled) is attached not to give the concerned member any new group key update. This expels the member from the group. However, the expelled member can still understand the group messages encrypted with older group keys.
- **Sub-leader Forced Leave** – The same procedure as a member leave is done. This leaves to partial disbanding of group. The hierarchy below the concerned sub-leader do not receive any group key updates. It is easy for a node in the disbanded hierarchy to detect such a case as it cannot decrypt any new group messages. It can rejoin the group at some other point in the hierarchy. To partially mitigate such cases use of *multi-join* procedure by a node is recommended.
- **Leader Forced Leave** – A leader cannot be forced out in a distributed tree structure. This is because of the way the tree is created with each node only having a localized view of the group. Therefore if a node looses confidence in a group it just leaves.

The member and sub-leader forced leaves can be exploited by a sub-leader to expel genuine group nodes. To prevent such nodes from joining the group in the first place, the group joining membership criteria (MC) should be strict.

### 5.5.3 Partitioning and Re-merger

One of the likely scenarios in Ad Hoc networks is that the group may become partitioned. If the network reforms before the next group key update, the group remains intact. In case the network is not able to reform in time then the intact hierarchy containing the leader is the group. The rest of the members will have to rejoin the group when they come in range again. It was decided against using complex tree restructuring algorithms in the wanderer scheme since in a large Ad Hoc networks accurately determining the topology and membership is difficult. Moreover, the join process of the group scheme is fast and consequently having no complex group reformation algorithm reduces the complexity of the wanderer scheme.

### 5.5.4 Summary of the Wanderer Scheme

The wanderer secure group collaboration balances the requirements of scalability, membership structure, robustness of group message propagation and security aspects. To make the group scalable the nodes in the network are organized into a distributed tree. This distributed tree group structure is able to cope with the frequent topological and membership changes in the Ad Hoc network. The scalable distributed tree structure is converted to a mesh-based group message propagation mechanism using the redundant links in the network. A node can join the group tree by satisfying the membership criteria. This is carried out by having a trust negotiation between the stranger node and the designated group node. The trust negotiation ensures that the group joining process gives both the stranger and designated group member to check each others trustworthiness while ensuring protection to the sensitive confidential certificates. If the membership criterion is not strict malicious group node(s) can cause partial or completer disbanding of the group.

One of the motivations of this research is that the secure group collaboration can work on commodity handheld wireless devices. These devices constitute most of the nodes deployed by an emergency service to participate in an Ad Hoc network. The performance of the implementation is investigated in the next section.

## *5.6 Implementation*

The wanderer scheme is implemented on the NTRG test-bed [md01a, md01b] having actual mobile handheld devices. Due to paucity of the handheld wireless devices in the testbed, emulations are performed to test the performance of the secure group message propagation. The discussion of the implementation starts with the presentation of the layer structure for the wanderer scheme.

### 5.6.1 Group Layer Structure

The stack structure for the wanderer prototype is shown in Figure 66. A node can have three different stack structures depending on its role in the network. The most general stack structure is for a mobile wireless node (e.g. handhelds, laptops). In this configuration the bottom layer is an interface to the 802.11 hardware. Next layer is the filter layer which monitors the traffic passing through the node and sorts out the group messages that are being relayed but are not destined for the node itself. On top of this is the DSR layer which uses the "Dynamic Source Routing" [bjm04] algorithm for packet forwarding. Next is the group layer which is the core implementation of the wanderer scheme. The filter layer passes the intercepted group messages directly to the group layer. This way the group messages not destined for the node can be overheard and the group tree structure converted to a mesh for group message propagation. At the top level, the two applications are the Instant Messaging (IM) and the Voice layer which provides text based instant messaging and the SIP based [sip] telephony services respectively. Some of the nodes can act in a gateway or a relay role between the conventional and Ad Hoc network. These nodes require the bottom of the stack to also contain an Ethernet layer. Moreover some of the nodes may be emulated with the help of the JEmu emulator [ftm02]. The JEmu is an Ad Hoc network emulator in which an emulated server is connected to the emulator clients. The emulation environment results in the bottom most layer replaced by a wireless link emulating system called the JEmu layer [ftm02]. This allows testing of scenarios with a large number of nodes having a software stack equivalent to the real nodes. The functioning and limitations of the JEmu emulator is explained after the presentation of the various stack structures for a node implementing the wanderer scheme.

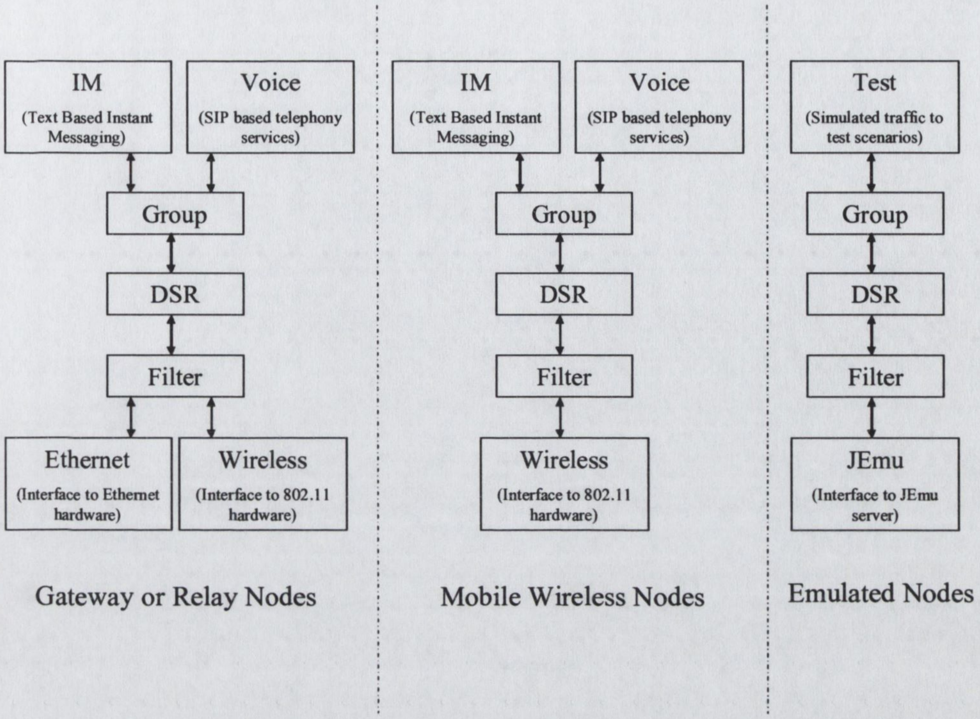
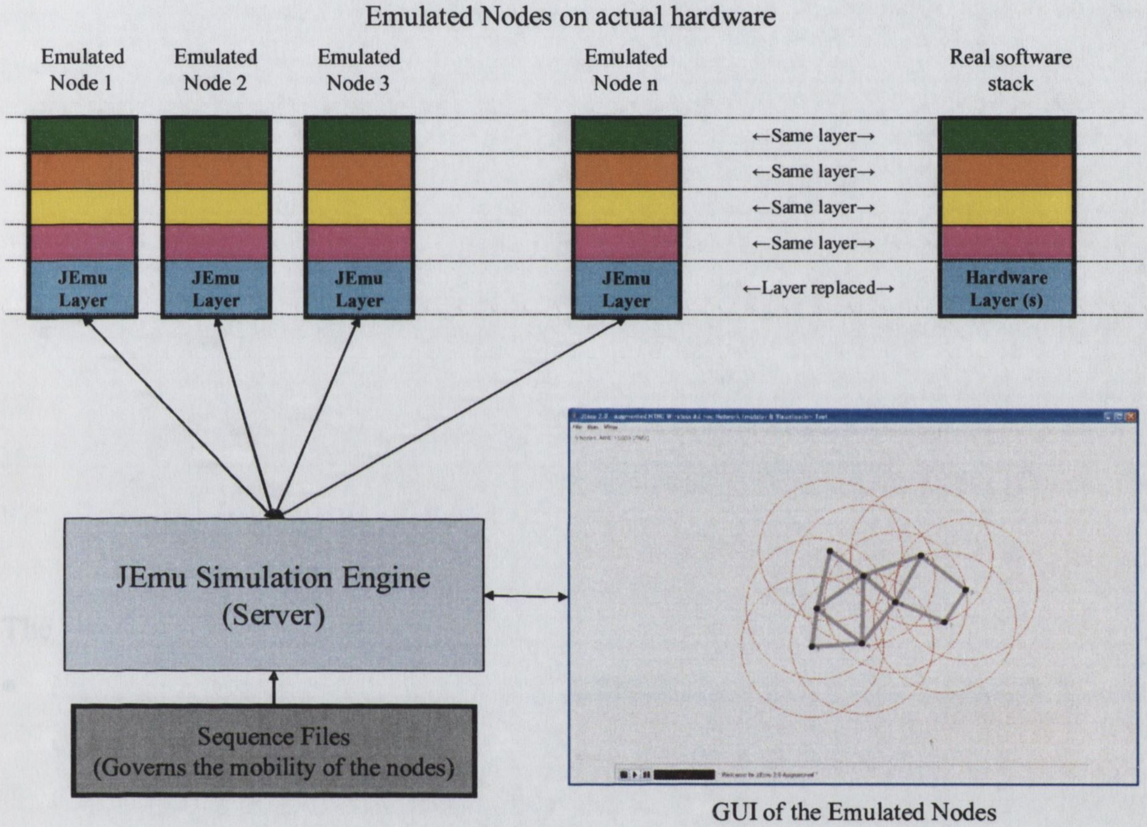

IM
(Text Based Instant Messaging)
Voice
(SIP based telephony services)
Group
DSR
Filter
Ethernet
(Interface to Ethernet hardware)
Wireless
(Interface to 802.11 hardware)
Gateway or Relay Nodes
IM
(Text Based Instant Messaging)
Voice
(SIP based telephony services)
Group
DSR
Filter
Wireless
(Interface to 802.11 hardware)
Mobile Wireless Nodes
Test
(Simulated traffic to test scenarios)
Group
DSR
Filter
JEmu
(Interface to JEmu server)
Emulated Nodes

**Figure 66 : Wanderer Layer Structure**

### 5.6.2 JEmu Emulator

In Ad Hoc networks there are three distinct methods for the testing and debugging of the networking protocols. The most commonly used method is the use of the free network simulators like NS-2 [ns2] and GlomoSim [glomosim]. In the simulated environment all the network components are modelled. The use of simulators is preferred by the Ad Hoc researcher community as it produces reproducible results for comparison. Moreover, the economic costs are low as access to actual wireless hardware is not required and a simulation for a scenario involving a large number of nodes can be done on a standard workstation. However, the comparative experiments by Cavin et al [css02] on the freely available simulators used by the Ad Hoc community show that the different simulators produce diverse results for the same scenario. Other experiments by Haq et al [hk05] showed that the simulated results varied greatly from that obtained on a testbed consisting of real hardware nodes. These experiments present a major dilemma for Ad Hoc researchers as the second method involving use of large-scale actual hardware is not economic. Moreover, complex mobility patterns are hard to perform on actual hardware unlike the simulators. A middle ground between the simulators and real testbed approaches is the use of emulators. In this method some of the network components are modelled while others are simulated. However, since the actual hardware available differs from researcher to researcher, the emulation results are difficult to compare. In the NTRG testbed, an emulator was developed to replicate the wireless connectivity between nodes. Thus a new software stack creation or recoding is not required for emulating a large number of nodes. This resulted in the emulation architecture illustrated in Figure 67 .



GUI of the Emulated Nodes

**Figure 67 : JEmu Emulation Architecture**

The emulation engine is connected to the emulated nodes over a fixed wired network. This allows the fast workstations to run multiple emulated nodes. Once the emulated nodes are connected to the server, the sequence files can be used to give the nodes simulated mobility. This simulated mobility is also reflected on the GUI displaying the mobility patterns in nearly real time. The GUI allows a user to modify the mobility scenario dynamically or even step through the entire mobility scenario manually. The GUI also has facilities to display the transmit/receive range of individual nodes along with the emulated network links. Moreover, the size of the simulation area can be fixed as needed by the scenario. The Figure 68 presents the JEmu emulation server algorithm.

1. Emulation server receives message from a JEmu client.
2. The emulation server timestamps the message and places the message along with parameters in a incoming queue. The parameters are
    1. Sending nodes position (Set by sequence files)
    2. Transmit/Receive range (Preset at start of the emulation)
    3. Interference range (Preset at start of the emulation)
3. Grinder thread moves the messages from the incoming queue to in-processing queue. These are messages with time stamp between the reference time interval and reference time interval + quantum time.
    1. The quantum time is the time period for which the emulation is done.
    2. The quantum time is fixed at the start of the emulation.
    3. Initial reference time is the time when the server starts.
    4. The new reference time interval is incremented by quantum time after messages are moved from the incoming queue to the in-processing queue.
4. Grinder thread iterates over the connected nodes and works out which of them can hear a message. The factors deciding if a node receives a message
    1. Node positions
    2. Transmit/Receive range of the nodes
    3. Interference range of the nodes (Presence of interference causes a packet to be dropped)
5. Grinder discards the in-processing queue and sleeps until messages in the incoming queue meet the conditions outlined in step 3. Therefore there are two independent loops (Steps 1 to 2 and steps 3 to 4).

**Figure 68 : JEmu Emulation Algorithm**

The main assumptions made in the JEmu algorithm are:

- No gradual degradation in the transmit/receive range of the nodes. The JEmu emulator assumes that circular transmit/receive range is solid. This implies that if another node is just outside this range it does not have any emulated network link (i.e. does not receive any packets).
- Similarly no gradual degradation in the interference range.
- Time quantum for the emulation is fixed at 1 ms.
- No provision made for the fact that there might be network congestion at the emulation server due to a large amount of clients connected. The network congestion can cause the late receipt of messages from some emulated clients.

Therefore the JEmu emulator does not completely model wireless network links and the network conditions like congestion and collisions. However, it provides a developer a platform to test networking protocols on a large number of nodes with actual software stack running on it.

### 5.6.3 Actual Implementation

The prototype of the wanderer secure group collaboration scheme was tested on the NTRG testbed handhelds with the same specifications and assumptions as in the one-to-one trust negotiation prototype. In the wanderer prototype the most time intensive operation involving timers is the group join operation. Therefore the analysis of the implementation starts with the presentation of the time required to do a typical join operation in the wanderer framework. The emergency scenario involved nodes from two countries coming together to form a group. A node *B* (from country B) decides to join the group (with *purpose* "Emergency_at_J20") after it receives an advertisement from node *A* (from country A). This node *A* is the leader of the group and sent the advertisement through an intermediate node (Figure 69). In the test scenario, the leader and the stranger both use 1024-bit identity certificates with Diffie-Hellman parameters embedded into them for key agreement. The leader finishes the key agreement phase (phase 1 of the join protocol) along with generation of fresh Diffie-Hellman parameters in 187 ms. The stranger requires 186 ms to finish the key agreement phase and then start the trust negotiation process.

The trust negotiation process (in phase 2 of the join protocol) with 1024-bit certificates on the leader and the stranger takes 272 ms and 354 ms respectively. The overall time for the entire join process was 546 ms and 540 ms respectively. Thus the join process involving exchange of four certificates and a key formation took a reasonable ½ second (approximately) on the commodity handheld wireless devices. Again, in this test the length of the certificate chain for each of the certificates is three. The size of the X.509 1024-bit modulus certificates used in the wanderer join tests varied from 600 to 1000 bytes with the variations in the size mainly due to the number of the embedded custom attribute name/value pairs. This ensured that the maximum sizes of the network messages in the join test to around 1400 bytes.

The timer intervals used in the join protocol test are same as those used in the RTM test in Chapter 4. It must be stressed again that these timer values where obtained for the maximum node density (i.e. nodes in transmit/receive range of each other) of six in the testbed.

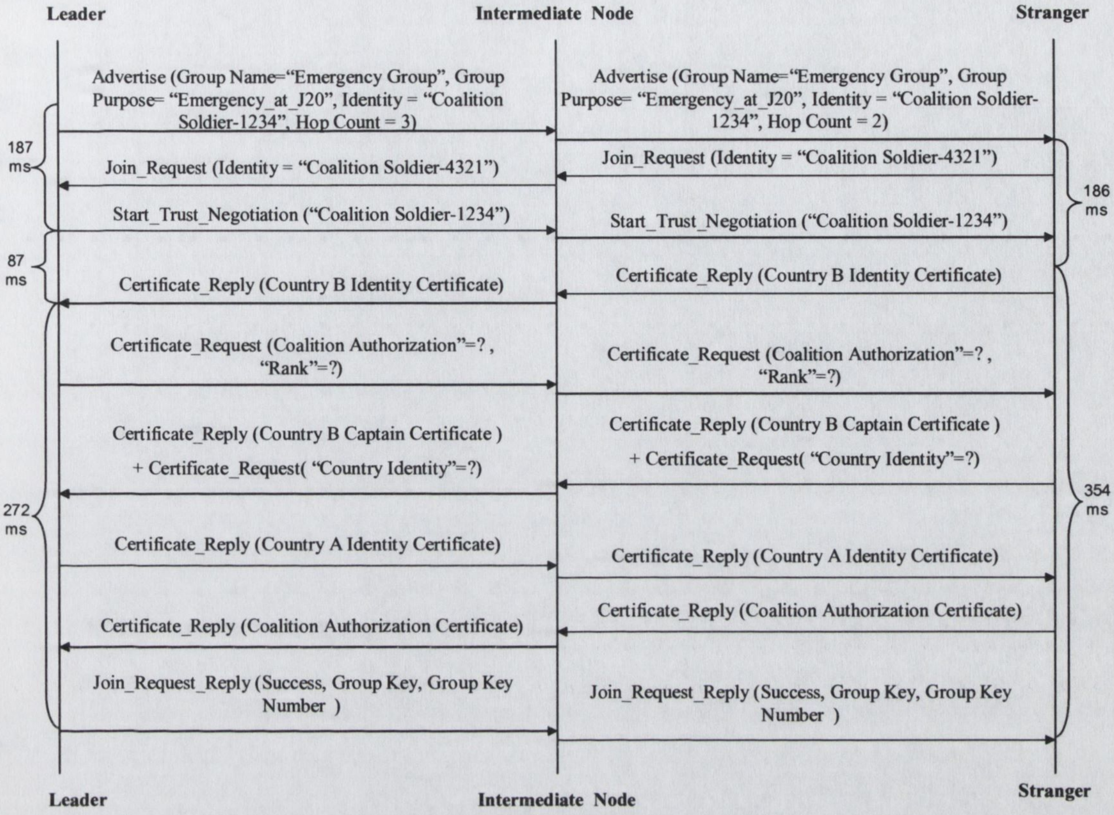
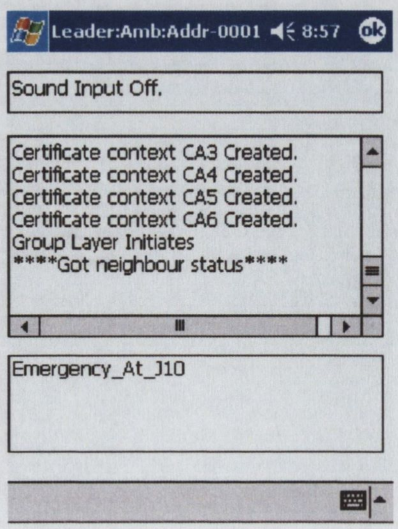

Leader
Intermediate Node
Stranger
Advertise (Group Name="Emergency Group", Group Purpose= "Emergency_at_J20", Identity = "Coalition Soldier-1234", Hop Count = 3)
Advertise (Group Name="Emergency Group", Group Purpose= "Emergency_at_J20", Identity = "Coalition Soldier-1234", Hop Count = 2)
187 ms
Join_Request (Identity = "Coalition Soldier-4321")
Join_Request (Identity = "Coalition Soldier-4321")
186 ms
Start_Trust_Negotiation ("Coalition Soldier-1234")
Start_Trust_Negotiation ("Coalition Soldier-1234")
87 ms
Certificate_Reply (Country B Identity Certificate)
Certificate_Reply (Country B Identity Certificate)
Certificate_Request (Coalition Authorization"=? , "Rank"=?)
Certificate_Request (Coalition Authorization"=? , "Rank"=?)
Certificate_Reply (Country B Captain Certificate ) + Certificate_Request( "Country Identity"=?)
Certificate_Reply (Country B Captain Certificate ) + Certificate_Request( "Country Identity"=?)
272 ms
354 ms
Certificate_Reply (Country A Identity Certificate)
Certificate_Reply (Country A Identity Certificate)
Certificate_Reply (Coalition Authorization Certificate)
Certificate_Reply (Coalition Authorization Certificate)
Join_Request_Reply (Success, Group Key, Group Key Number )
Join_Request_Reply (Success, Group Key, Group Key Number )
Leader
Intermediate Node
Stranger

**Figure 69 : Sample Join Process in Wanderer**

The screenshot of the wanderer group collaboration scheme initializing on a leader node is shown in Figure 70. It shows the information about the leader status of the node in the title bar. The screen shot also displays a debug window with information about the status of the group collaborations. At the bottom of the screenshot is the list of all group purposes which the node has membership.



Sound Input Off.

```
Certificate context CA3 Created.
Certificate context CA4 Created.
Certificate context CA5 Created.
Certificate context CA6 Created.
Group Layer Initiates
****Got neighbour status****
```

Emergency_At_J10

**Figure 70 : Screen Shot of the Wanderer**

The next performance test was done to test the group join protocol involved the setup of a group among the handhelds. In this test one of the nodes act as the group leader and the rest join it. The average result for 100 tests involving different number of member nodes is presented in Figure 71. This shows that the secure group can form in a reasonable time on the commodity handhelds using the wanderer scheme on the shared wireless medium.

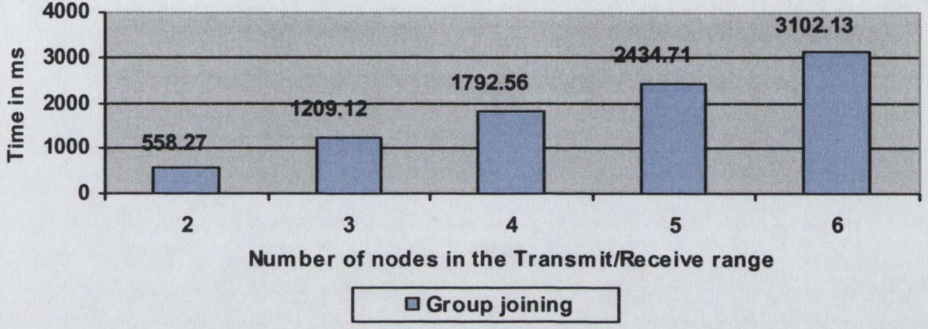


**Figure 71 : Performance of the Group Joining Protocol**

### *5.6.3.1 Joint Implementation*

The performance of the join protocol is also analysed in a scenario in which the node is participating both in the wanderer and one-to-one trust negotiation schemes. Therefore a joint implementation was done on the six handhelds of the NTRG testbed. The stack structure for this joint implementation is presented in Figure 73.

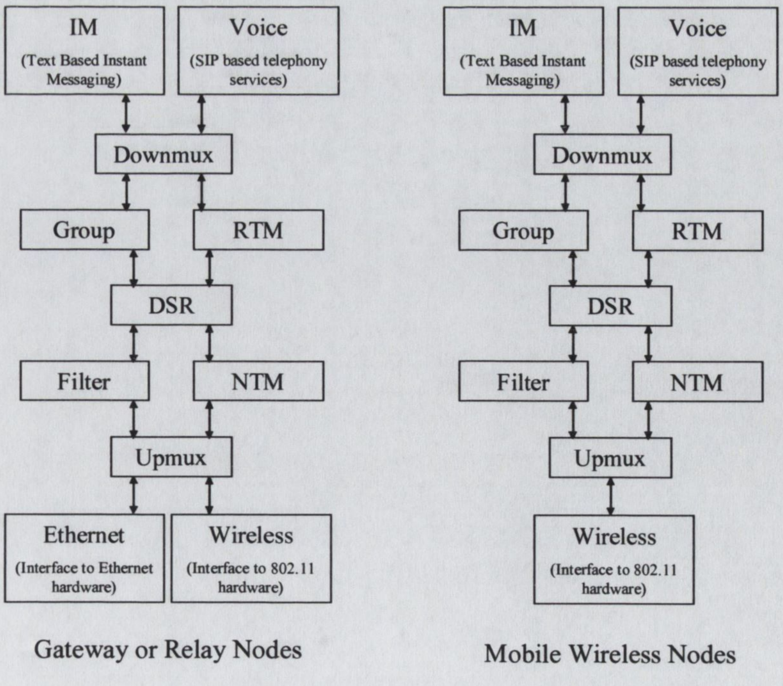

IM
(Text Based Instant Messaging)
Voice
(SIP based telephony services)
Downmux
Group
RTM
DSR
Filter
NTM
Upmux
Ethernet
(Interface to Ethernet hardware)
Wireless
(Interface to 802.11 hardware)
Gateway or Relay Nodes
IM
(Text Based Instant Messaging)
Voice
(SIP based telephony services)
Downmux
Group
RTM
DSR
Filter
NTM
Upmux
Wireless
(Interface to 802.11 hardware)
Mobile Wireless Nodes

**Figure 72 : Implementation of the Wanderer with the One-to-One Trust Negotaition**

The handhelds with the joint implementations where placed on a desk and switched on together. Then the time required to do the NTM key agreement (phase 1), RTM protocol (phase 1 & 2) and the join protocol of the wanderer scheme (phase 1 & 2 same as above example) was noted. In the test one node served as the leader and others joined it in the group. The time for the RTM and group formation was noted after the NTM phase 1 key agreement was completed between the nodes. It was difficult to find the individual times for the RTM and group join protocol interchanges as they ran concurrently. The joint result for average of 100 tests is tabulated in Figure 73. Despite the presence of interference in terms of the NTM and RTM protocol interchanges, the group join protocol finished forming the group in a reasonable time. The NTM and RTM protocol exchanges and timers are same as that used in the Chapter 4.

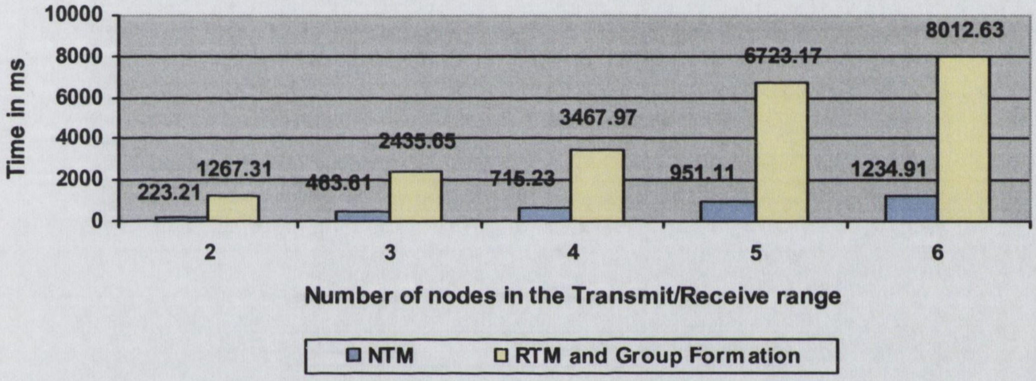


**Figure 73 : Performance of the Joint Implementation**

### 5.6.4 Emulation

Since only a small number of handhelds are available for testing, the wanderer scheme's content distribution is tested with a large number of emulated nodes. For the emulation environment, the bottom-most layers interfacing with the hardware are replaced by a wireless link emulating JEmu layer [ftm02]. The test involves a scenario in which 24 highly mobile nodes (Figure 74 and Figure 75) come together and form a secure group. The logical structure of group is different from the physical topology. In the screen shot (Figure 75) the red circles represent the range ring and the grey lines are the physical links between the nodes. It is also important to note that the Figure 75 gives the starting positions of the emulated nodes. Each of the node has a transmit/receive range of 30 units and an interference range of 45 units. The total area for the emulation is 150 x 250 units. If a node hits the invisible boundary it bounces back. At the start of the emulation each of the nodes is given varying random travel velocities by use of a sequence file. The mobile nodes bounce off the invisible boundary and consequently cause frequent changes in the network topology. Since the emulation boundary is small a node in the group tends to have at least one network link with the other nodes. In the emulation each node sends 100 messages to the group. The group leader "1" generates and distributes ten group keys during the simulation. A node during a typical run of the emulation receives an average 2270 out of 2300 group messages. Some nodes pass on more messages than the others which may cause congestion resulting in the dropping of a small number of messages.

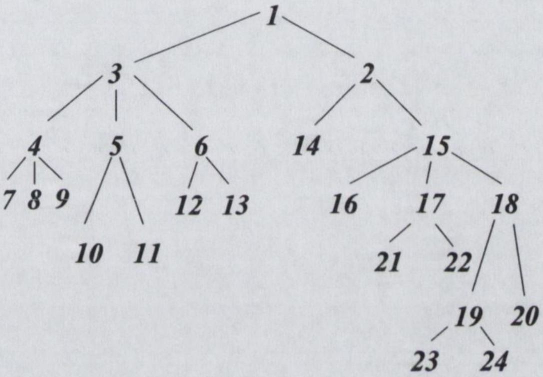


Figure 74 : Logical Structure of the Emulated Group

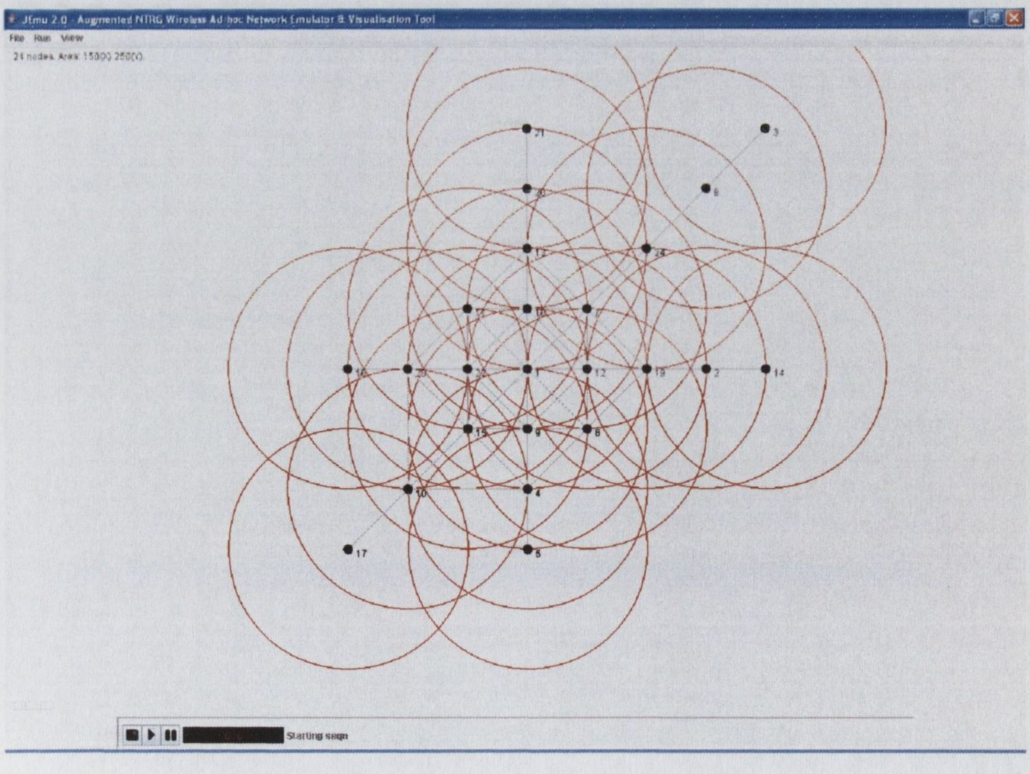

JEmu 2.0 - Augmented NTRG Wireless Ad hoc Network Emulator & Visualisation Tool
File Run View
24 nodes. Area: 1500() 2500()
21
3
6
24
1
12
2
14
10
4
17
5
Starting sequ

Figure 75 : Screen Shot at the Start of the Emulation

The results obtained from the emulation are plotted in Figure 76. In the graph the first bar shows the total number of group messages received by the node. The next two bars shows that due to mobility of nodes most of the group messages were received via the filter layer instead of being routed through DSR. This implies that the message was overheard and was not destined for the node. The message which the DSR layer passes to the group layer is sent specifically to the node by its parent or child node(s). Thus the messages routed through the DSR use the tree structure, while ones through the filter use the mesh. The predominant use of the mesh-based message propagation primarily results from the fact that the nodes were mobile and caused frequent topological changes. If the nodes in the emulation had same static topological layout as the logical group structure then the results would have shown predominant use of the tree-based (DSR in this case) group message propagation

Each node in the emulation relays a different number of group messages (last bar in the graph). This is primarily due to the mobility of nodes. Most of the small number of group messages not delivered to the nodes is due to congestion and running out of buffer space on the routing layer. However some of them may also be lost due to the partitioning of the group with some nodes not having any network links to the rest of the group.

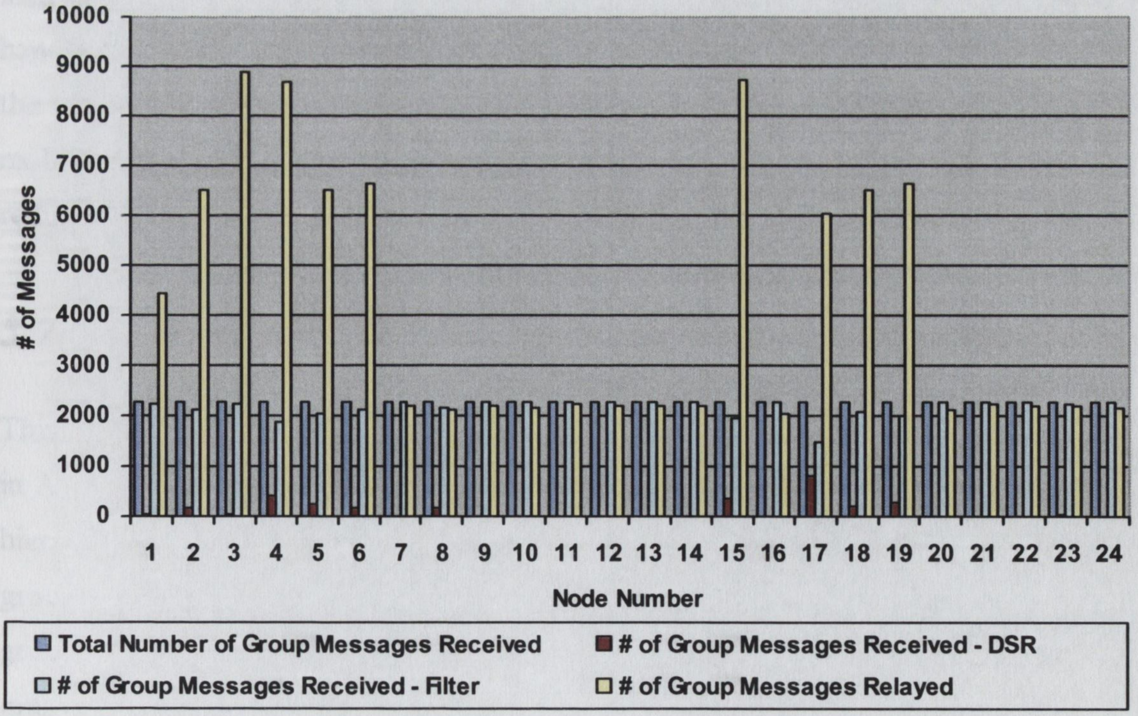


**Figure 76 : Emulation Results**

The emulation results show that for a mobile Ad Hoc group exhibiting frequent topological changes, the mesh-based group message delivery mechanism works better than the tree-based structure. Similar results are also demonstrated by the comparative simulation of Ad hoc wireless multicasting protocols carried out on GloMoSim [glomosim] simulator by Lee et al [lsh+00]. Therefore the emulation presented in this thesis generally agrees with the simulation result on the robustness of the mesh-based multicast message propagation. It is important to point out that it is difficult to directly compare the result obtained on the JEmu emulator and GloMoSim simulator due to different modelling algorithms used.

### 5.6.5 Implementation Summary

The implementation section demonstrated that the group join protocol is fast on the actual commodity handheld wireless devices. Moreover, the use of timers in the join protocol is feasible for forming a group on the six handheld devices in a reasonable time. The joint test of the secure group scheme with the one-to-one trust negotiation scheme on the handhelds showed that the two schemes are able to coexist together. These tests show that the secure group collaboration scheme is feasible to deploy on the handhelds forming the bulk of nodes for an emergency service. Since the actual number of handhelds is limited in the testbed, emulation for a mobile group was performed to test the robustness of the mesh based group message propagation. The test showed in the mobility scenario emulated, most of the group messages are delivered using the redundant links in the network instead of the using the group tree structure.

## *5.7 Conclusion*

This chapter presented a scalable, distributed and secure framework for group formation in Ad Hoc networks. The group formed by the wanderer framework has a distributed tree hierarchy for membership management which transforms into robust mesh structure for group message propagation. Since a group member has only a localized view of the group structure, the group is scalable for a large number of mobile nodes. Another innovation in the framework is that the logical structure of the group is overlaid on the actual physical topology. This allows the group structure to survive frequent topology and membership changes. The framework uses a novel way of authenticating new members to the group by progressively exchanging certificates. This authentication mechanism allows the node, having the ownership of a certificate, to use it in diverse roles instead of using it only for the purpose it was initially issued. The feasibility of this authentication method is demonstrated on real handheld wireless devices constituting the bulk of nodes in an emergency Ad Hoc network. In brief the secure group collaboration scheme meets the following criteria for an Ad Hoc security solution originally enumerated in the motivation section of Chapter 1.

- **Minimal Pre-Configuration** – The same certificates owned by a node can be used in multiple roles to satisfy the membership criteria of diverse groups. Thus, a node does not require additional pre-configuration when it moves to a new area of operation.
- **Little Coordination** – The coordination required to operate the secure group collaboration is localised. A node is only concerned about its parent and children node in the hierarchy.
- **Vary the Security Requirements** – The membership criterion lets the group leader set different group admission requirements depending on the scenario.
- **Computationally Feasible** – The join process which is the most computationally expensive operation in the wanderer scheme is shown to be reasonably fast on the actual commodity handheld devices.

# 6 Conclusions and Future Directions

## *6.1 Summary of Contributions*

The main goal of this thesis was to evaluate the use of trust negotiation techniques for building secure collaborations in a real Ad Hoc network suitable for use by emergency services. These secure collaborations allow a participating node to conduct confidential one-to-one and one-to-many communications for a range of applications. The main emphasis in this thesis is on researching, designing and developing the trust negotiation and associated protocols. This thesis also demonstrated the feasibility of use of these protocols in a real Ad Hoc network by testing them on real commodity handheld devices.

The trust negotiation approach was selected after a review of the existing conventional and Ad Hoc access control mechanisms. This approach was primarily chosen as it allowed each participating node to make its own decision about the trustworthiness of a remote node. In this approach only the two participating nodes need to coordinate unlike other Ad Hoc security approaches. Moreover, the role-free use of certificates in this approach allows a node to operate in a new area of operation with minimal re-configuration.

The first major contribution in this thesis is the development of a new one-to-one trust negotiation protocol suitable for use in an emergency Ad Hoc network. This was necessary as the original trust negotiation approach [wsj00] was found to lack a robust protocol for certificate exchange in an Ad Hoc scenario. The trust negotiation protocol developed allowed for confidential progressive exchange of certificates between the two participating nodes. The exchange of the certificates is governed by the local policies on the two nodes. One of the local policies maps from attributes in the received certificates into access control permissions on the service providing node. To ensure the confidentiality and integrity of the communications (including the trust negotiation) in the Ad Hoc network, a novel two-tier key framework independent of the routing algorithm was developed.

The second major contribution is the development of a secure group collaboration scheme suitable for emergency Ad Hoc networks. This new development was necessary for numerous reasons. Firstly, the one-to-one trust negotiation scheme is not scalable if a large number of nodes want to securely collaborate for some common task or share resources. Secondly after a review it was found that the existing secure group schemes proposed for Ad Hoc networks are not scalable or used computationally expensive public-key operations. Thus these schemes are infeasible in a large and dynamic Ad Hoc network consisting of CPU constrained commodity handheld devices. Such coordination is only possible to achieve in a small and stable Ad hoc network. Moreover during the review it was found that the mesh bases group message propagation scales better for a dynamic Ad Hoc network experiencing membership and topological changes. Thus a scalable, distributed and secure group formation framework suitable Ad Hoc networks deployed in an emergency scenario was proposed.

In the secure group collaboration scheme the nodes in the group are arranged in a tree structure. The tree is distributed and a participating node only keeps track of its parent and child nodes. It is not necessary that the parent and children nodes are adjacent to each other in the physical topology. The underlying routing layer maintains connectivity among the parent and children nodes. Thus the group tree is distributed and at the same time also scalable for a large number of nodes. However the tree-based group message propagation is not robust for a large and dynamic Ad Hoc network. Therefore in the proposed scheme the distributed group tree structure uses the redundant network links for robust mesh-based group message propagation. A stranger node can join the group by having a successful trust negotiation with the designated existing group member. This trust negotiation again involves progressive confidential exchange of certificates governed by the local policies of the two nodes. However in contrast to the one-to-one scheme, the trust negotiation in the secure group formation scheme is successful if the stranger node provides attributes in its certificates which satisfy the group membership criteria set by the group leader.

To demonstrate the feasibility of the two proposed schemes on real commodity handheld devices, benchmarking tests for the computationally heavy parts of the protocols were carried out and the results are presented in the thesis. Furthermore, emulation was performed on a mobile secure Ad hoc group to demonstrate the robustness of the mesh-based group message propagation under dynamic topological conditions.

In the Introduction Chapter, the motivation section presented the criteria for Ad Hoc security solutions which make them suitable for use by emergency services. The role-free use of certificates in the two proposed scheme lets a node operate in a new area of operation with *minimal pre-configuration*. Moreover, the two schemes require *little coordination* for operation compared to other Ad Hoc security solutions. Use of policies lets the two schemes *vary the security requirements* to suit the scenario. Furthermore, the schemes where found to be *computationally feasible* on real commodity handheld wireless devices expected to form the bulk of nodes used by emergency services. Therefore the requirements outlined in the Chapter 1 where met by the two security solutions presented in the thesis.

## *6.2 Directions for Future Research*

The research in this thesis focused on protocol issues and a used a very simple policy language for testing purposes. However to deploy the proposed schemes in an actual system, further work is required to refine the policy language and the associated compliance checkers. Another area where additional work needs to be done is in finding the optimal timer intervals for higher node densities than what was possible to explore in the implementation testbed. This is essential as higher node densities can cause an exponential drop in the network performance due to the increase in network congestion and retransmissions. In this thesis the JEmu emulator was used to test the performance of the Wanderer secure group scheme with a large number of nodes. The emulator approach is a middle ground between use of presently inaccurate simulators and the expensive actual hardware testbed approaches. Since the JEmu emulator uses a simplistic and basic modeling of the wireless medium therefore additional work needs to be done so that the emulator is able to more accurately model the various wireless technologies in use today. Another avenue of further research is to extend the proposed frameworks to integrate them in the existing access control frameworks.

The CPU, memory and the wireless networking capabilities of the commodity handheld wireless devices keep improving over time. This implies that the stronger cryptographic primitives with longer key lengths will become feasible to use on these devices. Moreover the improvements in the wireless network technologies will ensure the completion of longer and more secure key agreement exchanges in a reasonable amount of time. Therefore the schemes proposed in this thesis can be made more secure by incorporating the above-mentioned changes as the capabilities of the handheld devices improve over time. However despite the improvements in wireless network technologies, the problem of congestion remains as the wireless is a shared medium. This necessitates further work to mitigate and fully understand the effect of collision and interference on the key agreement phase of the two schemes proposed in this thesis. Moreover additional work needs to be done to understand the affects of the adverse wireless network conditions on the transformation of logical group structure from tree to mesh in the Wanderer Scheme.

The schemes proposed in this thesis can be made power-aware to conserve battery power on mobile wireless devises. This is particularly useful in remote emergency scenarios where access to power sources may be limited. In a similar way the proposed schemes should be able to use the wireless bandwidth intelligently. One of the ways of doing this is to make the new secure one-to-one or group associations use free channels to reduce congestions and retransmissions. Such a rationalization of the bandwidth will also indirectly help in conserving battery life of the mobile devices.

The security associations established by the wanderer framework can be extended to protect the self-routing used in Ad Hoc networks. Since in an Ad Hoc network there can be multiple routes between two nodes, the trust negotiation can help in selecting the best trusted route. A node can query the intermediate nodes on the multiple paths and come to a conclusion which is the best trusted node. However, this approach of best trusted path selection will only work in stable Ad Hoc network with few topological changes. The QoS of a route can also be factored in to determine the best choice.

# Appendix A Cryptographic Benchmarks

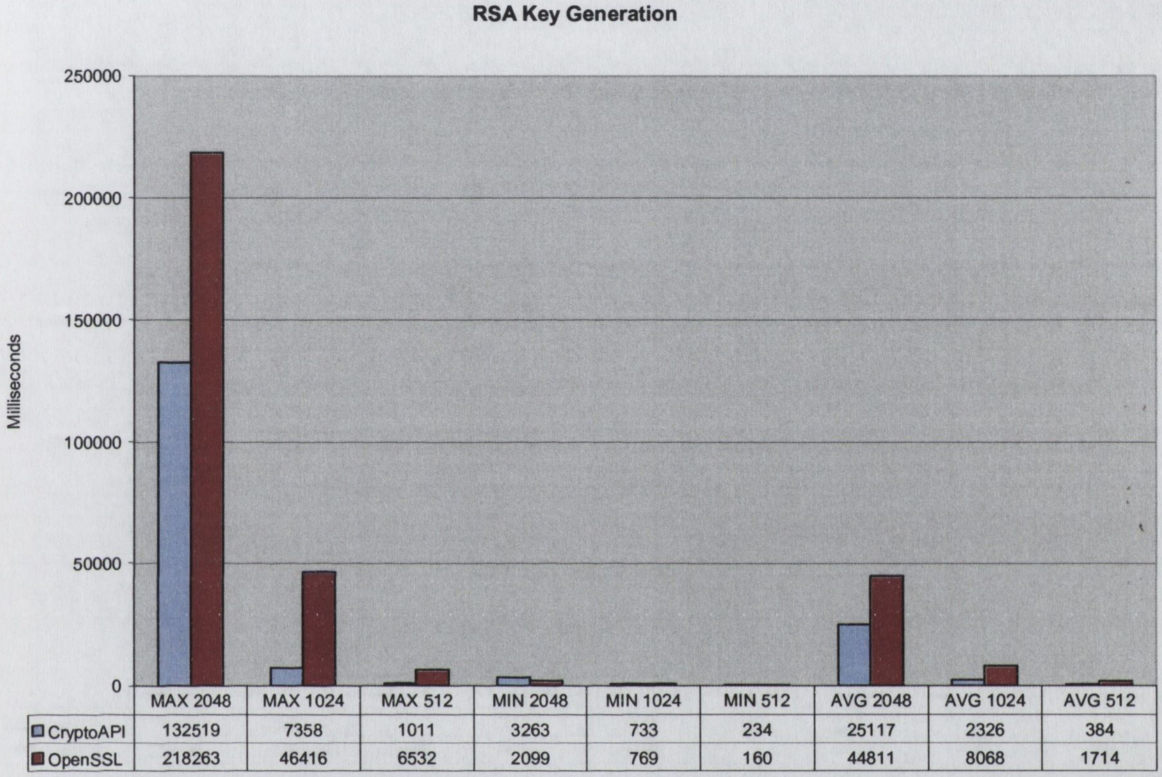
This appendix presents the benchmarks of the cryptographic primitives used in the thesis. The benchmarking was important for two reasons. Firstly, it helps in determining if the prototypes are computationally feasible on actual commodity handheld devices. Secondly, the benchmarks proved that the existing group management schemes using frequent generation of public-key certificates [mah00] are computationally infeasible on the actual handheld devices. The next section presents the test environment.

## *A.1 Test Environment*

The handheld computers used are the HP (Compaq) iPAQ H3630 [ipaq] with a 206 MHz StrongARM processor and 32MB RAM (16MB ROM), running the Windows CE Pocket PC 2002 [wince] operating system. For the benchmarking tests, the Windows CE port of the OpenSSL [openssl] cryptographic toolkit, version 0.9.7b is used. The same benchmarks where also performed utilizing the Microsoft Cryptography API [cryptoapi] and the results of the timing measurements were approximately the same. All the experiments are performed with RSA keys of 1,024 and 2,048 bits size, with small public exponents (*e* was given the value 65,537) making the public key operations significantly faster than the private key operations. The 512 bits keys are too short for sensitive data and therefore cannot be used in experiments that try to capture the realistic requirements of secure transactions. However, these are still included in the benchmarking process to compare it to previous benchmarking results [db99, gg01].

## *A.2 RSA Key Generation*

This test involved once off setting up of the needed data structures. The key generation was then performed 1,000 times on the iPAQ handhelds. At the end of the test the associated data structures were destroyed. The time measured for the key generation did not include the time required to initialize or destroy the associated data structures. The average, maximum and minimum time results for key generation are plotted in the graph presented in Figure 77. These results show that the average time to generate a reasonably secure 1024-bit modulus RSA key pair is approximately 8 seconds and 2 seconds using the OpenSSL and the CryptoAPI libraries respectively. Thus frequent generation of membership certificate can put strain on the CPU of a mobile commodity handheld device. This key generation time increases further if stronger 2048-bit RSA keys have to be generated. Thus it makes secure group management schemes [mah00] involving frequent generation of certificates infeasible on commodity handheld nodes.

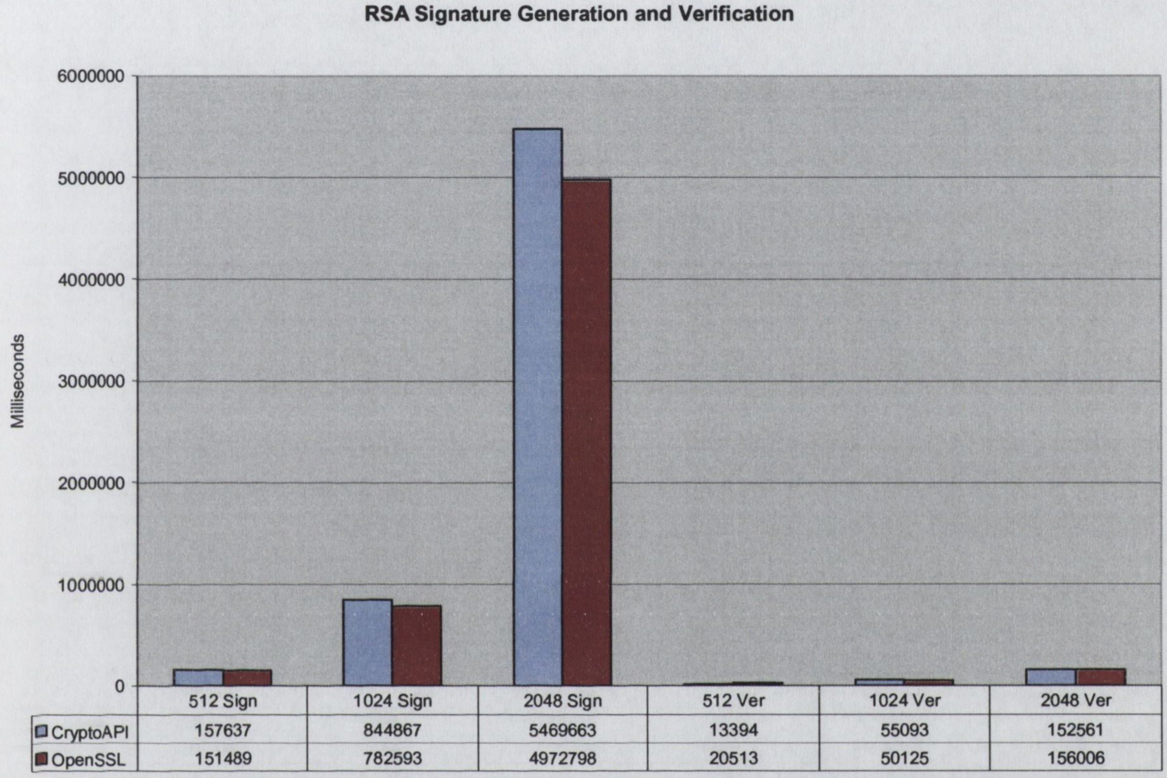


| | MAX 2048 | MAX 1024 | MAX 512 | MIN 2048 | MIN 1024 | MIN 512 | AVG 2048 | AVG 1024 | AVG 512 |
|---|---|---|---|---|---|---|---|---|---|
| 🟦 CryptoAPI | 132519 | 7358 | 1011 | 3263 | 733 | 234 | 25117 | 2326 | 384 |
| 🟥 OpenSSL | 218263 | 46416 | 6532 | 2099 | 769 | 160 | 44811 | 8068 | 1714 |

**Figure 77 : RSA Key Generation Benchmarks**

## *A.3 RSA signature generation and verification*

Asymmetric signature and verification operations are performed frequently by the two schemes presented in this thesis. The tests were performed to prove that these cryptographic primitive are fast enough on the actual handhelds to be used in the prototypes of the scheme. This test again involved once off setting up of the needed data structures needed. Then a 64 byte randomly generated string was generated. On the randomly generated string the signing and the verification operations where performed. The test was repeated for 10,000 iterations. At the end of the test the associated data structures and the randomly generated string was destroyed. The time measured for the signature and the verification primitives did not include the time required to initialize or destroy the associated data structures. The results are plotted in the graph presented in Figure 78. These tests show that the signing and the verification operation involving a 1024-bit public key take approximately 84ms and 5ms respectively using the OpenSSL cryptographic library. Therefore minimal CPU time is required on two nodes participating in a certificate exchange.

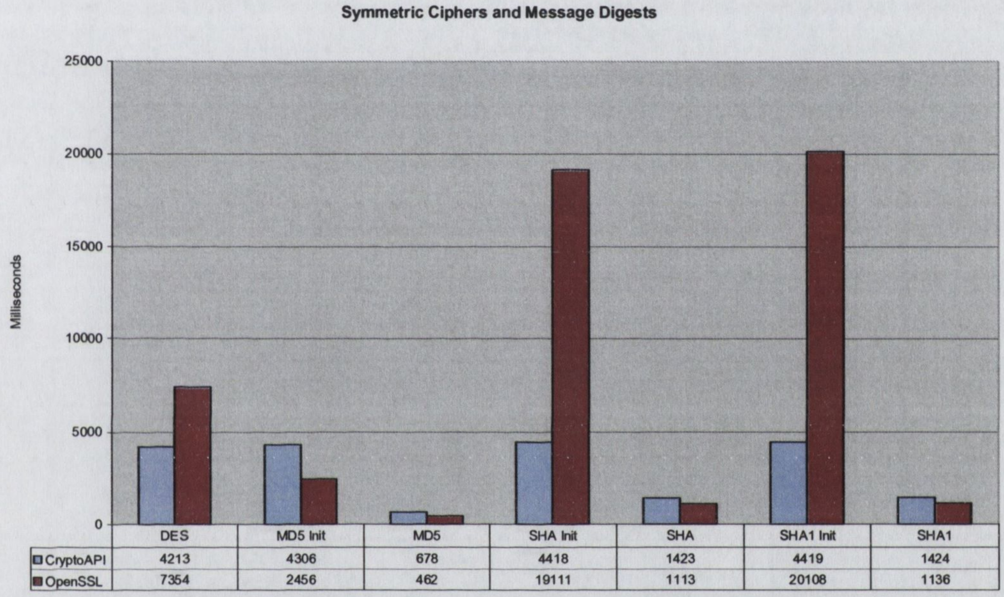


| | 512 Sign | 1024 Sign | 2048 Sign | 512 Ver | 1024 Ver | 2048 Ver |
|---|---|---|---|---|---|---|
| 🟦 CryptoAPI | 157637 | 844867 | 5469663 | 13394 | 55093 | 152561 |
| 🟥 OpenSSL | 151489 | 782593 | 4972798 | 20513 | 50125 | 156006 |

**Figure 78 : RSA Signature Generation and Verification Benchmarks**

## *A.4 Symmetric ciphers and message digests*

The data communication in this thesis used encryption to provide confidentiality. Moreover, the integrity of the communications was ensured using the Keyed-Hashing for Message Authentication (HMAC) [kbc97]. The implementation of the HMAC used the message digests. Thus the symmetric encryption and the message digest algorithms where benchmarked on the handheld devices. All the tests in this sub-section where performed for 100,000 iterations. The MD5, SHA and SHA1 message digests where tested with once off data structure initializations as in the previous tests. However, the prototype code also involved repeated use of code in which a data structure is initialized, cryptographic operation performed and then the associated data structure is destroyed. Such scenarios are benchmarked in the MD5 Init, SHA Init and the SHA1 Init tests. All the hashing algorithms tests had a 64 byte input. The DES test used a 64 bit plaintext input and 56-bit key. This test involved 100,000 encryptions and decryptions and one initialization of the data structures. The AES algorithm test involved 100,000 iterations on a 16 byte plaintext and three key lengths of 16, 24 and 32 bytes. The results plotted in graphs (Figure 79 and Figure 80) show that the symmetric encryption and the message digests are quite fast on the commodity handhelds. Thus they only contribute a marginal computational overhead in the prototypes of the schemes.



| | DES | MD5 Init | MD5 | SHA Init | SHA | SHA1 Init | SHA1 |
|---|---|---|---|---|---|---|---|
| ▨ CryptoAPI | 4213 | 4306 | 678 | 4418 | 1423 | 4419 | 1424 |
| ■ OpenSSL | 7354 | 2456 | 462 | 19111 | 1113 | 20108 | 1136 |

**Figure 79 : Benchmarks of Symmetric Ciphers and Message Digests**

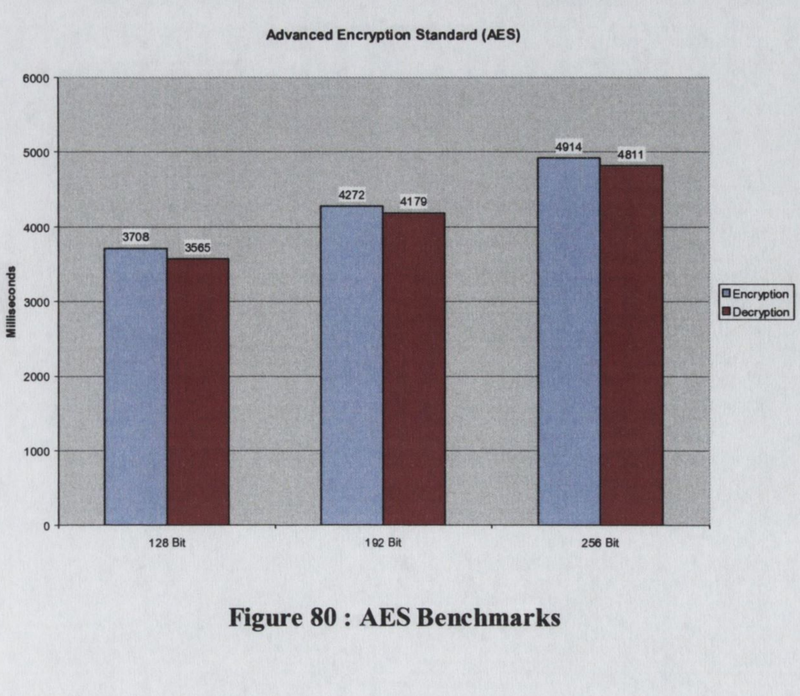


**Figure 80 : AES Benchmarks**

## *A.5 Related work*

In [gg01] the authors examine the performance of Kilobyte SSL (KSSL), a small footprint SSL client for the Java 2 Micro-Edition platform, on a 20 MHz Palm CPU with RSA keys of sizes 768 and 1,024 bits. Their results indicate that a full SSL handshake between a handheld client and a desktop server with only server-side authentication requires 10-13 seconds, which can be reduced to 7-8 seconds with certificate caching. RSA operations on the same platform require 0.5-1.5 seconds. RSA operations were also investigated in the context of electronic commerce through the use of handheld devices [db99]. The platform in this case was a PalmPilot Professional with a Motorola DragonBall chip at 16 MHz, running the PalmPilot port of the SSLeay cryptographic library. The observed results for RSA operations with 512 bits key pairs were 3.4 minutes for key generation, 7 seconds (7,028 ms) for signing and 1.4 seconds (1,376 ms) for verification.

# Appendix B Policy Language

This policy language is used to encode the service policy, certificate release policies and the membership criteria used by the schemes. In this appendix first the rudimentary language is presented and then its suitability for use in a trust negotiation environment is discussed. The policy language consists of two components the operators and the operands.

## *B.1 Language Specification*

<decimal digits> :: = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<letter> ::= A | B | C ... Z | a | b | c ... | z

<characters> :: = <decimal digits> | <letter>

<string1> ::= <characters> | <string1><characters>

<string2> ::= '*' | '?' | <characters> | <string1><characters>

<special string> ::= Free

Note:- Maximum string length can be 256

<logical operator> ::= = | < | > | <= | >= | <>

<boolean operator1> ::= AND | OR

<boolean operator2> ::= NOT

<exp1> ::= [(] <string1> <logical operator> <string2> [)]

<exp2> ::= [(] <exp1> <boolean operator1> <exp1> [)] | [(] <exp2> <boolean operator1> <exp2> [)]

<exp3> ::= <boolean operator2> [(] <exp1> [)] | <boolean operator2> [(] <exp2> [)]

<exp4> ::= [(] <exp3> <boolean operator2> <exp3> [)]

<expression> ::= <exp1> | <exp 2> | <exp3> | <exp4> | <special string>

In the <exp1> syntax the first <string1> is an attribute name and there is a table on each node mapping the attribute name to the X.509 certificate format [x.509] ASN.1 name. The second <string2> is the attribute value to be compared. If the <string2> ::= '*' it implies any non-null value will return the evaluation of the <exp1> to true. Other special character '?' is used when a node sends out a "Certificate Request". The brackets are evaluated from the innermost to the outermost. It is important to have same number of opening and closing brackets in an expression. A special expression *Free* is used when the appropriate policy specification does not require any attributes to satisfy it (i.e. signify an unlocked state).

## *B.2 Usage of the policy language*

### B.2.1 Service Policy

Entries in the service policy has the following format,

- <Service Name>: [Secret]: <expression> (to be satisfied to be granted the service).
- [<Disclosure Policy>: <expression> (to be satisfied to disclose the policy to a remote node)]

The square brackets signify the optional components. The <Service Name> ::= <string1> and <Disclosure Policy> ::=< string1>

### B.2.2 Certificate Release Policy

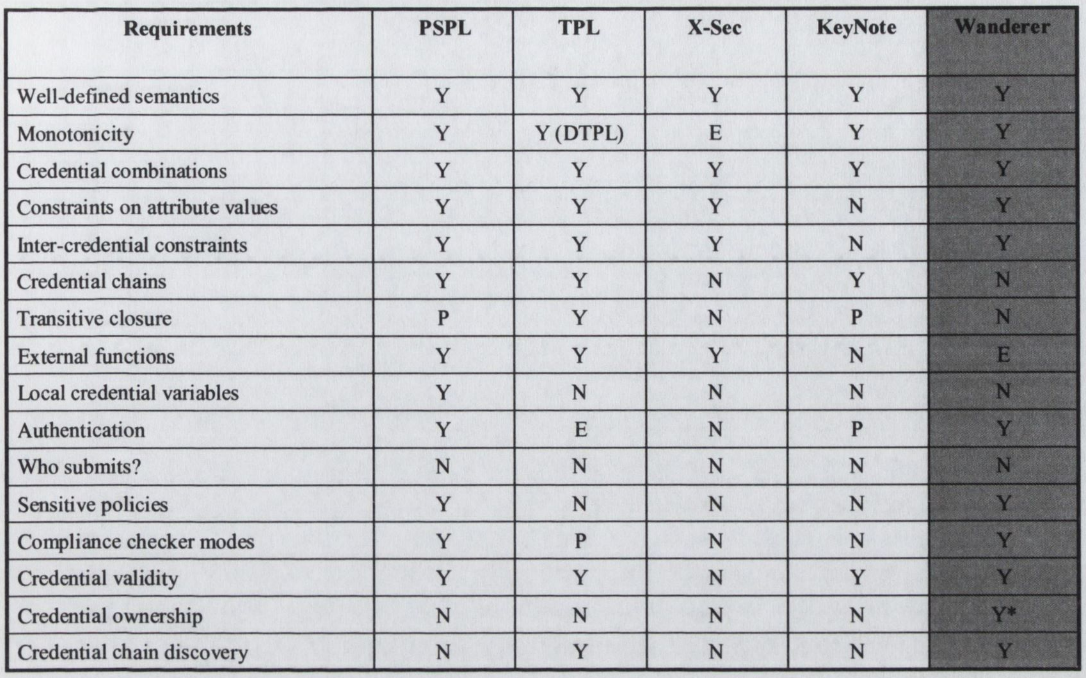
This policy is just an expression.

### B.2.3 Membership Criteria

This policy is an expression. The last line may be an optional disclosure policy using the format [<Disclosure Policy>: <expression> (to be satisfied to disclose the policy to a remote node]

## *B.3 Language Suitability*

A set of policy language requirements for trust negotiation have been proposed by Seamons et al [swy+02]. These requirements are enumerated (in Figure 81) and then compared with the policy language (i.e. Wanderer Policy Language) used in this thesis.



| Requirements | PSPL | TPL | X-Sec | KeyNote | Wanderer |
|---|---|---|---|---|---|
| Well-defined semantics | Y | Y | Y | Y | Y |
| Monotonicity | Y | Y (DTPL) | E | Y | Y |
| Credential combinations | Y | Y | Y | Y | Y |
| Constraints on attribute values | Y | Y | Y | N | Y |
| Inter-credential constraints | Y | Y | Y | N | Y |
| Credential chains | Y | Y | N | Y | N |
| Transitive closure | P | Y | N | P | N |
| External functions | Y | Y | Y | N | E |
| Local credential variables | Y | N | N | N | N |
| Authentication | Y | E | N | P | Y |
| Who submits? | N | N | N | N | N |
| Sensitive policies | Y | N | N | N | Y |
| Compliance checker modes | Y | P | N | N | Y |
| Credential validity | Y | Y | N | Y | Y |
| Credential ownership | N | N | N | N | Y* |
| Credential chain discovery | N | Y | N | N | Y |

A comparison of five languages for trust management with respect to the requirements for a policy language for trust negotiation (Key: Y-Yes, N-No, E-Easily extended, P-Partial support).
* - Provided by the protocols used in this thesis

**Figure 81 : Comparison of Five Policy Languages**

Furthermore the compliance checker algorithms using the wanderer policy language in this thesis is presented in the next section.

## *B.4 Compliance checkers*

The compliance checkers used in this thesis return true or false. A justification for false is evaluation is given in terms of attribute name/value pair(s). Thus the compliance checkers used in this thesis are of Type 1 [swy+02]. The other type (i.e. Type 2) returns a justification in either case of true or false evaluation. The compliance checkers in this thesis follows the following algorithm in verifying a certificate at runtime,

- The basis for the algorithm is the standard Windows XP methodology [k01] towards certificate status and revocation. In the abbreviated algorithm used in this thesis it is assumed that there is no cross-certification among the hierarchy of CA's in a chain. The abbreviated two interconnected processes in the algorithm are,

1. **Certificate Discovery** - This process collects the appropriate certificates for the certificate chain. All the certificates required for chain building are available locally. The chain building uses the exact match process using the Authority Key Identifier (AKI) extension. Thus, key and name matching is not used in the algorithm used for compliance checkers. Care is taken during the certificate generation time that the exact match only resulted in one chain. This was done to reduce the complexity of the verification process.

2. **Path Validation** - The certificate chain collected is processed in a hierarchical manner until a trusted self-signed root CA is reached. If the chain is not terminated in a trusted CA certificate, the received certificate is not trusted. Other reasons for the chain to fail the validation test are,
   - Time validation of any certificate in the chain failed
   - Unrecognized or malformed certificate format in the chain.
   - The intermediate or root CA certificates did not have power to issue new certificates.
   - The certificate in the chain is revoked using the CRL mechanism (note:- the Online Certificate Status Protocol (OCSP) mechanism is not used).

The outlines of the algorithm used by the schemes in the thesis are presented next. These algorithms use the credential verification mechanism discussed in this section.

### B.4.1 Compliance Checker used in the One-to-One Trust negotiation Scheme

The algorithm used for the compliance checker is,

1. On receiving inputs check if the node is service provider or service requester. If a service provider goto step 2 else goto step 3 (if a service requester). The inputs on which the decision is made are,
   a. Certificate Exchange History (CEH)
   b. Remote Node's Identity and UID
2. Service provider node
   a. Check if the local Service Policy (SP) is satisfied by using the following inputs,
      i. Service(s) Wanted by the remote node
      ii. Certificate Exchange History (CEH)
      iii. Remote Node's Identity and UID

```
        iv. Service Policy (SP)
        v. Remote Certificate(s) (if any) (Only valid remote certificates are considered)
    b. If the step 2(a) results in the remote node partially satisfying the Service Policy then,
        i. Issue the demands for remote certificate in form of attribute name/value pair(s) (if any)
        ii. Find any new service(s) unlocked (if any)
        iii. Set Status to Partial
        iv. Goto step 2(f)
    c. If the step 2(a) results in the remote node not satisfying the Service Policy then,
        i. Issue the demands for remote certificate in form of attribute name/value pair(s) (if any)
        ii. Find any free service(s) in the Service Policy (if any)
        iii. Set Status to No
        iv. Goto step 2(f)
    d. If the step 2(a) results in the remote node fully satisfying the Service Policy then,
        i. Set the output for the service(s) unlocked to everything wanted by the remote node.
        ii. Set Status to Yes
        iii. Goto step 2(f)
    e. If the step 2(d) results in the deadlock detection
        i. Set the deadlock output to true (By default the deadlock output is set to false)
        ii. Check if the Service Policy can be released to the remote node depending on,
            1. Certificate Exchange History (CEH)
            2. Disclosure Policy of the Service Policy
            3. Remote Node's Identity and UID
            4. Remote Certificates (if any)
        iii. Ask the user inputs- if this option is set by the user
        iv. Goto step 2(f)
    f. Add the certificates (if any) and the attributes requested (if any) to the Certificate Exchange
       History. The invalid certificates are also added but marked invalid. Goto step 3
3. Get the unlocked certificates to the remote node depending on,
    a. Eager Strategy
    b. Local Certificate Store (LCS)
    c. Certificate Exchange History (CEH)
    d. Service(s) Wanted by the local node
    e. Remote Node's Identity and UID
    f. Certificate Release Policies
```

### B.4.2 Compliance Checker used in the Wanderer Scheme

The algorithm used for the compliance checker is,

```
1. On receiving inputs check if the node is leader/sub-leader or stranger. If a leader/sub-leader goto step 2 else goto step 3 (if a stranger). The inputs on which the decision is made are,
    a. Certificate Exchange History (CEH)
    b. Remote Node's Identity and UID
2. Leader/Sub-leader node
    a. Check if the Membership Criteria (MC) is satisfied by using the following inputs,
        i. Certificate Exchange History (CEH)
        ii. Remote Node's Identity and UID
        iii. Membership Criteria (MC)
        iv. Remote Certificate(s) (if any) (Only valid remote certificates are considered)
    b. If the step 2(a) results in the remote node partially satisfying the Membership Criteria then,
        i. Issue the demands for remote certificate in form of attribute name/value pair(s) (if any)
        ii. Set Status to Partial
        iii. Goto step 2(f)
    c. If the step 2(a) results in the remote node not satisfying the Membership Criteria then,
        i. Issue the demands for remote certificate in form of attribute name/value pair(s) (if any)
        ii. Set Status to No
        iii. Goto step 2(f)
    d. If the step 2(a) results in the remote node fully satisfying the Membership Criteria then,
        i. Set Status to Yes
        ii. Goto step 2(f)
    e. If the step 2(d) results in the deadlock detection
        i. Set the deadlock output to true (By default the deadlock output is set to false)
        ii. Check if the Membership Criteria can be released to the remote node depending on,
            1. Certificate Exchange History (CEH)
            2. Disclosure Policy of the Membership Criteria
            3. Remote Node's Identity and UID
            4. Remote Certificates (if any)
        iii. Ask the user inputs- if this option is set by the user
        iv. Goto step 2(f)
    f. Add the certificates (if any) and the attributes requested (if any) to the Certificate Exchange History. The invalid certificates are also added but marked invalid. Goto step 3
3. Get the unlocked certificates to the remote node depending on,
    a. Eager Strategy
    b. Local Certificate Store (LCS)
    c. Certificate Exchange History (CEH)
    d. Remote Node's Identity and UID
    e. Certificate Release Policies
```

# Appendix C Code Fragments

The complete code for the two schemes presented runs to around 34,000 lines of code so the displayed code snippets are confined to ones which demonstrate the important aspects of the implementation. One of the important processes used in the schemes was the embedding of custom attribute name/value pair embedded in the certificates. Another noteworthy feature was how our NTRG architecture [md01a, md01b] created and handled the network primitives. To demonstrate the code for constructing a network primitive, the snippet used to create the “Advertise” network primate of the secure group collaboration is presented due to its simplicity.

## *C.1 Adding and extracting custom attribute into a X.509 certificate.*

This code is used often in the implementation to encode custom attribute name/value pair(s) into the X.509 certificate. The first code snippet is used to embed the Diffie-Hellman parameters “p” and “g” in an *address* certificate of the NTM at the certificate generation time. It also contains the “mac_id” to show that the node has the ownership of its MAC level network address. The simplified commented C++ code for addition of the “mac_id” attribute without any error handling is,

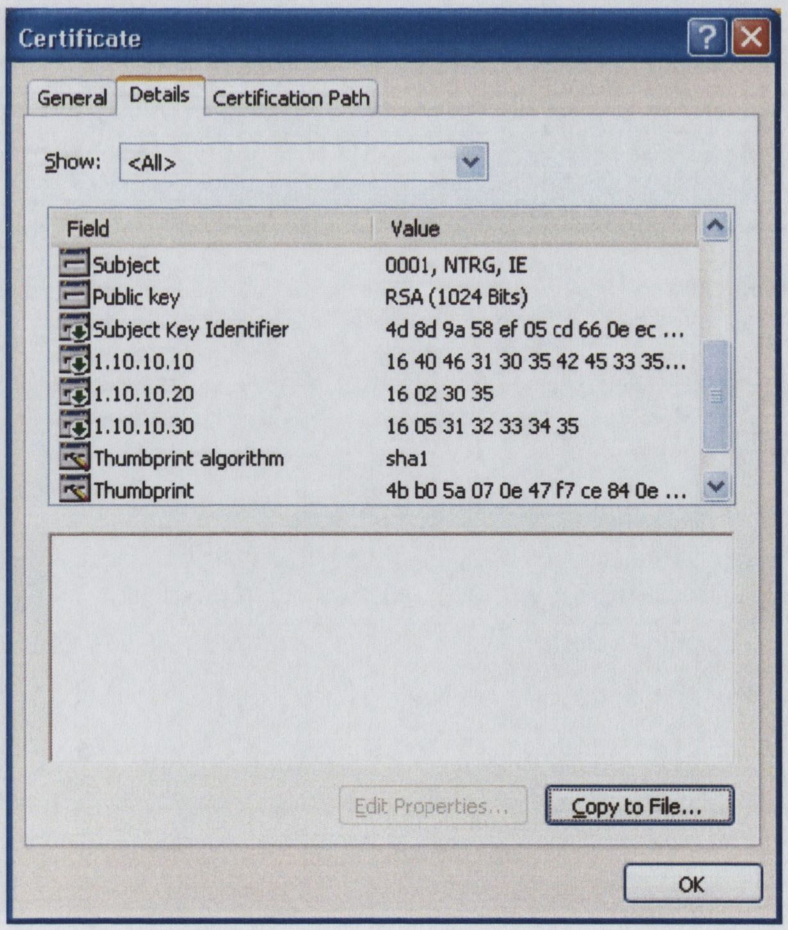
```
int nid;
X509_EXTENSION *ex;
X509V3_CTX ctx;
/* create a runtime mapping between ASN.1 string "1.10.10.30"and “mac_id” attribute
*/
nid=OBJ_create("1.10.10.10","myalias","mac_id");
X509V3_EXT_add_alias(nid,NID_netscape_comment);
/* This sets the 'context' of the extensions. A certificate context with no CRL or certificate
request is create, therefore two NULL’s in the function call*/
/*cert is the certificate to be signed and issuer is the CA certificate */
X509V3_set_ctx(&ctx, cert, issuer, NULL, NULL, 0);
/*create the certificate extension*/
```

```
ex = X509V3_EXT_conf_nid(NULL, &ctx, nid, “12345”);
/* add it to the certificate*/
X509_add_ext(cert, ex, -1);
```

Similar code can be used to add the values of “p” and “g”. The mapping between ASN.1 and attribute name used in this example is,

- 1.10.10.10 – p
- 1.10.10.20 – g
- 1.10.10.20 – mac_id

The windows dump of the generated 1024-bit *address* certificate is shown in Figure 82. Note that the dump only shows the ASN.1 string and associated value. Therefore the ASN.1 to attribute name mapping is lost in the certificate creation time. To get back the correct values this mapping should also be known to the decoder who wants to extract the custom attribute name/value pair.



Show: <All>

| Field | Value |
|---|---|
| Subject | 0001, NTRG, IE |
| Public key | RSA (1024 Bits) |
| Subject Key Identifier | 4d 8d 9a 58 ef 05 cd 66 0e ec ... |
| 1.10.10.10 | 16 40 46 31 30 35 42 45 33 35... |
| 1.10.10.20 | 16 02 30 35 |
| 1.10.10.30 | 16 05 31 32 33 34 35 |
| Thumbprint algorithm | sha1 |
| Thumbprint | 4b b0 5a 07 0e 47 f7 ce 84 0e ... |

**Figure 82 : Windows Dump of the *Address* Certificate**

The code to extract the custom attribute name/value pair on the receiver side the code snippet looks like,

```
/* create the mapping between the attribute name and the ASN.1 string*/
nid=OBJ_create("1.10.10.10","myalias","p");
X509V3_EXT_add_alias(nid,NID_netscape_comment);
nid=OBJ_create("1.10.10.20","myalias","g");
X509V3_EXT_add_alias(nid,NID_netscape_comment);
nid=OBJ_create("1.10.10.20","myalias","mac_id");
X509V3_EXT_add_alias(nid,NID_netscape_comment);
/*print the attribute name vale pairs on the console – can be easily redirected to a file*/
BIO *bio_out;
bio_out = BIO_new_fp(stdout, BIO_NOCLOSE);
/*get the certificate in an easily readable format*/
X509_CINF *ci;
ci=cert->cert_info;
/*enumerate the extensions embedded in the certificate*/
for (int i=0; i<sk_X509_EXTENSION_num(ci->extensions); i++)
{
        ASN1_OBJECT *obj;
        X509_EXTENSION *ex;
        ex=sk_X509_EXTENSION_value(ci->extensions, i);
        if(!X509V3_EXT_print(bio_out, ex, flag, 0))
        M_ASN1_OCTET_STRING_print(bio_out,ex->value);
        if(!X509V3_EXT_print(bio_value, ex, flag, 0))
        M_ASN1_OCTET_STRING_print(bio_value,ex->value);
}
```

## C.2 Creation and sending of an "Advertise" network message

The "Advertise" is used by the secure group scheme to tell the nodes in the vicinity of the group's existence. The function used to sent this message is presented below,

```
/* gn is the group name - gp is the group purpose - x_binary, p_binary and g_binary is
the Diffie_hellman parameters "X", "p" and "G" to be sent – hop_count determines
how far the group message will propagate – dest is the destination to which message is
to be sent (the neighbours detected by the NTM layer) */
void send_1(char gn[16],char gp[16],BYTE x_binary[32],BYTE p_binary[32],BYTE
g_binary,int hop_count,int dest)
{
        /* initialize the NTRG message structure*/
        Layer_Primitive *arg;
        arg = create_layer_primitive();
         /* The message is a normal message destined for transmission */
        arg->message_type = XFER_PRIM;
        /* Start filling the message buffer from fifth position so layers below can attach
its own header at start of the message*/
        arg->start = 5;
        /* Message length is 110 bytes*/
        arg->length = 110;
        /* identifier 1234 marks the beginning of a group message*/
        arg->buffer[arg->start]='1';
        arg->buffer[arg->start+1]='2';
        arg->buffer[arg->start+2]='3';
        arg->buffer[arg->start+3]='4';
        /* 1 denotes the message is a advertise message*/
        arg->buffer[arg->start+4]=1;
        /* Group Name added to the message*/
        memcpy(&arg->buffer[arg->start+5],&gn[0],16);
        /* Group purpose added to the message*/
        memcpy(&arg->buffer[arg->start+21],&gp[0],16);
        /* Diffie-Hellman parameter X added to the message*/
        memcpy(&arg->buffer[arg->start+37],&x_binary[0],32);
        /* Diffie-Hellman parameter p added to the message*/
        memcpy(&arg->buffer[arg->start+69],&p_binary[0],32);
        /* Diffie-Hellman parameter g added to the message*/
        memcpy(&arg->buffer[arg->start+101],&g_binary,1);
        /* Node name added to the message*/
        memcpy(&arg->buffer[arg->start+102],&self_name,sizeof(int));
        /*Hop count added to the message*/
        memcpy(&arg->buffer[arg->start+106],&hop_count,sizeof(int));
        char temp_dest[4];
        integer2char(dest,temp_dest);
        /*Destination added to the message*/
        add_entry(arg,"dest",temp_dest);
        /*Message is sent to layer below for processing*/
        send_downwards(me1,me1->neighbour_bot,arg);
}
```

## C.3 Receiving and deciphering of network messages by the group layer

The code for the main message handling code for the group layer is presented in this section.

```
/* Initialize the group layer*/
Layer_Primitive *argument;
int timerid=start_timer(5000);
initiate();
/* Initialize the certification authorities and associated structures */
create_ca();
log("Group Layer Initiates");
/* Begin the endless loops that does the processing*/
bool loop_true=true;
while(loop_true)
{
  /* Wait for a message – either form layer below or above or a timer expiry*/
  event_source=layer_wait_for_message(me1,&argument);
  if(event_source!=1020) // filter out nuisance Windows CE GUI message
  {
   if(event_source==MSG_FROM_BELOW) // message form layer below
   {
    if((argument->message_type==XFER_PRIM) || (argument->message_type==20)) // only type of
messages this layer is concerned above is normal group messages or special type 20 message from the
filter layer below
     {
      int lSize=argument->length; // size of message from below
      char * pass_in = (char *)malloc(lSize); // pass the message to appropriate handling function
      memcpy(&pass_in[0],&argument->buffer[argument->start],lSize); // copy the message
      if(argument->buffer[argument->start]=='1' && argument->buffer[argument->start+1]=='2' &&
argument->buffer[argument->start+2]=='3' && argument->buffer[argument->start+3]=='4') // is the
message a proper group layer message
       {
        log("****Got a message****\n");
        if(argument->buffer[argument->start+4]==1)
         {
          log("****Got  a Advertise Message****");
          handle_1(pass_in,lSize);
         }
        if(argument->buffer[argument->start+4]==2)
         {
          log("****Got a Join_Request message****");
          handle_2(pass_in,lSize);
         }
        if(argument->buffer[argument->start+4]==3)
         {
          log("****Got a Start_Trust_Negotiation message****");
          handle_3(pass_in,lSize);
         }
        if(argument->buffer[argument->start+4]==4)
         {
          log("****Got a Certificate_Request message****\n");
          handle_4(pass_in,lSize);
         }
        if(argument->buffer[argument->start+4]==5)
         {
          log("****Got a Certificate_Reply****");
          handle_5(pass_in,lSize);
         }
        if(argument->buffer[argument->start+4]==6)
         {
          log("****Got a Release_Membership_Criteria message****\n");
```

```
  handle_6(pass_in,lSize);
 }
 if(argument->buffer[argument->start+4]==7)
 {
  log("****Got a Join_Request_Reply message****");
  handle_7(pass_in,lSize);
 }
 if(argument->buffer[argument->start+4]==8)
 {
  log("****Got a Group_Key_Update message****");
  handle_8(pass_in,lSize);
 }
 if(argument->buffer[argument->start+4]==9)
 {
  log("****Got a Group_Key_Update_Acknowledgement message****");
  handle_9(pass_in,lSize);
 }
 if(argument->buffer[argument->start+4]==10)
 {
  log("****Got a Group_Message message****");
  handle_10(pass_in,lSize);
 }
 if(argument->buffer[argument->start+4]==11)
 {
  log("****Got a Sub_Leader_Leave message****");
  handle_11(pass_in,lSize);
 }
 if(argument->buffer[argument->start+4]==12)
 {
  log("****Got a Sub_Leader_Leave_Acknowledgement message****");
  handle_12(pass_in,lSize);
 }
 if(argument->buffer[argument->start+4]==14)
 {
  log("****Got a Leader_Leave message****");
  handle_14(pass_in,lSize);
 }
 if(argument->buffer[argument->start+4]==15)
 {
  log("****Got a Leader_Leave_Acknowledgement message****");
```

```
       handle_15(pass_in,lSize);
      }
     if(argument->buffer[argument->start+4]==16)
      {
       log("****Got a Member_Leave message****");
       handle_16(pass_in,lSize);
      }
     if(argument->buffer[argument->start+4]==17)
      {
       log("****Got a DISBAND message****");
       handle_17(pass_in,lSize);
      }
     if(argument->buffer[argument->start+4]==18)
      {
       log("****Got a MISBEHAVIOUR message****");
       handle_18(pass_in,lSize);
      }
     if(argument->buffer[argument->start+4]==19)
      {
       log("****Got a AVOID message****");
       handle_19(pass_in,lSize);
      }
     free(argument);
   }//end layer check
   else
   {
     if(argument->message_type==20) // message form filter layer giving a list of current neighbours
      {
       log("****Got neighbour status****");
       handle_type_20(pass_in,lSize);
       free(argument);
      }
     else
      {
       send_upwards(me1,me1->neighbour_top,argument);
      }
   }
  if(pass_in!=NULL)free(pass_in);
 }
 else // if not xprimtype or special message from filter layer
```

```
      {
        send_upwards(me1,me1->neighbour_top,argument);
      }
    }// message from below
    else // message from above or timer
    {
    if(event_source==MSG_FROM_ABOVE) // The MFC layer above has send data to be sent to the group
    {
      BYTE *got_m=(BYTE *)malloc(argument->length);
      memcpy(&got_m[0],&argument->buffer[argument->start],argument->length);
      send_group_message(got_m,encrypt_data); // send the data received to the appropriate group
      free(argument);
      free(got_m);
    }
    if(event_source==TIMER_EXPIRED) // a message is generated each time the main timer expires
    {
      current_time=GetTickCount();
      if((DWORD)(current_time-self_timer)>(DWORD)10000)
      {
        send_peroidic(); // send the periodic advertise message – only if this node is a leader
        self_timer=GetTickCount();
      }
      if((DWORD)(current_time-x_TimeStamp)>(DWORD)120000)
      {
        generate_x(); // generate a fresh Diffie-Hellman parameter X
        generate_group_key(); // generate a new group key – only if this node is a leader
        send_group_keys(); // send the group keys to – only if this node is a leader
      }
    } // end of timer check
  } // Nuisance 1020 Windows CE message
  success_meeting_mc(); // check if any remote node negotiating with this node to join has satisfied the
membership criteria – only if this node is a leader
  clear_sleepers(); // clear the entries in data structures for dormant remote nodes
  initiate_trust_negotiation(); // release certificates
 }
}//while true
```

# Bibliography

[802.11] IEEE Computer Society LAN/MAN Standards Committee, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", *IEEE Std. 802.11-1997. IEEE*, 1997.

[802.11] L. M. S. C. OF THE IEEE COMPUTER SOCIETY, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. " *IEEE Standard 802.11, 1999 Edition*, 1999.

[abb+04] W. Aiello, S. M. Bellovin, M. Blaze, R. Canetti, J. Ioannidis, A.D. Keromytis and O. Reingold, "Just fast keying: Key agreement in a hostile internet", *ACM Transactions on Information and System Security (TISSEC), Volume 7, Issue 2*, May 2004.

[adk92] Y. Amir, D. Dolev, S. Kramer, and D. Malki, "Transis: A Communication Sub-System for High Availability", *In Proceedings of the 22nd Annual International Symposium on FaultTolerant Computing*, Jul 1992, pp.76-84.

[afb+02] D. G. Andersen, N. Feamster, S. Bauer and H. Balakrishnan, "Topology Inference from BGP Routing Dynamics", *In Proceedings of the 2nd SIGCOMM Internet Measurement Workshop, Marseille, France*, Nov 2002, pp.243-248.

[an97] T. Aura and P. Nikander, "Stateless Connections", *Proceedings of the First International Conference on Information and Communication Security (ICICS '97)*, 1997.

[ao03] P. G. Argyroudis and D. O'Mahony, "A Survey of Secure Ad hoc Routing Protocols", *Technical Report, Department of Computer Science, University of Dublin, Trinity College*, 2003.

[as98] Y. Amir and J. Stanton, "The Spread Wide Area Group Communication System", *Technical Report CNDS 98-4*, 1998.

[b96] A. Ballardie, "Scalable multicast key distribution, RFC 1949", May 1996.

[b97] A. Ballardie, "Core Based Trees (CBT) Multicast Routing Architecture (RFC 2201)", Sep 1997.

[b97a] C. Boyd, "On Key Agreement and Conference Key Agreement", *Information Security and Privacy, LNCS 1270, Springer-Verlag*, 1997, pp.294-302.

[b99] D. Boneh, "Twenty years of attacks on the RSA cryptosystem", *Notices of the American Mathematical Society (AMS), Volume 46, Number 2*, 1999, pp.203-213.

[bd94] M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution system", *Advances in Cryptology -- EUROCRYPT'94*, May 1994, pp.425-438.

[bfk98] M. Blaze, J. Feigenbaum, and A. D. Keromystis, "The KeyNote Trust Management for Public-Key Infrastructures", *Security Protocols International Workshop, Cambridge, England*, 1998.

[bgs92] J.A. Bull, L. Gong and K. Sollins, "Towards Security in an Open Systems Federation", *In Proceedings of the ESORICS 92, Springer LNCS Volume 648*, Nov 1992, pp.3-10.

[bh03] L. Buttyan and J.-P. Hubaux (editors), G. Avoine, S. Buchegger, S. Capkun, JY. Le Boudec, S. Vaudenay et al., "Report on a Working Session on Security in Wireless Ad Hoc Networks", *Mobile Computing and Communications Review, Vol. 6, Number 4*, 2003.

[bhr+01] S. Basagni, K. Herrin, E. Rosti and D. Bruschi, "Secure pebblenets", *In Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2001, pp.156-163.

[bjm04] J. Broch, D. Johnson, and D. Maltz, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR) ", *IETF Internet Draft (work in progress), http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-10.txt*, July 2004.

[bm03] C. Boyd and A. Mathuria, "Protocols for Authentication and Key Establishment", 2003.

[bn03] C. Boyd and J.M. G. Nieto, "Round-optimal Contributory Conference Key Agreement", *Public Key Cryptography PKC 2003, Springer-Verlag, LNCS 2567*, 2003, pp.161-174.

[br94] K. P. Birman and R. V. Renesse, "Reliable Distributed Computing with the ISIS Toolkit", *IEEE Computer Society Press*, Mar 1994.

[bw98] K. Becker and U. Wille, "Communication Complexity of Group Key Distribution", *in Proceedings of the ACM Conference on Computer and Communications Security*, 1998, pp.1-6.

[cc89] G.H. Chiou and W.T. Chen, "Secure broadcasting using the secure lock", *IEEE Transactions on Software Engineering, Vol. 15, Number 8*, Aug 1989, pp.929-934.

[cdk+02] B. Cain, S. Deering, I. Kouvelas, B. Fenner and A. Thyagarajan, "Internet Group Management Protocol, Version 3, RFC 3376", 2002.

[cgz98] C. C. Chiang, M. Gerla and L. Zhang, "Forwarding Group Multicast Protocol (FGMP) for Multihop, Mobile Wireless Networks", *Baltzer Cluster Computing, Volume 1, Number 2*, 1998, pp.187-196.

[cla+96] G. Caronni, H. Lubich, A. Aziz, T. Markson and R. Skrenta, "SKIP: Securing the Internet", *in Proceedings of the IEEE Fifth Workshop on Enabling Technologies (WET ICE)*, 1996.

[clr90] T. H. Cormen, C. E. Leiserson and R. L. Rivest, "Introduction to Algorithms", *MIT Press*, 1990.

[cps03] Haowen Chan, Adrian Perrig, Dawn Song, "Random Key Predistribution Schemes for Sensor Networks", *In 2003 IEEE Symposium on Research in Security and Privacy*, 2003.

[crb01] R. Chandra, V. Ramasubramanian, K. P. Birman, "Anonymous Gossip: Improving Multicast Reliability in Mobile Ad-Hoc Networks", *in Proceedings of the The 21st International Conference on Distributed Computing Systems*, 2001, pp.275.

[crytopapi] D. Esposito, "Supporting CryptoAPI in Real-World Applications", *Microsoft Interactive Developer, http://www.microsoft.com/mind/0697/crypto.asp*, June 1997.

[css02] D. Cavin, Y. Sasson and A. Schiper, "On the Accuracy of MANET Simulators", *In Proceedings of the Workshop on Principles of Mobile Computing*, 2002, pp.38-43.

[da99] T. Dierks and C. Allen, "The TLS protocol version 1.0. RFC 2246", Jan 1999.

[dac] NCSC, "A guide to understanding discertionary access control in trusted systems", *Report NCSC-TG-003 version-1, National Computer Security Center*, Sep 1987.

[db99] N. Daswani and D. Boneh, "Experimenting with Electronic Commerce on the PalmPilot", *In proceedings of Financial Cryptography '99, Lecture Notes in Computer Science, Vol. 1648, Springer-Verlag*, February 1999, pp.1-16.

[db99] N. Daswani and D. Boneh, "Experimenting with Electronic Commerce on the PalmPilot", *In proceedings of the Financial Cryptography '99, Lecture Notes in Computer Science, Vol. 1648, Springer-Verlag*, Feb 1999, pp.1-16.

[ddg+01] B.T.DeCleene, L.R.Dondeti, S.P.Griffin, T.Hardjono, D. Kiwior, J. Kurose, D. Towsley, S. Vasudevan and C. Zhang, "Secure Group Communications for Wireless Networks", *In Proceedings of the IEEE MILCOM'01*, 2001.

[dow92] W. Diffie, P. van Oorschot, and M. Wiener, "Authentication and authenticated key exchanges", *Design Codes and Cryptograhpy, 2*, 1992, pp.107-125.

[dp00] P. McDaniel and A. Prakash, "Antigone: Implementing Policy in Secure Group Communication", *Technical Report CSE-TR-426-00, Electrical Engineering and Computer Science, University of Michigan*, May 2000.

[dr01] J. Daemen and V. Rijmen, "Rijndael, the advanced encryption standard", *Dr. Dobb's Journal, Volume 26, Number 3*, Mar 2001, pp.137-139.

[eg02] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks", *Conference on Computer and Communications Security. Proceedings of the 9th ACM conference on Computer and communications security, Washington, DC, USA*, 2002, pp.41-47.

[fhh+04] B. Fenner, M. Handley, H. Holbrook and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", *Internet Draft (Work in progress), http://www.ietf.org/internet-drafts/draft-ietf-pim-sm-v2-new-11.txt*, 2004.

[fk92] D.F. Ferraiolo and D.R. Kuhn, "Role Based Access Control", *15th National Computer Security Conference*, 1992.

[fkk96] A. O. Freier, P. Karlton and P. C. Kocher, "The SSL Protocol Version 3.0 ", *Internet Draft, http://wp.netscape.com/eng/ssl3/draft302.txt*, Nov 1996.

[fms01] S. Fluhrer, I. Mantin and A. Shamir, "Weaknesses in the key scheduling algorithm of RC4", *In Proceedings of the Eighth Annual Workshop on Selected Areas in Cryptography*, Aug 2001, pp.1-24.

[fn93] A. Fiat and M. Naor, "Broadcast encryption", *Advances in Cryptology CRYPTO'93*, 1993, pp.480-491.

[ftm02] J. Flynn, H. Tewari and D. O'Mahony, "A Real-Time Emulation System for Ad Hoc Networks ", *in Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002), San Antonio, Texas*, Jan 2002, pp.115-120.

[g89a] L. Gong, "A Secure Identity-Based Capability System", *in Proceedings of the IEEE Symposium on Security and Privacy*, May 1989, pp.56-64.

[gg01] V. Gupta and S. Gupta, "Securing the Wireless Internet", *Communications Magazine, IEEE*, Dec 2001, pp.68-74.

[gg01] V. Gupta and S. Gupta, "Securing the Wireless Internet", *Communications Magazine, IEEE*, December 2001, pp.68-74.

[gjk+96] R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin, "Robust and efficient sharing of RSA functions", *Journal of Cryptology*, 1996.

[glomosim]The glomosim project see, "http://pcl.cs.ucla.edu/projects/glomosim/".

[gs99] S. K. S. Gupta and P. K. Srimanis, "An Adaptive Protocol for Reliable Multicast in Mobile Multi-hop Radio Networks", 1999.

[gsp+02] T. Gopalsamy, M. Singhal, D. Panda and P. Sadayappan, "A Reliable Multicast Algorithm for Mobile Ad Hoc Networks", *in Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, 2002, pp.563.

[hbc01] J. Hubaux, L. Buttyan and S. Capkun, "The quest for security in mobile ad hoc networks", *in Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, 2001, pp.146-155.

[hbt+03] J. Hsu, S. Bhatia, M. Takai, R. Bagrodia and Michael J. Acriche, "Performance of Mobile Ad Hoc Networking Routing Protocols in Realistic Scenarios", *In Proceedings of the MILCOM '03*, 2003.

[hbt+04] J. Hsu, S. Bhatia, M. Takai, R. Bagrodia and Michael J. Acriche, "Performance of Mobile Ad Hoc Networking Protocols in Large Scale Scenarios", *MILCOM '04*, 2004.

[hch+00] H Harney, A Colegrove, E Harder, U Meth and R Fleischer, "Group Secure Association Key Management Protocol", *IETF Internet Draft, http://www.securemulticast.org/draft-irtf-smug-gsakmp-02.txt*, November 2000.

[hch+00] H Harney, A Colegrove, E Harder, U Meth and R Fleischer, "Group Secure Association Key Management Protocol", *IETF Internet Draft, http://www.securemulticast.org/draft-irtf-smug-gsakmp-02.txt*, Nov 2000.

[hjm+02] A. Hess, J. Jacobson, H. Mills, R. Wamsley, K. E. Seamons and B. Smith, "Advanced Client/Server Authentication in TLS", *Network and Distributed System Security Symposium, San Diego, CA*, Feb. 2002.

[hk05] F. Haq and T. Kunz, "Simulation vs. emulation: Evaluating mobile ad hoc network routing protocols", *In Proceedings of the International Workshop on Wireless Ad-hoc Networks (IWWAN 2005), London, England*, May 2005.

[hm97] H. Harney and C. Muckenhirn, "Group key management protocol (GKMP) specification, RFC 2093", Jul 1997.

[hot+99] C. Ho, K. Obraczka, G. Tsudik, and K. Viswanath, "Flooding for reliable multicast in multi-hop ad hoc networks", *In Proceedings of the International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communication (DIALM)*, 1999, pp.64-71.

[hss+04] T. W. van der Horst, T. Sundelin, K. E. Seamons and C. D. Knutson, "Mobile Trust Negotiation: Authentication and Authorization in Dynamic Mobile Networks", *In Proceedings of the Eighth IFIP Conference on Communications and Multimedia Security, Lake Windermere, England*, Sep 2004, pp.97-109.

[hy98] S. Hirose and S. Yoshida, "An Authenticated Diffie-Hellman Key Agreement Protocol ", *Public Key Cryptography, Springer-Verlag, LNCS 1431*, 1998, pp.135-148.

[ige00] C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks", *in 6th Annual Int. Conf. on Mobile Computing and Networking (MobiCOM □00), Boston, Massachusetts, United States*, 2000, pp.56-67.

[ikev1] Internet Key Exchange (IKEv1) Protocol see, "Request For Comments (RFC), http://www.ietf.org/rfc/rfc2409.txt", Nov 1998.

[ikev2] Internet Key Exchange (IKEv2) Protocol see, "Internet Draft, http://www.ietf.org/internet-drafts/draft-ietf-ipsec-ikev2-17.txt", Sep 2004.

[ipaq] iPAQ see, "http://www.compaq.com/support/handhelds/iPAQ_H3600.html."

[ipv6] IP Version 6 Working Group (ipv6) charter see, "http://www.ietf.org/html.charters/ipv6-charter.html".

[ipv6nd] G. Daley, "Preempting IPv6 Neighbour Discovery", *IETF Internet Draft (work in progress), http://www.ietf.org/internet-drafts/draft-daley-ipv6-preempt-nd-00.txt*, Jun 2004.

[itw82] I. Ingemarsson, D. Tang, and C. Wong, "A Conference Key Distribution System", *IEEE Transactions on Information Theory, Vol.28 Number 5*, 1982, pp.714-720.

[jb99] A. Juels and J. Brainard, "Client Puzzles: A Cryptographic Countermeasure Against Connection Depletion Attacks", *in Proceedings of the NDSS '99 (Networks and Distributed Security Systems)*, Feb 1999, pp.151-165.

[k00] V. Karpijoki, "Security in Ad-Hoc Networks", *Techical Report, TML-HUT*, 2000.

[k01] B. Komar, "Troubleshooting Certificate Status and Revocation", *http://www.microsoft.com/technet/prodtechnol/winxppro/support/tshtcrl.mspx*, Oct 2001.

[k01] H. Krawczyk, "The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is {SSL}?)", *Lecture Notes in Computer Science, Volume 2139*, 2001, pp.310-331.

[kbc97] H. Krawczyk, M. Bellare and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication. RFC 2104", Feb 1997.

[kig+03] A. D. Keromytis, S. Ioannidis, M. B. Greenwald and J. M. Smith, "The STRONGMAN Architecture", *In Proceedings of the Third DARPA Information Survivability Conference and Exposition (DISCEX III), Washington, D.C*, Apr 2003, pp.178-188.

[kka03] A. Khalili, J. Katz and W. Arbaugh, "Towards Secure Key Distribution in Truly Ad-Hoc Networks ", *IEEE Workshop on Security and Assurance in Ad hoc Networks in conjunction with the 2003 International Symposium on Applications and the Internet, Orlando, FL*, Jan 2003.

[klx+02] J. Kong, H. Luo, K. Xu, D. L. Gu, M. Gerla, S. Lu, "Adaptive Security for Multi-layer Ad-hoc Networks issue", *John Wiley InterScience Press journal 'Special Issue of Wireless Communications and Mobile Computing'*, Aug 2002, pp.533-547.

[kmm98] K. Kihlstrom, L. Moser, and P. Melliar-Smith., "The SecureRing Protocols for Securing Group Communication", *In Hawaii International Conference on System Sciences*, 1998.

[kmt03] Y. Kim, D. Mazzocchi and G. Tsudik, "Admission Control in Peer Groups ", *In Proceedings of the IEEE International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA*, Apr 2003.

[kv00] Y. B. Ko and N.H. Vaidya, "Geotora: A protocol for geocasting in mobile ad hoc networks", *Technical report, TR-00-010, Texas A&M University*, 2000.

[kv98] Y. B. Ko and N. H. Vaidya, "Geocasting in mobile ad hoc networks: Location-based multicast algorithms", *Technical Report TR-98-018, Texas A&M University*, Sep 1998.

[l71] B. W. Lampson, "Protection", *in Proceedings of the 5th Annual Princeton Conference on Information Sciences and Systems*, Mar 1971, pp.437-443.

[l73] B. W. Lampson, "A Note on the Confinement Problem", *Communications of the ACM, Volume 16, Number 10*, Oct 1973, pp.613-615.

[l96] G. Lowe, "Some New Attacks upon Security Protocols", *in Proceedings of the 9th IEEE Computer Security Foundations Workshop, IEEE Press*, 1996, pp.162-169.

[ldo03] B. Lehane, L. Doyle and D. O'Mahony, "Shared RSA Key Generation in a Mobile Ad Hoc Network", *In Proceedings of the MILCOM 2003*, Oct 2003.

[leh03] J. Luo, P. Eugster and J. P. Hubaux, "Route Driven Gossip: Probabilistic Reliable Multicast in Ad Hoc Networks", *In Proceedings of the IEEE INFOCOM 2003, San Francisco, CA, USA*, Apr 2003, pp.2229-2239.

[lk00] S. Lee and C. K. Kim, "Neighbor supporting Ad hoc multicast routing protocol", *In Proceedings of the ACM Symposium on Mobile Ad Hoc Networking & Computing*, Aug 2000, pp.37-44.

[ln03] G. Lin, G. Noubir, "Secure Multicast over Multihop Wireless Ad Hoc Networks", *in the Proceedings of Mobile Ad Hoc Networks Workshop (MADNET), Sophia-Antipolis, France*, 2003.

[lp03a] L. Lazos and R. Poovendran, "Energy-Aware Secure Multicast Communication in Ad-hoc Networks Using Geographic Location Information", *In Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing, Hong Kong, China*, Apr 2003.

[lp03b] L. Lazos and R. Poovendran, "Location-Aware Secure Wireless Multicast in Ad-Hoc Networks under Heterogeneous Path-loss", *UWEETR-2003-0012, UWEE Technical Report Series*, 2003.

[lsg00] S. Lee and W. Su and M. Gerla, "On-Demand Multicast Routing Protocol in Multihop Wireless Mobile Networks", *ACM/Baltzer Mobile Networks and Applications, special issue on Multipoint Communication in Wireless Mobile Networks*, 2000.

[lsh+00] S-J. Lee, W. Su, J. Hsu, M. Gerla and R. Bagrodia, "A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols", *In Proceedings of the IEEE Infocom 2000*, Mar 2000, pp.565-574.

[ltm99] M. Liu and R. Talpade and A. McAuley, "Amroute: Adhoc multicast routing protocol", *Technical Report TR 99-1, CSHCN*, 1999.

[lzk+02] H. Luo, P. Zerfos, J. Kong, S. Lu, and L. Zhang, "Self-securing ad hoc wireless networks", *In The 7th IEEE Symposium on Computers and Communications*, 2002.

[m94] J. Moy, "Multicast Extensions to OSPF", *Internet Draft, http://www.ietf.org/rfc/rfc1584.txt*, Mar 1994.

[m97] S. Mittra, "Iolus: A framework for scalable secure multicasting", *in Proceedings of the ACM SIGCOMM'97*, Sep 1997, pp.277-288.

[m99] C. Meadows, "A Formal Framework and Evaluation Method for Network Denial of Service", *In Proceedings of the 12th IEEE Computer Security Foundations Workshop (CSFW)*, Jun 1999, pp.4-13.

[mac] NCSC, "DoD trusted computer systems evaluation criteria", *Report 5200.28-STD, National Computer Security Center*, Dec 1985.

[mah00] S. Mäki, T. Aura, M. Hietalahti, "Robust Membership Management for Ad-hoc Groups", *In Proceedings of the 5th Nordic Workshop on Secure IT Systems (NORDSEC 2000)*, Oct 2000.

[manet] Mobile Ad-hoc Networks (manet) charter see, "http://www.ietf.org/html.charters/manet-charter.html".

[md01a] D. O'Mahony and L. Doyle, "Architectural Imperatives for 4th Generation IP based Mobile Networks", *in Procedings of the Fourth international symposium on wireless personal multimedia communications*, Sep 2001, pp.1319-1325.

[md01b] D. O'Mahony and L. Doyle, "An Adaptable Node Architecture for Future Wireless Networks ", *Mobile Computing: Implementing Pervasive Information and Communication Technologies, Kluwer series in Interfaces in OR/CS, Kluwer Publishing*, Aug 2001.

[mf99] E. L. Madruga and J.J.Garcia-Luna-Aceves fmadruga, "Multicasting Along Meshes in Ad-Hoc Networks", *In Proceedings of the IEEE ICC'99, Vancouver, Canada*, Jun 1999, pp.784-792.

[mgl04] P. Mohapatra C. Gui and J. Li, "Group Communications in Mobile Ad Hoc Networks", *Special Issue on Ad Hoc Networks, IEEE Computer*, Feb 2004, pp.70-77.

[mma96] L. E. Moser, P. M. Melliar-Smith, D. A. Agarwal, R. K. Budhia, and C. A. Lingley-Papadopoulos, "Totem: A fault-tolerant multicast group communication system", *Communications of the ACM, Volume 39, Number 4*, Apr 1996, pp.54-63.

[mov96] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, "Handbook of Applied Cryptography", Oct 1996.

[mss97] D. Maughan and M. Schertler and M. Schneider and J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)", *Internet Draft (draft-ietf-ipsec-isakmp-08)*, 1997.

[my99] A. Mayer and M. Yung, "Secure Protocol Transformation via "Expansion": From Two-party to Groups", *In Proceedings of the 6th ACM Computer and Communications Security Conference (CCS)*, 1999.

[ni97] J. C. Navas and T. Imielinski, "GeoCast Geographic Addressing and Routing", *Mobile Computing and Networking*, 1997, pp.66-76.

[ns2] The Network Simulator - ns-2 project see, "http://www.isi.edu/nsnam/ns/".

[o98] H. Orman, "The OAKLEY Key Determination Protocol", *IETF Internet Draft, http://www.ietf.org/rfc/rfc2412.txt*, Nov 1998.

[oks99] T. Ozaki, J. Kim, and T. Suda, "*Bandwidth Efficient Multicast Routing Protocol for Ad hoc Networks*", *In Proceedings of the IEEE ICCCN*, Oct 1999, pp.10-17.

[openssl] The OpenSSL Project see, "http://www.openssl.org/".

[p03] T. Pusateri, "Distance Vector Multicast Routing Protocol", *Internet Draft (work in progress), http://www.ietf.org/internet-drafts/draft-ietf-idmr-dvmrp-v3-11.txt*, Oct 2003.

[p99] A. Perrig, "Efficient collaborative key management protocols for secure autonomous group communication", *In Proceedings of the International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC'99)*, 1999, pp.192-202.

[pkd02] P. Poupyrev, M. Kosuga and P. Davis, "Analysis of wireless message broadcasts in large ad hoc networks of PDA's", *In Proceedings of the IEEE Fourth International Conference on Mobile and Wireless Communication Networks, Stockholm*, Sep 2002, pp.299-303.

[pl00] J. Pieprzyk and C. H. Li, "Multiparty key agreement protocols", *IEE Proceedings, Computers and Digital Techniques. Volume147, Number 4*, Jul 2000, pp.229-236.

[pml+03] R. D. Pietro, L. V. Mancini, Y. W. Law, S. Etalle and P. Havinga, "LKHW: A Directed Diffusion-Based Secure Multicast Scheme for Wireless Sensor Networks ", *In Proceedings of the 2003 International Conference on Parallel Processing Workshops, Kaohsiung, Taiwan.*, Oct 2003, pp.397-412.

[pr97] E. Pagani and G. P. Rossi, "Reliable broadcast in mobile multihop packet networks", *in Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking*, 1997, pp.34-42.

[r94] M. K. Reiter, "Secure agreement protocols: Reliable and atomic group multicast in rampart", *In Proceedings of 2nd ACM Conference on Computer and Communications Security*, Nov 1994, pp.68-80.

[rbd01] O. Rodeh, K. Birman, D. Dolev, "The Architecture and Performance of the Security Protocols in the Ensemble Group Communication System", *Journal of ACM Transactions on Information Systems and Security (TISSEC)*, 2001.

[rbm96] R. van Renesse, K. P. Birman, and S. Maffeis, "Horus: A flexible group communication system", *Communications of the ACM, Volume 39, Number 4*, Apr 1996, pp.76-83.

[rp99] E. M. Royer and C. E. Perkins, "Multicast Operation of the Ad hoc On-Demand Distance Vector Routing Protocol", *In Proceedings of the MobiCom '99, Seattle, WA*, Aug 1999, pp.207-218.

[rsa] The RSA Factoring Challenge see, "http://www.rsasecurity.com".

[s96] B. Schneier, "Applied Cryptography - Second Edition", Oct 1996.

[sa99] F. Stajano and R. Anderson, "The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks", *Security Protocols, 7th International Workshop Proceedings*, 1999.

[sip] Session Initiation Protocol (sip) charter see, "http://www.ietf.org/html.charters/sip-charter.html".

[sir01] A. Stubblefield, J. Ioannidis, A. D. Rubin, "Using the Fluhrer, Mantin, and Shamir Attack", *AT&T Labs Technical Report TD-4ZCPZZ*, 2001.

[sk00] B. Song and K. Kim, "Comparison of Existing Key Establishment Protocols", *KIISC (CISC 2000)*, Nov 2000, pp.677-690.

[so94] P. Syverson and P. van Oorschot, "On Unifying Some Cryptographic Protocol Logics", *in Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press*, 1994, pp.14-28.

[so96] P. Syverson and P. van Oorschot, "A Unified Cryptographic Protocol Logic", *Draft available from the authors*, 1996.

[spki] Simple Public Key Infrastructure (SPKI) see, "http://www.ietf.org/html.charters/spkicharter.html".

[ssb99] P. Sinha and R. Sivakumar and V. Bharghavan, "MCEDAR: Multicast Core-Extraction Distributed Ad hoc Routing ", *in Proc. of the Wireless Communications and Networking Conference*, 1999.

[ssd90] G. Steer, L. Strawczynski, W. Diffie, and M. Wiener, "A secure audio teleconference system", *Advances in cryptology*, 1990, pp.520-528.

[stw97] M. Steiner, G. Tsudik and M. Waidner, "Cliques: A protocol suite for key agreement in dynamic groups", *Research Report RZ 2984 (#93030), IBM Zurich Research Lab*, Dec 1997.

[stw98] M. Steiner, G. Tsudik and M. Waidner, "CLIQUES: A New Approach to Group Key Agreement", *in Proceedings of the 18th International Conference on Distributed Computing Systems*, 1998, pp.380-387.

[swy+02] K. E. Seamons, M. Winslett, T. Yu, B. Smith, E. Child, J. Jacobson, H. Mills, and L. Yu, "Requirements for Policy Languages for Trust Negotiation", *In Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY 2002), Monterey, CA*, Jun 2002, pp.68-79.

[tls] Transport Layer Security (tls) charter see, "http://www.ietf.org/html.charters/tls-charter.html".

[tol+02] K. Tang, K. Obraczka, S.J. Lee and M. Geria, "A reliable, congestion-control led multicast transport protocol in multimedia multi-hop networks", *The 5th International Symposium on Wireless Personal Multimedia Communications*, 2002.

[trust] TrustBuilder project architecture see, "http://isrl.cs.byu.edu/".

[tt00] W. G. Tzeng and Z. J. Tzeng, "Round-Efficient Conference Key Agreement Protocols with Provable Security", *Advances in Cryptology - ASIACRYPT 2000*, 2000, pp.614-628.

[ve03] E. Vollset and P. Ezhilchelvans, "A Survey of Reliable Broadcast Protocols for Mobile Ad-hoc Networks", *Technical Report CS-TR-792, School of Computing Science, University of Newcastle upon Tyne*, 2003.

[w03] M. Winslett, "An Introduction to Automated Trust Establishment", *1st International Conference on Trust Management, Crete, Greece*, May 2003, pp.275-283.

[wc02] B. Williams and T. Camp, "Comparison of broadcasting techniques for mobile ad hoc networks", *In Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2002), Lausanne, Switzerland*, Jun 2002, pp.194-205.

[wcs99] M. Waldvogel, G. Caronni, D. Sun, N. Weiler and B. Plattner, "The VersaKey Framework: Versatile Group Key Management", *IEEE Journal on Selected Areas in Communications, Vol. 17 Number 9*, 1999, pp.1614-1631.

[wince] Windows CE see, "http://www.microsoft.com/windowsce/".

[wm99] S. Blake-Wilson and A. Menezes, "Unknown Key-Share Attacks on the Station-to-Station (STS) Protocol", *in Proceedings of the Second International Workshop on Practice and Theory in Public Key Cryptography*, 1999, pp.154 -170.

[wmk94] B. Whetten, T. Montgomery and S. Kaplan, "A high performance totally ordered multicast protocol", *Dagstuhl Seminar on Distributed Systems*, 1994, pp.33-57.

[wsj00] W. Winsborough, K. Seamons and V. Jones, "Automated Trust Negotiation", *In Proceedings of the DARPA Information Survivability Conference and Exposition, Hilton Head Island, SC*, Jan 2000, pp.88-102.

[wt99] C. Wu and Y. Tay, "AMRIS: A Multicast Protocol for Ad hoc Wireless Networks ", *In Proceedings of the IEEE MILCOM'99, Atlantic City, NJ*, 1999.

[x.509] Public-Key Infrastructure (X.509) charter see, "http://www.ietf.org/html.charters/pkix-charter.html".

[yd02] A.Yasinsac and J. A. Davis, "Modeling Protocols for Secure Group Communication in Ad Hoc Networks", *In Proceedings of the Tenth International Workshop on Security Protocols, Cambridge, UK*, Apr 2002.

[yk03] S. Yi and R. Kravets, "Composite Key Management for Ad Hoc Networks ", *Report No. UIUCDCS-R-2003-2392*, Aug 2001.

[z00] R. Zuccherato, "Request For Comments (RFC) "Methods for Avoiding the "Small-Subgroup" Attacks on the Diffie-Hellman Key Agreement Method for S/MIME", see http://www.ietf.org/rfc/rfc2785.txt", Mar 2000.

[z97] P. Zimmermann, "The Official PGP User's Guide", *MIT Press*, June 1997.

[zh99] L. Zhou and Z. Haas, "Securing ad hoc networks", *IEEE Network Magazine*, 1999.